

ИДЕАЛЬНО!

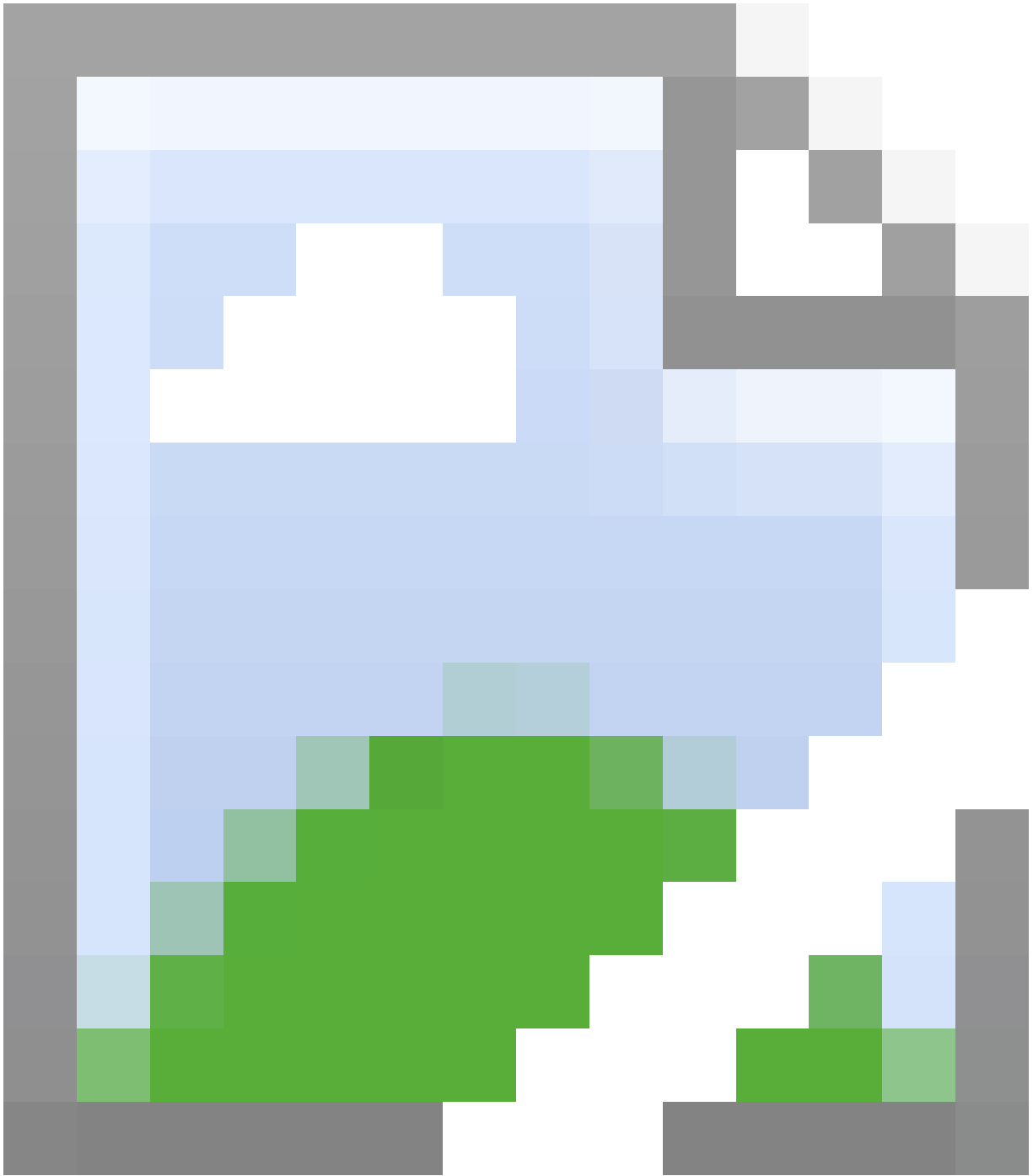
*как создать
и переделать*

СВОЙ САЙТ

*Правильный подход
и передовые
техники разработки*

PUSHBOOKS
КНИГИ КОТОРЫЕ ДОЧИТЫВАЮТ

Книга
02



Идеально! Как создать и переделать свой сайт. Правильный подход и передовые техники разработки

Издано с разрешения Smashing Media GmbH.

Под редакцией Александра Сарычева

© The Smashing Book #3 – Redesign the Web, 2012

© Smashing Media GmbH, 2012

© О. Серовская, перевод, 2013

© Издательство «СилаУма – Публишер», 2013

* * *

Предисловие от партнера российского издания

Множество авторов сейчас пытаются научить вас создавать сайты, приносящие прибыль своим владельцам. Однако книги, написанные настоящими мастерами своего дела, встречаются редко. Перед вами одна из них.

Эта книга достойна самого внимательного изучения. Здесь есть все инструменты для того, чтобы вы смогли создавать по-настоящему современные, удобные и эффективно работающие сайты.

Не важно, профессионал ли вы в деле сайтостроения или решили сделать первые свои страницы, в этом бесценном труде вы найдете ответы на многие вопросы.

Каждый раздел написан ведущим специалистом в своей отрасли.

- Первый научит вас выстраивать грамотный процесс работы с заказчиками. Этот блок будет полезен не только «технарям», но и всем, кто по роду службы сталкивается с заказчиками напрямую.
- Второй в подробностях описывает процесс создания или переделки сайта от выбора платформы и языка программирования до определения хостинга, интеграции платежных систем и управления контентом.
- Из третьего вы узнаете, как виртуозно управляться с версткой, JavaScript и современными языками программирования. После прочтения HTML5 и CSS3 станут вашими надежными союзниками.
- Четвертый посвящен преобразованию сайта, смене стилей и кодов. Здесь уделено внимание и веб-типографике, и грамотной разметке страниц.
- Пятый погрузит вас в мир Java-возможностей. Отныне перемещения, трансформации и анимация с JavaScript перестанут быть для вас загадкой.

- В последних разделах внимание уделяется повышению юзабилити, дизайнерским фишкам и приданию сайту индивидуальности. Не поверите, но тут вам в помощь привлекается даже психология. Вы узнаете, как вызвать нужные эмоции у пользователей и запрограммировать их поведение.

И все это обильно снабжено примерами, иллюстрациями и html-кодами, которые вы сможете сразу использовать в своей работе.

Эта книга способна полностью перевернуть ваш подход к работе над созданием сайтов, сделать его более продуктивным и денежным.

Напоследок хочется сказать о том, что вы – везунчик. Вы имеете отношение к одной из самых интересных и динамично развивающихся сфер – разработке и веб-дизайну. Но чтобы получать от своей работы не только удовольствие, но и материальное вознаграждение вкупе с вечной благодарностью от заказчиков, вы должны знать, как этого добиться. И мы уверены, что в этом вам поможет эта книга.

Удачи!

Команда iAGE,

Технологии вашего Интернет-будущего

(iage.net)

Вступительное слово научного редактора

Вы подумываете переделать свой сайт? Или вы специалист, который занимается их созданием? В любом случае всем вам будет полезно взять верный курс на разработку современного сайта.

Каждый год издаётся большое количество книг, скорее даже справочников, для технических специалистов. Есть книги и по маркетингу, юзабилити, проектированию целевых страниц (landingpages), оптимизации рекламы. Но вот куда копнуть глубже и где почитать про подход к разработке современного сайта? С чего начать, как выстроить процесс работы, какими технологиями воспользоваться и что не забыть по дороге?

Книга укажет направление для работы. Прочитав книгу, вы сможете разобраться, как добавить сайту «человеческое лицо», узнаете фишки юзабилити, поймёте как сделать отзывчивый (responsive) дизайн, а также убедитесь, что HTML5 и CSS3 уже давно в бою. Из книги вы узнаете и о перспективном подходе к вёрстке сайтов – Flexbox. Я уверен – за ним будущее.

Книга посеет и зерно здорового скептицизма – а по правильному ли подходу вы (или ваши подрядчики) разрабатываете сайт? Строите ли работу от контента, делаете ли прототипы, или до сих пор мыслите статичными макетами?

Мы в «ЛидМашине» уже частично использовали отдельные вещи, о которых рассказывается в книге, в своей работе и проектах наших клиентов. Теперь мы поглядим на весь процесс сверху и, вероятно, скорректируем процесс. Взгляните и вы.

Александр Сарычев,

маркетёр-аналитик лаборатории интернет-маркетинга
«ЛидМашина» (LeadMachine.ru)

Введение

Автор: Элиот Джей Стокс

Когда вы работаете в стремительно развивающейся индустрии, то каждый миг этого развития очень волнующий. Замечу, что сказать: «Самое захватывающее сейчас – это работа в веб-дизайне и разработке» – будет несколько неверно. Работа и жизнь на острие новых интернет-технологий всегда были захватывающими и всегда таковыми будут.

И все же в последние месяцы я ловлю себя на том, что говорю эти слова, потому что сейчас действительно захватывающий момент. Во времена широкого использования табличной верстки и инноваций новых медиа, мы были словно беспокойные дети, которые ищут поддержки у своих родителей из мира аналоговых медиа; в начале этого столетия мы превратились в неуклюжих подростков, которые экспериментируют с различными сервисами и техниками, как в лучшую сторону (веб-стандарты), так и в худшую (плагины). Сейчас мы выросли примерно до двадцатилетних и преобразовали все, чему научились, в новый, захватывающий опыт, который, в конце концов, окончательно охватит изменяемую, мягкую, открытую и изменчивую натуру Интернета.

Мало просто иметь продвинутые сервисы, равно как и мало уметь их использовать. Настоящая зрелость – это глубокое понимание того, что может и должно быть достигнуто.

Нет ничего революционного во множественной поддержке браузеров для экспериментальных CSS-характеристик. Не новость, что есть возможность использовать тысячи различных гарнитур. Обычное дело – вставка медиазапросов для изменения стилей в соответствии с шириной браузера. Но применение этих инструментов придает вашей работе изящество.

Эта концепция зрелой индустрии, более того – нашей собственной компетентности, в ней раскрылась на конференциях по всему миру, на

страницах печатных журналов, дневников и книг и, конечно же, в блогах. Создалось ощущение взросления Интернета, что не так давно побудило меня переделать сайт Smashing Magazine. С чем мы столкнулись при переделке, вы узнаете на страницах этой книги.

Тема переделывания сайта иногда принимает здесь форму делового обсуждения, ведь это тоже часть процесса, подчеркивает один из авторов, Пол Боуг. Также обсуждается модернизация веб-сайта при помощи таких сервисов, как HTML5, CSS3 and JavaScript, о чем написали Бен Шварц, Ли Веру, Дэвид Стори и Кристиан Хейлманн. Представлена тема и в рассказе про адаптивные мобильные техники (авторы Стивен Хей и Энди Кларк). И, конечно, она раскрывается в широком обсуждении, представленном в разделах, авторы которых Рэйчел Эндрю, Дмитрий Фадеев, Марк Эдвардс, Арэл Балкан и Аарон Уолтер.

Желание ускорить ход событий, которые запустили механизм переделывания Smashing Magazine, также повлияло на создание и контент книги. Эта страсть, которая связывает нас больше как единомышленников, а не конкурентов, – страсть, которую, я уверен, вы прочувствуете, когда будете читать эти страницы, – вдохновит каждого из нас сделать Интернет лучше с помощью... задания градиентов через CSS, использования веб-шрифтов, использования медиазапросов.

Бизнес-подход в переделывании сайтов

Автор: Пол Боуг

Рецензент: Коллис Тайед

Переделывание сайтов – это та область, в которой самые смелые разработчики и дизайнеры могут позволить разгуляться своей фантазии. Пустой документ Photoshop, библиотека фрагментов кода CSS3 – для них нет ничего более захватывающего, чем погружение в мир новых возможностей. Однако, как это ни печально, я здесь для того, чтобы разрушить их воздушные замки.

В последующих разделах представлены клевые дизайнерские техники, еще более крутые возможности для кодинга, а множество прекрасных наглядных примеров дизайна сайтов помогут разработчикам не заблудиться в лабиринте новых инструментов.

Я же затрону тему бизнес-подхода в реконструкции сайтов.

Почему я остановился на такой скучной теме, спросите вы? Ответ прост. Суть бизнес-подхода в дизайне состоит не только в том, чтобы создать удачный сайт, но и время от времени полностью переделывать его, несмотря на кажущуюся безупречность проекта. Я, как и все вы, принимал участие в разработке множества сайтов, и меня буквально швыряло по волнам изменяющихся технических заданий, нечетко сформулированных инструкций и разногласий с заказчиками. К счастью, годы опыта научили меня тому, что если ты достаточно подготовлен, можно избежать этих ловушек.

Цель этого раздела – вооружить вас знаниями и познакомить с техниками разработки для того, чтобы ваша последующая работа над реконструкцией сайта не превратилась в ночной кошмар. Рассмотрим тему по следующим пунктам:

- Подводные камни при переделывании сайта.

- Тщательное исследование проекта.
- Работа с клиентами.
- Тестирование дизайна.
- Жизнь сайта после реконструкции.

Итак, до того как вы воодушевитесь этим списком, давайте выясним, когда же пора кардинально переделывать сайт.

Пришло ли время переделывать сайт?

Как только начальник или клиент просят нас переделать сайт, мы готовы ринуться в бой. По-другому и быть не может! Это же то, что мы обожаем делать! К сожалению, глобальные перемены не всегда являются лучшим решением, и мы как эксперты по разработке обязаны объяснить почему.

С момента своего рождения Интернет переживал ряд периодических изменений. Также и каждые несколько лет кто-то из высшего руководства приходит в ужас при виде своего сайта и требует его сменить. Прежний сайт выбрасывается, а его место занимает новый.

Какое-то время этот сайт сияет на просторах Интернета. Однако контент не обновляется, технологии движутся вперед, а вкусы и пристрастия меняются. Некогда новый блестящий сайт постепенно блекнет до тех пор, пока компания не начинает стыдиться его.

Через пару лет кто-то из руководителей вновь видит необходимость сделать что-то с сайтом, и процесс начинается сначала.

Когда полная переработка сайта является расточительством

В следующий раз, когда клиент попросит вас полностью переделать сайт, будьте готовы вступить с ним в спор и объяснить, почему эта просьба может оказаться ошибкой.

Причины могут быть следующими:

- В процессе глобальной реконструкции затрагиваются все составные части сайта, даже те, которые все еще работают безупречно. И сайт компонуется заново.
- Большую часть своей жизни сайт работает неэффективно из-за того, что контент не обновляется и постепенно устаревает дизайн. И хотя раз в несколько лет сайт радует вас своей новизной, в остальное время он рассматривается, скорее, как помеха и поэтому работает не в полную силу.
- Пользователи редко положительно реагируют на большие изменения. Вам достаточно лишь посмотреть, какой шумный протест вызывает у них обновление Facebook, чтобы осознать эту проблему^[1].
- Периодическая глобальная реконструкция сайта невыгодно сказывается на расходе денежных средств, так как это требует солидных инвестиций каждые несколько лет.
- Тестирование эффективности комплексной переделки сайта – процесс сложный, так как изменяется очень многое.
- Затяжное отсутствие обновлений дает пользователям мало оснований для возвращения на ваш сайт.

Во многих случаях я советую своим клиентам отказаться от глобальной реконструкции в пользу постепенной перестройки сайта.

Хорошие дизайнеры создают новое, великие дизайнеры переделывают старое

В 2005 году Кэмерон Молл впервые популяризовал идею о предпочтении постепенной перестройки сайта, а не его полной переработке. Концепция существенно развилась с того момента, но ей все еще легко дать определение: перестройка или перегруппировка сайта – это серия поэтапных изменений, производимых со временем для того, чтобы решать отдельные бизнес-задачи.

«Перегруппировка сайта – это серия поэтапных изменений, проводимых со временем для того, чтобы выполнять характерные бизнес-технические задачи».

Кэмерон Молл

Другими словами, сторонники постепенного развития сайта отвергают идею его серьезной реконструкции каждые несколько лет, ради того чтобы он оставался актуальным. Вместо этого они предлагают программу непрерывной пошаговой разработки, которая увеличит эффективность сайта при выполнении бизнес-задач. Это поможет избежать подводных камней, возникающих при дорогостоящей модернизации каждые несколько лет.

Это мнение противоречит мысли, что сайт на самом деле может быть когда-либо готов. Суть подхода в том, что дизайн необходимо развивать на основании постоянного тестирования. Как только больше узнаешь о поведении и предпочтениях пользователей, сразу же эти новые знания нужно использовать в дизайне сайта. Это позволит ему заработать еще более эффективно. Следует сказать, что метод перестройки подходит не каждому проекту.

Когда сделать заново предпочтительней, чем улучшать постепенно

Несмотря на то что я фанат поэтапного усовершенствования сайтов (т. е. перегруппировки), чаще всего в моей работе, которой я занимаюсь в своем агентстве Headscape, все-таки приходится полностью переделывать их. Мы стараемся, где только возможно, использовать текущее состояние сайта как отправную точку, но часто его создание происходит с нуля, без опоры на предыдущий опыт. Все это потому, что при создании сайтов в редких случаях разработчики думали об их долгосрочном развитии. Жизни сайта после его создания придавали мало значения. Позже в этом разделе мы обсудим данную проблему.

К тому же перегруппировка сама по себе может послужить причиной для крупной реконструкции. Даже тот сайт, в план развития которого была заложена возможность поэтапных обновлений, время от времени нуждается в полном переделывании. На то есть две причины. Во-

первых, во многих постоянно изменяющихся сайтах будут накапливаться нестыковки, которые постепенно разрушат интерфейс.

«В конце концов, стратегия поэтапных изменений так или иначе, разрушает связанность, требуя новой архитектуры для пользовательского интерфейса»

В своей статье, написанной в поддержку поэтапных изменений[2], Якоб Нильсон говорит:

Другими словами, любой сайт со временем нужно будет переделывать полностью.

Вторая проблема, связанная с поэтапным изменением, это его влияние на базовый код. Как-то в нашем агентстве мы более шести лет работали с одним из клиентов. Это время было потрачено на непрерывный процесс улучшения и поэтапного изменения сайта. Добавлялись новые функции, в то время как другие выбрасывались. Дизайн менялся на основании обратной связи с заказчиком. Все эти изменения в конце концов превратились в кошмар. Одна часть сайта была написана на классической ASP[3], другая часть на .NET[4]. CSS-файлы[5] пухли от строк кода, в которых больше не было необходимости. Мы планировали и документировали так хорошо, как только могли, но в итоге стало понятно, что весь базовый код придется переписать.

Не удивительно, что клиент отказывался платить значительную сумму, не видя никаких визуальных отличий. Тогда мы совместили эту работу с изменением дизайна сайта и таким образом убили двух зайцев одним выстрелом.

Несмотря на то что процесс поэтапных изменений предпочтительнее, периодическое полное переделывание сайта все еще имеет смысл.

Суть в том, чтобы распознать признаки, которые дадут нам знать, когда требуется полная переделка.

Сигналы к масштабным изменениям

Мы уже определили два сигнала, которые указывают на то, что сайт лучше переделать заново, чем перегруппировать: если в интерфейс пользователя вкрадывается несовместимость и если код становится неуправляемым.

Вопросы кода обычно связаны с производительностью. Если сайт страдает от серьезных проблем с производительностью, а поэтапных изменений недостаточно, чтобы исправить их, тогда полная реконструкция могла бы стать решением. Создание целого сайта с нуля дает вам возможность оптимизировать производительность, удалив унаследованный программный код.

Полная переделка сайта может быть важной еще просто потому, что существующий дизайн исчерпал свои возможности. И хотя, как правило, еще есть возможность совершать поэтапные изменения, но некоторые аспекты все же ужасно сложно обновлять, не навредив остальному дизайну.

Например, изменение базовой модульной структуры сайта повлияет на все, начиная от навигации и заканчивая размером шрифта. Поэтому если нужно радикально изменить модульную сетку из-за нового несовместимого контента или изменений в разрешении экрана, то создание нового сайта может стать необходимым.

Также есть бизнес-причины, по которым считается, что переделать заново сайт будет лучше, чем поэтапно усовершенствовать его. Значительное переделывание несет в себе возможности продвижения, которые вы не получите от поэтапных изменений. И еще, большая переделка сайта дает радикальную встряску, которая иногда так необходима для того, чтобы старый сайт зажил новой жизнью.

И, наконец, есть еще одна простая причина, по которой я продвигаю идею о полном переделывании сайта. Это когда организация основательно поменяла свое позиционирование. В этом случае недостаточно будет слегка переделать логотип. Наоборот, если компания серьезно меняет свое позиционирование, то это повлияет на все, начиная от целевой аудитории и заканчивая контентом и внешним видом сайта.

Я часто мог наблюдать, что взять и лишь вклеить новый логотип в сайт бывает недостаточно. Бренд – это нечто большее, чем просто логотип. Итак, неважно, выбираете вы глобальную переделку или постепенную перегруппировку сайта, но делать это вы должны, определив причины.

Причины для изменений

Как отмечает в своей статье [\[6\]](#) Кэмерон Молл, переделывать сайт просто из-за того, что он внешне «постарел», недостаточно. Изменения должны приводиться в действие бизнес-задачами.

Типичные причины для изменений следующие:

- Изменение тенденций рынка.
- Изменение бизнес-модели.
- Падение конверсии.
- Рост требований о поддержке сайта со стороны клиентов.
- Смена позиционирования бренда.

Замечу, что с вашим бизнесом связаны не только эти задачи (что важнее, чем просто эстетика), но и выгода, которую вы получите от незамедлительных действий. Другими словами, если показатель конверсии вашего сайта падает или ваши бизнес-предложения изменились, то вы не захотите ждать два года до следующей существенной реконструкции сайта, чтобы воспользоваться им.

Вот поэтому часто предпочтительней делать поэтапные изменения. Все меняется быстро как в онлайн, так и в офлайн. Если вы хотите, чтобы ваш сайт работал с максимальной эффективностью, то не можете ждать, когда полностью переделаете его.

В зависимости от актуальности проблемы изменения сайта могут быть незначительными, такими как доработка текстов, или существенными, такими как полное переделывание интерфейса пользователя. Как бы

то ни было, думаете ли вы об изменениях или уже пришло время изменить ваш сайт, все решения должны подкрепляться бизнес-задачами и быть частью действующего и поэтапного процесса развития.

К сожалению, изменение сайта сопряжено с определенными опасностями.

Как избежать ловушек в проекте

Когда мы начинаем работу над новым проектом, перед нами открывается бесконечное множество возможностей. Мы возбуждены, полны энтузиазма и идей. Почему же тогда когда все подходит к концу, мы просто хотим запустить проект и никогда больше о нем не вспоминать?

Без разницы, полностью ли мы переделываем сайт или занимаемся умеренной перегруппировкой, для излишне неосторожных дизайнеров не будет недостатка в ловушках. Хуже всего, что нам свойственно повторять одни и те же ошибки. Перед тем как окунуться в новый проект, какое-то время обязательно обдумайте те общие вопросы, которые могут возникнуть, и то, как вы будете их решать.

Хотя любой проект индивидуален, ниже я представил самые большие проблемы, с которыми я столкнулся за 16 лет моей работы веб-дизайнером:

- Расширение рамок проекта.
- Принесение в жертву стиля и трендов.
- Создание сайта без учета того, будет ли возврат инвестиций.
- Негативная обратная связь.

Давайте рассмотрим каждую проблему по отдельности.

Борьба с расширением рамок проекта

Ничто так не бросает в дрожь любого дизайнера, чем фраза: «У меня есть идея!» – услышанная из уст заказчика.

Рамки каждого проекта почти неизбежно будут расширяться. Позицию заказчиков можно понять. Они не являются веб-экспертами, в отличие от нас, и поэтому не думают обо всем заранее. Только работая бок о бок с нами, они начинают реализовывать возможности.

Как же тогда поступить с изменением технического задания? Есть один выход – упереться рогами и сказать: «Нет». Но это может привести к конфронтации и разрушить ваши отношения с клиентом.

Методы, которые мы нашли для достижения успеха, – это составление списка пожеланий и идей, а также работа по этапам. Когда у заказчика или у нас возникнет идея, она будет вноситься в список пожеланий. Эти идеи не должны подвергаться цензуре или оценке, они просто должны добавляться.

К концу очередного этапа перечень пожеланий анализируется. Пункты тщательно выверяются, тем, которые остаются, отдается предпочтение, и они вносятся в следующую стадию разработки проекта.

Как только клиент ясно осознает этот процесс, у него создается определенное видение и формируются ожидания того, как будут осуществляться изменения. Клиенту также нравится быть все время «в процессе», что хорошо сказывается на повторных сделках.

Заказчик обязательно будет настаивать на том, чтобы некоторые его идеи включались в проект. Советую вам избегать споров по этому поводу на промежуточном этапе разработки. Сумейте объяснить заказчику, что реализация идей на данной стадии может навредить проекту. Предложите ему заморозить их до запуска следующего этапа, и обсудить потом.

Заморозка спорных вопросов до второй стадии имеет три преимущества:

- Заказчик гораздо реже навязывает идеи, как только видит свой восхитительный новый сайт.
- Создание нового сайта не будет тормозиться деталями, что вероятнее всего позволит запустить его вовремя.
- С новым живым сайтом ваши позиции в споре о том, что именно должно быть в его рамках, усиливаются.

Но не только обсуждение рамок проекта вселяет ужас в наши сердца.

Принесение в жертву стиля и тренда

Еще одно восклицание, от которого веб-дизайнеры бледнеют, это что-то вроде: «Мой сын подсел на Facebook. На нашем сайте должен быть Facebook». Не вопрос, Facebook – это не проблема. Но заказчик запросто может сказать: «У наших конкурентов крутой сайт. Мы хотим такой же» или «Нам нужен веб-два-нольный сайт».

Фишка в том, что заказчики часто кидаются с головой в то, что модно, престижно и успешно, и ждут, что мы будем делать это вместе с ними. Я не хочу сказать, что мы круче и лучше. Как веб-дизайнеры, мы любим новейшие тренды, будь то адаптивный дизайн, градиенты или красивые тени.

Но проблема в том, что модные тенденции будут меняться по запросам заказчика. То, что так нравится ему на момент начала работы над проектом, может стать ненавистным к моменту утверждения дизайна. Даже если клиент остается стойким в своих взглядах, через какое-то время сайт станет выглядеть несовременным, а это усложнит процесс внесения изменений и, возможно, приведет к тому, что клиент увидит вашу работу не в лучшем свете.

Самая мощная защита в борьбе с модой – это задать простой вопрос: «Зачем?» Ответить, что это «круто» или «в тренде», будет недостаточно. Нам нужно спросить, почему определенная идея хороша, и привести реальные деловые доводы. Мы должны увязать их с нашим выбором и выбором заказчика.

Если заказчик становится жертвой моды, не пытайтесь переубедить его. Наоборот, спокойно спросите его, почему он считает какую-либо идею хорошей. Зачастую мягкого подталкивания будет достаточно для того, чтобы заказчик понял, что он в сетях соблазна новизны.

Если это не даст желаемых результатов, действуйте дальше и задайте ему вопрос, откуда он ожидает возврата инвестиций.

Разработка сайта без учета возврата инвестиций

Большинство требований в заявках, которые я получал много лет, выглядели, скорее, как перечень пожеланий, чем исчерпывающий бриф. Я считал своим долгом отшлифовать каждое предложение так, чтобы оно было выгодно для бизнеса заказчика. И делал я это охотнее, чем в точности исполнял все, о чем он просил. Заказчики (как и все мы) соблазняются функциями и возможностями. Они не принимают во внимание стоимость внедрения в сравнении с возвратом инвестиций, потому что они некомпетентны в этом вопросе. Помогать тут им или нет, зависит от вас.

Помню, когда наше дизайнерское агентство только начинало раскручиваться, мы создавали сайты с многоязычной поддержкой. А так как в те годы многоязычная поддержка еще не являлась стандартной функцией, то и стоила она дорого. Я не задавался вопросом, для чего она нужна заказчику и уж тем более как будет переводиться их контент. В результате функция никогда не использовалась. В основном я внедрял их пожелания, а не удовлетворял нужды их бизнеса. Соответственно, я тратил кучу денег заказчика.

Чем больше функций вводится, тем больше сложностей и выше цена. Ваша задача – помочь заказчику придерживаться простых вещей.

Вы, возможно, уже заметили, что здесь красной нитью проходит одна тема: всегда поддерживайте простые решения. Ваша задача как веб-разработчика состоит в том, чтобы воздерживаться от излишеств как своих, так и заказчика. Вместо того чтобы глобально переделывать сайт, остановитесь на искусной перегруппировке. Вместо того чтобы

добавлять больше функций, определите набор простых функциональных возможностей и придерживайтесь его. Вместо слепого поклонения новейшим трендам сосредоточьтесь на простом, вечном и классическом.

Это гарантирует заказчику высокий показатель возврата средств, которые он вложит в сайт, работая с вами. А также сведет до минимума опасность последней ловушки при переделывании любых проектов: негативной обратной связи.

Что делать с негативными отзывами

Я уже говорил о том, что реакция людей на изменения редко когда бывает положительной. Некоторые из нас просто не любят изменений, а другие разочаровываются в них, потому что все изменилось не так, как они ожидали. В любом случае все изменения на сайте, вызовут реакцию и у пользователей, и у заинтересованных в бизнесе лиц.

Мы установили, что минимальные изменения уменьшат шансы на критику. Но это не всегда возможно, а даже если и получится, вы все равно получите негативную обратную связь. Не следует доверять первым откликам, будь они положительные или отрицательные.

Я вспоминаю, как мы обсуждали этот вопрос с Даниелем Буркой, когда он был ведущим дизайнером в Digg. Он рассказывал мне, как сложно ему было не реагировать на критику сразу же, после того как выходил в свет новый элемент его проекта. Он предпочитал тут же решать кажущуюся проблему. Но потом он научился тому, что если подождать пару недель, пользователи привыкают к изменениям и принимают их. В конце концов, это стало его стандартным методом. Он не делал дальнейших изменений до тех пор, пока элемент проекта не проживет хотя бы двух недель.

Когда заказчик или пользователь видят новую разработку, они дают поспешную оценку, которая редко отражает их конечное восприятие.

Кого-то новый дизайн может сразить наповал, и только со временем обнаружится, что сайтом совершенно неудобно пользоваться. Так же

кто-то может с первого взгляда возненавидеть изменения, а потом проникнуться к ним любовью.

Я овладел тремя простыми тактиками эффективного управления негативной обратной связью. Во-первых, в графике разработки проекта нужно оставлять промежуток между первым показом дизайна заказчику и внесением изменений в него. Это дает клиенту возможность привыкнуть к дизайну, перед тем как дать какой-либо отзыв.

Во-вторых, активно побуждайте клиента использовать это время для того, чтобы «переварить» дизайн. Объясните ему доходчиво, что первая реакция – не всегда самая правильная. Пусть заказчик вернется к дизайну через какое-то время, прежде чем он выскажет свое мнение.

И, наконец, заранее предупредите заказчика о том, что может произойти с новым дизайном, когда он выйдет в свет. Скажите о том, что пользователи могут выразить негативную реакцию. Критика обычно пугает заказчика. Ясно, что это может вызвать дрожь в его коленках и ухудшить положение вещей.

Возможно, заказчики испытают такое же напряжение, когда пропустят вашу разработку через себя. Возможно, что они тут же будут реагировать на каждый отрицательный комментарий, за исключением тех заинтересованных в проекте лиц, у которых не хватает времени на то, чтобы «впитывать» дизайн.

Предотвратите эту проблему, заранее обсудив ее с заказчиком. Вы сможете вернуться к этому моменту потом, если возникнет проблема. Заказчик убедится, что это обычное явление и вы к нему готовы.

Подготовка – это ключ к успеху. Это – путь к лучшему сайту для заказчика и наиболее стоящему проекту для вас. Но чтобы быть подготовленным на все сто, вы должны выполнить домашнее задание.

К исследованиям будь готов!

Когда сроки и бюджет ограничены, есть огромный соблазн сразу же погрузиться в процесс разработки дизайна. Однако делать это весьма неразумно. Во-первых, нам нужно четко понимать, что мы делаем и зачем мы это делаем. А для этого нужно провести некоторые исследования. Прежде чем объяснить, что я подразумеваю под этим, я хочу вкратце обрисовать, почему исследование так важно для процесса.

Зачем вам делать домашнее задание?

Есть две причины, по которым нужно проводить исследования до начала разработки любого проекта. Во-первых, это даст вам необходимые знания о проекте. Во-вторых (и, вероятно, что наиболее важно), вы будете во всеоружии, когда настанет время утвердить вашу работу.

Если мы имеем твердое представление о таких вещах, как бизнес-задачи, конкуренция, статистика и слабые стороны существующего проекта, тогда и объяснять наши дизайнерские решения, имеющие отношения к заказчику, будет значительно проще.

Доказывать, что пустое пространство вокруг контактной формы эстетически привлекательно, это не то, во что заказчик может вникнуть. Но если вы скажете ему, что свободное пространство поможет в решении бизнес-задачи, связанной с заполнением этой формы, – они вас поймут.

Итак, в чем же суть исследования?

Какое исследование я должен провести?

Количество проводимых вами исследований должно быть пропорционально ценности проекта. Размер неважен, вы должны провести хоть какие-то изыскания. Меня часто удивляет отсутствие базовой информации в обычной заявке. В требованиях часто не хватает таких фундаментальных вопросов, как:

- Для чего нам сайт?

- Чего мы ждем от сайта?
- Как мы сможем измерить его эффективность?
- Что будет делать пользователь на нашем сайте?

Как веб-разработчики, мы обязаны вытянуть эту информацию из клиента, до того как начнем работать над проектом. Бизнес-задачи должны встать во главе всего списка.

Заставить заказчика четко сформулировать свои бизнес-задачи может оказаться проблемой. Я часто задавал вопрос заказчикам, для чего им сайт и какие цели они преследуют. Внятного ответа я не получал.

Я уже не думаю о том, почему так было. Вместо этого я сажусь вместе с клиентом на старте проекта и методом мозгового штурма мы определяем с ним бизнес-задачи. Затем мы расставляем приоритеты по списку и определяем для них измеримые критерии успеха.

Например, такая расплывчатая бизнес-задача, как «увеличение продаж», должна превратиться во что-то более конкретное и напрямую связанное с призывом к действию. Так, «увеличение продаж» должно стать «увеличением числа горячих лидов через контактную форму сайта».

Расстановка бизнес-задач по приоритетам важна, потому что иногда они будут сталкиваться. Например, решение одной бизнес-задачи может принести клиенту больше лидов, в то время как решение другой задачи с более высоким приоритетом может демонстрировать пользователю продукцию. Поэтому, если кто-то предлагает заставить пользователя представить свой e-mail раньше, чем показать товар лицом, в противовес вы можете сказать, что демонстрация продукции важнее.

Постановка конкретных и умеренных бизнес-задач приведет к минимальному количеству исследований, проводимых в каждом проекте. Но если говорить о большинстве проектов, то тут мы должны

копнуть немного глубже. Один из способов – это проведение опроса заинтересованных лиц.

Ценность опроса заинтересованных лиц

Опрос заинтересованных лиц – это полуструктурированная дискуссия с теми, кто извлекает пользу с сайта. Это может быть тот, кто непосредственно работает на сайте (например, контент-редактор), или отдельные лица (например, руководители отделов), которые надеются с помощью сайта реализовать свои бизнес-задачи.

Опросы заинтересованных лиц дают четыре преимущества:

- **Они вводят веб-дизайнера в курс бизнес-требований**

Когда вы сталкиваетесь со сложной бизнес-моделью в новой области, опросы заинтересованных лиц ценны тем, что помогают вам понять требования заказчика. Из разговора с этими людьми вы узнаете о структуре организации и ее отрасли и определите, каким образом сайт будет соответствовать бизнес-требованиям заказчика.

- **Они дают более полный ракурс**

Многие веб-проекты крупных организаций влияют на многочисленные стороны бизнеса. Чтобы полностью осмыслить потенциальное влияние проекта, вы должны обсудить его со всеми участниками. Многие проекты заказываются одним отделом, у которого будет свое определенное видение целей. Беседуя с другими заинтересованными лицами, вы убедитесь, что проект, скорее, помогает, чем мешает другим людям внутри компании.

- **Они политически выгодны**

К сожалению, внутренняя политика является реальностью большинства крупных компаний. Это говорит о том, что нет недостатка в людях, которые хотят быть услышанными. Опросы заинтересованных лиц создают ту среду, где каждый может выразить свое мнение.

Я заметил, что если люди, работающие в организации, чувствуют, что к их мнению прислушиваются, они гораздо охотнее будут целиком и полностью поддерживать разработку. Опросы заинтересованных лиц также помогают утвердить дизайн, так как вы можете ссылаться на их комментарии как на оценку определенных элементов дизайна.

- **Они обеспечивают доступ к лицам, которые в действительности принимают решения**

При работе над большим проектом вы будете часто иметь дело с младшими сотрудниками, тогда как те, кто в действительности принимает решения, остаются за кулисами. И в этом может заключаться проблема. Опросы заинтересованных лиц позволят вам поговорить с этими людьми и понять, чего они хотят.

Умело построенные опросы гарантируют проекту ясные и определенные цели, из которых каждый в компании извлечет свою пользу и в то же самое время утвердит общую выгоду.

Опросы заинтересованных лиц сосредоточены на разработке и будущем сайта. Но многое нужно узнавать и из того, что уже есть на сайте.

Исследуем то, что имеем в настоящий момент

Мы начинаем работу над многими проектами с анализа онлайн-присутствия клиента. Этот анализ часто выливается в отдельный проект еще до того, как мы рассмотрим вопрос о переделывании сайта. Это дает заказчику возможность вместе с нами поработать над маленькими задачами, прежде чем выделить ресурсы на большой проект. И ему будет легче принять решение о долгосрочном сотрудничестве с нами без предварительной оплаты.

Допуская, что заказчик откровенен на фазе исследования, у нас есть пять вариантов (в большинстве случаев выберем только один или два):

- **Стратегический анализ**

Стратегический анализ помогает увидеть, как выглядит сайт сейчас. Он показывает сильные и слабые стороны сайта и дает советы по улучшению. Это хорошая возможность дать заказчику извлечь выгоду из вашего опыта и показать ему, что вы профессионал, а не просто человек,двигающий пиксели.

- **Эвристический анализ**

Так же, как и стратегический, эвристический анализ выявляет силу и слабость существующего сайта. Разница в том, что эвристический анализ делают с помощью оценки нескольких критериев (от 1 до 3). Эвристический анализ дает оценку сайту, но не в плане стратегии улучшения.

- **Конкурентный анализ**

Конкурентный анализ проводится по критериям, схожим с эвристическим, но относят их к конкуренции. Это дает ценное представление и помогает заказчику учиться на ошибках конкурентов.

- **Аналитический отчет**

Аналитический отчет освещает проблемы с сайтом и является отправной точкой для оценки изменений. Он может многое прояснить в поведении пользователя. Например, вы можете узнать, у какого из разделов сайта конверсия лучше. Этот анализ также покажет точки выхода и этим вскроет проблемные страницы.

- **Персонажи**

Персонажи – это мощный механизм ориентации на пользователей. Набор персонажей расскажет нам о пользователях сайта и о том, чего они пытаются достичь. Они дают не только демографическую информацию, но и понимание того, как пользователи пользуются сайтом, какие разделы посещают. И хотя это процесс трудоемкий, данный вид исследования очень ценен.

Мы обсудили выгоды, которые мы получаем, если понимаем смысл дизайнерских решений и можем их обосновать. Мы коснулись того, как это помогает клиенту лучше вас узнать. Последний вопрос, который важно обсудить, это слаженные взаимоотношения с клиентом при работе над проектом.

Принцип совместной работы при переделывании сайта

Многие взаимоотношения, которые я наблюдаю между дизайнерами и заказчиками, рушатся. Дизайнер подчиняется заказчику и заканчивает работу изолированно.

Традиционные взаимоотношения «клиент-исполнитель» опасны по ряду причин:

- Клиент не извлекает выгоды из богатого опыта дизайнера.
- Дизайнер разочаровывается, потому что его воспринимают только как человека,двигающего пиксели.
- Клиент чувствует себя исключенным из процесса работы над проектом.
- Недостаток общения между двумя сторонами приводит к непониманию. Из-за того, что дизайнер работает изолированно, велик шанс создать нечто такое, что заказчик не примет.
- Иногда заказчик вынужден принимать важные решения в области, в которой он просто некомпетентен.

Мой принцип таков – совместная работа с заказчиком на равных, вовлечение его в каждую деталь процесса, а также более активная собственная позиция в принятии решений.

Вместо работы над дизайном в одиночку, я работаю рядом с клиентом, показываю ему эскизы, прототипы, карты настроений и варианты макетов дизайна.

Когда утверждается окончательный дизайн, заказчик с большей охотой одобряет его по трем причинам.

Во-первых, дизайн для него не явится сюрпризом. Клиент будет видеть работу, которая проводилась над ним, и поэтому финальное творение – это ее естественный результат. Для того чтобы данный принцип сработал, вам необходимо привлекать клиента к разработке дизайна.

Во-вторых, клиент будет ощущать себя владельцем проекта. На каждом этапе процесса он будет получать обратную связь от вас, а поэтому станет считать проект больше своим, чем вашим.

И, наконец, заказчик будет в состоянии понять, как родился дизайн, потому что он внес свою лепту в его разработку. Плюс в том, что заказчик, который понимает и чувствует свою причастность к дизайну, сыграет хорошую роль, когда будет убеждать всех в его достоинствах. Для работы по такому принципу вам необходимо вовлекать заказчика в процесс разработки.

Как правильно работать над проектом

Если вы работаете с дизайном чисто интуитивно, вам, конечно, будет трудно взаимодействовать с заказчиками. Ключевым моментом в совместной работе клиентом является привлечение его к решениям по разработке. Я не предлагаю посадить заказчика рядом с вами, когда вы работаете. Но он обязательно должен участвовать в разнообразных этапах разработки.

Я вовлекаю заказчика в несколько этапов разработки дизайна:

- **На этапе обсуждения идей**

До начала своей работы над дизайном, я сажусь с заказчиком, чтобы обмозговать с ним идеи. Это момент хорош для того, чтобы взглянуть на сайт, посмотреть, что воодушевляет заказчика, и обсудить цвет, оформление и образы. Также это момент подходит для обсуждения персонажей.

- Основной вопрос, который я задаю: «Если бы сайт был знаменитостью, кто бы это мог быть?» Это помогает обеим сторонам визуализировать качества, которые вы сможете внести в дизайн.

- **После набросков карт настроений**

После первоначального мозгового штурма я иду и создаю карты настроений, где отражаю различные направления дизайна. Затем я обсуждаю их с заказчиком и продолжаю отшлифовывать. Я не слишком сильно усердствую. Нужно, чтобы их было легко создавать и чтобы я мог быстро воспроизвести их заново. Это мой шанс опробовать различные эстетические подходы одновременно.

- **Когда делаю прототипы**

Я всегда планирую встречу с клиентом (и даже с другими участниками) для обсуждения каркаса (wireframe). Мы садимся и делаем наброски ключевых страниц. Не нужно делать прототипы с высокой точностью – достаточно того, чтобы заказчик почувствовал, что он внес вклад в развитие данного направления. А я отшлифую их потом, после встречи.

- **Шлифовка макетов**

Когда придет время представить заказчику дизайн (или HTML-прототип), это не будет для него сюрпризом, потому что все основано на картах настроений и каркасах. Но я все же оставляю клиенту возможность обсудить напоследок любой вопрос, прежде чем перейду к финальному этапу.

Вы будете приятно удивлены, как мало итераций вам придется сделать в процессе выполнения проекта, если вы примените этот метод. Клиент часто счастливо одобряет дизайн всего лишь с несколькими изменениями, потому что он был вовлечен в процесс работы над ним с самого начала. Однако будьте осторожны: все может пойти наперекосяк, если вы покажете дизайн тем, кто не принимал участия в процессе разработки. И вот тут презентация дизайна просто необходима.

Презентация проекта

Все титанические усилия, которые вы прилагали в совместной работе с клиентом, могут пойти насмарку, если последнее слово осталось не за ним. Тесная работа с клиентами вероятнее всего приведет к тому, что внутренне они будут защищать дизайн, но тот, кто увидит его впервые, не сможет дать обоснованную оценку, потому что у него нет полной информации о разработке.

Единственный выход из данной ситуации, который я нашел, это презентация дизайна остальным лицам, которые также принимают решения.

Я уверяю вас, что таким образом они видят всю скрупулезность процесса моей работы, а также слышат мои идеи, вложенные в дизайн.

В идеале эта встреча может быть проведена в форме живого общения или конференции по телефону. Но это не всегда возможно. Другое решение, которое я нашел, это представить дизайн на рассмотрение с пояснениями. Оформленный как видеофайл с голосом за кадром, он дает хороший эффект. Это гарантирует, что все будут иметь возможность услышать объяснения процесса разработки.

Таким образом, зрители не дадут оценку дизайну по скудному первому впечатлению. Судя по своему опыту, скажу, что заказчикам эта идея тоже нравится. По их многочисленным отзывам, такой способ презентации дизайна впечатляет гораздо больше, чем статическое изображение. Видео также хорошо действует сейчас, когда сайты становятся все более и более динамичными.

Вместо статического изображения вы можете показать элементы JavaScript и даже, при необходимости, адаптивного дизайна. Лучше всего то, что сайту не надо быть кросс-браузерным, поскольку он работает в одном браузере. Конечно, дела могут пойти из ряда вон плохо, и неважно, насколько хорошо вы презентуете дизайн, если заинтересованные лица начнут давать обратную связь.

Обратная связь через тестирование

Мы любим считать себя разумными существами. На самом деле, мы не такие. На нас все имеет влияние, начиная с детского опыта до невыпитой чашки кофе сегодня утром.

Ни на что люди не реагируют так эмоционально, как на дизайн. Все знают, что им не нравится, и у всех нас разные вкусы. Один из моих клиентов отклонял даже цвет дизайна, потому что он напоминал ему о платье, которое носила его пожилая родственница!

С таким субъективным подходом к дизайну прийти к консенсусу в плане восприятия сайта трудно. Хотя существуют основные принципы дизайна, также присутствуют факторы, по которым одна и та же вещь кому-то кажется великолепной, а другому – ужасной. Если не обуздать процесс, то этап утверждения превратится в лотерею. К счастью, кое-что можно предпринять. Секрет кроется в психологии.

Так как мы люди, то мы любим считать себя последовательными во взглядах. Вот почему заказчик, который вовлечен в процесс создания дизайна, с гораздо меньшей охотой будет отклонять его. Тогда бы это было противоречием самому себе.

Другая причина, по которой данное свойство играет нам на руку, это то, что людям нравится считать себя логичными. Если вы представляете логичный, объективный аргумент в пользу дизайна, то слушатель гораздо быстрее примет его, ведь он должен реагировать логически. Желание быть последовательным и поддерживать свой имидж может даже превзойти его личную неприязнь к дизайну.

Вспоминаю, как несколько лет назад я делал дизайн в отрасли высшего образования. Сайт был прямо нацелен на то, что тогда называлось «поколением MySpace»[\[Z\]](#). Лично я ненавидел этот дизайн, и мой заказчик тоже.

Однако тестирование пользователей, на которых мы ориентировались, в конечном итоге показало, что живенький дизайн и кричащие краски

задают им нужный тон. Перед лицом этих фактов о нашем собственном мнении нам пришлось временно забыть.

Вот почему тестирование дизайна так важно. Оно отводит решения от субъективного мнения и приближает их к объективному исследованию. Оно также концентрируется на пользователе. Заказчик может ненавидеть дизайн, но если тот получает резонанс от пользователя и выполняет бизнес-задачи, то это должно подействовать на него.

А как вы будете уговаривать клиента принять метод тестирования при переделывании его сайта?

Продажа тестирования заказчику

Проблема тестирования в том, что на него тратятся время и деньги. Многие клиенты с неохотой платят за это. Им больше нравится опираться на своих «экспертов» и свои собственные предпочтения. Поскольку основная причина, по которой клиент не хочет проводить тестирование, кроется в деньгах, то вам придется сделать на них акцент при обосновании тестирования. Когда я убеждаю заказчика провести тестирование, я говорю ему о двух финансовых выгодах.

Во-первых, тестирование может значительно повысить эффективность сайта. Это гарантирует, что дизайн четко передаст ваши сообщения и побудит пользователя откликнуться на призыв к действию. В конце концов, это скорее заставит сайт генерировать значительный возврат на инвестиции.

Во-вторых, тестирование пользователей поможет сэкономить значительную часть денег. Без него вы можете пойти по неэффективному пути. Будут даже такие моменты, когда вы станете выбрасывать некоторый функционал, потому что тестирование покажет, что пользователю он не нужен. При своевременном и частом тестировании вы улавливаете проблемы с интерфейсом пользователя и набором функций до того, как потратить слишком много драгоценного времени.

Короче говоря, без тестирования пользователей процесс переделывания сайта будет проходить вслепую. Нет никакой гарантии, что результат будет эффективным. А в процессе непременно будут допускаться ошибки.

И раз уж заказчик принимает участие в тестировании, то вам нужно выбрать наиболее подходящий его вид.

Какой вид тестирования проводить

Есть несколько возможных вариантов, и выбрать правильный очень важно. Я остановлюсь на трех, которые я применяю чаще всего.

В среде веб-дизайнеров опрос как метод вытягивания информации из пользователей не используется широко. Но при определенных обстоятельствах он может стать эффективным. Я работал с клиентами из научных кругов, которые ставили под сомнение ценность тестирования при небольшом количестве участников.

На их взгляд, объем выборки был слишком мал, чтобы иметь статистическую важность. Опрос же оправдал себя как решение, потому что в него было вовлечено значительно большее количество людей.

Для достижения наибольшего эффекта нужно воспользоваться вопросами с ограниченным набором вариантов ответа, это даст возможность провести сравнение.

В опрос хорошо включить следующие пункты:

- Выбор между двумя подходами к дизайну, с едва уловимой разницей.
- Получение мнения по поводу того, правильно ли дизайн отражает ключевые качества, характеризующие бренд (например, свободный и небрежный либо профессиональный дизайн, консервативный или либеральный?).

- Вопрос пользователям, какими опциями из ограниченного набора они будут пользоваться.
- Проведение сортировки закрытых карт (где пользователь должен систематизировать страницы по категориям).

В инструментальных средствах (в сервисных программах, в серверах), поддерживающих этот вид онлайн-тестирования пользователей, нехватки нет. Я лично использую два – Verify4 и WebSort .5. Польза от опроса в том, что ответы подсказывают определенный тип дизайна, а клиент приобретает уверенность при принятии решения.

Как получить отзывы о дизайне

Второй вид тестирования пользователей – тестирование дизайна. Здесь проверяется, насколько эстетичен и удобен в использовании интерфейс. Обычно я провожу тестирование дизайна на начальной стадии разработки с относительно большой группой пользователей (20 и больше). Опрашивать их можно индивидуально или в малых группах. Я провожу несколько тестов, но наиболее распространенные – это тест на эмоциональную реакцию и флеш-тест.

В тесте на эмоциональную реакцию вы показываете пользователю дизайн и (как было сказано выше) просите его сделать выбор между ключевыми парами. Дизайн тесный или просторный, классический или современный? Смысл в том, чтобы оценить, соответствует ли дизайн тому образу, который вы представляете. Ключевые слова, которые выберет пользователь, должны содержать желаемые результаты. Если пользователи выбирают эти слова, я знаю, что я на правильном пути в разработке эстетики.

Иногда, правда, такие эмоциональные тесты могут преподносить сюрпризы. Мой коллега показывал один свой дизайн пожилой даме. Увидев его, она вдруг разрыдалась. Просто оказалось, что элемент дизайна – собачка – напомнила ей о ее больном питомце. Это говорит о том, что когда вы проводите тестирование дизайна, тенденции лучше всего искать по его результатам, чем по индивидуальным комментариям.

Флеш-тест основывается на контенте и визуальной иерархии. Вы показываете дизайн пользователям на несколько секунд и потом убираете его. Затем вы просите их воспроизвести элементы страницы.

Воспроизведенные элементы и их последовательность хорошо указывают на то, правильно ли расставлены акценты в дизайне. Например, если пользователю не удастся назвать ваш главный призыв к действию, значит, скорее всего, что-то нужно изменить. Оба эти теста могут проводиться в форме опроса и, таким образом, вовлекать намного больше пользователей. Но беседа с пользователями вживую имеет преимущества. Это помогает не только лучше понять их, но и дает вам шанс следить за их вопросами и копнуть немного глубже.

Юзабилити-тестирование

Наверное, самый хорошо известный вид тестирования – это проверка удобства пользования (юзабилити-тестирование). Как видно из названия, оно проверяет удобство и простоту использования вашего дизайна больше, чем его эстетический вид. Подобную тему широко раскрыли Стив Круг, Якоб Нильсон и Джаред Спул, поэтому я не хочу здесь слишком сильно углубляться в нее.

Я же хочу подчеркнуть, что успех юзабилити-тестирования зависит от многократных подходов в течение всего жизненного цикла проекта. В течение нескольких лет я откладывал тестирование пользователей на конец проекта, когда было уже слишком поздно вносить изменения. Тестирование на завершающей стадии также не дает возможности провести последующие, чтобы подтвердить эффективность усовершенствований.

Для эффективного тестирования его следует проводить на каждом этапе процесса переделки сайта. Мы должны тестировать абсолютно все, от эскизов, карт настроек и до завершеного дизайна. Проведение однократного теста недостаточно.

Когда и кого тестировать

Причина, по которой тестирование так часто сводится к формальной сессии в конце проекта, кроется в очевидных затратах на него. Люди думают, что найти участников и провести сессию – это процесс дорогой и трудоемкий. Позвольте мне убедить вас в том, что это не так.

Юзабилити-тестирование редко требует точного соответствия по демографическому признаку между выборкой респондентов и предполагаемой целевой аудиторией. Большинство преград, с которыми мы сталкиваемся и которые мешают удобству пользования, обычны для всех.

Для того чтобы набрать участников опроса нужно не больше и не меньше, чем ссылка на существующий сайт. Исключением является тестирование дизайна, для которого требуется приложить немного больше усилий, чем просто набор участников. Их количество должно быть сравнительно высоким, и подбор нужных людей действительно имеет значение. Но сами по себе тесты могут проводиться где угодно и без специального оборудования.

Только однажды я присутствовал на сессии по тестированию пользователей, которая проводилась в специальном центре тестирования с полупрозрачными зеркалами и подключенным видео. Но я думаю, что на самом деле это было менее эффективным, чем партизанское тестирование. У вас будет больше возможности регулярно устраивать тестирование, если вы будете проводить его облегченную версию.

В конце концов, тестирование помогает сделать сайты лучше и легче получить одобрение клиента. Это говорит о том, что клиенты хотят сказать свое слово, а поэтому мы должны уметь получать обратную связь от них.

Как быть с отзывами заказчика

Хотя тестирование реальных пользователей уводит заказчиков от собственного мнения, иногда им тоже нужно помочь правильно выразить свои отзывы о дизайне.

Мы ухудшим положение вещей, если спросим: «Как вы думаете?» или «Не могли бы вы дать мне свой отзыв?» Подобными вопросами мы стимулируем их выражать свое личное мнение. Вместо этого я задаю заказчикам ряд специальных вопросов, когда собираю их отзывы по определенному дизайну. Вот некоторые неплохие вопросы:

- Вы согласны с тем, что изменение дизайна отражает ценность бренда компании?
- Соответствует ли переделывание сайта тем бизнес-задачам, которые мы обговаривали?
- Соответствует ли переделывание сайта персонажам, которые мы обсуждали?
- Побудят ли пользователей утвержденные призывы к действию после переделки сайта?
- Отражает ли дизайн принятые карты настроений и прототипы?
- Все ли подходящие отзывы, полученные из тестирования, мы учли?

Весь ваш труд окупится авансом. Когда заказчик будет иметь возможность обращаться к утвержденным картам настроений, прототипам, результатам тестирования пользователей и т. п., он больше сосредоточится на бизнес-задачах и потребностях пользователей, чем на собственном мнении.

Одержимость личным мнением не единственная проблема. Часто обратная связь от клиента представляет собой список изменений в дизайне, которые вы должны внести по их желанию. К сожалению, эти изменения не всегда в лучшую сторону.

Я активно поддерживаю своих клиентов в том, чтобы они вносили свои предложения по улучшению дизайна. Ведь я же верю, что они могут дать ценную информацию, особенно если вы вовлекаете их в процесс дизайна и обучаете их.

Положение вещей становится угрожающим, когда клиент предлагает решение проблемы, которое он не может четко изложить. Например, клиент просит вас изменить цвет с голубого на розовый, не называя причины. А причина в том, что аудиторию составляют девочки в возрасте 9–12 лет.

Не зная сути проблемы, вы не предложите лучшее решение и не оцените, насколько к месту идея заказчика. Поэтому важно, чтобы он сначала мог высказать проблемы, а потом предложить решения.

Если у заказчика входит в привычку предлагать решения скорее, чем выражать проблему, то обычное напоминание часто возвращает их на место.

Не удастся – задайте вопрос «зачем?». Как я уже сказал, вопрос «зачем?» поможет клиенту разобраться в корне проблемы.

Суть данного раздела сконцентрирована на процессе работы с заказчиком при создании сайтов, ориентированных на бизнес-задачи. Мы увидели, как нужно проводить исследования и тестирование проекта, а также рассмотрели методы работы с заказчиками. Прежде чем подвести черту, давайте посмотрим, насколько долго может продлиться жизнь сайта, который вы создаете. И в особенности как обеспечить будущее сайту после его переделывания.

Вечная переделка

Я должен признаться, что как веб-дизайнер я не всегда был дальновидным в своем методе создания сайтов для заказчиков, что шло во вред мне и им. Уверен, что я не один такой.

Мы совершенно не виноваты в этой недальновидности. По большому счету она – порождение культуры, о которой я говорил ранее. Когда клиент каждые несколько лет утверждает новый сайт (зачастую, всякий раз с разными дизайнерами), мало толку планировать его долгое существование.

Я уже объяснял недостатки подобного подхода, как для клиента (с точки зрения эффективности сайта), так и для дизайнера (с точки зрения дальнейшего сотрудничества). Поэтому мы должны строить непрерывные рабочие отношения с заказчиками, а не ограничиваться однократными встречами. Это срабатывает независимо от того, дорабатываем ли мы сайты или переделываем их с нуля.

Как создать программу непрерывной работы

Как мы уже обсудили, есть веский довод для того, чтобы непрерывно инвестировать в сайт. Однако представлять что-то и осуществлять это на практике – две разные вещи. Важно настроить механизмы, которые обеспечат непрерывные инвестиции.

Я предложил составлять перечни пожеланий и поэтапное развитие сайта как способы подтолкнуть клиента к планированию. Но это не единственные варианты. Раз в три месяца мы устраиваем с заказчиками встречи, где обсуждаем, как можно усовершенствовать их сайты. Также мы проводим ежегодный анализ сайта и предлагаем варианты его усовершенствования на будущий год.

Какой бы механизм вы ни использовали, будь то инфобюллетени по электронной почте или ежегодные анализы, цель одна: показать заказчику, как сайт может перейти на следующий уровень. И помните, что вы должны не просто представить ему новые функциональные возможности и техники. Вы должны объяснить бизнес-выгоды, которые можно извлечь из этого. Только тогда они увидят возврат инвестиций, которые сделают, воплощая ваши идеи.

Однако кроме текущей программы инвестирования есть и другие способы обеспечить сайту его дальнейшее будущее. Существуют еще технические решения.

Использование технологий для обеспечения будущего сайту

Нам следует помнить, зачем и что мы делаем как веб-разработчики. Если мы об этом забудем, то начнем практиковать неправильные вещи. Речь идет о веб-стандартах.

В основе веб-стандартов лежат простые принципы: мы должны разделять контент, дизайн и логику работы сайта. Одним из многих преимуществ этого подхода является то, что это облегчает изменение сайта в будущем.

Веб-стандарты помогают обеспечивать будущее сайтов. Но все равно я вижу тревожное количество сайтов, созданных с CSS, HTML и JavaScript, в которых это разделение не так явно: строки JavaScript либо встроены непосредственно в код HTML, либо рассчитаны на присутствие определенных элементов HTML, а тот, в свою очередь, «нашпигован» классами и идентификаторами элементов, чья единственная цель – задание внешнего вида страницы. Не поймите меня неправильно: я не ярый борец за чистоту кода. Я знаю степень неминуемости таких накладок в коде. Когда вы пишете код, спросите себя, как ваши решения повлияют на обновления через год-другой.

Следующий вопрос, который возникает, когда мы говорим о будущем сайта, – это поддержка браузеров. Как веб-дизайнеры, мы много говорим о поддержке старых браузеров, но мало о поддержке будущих. Мы должны обеспечивать доступ к сайтам через старые браузеры, но нам также нужно думать и о будущей жизни сайта.

Поддержка старого браузера, чья рыночная доля снизится за счет появления новых, имеет мало смысла в финансовом плане. Я – ярый сторонник создания сайтов с использованием HTML5 и CSS3, так как мы знаем, что эти технологии будут все больше и больше поддерживаться. Мы создаем сайты для будущего, не закликиваясь на браузерах, чья рыночная доля идет на убыль. Понятно, что можно установить баланс, но кто-то может возразить, что поддержка старых браузеров – не лучшее вложение и без того ограниченных ресурсов. Говоря о поддержке, я не могу не посвятить часть раздела мобильному будущему сайта.

Планирование мобильного будущего сайта

Как веб-разработчики, мы входим в азарт от мобильности. Однако мобильность все еще не стоит на повестке дня у многих заказчиков.

Для многих из них использование мобильности еще сравнительно невысокое, и поэтому они неохотно вкладываются в него.

Несмотря на это, мало кто выражает свой скептицизм по поводу того, что мобильность может сыграть существенную роль в будущем сайта. Это означает, что даже если поддержка мобильности сайта не является одной из бизнес-задач компании в ближайшем будущем, клиентам нужно уже сейчас планировать ее.

Шаги, которые следует предпринять, будут зависеть от того, какую мобильную стратегию предпочтет клиент. У него есть несколько вариантов, и нам надо рассказать клиенту о них.

Для начала он должен решить, что он хочет: приложение или сайт. Приложения ориентированы на задачи и узко специализированы, тогда как сайт направлен на контент и его возможности намного шире. Если заказчик выбирает приложение, то вопрос в том, какое оно будет: основанное на веб-технологиях или «родное», т. е. написанное, например, под конкретную мобильную платформу. Это сложное решение, которое мы не можем подробно рассмотреть в этом разделе[\[8\]](#).

Так или иначе, здесь представлены некоторые моменты, которые вам необходимо обсудить с заказчиком:

- Какие «родные» характеристики должны быть доступны?
- Вы можете позволить себе разработку под множество платформ?
- Вы захотите разделить доходы с владельцем магазина приложений?
- Ваше приложение будет доступно через магазин приложений?
- Есть ли какие-либо выгоды от распространения приложения через такие магазины?

Когда клиент останавливается на сайте, то и выбор становится проще. Если сайт будет переделываться глобально, то сейчас самое время

сделать его отзывчивым (т. е. заставить его реагировать на доступное экранное разрешение). На маленьких экранах (таких как смартфоны) макет упрощается для удобства использования, тогда как на больших экранах у вас будет обычный макет под настольные компьютеры.

Если сайт не планируется серьезно переделывать в скором времени, тогда предпочтительней выбирать адаптивный подход. Адаптивный дизайн легче осуществлять на существующем сайте, это позволяет изменить формат ключевых экранных размеров (таких как экран планшетного компьютера).

Однако адаптивный дизайн, так же как и отзывчивый, подойдет не для каждого макета, и хотя адаптивный сайт может выглядеть великолепно на сегодняшних устройствах, есть вероятность, что он окажется неоптимизирован для будущих.

В настоящее время Интернет переживает еще одно глобальное развитие, где применяются новые технологии, новые устройства и методы. И нам, веб-разработчикам, выпала миссия помочь нашим заказчикам подготовиться к встрече с этим прекрасным новым миром.

Что дальше?

В этом разделе мы осветили много тем, но не слишком углублялись в них. Основная идея была в том, чтобы научить вас думать на более высоком уровне, чем только о технологии и дизайне, концентрироваться на сложных проблемах, с которыми сталкиваются клиенты, и обеспечивать эффективность их сайтов насколько это возможно. Все, что вам нужно, это взять на вооружение несколько принципов. Вот некоторые рекомендации:

- **Будьте кем-то большим, чем просто исполнителем проекта**

Неустанно работайте над тем, чтобы изменить ваши взаимоотношения с клиентами. Прекратите быть просто человеком,двигающим пиксели, работайте в команде с клиентами. Умейте возразить им, особенно когда они требуют глобальной переделки сайта. Вместо этого предложите им преобразование и вовлекайте их в этот процесс.

- **Подготовьтесь перед стартом**

Сдерживайте свои порывы приступить к переделыванию с ходу; сначала лучше выполните домашнее задание. Убедитесь, что вы готовы к тому, что проект может разрастись. Убедитесь с помощью тестирования заинтересованных лиц и аналитики действующего сайта, что вы понимаете бизнес-задачи.

- **Тестируйте все**

Не полагайтесь только на свой опыт разработчика. Проводите тестирования для того, чтобы процесс разработки был менее субъективным, и для обоснования изменений. Тестирование также поможет вам подсчитать потенциальный возврат вложений, который клиент ждет от вашей работы.

- **Планируйте будущее**

Разработайте с вашим заказчиком действующую программу развития, которая обеспечит будущее сайту. Подтолкните его к тому, что нужно принимать во внимание мобильный Интернет и будущие браузеры, а не ориентироваться на старые технологии, чья доля идет на убыль.

Вот и все. Надеюсь, что я не слишком деморализовал вас всеми этими разговорами о возврате инвестиций (ROI) и бизнес-механизмах. На самом деле это может быть захватывающей штукой. Мы разработчики, а не художники. Основное отличие в том, что мы создаем вещи, которые решают проблемы. Иногда это проблемы пользователей, но иногда это проблемы, которые испытывают сами клиенты.

Об авторе

* * *

Пол Боуг (р. 1972) вырос в Уашворде (Великобритания), учился в Университете Портсмута и начал свою веб-карьеру в IBM в 1994 году,

когда веб-дизайн включал в себя блокнот, серый фон и не имел возможности верстки. Пол – соучредитель «Хэдскэйп» (Headscape – компания, работающая в области веб-дизайна и адаптивного дизайна). Здесь он работает над веб-стратегиями, пишет об успешных сайтах и выступает на конференциях по всему миру. Время от времени он выпускает подкасты. Пол живет в небольшом городке в самом сердце английской сельской местности и активно участвует в жизни местной церкви. Он любит проводить время, играя в Skyrim (компьютерная игра). Пол считает себя любителем животных, потому что его отец – фотограф дикой природы, поэтому у них дома все время были все: от совы до оленей. Его любимые цвета – все оттенки серого. Самый большой урок, который он изучил за свою карьеру, – быть страстными и полным энтузиазма, никогда не терять любовь к Интернету и играть с инновациями. Послание Пола к читателям гласит, что успешность проекта обеспечивается сотрудничеством с клиентом. Возможно, вам непросто работать с клиентом, но без его знаний о бизнесе и сфере деятельности сайт будет провальным. Более того, клиент должен любить ваш дизайн. Если ему не нравится дизайн, он не будет вкладываться в него и использовать в полную силу. Вы обязаны работать совместно.

О рецензенте

* * *

Коллис Тайед – соучредитель и генеральный директор Evato. Он начинал работать в качестве веб-дизайнера, делая макеты в Photoshop и нарезая их для разработки большинства ранних сайтов Envato. Сейчас Коллис больше времени проводит, планируя, разрабатывая стратегии и работая с электронной почтой, но веб-дизайн всегда будет в его сердце!

Выбор платформы: технические решения для переделки вашего сайта

Автор: Рэйчел Эндрю

Рецензенты: Райан Карсон, Харли Финкельштейн

Этот раздел написан для тех, кто участвует в планировании работ по переработке сайта. В предыдущем разделе Пол Боуг рассматривал бизнес-сторону полной переделки и поэтапной доработки вашего сайта. В этом разделе мы обсудим некоторые технические проблемы, с которыми вы можете столкнуться в своей работе. Для переделывания интерфейса сайта не всегда требуется полностью переписывать код. Мы дадим вам несколько советов, как утрясти этот вопрос.

Прежде чем окунуться с головой в глобальное переделывание сайта, мы, по совету Пола, должны узнать все о существующем сайте и серверах, которые им управляют. При переработке системы очень легко потерять ценную информацию. Поэтому даже если вы полностью переделываете сайт или выбираете новую платформу, узнайте все, что только можно, о существующем сайте, чтобы не повторять ошибок ваших предшественников.

Прежде чем что-то предпринять, вы должны вникнуть во все технические требования. Возможно, переработка сайта добавит новые функции, которые нужно будет поддерживать. Например, для сайта электронной коммерции вам нужно знать, как принимаются платежи по Интернету и какие сервисы нужны для этого. Хостинг, который вы используете, должен удовлетворять техническим требованиям нового сайта или доработке существующей платформы. Поэтому не забудем подумать о том, как выбрать хороший хостинг.

Начало работы над проектом – прекрасный момент, чтобы оценить существующий сайт и убедиться, что как специалист вы в хорошей форме. В конце этого раздела вы найдете идеи о средствах разработки, контроле версии и об изменении версии существующего сайта. Как и в предыдущем разделе, мы обсудим здесь многое, только не слишком подробно. Формат книги не позволяет. Наша главная цель – заставить вас задуматься о различных аспектах сайта и выбрать те основы, которые вы будете применять в своей дальнейшей работе

Для кого этот раздел?

Неважно, с кем и для кого вы переделываете сайт: привлекаете ли подрядчиков для создания собственного сайта или работаете в команде разработчиков в качестве нетехнического специалиста, этот раздел – для вас. Изучив ее, вы поймете процесс переделывания, и это принесет вам пользу.

Возможно, вы тот, кто занимается разработкой сайта постоянно либо для себя (в этом случае заказчик вы сами), либо для стороннего заказчика или работодателя. В этом случае все в этом разделе будет для вас интересным. Для разработчика переделывание сайта – это фантастическая возможность. Сайт или приложение, которые у вас есть сейчас, имеют не только сильные и слабые стороны, но и базу пользователей, и посещаемость. Все это вы сможете изучить.

Проигнорировав на свой страх и риск эти данные, в конечном итоге вы просто воссоздадите те же самые проблемы, которые существуют и сегодня, либо не включите необходимые пользователям важные функции. Когда вы занимаетесь кардинальной переделкой сайта, у вас есть возможность узнать о его болевых точках и проблемах от пользователей и владельца. Система управления контентом (CMS^[9]) очень сложная, поэтому сайт не обновляется? Клиенты интернет-магазина постоянно звонят и просят помочь им сделать заказ? Дизайн ограничивает возможность добавления текста и фотографий? Если вам удастся решить эти проблемы – создать CMS, которой люди будут пользоваться с удовольствием, или наполовину сократить количество звонков от обескураженных покупателей, – вы получите бескрайнюю благодарность вашего клиента или начальника. Согласитесь, приятно знать, что ваша работа улучшает чью-то жизнь или помогает развить бизнес.

Если разработкой сайта занимается подрядчик, этот раздел все равно будет интересен для вас. Мы не станем слишком глубоко копаться в технических темах. Общее понимание вопросов, которые обсуждаются в этом разделе, сделает ваше общение с разработчиками легким и непринужденным.

Теперь допустим, что ваша роль – нетехническая. К примеру, вы работаете исключительно над графическим дизайном, либо

занимаетесь управлением проекта, либо копирайтингом. Опять же понимание того, что делают разработчики, только поможет общению.

Узнать все о действующей платформе

Как уже было сказано, все, что вы знаете о сайте, который собираетесь переделывать, очень важно. Поэтому нужно не только проанализировать сайт самому, но и поговорить с людьми, которые им пользуются: с его владельцами, посетителями и теми, кто его поддерживает.

Поинтересуйтесь, что им нравится в существующем сайте. Это может быть все, что угодно. Например, клиенту нравится, как выглядит сайт и как в нем представлен бренд, или, возможно, он находится на хороших позициях в поисковой выдаче. Пользователи могут сказать, что сайт удобный и можно обойтись без всяких прибабасов, которые так любят заказчики.

Поговорите с теми, кто обновляет контент и узнайте, что они думают о функциях существующего сайта. Если вы решите сменить платформу, берите на карандаш все, чем люди пользуются в своей работе. У многих она строится полностью вокруг системы. Бывает, что, отняв у людей возможность вести отчетность по заказам или делать записи в блог, вы явно усложните им жизнь. Если вы не заменяете эти функции, то вам лучше предоставить систему более высокого качества!

В каждой системе есть вещи, которые сводят пользователей с ума. Но даже если вы обслуживаете сайт, не считайте себя знатоком в них. Много раз я сталкивался с тем, когда недостаточно подкованные технически пользователи думали, что неполадки в системе возникли по их вине и поэтому молчали. И всякий раз, когда не удавалось сохранить данные, они думали: «Опять я это сделал!» – и повторяли все заново. Вместо того чтобы поднять проблему, пользователи списывали ее на то, что не умеют пользоваться компьютером.

Но даже если пользователь и виноват, а система допускает постоянную потерю данных, ваша задача – принять превентивные меры. И все же вопиющие ошибки часто не регистрируются по несколько месяцев, а пользователи предпочитают молчать и месяцами работать с ними. Если вы собираетесь сохранять уже существующую

систему, отыщите и исправьте эти болевые точки. Поверьте, те, кто пользуется системой каждый день, смогут тогда вздохнуть спокойно.

Есть системы, которыми люди пользуются в своей ежедневной работе постоянно. Так вот неплохо было бы отследить этот процесс. Что они делают изо дня в день? Очень часто человек даже не в курсе, что новая система может избавить его от изнурительной работы. Я видел пользователей, которым приходилось по несколько раз вводить в систему одни и те же данные. К примеру, они вводили одну и ту же информацию в разные места или копировали данные из одного документа в другой, тогда как очень простым решением стала бы автогенерация отчетов или добавление в CMS скрипта, который копирует данные из одного места в другое. Если вы будете просто смотреть на сайт с управляемым контентом или систему электронной коммерции, то ничего не увидите. Садитесь с рядом с администратором и наблюдайте, как пользователи работают в системе.

Если вы переделываете сайт, то придется решать не только видимые проблемы. Вам нужен сайт с такими функциями, которых у него нет сейчас? Хотите продавать онлайн? Решили наконец-то, что сайту нужна система управления контентом? Текущей CMS сложно пользоваться, или она не поддерживает контент, который вы хотите создавать?

Если вы будете знать, каких функций не хватает сайту, то решите, нужно ли вам сменить систему или дорабатывать ее дальше. Как только вы поймете, как пользователи и администраторы работают с сайтом или приложением, можете начинать собирать техническую информацию для переработки

Сбор технических требований

В предыдущем разделе мы обсуждали требования к сайту. Здесь мы поговорим о том, как выполнить их технически. Остерегайтесь заявлений типа: «Мы хотим, чтобы все было так, как в текущей системе» – если только не вы ее создали! Если клиент этого хочет, убедитесь, что вы получили подробную информацию обо всем том, что должно быть в новой системе.

Рисунок 2.1. Любой, кто выбирает CMS, столкнется с массой опций. Чтобы решить, что сработает для вашего проекта, нужно понимать технические требования

Не сделаете этого – готовьтесь к тому, что перед самым запуском вас спросят: «А как эта система поддерживает задачу, которую пол-офиса делает каждый день?» (хоть это и не так на самом деле). И таким образом заставят вас вносить значительные дополнения в проект. Знаю это по своему опыту!

Управление контентом

Почти всем сайтам требуется та или иная форма управления контентом. Это может быть обновление страниц, добавление продукта в магазин или редактирование текста в приложении. В какой степени клиенту необходимо управлять контентом и кто будет его редактировать? В этом разделе мы будем использовать термин CMS в очень широком смысле, для описания любого инструмента редактирования контента – от простого текстового редактора до полнофункциональной корпоративной CMS бизнес-уровня.

Вот некоторые мысли по поводу выбора CMS:

- Всем ли редакторам (контент-менеджерам) нужны одинаковые права доступа?

- Будут ли управлять контентом специальные сотрудники или это станет частью работы других людей?
- Нужно ли поддерживать многоязычность?
- Какой тип контента будет редактироваться?
- Какие нужны средства редактирования?

Всем ли редакторам (контент-менеджерам) нужны одинаковые права доступа?

Итак, для работы на сайте нужен один редактор (например, владелец малого бизнеса) или несколько? Если второй вариант, то все ли редакторы будут иметь равные права? Или один будет иметь доступ к каким-то разделам системы, а другие нет?

Вот несколько сценариев:

- Сайтом большой компании управляет один человек, который утверждает весь контент, а в подготовке контента принимают участие несколько сотрудников. Клиент хочет, чтобы эти редакторы могли создавать контент и отправлять его на рассмотрение. После одобрения, он публикуется управляющим редактором.
- Компания хочет, чтобы отдел кадров мог публиковать и снимать вакансии на сайте и вел раздел, который касается вопросов найма сотрудников. У этих пользователей не должно быть прав для внесения изменений в другие разделы.
- Владелец сайта электронной коммерции не хочет, чтобы его редакторы имели доступ к собранным в системе отчетам по продажам или информации о покупателях. В то же время он хочет, чтобы его бухгалтер имел доступ к данным о продажах, и ни к каким другим больше.
- Владелец сайта хочет, чтобы все желающие могли писать в блог, но не трогали контент других разделов.

Будут ли управлять контентом специальные сотрудники, или это станет частью работы других людей?

Очень важно понять, что могут люди, которые создают и редактируют контент. Речь здесь не только о технических умениях, к примеру хорошо ли они знакомы с CMS, но и о том, насколько они чувствуют дизайн и контролируют, как все выглядит. Также проверьте их копирайтерские способности. Если вначале контент пишет опытный копирайтер, а после поддержкой занимаются владелец или сотрудник, не занимающийся копирайтингом, то CMS должна помочь им понять, что и как писать. Об этом – в статье «Ваша CMS как куратор дизайна и контента»[\[10\]](#).

Нужно ли поддерживать многоязычность?

Даже если вы запускаете одноязычный сайт, но в ближайшем будущем нужно будет поддерживать несколько языков, подготовьтесь к этому заранее. Добавить на сайт поддержку нескольких языков гораздо сложнее, чем заложить эту функцию сразу. Конечно, это требование ограничивает выбор вариантов готового ПО.

Если сайт нужно переводить, выясните, как будет проходить этот процесс, чтобы поддерживать его в системе. Переводчики будут просто переводить весь контент в Word-документах или чем-то подобном, а потом отправлять редактору, чтобы он вносил их в систему? Не будет ли переводчику проще читать и переводить контент прямо в CMS?

Какой тип контента будет редактироваться?

У большинства сайтов, которым требуется CMS, относительно статические страницы и четкая стандартная древовидная структура. Наиболее логично создавать такой контент постранично, чтобы администраторы легко могли найти страницы для редакции.

В некоторых сайтах основная функция – блог и несколько поддерживаемых страниц. Тогда вы с заказчиком можете остановиться на использовании такой CMS, как WordPress. В ее ядре –

функциональный и качественный блог, в ней также можно добавлять отдельные страницы.

Чем больше сайт, тем серьезнее требования к контенту. Например, сейчас мы переделываем сайт для фестиваля искусств. Многие годы фестиваль остается привлекательным за счет подборки ценных видео- и аудиоматериалов, а также тысяч высококачественных фотографий, которые ежегодно снимают профессиональные фотографы

В сайтах с постраничной структурой предполагается, что контент-редакторы будут каждый раз отыскивать в архивах важный материал и перезагружать его заново на сайт. В данном случае это не лучший способ. Вместо этого мы создали отдельный медиасервер, со средствами поиска и маркировкой материалов. Все это для того, чтобы их было легко находить и вставлять в страницу. CMS готовит фотографии нужных размеров, включая те, что нужны для адаптивного дизайна. Изучив архивы материалов и, оценив их объем, мы поняли возникающие в связи с этим проблемы. Это позволило нам предложить такое решение.

Есть сайты, которые больше похожи на веб-приложения, неважно, система ли это электронной коммерции, или традиционные приложения. Эти системы ориентируются не на постраничную структуру (хотя они и могут включать в себя некоторые страницы), а на различные виды контента, требующие обновления.

Слишком часто в таких системах небольшие тексты (например, текст на кнопке) жестко кодируются в самом приложении. И поэтому только разработчик может изменять текст в призыве к действию. В идеале должно быть так, чтобы рядовые пользователи могли его редактировать.

A/B-тестирование

На некоторых сайтах, в особенности на тех, что продают товары, вам нужно тестировать разные версии страниц, чтобы узнать, какой контент, макет страницы или взаимное расположение блоков приводит к большему числу конверсий.

В зависимости от типа сайта конверсии[\[11\]](#) могут быть такие: покупка товара через сайт, подписка на получение пробной версии продукта, на e-mail-рассылку или заполнение формы. Если потребуются такой вид тестирования, как он будет осуществляться с помощью CMS? Если часть посетителей нужно отправить на одну версию страницы, а остальных – на другую, здесь не обойтись без стратегии. Об этом виде тестирования вы можете прочесть на сайте Smashing Magazine в исчерпывающей статье «Исчерпывающее руководство по A/B тестированию»[\[12\]](#) либо на сайте СилаУма (примеч. ред.)

Электронная коммерция

Добавить функцию электронной коммерции на сайт может оказаться простым делом. Возможно, достаточно будет всего лишь поставить несколько кнопок «Купить сейчас» платежной системы PayPal[\[13\]](#). Но могут быть и трудности, например, если вам придется настраивать свой магазин на поддомене готовой платформы онлайн-магазина или разрабатывать свой собственный. Конечно, сегодня процесс продажи товара напрямую через сайт не надо усложнять. Но все же есть над чем поломать голову, когда вам предстоит сделать правильный выбор в зависимости от вида продукции. В этом разделе мы пробежимся по некоторым из вариантов. Несомненно, вам будет это на руку, если вы в первый раз внедряете функцию электронной коммерции – ведь это все равно что ступить на необитаемый остров.

Что продаем?

Может быть, в вашем онлайн-магазине продаются товары, которые отправляются покупателю по почте или с курьером? Или, возможно, товары, которые поставляются электронно, такие как электронные книги, музыка или ПО. Также транзакциями на сайте могут быть пожертвования или подписка на сервис. Если товары скачиваемые, то подумайте над тем, как они будут доставляться клиенту после оплаты.

Как покупать?

Один ли товар будет продаваться (например, электронная книга), или посетители смогут просматривать товары и добавлять разнообразные

позиции в свою корзину? Можно ли будет выбрать товар по характеристикам (например, футболку по размеру и цвету)? Нужно ли делать категоризацию товаров, чтобы упростить их поиск? Товар будет ограничиваться одной категорией или его можно будет найти в нескольких? Пригодятся ли ярлыки и ссылки среди дополнительных или связанных товаров (скажем, позволяющие владельцу продвигать аксессуар к продукции)?

Будут ли на сайте специальные предложения – «При покупке одной вещи, вторая – в подарок!», «Скидка 20 %», «Две вещи по цене одной!», «Купите X, получите Y за полцены!»? Реализовать подобные предложения в созданной по заказу системе может быть достаточно непросто. А если вы используете коробочную CMS, то нужно знать, сможет ли она поддерживать их.

Дьявол (как и бюджет) в деталях. Из всех сайтов, над которым я работал, наиболее склонны к расширению рамок функционала онлайн-магазины. При планировании подумайте обо всех заданных выше вопросах. Конечно, есть много чего заманчивого и желание иметь как можно больше всевозможных функций велико. Но если вы создаете CMS самостоятельно и не хотите использовать коробочные решения, то лучше старайтесь разрабатывать более простые вещи. Ведь это значительно экономит вам время.

Рисунок 2.2. Вещь, которая, кажется простой, на самом деле может иметь ряд опций (например, футболка может быть женской или мужской и разных размеров)

Учетная запись и отслеживание заказа

У пользователей может быть опыт использования учетной записи (аккаунта) и отслеживания заказов. Возникают вопросы: обязательно ли пользователям создавать учетную запись? Смогут ли они отслеживать свои заказы и видеть их путь от «В обработке» до «Отгружено»? Аккаунт должен иметь такие основные функции

управления, как восстановление забытого пароля или обновление контактных данных.

Как принимать оплату?

Вероятно, вам понадобится принимать платежи от покупателей по кредитным и дебетовым картам. Есть несколько вариантов, разных по сложности и цене. Обычно платежи обрабатываются с помощью платежной системы, аккаунта продавца и платежного шлюза и программы-посредника SaaS (программное обеспечение как услуга). Принимать платежи онлайн через платежную систему – легко. Плюсы здесь в том, что создать аккаунт несложно, а интеграция может быть различной: от простого варианта – встраивания кода кнопки на страницу, до полной интеграции с вашей системой.

На платежном рынке есть и другие игроки, но большинство из них действуют в немногих странах. Причина в том, что большая часть законов о процедуре оплаты в каждой стране имеет свою специфику. Если вы будете принимать регулярные платежи, то вам могут пригодиться *Chargify* и *Recurly*. И хотя сейчас они доступны только в США[14], *Stripe* выглядит многообещающе как метод принятия платежей онлайн.

Чтобы напрямую принимать платежи по карте, избегая посредников, вам нужно завести аккаунт онлайн-продавца. Это позволит принимать платежи по кредитке и переводить деньги на ваш банковский счет. Если у вас есть действующий банковский счет для офлайновых продаж, вероятно, вы не сможете воспользоваться им для онлайн-овых транзакций. Транзакции через Интернет не совсем безопасны, поэтому прежде чем начать выполнять их, свяжитесь со своим банком. Банк посоветует защитить свой платеж, в большинстве случаев через провайдера платежных сервисов PSP (payment service provider), иногда называемым платежным шлюзом.

Рисунок 2.3. *Stripe.com* – это новый игрок на рынке, предлагающий простой способ приема платежей

Но что вы точно не должны делать, так это сохранять данные кредитной карточки, чтобы позже ввести их в терминал офлайн. Это противоречит условиям коммерческого договора. Поэтому, пока банк не даст вам свое письменное согласие и вы не выполните требования PCI DSS[\[15\]](#) (о них мы расскажем позже), просто не делайте этого.

Платежный шлюз

Платежи через шлюз позволят вам принимать оплату от клиентов по карте, проверять ее номер и состояние счета, а после этого безопасно переводить платеж в ваш банк. Взаимодействовать со шлюзом можно двумя способами:

- **Через страницу платежей**

Пользователь переходит с вашего сайта на защищенную страницу провайдера платежного сервиса и вводит данные карты.

- **Через интеграцию по API[\[16\]](#)**

Пользователь вводит данные своей карточки на вашем сайте (на странице с сертификатом защиты, который поддерживает протокол SSL), и эти данные затем отправляются в шлюз. Ваш сайт в этом случае служит посредником; пользователь не в курсе, какие транзакции проводит банк, а видит их только на вашем сайте.

Преимущества интеграции платежной страницы в том, что ваш сайт вообще не имеет отношения к данным карты, поэтому вы не несете ответственность за безопасность клиента. Но есть один большой минус. Это то, что вы уже не сможете контролировать процесс оплаты, потому что на финальном этапе требуется собрать все данные и отправить их на платежный сервер. Кроме того, редко встречается возможность настроить страницу оплаты в своем фирменном стиле или даже просто вставить в нее логотип.

Необходимость уводить клиента с сайта очень беспокоит многих владельцев магазинов. Они боятся, что пользователи будут отменять транзакцию до того, как попадут на платежную страницу WorldPay

или другого сервера. Но если отправлять клиентов на сайт известного банка для введения данных карты, то они будут больше доверять вам.

Скажу о себе. Когда на каком-то незнакомом сайте (например, мелкого розничного торговца) меня просят ввести данные моей карты, я начинаю нервничать и думать, а что с ними будут делать? Высветится номер моей карты в открытом виде на экране? Сохранятся ли ее данные в базе на сервере сайта?

Даже если страница имеет сертификат защиты и подтверждения о безопасности передаваемых данных, я все равно не имею понятия, что будет с моими данными, когда я кликну «Подтвердить». Но если в финале я окажусь на знакомой странице PSP, то буду спокоен за безопасность своих данных и за то, что они не попадут на сомнительный сайт. Я больше доверяю WorldPay, чем заурядному магазину виджетов.

Еще один плюс платежной страницы в том, что, если правила оплаты по карте изменяются, они будут обрабатываться на стороне провайдера PSP. Например, один из наших клиентов требовал 3D-Secure (на такой основаны технологии Verified by Visa и MasterCard SecureCode), чтобы принимать платежи Maestro. 3D защита требует от пользователей подтверждать платеж на странице своего банка, прежде чем он будет разрешен.

Если бы мы использовали API, нам нужно было бы изменить код, чтобы интегрировать 3D защиту. Но так как мы пользовались платежной страницей, мы просто уведомляли PSP, которая включала эту функцию для нашего аккаунта.

Все эти моменты подтолкнули владельцев многих сайтов к использованию платежных страниц. Они поняли, что нести ответственность за данные кредиток – только лишняя головная боль. Платежные страницы интегрируются со многими коробочными системами. После того как платеж сделан, пользователь перенаправляется обратно на ваш сайт, а специальный код позволяет идентифицировать пользователя и транзакцию и обрабатывать необходимые данные о совершенном заказе (например, поставить в

базе данных отметку «Оплачено» или обеспечить клиенту доступ к цифровой продукции).

Преимущество полной интеграции API в том, что вы можете контролировать процесс оплаты от начала до конца, в том числе оформлять страницы оплаты в своем стиле. Но вы также несете ответственность за безопасность данных пользовательских карт, и правила требуют, чтобы вы подтвердили, что используете передовые технологии.

PCI DSS

Стандарт безопасности данных индустрии платежных карт (The Payment Card Industry Data Security Standard (PCI DSS)) представляет собой совокупность 12 детализированных требований, которые распространяются на все компании, принимающие оплату по картам. Это относится не только к транзакциям онлайн. Обычный офлайновый магазин, который принимает платежи онлайн, должен тоже подчиняться требованиям PCI DSS для обоих способов оплаты: офлайн и онлайн.

Если вы принимаете онлайн-платежи через платежную страницу, но не получаете, обрабатываете или храните данные по картам, то вы можете заполнить сокращенный PCI DSS опросник (SAQ A), чтобы подтвердить, что ваш PSP совместим с PCI DSS. Если как средство интеграции вы используете API, то он полностью должен соответствовать PCI DSS (даже если вы не сохраняете данные карты) и включать ежеквартальные проверки безопасности, подтверждающих постоянное соблюдение требований.

В данном разделе мы не рассматриваем детали совместимости с PCI DSS, но если вы занимаетесь разработкой сайта, на котором платежи принимаются не через платежную страницу, тогда ознакомьтесь с ними или воспользуйтесь услугами тех, кто разбирается в данной теме.

Хранение данных карты

Я настоятельно советую вам не хранить данные карты на сервере клиента, даже в зашифрованном виде. Хранение данных требует совместимости с PCI DSS и поддержки сервера, а также Сети, в которых эти данные будут находиться в безопасности.

Если вам будет нужен доступ к ним, чтобы списать абонентскую плату, например вы можете поискать платежный шлюз, который предлагает сервис хранения данных. Если же вы собираетесь сохранять данные карты только для использования оплаты в один клик (как это делает Amazon), будьте осторожны! Вы в самом деле хотите взять на себя ответственность за данные вашего клиента? Хотите иметь дело с дополнительными и текущими расходами по поддержанию совместимости?

Валюта и местные налоги

Вполне возможно, что вам придется отчитываться по местным налогам, или НДС в Европе. Конечно, достаточно сложно разобраться в том, какие налоги нужны, но вы должны быть уверены, что ваша система обработает их правильно. Например, моя компания владеет скачиваемым продуктом, мини-CMS, которая называется Perch. Наша компания зарегистрирована в Великобритании, поэтому мы должны взимать налоги с британских покупателей. Еще нам нужно взимать НДС с покупателей из Евросоюза, пока у них есть действующий регистрационный номер плательщика НДС. Если покупатели из страны, которая не входит в состав Европейского Союза, мы не должны брать с них НДС. Итак, система должна уметь подтверждать номер налогоплательщика, а также правильно рассчитывать цены с или без НДС.

Большинство магазинов принимают оплату в одной валюте. Для того чтобы принимать платежи в разных валютах, чтобы посетители могли выбирать нужную им, смотреть соответствующие цены и вносить оплату, вам понадобится ввести нужную валюту в свой аккаунт продавца.

Рисунок 2.4. Платежная страница сайта. Включает НДС, скидку и приблизительную цену в долларах США. Несмотря на то что продается всего лишь один продукт, нужно принимать во внимание несколько величин

Другой вариант – это показывать цены в разных валютах, а оплату принимать только в местной. Вы можете обновлять курсы валют либо вручную, либо автоматизировать процесс с API. Если покупатели платят только местной валютой, то они должны быть в курсе, что стоимость показана чисто для информации и что реальная цена может слегка отличаться (из-за неустойчивых валютных курсов).

Обязательно посоветуйтесь с бухгалтером, когда будете иметь дело с деньгами, особенно, если вы принимаете платежи в иностранной или разной валюте. Если с самого начала знать, как обращаться с платежами и валютными курсами, то в будущем никаких проблем не возникнет.

О доставке

Если вы продаете физический товар, который нужно отправлять каким-то образом, вы должны включить расходы на транспортировку и, возможно, организовать отслеживание заказов. Так как вы продаете онлайн, то можно привлечь покупателей из других стран. Вам нужно решить, как рассчитать отправку в разные пункты и как вы будете доставлять товар: только в пределах вашей страны или в другие тоже.

Обычно онлайн-продавцы предлагают бесплатную доставку при заказе от определенной суммы. Также они практикуют доставку через разные службы доставки, такие как почта и курьерские службы (в зависимости от того, когда клиент хочет получить свой товар). Не забудьте учесть эти моменты, когда будете разрабатывать свой сайт.

Цифровые продукты

Покупатели ожидают, что они скачают цифровые продукты (такие как электронные книги, музыка и ПО) сразу же после их покупки. Доставка может быть оформлена в форме ссылки или страницы для

скачивания в своем профиле (вместе с лицензионным кодом если нужно).

Система должна обезопасить продукты до их приобретения и обеспечить в аккаунте клиента область, где они будут доступны (или как минимум отправлять ссылку на e-mail). Также можно генерировать код продукта. Опять же системы-посредники могут обеспечивать весь этот функционал в рамках оплачиваемых услуг.

Отчет и другие функции

Ваш клиент хочет обрабатывать заказы сразу же, как только они поступят, и вероятно, отмечать отправленные позиции, как только они будут обработаны. Возможность выгружать данные по заказам в CSV-файл, будет полезна как при организации e-mail-рассылки, так и при выгрузке платежных данных в офлайн-систему учёта продаж.

Вот ряд других вопросов, которые надо задать себе или заказчику:

- Нужно ли вам контролировать остатки через сайт?
- Как вы будете поступать с заказами, которые выполнены частично?
- Должен ли сайт генерировать счета и товарные накладные или это будет происходить в офлайне?

Интеграция с другими системами

Многие сайты не существуют автономно, а интегрируются с другими системами и сервисами. Интеграция реквизитов доступа (т. е. логина и пароля) от сети-посредника довольно стандартная процедура, и, возможно, вам будет нужно связываться с системой-посредником для контроля остатков и учета, особенно это касается магазинов электронной коммерции.

Несколько лет назад мы разрабатывали сайт для университета, который позволял студентам и персоналу обновлять свои данные и запрашивать определенные анкеты из офиса университета. По этой

причине нужно было обеспечивать логинам защиту. Для обычного сайта мы бы использовали свои собственные идентификационные классы с CMS. Но здесь нам пришлось работать с уже существующими логинами студентов. Эта информация сохранялась в LDAP (облегченный протокол службы каталогов), поэтому наш сайт должен был идентифицировать пользователей с помощью университетского сервера LDAP.

Так мы создали новый интерфейс для нашей стандартной CMS-системы, который идентифицировался через сервер LDAP. Когда работаешь с такой системой-посредником, написать код ничего не стоит.

Много времени ушло именно на то, чтобы получить доступ к серверу LDAP и понять, как подтверждается вход в систему. Мы использовали свою собственную CMS и создали идентификационный интерфейс, что упростило процесс написания кода.

В следующий раз мы разрабатывали обычную систему электронной коммерции для магазина одежды, торгующего онлайн. В нем продавались разнообразные дизайнерские модели в ограниченных количествах. Часто рубашки определенного размера и дизайна были представлены лишь в одном экземпляре. Из-за того, что рубашки продавались одновременно и в офлайн-магазинах, и онлайн, нужно было вести обновляемый учетный реестр. Если вещь продавалась онлайн, ее сразу же надо было убирать с полок обычных магазинов, а если она была куплена в обычном магазине, то информация на сайте должна была сейчас же обновляться.

Вообще, установка этой функции вызывает явные проблемы (за вещь, которую только что продали онлайн, кто-то может уже расплачиваться на кассе обычного магазина). Лучшее, что мы могли сделать, так это обеспечить одновременную двустороннюю связь между сайтом и системой электронно-кассовых продаж.

EPOS[17] была разработана другой компанией, и нам пришлось работать с ее разработчиками, чтобы скомпоновать две системы. Мы

ждали по три недели, пока эти разработчики соберут нужные ресурсы и выполнят наш запрос.

С этим мы ничего не могли поделать, и очевидно, что это время необходимо было включать в расписание проекта.

Ясность в нашем общении и представление четкой и полной информации, тестирование интерфейса, помогли свести к минимуму риск задержки. Но если вы знаете, что без посредников не обойтись, уточните, как они работают и сколько времени им нужно, чтобы обработать тот или иной запрос. Вы сможете тогда планировать свои действия с оглядкой на них.

Очень важно с самого начала узнать все о системах-посредниках, которые вам нужны. Не забывайте об этих важных задачах. Это упростит вам выбор технологии и коробочного ПО. Также это необходимо учитывать и при планировании сроков проекта как в своей компании, так и у партнеров.

Ограничения

Каждый проект имеет свои ограничения: во времени, в бюджете, в нехватке навыков и в несметном количестве других факторов. Прежде чем принять технические решения, вам нужно учесть их. Иначе эти ограничения встанут реальной преградой на вашем пути.

Бюджет

Когда будете планировать бюджет, учитывайте также и время. Наем разработчика или компании для создания сайта со стороны вносит ограничения в бюджет и временной график вашей работы над проектом.

Даже если посредник предлагает фиксированную сумму, она может колебаться в зависимости от того, сколько времени ему потребуется для работы. Разработка проекта своими силами также должна учитывать бюджет и время, будь то крайний срок его запуска или просто расчет времени на разработку.

Кроме чистых затрат на развитие, вам нужно будет учесть и затраты на посредников. Ведь мы все больше полагаемся на их помощь в обеспечении нашего сайта ресурсами. Хотите купить фотографии из фотобанка или шрифты? Затраты на хостинг могут увеличиться, если действующий провайдер не удовлетворяет техническим требованиям вашего нового сайта.

Если вы будете применять различные способы оплаты, то интернет-магазин понесет дополнительные затраты.

Программное обеспечение посредника может требовать оплату за лицензию, но это сократит время на разработку сайта. Вам, возможно, придется каждый месяц оплачивать услуги хост-системы. Так что включите в бюджет всевозможные затраты в начале проекта.

Выбор языка программирования

Возможно, ваша команда разработчиков или другое агентство по созданию сайтов имеют некий опыт в языке разработки. Он-то и определит базовый язык сайта, пока вам ужасно не захочется чего-то новенького. У хороших разработчиков всегда под рукой не один заранее заготовленный кусок кода и способы решения характерных задач на этом языке.

Но для того чтобы изменить язык на более «крутой», им потребуется серьезная переподготовка и много времени, потому что придется создавать новые библиотеки для базовых вещей. Это оправдывает себя в том случае, если существующий язык мешает продвижению сайта (или из-за того, что мало кто из разработчиков знает его, или из-за того, что его развитие застопорилось). Предположим, ваша система создана на классическом ASP, и команда, которая поддерживает ее годами, знает этот язык. Однако он уже вытеснен ASP.NET (и не развивается настолько активно), поэтому строить на нем новую систему не имеет смысла. Кроме всего прочего, найти разработчиков, которые знают Classic ASP не так-то просто. Да и язык устарел настолько, что он не подходит для модернизации сайта.

Если вам придется столкнуться с подобной ситуацией, вашей команде нужно будет освоить новый язык для переделки сайта. Если агентства со стороны предлагают вам преобразование в Classic ASP, спросите их напрямую, почему они проталкивают эту технологию. Объясните им, наконец, что вы хотите сайт, который будет жить долго, а устаревшие приемы вряд ли продлят его дни.

Если вы сами создаете сайт, и вас тянет изучить определенный язык, тогда придется утрясти вопросы с отсрочкой сдачи проекта насколько это возможно. Все в ваших руках.

Постигать что-то новое всегда интересно, но не кидайтесь от одного к другому, только потому, что это модно. Лучше знать один язык, но досконально, чем несколько, но поверхностно.

Если проект готовится для заказчика, скорее всего, он станет ограничивать вас во времени и бюджете. Ограничения могут иметь под собой как техническую подоплеку (например, требование создать и разместить сайт под определенную серверную архитектуру), так и политическую (например, требование использовать ПО с открытым кодом).

Технические ограничения

Если компания держит свои сайты на собственных серверах, скорее всего, у нее есть своя серверная архитектура. В этом случае нужно точно знать, что вы будете устанавливать.

Например, сервер работает на Windows, а компания могла бы установить PHP; это спасло бы вас от головной боли при инсталляции, потому что есть некоторые различия между PHP на Linux и PHP на Windows (в основном если сервер работает на IIS, а не Apache).

Если организация управляет собственными серверами, тогда выясните, будет ли у вас доступ к тестированию и развертыванию приложений. В некоторых проектах у меня его не было вообще, мне приходилось паковать код и данные и отправлять их IT специалистам организации для установки. Если это ваш случай, тогда скопируйте

ПО хостинга для тестирования у себя. Этим вы сэкономите себе время, потому что тестирование и изменение на рабочем сервере – процесс не из легких.

Интеграция с другой системой – своего рода ограничение. И определенные сторонние приложения могут быть исключены, если вы не можете написать плагины для их интеграции.

Политические ограничения

Вы можете столкнуться с ограничениями, которые продиктованы внутренней организационной политикой и предпочтениями клиента. К примеру, любое стороннее ПО должно быть с открытым кодом или использоваться должны только технологии Майкрософт.

Термин «открытый/исходный код» часто непонятен. Когда люди решают, что программное обеспечение должно быть открытым, обычно они не имеют в виду лицензию на него или бесплатное пользование. Все, что они хотят, это иметь возможность модифицировать (изменять) код, если нужно. Это позволит обезопасить себя в том случае, если разработчик стороннего продукта по каким-то причинам пропал или откажется от развития и поддержки продукта. Если от вас требуют использования ПО с открытым кодом, проясните для себя, что под этим подразумевается. Многие коммерческие продукты имеют незашифрованных и доступный для модифицирования код, но при то не имеют лицензии открытого исходного кода.

Иногда вы можете преодолеть политические ограничения, если приведете аргументы, почему какое-то решение лучше, чем то, которого от вас требуют.

Но они должны быть достаточно вескими. Ведь если клиент непоколебимо верит в какую-то технологию, то скептицизм с его стороны вам обеспечен.

Писать новый код или улучшать старый?

Решить, начинать ли преобразование сайта с полным обновлением серверного кода всегда непросто. Даже если существующая система имеет много проблем, может показаться, что вы выбросите на ветер кучу денег и приложите массу усилий, если будете заменять ее. В этой части мы разберем причины, по которым мы либо должны поддерживать уже существующую платформу, либо создать абсолютно новую.

Разработчик всегда подвержен искушению торжественно шагнуть вперед и создать что-то новое. Кому, в конце концов, понравится ковыряться в чужом коде? У всех нас свои методы работы; свои стандарты написания кода. Есть стандарты и системы, которые мы хорошо знаем и которым доверяем абсолютно. Но если мы выметим все подчистую из существующей системы и начнем все с нуля, мы рискуем потерять много полезного, что было в ней. И еще полбеда, если сайт предназначен просто для продвинутого управления контентом. А если вы решили переписать код сложной системы электронной коммерции, в которую в течение долгого времени вносили кучу различных доработок функционала?

Спросите себя, действительно ли существующая система не отвечает вашим требованиям настолько, что вы готовы полностью заменить ее, или же вам просто непривычно с ней работать.

Также не пренебрегайте опытом тех, кто пользуется системой. Если люди добавляют контент, обновляют продукцию или выполняют другие задачи в программе каждый день, то их нужно будет переобучать. Перенастройка и усовершенствование того, что есть, может быть даже поэтапное, избавит вас от этой необходимости.

Нельзя забывать и об ограничениях в сроках и бюджете. Начинать проект заново встанет заказчику в копеечку, да и времени на развитие будет затрачено больше. Внесение изменений в существующий код (рефакторинг) позволит вам распределить затраты, выкатывая обновления постепенно, по мере их готовности.

Но есть случаи, когда создание с нуля имеет смысл. Как мы уже говорили, программисты-новички будут рады заменить систему сайта

на ту, которую они знают отлично.

Если вы нанимаете собственного разработчика или команду либо начинаете работать с третьим лицом и хотите долго и плодотворно сотрудничать с ними, то тогда резонно перейти на платформу, в которой они асы.

Мы уже сказали вам, что язык программирования может устареть, или сайт был основан на более не поддерживаемом ПО, или функции, которые вы хотите добавить, перегружают существующую структуру. Не выбрасывайте деньги на ветер!

Еще одна причина, по которой надо бы избавиться от существующей платформы, – это то, что она может ограничивать вас в выборе дизайна или изменении ее структуры.

Например, сторонняя платформа систем электронной коммерции устанавливает шаблонные дизайн и процесс оформления покупки, и вы не можете внести изменения, которые повысят продажи. Трата времени на поиск проблем и решений, для того чтобы не допускать подобного, выбивает из рабочей колеи всех, кто этим занимается.

В конце концов, прежде чем принять решение установить новую платформу или перенастроить старую, досконально изучите существующую систему и посмотрите, как она уживается с настоящими и будущими требованиями.

Собственная разработка или коробочное ПО?

Предположим, вы захотите заменить сервер частично или полностью. Тогда подумайте, разработать ли вам персональное решение, или положиться на коробочное ПО, или, может быть, применить их комбинацию. В этом разделе мы оценим преимущества каждого варианта.

Зачем разрабатывать свое?

Если у вас специфичные и редкие требования, то продвижение собственной разработки может стать лучшим вариантом. Стороннее ПО должно привлекать широкий рынок, чтобы разработчики не тратили свое время зря. К тому же так заманчиво добавлять множество характеристик, чтобы платформой заинтересовалось больше народу. Вся проблема в том, что разработка системы может закончиться тем, что в ней будет сотни ненужных вам вещей.

И если движок не довести до ума, его функциональные свойства будут замедлять скорость вашего сайта, болтаться как бесполезные элементы и увеличат накладные расходы.

Если у вашего сайта большой трафик и вы с самого начала в курсе, что должны оптимизировать каждую строку серверного кода и внешнего интерфейса, то создание собственного решения позволит учесть весь список требований. Моя личная проблема в том, что я недостаточно использую язык SQL. Некоторые приложения делают сотни ненужных запросов базы данных, которые существенно повышают нагрузку на сайт.

Может случиться, что лицензия на стороннюю программу не подойдет к приложению, которое вы разрабатываете. Может быть, и так, что в системе есть все, что нужно, но какие-то 10 % функций, которые вы добавляете, требуют довольно глубокого проникновения в ядро программы, проще говоря, ее взлома. Вряд ли кто-то из разработчиков софта даст вам добро на это. Вот она и проблема! Сложность будет еще и в том, чтобы усовершенствовать программу.

Лучше делайте свои изменения через официальный API. Если не сможете, все равно не взламывайте программу. Потому что то, что у вас получится, вы едва ли сможете поддерживать.

Плюсы в разработке и развитии собственной системы в том, что вы точно сможете подогнать ее под свои требования. Стандартная программа может быть написана великолепно, и так же великолепно удовлетворять требования заказчика. Но при этом утечка конфиденциальной информации все равно возможна. А теперь ваша

задача – оценить, что лучше: использование готовых программ или создание своей, точно ориентированной на ваши задачи?

Выбираем коробочную платформу?

Коробочное ПО имеет явные преимущества. Вам не надо начинать с чистого листа. Вы получаете то, что уже находится в боевой готовности. Если вы ищете легкий способ управления контентом или ведения онлайн-магазина, а ваши требования достаточно просты, то наверняка одно из многих существующих в программе решений удовлетворит вас.

Есть бесплатные сторонние программы, а есть лицензионные, за которые приходится платить. У некоторых бесплатный только основной продукт, а дополнения – платные. Они добавляются быстро, поэтому заранее подумайте о том, что вам нужно для проекта. В одной системе может быть все включено в лицензионную оплату. Она, несомненно, обойдется вам дешевле, чем бесплатная альтернатива, для которой нужно будет приобрести несколько коммерческих дополнений.

Не забудьте узнать, на какую поддержку вы можете рассчитывать. У многих бесплатных программ официальная поддержка стоит денег или вовсе не существует. Еще не помешает зайти на форум, где пользователи отвечают друг другу на вопросы. Посмотрите, сколько людей участвует в нем и как быстро приходят ответы.

Рисунок 2.5. На WordPress оживленные форумы. Они для людей, которым нужно помочь в выборе правильного ПО

Если поддержка включена в стоимость, узнайте, в каком виде она предоставляется. Есть ли на нее ограничения или нет? Придется ли ждать? Существует ли общественная площадка поддержки и сразу ли персонал отвечает на вопросы? Twitter, кстати, тоже неплохой способ узнать о поддержке и общественном мнении о продукте. Просто спросите о нем и посмотрите, какие отзывы вы получите. Если

поддержка хорошая, платный продукт может быть более стоящим, чем бесплатный вариант.

Облачные решения

Третий способ решения, который нам нужно обсудить, – это программное обеспечение как услуга (SaaS), известные также как облачное ПО. Есть системы управления контентом (CMS), которые, к примеру, используются в электронной коммерции. В них реализован различный функционал, начиная от простого «положить в корзину» и страницы оплаты и заканчивая полноценным интернет-магазином. Такое решение привлекательно тем, что вам не нужно ничего устанавливать, чтобы запустить и поддерживать сайт. Может лишь возникнуть необходимость настроить некоторые опции и выбрать шаблон.

Использование внешнего размещения – быстрый способ запустить в работу ваш сайт. Вообще, в проектах электронной коммерции, такое решение перекладывает проблемы безопасности и сложных функций на компанию, которая специализируется на этом. Но есть кое-что, над чем нужно подумать.

Возможно, заказчик должен платить за пользование сервером ежемесячно и, естественно, это нужно внести в бюджет. Проверьте, изменится ли цена в зависимости от вашего трафика и количества заказов.

Вы также не можете изменять работу такого ПО. Если разработчик вносит изменения, которые влияют на ваш бизнес, придется с этим смириться. Вы не сможете переписать какую-то часть так, как вы это делаете с программой, находящейся на вашем сервере. Это и понятно. К ее исходному коду вы имеете доступ.

Ваш сайт будет зависеть от деятельности разработчиков облачного ПО, которые крутятся в бизнесе и не перестают предлагать услуги заказчикам по доступным для них ценам. Программа, установленная на вашем сервере, даже если провайдер отойдет от дел, все равно остается у вас. Правда, вы потеряете поддержку, но по крайней мере

сможете перейти на что-то свое абсолютно бесплатно. Если вы будете полагаться на внешний сервер, тщательно исследуйте его, чтобы убедиться, что он надежный, а компания – стабильная.

Бойтесь «евангелистов»!

Пользователи определенных продуктов порой проявляют свои странности. Вы это можете наблюдать на форуме, куда они заходят и наивно спрашивают: «А какая CMS лучше?»

По большому счету это бессмысленный вопрос. Он не прольет света на проблему, которую вы стараетесь решить. Читатель понятия не имеет, какой у вас сайт: крошечный с несколькими кусочками редактируемого текста или огромный трехязычный, вмещающий в себя тысячи документов.

И все равно кто-нибудь да расскажет вам о своей любимой CMS и в восторге станет доказывать, почему она лучшая из лучших. Этих людей мы называем «евангелистами»: они так любят софт, которым пользуются, что и представить себе не могут, что у кого-то может быть что-то другое. Я сам являюсь поставщиком CMS. Именно поэтому я иногда имею свою выгоду от «евангелистов» в среде наших пользователей. Но я и первый, кто в случае, если наше решение не вписывается в проект, скажет об этом прямо.

Бойтесь «евангелистов», когда вы изучаете стороннее ПО. Неважно, что это будет: целая платформа или простой скрипт. Задавайте специфические вопросы с учетом ваших требований. Тогда, скорее всего, в ответ вы получите более квалифицированное мнение.

Хостинг

Нас часто просят порекомендовать хороших хостинг-провайдеров и помочь с выбором хоста. Хостеров тысячи, но все они предлагают в конечном итоге серверы, которые могут мало отличаться друг от друга внешне, но по стоимости – в разы. Так что же выбрать? Может быть, просто хостера подешевле, у которого есть все, что вам нужно?

Дешевый хостинг – дорогой хостинг, или Скупой платит дважды

Многим хостинг-провайдерам удается сбывать свой продукт на рынке чисто из-за низкой стоимости. Можно достать хостинг за \$10 в год. Но то, что есть такой дешевый хостинг, еще не говорит о том, что надо пользоваться им. Он может в результате обойтись вам гораздо дороже, чем качественный хостинг с разумной, обоснованной ценой. Хороший хост стоит денег. А если заказчик толкает вас к дешевому провайдеру, то, надеюсь, что, прочитав этот раздел, вы вооружитесь неплохими контраргументами.

Веб-хост с оплатой \$10 в год, чтобы «сколотить» хоть какие-то деньги, должен разместить сотни сайтов на каждый физический сервер. Соответственно, это может привести к перегрузке, и какой-нибудь сайт с тяжелым трафиком застопорит остальные или даже выведет их в офлайн.

Google сейчас использует показатель скорости загрузки сайта для ранжирования выдачи[\[18\]](#), таким образом, медленный сайт не только раздражает пользователей, но и ухудшает позицию выдачи.

Рассчитывать на техническую поддержку за \$10 в год было бы глупо. Ни один провайдер не сможет этого сделать за такие деньги. Если ваш сайт стал хуже работать и проблема коснулась пользователей, знайте, что вы можете быстро связаться со специалистом в приоритетном порядке. Услуги клиентам стоят денег. Так что низкая цена хостинга окупается за счет платы за поддержку.

Как выбрать хост

Ваш сайт имеет определенные требования. Например, чтобы запустить сайт на WordPress, вам нужен следующий набор:

- PHP версии 5.2.4 или выше,
- MySQL версии 5.0.15 или выше,
- модуль `Apachemod_rewrite`.

Рисунок 2.6. Проверьте последнюю стабильную версию PHP на php.net

Если вы используете серверный язык и базу данных, убедитесь, что ваш хост поддерживает их, а также и минимальную версию, которую требует ПО. Правило номер один: держитесь подальше от хостов со старыми версиями серверных языков. Вы можете ознакомиться с современной версией PHP на сайте php.net, проверив раздел «Стабильный релиз». На момент написания последняя версия – 5.3.9. Если хост поддерживает обновления, то его версия – 5.3, естественно, версии продукта не должны быть моложе, чем 5.2.4, как требуется для WordPress.

Хост может поддерживать серверы со старыми версиями PHP, если нужна поддержка пользователей, работающих на них. Но он должен быть в состоянии также их обновлять их по требованию или создавать новые аккаунты (учетные записи) для новых серверов.

Базы данных

Если вам нужна база данных для сайта (например, контент блога WordPress хранится в базе данных MySQL), посмотрите, предоставляет ли хост хотя бы одну. Если у вас свои скрипты, то внедрить каждый с собственной базой данных, возможно, будет легче. Поэтому проверьте, какое количество баз допустимо. Что касается скриптового языка сервера, выясните, совместима или нет версия базы данных того софта, который предлагает хост, с вашими скриптами.

Вам нужна поддержка e-mail-сервера?

Если вы хотите использовать один и тот же сервер как для e-mail, так и хостинга, узнайте, что он предлагает. Есть ли у него нужный вам web-mail? Сможете ли вы использовать e-mail с протоколами POP или IMAP? Легко ли использовать интерфейс для установки почтовых сообщений? Есть ли антиспам-фильтр?

Нужно ли вам размещать несколько доменов на одном сайте?

Некоторые хостинговые компании разрешают регистрацию только одного домена на вашем сайте. Если вы собираетесь завести несколько доменов, развейдите, возможно ли это вообще.

Принимаем решение

Существует тысячи хостинговых компаний. Выбирайте, но с умом! Ведь если вы подберете ненадежного хостера, последствия могут быть весьма серьезными для той компании, чей сайт является «сердцем» ее бизнеса. Так как же выбрать надежный хост?

Хорошим признаком служат рекомендации. Они очень действенны, особенно если человек, который их дает, пользуется хостингом не один год. Когда вы сотрудничаете с разработчиками сайтов, возможно, что они посоветуют надежного провайдера, с которым работали, и скажут, от кого надо бежать как черт от ладана!

Стоит поискать хостеров на Google. Если у людей были с кем-то из них проблемы, они сообщат об этом на форумах или в блогах. Если увидите много плохих отзывов о какой-то компании, держитесь от нее подальше. Еще один неплохой способ получения информации – это Twitter. В сущности, люди любят «пощебетать» о своих проблемах в реальном времени, даже если они не склонны расписывать их на форумах и в блогах.

Как я уже говорил, достать очень дешевый хостинг – не проблема. Прекрасно, когда сайт ваш личный или создается для друзей. Но если же он крайне важен для вас и для бизнеса заказчика, тогда помните, за

что платите – то и получаете. Если хотите быстро справляться с проблемами, стоит заплатить немного больше.

Узнайте, как провайдер предоставляет поддержку. У некоторых хостов она осуществляется только через электронную почту, и это может быть для кого-то достаточным. Но ваш заказчик, возможно, захочет такого провайдера, которому он сможет позвонить, когда их почтовый сервер или сайт «упадет». Если вы знаете кого-нибудь, кто пользуется провайдером, который вы рассматриваете как вариант, узнайте, приходилось ли этим людям обращаться к нему за поддержкой и каков был опыт.

Как мы уже говорили, языки сервера нужно обновлять в целях безопасности и совместимости, как и настольные компьютеры. Если хостинг поддерживает очень старую версию языка, это говорит о том, что его серверы не современные. А это значит, что они уязвимы. Вы должны вникнуть в эти проблемы и попытаться установить скрипт с открытым кодом, который основан на современной версии языка программирования. Многие провайдеры дают гарантию на время работы, обычно в процентах: «Стопроцентная гарантия на время работы». Имеется в виду, что ваш сайт будет функционировать на 100 % в течение всего времени работы. Но не обольщайтесь и не рассчитывайте на многое!

Хостер обычно выплачивает компенсацию в случае, если сайт недоступен больше гарантированного времени, но вы сами должны контролировать такие моменты и предупреждать хостера перед тем как работа сайта будет восстановлена. Нет ничего приятного, если ваш сайт вдруг в 2 часа ночи перестал работать. Также гарантия доступного времени работы распространяется на сам сервер, но не на каналы связи.

Когда я выбираю хостинг, я мало обращаю внимания на заверения о времени доступности серверов по словам самого хостера, я спрашиваю пользователей этого хостера об этом и о работе техподдержки. Будьте осторожны! Не нарвитесь на хостинг компании, которая занимается перепродажей серверов крупных поставщиков. Любой может купить сервер у крупного хостера и начать распродавать

его дисковое пространство, при этом не имея малейшего представления о хостинге вообще. В этом случае, если ваш сайт упадет, вы столкнетесь с тем, что это посредник не в силах ничего с этим сделать. Он лишь может посоветовать вам обратиться напрямую к хостеру. С посредником, который болтался между вами и теми, кто действительно решит ваши проблемы, будет покончено.

Рисунок 2.7. Хостинг компания Memset предлагает ряд личных (приватных) виртуальных серверных пакетов

Типы хостинга

Много лет виртуальный (разделяемый) хостинг считался единственно прибыльным способом объединять почти все большие сайты. Сайт на виртуальном хостинге разделяет сервер с сотнями других. Вам будет выделено свое ограниченное пространство, поэтому вы не сможете конфигурировать или устанавливать софт на сервер.

Как альтернативу виртуальному можно привести выделенный сервер – физический компьютер, который вы получите целиком. С ним вам будет намного проще, потому что вы сможете изменять его установки, а также его «родной» софт. Но это удовольствие не из дешевых, именно поэтому многим сайтам не нужны пространство и ресурсы выделенного сервера.

В последние годы появился новый вид хостинга. Называется он «виртуальный выделенный сервер». В плане управления операционной системой он соответствует выделенному серверу, но по факту вы делите один физический сервер с другими виртуальными. Плюсы такого виртуального хостинга в том, что вы можете работать с какой угодно версией операционной системы и ПО, которое вам по душе.

А это значит, что вы можете изменять сервер и устанавливать на нем сложные сайты.

Многие компании, предлагающие этот вид хостинга, по умолчанию прилагают панель управления, чтобы облегчить людям работу на сервере, в том числе и тем, кто никогда не администировал такую систему раньше.

Две наиболее известные панели – это Plesk и cPanel (которые включают панель Web Host Manager). С помощью этих средств вы сможете установить новый сайт на сервере и актуализировать обновления, а также компоновать сервисы, доступные для индивидуальных сайтов. В сущности, имея виртуальный выделенный сервис, вы сами себе хозяева.

Виртуальные выделенные серверы также удобны для настройки показа сайта заказчиком, возможно на субдомене. Для дизайнеров, чье кредо – не передавать файлы до тех пор, пока заказчик не оплатит их, такая настройка позволит предоставить на утверждение сайт полностью, прежде чем отправить его клиенту.

Виртуальный выделенный сервер дает компаниям максимальную гибкость. Но если у вас неспецифические требования, то для их выполнения, скорее всего, подойдет стандартный общий пакет хостинговых услуг. Он имеет соответствующие спецификации и надежную поддержку, а также постоянно обновляется.

Облачный хостинг отличается от виртуального и виртуально-выделенного. В нем ресурсы, необходимые для вашего сайта, могут быть разбросаны по другим серверам, и вы вольны в том, чтобы добавлять их или, если нужно, убирать. Этот вид хостинга подходит для приложений, которым иногда требуются мощные ресурсы, а иногда – низкие.

Хороший тому пример – продажа билетов. Наплыв пользователей на сайт наблюдается, когда в продажу выбрасывают билеты на какое-то популярное событие, но все остальное время трафик остается низким. С облачным хостингом у вас будет, когда нужно, доступ к дополнительным мощностям (конечно же, не бесплатно), а резервная мощность сохраняется круглый год.

Среда разработки

Данный раздел больше подойдет не тем разработчикам, которые работают в большой команде, а для дизайнеров, которые впервые взялись за полную переделку сайта. Мы поделимся своим опытом, а затем займемся непосредственно переделкой.

Неважно, переделываете ли вы работающий в данный момент сайт или просто дополняете его новыми элементами или интерфейсом, это не должно навредить пользователям. Даже если изменения ничтожны, никогда не обновляйте сайт, пока не протестируете изменения.

Один из методов, у которого много поклонников, это вести разработку, добавляя подпапки на существующем сайте. Не самая хорошая идея, однако! Когда вы будете готовы к запуску сайта и перенесете новые файлы выше, в корневой раздел сайта, относительные ссылки станут неверными. Все изображения могут «слететь» из-за этого. Придется вам тогда все восстанавливать. Согласитесь, не лучшее начало проекта!

Для разработки лучше настроить локальное оборудование, включая серверные скрипты и правильные пути к корню. Затем загрузите его на вспомогательный сервер, где вы будете проверять сайт на вебе, показывать его заказчику, наполнять контент и получать добро на дальнейшее существование.

Если вы дорабатываете «живущий» сайт или перекраиваете интерфейс пользователя, а код оставляете более или менее нетронутым, мой вам совет: экспортируйте базу данных и скачайте все файлы (об этом прочтете позже в статье «Перенос вашего сайта»). Это нужно для того чтобы настроить разрабатываемую и промежуточную версии сайта в точности, как на действующем сайте. Если база данных с рабочего сайта содержит конфиденциальную информацию о пользователях, лучше использовать псевдозаполнение данных. Не только для того чтобы защитить данные, но и во избежание их случайной отправки 10 000 пользователям из вашей локальной системы, когда будете что-то проверять!

Вспомогательный сервер

Локальный сервер устанавливается на ваш собственный компьютер или какой-то другой у вас в офисе, и вы используете его для разработки сайта. Когда будете готовы показать заказчику свою работу, вам придется переместить его куда-то, где к нему будет доступ с других сетей. «Куда-то» означает перемещение на вспомогательный сервер.

Он так же прост, как и поддомен, который указывает папку на вашем текущем сервере. Это позволит вам протестировать сайт на реальном сервере с заданной конфигурацией и всеми корректными путями.

Если вы создаете большое количество сайтов для заказчиков и собираетесь использовать те же настройки сервера (тот же язык, базы данных и проч.), вы можете взять дешевый виртуальный выделенный сервер и задать для каждого клиентского сайта свой субдомен для тестирования.

В этом случае желательно обезопасить директорию. Чтобы зайти на сайт, посетители должны будут ввести пароль. Это помешает случайным людям увидеть вашу «сырую» разработку и спасет от индексирования этого сайта Google.

При разработке сайтов со сложной структурой и разным контентом для заказчиков, мы обеспечиваем им доступ к его промежуточной версии, где они смогут добавить свой контент и тщательно изучить сайт. Потом мы перемещаем сайт, базу данных и прочие необходимые объекты на текущий сервер, что обеспечивает быструю и легкую миграцию новой версии сайта без ошибок.

Контроль версий

Мы не можем говорить о среде разработки, не упомянув о контроле версий. Даже если вы работаете в одиночку, вы должны его использовать. И это всегда полезно для тех, кто работает в команде. Ее члены смогут править файлы и не волноваться о том, что они

«затирают» файлы друг друга. Вы всегда сможете вернуться к прежней версии, если что-то пойдет не так.

Если вы решили доработать сайт, а не переделать его полностью, ваш первый шаг – это импортирование всех файлов в одну из систем контроля версий. Так вы всегда сможете откатиться назад, если какие-то изменения чисто случайно затронут часть системы.

Даже когда вы работаете один, управление версиями защищает вас от самого себя. Например, от случайного удаления файла или быстрого изменения приложения. Я использую управление версиями, чтобы переносить работу с компьютера на компьютер, если это нужно. В конце каждого дня я фиксирую свою работу. Если моя дочь не пошла в школу и я работаю на дому, я могу взять текущие файлы и начать точно с того места, на котором я остановился позавчера вечером в офисе.

Рисунок 2.8. Beanstalk сервис для контроля версий

Существует несколько систем управления версиями. Вы, наверное, слышали о таких, как Subversion и Git. Обычно каждый выбирает, кому что нравится. Так как для выполнения работы вы можете менять сервер, то используя такие веб-сервисы, как GitHub или Beanstalk, вы можете быстро приступать к работе. Beanstalk имеет отличное руководство по управлению версиями на своем сайте[\[19\]](#).

Перенос вашего сайта

Вопрос, как переместить сайт с одного сервера на другой, мучает многих, особенно если сайт пользуется популярностью. Но если вы будете следовать рекомендациям ниже, то увидите, что сделать это возможно и без явного ущерба для посетителей. Этот процесс аналогичен переносу сайта с промежуточного сервера на основной.

Инструкции, данные ниже, предназначены для развертывания сайта на оборудовании обычного виртуального хостинга, с SFTP доступом.

Если ваше оборудование сложнее, чем это, тогда у вас, вероятно, уже есть специалист, который поможет вам разобраться.

Установка нового оборудования хостинга

Для начала вы должны настроить хостинг. Проверьте, есть ли в нем все необходимые возможности, включая поддержку нужного вам серверного языка и базы данных.

Перемещение файлов на новый сервер

Для загрузки файлов сайта на новый сервер используйте SFTP. Новый сервер может предоставлять вам временный домен для тестирования сайта. Если нет, то я обычно временно создаю на сервере поддомен, пока я проверяю, все ли работает как надо.

Избегайте предварительного просмотра сайта с тильдой (~) или с вашим именем пользователя, иначе пути и ссылки побьются.

Перенос базы данных на новый сервер

Если вы работаете на приложениях не от Microsoft, то, скорее всего, вы используете MySQL. MySQL – это база данных, используемая для большинства приложений с открытыми «исходниками». Ее любят разработчики, которые работают с PHP, Python и Ruby on Rails.

Первое, что вы должны знать при работе с такой базой данных, как MySQL, это то, что в ней нет файла данных для скачивания и пересылки. База хранится внутри MySQL сервера. Чтобы получить данные, вам надо будет подключиться к нему. Многие серверы устанавливаются с PHPMyAdmin – веб-приложением, которое позволяет управлять базами данных через браузер.

Вы также сами можете загрузить и установить PHPMyAdmin. С ним вам будет удобно переносить базу данных MySQL между локальным, вспомогательным и рабочим серверами.

Настройка почтовых ящиков

Если ваша почта будет располагаться на новом домене, тогда установите почтовые учетные записи так, чтобы они были готовы к использованию, как только домен будет прикреплен к новому серверу.

Изменение сервера доменных имен, или переназначение записи домена

Если вы изменяете DNS домена на ваш новый сервер, то нужно изменить и серверы доменных имен. Если ваш DNS расположен где-то в другом месте, а вам нужно изменить запись так, чтобы домен указывал на новый сервер, то измените лишь запись A. На это уйдет какое-то время. Пока новая запись между доменом и сервером не будет прописана по всему Интернету, продолжайте проверять почту со старого сервера день-другой, чтобы быть уверенным, что вы не потеряли ничего важного. Если прежний хост позволяет пользователям проверять почту в веб-интерфейсе без захода на домен, делайте так, пока почта не перестанет проходить через него.

Перенос сайта, работающего на базе данных

Для перемещения сайта, работающего на базе данных, нужно время: во-первых, чтобы запись DNS обновилась по всему Интернету, и во-вторых, для обновления кэширующих серверов. В течение этого промежутка времени, одни посетители могут направляться на старый хост, а другие – на новый. Если ваша база данных предназначена только для внутреннего управления контентом, и посетители не могут добавлять его, тогда просто попросите контент-менеджеров не изменять информацию на сайте до тех пор, пока они не будут уверены, что теперь они работают с сайтом на новом сервере.

В этом случае я проверяю сайт на копии базы данных и затем перемещаю базу непосредственно перед сменой доменных имен. Мне надо точно знать, что я перенес большую часть обновленного контента. Если посетители добавляют данные в базу (например, они делают заказы в онлайн-магазине), тогда вам ни в коем случае нельзя терять их при перемещении. Самый надежный способ – временно выключить сайт и разместить на нем страницу, с уведомлением, что идут технические работы.

Если так не получается, есть пара обходных маневров. Во-первых, сделайте перенос, как описано выше, а когда убедитесь, что DNS полностью переключился, сравните две базы данных и синхронизируйте записи от старых к новым. Если у сайта очень низкий трафик, а вы ожидаете только пару заказов или комментариев за это время, это сработает. Как вариант, вы можете сначала переместить базу данных. Пока будет настраиваться новый хостинг, чтобы внешний сайт мог соединиться с базой данных, вы можете переместить ее, а затем присоединить и старый, и новый сайты к базе данных в новых настройках сервера.

Если это ваш случай, то попросите вашего хостинг-провайдера (и разработчика, если он у вас есть) сделать все лучшим образом.

Куда дальше?

Мы пробежались по техническим вопросам, относительно переделки сайта или перехода на новую ступень развития. Не все в этом разделе пригодится именно вам, но все равно, обдумывайте свои технические решения, прежде чем принимать их. Даже если вы не один воплощаете их в реальность, избегайте спешки и не идите неверным, а значит и дорогим, путем!

Вспомните слова Пола Боуга из первого раздела: прежде чем «нырнуть с головой» в какое-то решение, сделайте домашнюю работу! Есть много способов достичь всего того, что вы хотите. А CMS или структура, которую вам пытаются продать, не всегда лучшее из того, что нужно вашему проекту. Всегда обращайтесь к своим исследованиям, бизнес-требованиям, имеющимся ресурсам и ограничениям. Тогда мудрое решение придет само собой! Все, что тщательно продумано, позволит сайту расти и развиваться.

Если вы полностью переделываете сайт, представьте, что вы это делаете в последний раз, и выбирайте такие решения, которые продвинут вперед ваш сайт и бизнес.

Об авторе

* * *

Рейчел Эндрю – веб-разработчик с навыками как в серверных языках, так и в верстке и разработке интерфейсов. Она написала множество книг, в том числе «Антология CSS: 101 основной совет, хитрость и уловка» (The CSS Anthology: 101 Essential Tips, Tricks & Hacks), четвертое издание которого было опубликовано в марте 2012 года. Рэйчел – основатель и управляющий директор edgeofmyseat.com (компания, занятая разработкой программного обеспечения), участвовала в создании «Пирч» (Perch), «действительно маленькой системы управления контентом». У Рэйчел есть личный сайт rachelandrew.co.uk, где она пишет о многих вещах, в том числе вебе и ведении бизнеса. Еще ее можно найти в Twitter под ником [@rachelandrew](https://twitter.com/rachelandrew).

О рецензентах

* * *

Харли Финкельштейн (р. 1983) – канадский гражданин, родившийся в Монреале, там же он получил ученую степень в области экономики. Также у него есть степень магистра в области права Университета Оттавы, города, где он живет сейчас. Он описывает Оттаву как холодный город с атмосферой провинции и радостью столицы. Помимо Интернета, он любит горные лыжи, бокс, готовить суши, диджеинг и красный цвет. Харли основал свою первую компанию в возрасте 17 и с тех пор работает в этой сфере. Сегодня он занимает должность руководителя платформы на Shopif, ведущей платформы электронной коммерции. На протяжении своей карьеры, Харли научился следовать за своими увлечениями и практиковать их каждый день. Его личный совет читателям: «Действуй энергично и находи изящные решения».

* * *

Райан Карсон – американец, живущий в Великобритании, отец семейства. Окончив Университет штата Колорадо со степенью в области компьютерных наук, он переехал в Великобританию в 2000 году. Он любит веб-технологии, кофе и фильмы. (Смотреть «Матрицу» в кино семь раз – это неправильно?) Он любит вовлекать и вдохновлять людей, и именно поэтому он увлеченно работает над мероприятиями для веб-сообщества. Он успешно построил и продал две компании и в настоящее время работает над третьей, Treehouse.

Погружение в HTML5

Автор: Бен Шварц

Рецензент: Рас Уикли

Если бы вы попросили меня рассказать вам о HTML5, я, пожалуй, начал бы с того, что сейчас ваша роль как веб-разработчика сильно изменилась по сравнению с тем, что было раньше.

Я буду обращаться к вам как к эксперту в области HTML (языка разметки), CSS (и всех ее вариаций в разных браузерах), JavaScript (и едва уловимой разницы между его воспроизведением в браузерах). Потом я перейду к теории дизайна, анимации, 3D, серверным технологиям и обработке звука.

Вы наверняка захотите узнать, зачем так много технологий собрано «под крышу» HTML, и, очевидно, удивитесь, почему вы решили взяться в первую очередь за веб.

HTML5 (как спецификация) «разбит» на мелкие кусочки, разбросанные по разным специализированным областям. Постарайтесь не раздражаться из-за этого.

Получив твердые базовые знания HTML, CSS и JavaScript вы сможете продолжить совершенствоваться самостоятельно и развить такие спецнавыки, которые другим и не снились. В сущности, когда вы овладеете не только узкоспециализированными, но и непрофильными навыками, вы станете просто бесценным и незаменимым для компании и коллег.

Для большей части этих новомодных фишек так называемый «передовой опыт» еще не накопили. Поэтому если вы хотите научиться чему-то действительно классному (и, возможно, прославиться в процессе), тогда самое время загрузить бета-версию браузера и начать эксперимент. Для этого, собственно, многие поставщики браузеров выпускают их бета-версии.

Ниже представлены, в частности, бета-версии так называемой «Большой пятерки» браузеров:

- У Google Chrome три версии релизов: «Beta» (для разработчиков) «Dev channel» (для разработчиков, которые хотят использовать возможности, созданные в течение последней недели) и «Canary» (так называемый ночной выпуск, полностью неопробованный). Вы можете скачать их по адресу smashed.by/chromedev.
- У браузера Safari одна версия: Webkit (webkit.org).
- У Opera – версия «Next»: smashed.by/operadev.
- Firefox имеет «ночную» версию (smashed.by/ffndev) и пре-бета сборку
- Aurora (smashed.by/ffadev).
- Последний по списку, но не последний по значимости: версия Internet Explorer от Microsoft выходит в ручном режиме (и поэтому не является «ночной» сборкой): smashed.by/iedev.

Поддержка браузерами новых свойств выпускается в модульном порядке. А с производителями браузеров (такими как Google и Mozilla), выпускающими новую версию каждые 6–8 месяцев, номер версии теперь несет значительно меньше смысла, чем ранее.

Кто-то может провести параллель с тем, как разработчики вносят изменения на сайт. Да, у сайта есть версия, но она не важна для конечного пользователя. Поэтому, как разработчик, определитесь, с какими свойствами вам лучше написать свою историю.

Так как веб-технологии постоянно развиваются, мы должны иметь представление об их прошлом. К счастью, эту цель преследуют оба комитета по стандартам языка HTML. Так что расслабьтесь и дышите ровно, HTML5 не отдалит от вас пользователей и не обременит работой. Какой бы DOCTYPE вы ни использовали, браузер пользователя отрендерит сайт в самом лучшем виде, насколько это

возможно. Если вы используете новый элемент HTML5 вместе со старым DOCTYPE, все равно он будет отрендерен верно.

Здесь мы не будем говорить о WebGL, аудио и видео, устройстве API, Web sockets или SVG. На каждую тему уйдет по разделу, поэтому оставляю их вам на откуп. Вместо этого я расскажу вам все с самого начала. Мы разберем все важные базовые моменты, прежде чем двигаться дальше, к более сложным темам.

Мы откуда и куда?

HTML5 – это уйма всего. Мы уже далеко ушли от последней основной «версии» HTML. Сообщество по развитию гипертекстовых технологий (WHATWG) ссылается на нее как на «HTML: Живой стандарт» (от упоминания числа 5 отказались). А это значит, что у HTML нет версий. Как мы говорили, производители браузеров выбирают лучшие свойства для рендера, поэтому и отличается поддержка разных элементов разными браузерами.

WHATWG, W3C и «Компания»

Может быть, вы слышали о Всемирном консорциуме Сети (W3C). Мы уже упомянули о сообществе WHATWG. Оно было основано производителями браузеров Apple, Mozilla и Opera. Обеспокоенные тем, что W3C недостаточно занимается развитием HTML, они решили организовать свою группу.

Большая часть работы группы WHATWG пересекается с W3C, а в лицензии на спецификацию говорится, что «вам предоставляется лицензия на использование, воспроизведение и создание производных этого документа».

W3C в самом деле помогает в работе. Организация не разрабатывает стандарты, но охотно дает рекомендации. И хотя W3C финансируют большие компании – производители компьютеров и браузеров, она тем не менее специализируется на открытых стандартах, которые не ставят во главе угла какую-либо одну компанию.

Что же, как разработчик, вы можете быть уверены, что все новые разработки в HTML (особенно в области веб-приложений) создаются при значительной финансовой поддержке производителей браузеров (Webkit, Gecko и Opera) и со временем одобряются W3C.

Эти «странные» взаимоотношения привели к тому, что технология с лицензионной платой или являющаяся строгой собственностью у

многих не пользуется популярностью. Браузеры соревнуются между собой сейчас столь же напряженно, как в самом начале.

Рисунок 3.1. Сайт Caniuse ([smashed.by/ciu](https://caniuse.com/)) демонстрирует, в каких случаях вы можете или должны использовать возможности HTML5

Знайте, какими функциями пользоваться

Современные разработчики должны понимать аудиторию, которую они обслуживают, уметь выбирать правильную технологию для работы и знать, каковы будут последствия, если какой-то функционал не поддерживается браузерами их аудитории.

Только маги и чародеи могут знать, насколько широко поддерживается какая-либо заданная функция. Если вы не относитесь к их числу, возьмите себе в помощь сайт When Can I Use [\[20\]](#). На нем находится список возможностей, которые поддерживаются в современных версиях всеми основными настольными и мобильными браузерами, а также перечень возможностей, которые будут представлены в будущих версиях. Он доступен для поиска и даже подключается к Google Analytics, чтобы показать вам, какие браузеры использует ваша аудитория. А теперь давайте погрузимся и посмотрим повнимательнее на основы HTML.

DOCTYPE описание типа документа (DOCTYPE)

[\[21\]](#)

Поройтесь в кладовой своей памяти. Можете припомнить полный DOCTYPE для HTML 4.01 (или для XHTML, коли на то пошло)? Не думаю. Давайте я покажу вам DOCTYPE для HTML5:

```
<!doctype html>
```

Вот и все! Можно печатать и заглавными, и строчными буквами. Этого вам достаточно, чтобы перевести браузер в режим соблюдения

стандартов. Спрашивается, зачем нам нужно все время копировать и вставлять верхние строки документа HTML? Конечно же, мы захламляли наш HTML кучей других важных тегов на протяжении нескольких лет. Давайте посмотрим, что еще упростилось.

Мета-теги

```
<meta http-equiv="Content-Type" content="text/html;  
charset=utf-8">
```

Ох, что за бардак! Этот метатег очень важный и его надо добавлять до тега title, чтобы убедиться, что браузер правильно установил кодировку символов. К счастью, его сделали легко запоминаемым:

```
<meta charset="utf-8">
```

У некоторых парсеров XML есть проблемы с тегами, которые не являются самозакрывающимися. Вот поэтому разработчики выбирают самозакрывающиеся теги (т. е. XHTML стиль). Конечно, вам решать, но мы все же советуем оставлять теги открытыми.

Теги со ссылкой на таблицу стилей и скрипты

Атрибут type можно задать и через ссылку на таблицу стилей `<link rel="stylesheet" href="layout.css">`, и в теге script.

В прежние времена атрибут type мог использоваться в теге script, если вы хотели применить VBScript вместо JavaScript, но сегодня это уже не актуально.

Вообще, когда ты опускаешь детали, которые «раздувают» и усложняют код, то чувствуешь себя просто великолепно. Но мы прошлись только по верхам. Давайте немного доработаем тег script.

Асинхронная загрузка шрифтов

Сначала о том, как браузер загружает файлы. После того как браузер загрузил и проанализировал HTML, он составляет список project

assets, категорий[22] (изображения, CSS, JavaScripts и т. п.) и расставляет приоритетность их загрузки в порядке появления.

Раньше мы подключались к Интернету через телефонную сеть, которая слабо справлялась с несколькими одновременными подключениями. Сейчас пропускные способности значительно варьируются (особенно с учетом мобильных устройств). Поэтому сегодня браузеры ограничены в одновременной загрузке файлов с домена верхнего уровня. Вот почему некоторые разработчики используют сети доставки и дистрибуции контента или размещают файлы на поддомене (таком как `assets.example.com`); использование нескольких доменов дает разработчику больше слотов загрузки для скриптов, таблиц стилей, изображений и фреймов.

Рисунок 3.2. Уерпоре: условный скриптовой загрузчик

Когда браузеры качают файлы JavaScript, они загружают по одному скрипту за раз, что позволяет им анализировать код и предзапускать магические оптимизации. Теперь, вместо того чтобы полагаться на удачу, мы можем использовать загрузчики скриптов (такие как LABjs, Уерпоре, RequireJS и многие другие), чтобы одновременно загружать множество скриптов, устанавливать зависимости и определять, нужен ли вообще какой-то определенный скрипт.

Имеет смысл улучшить производительность страниц везде, где возможно. Компания Amazon утверждает, что увеличение времени загрузки страницы на 100 миллисекунд уменьшает количество продаж на 1 %[23].

Вооружившись этими сведениями, поговорим о моем любимом скриптовом загрузчике, Уерпоре[24].

Уерпоре может использоваться для условно загружаемых скриптов на основе тестов. Попросту говоря, вы запрашиваете JavaScript только в том случае, если он нужен браузеру.

Для примера:

```
уерноре([  
  
{  
  
test: window.JSON,  
  
норе: '/javascripts/json2.js'  
  
}  
  
])
```

Этот умный кусочек JavaScript проверяет, есть ли у браузера свой «родной» анализатор (парсер) JSON. И туда, где его нет (у браузеров IE 6 и 7), он загружает скрипт `/javascripts/json2.js`, (полифилл [\[25\]](#) JSON).

Теперь, когда мы рассмотрели азы загрузок скрипта и поговорили о параллельной загрузке, самое время взглянуть на два новых атрибута в теге скрипта.

Первый – это асинхронный тег:

```
<scriptsrc="/javascripts/application.js" async></script>
```

Асинхронный тег – это булевый атрибут, означающий, что его явное присутствие в браузере задает значение `true`, или «Да, пожалуйста, используйте этот функционал». Он велит браузеру выполнить `application.js`, как только оно будет доступно. Скрипты, которые загружаются асинхронно, выполняются, как только будут загружены, то есть не в порядке их появления в коде HTML

Передаем файлы быстрее

Стоит упомянуть, что нашим самым большим достижением стало уменьшение размеров скриптов в целом. Сначала мы добились этого, применив `gzip`-сжатие (как и для таблицы стилей, и HTML файлов).

Чтобы добавить на сайт поддержку gzip, зайдите в репозиторий HTML5 Boilerplate на GitHub[\[26\]](#).

Если вы не знаете, как обслуживается ваш сайт, самое время ознакомиться с панелью инструментов для разработчика. На Webkit браузерах (Safari и Chrome) вы можете открыть ее, нажав Command + Option + I в системе Mac и Control + Shift + I в Windows.

Под панелью «Сеть» вы найдете список файлов, которые были загружены для текущей страницы. Можете проинспектировать запросы и заголовки ответов для каждого файла.

Рисунок 3.3. Панель инструментов для разработчика Safari отображает активность сети на Yerpore.

Рисунок 3.4. Файл был передан браузеру в предварительно сжатом виде

Второй совет, как улучшить производительность вашего сайта (вообще-то, лучший способ – это оптимизировать код), – объединить и сжать файлы.

Я советую вам применить компилятор UglifyJS[\[27\]](#).

Пытливые люди вроде Стива Саудерза посвящают себя изучению того, как браузеры загружают, анализируют и отображают сайты. Если вам интересно, как повысить производительность сайтов, почитайте работы Стива.

Новые семантические теги и их использование

Так как мы говорим о версии HTML5, по сути, все эти разговоры о производительности и скриптах к нам вроде бы не относятся. Давайте рассмотрим новые семантические теги и разработаем стратегию их применения.

Прежде чем использовать какой-то новый тег, не забудьте о так называемом HTML5 Shiv. Этот скрипт важен для нас. Без него Internet Explorer 6, 7 и 8 не смогут распознать незнакомые стили (т. е. новые HTML5-теги, которых не было, когда создавались указанные версии IE).

Вы можете скопировать HTML5 Shiv из сервиса Google Code, где выложен этот проект[\[28\]](#).

Также вы можете получить HTML5 Shiv, используя js-библиотеку Modernizr[\[29\]](#). В этом разделе мы не говорим об этом скрипте подробно, но тем не менее держите его на вооружении. Я использовал его на каждом сайте, созданном за последние два года.

Сброс HTML-стилей по умолчанию

Каждый браузер отображает элементы без присвоенных стилей немного по-разному. Поэтому, чтобы нормализовать теги для лучшей кросс-браузерной разработки и поддержки, вы можете использовать сброс стилей.

Применяйте одну из новых версий CSS reset, поскольку старые версии не задают нормированный стиль элементам HTML5.

Очень советую познакомиться с Normalize.css[\[30\]](#) Николаса Галлахера[\[31\]](#) и Джонатана Нила[\[32\]](#).

Многие из старых сценариев сброса (классический пример – сброс от Эрика Мейера) тяжеловесны: они сбрасывают каждый элемент, а некоторые их изменения можно оспорить, например тег strong при установке не выделяется жирным шрифтом по умолчанию. Normalize.css сбрасывает элементы более изящно и сглаживает некоторые причуды браузеров. Это позволит вам максимально нивелировать различия между ними. Джон Нил и Николас Галлахер «раскладывают по полочкам» все, что делают скрипты. Изучите подробно прокомментированный код – он фантастичен!

Строим сайт заново

Дойдя до этого пункта, вы, наверное, подумали: «ОК, пора начинать писать сайт, пользуясь самыми замечательными новейшими тегами!» Когда были установлены новые семантические теги, разработчикам пришлось потрудиться: проанализировать, какие классы идентификаторы они будут применять для своих сайтов. Ничего экстраординарного они не обнаружили: на деле все они пользовались одними и теми же именованиями (иногда с небольшими перестановками).

Итак, возможно, имена новых тегов уже подходят для того, что мы делаем.

Тег SECTION

Тег section может использоваться для того, чтобы разбить главную страницу. Допустим, ваш блог состоит из персональной информации о вас, ваших презентаций и регулярно размещаемых постов. Попробуйте разбить это на разделы:

```
<section class="статьи"></section>
<section class="о себе"></section>
<section class="презентации"></section>
```

Возможно, каждая из этих частей носит свою «шапку», поэтому мы можем считать их довольно важными. Так что наше решение по поводу разбивки на разделы оправдано. Если бы вы писали книгу наподобие этой, вы могли бы для каждой из глав создать раздел, а затем внутри каждого раздела – подраздел. Что касается семантики на сайте, общий совет: не «зависайте» на деталях. Выберите элемент, основанный на самой полной информации, которая у вас есть сейчас – и вперед! Семантика субъективна. И не парьтесь по пустякам!

Тег ARTICLE

Когда вы погружаетесь в HTML5, вам хочется знать, а в чем же разница между разделом и статьей. Возможно, вы уже предположили,

что тег `article` используется в основном в блогах и онлайн-новостях. Вы недалеки от истины.

Если вам нужен простой критерий распознавания статьи, используйте такой практический прием: если часть контента все еще имеет смысл вне текущего контекста (т. е. если пользователь не видит элементы страницы с этим контентом), тогда это, скорее всего, статья. Отсюда – блоги и онлайн-новости.

Я использую тег `article` для набора контента, скажем, для списка презентаций, которые я представлял раньше, возможно, с кратким обзором:

```
<section class="presentations">
<header>
<h1>Мои презентации</h1>
</header>
<article>
<h2>Введение в HTML5</h2>
<p>4-часовой мастер-класс, который я проводил
в Австралии</p>
</article>
<article>
<h2>Compass и SASS</h2>
<p>Используйте качественную библиотеку CSS,
и вы сможете сконцентрироваться на важных вещах.
</p>
</article>
</section>
```

Зоркие читатели могут спросить «Похоже на список! Почему бы не использовать тег `ul`?» И будут правы. Он, безусловно, может быть элементом списка. Но `article` показывает, что, хотя эти элементы похожи, они не имеют отношения друг к другу. Мы можем спорить об этом до бесконечности. Но, в конце концов, на вкус и цвет товарищей нет, и решение за вами.

Тег HEADER

Вы когда-нибудь использовали класс или идентификатор как «шапку» или баннер? Или даже для заголовка сайта? Тег `header` может использоваться как нечто большее, чем просто заголовок сайта. Его можно (но совсем не обязательно) применять внутри тегов `article` или `section`. Просто используйте его, когда нужно, чтобы блочный элемент разграничивал пространство на странице для ясности. Например, я часто храню заголовки и метаинформацию в шапке поста блога.

Тег FOOTER

Тег `footer` аналогичен тегу `header`. Вы можете использовать его внутри тегов `article` или `section`, или глобально, внутри `body`.

Тег ASIDE

Тег `aside` может использоваться на уровне страницы или внутри `article`. Его содержимое можно считать полезной информацией, но вовсе не основной.

Например, для мобильных версий своего сайта вы можете скрыть `aside`-элементы. Как бы вы ни обошлись с тегом, он заставит вас принять решение о контенте. Запись в блоге можно представить так:

```
<article>
<header>
<h1>Все о тракторах</h1>
<time      datetime="2012-01-01">1          января
2012</datetime>
</header>
<aside>
<p>Написано полностью Брюсом Лосоном</p>
</aside>
<! - Тело поста идет здесь.->
</article>
```

Тег TIME

Вы заметили в предыдущем примере новый тег? Тег `time` прост: используйте его для показа времени. Вы также можете представить машинно-читаемую версию.

```
<article> <p>Опубликовано <time datetime="1984-04-03" pubdate>3 April 1984</time></p> </article>
```

Атрибут `pubdate` может употребляться для указания исходной даты издания статьи. В спецификации сказано, что атрибут `pubdate` следует использовать только один раз для тега `article`.

NAVIGATION

Элемент `nav` явно предназначен для навигации сайта. Вы можете вставлять теги `nav`, чтобы создавать выпадающее меню. Тег не годится для списка презентаций, который я показывал вам, когда мы говорили о теге `article`. Приберегите `nav` для своего сайта, когда дело дойдет до структурной навигации сайта. Например:

```
<nav>
<ul role="navigation">
<a href="/products">Продукция</a>
<a href="/contact">Контактная информация</a>
<a href="/about">О компании</a>
</ul>
</nav>
```

(Хотите узнать, для чего нужен атрибут `role`? Тогда читайте дальше!)

Теги **FIGURE** и **FIGURE CAPTION**

Возможно, вы добавляете на свои страницы много изображений. А задумывались ли вы когда-нибудь о том, как лучше задать подписи к ним? Вот было бы здорово, если бы можно было аккуратно подписать изображение! Для этого-то нам и нужен тэг `figure`.

```
<figure>
```

```
  
<figcaption>Бокал виски, рядом  
маленькая бутылочка с водой.</figcaption>  
</figure>
```

Это еще не все. Вы можете использовать тэг `figure` для видео, SVG и всего визуального, к чему может потребоваться подпись.

Тег DIV

Познакомившись со всеми этими новыми тегами, вы наверняка считаете тег `div` пережитком прошлого. Это далеко не так. Уже несколько лет разработчики повсюду используют этот простой тег для всего на свете, словно их охватила эпидемия «дивификации».

`Div` (division) означает «раздел», и порой нет лучшего тега для описания куска контента. Возможно, все, что вам нужно, это контейнер для применения стилей. Так бывает. Это не в упрек вам. Семантика – штука коварная. Если вы на самом деле не умеете описывать куски контента с использованием каких-то тегов HTML, о которых мы говорили выше, берите `div` и не казните себя за это.

Несколько слов о семантической структуре

Теперь, когда у нас есть несколько новых элементов для разметки блоков (т. е. `section` и `article`), простая структура старого документа, которую мы использовали, чуть-чуть изменилась. Элементы разметки блоков можно рассматривать как блоки всего html-документа. Другими словами, теги заголовков от `h1` до `h6` могут использоваться внутри них.

Но будьте внимательны! Вы можете столкнуться с чем-то подобным:

```
<body>  
<header role="banner">  
<h1>Блог о велоспорте</h1>  
</header>
```

```
<article>
<h1>Раннее утро на дороге Блэкспур </h1>
...
</article>
</body>
```

Множественно вставлять h1 в один и тот же документ? Это безумие! Вместо этого я применяю заголовки, чтобы показать структуру внутри заданного раздела:

```
<body>
<header role="banner">
<h1>Полноформатный кадр: Блог о фотографии</h1>
</header>
<article>
<h2> Раннее утро на дороге Блэкспур </h2>
...
</article>
<section>
<h2>Купите нашу книгу!</h2>
...
<section>
<h3>Печатное издание</h3>
...
<button>Купитьза $90</button>
</section>
<section>
<h3>Электронное издание: PDF или eBook</h3>
...
<button>Купитьза $15</button>
</section>
</section>
</body>
```

Это не только упрощает оформление заголовков, но и выглядит лучше: меньше путаницы и не надо плыть против течения.

До того, как в наше распоряжение попали теги section и article, мы могли использовать для описания глубокой иерархии сайта только теги от h1 до h6. Теперь мы вольны описывать безграничные уровни глубины и можем точно представлять каждый уровень контента.

Если после всего этого вы все еще сомневаетесь, какой элемент применить, проштудируйте классную блок-схему по элементам разметки блоков в HTML5^[33]. Разработчики – Оли Студхолм и Петр Петрус. Распечатайте ее, приколите на стену, и вы всегда будете знать, какой элемент использовать.

Как обычно, вы захотите провести валидацию кода HTML, чтобы он был без ошибок.

Я предпочитаю это делать через Validator.nu^[34].

Работа со стандартом WAI-ARIA с самых азов

Роли стандарта доступности активных интернет-приложений WAI-ARIA (сокр. от Web Accessibility Initiative: Accessible Rich Internet Applications) всегда входили в состав современной технологии HTML. У многих разработчиков темнеет в глазах при одном только упоминании о них. Роли созданы для того, чтобы сделать сайты и приложения более доступными для пользователей с ограниченными физическими возможностями, которые полагаются на программы, читающие вслух текст с экрана. Разработчики, посвятившие себя этому благородному делу (на самом деле их не так уж и много!) постоянно говорят о важности WAI-ARIA, но профессиональное сообщество в целом их как будто не замечает.

Компании могут много говорить о том, как важна доступность. Исследуют ли они этот вопрос или разрабатывают его – это уже совсем другая история. А вот ваша задача, как разработчика, – сделать так, чтобы каждый человек мог пользоваться вашим сайтом.

Для слепых и людей с пониженным зрением WAI-ARIA роли описывают контекст и цель информации, лежащей за ним. Разделы страницы отличаются не только визуально, но и по контексту. А программа, читающая с экрана, может объяснять эту разницу пользователям и позволяет им без проблем взаимодействовать с разделами.

Я ни в коем случае не позиционирую себя как эксперта по стандартам доступности. Но очень постараюсь внятно и серьезно объяснить вам, почему важность стандарта WAI-ARIA не ограничивается только вопросами доступности. Если вы когда-нибудь пользовались блочными элементами, которые описаны в этом разделе, то, скорее всего, встречались с чем-то вроде этого:

```
<body>
<header>
<h1>Тракторы: Интерактивное руководство</h1>
</header>
<article>
<header>
<h2>Обслуживание тракторов</h2>
</header>
</article>
</body>
```

Заметили? У документа два тега header, и оба использованы правильно. Вся проблема в CSS:

```
hheader {
margin: 0 2em;
}
```

Этот селектор элемента описывает оба тега header. Мы могли бы использовать селектор наследования (т. е. body > header), но это кажется немного тяжеловесным, не говоря уже о том, что верхний заголовок может стать «шапкой» целого сайта. Здесь мы можем выгодно использовать роль WAI-ARIA, просто добавив role="banner" в HTML:

```
<body>
<header role="banner">
<h1>Тракторы: Интерактивное руководство</h1>
</header>
<article>
<header>
```

```
<h2>Обслуживание тракторов</h2>
</header>
</article>
</body>
```

Этот ролевой атрибут говорит о том, что элемент `<header role="banner">` – «глобальный» и его содержимое лучше применять ко всему сайту, а не только к текущей странице. А благодаря простому селектору атрибутов, несложно задать стиль:

```
header [role="banner"] {
margin: 0 2em;
}
```

Из-за того, что теги `header` и `footer` могут использоваться во многих местах, пожалуй, мы бы остались без тега для основного контента. И снова спасибо ролям ARIA, которые предоставляют богатый выбор.

Добавляя роль `"main"` к `article` (таким образом, получаем `<article role="main">`), мы можем легко определить, что основной контент для текущего документа расположен внутри `article`. (В примерах выше вы могли заметить, что тег `h1` используется в верхнем заголовке (корневом), а `h2` – во вложенном.) В сочетании они в лучшем виде описывают иерархию документа).

Вы, вероятно, уже отметили изящество этого подхода. Описание контента становится более детальным, и мы можем применять стили к нашим новым тегам, не особо напрягаясь. Третья из ролей ARIA – это `contentinfo`, которая часто применяется для конфиденциальных заявлений, уведомления об авторских правах и общей информации о текущей странице сайта. (Некоторые называют это «метаинформацией»).

И, наконец, четвертая и очень полезная роль ARIA, про которую нужно знать, – это `navigation`. Она легко отличает навигационный раздел от обычного старого списка ссылок. Добавление ролей ARIA – хороший способ сделать контент и контекст вашего существующего сайта более наглядным. Потом, когда вы решите усовершенствовать сайт, вы сможете прибегнуть к новым тегам.

Надеюсь, это небольшое введение помогло вам убедиться в пользе семантического контента. Роли ARIA – отличный пример тому.

Хранилище на стороне клиента

А теперь поговорим об абсолютно новой теме: хранилище на стороне клиента в HTML5. На сегодняшний день мы имеем невеликий выбор средств для хранения данных у пользователя. Наиболее распространенным способом стала скромная куки-сессия. Но этому методу сопутствует множество маленьких проблем. И самые выматывающие из них – это следующие:

- Данные, которые вы сохраняете в сессии, перемещаются туда и обратно между клиентом и сервером при каждом запросе.
- Данные, которые вы сохраняете, имеют лимит в 4 kB.
- Все куки ограничены во времени.

Впрочем, с использованием куки все обстоит не так уж плохо. Куки – то, что сохраняет данные пользователя для регистрации на сайте и помогает серверу идентифицировать его. Ясно, что нам нужны и другие варианты для сохранения данных. К счастью, у нас есть фантастическое решение в локальном и сессионном хранилище. Что это? Рад, что вам интересно.

С `localStorage` и `sessionStorage`, у нас есть два JavaScript API для сохранения строк для браузера. `SessionStorage` очищается, когда заканчивается сеанс пользователя (т. е. когда «вкладка» или «браузер» закрываются), а в это время `localStorage` хранится, пока разработчик (через JavaScript) или пользователь (через свои браузерные настройки) не решают удалить его.

Интерфейсы API виртуально идентичны – единственная разница будет в сроках хранения. Откройте свою панель инструментов разработчика в современном браузере (имеется в виду выпуск последних трех лет).

Введите `localStorage.setItem("name", "Ben")`. В Webkit-браузерах вы увидите мое имя, сохраненное под вкладкой «Ресурсы» (для этого вам нужно будет раскрыть “Local Storage”). Вы только что сохранили свой первый элемент данных в `localStorage`.

А теперь давайте извлечем то, что вы сохранили, используя `localStorage.getItem("name")`. Вы увидите “Ben”, четко напечатанное в консоли.

И, наконец, чтобы очистить все после себя, воспользуйтесь либо `localStorage.removeItem("name")`, чтобы удалить мое имя, либо `localStorage.clear()`, чтобы убрать все из `localStorage`. Когда пользователи вызовут `localStorage.clear()`, они лишь очистят его для текущего домена.

Итак, если пользователь сохраняет какие-либо данные на сайте, расположенном на `example.com`, а потом переключают вкладки на `google.com`, они увидят, что у них нет доступа к данным, сохраненным ими на вкладке `example.com`.

Интерфейс `localStorage` – чрезвычайно полезная штука. Допустим, вы создаете Twitter-клиент и хотите, чтобы он мог делать следующее:

- Использоваться в вашем интернет-браузере для настольных компьютеров и на мобильном устройстве.
- Отображать в режиме онлайн сообщения с вашего последнего сеанса.
- В режиме офлайн ставить сообщения в очередь и отправлять их позже.

Рисунок 3.6. Инспектор хранилища в браузере Safari

С `localStorage` все это возможно. Пример ниже наглядно демонстрирует это. (Он чисто гипотетический, поэтому не обращайте внимание на мелочи.)

```
postTweet = function(tweetText) {  
  // Проверяем, в онлайн ли мы  
  if(navigator.onLine) {  
    // Привет, мы – онлайн! Отправь это сообщение,  
    крошка!  
  } else {
```

```
// Хм, мы сейчас не в Сети. Лучше сохраните это  
для другого случая.
```

```
localStorage.setItem("queue-" + +new Date(),  
tweetText)}  
}
```

Несложно, правда? Чтобы увидеть все эти элементы в `localStorage`, мы должны повторить код и создать массив:

```
for (item in localStorage) { console.debug(item)  
}
```

Это выведет список всех ключей элементов, которые вы сохранили. Вы хотите вывести на экран твит, поставленный в очередь? Посмотрите, как это можно сделать:

```
for (item in localStorage) {  
console.debug(localStorage[item]) }
```

API `localStorage` и `sessionStorage` можно найти во всех современных браузерах (включая версию IE 8+). Итак, нет ни одной причины, мешающей вам создавать свои собственные приложения или просто начать экспериментировать с этим в клиентских приложениях.

В заключение

Прежде чем заменить все разметки на вашем текущем сайте, найдите время, чтобы изучить роли стандарта ARIA и производительности браузеров. Узнайте в общих чертах, как создавать код. Пробуйте новые теги, это гарантированно поднимет вам настроение. А использование стандарта ARIA и вовсе дело благородное – это поможет многим пользователям с ограниченными возможностями свободно использовать ваш сайт. Звучит почти поэтично, но на самом деле все гораздо проще: это – ваша работа.

Начиная применять новые технологии, не думайте, что вы должны использовать новый минимальный DOCTYPE. Браузеры будут использовать любые свойства, которые они смогут отобразить на вашем сайте.

Не существует «режима HTML5», так что смело ныряйте!

Это всего лишь апробация платформы, которую мы называем HTML5. Мы могли бы говорить о ней еще несколько дней, но лучше дадим вам несколько полезных ссылок:

- HTML5 Please, html5please.us Хотите знать, когда вам пора «латать» старые браузеры? Или определять, когда суперновые теги не совсем готовы к прайм-тайму? Этот сайт даст вам основу для того, чтобы подняться на ступеньку выше.

- HTML5: Техническая спецификация для веб-разработчиков, smashed.by/whatwg Это руководство – сокращенная версия полной спецификации HTML5. Из нее убраны все те бестолковые детали, которые нужны поставщикам браузеров для их создания. В ней есть поиск, она работает на мобильных устройствах (даже офлайн) и, к слову, была создана вашим покорным слугой.

- HTML5 Rocks, html5rocks.com Этот сайт поддерживается сотрудниками Google, почти каждая статья не просто познавательная, но и увлекательная.

- HTML5 Doctor, html5doctor.com Помимо того, что сайт создан компанией классных парней, HTML5 Doctor проник на такую глубину, на которую осмелится не каждый. Прекрасный источник знаний!

Я здорово горжусь тем, что знаком с известными людьми! По всему разделу я то и дело «вворачивал» их имена при каждом удобном случае. Но тому есть оправдание. Люди и сайты, о которых я говорил, – лидеры в этой индустрии.

Очень советую примкнуть к ним на Twitter или Google+, подписаться на их блоги, да просто купить им пива. Ничто не даст вам больше знаний и опыта в вебе, чем помощь в создании крепкого онлайн-сообщества. Засим я вас оставляю, а вы начинайте переделывать свой сайт. Желаю удачи!

Об авторе

* * *

Бен Шварц вкладывает свою любовь к хорошей еде (дома) и саке (в барах), разрабатывая сложные веб-приложения на технологиях,

основанных на стандартах. Больше всего на свете им движет маниакальное желание производить не только элегантный код, но и красивое программное обеспечение для своих пользователей. Он также член комитета Руби (Австралия) и присоединился к рабочей группе W3C CSS (World Wide Web Consortium) в качестве «приглашенного эксперта» в декабре 2011 года.

О рецензенте



* * *

Рас Уэкли (р. 1965) родился в Сиднее (Австралия), живет в Западном Чатсворде, зеленом пригороде на севере Сиднея. Имеет диплом в изобразительном искусстве и графическом дизайне и работы по пользователеориентированному веб-дизайну, разметке и программированию, управлению проектами, опыту пользовательского взаимодействия, доступности и обучению. Он работает с Интернетом с 1995 года. У Раса двое маленьких детей, поэтому у него нет времени на хобби. Раньше у него было три собаки, но с тех пор как они умерли, он не завел ни одной. Любимый цвет Раса – черный. Важный урок, который он получил за время своей карьеры, – «Это тоже пройдет», что относится ко всему в жизни, включая бизнес. Его личное послание читателям: «Займись делом!»

Меняем стиль, переписываем код и преобразуем сайт с помощью CSS3

Автор: Дэвид Стори, Ли Веру

Рецензент: Тэб Аткинс, мл.

К этому моменту мы все уже в курсе, что веб-стандарты являются фундаментальной базой в нашей работе, а семантический язык HTML – это мировая вещь. Мы создаем шаблоны без таблиц и боремся за правильную семантику. Однако многие из нас все еще пишут код по методам, популярным среди разработчиков в первые годы появления стандартов, – некоторые из них состоят из чрезмерно усложненных разметок и неприятных «причесываний» CSS, если не сказать больше.

Хотя эти приемы не являются неверными сами по себе, они уже не оптимальны и порой тянут нас назад, не давая нам развиваться как разработчикам в лучшую и эффективную сторону.

В предыдущем разделе мы научились переписывать разметку, чтобы она стала более изящной, семантической и современной. В этом мы узнаем, как с помощью изменения CSS сократить количество изображений, HTTP-запросов, кода на JavaScript для функций представления и оберточных тегов div, чтобы создать более гибкий и легкий в поддержке стиль.

Сайты не должны выглядеть одинаково во всех браузерах

Прежде чем продолжить, надо сказать, что есть один предрассудок, с которым нам надо, наконец, разобраться и помочь нашим друзьям и коллегам избавиться от него. Все мы должны понимать, что сайты не могут выглядеть одинаково в разных браузерах, хотя клиенты и дизайнеры старой школы часто не согласны с этим. По правде сказать, только вы, ваш клиент и ваши коллеги будут проверять сайт в различных браузерах. Ваши посетители обычно используют один из них. И вы просто счастливчик, если они вообще знают, что такое браузер.

Если сайт не разваливается в их браузере, они не станут включать четыре разных для сравнения, а останутся с тем, который есть. Если вы замените ваши длинные куски хаков на код CSS3, для одних посетителей это будет здорово, но для других – нет. И это нормально. Ни один человек не пожалуется на сайт из-за того, что у того нет закругленных углов, теней или градиентов. До тех пор, пока вы предусматриваете надлежащую нейтрализацию ошибок, никто не заметит что что-то не так.

Система нейтрализации ошибок

Использование изящной обработки ошибок таблицами CSS и каскадность – самый несложный способ обеспечить систему восстановления. Основная идея проста: в CSS, браузеры игнорируют то, чего они не понимают. Так, если они не понимают свойство или значение, они проигнорируют все описание. Если они не понимают селектор, они проигнорируют целое правило. Если они не понимают правило, начинающееся со знака @, они проигнорируют все внутри него.

Посмотрите на этот простой CSS-код:

```
a {  
  color: black;  
  color: super-cool-new-css-color;  
  super-cool-new-css-property: awesomesauce;  
}  
a: hawt-new-pseudoclass(awesomeness) {  
  color: hotpink;  
}
```

В браузерах, которые поддерживают: `hawt-new-pseudoclass`, цвет ссылок, которые размечены им, будет `hotpink`. Во всех остальных случаях цвет определяется первым правилом. В таких случаях, если `super-cool-new-css-color` – поддерживаемый браузером цвет, то ссылки будут такого цвета. В противном случае они будут черными. А если поддерживается свойство `super-cool-new-css-property`, цвет будет применяться ко всем ссылкам;

в противном случае он будет проигнорирован и не вызовет проблемы. Иногда (правда, редко), этот метод не помогает. Например, когда не поддерживаются новые приемы верстки, вам нужно задать полностью другой макет, используя множество свойств, которые не будут перезаписаны на новые. В таких случаях поможет определение поддержки фич.

Modernizr^[35] от Фарука Атеса и Пола Айриша считается одной из наиболее используемых библиотеки поддержки фич. Она добавляет классы к корневым элементам, которые вы потом сможете использовать в своей CSS-ветке соответственно:

```
no-flexbox section {  
  /* разметка макета со старыми стилями */  
}  
flexbox section {  
  /* разметка с поддержкой новых клевых штук */  
}
```

При скачивании библиотеки вы даже можете настраивать ее и «сгенерировать» облегченную версию для выявления только тех с вам нужны. Таким образом, вы получаете минимальные размеры файла.

Как читать этот раздел

Многие свойства библиотеки CSS3, описанные в этом разделе все еще используются с префиксами браузеров. Как только свойство превращается в стандарт и широко поддерживается браузерами, префиксы могут быть «выброшены». Для того чтобы воспользоваться ими сегодня, вам нужно приписывать к началу свойства один или несколько префиксов, которые даны ниже:

Префикс	Браузерный движок
-o-	Opera
-ms-	Internet Explorer
-moz-	Firefox
-webkit-	Браузеры на основе WebKit, такие как Chrome и Safari ²

^[36]

Есть другие и другие префиксы^[37], но они обычно не стоят того, чтобы поднимать шум. По большому счету префиксы производителя раздражают, но они очень полезны когда результат в браузерах отличается друг от друга, и спасают нас от ужасных кусков CSS, которыми мы пользовались в прошлом^[38].

Для краткости и простоты примеры кода в этом разделе будут даны только без префиксов, до тех пор пока код не нужно будет изменить для каких-нибудь реализаций.

Префиксы производителя часто необходимы. Присутствие в вашей таблице стилей всех префиксов -o-, -ms-, -moz-and-webkit – наихудший сценарий. Не всем свойствам они нужны. Вы можете посмотреть, какие префиксы еще нужны, в справочных таблицах по поддержке браузеров.

На ресурсах When Can I Use^[39], HTML5 Please^[40] или в документации Mozilla Developer Network^[41]. Кроме того, если не указано иное, каждый сниппет в этом разделе полагается на следующую простую разметку HTML:

```
<html>
<head>
<meta charset="utf-8" />
<title>Изучаем CSS3</title>
</head>
<body>
<section>
<h1>Изучаем CSS3</h1>
```

```
<p> Это образец контента. Не читайте его. Вы
просто потратите время. Почему вы продолжаете
читать? Мне что, нужно использовать Lorem Ipsum,
чтобы остановить вас? Чтож, пожалуйста. Lorem
ipsum dolor sit amet, consectetur adipi sicing
elit, sed do eiusmod tempor incidi dunt ut labore
et dolore magna aliqua. Читаете? Черт возьми, вы
невыносимы. Я закончу, чтобы поберечь вас.</p>
```

```
</section>
</body>
```

</html>

Мы рассмотрим некоторые основы в начале каждого раздела, но потом перейдем к более продвинутым техникам, так что, пожалуйста, будьте к нам снисходительны. Мы откроем различные способы стилизовать простую страницу, представленную выше, попутно изучая всякие свойства CSS3. Готовы? Тогда приступим.

Веб-типографика и техники работы с текстом

До сегодняшнего дня Web был скуден в оформлении. Пока мы не обратились к методам замены изображений с их преимуществами и недостатками, мы были привязаны к некоторому ограниченному набору базовых шрифтов в операционных системах, существовавших с незапамятных времен. Хотя проблема была вовсе не в шрифтах. У нас был недостаточно тонкий контроль за разметкой текста и лигатурами.

Но CSS3 меняет все. У вас никогда не было лучшего повода отойти от базовых шрифтов и расширить горизонты оформления. Перейдем к теме о том, как добавить ритма в вашу типографику с помощью величины шрифта в относительных единицах `rem`, эксперимента с новыми шрифтами, контролем переноса слов, и как полностью улучшить ваш текст. Тем не менее самый важный элемент на странице – это все-таки текст, и он заслуживает особого внимания.

Представляем REM

Если у вас есть некоторый опыт с работы с тянущимися макетами, то наверняка вы пользовались единицами `em`. В силу того что `em` – это краеугольные камни адаптивного веб-дизайна, их использование может быть сложным. Они основываются на размере текущего шрифта элемента (или его родительского элемента в случае `font-size`), поэтому тут нужно вспомнить математику, чтобы рассчитать размер 1 `em`. Но будем смотреть правде в глаза, математику любят не все.

Новая единица `rem` ("`root em`") – близкий родственник `em`, с теми же плюсами, только более легкая в использовании. Величина `rem` устанавливается по размеру шрифта в корневом элементе (это тег `<html>` для языка HTML). Поэтому 1 `rem` всегда одного размера, и неважно, где вы ее ставите в документе.

Чтобы `rem`, как и пиксели, было проще использовать, вы можете установить размер шрифта элемента `<html>` в 62,5 %, который будет пересчитан в 10 `px`, если только пользователь не настроил размер шрифта по умолчанию.

Теперь, когда вы задаете длину в `rem`, вы можете просто умножить на 10, чтобы получить величину в пикселях. Так как 10 пикселей может быть слишком мало для основного текста, вы можете настроить

какой хотите размер `rem` в элементе `<body>`. Потом эту величину унаследуют все элементы. Например, если вы хотите, чтобы размер текста «тела» был 15 пикселей, можете сделать следующее:

```
html { font-size: 62.5 %; }  
body { font-size: 1.5rem; /* 15px */ }
```

Один единственный недостаток величин `rem` – их поддерживают все основные браузеры, кроме Internet Explorer (IE) 8 и его ранних версий. Как альтернативу используйте каскадность стилей и установите размер в пикселях до настройки его в `rem`. Это может значительно увеличить вашу таблицу стилей. Поэтому вы можете использовать отдельную таблицу стилей с условными комментариями к IE 9 и его ранним версиям, если только вам не нужна поддержка iOS 3.2, Safari 4 и старых версий Opera.

Задание ритма в `rem`

Теперь, когда в нашем распоряжении величина, которую, как и пиксели, легко использовать и которая гибкая, как `em`, создать вертикальный ритм будет намного проще. О каком вертикальном ритме идет речь? Основная идея в том, что когда вы проматываете страницу вниз, текст и элементы страницы следуют заданному ритму.

Если мы проведем вниз страницы воображаемые строки с фиксированными интервалами, каждая линия текста и каждый отступ или другой элемент страницы будет занимать одну линию или кратную ей величину линий.

Они соблюдают ритм страницы, словно басист из блюз-группы. Это уплотняет дизайн и обеспечивает правильное выравнивание элементов страницы, даже вдоль колонок. Выдерживание ритма поможет добиться визуальной последовательности в макете страницы.

Для начала мы должны настроить ритм страницы. Например, на следующей странице мы установили его в `2.3 rem` или 23 пикселя. Он задается в элементе `body` с использованием свойства `line-height`. Мы также установили размер шрифта (`font-size`) в `1.5 rem`, для того, чтобы все основные элементы типографики, кроме заголовков унаследовали его.

Удобно иметь под рукой визуальные направляющие, чтобы видеть, где вы слегка перескочили. Для этого мы создали изображение в формате SVG для показа линий с 23-пиксельными интервалами. Также мы использовали базовый сброс стилей для удаления отступов и полей, чтобы они не нарушали наш ритм:

```
html {  
  font-size: 62.5 %;  
  background: url(line.svg) no-repeat;}  
/* reset */  
body, div, dl, dt, dd, h1, h2, h3, h4, h5,  
h6,pre, p, th, td,  
article, section, figure, img {  
  margin: 0;  
  padding: 0;  
}  
body {  
  font-size: 1.5rem;  
  line-height: 2.3rem;  
}
```

Если бы в нашем тексте были только текстовые параграфы, наша работа была бы почти закончена, даже если параграфы будут сжаты друг с другом без границ.

Заставляем заголовки соответствовать ритму

Настройка параграфов была несложной, так как каждая строка текста имеет междустрочный интервал (line height) меньше, чем 23-пиксельный вертикальный ритм страницы. Но что нам делать с такими элементами, как заголовки, которым нужен размер шрифта больше, чем высота нашей строки или дополнительные отступы сверху и снизу? Мудрость здесь в том, чтобы создать комбинацию из верхнего и нижнего отступов, при этом нужно, чтобы междустрочный интервал был кратен нашему ритму:

```
h2 {
```



```

font-size: 2.6rem; КОД (CODE)
line-height: 4.6rem; /* две базовых величины */
margin: 2.3rem 0 0 0; /* величина ритма сверху и
снизу */
}
h3 {
font-size: 2.1rem;
line-height: 2.3rem; /* базовый ритм */
margin: 2rem 0.3rem 0; /* 2 сверху + 0.3 снизу
== базовый ритм
}

```

Неважно, сколько строк заняли наши заголовки, они всегда будут занимать точные величины, кратные ритму. Так ритм будет идти вниз по странице. Потом мы сможем применить эту технику для всех элементов страницы, таких как изображения и фрагменты кода.

Имейте в виду, что рамки тоже занимают пространство, так что вам придется рассчитать их так же, уменьшая отступы. С другой стороны, тени не занимают пространство, поэтому вы можете делать с ними все, что хотите.

Если бы мы использовали величины `em` в этом примере, нам нужно было бы пересчитать высоту строки и отступы (`margin`), с учетом размера текста. С `rem` мы почти полностью можем игнорировать размер шрифта, только надо удостовериться, что высота строки достаточна, чтобы включить текст без нахлеста^[42].

GEORGIA у меня на уме

Мы любим Helvetica и Georgia, но не пора ли этим почтенным шрифтам взять отпуск? Весь мир типографики открывается нам, а веб-шрифты ждут, когда мы испробуем всю прелесть их экзотики.

Форматы шрифта имеют целую политическую историю, но мы не станем грузить вас этим здесь. Проблема в том, что вам нужно использовать массу форматов, если ищете надежное кросс-браузерное решение. К нашей великой радости, целая армия решений сама бросилась нам на помощь, поэтому можете не забивать себе этим сильно голову, если только этого вы и не хотели.

Добавить свой собственный @font-face

Ваш первый вариант – это использовать шрифт, установленный вами и подгружать его с собственного сервера. Но хотим заострить ваше внимание на том, что прежде, чем делать это, проверьте, лицензионный шрифт или нет, и есть ли у вас надлежащие права. Не всякий бесплатный и коммерческий шрифт разрешается использовать как веб-шрифт; а даже если это происходит, вам, скорее всего, велят купить дополнительную лицензию для этих целей.

Как только вы выберете шрифт, нужно будет подготовить его в правильном формате и загрузить его через правило @font-face. Мы открыли лучший способ – использовать бесподобный сервис Font Squirrel^[43]. Он позволяет загружать шрифт, потом «выдает» его в нужных вам форматах, с правилом @font-face внутри. Все, что от вас требуется, это перенести шрифты на ваш сервер, скопировать код в вашу таблицу стилей, а потом обращаться к шрифту, когда захотите.

Для примера в этом разделе мы взяли шрифт Museo Sans 500 для текста в body и Museo 700 для заголовков. Их предоставил Jos Buivenga абсолютно бесплатно^[44]. Давайте посмотрим на код CSS, созданный сервисом Font Squirrel для шрифта Museo 700:

```
@font-face {
  font-family: 'Museo-700';
  src: url('1F9920_0_0.eot');
  src:                                url('1F9920_0_0.eot?#iefix')
format('embedded-opentype'),
  url('1F9920_0_0.woff') format('woff'),
  url('1F9920_0_0.ttf') format('truetype');
}
```

Свойство font-family дает имя шрифта, на которое вам нужно будет сослаться в остальной таблице стилей, а свойство src ссылается на сам шрифт. Браузер будет проверять каждый URL по очереди, пока не найдет поддерживаемый формат. Изначально свойство src должно поддерживать старые версии IE, в которых наблюдается баг при использовании стандартного правила. Если вы вставляете его в свою таблицу стилей, вы можете ссылаться на шрифт обычным способом:

```
h2 {  
  font-family: "Museo-700", serif;  
}
```

Все должно быть в порядке, но вам нужно быть в курсе дополнительных деталей. Шрифт имеет плотность 700, что уже говорит о его жирном начертании. Запомните, по умолчанию браузеры будут считать, что шрифты, которые вы определяете в `@font-face`, это шрифты без выделения, курсива и без всяких остальных шрифтовых прибамбасов. Если вам нужен элемент со свойством `font-weight: bold`, он будет искусственно сделан полужирным начертанием. Так как шрифт Museo используется для заголовков, он уже выделен полужирным начертанием. Если вы наложите на него искусственное полужирное начертание, то получится грязное месиво. Чтобы этого не случилось, мы должны сообщить в правило `@font-face`, что шрифт уже полужирный, с помощью дескриптора `font-weight`:

```
@font-face {  
  font-family: 'Museo-700';  
  ...  
  font-weight: bold;  
}
```

Теперь современные браузеры будут знать, что это – полужирный шрифт и больше не станут пытаться сделать его полужирным сами. Добавляя правила `@font-face` с тем же свойством `font-family`, но с разными величинами для `font-weight`, `font-style` и `font-stretch`, вы можете объединять многочисленные шрифтовые файлы в одно имя шрифта. Тогда браузеры будут использовать правильный шрифт, когда нужно переключиться с полужирного начертания на курсивное.

Используем шрифтовые онлайн-сервисы

Вы можете пользоваться одним из доступных онлайн шрифтовых сервисов, вместо того, чтобы поддерживать свой собственный.

Наиболее популярны Google Web Fonts для бесплатных шрифтов и TypeKit – для тех, что предоставляются по подписке. Часто такие подписные сервисы – единственный законный путь получения доступа к известным коммерческим шрифтам.

Рисунок 4.1. Прекрасный ритмичный шрифт с использованием вертикального ритма и атрибута @font-face

Каждый сервис имеет свой собственный способ подключения шрифтов. В нашем случае мы использовали Google Web Fonts для основного заголовка страницы. Google представляет три способа: через тег `link`, через `@import` и через JavaScript. Мы остановились на использовании тега `link` и добавили его в шапку документа как:

```
<head>
<meta charset="UTF-8" />
<title>Типографика в CSS3 </title>
<link href='http://fonts.googleapis.com/
css?family=Abril+Fatface'
rel='stylesheet' type='text/css'>
...
</head>
```

Потом в таблице стилей шрифт может указываться, как раньше, по имени “Abril Fatface”:

```
h1 {
font-family: “Abril Fatface”;
font-weight: normal;
...
}
```

В этом случае мы настраиваем обычное начертание шрифта, потому что Firefox пытается искусственно сделать полужирным уже полужирный шрифт.

Из-за того что мы не можем контролировать, настраивает ли правило `@font-face` шрифт как полужирный, нам приходится делать все наоборот и устанавливать нормальную насыщенность каждый раз при его использовании. Как видите, на выходе мы имеем прекрасный ритмичный шрифт, без намека на базовые гарнитуры:

Встречайте нового члена семейства, используя FONT-STRETCH

Свойство `font-stretch` возможно имеет самое неудачное название в CSS.

Оно не расширяет шрифт искусственно, но взамен позволяет вам использовать растянутое и уплотненное начертание шрифта в гарнитуре. К примеру, если вы захотите использовать шрифт Helvetica Neue Condensed без свойства `font-stretch`, вам нужно будет подключить его как веб-шрифт (необходима лицензия) и обозначить его под другим именем в шрифтовом семействе. С `font-stretch` это делается проще:

```
h1 {  
  font-family: "Helvetica Neue";  
  font-stretch: condensed;  
}
```

Большинство шрифтов не имеют узкого или широкого начертания, так что при использовании другого шрифта, не Helvetica, вам, вероятно, придется подключать его как веб-шрифт. Свойство `font-stretch` имеет такие значения: `normal`, `ultra-condensed`, `extra-condensed`, `condensed`, `semi-condensed`, `semi-expanded`, `expanded`, `extra-expanded` и `ultraexpanded`. Браузер применит значения к ближайшему подходящему начертанию в семействе. В случае с Helvetica Neue ОС X все сжатые значения отображаются как Helvetica Neue Condensed до тех пор, пока вы не установите дополнительные шрифты.

Перенос слов

Мы привыкли видеть текст в вебе выровненным по левому краю с рваным правым. При печати часто применяется выравнивание вместе с переносами, чтобы избежать «рек» в тексте (больших пробелов между словами и символами). Не так давно перенос был недоступен в Сети. Но сейчас все меняется. Свойство `hyphens` (перенос) доступно с префиксами в Safari (с версии 5.1) и Firefox, и в IE 10.

В примере ниже мы настроили выравнивание текста по всей ширине и автоматический перенос:

Рисунок 4.2. Пример параграфа с переносом (слева) и без переноса (справа)

```
p {  
  text-align: justify;  
  hyphens: auto;  
  ...  
}
```

Свойство `auto` указывает браузеру использовать словари переноса для разбивки слов в нужных местах, и вставлять переносы в конце строк. Каждому языку присущи свои словари переноса, поэтому поддержка этой функции будет варьироваться. Если вы вставляете свой собственный символ мягкого переноса (), это аннулирует автоматическое поведение браузера. Если вы хотите настроить перенос вручную, можете использовать ручной режим. Отключение переноса (по умолчанию) происходит через ключевое слово `none`.

Вот пример с переносом (слева) и без (справа). Обратите внимание на неровные интервалы между словами во второй и шестой строках текста без переносов.

Журнальная верстка

В книгах и журналах текст обычно располагают несколькими колонками. Это делается для удобочитаемости. Длина строк остается короткой, при этом дизайнеры черпают полную выгоду от ширины страницы (или, в случае с Интернетом, от ширины окна браузера).

Многоколоночная спецификация наделяет этой возможностью CSS. Это позволяет вам точно определить ряд атрибутов для колонок, такие как их количество, ширина и размер интервала между ними.

Доработаем пример выше с использованием колонок, чтобы попробовать, каково это.

Определение колонок

Вы можете установить либо ширину, либо количество колонок. Браузер рассчитает другую величину, основанную на заданной, и доступную ширину контейнера. В этом примере мы зададим две колонки только для разделов, которые расположены в статье глубиной на два уровня. Это убережет нас от многочисленных колонок во вводном блоке и предотвратит разбивку колонок на ячейки, если мы добавим еще один вложенный блок.

Количество колонок может устанавливаться свойством `column-count`, которое допускает своей величиной целое число. Мы также применим такой интервал между колонками, чтобы они располагались красиво. В этом нам поможет свойство `column-gap`:

```
article> section > section {  
  column-count: 2;  
  column-gap: 2.6rem;  
}
```

Для того чтобы определить ширину колонок, вы можете использовать свойство `column-width`, которое допускает величины длины, так же как и `column-gap`.

Размещаемся в нескольких колонках

Часто бывает нужно, чтобы текст или элементы, такие как изображения, охватывали несколько колонок. Это можно достигнуть с помощью свойства `column-span`. В настоящее время можно все или ничего: вы можете либо охватить все колонки, используя значение `all` или не охватывать вообще, применяя значение `none`. Определить

точное число охватываемых колонок невозможно. В нашем примере заголовки `h3` охватывают полную ширину колонок^[45].

```
h3 { column-span: all; }
```

Результат будет выглядеть примерно таким образом, как ниже на рисунке^[46].

Еще одно предостережение в плане многоколоной верстки. Применение этого свойства для длинных кусков текста может привести к тому, что человек, чтобы ему было удобно читать, будет вынужден прокручивать вверх-вниз каждую колонку на странице. Поэтому применяйте это свойство только для коротких отрывков, где длина колонок, возможно, будет меньше, чем окно браузера пользователя. Но когда страница распечатывается, многоколоночная верстка будет удобнее, т. к. размещается на целой странице. Ее хорошо использовать в длинных текстах в таблицах стилей для печати.

Рисунок 4.3. Верхняя колонка как при журнальной верстке. Текстовый блок в верхней части захватывает ширину нескольких колонок в нижней части

Техники верстки/компоновки

Пожалуй, самое сложное на сегодняшний день при работе с CSS – это разметка страницы. Что-то вроде простейшей центровки элемента на странице бывает ужасно сложно сделать. Вы можете поседеть (если не уже), когда верстаете двух– или трехколоный макет с равными по высоте колонками. Эти моменты сводятся к тому, что CSS на самом деле никогда не предоставлял способ разбивки страниц. Годами мы неправильно использовали формат, основанный на свойстве `float`, но это только один хак. Как таблицы, так и свойство `floats`, не разрабатывались для разметки страницы. Они создавались для обтекания изображения внутри текстового блока. А когда нам не хватает подходящих инструментов, мы импровизируем. Мы должны быть благодарны скромному свойству `float` за помощь нам все эти годы. Теперь с CSS3 мы можем попросить `float` подвинуться, потому

что у нас есть новые инструменты для этого. Они из самых многообещающих – создание сетки с помощью тянущихся блоков.

Разметка с помощью тянущихся блоков

Flexible Box Layout (или Flexbox) – это новая блочная модель, оптимизированная под дизайн интерфейса. Дети блока, настроенные для использования модели Flexbox, располагаются вдоль горизонтальных либо вертикальных осей. Их ширина увеличивается или сокращается для того, чтобы заполнять имеющееся пространство на основе назначенной гибкой длины.

Flexbox имеет историческое прошлое. Сначала оно представляло собой свойство для XUL Mozilla (расширенного языка пользовательского интерфейса) и переписывалось тысячи раз. Спецификация только сейчас достигает зрелости, а у нас есть довольно полная поддержка в WebKit и ночных сборках Chrome. Об этом, вне сомнения, полезно знать. Потому что спецификация, возможно, будет реализовываться быстро, если это уже не произошло до того, как вы об этом прочитали. В особенности с быстротой графика релизов Chrome и Firefox.

Пример: Горизонтальная и вертикальная центровка, или Священный Грааль веб-дизайна

Расположить элемент по центру страницы, это, возможно, номер один среди всех запросов веб-дизайнеров. Он, пожалуй, даже превосходит запрос, как положить конец страданиям IE 6. С Flexbox это очень просто.

Начнем с основного шаблона HTML, с заголовка, который мы хотим разместить по центру:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<title>Центровка элемента на странице</title>
</head>
```

```
<body>
<h1>OMG, Я в центре</h1>
</body>
</html>
```

Ничего особенного здесь нет, даже оборачивающего div. Вся магия заключается в CSS:

```
html {
height: 100 %;
}
body {
display: flexbox; /* это значение нуждается в
префиксе */
flex-align: center; /* это свойство также */
flex-pack: center;
margin: 0;
height: 100 %;
}
h1 {
display: flexbox;
flex-align: center;
height: 10rem;
}
```

Это не совсем весь CSS, что нужен для примера, потому что мы убрали оформительские стили, т. к. вы их наверняка знаете, а мы сэкономим место.

По той же причине мы выбросили префиксы. Только WebKit поддерживает их (префиксом `-webkit-`), но не удивляйтесь, если в ближайшем будущем это тоже будут делать и Mozilla, и Opera и IE. Лучше добавлять префиксы на всякий случай. Давайте посмотрим на CSS для выравнивания по центру заголовка на странице. Сначала мы устанавливаем элементы `html` и `body` на 100 % высоты и убираем любые отступы. Это поможет контейнеру нашего `h1` полностью

принять высоту окна браузера. Теперь нам просто все нужно расположить по центру.

Включаем FLEXBOX

Так как элемент `body` содержит заголовок, который мы хотим поставить по центру, мы задаем значение свойства `display` на `flexbox`:

```
body {  
  display: flexbox;  
}
```

Это заставляет элемент `body` применять разметку `Flexbox` вместо обычной блочной. Все его дети в потоке документа (т. е. не абсолютно спозиционированные элементы) становятся элементами `Flexbox`.

Чего же мы добьемся тем, что наши элементы сделались гибкими, словно от занятий йогой? Они приобретают неслыханную силу: могут изменять свой размер и позицию относительно имеющегося пространства, могут располагаться и вертикально, и горизонтально, и даже могут достичь независимости позиционирования от положения в коде. (Два Священных Грааля в одной спецификации? Да у нас все хорошо!)

Центровка по горизонтали

Идем дальше. Мы хотим расположить наш элемент `h1` по центру горизонтально. Раз плюнуть, скажете вы. Да, это, в общем-то, проще, чем «играть» с автоматическими отступами. Всего-то и надо дать команду `Flexbox` расположить по центру его элементы. По умолчанию элементы `Flexbox` распределяются горизонтально, поэтому установка свойства `flex-pack` выравнивает элементы вдоль основной оси:

Рисунок 4.4. Простая горизонтальная и вертикальная центровка с использованием `Flexbox`

```
body {  
  display: flexbox;  
  flex-pack: center;  
}
```

Другие возможные значения – это `start` (начало), `end` (конец) и `justify` (по всей ширине). Значение `start` выравнивает влево (или вправо, если текст идет справа налево), `end` выравнивает справа, а `justify` четко распределяет элементы вдоль оси.

Если вы хотите точно установить ось, вдоль которой выравнивается элемент, то можете сделать это с помощью свойства `flex-flow`. Стандартная настройка – горизонтальная, она даст нам тот же эффект, которого мы только что добились. Чтобы сделать выравнивание по вертикальной оси, мы можем использовать свойство `flex-flow: column`. Если мы добавим его в наш пример, вы заметите, что элемент лег по центру вертикально, но мы потеряли горизонтальную центровку. Реверсирование порядка при добавлении `-reverse` к значениям `row` или `column` также возможно (`flex-flow: row-reverse` или `flex-flow: column-reverse`), но в нашем примере это мало что значит, так как у нас всего один элемент.

Центровка по вертикали

Центровка по вертикали такая же простая, как и горизонтальная. Все, что нам надо, это применить подходящее свойство для выравнивания поперечной оси. Выравнивания чего?! По существу, поперечная ось – это неосновная ось. Поэтому если элементы `Flexbox` выравниваются горизонтально, то поперечная будет вертикальная, и наоборот. Мы делаем эти настройки с помощью свойства `flex-align`:

```
body {  
  display: flexbox;  
  flex-pack: center;  
  flex-align: center;  
}
```

Вот и все, что нужно для центровки элементов с Flexbox! Мы также можем использовать значения `start` и `end`, и `baseline`, и `stretch`. Давайте посмотрим конечный результат. Зайдите для этого по адресу smashed.by/flxbox1.

Как вы могли заметить, текст также расположен вертикально по центру внутри элемента `h1`. Это можно было бы сделать отступами (`margins`) и междустрочным интервалом (`line height`), но мы снова использовали Flexbox, чтобы показать, как это работает с блоками (в этом случае линия текста лежит внутри элемента `h1`). Неважно, какой высоты элемент `h1`, текст всегда будет в центре:

```
h1 {  
  display: flexbox;  
  flex-align: center;  
  height: 10rem;  
}
```

Гибкость в размерах

Если бы центровка элементов было все, на что способен Flexbox, это было бы чертовски здорово, но он может больше. Давайте посмотрим, как элементы Flexbox могут растягиваться и сжиматься, чтобы уместиться в имеющемся пространстве внутри блока Flexbox. Для этого рассмотрим другой пример, который можно посмотреть в браузере по ссылке smashed.by/flxbox2.

Рисунок 4.5. Интерактивное слайд шоу, созданное с использованием `flex-order`

HTML и CSS для этого примера идентичны предыдущему. Мы подключаем Flexbox и центрируем элементы на странице тем же способом. Вдобавок к этому, мы хотим сделать так, чтобы заголовок (внутри элемента `header`) не изменял свой размер. При этом пять блоков (элементы `section`) нам нужно подогнать по размеру, чтобы заполнить ширину окна. Для этого мы используем новую функцию `flex` как значение свойства `width`:

```

    section {
      /* чтобы сэкономить место, мы убрали остальной
код */
      flex: 1;
      height: 250px;
    }

```

То, что мы сейчас сделали, это заставили каждый элемент `section` стать равным одной единице `flex`. Так как мы не задали точную ширину, каждый из пяти блоков будет иметь равную. Элемент `header` примет установленную ширину (277 пикселей), так как он негибкий. Мы разделим оставшуюся внутри элемента `body` ширину на 5, чтобы рассчитать ширину каждого элемента `section`. Теперь если мы изменим размеры окна браузера, они или увеличатся, или уменьшатся.

В этом примере мы установили неизменяемую высоту. Но ее можно было бы настроить на изменяемую точно таким же способом. Возможно, нам не всегда будет нужно, чтобы все элементы были одинакового размера, поэтому давайте сделаем один побольше. По наведению курсора мы задаем элементу две единицы `flex`:

Рисунок 4.6. Интерактивное слайд шоу, созданное с использованием Flexbox

```

section: hover {
  flex: 2;
  cursor: pointer;
}

```

Теперь имеющееся пространство делится на 6, а не на 5, и элемент, на который навели курсор, будет равняться двойной базовой величине. Обратите внимание, что элементу с двумя единицами `flex` не обязательно удваиваться по ширине, по сравнению с тем, у которого одна единица. Он становится вдвое больше, чтобы занять имеющееся пространство, которое добавлено к его «предпочтительной ширине». В наших примерах «предпочтительная ширина» равна 0 (по умолчанию).

Независимый источник порядка

Для нашего последнего трюка в этом разделе мы узнаем, как можно получить независимый источник порядка в нашем макете. Когда мы кликаем на блок, мы даем команду элементу отодвинуться влево ото всех остальных блоков, сразу следуя за заголовком.

Все, что нам нужно, это настроить порядок с помощью свойства `flex-order`. По умолчанию, все `flex`-элементы в нулевой позиции. Из-за того что они имеют одинаковую позицию, они располагаются согласно расположению в коде.

Чтобы выбранный нами элемент переместился на первую позицию, нам просто надо задать меньшую цифру. Мы выбираем `-1`. Нам еще нужно задать заголовку `-1` так, чтобы выбранный элемент не «выскочил» до него:

```
header {
  flex-order: -1;
}
section[aria-pressed="true"] {
  flex-order: -1; /* меньше, чем 0, поэтому идет
перед другими элементами section
*/
  flex: 3;
  max-width: 370px; /* ограничивает от слишком
большого расширения*/
}
```

Надеюсь, вы теперь воодушевлены и получили достаточно начальных знаний о Flexbox, чтобы экспериментировать с вашим дизайном.

Работа с изображениями

Теперь, когда мы лучше пониманием, как создавать продвинутые макеты с помощью CSS3, давайте рассмотрим несколько визуальных техник, которыми мы сможем их чуточку «оживить».

Множественные фоны и градиенты

Взгляните на следующий стиль в двух разных размерах^[47]:



Рисунок 4.7. Широкий вариант множественных фонов и градиентов



Рисунок 4.8. Узкий вариант

Исходя только из изображения, как можно задать его через CSS? Возможно, вы сделаете основную разметку за пару минут, как показано на следующей странице.

```
html {  
  background: #222;  
  min-height: 100 %;  
}  
body {  
  margin: 0;  
}  
section {  
  max-width: 25em;  
  margin: 0 auto;  
  padding-top: 150px;  
  color: white;  
  font: italic 100 %/1.5 'Palatino Linotype',  
  Georgia, serif;}
```

Но как добавлять звезды, силуэты на фоне неба и луну? Мы используем только два элемента (html и body – элемент section слишком узкий), а нам нужно четыре фоновых рисунка.

Раньше мы, скорее всего, пожали бы плечами и добавили несколько оборачивающих элементов div. Или же отказались бы от гибкости и объединили несколько изображений. Но сегодня мы можем урвать свой кусочек пирога. Познакомьтесь с множественными фоновыми изображениями:

```
html {  
  background: url("moon.png") no-repeat 100 % 1em,  
  url("stars.png") repeat-x 0 0,  
  url("city.png") repeat-x bottom,  
  linear-gradient(black, #444);  
  background-color: #222;  
  min-height: 100 %;  
}
```

Вы, наверно, заметили, что последний фоновый рисунок не является внешним файлом. В самом деле, теперь мы можем создавать градиенты прямо в CSS. Градиент, использованный выше, – это простой вертикальный градиент с двумя цветами. Но вы можете делать более сложные вещи с градиентами CSS: заливать разнообразные цвета в разные позиции или под разными углами, или даже создавать радиальные градиенты. А еще – все виды геометрических паттернов с несколькими CSS градиентами.

Однако текущая поддержка для CSS градиентов еще не такая хорошая, как для множественных фонов. Способ, которым мы написали код выше, даже несмотря на то что мы задали фоновый цвет на случай, когда множественные фоны не поддерживаются, мы задаем для случая, когда не поддерживаются даже градиенты. Лучше было бы иметь две альтернативы:

```
html {
background: url("moon.png") no-repeat 100 % 1em,
url("stars.png") repeat-x 0 0,
url("city.png") repeat-x bottom;
background: url("moon.png") no-repeat 100 % 1em,
url("stars.png") repeat-x 0 0,
url("city.png") repeat-x bottom,
linear-gradient(black, #444);
background-color: #222;
min-height: 100 %;
}
```

Правда, есть небольшой повтор? И дело еще хуже с браузерными префиксами. Но у нас ведь есть элемент `body`, так что мы можем извлечь из этого пользу и применить градиент к другому элементу:

```
html {
background: #222;
background: linear-gradient(black, #444);
height: 100 %;
}
body {
```

```
margin: 0;
background: url("stars.png") repeat-x 0 0;
background: url("moon.png") no-repeat 100 % 1em,
url("stars.png") repeat-x 0 0,
url("city.png") repeat-x bottom;
min-height: 100 %;
}
```

Намного лучше, и дает отдачу от возможностей каждого браузера.

Свойства BACKGROUND-ORIGIN, BACKGROUND-SIZE, OUTLINE

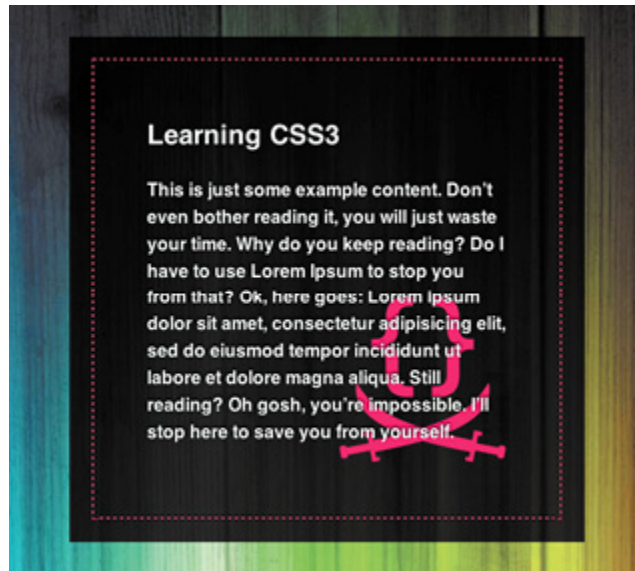


Рисунок 4.9. Дизайн, который мы будем воспроизводить на CSS3

Допустим, нам поручено создать дизайн, отображаемый справа, с использованием CSS^[48]. И опять мы уверены, что вы могли бы быстро сделать базовую разметку и, возможно, интегрировать несколько основ, которые вы знаете о системе цветопередачи RGBA и текстовых теней:

```
html {
min-height:100 %;
background: url('rainbow-wood.jpg');
```

```
}  
section {  
  max-width: 25em;  
  padding: 3em;  
  margin: 4em auto;  
  background: black url(logo.svg) no-repeat bottom  
right;  
  color: white;  
  font: bold 100 %/1.5 sans-serif;  
  text-shadow: 0 - .1em .2em black;  
}  
h1 {  
  margin-top: 0;  
}
```

Результат близок к тому, которого мы добивались. Но есть проблемы:

- Фоновое изображение элемента html – гигантское. Это хорошо для огромных экранов, но, как правило, мы хотим подогнать размер под экраны поменьше.
- Розовый логотип слишком мал и прилеплен к правому нижнему углу без отступа от края.
- Нет розовой «строчки».

Разберем эти вопросы по порядку.

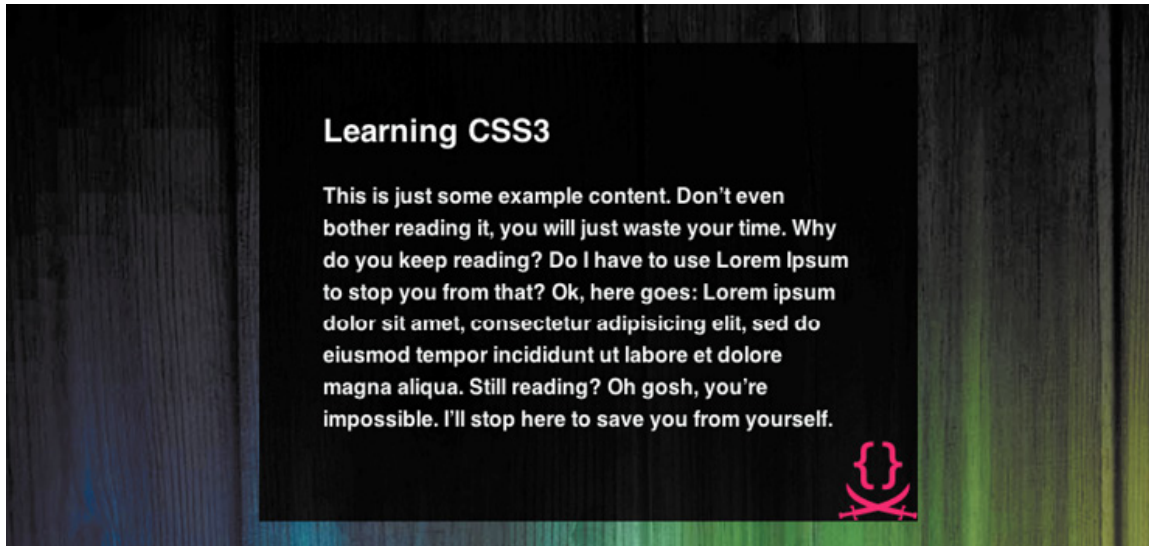


Рисунок 4.10. Результат после применения наших знаний о CSS2.1

Изменяем размер фонового рисунка по ширине

CSS3 дает нам новое свойство для контроля размера фонового изображения: `background-size`. Это позволяет нам использовать одно и то же изображение в разных размерах. Мы можем либо установить точно определенный размер, либо применить один из двух ключевых слов для задания размера: `contain` и `cover`.

- `cover`

изображение будет подстраиваться под полное содержание элемента. Это то, что нам нужно использовать для квадратных превью в стиле Flickr, где каждое изображение обрезается, как квадрат;

- `contain`

изображение будет настраиваться на подгонку элемента, но без обрезания.

В нашем случае мы не сможем применить `background-size: contain`, потому что мы не хотим, чтобы просвечивался фоновый цвет. Вместо этого нам надо, чтобы наше изображение уменьшалось или расширялось для заполнения всего окна экрана.

Правим логотип

Для увеличения размера SVG логотипа мы снова используем свойство `background-size`, но на этот раз устанавливаем точный размер:

```
background-size: 150 px;
```

При задании только одного значения вместо двух мы сохраняем соотношение сторон, при этом изображение расширяется до 150 пикселей по ширине. Но оно все еще «приклеено» к низу. В прежние времена мы бы просто пожали плечами и отредактировали изображение так, чтобы в нем был промежуток. Сегодня с CSS3 у нас есть немного больше контроля при размещении изображения. Если проанализировать примеры выше более детально, мы столкнемся с проблемой – изображение находится внизу справа от области полей, но нам хотелось бы, чтобы оно располагалось внизу справа от текста. CSS3 дает нам новое свойство `background-origin`, которое контролирует позиционирование изображения относительно края элемента с учетом толщины рамки, относительно рамки, либо содержимого элемента.

В этом случае надо поместить изображение в правом нижнем углу содержимого элемента, а не края, как задано по умолчанию. Эта строка CSS сделает что нужно:

```
background-origin: content-box;
```

Добавление розовой строчки

Это было бы просто, если бы розовая пунктирная рамка располагалась с краю контейнера. Такой кусок CSS мог бы сделать это:

```
border: 1px dashed #f06;
```

Но как мы переместим эту рамку внутрь контейнера? Не получается. Вместо этого мы сужаем контейнер, а потом добавляем дополнительный цвет за пределами рамки с помощью свойства `outline`:

```
border: 1px dashed #f06;  
outline: 20px solid rgba(0,0,0,8);
```

Другой способ – использовать свойство `outline` со значением `outline-offset` в `-20px` (здесь мы не применяем эту технику). Вот полный сниппет:

```
html {
  min-height: 100%;
  background: url('rainbow-wood.jpg');
  background-size: cover;
}
section {
  max-width: 20em;
  padding: 3em;
  margin: 4em auto;
  border: 1px dashed #f06;
  outline: 20px solid rgba(0,0,0,8);
  background: url(logo.svg) no-repeat bottom
right;
  background-color: black;
  background-color: rgba(0,0,0,8);
  background-origin: content-box;
  background-size: 150px;
  color: white;
  font: 100%/1.5 sans-serif;
  text-shadow: 0 -.1em .2em black;
}
h1 {
  margin-top: 0;
}
```

Обрезание фона

Давайте рассмотрим простой вариант предыдущего стиля, как показано ниже^[49].

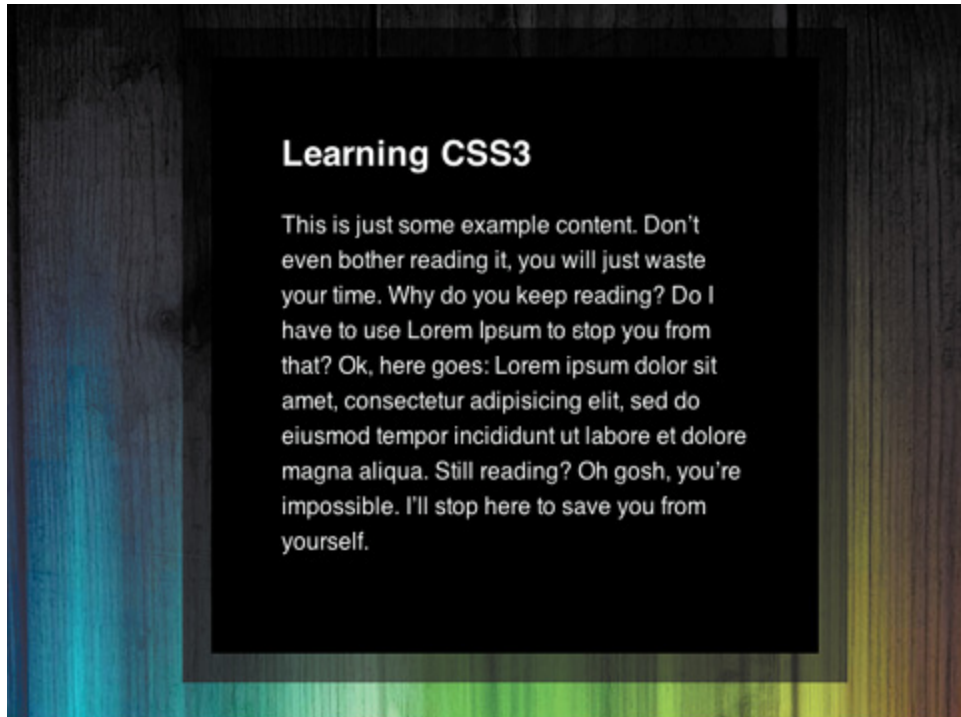


Рисунок 4.11. Полупрозрачная рамка с обрезанием фона

Для получения такого результата вы, вероятно, попробуете что-то подобное:

```
html {  
  min-height: 100 %;  
  background: url('rainbow-wood.jpg') top;  
  background-size: cover;  
}  
section {  
  max-width: 20em;  
  padding: 3em;  
  margin: 4em auto;  
  border: 20px solid rgba(0,0,0,5);  
  background-color: black;  
  color: white;  
  font: 100 %/1.5 sans-serif;  
  text-shadow: 0 - .1em .2em black; }  
h1 {  
  margin-top: 0;
```


}

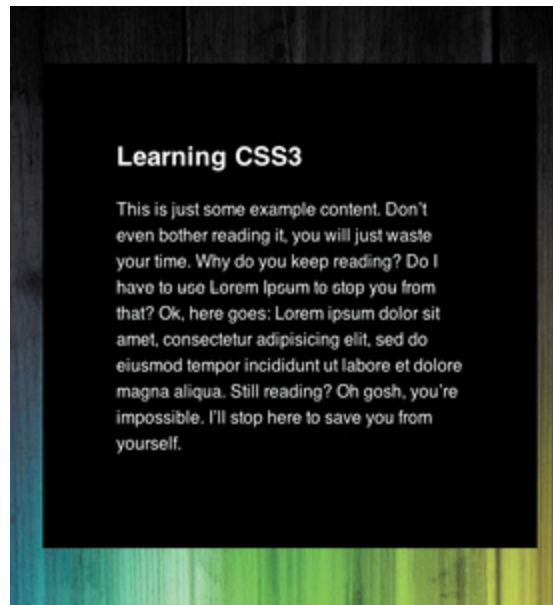


Рисунок 4.12. На самом деле наша граница не полупрозрачная. Почему?

Но когда вы попробуете, то увидите, что граница по факту не полупрозрачная. Почему так произошло? Причина в том, что по умолчанию фон также растягивается по границе. Вспомните все те разы, когда вы применяли стили рамки с пробелами (т. е. точечные, пунктирные и удвоенные) в дни существования CSS2.1? Это то же самое, но только сейчас у нас есть средства контроля поведения, а именно свойство `background-clip`. Его значение по умолчанию – `border-box`, результат которого аналогичен предыдущему опыту. Но мы должны изменить его на `padding-box`:

```
background-clip: padding-box;
```

Это заставит фон обрезаться точно там, где нам надо.

Граница рисунка

Как насчет этого стиля^[50]? Как вы справитесь с зигзагообразной рамкой? Наверное, вы начнете применять установленную ширину и высоту, и огромных размеров фоновое изображение со значением по-

repeat. Если средство отображения установило размер шрифта больше, чем по умолчанию в браузере, или у него нет шрифта, близкого к тому, что вы собираетесь задать, текст растечется над установленным фоновым изображением.

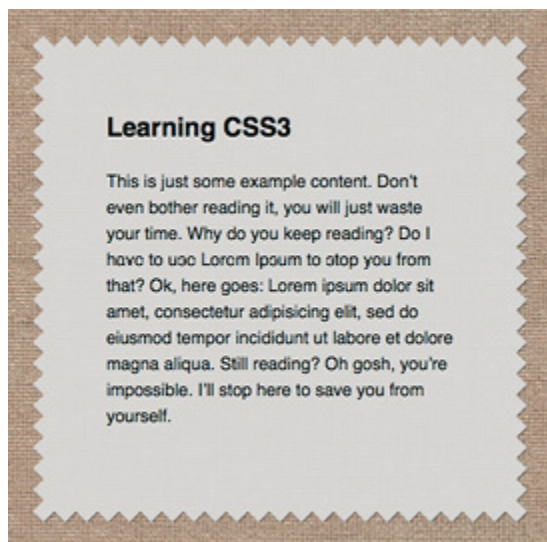


Рисунок 4.13. Пример рамки



Рисунок 4.14. Изображение cloth.svg, с которым мы и будем работать

Свойство `border-image` – новое и мощное, дает нам возможность использовать одно маленькое изображение и для фона, и для рамки элемента. Мы определяем, каков его масштаб и как оно повторяется. Они должно отличаться для каждого края. В этом примере мы использовали рис. 4.14.

А вот код CSS, который идет для него:

```

html {
  min-height:100 %;
  background: white url(background.jpg);
}
section {
  max-width: 20em;
  padding: 3em;
  margin: 4em auto;
  border: 20px solid transparent;
  - webkit-border-image: url(cloth.svg) 33.33 %
round;
  - moz-border-image: url(cloth.png) 33.33 %
round;
  - o-border-image: url(cloth.svg) 33.33 % round;
  border-image: url(cloth.svg) 33.33 % round fill;
  font: 100 %/1.5 sans-serif;
  text-shadow: 0 1px white;
}
h1 {
  margin-top: 0
}

```

Свойство `border-image` определяет, какое изображение будет использовано, какой должна быть ширина каждого «кусочка» (в этом случае мы хотим разрезать по три части, как горизонтально, так и вертикально, это лишь одна цифра) и как эти «кусочки» должны повторяться или растягиваться (ключевое слово `round` повторяет рисунок и масштабирует его так, чтобы на стороне элемента оказалось целое число изображений.). Ключевое слово `fill` сохраняет средний кусочек в виде фона, но оно не поддерживается браузерными префиксами. К счастью, у свойств с браузерными префиксами это поведение есть по умолчанию. Вы также можете заметить, что мы используем SVG (масштабируемую векторную графику) в каждом браузере, кроме Firefox. Дело в том, что в момент написания этого раздела Firefox «глючит» при применении SVG-картинок для рамок. Но Mozilla работает над этим. Хотите, проверьте сами.

Когда мы используем свойство `border-image`, стиль и цвет рамки, который мы определяем, игнорируется до тех пор, пока стиль рамки не устанавливается до значения `none`. Только свойство `border-width` имеет эффект. Но мы хотим задать что-то вроде `solid transparent`, так чтобы когда `border-image` не поддерживается, у нас не было бы уродливо-толстой рамки. Вы можете варьировать расстояние, а в некоторых случаях «реальный» запасной вариант рамки лучше, чем ничего.

Преобразования (трансформации)

Трансформации CSS дают возможность применить одно или несколько преобразований к элементу. Это относится к масштабированию, наклону изображения, вращению и переносу. Большинство этих эффектов не были раньше доступны в HTML вне преобразования контента в рисунок со всеми его недостатками.

Мы создадим простую галерею рисунков, чтобы продемонстрировать применение некоторых трансформаций. Наведение курсора и клик по нему запустит различные трансформации. Можете посмотреть пример, пройдя по ссылке smashed.by/trnsfrms.

Применение простой трансформации

В примере каждый рисунок включен в элемент `figure`, как здесь:

```
<figure>

</figure>
<! - повторяющийся тег figure для каждого
изображения->
```

Мы включили все изображения в их полном размере, уменьшив их масштаб, а не использовали отдельные превью. Это можно сделать с преобразованием масштаба:

```
figure {
float: left;
```

```
z-index: 1;  
margin: 1rem;  
width: 160px;  
height: 160px;  
transition-duration: 1s;  
transform: scale(0.25);  
}
```

Свойство `transform` допускает в качестве значения разделенный пробелами список функций трансформирования. Здесь мы используем функцию `scale` для уменьшения размера элемента вчетверо. Масштабирование пойдет от центра элемента по умолчанию. Значения меньше 1 уменьшат масштаб элемента, а значения выше этой цифры – увеличат.

Помните, что трансформации не влияют на поток документа, поэтому каждый элемент займет то же пространство, что и раньше (до трансформации). По этой причине мы также установили свойства ширины и высоты как четверть оригинальных размеров, поэтому у нас не много свободного пространства вокруг масштабированных изображений.

После применения дополнительных стилей к `body` и самим изображениям, мы получим что-то наподобие следующего:



Рисунок 4.15. Простая галерея изображений без трансформаций

Применение дополнительных трансформаций

Трансформации могут стать намного интереснее, если мы свяжем их в цепочку и объединим с перемещениями. Мы уже задали наше перемещение в предыдущем фрагменте кода, поэтому нам всего лишь нужно сделать преобразование по наведению курсора для достижения эффекта:

```
figure: hover {  
  transform: scale(0.33) rotate(2deg);  
  z-index: 100;  
  cursor: pointer;  
}
```

Здесь мы применяем два преобразования. Любое количество трансформаций может применяться при задании их через список, разделенный пробелами. Здесь мы уменьшаем масштаб рисунка в 3 раза, затем поворачиваем его на 2 градуса. Функция rotate перемещает элемент вокруг оси по часовой стрелке. Чтобы перемещение было против часовой стрелки, мы используем отрицательную величину:

```
figure: nth-of-type(even):hover {  
  transform: scale(0.33) rotate(-2deg);  
}
```



Рисунок 4.16. Преобразование, применяемое для изображения по наведению курсора с использованием CSS3

Каждая функция `transform` применяется по очереди, поэтому элемент сначала масштабируется, а потом вращается. Помните, что это важно! Порядок влияет на размер элементов при изменении масштаба, и на направление каждой оси при повороте.

Так как все функции `transform` применяются с использованием одного свойства, а вы хотите просто переместить одну величину, вам нужно будет составить полный перечень. А иначе остальные величины примут свои значения по умолчанию. При наведении курсора на изображение в нашем примере, оно будет масштабироваться и поворачиваться. Вы можете заметить, что изображение, которое находится после того, на которое наведен курсор, также перемещается и трансформируется после короткой паузы. Здесь мы добавили преобразование `translateX`, которое смещает элемент вдоль оси X.

```
figure: hover + figure {
  transform:      scale(0.25)      rotate(-1deg)
  translateX(15px);
  transition-duration: 1.5s;
```

```
transition-delay: .2s;  
}
```

Вы видите, что нет строгого движения вдоль оси X или преобразования на 15 пикселей, потому что преобразование scale уменьшает длину на четверть, а преобразование rotate поворачивает ось X на 1 градус против часовой стрелки.

Функция `translateX` принимает одно значение, используя действующую длину элемента. Также существует соответствующая функция `translateY`. Они обе могут устанавливаться вместе с использованием функции `translate`, которая допускает две величины (сначала X, затем Y), которые разделяются запятой.

Последнее семейство функций `transform` – это `skewX`, `skewY` и `skew`. Они определяются так же, как и функции `transform`, но они перекручивают элемент по одной или двум осям. Обычно это происходит с использованием моделируемой трехмерной перспективы.

Настройка базы трансформаций

Все трансформации в примере используют исходную точку по умолчанию, которая является центром элемента. Использование свойства `transform-origin` поможет установить ее. Оно допускает от одной до четырех величин, заданных через пробел. Это может быть длина, процент, или ключевые слова «сверху, слева, снизу, справа или в центре». Если установлена только одна величина, то вторая по умолчанию будет в центре. Если одна из величин не является ключевым словом, то тогда первая будет отвечать за горизонтальную позицию. При использовании ключевого слова, вы можете определить дополнительную величину смещения, которая устанавливается как процент или длина сразу после этого:

```
transform-origin: top 10 % left 25 %;
```

Так база преобразований установится на точке, которая находится на пересечении 10 % от верха элемента и 25 % слева от него.

Селекторы

Разработчики считают селекторы CSS часто чуть ли не самой прикольной игровой площадкой, на которой можно развернуться. Вы можете подумать, что нам на самом деле не нужны дополнительные селекторы для обращения к элементу в разметке. Или, может быть, вы из последних сил воюете с ненавистными искусственными приемами jQuery, чтобы перезаписать значения CSS по умолчанию в определенных ситуациях. И в том, и в другом случаях с селекторами CSS3 вы получите удовольствие. Давайте посмотрим, какие возможности у нас есть сейчас, когда селекторы CSS3 набирают поддержку в браузерах.

Подсвечивание текущего элемента

Давайте вспомним пример с вертикальным ритмом из раздела о типографике. Вы, возможно, в курсе, что можете ссылаться на определенные разделы страницы, используя идентификатор с решеткой после указателя URL. Так мы можем использовать что-то вроде этого `http://example.com/index.html#def` для ссылки на раздел с определениями типографики. Когда на странице несколько больших разделов, пользователю сразу же видно, в какой раздел он попал.

Однако когда существует много возможных целевых элементов, пользователь может растеряться и не знать, куда смотреть. В этих случаях было бы полезно высвечивать активный элемент на экране.

В прошлом для выполнения этой операции нам бы понадобился язык JavaScript. CSS3 дает нам новый псевдокласс для указания текущего элемента, т. е. элемент, чей ID-атрибут совпадает с текущим хешем в URL. Этот псевдокласс называется: `target`.

Давайте подсветим каждый заголовок в нашем примере полупрозрачным желтым, чтобы пользователь мог четко понимать, куда он попал на странице. Для этого мы можем использовать следующий код:

```
h1:target, h2:target,  
h3:target, h4:target,  
h5:target, h6:target {  
background: hsla(65,100 %,50 %,5);  
}
```

Указание цвета в профиле HSLA не нужно в данном случае, потому что в общем любой браузер, который поддерживает псевдокласс: `target` также поддерживает цвета HSLA. Старые браузеры, которые не поддерживают: `target` на самом деле не проблема, так как селектор улучшает юзабилити и его функциональность не ключевая. Википедия использует: `target`, когда вы щелкаете на сноску, подсвечивая ее в (обычно длинном) списке ссылок^[51]. Это реально помогает вам быстро опознать правильную ссылку, так ведь?

Указание на элементы, основанные на их позиции в DOM (объектной модели документа)

Все это уже известно. Иногда мы хотим обратиться к нечетной строке таблицы или каждому третьему изображению на странице, или к последнему элементу в списке, или первым четырем параграфам в статье.

В CSS 2.1 у нас был только один структурный псевдокласс: `first-child`. CSS3 расширяет это и дает нам изобилие новых псевдоклассов, что решает не только перечисленные задачи, но и другие вопросы:

- `nth-child`
- `last-child`
- `nth-last-child`
- `only-child`
- `first-of-type`
- `nth-of-type`
- `last-of-type`
- `nth-last-of-type`
- `only-of-type`

Количество псевдоклассов в этом списке может испугать, но как только вы поймете возможности: `nth-child` и как это работает, вы легко уловите все остальные и узнаете, как это применять. Потому что это всего лишь варианты или ярлыки. Те, что начинаются с `nth`, представляют концепцию, которой не было в CSS 2.1: псевдоклассы с параметрами. Подобно функциям, они задают параметр, который дифференцирует их поведение, в круглых скобках. Синтаксис этого параметра может быть следующим:

- Число в: `nth-child(1)` равно псевдоклассу: `first-child` в CSS 2.1.

- Чтобы выразить: `nth-child(5)` в CSS 2.1, вам нужно было бы написать: `first-child + * + * + *`, что недопустимо многословно, особенно для больших цифр.

- Выражение типа $5n$ или $3n+2$, где n равно любому числу от 0 до бесконечности. Например, `nth-child(3n+1)` равно: `nth-child(1)`, `nth-child(4)`, `nth-child(7)`, `nth-child(10)` и т. д., с бесконечным списком.

- Одно из ключевых слов нечетное или четное, $2n+1$ и $2n$, соответственно.

Например, чтобы затемнить фон каждой нечетной строки таблицы, мы могли бы написать что-то наподобие:

```
tr: nth-child(odd) {  
  background: rgba(0,0,0,15);  
}
```

Это, по существу, эффект полосок зебры, для которого мы обычно используем JavaScript.

Пожалуйста, обратите внимание на то, что разница между: `nth-*` и: `nth-last-*` – это чисто направление нумерации: `nth-child` начинается от первого «родителя», тогда как: `nth-last-child` стартует от последнего. Отсюда: `:last-child` по существу равен: `nth-last-child(1)`, а `:only-child` равен `:first-child: last-child`, потому что он подходит к элементам без «братьев и сестер». Интересно то, что мы можем обобщить псевдокласс: `only-child` так, что когда нам нужно обратиться к элементу ровно с пятью дочерними элементами, мы можем использовать: `first-child: nth-last-child(6)`, чтобы обратиться к первому, а затем применить: `first-child: nthlast-child(6) ~ *` для остальных.

Разница между псевдоклассами `*-child` и `*-of-type` в том, что последний поддерживает отдельный счет на тег. Например, `body: first-child` никогда не совпадет, потому что `body` всегда имеет

заголовочный дочерний элемент, а `body: first-of-type` всегда совпадает, так как у нас всего один элемент `body`.

Это может быть не особо важно (не очень значимо) для позиционирования `body`, но чрезвычайно полезно, если мы хотим обратиться, предположим, к каждому третьему изображению в разметке, у которой переменное количество параграфов между изображениям. В этом случае псевдокласс: `nth-child` рендерился бы непоследовательно, потому что он действует со всеми дочерними элементами, независимо от их типа.

Что насчет старых браузеров?

Обычно функциональные возможности, которые добавляются этими селекторами, не ключевые, так что страница обходится прекрасно и без них. Но если вам позарез нужна поддержка устаревших браузеров, тут может помочь полифил.

Наиболее популярный на данный момент – это Selectivizr^[52].

Сочетаем элементы без проблем

Итак, давайте снова вернемся к примеру в разделе о работе с изображениями^[53]. Предположим, нам сейчас надо заменить статической текст на веб-форму с текстом, который находится в элементе `textarea`, что позволяет людям редактировать его. Мы даем нашему элементу `textarea` внутренние поля в `1 em`, 1-пиксельную рамку и ширину `100 %`, потому что мы хотели чтобы он занял всю ширину контейнера. Вы, вероятно, видите, к чему это ведет – к Старой Ужасной Проблеме с Процентами в CSS™^[54].

Поля и рамки добавляются к тем `100 %`, которые делают весь прямоугольник (блок) намного больше, чем `100 %`, и выглядят не так «навороченно». В былые времена нам было бы нужно устанавливать отступы и рамки тоже в процентах, и определять ширину в `100 %` минус поле, минус ширина рамки. К счастью, в CSS3 мы властны изменять способ, которым рассчитывается ширина, и делать это так, как естественно для нас – включать в ширину поле и рамку. Свойство `box-sizing` отвечает за это изумительное изменение. Оно принимает три значения:

- **content-box**

значение по умолчанию, которое мы уже знаем и не любим;

- **padding-box**

с ним отступ включается в ширину, а рамка нет. Имеет не очень хорошую браузерную поддержку, поэтому пока избегайте его;

- **border-box**

и отступ, и рамки включаются в ширину.

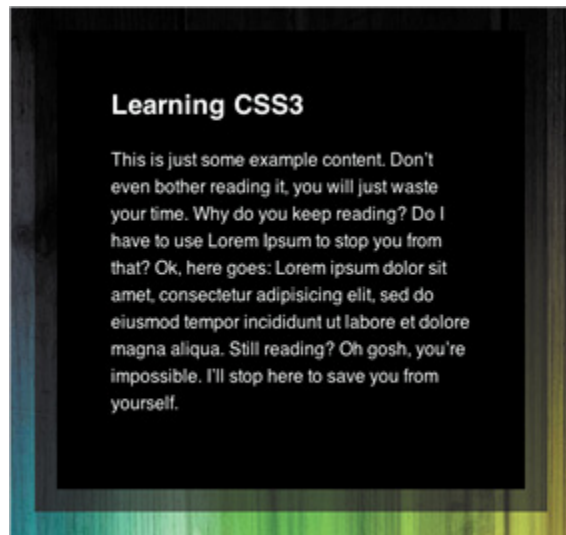


Рисунок 4.17. Наша отправная точка: стиль из предыдущего раздела

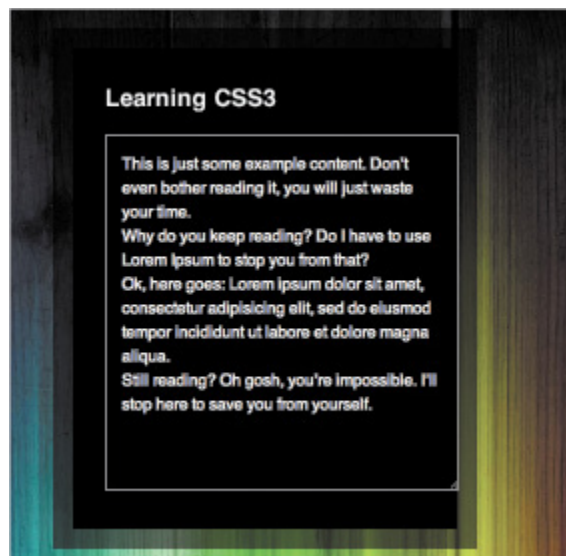


Рисунок 4.18. Старая Ужасная Проблема с Процентами в CSS™

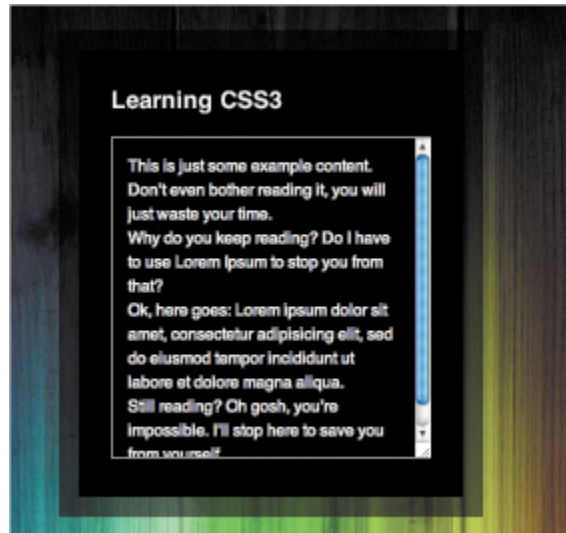


Рисунок 4.19. Смешивание процентов с пикселями и ems с использованием box-sizing

С применением `box-sizing: border-box` наша проблема теперь решена^[55]. И это свойство поддерживается не только каждым современным браузером, но также и IE 8!

Перемещения

До недавнего времени, сайты, которые разрабатывались по веб-стандартам были явно статическими. Если вы хотели добавить какую-нибудь анимацию между элементами на странице, вам нужно было делать это или с помощью Flash, или с комплексным JavaScript. Теперь, когда вы полностью вольны разрабатывать сайты для современных браузеров, вы можете достичь подобных эффектов, применяя CSS. Они дают дополнительные преимущества на мобильных устройствах, благодаря тому что режим анимации оптимизируется в браузере.

Перемещения представляют собой половину анимационных возможностей CSS. Они разработаны для простых анимаций между одним и другим положением элемента. В традиционном варианте, если вы меняете значение свойства в CSS, оно тотчас переключается между старым и новым состоянием. Теперь, с перемещениями CSS, браузер производит переход от старого значения к новому в течение определенного времени.

Использование перемещений

Для того чтобы продемонстрировать, как использовать перемещения, мы собираемся провести день на скачках. Помните тот автомат со скачущими лошадками, когда вы были маленькими? Если нет, не расстраивайтесь. Идея проста: лошадки скачут по дорожке с разными скоростями, а вы должны угадать какая из них придет первая. Эта игра переделана при помощи перемещений CSS. Вы можете поиграть дома, пройдя по ссылке smashed.by/trnsxpl. Просто наведите курсор на дорожки и смотрите, как скачут лошадки!



Рисунок 4.20. Игра со скачущими лошадками, созданная при помощи перемещений CSS

Разметка для дорожки такова:

```
<div id="track">
<h1>The<em>Smashing</em> Derby</h1>
<ol>
<li><div></div></li>
<! - добавленные лошадки ->
```

```
</ol>  
</div>
```

Каждый тег `li` представляет беговую дорожку, а тег `div` внутри содержит лошадь. Затем мы перемещаем ширину `div` по наведению курсора и задаем время:

```
#track div {  
  width: 3em;  
  height: 3em;  
  background: url(horse.png) no-repeat right  
center;  
  transition-property: width;  
  transition-duration: 6s;  
}  
#track: hover div {  
  width: 40em;  
}
```

Здесь мы говорим, что свойство `width` должно перемещаться около 6 секунд от 3 ems до 40 ems, когда на дорожку наведен курсор. Свойство `transition-property` настраивается по умолчанию ко всему, поэтому если вы не установите его точно, оно будет перемещать каждое анимируемое свойство.

Все свойства перемещения допускают разделяемый запятой список значений, поэтому вы можете устанавливать множественные свойства к перемещению. Если число величин в свойствах, таких как длительность перемещения, меньше, чем число свойств к перемещению, список будет ставить их в соответствие подобно тому, как было со свойством фона.

Настройка скорости передвижения лошадей

Скачки не были бы такими забавными, если бы лошади все время неслись с одной скоростью и приходили к финишу одновременно. Что ж, давайте настроим скорость каждой лошадки с помощью свойства `transition-timing-function`. У всех видов функций расчета времени

одинаковая продолжительность до завершения, но их ускорение и замедление происходит на разном уровне, в зависимости от установленной кривой Безье. Не волнуйтесь, если это звучит слишком по-математически: вы можете выбирать из набора встроенных функций расчета времени. Эти предустановки имеют следующие значения ключевых слов:

- **ease**
это значение по умолчанию;
- **linear**
перемещения с постоянной скоростью от А до В;
- **ease-in**
перемещения в медленном темпе, которые ускоряются при приближении к точке В;
- **ease-out**
перемещения в быстром темпе, которые замедляются при приближении к точке В;
- **ease-in-out**
перемещения быстрые до точки на полпути, затем замедляются при приближении к точке В.

Каждая из этих функций применяется именно в таком порядке для лошадей из примера:

```
li: nth-of-type(1) div { transition-timing-  
function: ease; }  
li: nth-of-type(2) div {  
transition-timing-function: linear;  
}  
li: nth-of-type(3) div {  
transition-timing-function: ease-in;  
}  
li: nth-of-type(4) div {  
transition-timing-function: ease-out;  
} li: nth-of-type(5) div {  
transition-timing-function: ease-in-out;  
}
```

Глядя на рисунок можно увидеть, что лошадь 1 (ease) вырывается в лидеры, это – явный победитель. О, нет! Она выбивается из сил на трети до финала. И все пять лошадей пересекают финиш одновременно!

Если вы недовольны предустановками, то можете установить свои собственные, используя кубическую функцию Безье. Для определения кубической функции, вам нужно задать координаты x и y для контрольных точек кривой. Начало всегда фиксируется на 0,0, а конец – на 1,1, поэтому их не нужно определять. Функцию перемещения ease-in можно определить, как показано ниже, с применением кубической функции:

```
transition-timing-function:    cubic-bezier(0.42,  
0, 1, 1);
```

Некоторое количество инструментов, таких как Lea Verou's Cubic Bezier preview tool^[56], позволят вам визуальным образом задать и «довести до ума» функцию cubic-bezier.

Придержите своих лошадей

Возможно, вы не захотите, чтобы перемещение произошло, как только изменятся значения. Вы можете контролировать это с помощью свойства transition-delay. Оно действует точно по такому же принципу, как и свойство transition-duration:

```
#track div {  
...  
transition-property: width;  
transition-duration: 6s;  
transition-delay: 1s;  
}
```

Со всеми этими свойствами за плечами ваши элементы «пролетят» по странице за два счета!

Заключение

Пока дизайнеры и разработчики продолжают выдвигать креативные технические решения, ясно одно: CSS3 будет использоваться дальше. Их применение не только значительно сокращает время на преобразование сайта от визуализации до кода, но также помогает создавать гибкий дизайн, который адаптируется к любой среде: изменениям кода, разному контенту и разным видам устройств.

Пока браузеры «снимают сливки» с CSS3, рабочая группа по CSS уже разрабатывает следующую версию, обычно упоминаемую как CSS4. Ожидается, что это привнесет свойства, которые уже жарко обсуждаются, такие как селекторы родителей, переменные, многоколоночная верстка макета страницы и конусные градиенты.

Восторгаясь от того, что CSS3 и CSS4, возможно, звучат как модные словечки, помните, что в целом живым стандартом является CSS. Каждый член рабочей группы CSS может утверждать, что в веб-стандартах нет таких понятий, как “CSS3” или “CSS4”. На деле, нет больше глобальной версии. После CSS 2.1, CSS была модулирована, и каждый модуль теперь имеет свою версию. И некоторые модули Уровня 1 фактически могли выходить позже, чем модули Уровня 4. Но нам не нужны модные словечки, чтобы приходить в восторг от всего, что приходит в мир CSS, так ведь?

Об авторах



* * *

Дэвид Стори имеет степень магистра в области Интернета и Распределенных систем университета Дарема, Великобритания. Член сообщества рабочей группы CSS, он является ярким сторонником открытых веб-стандартов. В настоящее время Дэвид живет в Маунтин Вью, штат Калифорния и работает в компании Motorola Mobility. До

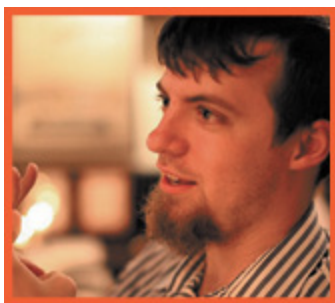
этого он работал в Opera, где создал команду разработчиков Developer Relations team и был бренд-менеджером продукта Opera Dragonfly. Так же он работал в ЦЕРН (CERN) – европейском центре ядерных исследований. Являлся автором тестов CSS3.info во времена его золотого периода. Он специализируется на HTML, CSS, SVG и JavaScript.



* * *

Ли Веру испытывает давнюю страсть к открытым веб-стандартам. Ее часто называют «гуру CSS». Ей нравится исследовать новые пути по извлечению преимуществ из новых веб-технологий и делиться своими изысканиями в своем блоге lea.verou.me. Ли также создает популярные инструменты и библиотеки, которые помогают веб-разработчикам изучать и применять эти стандарты. Она выступает на нескольких широко известных интернациональных конференциях по разработке сайтов и пишет публикации для ведущих изданий. Еще Ли является соавтором и лектором на курсах по веб-разработке в Афинском университете экономики и бизнеса.

О рецензенте



Тэб Аткинс, мл. Специалист во многих областях. Он работает в Google над разработкой браузера Chrome в качестве хакера веб-стандартов, хотя его вклад в код достаточно скромнен. Тэб в основном работает со спецификациями HTML и CSS. Также он является членом сообщества рабочей группы CSS и сотрудничает с несколькими другими рабочими группами в W3C.

Повторное открытие JavaScript: фишки и приемы для замены сложного jQuery

Автор: Кристиан Хейлманн

Рецензент: Пол Айриш

Когда появилась библиотека jQuery, это было абсолютным открытием. Ее первостепенной и главной задачей было обеспечивать одинаковое поведение браузеров. До этого поддержка основных функций, таких как доступ к частям документа, реагирование на взаимодействие системы с пользователем и даже стиль оформления элементов, очень отличалась в браузерах.

jQuery заменила спецификацию DOM, которая определяла доступный контент на странице со свойствами `getElementById()` и `getElementsByName()`, более простым методом: используя CSS-селекторы. Перед дизайнерами открылся совершенно новый мир разработки. Ведь они знали CSS, но страдали от ежедневных срывов браузеров, которые не поддерживали сложные селекторы. Другими словами, jQuery позволила нам использовать CSS завтрашнего дня уже сегодня. Упомянутый выше метод сцепления jQuery (это значит, что кодов будет создаваться намного меньше) позволил быстро достичь успеха.

Очень скоро, примерно через несколько лет (и к тому времени, как вы будете читать эту книгу), мы продвинемся еще дальше. У нас есть HTML5, мы владеем поддержкой CSS3, у нас очень много всего, с чем можно развлечься в браузерах, которые установили мы и наши пользователи. Да, IE 6 все еще наш бич, и IE 8 будет с нами еще какое-то время. Но в общем и целом наше положение не такое уж и плохое. Такие библиотеки, как jQuery, дают основные преимущества при исправлении недочетов в старых браузерах, но они также вызывают и некоторую неудовлетворенность. А причина в том, что мы злоупотребляем ими.

Как сообщество, мы стали зависимы от jQuery. Это и понятно, но далеко не хорошо. Библиотека jQuery написана на JavaScript, но это не одно и то же; она исходно не встроена в браузер. С расцветом мобильного Интернета многие отворачиваются от jQuery, потому что

она слишком «тормозит» и тяжела для наших модных карманных устройств. Найти хороших разработчиков JavaScript сложно: на каждое объявление о вакансии на должность разработчика JavaScript Вы получите около двадцати резюме от людей, которые в жизни не писали на чем-то, кроме jQuery. Это «обескровливает» наше ремесло.

Давайте посмотрим, что предлагают нам браузеры сегодня, и что мы можем применить для написания невероятно маленьких и удобных решений, не прибегая к jQuery. Многие из этих вещей также помогут нам написать более чистый и быстрый jQuery-код. Из-за того что библиотека jQuery абстрагируется от многих проблем, с которыми мы сталкиваемся как разработчики, чересчур легко написать код, который выглядит просто, но погружает результаты в пучину циклов и сравнений. А это – одна из причин того, что сайты «тормозят».

Сила смешивания и сочетания

В Интернете основной хитростью при разработке кода, который будет кратким, неизменяемым, эффективным и легким для поддержания, является разделение и делегирование. С jQuery мы забыли многое из этого. Чрезмерная длина ломает CSS селекторы в наших скриптах, когда код HTML изменяется.

Злоупотребление селекторами классов вынуждает разработчиков HTML добавлять много ненужных классов для того, чтобы использовать определенный плагин. При создании и задании стилей HTML через jQuery сложно найти, откуда берется цвет фона, когда вас просят изменить его. Чтобы написать код, который будет легко поддерживать, вспомните, какие технологии хороши для этого. С момента открытия jQuery они несколько изменились:

- **HTML – это структура и база для построения**

Ваш HTML должен иметь смысл и обеспечивать базовую функциональность: ссылки на другие сайты, кнопки, которые отсылают формы к скрипту для перезагрузки страницы, и элементы, создающие контент. Дело в том, что когда все «слетает», браузер остается только с HTML. Если HTML несет смысл, вы – победитель. Но если есть кнопка, которая ничего не делает, вы будете раздражать пользователей.

- **CSS определяет интерактивность и визуальность.**

Мы шагнули за пределы шрифтов и цвета, к анимации и перемещениям. Медиазапросы позволяют нам делать всевозможные макеты для разных устройств. Используя генерирование контента, мы можем создавать элементы для достижения определенного визуального эффекта без «загрязнения» тегами `div` и `span`.

- **JavaScript привносит дополнительные функции.**

Со скриптом загрузки и AJAX (асинхронный JavaScript и XML), вы можете загружать контент по требованию. Можете добавлять обработчики событий, чтобы элементы реагировали на нажатие пальцев и клик мыши, чтобы они могли считывать ориентацию устройства, и устанавливать, сколько информации прокрутил пользователь или где находится курсор мыши.

Хитрость в том, чтобы объять все эти новые возможности и не позволить старым браузерам «задушить» их. В конце концов, старые приемы никому не помогут. Да, вы могли бы анимировать меню в IE 6, но к чему напрягаться и писать эту функцию, которую «не мешало бы иметь», когда она встроена в другие браузеры?

Функции, доступные нам сейчас

Давайте посмотрим на некоторые функции браузеров, которые мы можем использовать сейчас. Чтобы получить последнюю информацию о том, какие браузеры их поддерживают, обратитесь к великолепному источнику [When Can I Use^{\[57\]}](#), который постоянно обновляется. Кое-что из того, что я рассказываю вам о браузерной поддержке сейчас, через несколько недель может устареть. Это темп нашей жизни и мы должны поддерживать его.

QUERYSELECTOR и QUERYSELECTORALL

Теперь, когда мы узнали об успехе jQuery, у браузеров есть способ обращения к элементам на странице с помощью селекторов CSS. Метод `querySelector` обращается к одному элементу, а `querySelectorAll` – к списку соответствующих элементов. Синтаксис селектора подобен CSS. Поэтому, `document.querySelector('#content p')` обратится к первому тегу параграфа в элементе с ID `content`;

`document.querySelector('nav li: last-child')` обратится к последней позиции списка в первом элементе `nav`; а `document.querySelectorAll('p')` обратится ко всем параграфам документа. Все просто.

Работаем с классами: CLASSLIST

Масштабный вариант использования jQuery – обратиться сразу ко многим элементам, и изменить стили по методу `css()`. Это удобно, но в тоже время, раздражает, потому что вы вставляете информацию о стиле в JavaScript. Гораздо проще было бы добавлять класс к рассматриваемому элементу, а остальное предоставить CSS. Из-за этого мы часто повторяли селекторы CSS в jQuery и наших таблицах стилей. Мы вынуждены были делать это, потому что браузеры не поддерживали селекторы CSS3, но сейчас-то поддерживают!

Невероятную мощь дает возможность тестировать классы в элементах HTML и динамично добавлять и удалять их. В JavaScript мы сейчас имеем свойство `classList` в HTML элементах, которое содержит коллекцию применяемых CSS классов. Раньше это делалось через `className`, который состоял из простой символьной строки и от нас зависело, найдем ли мы другие строки в ней, или добавим и удалим подстроки из нее. С помощью `classList` у нас для этого есть методы.

Мы можем использовать `element.classList.add(name)` для добавления класса, `element.classList.remove(name)` – для его удаления, `element.classList.contains(name)` для проверки, применяется ли класс, и `element.classList.toggle(name)` для включения и выключения класса. Позже в этом разделе мы увидим, как это действенно. Мы сможем избежать введения многих циклов, просто добавляя класс к элементу родителя.

Меняем плавно: CSS TRANSITIONS

Анимация в jQuery в самом деле легкая и смотрится очень ровно. Это из-за того, что jQuery включены `easing equations`^[58], и сегодня они есть также в CSS. Поэтому, если вы хотите растянуть заголовок и

поменять цвет его фона со светло-зеленого на оранжевый, вы можете использовать фрагмент со следующей страницы^[59].

```
h1 {
  background: #c0ffee;
  line-height: 1em;
  padding: 0.5em 1em;
  - webkit-transition: 1s;
  - moz-transition: 1s;
  - ms-transition: 1s;
  - o-transition: 1s;
  transition: 1s;
}
h1:hover, h1:focus {
  background: goldenrod;
  line-height: 3em;
}
```

Лучшее, что здесь есть, это то, что при установке времени перемещения, а не поиске свойств, которые надо изменить, мы можем задать браузерам плавный переход из одного состояния в другое, не вникая в подробности, как дальше будут вноситься правки.

В этом примере мы меняем цвет фона и высоту межстрочного интервала, но могли бы легко сделать это и позже. С jQuery это означало бы переписать JavaScript. Раздражать нас может то, что приходится повторять информацию о перемещении для всех префиксов браузера. Сейчас мы просто должны смириться с этим.

Другой плюс в том, что эти перемещения поддерживаются на аппаратном уровне. А это значит, что они могут происходить более плавно и тратить меньше зарядки в мобильных устройствах. Пока еще не все браузеры способны на это (некоторым нужна 3D-трансформация от 0 на оси Z в виде хака). Но, несомненно, со временем эта функция станет стандартом. Более подробно о перемещениях CSS3 вы можете прочесть в четвертом разделе этой книги.

Контент, сгенерированный посредством CSS

Порой разработчикам нужно больше HTML элементов для создания стиля (возьмем, как пример из прошлого, закругленные углы). В большинстве случаев мы используем для этого jQuery. С генератором контента CSS нам это больше не нужно. Мы можем создавать элементы в CSS и одновременно придавать им стиль. Допустим, вы хотите, чтобы все внешние ссылки имели красную стрелку в конце:

```
a[href^="http"]: after {  
  content: '  
                                ↗  
';  
  color: #c00;  
}
```

Этот простой CSS код имеет свою «изюминку». Мы определяем (устанавливаем), что для каждой ссылки с атрибутом href, которая начинается с http, текстовый узел – в данном случае это красная стрелка – нужно добавлять в содержимое ссылки.

Делегирование событий: от многих до одного

jQuery дает вам одну большую возможность. Вы можете быстро перебирать элементы, чтобы изменять что-нибудь в них или устанавливать обработчики событий. Но на самом деле это и не нужно. Делегирование событий^[60] – необычайно мощный инструмент для создания веб-интерфейсов. По существу, вместо того, чтобы устанавливать обработчики событий на каждый элемент внутри главного элемента контента, вы назначаете один обработчик для основного содержимого. А это уже позволит браузерам упорядочить события.

Здесь есть несколько плюсов. Для начала вы покончите с установкой множества мелких обработчиков в документе, что всегда полезно для экономии памяти. Что еще интереснее, вы сделаете обработку событий независимой от количества используемых

элементов. Например, если вы добавляете в ваш список дел новые пункты, вам вообще не нужно будет переназначать обработчики.

jQuery заимствовала этот принцип, когда добавила «живого» обработчика событий. Однако многие решения jQuery будут добавлять «живых» обработчиков к ID селекторам без детей. А по определению ID может появиться только один раз в документе и, поэтому, он не нуждается в делегировании.

Техники в примерах

Давайте применим эти техники в нескольких примерах. Начнем со списка дел, в котором используются делегирование событий и сгенерированный контент, потом перейдем к сайту-визитке, который использует перемещения, а в конце начнем применять HTML5 canvas для создания миниатюр в браузере.

Пример 1: Простой список дел

[\[61\]](#)

Чтобы создать список дел для всех браузеров, нам понадобится серверное решение для введения позиций списка, сохранения их в базе данных и отображения их в браузере. Мы не станем делать это здесь. Вместо этого мы просто используем решение на стороне клиента (в браузере), включающее хранение данных. Но с реальным продуктом вам потребуется резервный сервер.

Используя современные браузерные технологии, мы сможем сделать это в несколько строк кода, без каких-либо элементов цикла. Разметка в HTML достаточно проста:

```
<ul id="todolist"></ul>
<form action="#" method="post">
<div>
<label for="newitem">Add item</label>
<input type="text" name="newitem" id="newitem"
placeholder="new item">
<input type="submit" value="Add">
</div>
```

</form>

В коде JavaScript также нет ничего особенного:

```
var todo = document.querySelector('#todolist'),
    form = document.querySelector('form'),
    field = document.querySelector('#newitem');
form.addEventListener('submit', function(ev) {
    var text = field.value;
    if (text !== '') {
        todo.innerHTML += '<li>' + text + '</li>';
        field.value = '';
        field.focus();
    }
    ev.preventDefault();
}, false);
todo.addEventListener('click', function(ev) {
    var t = ev.target;
    if (t.tagName === 'LI') {
        t.parentNode.removeChild(t);
    };
    ev.preventDefault();
}, false);
```

Код, представленный выше, мы начинаем с ввода элементов в документ, который мы хотим создать, применяя `querySelector`. В данном случае мы будем вводить список, который добавит желаемые элементы, форму, в которой рождаются новые элементы, и поле, в которое вводится новая запись.

Потом мы добавим слушателя событий к форме, которая считывает значение поля и проверяет, было ли оно заполнено при отправке формы. (Это означает, что пользователь может добавлять новые позиции нажатием клавиши «Enter», помимо непосредственного добавления с помощью мыши. К сожалению, чересчур много jQuery решений используют вместо этого обработчик нажатия кнопки.) Если есть контент, мы добавляем новый элемент в список с помощью

innerHTML. После этого мы удаляем текущий текст из поля ввода и ставим на него курсор (чтобы было легко добавлять другие элементы).

Чтобы пользователь мог удалять из списка завершённые дела, мы добавляем в него обработчик нажатия, считываем цель события и сравниваем его tagName с элементом li. Удаляем мы это с помощью старого метода DOM removeChild(). Это то, что нам нужно сделать для создания списка дел с неограниченным количеством элементов, используя делегирование событий. Ни больше ни меньше.

Продвинутые CSS селекторы и сгенерированный контент для задания стиля

Теперь мы хотим задать альтернативные цвета элементам списка. Ещё мы хотим, чтобы при наведении курсора появлялся чекбокс (галочка), указывающая на то, что по клику список будет помечен как выполненный. Чтобы это сделать, нам не нужны ни JavaScript, ни изображения:

```
#todolist li {  
  background: #eee;  
  min-height: 20px;  
  position: relative;  
}  
#todolist li: nth-child(2n) {  
  background: #ccc;  
}  
#todolist li: hover: after {  
  content: '✓'  
  ;  
  color: #060;  
  position: absolute;  
  right: 5px;  
}
```

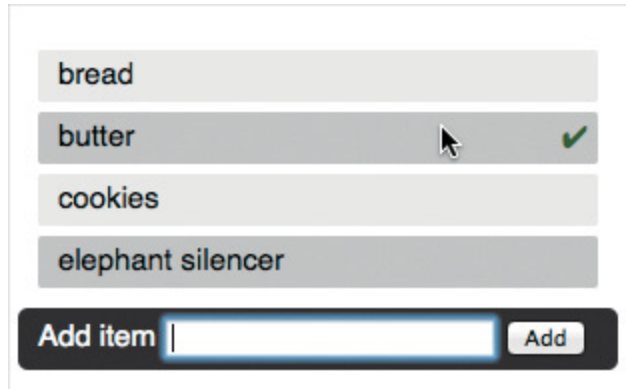


Рисунок 5.1. Наш список дел с «галочками»

Селектор `nth-child(2n)` указывает браузеру, что нужно окрасить каждую вторую строку в темно-серый цвет, а остальные оттенить светло-серым. Чтобы показать «галочку», когда пользователь проводит курсором над элементами, мы используем селектор: `after` и создаем «галочку» через таблицу символов UTF-8. Так как каждый элемент списка относительно позиционирован, любой абсолютно позиционированный «впадает» в него. Таким образом, правильное значение покажет зеленую «галочку» внутри блока, когда пользователь наведет курсор над элементом списка.

Делаем удаление элемента списка в два этапа

Что делать, если мы хотим, чтобы элементы в нашем перечне не только отмечались как завершенные, но и удалялись при втором клике? Все просто: мы всего лишь добавим другой режим и применим классы. Когда пользователь щелкает на элемент в первый раз, добавляется класс `done`, а когда второй раз, элемент удаляется. Все, что нам нужно сделать, – поменять обработчика событий:

```
todo.addEventListener('click', function(ev) {
  var t = ev.target;
  if (t.tagName === 'LI') {
    if (t.classList.contains('done')) {
      t.parentNode.removeChild(t);
    } else {
      t.classList.add('done');
    }
  }
});
```

```

}
};
ev.preventDefault();
}, false);

```

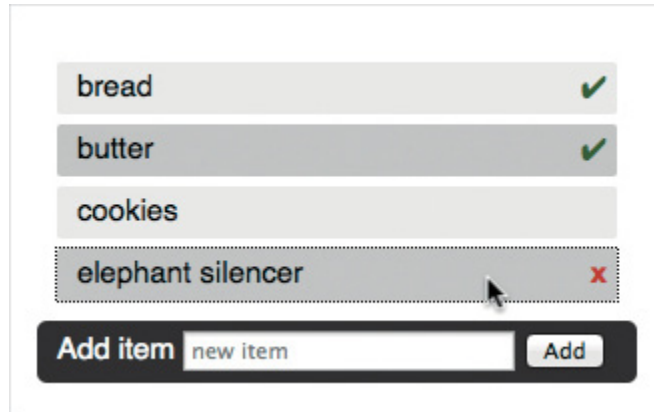


Рисунок 5.2. Список дел с «галочками» и иконкой удаления на втором этапе

Если элементу, на который мы кликаем, не присвоен класс `done`, он добавляется. Если же класс есть, мы удаляем элемент.

Это поддерживает функциональность, а также дает нам дополнительный класс для применения в нашей CSS. Мы можем использовать его для добавления подсказки об удалении (“x”), которая появляется, когда вы наводите курсор над завершенным элементом:

```

#todolist li: hover: after,
#todolist li.done: after {
  content: '
                                ✓
';
  color: #060;
  position: absolute;
  right: 5px;
}
#todolist li.done: hover: after {
  content: 'x';
  font-weight: bold;

```



```
color: #c00;  
position: absolute;  
right: 5px;  
}
```

Формы с проверкой корректного заполнения полей

Как вы помните, прежде чем создать новый перечень элементов, мы проверяем содержимое поля. Прямо сейчас это сделано на JavaScript. Но если мы храним верность среде, в которой творим, то можем обойтись и без этого. Добавление атрибута `required` в HTML гарантирует, что браузер проверит содержимое поля, прежде чем отправит форму:

```
<ul id="todolist"></ul>  
<form action="#" method="post">  
  <div>  
    <label for="newitem">Add item</label>  
    <input type="text" name="newitem" id="newitem"  
      placeholder="new item" required>  
    <input type="submit" value="Add">  
  </div>  
</form>
```

Если пользователь попытается отправить форму, не введя информацию, обработчик этого события точно не останется без работы. Это означает, что мы можем сократить код JavaScript очередной строкой:

```
form.addEventListener('submit', function(ev) {  
  todo.innerHTML += '<li>' + field.value +  
'</li>';  
  field.value = '';  
  field.focus();  
  ev.preventDefault();  
}, false);
```

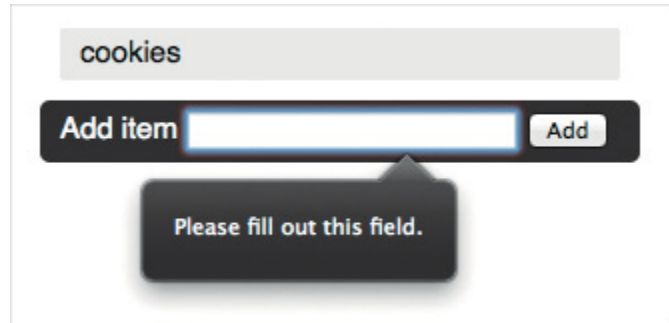


Рисунок 5.3. Firefox демонстрирует ошибку в заполнении поля и подсвечивает поле

Незаполненные поля автоматически помечаются браузером – что-то такое мы вынуждены были делать в прошлом (см. рис. 5.3). Если браузер не поддерживает атрибут `required`, форма будет отправлена. Вот что происходит, когда пользователь (или во многих случаях непрофессионал) исключает JavaScript. Тестирование в JavaScript – это удобное средство, но не мера защиты. В любом случае вам необходимо делать проверки и на сервере.

Сохранение состояния списка

Обычной процедурой в данный момент было бы найти способ сохранения информации в базе данных и запросить у пользователя учетные данные, чтобы сохранить их. Используя HTML5 и допуская, что это приложение будет использоваться на одном компьютере и его не нужно будет синхронизировать между многочисленными устройствами, мы можем использовать локальное хранилище для того, чтобы легко делать копию текущего состояния списка, каждый раз, когда он меняется.

`Session` и `localStorage` – это, по сути, не HTML5, а скорее, собственные стандарты. Для хранения большого объема данных на стороне клиента существуют базы данных `IndexedDB` и `WebSQL`. Однако локальное хранилище необычайно удобно в применении и более чем достаточно для наших нужд.

Для того чтобы сохранить состояние перечня и загружать его, когда пользователь возвращается на страницу, все, что нам нужно сделать, – это написать две функции, `storestate()` и `retrievestate()`:

```

function storestate() {
localStorage.todoList = todo.innerHTML;
};
function retrievestate() {
if (localStorage.todoList) {
todo.innerHTML = localStorage.todoList;
}
};

```

Потом мы должны вызывать эти функции всякий раз, когда меняется перечень:

```

form.addEventListener('submit', function(ev) {
  todo.innerHTML += '<li>' + field.value +
'</li>';
  field.value = '';
  field.focus();
  storestate();
  ev.preventDefault();
}, false);
todo.addEventListener('click', function(ev) {
var t = ev.target;
if (t.tagName === 'LI') {
if (t.classList.contains('done')) {
t.parentNode.removeChild(t);
} else {
t.classList.add('done');
}
storestate();
};
ev.preventDefault();
}, false);

```

Мы извлекаем данные при повторной загрузке страницы:

```
document.addEventListener('DOMContentLoaded',  
retrievestate, false);
```

Вот оно! Кэширование всех интерфейсов в локальном хранилище может показаться грязным приемом, но на самом деле в этом нет ничего сомнительного. Так как HTML не слишком тяжеловесен, и у нас есть 5 МВ памяти в браузерах, этот способ – простое решение распространенной проблемы.

Пример 2: Анимированные элементы страницы с использованием CSS3

[\[62\]](#)

Давайте еще раз быстренько взглянем на ловкий прием присвоения классов и делегирование событий. В этот раз мы поиграем с CSS и JavaScript для анимации разделов сайта без какой-либо библиотеки или инструмента анимации.

Возьмите следующий сайт, который я создал на скорую руку для моего любимого кафе (где я сижу и пишу эти строки прямо сейчас). Без JavaScript он выглядел бы примерно как изображение на следующей странице.

У нас есть несколько заголовков, изображений и поясняющих текстов на очень длинной странице – и это все. Код HTML очень прост: навигационное меню указывает на якоря в документе.

```
<header>  
<h1>Cafe Vintage</h1>  
<p>88 Mountgrove Road, London N5 2LT,  
England</p>  
<nav>  
<ul>  
<li><a href="#cafe">The Cafe</a></li>  
<li><a href="#fashion">Fashion</a></li>  
<li><a href="#food">Food</a></li>  
<li><a href="#gifts">Gifts</a></li>  
</ul>  
</nav>
```

```
</header>
<section>
<article id="cafe">[...]</article>
<article id="fashion">[...]</article>
<article id="food">[...]</article>
<article id="gifts">[...]</article>
</section>
<aside> <p>Opening hours:</p> [...] </aside>
<footer> <p>© 2012 Cafe Vintage and Chris
Heilmann</p> </footer>
```

Это HTML старой школы, с несколькими новыми элементами, представленными Беном Шварцом в третьем разделе. Он работает во всех браузерах, а между навигацией и основным контентом существует логическая связь: ID (идентификаторы).

Когда доступен JavaScript, страница показывает один раздел за один раз. Также существует анимация для перемещения между разделами: последняя страница движется вверх, а новая опускается вниз. А описание перемещается справа налево (см. рис. 5.5–5.7.).

Чтобы это случилось, все, что нам требуется, это добавить и удалить несколько классов. Остальное происходит в CSS. Вот логика, которой мы придерживаемся:

- Применяем класс с названием `js` к «телу» документа и прячем все элементы `article` в CSS по селектору `.jsarticle {...}` (в этом случае позиционируем их абсолютно и передвигаем их в экран).

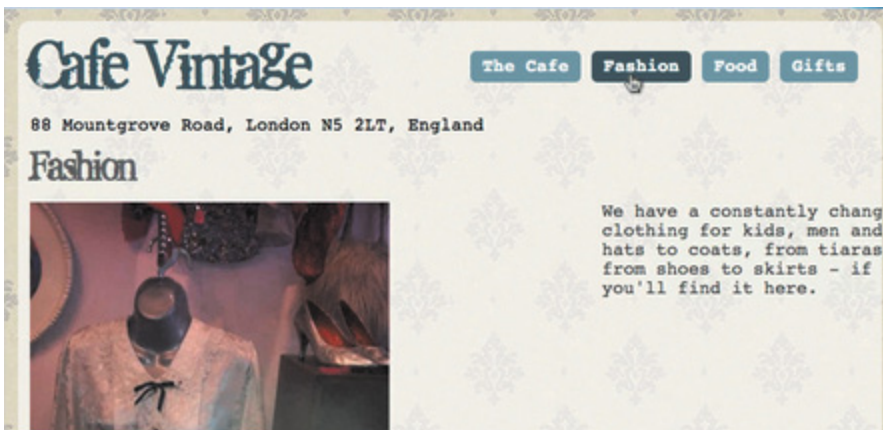
- К статье, выбранной пользователем, добавляем класс с названием `current`, который подменяется в CSS на `.jsarticle.current {...}`.

- Для того чтобы показать пользователю, где он находится, добавляем класс с названием `current` к меню ссылки, соответствующей текущей видимой статье.



Рисунок 5.4. Сайт простой кафешки с названием «Винтаж», где все разделы показаны один за другим

Сначала мы добавим класс с названием `js` к «телу» документа. Это позволит нам определить стили для версий с и без JavaScript. Потом мы возьмем элементы, которые нам нужны. В этом случае нам нужны первый элемент `article` и первая ссылка, потому что они будут доступны первыми по умолчанию.



Рисунки 5.5–5.7. Если JavaScript доступен, статьи на странице «оживают», когда пользователь нажимает на элемент навигации

```
document.body.classList.add('js');
var nav = document.querySelector('nav'),
    article = document.querySelector('article'),
    link = document.querySelector('nav a');
```

Мы установим их классы на current:

```
link.classList.add('current');  
article.classList.add('current');
```

Остальные функции – это делегирование событий:

```
nav.addEventListener('click', function (ev) {  
    var t = ev.target;  
    if (t.tagName === 'A') {  
        article.classList.remove('current');  
        link.classList.remove('current');  
        article =  
document.querySelector(t.getAttribute('href'));  
        link = t;  
        article.classList.add('current');  
        link.classList.add('current');  
    }  
    }, false);
```

Теперь мы назначим обработчик нажатия в навигационном меню, проверим, что на ссылку щелкнули и удалим класс current из ссылки и из статьи, которая демонстрируется. Потом мы доходим до новой статьи, которую нужно показать, обращаясь к ней через атрибут href – ссылки, на которую щелкаем – и затем назначаем класс current новым элементам.

Это почти все, что нам нужно сделать. Другой случай, который нам нужно учесть, это когда пользователь попадает на сайт через ссылку, содержащую хеш (когда ссылка указывает на статью, которая показывается не по умолчанию); в этом случае нам нужно показать правильную статью. Для этого мы можем использовать два селектора:

```
if (document.location.hash) {  
    var cleanhash =  
document.location.hash.replace(/^#/ , '');
```



```

    article =
document.querySelector(document.location.hash);
    link = document.querySelector('nav a[href$='
+ cleanhash + ']'
);
}

```

CSS-селектор проверяет, заканчивается ли ссылка тем, что мы передаем. Анимация сделана в CSS с использованием перемещений:

```

section {
overflow: hidden;
min-height: 340px;
position: relative;
}
article {
position: relative;
height: 350px;
}
body.js article {
width: 700px;
position: absolute;
top: -700px;
- webkit-transition: 0.8s;
- moz-transition: 0.8s;
- ms-transition: 0.8s;
- o-transition: 0.8s;
transition: 0.8s;
}
body.js article.current {
position: absolute;
top: 0;
}

```

Здесь мы видоизменяем статьи. Мы просто говорим браузеру расположить их относительно в разделе, когда JavaScript недоступен, и

разместить их абсолютно на 700 пикселей над верхней частью контейнера, когда JavaScript доступен. Так как в разделе применяется `overflow: hidden`, они никогда не показываются.

Если статья текущая, значение `top` изменяется на 0, и статья перемещается сверху вниз.

Параграфы действуют по тому же принципу:

```
article p {
position: absolute;
left: 320px;
width: 370px;
}
js article p {
left: 900px;
opacity: 0;
- webkit-transition: 1s ease 0.7s;
- moz-transition: 1s ease 0.7s;
- ms-transition: 1s ease 0.7s;
- o-transition: 1s ease 0.7s;
transition: 1s ease 0.7s;
}
js article.current p {
position: absolute;
left: 320px;
width: 370px;
opacity: 1;
}
```

В этом примере мы создали 1-секундное перемещение с задержкой в 0,7 секунды. Обратите внимание, что мы анимируем и положение слева и прозрачность одним махом. И нам не нужно ничего делать в JavaScript.

Пример 3: Создание миниатюры изображения в браузере

[63]

Лучшее, что есть в HTML5, – это элемент canvas (холст). Казалось бы, это просто элемент для рисования (а без JavaScript он и вовсе лишен смысла), но он же является мощным средством для чтения и управления изображениями и видеоданными. Вместе с FileReader и функцией DragandDrop («перетаски и оставь») в современных браузерах мы можем добиться реального успеха. Почему бы нам не создавать миниатюры в браузере?

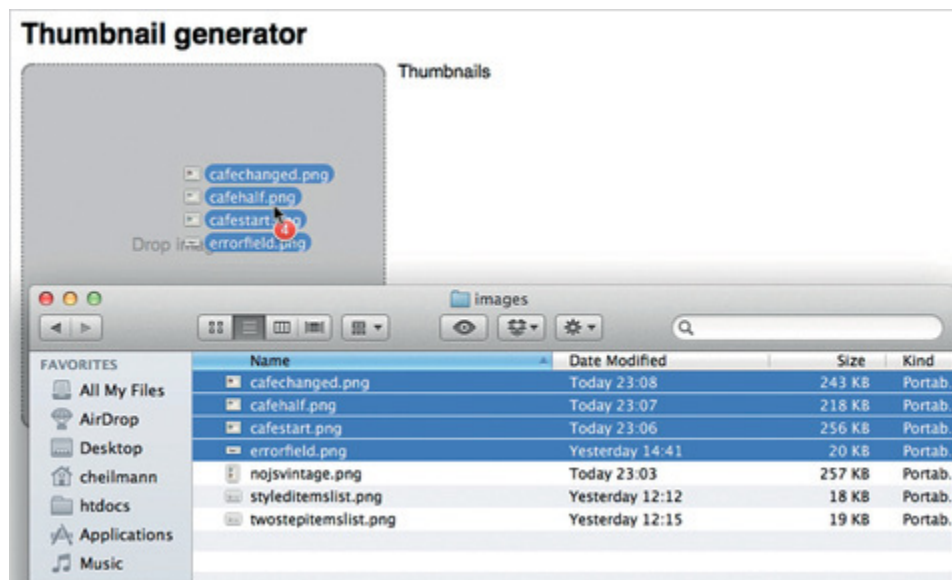


Рисунок 5.8. Перетаскивание с использованием canvas и FileReader

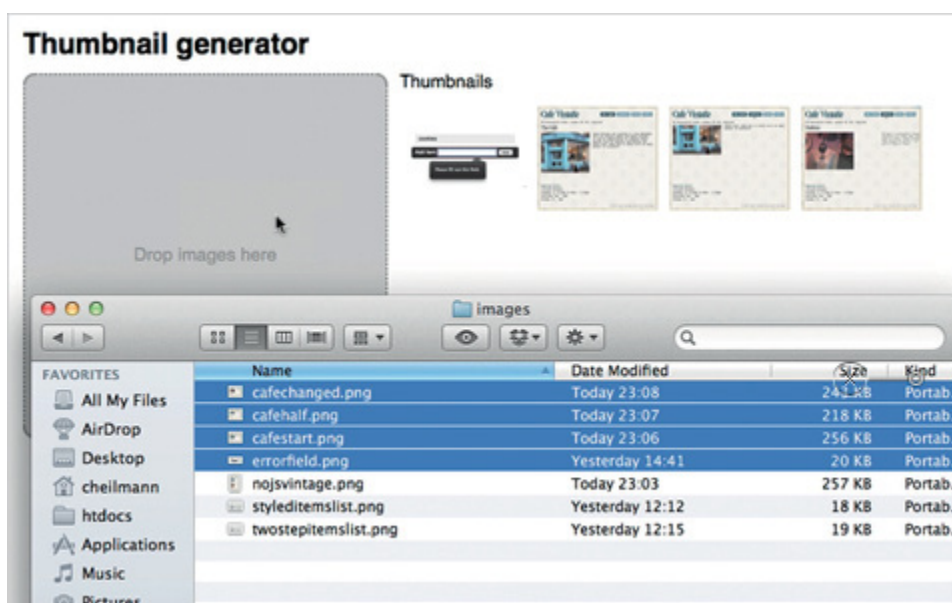


Рисунок 5.9. Созданные миниатюры

У большинства систем управления контентом и блогов есть загрузчики файлов для создания миниатюр из изображений. Поэтому давайте использовать их в качестве нашего резерва (к тому же мы здесь не будем погружаться в серверный код):

```
<section>
<form>
<label for="upload">Pick image</label>
<input type="file" id="upload" name="upload">
<input type="submit" value="Make it so!">
</form>
</section>
<output><p>Thumbnails</p></output>
```

Давайте добавим разные стили. И, если браузер поддерживает их, мы заменим форму на сообщение, где попросим пользователя перетащить изображения в раздел:

```
if (window.FileReader && (('draggable' in
document.body) ||
('ondragstart' in document.body && 'ondrop' in
document.body))) {
var s = document.querySelector('section'),
o = document.querySelector('output'),
c = document.createElement('canvas'),
cx = c.getContext('2d'),
thumbsize = 100;
c.width = c.height = thumbsize;
document.body.classList.add('dragdrop');
s.innerHTML = 'Drop images here';
```

Здесь мы проверили, как браузер поддерживает функции FileReader и DragandDrop (досадно то, что Safari не поддерживает первое, а Opera – последнее). Если браузер их не поддерживает, то мы берем те элементы, которые нам нужны. Мы создаем элемент canvas и

сохраняем его 2-Dcontext. Потом мы определяем размеры миниатюр и, соответственно, меняем размеры canvas. И, наконец, мы добавляем класс dragdrop для задания стиля, и заменяем форму на сообщение, приглашающее пользователей перетаскивать изображения.

Как правило, если вы перетаскиваете изображение в браузер, он просто заменяет им текущий документ. Мы хотим избежать этого. Вот почему мы меняем поведение браузера при перетягивании. Мы также добавили класс, чтобы показать пользователю, что что-то происходит:

```
s. addEventListener('dragover', function (evt) {  
s. classList.add('active');  
evt.preventDefault();  
}, false);
```

Мы удаляем класс, если перетаскивание отменяется:

```
s. addEventListener('dragleave', function (evt)  
{  
s. classList.remove('active');  
evt.preventDefault();  
}, false);
```

Все остальные функции действуют в drop-обработчике:

```
s. addEventListener('drop', function (ev) {  
s. classList.remove('active');  
var files = ev.dataTransfer.files;  
if (files.length > 0) {  
var i = files.length;  
while (i-) {  
var file = files[i];  
if (file.type.indexOf('image') !== -1) {  
createthumb(file);  
}  
}  
}  
ev.preventDefault();
```

```
}, false);
```

Первое, что мы сделали здесь, это убрали класс `active`, потому что делали его с перетаскиванием. Событие перетаскивания дало нам объект `dataTransfer`, в котором содержатся «перетащенные» файлы. Далее мы проверяем, был ли хоть один из файлов перемещен и затем начинаем повторять это действие для каждого из них (Цикл `while{}` – причудливый способ сделать цикл `for{}`, но не помещая длину в кэш-память или не используя вторую переменную итератора). Еще проверяем, является ли текущий файл изображением, а потом отсылаем его к функции `createthumb()` (создания миниатюры). В конце мы предотвращаем поведение браузера по умолчанию при перетягивании изображения. И все, с этим мы разобрались.

```
function createthumb(file) {
  var reader = new FileReader();
  reader.readAsDataURL(file);
  reader.onload = function (ev) {
    var img = new Image();
    img.src = ev.target.result;
    img.onload = function() {
      cx.clearRect(0, 0, thumbsize, thumbsize);
      var thumbgeometry = resize(this.width,
this.height,
  thumbsize, thumbsize);
      cx.drawImage(img, thumbgeometry.x,
thumbgeometry.y,
  thumbgeometry.w, thumbgeometry.h);
      var thumb = new Image();
      thumb.src = c.toDataURL();
      o.appendChild(thumb);
    };
  };
}
```

Функция `createthumb()` иницирует новый `FileReader` и читает изображение как цепочку данных. Если считыватель загружен удачно, то мы создаем новое изображение в браузере и устанавливаем его атрибут `src` для результата транзакции `FileReader`.

Когда изображение успешно загружено, мы очищаем элемент `canvas` методом `clearRect()`. Это обязательно. Иначе нам придется создавать миниатюры, которые добавляются друг к другу каждый раз, когда вызывается функция.

Потом мы задаем размер миниатюры, который хотим получить от функции `resize()` и вызываем метод `drawImage` для элемента `canvas`. Этот метод использует пять параметров: изображение для получения информации о пикселях, координаты левого верхнего угла, откуда рисуется изображение, ширину и высоту. Потом мы создаем новое изображение, сохраняем пиксельный контент холста методом `toDataURL()` и добавляем новое изображение к выведенному элементу.

```
function      resize(imagewidth,      imageheight,
thumbwidth, thumbheight) {
    var w = 0, h = 0, x = 0, y = 0,
    widthratio = imagewidth / thumbwidth,
    heightratio = imageheight / thumbheight,
    maxratio = Math.max(widthratio, heightratio);
    if (maxratio > 1) {
        w = imagewidth / maxratio;
        h = imageheight / maxratio;
    } else {
        w = imagewidth;
        h = imageheight;
    }
    x = (thumbwidth - w) / 2;
    y = (thumbheight - h) / 2;
    return { w: w, h: h, x: x, y: y };
};
```

Функция `resize` – математический помощник в изменении размера изображения определенной ширины и высоты на меньшее, но с теми же пропорциями. Ничего сверхъестественного, простая практика. И вот так мы получаем миниатюры в нашем браузере.

Заключение

Надеюсь, мне удалось убедить вас в том, что сегодня много отличных фишек стали «родными» для браузеров. Конечно, пока не все из них поддерживают эти функции, но по крайней мере все производители браузеров вместе работают над стандартами, и сейчас не то время, когда между первыми браузерами шла война, а инновации творились наощупь. При смещении и подгонке различных веб-технологий (HTML, CSS, JavaScript) мы можем создавать совершенно потрясающие вещи всего в нескольких строках кода. Все, что нам нужно – это пользоваться теми возможностями, которую дают нам браузеры.

Правильная технология работы

Взаимодействие CSS перемещений, трансформаций и анимаций с JavaScript – мощный инструмент, и нам следует использовать его гораздо больше. Сейчас, похоже, идет битва между теми, кто везде применяет jQuery или JavaScript, и теми, кто работает исключительно с CSS. Это не помогает нашим пользователям, а нас тормозит с написанием лаконичных, эффективных решений. Хороший веб-разработчик почерпнет достоинства из всех технологий, а не будет приверженцем какой-то определенной. Много ума не надо, чтобы использовать одну технологию на все случаи жизни. А потом оставить продукт работать только в одном браузере или на одном устройстве.

Если вам нужно, чтобы все необходимые свойства поддерживались в старых браузерах, используйте библиотеку, такую как jQuery. Вы также можете найти «заплаты» для старых браузеров в форме полифилов. А вообще, давайте уже прекратим пытаться заставить устаревшие технологии поддерживать то, что мы создаем для новых. Никто не перестанет использовать продукт только из-за того, что браузер IE 6 не делает плавных перемещений из одного состояния в другое. А это и есть самое важное свойство хорошего веб-продукта.

Большая игра

Как веб-разработчики мы всегда сетуем на то, что технология не отвечает нашим потребностям: люди пользуются давно устаревшими браузерами, а браузеры не дают того, что нам нужно. Основная причина использования старых браузеров кроется в прошлом. Тогда разработчики типа нас с вами создавали сайты только для этих браузеров, потому что они были современными. Считалось, что лучше ничего и быть не может.

Производители браузеров не добавляют каждое свойство, которое хотят видеть разработчики из-за того, что новая технология внедряется недостаточно широко. Мало пожаловаться на то, что что-то не работает.

Производители браузеров хотят, чтобы эта технология использовалась по полной программе. И еще им нужно получать информацию от потребителей о ее действии. Если обратная связь заключается только в вопросе: «Почему вы не поддерживаете функцию X?» – вам всегда ответят: «Потому что ею никто не пользуется».

Самый яркий пример тому – новые семантические элементы HTML5. Нам грех жаловаться на недостаточную поддержку алгоритма outline в браузерах, если мы сами не используем правильные элементы, а потом сообщаем производителю, что работает, а что – нет. У каждого производителя есть механизм обратной связи. От нас как от разработчиков зависит, давать ли быстрее реальные ситуации для решения или ждать безупречного внедрения.

Принимаем и применяем технологии будущего

Нам нужно ухватить новую технологию и использовать ее, где только возможно. Отбросим мысли о том, что надо ждать, пока все браузеры станут поддерживать определенную технологию и из-за этого не применять ее. Если мы не опробуем новые технологии, внедренные рабочими группами WHATWG и W3C, мы ни к чему не придем.

И все же мы не должны навязывать технологии людям. Сообщение типа «Обновите браузер X, чтобы увидеть этот контент» может расстроить пользователя, если у него нет права на загрузку нового браузера в свой компьютер или у него медленное соединение. Вместо

того чтобы создавать ракетные ранцы, давайте построим эскалаторы. Они позволят людям удобно подняться на более высокие уровни, но при этом будут работать как обычная лестница, если вдруг пропадет электричество или сломается механизм.

Об авторе



* * *

Кристиан Хейлманн (1975) родился в Швайнфурте, Бавария. Имеет диплом по немецкому и английскому языкам, истории и астрономии. Его девиз: «Если берешься за что-то, доводи до конца. Не хочешь – тогда брось!» Сейчас Кристиан живет в Северном Лондоне, куда стекаются люди из многих классных местечек. Он работает над тем, чтобы доводить технологии до людей, а людей до технологий. Когда он не занят этим, то переключается на фильмы. Его любимые цвета – синий и зеленый. Из-за того, что он редко бывает дома, его единственные питомцы – это множество резиновых уточек. Послание читателям от Кристиана: «Оставайтесь жаждущими и пытливыми. Новое всегда где-то рядом!»

О рецензенте



* * *

Пол Айриш (1982) родился в Питсфилде, штат Массачусетс. Окончил Вустерский политехнический институт, имеет степень бакалавра технических коммуникаций. В течение семи лет профессионально занимался разработкой пользовательского интерфейса. Последние два года обучает других разработчиков тому, как сделать Интернет более поразительным.

Пол живет в квартире-студии в районе Мишин в Сан Франциско. Все свое свободное время он любит слушать электромузыку и ходить на фуршеты. Главное, чему он научился на протяжении своей карьеры, – это добавлять причуды во всю тяжелую работу. Личный совет читателям от Пола: «Публикуйте то, чему учитесь». Слова эксперта! Так что стоит попробовать.

Техники для создания лучшего пользовательского опыта

Автор: Дмитрий Фадеев

Рецензент: Джошуа Потер

Дизайн и юзабилити сайтов и приложений изменяется в лучшую сторону. Чем больше компаний и программ переходят в онлайн, тем сильнее разгорается конкурентная борьба и компании начинают выискивать любое преимущество, чтобы остаться на плаву и выбиться в лидеры. Если вы думаете над еще нерешенной проблемой, то можете отделаться бессистемным сайтом, но если же вы боретесь в числе двадцати конкурентов, юзабилити и опыт использования вашего продукта начинает приобретать гораздо большее значение.

Опыт использования (UX, userexperience) сегодня – настолько горячее, модное словечко, что есть уже ряд специальностей, которые включают этот термин в своем названии. Я лично не люблю употреблять этот термин из-за того, что он не дает конкретики. Тем не менее он популярен, потому что то, для чего он служит, очень важно. Проще говоря, UX означает хороший дизайн. Не в поверхностном смысле, как отрада для глаз, а в том, как все слаженно увязывается между собой, как работает продукт и насколько он соответствует ожиданиям пользователя.

«Спроектируйте свой продукт так, чтобы он вел себя как располагающий к себе человек и пользователи любят его»

Ален Купер

Цель дизайна не в том, чтобы «украшать» проблемы, а в том, чтобы их решать. Неважно, означает ли это получение большего числа регистраций, приглашения пользователей к созданию контента или более простой и быстрый интерфейс. Все названное относится к видам дизайна, чья конечная цель – непременно оставить в душе пользователя большое впечатление от взаимодействия с продуктом.

Конечно, внешний вид тоже имеет значение и напрямую влияет на это. Но все же не настолько, чтобы вы бросали всю свою энергию на создание распрекраснейшего интерфейса и оставляли без должного

внимания такие вещи, как копирайтинг, структуру, контент и удобство использования (юзабилити).

Этот раздел вооружит вас новыми и мощными UX-техниками, которые вы сможете применять к вашим продуктам и сайтам. Они помогут вам получить преимущества в конкурентной борьбе и заставят пользователей проникнуться любовью к вашему творению. Этот раздел разбит на четыре части. Мы начнем с того, что посмотрим, как можно улучшить процесс регистрации и регистрационные формы, проследим за новейшими и наиболее экспериментальными техниками, обсудим идеи по усовершенствованию клиентского сервиса. Ну и, конечно, поговорим о тех техниках, которые вы предпочли бы избежать.

Улучшаем формы и процесс регистрации

«Я никогда не начну проектировать задание, пока не увижу место и не встречу с теми, кто будет им пользоваться»

Франк Ллойд Райт

Только после того как реальные люди начнут пользоваться вашим продуктом, вы узнаете, на правильном ли вы пути в разработке своего решения и набора функций. МэтМуленберг, разработчик – основатель системы управления контентом WordPress, проводит прекрасную аналогию между использованием и кислородом: «Вы никогда не сможете до конца предугадать, как аудитория отреагирует на то, что вы создадите, пока это будет оставаться за кадром. Это значит, что все, над чем вы работаете и не выпускаете в свет, просто умрет как от нехватки кислорода». Этот раздел представит ряд техник, которые помогут вам вызвать первоначальный интерес к продукту и создать непрерывный процесс регистрации после его выпуска.



Рисунок 6.1. Страница «Скоро открытие» сервиса SquidChef

Скрытый опрос

Первое, что нужно для любого большого продукта, это пользователи. И вы можете привлечь их еще до запуска своего детища, и даже до начала его создания. Возможно, вы знакомы со страницей Coming soon («Скоро открытие/запуск»). Она разработана для продвижения нового продукта или услуг, которые должны быть вскоре выпущены. На странице высвечиваются характеристики и преимущества продукта, запрашивается ваш e-mail на случай, если вы захотите получить уведомление о его выпуске.

Такие страницы представляют собой отличное средство для оценки потребности в вашем продукте, особенно если вы еще не приступили к его созданию. Количество регистраций покажет вам, есть ли спрос на продукт, и если да, то насколько он достаточен, чтобы ответить на него. Качество лидов зависит от количества информации на целевой странице.

Представление подробной информации о вашем продукте, а также потенциальных расценок поможет вам получить отзывы от людей,

которые в этом реально заинтересованы, а не просто проявляют любопытство.

Вот один из способов, как сделать так, чтобы целевая страница не просто собирала лиды, но и предоставляла вам ценную информацию о тех функциях, на которых следует сфокусироваться. Monotask применил эту умную технику на своей coming-soon странице. Если кого-то интересует продукт, то он оставляет свой e-мейл, чтобы попасть в список рассылки, уведомляющей о запуске продукта. При этом появляется краткий опросник под названием «Помогите нам создать, то, что вы хотите!» В опроснике заложено несколько вопросов о том, каким образом сейчас пользователь решает проблему, и сколько он готов заплатить за приложение.

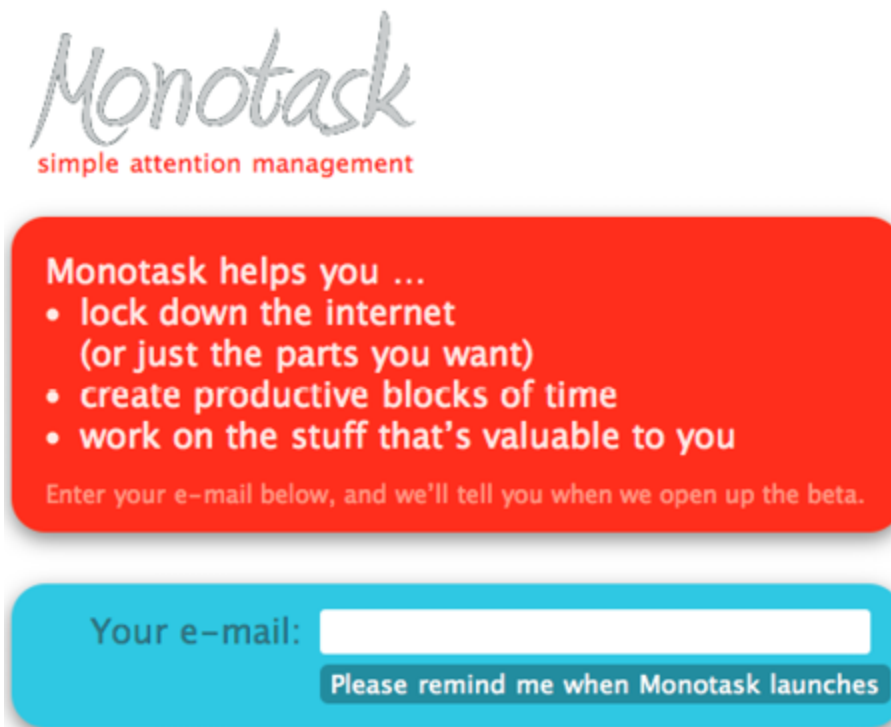
The image shows a landing page for 'Monotask'. At the top, the logo 'Monotask' is written in a grey, handwritten-style font, with the tagline 'simple attention management' in a smaller, red, sans-serif font below it. Below the logo is a large red rounded rectangle containing white text. It says 'Monotask helps you ...' followed by a bulleted list: '• lock down the internet (or just the parts you want)', '• create productive blocks of time', and '• work on the stuff that's valuable to you'. Below the list, it says 'Enter your e-mail below, and we'll tell you when we open up the beta.' Below the red box is a light blue rounded rectangle. Inside, on the left, it says 'Your e-mail:' followed by a white input field. To the right of the input field is a dark blue button with white text that says 'Please remind me when Monotask launches'.

Рисунок 6.2. Форма, собирающая e-mail на странице Monotask

Соккрытие опросника за формой собирающей e-mail преследует две цели. Во-первых, первоначальное побуждение к действию должно быть простым и ясным. Все, что нужно сделать пользователю, – это дать свой e-mail-адрес. Демонстрация формы сразу же могла бы отвлечь от этого действия.



Thanks for signing up! We'll let you know when we launch.

In the meantime, it would be awesome beyond words if you filled out a quick survey, to help us make Monotask something you'll really love. ↓

Help Us Build Something You Want

We haven't even launched yet, and we're looking to make Monotask more useful. Help us?

How do you currently block online distractions?

How damaging are distractions to you?

Business or Pleasure?

What would you be willing to invest in a service that helped you stay on target?

- | | |
|---|--|
| <input type="radio"/> \$6 / month | <input type="radio"/> \$8 / month |
| <input type="radio"/> \$10 / month | <input type="radio"/> \$12 / month |
| <input type="radio"/> \$14 / month | <input type="radio"/> \$16 / month |
| <input type="radio"/> I only use free tools | <input type="radio"/> more than \$16 / month |

Submit

Survey powered by Wufoo

Рисунок 6.3. После подтверждения вашего e-mail-адреса показывается краткий опросник

Во-вторых, только те, кто заинтересовался вашим продуктом, увидят опросник, что повысит качество лидов. Использование подобного опросника – не только прекрасный способ собрать данные, с помощью которых вы создадите то, что понравится людям. Он также укажет вам на основные «болевые точки», чтобы вы могли их устранить, и расскажет о том, для каких целей пользователь хочет приобрести ваш продукт.

Как только вы запуститесь, обращение к этим темам на ваших продающих страницах упрочит их положение.

Сделайте регистрацию скрытой

Существует UX-миф о том, что вы должны представлять ссылку на регистрацию или саму форму регистрации сразу же, т. е. вверху целевой страницы. Логика этого мифа исходит из идеи об устранении возможных препятствий из процесса регистрации. Однако в этой логике есть изъян, и кроется он в самом определении препятствия – в данном случае это все, что не является регистрацией.

Регистрационная форма – это не единственное, что нужно заполнить посетителям. Да, конечно, ссылка на нее и она сама важны для процесса, но прежде, чем люди начнут вводить информацию, должны произойти еще две вещи. Во-первых, посетители должны четко знать, что им дает регистрация. Во-вторых, они должны этого захотеть. Здесь не помешает хороший копирайтинг.

Если посетители еще не слышали о вашем продукте из внешних источников, им нужно рассказать о том, что он может и, самое важное, для чего он им.

Например, когда компания ZURB создавала домашнюю страницу для одного своего заказчика и перенесла кнопку регистрации вниз страницы, обнаружилось, что конверсия увеличилась на 350 %^[64] – неслабое улучшение. У Vendio регистрационная форма стояла первой позицией на одной из страниц, сместив текст вниз. При переделке дизайна форму перенесли на отдельную страницу, а ссылку на нее оставили надомашней. Что в итоге?

Конверсия поднялись на 60 %^[65].

vendio
Simply Powerful eCommerce

100% Free Online Stores

The Vendio Store

100% FREE

Create Username

Password

Confirm Password

Your Email

Confirm Email

User Agreement

☒ Sign me up to receive exclusive deals and Vendio news.

☐ I have read and agree to the Vendio [User Agreement](#)

[Sign Up Now](#)

[CLICK IMAGES TO ENLARGE](#)

100% Free Online Stores - No Credit Card Required

- Fast and easy setup: Our drag-and-drop editor will have your store up and running in minutes.
- Professional designs: Dozens of templates to choose from to make your store stand out.
- Analytics and SEO integrated: Effortlessly set up Google Analytics and optimize keywords.
- Multi-channel support: Increase exposure by listing your store items to eBay and Amazon.
- Completely FREE: Too good to be true? Nope. [Learn more.](#)

Partnered with:

[amazon.com](#) [eBay](#) [UPS](#) [FedEx](#) [Google](#) [TRUSTe](#)

Copyright © 1998 - 2010 Vendio Services, Inc. - [Privacy Policy](#)

Рисунок 6.4. Сайт Vendio с формой регистрации прямо на главной странице (исходная версия)

vendio
Simply Powerful eCommerce

100% Free eCommerce Websites

The Vendio Store

100% FREE Ecommerce Website

Create Username

Password

Confirm Password

Your Email

Confirm Email

User Agreement

☒ Sign me up to receive exclusive deals and Vendio news.

☐ I have read and agree to the Vendio [User Agreement](#)

[Sign Up Now](#)

[CLICK IMAGES TO ENLARGE](#)

Free eCommerce site - No hosting, listing or final value fees!

Partnered with:

[amazon.com](#) [eBay](#) [UPS](#) [FedEx](#) [Google](#) [TRUSTe](#)

Copyright © 1998 - 2010 Vendio Services, Inc. - [Privacy Policy](#)

Рисунок 6.5. Сайт Vendio с регистрационной формой, перенесенной на отдельную страницу (финальная версия)

Люди не станут нигде регистрироваться, пока не убедятся, что им это необходимо, даже если ссылка на регистрацию или ее форма расположена в самом начале страницы. Переместите регистрационную форму вниз страницы или, точнее говоря, поместите ваш текст наверх страницы, в первый экран. Когда люди прочтут сообщение от вас и будут готовы, они начнут действовать.

Плавное вхождение в контакт

Еще один UX-миф гласит, что регистрационная форма должна быть как можно короче. На самом деле это и важно, и нет. Что действительно играет роль, так это то, как происходит процесс регистрации и будет ли от него польза потребителю. Другими словами, имеет ли тот дополнительный шаг, который вы хотите ввести, ценность для самого пользователя?

Например, когда Twitter переделывал свою регистрационную форму, то ввел дополнительный шаг в процесс. Теперь на первом шаге Twitter пытается подключиться к e-mail-адресу посетителя для поиска друзей, которые уже пользуются Twitter. Если пользователь не хочет показывать контакты своей почты, то он окажется прямиком на пустой странице профиля – не самое лучшее начало.

Также добавился еще один шаг с предложениями. Сейчас пользователь может выбирать темы, которые ему интересны. А Twitter предлагает популярные аккаунты, на которые можно подписаться. Итак, даже если пользователь не найдет друзей, то узнает об интересных людях, а новый профиль сразу же покажет, о чем они «щебечут».

Редизайн сработал? Да: конверсия выросла на 29 %^[66].

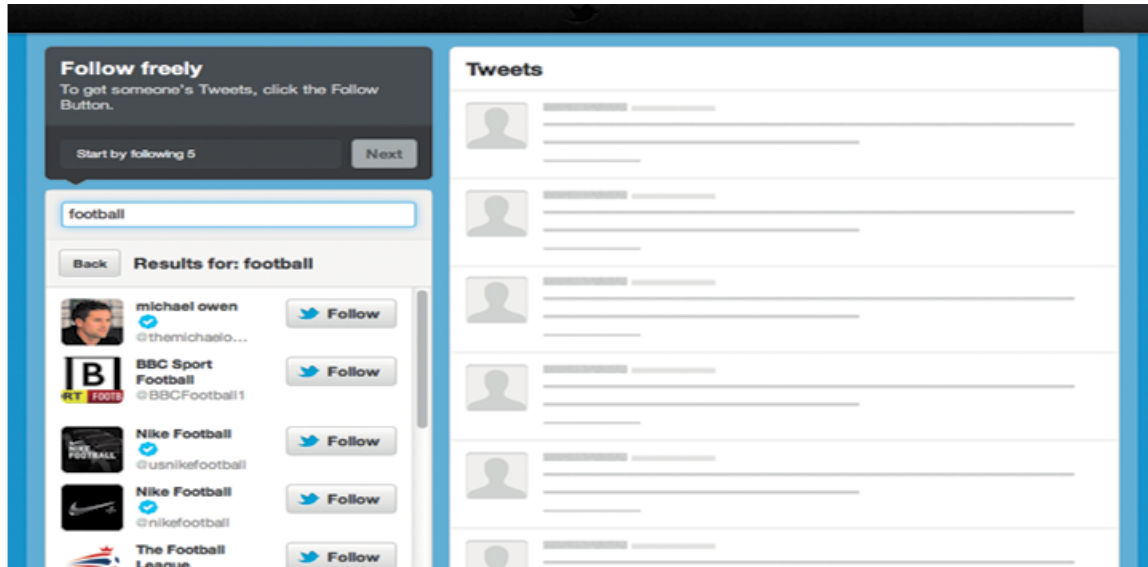
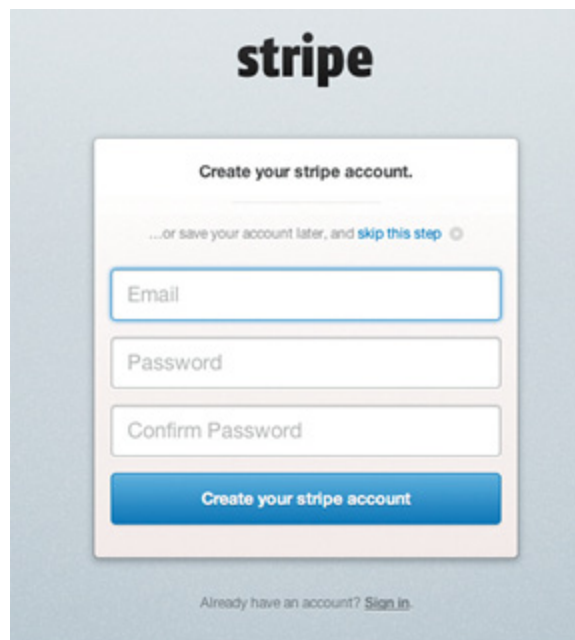


Рисунок 6.6. Twitter помогает вам найти людей для подписки на их ленты в процессе регистрации

Почему это сработало? Дополнительное предложение позволило пользователям более нацелено подключиться к сервису. Вместо заполнения пустой формы Twitter помогает вам сразу же перейти к интересующим вас людям. Даже если вы точно не знаете, кто вам нужен, вы можете пробежаться по самым популярным тематическим профилям. Например, если вас интересует веб-дизайн, вы можете начать поиск по теме прямо в режиме регистрации и получить список нужных вам аккаунтов. Это делает сервис полезным с самого начала и усиливает вхождение в контакт в процессе регистрации, когда люди создают свой новый аккаунт.

Последовательная регистрация

Как выяснилось, удлинение процесса регистрации – хорошее дело, если в этом есть смысл для пользователей. В то же время, вы можете и сократить его, а в некоторых случаях даже полностью исключить во время первоначального взаимодействия. Один из лучших способов показать потенциальным пользователям, что из себя представляет ваш продукт, это дать им его попробовать. В некоторых случаях вам не нужно, чтобы пользователь заполнял регистрационную форму до того, как вы покажете свой продукт. Просто дайте клиенту сразу проверить его в действии.



The image shows the Stripe registration form. At the top is the Stripe logo. Below it is a white box with the heading "Create your stripe account." and a link "...or save your account later, and [skip this step](#)". The form contains three input fields: "Email", "Password", and "Confirm Password". Below these is a blue button labeled "Create your stripe account". At the bottom of the form is a link "Already have an account? [Sign in](#)".

Рисунок 6.7. Регистрационная форма сервиса Stripe позволяет вам «пропустить этот шаг»

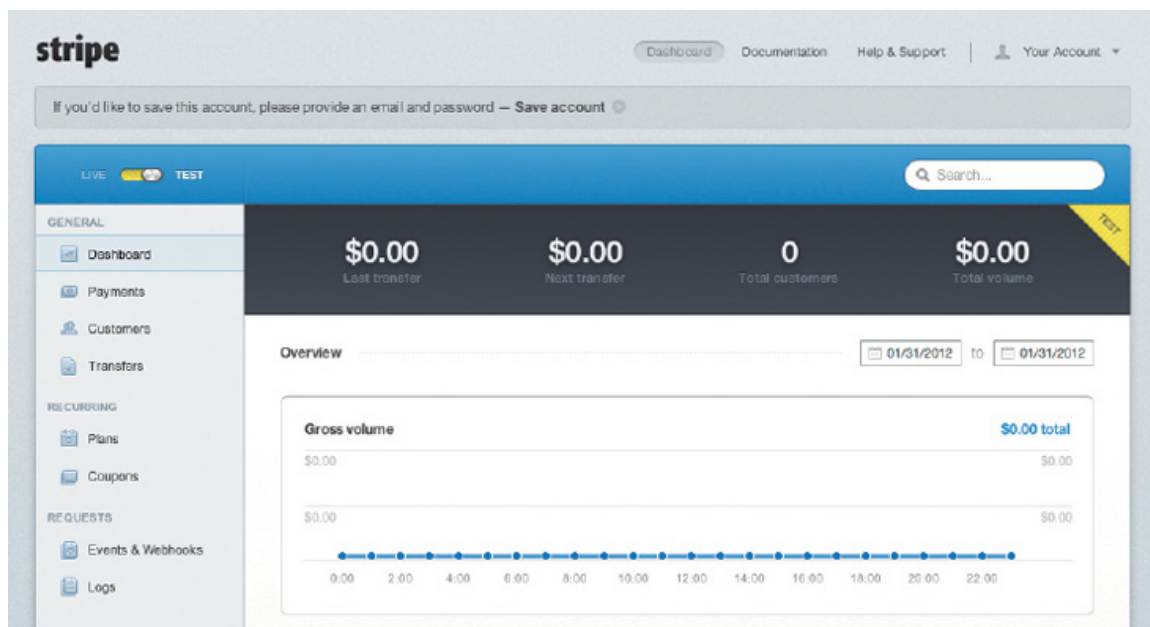


Рисунок 6.8. Вы можете сразу, без регистрации, «перескочить» на рабочий экран Stripe. Верхняя ссылка «Save account» («Сохранить учетную запись») позволит вам зарегистрироваться позже

Как это делает онлайн-редактор QuietWrite. Первое, что посетители видят, заходя на целевую страницу – это краткое описание продукта. Если им стало интересно, то они могут нажать на ссылку и начать пользоваться им. Понравилось то, что увидели? Следующим шагом будет заполнение имени пользователя и пароля. QuietWrite использует куки-файлы, чтобы отслеживать пользователей во время начального контакта. Поэтому даже если человек не указывает имя пользователя сразу же, он может продолжить работу. Кстати, есть стимул для введения имени пользователя для доступа к приложению с разных компьютеров. Потом у клиента появится хороший повод перейти к регистрации.

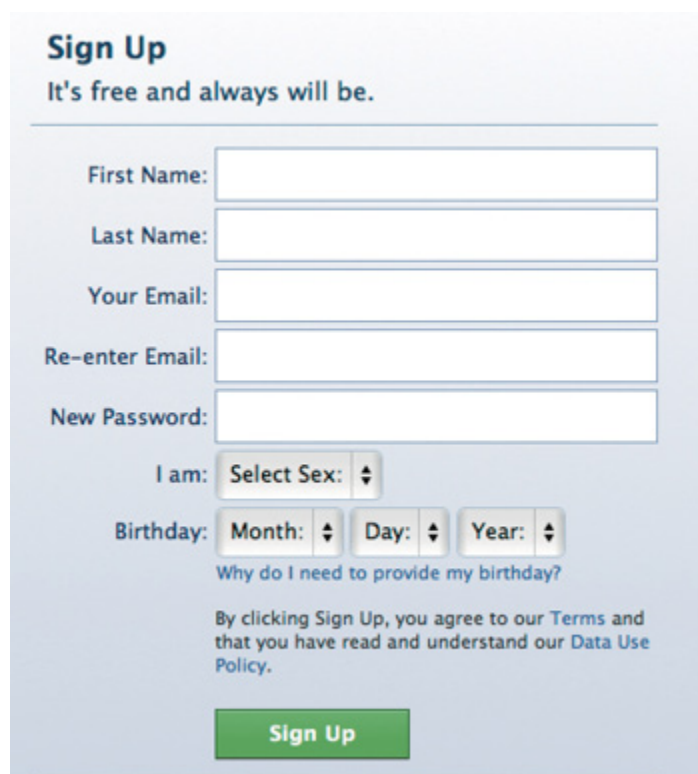
The image shows a screenshot of the Facebook registration page. At the top, it says "Sign Up" in blue, followed by "It's free and always will be." in a smaller blue font. Below this is a horizontal line. The form consists of several input fields: "First Name:", "Last Name:", "Your Email:", "Re-enter Email:", and "New Password:". Below these is a section for "I am:" with a "Select Sex:" dropdown menu. Then there is a "Birthday:" section with three dropdown menus for "Month:", "Day:", and "Year:". Below the birthday section is a link that says "Why do I need to provide my birthday?". Further down is a paragraph of text: "By clicking Sign Up, you agree to our [Terms](#) and that you have read and understand our [Data Use Policy](#)." At the bottom of the form is a green button with the text "Sign Up" in white.

Рисунок 6.9. Регистрационная форма Facebook с подтверждением адреса e-mail

QuietWrite продолжает сегментировать процесс, запрашивая дополнительную информацию позже. Например, чтобы опубликовать свои записи, вам нужно выбрать имя пользователя. Эта информация не была необходимой для сохранения вашей учетной записи, поэтому ее запрасят позже, когда она действительно понадобится.

Stripe, платежный процессор, еще один хороший тому пример. На его странице регистрации есть ссылка «Пропустить этот шаг», которая напрямую отправляет вас к приложению. Вы можете просмотреть его, а потом добавить свое имя и пароль, если решите начать им пользоваться. Демонстрация потенциальным клиентам того, чем они будут пользоваться, проводимая до регистрации – еще один способ для продвижения продукта.

В противовес, некоторые сервисы всегда будут запрашивать от вас детали до того, как они смогут пригодиться. Социальные сети, такие как Facebook, основаны на социальных взаимосвязях, которые не смогут осуществляться, пока вы не идентифицируете себя. Facebook заставляет вас пройти регистрацию сразу же, размещая форму на посадочной странице. Это противоречит высказанной ранее позиции о скрытии регистрационной формы. Но здесь все срабатывает, так как узнаваемость бренда Facebook настолько высока, что вы можете предположить, что когда люди заходят на эту страницу, они уже все знают о сервисе.

Интересная деталь в форме Facebook – это то, что пользователя повторно просят подтвердить свой e-mail-адрес, а не пароль. В этом есть свой смысл. Если вы забудете ваш пароль, ссылка на восстановление всегда может быть отправлена по почте. Но если ваш адрес некорректен, то повторная регистрация будет гораздо сложнее.

Формы – аккордеоны

Всякий раз, когда вы используете слишком длинную форму, возникает вопрос, а не разбить ли ее на несколько частей для удобства. Показ длинных форм отпугивает пользователя, но таков уж результат введения дополнительных шагов в процесс. Существует техника, способная решить эту проблему: форма-аккордеон.

Формы-аккордеоны состоят из нескольких разделов, каждый со своим подзаголовком. Все они располагаются на одной странице. Раздел, который пользователи заполняют, они видят. Остальные остаются скрытыми, показываются только их заголовки. Когда пользователь готов перейти к следующему разделу, он раскрывается, а остальные сворачиваются, отсюда и название.

The screenshot shows the Apple Store's Secure Checkout page. The page is divided into several sections:

- Items to be Shipped:** This section contains the 'Shipping Contact' and 'Shipping Address' forms. The 'Shipping Contact' form includes fields for First Name, Last Name, Area Code, Primary Phone, and Email Address (optional). The 'Shipping Address' form includes fields for Company Name (optional), Street Address, Apt, Suite, Bldg. (optional), Zip Code, and a checkbox for 'This is a business address'. A 'Continue' button is located at the bottom right of this section.
- Apple Shipping Policy:** A link to 'Learn more' is provided. The policy includes:
 - Signature is required for delivery
 - We do not ship to P.O. boxes
 - Delivery estimates below include item preparation and shipping time
 - We do not ship directly to APO/FPO addresses.
- Order Summary:** This section shows the Cart subtotal (\$1,199.00), Free Shipping (\$0.00), and Total (\$1,199.00). It also mentions '6 or 12 month special financing options' and a disclaimer about the terms of Apple's Sales and Refund Policy.
- Just Ask:** A section with a phone icon and the number 1-800-MY-APPLE.
- Frequently Asked Questions:** A section with links to 'How do I qualify for free shipping?', 'When will I get my items?', 'How do I track my shipment?', 'What if I will not be available to receive my shipment?', and 'How do I ship to an APO/FPO address?'.
- Shipping Method:** This section shows 'Standard Shipping — Free' and an 'Edit Shipping Method' link.
- MacBook Air, 11-inch:** This section shows the product image, price (\$1,199.00), quantity (1), and total (\$1,199.00). It also includes the part number (MC969LL/A) and a configuration list:
 - 1.6GHz Dual-Core Intel Core i5
 - 4GB 1333MHz DDR3 SDRAM
 - 128GB Flash Storage
 - Keyboard (English) & User's Guide
 - Accessory Kit

Рисунок 6.10. Компания Apple использует форму-аккордеон на их сайте для онлайн-покупки. Обратите внимание, что кнопка «Continue» («Продолжить») находится внутри каждого блока, а не внизу страницы

Насколько хорошо работает эта техника на практике? Etre, компания из Лондона, занимающаяся юзабилити, провела тестирование форм-аккордеонов^[67]. В нем приняли участие 24 простых пользователя в возрасте от 19 до 48 лет, с обычным опытом покупок через Интернет. Тестировались два варианта формы: с формой на нескольких страницах и с одной страницей, где все поля формы находится вместе. Один из вариантов предлагал пользователям нажать на следующий заголовок для того, чтобы развернуть его. В другом располагалась кнопка в конце каждого раздела.

Выяснилось то, что эти формы сильно не различаются по степени точности или удовлетворенности пользователей. Разница была только

в скорости их заполнения. Поразительно, но форма-аккордеон заполнялась даже быстрее, чем одна форма на всю страницу.

Это значит, что если ваш контент чувствителен ко времени, например если вы проводите аукцион, то форма-аккордеон может стать лучшим выбором.

Лучшим вариантом формы-аккордеона, заполняемой в процессе тестирования компанией Etre, стала та, где клавиша расположена внизу каждого раздела, а не та, в которой нужно нажимать на заголовок, чтобы развернуть раздел.

Когда люди заполняют формы, они ищут кнопку «Подтвердить».

А активизируемые мышью заголовки слишком далеки от привычек и ожиданий пользователей. Поэтому, если вы используете форму-аккордеон, убедитесь, что кнопка «Подтвердить» стоит внизу каждого раздела.


Гендерный вопрос

Иногда веб-сервису нужно знать пол пользователя, чтобы сделать доступными функции, которые зависят от этого. Ваши пользователи могут не захотеть представлять эту информацию и задумаются, зачем вам вообще это надо.

Еще хуже, если они заподозрят, что вы хотите продать ее рекламодателям. С другой стороны, фактическое использование нами этой информации может быть искренним и безвредным. Например, знание пола может нам помочь правильно формулировать статус в социальном приложении, например «Джон обновил свой профиль» или «Джейн загрузила новую фотографию в альбом Отдых».

Команда Bagcheck нашла отличный способ, как задать подобный вопрос. Вместо того чтобы спрашивать пользователя, мужчина он или женщина, им предлагается выбрать одно из притяжательных местоимений: «его», «ее» или «их». Этот метод проясняет, для чего вы хотите использовать эту информацию, и дает людям возможность выбрать нейтральное в плане пола «их».

Bagcheck



Create Your Account

Full Name

Your real name is required.

Location

Possessive Pronoun

☐ His ☐ Her ☒ Their

Your Email Address

Create a Password for Bagcheck

CREATE ACCOUNT

Рисунок 6.11. Гендерный вопрос в регистрационной форме Bagcheck

Хорошие значения по умолчанию

Когда пользователи вводят новые данные, может быть, было бы неплохо предоставить им стандартные значения по умолчанию. Например, магазин Etsy пытается догадаться о ваших региональных настройках, как только вы впервые заходите на его сайт. Вверху экрана высвечивается блок, и вам предлагают подтвердить ваш регион, который выбирается исходя из языковых настроек вашей системы. Ваш выбор влияет на такие вещи, как валюта, которая отражается в ценах за товар. Так как это предположение может сработать во многих случаях, Etsy понимает, что оно все же может быть ошибочным, и поэтому используется подтверждающее сообщение.

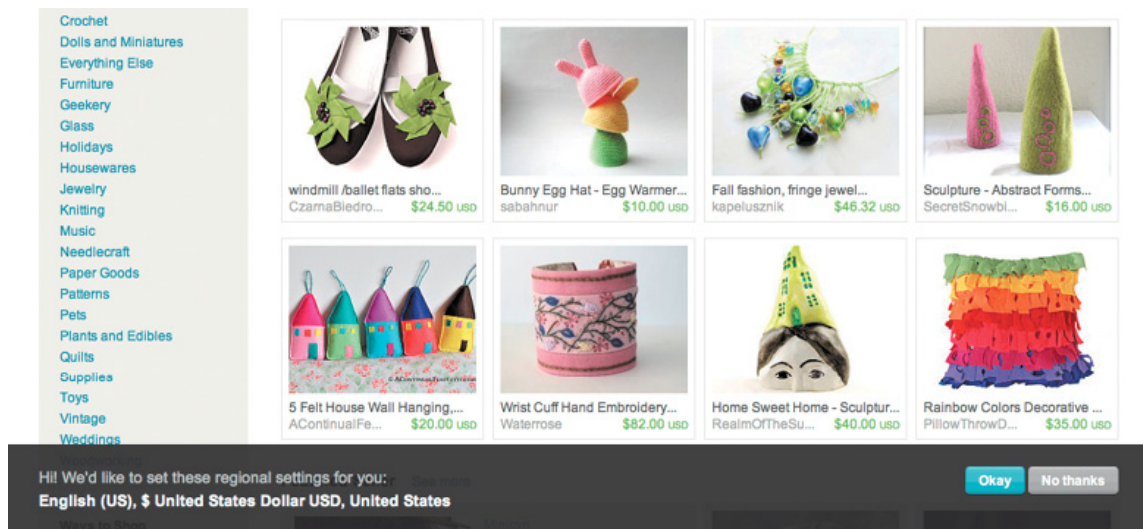


Рисунок 6.12. Etsy пытается догадаться о вашем местонахождении на основании языковых настроек вашей системы

Еще один хороший пример дает Homesite, посвященный страхованию жилья. Он содержит в своей форме некоторые средние значения. На странице для информации о собственности, в поля формы подставляются значения по умолчанию для размеров франшизы (нестрахуемого имущества), персональных обязательств и так далее.

Использование стандартных настроек таким способом действительно не везде. Например, при добавлении нового события в календарь, вы на самом деле хотите увидеть пустую форму, потому что все события разные. С другой стороны, если есть шаблон, то «умные» настройки по умолчанию могут стать хорошим способом упростить взаимодействие с пользователями.

Дизайн интерфейсов – это копирайтинг

Одним из наиболее важных аспектов любого визуального интерфейса является текст – это надписи на кнопках и формах, командных диалогах, заголовках, сообщениях об ошибке и т. п. Ваш текст – самый прямой способ общения с пользователем. И хотя часто советуют урезать текст до абсолютного минимума, потому что люди не читают, мы никогда не должны делать это в ущерб ясности. Кроме того, текст всегда должен быть написан для пользователей, а не для вас. Разница в том, что легко забыть, сколько времени вы проводите за

использованием вашего собственного продукта и что вы уже все знаете о нем.

Контекст может изменить значение слов. Институт Baymard провел исследование по удобству пользования и выяснилось, что слово «continue» («продолжить») в интерфейсе онлайн-магазинов сбивало с толку трех из десяти испытуемых^[68].

Почему так произошло? Все дело в контексте. Для тех, кто добавил товаров свою корзину и хочет перейти к заказу, «continue» означает «continue to checkout» («перейдите к заказу»). Для остальных, кто собирается выбрать еще другие товары, то же самое слово означает «продолжить выбор покупки». Всегда поясняйте значение надписи, если оно привязано к контексту.

Так, вместо «continue» используйте более описательное выражение, как «continueshopping» («продолжить покупку») или «continuetcheckout» («перейти к оформлению заказа»). Ясность лучше краткости.

Рисунок 6.13. Amazon использует кнопку Proceedtocheckout («Оформить заказ»)

Вот другой пример. У HubSpot была надпись на экране настроек домена: «Add domain» («Добавить домен») – что приводило в замешательство. Разработчики HubSpot имели в виду то, что клиенты могли пользоваться этим интерфейсом для добавления существующего домена к их учетной записи. А пользователи думали, что программа просит их создать новый домен, то, чего они не хотели делать, потому что у них уже был один.

Глагол «add» («добавить») сбивал всех с толку. HubSpot изменили надпись на «Connectyourdomain» («Подключите ваш домен») – и инцидент был исчерпан.

Рисунок 6.14. На сайте Target опускают глагол и просто используют слово «checkout» («оформление заказа»)

Первоначальное значение имело смысл для разработчиков HubSpot, использующих продукт, потому что они рассматривали проблему со своей колокольни. Однако их позиция не вызвала поддержки у клиентов. Пишите для тех, кто сталкивается с вашим продуктом впервые, и дважды проверьте, не имеет ли ваша надпись других значений.

Особое внимание деталям

Сегодня мы все привыкли к формам, в которых для подтверждения данных на ходу используется JavaScript. Square, новый платежный процессор, позволяющий вам использовать свой телефон как считыватель карт, пошел дальше и добавил другие динамические элементы, чтобы сделать взаимодействие с приложением более отлаженным.

В зависимости от того, данные какой карты вы заполняете (VISA, Master-Card и т. д.), используя приложение Square, иконка меняет свое изображение. Когда номер карты введен, все цифры, кроме последних четырех, исчезают. Потом иконка карты перевернется и подсветится область, где вы сможете ввести код проверки подлинности карты (CVV).

У Square очень много таких фишек. Когда вводите номер AmericanExpress, группировка цифр изменяется от 4 до 4, 6 и 5, отображая числовую структуру на карте, который вы пользуетесь. Так как CVV у AmericanExpress находится впереди, иконка карты не будет переворачиваться, а вместо этого покажет вам, где его найти на карте. Если вы вводите некорректное количество цифр кода, поле «вздрагнет» и станет красным. Это означает, что вы ошиблись. Также невозможно введение неправильного срока действия. Интерфейс просто не позволит вам этого сделать. Такое внимание к деталям приходится по душе клиентам. Это помогает им правильно заполнить форму и подсказывает направление, если затрудняются или ошиблись. Также берутся во внимание различные случаи использования, например расположение кода CVV на определенной карте может быть спереди, а не сзади.

Рисунок 6.15. Square скрывает полный номер карты и подсвечивает местонахождение CVV в иконке

Подобные случаи учитываются в реализации, для того чтобы у всех пользователей остались положительные впечатления от взаимодействия с продуктом.

Новые и экспериментальные техники

«На самом деле, сложно разрабатывать продукт фокус-группами. В большинстве своем люди не знают, чего они хотят, пока им это не покажут».

Стив Джобс

Эта цитата Стива Джобса хорошо созвучна со знаменитым высказыванием Генри Форда: «Если бы я спросил клиентов, чего они хотят, они бы ответили: более быструю лошадь». Инновации происходят, когда вы пробуете новое, а не заикливайтесь на старых способах решения проблем. Распространенные дизайны шаблонов и практический опыт показывают нам, что работает хорошо, но они не учат нас тому, как создать что-то лучшее. Слепое применение шаблонов, конечно, обеспечит вам получение надежного продукта. Но с другой стороны, вы будете плестись в хвосте тех компаний и дизайнеров, которые внедряют новаторские идеи в поисках лучших решений.

Иногда нам нужно нарушить правила и нормы, и поэкспериментировать. Например, во время бета-тестирования операционной системы Lion Apple пробовали внедрить новый дизайн для кнопок, в том числе для радиокнопок, где для выбора доступен лишь один элемент.

Рисунок 6.16. Эксперимент Apple со стилем кнопок представлен в виде селектора, похожего на слайдер

Рисунок 6.17. В конечном итоге Apple возвращается к более простому решению, с кнопками, похожими на кнопки, явно выделенному текущему пункту

К примеру, для переключения режима просмотра календаря на день, неделю, месяц или год. В новом дизайне подсвечивалась выбранная позиция, а остальные выглядели «утопленными». Задумка была такова, что кнопка могла передвигаться влево и вправо для выбора желаемого режима.

Однако это не сработало. Многие пользователи не восприняли это изменение на ура^[69], и Apple вернулась к прежнему дизайну, в котором кнопки реально выглядели как кнопки, а «утапливалась» выбранная в данный момент. Несмотря ни на что, эксперимент не провалился. Дизайнеры получили ценную информацию о том, какой сценарий лучше работает, и в конечном итоге было оставлено оптимальное решение. Если что-то идет вразрез с общепринятыми шаблонами, но помогает лучше решить проблему, то этот способ неизбежно станет рекомендуемым.

В этом разделе мы рассмотрим экспериментальные UX-техники. Может быть, они сработают для вашего продукта, может, нет. Но они подарят вам свежие идеи о том, как вывести дизайн на новый уровень.

Интерактивный сторителлинг

Занимательный рассказ всегда был мощным средством при представлении идей и продуктов. Браузер не ограничивается только показом текста и видео. Мы можем шагнуть дальше и создать интерактивные впечатления, позволяющие посетителям принять участие в истории, которую мы хотим поведать. Создание интерактивной истории процесс не из легких, но результат может увлечь пользователя гораздо глубже, чем привычное для нас пассивное поверхностное знакомство.

Когда NerdCommunications разрабатывал приложение «Ben the Bodyguard» («Бен-телохранитель»), то сделал сайт в виде интерактивной истории. Мы видим опускающуюся на город тьму и таящие угрозы улицы. В центре приложения представлен герой – Бен-телохранитель. Когда вы прокручиваете страницу вниз, он начинает двигаться с вами, а если вы прокрутите дальше, он раскроет идею приложения и то, для чего это вам нужно. Создателям приложения

удалось преобразовать ваш обычный маркированный список характеристик и преимуществ в интерактивные и захватывающие впечатления.

«Spent», сайт добровольных пожертвований Городского министерства Дарема, тоже демонстрирует это. Просить пожертвования нелегко, так почему бы не превратить этот процесс в игру? «Spent» – это веб-игра, в которой вы исполняете роль обычного безработного американца. Перед вами стоит задача – прожить один месяц без денег. В процессе вы должны искать работу, смотреть за детьми и пережить множество небольших кризисов. Успешным исходом считается заставить играющего проникнуться безработными и сделать пожертвования в конце игры.

Рисунок 6.18. Бен-телохранитель сопровождает вас на интерактивной прогулке по Злым улицам

Оба сайта не только прекрасно представляют соответствующие идеи и привлекают посетителей, но также являются отличными маркетинговыми средствами. И Бен-телохранитель, и «Spent» вызвали ажиотаж уже после первого выпуска. Люди рассказывали про эти сайты друг другу, потому что они были достаточно интересными сами по себе, даже если дело не заканчивалось покупкой приложения или пожертвованием. Сейчас, конечно, успех данных разработок в конечном итоге оценивается по факту покупки приложения или внесения пожертвований. Однако если вы знаете, что огромная часть вашей аудитории проводит много времени в социальных сетях, то сарафанное радио, привлекающее трафик, может работать на вас.

Рисунок 6.19. Дойдите до конца игры в «Spent». Без денег в течение месяца

Измеритель завершенности

Хотите, чтобы пользователи выполняли необязательные действия, допустим, добавляли больше информации в пользовательский профиль? Покажите им эти действия с помощью измерителя завершенности. Это что-то сродни индикатора выполнения, но в отличие от типичного индикатора, который показывает ход действия, измеритель завершенности сосредоточен на показе того, сколько еще требуется сделать. Он не должен быть индикатором, но должен иметь способ показывать прогресс. Если пользователю не по душе оставлять вещи недоделанными, он будет вынужден пойти по списку.

Например, у компании Klout из Сан-Франциско измеритель завершенности находится под заголовком «Connect» («Подключиться»). Это список дел в виде обычного текста, который предлагает вам выполнить различные действия, такие как подключить вашу учетную запись на Klout к сетям Twitter или Facebook, следовать за Klout в Twitter, поделиться вашими показателями на Klout и т. д. Как только вы выполняете какое-то действие, оно вычеркивается из списка. Другим примером может служить Groupon (американский сервис коллективных скидок). Его измеритель завершенности в боковой колонке – это список действий, такой же как у Klout, но с индикатором выполнения вверху, который заполняется по мере вычеркивания позиций. Почему это работает? Здесь на кону два психологических фактора. Во-первых, любопытство. Людям интересно узнать, что произойдет, когда измеритель достигнет 100 %. Будет ли какая-то награда в конце? Второй фактор – это цепочка обратной связи. Когда пользователь выполняет задачу, она отмечается, и показания измерителя растут. Так устанавливается четкая цель, и пользователи получают указания, что делать дальше.

Рисунок 6.20. Измеритель завершенности Klout приглашает пользователей быть активнее и вычеркнуть пункты из списка дел

Режим реального времени

Обновление в режиме реального времени всегда было возможным, но сегодня новые технологии еще больше упрощают этот процесс.

WebSockets (веб-сокеты) – свойство HTML5, которое позволит вам поддерживать интерактивный сеанс связи на сервере без его запроса и ожидания ответа через каждые несколько секунд. В действительности это намного упрощает работу в режиме реального времени, которая идет быстрее, а ваш сервер при этом не наводняется запросами.

Bagcheck использует веб-сокеты, чтобы обеспечить обновления счетчика посещаемости в режиме реального времени на каждой странице пользователя. Счетчик показывает количество просмотров и постоянно обновляет данные, когда кто-то еще открывает страницу. Bagcheck также отображает уведомления в реальном времени о том, что создает, лайкает или комментирует человек, на обновления которого вы подписаны.

Рисунок 6.21. Уведомления в режиме реального времени в Bagcheck

В приложении есть еще больше «штучек», работающих в режиме реального времени. Например, когда кто-нибудь дает комментарии по теме, они тотчас появляются.

Веб-сокеты открывают новые грани для экспериментов с веб-приложениями и сайтами. Аналитика и игры в реальном времени – это явное использование, но, как показывает Bagcheck, режим реального времени может применяться также к элементам интерфейса, которые мы привыкли расценивать как статические, таким как список комментариев.

Но то, что мы можем обеспечить этот режим, не значит, что мы должны это делать. Существует явный риск того, что пользователи, которые не хотят изменений на странице, расстроятся.

Селектор страны

Селекторы страны представляют собой выпадающее меню, открывающее на странице длинный список стран, который вы будете прокручивать, чтобы выбрать нужную вам. Можем ли мы придумать

что-то получше? Да. Кристиану Холсту и Джейми Эплсиду из института Baymard удалось переделать селектор страны, чтобы сделать процесс более удобным[70]. Идея такова – использовать живое поисковое поле для выбора страны. Так если пользователь начинает печатать название, включается поиск и выпадает список совпадений. Верхний вариант будет выделен, поэтому пользователь просто должен нажать клавишу Enter, и этот вариант окажется в поле. Пользователь уже знает, чего он хочет, поэтому впечатывание первых букв из названия страны намного быстрее, чем прокручивать список, в котором может стоять тысяча позиций.

Рисунок 6.22. Селектор страны в действии

Эта реализация работает, так как она решает три основных проблемы. Во-первых, вы должны учитывать опечатки и последовательность; пользователь может печатать слова в неправильном порядке или неверно по буквам. Во-вторых, некоторые страны имеют не одно название. Например, Нидерланды – это Голландия. Поэтому люди, которые пишут «Голландия», должны быть уверены, что получат правильное сочетание. То же касается и Америки, когда это название пишут вместо Соединенных Штатов.

И, наконец, некоторые страны более распространены среди ваших пользователей. Поэтому, если кто-то начинает впечатывать «United» (Соединенный, объединенный), первым сочетанием лучше сделать «United States» (Соединенные Штаты), а не «United Arab Emirates» (Объединенные Арабские Эмираты). Если только, конечно, большинство ваших пользователей не из ОАЭ.

Живое текстовое окно поиска может применяться и для других вещей, но три основных вопроса, которые я только что осветил, должны оставаться во главе угла. Таким образом, вы сможете улучшить вашу разработку.

Прогрессивный вход в систему

Сегодня многие сайты используют посредников, для того чтобы авторизовать пользователя. Люди, у которых уже есть учетная запись на Google или Facebook либо на любом другом сервисе, который позволяет идентифицировать и авторизовать их, могут разрешить доступ к данным авторизации и на других сайтах.

Подчас несколько логин-провайдеров используются одновременно, а это может привести к тому, что люди забудут, какой из них они использовали для регистрации.

Команда сайта Bagcheck провела эксперимент с прогрессивным входом в систему, который решает эту проблему. Вместо обычной логин-формы с двумя полями для имени пользователя и пароля, а также кнопками посредников, Bagcheck показывает отдельное текстовое поле для введения имени (или e-mail-адреса). Когда пользователь начинает печатать, в режиме живого поиска ищется его учетная запись. Когда она найдена, показываются дальнейшие логин-опции, в том числе кнопка посредника, через которого пользователь раньше зарегистрировался на Bagcheck.

Такой метод устраняет проблему того, что люди забудут, через какой аккаунт они зарегистрировались на сайте раньше. Однако есть две оговорки. Во-первых, пользователи больше не могут регистрироваться одним кликом, что они могли делать, когда логин-опции показывались сразу же. Во-вторых, живой поиск выдает других людей, которые зарегистрированы на Bagcheck. Так как учетные записи этого сайта по природе сервиса публичные, показ других учетных записей здесь норма. Однако это не подходит для тех сервисов, где они являются приватными.

Рисунок 6.23. Введение вашего имени на странице авторизации запускает поиск пользователей на сайте Bagcheck

Рисунок 6.24. Как только вы найдете свою учетную запись, вы увидите опции, которые можете использовать при авторизации

Отзывчивый мобильный интерфейс

При использовании мобильных интерфейсов, базирующихся на интернет-технологиях, всплывает ряд проблем. Управление кончиком пальца не настолько точно, как курсором мыши. Поэтому часто неясно, на ту вы нажали ссылку или нет. Скорость соединения медленная, и сложно понять, загружается страница или «зависла». Если данные в кэше, неясно, будут ли они обновляться и, если да, то когда. А это проблема, если пользователь ищет новый контент.

Когда компания 37signals приступила к разработке мобильной версии своего приложения управления проектами Basecamp, она пыталась найти пути решения этих проблем и создать интерфейс, который превзошел бы исходный вариант. Для управления касаниями 37signals убедилась, что все элементы имеют состояние «выбрано». Так, чтобы при прикосновении клавиша немедленно подсвечивалась, и пользователь мог узнать, что его выбор зарегистрирован, и он нажал верно.

Когда приложение загружается впервые, оно извлекает много данных, которые кэшируются в устройство. Это может занять некоторое время, поэтому компания создала специальный экран загрузки для первого запуска. Этот экран отсчитывает время, которое идет на загрузку, и когда оно переходит определенный рубеж, внизу «всплывает» сообщение, из которого пользователь узнает, что приложение загружается. Если оно грузится очень долго, то другое сообщение предлагает пользователю повторить попытку или перейти на версию в компьютере.

Из-за того что приложение использует много кэша, людям нужно знать, отражает ли представляемый контент актуальные данные. Вместо обновления всей страницы при проверке нового контента приложение показывает вращающуюся пиктограмму загрузки («спинер») в правом верхнем углу, которая говорит пользователю о

том, что Basecamr соединяется с сервером и позволяет ему (пользователю) продолжить работу.

Рисунок 6.25. Если начальная загрузка длится дольше положенного, появится сообщение, которое известит вас о том, что приложение все еще работает (грузится)

Меню пиктограмм

Пиктограммы предназначены не только для приложений. Вы можете также использовать их в меню выбора продукции на сайтах электронной торговли. Например, магазин по продаже винтажных очков Bonlook использует пиктограммы в своем выпадающем меню и дает пользователю возможность пролистать его по формам, которые им нравятся. Формы очков нелегко описать словами, но изображения представляют их сразу же.

Рисунок 6.26. Скромная вращающаяся пиктограмма в правом верхнем углу, которая говорит пользователю о том, что приложение проверяет обновления на сервере

Другой пример – шведский магазин обуви Nerro.se. Здесь вы снова видите выпадающее меню, где представлены различные модели туфель и сапог, которые вы можете купить.

Каждая позиция в меню сопровождается пиктограммой, изображающей форму обуви. В магазине также продаются шарфы, шляпы, щетки и другой товар. Все это тоже представляется в маленьких изображениях. Использование пиктограмм позволяет посетителям пользоваться магазином, даже если они не знают шведского языка. Такой вид интерфейса действительно работает, когда товар представлен во многих вариантах, таких как разные модели обуви, потому что покупатель имеет возможность просматривать и обрабатывать свой выбор тут же.

Рисунок 6.27. Селектор Bonlook для очков

Рисунок 6.28. Нерро.se показывает пиктограммы каждой модели обуви, которую они продают

Работа с клиентами – это опыт взаимодействия

Тони Шей создал электронному магазину Zappos историю успеха. С момента его присоединения в 2000 году к Zappos в роли исполнительного директора, годовой доход компании ежегодно удваивается. В 2009 году он достиг миллиарда долларов. Zappos – это обувной онлайн-магазин. Но самое интересное то, что Шей совсем не интересовала обувь. Что его на самом деле привлекало – так это работа с клиентами, а страсть к созданию у них сильных впечатлений принесла реальный успех компании.

«Мы задали себе вопрос, чего мы ждем от этой компании. Мы не хотели просто продавать обувь. Меня не интересовала обувь. Я был страстно увлечен работой с клиентами».

Тони Шей, основатель Zappos

Когда мы говорим об опыте взаимодействия, то обычно думаем о пользовательских интерфейсах. Но пользователи не просто взаимодействуют с интерфейсом: они читают вашу документацию и общаются с технической поддержкой.

Все это вносит вклад в общую картину, а гарантия того, что все будет хорошо, делает ваших клиентов счастливыми.

Ниже вы найдете идеи по поводу того, как улучшить работу с клиентами, начиная с вашей поддержки и заканчивая представлением лучшей документации.

Поддержка смайликом

Как повысить качество поддержки, которую предоставляет компания, с помощью визуализации? 37signals разработала рейтинговую систему, когда в конце каждого обращения по электронной почте люди оценивают поддержку, которую они получили. Рейтинговая система имеет три ссылки: одна для «great» («отлично»), другая для «just OK» («хорошо») и последняя «not so good» («не очень хорошо»). Каждая оценка сопровождается смайликом, для того чтобы выбор можно было сделать быстрее. Смайлики раскрашены под цвета светофора: зеленый для «отлично», желтый для «хорошо» и красный для «плохо».

Рисунок 6.29. 37signals представляет оценки своей службы поддержки на «Страничке смайликов»

Как только покупатель подтверждает оценку, его просят предоставить дополнительную информацию. Это необязательно, но это прекрасный способ собрать отзывы для того, чтобы использовать их для улучшения поддержки.

Так как клиенты оценивают отклики и действия, которые они получают на каждый свой запрос о поддержке, 37signals собирает данные о том, насколько хорошо работает компания, и даже отслеживает рейтинг отдельных сотрудников, чтобы посмотреть, не хромает ли у кого-то качество обслуживания. Это можно визуализировать, собрав все эти смайлики в ряд, получится большое изображение, ну, например, из ста последних оценок.

37signals также использует эту систему как мощное орудие маркетинга.

Страница создана для отображения 100 последних рейтингов, визуализированная с помощью смайликов, раскрашенных под цвет светофора. Таким образом, преобладание зеленого на большой стене из «улыбочек» натолкнет покупателя на мысль, что поддержка со стороны 37signals действительно хорошая.

Документация тоже впечатляет

Документация часто является слабым звеном во впечатлениях пользователя. На самом деле писать неувлекательно и читать незанимательно. Но так не должно быть. Команда MailChimp (сервис e-mail-рассылок) разработала отличный способ создания документации. Сначала даются основы, описываются основные функции. Потом каждый раз при получении запроса о поддержке компания отписывает основательные, детальные ответы. После отправки ответа клиенту MailChimp сохраняет его в базе знаний. Таким образом, потребитель получает отличный сервис, а затраченное время также идет на создание исчерпывающего ресурса базы знаний, которым он (потребитель) может пользоваться.

Чтобы убедиться в точности составления документации, создавайте ее в процессе. Это даст вам возможность получить кучу скриншотов, которые помогут быстрее объяснить шаги. Документация также не должна быть написана скучным языком, поэтому, где возможно, делайте ее веселой. Тогда она не станет утомительной для чтения. На одной из страниц, где объясняется, как настроить временные зоны, MailChimp разместила фото настенных часов, расположенных над настенным рисунком Чимпзиллы, разрушающего городские строения. Эта игривость делает интерактивное общение более человечным и интересным.

Однако будьте осторожны! Шутки проходят только тогда, когда читатели их воспринимают. При устранении серьезных проблем читатель, скорее всего, будет напряжен, и юмор на подобных страницах едва ли уместен.

Займитесь этими вопросами профессионально и заботливо. Если страницы не критичные, то возьмите пример с MailChimp и принесите немного веселья, чтобы разрушить утомительность скучных задач.

Рисунок 6.30. Картинка со страницы справки о временных зонах

Предоставление кредита

Райан Карсон на ThinkVitamin ратует за предоставление кредита пользователям, когда что-то идет не так. Если пользователи сталкиваются с какой-то проблемой на сервере, Райан компенсирует им убытки кредитованием их аккаунта (например, предлагает период бесплатной подписки). Это необычное решение проблемы. Клиенту показывают, что если что-то пойдет не так, ThinkVitamin возьмет на себя полную ответственность. Такая тактика отлично работает в пользу ThinkVitamin. Люди не привыкли к всеобъемлющему клиентскому сервису. Поэтому когда они соприкоснутся с этим, их благодарность не будет знать предела. И не только. Они также захотят поделиться своими впечатлениями. Предоставляя клиенту больше того, за что он заплатил, и вы приобретете больше, чем потеряете, и к тому же сделаете людей счастливее, терпимее. Они будут готовы рекомендовать ваш сервис всем друзьям.

«Ответьте, пожалуйста!»

Любое взаимодействие с вашими клиентами – это возможность для всестороннего контакта, укрепляющего ваши взаимоотношения. Удивительно, но многие компании сегодня все еще пользуются «безответными» почтовыми адресами (no-reply@sitename.com). Но, в конце концов, лучше показать вашим клиентам, что вас волнует их мнение, чем сказать о том, что им не нужно отвечать вам, так ведь?

Однако дела обстоят намного хуже. «Безответные» e-mail-адреса могут негативно отразиться на показателе доставки писем. Например, Google's Priority Inbox мониторит адреса, на которые отвечает большинство людей, и отмечает их как важные. Вполне вероятно, что их спам-алгоритм также учитывает ответы. Ведь если вы шлете ответ на почту, скорее всего, адрес подлинный, и это не спам.

Что же делать? А вот что! Превратите проблемный адрес «no-reply» («не отвечать») в «please-reply» («пожалуйста, ответьте»). Если у вас есть система техподдержки пользователей, вы можете переправлять ответы, которые получаете, туда. А потом обрабатывать их как обычно.

Рисунок 6.31. Современная система управления платежами обычно имеет опцию выдачи кредита на активный клиентский счет. Представленный пример – кредитная страница платежного сервиса Recurly

Вам также нужно установить фильтры для разграничения подлинных ответов, автоматических ответов (например, «я не в офисе») и сообщений о неудавшейся доставке. Вы захотите избавиться от автоматизированного хлама, чтобы не увязнуть в нем. Если в вашей системе еще нет фильтров, Gmail предлагает вам хорошие возможности фильтрации, а также сильную защиту от спама. Прежде всего, не забывайте использовать эту возможность для построения более крепких отношений с вашими клиентами, когда они дают ответ.

Поощрение преданных пользователей

При сбыте вашей продукции совсем необязательно выбрасывать кучу денег на рекламу. Лучшие ваши продавцы – это ваши страстно преданные клиенты. Это те люди, которые пользуются ею каждый день и знают, какая она классная. Рекомендация от друга не имеет под собой коммерческой основы, что делает ее гораздо более эффективной, чем оплата рекламы.

Рисунок 6.32. Компания Dropbox нашла способы простого обращения к людям через почту и соцсети

Раньше Dropbox оплачивала рекламу[71], но потом пришла к выводу, что слишком много денег тратится на привлечение клиентов, а вложения не возвращаются в доход. Компания пересмотрела свою стратегию и сосредоточилась на маркетинге, ориентированном на пользователя (customer-driven marketing). Новый подход заключался в поощрении преданных клиентов за рекомендации продукта своим друзьям. Dropbox стимулировал обе стороны: рекомендатели получали дополнительное бесплатное хранилище, а их друзья – большее пространство памяти, чем могли бы иметь.

Стимулирование лояльных пользователей к приглашению своих друзей и поощрение их принесло успех компании Dropbox. Еще одно преимущество этого метода по сравнению с обычным маркетингом в том, что вы приобретаете клиентов через сам продукт, а не посредством денег. Повышается ценность вашего продукта для людей, которых вы поощряете, и устраняет трудности процесса оплаты.

Спасибо!

Как часто вы благодарите своих покупателей за то, что они выбрали ваш продукт или сервис? Типичный ответ, который вы получите от большинства компаний, это автоматизированное сообщение по почте. Придумать что-то особенное не так уж и сложно. Почему бы не порадовать вашего клиента письменной благодарственной открыткой? Не по электронке, а именно настоящей! Этот небольшой жест не будет вам стоить ровным счетом ничего, но вашим покупателям будет приятно, а ваш бренд действительно выделится.

Рисунок 6.33. Благодарственная открытка от Wufoo

Так действует со своими главными покупателями компания Wufoo. Благодарственную открытку пишет каждый из сотрудников. Для ускорения процесса в него вовлекаются все. По пятницам, члены команды пишут примерно по 10 открыток людям, которые помогли их компании добиться успеха.

Ваши клиенты не ожидают, что вы будете прилагать столько усилий, чтобы сказать спасибо. Поэтому подобные открытки станут для них приятным сюрпризом.

Анти-UX: темные модели

Обычно плохой дизайн – это результат некомпетентности. Так или иначе бывают случаи, когда плохой дизайн создается преднамеренно. Цель его – таким обманчивым способом заставить пользователя сделать определенные вещи для выгоды разработчика. Эта техника получила

название «темные модели. Мы приведем здесь несколько примеров. Избегайте таких моделей в вашей работе, потому что они переходят грань между процессом убеждения и обманом. Введенный в заблуждение пользователь не стимулирует жизненную и долгосрочную стратегию ни с этических, ни с деловых соображений.

В примере ниже представлена типичная темная модель, которую использовали рекламщики на румынском сайте Softpedia, – замаскированная реклама. Softpedia показывает ссылку на загрузку файлов, однако над ней и слева расположены рекламные баннеры. Каждое изображение баннера выглядит как загружаемая ссылка с оценочными звездочками. Пользователь, который только что зашел на страницу и ищет ссылку к загрузке, может попасть в ловушку и нажать на одну из двух «подделок», представленных рекламой.

Рисунок 6.34. Замаскированные объявления на сайте Softpedia

Вот другой пример темной модели: вопрос-ловушка. Регистрационная страница магазина Wired использовала внизу страницы несколько окошек для ответов на вопросы, хотите ли вы получать предложения от компании и ее партнеров. Всего было 8 окошек. Зачем так много? Wired чередовал «согласие на рассылку» и «отказ от рассылки» на каждой строке, «танцуя джигу дважды». Компания разграничивала предложения по телефону и по почте, Wired и партнеров и имела отдельные галочки для отказа. Потом она разделила все пополам, на четыре более удобные кнопки.

Рисунок 6.35. Wired пытается заставить вас зарегистрироваться на их список рассылки 8 различными галочками

Еще один пример темной модели: вынужденное продолжение. Используется, чтобы заставить клиентов зарегистрироваться на получение периодической подписки. Например, вы зарегистрировались для получения бесплатного пробного продукта, а

потом увидели, что ваш счет автоматически превратился в схему оплаты, где срок пользования пробным продуктом заканчивается без предварительного предупреждения. Когда-то интернет-магазин Audible пробовал протолкнуть свой тарифный план без четкого объяснения ежемесячной платы. В окошке просто отражалась цена. Поэтому покупатели были уверены, что это одноразовая оплата. Каково же было их удивление, когда им нужно было платить по карте на следующий месяц. К счастью, компания Audible прислушалась к жалобам и внесла изменения в процесс для большей ясности.

Советуем вам сторониться таких моделей. Иногда вы можете не заметить, как вплотную подойдете к грани между убеждением и обманом. Но вы всегда должны знать, когда переходите ее. Использование техники дизайна как ловушки для пользователя, чтобы заставить его делать что-то, обычно означает, что он не понимает, что происходит. Это выражается в том, что пользователь не видит скрытых взносов, нажимает не на то окошко или замаскированное объявление и т. п. Дело в том, что пользователи совершают действия, которые они ни за что бы не сделали, владея они достаточной информацией. Темные модели всегда стараются скрыть ее. Но если вы применяете их намеренно, то должны также знать, что вы переходите грань между убеждением и обманом. Обман своих пользователей и клиентов никогда не был и не будет хорошей долгосрочной стратегией.

Рисунок 6.36. Audible не сделал прозрачной ежемесячную оплату

Значимость хорошего дизайна

24 января 2012 американская корпорация Apple опубликовала свои финансовые результаты за первый квартал финансового года: рекордный доход составил 46,33 миллиардов долларов. Это один из самых высоких показателей за квартал. Секрет успеха Apple в том, что она сосредоточена на исключительном дизайне продукта. Маркетинг эффективен, но он не будет работать без сильного продукта. Также на продажи влияет и культура. Если люди, работающие там, не любили

бы продукт, который они создают, он не продавался бы с таким размахом.

Вот история, наглядно демонстрирующая, как был важен дизайн продукта для Apple в период ее возрождения Стивом Джобсом. Джонатан Айв, главный дизайнер корпорации, хотел добавить ручку в полупрозрачный компьютер iMac, над которым он работал. Он знал, что с технологией люди чувствуют себя не в своей тарелке и боятся даже дотронуться до компьютера. Внедрение ручки дало бы пользователям возможность прикасаться к нему и сделало бы его более доступным.

Создание ручки вылилось бы в кругленькую сумму, и, естественно, Айв столкнулся с сильным сопротивлением со стороны других руководителей и инженеров. Они хотели знать, как окупится внедрение ручки и каков будет возврат на инвестиции (ROI). Когда Стив Джобс увидел ручку, он сразу же понял ее предназначение и дал изобретению зеленый свет. Он отверг все возражения и разослал своим сотрудникам разъяснение, что не анализ стоимости и подсчет ROI, а дизайн сильного продукта всегда был важной вехой в развитии корпорации Apple.

Спустя годы мы сможем увидеть, что его стратегия окупилась не только с позиций создания продукта и бренда, который понравился людям, а с тех, что этот продукт был создан самой прибыльной компанией в мире. Если вы сомневаетесь в том, стоит ли вам тратить дополнительное время и усилия на создание продукта, которым с удовольствием будете пользоваться вы и ваши клиенты, вспомните историю с ручкой iMac. Ее ценность может быть ясна не сразу, но сильный дизайн и акцент на деталях в конечном счете окупятся и помогут вам создать сильный бренд, приобрести преданных клиентов и продукт, к которому вы «прикипите».

Об авторе

* * *

Дмитрий Фадеев – основатель блога UsabilityPost, где он размещает свои мысли о хорошем дизайне и представляет интересные техники дизайна интерфейса. Он работал веб-дизайнером на фрилансе в течение нескольких лет, его последний примечательный проект – проработка UX для StackExchangenetwork (сеть сайтов для работы с вопросами и ответами в различных областях). Его последний проект – веб-приложение с названием Usauga, которое помогает дизайнерам проводить микротестирование удобства пользования с использованием скриншотов их интерфейсов и прототипов.

О рецензенте

* * *

Джошуа Потер – дизайнер интерфейсов и основатель Bokardo Design, где он сосредотачивается на дизайне социальных веб-приложений. «Помешан» на вебе вот уже в течение десятка лет, Джошуа создает простые, удобные интерфейсы для различных клиентов: от «стартапов» до гигантов. Также он консультирует компании, страдающие от тяжелых случаев деформации функционала и тех, кому нужен объективный совет. Джошуа написал книгу «Разработка для социальной сети», и он регулярно выступает на конференциях по веб-дизайну и других мероприятиях по всему миру. С 2003 года он пишет в популярный дизайн-блог bokardo.com, который достаточно хорошо известен тем, что либо делает вопросы дизайна доступными для понимания, либо усложняет их дальше некуда. Когда Джошуа не занят разработками, не консультирует и не пишет, он занимается скалолазанием или пытается заниматься своим ребенком.

Дизайн будущего: использование Photoshop

Автор: Марк Эдвардс

Рецензент: Джон Хикс

Как всякая развивающаяся платформа, Интернет претерпел существенные изменения с момента своего рождения. Он продолжил свое развитие и стал доступным на многих устройствах и многим людям, где бы они ни находились. Хороший дизайн, который раньше зачастую игнорировался, сегодня является обязательной составляющей успеха.

Граница между настольным, мобильным ПО и Интернетом продолжает размываться, так как дизайнеры и разработчики комбинируют то, что привычно, с тем, что под рукой и лучше всего подходит к ситуации. Средства, техники и требования продолжают изменяться.

Сегодня «родное» настольное и «родное» мобильное ПО часто содержат в себе HTML, CSS, JavaScript и другие языки – методы и техники, которые были разработаны для Интернета. А так как сайты и веб-приложения разрастаются и в свойствах, и в размерах, для дизайнеров стало привычным использовать техники, которые разрабатывались с учетом «родного» программного обеспечения. Базовые код и технологии могут представлять интерес для дизайнера, а могут и нет. Но они же могут ограничивать возможности и дизайнерские ресурсы, например изображения. К счастью, проблемы дизайна «родных» приложений и веб-приложений одинаковы, поэтому многие решения применяются в обоих случаях.

Давайте рассмотрим проблемы, с которыми дизайнеры сталкиваются сегодня и столкнутся в будущем. Потом мы обратимся к способам их решения, используя одно из наиболее распространенных доступных средств – Adobe.

Размеры экрана

С бурным ростом мобильных устройств правила игры изменились. Когда вы создаете сайт, то не можете предположить, на каком из устройств он будет просматриваться, каков будет размер экрана или разрешение. Мобильные и настольные приложения распространены немного больше, но дизайнерам нужно думать и о том, как их разработки будут выглядеть на разных устройствах.

Кажется почти неизбежным, что пост-ПК-устройства превзойдут по численности настольные компьютеры и ноутбуки.

Сегодня все важнее учитывать то, где появится ваш сайт: на 4– или 10-дюймовом мобильном экране, на 15-дюймовом ноутбуке или 27-дюймовом экране настольного компьютера.

Плотность пикселей

Пиксели – квадратные элементы, из которых состоит все изображение на дисплее компьютера – уменьшаются и это позволяет документу быть напечатанным более четко. Это быстро приближает нас к пределу желаемого.

Плотность пикселей обычно измеряется в пикселях на дюйм (PPI). Разрешение пикселя в 100 PPI означает, что в 1-дюймовой горизонтальной строке содержится 100 пикселей. Площадь в 1x1 дюйм на дисплее 100-PPI содержит 100×100 (или 10 000) пикселей. Звучит несколько сухоовато, но этот параметр жизненно важен, когда решается, насколько крупными должны быть элементы вашего дизайна и как они будут масштабироваться. Давайте рассмотрим примеры из жизни.

iPhone 3GS имеет дисплей шириной в 320 пикселей при высоте 480 пикселей. Экран с 3,5-дюймовой диагональю дает пиксельное плотность 163 PPI. Для сравнения, дисплей iPhone 4 Retina имеет точно такой же физический размер, диагональ в 3,5 дюйма, но его разрешение – 640 пикселей в ширину и 960 пикселей в высоту – это практически удвоенный экран iPhone 3GS. Пиксели на iPhone 4 ровно наполовину меньше в ширину и длину, соответственно пиксельное разрешение составляет 326 PPI. Другой способ взглянуть на это: для

каждого пикселя на iPhone 3GS существует 4 пикселя на 2×2 решетке на iPhone 4.

Рисунок 7.1. Разрешение и размер экрана iMac и некоторых мобильных устройств

С того момента как был разработан первый дисплей, появилась тенденция к увеличению пиксельного разрешения. Скорее всего, она продлится до тех пор, пока все дисплеи или их большинство не будет иметь разрешение от 200 до 350 PPI. Причина этому – существующий предел, при котором человеческий взгляд уже не может различать отдельные пиксели (несмотря на то что точная величина PPI для этого порога будет варьироваться в зависимости от того, насколько близко вы находитесь от экрана, какое у вас зрение и когда вы выпили последнюю чашечку кофе). Мы поговорим о том, как создавать Photoshop-документы так, чтобы они масштабировались равномерно для всех графических целей, требуемых для Web, iOS, Android, Windows Metro и других платформ.

Местоположение

Раньше сайты и приложения просматривались только на настольных компьютерах. Сегодня вы не можете предугадать, будут ли смотреть ваш дизайн на широком экране в офисе, кабинете или холле. Если ваше мобильное приложение предназначено для поиска отличных местных кафешек, то есть большая вероятность, что им будут пользоваться при ярком дневном свете в солнечный денек, и пользователь будет искоса поглядывать на отражение облаков в экране своего телефона. Жизненно важно создать правильный контраст и проверить его на разных устройствах по обычным сценариям. О нескольких методах тестирования дизайна на устройствах мы расскажем вам в конце этого раздела.

Реализм

Дисплеи с высокой плотностью пикселей, мультитачем и ускоренной графической обработкой данных дают прекрасные возможности для того, чтобы создать более реалистичный дизайн и анимацию для интерфейса пользователя (UI). Сейчас возможны великолепные дизайнерские решения с тонкими штриховками, затемнениями и текстурами. Можно добавлять декоративный дизайн, известный как скеоморфизм. Своим видом он демонстрирует, как нужно взаимодействовать с контентом. Мы можем увидеть открытую книгу. Край у одной из ее страниц загнут и показывает, что можно листать их справа налево. В приложении для игры на синтезаторе можно виртуально переподключать провода, для того, чтобы изменить выход аудиосигнала.

Рисунок 7.2. Приложение Early Edition 2 для чтения rss-лент от компании Glasshouse Apps

Рисунок 7.3. Ностальгический дизайн от GlasshouseApps. Функция «расшаривания» в программе Early Edition сделана в виде виртуального конверта

Такой дизайн необязателен. Для того чтобы приложение функционировало, не нужны отогнутые страницы или нарисованные кабели. Скеоморфизм больше похож на некое связующее звено между реальным и цифровым мирами. При правильном применении эти «украшения» должны сделать дизайн более удобным, потому что они помогают объяснять свойства привычным способом.

Однако будьте осторожны! Добавляя такие вещи в оформление, помните, что пользователи ждут от вас исполнения заложенных в них обещаний. Если страница выглядит перелистываемой, значит, она должна и быть такой. Некоторые связи/анalogии с реальным миром могут быть также слишком скучными. Если ваши пользователи молоды, они могут и не знать, что такое виниловая пластинка, 3,5-дюймовая дискета или вращающийся каталог Rolodex[72].

Закладываем масштабируемость

Как мы говорили, некоторые устройства уже имеют дисплей с высокой пиксельной плотностью. И мы видим, что выпускаются мобильные устройства с еще более высоким PPI. Весьма вероятно, что мы увидим настольные ПК и ноутбуки с высоким PPI [\[73\]](#) – Windows, Mac OS X и Сеть в целом будут следовать одной тенденции – мобильности.

Мы находимся на переломном этапе. Старые экраны с низким PPI еще будут использоваться какое-то время, поэтому нужно поддерживать экраны и с высоким, и с низким PPI. Разные платформы осуществляют этот процесс с небольшой разницей. Но, как правило, вам потребуется полный комплект изображений для каждой пиксельной плотности, с которой собираетесь работать.

iOS

Для разработки под iOS компания Apple подобрала две дисплейные плотности и поэтому два масштаба пользовательского интерфейса (UI). Новейшие дисплеи имеют в точности удвоенное пиксельное разрешение от устройств раннего поколения. Для iPhone его значение 163 PPI для ранее выпущенных моделей. А для дисплеев iPhone 4 Retina и последующих моделей – это 326 PPI.

Для масштабирования это идеальное решение, потому что все, что создано с правильными техниками для малого размера, будет отлично масштабироваться на дисплее Retina. Однако вам понадобятся два полных комплекта изображений: один для дисплеев не-Retina, другой для дисплеев Retina. Apple добавляет @2x к названиям файлов с изображениями для дисплеев Retina. Поэтому myimage.png должно выглядеть как myimage@2x.png.

Таким образом, если ваши первоначальные файлы не для размеров дисплея Retina, то Retina они должны быть отмасштабированы до 200 %.

Android

Операционная система Android схожа с iOS, за исключением того, что она имеет четыре пиксельных плотности вместо двух. Это потому что Android используется в огромном спектре устройств. Android представляет следующие разрешения:

- 120 PPI (низкая плотность),
- 160 PPI (средняя плотность,
- 240 PPI (высокая плотность),
- 320 PPI (сверхвысокая плотность).

Интерфейсы для всех устройств с системой Android масштабируются на основе одного из этих разрешений. Чтобы поддерживать все четыре вам понадобится полный комплект PNG изображений для каждого. Android устройства с низким разрешением – явление нечастое. Поэтому вы, скорее всего, станете поддерживать только три другие плотности. Если изначально ваш дизайн настроен на 160 PPI, потом вам нужно будет изменять масштаб до 100, 150 и 200 %.

Windows Metro

Подобно системе Android, Windows Metro была разработана для размещения в широком спектре устройств, так что здесь тоже понадобится большое количество изображений. Исходники для Windows Metro создаются под 100, 140 и 180 %, если только не используется масштабируемая векторная графика (формат SVG).

Mac OS X

Хотя и непровозглашенная, система MacOS X очень похожа на модель для iOS и работает с дисплеями не-Retina и Retina с настройками изображений в 100 и 200 %. Где это уместно, можно использовать для интерфейсных элементов Mac OS X PDF изображения, где в одном файле находятся два размера.

Сайты и веб-приложения

Немного неясно, как будут обрабатываться дисплейные пиксельные плотности в вебе в дальнейшем, но применяемые методы, скорее всего, будут очень похожи на методы iOS, Android, Windows Metro и других «родных» платформ.

Некоторые сайты уже включают множество изображений, базирующихся на PPI дисплее, похожем на iOS, Android и Windows Metro. Другие дизайнеры пытаются рисовать все с помощью кода, используя CSS и SVG изображения или встраивая пиктограммы и глифы в шрифтах, поэтому и наборы изображений не требуется.

Рисунок 7.4. Избегайте растровых изображений и всегда используйте векторные формы и эффекты

Очевидно одно: так как дисплеи с высоким PPI становятся более распространенными, придется усовершенствовать разные методы для веба. Любой дизайн будущего нужно создавать так, чтобы можно было изменять масштаб во многих размерах. Итак, как же мы можем добиться этого с помощью Photoshop?

Масштабируемые документы Photoshop

Когда дело дойдет до создания элементов, которые легко масштабировать в Photoshop, избегайте растровых изображений. Растровые изображения по своей природе – это сетки с квадратными элементами изображения (пикселями). Это означает, что нельзя добавлять детали, когда растровое изображение расширяется, потому что дополнительных деталей нет (дополнительные пиксели можно было бы интерполировать из соседних, но это привело бы к размытости).

Также нельзя уменьшать масштаб элементов. Это приводит к явным масштабным дефектам изображения и, отсюда, к созданию низкопробной графики.

Решение тут таково – создавать все, применяя векторные формы и эффекты, которые можно будет восстановить в любом размере. В Photoshop это значит использование однотонных, градиентных слоев с паттернами с векторными масками и стилями слоев. Это позволяет увеличить или уменьшить рисунок, а потом экспортировать его как растровое изображение с установленной пиксельной плотностью.

PDF, SVG, CSS и изображения с помощью кода

Другой метод поддержки разных разрешающих способностей и плотности пикселей – рисовать все, используя векторные изображения (такие как PDF или SVG) или код, либо создавать элементы с помощью CSS. В некоторых ситуациях эти методы работают очень хорошо, как правило, для простых объектов. Данные техники также подходят в тех случаях, когда размер, цвет или другие свойства объектов нужно изменять динамически.

Однако чем сложнее объекты, тем больше ресурсов они тратят. А это может привести к возникновению проблем. Повышенная потребность в ресурсах может обернуться большой проблемой для мобильных устройств. К тому же использование одного векторного SVG или PDF-изображения для всех пиксельных плотностей означает, что вы не сможете проконтролировать, как выглядит результат в каждом размере. А это может быть более важным при изменении масштаба для размера, промежуточного заданным рядом 100, 150, 200 %.

Изображение в SVG или PDF с масштабом в 200 % будет выглядеть ярко и четко, но уменьшение его до 140, 150 или 180 %, вероятнее всего, «размывает» резкость.

Однако прежде чем углубиться в техники Photoshop, давайте подготовим наше рабочее поле.

Подготовка вашего рабочего поля

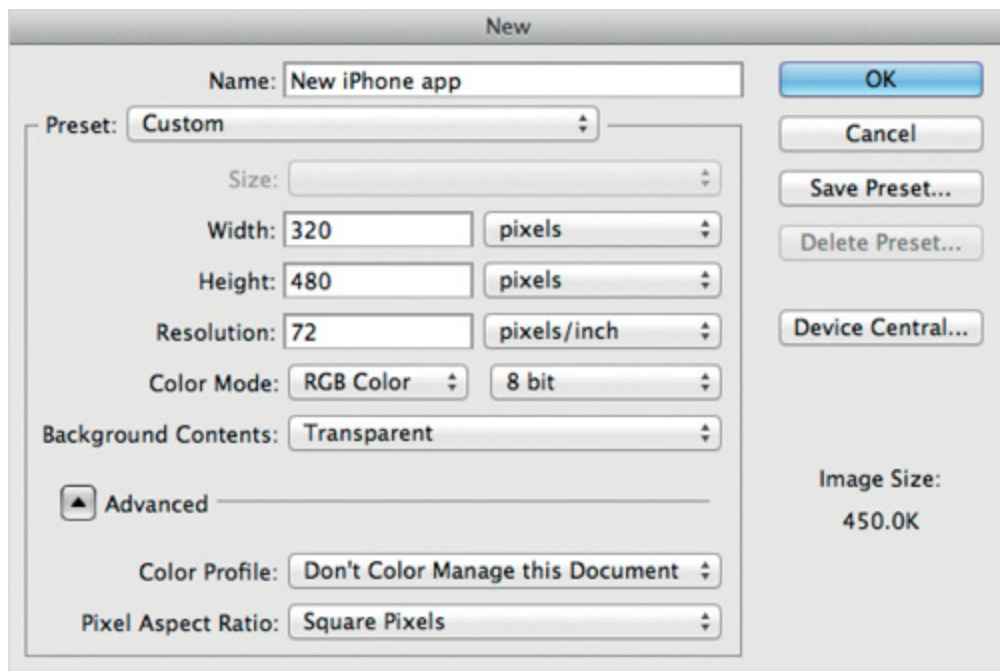


Рисунок 7.5. Решайте, с чего начнете новый документ: с маленькой пиксельной плотности с увеличением масштаба или наоборот

Начинаем работу над новым проектом в Photoshop. Вы хотите подогнать ширину и высоту вашего документа под размер конечного сайта или приложения. В нашем случае мы разрабатываем iPhone приложение в «портретной» ориентации. Поэтому мы начинаем с холста размером 320 × 480 пикселей. В качестве отправной точки я выбрал пиксельное плотность не-Retina iPhone (т. е. меньшую), потому что предпочитаю разрабатывать на исходном размере. А затем уже увеличиваю масштаб и экспортирую в 2× изображения, чтобы иметь возможность прикрепить объект изображения к пиксельной сетке, и я люблю, когда достаточно рабочего пространства. Это личный выбор, а вы, может быть, захотите начать с больших размеров дисплея Retina с уменьшением масштабов. У каждого метода есть свои за и против, и вам нужно было бы принять аналогично решение, если бы вы разрабатывали приложения Android, Windows Metro или сайты для дисплеев с высокой и низкой плотностью.

Пиксельная сетка

Начало работы с маленького размера гарантирует, что все, что вы делаете, «привязано» к сетке с шагом в 1 пиксель. Если бы мы начали с размера 2×, это означало бы, что нам нужно использовать только четные величины: четное позиционирование, четную высоту, ширину и четные величины стилей слоя. В противном случае, при уменьшении масштаба нечетные величины (1, 3, 5 и т. д.) сократятся на половину пикселей (0,5; 1,5; 2,5), что приведет к размытым очертаниям или ошибке округления.

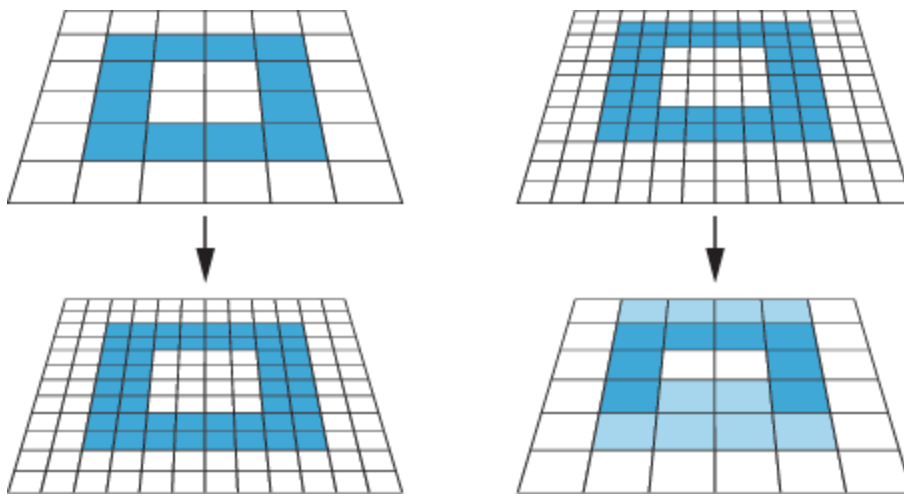


Рисунок 7.6. Увеличение масштаба до точных кратных величин всегда срабатывает. Уменьшение масштаба может вызвать проблемы

Размер предпросмотра

Работа в масштабе 1× означает, что превью 1:1 (где 1 пиксель на дисплее вашего компьютера отображает 1 пиксель в вашем дизайне) на дисплее вашего компьютера будет выглядеть меньше. В некоторых случаях это жизненно важно, потому что дисплей может быть недостаточно большим, чтобы сделать полный «портретный» предпросмотр на дисплее iPhone Retina (960 пикселей высотой плюс пространство для строки меню, окна и нижней панели). С дисплеем Retina на iPad ситуация становится только хуже.

Я знаю не так уж много дисплеев, которые могут вместить 2048 пикселей вертикально. Современный 27-дюймовый дисплей Mac Cinema имеет высоту в 1440 пикселей, поэтому даже если у вас

большой монитор, вы не сможете увидеть экран для Retina iPad необрезанным. Retina iPad имеет 3,145,728 пикселей – гораздо скромнее по сравнению с 27-дюймовым дисплеем Cinema в 3,686,400 пикселей. Но как только выйдут в свет компьютеры с высоким PPI^[74], предварительный размер уже не будет проблемой.

PPI документа

Вы могли заметить, что новый документ задается с разрешением в 72 PPI. Возможно, это выглядит несколько парадоксально, так как не увязывается с пиксельной плотностью устройства. Так или иначе есть причины работать именно с этой величиной.

Допустим, что один ваш документ имеет 100 PPI, а другой – 200 PPI. Если вы воспользуетесь функцией Copy Layer Style (Скопировать стиль слоя) на слое в документе с 100 PPI и Paste Layer Style (Вставить стиль слоя) на слое в документе с 200 PPI, стиль слоя изменит масштаб. Тень размером в 1 px (пиксель) превратится в двухпиксельную.

Вроде бы разумно с точки зрения Photoshop, но, возможно, не то, что вам нужно. Вот почему: если вы внимательно следите за PPI вашего документа, то могли бы установить ваши iPhone-документы на 163 PPI или 326 PPI, а ваши iPad-документы – на 132 PPI и 264 PPI.

Если вы работаете над приложением, которое подходит и для iPhone, и для iPad, велик шанс, что вы будете копировать элементы и стили слоя между вашими макетами под iPhone и iPad. Если разрешение PPI вашего документа устанавливается в соответствии с устройствами, ваши стили слоя каждый раз будут изменять масштаб на 20 %.

Это не проблема для однопиксельных теней, когда они будут округляться больше или меньше до ближайшего пикселя и, скорее всего, такими и останутся. Но любая величина больше 5 пикселей будет масштабироваться, и вам захочется узнать, почему элементы слегка смещены после своего копирования между документами. То же самое происходит и с устройствами Android или Windows Metro.

Итак, как обсуждалось раньше, плотность изображения не имеет значения для веба и приложений. В расчет берутся размеры пикселей конечного изображения, а не PPI, которое вы настраиваете в Photoshop.

Поэтому я настоятельно рекомендую всегда задавать документы на 72 PPI. Это сделает создание Photoshop-документа более предсказуемым.

Цветовой профиль

Наш цветовой профиль установлен в положение «Не управлять цветом этого документа». Это не случайно и необходимо, если вы хотите, чтобы цвета в Photoshop и Illustrator совпадали для других приложений и не изменялись при экспортировании.

В мире печати управление цветом включает в себя калибровку всего технологического процесса, от сканера или цифровой камеры до дисплея компьютера, от пробного оттиска до готовой печатной продукции.

Это может показаться невыполнимым, особенно когда устройства используют разные цветовые пространства – подгонка под устройства RGB и CMYK печально известна своей сложностью.

При разработке дизайна и редактировании для ТВ обычно применяется калибровка главного редакционного видеотерминала и использование контрольного видеомонитора. Пробное изображение в реальном времени показывает, как оно будет выглядеть на обычном домашнем телевизоре.

В этих сценариях рекомендуется управление цветом, которое предлагает много преимуществ.

При разработке веб-приложений и интерфейсов приложений, ситуация несколько иная. Готовый продукт отобразится на том же устройстве (или том же типе устройства), на котором вы обычно создавали свои произведения: на дисплее компьютера. Однако здесь есть кое-какие сложности. Даже если устройство, на котором вы будете делать веб-приложение или интерфейс приложения будет тем же либо подобным тому, на котором вы собираетесь использовать готовый продукт, у вас будут разные ресурсы для цветопередачи: изображения (как правило, в форматах PNG, GIF и JPEG), разметка стилей (CSS) и код (JavaScript, Java, HTML, Objective-C и т. д.). Подогнать их может оказаться достаточно сложным процессом.

При разработке сайта или приложений мы хотим, чтобы цвета, отображаемые на экране в Photoshop и сохраняемые в файлах,

великолепно подходили к тому, что отображается в других приложениях, включая Firefox, Safari и симулятор iOS.

Цвета не должны смещаться или производить впечатление смещения, как говорят, при любых обстоятельствах и при любой погоде. Поэтому мы не хотим, чтобы управление цветом редактора Photoshop в приложении изменяло цвета на экране или в файлах для хранения.

Отключение системы управления цветом RGB в Photoshop

Чтобы отключить систему управления цветом RGB в Photoshop, выберите Edit → Color Settings (Редактировать → Настройки цветов) и установите рабочее пространство для RGB на Monitor RGB. Убедитесь, что цветовой профиль каждого документа, с которым вы работаете, настроен на положение Don't Color Manage This Document (Не управлять цветом этого документа). Это делается с помощью выбора Edit → Assign Profile (Редактировать → Назначить профиль) или с помощью конфигурации дополнительных опций при создании нового документа. Если вы не будете делать этого для каждого документа, с которым работаете, цвета будут неправильно отображаться в самом редакторе Photoshop.

Каждый Photoshop документ содержит цветовой профиль, который отделен от фактических цветовых данных, сохраняемых для каждого пикселя. Опция Assign Profile (Назначить профиль) просто изменяет профиль в документе, не затрагивая цветовые данные. Это действие ничего не разрушит. Вы можете устанавливать новый профиль на ваши документы столько, сколько вам захочется, ничего не повредив при этом. Назначение нового профиля может изменить вид документа на экране, но данные, которые содержатся в файле, останутся нетронутыми.

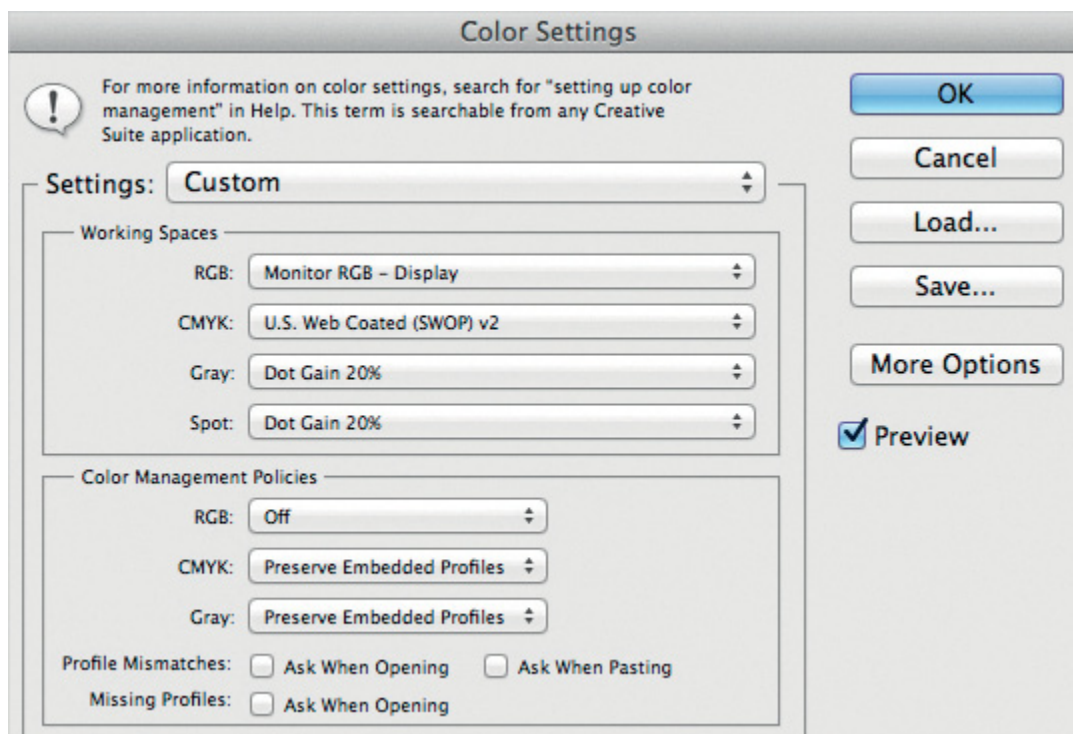


Рисунок 7.7. Отключение системы RGB в Photoshop помогает избежать настройки цвета Photoshop на экране или в сохраненных файлах

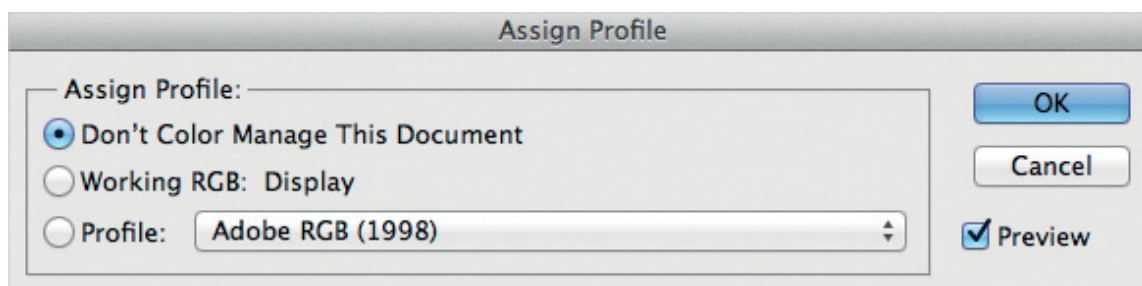


Рисунок 7.8. С опцией Assign Profile (Назначить профиль) в значении «Не менять» вы изменяете то, как ваш документ будет отображаться на экране, не затрагивая цветовые данные

Рисунок 7.9. При сохранении файлов через опцию Save For Web (Сохранить для веба) в Photoshop, убедитесь, что Convert to sRGB (Преобразовать в sRGB) отключена, также и когда сохраняете в формат JPEG. Иначе вы измените цвет и вызовете несовпадение величин

А вот опция Convert to Profile (Преобразовать профиль) – это нечто иное. С ее помощью можно не только назначить цветовой профиль документа, но и попытаться сохранить вид вашего изображения таким же и на экране. Это достигается путем обработки цветовых данных в каждом файле для каждого пикселя. Преобразование в новый профиль, скорее всего, будет поддерживать способ, каким документ появится на экране, но данные, хранящиеся, в файле будут изменены навсегда. Поэтому используйте эту опцию с осторожностью.

Если вы копируете слой из одного Photoshop документа в другой, проверьте, чтобы для обоих был назначен один цветовой профиль. Если этого нет, то информация о цвете может быть разрушена, так как она перемещается между документами. Вам также нужно убедиться в том, что опция View → Proof Colors (Вид → Цвета пробного отпечатка) отключена. Если нет, то опция Proof Colors изменит способ отображения цветов в вашем документе. А это значит, что они не подойдут другим приложениям.

И, наконец, при сохранении файлов опцией Save for Web (Сохранить для веба), посмотрите, отключена ли Convert to sRGB (Преобразовать в sRGB). Если она активирована, то изображение преобразуется из текущего цветового профиля в цветовой профиль sRGB, таким образом, изменяя цветовые значения, внося деструктивные изменения в файл и вызывая несоответствия с закодированными цветами.

Если вы сохраняете файл в формате JPEG, то также отключите опцию Embed Color Profile (Вставить цветовой профиль) (в некоторых случаях для фотографий вы, возможно, захотите ее оставить и отключить для элементов интерфейса и пиктограмм).

Если вы пользуетесь Adobe Illustrator вместе с Photoshop и стремитесь к тому, чтобы ваши цвета оставались неизменными, когда вы вставляете элементы между ними двумя, тогда вам нужно установить Illustrator с такими же настройками.

Рисунок 7.10. При создании интерфейсов всегда устанавливайте цветовой профиль документа Illustrator в положение Don't Color Manage this Document (Не управлять цветом этого документа) через Edit → Assign Profile (Редактировать → Назначить профиль)

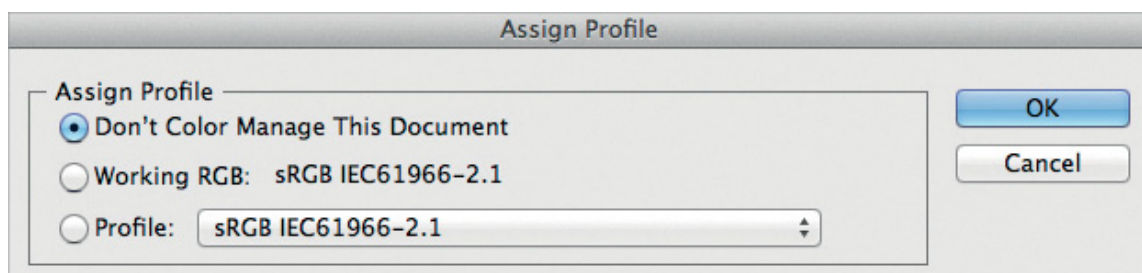


Рисунок 7.11. Установите положение Don't Color Manage This Document для каждого Illustrator документа, с которым вы работаете. Также отключите View → Proof Colors

Рисунок 7.12. Сохраняя файлы с опцией Save For Web And Devices (Сохранить для веба и устройств) в Illustrator, проверьте, чтобы опция Convert to sRGB (Преобразовать в sRGB) была отключена

Отключение системы управления RGB в Illustrator

Система управления цветом в Illustrator очень схожа с Photoshop, в случае, если настройки производятся для дизайна под веб или экранные приложения. Для отключения системы управления цветом RGB в редакторе Illustrator перейдите к опции Edit → Color Settings (Редактировать → Настройки цвета) и установите рабочее пространство для RGB на Monitor RGB. Проверьте, чтобы в каждом документе, с которым вы работаете, цвет профиля был установлен в положение Don't Color Manage This Document (Не управлять цветом этого документа). Для этого выберите Edit → Assign Profile (Редактировать → Назначить профиль). Это должно быть сделано для каждого документа, с которым вы работаете.

Вам также нужно отключить View → Proof Colors (Просмотр → Цвета пробного отпечатка). Если не сделаете этого, Proof Colors изменит способ отображения цветов в вашем документе. А это значит, что они не подойдут другим приложениям. При сохранении файлов опцией Save for Web (Сохранить для веба), посмотрите, отключена ли опция Convert to sRGB (Преобразовать в sRGB).

Фигуры

Слои фигур в Photoshop создаются и редактируются как векторные контуры, которые могут отображаться в оптимальном качестве и в любом размере. Это идеальная отправная точка для масштабируемых, гибких глифов, пиктограмм и элементов интерфейса.

Рисунок 7.13. Пиктограмма выше – один векторный слой, созданный из нескольких контуров

Почти любая сложная фигура может быть создана с использованием комбинаций фигур Photoshop (прямоугольник, закругленный прямоугольник, эллипс, многоугольник, линия и произвольная фигура).

Рисунок 7.14. Фигуры Photoshop «Прямоугольник» и «Закругленный прямоугольник» допускают четкие края на всех сторонах. Чекбокс Snap to Pixels можно найти в окне опций, как правило, вверху экрана

Привязка к пиксельной сетке

Фигуры «Прямоугольник» и «Закругленный прямоугольник» имеют хорошо скрытую опцию, которая позволяет им прикрепляться к пиксельной сетке, обеспечивая четкие края всем сторонам. Кнопку-флажок Snap to Pixels (Попадать в пиксели) можно найти в панели опций (Option bar), обычно вверху экрана.

К сожалению, у фигуры «Эллипс» нет опции Snap to Pixel (Попадать в пиксели). Если вам нужно нарисовать пиксельный круг, вам поможет в этом «Закругленный прямоугольник с большим радиусом угла».

Устранение проблем с вращением

Вращающиеся слои в Photoshop при 90 или 270° с выбором опций Free Transform Path, Rotate 90° CW или Rotate 90° CCW в меню редактирования могут вызвать проблемы с векторными и растровыми

слоями. Качество готового продукта определяется размером иллюстрации. Если слой имеет четные ширину и высоту, все хорошо.

Рисунок 7.15. Обход ошибки вращения Photoshop с помощью вращения Bjango. smashed.by/rotation

Если ширина и высота нечетные, то тоже все нормально. Но если они нечетно-четные или четно-нечетные, то результат может быть похож на то, что изображено справа (см. рис. 7.15). Изменение точки вращения с центральной на верхнюю левую, верхнюю правую, нижнюю левую или нижнюю правую до вращения приведет к тому, что все останется четким после трансформации.

Сдвиг точек на один пиксель

При перемещении якорных точек векторных контуров Photoshop может продемонстрировать весьма странное поведение, в зависимости от того, насколько увеличен масштаб. При 100 % перемещение стрелкой подвинет вашу векторную точку точно на 1 пиксель. При 200 % перемещение продвинет точку на половину пикселя. При 300 % – на одну треть пикселя. Если вы хотите иметь прекрасные пиксельные векторные фигуры, возможно, вы предпочтете перемещать с шагом в 1 пиксель, даже если вы редактируете при увеличенном масштабе.

Рисунок 7.16. Изменение точки вращения на верхнюю левую до вращения обеспечивает поддержку качества

Мы можем воспользоваться сдвигом в Photoshop даже при 100 %. Откройте ваш документ, выберите Window → Arrange → New Window для создания второго окна документа. Потом вы сможете изменить размер нового окна и переместить его куда-нибудь.

В оригинальном окне редактируйте как обычно, и увеличивайте масштаб так, как вы хотите.

Если вам нужно переместить точку, просто нажмите команду Command + `, чтобы включить окно, в котором выставлен масштаб

100 %, переместите элемент с помощью кнопок клавиатуры, потом нажмите Команду Command + ` , чтобы переключить обратно. Так как другое окно показывает изображение в масштабе 100 %, перемещение сместит выбранные векторные точки точно на 1 пиксель. Немного неудобно, но гораздо быстрее, чем уменьшать масштаб до 100 %, а потом наоборот, чтобы отредактировать тонкие детали.

Как бы то ни было, перетаскивание якорных точек векторных контуров мышкой прикрепляет их к пиксельной сетке. А удерживая Shift с использованием стрелок курсора при перемещении всегда продвигает элемент на 10 пикселей, неважно насколько вы увеличили масштаб.

Заливка и форма

Заливка и тени добавляют дизайну тело и форму, при этом элементы выглядят натурально, как будто в трехмерном измерении.

Приподнятые выпуклые элементы своим видом показывают, что их можно нажать. Впалые элементы выглядят так, как будто их вырезали. Тени указывают высоту, структуру взаиморасположения и иерархию. Эти важные «подсказки» при быстром просмотре показывают, как функционирует пользовательский интерфейс, играя на нашем опыте с реальными объектами и освещением.

Однако прежде чем мы начнем экспериментировать с заливкой, тенями и другими техниками, нам нужно решить вопрос с источниками света. Как правило, дизайн интерфейсов и иллюстрации выглядят освещенными от верха экрана параллельными световыми лучами. Эта мнемосхема – обычная дневная сцена, где отовсюду светит солнце. Она напоминает сцену внутри помещения, где свет исходит сверху, с потолка. Вы можете решить, использовать вам другой световой источник или даже несколько. Это хорошо, потому что демонстрирует вашу последовательность в стиле, а значит и весь дизайн соблюдает те же правила.

Как в Photoshop, так и в Illustrator, заливка часто завершается градиентами. В Photoshop это лучше всего достигается при применении опций Gradient Fill Layers (Слой заливки градиентом) или Gradient Layer Styles (Градиентные стили слоя), потому что они могут

масштабироваться безгранично. В Illustrator заливка градиентом может применяться к любому контуру.

Выпуклая заливка

Выпуклые фигуры выступают наружу в направлении смотрящего. Они часто выражены линейными градиентами от светлого к темному, так как свет «падает» сверху. Выпуклые фигуры выглядят приподнятыми, и это отлично подходит для кнопок.

Вогнутая заливка

Вогнутые фигуры выглядят полыми или вдавленными и могут быть нарисованы, как линейные градиенты от темного к светлому (в противоположность выпуклым фигурам). Используя комбинацию нескольких фигур с градиентной заливкой, мы можем создать простой вид в каком-то измерении.

Рисунок 7.17. Выпуклые фигуры выступают наружу, выражены линейными градиентами от светлого к темному. Выпуклые фигуры выглядят приподнятыми, отлично подходят для кнопок

Рисунок 7.18. При легком изменении градиента мы можем сделать фигуру как будто с большим сплошным участком в середине. Этого можно также добиться, добавив второй слой со сплошной цветовой заливкой

Рисунок 7.19. Вогнутые фигуры могут быть нарисованы как линейные градиенты от темного к светлому. Поперечное сечение (слева). Элементы с градиентной заливкой (справа)

Рисунок 7.20. Линии сверху и снизу выглядят как высеченные каналы. Углубленные линии часто используются как перегородки в

пользовательских интерфейсах

Сферические фигуры

Сферические фигуры часто создаются радиальными градиентами. Радиальные градиенты начинаются от центра и перерастают в круговую диаграмму. В Photoshop центральная точка градиента может перетаскиваться мышкой по холсту, когда открыто окно задания градиента (для градиентных заливок) или когда открыто окно стилей слоя (для градиентов стилей слоя).

Для сферических фигур характерно отражение рассеянного света с поверхности ниже. Воспроизвести этот эффект возможно с несколькими незначительными изменениями градиентов. Использование фотографий и других исходников может помочь вам точно настроить величину контраста, который вам необходим для воспроизведения заданного материала (см. рис. 7.22).

Рисунок 7.21. Для градиентных заливок центральная точка градиента может перетаскиваться мышкой по холсту, когда градиентное окно открыто

Рисунок 7.22. Одни и те же рассеянные отражения могут существовать в других реальных объектах. Тонкое использование градиентов от светлого к темному и от темного к светлому может выглядеть впечатляюще, так же, как и добавление реализма в дизайн

Отраженные градиенты

Отраженные градиенты в Photoshop содержат линейный градиент, который можно редактировать, плюс зеркальное повторение того же градиента. Отраженные градиенты делают редактирование симметричных градиентов не таким нудным и производят ровно тот эффект, которого вы пытаетесь добиться.

Угловые градиенты

Угловые градиенты – это прекрасный способ имитации отражения окружающих предметов в металлических объектах. Как и у радиальных градиентов, центральная точка градиента может перемещаться в Photoshop при помощи мышки и курсора, когда градиентное окно открыто (для градиентных заливок) или когда открыто окно стилей (для градиентов стилей слоя).

Градиенты на градиентах

Комбинирование слоя заливки градиентом (Gradient Fill Layer) с градиентным стилем слоев (Gradient Layer Style) позволяет вам наложить два различных градиента, создавая более сложные и динамичные результаты. Чтобы соединить градиенты, вам нужно установить режим смещения для градиентов стилей слоя (Gradient Layer Style). Это позволит слою градиентной заливки (Gradient Fill Layer) проявиться через градиент стилей слоя (Gradient Layer Style). В примере, данном ниже, я использовал режим смещения Screen (хорошо для осветления) либо Multiply (хорошо для затемнения).

Рисунок 7.23. Центральная точка углового градиента может перетаскиваться курсором по холсту при открытом градиентном окне

Рисунок 7.24. Для комбинирования градиентов вам нужно установить режим смешивания для стиля градиентного слоя (Gradient Layer Style), т. е. Screen (хорошо для осветления) или Multiply (хорошо для затемнения)

Рисунок 7.25. Неразмытый градиент (слева) в сравнении с размытым (справа). Градиент справа выглядит более гладким (ровным)

Размытие градиентов

Добавление размытия в градиент дает более ровные результаты, потому что дает изысканно-узорчатый эффект или белый шум.

Неразмытые градиенты часто содержат визуальную сегментацию (полосатость). Размытие даже больше важно, если ваш рисунок будет просматриваться на дешевых нематических жидкокристаллических экранах (TN LCD) с шестью битами на канал и некоторых других видах дисплеев, где присутствуют проблемы с постеризацией (приданием «плакатного стиля»). Photoshop может размывать слои градиентной заливки, а также градиенты, нарисованные с помощью градиентного инструмента, поэтому советую вам включать эту опцию. Градиенты, нарисованные в Illustrator, не могут размываться, так же как и векторные смарт-объекты (Smart Objects), которые вставляются в Photoshop из Illustrator.

Если вы используете прозрачность как часть градиента, он тоже не будет размываться, что иногда приводит к визуальной сегментации (полосатости). Для определенных случаев есть решение: если вы используете градиент с прозрачностью для того, чтобы осветлить область белым цветом, то использование неразмытого градиента с режимом смешивания слоев на экране (Screen layer blending mode) позволит вам размыть его. Та же техника применяется для затемнения, с режимом смешивания Multiply. Размытие цветов – это ювелирная работа. И подчас оно едва уловимо. В примере слева мы повысили контраст для того, чтобы выделить размытый паттерн. Градиент справа выглядит более ровным, потому что он размыт.

Текстуры

В реальной жизни большинство вещей имеет свою текстуру. Текстура может быть явно видимой (например, крупные волокна на деревянном бруске) или едва уловимой (например, тонкий узор очищенного алюминия).

Добавление текстуры может стать отличным способом при обозначении разных областей и поверхностей и придать элементам более тактильный, реалистичный вид.

Однако здесь есть небольшая сложность. Текстуры, как правило, основаны на растровом изображении, поэтому они выглядят как фотография. Как мы уже говорили, растровые изображения или растровые слои масштабируются неудачно, и их нужно избегать, где только возможно. Photoshop предлагает три способа добавления

текстуры, которая также будет без разрушений изменять масштаб ваших документов: Pattern Fill Layers (Заливка слоев паттернами), Pattern Layer Styles (Стили заливки паттернами) и Smart Objects (смарт-объекты).

Создание паттерна

Чтобы создать паттерны в Photoshop мы можем использовать опцию выделения с помощью рамки (Marquee Selection tool) для выбора прямоугольной площади, а затем выбрать опции Edit → Define Pattern (Редактировать → Задать паттерн). После создания паттерн готов к использованию как через слой заполнения паттерном (Pattern Fill Layer), так и через паттерный стиль слоя (Pattern Layer Style).

Если вы собираетесь увеличить масштаб вашего документа, то создавать текстуру своего шаблона придется в большем, чем нужно, размере.

Если хотите проверить точно, как выглядят паттерны в определенных масштабах, то вам придется создавать версии для каждого нужного размера. Например, вы могли бы создать два шаблона для iOS текстуры: один для масштабирования в Retina-устройствах, а другой – для масштабирования в не-Retina-устройствах. Потом при экспорте всех финальных исходников изображения вы бы включили паттерны, используемые в ваших Pattern Fill Layers и Pattern Layer Styles.

Слои заливки паттерном

Слои заливки паттерном (Pattern Fill Layers) в точности соответствуют своему названию: это слои, заполняемые узорами. Они могут также факультативно содержать векторную маску, поэтому, даже если растровое изображение внутри слоя с паттерном сглаживается при масштабировании, границы будут оставаться резкими. Двойной щелчок на миниатюру Pattern Fill Layer на панели слоев открывает опции, позволяющие вам установить масштаб паттерна независимо от масштаба документа.

Стили заливки паттерном

Стили заливки паттерном (Pattern Layer Styles) подобны заливкам паттерном (Pattern Fills), но применяются как стиль слоя. Это значит, что они не могут использоваться вместе с опциями Solid Color Fill Layers (Заливка слоев однотонным цветом), Gradient Fill Layers (Слой заливки градиентом) и даже с другими слоями Pattern Fill Layers (Заливка паттерном).

Смарт-объекты (Smart Objects)

В Photoshop смарт-объекты (Smart Objects) – это документы, расположенные внутри слоя, что дает идеальную возможность вставлять текстуру высокого разрешения в документ с низким разрешением. Смарт-объекты формируются, когда формируется их исходный файл. Смарт-объекты, созданные внутри Photoshop, формируются в их исходном размере, затем их масштаб увеличивается или уменьшается с использованием растрового масштабирования.

Лучше всего использовать смарт-объекты в виде прямоугольной области с векторной маской, применяемой для фигуры. Это означает, что сама текстура будет масштабироваться в растровом изображении, а фигура будет перерисована по размерам как векторный объект, создавая четкие границы.

Чтобы создать смарт-объект в Photoshop, щелкните правой кнопкой мыши или кликните левой клавишей с зажатой клавишей Ctrl в слое или группе, затем выберите опцию Convert to Smart Object (Преобразовать в смарт-объект).

Смарт-объекты, созданные вставкой или импортированием файла из редактора Illustrator, по своей природе векторы, поэтому они могут масштабироваться в любой размер и, если нужно, формироваться заново. Смарт-объекты из Illustrator могут иметь множество заливок цвета, обводок и свойств, недоступных векторным слоям в Photoshop. Однако будьте осторожны: смарт-объекты из Illustrator не рисуются размытыми градиентами, поэтому у них могут быть проблемы со сглаживанием.

Чтобы создать смарт-объект из файла Illustrator, выберите File → Place from Photoshop (Файл → Вставить из Photoshop). Также возможно копирование объектов в Illustrator с последующей вставкой в Photoshop.

Передача дизайна

Все дизайнеры, разработчики и команды работают по-разному. Некоторые рассматривают дизайн как процесс, который проходит в параллели с разработкой. Другие – как задачу, которую лучше выполнить до написания кода. По возможности, я предпочитаю последнее, потому что могу делать быстрые итерации в дизайне без изменения кода.

Какой бы метод вы ни применяли, помните всего о нескольких простых вещах, когда передаете дизайн разработчику или команде разработчиков. Кроме полного комплекта изображений, вам нужно рассматривать и другие аспекты.

Информация о шрифте

Некоторым разделам вашего дизайна, скорее всего, нужен текст, который будет генерироваться в коде. Отправьте разработчику полные характеристики для каждого вида текста, включая шрифт, его размер и соответствующие тени.

Проверьте, чтобы используемые шрифты были в заданных устройствах. Для сайтов и веб-приложений вам, возможно, будет нужно создать группу шрифтов, список, определяющий порядок предпочтений для шрифтов на случай, если нет какого-то определенного шрифта. Если шрифт заменяется, слова или фразы могут расширяться, поэтому проверяйте все шрифты в вашей группе.

Кроме того, возможности кода – это обычно только часть того, что возможно в Photoshop. Заливка градиентом или сложные стили заливки текста в Photoshop может оказаться непростой, невозможной и нежелательной задачей в приложении или сайте.

Если вы разрабатываете дизайн для iOS, то значение 72 PPI в документе Photoshop задаст размер шрифта близкий к размеру в среде разработки Xcode: текст размером в 10 пунктов в документе Photoshop с масштабом 1x (320 на 480 пикселей для iPhone) будет эквивалентен тексту размером 10 пунктов в iOS. Но помните, что во внешнем виде текста будут некоторые различия в результате рендеринга между Photoshop и iOS.

Растягиваемые изображения

Изображения интерфейса с динамической шириной или высотой, как правило, растягиваются путем повторения сегмента в середине элемента. Если ваши растягиваемые изображения содержат паттерны или размытые градиенты, тогда вы, возможно, захотите добавить комментарий для разработчиков. Повторяющиеся текстуры шириной в 1 пиксель лишены эффекта размытия и обычно выглядят ужасно. Ведь для того, чтобы мы заметили воплощение магии, размытым паттернам нужно больше пространства.

Изображения и плотность пикселей

Информация о PPI почти всегда игнорируется, когда изображения используются в Сети или как элементы интерфейса пользователя в приложениях iOS, Android, Windows Metro и Mac.

Тем не менее пиксельные размеры ваших изображений все-таки имеют значение, так что делайте их правильно. Для iOS проверьте, что ваши 2× изображения в точности удваивают размеры ваших 1× изображений и что элементы внутри изображений находятся в одинаковых положениях; ваши изображения Retina должны быть идентичны их меньшим аналогам, но и быть более детальными.

Помните, что изображение в формате PNG сохраняет свое пиксельное разрешение в пикселях на метр (PPM). Это может вызывать ошибки округления, когда величины преобразуются между PPI и PPM. Если вы когда-нибудь видели, как при сохранении изображения в формате PNG его разрешение изменялось с 300 до 299,999 PPI, то вы меня понимаете.

Сжатие PNG и приложения iOS

Обработка ваших изображений инструментами сжатия PNG, такими как OptiPNG, PNGcrush, AdvPNG или PNGOUT, может показаться классной идеей! Инструмент работает вовсю, «обрезая» килобайты и байты в каждом файле в надежде улучшить скорость скачивания приложения и ускорить запуск.

Но для кардинального увеличения производительности приложений iOS встроенная среда разработки Xcode перекодирует/пересжимает файлы PNG. Она предварительно умножает разряды альфа-канала в обратном порядке, и байты в красном, зеленом и синем

каналах меняют последовательность на синий, зеленый и красный. Затем Xcode перекодирует изображения, используя утилиту PNGCrush. Результат оптимизируется для iOS, но, как побочный эффект, любое предшествующее сжатие остается необратимым из-за того, что изображения перестраиваются.

Определенно, есть причины для оптимизации ваших изображений для веба, есть хорошие инструменты, но если вы создаете приложения iOS в программной среде Cocoa и используете встроенное в программу Xcode сжатие PNG, то не получаете особых преимуществ от предварительного сжатия файлов.

Проверка устройств

Если вы разрабатываете дизайн для мобильных устройств, проверьте ваши готовые макеты на целевом или целевых устройствах. В этом много плюсов. Вы сможете протестировать эргономику и области нажатий, увидите дизайн в контексте, с характеристиками компоновки во всей красе. И у вас при этом будет роскошная возможность внести в разработку изменения.

Проверка устройств – дело важное, потому что типы экранов, теплота тонов, и даже субпиксельные паттерны значительно различаются. Поэтому вы, возможно, захотите «причесать» дизайн после того как увидите все на своих местах. Некоторые типы дисплеев, например AMOLED, могут выглядеть гораздо насыщеннее и с более высоким контрастом, чем обычные ЖК экраны компьютеров. Я уже не говорю о том, каким захватывающим зрелищем является просмотр вашего дизайна на устройствах.

Есть много способов посмотреть ваш готовый макет на мобильных устройствах. Сброс изображений себе на почту или использование таких сервисов, как Dropbox, срабатывают отлично, так же как и более нестандартные решения, такие как LiveView и Skala Preview. По возможности, проверяйте методом, который поддерживает качество изображения: 32-битовые PNG изображения прекрасны, но сжатие с потерями, например в формате JPEG могло бы показать все артефакты.

Техники для экспорта изображений из Photoshop

Экспорт всех изображений, необходимых для создания сайта или приложения, возможно, не самое интересное занятие в процессе разработки дизайна. Это, как правило, нудная работа, отнимающая много времени.

Сохранение изображений во многих масштабах, которые требуют iOS, Android, Windows Metro и другие платформы, еще усложняет задачу.

Но есть способы упростить или автоматизировать экспорт.

Скопировать объединенные данные

Разрезать ваш дизайн с помощью опции Copy Merged (скопировать объединенные данные) довольно легко: сделайте соответствующие слои видимыми, выделите элемент с помощью опции Marquee selection, выберите Edit → Copy Merged (Редактировать → Скопировать объединенные данные), потом File → New (Файл → Новый), нажмите Enter и затем Paste (Вставить). В результате получится новый документ с вашим изолированным элементом, установленном на минимально возможный размер. С этого момента все, что нужно, это сохранить изображение, используя команды Save As (Сохранить как) или Save For Web And Devices (Сохранить для сети и устройств).

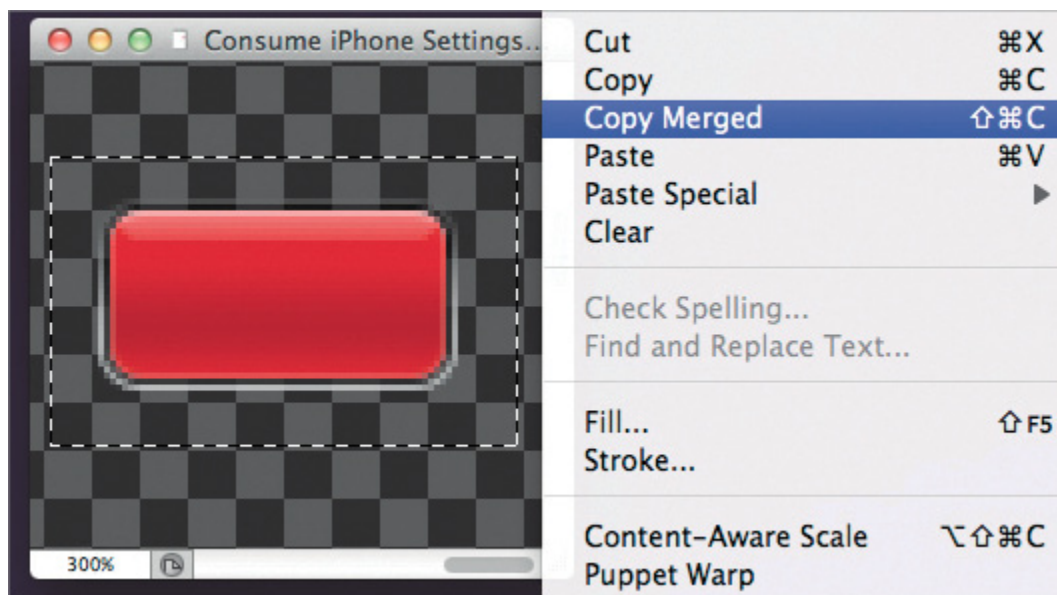


Рисунок 7.26. Разрезать ваш дизайн с помощью Copy Merged достаточно легко: сделайте соответствующие слои видимыми, выделите ваш элемент с помощью опции Marquee selection, выберите Edit → Copy Merged, затем File → New, нажмите Enter и потом Paste

Повторяйте эти действия для каждого изображения вашего приложения или сайта. Техника несложная и быстрая, но требует много повторяющейся работы, и если вам когда-нибудь будет нужно снова экспортировать изображение, то придется начинать с нуля.

В моей практике это наиболее распространенный, а зачастую и единственный метод, который применяют дизайнеры. Стыдно! Ведь существуют техники лучше.

Вы могли бы создать экшен^[75] (операцию), который последовательно запускает опции Copy Merged, New, Paste. Это хоть немного сэкономило бы вам время и улучшило технологический процесс^[76].

Экспортирование слоев в файлы

Если бы вам выпало счастье экспортировать множество одинаковых изображений (как правило, с идентичными размерами), вы могли бы использовать опцию Export Layers (экспортирование слоев в Photoshop) в Files script (Сценарий файла).

При выборе команды File → Scripts → Export Layers to Files (Файл → Сценарии → Экспортировать слои в файлы), каждый слой вашего документа будет сохраняться как отдельный файл, с названием, которое соответствует названию слоя. Это означает, что, скорее всего, вы должны подготовить свой документ, объединив с растровыми слоями все элементы, которые хотите экспортировать. Процесс трудоемкий, но его можно ускорить, применив опцию Copy Merged. Также это может сократить итоговый файл, если вы хотите полностью удалить прозрачную область. Экспортировать слои в файлы удобно, если желаемый результат соответствует вариантам его использования.

Ломтики (Slices)

Photoshop-инструмент Slice tool (Нарезка ломтиками) позволяет вам определить прямоугольные участки для экспортирования в виде

отдельных изображений с некоторыми ограничениями: на один документ может приходиться только один комплект ломтиков, и ломтики не должны перекрываться (если так будет, то сформируются прямоугольные ломтики меньшего размера). В 1990-е годы опция Slice tool была хорошим способом создания табличных веб-макетов, заполненных изображениями. В наши дни нам нужен более тщательный контроль над тем, как нарезаются изображения, особенно если мы хотим иметь эффективный и динамичный дизайн с прозрачными изображениями. Так или иначе инструментом Slice tool можно отлично пользоваться.

Рисунок 7.27. С выбором команды File → Scripts → Export Layers to Files каждый слой вашего документа будет сохраняться, как отдельный файл, с названием, соответствующим названию слоя

Спрайты с ломтиками

Спрайты обычно используются в CSS и играх на OpenGL, где их использование может давать значительные преимущества в производительности. Спрайт – это большое изображение, которое состоит из более мелких графических элементов. При использовании спрайтов только одно большое изображение загружается в браузер, уменьшая количество HTTP-запросов. Подобное улучшение производительности происходит с OpenGL играми, где один файл, сохраненный в памяти GPU, может использоваться со ссылкой на множество меньших изображений в нем.

Рисунок 7.28. Спрайты требуют времени на создание, но они помогают автоматизировать экспортирование изображений

Рисунок 7.29. После создания спрайта с ломтиками все устанавливается корректно. Вы сможете экспортировать все изображения сразу же, используя команду Save for Web & Devices

Визуально схожий метод может применяться, чтобы экспортировать из Photoshop элементы интерфейса пользователя, даже если результат больше представляет собой набор изображений, а не одно большое.

При распределении элементов, которые нужны вам для экспортирования в виде спрайта, нет необходимости в перекрытии ломтиков. Если есть слишком много элементов, которые уютно располагаются в одном документе, то вы можете создать несколько документов, без необходимости создавать более одного набора ломтиков на документ.

Выгода от такого метода работы в том, что вам больше не нужно будет создавать ваши основные документы по дизайну с одинаковым уровнем точности. И не страшно иногда использовать растровые изображения или забыть дать название слою, так как у вас появится возможность привести все в порядок при подготовке вашего спрайта к экспортированию. Но это не означает, что ваш оригинальный макет документа не должен синхронизироваться с вашими последними изменениями при экспортировании документов (если вы, например, изменяете цвет или эффекты слоя).

Так как нас интересуют только ломтики, которые создает пользователь, возможно, вам захочется «щелкнуть» на команду Hide Auto Slices (Скрыть автоломтики) в окне опций при использовании инструмента Slice Select tool и отключить Show Slice Numbers (Показать количество ломтиков) – под Guides, Grid & Slices в настройке. Таким образом, будет удален ненужный беспорядок из Photoshop ломтиков пользовательского интерфейса. После создания спрайта с ломтиками все устанавливается корректно, вы можете экспортировать изображения сразу же, используя команду Save for Web & Devices. А чтобы убедиться, что вы все сделали как следует, вы можете увеличить масштаб до 200 %, сохранить все ваши изображения для дисплеев Retina, а потом добавить @2x к названиям файлов пакетным переименованием (или уменьшить масштаб, если вы сначала все создаете на размерах дисплеев Retina).

Ломтики на основе слоя

Если ваш элемент пользовательского интерфейса однослойный, а вы хотите, чтобы экспортируемое изображение было маленьким насколько это возможно, то рассмотрите использование опции Layer Based Slice. Для того, чтобы создать ломтик для выбранного слоя, используйте опцию New Layer Based Slice (Новый ломтик на основе слоя) из Layers menu (Меню слоев). Ломтики на основе слоя (Layer Based Slices) передвигаются, увеличиваются и уменьшаются вместе со слоем, к которому привязаны. Они также учитывают эффекты слоя: контуры и тени включены, и увеличивают размер ломтика на основе слоя. Меньше контроля, больше автоматизации!

Как я экспортирую

В течение нескольких лет я пользовался Copy Merged в качестве основного метода экспортирования и использовал Export Layers to Files, когда это имело смысл. Это был скучный выбор. Спрайты имеют так много плюсов, особенно для средних и больших проектов, что время на первоначальное их создание проходит очень быстро. Это еще более оправдано при экспорте множества комплектов изображений с разными пиксельными плотностями. Каждый комплект может экспортироваться несколькими кликами, а до проблем с названием или размером далеко благодаря автоматизированному процессу.

Также формируется такая среда, в которой легко изменять ресурсы продукта, делать более быструю итерацию и проводить больше экспериментов. Становится проще улучшать ваш продукт в процессе разработки и при каждом просмотре вашего сайта или приложения. А это очень хорошо.

Рисунок 7.30. Отключение зеленого канала помогает увидеть разницу между макетом и настоящим приложением

Сравнение и настройка

Итак, мы в поте лица трудились над деталями, а теперь давайте убедимся, что готовый продукт уживается с макетами из Photoshop. Мой излюбленный метод проверки ошибок в живом приложении или сайте – это сделать скриншот, открыть его в Photoshop, а затем

расположить оригинальный макет сверху, удалив зеленый канал. Это можно сделать с помощью опций смешивания (Blending Options) в окне стилей слоя. В большинстве своем это отлично срабатывает для нейтральных стилей. Если есть различия, оригинальный макет появится зеленым, а приложение и сайт – пурпурными. Определение разницы, измерение требуемых изменений и отправка комментариев разработчику (или, если вы сам разработчик, самостоятельная отшлифовка) – все это легко.

При сравнении текст в готовом продукте будет по внешнему виду отличаться от текста в макете Photoshop или Illustrator. Этого и следовало ожидать. iOS, Android, Windows Metro и все браузеры передают текст по-разному, иногда едва различимо, иногда со значительной разницей.

Новые проблемы

Мир программного обеспечения и, соответственно, дизайн ПО всегда находится в состоянии изменения. Новые технологии и новые возможности влекут за собой новые проблемы. Но перечень проблем, с которыми сталкиваются дизайнеры сегодня, никогда не был таким длинным и сложным.

Наряду с уже привычными трудностями создания теперь мы должны столкнуться с проблемами, которые создают дисплеи с высоким пиксельным разрешением. В большинстве случаев наши разработки кодифицируются.

Это механизмы; они должны масштабироваться и быть гибкими. Они также должны поддерживать человеческий элемент, потому что люди разговаривают с ними. Надеюсь, этот раздел помог вам подготовиться ко встрече с новыми проблемами.

Об авторе

* * *

Марк Эдвардс является директором и ведущим дизайнером в Bjango, независимой компании по разработке систем для Mac и iOS в Австралии. Марк – соведущий подкаста Iterate, иногда выступает на

конференциях и пишет статьи о дизайне для журнала Smashing Magazine и сайта компании Vjango.

Фото: Мэт Адамс

О рецензенте

* * *

Джон Хикс (р. 1972) родился в Лемингтоне Спа (Великобритания). В настоящее время он живет в небольшом городке с выездными ярмарками под Оксфордом.

Это прямо на краю великолепного местечка под названием Котсволдс, прекрасно подходящего для семейного очага и для верховой езды! Он до сумасшествия любит кататься на велосипеде и мотоцикле, а также обожает Lego и Dr. Who.

Джон работает дизайнером 18 лет, 10 из них – как фрилансер. Он известен как создатель логотипов Firefox, MailChimp и Shopify, но он работает в различных средах. У Джона есть две морских свинки (Том-Том и Тафти) и золотистый ретривер по кличке Олив. Его любимые цвета – оранжевый и черный. Самый большой урок, который он усвоил из своей карьеры, – это не упускать возможность заниматься своими небольшими проектами. По возможности каждую неделю посвящайте полдня просто экспериментам. Это принесет вам дивиденды, ваши старания окупятся. Он максималист по жизни и его кредо: «Что посеешь, то пожнешь!»

Переделка сайта с использованием индивидуальности

Автор: Аарон Уолтер

Рецензент: Дениз Джекобс

Переделка сайта может быть подобна семи кругам ада. Вы искали вдохновения в дюжине сайтов, сохраняли скриншоты, делали заметки, консультировались у друзей и коллег, возможно, брали интервью у пользователей. Но несмотря на ваше должное прилежание, ваше видение нового сайта остается неясным. Я понимаю вашу боль, друзья мои.

Я сам не раз испытывал ее. Переделка сайта несет с собой острую необходимость введения новшеств, вдохновения нового образа, создания лучшей версии. Так что это может длиться годами и в конце концов лишить вас энергии.

Неважно, для кого вы делаете сайт: для клиента, или для себя. Если вы бьетесь в поисках своего решения, скорее всего, вы не с того начали. Вдохновение, которого вы жаждете, вы не там ищите. Оно не в блоге с заголовком «25 поразительно великолепных сайтов». Оно не в ленте вашего Twitter и не в Facebook. Но даже не в Сети. Оно прямо там, где вы! Это – вы сами.

Перестаньте хоть на мгновение думать о семантике языка HTML, магии таблиц CSS и ловких трюках jQuery. Вместо этого спросите себя: «Кто я и что я хочу сказать?» Чего вы придерживаетесь, что важно для вас и с кем вы разговариваете? Давайте дадим ответы на эти вопросы в начале нашего пути.

Мы, веб-дизайнеры, имеем под рукой много привлекательных инструментов. А так как веб – это огромное сообщество, базирующееся на коллективном использовании данных, новые идеи и модные техники ежедневно попадают в поле нашего зрения. Но в этом разделе я хотел бы отвернуть ваш пристальный взор от этих манящих объектов и сосредоточить его на том, что мы пытаемся сделать в реальной жизни. Наша истинная цель – общаться ясно и строить человеческие взаимоотношения.

Мы достигнем этой цели не тем, что будем собирать прибалбасы для нашего проекта, а постижением самих себя и нашей миссии. Интерфейсы, которые мы создаем, это не стены, о которые наши пользователи щелкают мышкой и нажимают пальцем. Это окна, через которые мы показываем миру, кто же мы есть на самом деле. Как мы увидим из правил и примеров, раскрыв свою индивидуальность, мы построим долгие отношения с людьми, которые пользуются нашим сайтом, и сможем улучшить непосредственно наш бизнес.

Индивидуальность вашего бренда удалит от вас соперников и поможет навести мосты к преданной аудитории. Возможно, переделывая сайт, вы побоитесь проводить индивидуальность красной нитью, особенно, если вы работаете с большой корпорацией, привыкшей говорить, как Борг^[ZZ]. Но, даже самые огромные корпорации общаются на человеческом языке.

Кто они?

Большие проекты по переделке часто начинаются с «прощупывания» заказчиков. Мы садимся с людьми, чтобы обсудить их цели для нашего сайта и их ожидания. Мы смотрим на демографические показатели, аналитику и историю поисковых запросов. Нужно будет перелопатить много данных, но это не пустая трата времени. Из этого исследования мы можем создать портреты наших основных пользователей. Это досье на личности в нашей целевой аудитории называется «персонаж». Оно отвечает на важный вопрос в процессе редизайна: кто они, люди, с которыми мы общаемся, и что они от нас ожидают?

Есть вероятность того, что даже если вы провели мало времени за дизайном, то слышали о персонажах. Может быть, вы даже создали несколько. Мы спрашиваем себя: «Кто же они?» с момента, когда Алан Купер представил персонажей в интерактивном дизайне в 1995 году, и с тех пор они стали главным продуктом дизайна, ориентированном на пользователя. Если персонажи для вас неосвоенная территория, вы можете найти краткое введение по теме в «The Project Guide to UX Design»^[78] Руса Унгера и Каролин Чендлер. Если вы желаете покопаться глубже в исследовании пользователей, раздобудьте

специализированную книгу Алана Купера «The Inmates Are Running the Asylum»^[79].

С персонажами под рукой, мы получаем надежную отправную точку для начала редизайна. Но чего-то не хватает. Персонажи показывают нам только половину того, что нам нужно видеть. Истинно эффективное общение двунаправленное. Теперь мы знаем, кто они, но кто же мы? Если мы вложим чуточку нашей души в разработку, мы не только завоюем доверие публики, но и вдохновим страстных пользователей.

THE INFLUENCER


Julia

Age: 19 • 22; Sophomore; Journalism & Communications

Goals: Get a "Big City College" education, cosmopolitan experience; Build resume with internship; Take new/different courses; Make new/different friends; Experience different cultures

Pain Points: Limited courses offered; Costs; Organization (too much or not enough); Advantages are hidden; Challenging to transfer credits

My internship provided me with the opportunity to work in Times Square. I just love all of the lights, action, and excitement!

Julia has been taking Spanish since high school and is excited to study abroad in Buenos Aires next spring. She's traveled a little in the past—to Great Britain for a vacation with her family and to Mexico for a missions trip—but this is her first time going abroad alone. Though she has other friends who also plan to study abroad, she wanted to go at a different time so she would be forced to make friends with the locals and truly immerse herself in the culture. She's heard from friends that the maturity level of some of the students plummets the moment they step on the plane to study abroad. She hopes they don't make her look like a "stupid American."

She's also heard that the dorms in Buenos Aires aren't great, which solidified her decision to do a homestay. However, she's concerned about commuting to classes, which she hopes to take at the NYU campus as well as a local university—if the credits transfer. She doesn't have a lot of extra cash and is interested in a work study to pay for souvenirs and some travel around Argentina. Speaking Spanish on the job would also be great practice, but she isn't sure what sort of opportunities there are, or even if she's allowed to work.

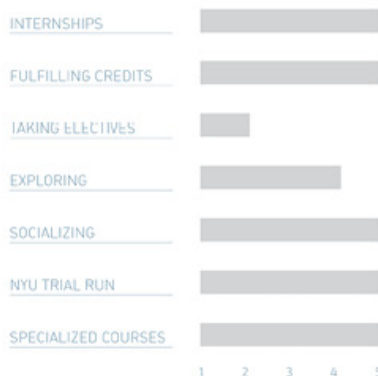
Knowledge



Lifecycle



Activities and Interest



Influencers

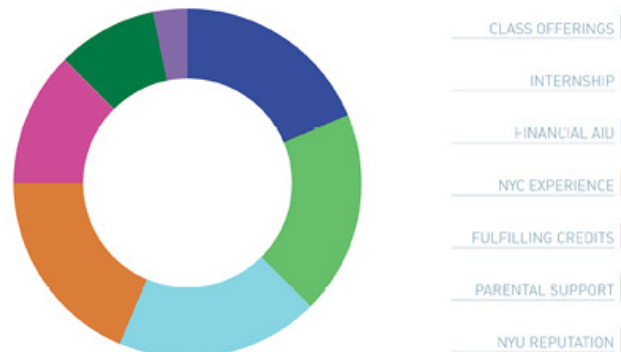


Рисунок 8.1. Персонаж описывает портрет основного пользователя в нашей целевой аудитории и одушевляет наши дизайнерские

решения. Этот пример создал Тод Заки Ворфел из messagefirst.com

Индивидуальность

Длительные взаимоотношения базируются на индивидуальных качествах и перспективах, которыми мы все владеем. Это называется «смесь личностных характеристик». Через личность мы выражаем полный спектр ощущений. Личность – это магическая сила, которая привлекает определенных людей и выделяет нас среди других. Это как знак совместимости, порождающий эмоциональную реакцию, которую мы не можем не заметить.

Мы все испытывали магию встречи с тем, чья личность просто очаровывает. Случайная встреча свела нас, а магнетизм наших личностей удерживает нас вместе. Когда мы встречаем кого-то, личность помогает нашему интеллекту проводить несложный и экономически эффективный анализ.

Хотя порой и неосознанно, но мы оцениваем мир вокруг нас, задавая себе вопрос: «Хорошо это или плохо для меня?» Личность дает нам все сигналы того, стоит ли нам общаться с новым знакомым, будет от этого польза или вред. В силу того, что личность играет очень важную роль в процессе принятия наших решений в социальных кругах, она может стать мощным оружием в создании дизайна.

Давным-давно, в далекой-далекой галактике

Не так давно те из нас, кто публиковался в Сети, были вынуждены выглядеть более пафосно перед публикой. У вас был сайт во время бума доткомов в конце 1990-х? У меня – да, и, уподобляясь многим, я писал тексты для моих сайтов с королевским «мы», чтобы создать впечатление того, что моя корпорация такая же огромная, как iXL или Razorfish^[80] (дети, спросите у родителей!). Выбросите стоковые фотографии фиктивных партнеров и встреч, и вы превратитесь в одиночку, который сидит в своей спальне и выдает себя за «всемирную корпорацию».

Таким был фасад, который я создал, потому что думал, что никто не воспримет меня всерьез, если я скажу все честно. Это было еще не все, чем я дурачил мою аудиторию. В те времена много мелких предприятий и фрилансеров «разрисовывали картины» грандиозности

масштабов и богатства своего бизнеса, а на самом деле ничего этого не было, только один-два индивида «за кулисами».

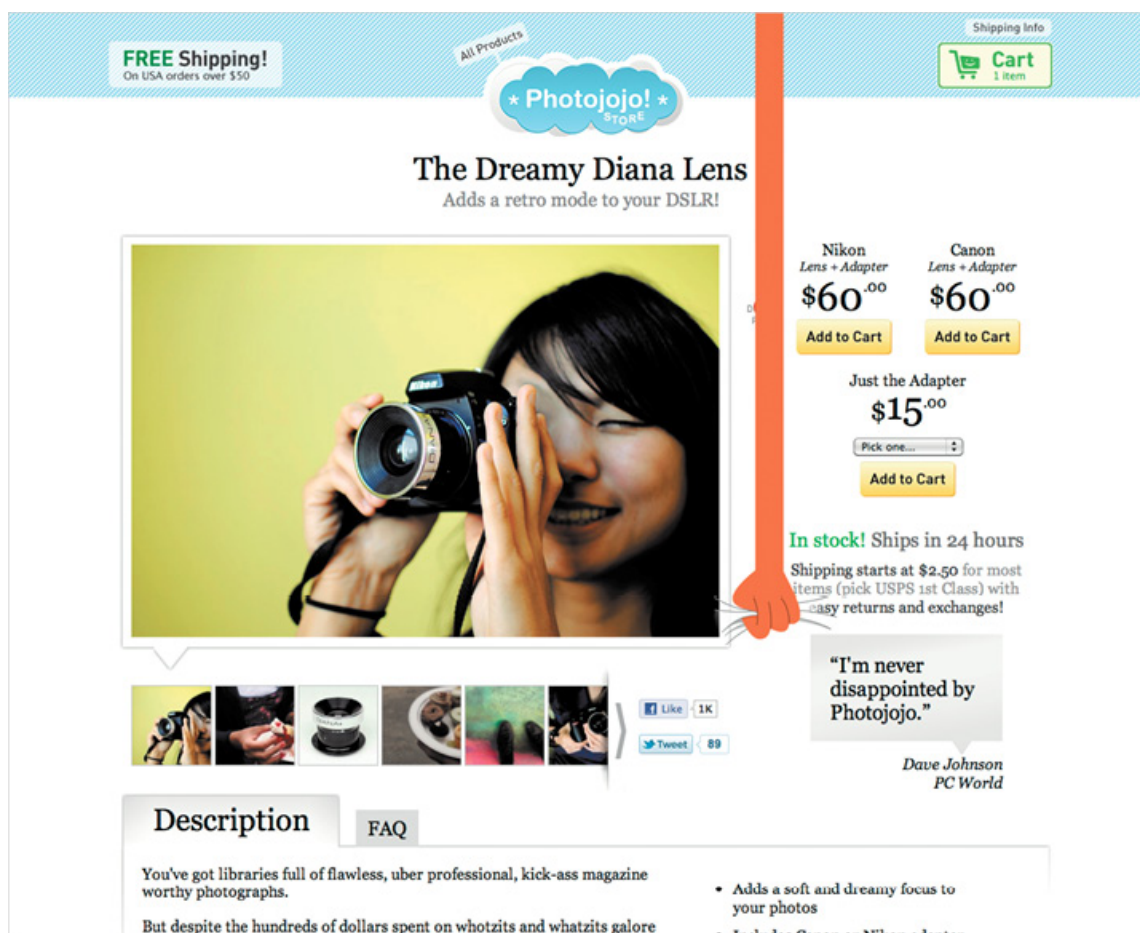


Рисунок 8.2. И Carbonmade, и Photojojo имеют яркую индивидуальность, которая выделяет их среди конкурентов и привлекает страстных поклонников

Сегодня все по-другому. Мы раскрываем нашу жизнь миру в Twitter, Facebook и других социальных сетях. Мы больше не создаем отдельных персонажей для личной и профессиональной жизни. Линия, которая когда-то разделяла эти два понятия, сейчас размыта, и хорошо это или плохо, теперь мы предлагаем миру невымышленную личность.

Раз мы раскрылись друг перед другом, то ждем того же и от брендов, с которыми взаимодействуем. В мире, ориентированном на потребление, где нам «толкают» товар, а компании, вместо того, чтобы отдавать только берут, мы жаждем реального человеческого общения, вежливого и обоюдного.

Некоторые компании поняли это и сейчас налаживают эмоциональные связи со своими клиентами с учетом их индивидуальности. Мы сейчас созрели до того, чтобы ожидать этого от маленьких «стартапов», таких как Carbonmade^[81] – это забавное веб-приложение, которое помогает дизайнерам создавать портфолио, и Photojojo^[82], площадки для фотографов. Эти компании свободны от корпоративных ограничений и переполнены креативностью. Их дизайн и маркетинг блещет юмором, что выделяет их среди конкурентов. Просто индивидуальность – это мощное оружие при создании дизайна, которое работает даже с гигантскими конгломератами.

Компания General Electric, компания из списка Fortune-100 со штатом в 287 000 сотрудников (хм, людей в 36 раз больше, чем в городе, где я вырос), выглядит гигантом по сравнению с почти любой компанией на планете. Многие корпоративные монстры любят размещать на своем сайте стандартные тексты, стоковые фотографии и обещания. Но GE пробует кое-что другое. Она рассказывает истории устами своих сотрудников о том, как изменяет мир.

В видео на домашней странице GE^[83] рабочие из Дарема (штат Северная Каролина, США) делятся с нами вкратце рассказами о своей работе, о том, как они создают мощные реактивные двигатели, которые поднимают коммерческие самолеты на высоту 1000 метров над землей. Это видео не о том, насколько сложны технологии, заложенные в двигатели, а собрание личных историй Сеза, Марка, Карима, Тома и их коллег, которые вкладывают душу в свою работу.

В них выражается вся любовь к своему делу. Становится понятно, что эта небольшая группа людей ходит каждый день на работу не просто получать зарплату, а потому что у них присутствует чувство ответственности за пассажиров. Их волнует то, как люди полетят на самолете с двигателем их компании.



Рисунок 8.3. Компания GE представляет на своей домашней странице рассказы тех людей, которые делают ее продукцию. Индивидуальность компании освещена личностями, которые страстно привержены своему делу

Как видно из видеоролика, члены команды идут по взлетной полосе и видят, как взлетает авиалайнер-гигант, чье сердце – это двигатели, собранные ими.

Все поглощены тем зрелищем, как лайнер набирает высоту.

Чувство гордости переполняет одного из мужчин, и из его глаз текут неподдельные слезы. Тем, кто видит это, трудно не поверить в искренность и честность этих людей, как и невозможно не проникнуться ими и не воодушевиться.

Продукция и люди – одно и то же

Личность определяет степень выражения наших эмоций. Это те рамки, где мы шутим, определяем круг друзей и даже находим соратников. Это – сердце любого человеческого общения.

Стивен Пинкер, профессор кафедры психологии Гарвардского университета в своем бестселлере «How the Mind Works»^[84] («Работа

ума») отмечает: «Многие изменения в личности – около пятидесяти процентов – имеют генетические корни». И это так! В момент, когда вы входите в этот мир, половина качеств вашей личности уже предрешены свыше.

Другая половина определяется в основном влиянием общества и культуры. И только 5 % вашего характера зависит от воспитания родителями. Как отец, я считаю это до боли тягостным. Факт того, что так много в нашей личности заложено генетически, определяет стиль нашей жизни и обеспечивает выживание.

Итак, что если бы сайты, продукты и сервисы, которые мы создаем, были наделены личностными качествами? Личность – это естественный интерфейс, известный людям. Мы уже знаем, как нам реагировать на людей, которых мы встречаем, по сигналам: по интонации голоса, языку, внешности и позам. Мы подсознательно обрабатываем эту информацию и начинаем вести себя соответственно. Если мы замечаем в человеке вежливость и надежность, мы можем заболтаться и рассказать о себе многое. Если же он грубый и подозрительный тип, то мы извинимся за наш резкий уход.

Сигналы, которые мы обычно считываем с личности человека, могут тоже отразиться и в дизайне. Цвет, изображения, текст и модели общения – все это средства передачи индивидуальности. Так же, как и личность влияет на поведение людей вокруг нас, так и индивидуальность в дизайне может определить поведение посетителей сайта.

Есть четыре основных выгоды от выражения вашей личности через дизайн:

- На переполненном рынке индивидуальность поможет вам отделиться от конкурентов.
- Индивидуальность вызывает у аудитории эмоциональную реакцию, и ваш бренд запоминается надолго.
- Личность привлекает тех, кто понимает вас, и удерживает на расстоянии всех остальных.
- Индивидуальность вызывает у пользователей страсть, и они становятся самым мощным маркетинговым каналом.

Теперь каждый пункт рассмотрим в отдельности.

Один из этих сайтов не похож на другие

Неважно, какой у вас сайт, в Сети найдется десяток, похожих на него. Поставьте себя на место вашей аудитории. Как они отличат ваш сайт от других? В чем разница между ними? Компания по бронированию путешествий Hipmunk^[85] тщательно рассматривает эти вопросы прежде, чем окунуться в кишачий конкурентами рынок. Travelocity, Orbitz, Expedia и несколько других компаний одно время крепко держались за рынок онлайн-бронирования путешествий, но Hipmunk удалось выделиться из толпы.

Зайдите на некоторые известные сайты для путешествий, и вы найдете в них общие черты, которые выражают их индивидуальность. Их домашние страницы пестрят объявлениями о последних сделках по путешествиям и гарантиями на возврат денег. Растерянный пользователь отвлекается от своей основной цели, с которой он зашел на сайт, – забронировать рейс. Каждый из этих элементов чего-то требует от пользователей. «Дай мне! Дай мне!» – вопят они.

Ненавижу строить планы на путешествие. Это сплошной стресс. Утрясать расписание, выяснять ограничения по времени, платить высокую пошлину за основные услуги, да еще и бояться, что авиакомпания что-нибудь изменит, – все это вымораживает меня, и я начинаю дергаться.

Дизайн большинства сайтов туризма и путешествий, привлекающий внимание «агрессивными нападками», не улучшает такое эмоциональное состояние. А вот спокойный, сосредоточенный образ – лучшее лекарство от стресса. Это в точности то, что предлагает сайт компании Hipmunk. Его домашняя страница прямо сосредоточена на одном: выбор рейса. Забавный бурундучок добавляет необходимой ветреной легкости в противовес стрессовому взаимодействию. Нет никаких предложений о скидках, которые отвлекают или еще больше напрягают пользователя.

Результаты поиска на Hipmunk также целенаправленны и показывают пользователям свое участие, чего не наблюдается у конкурентов. Вместо простого списка рейсов и самолетов сайт представляет индекс «страданий», в котором пользователь может найти самые проблемные рейсы. Рейсы с ранней отправкой, поздним прибытием или с длительной задержкой ранжируются в этом индексе.

Кто-то может поспорить и сказать, что эта функция – просто умный шаблон дизайна, но есть кое-что еще. Индекс «страданий» буквально говорит пользователям о том, что сайт Hipmunk проникается ими: он чувствует их мучения.

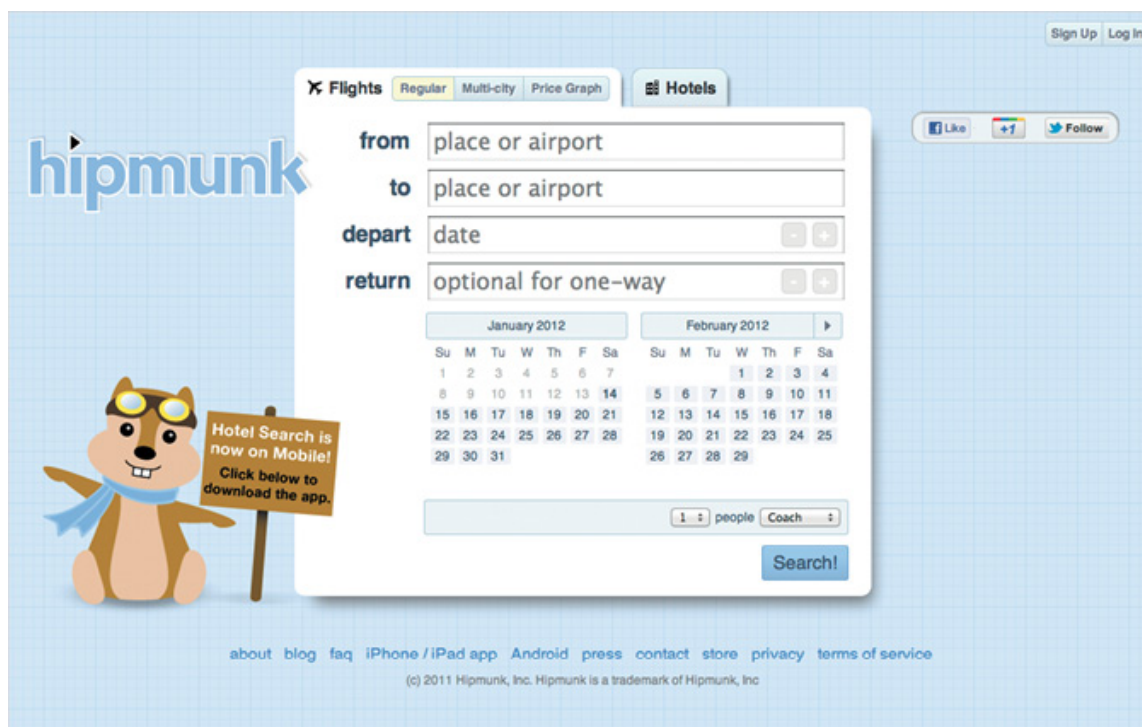


Рисунок 8.4. Забавный персонаж – бурундучок – и однонаправленность сайта – прекрасное лекарство от вывода из стрессового состояния многих людей, которые пользуются услугами сервиса бронирования поездов

Очень мало сайтов наделено такими чертами. Так же, как и акт милосердия на улице что-то говорит о личности, которая его совершает, так и сочувствующий интерактивный дизайн может передавать участие дизайнера в жизни его аудитории. Это такая штука, которую пользователи забудут нескоро.

Я помню

Эмоциональные переживания оставляют глубокий отпечаток в нашей памяти. И возникновение эмоций, и их запоминание происходят в лимбической системе – совокупности отделов головного мозга. Есть хорошая причина, по которой лимбическая система объединяет эти важные функции. Мозг объединяет эмоцию и долговременную память потому что, в противном случае, человек был бы обречен на повторение негативных впечатлений и не мог бы осознанно повторять позитивные. Как Джон Медина разъясняет в своей книге «Brain Rules: Principles of Surviving and Thriving at Work, Home and School»^[86] («Правила мозга: 12 принципов выживания и процветания на работе, дома и в школе») наш мозг тонко подмечает эмоционально наполненные события:

«Мозжечковая миндалина битком набита нейромедиаторным дофамином. И она использует дофамин так же, как если бы офис-менеджер пользовался самоклеющимися цветными стикерами для записи. Когда мозг обнаруживает эмоционально окрашенное событие, мозжечковая миндалина выпускает дофамин в систему. Так как дофамин сильно способствует обработке информации в памяти, это можно сравнить с записью на стикере: «Запомни это!» Заставить мозг химически записывать определенную информацию означает, что она будет обрабатываться более прочно. Это то, чего хочет каждый учитель, родитель и руководитель».

Впечатления, скудные эмоционально, имеют тенденцию стираться из памяти. Так что консервативный, привычный дизайн, скорее всего, забудется. Медицина красноречиво обобщает основную причину, по которой мы, дизайнеры должны включать образы и эмоции в дизайн: «Мозг не обращает внимания на скучные вещи». Когда мы делаем дизайн с индивидуальностью, мы создаем рамки, через которые эмоциональные переживания остаются в памяти наших пользователей.

Но буду честен. Риск при разработке дизайна с индивидуальностью существует. Не всем может понравиться результат. Но если вы создаете дизайн, чтобы угодить всем, вы не угодите никому. В следующей

статье мы увидим, что дихотомия позитивных и негативных эмоций вызывается выражением индивидуальности.

Люблю и ненавижу

Это нормально, когда некоторые ненавидят ваш редизайн и не реагируют на образы, которые вы включаете. Это знак того, что вы в самом деле контактируете с вашей аудиторией на эмоциональном уровне. Пренебрежение все же лучше, чем безразличие.

Демонстрация образов в вашем дизайне всегда сопряжена с некоторым риском. Одни люди чувствуют себя очень привязанными к ним, а других они не интересуют. Когда вы отдаете частичку себя миру, кто-то этого не делает.

Но это в порядке вещей. Люди, которым наплевать на ваши образы, явно вам не судьи. Они те, кто мог бы вызвать много проблем в поддержке вашего продукта или кто постоянно настаивал бы на превращение вашего продукта не понятно во что.

Если вы фрилансер или агентство и охотитесь за новыми проектами, выражая свои образы на сайте, вы можете уберечь своих клиентов от чего-то плохого. Люди, которым неинтересен ваш сайт, не захотят работать с вами. Говорю вам из своего опыта, это не конец света!

Даже процесс разработки, которому мы обычно следуем, напоминает нам, что мы не работаем на всех. Это та причина, по которой мы изучаем пользователей. Исследование действительно дает нам понять, для кого мы создаем проект. Если бы мы ставили целью делать дизайн для всех, нам не нужно было бы ничего исследовать.

В нашей личной жизни есть несметное количество случайных лиц, которых мы и в глаза не видели. Конечно, тяжело осознавать, что некоторые люди не принимают наш образ на ура. Глупо пытаться подстраиваться под желания всех, с кем нас так или иначе сводит судьба. Самое лучшее, что мы можем сделать, оставаться самими собой и верить, что есть люди на планете, которые принимают нас такими, какие мы есть. (Бог мой! По-моему, мы под шумок внесли лекцию о жизни в книгу о дизайне, вам не кажется?)

Эта лекция имеет отношение к дизайну. Образы помогут вам отфильтровать свою аудиторию на тех, с кем у вас общие ценности,

интересы и цели. Эти люди – ваши пользователи. Вот им и надо стараться угодить. А они, в свою очередь, открыто выразят всю свою любовь вашему бренду.

Как ведущему дизайнеру пользовательского опыта в Мейлчимпе, мне это известно из первых рук. Юмор, остроты и веселые времена, что я переживаю каждый день с моими коллегами, отражаются в текстах, иллюстрациях и интерактивном дизайне, которые мы создаем. Приколы Фредди фон Чимпенхаймера IV, нашего талисмана, шимпанзе, который сидит вверху каждой страницы приложения, собраны людьми нашей компании. Так выражается наше чувство юмора. Когда вы взаимодействуете с сайтом MailChimp, вы взаимодействуете с теми, кто его создавал.

Мы слышали от некоторых людей что юмор, «вплетенный» в сайт, раздражает или отвлекает. Если ваш образ нестандартный, готовьтесь к тому, что можете получить такую колкую обратную связь.

Мы заволновались, что такое восприятие широко охватило наших пользователей, поэтому провели кое-какие исследования. Мы придумали опцию, которая называется режим «кислой мины». Он блокирует шутки Фредди и закрывает неформальные тексты приложения.

Проследив, как он используется, мы обнаружили, что только 0,007 % нашей аудитории отключают наш персонаж. Из этого урока мы вынесли то, что незначительная критика – далеко не причина для того, чтобы менять себя.

Мы поняли, что выражение вещей через образы приносит намного больше выгоды, чем риска. Это выделяет наш бренд в среде конкурентов. Многие наши клиенты влюбляются в наш продукт. Да, кстати, и это не сильно вредит нашему маркетинговому бюджету. Об этом в следующем разделе.

Наши пользователи умеют обращаться со слухами

Образы в дизайне помогают распространить слух о нас. Не надо никакой дополнительной рекламы. Wufoo^[87], разработчики веб-приложения, которое упрощает создание веб-форм и управление собранными данными, полностью ликвидировали свой маркетинговый

бюджет, когда увидели, что пользователи рекламируют приложение за них.

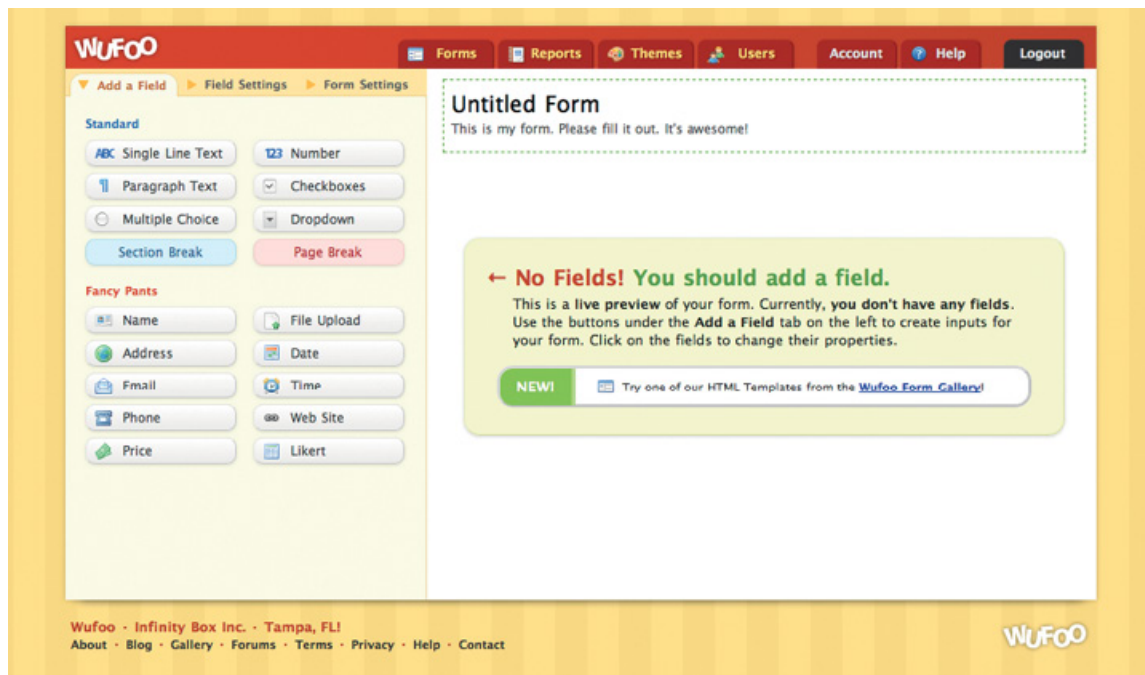


Рисунок 8.5. Уникальный метод разработки дизайна приложения Wufoo, сосредоточенный на бизнесе, вызывает восхищение и восторг пользователей, которые горят желанием поделиться с миром их положительными впечатлениями

Записи в Twitter, посты в блогах и простая молва привлекли новых пользователей более эффективно, чем баннеры. Неизбитый дизайн сайта, остроумный текст и озорник Динозаврик резко выделяют Wufoo в среде конкурентов, которые с большой неохотой используют образы. Пользуясь сайтом Wufoo, мы легко можем увидеть образы тех людей, кто его создавал^[88]. Владение продуктом, любимым пользователями и который постоянно растет за счет сарафанного радио, сделало Wufoo привлекательным и ценным приобретением для компании SurveyMonkey^[89], что купила его в 2011 году приблизительно за 35 миллионов долларов.

Образ: уже не только для дизайнеров

Уважаемые дизайнеры пользовательского опыта, такие как Стивен Андерсон и Энди Бад, размышляют и говорят о связи между образами, эмоциями и дизайном вот уже несколько лет. Тема нашла поддержку в дизайнерской среде. Но мы не единственные, кто осознает силу образов на переполненном рынке. Люди, которые создают стартапы, тоже понимают это.

В своем блоге Фред Уилсон, венчурный капиталист и директор компании Union Square Ventures, выступает за ясность речи и образов в дизайне продукта, и высказывает мысль, что это является «решающим в создании успешного продукта»^[90].

Дейв Мак-Клер, «инвестор-ангел» и основатель бизнес-инкубатора из 500 мелких компаний, придерживается того же мнения. Он рассматривает образы и эмоции как важные факторы успеха продукции. В своей речи на конференции Warm Gun conference в 2011 году Мак-Клер обратился к дизайнерам и предпринимателям: «Будьте собой, суперсамими собой. В чем вы можете быть искренними и где вам уютно? Ищите свою сущность и расширяйте ее. Найдите чувство или эмоции, которые вы можете поддерживать»^[91].

Бизнес-СМИ также раскрывают роль образов в дизайне. Спонсор журнала Forbes Энтони Винг Коснер подтверждает, что образы могут сделать сайт более запоминающимся:

«Почему сайты одних компаний запоминаются больше, чем остальные? На первый взгляд, может показаться, что все дело в оригинальности, визуальном воздействии и создании бренда. А если я вам скажу, что наиважнейшим фактором в создании сайта являются чувства посетителей, которые он вызывает?»^[92]

Уилсон, Мак-Клер и Коснер – все они описывают что-то, с чем мы ознакомились в этом разделе ранее. GE создала более человеческое восприятие своей гигантской корпорации через образы людей, работающих в ней. Она показала нам страсть и гордость за свое ремесло, и свои ощущения.

Неформальный персонаж сайта Nirmunk смягчает напряжение и негативность, которую испытывают путешествующие при бронировании рейса. Индекс «страданий» буквально ранжирует результаты по эмоциональной реакции, что не просто помогает

клиентам принять решение, но и вызывает у них чувство благодарности за такое участие сайта Hirtmunk.

Компания Wufoo аннулировала полностью свой маркетинговый бюджет после того, как поняла, что страстные пользователи распространяют хорошую славу о ней. Образ в приложении освещает дни тысячи работников, которые заняты сбором данных по приказу своих корпоративных боссов, сидя в серых кабинках. Легкость, цвет и образы сайта для пользователей – словно глоток воды для странника в пустыне.

Раньше образы и эмоции были инновацией мелких сайтов, создаваемых одной или двумя креативными личностями. Сегодня мы наблюдаем, как образы осторожно вливаются в сайты многих брендов, маленьких, больших, гигантских и всяких разных. Дизайнеры не единственные, кто видит ценность в том, чтобы оживить впечатления пользователей; сейчас инвесторы понимают, что образы – это ключевой фактор в успехе продукции.

Однако начало переделки дизайна с мыслями об образах может быть туманным. Четкое определение характера вашего персонажа перед тем, как погрузиться с головой в Photoshop или Illustrator было бы гораздо полезнее.

Вот для чего именно нужны персонажи.

Определяем ваш образ с помощью дизайн-персонажа

Ранее в этом разделе мы говорили о персонаже пользователя, что составленном на основании исследований вашего реального пользователя. Это понятие отвечает на вопрос, кто они. А дизайн-персонаж зеркально отражает его, спрашивая: «Кто мы?» Как личность пользователя, так и личность дизайнера устанавливают параметры для процесса разработки. Они помогают нам перебороть синдром пустого холста.

Когда столько возможностей, с чего же начать? Понимая и себя, и пользователей, море вариантов больше не кажется таким безбрежным, а курс на нужный становится яснее.

Подумайте вот о чем: если бы ваш сайт был личностью, то какой? Была бы это серьезная, строгая, всесторонняя, надежная и одаренная персона? Или остроумный паренек, который превращает обычные

задачи в забавы? Дизайн-персонаж это документ, в общих чертах охватывающий ключевые характеристики образа, который вы хотите включить в дизайн. Совсем скоро мы увидим живые примеры дизайн-персонажа, но сначала давайте посмотрим на структуру документа.

Дизайн-персонаж состоит из девяти частей^[93]

1. Название бренда

Название вашей компании или продукта.

2. Обзор

Краткий обзор образа вашего бренда. В чем его уникальность?

3. Образ индивидуальности/личности

Это реальный портрет личности, в которой вы хотите воплотить определенные характерные черты и включить ее в сайт. Это делает образ менее абстрактным. Выберите известную личность или кого-то, кто знаком вашей команде. Если у вашего бренда уже есть талисман или представитель, который выполняет эту роль, используйте его. Опишите качества талисмана, которые соответствуют образу бренда.

4. Качества бренда

Составьте список из пяти-семи качеств, которые лучше всего описывают ваш бренд, включая нежелательное. Это поможет тем, кто разрабатывает и пишет сайт создать последовательный образ, избегая тех качеств, которые собьют ваш бренд с правильного пути.

5. Схема образа

Мы можем изобразить образ графически. Ось X изменяется от враждебной к дружелюбной, а ось Y – от покорной до господствующей^[94].

6. Речь

Если бы ваш бренд мог говорить, как бы он это делал? Что бы он сказал? Разговаривал бы он компанейским простым языком, или утонченным?

Опишите специфические аспекты голоса вашего бренда и как он может изменяться в зависимости от ситуации. Люди изменяют свой язык и тон по ситуации, и так должен делать ваш бренд.

7. Примеры текста

Подготовьте примеры текстов, которые могли бы использоваться в различных сценариях на вашем сайте. Это поможет авторам понять, как должен взаимодействовать дизайн.

8. Визуальный словарь

Если вы создаете этот документ для себя как дизайнер или для команды дизайнеров, разработайте такой словарь, который объединяет в единое целое цвета, типографику и визуальный стиль образа вашего бренда. Вы можете быть универсальны в концепции или включить карту настроений.

9. Методы привлечения

Опишите методы, которыми вы могли бы эмоционально привлечь ваших пользователей, чтобы создать незабываемые впечатления. Пачка карточек с ментальными заметками Стивена Андерсена очень удобное собрание таких методов^[95].

Разработка дизайн-персонажа

Процесс создания дизайн-персонажа столь же важен, как и его результат. Когда вы перестаете рассматривать характерные особенности, которые хотите внести в свой дизайн, вам становится ясно, что вы могли забыть, если бы прямоком окунулись в ваш любимый Photoshop. Определение образа путем мозгового штурма поможет авторам текстов, дизайнерам, информационным архитекторам и разработчикам воспринимать ваш сайт как личность и увидеть границы, в которых они работают.

Чтобы начать работать над вашим дизайн-персонажем, сначала скачайте образцы и шаблоны^[96]. Если вы работаете в команде, соберитесь все вместе с легкой закуской и маркерной доской, чтобы проработать первоначальные идеи по поводу образа. Будет весело, и никому не повредит холодильник со взрослыми напитками.

Начните дискуссию с вопроса: «Какие семь слов могут описать нас лучше всего?» Будьте честными. Вы можете услышать такие характеристики, от которых не загордитесь. А теперь уточните: «Какие семь слов могут описать того, кем мы надеемся стать после редизайна?» Переделка сайтов по своей сути желательна и полезна, так как основывается на наших стремлениях.

Давайте посмотрим, совпадают ли эти два списка. Обсудите, как вы могли бы видоизменить качества, которые вам не нравятся, в положительные характеристики. Если оставшийся список содержит больше семи характерных особенностей, продолжайте «урезать» его

до семи или меньше, потому что образ может стать размытым и неискренним, если вы попытаетесь стать всем для всех.

В окончательном списке, «нацарапанном» на доске, отразите границы каждой характеристики. Каким бы вы не хотели видеть свой образ? Например, если в списке стоит характеристика «веселый», это может означать много чего. Веселый, как Элмо^[97], или веселый от того, что с ветерком едешь за рулем Ferrari California вдоль Блуриджской парковой автострады? Рамки привнесут ясность в доску дизайн-персонажа и помогут вам увидеть во время проекта, когда пересекается граница.

Чтобы лучше понимать, как работают характеристики и рамки, посмотрим, как мы определили их для образа MailChimp:

1. **Веселый**, но не по-детски.
2. **Забавный**, но не тупой.
3. **Функциональный**, но не сложный.
4. **Неформальный**, но не разгильдяйский.
5. **Простой**, но не простецкий.
6. **Надежный**, но не нудный.
7. **Неформальный**, но не чуждый.

Этот маленький список представляет собой систему ценностей. Он говорит вам о том, кто вы, сопровождает ваш голос, и помогает сформировать впечатления публики от вашего бренда.

Теперь, когда вы знаете, какие качества образа вы хотите или не хотите, можете подумать о личности – какой-нибудь знаменитости или знакомом. Сможет ли он/она стать отправной точкой для вас и вашей команды? Оживление этих характеристик упрощает ответ на вопрос в ходе разработки: «Что бы сделал мой персонаж в этой ситуации?»

Напишите обзор характеристик, чтобы затем конкретизировать их. Опишите голос вашего образа, напишите примеры текста. Как образ выражен в цвете, типографике и визуальном стиле? В конце этого задания вы многое узнаете об отправной точке для дизайна.

При создании образа для вашего сайта помните одно важное правило.

Отражайте образ компании, в которой вы работаете или свой собственный честно. Будет сложно оставаться верными дизайн-персонажу, если образ составлен с натяжкой. Во всех примерах,

которые мы видели в этом разделе раньше, образы людей, которые создавали сайты, были настоящие. Они не были идеалистическим вымыслом.

Мы можем сказать сразу же, когда человек выдает себя не за того, кто он есть на самом деле, то же самое действует и в дизайне. Самое главное в передаче образа в дизайне – вложить больше себя, так, чтобы мы могли сковать значимые связи с другими людьми. Фальшь здесь не пройдет. Не потому, что ее трудно создать, а потому что вы потеряете нить, связывающую вас с идеальной аудиторией.

Когда мы делали нашего дизайн-персонажа в MailChimp, мы просто документировали коллективный образ людей нашей команды. В действительности, мы обнаружили, что задача не была сложной, потому что наняли людей с одинаковыми взглядами, одинаковым чувством юмора и другими качествами. Хотя наша приоритетная цель была найти людей, которые сработаются, процесс также оказался полезным для определения образа бренда. Представляем вам дизайн-персонаж [\[98\]](#).

Название бренда

MailChimp.

Обзор

Фредди фон Чимпенхаймер IV – лицо MailChimp и воплощение образа бренда. Крепкая фигура Фредди символизирует мощь приложения, а его поза демонстрирует, что он находится в движении. У Фредди всегда милая улыбка, которая привлекает пользователей, и они чувствуют себя уютно на сайте.

Мультипликационный жанр говорит о том, что бренд предлагает непринужденное и веселое взаимодействие. Да, это обезьяна – мультяшка, но все равно Фредди такой классный! Он любит отпускать остроумные шуточки, но когда ситуация серьезная, потехи прекращаются.

MailChimp часто подбрасывает пользователям забавные «пасхальные яйца» [\[99\]](#) или ссылки на смешные видеоролики на YouTube. Веселье и забавы живут в каждом уголке сайта, но никогда не станут мешать вашей работе.

Характеристики бренда

Веселый, но не по-детски. Забавный, но не тупой. Функциональный, но не сложный. Неформальный, но не разгильдяйский. Простой, но не простецкий. Надежный, но не нудный. Неформальный, но не чуждый.

Голос

Голос MailChimp знакомый, дружелюбный, а главное – человеческий. Понятно, что за брендом на самом деле стоят образы людей. MailChimp отпускает шуточки (вы сможете рассказать их маме), рассказывает истории и говорит в непринужденной манере, в какой вы могли бы разговаривать со старым приятелем. MailChimp употребляет конструкцию don't вместо do not, потому что так говорят нормальные люди. MailChimp вставляет неформальные междометия типа «хм-м-м», когда думает, и «Блин! Это ужасно!» – чтобы выразить сочувствие.

Примеры текста

- Сообщение об успехе: «Дай пять! Твой список импортирован».
- Сообщение об ошибке: «Упс! Кажется, ты забыл ввести адрес электронной почты».
- Критический отказ: «Один из наших серверов временно не работает. Наши инженеры уже работают и скоро все исправят. Спасибо за понимание».



Рисунок 8.6. Досье дизайн-персонажа, созданное для образа сайта MailChimp

Визуальный словарь

- **Цвет.** Яркая, но все же слегка ненасыщенная палитра MailChimp передает радость – цвета чистые, но не как в детских телепередачах. Следуя характеристикам бренда, цвета воплощают юмор, и при этом они действенные и изысканные.

- **Типографика.** MailChimp легкий в обращении, эффективный и простой. Это все отражается в его типографике. Простые шрифты без засечек в заголовках и сообщениях отличаются размером, плотностью и цветом, которые передают иерархию сообщений. MailChimp представляется как привычный уютный, теплый свитер, который выполняет свои функции и в то же время любим.

- **Основной стиль.** Элементы интерфейса – простые и ровные, делают восприятие всего доступным и не отпугивающим. Мягкая, утонченная текстура может проявиться в каплях, согревая пространство. Фредди используется изредка, и то, только чтобы впустить струйку юмора. Он не влияет ни на обратную связь или статистику, ни помогает выполнять задачи.

Методы привлечения

- **Сюрприз и развлечение.** Тематические экраны напоминают о праздниках, культурных событиях или вашем любимом человеке. «Пасхальные яйца» создают эффект неожиданного юмора, иногда передавая ностальгию или обращение к штампам поп-культуры.

- **Предвкушение.** Случайные веселые приветствия Фредди вверху каждой основной страницы создают предвкушение следующей. Эти приветствия не несут информацию или обратную связь. Это просто веселые вставки, которые никогда не помешают функциональности и удобству использования.

Мы создали это досье дизайн-персонажа в основном как руководство для дизайна приложения. Это помогло нам озвучить то, что мы хотим для таких важных элементов, как сообщения об успехе и ошибке, и помогло нам пересмотреть малые элементы дизайна, которые начали выпадать из нашего образа.

Дизайн-персонаж – это НЕ руководство по стилю. В нем нет того, каким должен или не должен быть логотип. Нет специальных указаний ни на цвет, ни на типографику. Это просто краткое изложение духа дизайна. Когда вы творите дизайн-персонажа, вы многое раскрываете о себе и о своих коллегах и потом сможете поделиться этим со своей аудиторией. Это не документ, которому надо придавать законную силу, а компас, который поведет вас в правильном направлении.



Рисунок 8.7. Талисман Фредди фон Чимпенхаймер IV, талисман (лицо) MailChimp

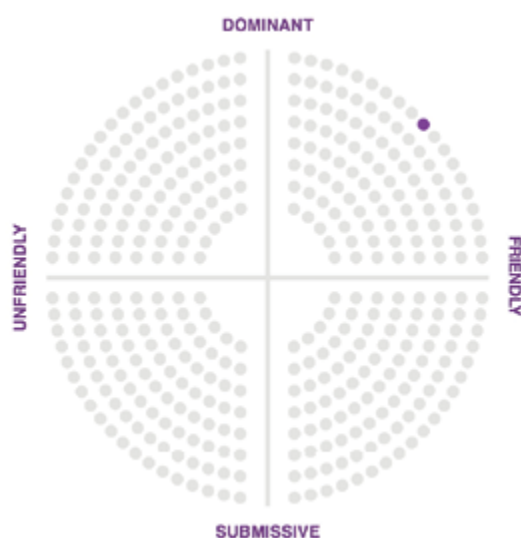


Рисунок 8.8. Схема образа MailChimp

Держа образ в голове, мы начали осознавать его присутствие во всем, что создаем, особенно, пишем. Посты в блог, тексты о продукте, статьи базы знаний руководство по поддержке – везде есть наш образ. Хотя мы не всегда делали все классно. Мы допускали и неуместный юмор. Когда пользователь нервничает, потому что ему, кровь из носа, надо выполнить задание в срок, а он никак не может сосредоточиться, шутки и неформальный тон не к месту.

Хотя дизайн-персонажа помог установить наш голос, мы увидели, что нужна еще ясность. Наш контент-куратор Кейт Кифер Ли сделала вывод, что несмотря на то что голос бренда должен оставаться единым по тембру в дизайне и письме, тон должен изменяться в зависимости от эмоционального состояния пользователя. Установить связь между голосом и тоном штука хитрая, но Кейт нашла интересное решение.

Задание голоса и адаптация тона

Вы когда-нибудь писали руководство по стилю, которое закладывает стандарты дизайна или текстов для компании? Руководства по стилю устанавливают мириады сценариев, в которых дизайн или текст могут пострадать, и они задают рамки, в которых можно долго топтаться на месте.

Многие переделки начинаются с руководств, кишащих благими намерениями, но в большинстве своем проекты забрасываются, потому что постоянное обращение к руководству непрактично. У людей создается впечатление, что у них за плечами стоит «полиция нравов». Кейт Кифер Ли написала для нас содержательное руководство по стилю, которое помогает опытным и новоиспеченным авторам. У нее были некоторые оговорки по тому, как коллеги будут воспринимать его. Будут люди использовать его реально, или для них это будет обузой? В плане грамматики и пунктуации сложностей не возникло, но вот как было донести до коллег взаимосвязь между голосом и тоном?

Когда Кейт объединила посты в блог, базу знаний, средства обучения и руководства, чтобы оценить голос компании, она отметила участок, где голос MailChimp был чистым, без интонации.

Стали бы вы шутить, чтобы утешить вдову на похоронах? Или использовать неформальную речь на встрече с королевой Англии?

Думаю, что нет! Естественно, вы бы захотели настроить свой тон по ситуации.

Выше, в теме о дизайн-персонаже, мы включили несколько примеров текста, чтобы показать стили языка и частично варианты тона в различных ситуациях. Сообщения об ошибке в приложении не место для шуток, потому что пользователь в это время растерян и напряжен. Но вот сообщение об успехе – отличное поле для юмора и неформального тона, потому что у человека сейчас позитивные эмоции.

Вместо того чтобы писать сухие руководства по стилю управляющие общением, Кейт решила прибегнуть к другому методу, который бы помог понять голос бренда и вариации тона легко и непринужденно. В команде с коллегами она создала «Голос и тон»^[100], интерактивное руководство, которое привязывает голос и тон к эмоциональному состоянию читающего.

Этот метод определения образа и стиля написания был воспринят хорошо. Просто понимать, весело использовать. Нет кучи правил и ограничений. Никто не хочет быть под прицелом «полиции нравов». Это руководство дает людям свободу писать, используя голос MailChimp, когда тон правильно подобран.

Кейт увидела тесную связь между голосом и индивидуальностью: голос тесно привязан к образу. Он – выражение нас. Наша перспектива и то, что мы привносим в каждый кусочек контента, который будут читать наши клиенты. Голос исходит у нас изнутри. Это все о нас. А тон – это то, что мы подстраиваем под чувства наших читателей.

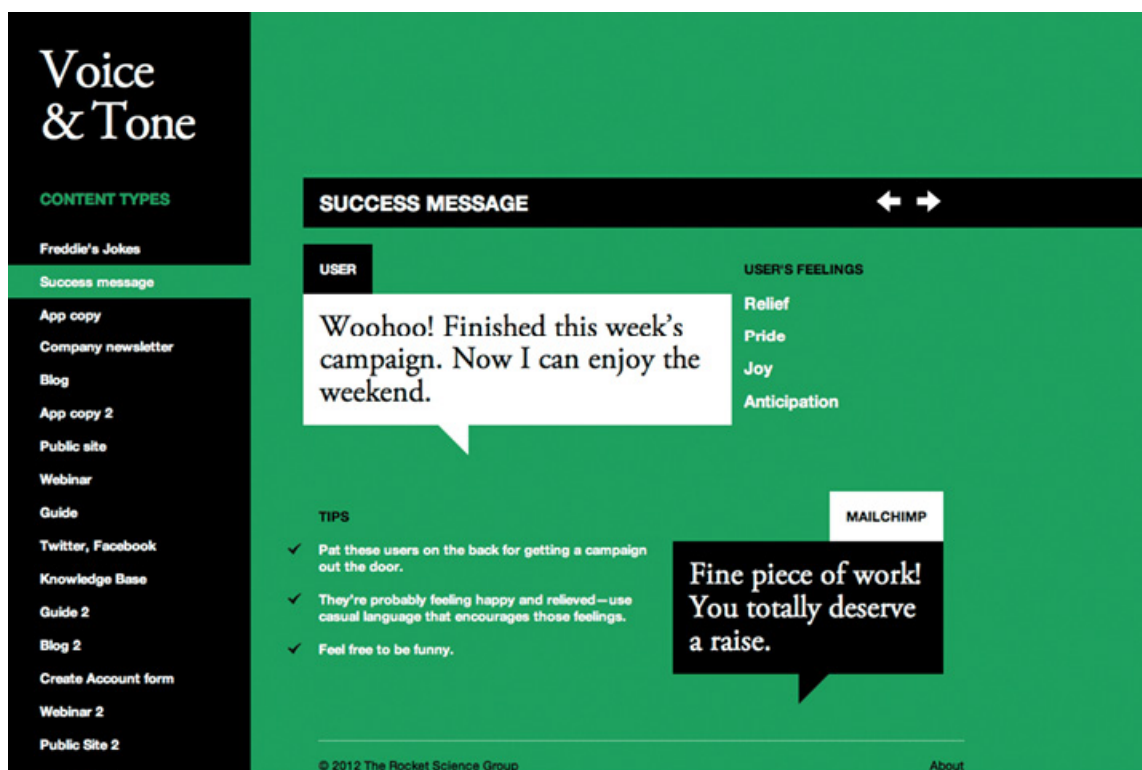


Рисунок 8.9. Руководство «Голос и тон» помогает всем, кто пишет для сайта MailChimp, поддерживать голос и индивидуальность бренда при адаптации тона под эмоциональный настрой читателя

Примеры текста в руководстве «Голос и тон» не только определяют чувства читателей в конкретных ситуациях, но и сопровождаются соответствующими цветами на фоне страницы. Красный фон символизирует злость и расстройство, в то время как зеленый — восторг и радость. Как любое руководство по стилю, «Голос и тон» помогает авторам устанавливать и поддерживать постоянство, часто проповедуемое качество бренда, которое возвращает доверие пользователей и укрепляет взаимоотношения с ними. Руководство так же обращает внимание на важность адаптации. Вместо того чтобы быть категоричным в стиле, оно просто представляет спектр стилей общения, которые авторы смогут применять в различных ситуациях^[101].

Заключение

Очень часто в нашей индустрии мотивацией к началу проекта является использование новой технологии или приемов. Несмотря на то что расширение наших знаний – это прекрасно и мы это любим, техника все же не основной повод для этого. Истинной наградой нашей работы являются отношения, которые мы строим в Сети с нашими коллегами – дизайнерами, разработчиками и людьми, для которых творим. Чем больше мы вкладываем своей души в наш сайт, тем прочнее эти связи.

В этом разделе мы рассмотрели много примеров образа/индивидуальности и много способов его совершенствования. Nipmunk выражает участие через индекс «страдания». Wufoo удалось аннулировать маркетинговый бюджет после введения своих образов в интерфейс и позволить клиентам говорить о себе. MailChimp оформил свои образы в дизайн-персонаже и в интерактивном письменном руководстве.

Эти примеры даны здесь как совет вам не только делать то же самое в вашем следующем проекте, но и зажечь новые идеи в вашей работе и помочь вам разглядеть силу образа в дизайне. Дизайн, который на переполненном рынке станет выдающимся в силу присутствия в нем ясных образов.

Он вызовет эмоциональную реакцию аудитории, и ваш бренд запомнится надолго. Он привлечет желанных вам людей и отпугнет обременительных. Он внушит людям страстную любовь, и они станут для вас самым мощным маркетинговым каналом.

Начиная переделывать очередной дизайн, задайте себе несложный вопрос: «Кто я и что хочу сказать?» Как только вы найдете на него ответ, появятся элементы вашего дизайна и стиль написания, которые принесут образ в сердце вашей работы.

Об авторе



* * *

Аарон Уолтер родился после динозавров, но до появления Интернета, в городе, недалеко от места, где через 200 лет родится капитан Кирк. Его образование включает степень бакалавра изобразительных искусств, Университет Айовы в США, степень магистра изящных искусств, Тайлерская школа искусств (штат Техас, США). Он живет в Афинах (штат Джорджия, США) где музыка такая же прекрасная, как и люди. Аарон – ведущий дизайнер пользовательских интерфейсов в команде MailChimp. Он написал две книги, последняя – «Эмоциональный веб-дизайн» (Designing for Emotion, 2011). До прихода в MailChimp в 2008 г. восемь лет он проводил по стране интерактивные дизайнерские курсы для нескольких колледжей. Его любимый цвет – черный. Он мечтает стать бариста (специалист по приготовлению кофе), когда подрастет. Его совет читателям: «Пробовать лучше, чем спрашивать: „А что если?..“»

О рецензенте



* * *

Дениз Джекобс (р. 1968) родилась в Спрингфилде (штат Огайо, США). Окончила Стэндфордский университет, Университет Вашингтона в Сиэтле и Университет Поля Валери во Франции. Ее

жизненное кредо: «Глаза боятся – руки делают». Дениз живет в небольшом домике недалеко от города Корал-Гейблс в Майами (штат Флорида, США). Она почти полностью обустроила ландшафт своего двора в 2009, но затем села писать книгу. Она с тех пор так и не смогла вернуть своему саду былую славу, но все же у нее растут банановые деревья, а соседских авокадо – просто в изобилии.

Дениз выступает на конференциях, пишет книги и статьи. Она делает дизайн разных вещей и любит обучать дизайну людей. Дениз любит живое садоводство, танцы (особенно самбу и сальсу), импровизированные комедии и постановки, ей нравится делать ювелирные украшения, работать над собой и читать.

Самый важный урок, который усвоила Дениз, – нельзя объять необъятное.

Стремиться стать суперчеловеком – слишком завышенная планка. Это подходит больше к жизни, чем к работе. Ее личное пожелание читателям: «Следуйте своей мечте и страсти, мечтайте глубоко. Тогда ваша жизнь изменится в лучшую сторону».

Размышления о мобильном опыте использования в дизайне: сетевой или «родной»^[102]?

Автор: Арэл Балкан

Рецензенты: Джош Кларк, Андерс М. Андерсен

Возможно, вы знаете, что дизайн опыта взаимодействия – это дисциплина, включающая все аспекты разработки интерактивных продуктов. Хотя она объединяет важные элементы графического и видеодизайна, в основном касается интерактивного дизайна. В сфере дизайна ближайшим родственником ему является промышленный дизайн, т. е. дизайн предметов и продуктов.

Как дизайнеры опыта взаимодействия, мы создаем виртуальные продукты. Более того, так как дизайн оборудования и дизайн программного обеспечения по существу связаны и нераздельны, грань, которая разграничивает дизайн продукта и интерактивный дизайн (если она вообще существует), – расплывчатая.

Веб-разработчик – это разработчик опыта пользовательского взаимодействия

По существу, веб-разработчик – это разработчик, создающий опыт взаимодействия, владеющий специальными знаниями о средствах Интернета. В качестве материалов веб-разработчик использует основные (HTML, CSS, JavaScript) и вспомогательные (LESS, Stylus и т. п.; HAML, Jade и т. п.; jQuery, MooTools и т. п.) фреймворки Сети и их компоненты. Эти фреймворки и компоненты внутри них созданы из кода. Того самого, который определяет границы дизайна и поведение этих материалов.

Так же как конструктор автомобилей глубоко разбирается в различных материалах, из которых делают машину, веб-разработчик должен понимать суть материалов, которыми он создает сайт.

Мы интерактивные разработчики и нас интересует не просто эстетика интерактивного объекта, но и его поведение.

Особенно это важно, когда вы разрабатываете приложения (которые являются объектами, на основе поведения) в противовес

разработке документов (которые являются объектами, на основе содержания).

Дизайн документов против дизайна приложений

Разработка интерактивного документа для Сети – особенно адаптивным способом, – требует специальных знаний. Как минимум нужно понимать принципы адаптивного дизайна и прогрессивного улучшения^[103]. Рисование милых картинок – это искусство, а не разработка.

Интерактивные продукты или приложения – совершенно другое дело. Разработка для интерактивной среды требует знаний графического и динамического дизайна, и, что самое главное, интерактивного дизайна. Важнейшим аспектом интерактивного продукта являются его взаимодействия. Они заложены в код.

Процесс дизайна приложения не ограничивается лишь рисованием красивой картинки, как проектирование автомобиля не ограничивается рисунком машины

К сожалению, из-за увеличения числа различных специализаций в команде, роли веб-дизайнера и веб-разработчика искусственно разделились. Хотя такое разделение может быть необходимым при работе в команде, тем не менее названия специализаций определяют скорее текущие задачи для членов команды, но не копилку их знаний. Вы можете сосредоточиться на определенных областях больше, особенно в специфичных проектах. Но вы должны понимать, что первоначальная причина, по которой мы создаем продукт – удовлетворить потребности пользователей, и каждый в команде разработчиков играет важную роль и влияет на опыт пользователя. Вот почему обязательно нужно работать в маленьких междисциплинарных группах, где каждый член думает в первую очередь о пользователе.

Разработка в первую очередь для пользователей

При создании продукта дизайн руководит разработкой, а разработка одушевляет дизайн. Это циклический, повторяющийся процесс, цель которого постоянное улучшение продукта для лучшего удовлетворения нужд пользователя^[104].

В основе каждого решения, которое вы принимаете по своему продукту, должны быть потребности пользователя. В первую очередь вы должны думать об этом, и только потом о себе. Другими словами, практикуйте то, что мы называем «outside-in design» («дизайн “снаружи внутрь”»). Думайте о нуждах пользователя и их контексте, создавайте то, что он увидит и с чем соприкоснется, а уж потом переходите к решению проблем, которые предстанут перед вами.

Цель главы

Два основных решения, над которыми вы должны «поломать» голову во время процесса разработки, – делать ли продукт кроссплатформенным или «родным».

Снаружи – это хорошо. Изнутри – это плохо

Ваш первый вопрос в новом проекте: «Какую использовать серверную технологию, или как должна выглядеть схема базы данных?» Стоп! Неверный подход! Вы пытаетесь решить не проблемы пользователя, а свои собственные. Получается дизайн «изнутри наружу». А это – Очень Плохая Штука!TM

Цель этого раздела помочь вам как разработчику пользовательского опыта понять вашу среду так, чтобы вы были подкованы, отвечая на эти вопросы. Начнем с того, что посмотрим, что же именно представляет собой «родное» приложение.

Что значит «родной»?

Вы замечали, как люди волей-неволей бросаются термином «родной», не до конца понимая, что это на самом деле значит? Давайте исправим ситуацию. Начнем с того, что выясним, чем «родной» не является.

Все дело в нулях и единицах

Если уж быть педантичными, то «родной» – насколько это касается цифровых устройств – относится к наличию или отсутствию электрического тока в транзисторах наших компьютеров. Мы, как правило, визуализируем это как базовое клише цифрового вычисления:

бинарный код, серия «бегущих нулей и единиц». Мы называем эти бинарные инструкции машинным языком.



Рисунок 9.1. Да, существует большое количество устройств. Но! Наше приложение не должно поддерживать их всех. Сосредоточение на пользователе означает определение вашей целевой аудитории и оптимизацию ваших приложений под платформы и устройства, которыми они пользуются. Поддержка огромного количества платформ через прогрессивное улучшение проще, если вы первоначально создаете сайты с учетом содержания в противовес приложениям с учетом поведения.

Фото: Дэвид Джонс / smashed.by/davidjones

Когда-то компьютеры запрограммировали бинарно с использованием переключателей, но мы больше не пишем программы на таком низком аппаратном уровне. Однако каждый второй компьютерный язык программирования, который мы создавали, – язык ассемблера, Си, Питон, JavaScript и т. д. – в конечном итоге переводится на язык «бегущих нулей и единиц». А они уже обеспечивают наличие или отсутствие электрического тока в транзисторах.

Каждая из этих технологий основана на абстракциях. Например, Питон написан на Си. Цель каждого абстрактного слоя – каждого более высокого слоя в слоеном торте технологий, которые составляют современную компьютерную экосистему – упростить разработчикам создание приложений. Так что употребление термина «родной» в смысле бинарного программирования сегодня поблекло.

Итак, теперь, когда мы знаем, что такое «неродной», давайте выясним обратное.



Рисунок 9.2. Веб-технологии могут быть «родными» для определенных операционных систем. Здесь вы видите лэптоп Samsung с системой Chrome OS, чьи первостатейные граждане – HTML, CSS и JavaScript и веб-приложения. Фото: промо-изображение Google

Родной как культура

Термин «родной» относится к технологиям, т. е. языкам и объектным структурам – всему тому, что формирует культуру, язык, условные обозначения и нормы платформы. Это базовый уровень абстракции. Он включает в себя базовые символы, жесты и взаимосвязи, применяемые пользователем для взаимодействия с приложениями на установленной платформе. Эти элементы крайне важны, потому что они составляют культуру и нормы платформы^[105].

Это языки, как визуальные, так и поведенческие, которые пользователь изучает для взаимодействия при пользовании платформой. И наоборот, это также слова, фразы и концепты, которые приложения на установленной платформе применяют для взаимодействия с пользователями. Чем больше они используются и постоянны на установленной платформе, тем больше плюсов для создания приложений для нее.

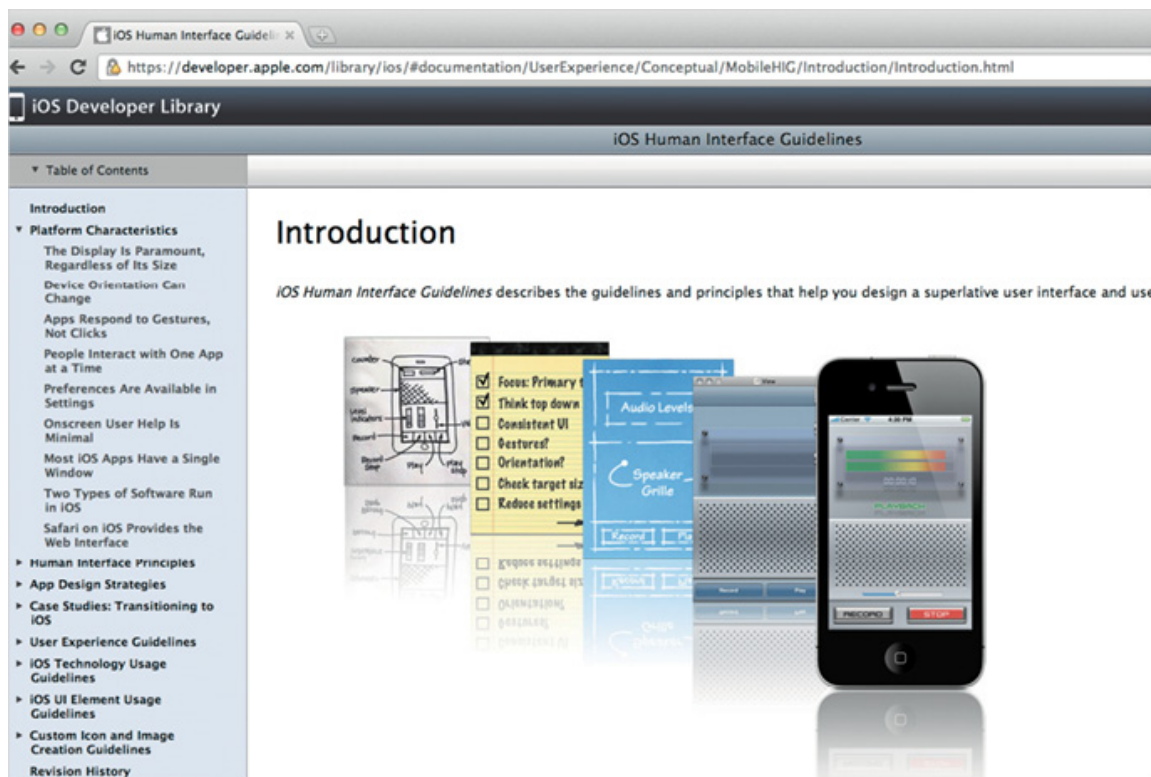


Рисунок 9.3. «Руководство по интерфейсу» от Apple дает ясные инструкции, как должно выглядеть и вести себя приложение на платформе iOS. Это поможет определить культуру платформы

В конце спектра мы имеем платформу iOS от Apple с ее подробным «Руководством по интерфейсу»^[106] и элегантным и единообразным фреймворком Cocoa Touch. «Родное» приложение от производителя, которое соответствует руководству, унаследует много от практичности базовых фреймворков и сразу же покажется знакомым пользователю, уже знающему другие приложения этой платформы.

На другой чаше весов мы имеем такие платформы, как Android. Они основательно переработаны производителями устройств,

сотовыми операторами и пользователям из-за того, что существует маленькая совместимость (если она вообще есть) между различными телефонами Android и приложениями. Разработчикам «родных» приложений для таких платформ придется затратить больше усилий, чтобы обеспечить постоянный пользовательский опыт взаимодействия.



Рисунок 9.4. Swype-клавиатура на самом деле восхитительна. Просто скользите пальцем от буквы к букве, а она автоматически будет угадывать слово, которое вам нужно. К сожалению, такая система есть не во всех устройствах Android, а лишь в некоторых. У других клавиатура похожа на клавиатуру iPhone, и пользователи могут купить и использовать кучу других сторонних клавиатур. Огромный выбор поначалу может показаться плюсом, однако это значит, что нет единого пользовательского опыта системой Android. Фактически есть столько опыта пользования платформами Android, сколько существует различных версий Android, которые адаптируются производителями, сотовыми операторами и самими пользователями. Поэтому так трудно использовать общий язык дизайна, когда создаете приложения

Например, мое iPhone-приложение Feathers^[107] имеет собственную клавиатуру, которую я сделал, чтобы вводить расширенные уникальные символы. На iPhone она работает в точности, как встроенная программная клавиатура iPhone. Чтобы добиться этого, пришлось попотеть, но невозможное возможно. Если бы я переносил приложение на Android, мне пришлось бы вычислять, какую именно из кучи программных платформ установил пользователь, а потом адаптировать ее поведение для совместимости. Не факт, что это было бы вообще реально сделать, не говоря уже о том, что точно потребовалось бы

больше усилий. Клавиатура Swype на телефонах Android, например, патентована. Поэтому на устройстве Android со Swype-клавиатурой я не смогу заставить свою вести себя точь-в-точь как системная.

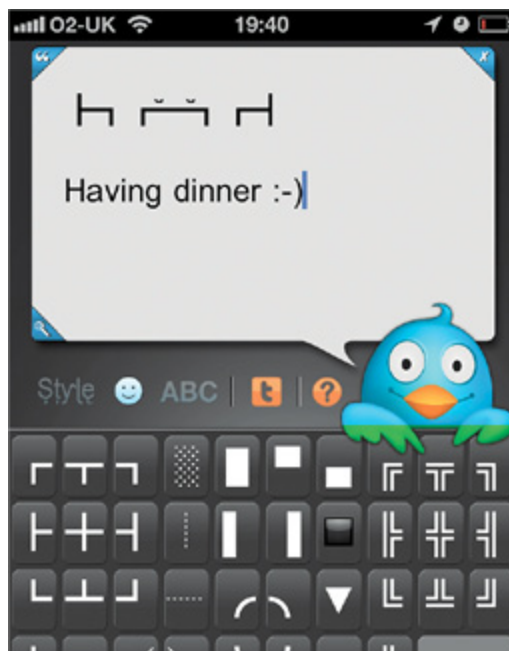


Рисунок 9.5. Портирование Feathers на Android требует создания различных версий пользовательской раскладки клавиатуры под различные типы клавиатур

Компромиссное решение — создать единую пользовательскую клавиатуру и использовать ее независимо от системной клавиатуры пользователя. Конечно, она не будет ни смотреться, ни ощущаться как основная клавиатура и поэтому не будет обеспечивать такое же удобство работы, как и в Feathers на iOS. Вместо этого пользователям приложения придется учиться управлять двумя разными видами клавиатур в приложении и напрягаться при переключении с одной на другую.

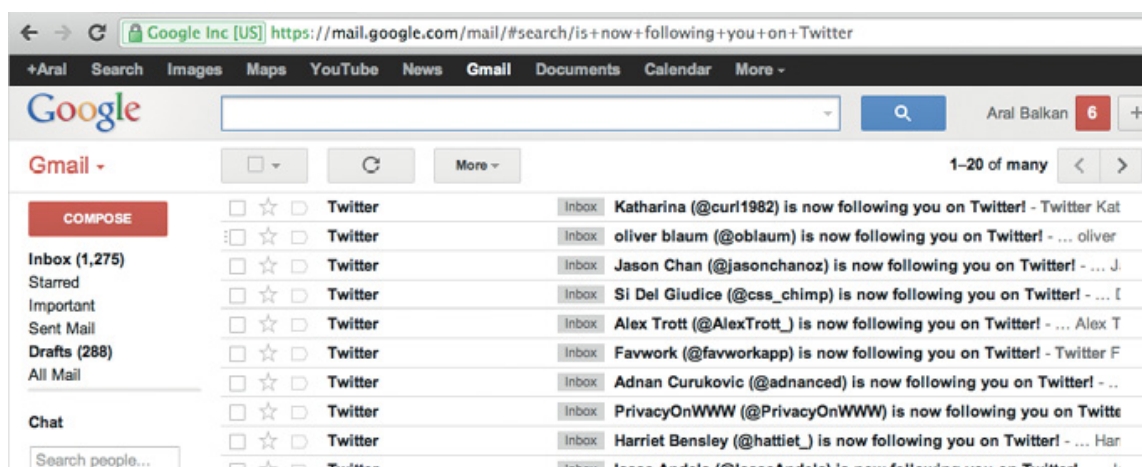


Рисунок 9.6. Веб-приложение Gmail обеспечивает постоянный опыт взаимодействия на всех платформах. Пользователям не нужно ничего устанавливать или беспокоиться о синхронизации своей почты на разных устройствах

Аналогично на платформах, в которых нет сильного и постоянного визуального языка и культуры, «неродные» приложения будут иметь меньше недостатков и даже смогут, как это ни парадоксально, обеспечивать больше удобства при использовании в некоторых ситуациях.

Великолепный пример «неродного» приложения, которое обеспечивает лучший опыт взаимодействия на определенных «родных» платформах, – это Gmail. Использование почтового клиента компьютера в операционной системе, такой как Windows, может потребовать от вас установки самого приложения, его обновления, хранения вашей почты, синхронизации в различных устройствах. Нужно будет проверять, нет ли в сообщениях вирусов и другого вредоносного ПО. Оцените простоту Gmail, когда вводите URL-адрес в браузер любого устройства – раз! – вот и ваша почта. Все! Все отлично!

Gmail также прекрасный пример того, как создание хорошего кросс-платформенного опыта взаимодействия может требовать много оптимизаций, характерных для разных платформ. Хотя приложение Gmail работает на настольных и мобильных платформах, фактически у него есть несколько высоко оптимизированных версий.

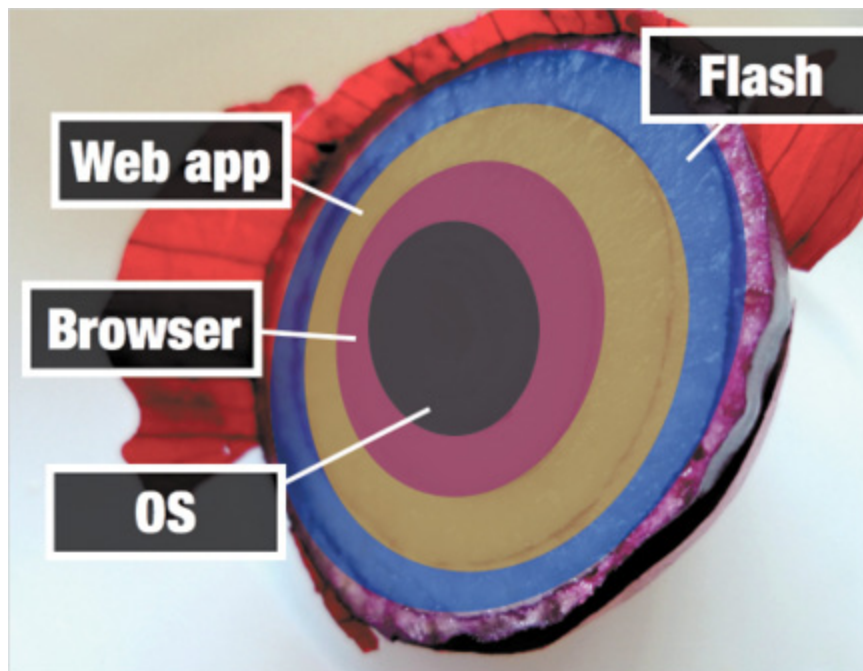


Рисунок 9.7. Браузер – «родное» приложение операционной системы (ОС), работающий в ее контексте. Веб-приложение работает в контексте браузера. Веб-приложение так или иначе не является «родным» приложением ОС. Между ним и операционной системой одна степень разделения (браузер). Аналогично Flash-приложение работает в виртуальной машине, «сидит» поверх браузера и не является «родным» по отношению к нему.

Изображение: Розмари Воегтли / smashed.by/voegtli

Однако Интернет как платформа имеет мало фиксированных пользовательских опытов. Хотя веб-приложения пользуются общими свойствами, «Руководства по интерфейсу» для Сети нет (но наверно, должно бы быть [\[108\]](#)). Вместо этого, мы концентрируем свое внимание на документировании и продвижении хорошего кода и дизайн-практик, таких как прогрессивное улучшение. Различные браузеры по-разному реализуют управление основными элементами форм. Вот почему поведение веб-приложения отличается в различных браузерах, даже если у него такие же компоненты, разметка и код.

Гибридные приложения

Итак, у нас есть «родные» приложения, которые относятся к культуре платформы, на которой они работают. И у нас есть веб-приложения, которые работают в браузере. Но мы забыли о третьей категории, под которую подпадают многие современные приложения: гибридные приложения.

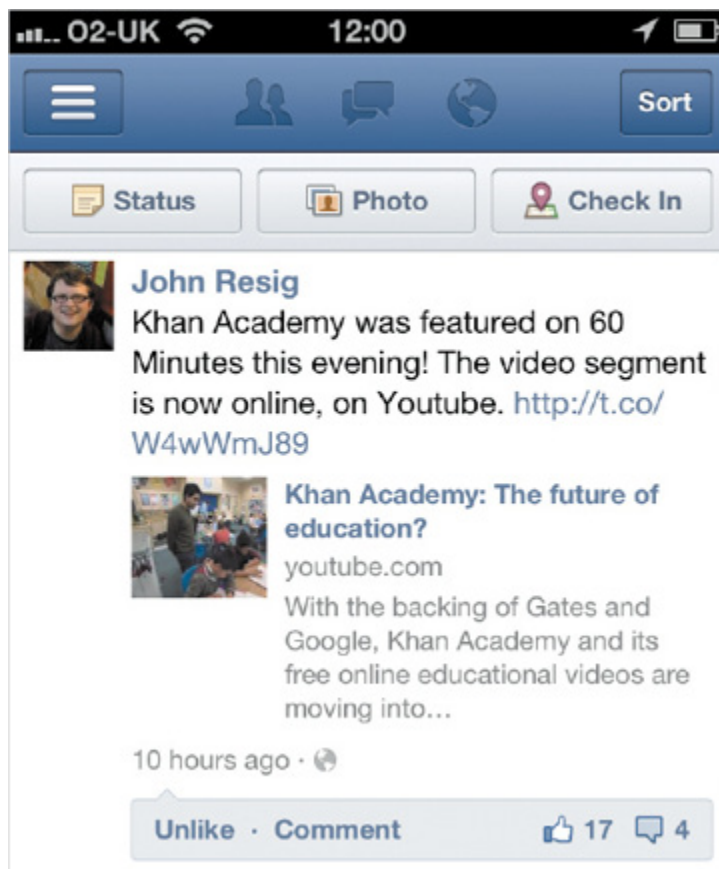


Рисунок 9.8. Гибридное приложение Facebook на iPhone

Мы должны понимать всю силу различных технологий и использовать их где возможно. Официально провозглашенные авторские технологии (главным образом HTML и CSS) прекрасно подходят для создания сложных документов, и придают им бесподобный стиль. Таким образом, многие дизайнеры «родных» приложений используют язык HTML и библиотеки CSS, когда им нужно отобразить разнообразный контент. Эти виды приложений носят название «гибридные приложения», потому что они являются смесью «родных» и сетевых технологий.

Приложение Facebook на iPhone является одним из примеров гибридного приложения, в котором определенные разделы (такие как новостные ленты) воспроизводятся с помощью веб-технологий.

Аналогично тому, что мы видели раньше, веб-приложения тоже могут быть гибридными. Сайт, написанный на HTML, CSS и JavaScript и использующий Flash для отображения богатого интерактивного контента, – пример гибридного приложения.

Многие приложения сегодня – гибриды. И если вы ас в HTML, CSS и JavaScript, могу сказать, вы не останетесь голодным независимо от того, какая платформа или платформы в конечном итоге станут популярными через годы.

Преодоление идеологических предрассудков

Гораздо чаще технологические и дизайнерские решения основываются не на желании выбрать лучшие средства и материал для работы, а на идеологии. К сожалению, почти повсеместно веб-стандарты поддерживают те, кто слепо предлагает платформы и «родные» авторские технологии для любого проекта, не учитывая нужды пользователей. Веб-специалисты – практики, которые без оглядки советуют Flash и «родные» iPhone– или iPad-приложения для любого проекта, тоже явление распространенное. На ум приходит старая поговорка «Против лома нет приема». Поэтому важно увидеть такие предрассудки и строить свои решения на нуждах ваших пользователей, а не на идеологии. Еще важно уметь распознавать идеологические взгляды так, чтобы направить дискуссию в область разработки.

Иные сторонники веб-стандартов убеждены, что платформа и авторские технологии по своей природе являются некой силой добра. Никто не спорит, что Интернет и Сеть как основы оказали на мир такое же (если не большее) демократическое воздействие, как печатный станок Гутенберга. Однако веб-платформа и авторские технологии по сути своей ни плохие и ни хорошие, и они могут легко использоваться в обоих случаях.

В случае веб-платформы принято считать, что она хорошая по своей природе, потому что к любому документу или приложению, расположенному на ней, повсюду есть доступ. Хотя это может быть

верно для открытых коллекций документов (это было нормой для контента на заре веба), но не подходит для того, что мы сегодня называем современными приложениями. Возьмем, к примеру, Facebook. Facebook – это сетевое приложение. Оно – закрытое. Оно бесплатное. Но это значит, что вы, пользователь, не клиент Facebook. Вы – его продукт^[109]. Ваши личные данные и поведенческие характеристики – это то, что Facebook продает своим реальным клиентам, рекламодателям. Так может это лучше и более открыто, чем покупать коммерческое приложение в магазине приложений от Apple? На самом деле нет.

По сути, вы легко можете привести аргументы, что купить лицензию на коммерческое приложение iPhone более честная и открытая сделка. Вы платите за нее и таким образом владеете лицензией, чтобы пользоваться. Вы становитесь клиентом компании или личности, которая сделала приложение. Это прозрачность, с которой компания делает деньги. Во многих отношениях это больше традиционные коммерческие отношения.

Естественно, даже коммерческие приложения могут окольными путями использовать ваши данные, поэтому сегодня важно быть начеку. Но суть в том, что просто веб-приложение не сделать силой добра каким-то магическим образом.

«Родные» приложения и платформы – исчезающие виды?

Как отлично подметил Джереми Кейс на конференции по модернизации, «писать „родное“ приложение все равно что писать код под лазерный диск». Смысл здесь в том, что Сеть все время развивается, и «родные» платформы, как и компакт-диски (CD-ROM) и лазерные диски (LaserDisc) скоро устареют.

Я сильно надеюсь, что это не так, потому что Сеть сама по себе «родная» платформа и все больше становится таковой (в общепринятом смысле) с ростом операционных систем, таких как WebOS и Chromium, которые созданы на основе «родных» веб-технологий. Мы должны понимать, что Интернет развивается – это движущаяся цель. Новые свойства, расширение опыта взаимодействия и т. д. постоянно добавляется к «родным» платформам. Это не похоже

на решение Apple с 1 сентября 2012 года прекратить создание новых операционных систем и моделей iPhone и iPad, потому что создана система iOS и может понадобится несколько лет, чтобы Сеть догнала ее.

Мы знаем, что термин «родной» относится к базовой культуре и языку платформы, поэтому утверждать, что «родные» приложения исчезнут, значит, признавать, что различные устройства не будут иметь особую культуру в будущем.

Исходя из этого, можно предположить, что в будущем разовьется монокультура, где каждое приложение на каждом устройстве будет «говорить» через элементы системы и авторизовываться с использованием «родных» сетевых авторских технологий. Более того, эти приложения будут обслуживаться с помощью веб-платформы. Вообще, это ужасное и безрадостное видение будущего. Такого, где люди все меньше смогут управлять своими данными, а их устройства попросту превратятся в «немые» терминалы, подвешенные к огромным облачным хранилищам, подконтрольным большим корпорациям.

Например, вместо того чтобы владеть лицензией на приложение по обработке текста, вы могли бы написать все в Google Docs. Google, со своей стороны, анализировал бы каждое слово и предложение и старался понять больше о том, кто вы и что вам интересно, и использовал эту информацию для манипуляции вашим поведением в коммерческих целях.

Не сказать, что веб-приложения – это неизбежное зло, но по своей природе они точно нехорошие.

Размывание границ

Пока разработчики свято верят в то, что будущее Сети – исключительно в разнообразных авторских веб-технологиях, таких как HTML, CSS и JavaScript, которые обеспечат лучший доступ к таким свойствам устройств, как тач, GPS, поддержка акселерометра и камеры, я буду спорить. Очень редко говорят о том, как «родные» приложения догоняют веб-приложения.

В некоторых отношениях «родные» приложения делают серьезные успехи и уже почти не уступают в преимуществах, которыми долгое

время могли похвастаться лишь веб-приложения.

Вот три основные области, в которых они догоняют веб-приложения, – это легкость в использовании и доступе, автоматическое обновление и прямой доступ к данным.

Рисунок 9.9. Граница между потоком использования «родных» веб-приложений или «родных» OS-приложений размытая. Фактически в операционных системах, основанных на веб-технологиях, «родные» веб-приложения – это «родные» OS-приложения

Легкость в использовании и легкий доступ

Одно из основных преимуществ опыта взаимодействия, которое Сеть имеет от «родных» приложений, это – легкость, с которой они могут использоваться и быть доступными. Нажмите клавишу Upload (Загрузить) в вашем FTP-клиенте по выбору^[110], и разработанное приложение станет доступным каждому, у кого есть ссылка на него, в любом уголке мира^[111]. Все просто.

Не нужно скачивать Zip-файл, потом искать приложение, чтобы разархивировать его, потом искать, куда оно скачалось, потом распаковывать архив, устанавливать, запускать. И все это только для того, чтобы узнать, что оно не поддерживается вашей видеокартой. Ик! Вас не удивляет, что такие веб-приложения, как Gmail и Google Docs обрели феноменальный успех, особенно на «родных» платформах с низким уровнем удобства в работе?

Так или иначе «родные» приложения стремятся к легкости размещения и доступа, которые предлагают веб-приложения благодаря разработке хранилища приложений. С хранилищем приложений типа Apple процесс поиска так же прост, как клик по ссылке в браузере. По сути, вы можете кликнуть по URL, чтобы найти приложение в хранилище, там просто нажать клавишу, чтобы загрузить и установить его.

Автоматические обновления

По своей природе веб-приложения всегда находятся в обновленном состоянии. На самом деле мы совсем не думаем о «версии» веб-приложения, потому что Сеть, по сути, независима от версий.

Вы всегда пользуетесь последней версией Gmail, и вам все равно, что это за версия.

Это не Gmail версии 7; это просто Gmail^[112].

Для сравнения, «родные» приложения имеют громоздкий механизм обновления, который прерывает процесс использования.

Это тоже меняется. Все больше и больше «родных» приложений выполняют непрерывное обновление. К примеру, когда вы в последний раз видели, как Google Chrome обновлялся? Да никогда. Он делает это тихо.

Непрерывный доступ к данным

Другое огромное преимущество, которым пользователи веб-приложений обычно наслаждаются, это непрерывный доступ к своим данным с любого устройства. Вам никогда не нужно волноваться о синхронизации e-mail с вашего компьютера на мобильный телефон, когда вы пользуетесь сервисом Gmail. Это всегда в нем. Сравните это с ночным кошмаром, который обычно «обрушивается» на синхронизацию на «родных» платформах.

Однако, как ни крути, «родные» приложения снова наверстывают упущенное. Например, с сервисом iCloud от Apple, синхронизация вручную становится вчерашним днем. Ваши данные легко и просто доступны автоматически на ваших устройствах Apple и постоянно находятся в процессе синхронизации, и у вас не должна болеть об этом голова. Хотя сервис iCloud по большому счету клиентское решение Apple, кроссплатформенные технологии типа Dropbox дают похожие преимущества другим платформам.

Просто еще один клиент

Вы внимательно прочли предыдущий раздел? Отлично. Тогда вы могли заметить как красной нитью во всех трех областях проходит то, что «родные» приложения должны «наверстывать» веб-приложения. Это все области, где плюсы опыта взаимодействия – это заслуга Интернета, а не Сети. Сеть – это просто стек технологий, а именно

HTTP и адресов URL на вершине Интернет-стека. Поэтому «родные» приложения «наверстывают» Сеть с помощью тех же характеристик Интернета, что и она сама.

Кроме того, мы наблюдаем рост новых моделей опыта взаимодействия, которые называются непрерывный клиент. Непрерывный клиентский опыт, изначально предлагаемый Джошуа Топольски^[113], позволяет пользователю непрерывное взаимодействие с устройствами и контекстами. Например, когда вы читаете Twitter на своем компьютере, а затем хватаете свой телефон, вам нужно, чтобы вы могли продолжить чтение ленты с того места, на котором остановились. А когда вы приходите домой, вам надо продолжить с того же места на телевизоре.

Рисунок 9.10. Келли Соммерс представляет хороший пример приложения, которое демонстрирует опыт непрерывного использования: smashed.by/multi. Пользователи могут начать просмотр видео на своих Windows Phone 7, продолжить в веб-клиенте, а затем перейти на свои iPhone

Рисунок 9.11. Приложение по передаче непрерывных сообщений сервиса Trillian – хороший пример непрерывного клиента.

Фото: Trillian Blog, smashed.by/trillian

Хорошим примером непрерывного клиента служит приложение мгновенного обмена сообщениями программы-мессенджера Trillian. Оно может сохранять сообщения в облаке, распределять разговоры между всеми вашими устройствами в реальном времени, отслеживать и синхронизировать ваши прочитанные и непрочитанные сообщения и даже воспользоваться «присутствующей технологией», чтобы узнать, на каком устройстве вы в данный момент активированы. И это для того, чтобы посылать текущие сообщения только туда.

Как вы понимаете, в эпоху использования приложений типа «непрерывный клиент» Сеть и сама становится просто еще одним клиентом. В определенных контекстах он может быть лучшим для использования, но у пользователей все-таки есть выбор –

переключиться на «родных» клиентов, не беспокоясь о синхронизации данных. Скоро пользователи в основном будут ожидать приложений типа «непрерывный клиент», а не инноваций, тем более что технологии высокого уровня, например iCloud, упрощают задачу разработчиков при их внедрении.

Будущее за родным

Сегодня Сеть – это просто еще одна «родная» платформа, которая набирает обороты и должна конкурировать по своим уникальным достоинствам, а не по тем, которым ее одаривает Интернет, ведь другие платформы тоже реализуют такие же возможности. Вопрос на засыпку! Что вы запланируете при выборе веб-платформы для вашего следующего приложения как его достоинство: интерфейс пользователя «родного» приложения или показ через URL и обслуживание его через HTTP?

Веб-приложения «догоняют» «родные» платформы по таким свойствам, как локальное хранилище и способность работать, когда устройство не в Сети, граница между веб и «родными» приложениями стирается все больше. Она достигает своего логического предела в операционных системах типа WebOS от Palm и Google Chrome, где «родные» технологии – это веб-технологии.

Нам надо понимать, что веб-приложения, работающие на таких ОС, являются «родными» приложениями. Поэтому мы выбираем между структурами различных ОС и «родными». Этот выбор определяет то, какая «родная» ОС предложит лучший опыт взаимодействия. Потом наш выбор будет проходить между различными «родными» авторскими технологиями – HTML, CSS и JavaScript для «родных» веб-приложений, Objective-C и Cocoa Touch для iOS приложений, Java и Android SDK для Android приложений, C# и .NET для Windows Phone-приложений, и т. д., и т. п.

Неважно, какие платформы или технологии в конце концов победят. Будущее явно за «родным», а веб – это просто еще один клиент. Сейчас вопрос на миллиард долларов звучит не как «Выберем мы сетевое или «родное»?» – а, скорее, так: «Какая платформа или платформы, какая клиентская технология или технологии должны поддерживать наш продукт?»

Чтобы ответить на него, нам нужно понимать сущность нашего продукта и особенно то, к чему он по сути ближе – к сайту или приложению.

Документ или приложение

В Сети один из способов классификации продуктов – это определить их ядро: содержание или поведение. Мы называем совокупность документов с концентрацией на содержании сайтом. Продукт, сконцентрированный на поведении, называется приложение. Вместо того чтобы полностью вставать на сторону одного или другого лагеря, ваш продукт, возможно, расположился где-то между двумя этими полюсами.

Рисунок 9.12. Континуум документ-приложение

Рисунок 9.13. Принцип универсальности, выдвинутый создателем Всемирной паутины, Тимом Бернерсом Ли, был написан в пору, когда Сеть представляла собой в основном совокупность связанных документов. Не обязательно применять его полностью к приложениям, которые по своему дизайну имеют различные ограничения и требования. См. комментарии Ли Веру на возражения Джона Олсопа по поводу моего манифеста одной версии в. Net magazine: smashed.by/netmag

Когда продукт «приближается» к документальной стороне, можно постепенно улучшать свойства ядра, основанного на содержании, сохраняя при этом доступ большого количества людей к ядру. Вносимые изменения обычно либо расслаивают продвинутое форматирование, либо размещаются на этих документах, либо добавляют невообразимые взаимодействия для навигации в них или между ними. Мы можем адаптировать контент под различные размеры экрана и адаптировать ограниченные взаимодействия, которые используются для навигации контента под различные механизмы ввода. Это непростая, но все же выполнимая задача.

Когда продукт перемещается от документа к приложениям, введение прогрессивного улучшения, так или иначе, усложняется. Фактически это может стать полностью бессмысленным и невозможным.

Это напоминает скорее изящную деградацию^[114] онлайн-функционала редактора изображений? Как бы редактор изображений работал на телефонах без полноцветного дисплея?

Ни одна команда разработчиков продукта на земле не имеет ресурсов для создания приложений, которые предоставят наилучшие возможности опыта взаимодействия для каждого пользователя

Какой контент вы бы отображали на таких устройствах?

Приложения не основаны на контенте. Они основаны на поведении. Изящная деградация к упрощенному представлению любого контента приложения не всегда имеет смысл. Приложения часто полностью составлены на основе поведения, которое позволяет пользователю создавать контент. Вернемся к редактору приложений: в нем нет никакого контента, но он позволяет пользователю создать его.

Чтобы создать образцовый опыт взаимодействия, нам нужно поддерживать фокус. Это фокус должен устанавливаться с учетом потребностей пользователей самым наилучшим способом, какой только возможен.

Предоставляя неограниченное время и ресурсы, мы могли бы оптимизировать опыт использования наших приложений на любом устройстве и платформе, известных человечеству.

Однако, используя ограниченное время и бюджет, с которыми нам приходится работать в реальном времени, мы обязательно должны быть разборчивы в выборе нашей аудитории, платформ и устройств. Мы делаем это не для того, чтобы исключить ненужных людей, а потому что мы понимаем, что включать всех и предоставлять всем огромный опыт взаимодействия – непрактично.

**Соображения по мобильному дизайну для
сконцентрированных на документах продуктах –
сайтах**

Документальную сторону непрерывной среды – континуума – представляют сайты. Сайт содержит контент, а также представление этого контента. Контент может представлять собой текст, изображения, аудио, видео и т. д., размеченные с помощью языка HTML, для того, чтобы добавить семантику и структуру. Презентация включает в себя как визуальную разметку, так и интерактивные элементы (такие как навигация между страницами и положениями), и обычно осуществляется с использованием комбинаций CSS и JavaScript.

Протестируйте свои дизайнерские разработки на действующих устройствах

Симулятор или эмулятор прекрасно подходят для проверки эффективности изменений кода в ходе разработки, но они не смогут восстановить уникальную эргономику самого устройства. Контекст также является ключевым фактором, который влияет на удобство мобильного использования. А, например, дизайн, который прекрасно действует в офисе при отличных условиях освещения, может не работать при солнечном свете.

Помните еще одно! Когда вы проводите тестирование с симулятором, он потребляет мощность «железа» вашего компьютера, чтобы запустить приложение. Вы можете наблюдать низкую производительность и даже слегка разное поведение, когда работаете с реальными устройствами.

Если у вас есть сайт, первое, что вы должны сделать, протестировать его в мобильной среде и посмотреть, как он отображается и действует. Я сталкивался со слишком многими компаниями, которые не считали важным делать свой сайт удобным для мобильного использования и сразу же «прыгали» в создание «родного» приложения. Будьте осторожны! Не повторяйте таких ошибок, особенно если ваш сайт является важным генератором прибыли. В сайтах, основанных на документах, используйте по возможности прогрессивное улучшение, для того чтобы обеспечить контент и основные взаимодействия при навигации, и чтобы убедиться, что контент остается доступным самой широкой публике.

Для этого вы можете выполнить следующие шаги:

1. Убедитесь, что ваш контент доступен для всех, сделайте правильную семантическую разметку. Главное – отделить контент от

представления.

2. Прогрессивно улучшайте контент, чтобы люди, которые просматривают его с экранов разных размеров, получили больше оптимизированного опыта (в настоящее время на этом сосредоточен отзывчивый/адаптивный дизайн).

3. Прогрессивно улучшайте контент, чтобы семейства устройств, основанные на поддерживаемых функциях и возможностях (например, тач), получили больше оптимизированного опыта. Проще говоря, сделайте сайт чувствительным в поведении, а не только в разметке. Здесь вы оптимизируете для особенностей, а не для специфических устройств (так сказать, для всех мобильных телефонов, которые поддерживают тач, а не только для одного iPhone).

4. Прогрессивно улучшайте контент для поддержки уникальной культуры и возможностей различных устройств, которые вы хотите поддерживать. Здесь оптимизация для специфических устройств вполне нормальна. Нет ничего плохого в том, что вы пытаетесь сделать опыт взаимодействия прекрасным, насколько это возможно, даже если он создается для специфических телефонов.

Конечно, каждый из этих шагов занимает и время, и много ресурсов. И в соответствии с этим, вам нужно планировать и составлять бюджет. А что если ваша задумка сделать действующий сайт адаптивным не удовлетворяет нужды пользователей в определенных случаях? Возможно, вам нужно предоставить больше оптимизированного и специального опыта, чем допускает прогрессивное улучшение, с учетом бюджета и сроков. Или может быть вам нужен такой уровень интеграции устройства, который просто не совместим с действующим браузером? В этом случае вам в голову могут прийти мысли о том, создавать ли приложение, и если да, то будет ли оно поддерживать множественные платформы, и какие технологии нужно будет использовать.

Кроссплатформенность или единственная платформа?

Как лучше удовлетворить потребности пользователя: запуская приложение на многих платформах или на единственной, по крайней

мере сначала? Когда будете отвечать на этот вопрос, не забудьте, что создание одной платформы не означает, что вы позже не сможете создать отдельное приложение для другой платформы.

Это значит лишь то, что в начале работы над проектом в центре ваших дизайнерских изысканий будет единственная платформа (или даже единственное устройство).

Само по себе не имеет значения, будет ли работать ваше «родное» приложение или оно будет использовать «родные» компоненты устройства или устройств, на которых запускается. (Использование кроссплатформенных авторских технологий для создания приложения, которое вы оптимизировали для единственной платформы, легко осуществимо.)

Ответ на этот вопрос в любом случае определит, какую оптимизацию вы должны сделать на различных платформах. Если вы заботитесь об опыте взаимодействия, то должны оптимизировать ваше приложение на каждой платформе, которую поддерживаете. Ваше решение также повлияет на то, сколько тестирований вы должны сделать (потому что вам нужно будет тестировать каждую платформу, которую поддерживаете), на размер отдела техподдержки (потому что вам нужно будет разбираться с проблемами пользователей под каждую платформу) и на то, сколько времени и бюджета вам понадобится, чтобы все утрясти для этих различных функций.

Миф: Один раз написанный код работает везде

Типичная ошибка, которую допускают дизайнеры, – это предположение, что при использовании кроссплатформенных авторских технологий они могут написать код один раз, а работать он будет везде. Это миф. И он может привести к достаточно дорогим последствиям.

Ваше приложение должно запускаться на множестве платформ, но это в редких случаях означает, что оно будет хорошо запускаться на множестве платформ

Единственный способ заставить приложение хорошо работать на различных платформах – это оптимизировать его для каждой платформы и устройства. Как мы говорили раньше, каждая платформа имеет свою культуру, язык и нормы, которым, как ожидает

пользователь, соответствуют приложения. И многих пользователей не интересует, на скольких устройствах работает ваше приложение. Их заботит только то, насколько хорошо ваше приложение будет работать на их устройстве.

Дизайнеры, которые не учитывают уникальную культуру, традиции, язык и нормы платформ, рискуют сделать свои приложения выглядящими неуместно. Приложение будет чужеродным, часто даже вызывающим, просто потому что оно не учитывает культуру платформы.

Так как мы не хотим, чтобы наши приложения демонстрировали такое отвратительное поведение, мы должны оптимизировать их под каждую платформу, которую поддерживаем. Наша цель – создать приложения, культурно восприимчивые к языку, нормам и традициям платформ, на которых они запускаются. Отступить от нее – значит, проявить неуважение к определенному сегменту пользователей.

Смерть от тысячи ранений

Худшее, что вы можете сделать, это, конечно, проявить неуважение к вашим пользователям, создав приложение с наименьшими сходными характеристиками, которое предоставляет им неоптимизированный опыт взаимодействия на каждой платформе. Здесь вы уязвимы больше всего. Даже если ваше кроссплатформенное приложение потенциально может иметь большое количество пользователей, потому что запускается на многих устройствах и платформах, на каждой из них оно будет неудобно в использовании. Проще говоря, кого волнует, на какой платформе работает приложение, если оно работает ужасно?

Еще важнее, что будет, когда на одной из этих платформ появится конкурент со своим великолепным, восхитительным, детально оптимизированным опытом взаимодействия и «вышибет» ваше приложение? А на другой платформе то же самое с вами сделает другой конкурент.

Это равносильно смерти от тысячи ранений. Поддержка множественных платформ не станет функционировать до тех пор, пока вы не сможете хорошо поддерживать все из них. У вас может быть преимущество первого игрока на рынке, но только до той поры, пока вас не превзойдет лучший конкурент.

Напиши один раз, оптимизируй везде

Что ж, написанное один раз работает везде – опасный миф. Кроссплатформенные приложения, которые удачно конкурируют, пишутся один раз, оптимизируются везде. Вы должны понимать, что это означает для вашего бюджета, сроков и плана оптимизации, тестирования и поддержки приложения на каждой платформе, которую вы выбрали.

Пожалуйста, не берите демонстрационные версии различных производителей, обещающие создание кроссплатформенного приложения за пять минут. Это все будет ерундой, пока вы не составите простейший план действий с приложением. Правильное тестирование технологии заключается не в том, чтобы легко создать пятиминутную демоверсию или быстро пройти 7 % пути к своей цели, а в том, насколько легко вам будет «удержать» последние 3 % вашей разработки, включая все важные оптимизации для опыта взаимодействия, что может забрать последний 1 %. На эти детали в разработке приложения у вас может уйти масса времени и усилий.

Рисунок 9.14. Бинарно-каркасная матрица

Сетевые и другие кроссплатформенные технологии

Вообще говоря, кроссплатформенные технологии можно разбить на две группы: те, которые создают «родной» бинарный файл и которые не делают этого. Мы можем дальше классифицировать их на те, которые используют «родные» фреймворки. Эта комбинация дает нам бинарно-каркасную матрицу, представленную ниже:

«Родной» бинарный файл – это пакет приложений, которые могут запускаться напрямую операционной системой данного устройства. Мы, как правило, говорим об этом, когда ссылаемся на «родное» приложение.

Однако гораздо важнее проверить, использует ли приложение «родные» фреймворки. Эти фреймворки воплощают в себе культуру, язык, жесты, символы и нормы платформы.

Чужие приложения в обертке от «родных» бинарных файлов, или Волки в овечьих шкурах

С помощью технологии типа Adobe PhoneGap вы можете (и вполне легко) создать для iPhone «родной» бинарный файл, не использующий «родные» компоненты в фреймворке Cocoa Touch для iOS. PhoneGap «оборачивает» существующее веб-приложение, и создает из него «родное» приложение операционной системы (в примере выше «родное» приложение системы iOS). Ваше приложение становится похожим на «родное» для iPhone и запускается как «родное» приложение, но пользовательский интерфейс приложения при этом воспроизводится с использованием веб-компонентов^[115].

Ваше бинарное приложение это всего лишь оболочка, которая состоит из WebKit компонента. Этот WebKit-компонент визуализирует ваше веб-приложение с помощью веб-компонентов. Из-за того что веб-компоненты не могут соответствовать «родным» ожиданиям, я бы не советовал использовать PhoneGap при создании приложений для повышения продуктивности^[116], таких как календарь, различные списки дел и т. п.

Однако при разработке таких приложений, как электронные книги и игры, т. е. приложений с эффектом присутствия, недостаточная поддержка «родной» структуры в похожих технологиях не является большой проблемой. В среде создателей игр и электронных книг Adobe Flash и ActionScript, Unity и Anscas Corona считаются фаворитами, даже если они не используют «родные» структурные элементы и компоненты. Это из-за того, что подобные приложения редко используют «родные» компоненты. Вместо этого дизайнеры стремятся создать полностью иную среду, возможно, скевоморфную, которая выглядит и ведет себя, как настоящая книга, – со своими правилами, взаимодействиями и культурой.

Я бы не советовал использовать Adobe Flex (или приложения, которые пользуются компонентами Flash) по той же причине, что и не советовал использовать PhoneGap для создания «родных» приложений без эффекта погружения. Они не соответствуют культуре «родной» платформы и будут вести себя иначе, чем «родные» компоненты системы.

В общем, будьте осторожны, когда создаете «родные» бинарные файлы, которые легко «обволакивают» «неродные» приложения. Выглядят они как «родные», но ведут себя иначе, потому что не используют собственные компоненты в «родных» фреймворках. PhoneGap приложения, которые используют структуру jQuery, могут отображать нечто, похожее на таблицу интерфейса iOS при запуске на iPhone. Но это всего лишь HTML, продукт умного использования таблицы CSS и языка JavaScript. Он выдает себя за таблицу вида iOS, но не может соответствовать поведенческим характеристикам настоящего компонента представления таблицы из каркаса Cocoa Touch и поэтому перестает соответствовать ожиданиям.

Неудовлетворенные ожидания – это главный враг для удобства взаимодействия. Бегите от этого, как от чумы.

Приложения с эффектом присутствия против приложений без эффекта присутствия

Важно понимать разницу между приложениями, создающими эффект присутствия и не создающими. Значительно меньше проблем вы получите для приложений с таким эффектом, сделав их «неродными».

Эти приложения обычно занимают все устройство и по определению создают собственную культуру и язык. Игры – хороший тому пример. Игра может иметь собственную систему управления бегом, прыжками и стрельбой. В отрыве от культуры платформы, хорошая игра переносит пользователя в созданный свой собственный мир. Таким образом, в рамках платформы такие приложения используют очень немногие. По сути, это первые кандидаты на использование «неродных» технологий.

Вот почему технология Flash, не являясь «родной» для веба, так популярна при создании игр в Сети. Или почему Unity, не пользуясь «родными» компонентами в Cocoa Touch, применяется для создания многих основных игр на iOS и других платформах.

Приложения без эффекта присутствия, например повышающие продуктивность, как правило, усложняют использование стандартных элементов интерфейса пользователя, таких как кнопки и таблицы. Они используют определенные платформенные взаимодействия, например

переход в полноэкранный режим в iOS либо древовидное управление в приложениях Windows или OS X. Таким образом, очень важно, чтобы приложения без эффекта присутствия говорили на «родном» языке и соответствовали «родной» культуре и нормам той платформы, на которой они запускаются.

Суммируя вышесказанное, даже для приложений с эффектом присутствия, главным критерием для выбора «родной» или «неродной» технологии все еще может быть производительность.

У некоторых игр, таких как стрелялки от первого лица, нужно выжимать каждый бит для эффективной работы в системе. В подобных случаях некоторые «неродные» технологии могут не совсем подходить для ваших нужд. Например, ранние версии приложений на основе Adobe AIR Flash для iPhone были печально известны как медленные. Впрочем, в последних версиях Adobe AIR на iOS Adobe улучшил эффективность работы.

«Родные» приложения, переведенные с «неродных» языков

Если вы стремитесь оптимизировать опыт взаимодействия, лучшим кроссплатформенным методом при создании «родных» приложений без эффекта присутствия будет использование собственных объектных структур и компонентов. Это не значит, что вы обязательно должны применять «родной» язык программирования данной платформы, чтобы авторизовать ваше приложение.

Продукт	Авторская технология	Внедрение		
		«Родной» бинарный файл?	«Родные» компоненты?	Кроссплатформенность?
Web app	Web (HTML, CSS, JavaScript)	Да ¹	Да ² и Нет ³	Да
Appcelerator Titanium	JavaScript	Да	Да	Да
PhoneGap	Web (HTML, CSS, JavaScript)	Да	Нет	Да
Xcode (и/или GCC, LLVM)	Objective-C, Cocoa Touch (и C, C++, Ruby, Python)	Да	Да	Нет
Unity	C# и другие языки	Да	Нет	Нет
Mono (для Android и Mono Touch)	C#, базовые фреймворки устройств	Да	Да	Да

Рисунок 9.15. Сравнительная таблица некоторых общеизвестных кроссплатформенных технологий^[117]

Например, используя Titanium Mobile SDK от Appcelerator, вы можете написать на JavaScript, чтобы создать «родные» компоненты^[118]. Таким образом, на iOS, когда вы создаете представление в виде таблицы в Titanium Mobile, «родной» компонент представления таблицы Cocoa Touch создается в вашем интерфейсе пользователя. Оно не просто выглядит как «родное» представление таблицы (как могло быть в случае приложения на PhoneGap, которое имитирует «родные» компоненты), но и фактически является

«родным» представлением таблицы. И, что самое важное, оно ведет себя так, как и должно это делать представление таблицы.

«Родное» не обязательно лучше, но все же это «родное»

Единственный способ создания приложений, которые соответствуют нормам, т. е. культуре и языку данной платформы, – это использовать «родные» технологии. Например, пока возможно создание Flash-приложений, которые используются на веб-платформе, они не будут выглядеть или восприниматься как «родные» веб-приложения, которые используют «родные» авторские веб-технологии, такие как HTML, CSS и JavaScript. Также, пока возможно использование этих технологий для создания приложений на таких платформах, как iPhone, они не будут выглядеть или восприниматься как «родные» iPhone-приложения, которые создаются при помощи компонентов структуры Cocoa Touch. Не сказать, что Flash-приложения не могут быть представлены лучше, чем HTML приложения. В определенных случаях использования, особенно в играх, Flash-приложения могут обеспечить лучший опыт взаимодействия. Например, Machinarium, прекрасная игра, созданная с помощью Flash, прекрасно работает на iPad. Опять же для таких приложений, как игры и электронные книги, кроссплатформенное применение технологий типа Unity или Corona поможет сократить время на разработку и облегчить внедрение свойств, что было бы сложнее сделать с родными технологиями (такими как 3D-среда или физический движок).



Рисунок 9.16. Machinarium на iPad, интерактивное «родное» приложение, созданное на технологии Flash

Использование скриптового языка типа JavaScript может облегчить написание приложений и в очередной раз применить навыки по веб-разработке, которыми владеет ваша команда. При этом вы получите все преимущества использования «родных» компонентов в ваших «родных» приложениях.

Кроме того, поскольку Titanium Mobile является кроссплатформенной технологией, она обрабатывает «родные» iOS-компоненты для вашего «родного» iOS-приложения и использует «родные» Android-компоненты, когда компилирует «родное» приложение под Android. Преимущества налицо: вы используете простой скриптовый язык (JavaScript), и вам нужно всего лишь поддерживать один базовый код, вместо того, чтобы изучать и применять Objective-C в iOS и Java в Android, и поддерживать два отдельных языка.

Минус в том, что у вас есть еще один слой абстракции для работы. В конце концов, качество ваших приложений зависит от качества кода, который напишет Appcelerator в своих абстрактных фреймворках. А вы будете зависеть от того, как быстро Appcelerator поддержит новые свойства, которые выпускаются для «родных» фреймворков и SDK.

Хотя Appcelerator пытается сделать разницу между платформами прозрачной, насколько это возможно в Titanium, есть различия, не в последнюю очередь культурные, к которым вам еще нужно будет обращаться и оптимизировать (помните наше кредо: напиши один раз, оптимизируй везде).

Это не к тому, что вы должны бояться кроссплатформенных технологий, а, скорее, к тому, что вам нужно проводить исследования, взвешивать все за и против и принимать обоснованное решение по поводу того, добавлять ли еще один слой абстракции в ваш процесс разработки. Каждая кроссплатформенная технология имеет свои плюсы и минусы, а также ситуации, при которых она лучше приспособлена для определенных видов приложений. В то время как Corona – это прекрасный выбор для 2D-игр, Titanium Mobile может быть лучше для построения кроссплатформенного рабочего приложения.

Конечно, Titanium не единственная кроссплатформенная технология, которая может создавать «родные» бинарные файлы и использовать фреймворки. Если у ваших разработчиков есть навыки в разработке C# и .NET, вы также могли бы рассмотреть технологию Mono (а особенно MonoTouch для iOS и Mono для Android). Mono работает по такому же принципу, как и Titanium, но вместо использования JavaScript, вы берете язык программирования C# (...и с осторожностью другие языки программирования. .NET), .NET паттерны и инструменты для создания «родных» приложений.

Веб-приложения

Если вы читаете эту книгу, скорее всего, вы либо веб-дизайнер, либо веб-разработчик (либо и то и другое), либо вы только учитесь. Ваша роль как веб-дизайнера заключается в разработке набора документов (в этом случае вы могли бы здорово положиться на ваши навыки в графическом дизайне) для приложений с разнообразным поведением (здесь вы бы поупражнялись в применении ваших интерактивных дизайнерских способностей).

У вас также есть богатый выбор материала для разработки вашего продукта. Исходя из нужд вашей аудитории, вы можете выбрать использование «родных» авторских веб-технологий, таких как HTML,

CSS и JavaScript, для создания «родных» веб-приложений. В качестве альтернативы вы можете использовать «неродные» кроссплатформенные авторские технологии, такие как Adobe Flash или Unity, для создания «неродных» веб-приложений. И как третий вариант, вы можете использовать комбинацию обеих авторских технологий: «родной» и «неродной» для создания гибридных веб-приложений. Например, сайт, изначально построенный на HTML, CSS и JavaScript, но дополненный технологией Flash, чтобы обеспечить богатый игровой опыт.

Независимо от того, что вы выберете— «родные» или «неродные» веб-технологии, обратите внимание на то, как ваше приложение запускается, откуда (как) оно доступно и есть ли доступ к нему из Сети. Проще говоря, это означает, что у вашего сайта или приложения должен быть URL-адрес^[119] и оно должно обслуживаться через протокол HTTP. Это то, что делает веб-приложение веб-приложением.

«Родные» приложения одиночной платформы

В процессе работы, описанном выше, вы, возможно, придете к выводу, что лучше удовлетворите нужды пользователя, если оборудуете одиночную платформу или устройство под ваше приложение. А свое ограниченное время и бюджет вы вложите в оптимизацию опыта взаимодействия именно на ней.

Если все же вы решаете поддерживать одиночную платформу, вам все равно нужно выбрать технологию. Если вы делаете игровое или другое интерактивное приложение, то можете взять кроссплатформенную технологию, такую как Corona от Anscas, Unity или Adobe Flash, которые существенно облегчают разработку подобных приложений.

Если вы создаете не интерактивное приложение, у вас есть выбор между такими кроссплатформенными наборами средств, как Titanium Mobile или Mono. Недостаток здесь в том, что процесс разработки потребует еще одного дополнительного преобразования, и вы мало сможете контролировать оптимизацию вашего приложения, потому что в какой-то степени будете полагаться на «родной» код, написанный специалистами Titanium. Даже если вы погружаетесь в Titanium (или похожий фреймворк типа Mono), помните, что вам все равно нужно

изучить «родные» фреймворки (или API) платформы, на которой вы собираетесь работать. Изучать новые фреймворки гораздо сложнее, чем новые языки программирования.

Хотя опытный программист может освоить такой язык программирования, как Objective-C, в считанные дни, у разработчиков уйдут недели (а то и месяцы, даже года) на то, чтобы полностью вникнуть и привыкнуть к паттернам, культуре и тонкостям такого фреймворка, как Cocoa Touch.

В конце концов, вы можете использовать «родной» набор средств платформы, которую выберете. Например, вы можете использовать XCode с Objective-C и Cocoa Touch для разработки «родного» iOS-приложения, или Eclipse с Android SDK для создания «родного» Android-приложения, или Visual Studio с C# и фреймворком .NET для разработки Windows Phone-приложения.

Это может включать в себя изучение нового языка программирования (Objective-C для iOS, Java для Android и C# для Windows Phone) или привлечение группы специалистов, которые знают эти языки и имеют опыт работы с «родными» фреймворками. Как мы уже говорили, выбрать новый язык легко, гораздо сложнее изучить новый фреймворк. Не забывайте об этом, когда будете решать, идти ли вам таким путем, если ни вы, ни ваши специалисты не обладают необходимыми навыками.

Плюсов в создании родного приложения с использованием «родных» технологий множество. Во-первых, вы имеете полную гибкость в оптимизации приложения и опыта взаимодействия. Когда вы используете «родные» компоненты и придерживаетесь руководства по интерфейсу для выбранной платформы, ваше приложение будет соответствовать культуре, языку и нормам данной платформы. Пользователям будет легче его изучить и использовать. Также, если вы полагаетесь на чужой абстрактный или преобразующий слой, вы будете постоянно работать с последними версиями кода и фреймворками от производителя, которые дают вам дополнительную гибкость для подгонки новых свойств и обновлений, когда они выпускаются для платформы.

Самое важное то, что если вы направите свои усилия на одну конкретную платформу, время, которое вы потратили бы на оптимизацию, тестирование и поддержку другой платформы, можно

сначала уделить оптимизации опыта взаимодействия вашего приложения, а потом, если нужно, добавить поддержку для других платформ.

Дизайн для людей

Платформы и технологии, которые вы выбираете для создания вашего нового продукта, существенно повлияют на то, какой к нему будет доступ, как он будет выглядеть и восприниматься, будет ли он соответствовать, и, будем надеяться, превышать запросы ваших пользователей. Такое решение нельзя принимать на скорую руку.

Если выбор платформы и технологии основан просто на ваших сиюминутных деловых нуждах или на имеющейся у ваших специалистов компетенции, то в этом случае вы принимаете решение в своих интересах, но никак не в интересах пользователя. Это может сработать на какой-то короткий промежуток времени, но вы не сможете долго тягаться с теми, кто изначально решал проблемы пользователей. Выбор платформ и технологий должен базироваться на том, как лучше удовлетворить нужды пользователей, а не на идеологических пристрастиях или стремлении к краткосрочным целям в ущерб долгосрочным.

Помните, что и так существует слишком много нестандартных вещей. Нам не надо больше. Нам нужны вещи, которые будут хорошо работать. Создавайте то, что информирует, оказывает поддержку, развлекает и вызывает восторг. И при этом используйте в работе правильный инструментарий и технологии.

Об авторе



* * *

Арэл Балкан – дизайнер, разработчик, автор, учитель, предприниматель. Страстный поклонник ренессанса, помешан на разработке опыта использования и желании сделать мир лучше с помощью технологии и ораторского искусства. В настоящее время является организатором Конференции по модернизации как части Фестиваля цифровой техники в Брайтоне, а также создает iPhone-приложения, например тепло принятый критиками Feather.

О рецензентах



* * *

Джош Кларк окончил Гарвардский колледж (Кембридж, штат Массачусетс, США). Джош постоянно выступает на интернациональных технологических конференциях, делаясь своим видением мобильной стратегии и дизайна для телефонов, планшетов и других перспективных устройствах. Его девиз одинаков как для фитнеса, так и для опыта использования программ: «Без труда не вытащишь рыбку из пруда». Джош является основателем агентства Global Moxie.



* * *

Андерс М. Андерсен имеет степень магистра компьютерной науки, университет La Trobe University, Мельбурн. Его кредо – следующие строки Дугласа Адамса: «Я люблю дедлайны. Мне нравится свист, с которым они пролетают». Благодаря своей работе он делает информацию доступной людям. Самый большой урок, который он извлек из своей карьеры: если хочешь, чтобы что-то было сделано, сделай это сам. Его личное пожелание вам: «Старайтесь узнавать что-то новое каждый день».

Перестраиваем рабочий процесс: дружелюбный подход к будущему проекту

Автор: Стивен Хей

Рецензент: Брайан Ригер

Мне часто вспоминаются карты на стенах, которые показывают территорию офисного здания, торгового центра, метро или городскую зону, с выделенной обычно красной стрелой и пунктиром надписью «Вы здесь», указывающей на то место, где я стою. Эти карты показывают, где мы, что вокруг нас и куда нам надо идти. Конечно, нам нужно и то и другое. Мы должны четко знать направление, но это не имеет смысла, если мы не имеем понятия, где мы находимся сейчас.

Рисунок 10.1. Достичь места назначения важно, но знать отправную точку гораздо важнее. Фото: Джо Голдберг, smashed.by/joegraph

Карта все облегчает. Надпись «Вы здесь» показывает наше настоящее местоположение как данное; найти наш пункт назначения можно через индекс, а координаты посмотреть в таблице. Допустим, вы знаете, куда хотите пойти сначала. Тогда все, что вам остается, – это идти.

Философ Альфред Корзыбски заметил, что карта – это не территория: модель реальности – еще не сама реальность. Вот почему многие проектные планы не срабатывают. Не могу удержаться, но чувствую, в глубине души мы все это знаем. Примеры этого окружают нас везде: пробки, строительство дорог, улицы с односторонним движением, те вещи, которые мы не видели и не смогли бы увидеть на нашей карте, но мимо которых мы проходим.

Движение – это форма технологического рабочего процесса. Мы шныряем туда-сюда в транспорт, сокращаем путь, обходим стороной преграды, останавливаемся попить кофе, случайно сталкиваемся со

старым другом и узнаем от него, что магазин, который мы ищем, находится уже в другом месте. Снова хватаем карту и корректируем план.

Технологический рабочий процесс должен быть плавным. Он должен приспосабливаться к различным обстоятельствам. Он должен быть... адаптивным, если хотите. Для нас, искателей шаблонов и любителей пошаговых руководств, нет ничего дороже, чем набор команд для нашего технологического рабочего процесса:

- пройдите прямо два квартала,
- поверните налево,
- пройдите еще два квартала,
- первое здание слева.

Это выглядело бы фантастически в том мире, где карта – это территория. Но так не бывает. Никогда.

Во-первых, мы часто не знаем, с чего нам надо начать проект. «Вы – здесь» не существует. Клиенты часто говорят нам: «Вы – здесь», но наша задача, как дизайнеров, спросить и выяснить, действительно ли это так.

Даже фраза «Я думаю, что мы здесь» – рискованное дело. Если вы хотите попасть в Нью-Йорк, а вы, скорее всего, в Париже, а не в Чикаго, у вас серьезные проблемы: машина, которую вы взяли в аренду, не лучший способ передвижения. Вам все-таки нужно добраться до Нью-Йорка, но это будет гораздо дольше.

Итак, технологический процесс должен оставаться подвижным, потому что любой фактор влияет на другие факторы. Комбинация исходной и конечной точек (пункт назначения или цель) определяет наш выбор для путешествия из пункта А в пункт В. Летите до Нью-Йорка самолетом, до отеля возьмите такси, до здания прогуляйтесь.

В веб-дизайне местность изменяется быстро. Польза от карт минимальна. В мультиплатформенном дизайне, где сайты и приложения будут использоваться на многих различных устройствах, мы сталкиваемся лицом к лицу с разнообразными пунктами назначения. Перечень устройств, которые нужно поддерживать, может быть нашей картой, но ее полезность в этом изменяющемся ландшафте ограничена. Мы не можем сказать: «Это должно хорошо выглядеть на Android», что это значит вообще? Возможно, вы столкнетесь со списком переменных, с которыми вам придется иметь дело: разрешение экрана, плотность пикселей, размер экрана, поддержка CSS (или вдобавок поддержка любой технологии), доступность, ввод через клавиатуру и мышкой против сенсорного ввода. Список можно продолжить. И это только технические факторы. Давайте не будем забывать о «туманных» (хотя все же технических) изменениях, например, почему у клиента сайт на устройстве BlackBerry выглядит иначе, чем в вашем макете Photoshop. Упс!

Итак, как же установить технологический процесс для создания или переделки нашего сайта так, чтобы оставаться гибкими и легко адаптироваться и при этом не блуждать туда-сюда? Первый и хороший шаг — это начать сначала. Давайте посмотрим, что мы имеем и откуда мы хотим начать наше путешествие. Для того чтобы обеспечить себе оптимальную гибкость, сосредоточимся сначала только на контенте.

Сначала структурированный контент: мышление, независимое от устройства или платформы

Намеренное игнорирование нашего пункта назначения и фокусирование на том, где мы есть и на самом ли деле хотим начать отсюда, требует от нас немалой гибкости. На сегодня одна из больших проблем Сети в том, что технологии влияют на принятие решений больше, чем следует. Возможности устройства становятся поводом для используемых функций. Система управления контентом выбирается еще до того, как мы решаем, что будет делать приложение и для кого оно. «Хотим использовать HTML5», – заявляем мы.

Естественно, что люди сосредотачиваются на технологии. Ведь технология – это захватывающая и такая классная забава. Веб-технологии – это игрушки, инструменты, всякие штуки, который используют «все». Вместо того чтобы дать нашей проблеме подвести нас к выбору правильной технологии для решения, мы смотрим на классные демоверсии и придумываем повод использовать их. Мы забываем о том, что эти демоверсии делают люди, которые...ну, делают демоверсии. Когда вы занимаетесь проектом редизайна, это реальная жизнь. Вам нужно сделать вещь, которая действует. «Круто» – это, конечно, хорошо, но только в том случае, если это работает и решает проблему. Сегодня все внимание слишком часто приковывается к решениям, а не проблемам. Нам надо на время превратиться в пессимистов и сосредоточиться на всех наших проблемах – сосредоточиться на них пристально, позволить им вырасти. Тем, кто имеет дело с веб-технологиями (т. е. вам), ответы придут на ум прямо в душе.

Перечень контента

Есть определенное мнение о контенте, которое поможет разработчикам (и их клиентам) сосредоточиться на реальной сущности проекта. На протяжении всего времени действительно имеют значение только две вещи – это основной контент и функциональные возможности сайта или приложения. В этом причина того, почему люди, в первую очередь, используют продукт. Не обращайтесь внимания на платформу. Игнорируйте браузеры. Сосредоточьтесь на причинах

того, почему так. Назовем это мышлением, независимым от устройства или платформы, так как нас не волнуют эти вещи, по крайней мере, сейчас. Давайте применим это мышление к основному контенту и функциональным возможностям.

Как всегда, вопросы помогут нам мыслить критично:

1. Во-первых, почему люди будут использовать этот сайт? Найдите веские причины... Я не уверен. Поищите еще. А теперь «отшлифуйте» эти причины. Они должны быть истинными.

2. Если бы в мире существовал только один браузер, и он воспроизводил бы только язык HTML, со стилями по умолчанию и функциями, которые обрабатывает сервер, какую совокупность контента и функциональных возможностей вы могли бы предложить? Проще говоря, что смогли бы увидеть и сделать пользователи в таком случае?

Напишите ответы на эти два вопроса в список. Назовем его перечнем контента. Каждый элемент контента или функциональная возможность, которую вы будете добавлять в перечень, должен подходить под критерий цели: он должен активно поддерживать причины, которые вы внесли в список как ответ на первый вопрос.

Когда все сделано (это займет какое-то время и будет достаточно для промежуточного этапа разработки), посмотрите внимательно на результат. Он гласит: «Вы – здесь!»

В этом разделе мы расскажем о том, что можно изменить в технологическом процессе и в обычном веб-дизайне, чтобы в будущем сайт стал более удобным для пользователя. Мы поймем, как применять наш новый технологический процесс в простом проекте. Проект будет маленьким и умышленно не полным. Он просто покажет, как можно осуществить эти идеи. Попробуйте применить несколько или все техники в ваших личных проектах. Берите то, что работает для вас, и изменяйте или убирайте то, что не нужно. Я использовал все элементы в этом технологическом процессе в реальных проектах и остался доволен результатами. Надеюсь, что вы почерпнете из этих идей столько же полезного, сколько и я.

Пример: «Три маленьких прямоугольника»

Предположим, что мы работаем над сайтом, который обучает разработчиков CSS гибкой модульной верстке (или Flexbox), относительно новому модулю для разметки, которая имеет большие возможности в мультиплатформенном пространстве^[120].

Смысл в том, чтобы объяснения и примеры были до смешного простыми. В силу того, что большая часть свойств CSS может объясняться текстом и несколькими прямоугольными блоками, назовем наш сайт «Три маленьких прямоугольника». Со временем мы можем расширить наши уроки и включить другие CSS-модули.

«Три маленьких прямоугольника» будут состоять из теории и синтаксиса, а также нескольких заданий, каждое из которых строится на основе предыдущего задания. Пользователь сможет написать их код в редакторе на странице, и кнопка «результат» покажет результат кода. Щелчком по клавише «Запустить» запускается введенный код. Каждое задание имеет два вида текстового контента: теорию и задание. Теория объясняет синтаксис и концепцию, а задания являются упражнениями, которые даются студентам для закрепления.

Давайте накидаем перечень контента:

1. Логотип.
2. Глобальная навигация.
3. Вводный текст (только на домашней странице).
4. Редактор кода (только на обучающей странице).
5. Область результата (только на обучающей странице).
6. Индикатор выполнения задания (только на обучающей странице).
7. Теоретический текст (только на обучающей странице).
8. Текст задания (только на обучающей странице).
9. Навигация урока (только на обучающей странице).
10. Регистрационная форма (только на домашней странице).

У нас есть два вида страницы:

- домашняя страница (для вводного текста и регистрационной формы) и
- обучающая страница.

Здесь остановимся. Понятно, что это до смешного просто и не завершено, но для примера хватит. Конечно, мы не включили сюда много другого материала (страницу профиля, статистику по урокам и т. д.) На большом сайте перечень контента может оказаться громадным и, возможно, разобьется на несколько частей. Но помните! Это не функциональные требования. Мы не должны включать каждую деталь каждой части контента. Мы разделили его на большие специфические группы, которые отражают суть.

Как правило, мы присваиваем, номерной или буквенный идентификатор каждой части контента. Так как наш пример несложный, мы будем просто использовать нумерацию от 1 до 10, как в списке. Дизайн перечня не важен; годится простой старый текстовый файл. Но вы можете сделать его таким, каким позволит ваше воображение.

В перечне контента классно то, что он никогда не подведет. Все срабатывает. Клиент может выкрикивать все, что хочет, а вы просто записываете это. Мы скоро выясним, что будет работать, а что нет.

Записи перечня можно украсить (например, описаниями или скриншотами готовых компонентов), но знайте меру. Я считаю, что расстановка соответствия контента и типов страниц очень полезна, так что:

- домашняя страница → 1, 2, 3, 10,
- обучающая страница → 1, 2, 4, 5, 6, 7, 8, 9.

Порядок позиций в данном случае не имеет значения. Мы займемся этим во время этапа прототипирования.

Прототипирование со ссылкой на контент

Один из первых шагов, которые делают многие из нас в веб-проектах (будем надеяться, что после тщательной инвентаризации контента) – это создание прототипов. Прототипы, которые сегодня разрабатывают в основном дизайнеры, эволюционировали от простых чертежей прямоугольников на холсте до почти полноценного веб-дизайна, разве что без цвета, изображений и подробной типографики. Изначально это были средства работы с дизайном: расположение блоков, эффект близости, визуальная иерархия, приоритет пространства и т. п. Контент был представлен «грубыми» блоками.

Некоторые дизайнеры еще продолжают так делать. Но многие поменяли свои цели.

Сегодня прототипы часто представляются для показа клиенту, чтобы он посмотрел, как будет выглядеть сайт, и как он будет функционировать до фактической разработки. На самом деле эти прототипы и есть дизайн. Но они не доделаны, и поэтому их потенциально опасно показывать клиенту. Если клиенту нравятся его определенные аспекты, любое отклонение от них (шрифт ли это, цвет и т. д.) может привести к пропаже интереса со стороны клиента. Если дело закончится тем, что клиенту не понравится сам дизайн, что ж, тогда вам придется изрядно потрудиться.

Прототипы прошлых лет (впрочем, многие еще используют их) больше были похожи на наброски традиционных графических дизайнеров. Многие наброски могут делаться быстро, с адекватным объяснением того, что будет работать, а что нет. Дизайнерам нужно обратить внимание только на визуальную рельефность и композицию. Эти прототипы – их средство проектирования. Они не для клиента. Конечно, это не означает, что нельзя привлекать его на начальном этапе разработки, но делать это надо вскользь, для ознакомления, но никак не для утверждения нашей работы. Одобрение – это наименьшая проблема, которая может возникнуть, когда клиент решит сопровождать вас в процессе разработки.

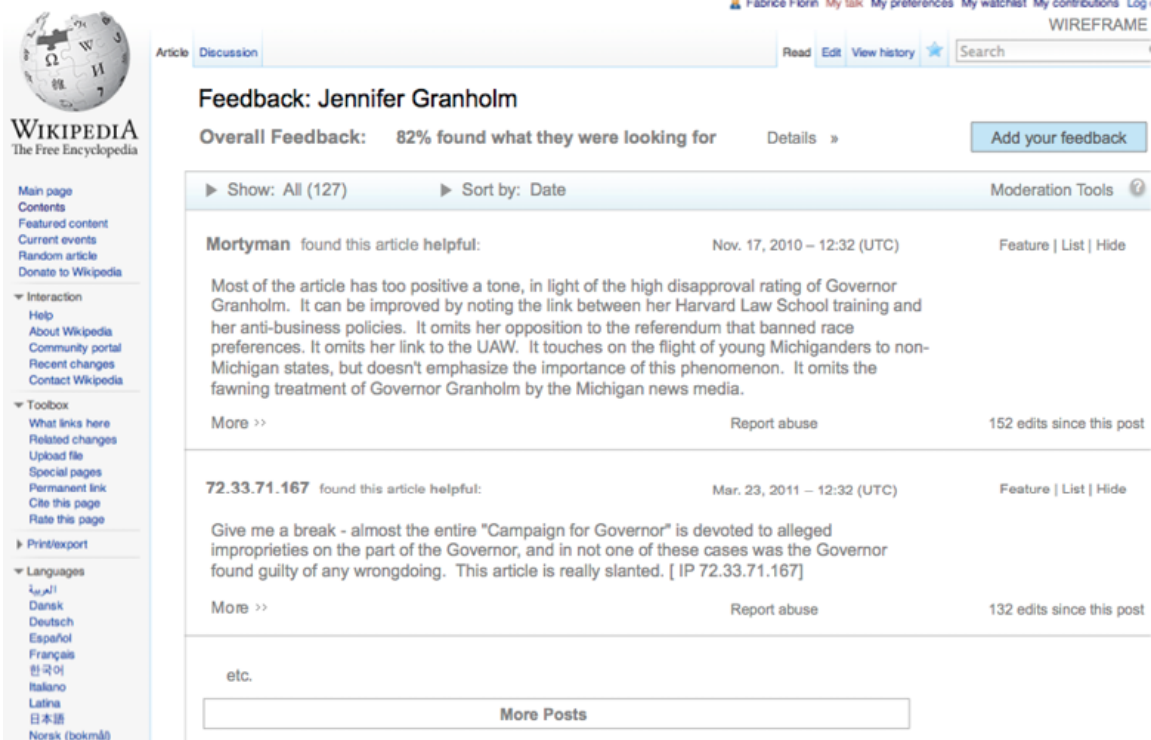


Рисунок 10.2. Такой стиль прототипа обычен. Хотя, скорее всего, он и полезен, он здорово похож на реальный сайт, и только надпись «Прототип» выдает, что это не так. И визуальный дизайн, и контент подвергаются здесь детальной проработке, комбинация, которую возможно лучше обработать через прототип HTML, потому что создание нескольких вариаций для различных размеров экранов на такой ранней стадии проекта в лучшем случае будет нудным. Честно говоря, такая ранняя модель прототипа страницы из Википедии, возможно, служит своей цели. Но минусы для технологического процесса, дружелюбного к будущему проекта, очевидны.

Иллюстрация: Fabrice Florin, Wikimedia Foundation под лицензией CC-BY-SA и GFD / smashed.by/wkpmckp

Я называю эти уменьшенные модели «прототипами со ссылкой на контент». Если клиенту не понравятся ваши прототипы, вы потеряете всего лишь несколько минут работы. Цель такова: показать клиенту не «дизайн», а то, какой контент будет отражен в данном контексте, приблизительно обрисовав расположение и композицию, как предшествование визуальному дизайну. Это создает контраст с тем, как

сегодня оформляются многие шаблоны, где графический дизайн уменьшается до размеров детской книжки-раскраски.

Создать прототип со ссылкой на контент несложно. Воспользуйтесь перечнем контента, определите, какое содержимое и функции должны быть на данной странице, а затем нарисуйте прямоугольники, в каждом из которых будут представлены эти части содержимого и функций.

Сделать это нужно для каждого типа страницы. Вы можете озаглавить каждый блок, или обозначить его буквой или номером в соответствии с пунктом в перечне контента. Вот и все. Совсем непривлекательно, но достаточно эффектно, особенно если мы не пытаемся навязать свой продукт клиенту.

Этот метод может показаться странным поначалу, но мы все еще на том этапе, где вычисляем, появится ли форма регистрации на странице, и ориентировочно прикидываем, в каком месте она была бы уместней. Мы должны просто подтвердить эти основные факты. Пока нет необходимости их визуализировать.

Прототипы со ссылкой на контент можно делать на бумаге или в графическом приложении. Но я подбиваю вас создавать их в HTML и CSS. Да, вам придется сделать несколько CSS-макетов, но это просто. Плюсы в том, что вы можете создавать прототипы, которые адаптируются под размеры экрана и позволяют вам принять решение об отзывчивости (адаптивности) дизайна на начальном этапе. Прототипы со ссылкой на контент могут разрабатываться так, чтобы прямоугольник имел фоновый цвет, а не просто очертания. И это хорошо. Типографические свойства тоже подойдут. Но делайте все в упрощенном варианте.

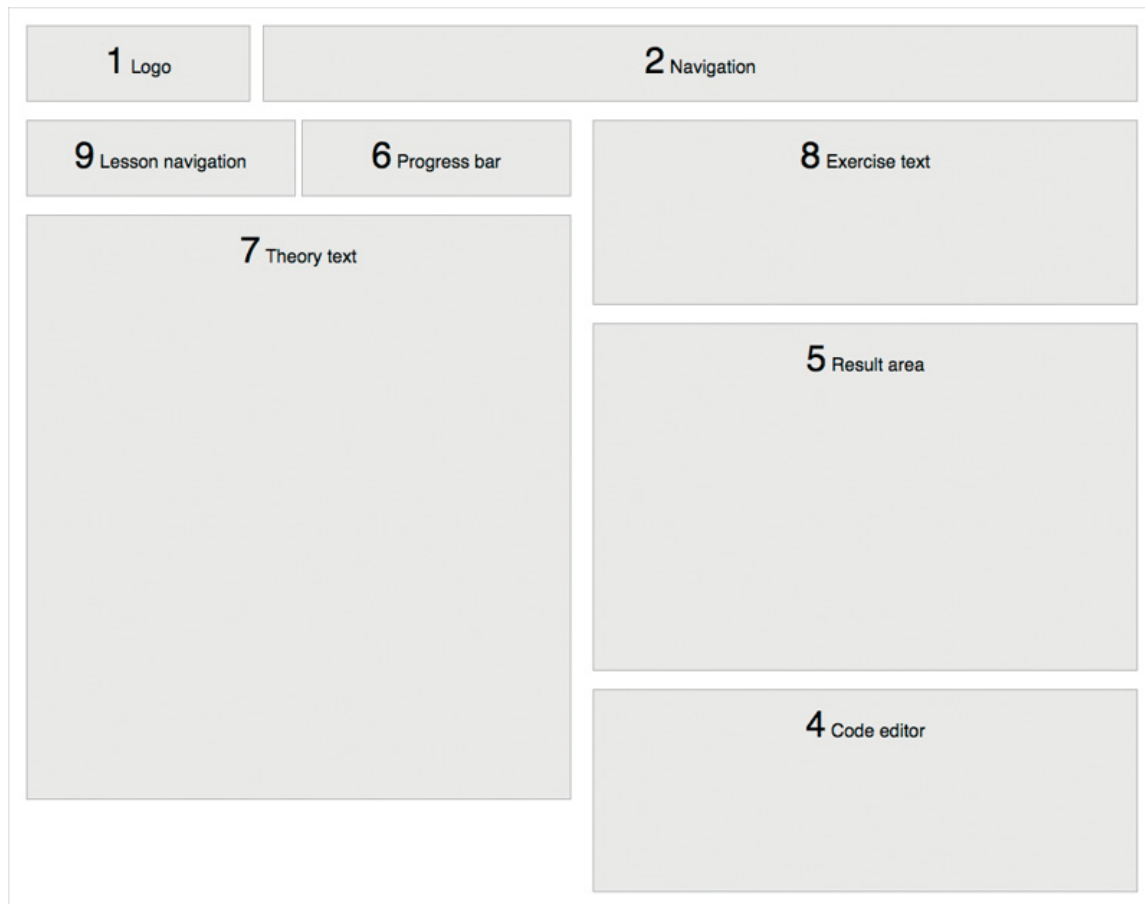
С этого момента прототипы становятся интереснее. Используя медиазапросы CSS, мы сможем начать работу над грубыми разметками для экранов различных размеров. Благодаря этому два наших прототипа превратятся в бесчисленное количество, потому что их так легко загрузить в браузер любого устройства. Потом мы сможем проанализировать возможности разметки и показать прототипы клиенту^[121].

Я часто показываю клиентам скриншоты прототипов в различных размерах (чтобы не создать впечатление того, что процесс разработки будет скорым).

Но для самих себя просмотр прототипов на различных устройствах имеет свои преимущества.

Мы заверстали два прототипа (рис. 10.3 и 10.4.) для нашего сайта «Три маленьких прямоугольника», по одной на каждую главную страницу, пронумеровали согласно пунктам в нашем списке перечня контента. Я также добавил названия позиций, чтобы мы то и дело не заглядывали в список:





Рисунки 10.3 и 10.4

1 Logo

3 Introduction

4 Signup form

2 Navigation

1 Logo

2 Navigation

3 Introduction

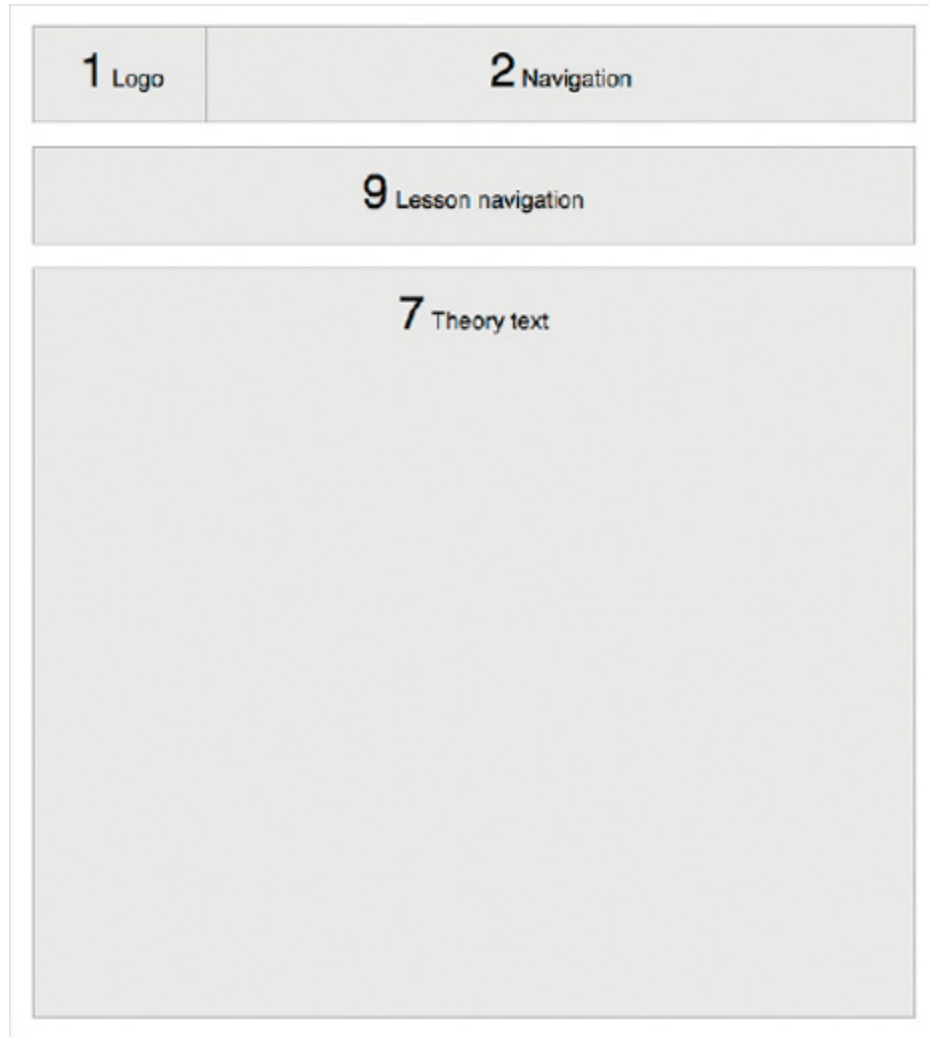
4 Signup form

1 Logo

9 Lesson navigation

7 Theory text

2 Navigation



Рисунки 10.5–10.8

Если мы что-то изменяем в перечне, прототип обновляется незначительно. Здесь мы использовали блоки (div) для ускорения процесса, и добавили немного CSS для оформления блоков и разметки.

Рис. 10.3–10.8. показывают наши два, да, только два прототипа на экранах разной ширины. Снаряженные медиазапросами CSS наши прототипы, основанные на HTML, позволяют нам изучить мультиплатформенную верстку на ранней стадии разработки. Мы уже можем начать подумывать о приоритетах и эффекте близости на экранах разных размеров. Представьте, что вы используете графические приложения для визуализации этих изменений в шаблоне, который богат на контент так же, как и модель Википедии (мы видели

ее выше). Представьте объем работы над каждым незначительным изменением.

Проектирование структурированного контента

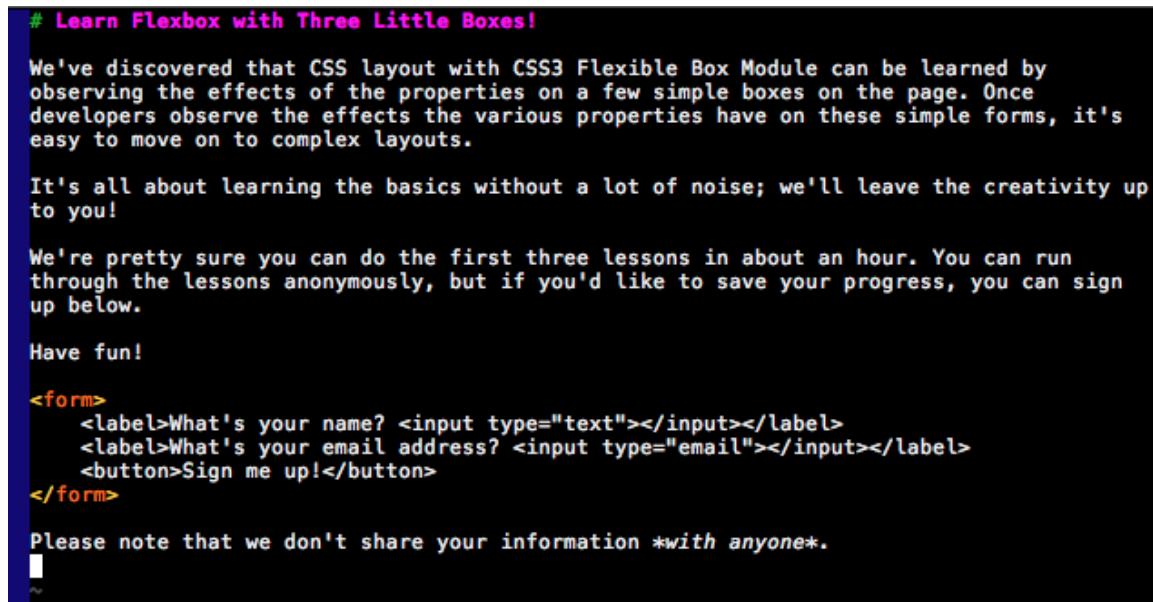


Рисунок 10.9. Простой структурированный контент. Это домашняя страница. «Три маленьких прямоугольника», которую я написал на Markdown. Преобразовать Markdown в HTML просто. Для этой цели я использовал Pandoc. Pandoc – это гибкий, универсальный конвертер: smashed.by/pandoc

В начальной точке создания прототипа, подумайте, как бы мы организовали наш контент, если бы не было никакой разметки, кроме линейной, той, что мы часто встречаем в мобильных версиях сайтов, когда один кусок контента располагается под другим. Эта философия «Сначала мобильные» («Mobile first»), которую популяризировал Люк Вроблевски, основывается на доступности и прогрессивном улучшении^[122]. Вы можете сказать, что цель заключается в разработке основного отображения приложения, которое функционирует достаточно хорошо везде, где может быть отрендерен простой HTML.

Learn Flexbox with Three Little Boxes!

We've discovered that CSS layout with CSS3 Flexible Box Module can be learned by observing the effects of the properties on a few simple boxes on the page. Once developers observe the effects the various properties have on these simple forms, it's easy to move on to complex layouts.

It's all about learning the basics
without a lot of noise; we'll leave the

Learn Flexbox with Three Little Boxes!

We've discovered that CSS layout with CSS3 Flexible Box Module can be learned by observing the effects of the properties on a few simple boxes on the page. Once developers observe the effects the various properties have on these simple forms, it's easy to move on to complex layouts.

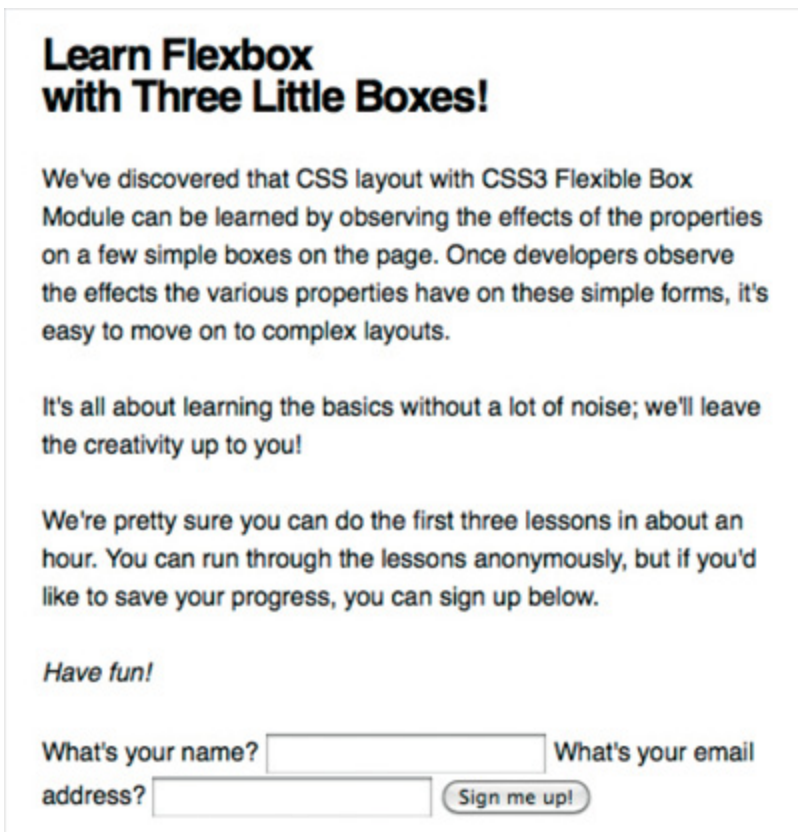
It's all about learning the basics without a lot of noise; we'll leave the creativity up to you!

We're pretty sure you can do the first three lessons in about an hour. You can run through the lessons anonymously, but if you'd like to save your progress, you can sign up below.

Have fun!

What's your name? What's your email address?

Please note that we don't share your information *with anyone*.



Рисунки 10.10–10.12. Эти скриншоты показывают наш структурированный в HTML-контент в браузере. Иллюстрации (вверху слева и внизу) показывают стили браузера по умолчанию в Opera Mobile Emulator и Firefox на компьютере соответственно. В варианте справа вверху мы начали работать над типографикой

Задайте себе вопрос о том, какой контент должен идти первым, какой вторым, какой последним и как это может повлиять на порядок нашего исходного кода.

Дизайнер и разработчик Брайан Ригер постоянно работает над дизайном текста на ранней стадии разработки. Это, по сути, то, о чем мы здесь говорим: структурирование и определение приоритетов, как если бы мы создавали линейный документ.

Вы даже можете подумать, что это дизайн для текстового документа или статьи, или книги. Нас интересует основополагающая структура контента: это заголовок, это цитата, это список и т. п.

В идеале мы должны использовать реальный контент, именно поэтому этот шаг связан с прототипами со ссылкой на контент. Так как наш контент структурирован, мы можем начать заниматься

типографикой: использовать гарнитуры и разные размеры, определять сетку, устанавливать межстрочный интервал (или высоту строки).

Кстати, все это можно сделать в HTML и CSS. Если вы берете разметку в формате plain-text, такую как Markdown, и потом преобразуете ее в HTML, вы можете создать структурированный дизайн контента так же быстро, как и в текстовом редакторе. Таким способом вы также можете взять базовый HTML и CSS, чтобы использовать для запуска адаптивного (отзывчивого) прототипа.

Когда вы пройдете этот шаг, мы получим настоящую базовую конструкцию для маленьких экранов. Загружайте страницу на мобильное устройство и смотрите.

Улучшаем опыт взаимодействия: специфика платформ и устройств, с учетом возможностей браузеров и устройств

Если мы все сделаем правильно, то наш основной контент и функции будут работать в большинстве веб-программ. Небольшое предостережение, если вы создаете веб-приложение, которое требует специальной технологии (например, картографическое приложение, где нужны CSS, JavaScript, изображения и возможности GPS в основе функционала). В этом случае линейный дизайн может не сработать или просто не подойти. В некотором смысле это очень плохо, потому что мы отгораживаемся от пользователей. Постарайтесь сделать доступным так много контента, насколько это возможно, и используйте ваши технологические основы как отправную точку. Наш линейный дизайн даст нам представление о том, как может выглядеть и работать сайт на некоторых мобильных устройствах.

А теперь мы бы хотели улучшить наш дизайн и опыт взаимодействия в браузерах и на платформах, которые поддерживают различное расширение функциональных возможностей. Несколько примеров:

- Линейный дизайн – это просто одна колонка контента. Мы можем изменять его для больших экранов и для случаев, когда на некоторых устройствах пользователь переходит от портретного режима к ландшафтному и наоборот. Можно делать разметку для большего количества колонок, если позволяет пространство. Различное размещение элементов может иметь смысл, потому что пользователи тоже могут по-разному взаимодействовать с большими экранами. Например, вы захотите перестроить или по-другому спозиционировать навигацию. Вам также придется пересмотреть важные части контента, потому что речь теперь не о том, чтобы просто расположить их поближе к верху.

- Некоторые устройства имеют возможности, которые мы бы хотели использовать в наших целях, например камеру или функцию GPS. Многие из нас хотели бы использовать функциональные свойства JavaScript, если они доступны (и часто это так и есть, правда, не всегда, особенно в устройствах низкого уровня). Как насчет

визуальных улучшений, таких как встраивание шрифтов и CSS-градиентов? Разработка в актуальных браузерах современных устройств позволяет нам проверить работают ли эти свойства (и работают ли как следует) и как они влияют на качество функционирования. Нам надо подумать о свойствах, которые лучше исключить из определенных устройств и платформ, и поддерживать их на других.

- Вполне возможно, что мы даже захотим добавлять, удалять или изменять сам контент, в зависимости от устройства или платформы. Например, мы однозначно захотим использовать уменьшенные версии изображений по умолчанию на маленьких экранах и мобильных устройствах, при этом использовать большие версии где-то еще. Также мы не хотим предлагать контент, который не подходит под определенные случаи использования. GPS-зависимый контент будет уместен только там, где поддерживается GPS, поэтому добавлять его мы сможем только для этих устройств.

Чтобы улучшить опыт взаимодействия структурированного контента, нам, как минимум, будет нужно составить список типов устройств, которые мы хотели бы поддерживать, группируя их в «классы» устройств. Проще говоря, сформируйте группу устройств с одинаковыми характеристиками, с тем чтобы сосредоточиться на классах устройств, а не на отдельных устройствах. Мы могли бы поддерживать только iOS или Android, но это уж было бы слишком ограничено. В результате мы можем наладить все так, чтобы приложение правильно выглядело и работало на любом устройстве.

Не классифицируйте в соответствии с основной физической формой, например такой как компьютер, смартфон, планшет и т. д. Эти категории важны менее, чем вы могли бы подумать. Вместо этого проанализируйте устройства (и заодно их браузеры по умолчанию) в соответствии с функциями, которые требует ваше приложение. Любой фактор может играть роль, будь то сенсорный ввод, размер экрана, пиксельное разрешение, геолокация, локальное запоминающее устройство, поддержка SVG и т. д.

Концентрация на свойствах гарантирует то, что наши решения останутся значимыми даже когда появляются новые устройства, которые трудно соотнести со средней потребительской и

маркетинговой категориями. Маркетинговые категории не скажут нам о том, что нам нужно знать (например, существует ли поддержка SVG и хорошо ли она представлена в этом браузере и на этом устройстве).

Чуть ближе: классы устройств

Для сайта «Три маленьких прямоугольника» нам действительно нужно обратить внимание на классы устройств. Сайт будет образовательным и требует ввода кода пользователем, чтобы узнать, как он работает. Код будет интерпретироваться в браузере, а результат воспроизводиться на экране.

Давайте реально смотреть на вещи: проверить все это на большинстве мобильных телефонов невозможно. В общем, вот классы устройств, с которыми мы будем иметь дело:

- с поддержкой HTML: необходима для обучающего текстового контента (теория и синтаксис);
- с поддержкой JavaScript: обязательна для интерактивных заданий;
- с большими экранами: не требуются для текстового контента, но удобны для упражнений;
- с кнопочной (неэкранной) клавиатурой: предпочтительна для набора кода;
- с браузером, который должен поддерживать последние спецификации Flexbox (иначе не будут работать упражнения).

Но у нас серьезная проблема. На момент написания этой книги новейшие спецификации Flexbox поддерживаются только в Chrome Canary^[123], который ограничивает наши возможности работы с многочисленными устройствами. Если бы это был реальный сайт, мы бы столкнулись с тяжелым выбором. Предположим, мы не хотим создавать механизм разметки Flexbox в JavaScript, мы застряли в Chrome Canary – единственном браузере, в котором будет исполняться код для задания.

Однако согласно нашему перечню контента и классам устройств мы точно можем обеспечить ценный текстовый контент (теория по Flexbox, синтаксис и т. п.), а потом уже предусмотреть интерактивные элементы, только если они на самом деле используются (т. е. когда требуемые функции доступны).

Определить лучший способ для этого очень сложно, и это выходит за рамки данного раздела. Но давайте представим, что мы используем JavaScript чтобы проверить, поддерживаются ли определенные свойства Flexbox в браузере. Если да, тогда интерактивные компоненты можно добавлять на сайт.

Улучшение опыта взаимодействия против ключевых сценариев

Когда мы доходим до фазы улучшения, мы должны спросить себя: «Эта функция настолько необходима для сайта в том виде, в каком она есть сейчас?» Если это так, тогда это то, что я и кое-кто еще любим называть «ключевым сценарием», т. е. это одна из базовых задач пользователя на сайте, которая в большинстве случаев не рассматривается как «улучшение».

Значит ли это, что улучшения несерьезны и не важны? Конечно же, нет.

Возьмем, к примеру, простое приложение для списка дел. Цель такова — сделать приложение используемым независимо от платформы; браузер должен только поддерживать HTML, включая основные функции для форм. Спору нет, что приложение списка дел должно позволять пользователю совершать по крайней мере несколько вещей:

- добавлять пункты дел,
- редактировать текущие пункты,
- отмечать пункты как выполненные,
- архивировать и удалять старые пункты.

Это наши ключевые сценарии. Конечно, мы могли бы реализовать больше (и многие приложения списков дел умеют больше), но пока давайте сосредоточимся на этих действиях, которые необходимы для целей приложения.

Одним словом, без JavaScript мы отображаем форму, которая содержит одно или несколько текстовых полей, куда пользователь может вводить один или несколько пунктов, а также клавишу для подтверждения. Содержимое формы посылается на сервер, перед пользователем появляется новая страница с их пунктами дел. Каждый

пункт может иметь пару опций (возможно, в виде чекбоксов), и вы можете выполнять действия кликом на клавишу. Еще одно – данные должны передаваться на сервер и обратно, но по крайней мере это будет работать.

В любом браузере, который поддерживает JavaScript, мы хотим мгновенных действий: отметка задания как выполненного, возможно, сотрет текст и/или добавит галочку, а пользователь увидит изменения сразу же.

Это и не ключевой сценарий, и не бесполезное улучшение: это очень полезное улучшение, которое добавляет больше удобства использования нашему приложению. Зависимость от JavaScript не значит, что это плохо, только ее, возможно, следует избегать в определенных случаях.

И если этого можно избежать, почему бы тогда не начать с базовых функций на устройствах более низкого уровня? Помните, что многие неотразимые сайты с высоким трафиком, известные на сегодня, создавались, когда еще JavaScript не был распространен так широко. И все же эти сайты удовлетворяют потребности миллионов пользователей. Создать великолепный опыт взаимодействия без JavaScript вполне возможно. Огромное значение имеют контент, форма и исполнение, а не инструментальные средства.

«Три маленьких прямоугольника» представляет собой слегка иную проблему. Так как задания – это первичное использование сайта, есть определенная образовательная ценность в представлении четкой информации о спецификации Flexbox, ее синтаксиса и применения. Это не «все или ничего». Способность читать текстовую информацию – это тоже ключевой сценарий. Это текстовая информация содержит примеры кода и читается, как обучающее руководство.

Так или иначе опыт улучшается в Chrome Canary (вероятно, на настольных компьютерах и ноутбуках), где интерактивные компоненты добавляются к заданиям. Это пример изменения контента на базе свойств устройства и браузера. Когда все сделано хорошо, пользователи неподдерживаемых платформ не ощущают потери важного контента.

Пользователь, у которого телефон семилетней давности, не будет ожидать того, что последний код на Flexbox, который он введет,

воспроизведется верно. И также он не столкнется с нефункционирующими интерактивными компонентами.

Нам надо понять, где и когда представлять определенный контент и размещение. Я пользуюсь визуальным инструментом, который называю графиком точек прерывания. Он помогает мне отметить, где контент и дизайн могут изменяться.

График точек прерывания

Раньше мы рассматривали три вещи, которые хотели бы изменить по возможности для определенных классов устройств и/или размеров экранов:

- дизайн и расположение,
- функциональность и свойства,
- контент.

Точки, где эти изменения имеют место, называются точками прерывания. Они очень часто устанавливаются в отношении ширины окна просмотра в пикселях, в таких случаях они обычно вводятся в медиазапросы CSS и/или JavaScript. Например, вы могли бы сказать, что когда окно просмотра шире, чем 600 пикселей, обратитесь к макету X.

Точки прерывания не ограничиваются шириной окна просмотра. Любая характеристика класса устройства может быть точкой прерывания: GPS (против отсутствия GPS), камера (против устройства без камеры) и т. п. Мы можем отметить эти точки на графике точек прерывания^[124].

График точек прерывания дает дизайнерам и разработчикам видение аспектов сайта, которые будут изменяться по определенным параметрам устройства. Эти изменения могут быть представлены уменьшенными прототипами (чтобы показать изменения в разметке) или метками, которые идентифицируют некоторые аспекты (чтобы показать изменения в функционировании и контенте). Мы можем отметить классы устройств (и даже специфические устройства) на графике точек прерывания. Это не только важно для группы разработчиков, но и позволяет клиенту увидеть, как сайт адаптируется в определенных условиях.

Сделать график точек прерывания достаточно быстро и легко. Он выглядит как временная шкала с горизонтальными линиями, представляющими полный спектр параметров. Эти параметры могут быть какими угодно: размером экрана, поддержкой специфических технологий или даже классом устройства.

Потом параметры группируются логически и располагаются над и под горизонтальной линией. Расположение параметров таким способом показывает, что их отношение друг к другу может быть полезным (например, класс устройства может совпасть с размером экрана), но такое не всегда возможно. После исследования, когда вы определили точки прерывания, каждая может быть представлена символом на линейке (мы использовали круг в примере ниже). Наборы свойств представляются прямоугольными затенениями за горизонтальной линией, охватывая точки прерывания среди которых они поддерживаются. наброски миниатюр разметки для различных размеров могут быть включены под соответствующими точками прерывания.

Будьте осторожны! Установка точек прерывания для определенных устройств, хотя иногда это и требование, может вызвать проблемы. Изменения, которые вы делаете для платформы, могут хорошо выглядеть и хорошо работать на определенных устройствах, но, возможно, не на других с подобными функциями. По этой причине многие сайты сегодня используют функцию определения устройства.

То, что мы здесь делаем, – это планируем прогрессивное улучшение. Цель такова: обеспечить как можно лучший опыт взаимодействия в контексте каждого устройства. Мы осуществляем это с помощью обеспечения самого основного опыта на самых основных устройствах (в некоторых случаях просто с HTML) и расширяем опыт через свойства, так как все больше и больше навороченных устройств и браузеров поддерживает их. Прогрессивное улучшение сводит до минимума риск, что продукт не будет вообще работать на основных классах устройств. Не все пользуются iOS. Такое планирование дизайна важно. Оно дает ценную информацию дизайнерам и разработчикам и шанс испытать удовольствие от свободы и скорости создания отзывчивого прототипа.

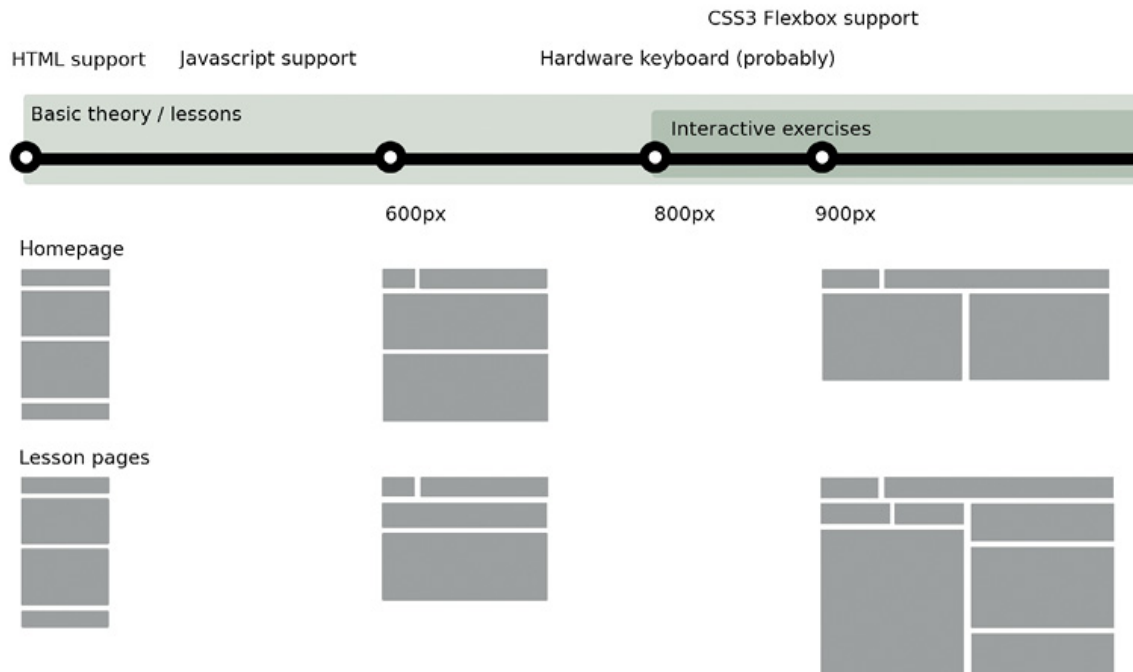


Рисунок 10.13. График точек прерывания для «Трех маленьких прямоугольников»

Проектирование в браузере: Адаптивное и отзывчивое прототипирование

Вы могли слышать о практике проектирования в браузере. Должно быть, вы подумали, как и я: «Звучит великолепно, но только как я смогу сделать это быстро, когда дизайн еще не утвержден? Не похоже ли это на создание статических шаблонов на стадии разработки? Гораздо проще сделать макет в Photoshop». Но вот дилемма: мы хотим сделать что-то, что клиент может увидеть и одобрить, но мы не хотим начинать сейчас реальную разработку сайта.

Я сталкивался с этой проблемой очень долгое время, и решение менялось в зависимости от проекта. Я заметил, что если я начинаю работать в HTML на стадии прототипа, потом гораздо легче и быстрее расширить шаблон до полноценного визуального дизайна, чем в случае, когда я начинаю разработку с графического редактора. Те же плюсы, которые применяются для прототипов в HTML, действуют и здесь: вы должны работать с целевой технологией, и вы можете оценивать варианты (и показывать их клиенту), такие как изменения в ширине окна просмотра.

Мы можем рассматривать эти HTML-макеты как развитие прототипов на основе веба. В этом случае прототип дает нам основу HTML для построения. Мы просто берем дизайн структурированного контента, вставляем его в шаблон, и мы на правильном пути к отзывчивому дизайну – прямо в браузере! Этот дополнительный плюс нам, что мы можем сразу протестировать дизайн на реальных устройствах на очень ранней стадии. Так как технологии развиваются, например дисплеи с высоким разрешением, расширяется брешь между веб-опытом и статическими изображениями, потребность в проектировании в браузере увеличивается.

Мы увидели, что наш основной линейный дизайн на структурированном контенте лучше создавать в HTML и CSS, а причина в том, что мы начинаем расширять основной типографический дизайн в CSS. Мы будем загружать эти файлы в несколько браузеров на нескольких устройствах, чтобы видеть, что происходит и дорабатывать их.

Возможно, это выглядит как специальный дизайн, но это не так. Я не говорю о том, что вы пропустите проверенные этапы графического дизайна и скетчей. Мы всегда должны думать и делать наброски, прежде чем что-то выполнять на компьютере. Пожалуйста, продолжайте делать это. Я предлагаю выполнять проект или предложение для клиента в веб-технологиях, а не в графическом редакторе, таком как Photoshop.

Богохульство? Думаю, нет. Наши средства позволяют нам роскошь немедленной публикации. Каждый может создать веб-страницу за секунды. Вы не смогли бы это сделать в печатном виде. Когда работа выходит с печатного станка, существует легкое напряжение, как будто мы должны проверить, как все обернется. Почему мы настаиваем на разработке статических изображений, заставляя клиента утверждать их, и потом делаем ту же работу снова в HTML? Возможно, мы думаем, что дизайнеры не могут или не должны писать код. Спорный вопрос, который не является неминуемым препятствием. В худшем случае дизайнеры могли сделать свои наброски, а потом сесть с верстальщиком и создавать дизайн в браузере.

Короче говоря, мы можем пропустить этап использования графических редакторов для проектирования и использовать их отдельно для редактирования изображений, что имеет свой смысл.

Создание нашего дизайна в HTML и CSS дает следующие преимущества:

- Дизайнеры видят, что нужно отшлифовать в дизайне, чтобы он лучше выглядел в определенных браузерах и на определенных платформах. В качестве бонуса в процессе они узнают подробности о разработке в Сети.

- Разработчики видят, где правильно, а где нет, с технической точки зрения. Как бонус они постигают процесс графического дизайна (если они не дизайнеры).

- Клиент и другие участники видят – и учатся принимать – разницу между браузерами и платформами. Когда они видят рождение своего проекта, они начинают понимать несколько больше о том, как работает веб и понимать ценность того, что сила веба заложена в контенте, правильно отображаемом на всех устройствах.

Полезно будет также показать клиенту скриншоты этих HTML-прототипов. Я делаю скриншоты в различных браузерах, с разной шириной окна просмотра и/или устройств^[125] и говорю клиенту: «Вот несколько изображений того, как этот дизайн может работать в различных сценариях». Показ их сначала именно в качестве изображений помогает проводить обсуждение вопросов по дизайну и избежать глубинного анализа отдельных частей контента. Но если клиент хочет изменить стиль, допустим, «шапки», вы можете сделать это в CSS и сделать новые скриншоты очень быстро. Представьте, что вы делаете эти изменения в различных документах Photoshop – невесело! Клиент не заметит (или его не волнует) то, что эти изображения не созданы в Photoshop.

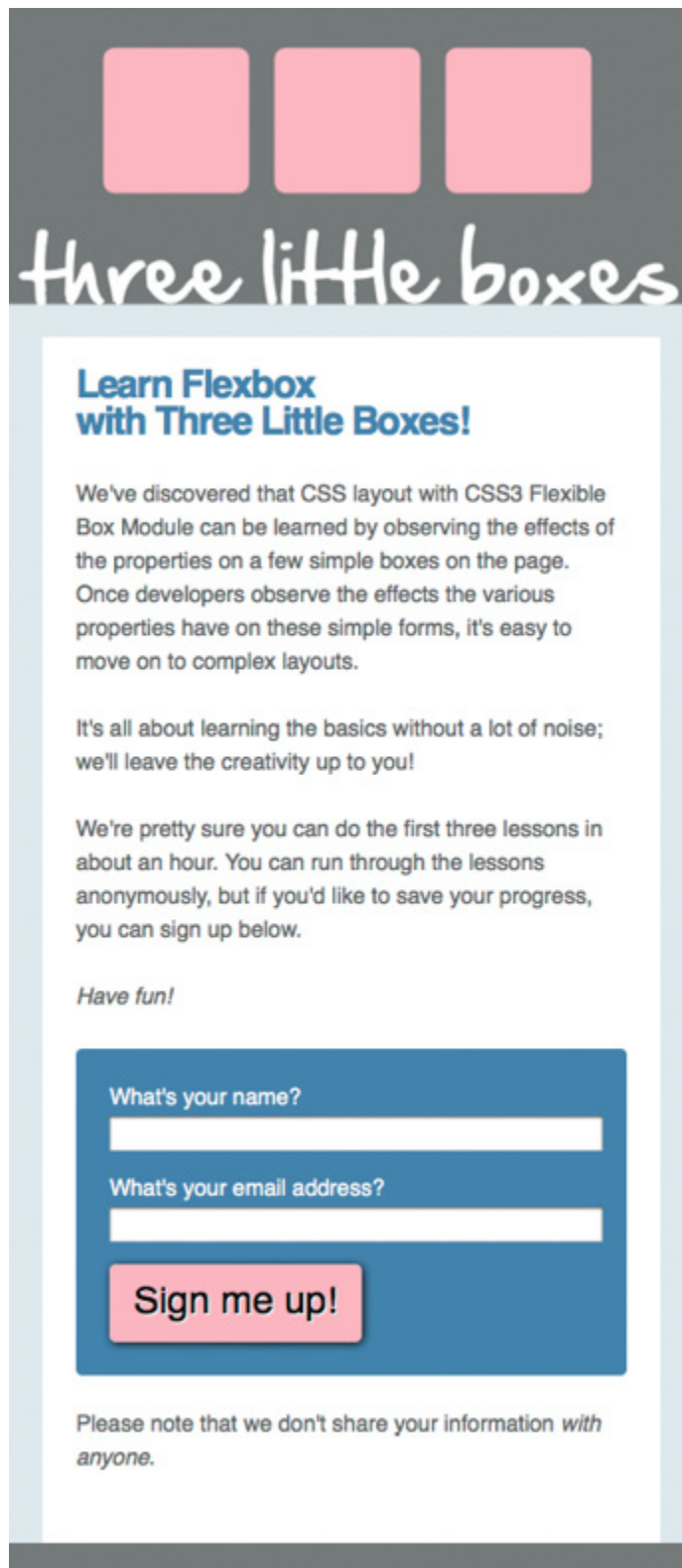


Рисунок 10.14. Взяв наш простой, структурированный контент в HTML, мы создали дизайн для него в линейной форме. Опираясь на

скетчи, мы начали добавлять стили к нему. Потом, ссылаясь на наш график точек прерывания, мы сможем использовать медиазапросы, чтобы начать писать код для невероятно больших экранов и браузеров, которые поддерживают наши желаемые свойства

Использование прототипов в виде изображений не говорит о том, что мы обманываем клиента. Скорее, это помогает избежать впечатления, что вы далеко продвинулись в разработке, когда на деле это не так. И это важно, потому что вы должны на данном этапе сосредотачиваться на дизайне. А к удобству пользования и внедрения обратитесь позже.

На самом деле, вы продвинулись далеко, но вашего клиента не должно волновать это. Следование технологическому процессу – это преимущество для вас и вашей команды. Клиент тоже останется доволен, когда сайт будет завершен, он будет выглядеть почти так же, как и в прототипе. Как вы будете разрабатывать прототип – это ваше дело. Некоторые генераторы статических сайтов могут приносить пользу: контент утверждается в файлах с разметкой Markdown, которые передаются по конвейеру через шаблоны, чтобы создать статический сайт. В конце концов, это просто инструменты. Вам все же придется сделать несколько шаблонов в HTML и CSS-файлов для дизайна. Если вы уже таким образом сделали исходные прототипы, то обычно контент распределяется по различным прямоугольникам на странице, к которым применяется CSS, основанный на ваших дизайнерских набросках.

Многие дизайнеры эффективно используют метаязыки на основе CSS и/или препроцессоры или компиляторы, такие как SASS и LESS^[126] для значимых изменений в CSS в относительно короткие сроки. В сочетании с контролем версий эти средства могут осуществлять разработку в браузере быстрее, проще и намного интереснее, чем работу в графическом редакторе. Апробирование Великой Новой Идеи клиента (или вашей собственной) становится обычным делом, а процесс может помочь вам держать проект «в тонусе» и сохранить бюджет, сократив объем работ, требующийся на освоение новых идей.

Когда клиент одобрит базовый дизайн, вы можете точно настроить и презентовать несколько важных страниц как полнофункциональных

прототипов HTML в противоположность скриншотам. Это не сложно: вы проделали почти весь путь, так как ваш дизайн уже в HTML. Используйте это как свое преимущество, сосредоточьтесь на дополнительном дизайне: сообщения об ошибке, изменение состояния и т. п.

Сейчас у вас также есть время, чтобы рассмотреть эти вопросы, потому что вы пропустили шаг визуального анализа файла Photoshop и не вывели из него тонкости дизайна.

Все-таки не будем слишком жестоки к графическим редакторам. Приложения типа Photoshop прекрасно подходят для быстрых дизайнерских решений, работы с цветом и для проработки графических исходников. Используйте Photoshop для быстрых экспериментов с такими вещами, как цвет, фотография и кое-что из типографики. Photoshop хорошо работает для сбора карт настроек из всех этих элементов в их концептуальной фазе перед созданием дизайнерских впечатлений.

В конечном итоге разработка в браузере таким способом приносит огромную пользу:

- Мало или вообще несколько времени не тратится на графические редакторы. Все время посвящается коду, который мог бы использоваться полностью или частично в процессе разработки самого сайта. Использование контроля версий наподобие Git позволит вам свободно экспериментировать со своим дизайном непосредственно в коде и не волноваться о постоянных изменениях.

- Выполнение многих требований клиента, как правило, история обычная (за исключением возможных изменений в макете). Эти изменения будут видны немедленно в окнах просмотра различной ширины и т. д. Делать такие изменения в редакторе – это огромная трата времени и ничего веселого.

- Демонстрация клиенту изменений состояний (например, авторизован или разлогинен пользователь) возможна с минимумом усилий.

- Если вы внимательно относитесь к качеству вашего кода на фазе разработки прототипа, многое из него пригодится для создания сайта.

Создание прототипа, как дизайна впечатлений и рабочего прототипа может показаться нелогичным, но я настаиваю, чтобы вы попробовали так сделать несколько раз, и вы привыкнете к идее. Это реально может сэкономить время. А вам никогда не нужно будет бросать свою целевую технологию. В отличие от традиционного процесса разработки сайта, она преподносит несколько сюрпризов.

Новое мышление, новый подход к дизайну

Давайте взглянем на наш новый технологический процесс:

1. Создать перечень, включив в список наш контент и функции.
2. Создать быстрые вайрфреймы в HTML и CSS, пометив прямоугольники в них соответствующим номером из перечня.
3. Используйте некоторое количество реального контента в структуре.
4. Создайте линейный дизайн для этого контента.
5. Рассмотрите классы устройств и создайте график точек прерывания.
6. Сделайте скетчи, обдумайте и составьте концепцию дизайна для различных точек прерывания.
7. Объедините контент из линейного дизайна и вайрфрейма на HTML, создавая дизайн в виде прототипа на HTML.
8. Покажите клиенту скриншоты прототипов, представляя их как графический дизайн.
9. После одобрения дизайна (необязательно) доработайте HTML-прототип и представьте его (например, для пользовательского тестирования).
10. Если все сделали правильно, большая часть кода может использоваться для создания сайта.

Этот технологический процесс эффективен как для переделки, так и для создания нового сайта. Это метод ориентированный на будущее, потому что мы рассматриваем различные классы устройств с самого начала. Поэтому новые устройства, которые появляются на рынке, не навредят сайту.

Этот технологический процесс эффективен, потому что каждый шаг нераздельно соединен с другим. И дизайнеры, и клиенты

постоянно сталкиваются с реальными жизненными проблемами, такими как отображение в браузере.

Из своего опыта скажу, что клиентам нравится такой технологический процесс. Он не создает визуального напряжения так, как это бывает при традиционном дизайне технологического процесса, где продукция обрушивается как снег на голову. Он строится медленно, пошагово, клиент может включаться во весь процесс. Ничего удивительного. Каждый шаг приводит к более привлекательному результату по сравнению с предыдущим. Вещи конкретизируются очень быстро, а клиенты с самого начала видят эффект от своих решений. Некоторые дизайнеры и разработчики уже работают таким способом. Для некоторых, может быть, и для вас тоже, нужно время, чтобы привыкнуть. Попробуйте эти приемы в вашем проекте сегодня и откройте то, что работает для вас.

Возможно, процесс радикально отличается от того метода, которым мы привыкли работать, но это эффективный способ совершить путешествие как дизайнер Сети.

Об авторе



* * *

Стивен Хей (р. 1970) родом из Оранжа (штат Калифорния, США). Живет в городе Леуварден на севере Нидерландов. После получения степени бакалавра по графическому дизайну и изобразительному искусству он отложил в сторону свои мечты стать художником и прошел путь от дизайнера до арт-директора в дизайнерском агентстве, где специализировался на фирменном стиле и дизайне упаковки. Стивен сделал свой первый сайт в 1995 году и влюбился в веб. Сеть казалась ему идеальным сочетанием дизайна и технологий. После 10 лет работы креативным директором в фирме, связанной с веб-

разработками, он стал независимым консультантом, и это дало ему больше времени для занятий любимыми делами. Ему нравится помогать людям учиться, он получает удовольствие, когда рассказывает или пишет про веб. Работа занимает не всю его жизнь: он играет на баритон-саксофоне и состоял в джазовом квинтете. Еще он любитель фокусов, искусства и кино. Важнейшие уроки, которые Стивен получил за свою карьеру: критически мыслить, заниматься тем, что доставляет удовольствие, делать все настолько хорошо, насколько ты можешь, и продолжать учиться.

О рецензенте



* * *

Брайан Ригер родился в год, когда распались «Битлз», Джими Хендрикс стал номером один, а население мира достигло 3,6 млрд человек. Он родом из Торонто и обычно большую часть года проводит в Эдинбурге, а оставшуюся часть старается провести в Бангкоке. Он довольно часто переезжал и рассматривал много мест в качестве своего дома, включая Шарлоттаун, Ванкувер, Сурабаю, Пхукет, Гонконг, Манилу, Лондон, Глазго и Брайтон. Дом там для него, где оказался сейчас, что хорошо сочетается с его жизненной позицией: «Удобно где бы то ни было!» Образование Брайана включает немного занятий дизайном, театром и анимацией в разных школах, в разное время и на разных уровнях. Брайан любит путешествовать, его любимый цвет белый, потому что он дает ощущение возможностей. Самый большой урок, который он получил за свою карьеру, — спрашивать все и всегда. Его послание читателям: «Plus ça change, plus c'est la même chose („Чем больше меняются вещи, тем больше они остаются прежними“. — фр.). Будьте гибкими, принимайте перемены...»

Стать невероятно гибким: создание атомов и элементов

Автор: Энди Кларк

Есть три слова, которые, я думаю, обобщают работу в Интернете для многих из нас. Вот они:

- Отзывчивый
- Сеть
- Дизайн

Итан Маркотт дал имя комбинации из тянущихся сеток, изображений и медиазапросов CSS3. И после этого разработчики со всего мира разработали ряд классных шаблонов, скриптов и шаблонных решений для многих сложных задач, основанных на том, что Итан Маркотт назвал отзывчивый веб-дизайн.

Тем не менее отзывчивый дизайн это не просто то, как дизайнеры и разработчики используют такие технологии, как медиазапросы CSS3, и это не то, как мы решаем проблемы с отображением картинок разных размеров или как справляемся с табличными данными. Это просто технические проблемы; стать отзывчивым – это не просто преодоление технических проблем. Это не означает, что надо изучать языки и то, как их лучше использовать. Хотел бы я, чтобы все было просто. Но, к сожалению, это не так. Отзывчивый дизайн очень и очень сложный процесс.

Нравится вам это или нет, отзывчивый веб-дизайн ставит под сомнение все, что мы знаем о Сети у всех, кто включается в процесс. Вот почему в этом разделе мы хотим продемонстрировать, что старые способы разработки больше неуместны, а я хочу представить новый путь создания отзывчивого веб-дизайна, который работает на меня и моих клиентов.

Ядро ваших действий

Не думаю, что я одинок в своем мнении о том, что отзывчивый веб-дизайн представляет собой фундаментальное изменение того, что для нас значит разработка для Интернета. Энди Хьюм пишет^[127]:

«Для меня отзывчивый дизайн – это еще один пример того, как веб-дизайн возвращается на путь Дао. Вот почему он не является дополнением или свойством. Он является ядром того, что вы делаете».

Я согласен с Энди. И хотя я знаком со многими, кто рассматривает отзывчивый дизайн просто, как одно из веяний, но надеюсь, что это, возможно, одно из самых больших и важных изменений в веб-дизайне с рождения Интернета. Как-то я написал^[128]:

«Все, что неподвижно и нечувствительно – это уже не веб-дизайн, это нечто другое. Если вы больше не улавливаете прирожденную изменчивость веба, вы не веб-дизайнер, а некто другой. Веб-дизайн – это отзывчивый дизайн. Отзывчивый веб-дизайн – это самый правильный веб-дизайн».

Я не преувеличиваю. Я настаиваю на этом, даже если некоторые думают, что я немного помешался на этом. Я считаю, что принимать во внимание реакцию дизайна при отображении на миллиардах устройствах разных размеров, форм и возможностей, это один из самых важных аспектов работы веб-дизайнера на сегодняшний день.

Как, когда и зачем?

Отзывчивый веб-дизайн изменяет то, что мы делаем для Сети, поэтому это также означает, что мы заставляем его изменяться. Это касается не только дизайнеров и разработчиков. Отзывчивый веб-дизайн затрагивает каждого, кто придумывает, разрабатывает, создает, оплачивает или пользуется Сетью.

- Контентный стратег? Вы.
- Интерактивный, графический дизайнер или разработчик пользовательского опыта? Да.
- Разработчик пользовательского интерфейса или серверный программист? Тоже вы.
- Босс, клиент и заказчик? Держу пари, вы.
- Пользователь? Тоже вы, в хорошем смысле этого слова.

Отзывчивый веб-дизайн задает больше вопросов, чем дает ответов. Он влияет на рабочие взаимоотношения и взаимодействие между

всеми, кто участвует в каждом процессе, – от специалистов по работе с контентом, дизайнеров и разработчиков всех мастей, до наших боссов и заказчиков, которые, в конце концов, утверждают нашу работу и платят за нее.

Отсюда возникают проблемные вопросы: как, когда и зачем.

Эти проблемы не всегда будет легко преодолеть, а изменения, которые они привносят, не всегда популярны. Скорее встретишь стойкое сопротивление со стороны тех людей, которые не могут или не будут видеть нужды приспособливаться.

Просто. На самом деле просто

В 1998 году, когда я открывал нашу крошечную дизайн-студию, самыми серьезными техническими вопросами, с которыми я столкнулся, были различия в том, как отображается мой дизайн в браузерах Internet Explorer 4 и Netscape 4.

По правде говоря, даже если мне приходилось работать с незрелыми технологиями и ужасными браузерами, все было тогда достаточно просто. У всех.

Действительно просто.

Посмотрите, как я работал. Спорим, что вы делали то же самое? А может, и еще делаете?

1. Я делал дизайн в Adobe Photoshop или Macromedia Fireworks – дизайн, который предназначался для всеобщего обозрения. Без разницы, какой бы браузер использовался или какого размера был бы экран.

2. Потом я показывал этот дизайн – в виде композиции или макета (я называю это статическими изображениями, потому что они неоживленные и не интерактивные) – своим клиентам.

3. После этого я делал изменения, основанные на обратной связи. Я переделывал и экспортировал новые статические изображения. Когда они утверждались по второму, третьему или четвертому кругу, я нарезал дизайн для верстки на HTML и CSS, публиковал его, забегал домой на чай и включал телесериал Animal Magic^[129].

Оглядываясь назад, я понимаю, что создать простой дизайн было несложно, потому что для нас и наших боссов и заказчиков Интернет

был доступен только на настольных компьютерах. Но сегодня мы не можем делать только один дизайн. Это уже не так легко. Если раньше у нас было два важных браузера на компьютере, то сейчас у нас их сотни на всех видах устройств, которые люди используют для доступа в Интернет.

Превратности быстрой смены положения дел

Даже на рубеже тысячелетий, мы должны понимать, что Интернет фундаментально отличается от других медиа, и что часть его уникальности в том, что мы не можем достаточно контролировать, как люди просматривают и взаимодействуют с контентом и сервисами, которые мы предоставляем. Правда в том, что мы всегда должны были разрабатывать дизайн для гибкой природы Интернета.

На самом деле некоторые люди это понимали, и они пытались предостеречь нас от глупых решений. В 2000 году Джон Олсоп написал статью, которую многие считают самой важной о веб-дизайне, которая когда-либо была написана: «Дао веб-дизайна». В ней Джон писал^[130]:

«Контроль, которым дизайнеры владеют на печати и часто жаждут в интернет-среде – это просто функция ограничения печатной страницы. Мы должны принять тот факт, что у Интернета нет таких ограничений, и его дизайн гибкий. Но сначала мы должны принять превратности смены вещей».

Хороший совет.

Приняли ли мы его?

Нет. Глупцы.

Что мы делали?

Мы дурачили себя тем, что думали, будто у нас все под контролем. Мы пытались влиять на те же уровни контроля, что и для печати. Мы совершенно игнорировали проблему гибкости Интернета и старались сделать его неподвижным. Вот как.

Разработка в сжатых рамках

Когда я перешел от печати к Интернету, это было новшеством. Правила отсутствовали. Поэтому я и дизайнеры вроде меня вносили

принципы, идеи и средства, хорошо нам известные. Наши работодатели тоже были новичками в вопросе Интернета, поэтому они привносили с собой приемы допечатной подготовки и печати, что работало на них.

Я использовал Photoshop для разработки сайта, и начинал каждый проект с создания нового холста. Я рисовал прямоугольник и затем заполнял его материалом.

Те первые прямоугольники были размерами: 640 пикселей шириной на 480 пикселей высотой, потому что таким было разрешение большинства мониторов в то время. Разработчики привыкли все контролировать, поэтому делали безумные вещи наподобие написания скриптов для фиксирования размеров окон браузера, чтобы точно вписаться в размер и размещали контент, используя пять или более фреймов^[131]. Этот 640 пиксельный прямоугольник был тесным. Если вы помните, заказчики не хотели, чтобы люди прокручивали сайт для просмотра (вообще), и все, что было на нем, должно было вмещаться в первый экран без прокрутки. 640-пиксельные экраны вскоре уступили место 800 (на 600) пиксельным, поэтому дизайнеры стали рисовать большие прямоугольники. Ощущение широко открытого пространства было опьяняющим. Но продлилось оно недолго, до того момента когда эти большие прямоугольники стали тесными, и мы начали рассматривать размеры от 800 до 1024 (на 768) пикселей.

Снова и снова, на каждом этапе расширения мы изучали серверные логи и статистику посещений сайтов, и мучились над решением, было ли безопасно расширять сайт.

Может показаться странным, что мы сегодня смотрим назад, но в 2005 году одной из самых горячих тем для обсуждения было противостояние фиксированного и тянущегося дизайна. Нет, без шуток, так оно и было.

Я брал интервью у Джейсона Санта Марии о его редизайне сайта A List Apart, который он создал для людей, делающих сайты^[132]. Мне хотелось знать, почему Джейсон выбрал для сайта фиксированную ширину 1240 пикселей вместо того, чтобы внедрить тянущуюся разметку, основанную на процентном соотношении. Джейсон ответил мне^[133]:

«ALA всегда старался быть одним из тех сайтов, которые идут впереди. Мы уже не поддерживаем размеры 800 × 600 и 640 × 480. А вы? Людей бесило, когда сайты переставали поддерживать 640 × 480... теперь никто не говорит ни слова. Времена меняются. Поверьте мне, вы увидите больше сайтов, которые расслабились».

Люди возмущались этим новым расширенным сайтом List Apart. Но не серьезно.

Джереми Кейт поддержал мысли Джона Хикса о отдельной дихотомии^[134] «между чувствительностью дизайнеров и предпочтениями пользователя». Джереми сказал^[135]:

«Спорить о 640, 800 и 1200 пикселях – это все равно, что спорить о том, что вкуснее – «Пепси» или «Кока-Кола», когда на самом деле стакан простой воды был бы намного более освежающим. Игра в цифры – отвлекающий маневр. Большой отвлекающий маневр постоянной ширины».

Годами мы дурачили сами себя, думали, что если 640, 800, 1024 и выше – это самые распространенные разрешения, то мы можем разрабатывать дизайн для этих постоянных размеров. Мы отчаянно цеплялись за дизайн с постоянной шириной, потому что в реальности Интернет – эта изменчивая среда без общепринятого размера холста, без границ – был слишком устрашающим.

Интернет не знает границ

Так как сегодня мы не можем предвидеть размер или формат, в котором будет просматриваться наш контент, Интернет фактически не имеет границ. Так что же делать дизайнерам?

Когда 640 пикселей превращались в 800, а затем в 1024, мы использовали все большие холсты с постоянными размерами как отправные точки наших разработок. Мы рисовали все большие и большие прямоугольники и заполняли их нашим контентом от границ холста внутрь.

Когда в 2007 году Стив Джобс (благослови Господь его душу) вытащил из кармана свой первый iPhone с браузером Safari, в одном устройстве было две ориентации. «Жесты» прокрутки, уменьшения и увеличения также подкрепили тот факт, что границы не имеют

значения. С того самого дня благодаря этому одному устройству Интернет изменился навсегда. Так что же сделали дизайнеры?

Мы нарисовали маленький прямоугольник. Шириной в 320 пикселей и высотой в 480 пикселей.

После того как Стив присел на диван и продемонстрировал первый iPad, мы сели и создали наш холст размерами 1024 на 768 пикселей. Мы придерживались постоянных размеров холста, потому что это было то, что мы знали и то, чего ожидали наши боссы и заказчики.

Вы слышали, как определяется невменяемость? Делать каждый раз одно то же, но ожидать при этом разных результатов. Вот так мы работали до сегодняшнего дня. Расширяли прямоугольники, сужали, потом снова расширяли. Все то время, пока они изменялись, наши размышления оставались одними и теми же.

Нож в перестрелке

Нам не следует слишком жестко относиться к себе. Программные средства, которые мы холили и лелеяли, сделали все что могли, чтобы сохранить жизнь постоянным размерам. Подумайте об этом. Что мы делаем прежде всего, когда начинаем разрабатывать новый дизайн в Fireworks и Photoshop?

Жмем Файл → Новый
или клавиши Ctrl+N.

Потом мы задаем документу постоянный размер холста.

Спросите себя, в чем причина того, что многие сайты имеют постоянную ширину и центрированы на экране. Не результат ли это видения наших клиентов, подписания ими фиксированных изображений?

Хотя поставщики ПО наподобие Adobe встраивают средства веб-разработки в свою продукцию, в них нет ничего, что может помочь нам создать отзывчивый дизайн. Они даже не могут помочь нам продемонстрировать состояние по наведению или другие события, которые могут показать как гибкий макет влияет на изменение положения текста и других элементов. И снова Джейсон Санта Мария^[136]:

«Каждый элемент веб-страницы имеет способность влиять на размещение других элементов. Мы должны уметь определять, какие действия предпринимать, когда это происходит. Кроме того, окно браузера – это изменчивый холст; дизайн приложений для настольного компьютера взаимодействует только с неизменными размерами холста, делая работу с гибким дизайном немногим меньше, чем догадка».

Наше текущее ПО, в частности Photoshop и Fireworks, просто неспособны к обработке требований адаптивного дизайна. Они как нож в перестрелке.

Этот неудобный факт

Долгое время самая трудная часть разработки сайта не имела отношения к тому факту, что люди, использовали наш дизайн на экранах разных размеров. Все это потому, что большую часть времени мы не занимались этим.

Вместо этого мы игнорировали этот неудобный факт и продолжали верить, что если у большинства людей экран достаточно велик для отображения нашего дизайна, то все будет в порядке.

Когда iPhone заставил нас осознать, что наша работа не всегда будет «[наилучшим образом] просматриваться на современном браузере в 1024 пикселей или больше», нашей первой реакцией было создать специфический дизайн страницы для iPhone в дополнение к тому, что есть для настольного компьютера. Это тут же увеличило время на разработку, обратную связь, корректировки и утверждение.

Чем больше появлялось смартфонов, электронных читалок и планшетов, тем прочнее у боссов и заказчиков входило в привычку просить специфические версии сайта или приложения для них ^[137].

- iPhone
- Android
- iPad
- Kindle Fire

Но разработка отдельных версий могла означать увеличение работы в три, четыре, а то и пять раз. В три, четыре, пять раз

увеличивался период утверждения. Встречи назначались три, четыре, пять раз!

Сегодня иметь дело с таким большим количеством характеристик, возможностей и размеров, создавать множество фиксированных макетов неуместно, неэкономично и непрактично, и не лучшее применение времени и талантов дизайнера. Также мы ничего не выгадываем технически, создавая несколько фиксированных макетов, когда мы хотим преобразовать эти изображения в код.

Как говорит Стефани Ригер в своей статье «„Проблема“ с Андроидом» («The ‘Trouble’ With Android»^[138]):

«По факту разработка под фиксированный экраный размер никогда не была хорошей идеей. [...] слишком много вариантов, даже среди «популярных» устройств».

Так, вместо того чтобы думать об индивидуальном дизайне для отдельных устройств, мы должны подумать, как советует Итан Маркот, о простом дизайне, имеющем много аспектов, которые находятся в непрерывном и гибком ряду^[139]:

«Вместо того чтобы адаптировать разрозненные макеты к каждому все более возрастающему числу устройств, мы можем принимать их как аспекты одного и того же опыта взаимодействия».

Я люблю такой метод объяснения дизайна в отзывчивом контексте – много аспектов в одном опыте использования, потому что он хорошо отражает, что такое «смена положения дел», как описывал в своем «Дао» Джон Элсоп.

Мне также нравится думать о дизайне как о реке, которая узкая в истоке, а потом, ближе к морю, становится широкой. Мы никогда не можем предсказать, на каком устройстве кто-то будет просматривать наш дизайн, поэтому должны сделать его гибким.

Отсюда возникают вопросы. Если бы мы не могли делать отдельные макеты с неизменяемой шириной, что бы мы делали? Как бы выстроили процесс? Как бы создавали дизайн для Интернета без границ?

«Эй, а я вот что сделал!»

Сегодня это в порядке вещей для нас создавать одно или несколько статических изображений с фиксированной шириной, чтобы представить сайт, который мы делаем. Есть несколько причин, почему мы так делаем.

Для дизайнеров

Photoshop и Fireworks – невероятные инструменты для творческих экспериментов и наши помощники в решении проблем и определении визуального направления проекта. Иногда даже сам процесс использования Photoshop и Fireworks помогает нам, порой случайно, достичь результата, которого трудно добиться другими средствами. В конце концов, результат – это нечто, на что мы можем ориентироваться и сказать нашим боссам и заказчикам: «Эй, а я вот что сделал!»

Проблема со статическими изображениям, я говорил об этом не раз, не просто в том, что они плохо передают реальность современного интерактивного сайта. Дело в том, что при неправильном использовании, они устанавливают неверные ожидания. Так как дизайнеры используют их, чтобы получить утверждение дизайнера, наши боссы и заказчики, вне сомнения, ожидают, что на выходе сайт будет выглядеть именно так, как на картинке. В конце концов, мы же для этого им его показываем.

Но не все браузеры могут воспроизводить дизайн одинаково. У всех разные возможности, которые статические изображения игнорируют. Так что в нашей глупости никто не виноват, когда мы тратим часы, влезая в тонкости наших HTML и CSS или применяя «заплатки» на JavaScript в тщетной попытке сделать так, чтобы наш сайт выглядел одинаково во всех браузерах, как на статичном макете.

Поэтому в будущем дизайнерам нужно согласиться с тем, что то, что они включают в статический макет, не может выглядеть одинаково для каждого человека и устройства. Они также должны научиться настраивать верные ожидания от статического макета, и объяснять боссам и заказчикам, что макеты Photoshop всего лишь показывают то, как сайт может выглядеть в некоторых браузерах на некоторых устройствах.

Для разработчиков

Статические макеты могут быть полезными инструментами для разработчиков, занимающихся переводом дизайна в код. Макеты являются проектом для разработчиков, а для боссов и заказчиков контрольной точкой для оценки того, насколько хорошо окончательный дизайн будет соответствовать тому видению, которое выражается в макете.

Но на сегодня разработчики должны согласиться с тем, что когда они получают статический макет, он представляет собой всего лишь руководство о том, как может выглядеть сайт. Разработчики должны научиться принимать решения по верстке макета, включая адаптацию к различным размерам экрана и устройств.

Для заказчиков

Беседы об этих статических макетах между дизайнерами, нашими боссами и заказчиками могут стать самой ценной частью креативного процесса. Если установить правильные ожидания – боссы и заказчики поймут значение того, что они видят. Потом статические макеты могут быть чрезвычайно полезны в описании целей, сбора комментариев и конструктивной критики.

Проблема в том, что правильные ожидания устанавливаются редко. И чем больше мы требуем от макетов с фиксированными размерами, тем меньше они становятся соответствующими и тем меньше они эффективны как средства для взаимодействия с дизайном. В частности, статические макеты с фиксированным размером не работают в отзывчивом контексте, потому что люди легко смогут перепутать дизайн с разметкой макета.

Дизайн – это не всегда разметка

Вам когда-нибудь говорил клиент, что ему не нравится дизайн? Если это случалось, вы копались немного глубже и выясняли, что это за детали, против которых они возражали. Спорю, что это была не гарнитура или способ обработки, который вы выбрали. Они прошли незамеченными. Дело не в цвете, который вы используете. И не в вашем «ювелирном» вычерчивании линий, рамок и затенений. Возможно, это была фраза: «Колонка должна быть слева, а не справа».

Короче говоря, ваш заказчик прокомментировал разметку, а критику навел на дизайн. Для боссов и заказчиков привычно включать разметку в тот же диалог, как и другие аспекты дизайна: типографика, цвет и текстура. Дизайнеры тоже так делают, потому что годами мы создавали и демонстрировали макеты с неизменяемой шириной, которые включали все эти вещи.

С этого момента нам надо понять, что дизайн и разметка должны быть разделены. Дизайн компонентов и их расположение на сетке горизонтально и вертикально – это два разных вопроса. Несмотря на то что расположение компонентов в разметке вне сомнения будет разным в зависимости от размеров экрана, дизайн тех самых компонентов почти точно, будет за ее пределами.

Сначала дизайн компонентов

Подумайте на мгновение о компонентах, которые вы разместили почти в каждом дизайне. Ваш список почти всегда будет включать следующие пункты:

- прямоугольники (блоки);
- таблицы данных и другие виды информационных панелей;
- изображения (контент и иконографика);
- элементы интерфейса (карусели и галереи изображений и пр.);
- навигация (несколько уровней);
- шрифт.

Обратите внимание: это важная часть! Дизайн в отсутствии разметки становится стилизацией этих компонентов. Как они выглядят сейчас – это то, что мы выражаем через дизайн. Более точно: впечатления и ощущения от этих компонентов – это то, над чем дизайнеры могут и должны работать в первую очередь, задолго до того как у них возникнет мысль о расположении. Мне нравится размышлять о подходе, каким я разрабатываю компоненты страницы сначала на атомарном, самом крошечном уровне.

Теперь я знаю, что идея разработки компонентов, подобная этой, вне контекста и без разметки может звучать странно, особенно если вы привыкли создавать страницы традиционным способом. Но, по правде говоря, мы долгое время рассматривали идеи, подобные этой.

Мудборды

Мудборды (mood boards), или карты настроений, долгое время оставались невероятным методом для собирания вдохновляющего материала. Хранили ли мы эту коллекцию на наших компьютерах в приложениях наподобие Evernote, использовали ли онлайн-сервисы типа Pinterest, или просто крепили их на паспарту или пробковую доску, все это комбинация различных элементов, выражающих настроение. Наши мудборды могут иметь современное, безвкусное или традиционное восприятие, но как бы мы их ни называли – образом, настроением или индивидуальностью, – необходимо помнить о важном: мы описываем стиль. Конечно, мы не должны начинать процесс разработки с бумаги, ножниц и клея. Мы можем, если захотим, и не пачкать наши руки.

Теперь, когда мы делаем отзывчивый дизайн, мы можем начать с моделирования компонентов как индивидуальных, но связанных частей дизайна.

Adobe Photoshop и Fireworks могут оказаться и не лучшими средствами для веб-разработки, но все же будут ценными для создания любого вида компонентов. Чтобы помочь вам избавиться от мыслей о разметке на данном этапе, попробуйте начать с холстов Photoshop и Fireworks, размеры которых составляют 10,000 на 10,000 пикселей и больше.

На этой дизайн-схеме группируйте элементы в заголовки, стили основного текста, формируйте элементы, кнопки, сообщения об ошибках и т. д. Если необходимо, перемещайте компоненты по схеме, чтобы посмотреть, как они будут выглядеть в соотношениях друг с другом.

Трент Уолтон писал^[140]:

«Веб-дизайнерам нужно выходить за пределы разметки и представлять, как ее элементы будут переставляться и заключаться в различную ширину при сохранении формы и иерархии. Медиазапросы могут использоваться не только для того, чтобы исправить нарушенное расположение: при правильном планировании мы можем начать выстраивать контент пропорционально размеру экрана, предоставляя самый лучший опыт взаимодействия, который возможен, на любой ширине».

Отделение дизайна от разметки, которого мы добились, когда занимались разработкой на атомарном уровне важно, потому что это помогает всем – дизайнеру и его боссам или заказчику – сконцентрироваться на деталях дизайна, при этом не устанавливая ожиданий, как элементы в конечном счете будут располагаться на экранах разных размеров или устройствах.

Последние несколько месяцев я использовал схемы дизайна для демонстрации моим клиентам. Я считал это чрезвычайно эффективным, несмотря на то что требуется несколько минут, чтобы объяснить суть клиенту, который до этого ничего не знал о таком методе работы раньше. Мы можем спросить у любого, кто заинтересован: «Это то настроение сайта, к которому мы стремились?»

Схемы дизайна – это потрясающие средства для прояснения желаний клиента на раннем этапе процесса разработки. Вместо того чтобы поощрять смутные высказывания типа «Мне не нравится дизайн», этот метод помогает нам быть определенными, когда мы обсуждаем дизайн.

- Насколько чистым или минималистичным должен быть дизайн?
- Как мы будем использовать цвет для передачи действий и смысла?
- Сколько будет шагов в тоне и контрасте?
- Как будут использоваться гарнитуры для различия типов контента?
- Сколько уровней будет в типографической иерархии?

Проектирование компонентов таким способом может помочь нам найти ответы на эти вопросы и предоставить всем заинтересованным лицам альтернативы и возможности изменить свое мнение до вложения значительных инвестиций в дизайн или разработку.

Использование схем дизайна также позволяет нам продолжить циклы дизайна, даже в то время, когда развиваются другие аспекты сайта. Это помогает нам прервать отживший рабочий процесс, который мы переняли из допечатной подготовки, и печатного дизайна для Интернета.

Одна схема вам известна:

***Планирование → Утверждение → Дизайн →
Утверждение → Разработка***

При более гибкой работе дизайн может иметь место на различных этапах, иногда даже до того как будет завершено планирование или после того как начнется разработка. Таким образом, дизайн становится глубже интегрированным и продолжается в циклах разработки.

Мозайка стилей

Абстракция дизайна во впечатлениях и ощущениях от элементов – это нечто такое, о чем думает дизайнер Саманта Уоррен. Она называет эту технику мозаикой стилей.

Саманта поясняет^[141]:

«Мозаика стилей – это визуальный трей из элементов цветовой палитры, текстурных узоров и цветовых вариантов, которые поддерживают цели клиентов. У меня есть шаблон в Photoshop со специально замаскированными участками, где [...] я представляю образцы стилей клавиш, решения навигации и типографических возможностей».

Саманта считает эту технику эффективным средством коммуникации, так же как и для дизайна на уровне компонентов^[142].

BOOTSTRAP

^[143]

Для других, элементы страницы могут быть компоновочными блоками для будущего дизайна. Возьмем Марка Отто и Джейкоба Торнтон – тандем дизайнеров-разработчиков, которые работают в Twitter. Эти двое стоят за Bootstrap^[144]. Интернет может быть безграничным, но контент у него есть всегда. Мы выражаем этот контент через соответствующие HTML-элементы:

- заголовки и их уровни от 1 до 6;
- параграфы, цитаты и элементы на уровне текста;
- списки: определения, нумерованный и ненумерованные;
- медиа: диаграммы, заголовки и изображения;
- таблицы;
- формы: кнопки, элементы и метки.

Так же как у Twitter есть своя сетка, так и Bootstrap включает в себя дизайн по умолчанию для «типографики, форм, клавиш, меток, сеток, навигации и остального». Bootstrap интересен по ряду причин. Не только потому, что это мощный шаблон для разработки, но и потому что в нем есть полный список HTML-элементов, вместе с несколькими стилями по умолчанию, что является идеальной отправной точкой для нового дизайна.



Рисунок 11.1. Сначала вы должны узнать, почему такие средства разработчика, как Bootstrap важны для отзывчивого рабочего процесса. Возможно, вы удивитесь, когда узнаете, что Bootstrap – это великолепная исходная точка для разработки элементов страницы

Вы на Дрибле?

Dribbble^[145] – это сайт, где дизайнеры обмениваются маленькими скриншотами (максимум 400 на 300 пикселей) дизайнов, над которыми они работают.

Рисунок 11.2. Dribbble показывает элементы в нейтральном окружении

Совершите путешествие по страницам Dribbble, и вы найдете много примеров дизайна элементов и деталей, над которыми потеют дизайнеры. Вы найдете дизайн навигации и решения для встраивания изображений. Если формы – это ваше, то вы найдете стили для каждого элемента формы, плюс стили клавиш для любого состояния. О типографике здесь тоже не забыли. Для вас здесь сколько угодно примеров гарнитур и решений.

Шоты сайта Dribbble – это прекрасные примеры разработанных элементов без разметки. Они показывают, как мы можем разработать дизайн и потом продемонстрировать нашим боссам и заказчикам направление дизайна в формате, который не путает стиль с расположением.

Руководства по стилю

Когда вы работаете над проектом, в котором участвует группа людей, подробное и хорошо написанное руководство по стилю может стать жизненно необходимым для поддержки целостности дизайна. При организации пользуйтесь руководством по стилю для разных целей – издатели используют его для орфографии и правил пунктуации. А специалисты по печати и рекламе используют их для задания стиля в письменных публикациях. Лучшие руководства по стилю четко устанавливает стиль сайта.

Рисунок 11.3. Global Experience Language (GEL) – 18 документов полного визуального стиля для веба компании BBC

Один из лучших приведенных примеров полного комплекта руководств по стилю «пришел» из BBC. Его руководство Global Experience Language (GEL)^[146] документирует полный визуальный язык BBC для использования в Интернете, с подробной информацией

о типографике, иконографии и дизайне элементов интерфейса, таких как наложение, «аккордеон» и карусель изображений.

Е-mail-эксперт MailChimp^[147] создал свою собственную библиотеку образцов пользовательского интерфейса, чтобы документировать то, как задается стиль таким элементам, как клавиши, формы, таблицы и табы (вкладки).

Возможно, вы захотите знать, почему руководства по стилю так важны в разговоре об отзывчивом дизайне, особенно если руководства по стилю создаются для документирования принципов дизайна, как правило, после того как сайт уже разработан.

Вещи, подобные Bootstrap в Twitter, GEL от BBC и библиотеке образцов пользовательского интерфейса от MailChimp, – великолепные примеры дизайна элементов. Они детально описывают, как должен выглядеть каждый элемент, появляющийся в дизайне. И делают они это с небольшой ссылкой на разметку или вовсе без нее.

Самое детальное руководство по стилю не просто документирует дизайн сайта – оно само его дизайн. Поэтому зачем ждать окончания процесса до того, как сделать руководство по стилю? Вместо этого рассматривайте эти руководства как еще один пример дизайна на уровне элементов.

Используйте их для документирования, а потом включайте в дизайн каждый элемент, который появится на вашей странице. Вы потом сможете использовать их для демонстрации направления вашего дизайна, без использования конкретной разметки, и использовать их как средства, которые помогут вам получить утверждение без ожиданий от разметки.

Рисунок 11.4. MailChimp создал свою собственную библиотеку образцов пользовательского интерфейса, чтобы документировать, как задается стиль элементам пользовательского интерфейса

Сохранение статических макетов, когда это уместно

Я обнаружил, что если мы начинаем с дизайна компонентов, то это приносит реальные дивиденды моим заказчикам. Я видел, как проекты продвигались более плавно, быстрее и с наименьшим

недопониманием. Но разработка элементов в начале – это такой процесс, который люди считают сложным для понимания. Поскольку многие работают в организациях, где статический макет со строгой разметкой и определенным положением элементов все еще котируется среди дизайнеров, разработчиков и их боссов или клиентов, не каждый может оставить статические макеты.

Дизайнеры должны видеть элементы в отношениях друг к другу. Вот почему важны размеры и пропорции, и ничего не остается, как смотреть, как они подходят друг другу. В этой ситуации, может быть, очень полезно сделать статический макет с одной разметкой. Так я делаю почти в каждом проекте.

Мы также не можем убежать от того факта, что годами наши боссы и заказчики были так проникнуты ожиданием композиции полной страницы, что некоторые из них могли упасть духом при виде листа, на котором был просто дизайн элементов и никакого взаимного расположения.

Я столкнулся с этим сразу же, как только начал заниматься разработкой дизайна от элементов. Иногда требуется тщательный подход к решению и уверенность в том, что результат будет лучше после принятия этого процесса. Вот почему я вынужден иногда демонстрировать образец дизайна клиенту как полностраничный статический макет.

Если наш рабочий процесс вынуждает нас придерживаться полностраничных статических макетов, пусть и немного, не означает ли это, что мы должны отказаться от плюсов разработки дизайна от элементов? Нет. Если мы установим правильные ожидания. Особенно наши боссы и клиенты должны понять, что статические макеты, которые мы можем показать им, – просто расширение атмосферы дизайна и всего лишь возможное его выражение.

Атмосфера дизайна

Если мы должны продолжить использование статических полностраничных макетов, мы все еще можем выделить дизайн элементов и использовать его на экранах разных размеров и устройствах.

Давайте разобьем любой дизайн на его компоненты:

- Типографика: гарнитура, работа с текстом и пустым пространством;
- Цвет: для пробуждения настроения и операций выделения;
- Текстура: декоративные аспекты, в том числе рамки, затенение и фигуры;
- Разметка: горизонтальное и вертикальное расположение элементов по сетке.

Для начала давайте отделим разметку от остальных компонентов. То, что остается, – цвет, текстура и типографика – я люблю называть атмосферой дизайна, потому что я представляю, как будто все побывало на концерте или на бейсбольном матче, атмосферу которого можно назвать накаленной. Держу пари, что все мы были на одной-двух вечеринках с вялой атмосферой.

Вероятно, вам не посчастливилось ощущать борьбу двух людей под действием атмосферы в комнате.

В дизайне атмосфера описывает чувства, которые вызываются у нас цветом, текстурой и типографикой. Возможно, вы уже думаете об атмосфере в различных терминах. Вы можете называть ее «ощущением», «настроением» или даже «визуальным определением». Какое бы слово вы ни выбрали, атмосфера дизайна не будет зависеть от разметки: расположения и визуального размещения. Она будет видимая или ощутимая в любом размере экрана или на любом устройстве.

Пока мы или наша организация движемся к новому и улучшенному процессу дизайна, важно понять, как извлечь атмосферу из статического макета или из уже существующего сайта. Это умение, которому и разработчики, и дизайнеры должны научиться как можно скорее, потому что знание того, как увидеть, извлечь и перегруппировать элементы при изменении разметки, – это ключ к созданию хорошего отзывчивого веб-дизайна.

Извлечение атмосферы

Давайте рассмотрим подробно небольшую выборку сайтов. Мы разделим их атмосферу и разметку и выявим, чем отличается атмосфера каждого из них.

Food Sense

Начнем с Food Sense^[148], великолепно разработанного, отзывчивого сайта, в котором говорится о растительной пище. Говоря о статических элементах, мы могли бы сначала рассмотреть его двухколонную разметку на компьютере. Но мы ищем нечто большее. Нам нужна атмосфера сайта. Мы разбиваем ее на цвет, текстуру и типографику.

Цвет. Внутри атмосферы любого дизайна цвет вызывает настроение и эмоции, но мы также можем использовать цвет в словах для призыва к действию, сообщений и других точек соприкосновения. Food Sense использует нежные оттенки зеленого, черного и белого для связи элементов дизайна.

Рисунок 11.5. Сайт Food Sense

Текстура. Когда я говорю о текстуре, я не имею ввиду физическую текстуру, хотя ничто не помешает вам использовать искусственную кожу и рваную бумагу, если вы хотите этого. Я говорю о том, что текстура имеет отношение к декоративным аспектам дизайна, включая линии (границы, делители и разделители), затенение и форму.

Как пример, мы можем описать работу с контурами. Какие они: простые, двойные или пунктирные? С постоянной шириной? Существует ли иерархия ширины разделительной линии между второстепенными и основными разделами? Если да, то, как выстраивается эта иерархия?

Как затеняется контент? Блоки имеют однородный цвет фона? Они градуированные или с паттернами? Углы блоков закругленные или квадратные? Все это относится к текстуре.

Food Sense имеет плоское цветное затенение для блоков и простые серые пунктирные или сплошные линии, которые разделяют область контента.

Его кнопки и активные ссылки с тонкими серыми подчеркиваниями и тенями. Изображения в контенте даны в белой рамке с серой границей. Представленные в виде рисованных

набросков, пиктограммы усиливают атмосферу дизайна сайта Food Sense.

Типографика, возможно, представляет собой самый простой элемент атмосферы, который можно извлечь, но она значит больше, чем просто выбор гарнитуры. Типографическая атмосфера включает в себя межстрочный интервал (высота строки), трекинг (межбуквенное расстояние), а также свободное пространство разделяющее элементы.

Элиот Джей Стокс пишет^[149]:

«Я чувствую, что метод разработки дизайна, где сначала внимание уделяется типографике, уводит нас на шаг вперед от ненужных отвлечений на дизайн ради дизайна и приближает нас к тому, чтобы стать настоящими типографами».

Дизайнеры сайта Food Sense выбрали строчный шрифт с засечками FF Tisa Web Pro для заголовков, основного текста и навигации. Этот дизайн допускает много свободного пространства, которое создает открытую и легкую атмосферу.

Джейми Оливер (Jamie Oliver)

Изучая атмосферу дизайна, наш взгляд быстро настраивается на разницу между ними. С той же тематикой – о еде – сайт Джейми Оливера^[150] выглядит иначе, чем сайт Food Sense. Теперь ваша очередь извлекать атмосферу из сайта Джейми Оливера и анализировать цвет, текстуру и типографику.

Рисунок 11.6. Сайт Джейми Оливера

Цвет. Как видно из логотипа, на сайте используется основное сочетание коричневого или голубого. Вы можете выстроить систему цветов для ссылок? Как используется цвет в навигации?

Текстура. Как на сайте используются фоновые иллюстрации для создания блоков контента? Вы заметили паттерны? Например, как заголовки отделены от контента под ними? Какой стиль у кнопок? Их углы закругленные или квадратные? Есть ли последовательность?

Типографика. На сайте используются обычные предустановленные шрифты. Есть ли легко распознаваемая типографическая иерархия заголовков? Существует ли разница между видами основного текста? Если да, то в чем? Сайт воспринимается как открытый или сжатый? Как поля и отступы влияют на чувство пространства внутри дизайна?

BBC Food

И, наконец, оставаясь в теме о еде, рассмотрим сайт BBC^[151], где мы видим блоки по рекомендациям BBC GEL. И опять мы не ищем разметку. Вместо этого мы ищем атмосферу и то, что отличает этот сайт от других с подобным контентом или той же целью.

BBC Food использует цвет, чтобы пояснить, какие элементы являются ссылками. У него приглушенная палитра, возможно для того, чтобы сделать акцент на цвете фотографий. Там, где BBC использует иконки из своего комплекта иконок GEL, они плоские и одноцветные – такие же, как фон блоков контекста.

Края блоков сайта BBC остаются квадратными и острыми. Где контент перекрывает изображения, дизайнеры BBC Food используют полупрозрачные черно-белые прямоугольники. Сами изображения по большому счету необработанные и у них нет никаких границ.

Рисунок 11.7. Сайт BBC Food

Сайт BBC использует полужирный текст «для создания сильной иерархии и драматизма на сайте»^[152]. Шрифт сайта BBC по умолчанию – Arial, хотя, если копнуть глубже, вы наткнетесь на Helvetica Neue для Mac. Эти шрифты используются для заголовков и основного текста.

Это упражнение, которое будет работать на всех сайтах, и особенно в группе дизайнеров и разработчиков. Оно стало ключевым моментом моих мастер-классов по отзывчивому веб-дизайну и доказало свою высокую эффективность.

Стать невероятно гибким

Учимся ли мы сначала разрабатывать компоненты или извлекать атмосферу из статических макетов, нам больше необязательно разделять макеты Photoshop или Fireworks и разметку для каждой страницы, не говоря уже о каждом устройстве. На самом деле я слишком забегаю вперед и скажу, что дни разработки множественных статических макетов сочтены. Мы, наши боссы и заказчики понимаем свойственные им неточность и неэффективность. Это значит, нравится нам или нет, мы должны находить новые пути к отзывчивому дизайну.

Знаю по своему опыту за последний год, углубляясь в отзывчивый дизайн, что найти новые пути разработки и рассказать про них не так-то просто. Хотя я продвигаю идею того, что сайты не должны выглядеть одинаково на каждом браузере годами, так или иначе было сложно отпустить контроль над разметкой. Я понял, что моя работа как дизайнера изменилась коренным образом. Я больше не мог ожидать привычного уровня контроля над разметкой на экранах разных размеров.

И все же в отзывчивом дизайне есть достаточно позитивных моментов для дизайнеров, особенно в гибком технологическом процессе. Теперь я могу сосредоточиться на изматывающих деталях дизайна не только в начале, но и в ходе всего процесса разработки. Вместо того чтобы подробно описывать, как должна выглядеть каждая разметка для каждого экранного размера и устройства, я могу объяснить свои мысли в дизайне, а потом дать другим возможность интерпретировать его и адаптировать при разработке. Этот технологический процесс работает сказочно для меня и моих клиентов.

Для того чтобы все шло ровно, разработчики, с которыми я работаю, должны тоже узнать о дизайне и о том, как из него извлекается атмосфера. Они должны быть ответственны за принятие своих решений по дизайну. Некоторым это может показаться пугающим изменением, но разработчики, с кем я сотрудничал, рассматривали это больше как возможность развития понимания и навыков в новой области.

В начале раздела я говорил о том, что отзывчивый веб-дизайн задает больше вопросов, чем дает ответов, потому что он изменяет рабочие взаимоотношения и взаимодействия между всеми, кто вовлечен в любой процесс. Я знаю, что эти проблемы не всегда будут

легко решить. Я сам видел, как увлеченно дизайнеры и разработчики решают эти проблемы и какими чуткими могут быть наши боссы и заказчики, когда они используют преимущества, которые дает отзывчивый веб-дизайн.

Работая вместе, мы можем сделать Интернет отзывчивым, каким он всегда и задумывался.

Об авторе

* * *

Энди Кларк получил много прозвищ, с тех пор как 10 лет назад начал заниматься веб-дизайном. Его самолюбие тешат такие как «вестник CSS», «пророк индустрии» и «вдохновляющий». Но больше всего он гордится тем, как однажды назвал его Джеффри Зельдман («крестный отец» веб-стандартов). Он назвал его «трижды талантливым ублюдком».

Энди управляет Stuff and Nonsense (stuffandnonsense.co.uk), маленькой дизайн-студией, где он работает с такими клиентами, как ISO, STV и правительством Великобритании. Он выступает на конференциях по веб-дизайну по всему миру. Является автором манифеста «Transcending CSS» и принятой на ура книгой «Hardboiled Web Design».

Сноски

1

Facebook Changes Confuse Users, as a Major Overhaul Looms // The Washington Post, smashed.by/fbc

2

Nielsen, Jacob. Fresh vs. Familiar: How Aggressively to Redesign, smashed.by/nielsen

3

Active Server Pages – технология создания веб-приложений корпорации «Майкрософт». – Примеч. ред.

4

NET Framework – программная платформа. Основой платформы является исполняющая среда CLR, способная выполнять как обычные программы, так и серверные веб-приложения. – Примеч. ред.

5

Cascading Style Sheets – каскадные таблицы стилей – формальный язык описания внешнего вида документа, написанного с использованием языка разметки (например, HTML). – Примеч. ред.

6

Moll, Cameron. Good Designers Redesign, Great Designers Realign, smashed.by/realign

7

Некогда популярная социальная сеть. – Примеч. ред.

8

Вы можете узнать больше о мобильной платформе в редизайне в разделе Арэла Балкана в этой книге.

9

Content management system – система, позволяющая изменять контент на сайте, создавать новые страницы с помощью так называемой «админки» сайта.

10

Andrew, Rachel. Your CMS as Curator of Your Design and Content / smashed.by/cms-curator.

11

Конверсия – один из важнейших параметров эффективности сайта. Обычно конверсией называется совершение целевого действия. Показатель конверсии считается как отношение числа пользователей, совершивших целевое действие к общему числу посетителей сайта или целевой страницы. – Примеч. науч. ред.

12

Chopra, Paras. The Ultimate Guide to A/B Testing / smashed.by/abtesting.

13

В России PayPal не распространена, и лучше использовать другие платежные системы. – Примеч. ред.

14

В России лучше воспользоваться услугами процессинговых компаний, действующих по законодательству РФ. – Примеч. ред.

15

Payment Card Industry Data Security Standard – стандарт безопасности данных индустрии платежных карт. – Примеч. ред.

16

Набор функций, предоставляемых сервисом для использования во внешних программных продуктах. Используется программистами для написания приложений для взаимодействия с исходным сервисом. Примером такого API могут служить, к примеру, Яндекс. Карты, которые можно встроить на свой сайт. – Примеч. ред.

17

Electronic point of sale – оборудование для калькуляции и учета транзакций. – Примеч. ред.

18

Google Webmaster Central Blog. Using Site Speed in Web Search Ranking / smashed.by/googlespeed.

19

Beanstalk, «An Introduction to Version Control» / smashed.by/version-control.

20

smashed.by/wcai.

21

Тег `<!DOCTYPE>` служит для указания типа текущего документа – DTD (document type definition, описание типа документа). Необходимо, чтобы браузер понимал, как следует интерпретировать страницу, поскольку HTML существует в нескольких версиях. Чтобы браузер «не путался» и понимал, согласно какому стандарту отображать веб-страницу, и нужно в первой строке кода задавать `<!DOCTYPE>`. – Примеч. науч. ред.

22

Общие части страниц (группы элементов), которые кочуют из страницы в страницу называются project assets.

23

smashed.by/amzspeed.

24

smashed.by/yepnope.

25

Polyfill – загружаемый скрипт, который дополняет функционал браузера. Например, старые версии браузеров не поддерживают весь функционал HTML5, и такой скрипт позволяет полноценно работать в старых браузерах недоступным по умолчанию функциям. – *Примеч. науч. ред.*

26

smashed.by/configs.

27

smashed.by/uglify.

28

smashed.by/steve.

29

smashed.by/modernizr.

30

smashed.by/normalize.

31

smashed.by/nicolas.

32

smashed.by/jon.

33

smashed.by/h5doc.

34

smashed.by/vldnu.

35

smashed.by/modernizr.

36

² В начале 2013 года анонсирован переход Оперы также на движок Webkit. – *Примеч. науч. ред.*

37

С полным перечнем префиксов можно ознакомиться по ссылке smashed.by/vndrprfx.

38

Если у вас нет времени разбираться, какой префикс нужно использовать для каждого свойства, вы можете воспользоваться постпроцессором, например `prefix-free` (smashed.by/prfxfree), который позволяет вам писать CSS без префиксов и сам добавляет их через JavaScript.

39

smashed.by/ciuse.

40

smashed.by/html5pl.

41

smashed.by/mozdevnet.

42

Примеры смотрите в smashed.by/fntexmpl.

43

smashed.by/fntsqr.

44

smashed.by/exjlb.

45

Свойство `column-span` еще не поддерживается широко, поэтому ваша верстка может «слететь», если будете сильно полагаться на него.

46

Вы можете попробовать сами, посетив smashed.by/mltclmn.

47

smashed.by/multbgs.

48

smashed.by/bgexample1.

49

smashed.by/bgclipping.

50

smashed.by/brdimages.

51

Попробуйте это по ссылке smashed.by/wikitarget.

52

smashed.by/slctvzr.

53

smashed.by/mixunits.

54

smashed.by/percproblem.

55

smashed.by/box-sizing.

56

smashed.by/cbcbz.

57

smashed.by/caniuse.

58

smashed.by/easing.

59

Испробуйте этот пример в действии: smashed.by/transition.

60

smashed.by/sandbox.

61

Проверьте этот пример в действии: smashed.by/todolist.

62

Попробуйте этот пример на практике: smashed.by/cafevintage.

63

Проверьте этот пример в действии: smashed.by/cafevintage

64

ZURB. Why Burying Sign Up Buttons Helps Get More Sign Ups / smashed.by/signupbuttons.

65

VWOblog. Регистрация возросла до 60 % после фактического перемещения ее формы / smashed.by/abtesting.

66

Вроблевский, Люк. Плавное вхождение в контакт повышает количество регистраций в Twitter на 29 % / smashed.by/twsignups.

67

Вроблевский, Люк. Тестирование форм-аккордеонов /
smashed.by/testing-forms.

68

The Baymard Institute. Contextual Words Like ‘Continue’ are Usability
Poison/ smashed.by/poison.

69

Apple Insider. Inside Mac OS X 10.7: Lion / smashed.by/appinside.

70

Холст, Кристиан. Redesigning The Country Selector/ smashed.by/selector.

71

Porter, Joshua. Reward The Passionates / smashed.by/dropboxads.

72

Вращающийся каталог с карточками, используемыми для хранения
контактной бизнес-информации (ROLLing + inDEX). – Примеч. науч.
ред.

73

Сейчас уже выпущены ноутбуки с высокой плотностью пикселей,
например ноутбуки Apple с экраном Retina. – Примеч. науч. ред.

74

На момент перевода книги уже выпущен ноутбук Apple Macbook
Pro 15” с экраном Retina и разрешением 2880 на 1800, который
позволяет уместить целиком разрешение экрана iPad, правда, лишь в
альбомном режиме. – Примеч. науч. ред.

75

Action– скрипт с записанной последовательностью действий для автоматизации рутинных операций. – Примеч. науч. ред.

76

Вы можете воспользоваться моим экшеном для Copy Merged в smashed.by/copyaction.

77

Wikipedia: Borg (Star Trek) / smashed.by/borg // [http://ru.wikipedia.org/wiki/Борг_\(Звёздный_путь\)](http://ru.wikipedia.org/wiki/Борг_(Звёздный_путь)).

78

Unger, Russ and Chandler, Carolyn. A Project Guide to UX Design: For user experience designers in the field or in the making. New Riders Press, 1st edition, 2009. Выпущена издательством «Символ-Плюс» в 2010 под названием «UX-дизайн. Практическое руководство по проектированию опыта взаимодействия».

79

Cooper, Alan. The Inmates Are Running the Asylum. Sams – Pearson Education, 1st edition, 2004. Издана в России под названием «Психбольница в руках пациентов». – Примеч. ред.

80

Компании, работающие в сфере интернет-консалтинга, маркетинга и диджитал-стратегий.

81

smashed.by/carbon.

82

smashed.by/jojo.

83

[ge.com](#).

84

Pinker, Steven. How the Mind Works. W. W. Norton & Company, Reissue edition, 2009.

85

[smashed.by/hipmunk](#).

86

Medina, John. Brain Rules: 12 Principles for Surviving and Thriving at Work, Home, and School. Pear Press, Reprint edition, 2009.

87

[smashed.by/wufoo](#).

88

Как отмечает в своем разделе Дмитрий Фадеев, сайт Wufoo выходит за рамки обычного, делая свой образ поистине выдающимся, когда сотрудники рассылают клиентам написанные от руки благодарственные открытки.

89

[smashed.by/monkey](#).

90

Wilson, Fred. Minimum Viable Personality / [smashed.by/mvp](#).

91

Wroblewski, Luke. Warm Gun: Designing for Emotion / smashed.by/warmgun.

92

Why Does Emotional Design Work on the Web. Forbes / smashed.by/forbes.

93

Загрузите образцы для досье user personas из smashed.by/persona.

94

Чтобы узнать больше о схеме образа и ее исследовании, см.: Van Gorp, Trevor. Emotional Design With A.C.T.: Part 1 // Boxes and Arrows / smashed.by/emodesign.

95

Anderson, Stephen P. Mental Notes / smashed.by/notes.

96

smashed.by/persona.

97

Детская кукла Элмо, персонаж программы «Улица Сезам». – Примеч. ред.

98

Загрузите эти примеры из smashed.by/mailchimp.

99

«Пасхальное яйцо» (англ. Easter Egg) – разновидность секрета, оставляемого в игре, фильме или программном обеспечении

создателями. «Пасхальные яйца» играют роль своеобразных шуток для внимательных игроков и зрителей.

100

smashed.by/voicetone.

101

Дополнительная информацию на smashed.by/bonus.

102

В оригинале используется термин *native* – термин, вошедший в русскоязычную практику, обозначающий «родные» приложения для мобильных устройств, в противовес веб-версиям этих же сервисов. Например, сервисом Twitter можно пользоваться через обычный сайт, через мобильную версию и через приложение. Приложения зачастую обеспечивают более высокие производительность и возможности. – Примеч. науч. ред.

103

Progressive enhancement – методика проектирования, когда сначала сайт проектируется для устройств с малыми возможностями (например, телефонов с небольшими экранами), а затем функционал прогрессивно улучшается для других устройств (планшетных и настольных компьютеров). – Примеч. науч. ред.

104

Узнать больше о методах такой работы вы сможете, изучив материалы по живым методологиям и развитию продукта, нацеленного на пользователя. Постоянное тестирование и добавление по его результатам элементов дизайна – это то, что я называю динамичное развитие продукта, нацеленного на пользователя («*user-centered agile product development*»): smashed.by/cfe.

105

Эти нормы обычно выражаются в рекомендациях по интерфейсу к платформам, а именно рекомендации по интерфейсу с пользователем для платформ Mac, iPhone, iPad и Android. Android можно выделить, так как это не единичная платформа, имеющая много разновидностей, каждая настроена под устройство производителя и мобильные носители. Вот почему для Google очень сложно проконтролировать опыт пользователя на платформе Android. Хороший пользовательский опыт – это контролирующая функция. А Google очень слабо контролирует опыт пользователей телефонами с платформой Android.

106

smashed.by/apple.

107

smashed.by/fapp.

108

Смотрите, как Тантек Челик требует «Руководство интерфейса человека для сети» (smashed.by/wehuin), и читайте сообщение Джо Хьюитас призывом обращать больше внимания на веб-технологии (smashed.by/owned).

109

Как утверждает Эндрю Люис, «если вы не платите за это, вы – не клиент. Вы – продукт на продажу» [/smashed.by/discount](http://smashed.by/discount).

110

Если вы действительно рассуждаете и используете Git для управления версиями разработки (github.com), вы можете автоматически выгружать последнюю актуальную версию на сервер.

111

Во всяком случае здесь URL не заблокируется автократичным режимом.

112

Изучите «манифест одной версии» и характер меньшей версии сети в моем отзыве в Netmagazines с названием «Мой веб-сайт будет поддерживать только новейшие версии браузера» («My Websites Will Only Support the Latest Browser Versions») /smashed.by/netsu

113

Topolsky, Joshua. A modest proposal: the Continuous Client / smashed.by/client

114

Graceful degradation – изящная деградация – прием, обратный прогрессивному улучшению. В этом случае продукт разрабатывается, например, для браузеров с полной поддержкой современных функций, а для устаревших версий, таких как, например, Internet Explorer 6, функционал уменьшается – убирается поддержка «украшательств» и менее важных опций, оставляя базовый функционал. – Примеч. науч. ред.

115

PhoneGap не навязывает использование какой-то определенной структуры веб-интерфейса пользователя (например, jQuery Mobile). Все это позволяет вам представить веб-приложение «родным». Однако, независимо от того, какую структуру интерфейса пользователя вы используете (или даже если вы решили вообще не использовать структуру веб-интерфейса пользователя), воспроизводимые компоненты будут компонентами веб-интерфейса пользователя, а не компонентами «родного» интерфейса пользователя той платформы, на которой запускается ваше приложение.

116

То же самое можно сказать о веб-приложениях, которые добавляются на экраны телефонов. К тому же веб-приложение, о котором идет речь, выглядит и запускается как «родное» приложение, но ведет себя иначе. У веб-приложения, работающего в браузере, нет таких недостатков, так как у него нет цели имитировать «родное» приложение операционной системы. Это «родное» веб-приложение, а они – так же как и «родные» приложения на других платформах – имеют свою культуру, условные обозначения, язык, нормы и ожидания (даже если они могут быть не так ярко выражены, как на некоторых других «родных» платформах).

117

¹ Не использует бинарные файлы.

² Если запускается в браузере.

³ Если запускается по ссылке с домашнего экрана.

118

Не путайте Titanium Mobile с Titanium Desktop. Последний функционирует просто как PhoneGap. Appcelerator недавно открыл доступ к исходникам Titanium Desktop и сфокусировался на улучшении Titanium Mobile: smashed.by/tita.

119

Более точно известен как URI, хотя между двумя обозначениями существует техническая разница. Разработчики W3C-стандартов вскипели бы, если бы увидели, как здесь пренебрегают ею.

120

В настоящее время The CSS Flexible Box Layout Module (smashed.by/w3flexbox) является одной из нескольких мощных разметок в разработках. До сих пор ее внедрение в браузеры ограничено. Я написал учебное пособие, которое демонстрирует спецификацию

Flexbox с использованием трех маленьких прямоугольников на одной странице: smashed.by/learnflexbox.

121

Потрясающая статья Итан Маркотта «Отзывчивый веб-дизайн» (smashed.by/rwdala) больше чем о медиазапросах, но медиазапросы рассматриваются в полезном контексте, и я очень рекомендую прочитать статью всем, кто не знаком с ними.

122

Люк много говорил и писал о методе «Сначала мобильные»: smashed.by/mfirst.

123

Специальная версия браузера Chrome, в которой тестируются самые новые функции. – Примеч. науч. ред.

124

Труд Стефана Фью «BulletGraphDesignSpecification» воодушевил меня на создание графика точек прерывания: smashed.by/bullgraph (PDF). Для более подробной информации смотрите статью в Википедии: smashed.by/bullgraphwiki. Графики точек прерывания и буллит-графики отличаются, тем не менее я еще пока не написал спецификацию для графика точек прерывания.

125

Я не могу не подчеркнуть то, что тестирование важно для актуальных браузеров на актуальных устройствах. По возможности, сделайте скриншот на устройствах, потому что простое изменение размеров окна браузера настольного компьютера редко хорошо эмулирует, что действительно происходит на других устройствах. Эмуляторы тоже не всегда рассказывают всю правду, но они все же лучше, чем измененный в размерах браузер настольного компьютера.

126

Препроцессоры CSS позволяют писать ясный CSS с использованием программных конструкций (например, использовать переменные) вместо статических правил. Предназначены для увеличения уровня абстракции кода CSS и упрощения файлов каскадных таблиц стилей. – Примеч. науч. ред.

127

Хьюм, Энди. Responsive by Default («Отзывчивый по умолчанию») / smashed.by/respdef.

128

Кларк, Энди. I Don't Care About Responsive Web Design («Чувствительный веб-дизайн? Мне это не интересно») / smashed.by/respc.

129

smashed.by/anmag.

130

Allsopp, John. A Dao of Web Design / smashed.by/dao.

131

Dreamweaver FAQ: Using Frames To Align Page Content / smashed.by/tut.

132

Santa Maria, Jason. A List Apart Redesign / smashed.by/mari.

133

Clarke, Andy. A List (taken) Apart / smashed.by/andy.

134

Hicks, John. Is 1024 OK? / smashed.by/hicks

135

Keith, Jeremy. A List Too Far Apart? / smashed.by/runaway

136

Санта Мария, Джейсон. A Real Web Design Application / smashed.by/rwdapp

137

См. седьмой раздел для обзора общепринятых разрешений экрана смартфона и пиксельных разрешений.

138

Rieger, Stephanie. The «trouble» with Android / smashed.by/trouble

139

Marcotte, Ethan. Responsive Web Design / smashed.by/rwdala

140

Walton, Trent. Content Choreography / smashed.by/trent

141

Warren, Samantha. Style Tiles as a Web Design Process Tool / smashed.by/styletiles

142

Более подробный обзор процесса дизайна Саманты представлен на www.styletil.es

143

Набор инструментов (фреймворк) для создания сайтов и веб-приложений. Включает в себя HTML и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов, включая JavaScript расширения. Разработан в Twitter. – Примеч. науч. ред.

144

TwitterBootstrap /smashed.by/boots .

145

smashed.by/drbb .

146

smashed.by/gel .

147

smashed.by/mc .

148

Стивен Хей описывал удобный технологический процесс будущего в предыдущем разделе этой книги.

149

Jay Stocks, Elliot. The Typography-Out Approach In The World Of Browser-Based Web Design / smashed.by/typeout .

150

smashed.by/jam .

151

smashed.by/bbc .

