Государственное казенное образовательное учреждение
высшего профессионального образования
«РОССИЙСКАЯ ТАМОЖЕННАЯ АКАДЕМИЯ»

К.Н. МЕЗЕНЦЕВ, И.И. НИКИТЧЕНКО,
А.В. СМИРНОВ

Основы работы в сети Интернет

УЧЕБНОЕ ПОСОБИЕ

по дисциплине

«Основы Интернета»

Москва
2012

УДК 339.543
ББК 32.81
М44

Допущено
Учебно-методическим советом Российской таможенной академии
в качестве учебного пособия для студентов, обучающихся
по специальностям «Экономика и управление на предприятии (таможня)»
и «Мировая экономика»

Рецензенты:

И.С. ШУВАЛОВА, доцент кафедры автоматизированных систем управления Московского автомобильно-дорожного государственного технического университета (МАДИ), канд. техн. наук;

Н.Г. ЛИПАТОВА, начальник научно-исследовательского центра, канд. техн. наук, старший научный сотрудник

Мезенцев К.Н. Основы работы в сети Интернет: учебное пособие / К.Н. Мезенцев, И.И. Никитченко, А.В. Смирнов. М.: Изд-во Российской таможенной академии, 2012. 80 с.

ISBN 978-5-9590-0300-5

В учебном пособии рассмотрены основные принципы организации сети Интернет и технология использования и разработки информационных ресурсов сети. Учебное пособие позволяет изучить принципы организации доступа к информационным ресурсам, технологию использования языков разметки. В пособии изучается также технология создания и обработки клиентских HTML-форм.

Учебное пособие предназначено для студентов Российской таможенной академии, обучающихся по специальностям «Экономика и управление на предприятии (таможня)» и «Мировая экономика» и по направлению «Менеджмент» (профили «Финансовый менеджмент» и «Производственный менеджмент»).

ВВЕДЕНИЕ

По своему содержанию учебное пособие полностью соответствует требованиям государственного образовательного стандарта для слушателей, обучающихся по специальностям: «Экономика и управление на предприятии (таможня)» и «Мировая экономика».

Учебное пособие позволяет изучить следующие вопросы курса «Основы Интернета»:

– структуру и особенности модели сетевого взаимодействия TCP/IP;
– семейство протоколов модели взаимодействия TCP/IP;
– технологию коммутации пакетов;
– классификацию подсетей Интернета;
– классификацию служб сети;
– способы адресации ресурсов сети;
– особенности доступа к файловым архивам;
– особенности современных языков разметки;
– приемы создания HTML-страниц и сайтов;
– назначение и особенности языка разметки XML;
– технологию создания корректных и валидных XML-документов;
– технологию вывода содержания XML-документа с помощью каскадных стилевых таблиц CSS;
– технологию трансформации содержания XML-документа в HTML-страницу с использованием инструкций языка XSL;
– назначение и структуру клиентских HTML-форм;
– технологию взаимодействия с клиентскими формами с помощью сценариев, написанных на языке программирования JavaScript.

Глава 1

ОСНОВНЫЕ ПРИНЦИПЫ ОРГАНИЗАЦИИ СЕТИ ИНТЕРНЕТ

Сеть Интернет является глобальной сетью, созданной для свободного доступа к информационным ресурсам. В основе работы сети лежит сетевая модель взаимодействия TCP/IP.

1.1. МОДЕЛЬ TCP/IP

Модель взаимодействия определяет способ обмена информацией в компьютерной сети. В сети Интернет передача информации происходит путем коммутации пакетов (далее – КП) байтов данных.

При КП исходное сообщение, предназначенное для передачи от адресата к приемнику, разбивается на меньшие части – пакеты байтов. Каждый из пакетов имеет установленную максимальную длину. Передача осуществляется методом промежуточного хранения пакетов в узлах сети при передаче сообщения адресату. Узел сети – это определенный компьютер или группа компьютеров. Такие узлы называются узлами коммутации (далее – УК).

Сущность метода КП:

- вводимое в сеть сообщение разбивается на части – пакеты. Разбиение осуществляется либо в источнике сообщения, либо в ближайшем УК;
- при разбиении в УК дальнейшая передача ведется по мере их формирования, не дожидаясь окончания приема в УК всего сообщения;
- в УК пакет запоминается в оперативной памяти и по адресу определяется канал, по которому его надо передать;
- если канал свободен, то пакет немедленно передается на соседний узел коммутации;
- если канал занят, то пакет хранится в ОЗУ до освобождения канала;
- сохраняемые пакеты помещаются в очередь по направлению передачи, длина очереди не должна превышать 3,4 пакета, в противном случае пакеты стираются из ОЗУ и передача возобновляется снова.

Пакеты, относящиеся к одному сообщению, могут передаваться по разным маршрутам.

1.2. МЕТОДЫ ПАКЕТНОЙ КОММУТАЦИИ

Датаграммный метод (далее – ДМ) используется при передаче коротких сообщений. Датаграмма – самостоятельный пакет, движущийся по сети независимо от других пакетов. Маршруты доставки пакетов определяются сложившейся динамической ситуацией в сети. Поступают пакеты на прием в произвольной последовательности. Поиск маршрута происходит с помощью алгоритма ранжирования УК. Ближайший узел получает ранг 1. Пакет сначала посылается в узел первого ранга, при неудаче – во 2 и т.д. Выбор маршрута может носить случайный характер.

Виртуальный метод (далее – ВМ) предполагает предварительное установление маршрута от получателя к отправителю с помощью специального служебного пакета – запроса вызова. Для этого пакета выбирается маршрут. В случае согласия получателя начинается передача всего трафика.

Виртуальный канал – логическая связка между отправителем и получателем.

После того как отправитель получил подтверждение от приемника, что все пакеты получены, виртуальный канал разрывается.

Пакетная передача данных выполняется в рамках четырехуровневой модели взаимодействия. Нумерация уровней ведется от старшего уровня к младшему.

1.3. УРОВНИ СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ

Таблица 1.1

Модель TCP/IP

Номер уровня	Наименование	Описание
4	Прикладной	Программы конечных пользователей для работы в сети
3	Транспортный	Доставка пакетов между узлами сети
2	Сетевой	Адресация и маршрутизация пакетов
1	Канальный	Сетевые аппаратные средства и их драйверы

На каждом уровне сети передача данных определяется протоколом. Протокол – это набор правил взаимодействия на данном уровне и правил взаимодействия с вышестоящим уровнем и нижестоящим. Схема уровней и имена протоколов сетевой модели показаны на рис. 1.1.

Р и с . 1.1. Взаимодействие в сетях TCP/IP

Самый низкий уровень в иерархии – уровень 1 – уровень канала связи. Это физический уровень, на котором используется Ethernet-адрес с разрядностью шесть байтов.

Передача данных на сетевом уровне осуществляется с помощью IP-адресов с разрядностью четыре байта. Преобразование адресов выполняется по протоколу ARP (Address Resolution Protocol) – протоколу разрешения адресов.

На сетевом уровне передачи данных требуется также установка правил адресации ресурсов сети. Эти правила определяются протоколом IP (Internet Protocol – протокол сети Интернет). Согласно этому протоколу ресурсы сети должны обладать уникальным четырехбайтовым адресом. По названию протокола такие адреса принято называть IP-адресами.

На транспортном уровне передача данных должна отвечать правилам, которые определяют два протокола TCP (Transmission Control Protocol) и UDP(User Datagram Protocol). Первый протокол используется для доставки пакетов с контролем при передаче по виртуальному каналу. Второй протокол используется для передачи данных датаграммами – пакетами без контроля передачи.

При движении данных в рамках четырехуровневой модели взаимодействия используется механизм стека протоколов.

Движение данных по сети требует выполнения определенных преобразований форматов данных:

- инкапсуляции/экскапсуляции;
- фрагментации/дефрагментации.

Инкапсуляция – способ упаковки данных в формате вышестоящего протокола в формат нижестоящего протокола. При этом один или несколько первичных пакетов преобразуются в один вторичный пакет и снабжаются управляющей информацией, характерной для принимающего уровня.

При возврате на верхний уровень исходный формат восстанавливается в соответствии с обратной процедурой – экскапсуляцией (рис. 1.2).

Фрагментация – реализуется, если разрешенная длина пакета нижнего уровня недостаточна для размещения первичного пакета, при этом осуществляется «нарезка» пакетов, при возврате на первичный уровень пакет должен быть дефрагментирован.

Р и с . 1.2. Стек протоколов

1.4. КЛАССИФИКАЦИЯ СЕТЕЙ

Глобальная сеть Интернет используется для объединения в единое адресное пространство различных сетей и локальных компьютеров. В зависимости от особенностей IP-адреса в сети принята определенная классификация подсетей, показанная в табл. 1.2.

Таблица 1.2

Классификация подсетей

Класс	Первый байт	Маска	Комментарий
A	1–126	C.M.M.M	Крупные сети, не используется
B	128–191	C.C.M.M.	Большие узлы с подсетями
C	192–223	C.C.C.M.	Выделяются юридическим лицам
D	224–239	Нет	Групповая адресация
E	240–254	Нет	В стадии разработки

В основе классификации лежит значение кода первого байта IP-адреса подсети и тип сетевой маски. Адрес IP состоит из четырех байтов. В десятичной системе исчисления значение байта адреса может хранить число, равное 2^8 (256). Диапазон кода для байта составляет интервал от 0 до 255. Маска состоит из байтов, которые кодируют адрес подсети. Условные обозначения в маске: C – адрес подсети, условное обозначение M – адрес компьютера. В числовом виде маска показывает, какие байты адреса могут быть использованы для задания адреса компьютера подсети. Такие байты адреса в маске нулевые. Байты, отводимые для адреса подсети в маске, помечаются кодом 255.

Соответственно для сетей A, B, C маски в числовом виде примут вид:

A – 255.0.0.0 B – 255.255.0.0 C – 255.255.255.0

Например, IP-адрес 193.120.100.003 говорит о том, что это адрес подсети класса C, так как значение первого байта равно 193. Адрес подсети, согласно маске, равен 193.120.100. В подсети находится компьютер с адресом 003. Для размещения в этой подсети еще одного компьютера ему нужно присвоить адрес 193.120.100.004.

1.5. СЛУЖБЫ СЕТИ

При работе в сети Интернет нужно представлять, как адресуются ресурсы сети на прикладном уровне.

Для этого используют понятие порта. Понятие «порт» присуще как протоколу UDP, так и протоколу TCP. Поле порта в пакете занимает 2 байта (16 бит). Номер порта по традиции записывается в виде десятичного числа. Порты нумеруются с нуля. Если IP-адрес характеризует компьютер в сети, то порт характеризует ту или иную прикладную программу пользователя для работы в сети на этом компьютере.

На прикладном уровне используются две составляющие адреса ресурса. При этом IP-адрес определяет, куда передаются данные, а номер порта – прикладной процесс и связанную с ним программу для работы с данными.

Прикладной процесс, предоставляющий некоторые услуги другим процессам (сервер), ожидает поступления сообщений по некоторому специально выделенному порту, как говорят, «слушает порт». Запросы на предоставление услуг посылаются процессами – клиентами. В этом случае говорят об организации работы по типу «клиент-сервер».

Некоторые общеизвестные порты и службы показаны в табл. 1.3.

Таблица 1.3

Службы Интернет

Порт	Протокол	Служба
21	TCP	FTP (передача файлов)
25	TCP	SMTP (электронная почта)
80	TCP	HTTP (передача гипертекста)
110	TCP	POP (электронная почта)
53	UDP	DNS (служба доменных имен)

1.5.1.Система доменных имен

Служба DNS (Domain Name System) представляет собой систему доменных имен. Доменное имя – мнемоническое имя компьютера в сети. Доменные имена строятся по иерархическому принципу. Домены верхнего уровня имеют значение для стран, и они стандартизованы.

Например: us (США), uk (Великобритания), ru (Российская Федерация). Затем могут быть указаны доменные имена организаций. Они также стандартизованы и показаны в табл. 1.4.

Таблица 1.4

Стандартные домены

Домен	Назначение
gov	Правительственная организация
mil	Военная организация
edu	Образовательное учреждение
com	Коммерческая организация
net	Организация управления сетью
org	Прочие организации

Затем в доменном имени идут обозначения регионов и наименование компьютера. Преобразование доменных имен в числовые адреса IP происходит с помощью специальных серверов доменных имен DNS.

Для этого используется специальное программное обеспечение BIND (Berkley Internet Name Domain) – программа поддержки DNS.

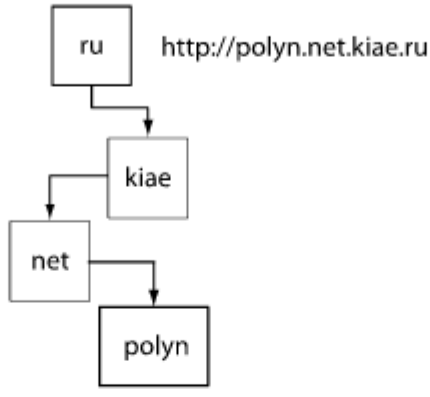
Данная программа функционирует на серверах сети. При этом выделяют типы серверов сети:

- Primary master сервер. Поддерживает свою базу имен и местный доменный;
- Secondary master сервер. Обсуживает свой домен, но данные об адресах части своих компьютеров получает по сети от другого сервера;
- Caching сервер. Не имеет своего домена, данные получает от одного из серверов либо из буфера;
- Remote server. Удаленный сервер. Обычный – сервер, установленный на удаленной ЭВМ, к которому обращаются программы по сети.

В совокупности доменные адреса образуют адрес компьютера сети, который принято называть URL (Uniform Resource Locator) адресом. В начале такого адреса принято указывать протокол прикладного уровня, кото-

рый определяет способ передачи данных и их тип, а затем следует доменный адрес.

На рис. 1.3 показан пример адресации ресурса по протоколу передачи гипертекста HTTP.



Р и с . 1.3. Пример URL-адреса

1.5.2. Электронная почта

Электронная почта обслуживается службой, которая использует ряд протоколов.

Протокол SMTP (Simple Mail Transfer Protocol) использует транспортный протокол TCP. Сообщение отправляется только в том случае, если установлено интерактивное соединение с программой-сервером на машине получателя почты.

Протокол UUCP (Unix to Unix Communication Protocol). Почта передается по цепочке почтовых серверов, пока не достигнет машины получателя.

Выборка получателем почты происходит по правилам, определяемым протоколом POP3 (Post Office Protocol) или IMAP(Interactive Mail Access Protocol). Отличительной особенностью этих протоколов является наличие функции поиска нужного сообщения и правил разбора заголовка сообщения.

Собственно почтовое сообщение должно отвечать определенным правилам. Эти правила определяют формат сообщения.

Формат электронной почты должен отвечать спецификации RFC(Request for Comments) – 822.

Согласно этой спецификации почтовое сообщение должно состоять из «конверта», заголовка и тела сообщения. Заголовок и тело сообщения формирует пользователь почтовой службы. «Конверт» используется специальными программа сети, которые называются транспортными агентами.

Основными полями данных «конверта» являются:

- Date (дата отправки сообщения);
- From (сведения об отправителе);
- To (сведения о получателе).

Спецификация RFC-822 определяет почтовое сообщение как текстовый документ, состоящий из набора символов.

Для расширения возможностей почтовой службы используется дополнительный стандарт на электронную почту MIME (RFC-1341).

Стандарт MIME (Multipurpose Internet Mail Extension) предназначен для описания тела почтового сообщения и дополняет RFC-822.

Спецификация RFC-822 подробно описывает заголовок почтового сообщения, текстовое содержание и механизм рассылки. Спецификация MIME главным образом ориентирована на описание в заголовке письма структуры тела почтового сообщения и возможности составления письма из информационных единиц различных типов.

В стандарте зарезервировано несколько способов представления разнородной информации. Для этой цели используются специальные поля заголовка почтового сообщения:

- поле версии MIME, которое указывается в заголовке почтового сообщения и позволяет определить программе рассылки почты, что сообщение подготовлено в стандарте MIME;
- поле описания типа информации в теле сообщения, которое позволяет обеспечить правильную интерпретацию данных;
- поле типа кодирования сообщения.

Стандарт MIME определяет семь типов данных, которые можно передавать в теле письма:

- текст (text);
- смешанный тип (multipart);
- почтовое сообщение (message);
- графический образ (image);
- аудиоинформация (audio);
- фильм или видео (video)
- приложение (application).

Поле типа кодирования почтового сообщения введено для того, чтобы при получении данные были правильно распакованы.

Стандарт определяет еще два дополнительных поля:

- Content – ID;
- Content – Description.

Первое поле определяет уникальный идентификатор содержания, а второе служит для комментария содержания. Программами просмотра эти поля обычно не отображаются.

Для обеспечения конфиденциальности передачи почтовых сообщений используется специальная спецификация S/MIME (Secure/Multipurpose Internet Mail Extensions).

Спецификация S/MIME позволяет использовать шифрование почтового сообщения с помощью алгоритмов шифрования с асимметричным ключом.

Спецификация определяет также правила создания электронной подписи в почтовом сообщении.

Чтобы пользователь сети мог воспользоваться электронной почтой, он в общем случае должен обладать электронным адресом. Такой адрес имеет формат:

```
Условное_Имя_Пользователя@Доменный_Адрес_Сервера
```

Например:

```
User_AP@RAMBLER.RU
```

Для приема и передачи почтовых сообщений необходимо в общем случае выполнить настройку почтового программного обеспечения.

Процесс настройки требует:

- указать адрес сервера исходящей почты SMTP;
- указать адрес сервера входящей почты POP3;
- задать параметры учетной записи пользователя на почтовом сервере – условное имя и пароль.

В настоящее время процесс настройки почтовой службы упрощается для конечных пользователей. Современные поисковые системы, например Rambler, Yandex, предоставляют пользователям бесплатную функцию – организацию почтового «электронного» ящика. При использовании этой услуги пользователь формирует свой электронный адрес и задает пароль для входа в свой «электронный» почтовый ящик. Функции настройки серверов берет на себя поисковая система.

1.5.3. Передача файлов в сети

Передача файлов по сети выполняется FTP-службой. Этой службе соответствует одноименный протокол FTP (File Transfer Protocol). Обмен данными проходит по TCP виртуальному каналу.

Файлы в сети собраны в специальные сетевые архивы. Сетевой архив – это распределенный банк данных. Пользователь может реализовать аноним-

ный доступ к этому хранилищу и скопировать интересующие его материалы. Информация в архивах разделена на три категории.

Защищенная информация, режим доступа к ней определяется ее владельцами и разрешается по специальному соглашению с пользователем.

Информационные ресурсы ограниченного использования (shareware), свободно распространяемые информационные ресурсы (free-ware).

Ресурсы ограниченного использования – это обычно программное обеспечение, использование которого имеет ограниченный срок или часть функций которого заблокирована до момента покупки программного обеспечения.

Свободно распространяемые ресурсы – это программное обеспечение и материалы, которые пользователь сети может использовать без ограничения их функций, срока и дальнейшей покупки.

Для поиска архивов и файлов по сети пользователи используют поисковые серверы. Основной особенностью использования таких серверов является формирования на естественном языке поискового требования. Поисковое требование вводится в специальной строке ввода поисковой программы – сервера. При вводе поискового требования можно использовать язык формирования требований.

Например, в поисковой системе Yandex для формирования поискового требования могут быть использованы приемы его формирования, показанные в табл. 1.5.

Таблица 1.5

Поисковый язык

Номер	Образец запроса или его фрагмент	Действие
1	«Примеры программ JavaScrip»	Поиск производится с учетом точного следования слов в предложении
2	«Примеры * JavaScrip»	Между словами предложения пропущено произвольное слово
3	Программы&JavaScript	Поиск слов в пределах одного предложения
4	Отображение&&XML	Поиск сочетания слов в пределах одного документа
5	JavaScript|XML|HTML	Поиск любого из слов в документе
6	Библиотека книг<<HTML	Неранжирующее «и»: слово после оператора не влияет на позицию документа в списке выданных

Окончание таблицы 1.5

Номер	Образец запроса или его фрагмент	Действие
7	Создание/2сайтов	Расстояние в пределах двух слов в любую сторону (т.е. между заданными словами может встречаться одно слово)
8	Публикация&&/3документов	Расстояние в три предложения в любую сторону
9	Использование~~глобальных описаний	Исключение слова «глобальных»
10	Публикация/+2статей	Расстояние в пределах двух слов в прямом порядке
11	Публикация ~книг статей	Исключение слова «книг»
12	!Пример!xml!HTML	Слова в точной форме с заданным регистром
13	title:(администрирование)	Поиск в заголовках документа
14	url:myserver.ru/index.htm	Поиск по URI
15	Политика inurl:libdoc	Поиск с учетом фрагмента URI
16	site:http://www.serv.ru/doc/xml	Поиск по всем под доменам и страницам заданного сайта
17	domain:ru	Поиск с ограничением по домену
18	date:200712* date:20071225	Поиск с ограничением по дате: 2007 год, 12 месяц Поиск по дате: 2007 год, 12 месяц, 25 число
19	date:20071215..20080101, date:>20091231	Поиск с ограничением по интервалу дат

С учетом особенностей формирования требований, указанных в таблице, может быть сформировано такое поисковое предписание:

Электронные книги JavaScript|XML|HTML date:2005*...2010*

Качество работы поисковой программы оценивается по релевантности полученных результатов. Под релевантностью понимают степень соответствия полученных результатов поиска поисковому требованию.

В общем случае поисковую систему сети Интернет можно рассматривать как документальную информационную поисковую систему. Для оценки качества работы таких систем предлагается использовать следующую методику [2].

Вводится набор обобщенных показателей, характеризующих процесс выдачи пользователю документов. При таком подходе выделяются классы документов, приведенные в табл. 1.6.

Таблица 1.6

Типы документов

	Выданные	Не выданные
Релевантные	A	C
Нерелевантные	B	D

В таблице 1.6 :

A – массив выданных релевантных документов;

B – массив выданных нерелевантных документов;

C – массив не выданных релевантных документов;

D – массив не выданных нерелевантных документов.

Исходя из приведенной классификации документов, вводятся следующие показатели качества работы информационно-поисковой системы:

a – количество выданных релевантных документов;

b – количество выданных нерелевантных документов;

c – количество не выданных релевантных документов;

d – количество не выданных нерелевантных документов.

Оценка качества производится на основе следующих коэффициентов:

Коэффициент полноты p – характеризует долю выданных релевантных документов во всем массиве релевантных документов:

$$p = \frac{a}{a+c}. \tag{1.6}$$

Коэффициент точности t – характеризует долю выданных релевантных документов во всем массиве выданных документов:

$$t = \frac{a}{a+b}. \tag{1.7}$$

Коэффициент шума s – характеризует долю выданных нерелевантных документов во всем массиве выданных документов:

$$s = \frac{b}{a+b}. \tag{1.8}$$

Коэффициент осадка q, характеризующий долю выданных нерелевантных документов во всем массиве нерелевантных документов:

$$q = \frac{b}{b+d}. \tag{1.9}$$

Коэффициент специфичности k, характеризующий долю не выданных нерелевантных документов во всем массиве нерелевантных документов:

$$k = \frac{d}{b + d}. \tag{1.10}$$

Контрольные вопросы

1. Как осуществляется коммутация пакетов в сети Интернет?
2. Из каких уровней состоит модель TCP/IP?
3. Дайте краткую характеристику уровням модели TCP/IP.
4. Что такое инкапсуляция и экскапсуляция?
5. Как классифицируются подсети в сети Интернет?
6. Задайте два IP-адреса для размещения компьютеров в подсеть класса B.
7. Напишите маску для подсети класса B.
8. Как организуется доступ к службам сети Интернет?
9. Как связаны между собой URL-адрес и IP-адрес компьютера сети?
10. Перечислите стандартные домены организаций.
11. В чем разница между стандартом на электронную почту RFC-822 и стандартом MIME?
12. Перечислите действия, необходимые для настройки электронного почтового ящика.
13. Дайте классификацию информационных ресурсов, доступных в файловых архивах сети Интернет.
14. Что такое поисковое требование?
15. Как используется поисковая система для поиска информации?
16. Какие показатели используются для оценки качества работы поисковой системы?

Глава 2

ИНФОРМАЦИОННЫЕ СЕТЕВЫЕ ТЕХНОЛОГИИ

2.1. ЯЗЫКИ РАЗМЕТКИ ДОКУМЕНТОВ

Передача информации по сети осуществляется в виде документов, оформленных по определенным правилам. В основе этих правил лежат принципы разметки содержания документа. Данные правила реализуются специальными языками разметки.

В основе современных языков разметки лежат принципы, определенные в языке разметки SGML (Standard Generalized Markup Language). Это обобщенный мета-язык разметки, который определяет правила формирования разметок содержания документа.

В настоящее время на основе этого мета-языка было создано два языка разметки HTML (Hyper Text Markup Language – язык гипертекстовой разметки) и XML (eXtensible Markup Language – расширяемый язык разметки).

Язык разметки HTML является реализацией SGML, а язык разметки XML – это подмножество языка SGML.

Документ, созданный с помощью языков разметки, представляет собой текстовый документ, состоящий из кодов символов национального языка. В составе такого документа есть специальные структуры, которые отмечают части документа. Эти структуры называют тегами разметки.

Правила формирования тегов определяются конкретным языком разметки.

В языке HTML тег отмечает определенную часть документа и несет информацию о том, как будет выглядеть отмеченная часть при ее просмотре пользователем информационного ресурса.

Документ, содержащий разметку, передается по сети в виде файла и может быть просмотрен специальной программой браузером, в состав которой входит интерпретатор языка разметки. Интерпретатор выполняет определенные действия над содержанием разметки, которые определены тегами, и выводит документ пользователю для просмотра.

2.2. ЯЗЫК ГИПЕРТЕКСТА HTML

Понятие «гипертекст» было введено Тедом Нельсоном в 1965 г. для обозначения «текста, ветвящегося или выполняющего действия по запросу».

Обычно гипертекст – это набор текстов, содержащих узлы перехода между ними, которые позволяют выбирать читаемые сведения или последовательность чтения.

Гипертекстовые документы создаются с помощью языка гипертекстовой разметки HTML и называются страницами.

Для создания страниц можно использовать любой текстовый редактор, поддерживающий кодировку ANSI, например редактор «Блокнот», который входит в состав стандартных приложений операционной системы Windows.

Страницы хранятся в файлах с расширением HTM или HTML. Они передаются по сети по правилам, определяемым протоколом передачи HTTP (HyperText Transfer Protocol – протокол передачи гипертекста). Этот протокол предусматривает взаимодействие в сети по принципу «клиент-сервер».

Пользователь сети, используя специальную программу, браузер, формирует запрос к серверу для получения страницы, страница передается пользователю. Для доступа к страницам нужно указать ее адрес URI (Uniform Resource Identifier) – универсальный идентификатор ресурса. Этот адрес складывается из двух составляющих URL и URN (Uniform Resource Name) единообразного названия ресурса.

Первая составляющая адреса – это доменное имя сервера, на котором расположена страница, а вторая – адрес, определяющий местоположение страницы – ресурса на сервере.

Пример:

```
http://myserver.org.ru/doc/html/index.html
```

Здесь:

```
URL=http://myserver.org.ru
URN= doc/html/index.html
```

Страница HTML состоит из символов текста и тегов. В общем виде теги имеют формат:

а) `<T>содержание</T>`

б) `<T>`

Здесь `T` – наименование тега. Теги первого типа называются парными, так как они имеют открывающую и закрывающую часть. Тег второго типа называется одиночным. Наименования тегов могут записываться в произвольном регистре.

2.2.1. Структура страницы

Структура страницы определяется набором тегов HTML, HEAD, TITLE, BODY.

- HTML – отмечает начало страницы;
- HEAD – помечает головную часть страницы;
- TITLE – используется для задания текста, этот текст выводится в заголовке окна браузера при загрузке страницы;
- BODY – отмечает основную часть страницы, в этой части находится информация, предназначенная для пользователя. Это содержание страницы.

Эти теги должны быть определенным образом вложены друг в друга:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="c1" TEXT="c2" LINK="c3" VLINK="c4"
BACKGROUND="uri">
</BODY>
</HTML>
```

Внутри тега BODY можно использовать ряд параметров для оформления страницы. Ниже перечислены значения для этих параметров.

- c_1 – цвет фона страницы;
- c_2 – цвет текста страницы;
- c_3 – цвет ссылок;
- c_4 – цвет просмотренной ссылки;
- uri – адрес фонового рисунка.

2.2.2. Теги форматирования текста

Теги форматирования используются для оформления информационного содержания страницы. Ниже перечислены основные теги этой группы:

<B>содержание</B> – жирный текст;

<U>содержание</U> – подчеркнутый текст;

<S>содержание</S> – перечеркнутый текст;

<PRE>содержание</PRE> – сохранение форматирования текста;

<I> содержание</I> – курсивный текст;

<TT>содержание</TT> – моноширинный шрифт.

При форматировании текста страницы надо помнить, что интерпретатор HTML рассматривает содержание страницы как последовательность символов, на которые воздействует определенный тег. При этом управляющие коды конца строки игнорируются, ведущие пробелы строки также игнорируются. Если нужно чтобы содержание выводилось «как есть», используют специальный тег PRE. Его содержание не изменяется. Например, если требуется вывести в окно браузера текст:

HTML
TEST
OK !!!,

то его нужно разметить в виде:

```
<PRE>
HTML
TEST
OK !!!
</PRE>
```

Формирование строк в разметке выполняется тегом <BR>, его указывают в конце строки или между словами строки.

Пример:

```
<b>Hello World !</b><BR><i>Привет МИР !</i>
```

В окне браузера такой текст будет иметь вид:

Hello World !

Привет МИР !

Использование тега TT позволяет получить оформление текста шрифтом, похожим на шрифт пишущей машинки. Размеры символов такого шрифта по высоте и ширине одинаковые.

Формирование абзацев в тексте выполняется тегом P. Форматы тега:

<P> – начать новый абзац.

<P ALIGN="Type">содержание</P> – создать абзац с выравниванием. Параметр Type определяет тип выравнивания содержания внутри абзаца: center (по центру), right (по правому краю), left (по левому краю).

Создание заголовков в тексте выполняется тегом H. Принято говорить о заголовках определенного типа размера n.

Формат тега:

<Hn></Hn> , где n=1...6 размер символов. Размер символов текста является условным и определяется браузером. Размер 1 – крупные символы, далее символы уменьшаются.

Текст на странице можно разделить линиями с помощью тега HR.

Формат тега:

<HR color="цвет"> – вывод линии.

Текст страницы может центрироваться по границам окна браузера тегом <CENTER>содержание</CENTER>. Выделить текст можно как цитатный блок тегом <BLOCKQUOTE>содержание</BLOCKQUOTE>.

Управление видом отдельных частей страницы можно выполнить, используя тег FONT. С помощью этого тега изменяют цвет символов, шрифт текста, размер символов текста. В составе тега используют параметры:

FACE – в него записывают системное имя шрифта;

COLOR – хранит цветовой код;

SIZE – хранит размер символов шрифта.

Размер шрифта задается в относительных единицах от 1 до 7. Размер 1 – символы наименьшего размера.

Пример.

```
<font size="7" face="Courier">
<blockquote>Hello World!<br>Привет Мир!</blockquote>
</font>
```

Наименование шрифта вводится с соблюдением регистра. Если такого шрифта нет на компьютере клиента сети, то используется шрифт по умолчанию браузера.

При создании информационного содержания страницы могут применяться специальные символы. Такие символы используются, если нужно ввести символ, которого нет в явном виде на клавиатуре либо его использование вызывает противоречие в разметке.

Специальные символы вводятся по ссылке &name; здесь name – идентификатор символа. Ссылки на специальные символы приводятся в табл. 2.1.

Таблица 2.1

Таблица специальных символов

Символ	Назначение	Символ	Назначение
`&copy;`	Символ авторского права ©	`>`	Знак >
`&reg;`	Зарегистрированная торговая марка ®	`<`	Знак <
` `	Неразрывный пробел	`&`	Символ &
		`&quot;`	Символ “

С помощью ссылки на неразрывный пробел можно получить красную строку при выводе текста в окно браузера.

Например, нужно вывести текст:

Hello World !

Привет МИР !

Разметка примет вид:

```
   <b>Hello World !</b><br>
<i>Привет МИР !</i>
```

В разметке можно использовать комментарий. Это поясняющий текст, который не выводится в окно браузера. Оформляется комментарий тегом `<! - -Текст комментария - ->`. Комментарий должен начинаться с символа «!».

2.2.3. Упорядочивание информации на странице

Может быть выполнено с помощью следующих средств языка разметки:

- списков;
- таблиц.

Списки

С помощью списков можно упорядочить строки текста, снабдив их определенными маркерами, и получить определенное взаимное расположение.

Существуют три вида списков, показанных в табл. 2.2.

Таблица 2.2

Типы списков

Нумерованный список	Маркированный список	Список определений
<OL>	<UL>	<DL>
<LI>строка$_1$	<LI>строка$_1$	<DT>Термин$_1$<DD>Определение$_1$
<LI>строка$_2$	<LI>строка$_2$	<DT>Термин$_2$<DD>Определение$_2$
...	...	...
<LI>строка$_i$	<LI>строка$_i$	<DT>Термин$_i$<DD>Определение$_i$
</OL>	</UL>	</DL>

Теги OL, UL и DL отмечают область страницы, где находится список. Тег LI отмечает строку – элемент списка, его можно закрыть парным тегом </LI>. Тег DT отмечает строку термин, а тег DD – строку определения этого термина, она выводится ниже термина с отступом.

В табл. 2.3 приведены теги для формирования трех разных типов списков.

Таблица 2.3

Примеры списков

Нумерованный список	Маркированный	Список определений
		<dl>
Yes	Yes	<dt>Yes<dd>Да
No	No	<dt>No<dd>Нет
Ignore	Ignore	<dt>Ignore<dd>Отмена
		</dl>

Результат действия тегов показан на рис. 2.1.

Нумерованный список **Маркированный список** **Список определений**

1. Yes
2. No
3. Ignore

- Yes
- No
- Ignore

Yes
Да
No
Нет
Ignore
Отмена

Рис. 2.1. Фрагмент страницы со списками в браузере

Таблицы

Создание таблицы требует использования ряда тегов. Тегом TABLE отмечают область страницы, где будет находиться таблица. Строки таблицы размечаются тегом TR. Для создания ячеек таблицы служит тег TD, а для создания ячеек шапки таблицы можно использовать тег TH, его содержание выводится жирным шрифтом.

Таблицу можно снабдить подписью – заголовком, используя тег CAPTION. Формат тега:

<CAPTION ALIGN=»Type»></CAPTION> – заголовок таблицы.

Тип выравнивания Type может принимать значения Top (заголовок над таблицей), Bottom (заголовок под таблицей).

Теги формирования ячеек можно снабдить параметром ALIGN, который задает выравнивание содержания ячеек. Форматы этих тегов примут вид:

<TH ALIGN=»Type»></TD>

<TD ALIGN=»Type»></TD>

Здесь значение Type – тип выравнивания в ячейке: center (выравнивание по центру), left (выравнивание по левому краю), right (выравнивание по правому краю).

При создании сложных таблиц в тегах TH и TD используют параметры COLSPAN и ROWSPAN, которые позволяют задавать ширину ячейки в клетках и высоту ячейки в строках соответственно.

Пример.

Дана таблица, нужно сформировать разметку для ее вывода.

Table sample

<table>
<tr><th>One</th><th>Two</th><th colspan="2">Three</th></tr>
<tr><td>Yes</td><td rowspan="3">Ok</td><td>One</td><td>Four</td></tr>
<tr><td>No</td><td>Two</td><td>Five</td></tr>
<tr><td>Cancel</td><td>Three</td><td>Six</td></tr>
</table>

Разметка для вывода данной таблицы примет вид:

```
<table border>
<caption align=»top»><b>Table sample</b></caption><tr>
<th>One</th><th>Two</th><th colspan=»2»>Three</th>
</tr><tr>
<td>Yes</td><td rowspan=»3» align=»center»>Ok</td>
```

```
<td>One</td><td>Four</td>
</tr><tr>
<td>No</td><td>Two</td><td>Five</td>
</tr><tr>
<td>Cancel</td><td>Three</td><td>Six</td></tr>
</table>
```

Table sample

One	Two	Three	
Yes	Ok	One	Four
No		Two	Five
Cancel		Three	Six

Рис. 2.2. Таблица в браузере

Параметр BORDER тега TABLE предписывает выводить разделительные линии в таблице. Вид таблицы в окне браузера показан на рис. 2.2.

Допускается задавать толщину линий разграфки таблицы с помощью параметра BORDER. Например, BORDER=»2» задает толщину линий в 2 пиксела.

2.2.4. Организация гипертекстовых переходов

Гипертекстовые переходы на страницах бывают двух типов: внешние ссылки и якоря.

Внешние ссылки

Внешняя ссылка оформляется тегом:

```
<A HREF = «uri»>Содержание</A>
```

Здесь uri – адрес страницы.

Заметим, что возможности протокола передачи гипертекста HTTP гораздо шире, uri-адрес может ссылаться на любой файл с данными, например, на файл с графическим изображением, на файл с мультимедиа данными. Текст содержания подчеркивается в тексте страницы и является ссылкой. Механизм внешних ссылок позволяет объединить несколько страниц. Такое объединение страниц по определенной тематике называется сайтом.

В составе сайта обычно есть главная страница, которая открывается пользователем, на этой странице находится оглавление сайта в виде гиперссылок. Такую страницу можно организовать в виде набора фреймов.

Создают такую страницу с помощью тегов FRAMESET, FRAME. Первый тег создает набор фреймов. Фрейм – часть окна браузера, куда может загружаться информация. Второй тег определяет адрес страницы, помещаемой во фрейм.

Страница, содержащая набор фреймов, не должна содержать тега BODY.

Формат тега для создания фреймов имеет вид:

```
<FRAMESET COLS=»n1,n2» | ROWS=»m1,m2»></FRAMESET>
```

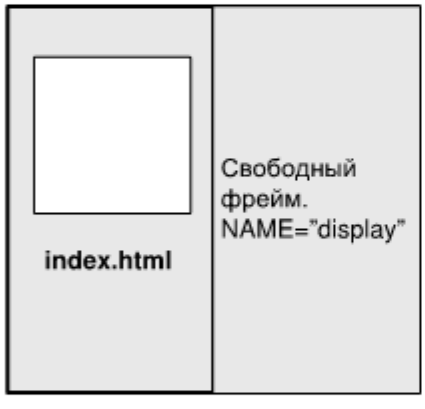
Параметры COLS и ROWS определяют пропорции деления окна браузера по вертикали и горизонтали. Использовать можно только один параметр. Пропорции деления задаются в процентах.

Тег создания фрейма имеет формат:

```
<FRAME NAME=»fraName1» SRC=»uri»>
```

Здесь uri – адрес страницы, загружаемый во фрейм. Параметр NAME содержит идентификатор фрейма. Это условное имя задается латинскими буквами и используется для организации обращения к фрейму. Теги FRAME вкладываются в тег FRAMESET. Их число должно соответствовать частям, заданным в параметре ROWS или COLS.

Страница сайта с оглавлением загружается в один из свободных фреймов, а другой свободный фрейм используется для просмотра содержания страниц, загружаемых по ссылке оглавления. Для этого в тег <A HREF...> страницы оглавления нужно включить параметр TARGET, значение которого должно быть идентификатором фрейма загрузки.



Р и с . 2.3. Главная страница с двумя фреймами

Например, требуется построить главную страницу сайта по следующей схеме, показанной на рис. 2.3.

Такая страница состоит из двух фреймов: левого и правого. Правый фрейм свободный, в него будут загружаться страницы при выборе их в оглавлении. Оглавление содержится в файле index.html. Оно должно содержать теги, загружающие страницы в правый фрейм.

Страница, делящая окно браузера на фреймы, должна содержать разметку:

```
<html>
<frameset cols=»50%,*»>
<frame src=»index.html»>
<frame src=»» name=»display»>
</frameset>
</html>
```

При задании пропорций оставшуюся часть окна можно заменять символом «*».

Пусть в составе сайта есть три страницы: p1.htm, p2.htm, p3.htm, которые хранят информацию по трем главным разделам сайта. Тогда разметка страницы index.htm будет иметь следующий вид:

```
<html>
<body>
<a href=»p1.htm» target=»display»>Первая глава</a><br>
<a href=»p2.htm» target=»display»>Вторая глава</a><br>
<a href=»p3htm» target=»display»>Третья глава</a><br>
</body>
</html>
```

При организации внешних ссылок необходимо помнить, что если загружаемая страница находится на удаленном сервере сети, то нужно указывать ее полный URI-адрес. Если страница находится в каком-либо каталоге клиента сети, то передача такой страницы выполняется по локальному URI-адресу по протоколу file. Формат такого адреса имеет вид:

```
file:///имя_диска|/путь_к_странице/my.htm
```

Пример:

```
file:///e|/doc/html/my.htm
```

Такой адрес означает, что страница my.htm находится в папке doc/html на диске E.

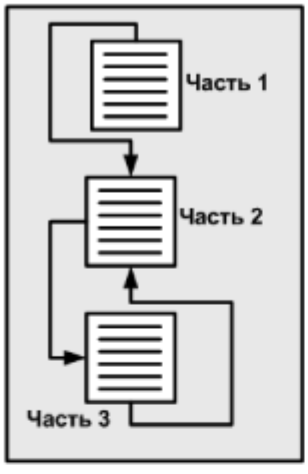
Использования полного имени файла, принятого в операционной системе Windows `e:\doc\html\my.htm`, следует избегать, так как такие адреса распознаются корректно только браузером MS Internet Explorer. Если в качестве URI-адреса указано имя файла страницы, например my.html, то поиск этого файла будет происходить в папке, где находится страница, содержащая тег внешней ссылки.

Организация якорей

Якорем называется технология организации ссылок на части одного и того же документа. Технология использования якоря показана на рис. 2.4.

Из рисунка видно, что страница состоит из трех частей. Первая часть связана со второй, вторая часть с третьей, третья часть со второй. Организация якоря требует использования двух тегов. Первый тег формирует ссылку на определенную часть страницы и имеет вид:

```
<A HREF=»#key»>содержание</A>
```



Р и с . 2.4. Документ с якорями

Второй тег помечает ту часть текста, на которую попадает пользователь при обращении к содержанию якоря. Формат тега:

```
<A HREF="#key">текст</A>
```

В обоих тегах `key` – это идентификатор перехода якоря, задается латинскими буквами и его название выбирается разработчиком страницы. Это название должно использоваться с учетом регистра символов.

Пример организации якоря:

```
<a href="#part1" Глава 1.</a> В которой появляется
герой романа.
.....
<a name="part1">Глава 1.</a> Явление героя.
Было теплое весеннее утро. Солнце уже встало над
крышами города и осветило ярким светом черепичные
красные крыши...
```

2.2.5. Основные понятия дизайна страниц

Чтобы страница привлекала внимание пользователя и позволяла лучше воспринимать материал, используют различные приемы оформления.

Рассмотрим основные из них.

Графические изображения

Чтобы поместить иллюстрации в тексте страницы, используют специальный тег:

```
<IMG SRC=»uri» WIDTH=»w» HEIGHT=»h» ALT=»Text»
ALIGN=»Type»>
```

Здесь `uri` – адрес графического файла с изображением. Параметры `WIDTH` и `HEIGHT` задают размеры области вывода графического изображения. Параметр `ALT` хранит текст, который выводится как подсказка для курсора мыши при наведении его на изображение. Кроме того, этот текст выводится, если в браузере отключен показ изображений. Параметр `ALIGN`

позволяет управлять размещением изображения относительно текста. Выравнивание Type может принимать следующие значения:

- Middle (текст выравнивается по центру изображения);
- Bottom (нижняя граница изображения выравнивается по окружающему ее тексту, используется по умолчанию);
- Top (верх изображения выравнивается по верхней строке окружающего его текста);
- Left (текст обтекает изображение справа);
- Right (текст обтекает изображение слева).

Изображение можно взять в рамку, если в тег IMG добавить параметр BORDER="n". Здесь n – ширина рамки в пикселах.

Для оформления обычно используют графические изображения формата GIF и JPEG.

Изображения первого типа занимают мало места в памяти, так как их цветовая палитра равна 256 цветам. Кроме того, они поддерживают анимацию и прозрачные цвета.

Изображения второго типа позволяют хранить и передавать в сжатом виде иллюстрации, которые характеризуются насыщенной цветовой палитрой и большим разрешением.

Графические изображения могут использоваться в качестве ссылок, тогда их помещают в качестве содержания в тег:
<A HREF…><IMG SRC=»uri»></A>.

Изображения можно использовать как фон страницы, такие изображения называются обоями и тиражируются по полю страницы при ее просмотре в браузере. Размещают обои с помощью параметра BACKGROUND=»uri» тега BODY. Здесь uri – адрес файла изображения.

Фоновые изображения можно задавать для ячеек таблицы с помощью этого же параметра, помещая его в тег создания ячейки таблицы TD.

Цветовая палитра браузера

При разработке гипертекстовых документов используется R(ed)G(reen) B(lue) цветовая модель. В такой модели цвет образуется из красной, синей и зеленой составляющих. Каждая составляющая задает определенную интенсивность базового цвета. Итоговый цвет – это смешение интенсивностей. Задается цвет в шестнадцатеричном формате в виде трех байтов. Код каждого байта изменяется в пределах от 00 до FF. Цветовая константа должна начинаться с префикса #.

В настоящее время существуют определенные соглашения по цветам гипертекста. Каждому цвету поставлена в соответствие определенная сим-

вольная константа. В табл. 2.4 приводятся основные цвета палитры гипертекста.

Таблица 2.4

Стандартные цвета

Системная константа, цвет	RGB значение	Системная константа	RGB значение
Black (черный)	#000000	Green (зеленый)	#008000
Navy (морская волна)	#000080	Teal (темная морская волна)	#008080
Silver (серый)	#c0c0c0	Lime (яркозе-леный)	#00ff00
Blue (голубой)	#0000ff	Aqua (прозрачная морская волна)	#00ffff
Maroon (фиолетовый)	#800000	Olive (оливковый)	#808000
Purple (пупрпурный)	#800080	Gray (темно-серый)	#808080
Red (красный)	#ff0000	Yellow (желтый)	#ffff00
Fuchsia (рыжий)	#ff00ff	White (белый)	#ffffff

Цвет может быть использован в параметре COLOR тегов HR, FONT. В параметрах тега BODY: TEXT, BGCOLOR, LINK, VLINK (см. описание тега BODY).

Цвет можно использовать при работе с таблицами. Так, в тег TD формирования ячейки таблицы можно включить параметр:

<TD BGCOLOR="цвет"></TD>, параметр можно также поместить в тег TABLE и задать цвет для всей таблицы. Кроме того, в тег TABLE можно поместить параметр BORDERCOLOR и задать цвет линиям разграфки таблицы.

Контрольные вопросы

1. Какие современные языки разметки используются для создания информационных ресурсов?
2. Как адресуется сетевой ресурс с помощью URI-адреса?
3. Перечислите структурные теги и их особенности.
4. Как строится разметка, если нужно создать текст с красной строкой и заданным переносом слов?
5. Как выполняется выравнивание текста на странице?

6. Для чего нужны ссылки на символы при создании разметки?
7. Дайте классификацию списков.
8. Перечислите теги, необходимые для создания таблицы.
9. Как выполняется форматирование таблиц?
10. Как управлять шрифтами на странице?
11. Дайте классификацию гипертекстовых переходов.
12. Как формируются страницы с фреймами?
13. Как выполняется размещение графических изображений на страницах?
14. Перечислите основные особенности использования цветов при оформлении страницы.

2.3. ТЕХНОЛОГИЯ XML

Технология XML представляет собой протокол хранения и передачи информации. С другой стороны, – это семейство технологий, позволяющее формировать различные документы и выполнять обработку информации.

При использовании XML разработчик создает документ с разметкой. Документ хранится в текстовом файле с расширением xml. Как и для создания HTML-страниц разработку документов можно вести в любом текстовом редакторе, поддерживающем кодировку ANSI.

Документ состоит из следующих основных частей: пролога и основной части.

В прологе документа располагаются директивы, необходимые интерпретатору XML для обработки документа. Директива начинается и заканчивается специальным символом «?». В прологе обязательно должна быть директива:

```
<?xml version=»1.0» encoding=»type»?>
```

В ней указываются версия – XML и тип кодировки документа – `type`. Для использования кириллицы применяются кодировки:

windows -1251 или ISO-8859-5.

Первая кодировка используется в текстовых редакторах операционной системы Windows, и ее использование предпочтительнее.

Основная часть документа следует за прологом. Эта часть документа содержит собственно разметку. Разметка XML – выделение структурных составляющих документа тегами. Разработчик документа сам выбирает названия тегов по смыслу документа. Для названия тегов используются латинские символы.

Разметка должна отвечать определенным правилам. Ниже рассмотрены эти правила.

Теги в документе могут быть двух типов:

Парные:

```
<name>информация</name>
```

Одиночные:

```
<name/>
```

Здесь `name` – наименование тега.

Документ должен отвечать следующим требованиям:

- содержать корневой тег;
- внутри корневого тега должны располагаться дочерние;
- в дочерний тег может вкладываться другой;
- число вложений не ограничено;
- теги должны быть замкнуты, если они парные;
- регистр символов в парных тегах должен совпадать;
- пересечение тегов не допускается.

Соблюдение перечисленных правил необходимо для того, чтобы получить «корректно форматированный документ».

Пример корректного XML-документа:

```
<?xml version="1.0" encoding="windows-1251"?>
<TEST>
<NODE>
<ENG>Hello World !</ENG>
<RU>Привет МИР !</RU>
</NODE>
</TEST>
```

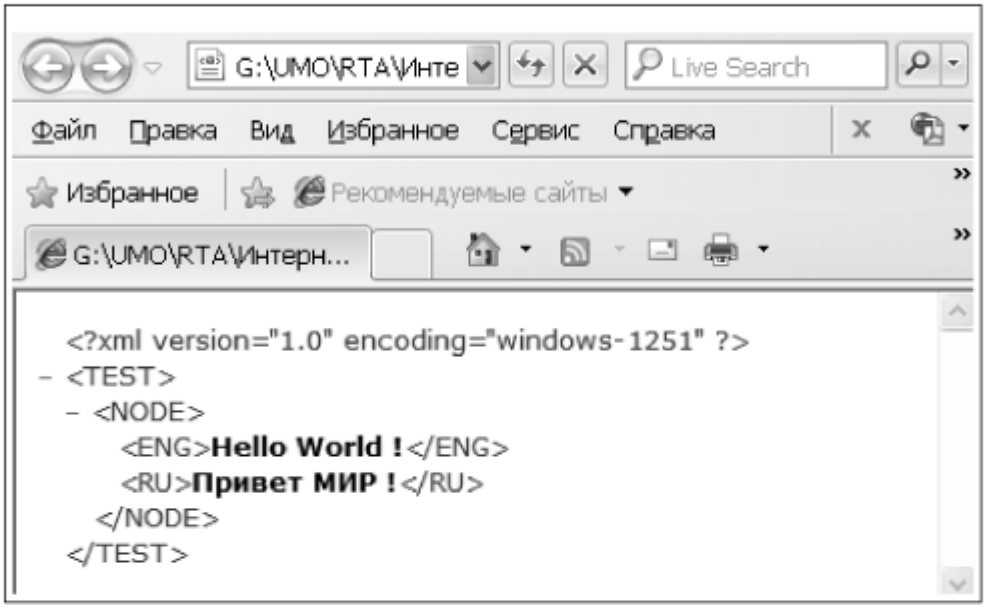
Р и с . 2.5. Структура документа

На рис. 2.5 показана его структура.

Документ XML представляет собой иерархическую структуру.

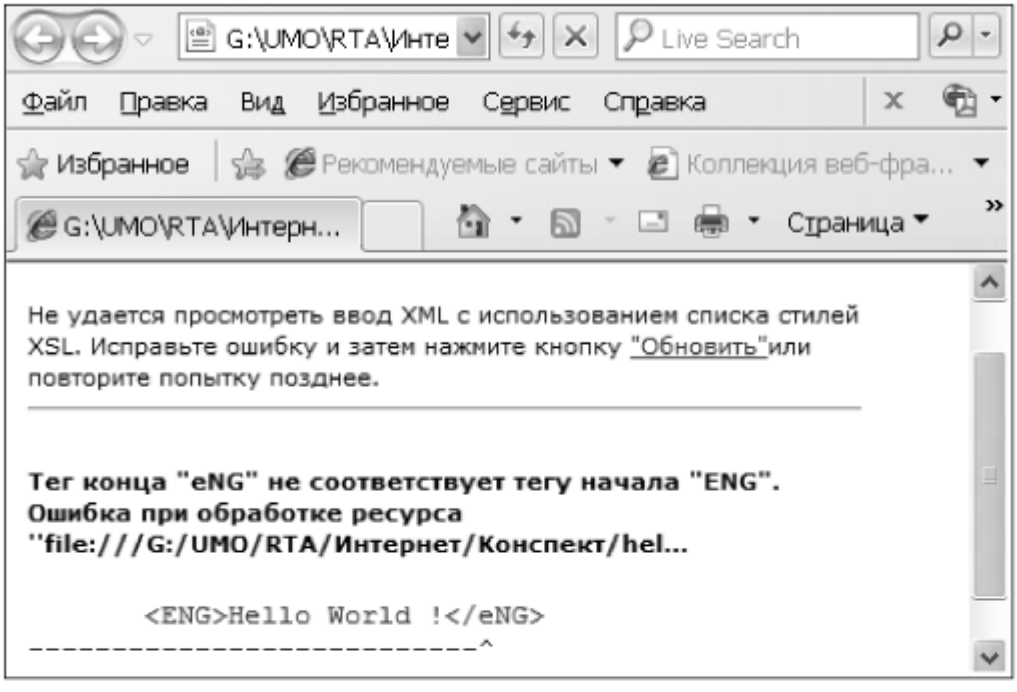
В начале иерархии располагается корневой элемент (не путать с корневым тегом!). Это начало документа, его пролог. Затем располагается корневой тег, который содержит остальные теги разметки, – это тег `TEST`. Он содержит тег `NODE`, который содержит два тега `ENG` и `RU`. Содержание двух последних тегов – сообщения на английском и русском языках.

Если сохранить рассмотренную разметку документа в текстовый файл hello.xml и загрузить в браузер, то она отобразится в иерархическом виде так, как это показано на рис. 2.6. Рядом с тегами выводится пиктограмма «+» или «–». Знак «–» означает, что тег раскрыт и видно его содержание, его можно закрыть, сделав на пиктограмме щелчок мышью.



Р и с . 2.6. Документ в браузере MS Internet Explorer

Следует иметь в виду, что браузер отображает только корректные XML-документы, если документ содержит ошибку в разметке, то выводится сообщение об ошибке и загрузка прекращается. Например, в закрывающем теге </ENG> изменим регистр первого символа </eNG> и повторим загрузку. Вид окна браузера показан на рис. 2.7.



Р и с . 2.7. Окно браузера при ошибке в разметке

2.3.1. Встроенные объектные ссылки

При формировании разметки можно использовать ссылки на символы. В XML имеется фиксированный набор именованных ссылок, показанных в табл. 2.5. При использовании ссылки обязательно должны оканчиваться символом «;».

Допускается использовать ссылки в виде кода. В качестве кода может быть указан любой десятичный символ в кодировке UNICODE. Примеры кодов некоторых символов:

- © (авторское право ©);
- ® (торговая марка ®);
- — (длинное тире).

Таблица 2.5

Ссылки XML

Ссылка	Символ
&amp;	&
&apos;	‘
&gt;	>
&lt;	<
&quot;	«

В разметке можно использовать комментарий. Это любой текст в паре треугольных скобок. Первый символ – знак «!», за которым нужно указать два знака «–», эти знаки должны закрывать комментарий.

Пример:

```
<!-- Текст комментария -->
```

2.3.2. Создание валидных документов

Документы XML обязательно должны быть корректными, дополнительно они могут отвечать требованию валидности. Валидность означает, что документ отвечает определенной структуре, заданной в описании шаблона документа DTD (Data Type Definition).

Схема DTD задается таблицей в прологе документа в следующем виде:

```
<!DOCTYPE name
[
Описание элементов
]>
```

Здесь `name` – имя корневого тега.

Описание элементов задается с помощью объявления:

```
<!ELEMENT Имя описание_содержимого>
```

Описание содержимого:

`EMPTY` – пустой элемент, который не может содержать дочерних тегов;

`ANY` – любое содержимое, содержание не определено, но элемент может содержать дочерние теги;

`(#PCDATA)` – символьное содержание.

В качестве описи содержимого могут указываться другие теги, которые вложены в тег:

```
Тег(дочерние теги).
```

При описании дочерних структур действуют следующие правила:

– перечисление через запятую – обязательный перечень элементов и порядок элементов;

– использование символа «|» – означает выбор одного из дочерних тегов.

Пример.

Составим описание для документа с двумя предложениями на русском и английском языках:

```
<?xml version=»1.0» encoding=»windows-1251»?>
```

```
<!DOCTYPE TEST[
     <!ELEMENT TEST (NODE)>
     <!ELEMENT NODE (ENG,RU)>
     <!ELEMENT ENG (#PCDATA)>
     <!ELEMENT RU (#PCDATA)>
]>
<TEST>
     <NODE>
       <ENG>Hello World !</ENG>
       <RU>Привет МИР !</RU>
</NODE>
</TEST>
```

Следует иметь в виду, что интерпретатор XML-браузера проверяет только корректность документа, проверить валидность можно с помощью специальных программ либо специальных XML-редакторов кода.

При создании DTD для уточнения содержания используются символы, показанные в табл. 2.6.

Таблица 2.6

Уточнение содержания

Символ	Значение
?	Один дочерний элемент или ни одного
+	Один или несколько дочерних элементов
*	Ни одного или несколько дочерних элементов

Пример DTD описания шаблона со специальным символом:

```
<?xml version=»1.0» encoding=»windows-1251»?>
<!DOCTYPE firma[
     <!ELEMENT firma (sluj+)>
     <!ELEMENT sluj (id, fam)>
     <!ELEMENT id (#PCDATA)>
     <!ELEMENT fam (#PCDATA)>]>
<firma>
```

```
    <sluj>
      <id>m100</id>
      <fam>Петров</fam>
    </sluj>
    <sluj>
      <id>m200</id>
      <fam>Хватова</fam>
    </sluj>
</firma>
```

Здесь тег `sluj` – это блок данных о сотруднике фирмы, тег `id` – хранит личный номер сотрудника, а тег `fam` – его фамилию. Блок `sluj` должен быть как минимум один.

Можно повысить информационную емкость XML-тега, используя атрибуты (табл. 2.7). Атрибут описывается в DTD-шаблоне в виде:

```
<!ATTLIST Тег Атрибут Тип Спецификация>
```

Здесь `Тег` – имя тега, которому принадлежит атрибут. Тип атрибута может принимать значения из таблицы.

Таблица 2.7

Типы атрибутов

Тип	Особенности
CDATA	Произвольный текст
ID	Уникальное значение
IDREF	Ссылка на другой ID атрибут
IDREFS	Ссылка на другие ID атрибуты

Параметр *Спецификация* определяет правило задания значения атрибуту. Допустимые значения приводятся в табл. 2.8.

Таблица 2.8

Спецификация атрибутов

Значение	Особенности
#REQUIRED	Значение должно задаваться обязательно
#IMPLIED	Значение задавать не обязательно

Окончание таблицы 2.8

Значение	Особенности
#FIXED	Заданное значение по умолчанию изменять нельзя
Значение по умолчанию	Значение по умолчанию принимается автоматически
Перечисляемые значения	Список допустимых значений, разделенных символом «|»

Пример. Документ с атрибутами.

```
<?xml version=»1.0» encoding=»windows-1251»?>
<!DOCTYPE firma[
<!ELEMENT firma (rukov+)>
<!ELEMENT rukov (sotr+)>
<!ELEMENT sotr (#PCDATA)>
<!ATTLIST rukov
     zip ID #REQUIRED
     podch IDREFS #REQUIRED
     otdel (Маркетинг | Склад) #REQUIRED
     name CDATA #REQUIRED
>
<!ATTLIST sotr
     zip ID #REQUIRED
     firma_name CDATA #FIXED «ООО Иволга»>
]>
<firma>
<rukov zip=»r100» podch=» m101 m102»
otdel=»Маркетинг» name=»Артамонов И.С.»>
     <sotr zip=»m101» >Иванов П.И.</sotr>
     <sotr zip=»m102»>Сидорова О.П.</sotr>
</rukov>
</firma>
```

Данный документ задает структуру фирмы. Фирма состоит из нескольких руководителей тег `rukov`. Руководителю подчиняются несколько сотрудников тег `sotr`. Руководитель характеризуется атрибутами:

`zip` – личный номер руководителя;

`podch` – ссылка на личные номера подчиненных;

`otdel` – наименование отдела;

`name` – фамилия руководителя.

Задавать значения всех атрибутов обязательно.

Сотрудник характеризуется атрибутами:

`zip` – личный номер сотрудника

`firma_name` – наименование фирмы.

Код сотрудника задавать обязательно, а название фирмы нет, но если задано, то оно должно быть: ОООИволга.

В описании документа можно определить некоторую сущность – примитив и тиражировать ее в документе XML по ссылке. Задается сущность описанием:

```
<!ENTITY ИмяПримитива Значение>
```

Ссылка на сущность имеет вид:

```
&ИмяПримитива;
```

как это показано в следующем примере.

П р и м е р . Задание перевода предложения в виде сущности.

```
<?xml version=»1.0» encoding=»windows-1251»?>
<?xml-stylesheet type=»text/css» href=»entity.css»?>
<!DOCTYPE document[
<!ELEMENT document (verse)>
<!ELEMENT verse (#PCDATA)>
<!ENTITY hello «Hello World !»>
]>
<document>
<verse>&lt;Привет мир ! (&hello;)&gt;</verse>
</document>
```

Кроме определенной сущности в шаблоне `hello` используются ссылки на две встроенные сущности XML `lt` и `gt`.

2.4. ПРЕОБРАЗОВАНИЕ XML-ДОКУМЕНТОВ

Документы XML по умолчанию отображаются в браузере как иерархическая структура тегов с их содержанием. Разметка документа отмечает только смысловые – семантические его части и не несет никакой информации о том, как должно быть выведено содержание пользователю документа.

В настоящее время существуют различные технологии преобразования содержания документов XML. В данном пособии будут рассмотрены две такие технологии: каскадные таблицы стилей CSS (Cascading Style Sheets) и технология расширяемых стилевых таблиц XSLT (eXtensible Stylesheet Language Transformations).

2.4.1. Использование каскадных стилевых таблиц

Использование каскадной стилевой таблицы предусматривает создание специального текстового файла с расширением css. Этот файл должен содержать набор правил для отображения содержания тегов. Затем в пролог XML-документа нужно добавить директиву присоединения этого файла.

Формат директивы:

```
<?xml-stylesheet type="text/css" href="uri"?>.
```

Здесь `uri` – путь к CSS файлу. Если он находится в одной папке с XML-документом, достаточно указать его имя.

Файл каскадной стилевой таблицы состоит из правил отображения содержания тега документа. Правила называют селекторами, и они имеют в общем виде следующую структуру:

```
Тег|Теги {параметр1: значение;
параметр2: значение;
....
}
```

Здесь `параметр` – определенный стилевой параметр, ему нужно задать значение, которое должно быть закрыто символом «`;`».

Правила применяются к тегам или списку тегов, разделенных запятыми. Если правило назначено головному (родительскому) тегу иерархии, то оно распространяется на дочерние теги. Имена параметров и их значений регистрозависимые.

Пример. Рассмотрим ранее созданный XML-документ с двумя сообщениями и выведем его в следующем виде:

Hello World !

Привет МИР !

Текст выводится жирным шрифтом, русский текст курсивом. Английский текст – синим цветом, русский текст – красным.

Для решения этой задачи нужно составить таблицу стилей и поместить ее в файл hello.CSS, а затем подключить к XML-файлу и загрузить документ в браузер.

В CSS файл нужно поместить правила:

```
NODE {font-weight:bold;}
RU,ENG{display:block;}
ENG{color:#0000FF;}
RU{color:#FF0000;
    font-style:italic;}
```

Здесь:

параметр `font-weight` отвечает за яркость символов (жирность), значение `bold` означает задать стандартную яркость. Его действие распространяется на дочерние теги `RU` и `ENG` тега `NODE`.

Параметр `display` определяет, будет ли содержание тегов `RU` и `ENG` выводиться отдельным блоком. При значении `block` содержание тегов выводится как строки.

Параметр `color` задает цвет содержанию тегов `ENG` и `RU`.

Параметр `font-style` определяет начертание шрифта для содержания тега `RU`. Значение `italic` означает вывод текста курсивом.

В приложении 1 приводится описание базового набора стилевых параметров и их значений.

Если требуется выводить графические изображения при выводе содержания XML-документа, то для их размещения в составе разметки определяют специальный тег с пустым содержанием. В CSS-файле нужно создать правило для этого тега со следующими особенностями:

- нужно загрузить графический файл в элемент-тег;
- запретить его тиражирование по полю содержания тега;
- задать размеры области вывода;
- задать тип обтекания текстом.

На рисунке 2.8 показан вид окна браузера с загруженным документом, для которого создана стилевая таблица, обеспечивающая загрузку изображения формата BMP с разрешением 100 на 67 пикселей и форматирование текста.

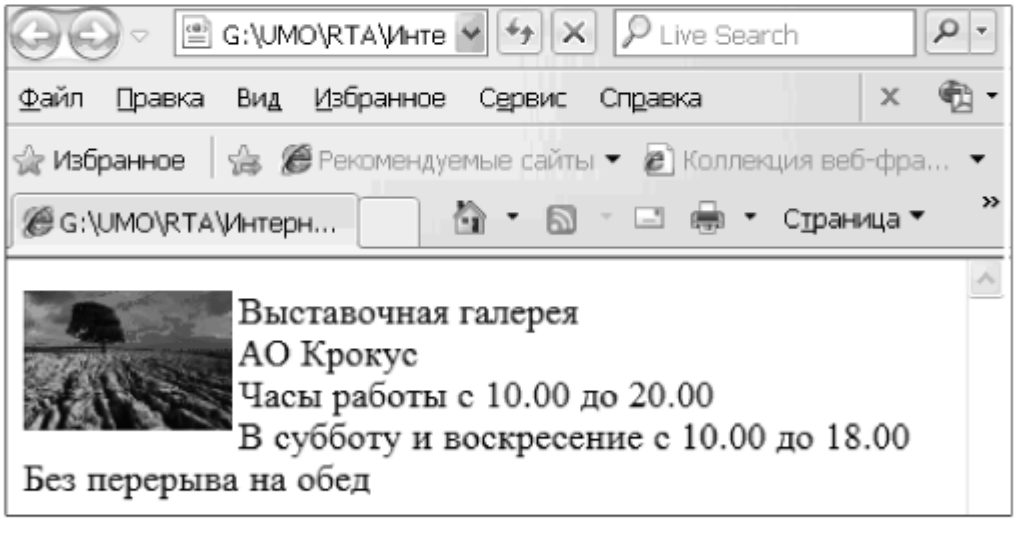
Документ XML содержит разметку:

```
<?xml version=»1.0» encoding=»windows-1251»?>
<?xml-stylesheet type=»text/css» href=»picture.css»?>
```

```
<doc>
     <picture/>
     <s>Выставочная галерея</s>
     <s>АО Крокус</s>
     <s>Часы работы с 10.00 до 20.00</s>
     <s>В субботу и воскресение с 10.00 до 18.00</s>
     <s>Без перерыва на обед</s>
</doc>
```

Стилевая таблица, хранящаяся в файле picture.CSS, имеет структуру:

```
s{display:block;
font-size:18px;
}
picture{background-image:url(739000.bmp);
background-repeat:no-repeat;
width:100px;
height:67px;
float:left;
}
```



Р и с . 2.8. Вывод графического изображения

Создавая разметку, можно обращаться к пространству имен HTML тегов:

```
<html:T
xmlns:html="http://www.w3c.org/TR/REC-html40/">
     содержание
</html:T>
```

Здесь T – требуемый тег.

Например, вывод разделительной линии при просмотре содержания XML документа можно выполнить тегом

```
<html:hr
xmlns:html=»http://www.w3c.org/TR/REC-html40/»>
</html:hr>
```

2.4.2. XSL-таблица стилей

Таблица стилей XSL (Extensible Stylesheet Language) – корректный документ XML, который определяет способы трансформации исходного документа с помощью специальных инструкций-тегов расширяемого языка стилевой разметки. Инструкции XSL служат для извлечения информации и передачи ее специальному процессору XSLT. Процессоры, входящие в состав браузера, преобразуют исходный XML-документ в HTML-страницу, которая отображается браузером. В составе XSL-документа могут находиться теги XHTML (расширяемого HTML). Написание тегов должно отвечать требованиям корректного XML-документа.

Примеры XHTML тегов:

```
<b>содержание</b>
<br/>
```

Для использования XSL-тегов необходимо объявить пространство имен

```
<xsl:stylesheet version=»1.0»
xmlns:xsl=»http://www.w3.org/1999/XSL/Transform»>
<!--теги XSL и XHTML-->
</xsl:stylesheet>
```

Просмотр информации требует задания корня документа с помощью тега:

```
<xsl:template match=»/»>
</xsl:template>
```

Тег должен содержать инструкции для вывода информации и теги XHTML. Извлечение информационного содержания элементов выполняется с помощью тега:

```
<xsl:value-of select=»path»/>
```

Здесь `path` – путь к требуемому элементу-тегу XML-документа, сформированный с учетом иерархии тегов. В качестве разделителя тегов иерархии служит символ «/».

Таблица XSL должна подсоединяться к XML-документу с помощью инструкции:

```
<?xml-stylesheet type=»text/xsl» href=»table.xsl»?>
```

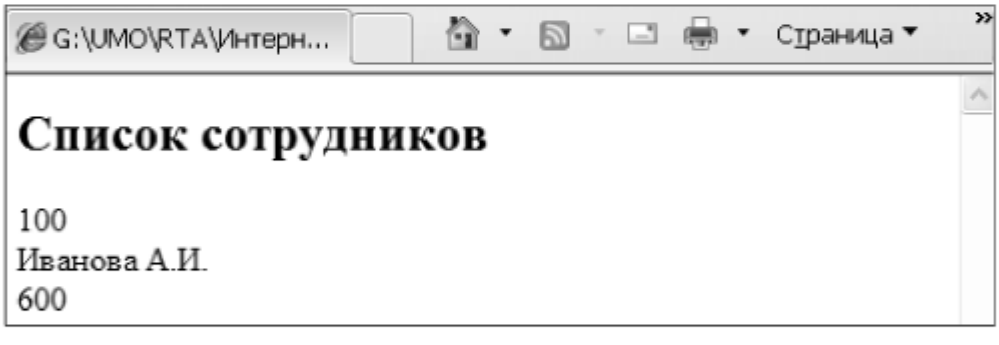
Пример. Рассмотрим XML-документ, в котором зафиксирована информация о служащих: личный номер, наименование отдела служащего, фамилия с инициалами и ставка в условных единицах.

```
<?xml version=»1.0» encoding=»windows-1251»?>
<?xml-stylesheet type=»text/xsl» href=»table.xsl»?>
<firma>
     <worker>
          <id dept=»бухгалтерия»>100</id>
          <name>Иванова А.И.</name>
          <pay>600</pay>
     </worker>
     <worker>
          <id dept=»бухгалтерия»>500</id>
          <name>Козырева А.П.</name>
          <pay>1000</pay>
     </worker>
     <worker>
          <id dept=»склад»>300</id>
          <name>Свиридова С.А</name>
          <pay>1500</pay>
     </worker>
</firma>
```

К документу присоединен XSL-файл table.xsl. В этом файле хранятся инструкции трансформации документа.

```
<?xml version=»1.0» encoding=»windows-1251»?>
<xsl:stylesheet version=»1.0»
xmlns:xsl=»http://www.w3.org/1999/XSL/Transform»>
   <xsl:template match=»/»>
      <html>
      <body>
         <h2>Список сотрудников </h2>
         <xsl:value-of select=»firma/worker/id»/><br/>
         <xsl:value-of select=»firma/worker/name»/><br/>
         <xsl:value-of select=»firma/worker/pay»/><br/>
      </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

Результат преобразования документа показан на рис. 2.9.



Р и с . 2.9. Трансформация документа

Как видно из рисунка, сформирован HTML-документ, который содержит информацию из первого блока данных, отмеченных тегом `worker`. Из рисунка видно также, что нет сведений об отделе, отсутствуют комментарии к извлеченной информации.

Просмотреть содержание всех узлов документа можно с помощью тега:

```
<xsl:for-each select=»path»>
</xsl:for-each>,
```

где path – путь к узлу, содержимое которого нужно вывести для всего документа. Внутри располагаются теги для извлечения информации из дочерних узлов. Данный тег представляет собой цикл перебора содержания узла, к которому указан путь.

Чтобы извлечь значение атрибуты тега, нужно указать к нему путь и снабдить специальным символом @ (ссылкой на атрибут).

Новый вариант XSL-документа примет вид:

```
<?xml version=»1.0» encoding=»windows-1251»?>
<xsl:stylesheet version=»1.0»
 xmlns:xsl=»http://www.w3.org/1999/XSL/Transform»>
     <xsl:template match=»/»>
     <html>
     <body>
          <h2>Список сотрудников </h2>
          <xsl:for-each select=»firma/worker»>
               Отдел:
               <xsl:value-of select=»id/@dept»/><br/>
               Личный номер:
               <xsl:value-of select=»id»/><br/>
               Фамилия:
               <xsl:value-of select=»name»/><br/>
               Зарпплата у.е:
               <xsl:value-of select=»pay»/><br/>
               <hr/>
          </xsl:for-each>
     </body>
     </html>
     </xsl:template>
</xsl:stylesheet>
```

Результат преобразования показан на рис. 2.10. Из рисунка видно, что из документа извлечена вся информация, которая в нем содержится.

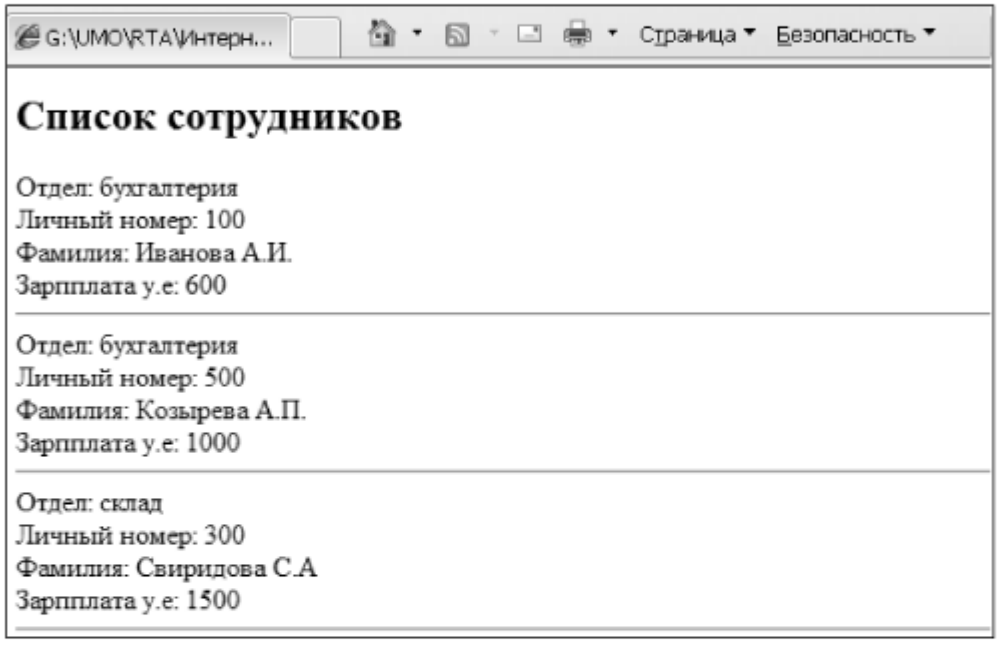
Технология использования трансформации документов позволяет выполнить ряд специальных действий над содержанием XML-документа. К ним относятся:

- отбор информации по условию;

- множественный выбор информации;
- сортировка;
- использование шаблонов.

Отбор информации по условию выполняется с помощью тега:

```
<xsl:if test="условие отбора">
      содержание
</xsl:if>
```



Р и с . 2.10. Вывод информации о служащих

Содержание тега обрабатывается, если выполнено условие. Например, набор тегов:

```
<xsl:if test="id/@dept='бухгалтерия'">
      Отдел:
      <xsl:value-of select=»id/@dept»/><br/>
      Личный номер:
      <xsl:value-of select=»id»/><br/>
      Фамилия:
      <xsl:value-of select=»name»/><br/>
      Зарплата у.е:
```

```
      <xsl:value-of select=»pay»/><br/>
      <hr/>
</xsl:if>
```

Выводит информацию о служащих бухгалтерии.

Для селекторного отбора информации используют теги:

```
<xsl:choose></xsl:choose>
<xsl:when test=»условие»> </xsl:when>
```

Во второй тег вкладывается первый. Дополнительно может быть использован тег `<xsl:otherwise></xsl:otherwise>`. Его содержание выполняется только в том случае, если не найдены соответствия в селекторах.

П р и м е р . Отбор сотрудников с личными номерами 100 и 500.

```
<xsl:choose>
      <xsl:when test=»id='100'»>
      Фамилия:
      <xsl:value-of select=»name»/><br/>
      <hr/>
      </xsl:when>
      <xsl:when test=»id='500'»>
      Фамилия:
      <xsl:value-of select=»name»/><br/>
      <hr/>
      </xsl:when>
<xsl:otherwise>
      <p></p><hr/>
</xsl:otherwise>
```

Если служащий не обнаружен, выводится пустая строка и разделительная линия.

Отбор по условию и селективный выбор должны производиться внутри тега `xsl:for-each`.

При выводе содержания XML его можно отсортировать-упорядочить с помощью тега `xsl:sort`. Тег должен отвечать следующему формату:

```
<xsl:sort select=″тег″ order=″type″/>
```

Здесь `тег` – наименование тега, по содержанию которого выполняется сортировка. Значение `type` определяет тип сортировки:

- `ascending` – по возрастанию, используется по умолчанию;
- `descending` – по убыванию.

Этот тег должен быть первым внутри тега `xsl:for-each`.

Пример. Сортировка содержания по убыванию личного номера сотрудника.

```
<xsl:for-each select=»firma/worker»>
      <xsl:sort select=»id» order=»descending»/>
      Личный номер:
      <xsl:value-of select=»id»/><br/>
      Фамилия:
      <xsl:value-of select=»name»/><br/>
      <hr/>
</xsl:for-each>
```

При использовании сортировки можно использовать дополнительный параметр `case-order` для учета регистра символов при сортировке:

- `lower-first:` сортировка по строчным буквам имеет больший приоритет;
- `upper-first:` сортировка по прописным буквам имеет больший приоритет.

Например, сортировка фамилий с учетом регистра:

```
<xsl:sort select=»name» case-order=»upper-first»/>
```

Числовые данные сортируются по умолчанию не как числа, а как строки символов с кодами символов цифр. С помощью параметра `data-type="number"` можно указать, что строки с числами должны сортироваться как числа:

```
<xsl:sort select=»id» data-type=»number»/>
```

В результате будет выполнена сортировка по числовому значению личного номера сотрудника.

При извлечении информации можно использовать шаблоны.

Шаблон задается один раз и может быть использован при разработке XSL-документа. Задается шаблон тегами:

```
<xsl:template match="имя">
содержание
</xsl:template>
```

Обращение к шаблону выполняется с помощью тега:

```
<xsl:apply-templates select=»path»/>
```

Здесь имя – это наименование тега документа XML, откуда будет произведено извлечение информации. Значение path задает путь в документе к тегу, указанному в имя.

Пример. Создадим шаблон для извлечения информации из тега worker и используем его в XSL таблице.

```
<xsl:template match=»worker»>
     Отдел:
     <xsl:value-of select=»id/@dept»/><br/>
     Личный номер:
     <xsl:value-of select=»id»/><br/>
     Фамилия:
     <xsl:value-of select=»name»/><br/>
     Оплата у.е:
     <xsl:value-of select=»pay»/><br/>
     <hr/>
</xsl:template>
<xsl:template match=»/»>
     <html>
     <body>
<h2>Список сотрудников </h2>
     <xsl:apply-templates select=»firma/worker»/>
     </body>
     </html>
</xsl:template>
```

Результат вывода будет таким же, как при использовании тега xsl:for-each. Шаблоны многократно применяются при выводе информации из документа XML.

Контрольные вопросы

1. Перечислите основные части XML-документа.
2. Дайте характеристику корректному XML-документу.
3. Перечислите ссылки на символы, используемые в XML.
4. Что понимается под валидностью XML-документа?
5. Как устроен шаблон DTD-документа?
6. Как задаются элементы в шаблоне DTD-документа?
7. Как определить и использовать сущность XML?
8. Что значит определить атрибут в шаблоне документа DTD?
9. Как выполнить преобразование содержания XML-документа?
10. Как устроена каскадная стилевая таблица?
11. Как используется каскадная стилевая таблица для отображения содержания документа?
12. Что такое XSL-документ и XSL-тег?
13. Как строится и используется XSL-документ?
14. Дайте классификацию XSL-тегов и укажите их особенности.
15. Как используются шаблоны XSL?

Глава 3

СЦЕНАРИИ И ИХ ИСПОЛЬЗОВАНИЕ

Под сценарием подразумевается программный код, который выполняется при обращении пользователя к информационному ресурсу. Существуют сценарии, которые выполняются на удаленном сервере сети Интернет, такие сценарии называются серверными. Сценарии, которые находятся внутри ресурса, полученного пользователем сети, называются клиентскими. Их выполняет программа пользователя сети.

В учебном пособии рассматриваются клиентские сценарии, написанные на языке JavaScript. Запуск таких сценариев может производиться с помощью формы HTML-страницы.

Форма представляет собой контейнер на странице, в который собраны элементы диалога между пользователем и сценарием.

В общем виде формы бывают, как уже говорилось, двух основных типов: серверные и клиентские. В первом случае обработку данных, полученных из формы, и возврат результатов выполняет серверная программа, расположенная на другом компьютере сети. Во втором случае программный код для работы с формой располагается с ней на одной странице. На рис. 3.1 показаны элементы диалога формы HTML.

Р и с . 3.1. Элементы диалога HTML-формы

1– текстовое поле, 2 – блок радиокнопок, 3 – ячейка, 4 – командная кнопка, 5 – список с однозначным выбором, 6 – список с многозначным выбором, 7 – текстовая область

Для создания формы служит тег:

```
<form name=»frmName» action=»URI» method=»Type»>
     <!- - Элементы диалога - ->
</form>,
```

где:

`frmName` – идентификатор формы;

`URI` – адрес серверной программы;

`Type` – тип взаимодействия с серверной программой (POST или GET).

Клиентская форма задается тегом аналогичной структуры, но без задания параметров `action` и `method`.

Следует отметить, что элементы диалога формы создаются тегами, и в языке HTML нет средств позиционирования элементов диалога в контейнере формы. Задачу позиционирования можно решить, если использовать таблицу HTML. В этом случае в ячейки таблицы, задаваемые тегом TD, помещают теги элементов диалога, а таблицу не очерчивают.

3.1. ЭЛЕМЕНТЫ ДИАЛОГА ФОРМЫ

3.1.1. Командная кнопка

Тег для размещения кнопки имеет вид:

```
<input type=»ButtonType» name=»cmdName»
value=»подпись»>
```

Здесь:

`ButtonType` – тип кнопки:

- `button` – кнопка, требующая написания кода для ее обработчика;
- `reset` – кнопка для сброса значений в элементах диалога;
- `submit` – кнопка для соединения с сервером;

`cmdName` – идентификатор кнопки;

`подпись` – надпись на кнопке.

При написании скрипта локальной обработки формы в тег создания кнопки добавляют обработчик события. Каждому событию на языке JavaScript поставлены в соответствие системные идентификаторы. При работе с формой пользователь делает щелчок мышью на кнопке. Такое событие называется `OnClick` и его обработчик будет иметь вид: `OnClick="functionName()"`. Здесь `functionName` – имя функции JavaScript, в которой записан код обработки события. Такую функцию

называют событийной функцией. При создании обработчиков идентификатор кнопкам можно не задавать и не использовать параметр name тега кнопки.

Событийная функция содержит код, написанный на языке JavaScript. Событийную функцию помещают внутри специального тега страницы SCRIPT в головной ее части. Формат тега:

```
<script language="JavaScript">
<! - - Код JavaScript - ->
</script>
```

Для задания функции используется следующее описание JavaScript:

```
function functionName(){
//Операторы
}
```

Среди операторов можно использовать оператор принудительного завершения работы функции return. Подробнее об операторах и особенностях языка JavaScript можно посмотреть в приложении 2.

Если требуется, то функция может принимать исходные данные для своей работы. Тогда она должно обладать формальными параметрами.

Пример 1. Создание формы с командной кнопкой. При ее нажатии выводится сообщение Hello World!, в системное окно браузера.

```
<html>
<head>
<script language=»JavaScript»>
     function fHello(){
          alert(«Hello World !»);
     }
     </script>
</head>
<body>
<form name=»frmGo»>
<input type=»Button» value=»GO» onClick=»fHello()»>
</form>
</body>
</html>
```

Пример 2. Изменим структуру функции, снабдив ее формальным параметром для вывода произвольного сообщения.

Код функции примет вид:

```
function fHello(mes){
     alert(mes);}
```

Тогда обращение к функции изменится:

```
<input type=»Button» value=»GO»
onClick=»fHello('Hello World !')»>
```

Фактическое значение для формального параметра `mes` будет строка `'Hello World !'`.

Пример 3. Текст сообщения вводится с помощью системного окна ввода, затем пользователь принимает решение о продолжении работы со страницей. Функция `fHello` примет вид:

```
function fHello(){
var mes="";
     mes=prompt("message","Hello World","Dialog");
     if (mes==null) mes="nothing";
     alert(mes);
     bDialog=confirm("Stop working?");
     if (bDialog) window.close();}
```

Если пользователь не ввел строку – получено значение `null`, тогда принимается значение строки `nothing`. Окно браузера закрывается методом `close` системного объекта `window`.

3.1.2. Текстовое поле

Элемент задается тегом:

```
<input type=»text» name=»txtName» size=»n»
maxlength=»m» value=»значение»>,
```

где:

`txtName` – идентификатор поля;

`n`* – длина поля в символах;

`m`* – максимальное число символов при вводе;

`значение`* – значение по умолчанию;

* – необязательные параметры.

Если изменить тип поля на `password`, то ввод в поле становится закрытым символом «*». Поле хранит введенную (выведенную) строку символов в свойстве `value`.

П р и м е р . Страница с формой из трех полей ввода. В первое поле пользователь вводит сумму валюты, во втором поле он может изменить курс или оставить значение по умолчанию, а в третье поле выводится результат в рублях.

```
<html>
<head>
<script language=»JavaScript»>
 function fCalc(){
     var forma=document.frmGo
     var s,kurs,itogo;
     s=forma.txtSumma.value;
     kurs=forma.txtKurs.value;
     s=parseFloat(s);
     kurs=parseFloat(kurs);
     itogo=kurs*s;
     itogo=Math.round(itogo*100)/100;
     forma.txtItog.value=itogo;
}
  </script>
</head>
<body>
<form name="frmGo">
<input type="text" name="txtSumma" value="0"><br>
<input type="text" name="txtKurs" value="30.95"><br>
<input type="text" name="txtItog" value="0"><br>
<input type="Button" value="GO" onClick="fCalc()">
</form>
</body>
</html>
```

Чтобы работать с элементами формы с помощью кода, нужно получить объект-указатель на форму с помощью оператора:

```
var forma=document.frmGo.
```

Идентификатор `forma` – экземпляр нашей формы.

Ссылка на форму выполняется с помощью объекта браузера `document` в виде `document.frmGo`. Здесь `frmGo` – имя нашей формы. Для обращения к текстовым полям формы служит оператор разъименования, который представляет собой точку. Так, оператор `s=forma.txtSumma.value;` означает запись в переменную `s` значения, хранящегося в текстовом поле. Из текстового поля поступает строка символов, их преобразование в число выполняется с помощью функции `parseFloat(s)`, `s` – строка с символами числа. Для округления полученного значения в рублях до второго знака после запятой служит метод `round` класса `Math` JavaScript (см. приложение 2).

При вводе пользователем чисел полезно проверять корректность ввода. Это можно сделать с помощью функции `isNaN(s)`. Если строка `s` хранит символы, пригодные для преобразования в число, то она возвращает логическое значение `false`, иначе `true`. Разделителем дробной части от целой части в языке JavaScript служит точка.

Перепишем нашу функцию, введя контроль над вводом данных.

```
function fCalc(){
     var forma=document.frmGo
     var s,kurs,itogo;
     s=forma.txtSumma.value;
     kurs=forma.txtKurs.value;
     if (isNaN(s)||isNaN(kurs)){
          alert(«Bad numbers input!»);
          return;}
     s=parseFloat(s);
     kurs=parseFloat(kurs);
     itogo=kurs*s;
     itogo=Math.round(itogo*100)/100;
     forma.txtItog.value=itogo;}
```

Теперь при неверном вводе курса или суммы будет выводиться сообщение об ошибке.

3.1.3. Текстовая область

Элемент диалога служит для вывода/ввода многострочного текста. Создается тегом:

```
<textarea name="areName" rows="r" cols="c">
</textarea>,
```

где:

`areName` – идентификатор элемента;

`r` – высота области в строках;

`c` – ширина области в символах.

Текст элемента хранится в свойстве `value`. При выводе текста для формирования строк в конец строки нужно включать ESC последовательность `\n`.

3.1.4. Флажки

Используются флажки для предоставления выбора пользователю данных из предопределенного набора значений. Флажок, используемый в блоках данных, с многозначным выбором называется ячейкой и создается тегом:

```
<input type=»checkbox» name=»chkName»>,
```

где:

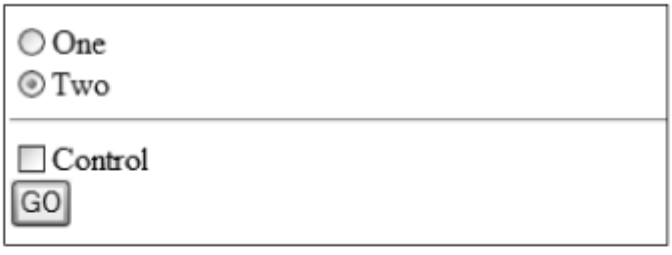
`chkName` – идентификатор флажка.

Для предварительной активизации флажка используется параметр `checked` (признак активности флажка – наличие символа «√» в его поле). Проверка состояния флажка и изменение его состояния в коде выполняется с помощью одноименного свойства `checked`. Значение `false` соответствует выключенному состоянию, а значение `true` – активному.

В блоках с однозначным выбором используется флажок радиокнопка. Создается аналогичным тегом, в качестве типа следует указать значение `radio`.

Чтобы создать блок с однозначным выбором, следует всем флажкам блока задать одинаковый идентификатор.

Пример. На рис. 3.2 показана страница с тремя элементами диалога, блок однозначного выбора с двумя радиокнопками, ячейка и командная кнопка для опроса состояния блока и ячейки. Результат опроса выводится в системное окно службы сообщений браузера.



Р и с . 3.2. Форма с флажками

Теги для создания формы примут вид:

```
<form name=»frmGo»>
<input type=»radio» name=»rdoBlock» >One<br>
<input type=»radio» name=»rdoBlock» checked>Two<br>
<hr>
<input type=»checkbox» name=»chkBox»
checked>Control<br>
<input type=»Button» value=»GO» onClick=»fState();»>
</form>
```

Код функции `fState` для события кнопки `onClick` будет иметь вид:

```
function fState(){
     var mes=»»;
     mes+=»Block action\n»;
var forma=document.frmGo
     if (forma.elements[0].checked){
         mes+=»One»;
     }
     if (forma.elements[1].checked){
         mes+=»Two»;
     }
     if (forma.chkBox.checked)
         mes+=»\nControl- OK»;
     else
         mes+=»\nControl- NO»;
     alert(mes);
}
```

Для опроса состояния блока кнопок используется коллекция `elements` формы. Она позволяет по индексу обратиться к любому элементу формы. Индексация ведется с нуля. Элементы диалога в коллекцию помещаются в порядке следования тегов их размещения в форме.

3.1.5. Списки

Элемент диалога позволяет пользователю выбрать значение из определенного набора данных. Пользователь может сделать однозначный выбор либо отобрать несколько значений в произвольном порядке.

Список с однозначным выбором создается тегами:

```
<select name=»lstName»>
<option value="значение1">строка1
<option value="значение2">строка2
...
</select>,
```

где:

`lstName` – идентификатор списка;

`строкаi` – текст элемента списка, выводимый на экран.

Список с многозначным выбором создается с помощью аналогичных тегов. В головной тег следует добавить ключевое слово `multiple`. Параметр `value` хранит символьное данное значение, связанное с конкретным элементом списка – строкой, на экране не отображается и является не обязательным.

При опросе списков однозначного выбора используется коллекция строк элементов списка `options`. Символы выбранной строки хранятся в свойстве `text`. Для доступа к строке выбранного элемента нужно указать ее индекс. Индексируются строки с нуля. Значение индекса выбранной строки хранится в свойстве `selectedIndex` коллекции. Доступ к значению строки осуществляется с помощью свойства `value`.

При многозначном выборе следует выполнить перебор всех строк коллекции и отобрать активные строки путем проверки свойства `selected` коллекции строк `options`. Для определения количества строк нужно использовать свойство `length` коллекции.

Пример 1. Требуется выполнить опрос списка с однозначным выбором. Нужно вывести выбранную строку списка и значение, связанное с этой строкой. Теги для создания формы:

```
<form name="frmGo">
```

```
<select name="lstMes" onChange="fState()">
<option value="YES!">message 1
<option value="NO!" selected>message 2
<option value="OK!">message 3
</select>
</form>
```

В списке три строки, вторая строка выбрана по умолчанию.

При выборе строки списка порождается событие `onChange` и выполняется функция `fState`, в которой написан код опроса списка.

Код функции:

```
function fState(){
     var mes="";
     var idx;
     var forma=document.frmGo;
     idx=forma.lstMes.selectedIndex;
     mes=forma.lstMes.options[idx].text
     +"\n"+forma.lstMes.options[idx].value
     alert(mes);
}
```

Результат накапливается в символьную переменную `mes` и ее значение выводится службой сообщений браузера.

Пример 2. Требуется опросить список из примера номер 1. Список многозначный. Результат опроса выводится в текстовую область формы (рис. 3.3).

Теги для создания формы примут вид:

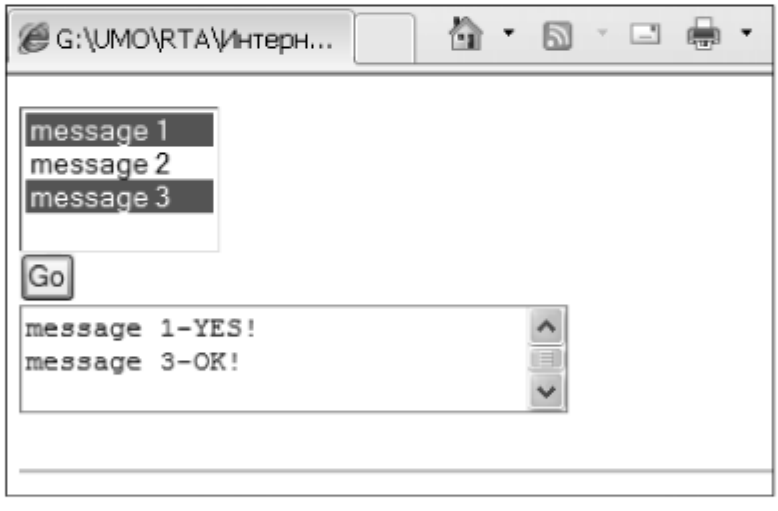
```
<form name=»frmGo»>
<select name=»lstMes» multiple>
<option value=»YES!»>message 1
<option value=»NO!»>message 2
<option value=»OK!»>message 3
</select><br>
<input type=»button» value=»Go»
onClick=»fState()»><br>
```

```
<textarea name=»display» rows=»3» cols=»30»>
</textarea>
</form>
```

Опрос производится при нажатии командной кнопки.

Код функции опроса:

```
function fState(){
     var mes=»»;
     var forma=document.frmGo;
     for (i=0;i<forma.lstMes.length;i++){
          if (forma.lstMes.options[i].selected){
          mes+=forma.lstMes.options[i].text+»-»
          +forma.lstMes.options[i].value+»\n»;}//if
          }//for
          forma.display.value=mes;
     }
```



Р и с . 3.3. Опрос списка многозначного выбора

Вывод происходит в текстовую область `display`. На рис. 3.3 показан результат работы функции. Выбор в списке производится щелчками мыши при нажатой клавише CTRL. Можно отметить первый элемент списка при нажатой клавише CTRL, а последний элемент выбрать при нажатой клавише SHIFT. В результате будет выделено сразу несколько строк списка.

Контрольные вопросы

1. Какие типы сценариев используются в сети Интернет?
2. Дайте классификацию элементов диалога в клиентских HTML-формах.
3. Как создать клиентскую форму HTML?
4. Что такое событие JavaScript и событийная процедура?
5. Как программируется реакция при нажатии кнопки формы?
6. Какую структуру имеет страница при размещении скрипта обработки формы?
7. Как получить доступ в коде JavaScript к форме и ее элементам диалога?
8. Какие типы текстовых полей используются при создании форм?
9. В чем разница между текстовым полем и текстовой областью формы?
10. Как программируется опрос блока однозначного выбора формы?
11. Как программируется опрос блока многозначного выбора формы?
12. Как опрашивается список однозначного выбора формы?
13. Как опрашивается список многозначного выбора формы?

Приложение 1

Стилевые параметры

Стилевые параметры применяются к конкретному тегу XML. Тег документа образует узел с определенным содержанием.

Если используются параметры, значением которых является некоторый размер – число, то это число задается в виде `Ne`, где `N` – число, `e` – обозначение единицы измерения. Единицы измерения могут быть:

`px` – пикселы;

`em` – высота текущего шрифта, используемого в узле;

`ex` – высота строчной буквы x текущего шрифта.

Например: `24px` – 24 пиксела.

Стилевые параметры разделяют на группы.

Первая группа параметров. Эти параметры служат для оформления узла, они даны в табл. 1.1.

Таблица 1.1

Оформление узла

Параметр	Значение
`background-image`	Загрузка изображения: `url(path)`. Здесь `path` – путь к файлу изображения
`background-color`	Цвет фона узла. Это системная цветовая константа как в HTML либо шестнадцатеричное число `#rrggbb` – код цвета в формате RGB
`background-repeat`	Способ вывода изображения: `repeat` (повторять), `repeat-x` (повторять горизонтально), `repeat-y` (повторять вертикально), `no-repeat` (не повторять)
`background-position`	Выравнивание изображения: `left, right, top, bottom, center` (рис. 1)
`width`	Ширина области вывода изображения
`height`	Высота области вывода изображения

Размещение изображения в узле выполняется с указанием его выравнивания. На рис. показаны способы выравнивания по полю узла.

left top (по умолчанию)	center top или top	right top
left	center	right
left bottom	center bottom или bottom	right bottom

Р и с . Выравнивание по полю узла

Вторая группа – это параметры для работы с текстом. Они показаны в табл. 1.2.

Таблица 1.2

Оформление текста

Параметр	Значение
display	Отображение текста: block (отдельный фрагмент, строка), none (отмена действия)
font-family	Тип шрифта
font-size	Размер шрифта
font-style	Начертание текста: normal (обычный), italic (курсивный), oblique (рукописный)
font-weight	Яркость шрифта: normal (обычный), bold (жирный), bolder(усиление яркости); lighter (максимальная яркость) либо диапазон 100–900. Выбранное значение должно быть кратно 100
color	Цвет символов

Окончание таблицы 1.2

Параметр	Значение
text-transform	Регистр символов: uppercase (прописные буквы), lowercase (строчные буквы), capitalize (все первые буквы слов прописные)
text-align	Выравнивание текста в узле: left, right, center
text-decoration	Оформление текста: underline (подстрочный), overline (надстрочный), line-through (перечеркнутый)

Третья группа параметров позволяет оформлять текстовые области.

С их помощью определяют обрамление, отступы, характер обтекания текстом узла и выравнивание по границе окна программы просмотра. Эти параметры даны в табл. 1.3.

Таблица 1.3

Оформление областей

Параметр	Значения
border-style	Тип рамки: solid, double, none
border-width	Толщина рамки: thin, medium, thick или размер
border-color	Цвет рамки
padding-top	Просвет между рамкой и окружающим ее текстом сверху
padding-bottom	Просвет между рамкой и окружающим ее текстом снизу
padding-left	Просвет между рамкой и окружающим ее текстом слева
padding-right	Просвет между рамкой и окружающим ее текстом справа
padding	Просвет между рамкой и окружающим ее текстом
float	Расположение элемента относительно другого: left, right, none
clear	Отмена действия float: left, right
margin-left	Левый отступ от окна программы
margin-right	Правый отступ от окна программы
margin-bottom	Отступ снизу от окна программы
margin-top	Отступ сверху от окна программы

Основы языка JavaScript

Особенности языка программирования

Язык создания скриптов JavaScript требует при написании кода учитывать регистр символов в операторах и выражениях.

При написании кода разработчик может использовать данные различных типов. Существуют следующие типы данных:

– числовой;
– булевый – логический;
– строковый;
– неопределенный – нулевой.

Примеры данных:

– числовые: 3.14, 200;
– логические: `true`, `false`;
– строка символов: «Привет Мир !»
– неопределенное значение: `null`.

При создании кода скрипта допускается использовать комментарии. Текст комментария должен начинаться с пары символов //. В этом случае можно задать одну строку комментария. Для ввода нескольких строк комментария используются скобки `/* */`.

Примеры:

```
//JavaScript
/*JavaScript
Локальный сценарий
*/
```

Явно тип переменных не задается, но допускается явно вводить переменные:

```
var a;//явное задание
a="Привет Мир!";
PI=3.14;//Неявное задание
```

Тип переменной зависит от значения, которое хранится в ней в текущий момент.

Разработчик может использовать функции преобразования к типам:

`parseInt(s)`	Преобразует значение из строки `s` в число целого типа
`parseFloat(s)`	Преобразует значение из строки `s` в число вещественного типа

Для проверки наличия в строке символов числа служит функция `isNaN(s)`. Если строка содержит символы числа возвращается значение `false`.

При программировании допускается использовать массивы. Массив – именованная последовательность индексированных однотипных данных. Создается массив с помощью конструктора:

`v = new Array(n). n - число элементов.`

Индексирование массива производится с нуля. Для обращения к элементам массива служат скобки []. Например, `v[1]=50` – это запись числа во второй элемент массива.

Массив представляет собой объект, который обладает методами и свойствами. Например:

`length` – свойство. Возвращает число элементов в массиве.

`sort()` – метод. Служит для сортировки элементов массива. В результате создается новый массив.

П р и м е р. Дан массив из пяти латинских прописных букв G, A, K, Z, L. Требуется его отсортировать, а результат поместить в строку.

```
sym=new Array(5);
mes=»»;//Строка для приема символов
sym[0]=»G»;sym[1]=»A»;
sym[2]=»K»;sym[3]=»Z»;
sym[4]=»L»;
sortA=sym.sort();//Отсортировать символы
//Вывести сортированный массив символов:
for (i=0;i<sortA.length;i++){
mes+=sortA[i]+» «}
```

После вызова метода `sort` создается новый массив символов, помещаемый в массив `sortA`. Затем элементы массива в цикле записываются в символьную переменную `mes`. Для этого используется присваивание с накоплением и оператор цикла с заданным числом шагов (см. ниже).

Основные операции

Арифметические операции: *, /, %, +, – (умножение, деление, деление с выделением остатка, сложение, вычитание).

Операции сравнения: <,<=,>,>=,==,!= (меньше, меньше либо равно, больше, больше либо равно, равно, не равно).

Логические операции: ||, &&, ! (или, и, отрицание).

Инкрементирование (увеличение на единицу операнда) ++.

Декрементирование (уменьшение на единицу операнда) – –.

Операции инкрементирования и декрементирования бывают префиксные и постфиксные. Префиксные операции выполняются в первую очередь.

Пример:

```
x=10                    x=10
y=++x; //y=11           y=x++;//y=10
```

На основе операций строятся выражения. Выражение возвращает определенный результат. Операции в выражении выполняются с учетом приоритета, для изменения приоритета используются круглые скобки. Наибольшим приоритетом обладают операции умножения и деления.

Общая форма выражения имеет вид: `v=выражение.`

Здесь `v` – переменная, получившая результат выражения.

Выражение может заканчиваться символом «;». Наличие этого символа обязательно только в том случае, если выражение переносится на другую строку.

Сохранить результат выражения можно с помощью оператора присваивания. В JavaScript можно использовать различные его виды:

= (простое присваивание);

*= (накопление с умножением);

/= (накопление с делением);

-= (накопление с вычитанием);

+= (накопление со сложением).

Пример:

```
x=20;
x+=10//x=30
x*=10//x=300.
```

Системные средства ввода/вывода

Для использования системных средств ввода/вывода браузера служат операторы `alert`, `confirm`, `prompt`.

Вывод системного окна с текстом: `alert("Текст");`

Для создания текста сообщения можно использовать операцию конкатенации строк +, переход на следующую строку требует включения в текст ESC кода `\n`. При соединении со строчкой числа оно автоматически преобразуется к строковому типу.

Примеры:

```
alert(«Hello World\nПривет Мир !»);
alert(«x=»+Math.sin(x));
```

Вывод окна для диалога с пользователем: `confirm("Текст")`.

Если нажата кнопка OK в окне, то возвращается значение `true` (истинна), иначе `false` (ложь) .

Запрос строки текста из системного окна:

```
v=prompt("подпись к текстовому полю",
"значение по умолчанию","Заголовок окна");
```

Возвращает оператор введенную строку в переменную `v`.

Основные операторы

Операторы используются для выполнения определенных алгоритмических действий. Для выполнения группы выражений в операторе используется пара скобок `{}` – блок.

Ветвление

Оператор ветвления позволяет изменить последовательность выполнения кода скрипта. Общая форма оператора имеет вид:

```
if (expr) {
//операторы
}else {
//операторы
}
```

Здесь `expr` – логическое выражение, построенное с помощью операций сравнения. Логическое выражение можно построить, используя также логические операции.

Пример 1. Даны два числа x1 и x2, требуется найти максимальное число.

```
if (x1>x2) {xmax=x1;}
else {xmax=x2;}
```

Пример 2. Требуется проверить, находится ли число x в интервале [1..40].

```
if ((x>=1) && (x<=40)) {bFlag=true;}
else {bFlag=false;}
```

Здесь bFlag – логическая переменная.

При использовании логических операций условия должны быть взяты в пару скобок (), так как логические операции имеют больший приоритет, чем операции сравнения.

Оператор множественного выбора

Оператор позволяет выбрать определенный блок case в зависимости от значения селектора.

```
switch(селектор){
case n1:{
//операторы
break
}
case n2:{
//операторы
break
}
...
default:{
//операторы
}}
```

Выполняется тот блок, параметр которого n_i совпадает со значением селектора. Блок должен заканчиваться оператором break для передачи управления оператору, который следует за оператором case. В операторе выбора можно использовать не обязательный блок default, который выполняется, если селектор не совпал ни с одним ni. Обычно в качестве селектора и значений ni используют целые числа или строки.

Пример. Выбор названия весеннего месяца по его номеру:

```
var m=»»;
switch (n){
     case 3:{
          m=»march»
          break
}
     case 4:{
          m=»april»
          break
}
     case 5:{
          m=»may»
          break
}
     default:{
          m=»nothing»
}}
```

Циклы

Цикл позволяет выполнить определенный оператор или группа операторов в скобках {} несколько раз. Существует три типа циклов, показанных в табл. 2.1.

Таблица 2.1

Типы циклов

Цикл с постусловием:	Цикл с заданным числом шагов:	Цикл с предусловием:

`do`
`//операторы`
`while (expr)`
	`for (n1,n2,n3) {`
	`// операторы`
	`}`
		`wile(expr){`
		`// операторы`
		`}`

Здесь:

`expr` – логическое выражение (условие).

`n1` – начальное значение счетчика цикла;

`n2` – условие выполнения цикла;

`n3` – выражения для приращения счетчика цикла.

Цикл с постусловием выполняется до тех пор, пока условие истинно и выполняется минимум один раз.

Пример.

```
x=0;
do
      x+=10;
while (x<=100)
```

Цикл с заданным числом шагов:

```
x=1;
for (i=1,i<=100,i++) {
      x*=i;
}
```

Цикл с предусловием аналогичен первому циклу, но условие проверяется в начале. Он может не выполниться ни одного раза.

```
x=0;
while (x<=100){
      x+=10;
}
```

Математические встроенные функции

При использовании в операторах математических функций для их вызова следует использовать класс `Math`. В общем виде обращение к функции имеет вид: `Math.f(d);`

где:

`f` – название функции;

`d` – аргумент;

Основные функции приведены в табл. 2.2.

Таблица 2.2

Математические методы

Метод (функция)	Назначение
`abs(x)`	Модуль
`sin(x)`	Синус
`cos(x)`	Косинус

Окончание таблицы 2.2

Метод (функция)	Назначение
`exp(x)`	Натуральное число в степени x (e^x)
`max(x1,x2)`	Определения максимума из двух чисел
`min(x1,x2)`	Определение минимума из двух чисел
`pow(x,y)`	x^y
`sqrt(x)`	Корень квадратный
`ceil(x)`	Округление «вверх»
`floor(x)`	Округление «вниз»
`round(x)`	Округление до ближайшего целого
`random()`	Случайное от 0 до 1

Пример. Требуется получить случайное число из диапазона от 1 до 10:

```
var a=1,b=10;
var n;
n=(b-a)*Math.random()+a;
```

Обработка строк

Строка задается в виде набора символов:

```
var mes="Hello World !"
```

Строка представляет собой объект, который имеет свойство `length`, которое хранит общее количество символов в строке. Индексирование символов в строке выполняется с ноля. Для выполнения действий над символами используются методы, показанные в табл. 2.3.

Таблица 2.3

Строковые методы

Метод	Назначение
`indexOf(s)`	Отыскание вхождения подстроки `s` в строке. Возвращает позицию вхождения либо –1
`split(«c»)`	Расщепление строки по заданному символу. В результате создается массив
`toUpperCase(s)`	Перевод символов строки `s` к верхнему регистру

Окончание таблицы 2.3

Метод	Назначение
`toLowerCase(s)`	Перевод символов строки `s` к нижнему регистру
`substring(n1,n2)`	Выделение подстроки. `n1` – стартовая позиция, `n2` – количество выделяемых символов, включая стартовую позицию

Пример. Поиск символа в строке:

```
var mes="Hello World !";
pos=mes.indexOf("W")
if (pos!=-1) {alert("OK !")}
else {alert("Error !")}
```

Работа с календарем

Для получения значений даты и времени необходимо создать экземпляр системного класса `Date`

```
d=new Date()
```

После создания объекта доступны методы, приведенные в табл. 2.4.

Таблица 2.4

Методы работы с датой и временем

Метод	Назначение
`toString()`	Преобразования в строку показаний календаря и часов
`toGMTString()`	Получение показаний системных часов по Гринвичу
`getHours()`	Получение часа
`getMinutes()`	Получение минут
`getSeconds()`	Получение секунд
`getMonth()`	Получение номера месяца: 0 – январь, 11-декабрь.
`getDay()`	Получение дня недели: 0 – воскресение, 6 – суббота.
`getDate()`	Текущая дата: от 1 до 31
`getYear()`	Возвращает год объекта минус 1900

Пример. Получение текущих показаний таймера компьютера пользователя:

```
var mes=»»;
myDate=new Date();
alert(myDate.toString());
```

Создание объектов

В JavaScript разработчик может создавать объекты и использовать их экземпляры в программном коде. Для получения экземпляра объекта используется оператор `new`.

Пример создания объекта:

```
function Person(name,bday) {
     this.name=name;//Имя
     this.bday=bday;//Дата рождения
}
function get(){
     p1=new Person(«Olga»,»12.04.87»)
     alert(p1.name+" "+p1.bday) }
```

ЗАКЛЮЧЕНИЕ

Изучение учебного пособия «Основы работы в сети Интернет» позволит студентам получить теоретические навыки и практические знания, используя которые, можно:

– обращаться к ресурсам сети Интернет;

– создавать страницы и сайты;

– разрабатывать корректные и валидные XML-документы;

– выполнять извлечение и преобразование содержания XML-документов;

– создавать и использовать клиентские формы HTML.

В учебное пособие включены два приложения.

В первом приложении собраны сведения, позволяющие использовать каскадные стилевые таблицы CSS для решения различных задач, возникающих при отображении информации, хранящейся в XML-документе.

Второе приложение содержит сведения по языку программирования сценариев JavaScript в объеме, необходимом для решения задач, связанных с программированием реакции при взаимодействии пользователя HTML страницы с клиентской формой.

Студенты после изучения учебного пособия обязаны:

– усвоить полный объем программного материала и излагать его на высоком научном уровне;

– изучить литературу к курсу и использовать ее при ответах;

– свободно владеть методологией дисциплины, свободно излагать основные понятия дисциплины;

– уметь творчески применить теоретические знания при решении практических задач, используя ЭВМ и современные методы исследования;

– показать способность самостоятельно пополнять и обновлять знания в процессе дальнейшей учебы и профессиональной деятельности.

Список литературы

1. *Баррет Д., Ливингстон Д., Браун М.* JavaScript. Web – профессионалам. К.: Издательская группа BHV, 2001. 240 с.
2. *Васютин С.* Базы данных. Интеллектуальная обработка информации / С. Васютин, А. Гареев, В. Корнеев, В. Райх (2-е изд.). М.: Нолидж, 2003. 494 с.
3. *Дариелл Р.* JavaScript: Справочник. СПб.: Питер, 2001. 192 с.
4. *Дмитриева М.В.* Самоучитель JavaScript. СПб.: БХВ – Петербург, 2003. 512 с.
5. *Максимов Н.В., Попов И.И.* Компьютерные сети: учебное пособие. М.: ФОРУМ: ИНФРА – М, 2004. 336 с.
6. *Рэй Э.* Изучаем XML. СПб.: Символ – Плюс, 2001. 408 с.
7. *Сергеев А.П.* HTML и XML профессиональная работа. М.: Издательский дом «Вильямс», 2004. 880 с.
8. *Янг М.* XML. Шаг за шагом. М.: Изд-во ЭКОМ, 2002. 384 с.

Содержание

Константин Николаевич МЕЗЕНЦЕВ

Основы работы в сети Интернет

УЧЕБНОЕ ПОСОБИЕ

по дисциплине

«Основы Интернета»

Редактор
Н.А. Панкратова

Макетирование и верстка
Л.А. Бутузовой

Дизайн обложки
Н.С. Тресковой

Подписано в печать 09.02.2012 г.
Формат 70×100/16. Усл. печ. л. 6,5.
Тираж 80 экз. Изд. № 151. Заказ № 029.

Изд-во Российской таможенной академии,
140009, г. Люберцы Московской обл.,
Комсомольский пр., 4.