МИНИСТЕРСТВО РАЗВИТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

Ташкентский университет информационных технологий

Факультет «Программный инжиниринг»

РАЗРАБОТКА WEB ПРИЛОЖЕНИЙ

Учебное пособие

Ташкент – 2017

Автор(ы): К.Ф. Керимов, И.В. Хан, Ш.Ш. Мухсинов. "Разработка web приложений". Учебное пособие /ТУИТ. 298 с. Ташкент, 2015

Данное учебное пособие предназначено для студентов высших учебных заведений подгатавливаемых в области образования компьютерные технологии и информатика.

Учебная дисциплина "Разработка web приложений" является одной из специальных дисциплин в учебной программе подготовки бакалавров по направлению образований "5330500 – Компьютерный инжиниринг" и "5330600 – Программный инжиниринг", который изучается на третьем курсе.

После изучения данного учебного курса студенты смогут самостоятельно разрабатывать веб-приложения малой и средней сложности и быть готовыми к изучению более сложных технологических методов и приемов разработки сложных веб-приложений.

Напечатано на основе утверждения учебно-методическим советом Ташкентского Университета Информационных Технологий

Рецензенты:

Акбаралиев Б.Б. – ТУИТ, кафедра «Программное обеспечение информационных технологий», к.т.н., заведующий кафедрой

Худайбердиев М.Х. – Центр разработки программных продуктов и аппаратно – программных комплексов при ТУИТ, к.т.н., директор

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Настоящее пособие предназначено в первую очередь для студентов ТУИТ, изучающих основные предметы информационно-коммуникационных технологий. Особенностью данного пособия является мультимедийное сопровождение практических занятий, которое будет играть роль электронного наставника, показывающего как научиться выполнять тот или иной прием процесса веб-разработки. В виду такой особенности, изучение основ веб-разработки лучше начать с самого начала, однако, студенты имеющие знания и навыки могут пропустить главы, изучение которых, по их мнению, будет напрасной тратой времени.

Предполагается, что после изучения настоящего курса студенты смогут самостоятельно разрабатывать веб-приложения малой и средней сложности и быть готовыми к изучению более сложных технологических методов и приемов разработки сложных веб-приложений.

При разработке пособия авторы придерживались «принципа Паретто» : 20 процентов первоначально передаваемых студентам знаний могут решать 80 процентов задач веб-разработчика. И, как уже отмечалось, главная цель мультимедийного сопровождения настоящего пособия заключается в реализации принципа «делай как я».

Ниже представим кратко содержимое пособия, охарактеризовав каждую главу.

Пособие состоит из трех частей, разбитых на 15 глав.

Часть 1 Введение в веб-разработку

Глава 1. Введение в разработку приложений для Интернета.

Глава 2. Типы веб-приложений и платформы для их разработки.

Глава 3. Системное программное обеспечение интернет-приложений (веб-сервера).

Глава 4. Основы Интернет: протоколы HTTP , URL, HTML. Другие протоколы и стандартизация.

Часть 2 Разработка клиентской части веб-приложения

Глава 5. HTML5 – язык гипертекстовой разметки пятой версии.

Глава 6. Табличная и контейнерная разметки.

Глава 7. Программирование форм и элементов форм.

Глава 8. Объектная модель документа.

Глава 9. CSS3 – Каскадные таблицы стилей.

Глава 10. Программирование на языке JavaScript.

Глава 11. Использование фреймворков JavaScript (JQuery и т.д.).

Глава 12. Программирование насыщенных интернет приложений (RIA)

Часть 3 Разработка на стороне сервера

Глава 13. Программирование на стороне сервера.

Глава 14. Программирование веб-приложений с использованием систем управления контентом и фреймворков.

Глава 15. Публикации веб-сайта, вопросы администрирования и модерации содержимого.

Часть 1 Введение в веб-разработку состоит из 4 глав. В **главе 1** рассматриваются основные технологии, применяемые в Интернет, сходства и различия процессов разработки прикладных программ и веб-приложени. В **главе 2** дается обзор основных типов веб-приложений и особенностей технологических приемов при их разработке. Также рассматриваются платформы для разработки веб-приложений от использования простых текстовых редакторов и веб-серверных сборок, до интегрированных систем с вспомогательными интеллектуальными средствами разработки и отладки. **Глава 3** посвящена вопросам установки и настройки системного программного обеспечения – веб-серверов, интерпретаторов языков программирования, серверов и других приложений баз данных. В **главе 4** приводится общий обзор применяемых протоколов, объясняется принцип отработки запросов пользователей и ответов серверов.

В части 2, которая состоит из 8 глав, рассматриваются аспекты разработки веб-приложений на стороне клиента. В **главе 5** приводится информация о гипертекстовом языке разметки, с акцентом на новинки реализованные в пятой версии. **Глава 6** сосредоточена на сравнении различных методов верстки с использованием языка гипертекстовой разметки. Описаны достоинства и недостатки обоих методов, приводятся рекомендации по их использованию. **Глава 7** посвящена изучению основ верстки – программированию форм и элементов управления форм. **Глава 8** вводит понятие объектной модели документа и объясняет почему важно использовать эту модель. Оформление веб-страниц невозможно представить без использования каскадных таблиц стиля. Описанием их применения занята **Глава 9**. **Главы 10** и **11** посвящены языку JavaScript – основному средству обеспечения интерактивности веб-приложений на стороне клиента. **Глава 12** посвящена вопросам разработки насыщенных веб-приложений, по большей части это относится к мультимедийной составляющей веб-приложений. Это может быть реализовано как с использованием вставок специальных объектов (Flash, Flex, Java-applet etc.) на веб-страницу, так и с использованием стандартных средств HTML5, CSS3 и JavaScript с его библиотеками.

Часть 3 Посвящена разработке веб-приложений на стороне сервера, и состоит из 3 глав. Глава 13 рассматривает общие вопросы программирования на стороне сервера. Приводит краткое описание возможных платформ и технологий. Подробно рассмотрено использование языка веб-программирования PHP и программного обеспечения баз данных MySQL. В глава 14 рассматриваются современные средства разработки веб-приложений на основе широкого использования шаблонов программирования, реализованных в различных сборках: системах управления контентом и фреймворках. Глава 15 посвящена вопросам публикации разработанного веб-сайта, администрирования и модерации содержимого веб-сайта.

Минимальные и оптимальные требования к техническому обеспечению

Веб-разработка – это наиболее демократичное занятие, поскольку требования к используемому для нее оборудованию выполнимы практически на всех используемых компьютерах. Например, для начала достаточно только наличие браузера и текстового редактора. Т.е. простые веб-страницы можно формировать даже на мобильных устройствах с самой простой операционной системой, главное чтобы там был текстовый редактор и браузер. Для более продвинутой работы – разработку серверной части – необхродимы уже веб-сервер , интерпретатор и СУБД. Все это, как правило, доступно через использование многочисленных LAMP или WAMP сборок (веб-сервер, интерпретатор PHP, сервер баз данных) – денвер, XAMPP, vertrigo, WAMP и многих (более 50) других.

Минимальные требования к знаниям студентов.

Поскольку пособие пишется для студентов Ташкентского университета информационных технологий, то предполагается, что студенты обладают навыками работы на персональном компьютере: с клавиатурой, текстовыми редакторами, интегрированными оболочками, браузерами. Также предполагается, что студенты знакомы с архитектурой современных компьютеров и средств связи, а также имеют элементарные знания о хранящейся и обрабатываемой информации и ее представлении в компьютере. Кроме того, желательно горячее стремление к самостоятельному изучению учебных дисциплин. Без такого отношения к учебе и работе не помогут никакие мультимедийные видеоуроки.

ЧАСТЬ 1. ВВЕДЕНИЕ В ВЕБ-РАЗРАБОТКУ

ГЛАВА 1. ВВЕДЕНИЕ В РАЗРАБОТКУ ПРИЛОЖЕНИЙ ДЛЯ ИНТЕРНЕТА

Настоящий курс посвящен процессам разработки приложений в сети и для сети. В этом смысле термины Интернет и веб (или world wide web) будем считать синонимичными. Вообще говоря, приложения, которые мы будем рассматривать в настоящем пособии с таким же успехом могут использоваться и в локальной сети.

Процессы разработки приложений для Интернета (или веб-разработки) хотя и имеют много схожего с процессами разработки настольных приложений, отличаются от них некоторыми особенностями, которые и будут предметом нашего рассмотрения в этой и последующих главах пособия.

К таким особенностям относятся следующие:

1. Все веб-приложения предназначены для использования в сети.
2. Следствием первой особенности является необходимость использования стандартов и интерфейсов, к которым относятся стандарты протоколов передачи данных (HTTP, FTP, URI, URL и др.), необходимость использования стандартных языков и средств разметки (HTML, CSS, XML)
3. Учет архитектурных решений: необходимость использования архитектур распределенных систем (клиент-сервер, мобильные агенты, P2P и др). т.е для проверки работоспособности веб-приложений необходимо иметь реально работающую сеть, либо полную ее имитацию. При этом, окончательное решение о приеме-сдаче веб-приложения в эксплуатацию принимается после апробации на реальном сегменте Интернета.

Таким образом, в отличие от традиционных настольных приложений, где, порой, бывает достаточно знаний одного языка (C, C++, C#, Java, Delphi), при разработке веб-приложений необходимо уверенное знание как минимум 4 языков: HTML, CSS, JavaScript и далее на выбор PHP, Perl, C#, Java, Python и еще много других (XML, XSL, XSLT, XPATH, SQL etc.).

Прежде чем перейти к дальнейшему изложению материала приведем перечень некоторых терминологических пояснений, которые были взяты из открытых источников Интернета.

Фундаментальные концепции

Понимание терминологии, составляющей существо этих технологий является необходимым для успешного изучения дисциплины нашего курса, а именно веб-разработки. Приведем список основных терминов и концепций:

- **Hypertext** позволяет словам (или другим объектам) в одном документе быть связанными с другими документами. Он придает динамический смысл организации и доступу информации, где страницы информации связаны вместе посредством гипертекстовых ссылок. **Hyperlink**

(Hypertext Link) может быть текстом или картинкой, которые ассоциируются с размещением (путем или именем файла) другого документа и использоваться в качестве ссылки на этот другой документ. Эти документы называются **web pages** и могут содержать текст, графику, video и audio а также гиперссылки (hyperlinks). А **website** это набор связанных страниц, хранящихся на **web server**. Web server сделан из компьютерных частей и программного обеспечения, на базе PC или другого большего компьютера. Но прежде всего **web server** – это компьютерная программа (примеры: Apache, IIS, Tomcat)

- **URL** (Uniform Resource Locator – Универсальный локатор ресурсов) задает адрес или расположение любого специфического сайта. Каждый URL определяет путь, который будет передавать документ, используемый протокол Internet, и сервер на котором расположен website. Каждый адрес Internet преобразуется в последовательность чисел, называемую **IP address**. **Доменное имя (domain name)** используется организационным объектом, чтобы идентифицировать его сайт и базируется на иерархии системы доменных имен (**Domain Name System** (**DNS**)).
- **Web браузер** – это программа используемая для получения доступа к веб страницам и просмотра их. Наиболее известными web браузерами являются: Google Chrome, Internet Explorer, Firefox, Safari, Opera и Netscape Navigator.
- **Коммуникационная модель (Communications Model):** Коммуникация включает в себя отправителя посылающего некоторую форму сообщения на приемник. Это происходит в конкретном контексте или установке, и включает в себя передачу некоторой формы информации от отправителя к приемнику по некоторому типу канала связи (Davison et al., 2008). Сообщение кодируется в соответствующую форму отправителем перед передачей, и позднее декодируется принимающей стороной. Обратная связь позволяет отправителю знать, как передавалось сообщение. Шум или интерференция могут сорвать передачу и вызвать отличие принятого сообщения от посланного.
- **Переключение пакетов (Packet switching)**. Данные, подлежащие передаче разбиваются на дискретные пакеты, состоящие из групп символов, которые затем отправляются независимо по пути между отправителем и получателем являющимся наиболее удобным в это время. Соединение является виртуальным и поэтому данные могут следовать разными путями. Каждый пакет имеет электронную метку с кодами, для указания его происхождение и назначение, и может пойти по другому сетевому пути, чем другие пакеты. Достигнув места назначения пакеты вновь собираются воспроизводят исходное сообщение.
- **Internet connection protocols** (протоколы соединения Интернет): инструкции для подключения компьютера к сети Интернет. Важными протоколами являются PPP (точка-точка) и SLIP (Serial Line Internet Protocol).

- **HTTP** (HyperText Transfer Protocol – гипертекстовый протокол передачи) это набор правил, которые размещают и перемещают файлы всех типов, включая файлы мультимедиа по сети Internet. Он не обрабатывает пакеты данных, которые он передает, а просто обеспечивает им достижение места назначения и так определяет компьютерные ссылки на Web. HTTPS (протокол передачи гипертекста (Secure)) показывает, что HTTP должен быть использован, но с другим портом по умолчанию и дополнительным слоем шифрования/аутентификации между HTTP и TCP.
- **Open Systems Interconnection (OSI)**. Поскольку сети работают в нескольких различных способах, существует необходимость определить стандарты для обмена данными, чтобы обеспечить средства для компьютеров от разных производителей, чтобы общаться друг с другом. Данные, передаваемые по сети должны прибывать в пункт назначения в срок, в правильной и узнаваемой форме и содействовать этому процессу, модель OSI состоит из семи уровней, каждый из которых выбран для выполнения четко определенных функций. Хотя OSI является международным стандартом, более важным, как понятие является стандарт TCP/IP который, на самом деле используется в настоящее время.
- **TCP/IP** (Transmission Control Protocol, Internet Protocol). Это два протокола, которые поддерживают основную работу в Интернете и включают правила, которые компьютеры объединенные в сети используют для установления и разрыва соединения. TCP/IP управляет разбиением сообщения на небольшие пакеты перед передачей, контролирует сборку пакетов, как только они достигнут места назначения, есть правила для маршрутизации отдельных пакетов данных от их источника к месту назначения и позволяет пользователям отправлять и получать сообщения, искать информацию, обмениваться данными и скачивать программное обеспечение.
- **HTML** (**HyperText Markup Language – гипертекстовый язык разметки**) используется для описания того, как web браузер должен отобразить текстовый файл полученный от сервера. HTML определяет как отображается страница. Он унаследован от **SGML** – Standard Generalised Markup Language, который является стандартной системой (an ISO standard), используемой для специфицирования структуры документа. HTML позволяет разработчику web страницы определять гиперссылки между этими документами и документами, которые требуются.
- **XML** (**Extendible Markup Language – расширенный язык разметки**) определяет фактическое содержимое страницы, а также интерпретирует значение данных. Он определяет, какие данные отображаются, в то время как HTML определяет только способ отображения страницы. Элементы структурированной информации включают в себя: контент (фотографии, графики, текста и т.д.), а также роль, которую контент играет в документе; где контент расположеный в документе влияет на значение определенное для него.

- **XSL** (Extensible Stylesheet Language) – расширяемый язык стилей
- **XHTML** (Extensible Hypertext Markup Language) – новейшая версия HTML, основанная на XML и разработанная для поддержки новейших устройств.
- **HTML5**(Extensible Hypertext Markup Language – гипертекстовый язык разметки 5 версия) – Это пятая версия HTML, последняя (четвёртая) версия которого была стандартизирована в 1997 году. По состоянию на октябрь 2013 года, HTML5 ещё находится в разработке, но, фактически, является рабочим стандартом (англ. HTML Living Standard). Цель разработки HTML5 — улучшение уровня поддержки мультимедиа-технологий, сохраняя при этом удобочитаемость кода для человека и простоту анализа для парсеров.
- **CSS3** (англ. Cascading Style Sheets 3 — каскадные таблицы стилей третьего поколения) — активно разрабатываемая спецификация CSS. Представляет собой формальный язык, реализованный с помощью языка разметки. Самая масштабная редакция по сравнению с CSS1, CSS2 и CSS2.1. Главной особенностью CSS3 является возможность создавать анимированные элементы без использования JavaScript, поддержка линейных и радиальных градиентов, теней, сглаживания и многое другое. Преимущественно используется как средство описания и оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.
- **JavaScript** – прототипно-ориентированный сценарный язык программирования. Является диалектом языка ECMAScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.
- **Java** – язык программирования, используемый для создания web-приложений. Многое в синтаксисе Java унаследовано из C++, а компилируемая версия может запускаться на любой вычислительной архитектуре в виртуальной Java машине.
- **Интранет (intranet)** – это частная (внутренняя) сеть доступная только отдельным пользователям внутри организации. Она строится на базе web технологии, но доступ к ней ограничивается, чтобы исключить внешний. Интранет использует web браузеры и гипертекстовые ссылки точно так же как и World Wide Web, с единственной разницей в том, где web страницы расположены и кто может иметь доступ к ним.
- **Экстранет (extranet)** – это также приватная или частная сеть (на основе web технологий) которая соединяет отдельные части intranet компании с ее заказчиками, поставщиками, или иными авторизованными бизнес-партнерами.
- **EDI** (англ. Electronic data interchange — электронный обмен данными) — серия стандартов и конвенций по передаче структурированной

цифровой информации между организациями, основанная на определенных регламентах и форматах передаваемых сообщений. Основная задача EDI – стандартизовать обмен транзакционной цифровой информацией, обеспечить возможности программного взаимодействия компьютерных систем различных сегментов, организаций, предприятий. EDI в течение многих лет оставался единственной формой существования электронной коммерции.

- **RFID** (англ. Radio Frequency IDentification, радиочастотная идентификация) — способ автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID-метках. Любая RFID-система состоит из считывающего устройства (считыватель, ридер или интеррогатор) и транспондера (он же RFID-метка, иногда также применяется термин RFID-тег).
- **Bluetooth** является открытым стандартом для ближней беспроводной связи (до 10 метров) между цифровыми устройствами. Включает в себя оборудование, программное обеспечение и взаимодействие стандартов, протоколов и требований и реализуется через небольшой, недорогой радио чип, который может быть включен в мобильные телефоны, КПК, КПК, компьютеры, принтеры и аналогичных устройств. (Первоначально она была разработана компанией Эрикссон и названа в честь Харальда Bluetooth, викинга и короля Дании в 940-981 гг.)
- **The IEEE 802.11 Protocols**. Ассоциация по стандартам Института инженеров по электротехнике и электронике (IEEE) разработала серию стандартов для беспроводной локальной сети (WLAN) устройств, работающих в непосредственной близости (до 100 м) под общим названием **IEEE 802.11**. Имеется множество вариантов этого протокола, такие как **IEEE 802.11a**, **IEEE 802.11b** и **IEEE 802.11g** (Wave Repo2001; O'Hara and & Petrick, 2005).
- **PDA** (Personal Digital Assistant). «личный цифровой секретарь», а также Handheld Computer — портативное вычислительное устройство, обладающее широкими функциональными возможностями. КПК часто называют наладонником (англ. palmtop) из-за небольших размеров. Изначально КПК предназначались для использования в качестве электронных органайзеров. С «классического» КПК невозможно совершать звонки, и КПК не является мобильным телефоном, поэтому к настоящему времени классические КПК практически полностью вытеснены коммуникаторами — КПК с модулем сотовой связи и смартфонами.
- **Personal Area Networks** (ПАН) являются специальными сетями персональных цифровых устройств, таких как ноутбуки, персональные цифровые помощники (PDA) и мобильные телефоны, способные передавать данные и связываться с другими персональными цифровыми устройствами по Bluetooth, или какой-то другой технологии беспроводной связи.
- **Семантическая Сеть** информационная модель предметной области, имеющая вид ориентированного графа, вершины которого

соответствуют объектам предметной области, а дуги (рёбра) задают отношения между ними. Объектами могут быть понятия, события, свойства, процессы. Таким образом, семантическая сеть является одним из способов представления знаний. В названии соединены термины из двух наук: семантика в языкознании изучает смысл единиц языка, а сеть в математике представляет собой разновидность графа — набора вершин, соединённых дугами (рёбрами), которым присвоено некоторое число. В семантической сети роль вершин выполняют понятия базы знаний, а дуги (причем направленные) задают отношения между ними. Таким образом, семантическая сеть отражает семантику предметной области в виде понятий и отношений.

- **Web 2.0** веб-сайты построенные из интерактивных объектов в Интернете и так, чтобы пользователи могли сделать больше, чем просто получить информацию. Это привело к разработке веб-сообществ и сайтов социальных сетей.
- **Internet service provider** (ISP). Это компания, которая предоставляет заказчикам услуги по доступу в Интернет.
- **Web hosting service provider**. Эти организации предоставляют место на своем сервере для веб-страниц организаций или физических лиц. Они также предлагают подключение своих серверов к Интернету.
- **Social networking** – Сайты социальных сетей (например, Facebook, MySpace, Twitter и LinkedIn) направлены на создание интернет-сообществ людей, которые разделяют интересы.
- **Wiki** это сайт, который позволяет легко создавать и редактировать взаимосвязанные совместные страницы. Википедия энциклопедия является одним из самых известных из этих вики (http://en.wikipedia.org/wiki/Wiki).
- **Blog** (сокращение для вебблога) является сайтом, созданным физическим лицом для описания событий и материалов, представляющих интерес для них их отношения к жизни.
- **Интерактивность**— это принцип организации системы, при котором цель достигается информационным обменом элементов этой системы. Элементами интерактивности являются все элементы взаимодействующей системы, при помощи которых происходит взаимодействие с другой системой/человеком (пользователем).
- **Мультимедиа –** интерактивная система, обеспечивающая одновременное представление различных медиа – звук, анимированная компьютерная графика, видеоряд. Например, в одном объекте-контейнере (англ. container) может содержаться текстовая, аудиальная, графическая и видеоинформация, а также, возможно, способ интерактивного взаимодействия с ней. Термин мультимедиа также, зачастую, используется для обозначения носителей информации, позволяющих хранить значительные объемы данных и обеспечивать достаточно быстрый доступ к ним (первыми носителями такого типа были Компакт-диски). В таком случае

термин мультимедиа означает, что компьютер может использовать такие носители и предоставлять информацию пользователю через все возможные виды данных, такие как аудио, видео, анимация, изображение и другие в дополнение к традиционным способам предоставления информации, таким как текст.

Ключевые слова: *сеть, приложение, Интернет, веб-разработка, программное обеспечение, стандарт, протокол, язык программирования, гипертекст, универсальный локатор ресурсов URL, браузер, мультимедиа*.

Контрольные вопросы

1. Отличительные особенности процесса разработки веб приложений от настольных.
2. Какие основные языки используются при разработке веб приложений?
3. Перечислите фундаментальные концепции веб программирования.
4. Что такой web браузер?
5. В чем отличие Wiki от блога?
6. Мультимедиа. Что может содержать мультимедиа контейнер?

ГЛАВА 2. ТИПЫ ВЕБ-ПРИЛОЖЕНИЙ И ПЛАТФОРМЫ ДЛЯ ИХ РАЗРАБОТКИ

Понятие Интернет-приложения

Web-приложение — это web-система, позволяющая пользователям реализовать доступ к бизнес-логике через браузер. Web-система — это система гипермедиа, поскольку ее ресурсы связаны между собой. Термин "web" означает, что система рассматривается как набор узлов с перекрестными ссылками.

Типы Интернет-приложений

Существует четыре типа Интернет-приложений:

- Web-приложения, которые работают на сервере, передавая через Интернет данные на клиентские машины. Для их применения требуются Web-браузеры, такие, как Google Chrome, Firefox, Opera, Microsoft Internet Explorer и Netscape Navigator;
- Web-сервисы, которые позволяют приложениям обрабатывать их данные на сервере. При этом передача подлежащих обработке данных на сервер и возврат результатов осуществляется через Интернет;
- приложения с поддержкой Интернета автономные программы со встроенными механизмами, позволяющими их пользователям регистрироваться, получать обновления, а также предоставляющими доступ к справочной системе и другим вспомогательным службам через Интернет;
- одноранговые приложения автономные программы, использующие Интернет для взаимодействия с другими программными продуктами этого же типа.

Архитектурные шаблоны Интернет-приложений

Под архитектурой системы понимается высокоуровневое представление архитектурно-значимых компонентов системы. В этом смысле компонент представляет собой отдельную сущность с открытым интерфейсом. Архитектурно-значимые компоненты это те, которые входят в представление системы на самом высоком уровне. Обычно эти компоненты или их набор нельзя сгруппировать или объединить в пакет с другими аналогичными компонентами. Свойства отдельных компонентов будут зависеть от особенностей представления системы.

Существует широкий спектр продуктов и технологий, которые могут быть частью архитектуры web-приложений. Архитектурный шаблон отражает фундаментальную структурно-организационную схему программных систем. Он представляет набор предопределенных подсистем, описывает спектр их обязанностей, а также определяет правила и рекомендации для организации взаимодействия между ними.

Можно выделить следующие архитектурные шаблоны web-приложений:

- Шаблон Thin Web Client (на основе "тонкого" Web-клиента) используется в большинстве приложений Internet и предоставляет ограниченные возможности по управлению конфигурацией клиента. В распоряжении клиента должен быть только стандартный браузер, поддерживающий формы. Все операции, связанные с бизнес-логикой, выполняются на сервере. Этот шаблон больше всего подходит для Web-приложений, в которых клиент обладает минимальными вычислительными возможностями или не может управлять своей конфигурацией.
- Шаблон Thick Web Client (на основе "толстого" Web-клиента) предполагает, что значительная часть бизнес-логики выполняется на клиентской машине. Обычно для выполнения бизнес-логики клиентом используется DHTML, аплеты Java или управляющие элементы ActiveX. Взаимодействие с сервером также происходит через протокол HTTP.
- Шаблон Web Delivery (на основе механизма Web-доставки). При взаимодействии клиента и сервера, кроме протокола HTTP, используются и другие протоколы, такие как IOOP (Internet Inter-Orb Protocol) и DCOM, которые могут применяться для поддержки системы распределенных объектов. В данном случае браузер функционирует как контейнерный модуль системы распределенных объектов.

Ключевые слова: Web-приложение, Web-система, Web-браузеры, Web-доставка, архитектурная система, архитектурный шаблон, Thin Web Client, Thick Web Client, Web Delivery.

Контрольные вопросы

1. Что такое Интернет-приложение?
2. Какие существуют типы Интернет-приложений?

3. Что такое архитектурная система?
4. Для чего используется шаблон Thin Web Client?
5. Для чего используется шаблон Thick Web Client?
6. Для чего используется шаблон Web Delivery?

ГЛАВА 3. СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНТЕРНЕТ-ПРИЛОЖЕНИЙ (ВЕБ-СЕРВЕРА)

Виды web-серверов

Различают статические и активные серверы Web. Если страницы сервера содержат только статическую текстовую и мультимедийную информацию, а также гипертекстовые ссылки на другие страницы, то сервер называется статическим. Если страницы web-сервера изменяют своё содержимое в зависимости от действий пользователя, то такие серверы называют активными. Статический сервер Web не может служить основой для создания интерактивных приложений с доступом через Интернет, так как он не предусматривает никаких средств ввода и обработки запросов.

Установка и настройка ПО

Поскольку в качестве практической основы курса мы будем рассматривать задачи, решаемые с помощью технологии клиент-сервер, и PHP соответственно будет использоваться для создания скриптов, обрабатываемых сервером, нам нужно установить web-сервер и интерпретатор PHP. В качестве web-сервера выберем, например, Apache, как наиболее популярный среди web-разработчиков. Для просмотра результатов работы программ нам понадобится web-браузер, например Internet Explorer.

Установка Apache 1.3.29 под Windows XP

Чтобы что-нибудь установить, нужно для начала иметь соответствующее программное обеспечение (ПО). Скачать ПО для установки Apache можно, например, с его официального сайта http://www.apache.org. Мы скачали файл apache_2.4.9-win32-x86-no_src.exe. Это автоматический установщик (иначе – wizard) сервера Apache под Windows. Эта программа попытается почти самостоятельно (а точнее, с минимальными усилиями с вашей стороны) установить на компьютер какое-либо программное обеспечение, а в данном случае сервер. После запуска файла установщика появляется следующее окошко (рис. 3.1).

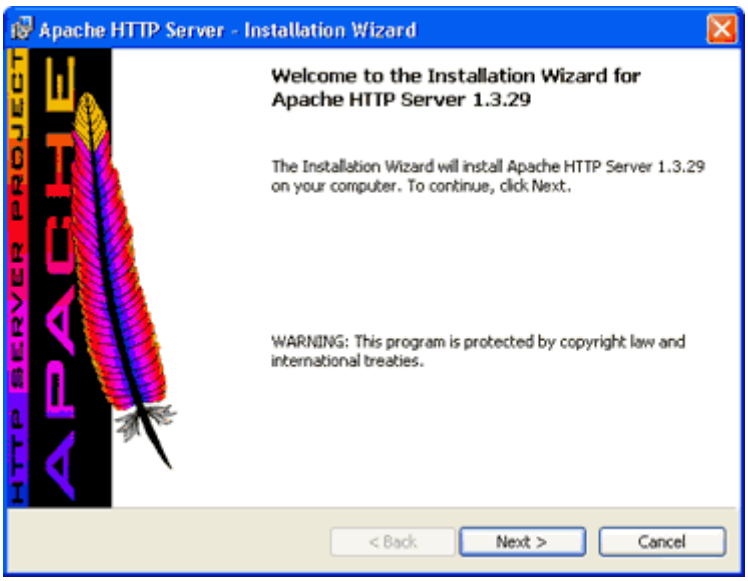


Рис. 3.1. Автоматическая установка сервера Apache

Чтобы установить HTTP- сервер Apache версии 1.3.29 на свой компьютер, нужно нажать на кнопку Next. Кстати говоря, эта же программа позволит изменить или удалить уже установленный web-сервер.

После нажатия кнопки Next программа предложит согласиться с условиями лицензии (рис. 3.2).

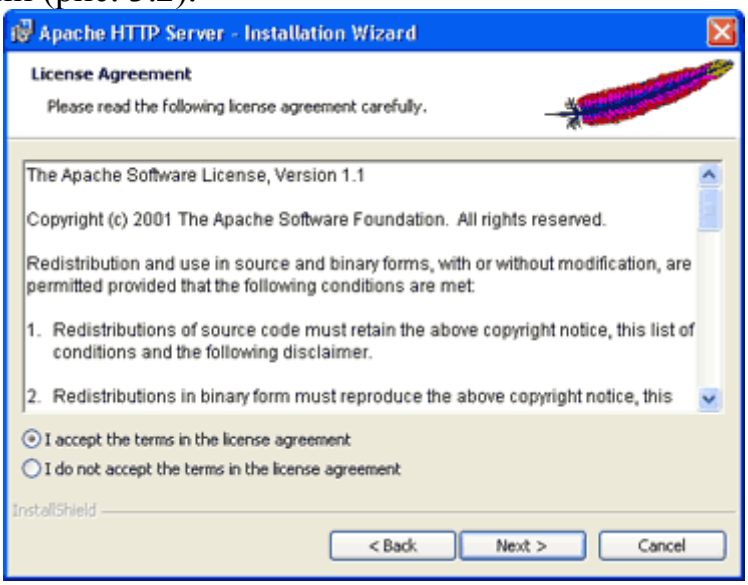


Рис. 3.2. Лицензионное соглашение

Следующий экран будет содержать информацию о сервере Apache, и в частности о его Windows-реализации (его изображение не приводим).

На следующем шаге нужно ввести имя сетевого домена, имя сервера и e-mail администратора. Программа попытается автоматически определить ваш домен и хост по настройкам компьютера (рис. 3.3).

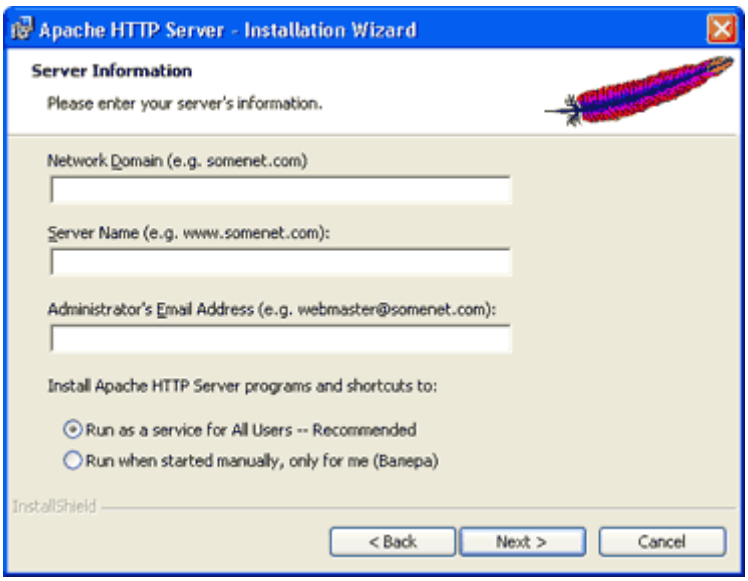


Рис. 3.3. Основная информация о сервере

После того как вы ввели данные в вышеприведенную форму, нужно выбрать тип установки: полная (устанавливаются все компоненты сервера) или определяемая пользователем (можно выбрать компоненты для установки) (рис. 3.4.).

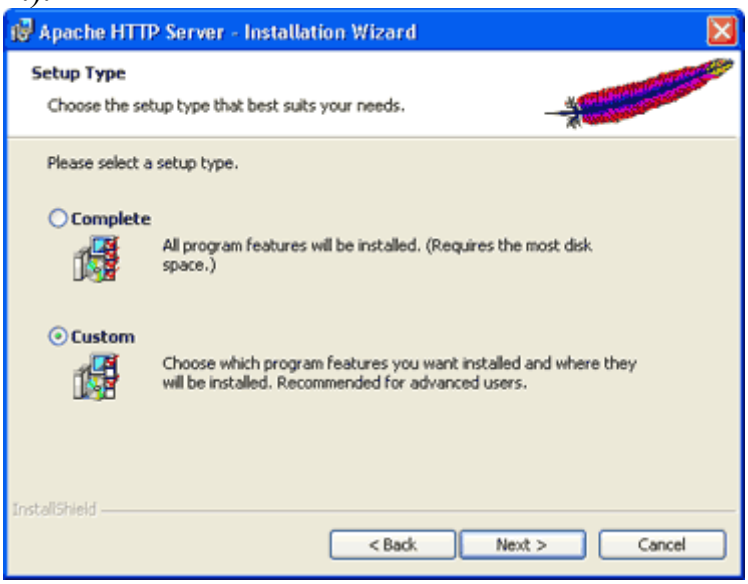


Рис. 3.4. Тип установки

Выбор компонентов сервера не очень большой – это инструменты, необходимые для работы сервера и документация к нему (рис. 3.5).

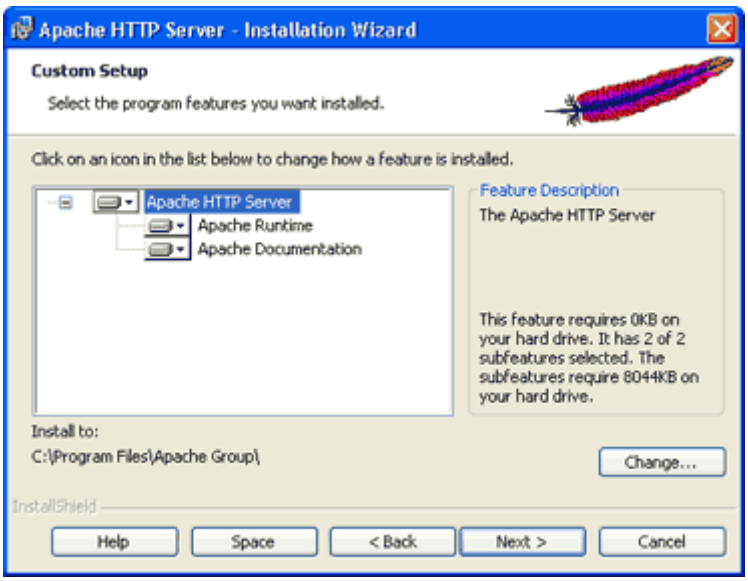


Рис. 3.5. Выбор компонент пользовательской установки

Мы выберем полную установку. Тогда на следующем шаге программа предложит выбрать папку, в которую будет установлен сервер. По умолчанию сервер устанавливается в папку c:\Program Files\Apache Group\ (рис. 3.6).

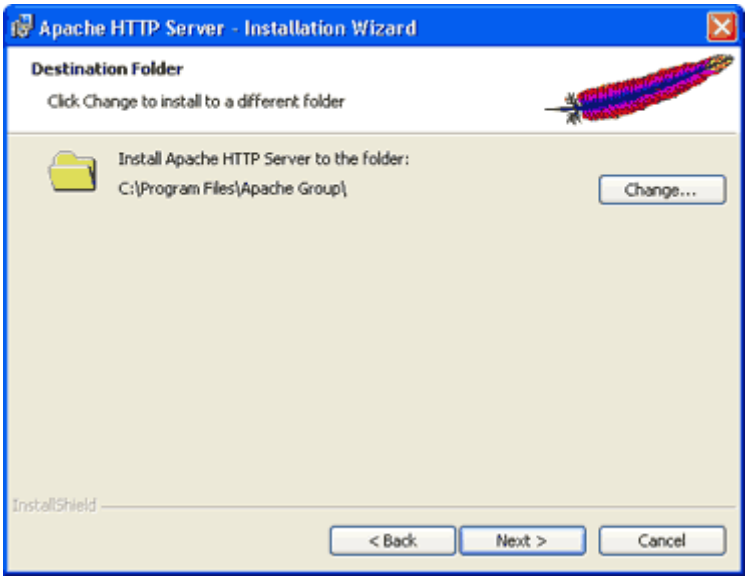


Рис. 3.6. Папка, в которую будет установлен сервер

На следующем экране (рис. 3.7) потребуется подтвердить правильность введенных данных и начать установку. Из любого окна установки, включая и это, можно вернуться назад и изменить введенные ранее данные.

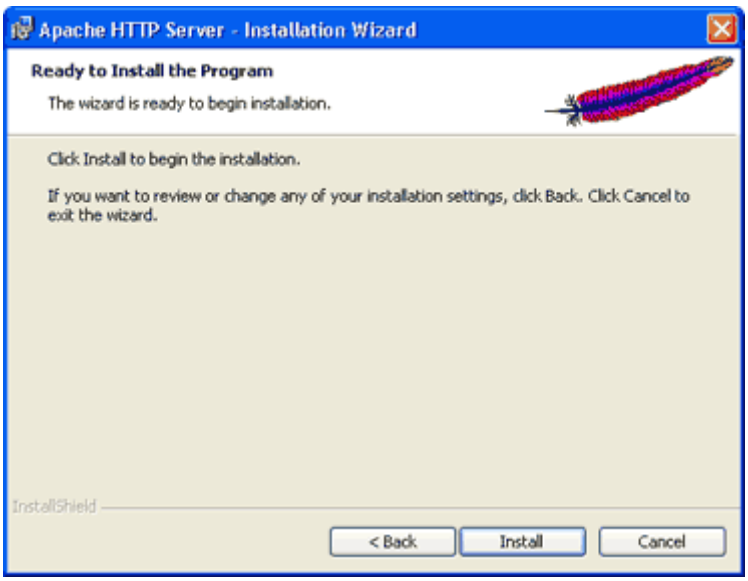


Рис. 3.7. Начало установки

После нажатия кнопки Install начнется непосредственная установка сервера. От пользователя никаких дополнительных действий больше не требуется. С одной стороны, это хорошо, но есть у такой автоматизированной установки и некоторые недостатки. Например, домашние директории пользователей оказываются там же, где и файлы настроек сервера (c:\Program Files\Apache Group\Apache\users\ ). Это небезопасно, если на компьютере работает несколько пользователей, не являющихся администраторами сервера. Но для начала можно ничего не менять. Допустим, мы запустили установщик, ввели все необходимые данные, он выполнил все операции успешно и говорит, что сервер установлен. Как проверить, действительно ли сервер установлен? Набираем в окне браузера http://localhost/. Если все установилось правильно, мы получим страничку приветствия сервера Apache (см. рис. 3.8).

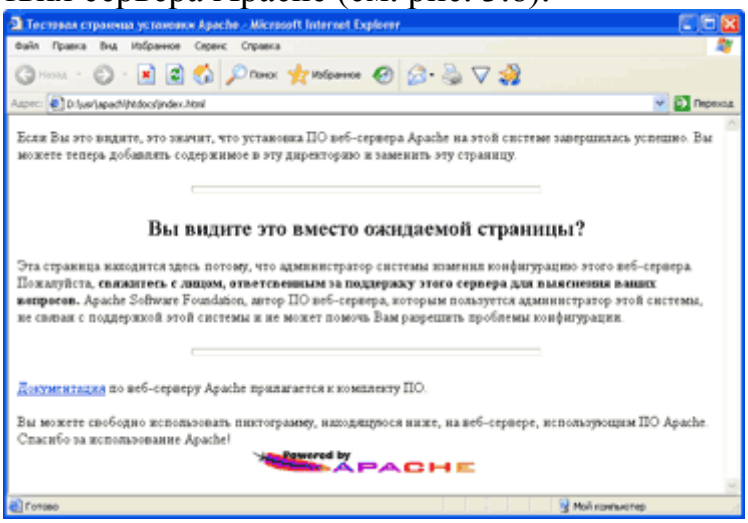


Рис. 3.8. Страница приветствия

Итак, сервер установлен. Как теперь с ним работать? Откуда можно запускать скрипты и где должны находиться файлы пользователей? Файлы, которые должны быть обработаны сервером, можно сохранять либо в корне сервера (в нашем случае это c:\Program Files\Apache Group\Apache\htdocs), либо в директориях пользователей (в нашем случае это c:\Program

Files\Apache Group\Apache\users\ ). Местоположение корня сервера и директорий пользователей прописано в настройках сервера, а точнее, в файле конфигурации httpd.conf (найти его можно в c:\Program Files\Apache Group\Apache\conf). Для изменения этих путей нужно изменить соответствующие переменные в файле конфигурации сервера. Переменная в файле конфигурации ServerRoot отвечает за корневую директорию сервера, а переменная UserDir – за директорию, где будут располагаться файлы пользователей сервера (для более безопасной работы советуем изменить переменную UserDir на что-нибудь типа c:\users\ ). Чтобы получить доступ к файлу test.html, находящемуся в корне сервера, нужно набрать в браузере http://localhost/test.html (т. е. имя хоста, имя файла). Если же файл test.html находится в директории пользователя user, то для его просмотра нужно набрать в браузере http://localhost/~user/test.html.

Ключевые слова: *Web-сервер, web-разработчик, Apache 1.3.29,клиент-сервер, HTTP- сервер, Internet Explorer, Windows-реализация, user, инсталяция.*

Контрольные вопросы

1. Какие существуют виды web-серверов?
2. Как установить и настроить ПО?
3. Что такое Apache 1.3.29?
4. Опишите процесс установки Apache 1.3.29 под Windows XP?
5. Как работать с сервером?

ГЛАВА 4. ОСНОВЫ ИНТЕРНЕТ: ПРОТОКОЛЫ HTTP, URL, HTML. ДРУГИЕ ПРОТОКОЛЫ И СТАНДАРТИЗАЦИЯ

Что такое Интернет?

Поскольку физической основой сети Веб является Интернет, то для более глубокого понимания многих вопросов данного курса потребуется кратко ознакомиться со структурой и протоколами Интернета.

Что же такое Интернет? По-сути, это самая большая в мире сеть, не имеющая единого центра управления, но работающая по единым правилам и предоставляющая своим пользователям единый набор услуг. Интернет можно рассматривать как «сеть сетей», каждая из которых управляется независимым оператором – поставщиком услуг Интернета (ISP, Internet Service Provider).

С точки зрения пользователей Интернет представляет собой набор информационных ресурсов, рассредоточенных по различным сетям, включая ISP-сети, корпоративные сети, сети и отдельные компьютеры домашних пользователей. Каждый отдельный компьютер в данной сети называется хостом (от английского термина host).

Сегодняшний Интернет обязан своему появлению объединенной сети ARPANET, которая начиналась как скромный эксперимент в новой тогда технологии коммутации пакетов (табл. 1). Сеть ARPANET была развернута в 1969 г. и состояла поначалу всего из четырех узлов с коммутацией пакетов, используемых для взаимодействия горстки хостов и терминалов. Первые линии связи, соединявшие узлы, работали на скорости всего 50 Кбит/с. Сеть ARPANET финансировалась управлением перспективного планирования научно-исследовательских работ ARPA (Advanced Research Projects Agency) министерства обороны США и предназначалась для изучения технологии и протоколов коммутации пакетов, которые могли бы использоваться для кооперативных распределенных вычислений.

Таблица 1. Хронология развития Интернета (с 1966 по 2000 г.)

Год	**Событие**
1966	Эксперимент с коммутацией пакетов управления ARPA
1969	Первые работоспособные узлы сети ARPANET
1972	Изобретение распределенной электронной почты
1973	Первые компьютеры, подключенные к сети ARPANET за пределами США
1975	Сеть ARPANET передана в ведение управления связи министерства обороны США
1980	Начинаются эксперименты с TCP/IP
1981	Каждые 20 дней к сети добавляется новый хост
1983	Завершен переход на TCP/IP
1986	Создана магистраль NSFnet
1990	Сеть ARPANET прекратила существование
1991	Появление Gopher
1991	Изобретение Всемирной паутины. Выпущена система PGP. Появление Mosaic
1995	Приватизация магистрали Интернета
1996	Построена магистраль OC-3 (155 Мбит/с)
1998	Число зарегистрированных доменных имен превысило 2 млн.
2000	Количество индексируемых веб-страниц превысило 1 млрд.

На рисунке 4.1. представлен график, показывающий динамику роста числа хостов (как формально зарегистрированных и так активно функционирующих).

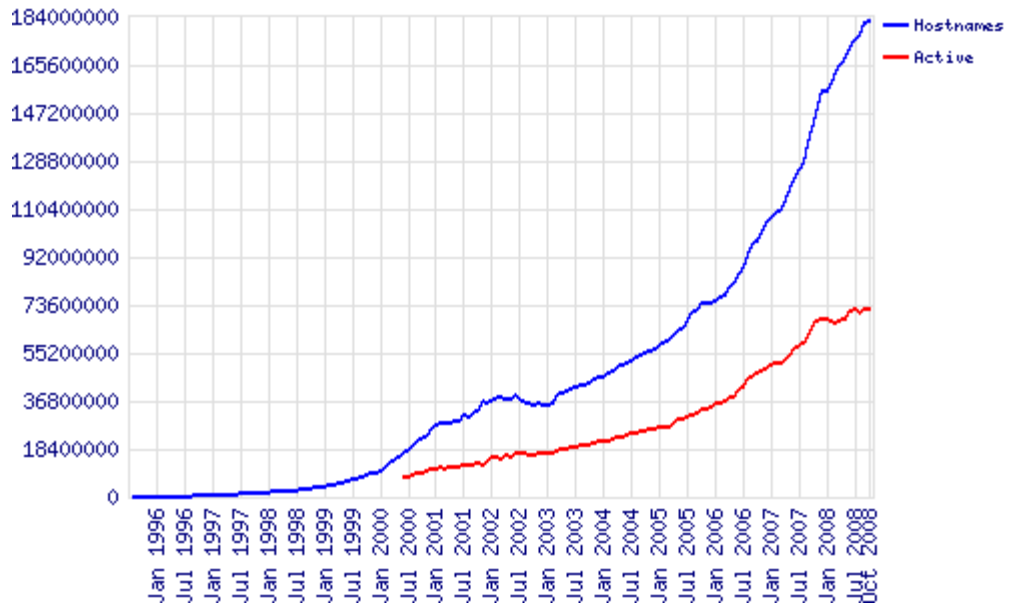


Рис 4.1. Динамика роста числа хостов в Интернет (взято с сайта www.netcraft.com).

Возможно ли централизованное управление в такой глобальной сети? Ответ на данный вопрос будет отрицательным, поскольку, во-первых, данная сеть является транснациональной и, во-вторых, в силу исторических предпосылок ее формирования.

Тем не менее, в Интернете могут проявляться опосредованные формы централизации в форме единой технической политики, согласованном наборе технических стандартов, назначении имен и адресов компьютеров и сетей, входящих в Интернет.

То есть Интернет является децентрализованной сетью, что имеет свои достоинства и недостатки.

1. Достоинства:

– Легкость наращивания Интернета путем заключения соглашения между двумя ISP.

2. Недостатки:

– Сложность модернизации технологий и услуг Интернета, поскольку требуются согласованные усилия всех поставщиков услуг.

– Невысокая надежность услуг Интернета.

– Ответственность за работоспособность отдельных сегментов этой сети возлагается на поставщиках услуг Интернета.

Существуют различные типы поставщиков услуг Интернета:

– просто поставщик услуг Интернета выполняет транспортную функцию для конечных пользователей – передачу их трафика в сети других поставщиков услуг Интернета;

– поставщик интернет-контента имеет собственные информационно-справочные ресурсы, предоставляя их содержание в виде веб-сайтов;

– поставщик услуг хостинга предоставляет свои помещения, каналы связи и серверы для размещения внешнего контента;

– поставщик услуг по доставке контента занимается только доставкой контента в многочисленные точки доступа с целью повышения скорости доступа пользователей к информации;

– поставщик услуг по поддержке приложений предоставляет клиентам доступ к крупным универсальным программным продуктам, например SAP R3;

– поставщик биллинговых услуг обеспечивает оплату счетов по Интернету;

Обычный сеанс связи с Интернет

Прежде чем перейти к описанию вопросов стандартизации рассмотрим как осуществляется доступ в Интернет по телефонной линии общего пользования. Абонент автоматической телефонной станции (АТС) должен купить у провайдера сети Интернет (Internet Service Provider — ISP) карту с предоплатой, в которой указан телефонный номер провайдера для доступа в Интернет, имя пользователя (User ID) и пароль (Password). Эти данные абонент должен ввести в персональный компьютер. При установлении модемного соединения с телефонным номером провайдера компьютер соединится с сервером сетевого доступа (Network Access Server, NAS), который запросит у компьютера имя и пароль. Компьютер автоматически перешлет ему запрошенную информацию. После этого NAS запросит те же данные (имя и пароль) у сервера аутентификации, авторизации и учета (Authentication, Authorization, Accounting) и сравнит данные имени и пароля, полученные от абонента и от AAA-сервера. В случае их совпадения NAS откроет домашнюю страничку провайдера и начнет обслуживание запросов абонента. Для реализации запроса может потребоваться соединение через магистральную сеть (Backbone Network-BN), которая использует высокоскоростные (от 622 Мбит/с до 1.28 Гбит/с) каналы связи и высокопроизводительные маршрутизаторы (R) для объединения зоновых сетей различных провайдеров. (Рис. 4.2.)

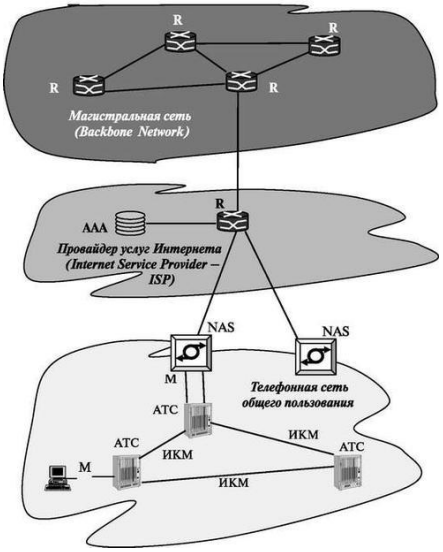


Рис. 4.2. Схема доступа абонента ТФОП в Интернет

В настоящее время широко внедряются методы доступа по цифровой абонентской линии (DSL — Digital Subscriber Line) и по каналам кабельного телевидения, которые обеспечивают непосредственный доступ абонента к NAS.

Модель OSI. Понятие об интерфейсах и протоколах. Рекомендация ITU-T X.200

Организация взаимодействия между элементами сети является сложной задачей, поэтому ее разбивают на несколько более простых задач.

Международной организацией по стандартизации (ISO) был предложен стандарт, который покрывает все аспекты сетевой связи, — это модель взаимодействия открытых систем (OSI). Он был введен в конце 1970-х.

Открытая система — это стандартизированный набор протоколов и спецификаций, который гарантирует возможность взаимодействия оборудования различных производителей. Она реализуется набором модулей, каждый из которых решает простую задачу внутри элемента сети. Каждый из модулей связан с одним или несколькими другими модулями. Решение сложной задачи подразумевает определенный порядок следования решения простых задач, при котором образуется многоуровневая иерархическая структура на рис. 4.3.. Это позволяет любым двум различным системам связываться независимо от их основной архитектуры.

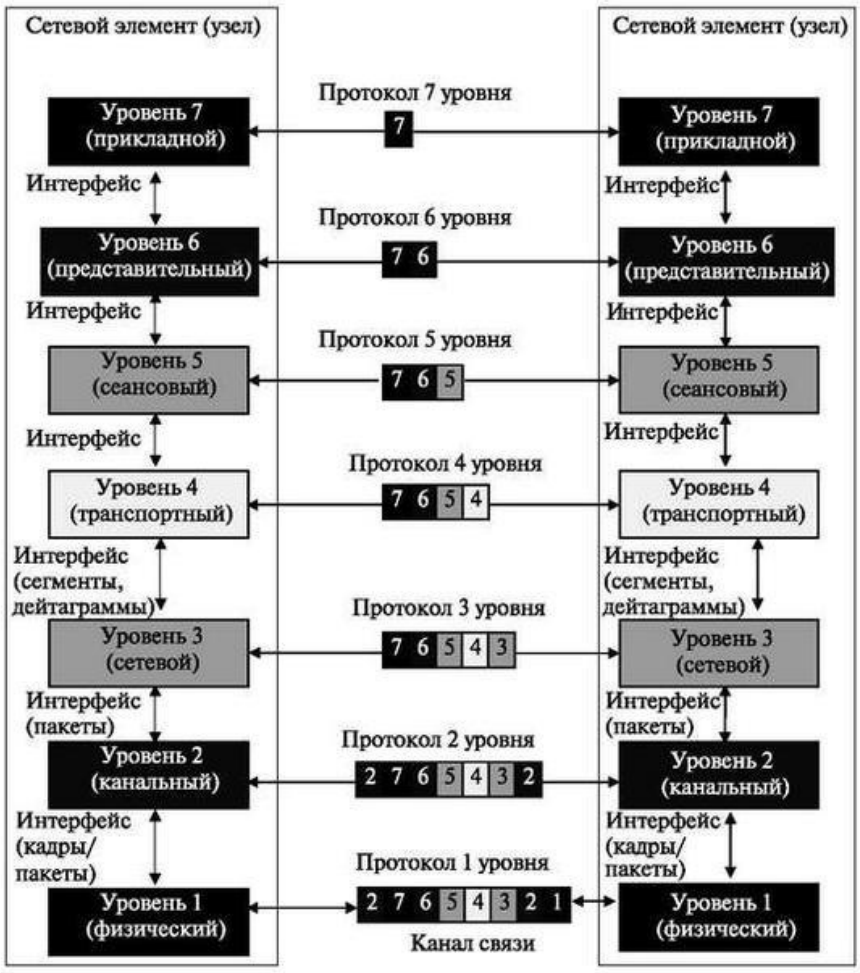


Рис. 4.3. Модель взаимодействия открытых систем OSI

Модель OSI составлена из семи упорядоченных уровней: физического (уровень 1), звена передачи данных (уровень 2), сетевого (уровень 3), транспортного (уровень 4), сеансового (уровень 5), представления (уровень 6) и прикладного (уровень 7).

Обмен информацией между модулями происходит на основе определенных соглашений, которые называются интерфейсом. При передаче сообщения модуль верхнего уровня решает свою часть задачи, а результат, понятный только ему, оформляет в виде дополнительного поля к исходному сообщению (заголовка) и передает измененное сообщение на дообслуживание в нижележащий уровень. Этот процесс называется инкапсуляцией.

Заголовки добавляются к началу передаваемых данных, как это показано на рис. 4.3. в уровнях 6, 5, 4, 3 и 2. На уровне 2 кроме заголовков добавляются конечные метки (окончания). На уровне 1 полный комплект преобразуется к форме, которая может быть передана к приемному устройству.

С другой стороны, при приеме сообщения нижележащий уровень после обработки своей части сообщения удаляет его и оставшееся сообщение передает вышележащему уровню. Например, уровень 2 удаляет данные, предназначенные для него, затем передает остальные к уровню 3. Уровень 3 затем удаляет данные, предназначенные для него, и передает остальные к уровню 4, и так далее.

Прохождение данных и сетевой информации вниз через уровни устройства передачи и назад через уровни устройства приема делается возможным с помощью интерфейсов и протоколов между каждой парой смежных уровней.

Интерфейс определяет формат, физические и электрические свойства сигналов обмена между модулями различных уровней, а протокол описывает логические процедуры по обработке сообщения удаленному узлу сети равного уровня.

Четкие интерфейсы и протоколы обеспечивают модульность, реализация функций каждого уровня может быть обновлена или удалена, не требуя изменений уровней, находящихся выше или ниже его.

Семь уровней можно рассматривать, исходя из принадлежности их к трем подгруппам. Нижние уровни 1, 2 и 3 — физический, звена данных и сетевой — имеют дело с физическими аспектами данных, перемещающихся от одного устройства до другого (таких как электрические спецификации, физические подключения, физическая адресация и синхронизация передачи и надежность). Верхние уровни 5, 6 и 7 — сеансовый, представления и прикладной — позволяют обеспечивать способность к взаимодействию среди несвязанных программных систем. Уровень 4 — транспортный уровень — связывает эти две подгруппы и гарантирует, что более низкие уровни передачи находятся в формате, который верхние уровни могут использовать. Верхние уровни OSI почти всегда реализовывались в программном обеспечении; более низкие уровни — комбинация аппаратных

средств и программного обеспечения, исключая физический уровень, который является главным образом аппаратным.

Наименование уровней и распределение функций между ними следующее.

Физический уровень (Physical Layer — PL) обеспечивает побитовую транспортировку кадров (часто называемую пакетом) между узлами по требуемой физической среде передачи (металлический кабель, оптоволоконная линия связи, радиоканал).

Физический уровень определяет следующие процедуры и функции, которые физические устройства и интерфейсы должны выполнять в ситуациях, возникающих при передаче информации:

- **Физические характеристики интерфейсов и сред передачи**. На физическом уровне задают характеристики интерфейса между устройствами и средами передачи. Он также определяет тип среды передачи.
- **Представление бит**. Физические данные уровня состоят из потока битов (последовательность нулей или единиц) без любой интерпретации. Чтобы быть переданными, биты должны кодироваться электрическими или оптическими сигналами.Физический уровень определяет тип кодирования (каким именно образом нули и единицы представляются в форме электрических сигналов).
- **Скорость данных**. Скорость передачи — число бит, передаваемых каждую секунду — также определяется физическим уровнем. Другими словами, физический уровень задает продолжительность бита, которая определяет, как долго длится передача блоков информации.
- **Синхронизация битов**. Передатчик и приемник могут иметь расходящиеся по своим значениям скорости, которые должны быть синхронизированы на уровне разряда.
- **Конфигурация линии.** Физический уровень определяет подключение устройств к среде передачи. В конфигурации "точка-точка" два устройства связаны вместе через приданную им линию связи. В многоточечной конфигурации линия связи разделена между несколькими устройствами.
- **Физическая топология**. Физическая топология определяет, как устройства связаны для того, чтобы создать сеть. Устройства могут быть связаны, используя топологию "каждый с каждым" (каждое устройство связано с каждым другим устройством), звездную топологию (устройства связаны через центральное устройство), кольцевую топологию (каждое устройство связано со следующим, формируя кольцо) или топологию типа "шина" (каждое устройство на общей линии связи).
- **Режим передачи**. Физический уровень также определяет направление передачи между двумя устройствами: симплекс, полудуплекс или дуплексный. В **симплексном режиме** только одно устройство может передать, а другое может только получить. Симплексный режим — однонаправленная связь. В **полудуплексном режиме** два устройства могут передавать и получать, но не в одно и то же время. В **полнодуплексном** (или

просто дуплексном) режиме два устройства могут передавать и получать информацию одновременно.

На **канальном уровне (Data Link Layer — DLL)** реализуются механизмы обнаружения и коррекции ошибок, возникающих в канале связи между узлами.

Задачи уровня звена передачи данных состоят в следующем:

Задачи уровня звена передачи данных состоят в следующем:

- **Цикловая синхронизация.** Канальный уровень данных преобразует поток битов, полученных от сетевого уровня вуправляемые модули данных, которые называются **кадрами.**
- **Физическая адресация**. Если кадры должны быть распределены между несколькими различными приемниками, уровень звена передачи данных добавляет заголовок к кадру, чтобы определить конкретный передатчик и/или приемник кадра. Если кадр предназначен для системы вне сети передатчика, добавляется адрес приемника или адрес устройства, которое подключает его к другой сети.
- **Управление потоком**. Если скорость, на которой данные поглощаются приемником, меньше, чем скорость, порождаемая в передатчике, уровень звена передачи данных применяет механизм управления потоком, чтобы предотвратить переполнение приемника.
- **Контроль ошибок**. Для этого пакет, поступающий с вышележащего (сетевого) уровня, преобразуется в кадр, т. е. дополняется контрольной суммой и обрамляется специальной последовательностью "Флаг", позволяющей определить начало и конец кадра. На приеме "Флаги" отбрасываются, и снова вычисляется контрольная сумма. Если вычисленная контрольная сумма совпадает с суммой, принятой из кадра, то кадр считается правильным и в виде пакета передается на сетевой уровень, а на передающую сторону высылается квитирующий кадр. В случае искажения или пропажи кадра квитирующий кадр не высылается, и передающая сторона через некоторый промежуток времени возобновляет повторную передачу. Поскольку к узлу (например, маршрутизатору) обычно подключено несколько каналов связи с различными технологиями передачи кадра, то для каждой технологии передачи канальный уровень добавляет к пакету соответствующее дополнительное поле. Сетевому уровню поставляются пакеты единообразного вида.
- **Управление доступом**. Когда два или более устройств могут использовать одну и ту же линию связи, протоколы уровня звена передачи данных необходимы для того, чтобы определить, какое устройство может иметь доступ к линии связи в конкретный момент времени.

Сетевой уровень (Network Layer — NL) служит для образования сквозной транспортной системы между оконечными устройствами пользователя через все промежуточные сети связи — "из конца в конец".

Он выполняет следующие задачи:

- **Логическая адресация**. Чтобы передать пакет, средства сетевого уровня собирают информацию о топологии сетевых соединений и

используют ее для выбора наилучшего пути. Каждый пакет содержит адрес получателя, который состоит из старшей части — номера сети и младшей — номера компьютера (узла) в этой сети. Все компьютеры одной сети имеют один и тот же номер сети, т. е. сеть — это совокупность компьютеров, сетевой адрес которых содержит один и тот же номер сети.

Сетевой уровень добавляет заголовок к пакету, прибывающему от верхнего уровня, который среди других атрибутов включает логические адреса передатчика и приемника.

- **Маршрутизация**. Когда независимые сети или линии связи включены вместе, чтобы создать **интернет-сети** (сеть сетей) или большую сеть, то используются подключающие устройства (называемые **маршрутизаторами**, или **коммутаторами**). Они последовательно направляют или коммутируют пакеты к конечному пункту назначения. Одна из функций сетевого уровня должна обеспечить этот механизм.

О роли стандартизации в Интернет

Как следует из всего вышеизложенного, Интернет является очень сложной сетью, и соответственно такой же сложной является задача организации взаимодействия между устройствами сети. Для решения такого рода задач используется декомпозиция, т.е. разбиение сложной задачи на несколько более простых задач-модулей. Одной из концепций, реализующих декомпозицию, является многоуровневый подход. Такой подход дает возможность проводить разработку, тестирование и модификацию каждого отдельного уровня независимо от других уровней. Иерархическая декомпозиция позволяет, перемещаясь в направлении от более низких к более высоким уровням переходить к более простому представлению решаемой задачи.

Специфика многоуровневого представления сетевого взаимодействия состоит в том, что в процессе обмена сообщениями участвуют как минимум две стороны, для которых необходимо обеспечить согласованную работу двух иерархий аппаратно-программных средств. Каждый из уровней должен поддерживать интерфейс с выше- и нижележащими уровнями собственной иерархии средств и интерфейс со средствами взаимодействия другой стороны на том же уровне иерархии. Данный тип интерфейса называется протоколом (см. рис. 4.4.).

Рис 4.4. Организация взаимодействия между уровнями иерархии при иерархического декомпозиции в сети Интернет.

Иерархически организованный набор протоколов, достаточный для организации взаимодействия узлов в сети, называется стеком протоколов.

В начале 80-х годов международные организации по стандартизации ISO (International Organization for Standardization), ITU (International Telecommunications Union) и другие разработали стандартную модель взаимодействия открытых систем OSI (Open System Interconnection). Назначение данной модели состоит в обобщенном представлении средств сетевого взаимодействия. Ее также можно рассматривать в качестве универсального языка сетевых специалистов (справочной модели).

Поскольку сеть – это соединение разнородного оборудования, актуальной является проблема совместимости, что в свою очередь, требует согласования всеми производителями общепринятых стандартов. Открытой является система, построенная в соответствии с открытыми спецификациями.

Спецификация представляет собой формализованное описание аппаратных (программных) компонентов, способов их функционирования, взаимодействия с другими компонентами, условий эксплуатации, особых характеристик. Под открытыми спецификациями понимаются опубликованные, общедоступные спецификации, соответствующие стандартам и принятые в результате достижения согласия после всестороннего обсуждения всеми заинтересованными сторонами. Использование открытых спецификаций при разработке систем позволяет третьим сторонам разрабатывать для этих систем аппаратно-программные средства расширения и модификации, а также создавать программно-аппаратные комплексы из продуктов разных производителей.

Если две сети построены с соблюдением принципов открытости, это дает следующие преимущества:

– Возможность построения сети из аппаратных и программных средств различных производителей, придерживающихся стандарта;

– Безболезненная замена отдельных компонентов сети другими, более совершенными;

– Легкость сопряжения одной сети с другой.

В рамках модели OSI средства взаимодействия делятся на семь уровней: прикладной, представления, сеансовый, транспортный, сетевой, канальный и физический. В распоряжение программистов предоставляется прикладной программный интерфейс, позволяющий обращаться с запросами к самому верхнему уровню, а именно, - уровню приложений.

Сеть Интернет строилась в полном соответствии с принципами открытых систем. В разработке стандартов этой сети принимали участие тысячи специалистов-пользователей сети из вузов, научных организаций и компаний. Результат работы по стандартизации воплощается в документах RFC.

RFC (англ. Request for Comments) – документ из серии пронумерованных информационных документов Интернета, содержащих технические спецификации и Стандарты, широко применяемые во Всемирной сети. В настоящее время первичной публикацией документов

RFC занимается IETF под эгидой открытой организации Общество Интернета (ISOC). Правами на RFC обладает именно Общество Интернет. Формат RFC появился в 1969 г. при обсуждении проекта ARPANET. Первые RFC распространялись в печатном виде на бумаге в виде обычных писем, но уже с декабря 1969 г., когда заработали первые сегменты ARPANET, документы начали распространяться в электронном виде. В таблице 2 приведены некоторые из наиболее известных документов RFC.

Таблица 2. Примеры популярных RFC-документов.

Номер RFC	**Тема**
RFC 768	UDP
RFC 791	IP
RFC 793	TCP
RFC 822	Формат электронной почты, заменен RFC 2822
RFC 959	FTP
RFC 1034	DNS – концепция
RFC 1035	DNS – внедрение
RFC 1591	Структура доменных имен
RFC 1738	URL
RFC 1939	Протокол POP версии 3 (POP3)
RFC 2026	Процесс стандартизации в Интернете
RFC 2045	MIME
RFC 2231	Кодировка символов
RFC 2616	HTTP
RFC 2822	Формат электронной почты
RFC 3501	IMAP версии 4 издание 1 (IMAP4rev1)

Основным организационным подразделением, координирующим работу по стандартизации Интернет, является ISOC (Internet Society), объединяющее порядка 100 тысяч участников, которые занимаются различными аспектами развития данной сети. ISOC курирует работу IAB (Internet Architecture Board), включающую две группы:

– IRTF (Internet Research Task Force). Координирует долгосрочные исследовательские проекты, относящиеся к TCP/IP;

– IETF (Internet Engineering Task Force). Инженерная группа, определяющая спецификации для последующих стандартов Интернет.

Разработкой стандартов для сети Веб, начиная с 1994 года, занимается Консорциум W3C (World Wide Web Consortium), основанный и до сих пор возглавляемый Тимом Бернерсом-Ли.

Консорциум W3C – организация, разрабатывающая и внедряющая технологические стандарты для Интернета и WWW. Миссия W3C формулируется следующим образом: «Полностью раскрыть потенциал Всемирной паутины путём создания протоколов и принципов, гарантирующих долгосрочное развитие Сети». Две другие важнейшие задачи

Консорциума – обеспечить полную «интернационализацию Сети» и сделать ее доступной для людей с ограниченными возможностями.

W3C разрабатывает для WWW единые принципы и стандарты, называемые «Рекомендациями», которые затем внедряются разработчиками программ и оборудования. Благодаря Рекомендациям достигается совместимость между программными продуктами и оборудованием различных компаний, что делает сеть WWW более совершенной, универсальной и удобной в использовании.

Все Рекомендации W3C открыты, то есть, не защищены патентами и могут внедряться любым человеком без каких-либо финансовых отчислений Консорциуму.

Для удобства пользователей Консорциумом созданы специальные программы-валидаторы (англ. Online Validation Service), которые доступны по сети и могут за несколько секунд проверить документы на соответствие популярным Рекомендациям W3C. Консорциумом также созданы многие другие утилиты для облегчения работы веб-мастеров и программистов. Большинство утилит – это программы с открытым исходным кодом, все они бесплатные. В последнее время, повинуясь мировым тенденциям, Консорциум, в целом, гораздо больше внимания уделяет проектам с открытым исходным кодом.

В российском сегменте Интернета имеется своя организация – Российский НИИ Развития Общественных Сетей РОСНИИРОС (Russian Institute for Public Networks, RIPN). РОСНИИРОС занимается координацией российских исследований и разработок в Интернете.

Прежде чем перейти к описанию структуры, принципов работы и основных протоколов сети Веб, рассмотрим основной стек протоколов сети Интернет - стек TCP/IP.

Стек протоколов TCP/IP

Эти протоколы изначально ориентированы на глобальные сети, в которых качество соединительных каналов не идеально. Он позволяет создавать глобальные сети, компьютеры в которых соединены друг с другом самыми разными способами от высокоскоростных оптоволоконных кабелей и спутниковых каналов до коммутируемых телефонных линий. TCP/IP соответствует модели OSI достаточно условно и содержит 4 уровня. Прикладной уровень стека соответствует трем верхним уровням модели OSI: прикладному, представления и сеансовому.

В сети данные всегда передаются блоками относительно небольшого размера. Каждый блок имеет префиксную часть (заголовок), описывающую содержимое блока, и суффиксную, содержащую, например, информацию для контроля целостности передаваемого блока данных.

Название стека протоколов TCP/IP состоит из названий двух разных протоколов. Протокол IP (Internet Protocol) представляет собой протокол нижнего (сетевого) уровня и отвечает за передачу пакетов данных в сети. Он относится к так называемым протоколам датаграмм и работает без

подтверждений. Последнее означает, что при его использовании доставка пакетов данных не гарантируется и не подтверждается. Не гарантируется также и то, что пакеты достигнут пункта назначения в той последовательности, в которой они были отправлены.

К протоколам сетевого уровня относится также протокол межсетевых управляющих сообщений ICMP (Internet Control Message Protocol), предназначенный для передачи маршрутизатором источнику информации об ошибках при передаче пакета.

Очевидно, что намного удобнее передавать данные по каналу, который работает корректно, доставляя все пакеты по порядку. Поэтому поверх протокола IP работает протокол передачи данных более высокого (транспортного) уровня – TCP (Transmission Control Protocol). Посылая и принимая пакеты через протокол IP, протокол TCP гарантирует доставку всех переданных пакетов данных в правильной последовательности.

Следует отметить, что при использовании протокола IP обеспечивается более быстрая передача данных, так как не тратится время на подтверждение приема каждого пакета. Есть и другие преимущества. Одно из них заключается в том, что он позволяет рассылать пакеты данных в широковещательном режиме, при котором они достигают всех компьютеров физической сети. Что же касается протокола TCP, то для передачи данных с его помощью необходимо создать канал связи между компьютерами. Он и создается с использованием протокола IP.

Для идентификации сетевых интерфейсов используются 3 типа адресов:

- аппаратные адреса (или MAC-адреса);
- сетевые адреса (IP-адреса);
- символьные (доменные) имена.

В рамках IP протокола для создания глобальной системы адресации, не зависящей от способов адресации узлов в отдельных сетях, используется пара идентификаторов, состоящая из номера сети и номера узла. При этом IP-адрес идентифицирует не отдельный компьютер или маршрутизатор, а одно сетевое соединение в составе сети, в которую он входит; то есть конечный узел может входить в несколько IP-сетей.

Система доменных имен DNS

Несмотря на то, что аппаратное и программное обеспечение в рамках TCP/IP сетей для идентификации узлов использует IP-адреса, пользователи предпочитают символьные имена (доменные имена).

Первоначально в локальных сетях из небольшого числа компьютеров применялись плоские имена, состоящие из последовательности символов без разделения их на отдельные части, например MYCOMP. Для установления соответствия между символьными именами и числовыми адресами использовались широковещательные запросы. Однако для больших территориально распределенных сетей, работающих на основе протокола TCP/IP такой способ оказался неэффективным. Поэтому для установления

соответствия между доменным именем и IP-адресом используется специальная система доменных имен (DNS, Domain Name System), которая основана на создаваемых администраторами сети таблиц соответствия.

В сетях TCP/IP используется доменная система имен, имеющая иерархическую (в виде дерева) структуру. Данная структура имен напоминает иерархию имен, используемую во многих файловых системах. Запись доменного имени начинается с самой младшей составляющей, затем после точки следует следующая по старшинству символьная часть имени и так далее. Последовательность заканчивается корневым именем, например: company.yandex.ru.

Построенная таким образом система имен позволяет разделять административную ответственность по поддержке уникальности имен в пределах своего уровня иерархии между различными людьми или организациями.

Совокупность имен, у которых несколько старших составных частей совпадают, образуют домен имен.

Корневой домен управляется центральными органами Интернета: IANA и Internic.

Домены верхнего уровня назначаются для каждой страны, а также для различных типов организаций. Имена этих доменов должны следовать международному стандарту ISO 3166. Для обозначения стран используются двухбуквенные аббревиатуры, например, uz (Узбекистан), ru (Российская Федерация), us (США), it (Италия), fr (Франция).

Для различных типов организаций используются трехбуквенные аббревиатуры:

- net – сетевые организации;
- org – некоммерческие организации;
- com - коммерческие организации;
- edu – образовательные организации;
- gov – правительственные организации.

Администрирование каждого домена возлагается на отдельную организацию, которая делегирует администрирование поддоменов другим организациям.

Для получения доменного имени необходимо зарегистрироваться в соответствующей организации, которой организция InterNIC делегировала свои полномочия по распределению доменных имен.

Регистратором доменных имен в зоне ru до 2005 г. являлся Российский научно-исследовательский институт развития общественных сетей (РосНИИРОС). В настоящее время регистрация доменов осуществляется одним из действующих регистраторов.

В TCP/IP сетях соответствие между доменными именами и IP-адресами может устанавливаться как локальными средствами, так и централизованными службами. Первоначально соответствие задавалось с помощью создаваемого вручную на хосте файла hosts.txt, состоящего из

строк, содержащих пару вида «доменное имя – IP-адрес». Однако с активным ростом Интернета такое решение оказалось немасштабируемым.

Альтернативное решение – централизованная служба DNS, использующая распределенную базу отображений «доменное имя – IP-адрес». Сервер домена хранит только имена, которые заканчиваются на следующем ниже по дереву уровне. Это позволяет распределять более равномерно нагрузку по разрешению имен между всеми DNS-серверами. Каждый DNS-сервер помимо таблицы отображения имен содержит ссылки на DNS-серверы своих поддоменов.

Существуют две схемы разрешения DNS-имен.

Нерекурсивная процедура:

1. DNS-клиент обращается к корневому DNS-серверу с указанием полного доменного имени;
2. DNS-сервер отвечает клиенту, указывая адрес следующего DNS-сервера, обслуживающего домен верхнего уровня, заданный в следующей старшей части имени;
3. DNS-клиент делает запрос следующего DNS-сервера, который отсылает его к DNS-серверу нужного поддомена и т.д., пока не будет найден DNS-сервер, в котором хранится соответствие запрошенного имени IP-адресу. Сервер дает окончательный ответ клиенту.

Реккурсивная процедура:

1. DNS-клиент запрашивает локальный DNS-сервер, обслуживающий поддомен, которому принадлежит клиент;
2. Далее
3. Если локальный DNS-сервер знает ответ, он возвращает его клиенту
4. Если локальный сервер не знает ответ, то он выполняет итеративные запросы к корневому серверу. После получения ответа сервер передает его клиенту.

Таким образом, при рекурсивной процедуре клиент фактически перепоручает работу своему серверу. Для ускорения поиска IP-адресов DNS-серверы широко применяют кэширование (на время от часов до нескольких дней) проходящих через них ответов.

Структура и принципы WWW

Сеть WWW образуют миллионы веб-серверов, расположенных по всему миру. Веб-сервер является программой, запускаемой на подключённом к сети компьютере и передающей данные по протоколу HTTP.

Для идентификации ресурсов (зачастую файлов или их частей) в WWW используются идентификаторы ресурсов URI (Uniform Resource Identifier). Для определения местонахождения ресурсов в этой сети используются локаторы ресурсов URL (Uniform Resource Locator). Такие URL-локаторы представляют собой комбинацию URI и системы DNS.

Доменное имя (или IP-адрес) входит в состав URL для обозначения компьютера (его сетевого интерфейса), на котором работает программа веб-сервер.

На клиентском компьютере для – просмотра информации, полученной от веб-сервера, применяется специальная программа – веб-браузер. Основная функция веб-браузера – отображение гипертекстовых страниц (веб-страниц). Для создания гипертекстовых страниц в WWW изначально использовался язык HTML. Множество веб-страниц образуют веб-сайт.

Прокси-серверы

Прокси-сервер (proxy-server) – служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым службам.

Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс, расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (если имеется). В некоторых случаях запрос клиента или ответ сервера может быть изменен прокси-сервером в определённых целях. Также прокси-сервер позволяет защищать клиентский компьютер от некоторых сетевых атак.

Чаще всего прокси-серверы применяются для следующих целей:

– обеспечение доступа с компьютеров локальной сети в Интернет;

– кэширование данных: если часто происходят обращения к одним и тем же внешним ресурсам, то можно держать их копию на прокси-сервере и выдавать по запросу, снижая тем самым нагрузку на канал во внешнюю сеть и ускоряя получение клиентом запрошенной информации.

– сжатие данных: прокси-сервер загружает информацию из Интернета и передаёт информацию конечному пользователю в сжатом виде.

– защита локальной сети от внешнего доступа: например, можно настроить прокси-сервер так, что локальные компьютеры будут обращаться к внешним ресурсам только через него, а внешние компьютеры не смогут обращаться к локальным вообще (они «видят» только прокси-сервер).

– ограничение доступа из локальной сети к внешней: например, можно запретить доступ к определённым веб-сайтам, ограничить использование интернета каким-то локальным пользователям, устанавливать квоты на трафик или полосу пропускания, фильтровать рекламу и вирусы.

– анонимизация доступа к различным ресурсам. Прокси-сервер может скрывать сведения об источнике запроса или пользователе. В таком случае целевой сервер видит лишь информацию о прокси-сервере, например, IP-адрес, но не имеет возможности определить истинный источник запроса. Существуют также искажающие прокси-серверы, которые передают целевому серверу ложную информацию об истинном пользователе.

Протоколы Интернет прикладного уровня

Самый верхний уровень в иерархии протоколов Интернет занимают следующие протоколы прикладного уровня:

- DNS - распределённая система доменных имён, которая по запросу, содержащему доменное имя хоста сообщает IP адрес;
- HTTP - протокол передачи гипертекста в Интернет;

- HTTPS - расширение протокола HTTP, поддерживающее шифрование;
- FTP (File Transfer Protocol - RFC 959) - протокол, предназначенный для передачи файлов в компьютерных сетях;
- Telnet (TELecommunication NETwork - RFC 854) - сетевой протокол для реализации текстового интерфейса по сети;
- SSH (Secure Shell - RFC 4251) - протокол прикладного, позволяющий производить удалённое управление операционной системой и передачу файлов. В отличие от Telnet шифрует весь трафик;
- POP3 – протокол почтового клиента, который используется почтовым клиентом для получения сообщений электронной почты с сервера;
- IMAP - протокол доступа к электронной почте в Интернет;
- SMTP – протокол, который используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю;
- LDAP - протокол для доступа к службе каталогов X.500, является широко используемым стандартом доступа к службам каталогов;
- XMPP (Jabber) - основанный на XML расширяемый протокол для мгновенного обмена сообщениями в почти реальном времени;
- SNMP - базовый протокол управления сети Internet.

Рассмотрим более подробно некоторые из этих протоколов.

1. **FTP**

FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер; кроме того, возможен режим передачи файлов между серверами; FTP позволяет обмениваться файлами и выполнять операции над ними через TCP-сети. Данный протокол работает независимо от операционных систем. Исторически протокол FTP предложил открытую функциональность, обеспечивая прозрачный перенос файлов с одного компьютера на другой по сети. Это не так тривиально, как может показаться, так как у разнотипных компьютеров могут различаться размеры слов, биты в словах могут храниться в неодинаковом порядке или использоваться разные форматы слов.

2. **Telnet**

Название «telnet» имеют также некоторые утилиты, реализующие клиентскую часть протокола. Протокол telnet работает в соответствии с принципами архитектуры «клиент-сервер» и обеспечивает эмуляцию алфавитно-цифрового терминала, ограничивая пользователя режимом командной строки. Приложение telnet предоставило язык для общения терминалов с удаленными компьютерами. Когда появилась сеть ARPANET, для каждой компьютерной системы требовались собственные терминалы. Приложение telnet стало общим знаменателем для терминалов. Достаточно было написать для каждого компьютера программное обеспечение,

поддерживающее «терминал telnet», чтобы один терминал мог взаимодействовать с компьютерами всех типов.

3. **SSH**

Сходен по функциональности с протоколами telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH-клиенты и SSH-серверы имеются для большинства операционных систем.

4. **Почтовые протоколы.**

Хотя telnet и FTP были (и остаются) полезными, первым приложением, совершившим переворот в сознании пользователей компьютеров сети ARPANET, стала электронная почта. До сети ARPANET существовали системы электронной почты, но все они были однокомпьютерными системами. В 1972 г. Рэй Томлинсон (Ray Tomlinson) из компании BBN написал первый пакет, предоставляющий распределенные почтовые услуги в компьютерной сети из нескольких компьютеров. Уже к 1973 г. исследования управления ARPA показали, что три четверти всего трафика сети ARPANET составляла электронная почта. Польза электронной почты оказалась столь велика, что все больше пользователей стремилось подключиться к сети ARPANET, в результате чего возрастала потребность в добавлении новых узлов и использовании высокоскоростных линий. Таким образом, появилась тенденция, сохраняющаяся и по сей день.

- POP3 (Post Office Protocol Version 3 – RFC 1939) – протокол, который используется почтовым клиентом для получения сообщений электронной почты с почтового сервера;
- IMAP (Internet Message Access Protocol – RFC 3501) – протокол доступа к электронной почте. Аналогичен POP3, однако предоставляет пользователю богатые возможности для работы с почтовыми ящиками, находящимися на центральном сервере. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без необходимости постоянной пересылки с сервера и обратно файлов с полным содержанием писем.
- SMTP (Simple Mail Transfer Protocol – RFC 2821) – протокол, предназначенный для передачи электронной почты. Используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю. Для приёма почты почтовый клиент должен использовать протоколы POP3 или IMAP.

Базовым протоколом сети гипертекстовых ресурсов Веб является протокол HTTP. В его основу положено взаимодействие «клиент-сервер», то есть предполагается, что:

1. Потребитель-клиент инициировав соединение с поставщиком-сервером посылает ему запрос;
2. Поставщик-сервер, получив запрос, производит необходимые действия и возвращает обратно клиенту ответ с результатом.

При этом возможны два способа организации работы компьютера-клиента:

Тонкий клиент – это компьютер-клиент, который переносит все задачи по обработке информации на сервер. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с веб-приложениями.

Толстый клиент, напротив, производит обработку информации независимо от сервера, использует последний в основном лишь для хранения данных.

Прежде чем перейти к конкретным клиент-серверным веб-технологиям, рассмотрим основные принципы и структуру базового протокола HTTP.

Протокол HTTP

HTTP (HyperText Transfer Protocol – RFC 1945, RFC 2616) – протокол прикладного уровня для передачи гипертекста.

Центральным объектом в HTTP является ресурс, на который указывает URI в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т.д. Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя изначально данный протокол предназначен для передачи символьной информации. На первый взгляд это может показаться излишней тратой ресурсов. Действительно, данные в символьном виде занимают больше памяти, сообщения создают дополнительную нагрузку на каналы связи, однако подобный формат имеет много преимуществ. Сообщения, передаваемые по сети, удобочитаемы, и, проанализировав полученные данные, системный администратор может легко найти ошибку и устранить ее. При необходимости роль одного из взаимодействующих приложений может выполнять человек, вручную вводя сообщения в требуемом формате.

В отличие от многих других протоколов, HTTP является протоколом без памяти. Это означает, что протокол не хранит информацию о предыдущих запросах клиентов и ответах сервера. Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами. Например, клиентское веб-приложение, посылающее запросы, может отслеживать задержки ответов, а веб-сервер может хранить IP-адреса и заголовки запросов последних клиентов.

Всё программное обеспечение для работы с протоколом HTTP разделяется на три основные категории:

- Серверы – поставщики услуг хранения и обработки информации (обработка запросов).
- Клиенты – конечные потребители услуг сервера (отправка запросов).
- Прокси-серверы для поддержки работы транспортных служб.

Основными клиентами являются браузеры например: Internet Explorer, Opera, Mozilla Firefox, Netscape Navigator и другие. Наиболее популярными реализациями веб-серверов являются: Internet Information Services (IIS), Apache, lighttpd, nginx. Наиболее известные реализации прокси-серверов: Squid, UserGate, Multiproxy, Naviscope.

"Классическая" схема HTTP-сеанса выглядит так.

1. Установление TCP-соединения.
2. Запрос клиента.
3. Ответ сервера.
4. Разрыв TCP-соединения.

Таким образом, клиент посылает серверу запрос, получает от него ответ, после чего взаимодействие прекращается. Обычно запрос клиента представляет собой требование передать HTML-документ или какой-нибудь другой ресурс, а ответ сервера содержит код этого ресурса.

В состав HTTP-запроса, передаваемого клиентом серверу, входят следующие компоненты.

- Строка состояния (иногда для ее обозначения используют также термины строка-статус, или строка запроса).
- Поля заголовка.
- Пустая строка.
- Тело запроса.

Строку состояния вместе с полями заголовка иногда называют также заголовком запроса.

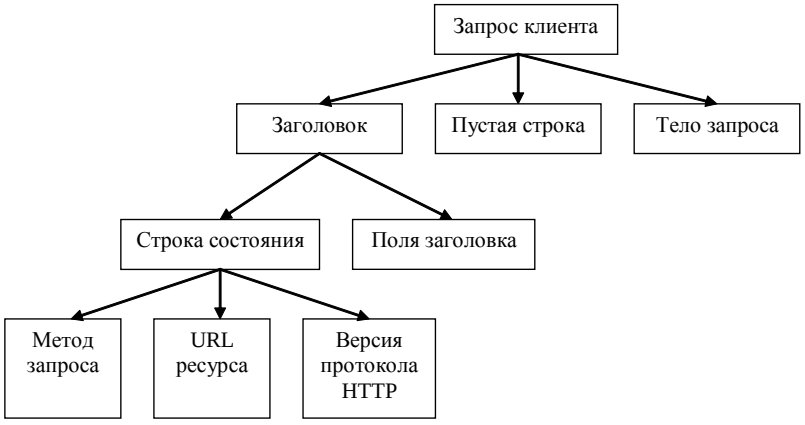


Рис 4.5. Структура запроса клиента.

Строка состояния имеет следующий формат:

метод_запроса URL_ресурса версия_протокола_HTTP

Рассмотрим компоненты строки состояния, при этом особое внимание уделим методам запроса.

Метод, указанный в строке состояния, определяет способ воздействия на ресурс, URL которого задан в той же строке. Метод может принимать значения GET, POST, HEAD, PUT, DELETE и т.д. Несмотря на обилие методов, для веб-программиста по-настоящему важны лишь два из них: GET и POST.

- GET. Согласно формальному определению, метод GET предназначается для получения ресурса с указанным URL. Получив запрос

GET, сервер должен прочитать указанный ресурс и включить код ресурса в состав ответа клиенту. Ресурс, URL которого передается в составе запроса, не обязательно должен представлять собой HTML-страницу, файл с изображением или другие данные. URL ресурса может указывать на исполняемый код программы, который, при соблюдении определенных условий, должен быть запущен на сервере. В этом случае клиенту возвращается не код программы, а данные, сгенерированные в процессе ее выполнения. Несмотря на то что, по определению, метод GET предназначен для получения информации, он может применяться и в других целях. Метод GET вполне подходит для передачи небольших фрагментов данных на сервер.

- POST. Согласно тому же формальному определению, основное назначение метода POST – передача данных на сервер. Однако, подобно методу GET, метод POST может применяться по-разному и нередко используется для получения информации с сервера. Как и в случае с методом GET, URL, заданный в строке состояния, указывает на конкретный ресурс. Метод POST также может использоваться для запуска процесса.
- Методы HEAD и PUT являются модификациями методов GET и POST.

Версия протокола HTTP, как правило, задается в следующем формате:

HTTP/версия.модификация

Поля заголовка, следующие за строкой состояния, позволяют уточнять запрос, т.е. передавать серверу дополнительную информацию. Поле заголовка имеет следующий формат:

Имя_поля: Значение

Назначение поля определяется его именем, которое отделяется от значения двоеточием.

Имена некоторых наиболее часто встречающихся в запросе клиента полей заголовка и их назначение приведены в таблице 1.

Таблица 1. Поля заголовка запроса HTTP.

Поля заголовка HTTP-запроса	**Значение**
Host	Доменное имя или IP-адрес узла, к которому обращается клиент
Referer	URL документа, который ссылается на ресурс, указанный в строке состояния
From	Адрес электронной почты пользователя, работающего с клиентом
Accept	MIME-типы данных, обрабатываемых клиентом. Это поле может иметь несколько значений, отделяемых одно от другого запятыми. Часто поле заголовка Accept используется для того, чтобы сообщить серверу о том, какие типы графических файлов поддерживает клиент

Accept-Language	Набор двухсимвольных идентификаторов, разделенных запятыми, которые обозначают языки, поддерживаемые клиентом
Accept-Charset	Перечень поддерживаемых наборов символов
Content-Type	MIME-тип данных, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Content-Length	Число символов, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Range	Присутствует в том случае, если клиент запрашивает не весь документ, а лишь его часть
Connection	Используется для управления TCP-соединением. Если в поле содержится Close, это означает, что после обработки запроса сервер должен закрыть соединение. Значение Keep-Alive предлагает не закрывать TCP-соединение, чтобы оно могло быть использовано для последующих запросов
User-Agent	Информация о клиенте

Во многих случаях при работе в Веб тело запроса отсутствует. При запуске CGI-сценариев данные, передаваемые для них в запросе, могут размещаться в теле запроса.

Ниже представлен пример HTML-запроса, сгенерированного браузером

GET http://oak.oakland.edu/ HTTP/1.0

Connection: Keep-Alive

User-Agent: Mozilla/4.04 [en] (Win95; I)

Host: oak.oakland.edu

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*

Accept-Language: en

Accept-Charset: iso-8859-l,*,utf-8

Получив от клиента запрос, сервер должен ответить ему. Знание структуры ответа сервера необходимо разработчику веб-приложений, так как программы, которые выполняются на сервере, должны самостоятельно формировать ответ клиенту.

Подобно запросу клиента, ответ сервера также состоит из четырех перечисленных ниже компонентов.

- Строка состояния.
- Поля заголовка.
- Пустая строка.
- Тело ответа.

Ответ сервера клиенту начинается со строки состояния, которая имеет следующий формат:

Версия_протокола Код_ответа Пояснительное_сообщение

• Версия_протокола задается в том же формате, что и в запросе клиента, и имеет тот же смысл.

• Код_ответа – это трехзначное десятичное число, представляющее в закодированном виде результат обслуживания запроса сервером.

• Пояснительное_сообщение дублирует код ответа в символьном виде. Это строка символов, которая не обрабатывается клиентом. Она предназначена для системного администратора или оператора, занимающегося обслуживанием системы, и является расшифровкой кода ответа.

Из трех цифр, составляющих код ответа, первая (старшая) определяет класс ответа, остальные две представляют собой номер ответа внутри класса. Так, например, если запрос был обработан успешно, клиент получает следующее сообщение:

HTTP/1.0 200 OK

Как видно, за версией протокола HTTP 1.0 следует код 200. В этом коде символ 2 означает успешную обработку запроса клиента, а остальные две цифры (00) – номер данного сообщения.

В используемых в настоящее время реализациях протокола HTTP первая цифра не может быть больше 5 и определяет следующие классы ответов.

• 1 – специальный класс сообщений, называемых информационными. Код ответа, начинающийся с 1, означает, что сервер продолжает обработку запроса. При обмене данными между HTTP-клиентом и HTTP-сервером сообщения этого класса используются достаточно редко.

• 2 – успешная обработка запроса клиента.

• 3 – перенаправление запроса. Чтобы запрос был обслужен, необходимо предпринять дополнительные действия.

• 4 – ошибка клиента. Как правило, код ответа, начинающийся с цифры 4, возвращается в том случае, если в запросе клиента встретилась синтаксическая ошибка.

• 5 – ошибка сервера. По тем или иным причинам сервер не в состоянии выполнить запрос.

• Примеры кодов ответов, которые клиент может получить от сервера, и поясняющие сообщения приведены в таблице 2.

Таблица 2. Классы кодов ответа сервера.

Код	**Расшифровка**	**Интерпретация**
100	Continue	Часть запроса принята, и сервер ожидает от клиента продолжения запроса
200	OK	Запрос успешно обработан, и в ответе клиента передаются данные, указанные в запросе
201	Created	В результате обработки запроса был создан новый ресурс
202	Accepted	Запрос принят сервером, но обработка его не окончена. Данный код ответа не гарантирует, что

		запрос будет обработан без ошибок.
206	Partial Content	Сервер возвращает часть ресурса в ответ на запрос, содержавший поле заголовка Range
301	Multiple Choice	Запрос указывает более чем на один ресурс. В теле ответа могут содержаться указания на то, как пра¬вильно идентифицировать запрашиваемый ресурс
302	Moved Permanently	Затребованный ресурс больше не располагается на сервере
302	Moved Temporarily	Затребованный ресурс временно изменил свой адрес
400	Bad Request	В запросе клиента обнаружена синтаксическая ошибка
403	Forbidden	Имеющийся на сервере ресурс недоступен для данного пользователя
404	Not Found	Ресурс, указанный клиентом, на сервере отсутствует
405	Method Not Allowed	Сервер не поддерживает метод, указанный в запросе
500	Internal Server Error	Один из компонентов сервера работает некорректно
501	Not Implemented	Функциональных возможностей сервера недоста¬точно, чтобы выполнить запрос клиента
503	Service Unavailable	Служба временно недоступна
505	HTTP Version not Supported	Версия HTTP, указанная в запросе, не поддержи¬вается сервером

В ответе используется такая же структура полей заголовка, как и в запросе клиента. Поля заголовка предназначены для того, чтобы уточнить ответ сервера клиенту. Описание некоторых из полей, которые можно встретить в заголовке ответа сервера, приведено в таблице 3.

Таблица 3. Поля заголовка ответа веб-сервера.

Имя поля	**Описание содержимого**
Server	Имя и номер версии сервера
Age	Время в секундах, прошедшее с момента создания ресурса
Allow	Список методов, допустимых для данного ресурса
Content-Language	Языки, которые должен поддерживать клиент для того, чтобы корректно отобразить передаваемый ресурс
Content-Type	MIME-тип данных, содержащихся в теле ответа сервера
Content-Length	Число символов, содержащихся в теле ответа сервера
Last-Modified	Дата и время последнего изменения ресурса
Date	Дата и время, определяющие момент генерации ответа
Expires	Дата и время, определяющие момент, после которого информация, переданная клиенту, считается устаревшей

Location	В этом поле указывается реальное расположение ресурса. Оно используется для перенаправления запроса
Cache-Control	Директивы управления кэшированием. Например, no-cache означает, что данные не должны кэшироваться

В теле ответа содержится код ресурса, передаваемого клиенту в ответ на запрос. Это не обязательно должен быть HTML-текст веб-страницы. В составе ответа могут передаваться изображение, аудио-файл, фрагмент видеоинформации, а также любой другой тип данных, поддерживаемых клиентом. О том, как следует обрабатывать полученный ресурс, клиенту сообщает содержимое поля заголовка Content-type.

Ниже представлен пример ответа сервера на запрос, приведенный в предыдущем разделе. В теле ответа содержится исходный текст HTML-документа.

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
X-Powered-By: ASP.NET
Date: Mon, 20 OCT 2008 11:25:56 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Sat, 18 Oct 2008 15:05:44 GMT
ETag: “b66a667f948c92:8a5”
Content-Length: 426

<html>
<body>
    <form action='http://localhost/Scripts/test.pl'>
        <p>Operand1: <input type='text' name='A'></p>
        <p>Operand2: <input type='text' name='B'></p>
        <p>Operation:<br>
        <select name='op'>
            <option value='+'>+</option>
            <option value='-'>-</option>
            <option value='*'>*</option>
            <option value='/'>/</option>
        <select></p>
        <input type='submit' value='Calculate!'>
    </from>
</body>
</html>
```

Поля заголовка и тело сообщения могут отсутствовать, но строка состояния является обязательным элементом, так как указывает на тип запроса/ответа.

Поле с именем Content-type может встречаться как в запросе клиента, так и в ответе сервера. В качестве значения этого поля указывается MIME-

тип содержимого запроса или ответа. MIME-тип также передается в поле заголовка Accept, присутствующего в запросе.

Спецификация MIME (Multipurpose Internet Mail Extension – многоцелевое почтовое расширение Internet) первоначально была разработана для того, чтобы обеспечить передачу различных форматов данных в составе электронных писем. Однако применение MIME не исчерпывается электронной почтой. Средства MIME успешно используются в WWW и, по сути, стали неотъемлемой частью этой системы.

Стандарт MIME разработан как расширяемая спецификация, в которой подразумевается, что число типов данных будет расти по мере развития форм представления данных. Каждый новый тип в обязательном порядке должен быть зарегистрирован в IANA (Internet Assigned Numbers Authority).

До появления MIME компьютеры, взаимодействующие по протоколу HTTP, обменивались исключительно текстовой информацией. Для передачи изображений, как и для передачи любых других двоичных файлов, приходилось пользоваться протоколом FTP.

В соответствии со спецификацией MIME, для описания формата данных используются тип и подтип. Тип определяет, к какому классу относится формат содержимого HTTP-запроса или HTTP-ответа. Подтип уточняет формат. Тип и подтип отделяются друг от друга косой чертой:

тип/подтип

Поскольку в подавляющем большинстве случаев в ответ на запрос клиента сервер возвращает исходный текст HTML-документа, то в поле Content-type ответа обычно содержится значение text/html. Здесь идентификатор text описывает тип, сообщая, что клиенту передается символьная информация, а идентификатор html описывает подтип, т.е. указывает на то, что последовательность символов, содержащаяся в теле ответа, представляет собой описание документа на языке HTML.

Перечень типов и подтипов MIME достаточно велик. В таблице 4 приведены примеры MIME-типов, наиболее часто встречающиеся в заголовках HTML-запросов и ответов.

Таблица 4. MIME типы данных.

Тип/подтип	**Расширение файла**	**Описание**
application/pdf	.pdf	Документ, предназначенный для обработки Acrobat Reader
application/msexcel	.xls	Документ в формате Microsoft Excel
application/postscript	.ps, .eps	Документ в формате PostScript
application/x-tex	.tex	Документ в формате TeX
application/msword	.doc	Документ в формате Microsoft Word
application/rtf	.rtf	Документ в формате RTF, отображаемый с помощью Microsoft Word
image/gif	.gif	Изображение в формате GIF

image/ jpeg	.jpeg, .jpg,	Изображение в формате JPEG
image/tiff	.tiff, .tif	Изображение в формате TIFF
image/x-xbitmap	.xbm	Изображение в формате XBitmap
text/plain	.txt	ASCII-текст
text/html	. html , . htm	Документ в формате HTML
audio/midi	.midi, .mid	Аудиофайл в формате MIDI
audio/x-wav	.wav	Аудиофайл в формате WAV
message/rfc822		Почтовое сообщение
message/news		Сообщение в группы новостей
video /mpeg	.mpeg, .mpg, .mpe	Видеофрагмент в формате MPEG
video/avi	.avi	Видеофрагмент в формате AVI

Для однозначной идентификации ресурсов в сети Веб используются уникальные идентификаторы URL.

Единообразный идентификатор ресурса URI (Uniform Resource Identifier) представляет собой короткую последовательность символов, идентифицирующую абстрактный или физический ресурс. URI не указывает на то, как получить ресурс, а только идентифицирует его. Это даёт возможность описывать с помощью RDF (Resource Description Framework) ресурсы, которые не могут быть получены через Интернет (имена, названия и т.п.). Самые известные примеры URI – это URL и URN.

- URL (Uniform Resource Locator) – это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса.
- URN (Uniform Resource Name) – это URI, который идентифицирует ресурс в определённом пространстве имён, но, в отличие от URL, URN не указывает на местонахождение этого ресурса.

URL имеет следующую структуру:

<схема>://<логин>:<пароль>@<хост>:<порт>/<URL-путь>

где:

- схема – схема обращения к ресурсу (обычно сетевой протокол);
- логин – имя пользователя, используемое для доступа к ресурсу;
- пароль – пароль, ассоциированный с указанным именем пользователя;
- хост – полностью прописанное доменное имя хоста в системе DNS или IP-адрес хоста;
- порт – порт хоста для подключения;
- URL-путь – уточняющая информация о месте нахождения ресурса.

Общепринятые схемы (протоколы) URL включают протоколы: ftp, http, https, telnet, а также:

- gopher – протокол Gopher;
- mailto – адрес электронной почты;
- news – новости Usenet;

- nntp – новости Usenet через протокол NNTP;
- irc – протокол IRC;
- prospero – служба каталогов Prospero Directory Service;
- wais – база данных системы WAIS;
- xmpp – протокол XMPP (часть Jabber);
- file – имя локального файла;
- data – непосредственные данные (Data: URL);

Обеспечение безопасности передачи данных HTTP

Поскольку протокол HTTP предназначен для передачи символьных данных в открытом (незашифрованном) виде, то лица, имеющие доступ к каналу передачи данных между клиентом и сервером, могут без труда просматривать весь трафик и использовать его для совершения несанкционированных действий. В связи с этим предложен ряд расширений базового протокола направленных на повышение защищенности интернет-трафика от несанкционированного доступа.

Самым простейшим является расширение HTTPS, при котором данные, передаваемые по протоколу HTTP, «упаковываются» в криптографический протокол SSL или TLS, тем самым обеспечивая защиту этих данных. В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443. Чтобы подготовить веб-сервер для обработки HTTPS соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера.

SSL (Secure Sockets Layer) – криптографический протокол, обеспечивающий безопасную передачу данных по сети Интернет. При его использовании создаётся защищённое соединение между клиентом и сервером. SSL изначально разработан компанией Netscape Communications. Впоследствии на основании протокола SSL 3.0 был разработан и принят стандарт RFC, получивший название TLS. Этот протокол использует шифрование с открытым ключом для подтверждения подлинности передатчика и получателя. Поддерживает надёжность передачи данных за счёт использования корректирующих кодов и безопасных хэш-функций. На нижнем уровне многоуровневого транспортного протокола (например, TCP) он является протоколом записи и используется для инкапсуляции различных протоколов (POP3, IMAP, SMTP или HTTP). Для каждого инкапсулированного протокола он обеспечивает условия, при которых сервер и клиент могут подтверждать друг другу свою подлинность, выполнять алгоритмы шифрования и производить обмен криптографическими ключами, прежде чем протокол прикладной программы начнет передавать и получать данные.

Для доступа к веб-страницам, защищённым протоколом SSL, в URL вместо схемы http, как правило, подставляется схема https, указывающая на то, что будет использоваться SSL-соединение. Стандартный TCP-порт для соединения по протоколу https – 443. Для работы SSL требуется, чтобы на сервере имелся SSL-сертификат.В сети Веб поддерживаются 3 типа аутентификации при клиент-серверных взаимодействиях:

- Basic – базовая аутентификация, при которой имя пользователя и пароль передаются в заголовках http-пакетов. Пароль при этом не шифруется и присутствует в чистом виде в кодировке base64. Для данного типа аутентификации использование SSL является обязательным.
- Digest – дайджест-аутентификация, при которой пароль пользователя передается в хэшированном виде. По уровню конфиденциальности паролей этот тип мало чем отличается от предыдущего, так как атакующему все равно, действительно ли это настоящий пароль или только хэш от него: перехватив удостоверение, он все равно получает доступ к конечной точке. Для данного типа аутентификации использование SSL является обязательным.
- Integrated – интегрированная аутентификация, при которой клиент и сервер обмениваются сообщениями для выяснения подлинности друг друга с помощью протоколов NTLM или Kerberos. Этот тип аутентификации защищен от перехвата удостоверений пользователей, поэтому для него не требуется протокол SSL. Только при использовании данного типа аутентификации можно работать по схеме http, во всех остальных случаях необходимо использовать схему https.

Cookie

Поскольку HTTP-сервер не помнит предыстории запросов клиентов, то каждый запрос обрабатывается независимо от других, и у сервера нет возможности определить, исходят ли запросы от одного клиента или разных клиентов.

Если сервер будет проверять TCP-соединения и запоминать IP-адреса компьютеров-клиентов, он все равно не сможет различить запросы от двух браузеров, выполняющихся на одной машине. И даже если допустить, что на компьютере работает лишь одна клиент-программа, то никто не может утверждать, что в промежутке между двумя запросами она не была завершена, а затем запущена снова уже другим пользователем.

Тем не менее, если вы когда-нибудь пользовались почтовым ящиком на mail.ru или на другом сервере, предоставляющем почтовые услуги пользователям Веб, вспомните, как вел себя клиент после того, как вы создали для себя почтовый ящик на сервере. Когда вы в следующий раз обратились с того же компьютера к mail.ru, вы, вероятно, заметили, что после загрузки веб-страницы ваше регистрационное имя уже отображалось в соответствующем поле ввода.

Такие сведения позволяет получить дополнительное средство под названием cookie. Механизм cookie позволяет серверу хранить информацию на компьютере клиента и извлекать ее оттуда.

Инициатором записи cookie выступает сервер. Если в ответе сервера присутствует поле заголовка Set-cookie, клиент воспринимает это как команду на запись cookie. В дальнейшем, если клиент обращается к серверу, от которого он ранее принял поле заголовка Set-cookie, помимо прочей информации он передает серверу данные cookie. Для передачи указанной информации серверу используется поле заголовка Cookie.

Для того чтобы в общих чертах представить себе, как происходит обмен данными cookie, рассмотрим следующий пример. Предположим, что клиент передает запросы на серверы A, B и C. Предположим также, что сервер B, в отличие от A и C, передает клиенту команду записать cookie. Последовательность запросов клиента серверу и ответов на них будет выглядеть приблизительно следующим образом.

1. Передача запроса серверу A.
2. Получение ответа от сервера A.
3. Передача запроса серверу B.
4. Получение ответа от сервера B. В состав ответа входит поле заголовка SetCookie. Получив его, клиент записывает cookie на диск.
5. Передача запроса серверу C. Несмотря на то что на диске хранится запись cookie, клиент не предпринимает никаких специальных действий, так как значение cookie было записано по инициативе другого сервера.
6. Получение ответа от сервера C.
7. Передача запроса серверу A. В этом случае клиент также никак не реагирует на тот факт, что на диске хранится cookie.
8. Получение ответа от сервера A.
9. Передача запроса серверу B. Перед тем как сформировать запрос, клиент определяет, что на диске хранится запись cookie, созданная после получения ответа от сервера B. Клиент проверяет, удовлетворяет ли данный запрос некоторым требованиям, и, если проверка дает положительный результат, включает в заголовок запроса поле Cookie.

Таким образом, процедуру записи и получения cookie можно представить себе как своеобразный "запрос" сервера, инкапсулированный в его ответе клиенту. Соответственно получение cookie также можно представить себе как ответ клиента, инкапсулированный в составе запроса тому же серверу.

Рассмотрим подробнее, какие данные передаются в поле заголовка Set-cookie и как они влияют на поведение клиента.

Поле Set-cookie имеет следующий формат:

Set-cookie: имя = значение; expires = дата; path = путь; domain = имя_домена, secure

где

- Пара имя = значение – именованные данные, сохраняемые с помощью механизм cookie. Эти данные должны храниться на клиент-машине и передаваться серверу в составе очередного запроса клиента.
- Дата, являющаяся значением параметра expires, определяет время, по истечении которого информация cookie теряет свою актуальность. Если ключевое слово expires отсутствует, данные cookie удаляются по окончании текущего сеанса работы браузера.
- Значение параметра domain определяет домен, с которым связываются данные cookie. Чтобы узнать, следует ли передавать в составе запроса данные cookie, браузер сравнивает доменное имя сервера, к которому

он собирается обратиться, с доменами, которые связаны с записями cookie, хранящимися на клиент-машине. Результат проверки будет считаться положительным, если сервер, которому направляется запрос, принадлежит домену, связанному с cookie. Если соответствие не обнаружено, данные cookie не передаются.

- Путь, указанный в качестве значения параметра path, позволяет выполнить дальнейшую проверку и принять окончательное решение о том, следует ли передавать данные cookie в составе запроса. Помимо домена с записью cookie связывается путь. Если браузер обнаружил соответствие имени домена значению параметра domain, он проверяет, соответствует ли путь к ресурсу пути, связанному с cookie. Сравнение считается успешным, если ресурс содержится в каталоге, указанном посредством ключевого слова path, или в одном из его подкаталогов. Если и эта проверка дает положительный результат, данные cookie передаются серверу. Если параметр path в поле Set-Cookie отсутствует, то считается, что запись cookie связана с URL конкретного ресурса, передаваемого сервером клиенту.
- Последний параметр, secure, указывает на то, что данные cookie должны передаваться по защищенному каналу.

Для передачи данных cookie серверу используется поле заголовка Cookie. Формат этого поля достаточно простой:

Cookie: имя=значение; имя=значение; ...

С помощью поля Cookie передается одна или несколько пар имя = значение. Каждая из этих пар принадлежит записи cookie, для которой URL запрашиваемого ресурса соответствуют имени домена и пути, указанным ранее в поле Set-cookie.

Как правило, Веб-приложение – приложение, в котором клиентом выступает браузер, а сервером – веб-сервер.

Рассмотрим типы программ, обеспечивающих работу Веб и использующих HTTP-протокол.

Никакой HTTP-обмен невозможен без клиента и сервера. Однако помимо клиента и сервера в веб-сеансе могут участвовать и другие программы, которые и являются объектом веб-программирования.

Результатом работы веб-приложения является веб-страница, отображаемая в окне браузера. При этом само веб-приложение может выполняться как на компьютере клиента, так и на компьютере сервера.

Рассмотрим подробнее обе схемы.

Программы, выполняющиеся на клиент-машине

Одним из типов программ, предназначенных для выполнения на клиент-машине, являются сценарии, например, JavaScript (VBScript). Исходный текст сценария представляет собой часть веб-страницы, поэтому сценарий JavaScript передается клиенту вместе с документом, в состав которого он входит. Обрабатывая HTML-документ, браузер обнаруживает исходный текст сценария и запускает его на выполнение.

Ко всем программам, которые передаются с сервера на клиент-машины и запускаются на выполнение, предъявляется одно общее требование: эти

программы должны быть лишены возможности обращаться к ресурсам компьютера, на котором они выполняются. Такое требование вполне обосновано. Ведь передача по сети и запуск Java-апплетов и JavaScript-сценариев происходит автоматически без участия пользователя, поэтому работа этих программ должна быть абсолютно безопасной для компьютера. Другими словами, языки, предназначенные для создания программ, выполняющихся на клиент-машине, должны быть абсолютно непригодны для написания вирусов и подобных программ.

Программы, выполняющиеся на сервере

Код программы, работающей на сервере, не передается клиенту. При получении от клиента специального запроса, предполагающего выполнение такой программы, сервер запускает ее и передает параметры, входящие в состав запроса. Средства для генерации подобного запроса обычно входят в состав HTML-документа.

Результаты своей работы программа оформляет в виде HTML-документа и передает их веб-серверу, а последний, в свою очередь, дополняет полученные данные HTTP-заголовком и передает их клиенту. Взаимодействие клиента и сервера в этом случае показано на рисунке 4.6.

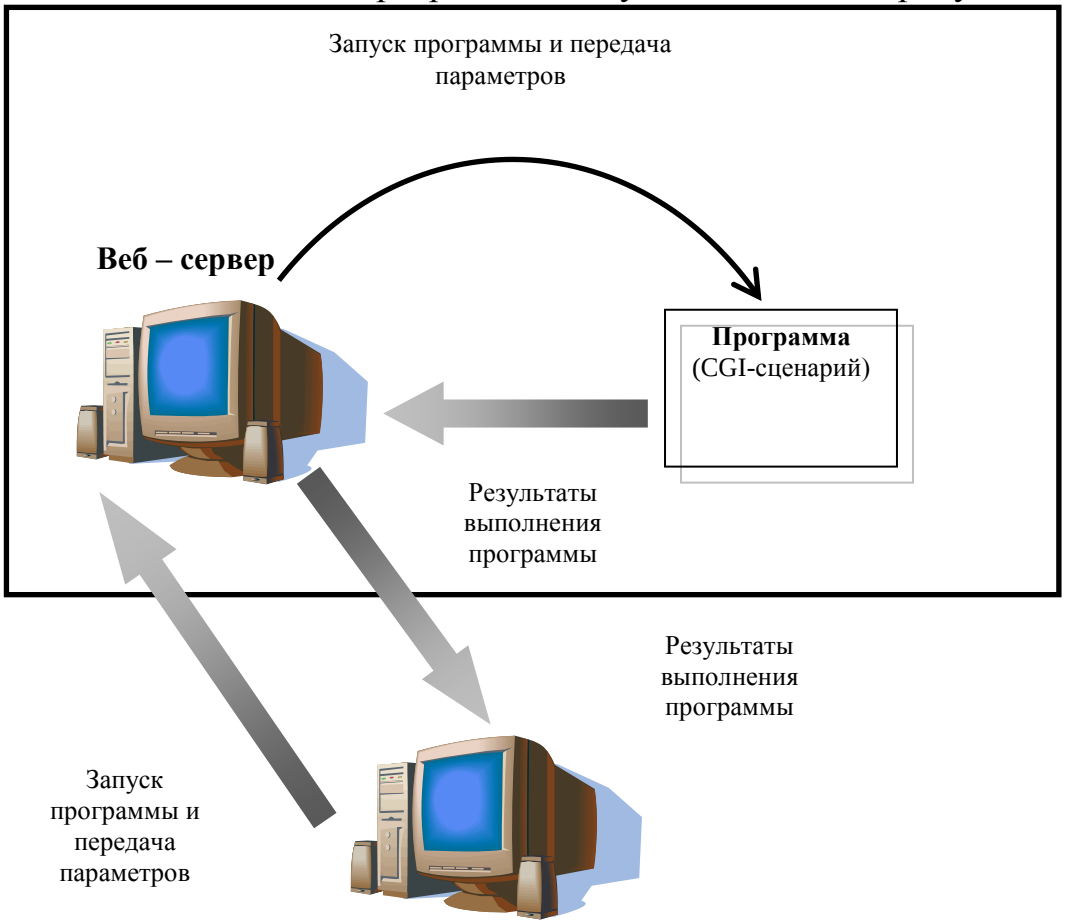


Рис 4.6. Взаимодействие клиента с программой, выполняющейся на сервере.

Насыщенные интернет-приложения

Насыщенное интернет-приложение (Rich Internet application) – еще один подход, который заключается в использовании Adobe Flash или Java-апплетов для полной или частичной реализации пользовательского интерфейса, поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов).

Возникновение данного подхода обусловлено тем, что в рамках веб-приложений с "тонким" клиентом взаимодействие пользователя с приложением реализуется в существенной степени через сервер, что требует отправки данных на сервер, получение ответа от сервера и перезагрузку страницы на стороне клиента.

При использовании Java-апплетов в состав HTML-документа включается специальный дескриптор, описывающий расположение файла, содержащего код апплета, на сервере. После того как клиент получает HTML-код документа, включающего апплет, он генерирует дополнительный запрос серверу. После того как сервер пересылает клиенту код апплета, сам апплет запускается на выполнение. Взаимодействие между клиентом и сервером при получении апплета показано на рисунке 4.7.

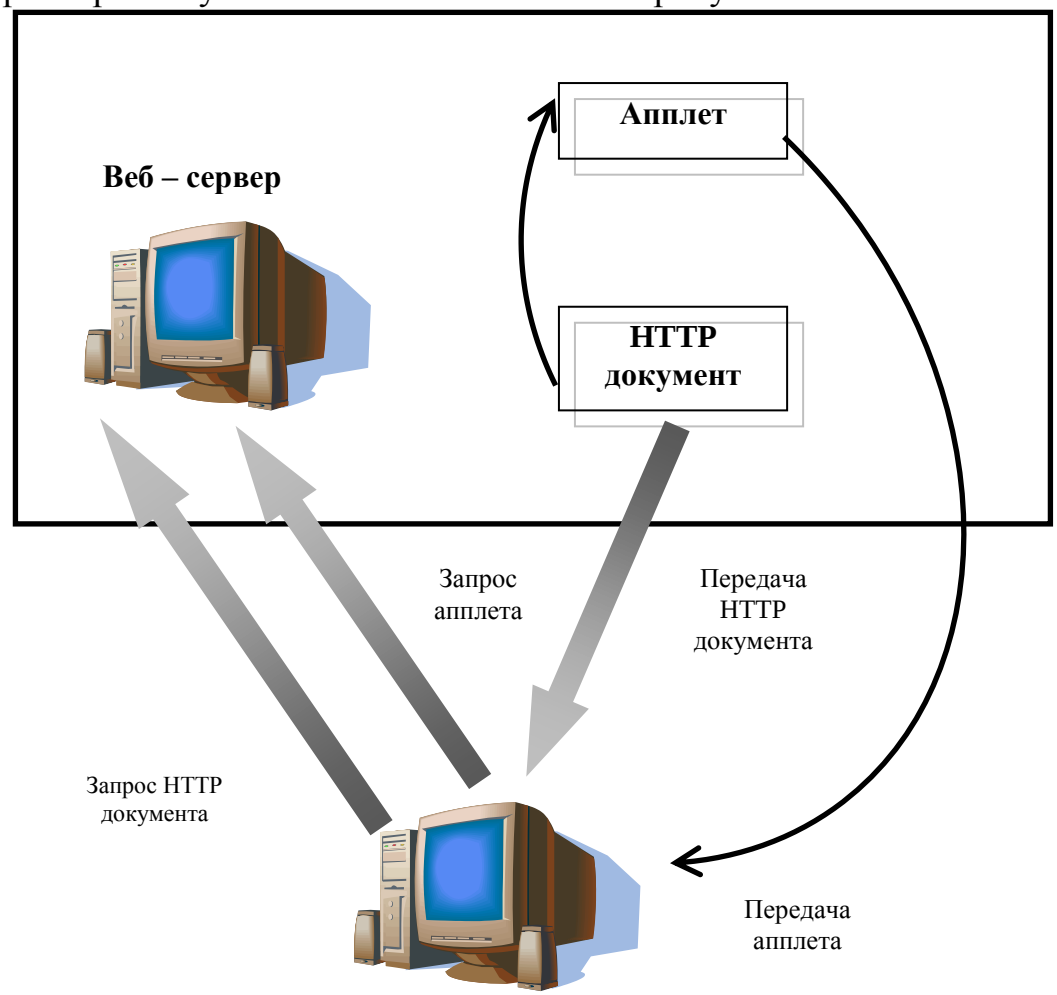


Рис. 4.7. Передача клиенту Java-апплета.

При использовании насыщенных интернет-приложений приходится сталкиваться со следующими проблемами:

– необходимость обеспечения безопасной среды выполнения («песочница»);

– для исполнения кода должно быть разрешено исполнение сценариев;

– потеря в производительности (т.к. выполняется на клиентской стороне);

– требуется много времени на загрузку;

Для разработки насыщенных интернет-приложений используются пакеты Curl, Adobe Flex и Microsoft Silverlight.

Ключевые слова: *Интернет, протокол HTTP, Java-апплет, веб-клиент, URL, HTML-документа, RFC-документов, интерфейс, протокол, Cookie, IP-адрес.*

Контрольные вопросы

1. Что такое интернет? Этапы развития интернета.
2. Какие достоинства и недостатки интернета вы знаете?
3. Опишите схему доступа абонента ТФОП в Интернет.
4. Понятие интерфейса и протокола.
5. Какова роль стандартизации в Интернет?
6. Опишите структуру URL?
7. Какие классы кодов ответа сервера, вы знаете?

ЧАСТЬ 2. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ

ГЛАВА 5. HTML5 – ЯЗЫК ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ ПЯТОЙ ВЕРСИИ

В предыдущих главах приводились примеры использования языка гипертекстовой разметки. В настоящей главе приведем более подробное описание языка гипертекстовой разметки. Напомним, что независимо от выбора платформы веб-разработки, язык HTML можно считать неотъемлемым инструментом верстки веб-страниц.

Любая веб-страница, вне зависимости от контента, стиля оформления, баннеров и прочих элементов внешнего вида, описывается обычным текстом.

Прочитав эту главу вы узнаете:

- Типы тегов
- Какие теги считаются устаревшими
- Какие новинки введены с появлением пятой версии HTML

Коротко о HTML

HTML (Hyper Text Markup Language) означает язык разметки гипертекста. Этот язык был разработан Тимом Бернерсом-Ли в рамках создания проекта распределенной гипертекстовой системы, которую он назвал World Wide Web (WWW) или Всемирная паутина. HTML предназначен для написания гипертекстовых документов, публикуемых в World Wide Web. Документ на языке HTML может включать следующие компоненты:

- стилизованный и форматированный текст,
- команды включения графических и звуковых файлов,
- гиперсвязи с различными ресурсами Internet.
- скрипты на языке JavaScript и VBScript.
- различные объекты, например Flash-анимацию

Документы HTML являются обычными текстовыми файлами, содержащими специальные теги (или управляющие элементы) разметки. Теги разметки указывают браузеру Web (программе пользователя для отображения web-страниц, например, Google Chrome, Internet Explorer, Mozilla FireFox, Safari, Netscape или Opera), как надо вывести страницу.

Файлы HTML обычно имеют расширения htm или html. Их можно создавать при помощи любого текстового редактора.

Язык HTML является подмножеством мощного языка SGML (Standard Generalized Markup Language), который широко используется в издательской деятельности. Основной выигрыш от использования этих языков состоит в переносимости текста между разными издательскими системами. Эта же особенность сохраняется и в HTML. Так, читая документ, пользователи

могут устанавливать способы выделения текста, гарнитуру и размер шрифтов по своему вкусу; они могут отменить просмотр рисунков.

В документе HTML можно выделить два основных блока: головная часть и тело документа. Содержимое головной части не выводится на экран пользователя, за исключением заголовка, в ней, как правило, указывают ключевые слова, авторов и другую служебную информацию, а также подключают внешние таблицы стилей и скрипты. В теле документа размещают ту информацию, которая будет выведена пользователю.

Мой первый сайт

В разных операционных системах имеются различные редакторы, которые можно использовать для создания документов HTML. Если вы используете:

- Microsoft Windows запустите редактор Notepad;
- Mac OS запустите редактор SimpleText;
- OSX запустите редактор TextEdit (обязательно измените настройки "Rich Text" или "Расширенный текст" на "Plain text" или "Простой текст" и отметьте пункт "Ignore rich text commands in HTML files" или "Игнорировать команды расширенного текстового формата в файлах HTML").

Файлы HTML можно создавать и в редакторе Microsoft Word, в котором имеется возможность сохранить документ как Web-страницу (в меню "Файл"), однако использовать эту возможность не рекомендуется. Во-первых, потому что HTML-код, генерируемый MS Word не оптимален и содержит множество ненужных элементов разметки, и, во-вторых, автоматическая генерация кода не будет способствовать изучению и правильному пониманию HTML.

Имеется также большое количество специализированных редакторов для создания файлов HTML, таких как FrontPage, Macromedia Dreamweaver или Adobe Web Bundle, которые обладают возможностью WYSIWYG (What You See Is What You Get - что видишь, то и получишь). С их помощью можно легко создавать документы HTML, при помощи кнопок и элементов меню, а не писать самому теги разметки. Однако, как уже отмечалось выше, тем, кто хочет стать технически грамотным разработчиком Web, настоятельно рекомендуется использовать простой текстовый редактор для начального изучения HTML.

Напечатайте следующий текст:

```
<html>
<head>
<title>Это заголовок страницы</title>
</head>
<body>
<h1>Здравствуйте!</h1>
<p>Это моя первая страница HTML. <b>Этот текст выводится жирным шрифтом.</b></p>
</body>
```

</html>

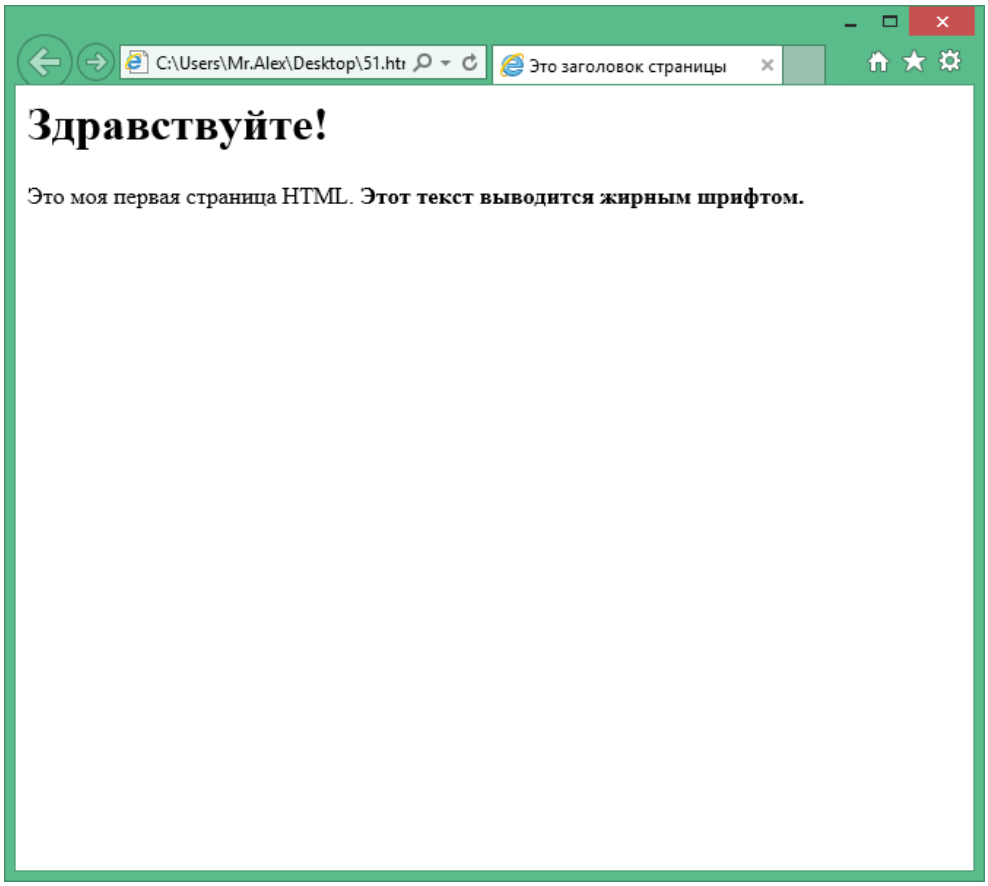


Рис. 5.1. Пример выполнения данного HTML-кода

Сохраните файл как "page1.htm".

При сохранении файла HTML можно использовать расширение .htm или .html. Расширение .htm было принято для старых версий операционных систем, которые допускали трехбуквенное расширение для файлов. В настоящее время практически все операционные системы не имеют подобного ограничения и можно использовать расширение .html.

Теперь посмотрите, как браузер отобразит вашу первую страницу. Запустите браузер Интернет. Выберите "Open" или "Open Page" ("Открыть" или "Открыть страницу") в меню File (Файл) браузера. Появится диалоговое окно. Выберите "Browse" или "Choose File" ("Просмотр" или "Выбрать файл") и найдите только что созданный файл HTML - "page1.htm" - выберите его и щелкните на кнопке "Open" ("Открыть"). В диалоговом окне должен появиться адрес, например "C:\MyDocuments\page1.htm". Щелкните на кнопке ОК, и браузер выведет на экран вашу страницу.

Разбор примера

Ваш первый HTML-документ начинается с тега <html>, который сообщает браузеру о начале документа HTML и заканчивается тегом </html>, который информирует браузер о достижении конца документа HTML.

Текст между тегами <head> и </head> является информацией заголовка документа. Эта информация не выводится в окне браузера.

Текст "Это заголовок страницы" между тегами <title> и </title> является заголовком документа. Этот заголовок выводится в строке заголовка окна браузера.

Текст между тегами <body> и </body> является текстом, который будет выведен в окне браузера. Текст "Здравствуйте!" между тегами <h1> и </h1> будет отображен стилем заголовка, обычно жирным шрифтом большего размера.

Тег означает, что начинается новый параграф, тег означает конец параграфа.

Текст "Этот текст выводится жирным шрифтом." между тегами и будет выведен жирным шрифтом.

Элементы HTML

Страницы Всемирной паутины отображаются при помощи документов HTML, которые являются обычными текстовыми файлами, содержащими специальные теги разметки. Теги разметки определяют элементы HTML, из которых состоят документы HTML.

Коротко о тегах

Теги HTML используются для выделения элементов HTML. Обычно теги HTML используются парами и заключены между двумя символами угловых скобок <(начальный тег)> и </(конечный тег)>. Текст между начальным и конечным тегами является содержимым элемента. Некоторые теги не имеют конечного, например, тег принудительного переноса строки <br>, для таких тегов рекомендуется использовать следующее написание <br/>.

Регистр символов для отображения тегов не важен, например, и <P> означает одно и то же. Однако в этом курсе используется нижний регистр для написания тегов. Это связано с тем, что консорциум WWW (W3C), который занимается стандартизацией спецификации HTML, рекомендует использовать теги в нижнем регистре, поскольку в следующем поколении стандартов будет именно такое требование.

Коротко об элементах HTML

Рассмотрим тот же пример документа HTML:

```
<html>
<head>
<title>Это заголовок страницы</title>
</head>
<body>
<h1>Здравствуйте!</h1>
<p>Это моя первая страница HTML.
<b>Этот текст выводится жирным шрифтом.</b>
</p>
</body>
</html>
```

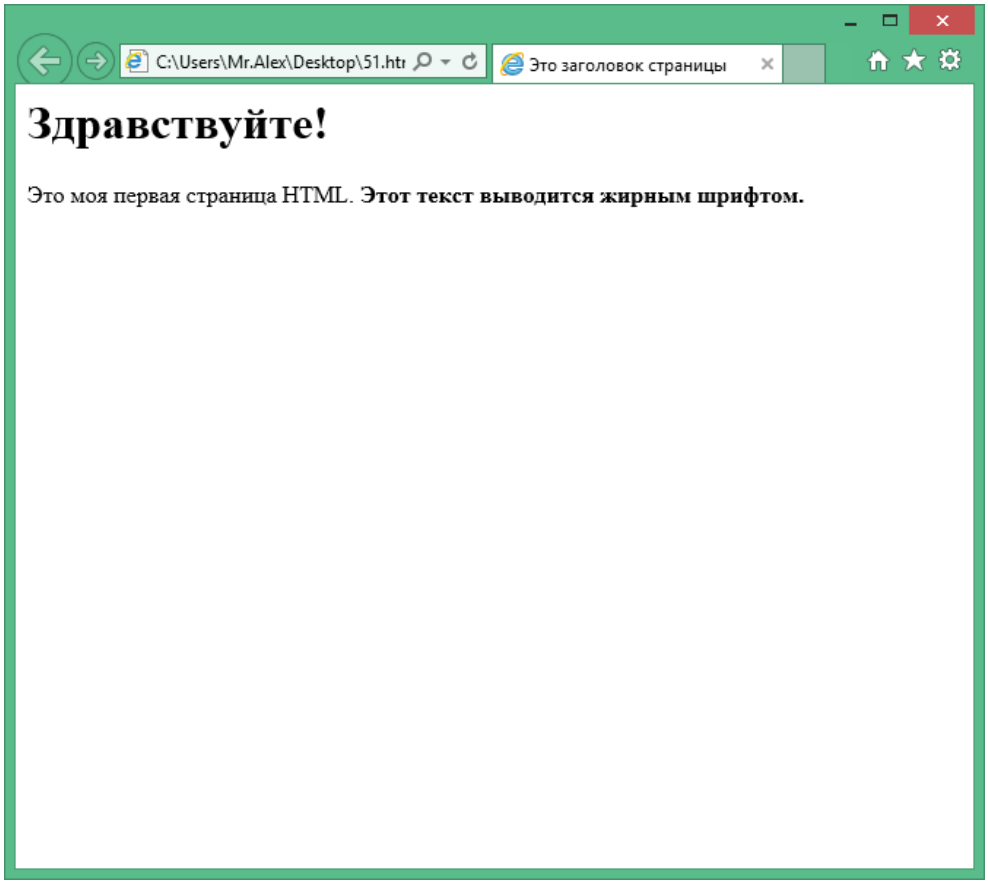


Рис. 5.2 Пример выполнения данного HTML-кода

Элементом HTML является:

<h1>Здравствуйте!</h1>

Этот элемент начинается с тега <h1>, имеет содержимое "Здравствуйте!" и заканчивается тегом </h1>.

Также элементом HTML является:

Это моя первая страница HTML.

Этот текст выводится жирным шрифтом.

Этот элемент, начинается с начального тега , заканчивается конечным тегом и означает, что содержимое элемента "Это моя первая страница HTML. Этот текст выводится жирным шрифтом. " является отдельным параграфом. При этом внутри этого элемента находится другой элемент:

Этот текст выводится жирным шрифтом.

Этот элемент HTML начинается с начального тега: Содержимым элемента HTML является: Этот текст выводится жирным шрифтом. Этот элемент HTML заканчивается конечным тегом . Назначение тега состоит в определении элемента HTML, который должен выводиться жирным шрифтом.

Все описанные элементы HTML содержатся в элементе:

<body>

<h1>Здравствуйте!</h1>

Это моя первая страница HTML.
Этот текст выводится жирным шрифтом.

</body>

Этот элемент HTML начинается с начального тега <body>, и заканчивается конечным тегом </body>. Назначение тега <body> состоит в определении элемента HTML, который содержит основную часть (или тело) документа HTML.

Атрибуты тегов

Теги могут иметь атрибуты, которые предоставляют дополнительную информацию об элементах HTML. Атрибуты всегда используются в виде пары "имя/значение". Общий формат задания атрибутов имеет вид:

<имя_тега имя_атрибута="значение">

Например, тег:

<body bgcolor="red">

означает, что цвет фона страницы должен быть красным.

А тег:

означает, что параграф необходимо выровнять по центру страницы отображения браузера.

Атрибуты всегда помещаются в начальном теге элемента HTML. Значения атрибутов всегда полезно заключать в кавычки. Наиболее широко используются двойные кавычки, но одиночные кавычки также допустимы.

В некоторых редких ситуациях, когда, например, значение атрибута само содержит кавычки, необходимо использовать одиночные кавычки:

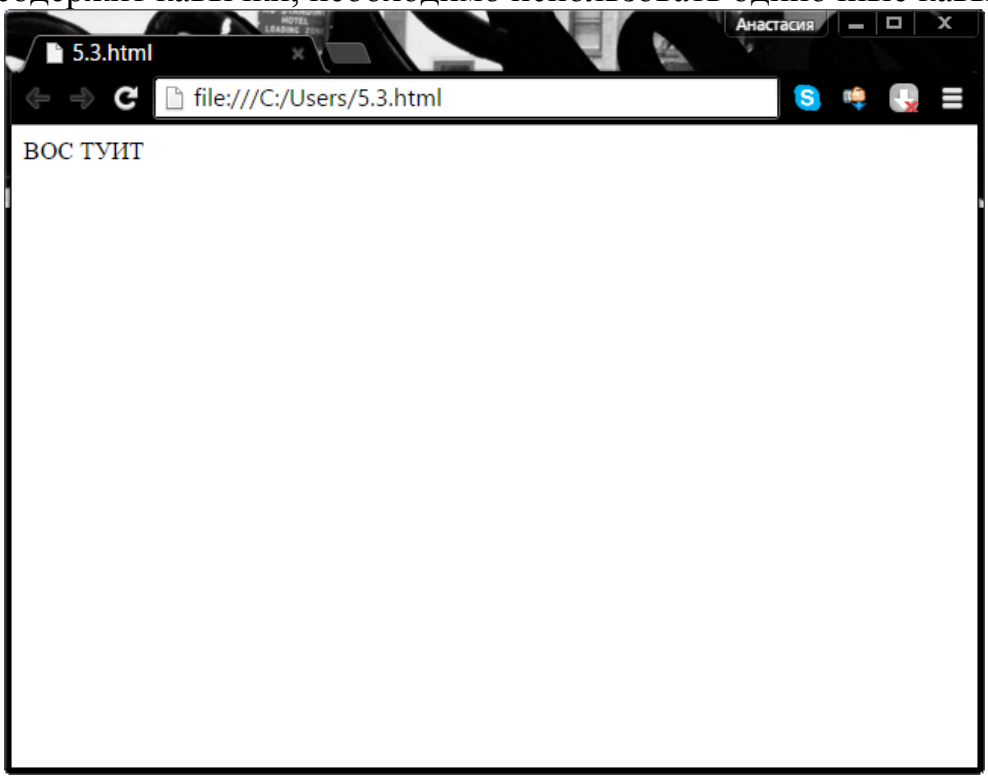


Рис. 5.3 Пример выполнения данного HTML-кода

Кроме атрибутов, записываемых вышеописанным способом, для некоторых элементов определены специальные флаги, которые просто указываются как <тег имя_флага>.

```
<html>
<body>
<p>Попробуйте исправить текст в этих полях ввода</p>
<form action="index.php" method="GET">
<input type="text" name="blocked"
 value="Пример поля ввода, у которого указан флаг readonly"
  readonly size="100"><br/><br/>
<input type="text" name="unblocked"
 value="Обычное поле ввода у которого нет флагов"
 size="100">
</form>
</body>
</html>
```

Основные теги HTML

Параграфы.

Прежде чем изучать теги форматирования HTML, посмотрим как введенный текст отобразится, если не будут применены никакие теги кроме тегов <html> и <body>. Следующий пример демонстрирует такой документ HTML

```
<html>
<body>
Этот текст будет показан в окне браузера.
</body>
</html>
```

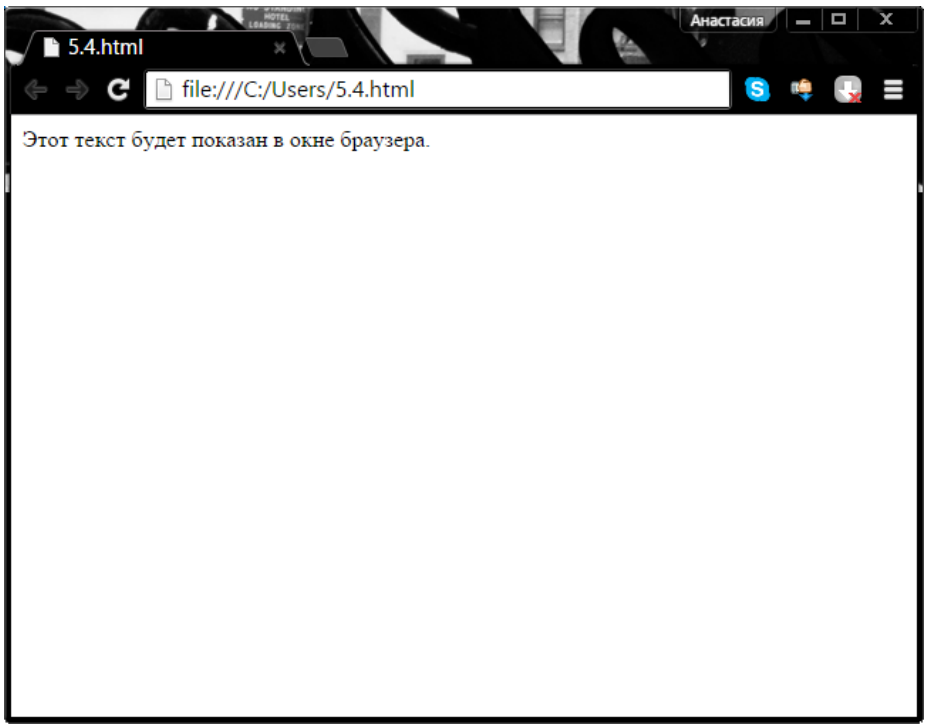


Рис. 5.4 Пример выполнения данного HTML-кода

Этот простой пример документа HTML, который содержит минимальное количество тегов HTML и демонстрирует, как текст внутри элемента body отображается в браузере.

Если ввести большой объем текста таким способом, то читать его будет очень неудобно. Логичнее разбить его на параграфы, как в книге, которые повышают читабельность текста, и кроме того выделяют смысловые блоки.

Следующий пример показывает, как отображаются параграфы

```
<html>
<body>
<p>Это параграф 1.</p>
<p>Это параграф 2.</p>
<p>Это параграф 3.</p>
</body>
</html>
```

Рис. 5.5 Пример выполнения данного HTML-кода

Этот пример демонстрирует, как в браузере выводится текст внутри элементов параграфа. Можно видеть, что по умолчанию текст каждого параграфа выводится в виде отдельного блока. Каждый из таких блоков отделяется от предыдущих и последующих блоков страницы пустой строкой. Однако отображение параграфа браузером может быть легко изменено посредством таблицы стилей.

Можно заметить, что параграфы можно записывать без закрывающего тега </p>, однако лучше этого не делать, в следующей версии HTML все теги нужно будет закрывать.

В разных браузерах на разных мониторах с разным разрешением страница будет отображаться по-разному, поэтому не стоит форматировать при помощи добавления пустых строк и пробелов. Любое число пробелов заменяется одним.

Использование пустых параграфов <p> для вставки пустых строк является плохим стилем, вместо этого используйте тег <br />.

Заголовки.

Заголовки определяются с помощью тегов от <h1> до <h6>. <h1> определяет заголовок самого большого размера, а <h6> определяет заголовок самого маленького размера.

```
<h1>Это заголовок первого уровня</h1>
<h2>Это заголовок второго уровня</h2>
<h3>Это заголовок третьего уровня</h3>
<h4>Это заголовок четвертого уровня</h4>
<h5>Это заголовок пятого уровня</h5>
<h6>Это заголовок шестого уровня</h6>
```

Заголовки автоматически отделяются дополнительными промежутками от остальных элементов документа.

Переносы строк.

Для переноса внутри параграфа используется тег <br>, который выполняет принудительный перенос строки.

```
<html>
<body>
<p>Это <br>пара<br>граф с переносами строк</p>
</body>
</html>
```

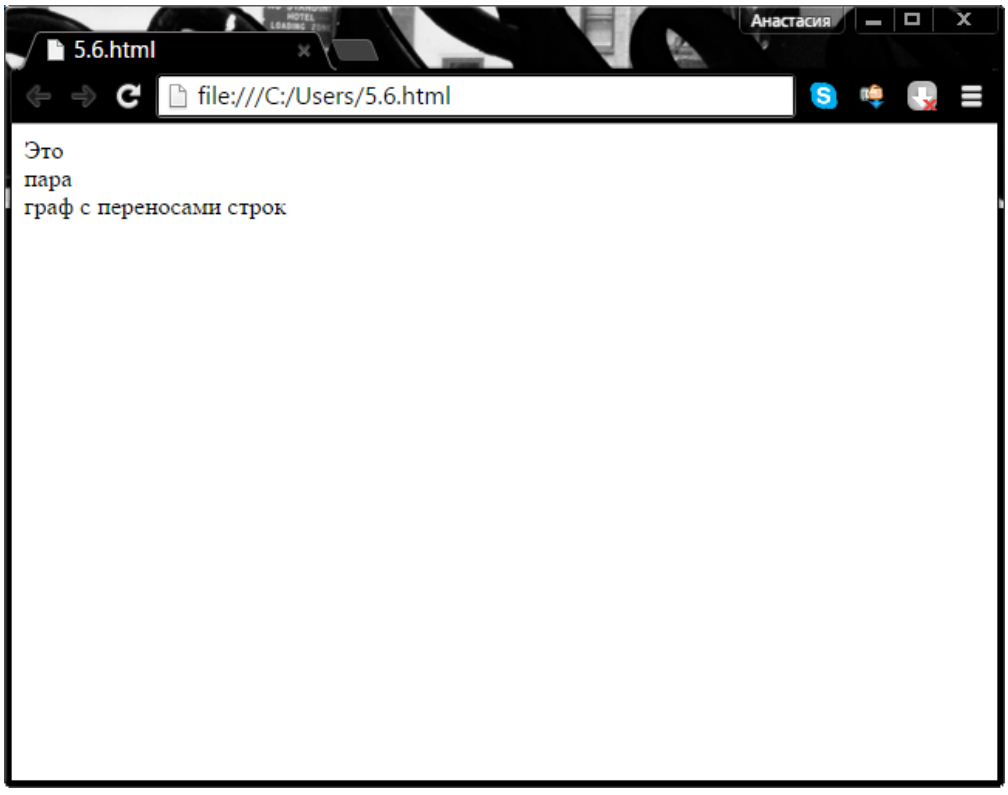


Рис. 5.6 Пример выполнения данного HTML-кода

Тег
 не имеет закрывающего тега. Поэтому для совместимости с будущими версиями стандарта рекомендуется следующее написание тега

Горизонтальная линейка

Разделять различные элементы можно при помощи горизонтальной линейки, для этого используйте тег <hr>:

```
<html>
<body>
<p>
Этот параграф отобразится сверху горизонтальной полосы.
</p>
<hr>
<p>
Этот параграф отобразится снизу горизонтальной полосы.
</p>
</body>
</html>
```

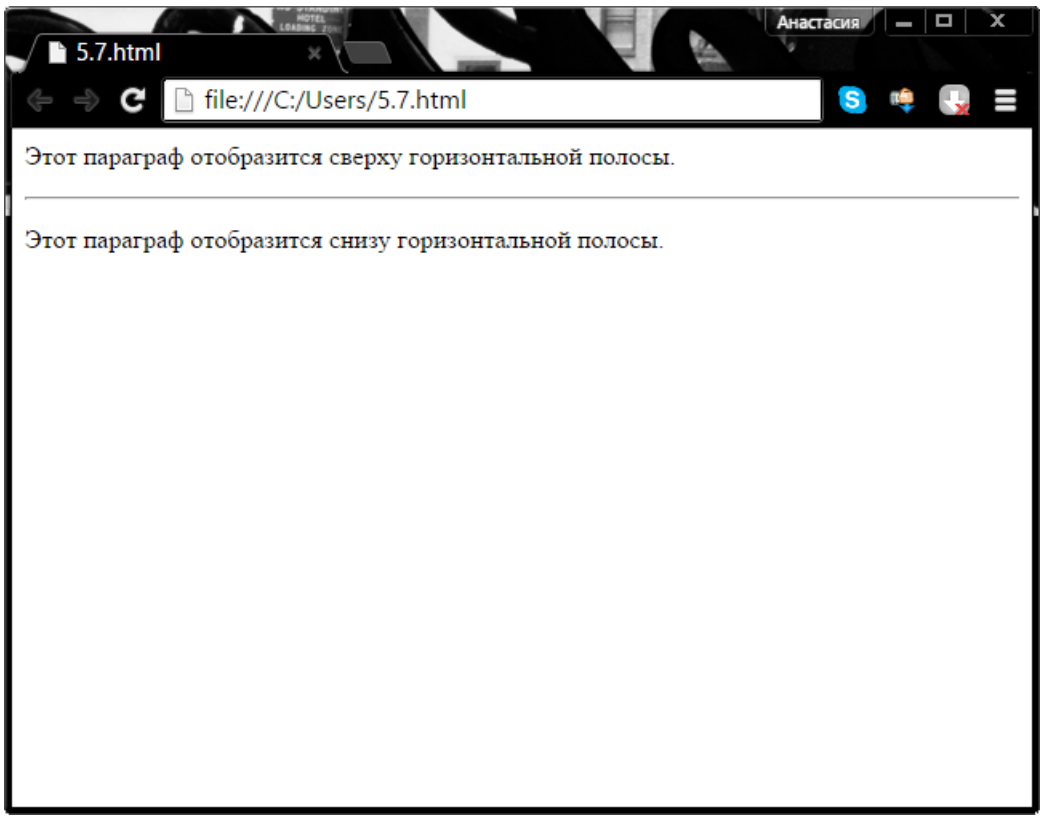


Рис. 5.7 Пример выполнения данного HTML-кода

Тег <hr> не имеет закрывающего тега. Поэтому для совместимости с будущими версиями стандарта рекомендуется следующее написание тега <hr />. Для этого тега определен ряд атрибутов, но они являются устаревшими. И хотя их применение возможно, но консорциум W3C их использовать не рекомендует. Вместо них следует использовать таблицы стилей.

Комментарии в HTML

Тег комментария используется для вставки комментариев в исходный код HTML. Комментарии будут проигнорированы браузером. Комментарии

можно использовать для пояснения кода, что может помочь при редактировании исходного кода в будущем.

<!-- Это комментарий -->

Вот пример:

<html>

<body>

Этот текст будет показан в окне браузера.

<!-- Этот текст не будет показан, это комментарий. -->

</body>

</html>

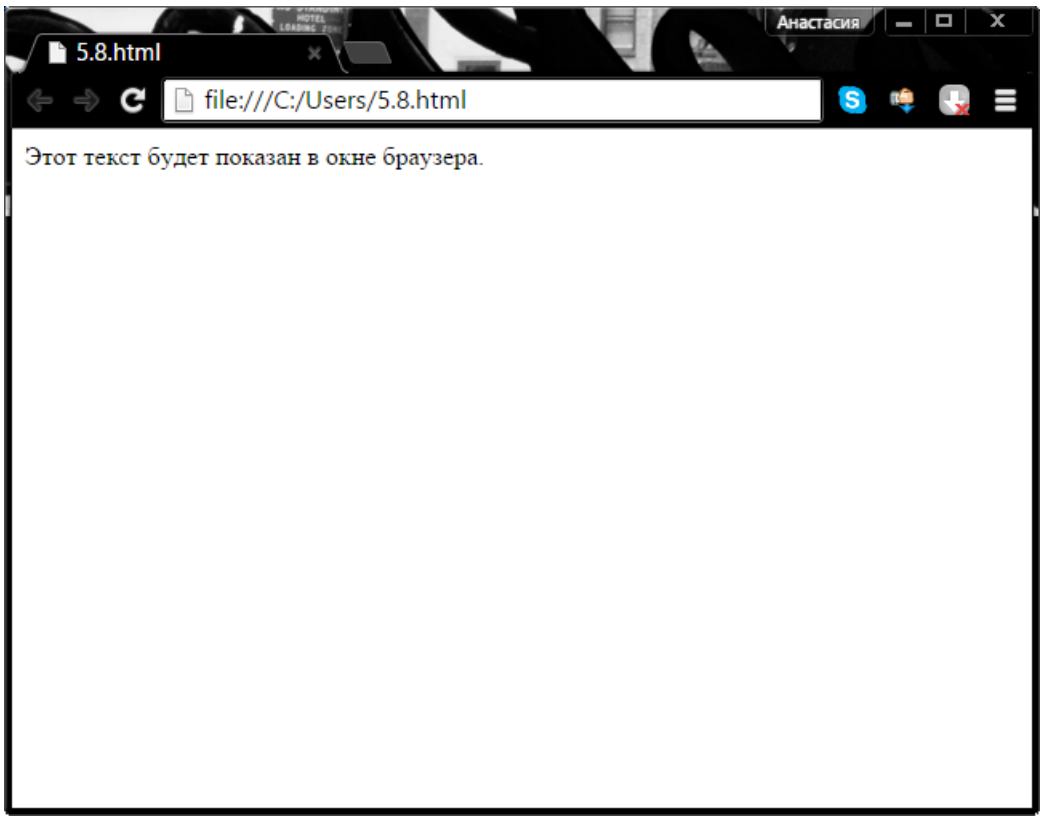


Рис. 5.8 Пример выполнения данного HTML-кода

Дополнительные примеры

Лучшим способом изучения HTML является работа с примерами. Рассмотрим несколько примеров, которые иллюстрируют некоторые элементы форматирования документов.

Дополнительные параграфы

Этот пример демонстрирует некоторые особенности поведения по умолчанию элементов параграфа.

<html>

<body>

<p>

Этот параграф

содержит много строк

в исходном коде,

но браузер

это игнорирует.

```
</p>
<p>
Этот параграф
содержит     много пробелов
в исходном    коде,
но     браузер
это игнорирует.
</p>
</body>
</html>
```

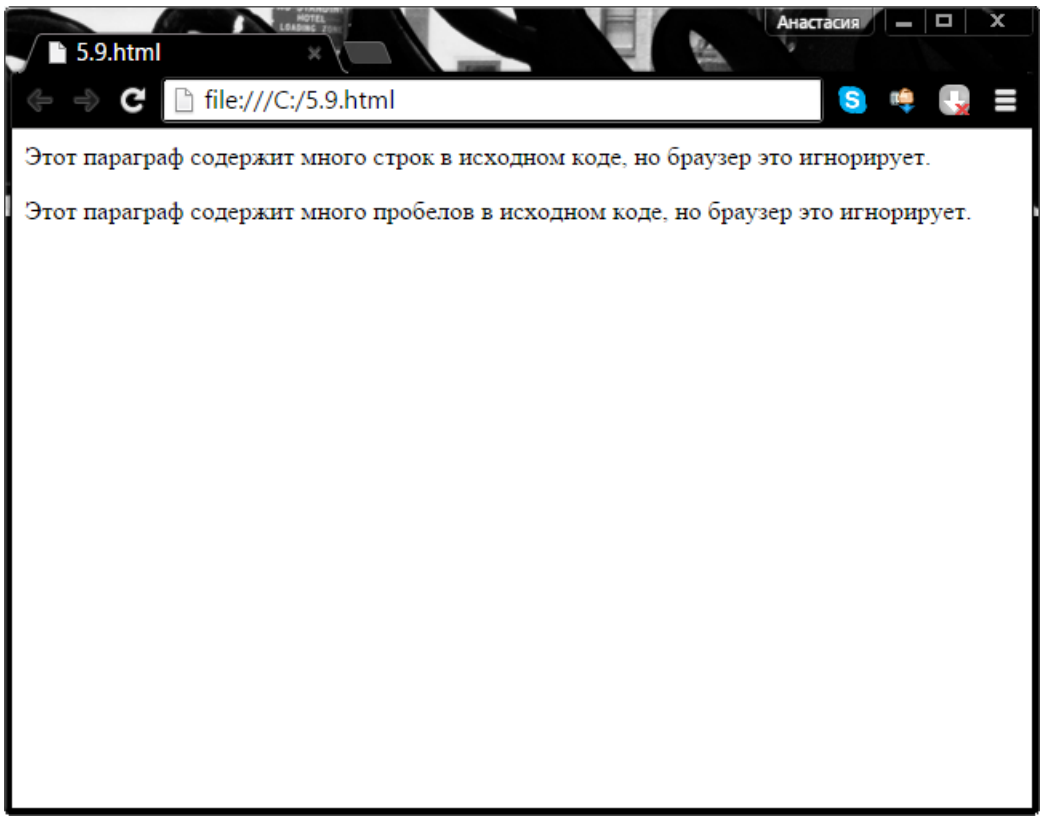


Рис.5.9 Пример выполнения данного HTML-кода

Перенос строк

Этот пример демонстрирует использование переноса строк в документе HTML.

```
<html>
<body>
<p>
Чтобы выполнить перенос<br>строк<br>в
<br>параграфе,<br>используйте тег br.
</p>
</body>
</html>
```

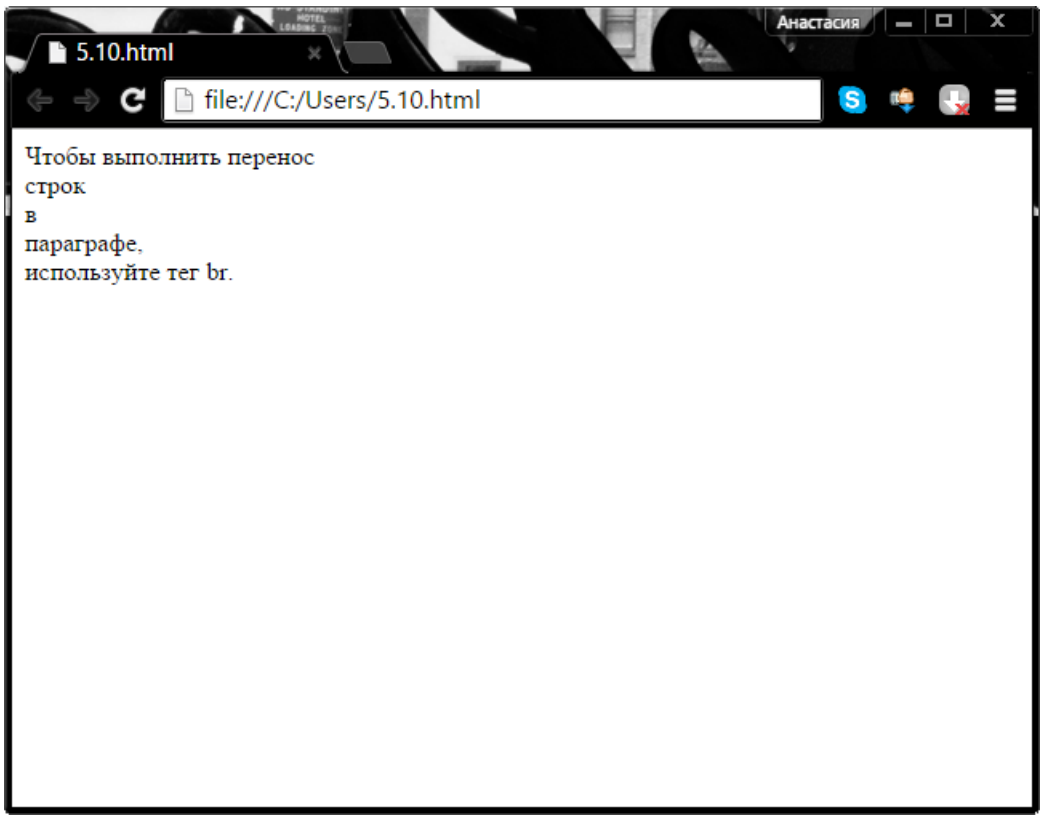


Рис. 5.10 Пример выполнения данного HTML-кода

Этот пример демонстрирует некоторые проблемы с форматированием HTML. Попробуем сформатировать стихи:

```
<html>
<body>
<p>
  В лесу родилась елочка.
  В лесу она росла.
  Зимой и летом стройная,
  Зеленая была.
</p>
<p>
Обратите внимание, что браузер просто проигнорировал использованное форматирование!
</p>

</body>
</html>
```

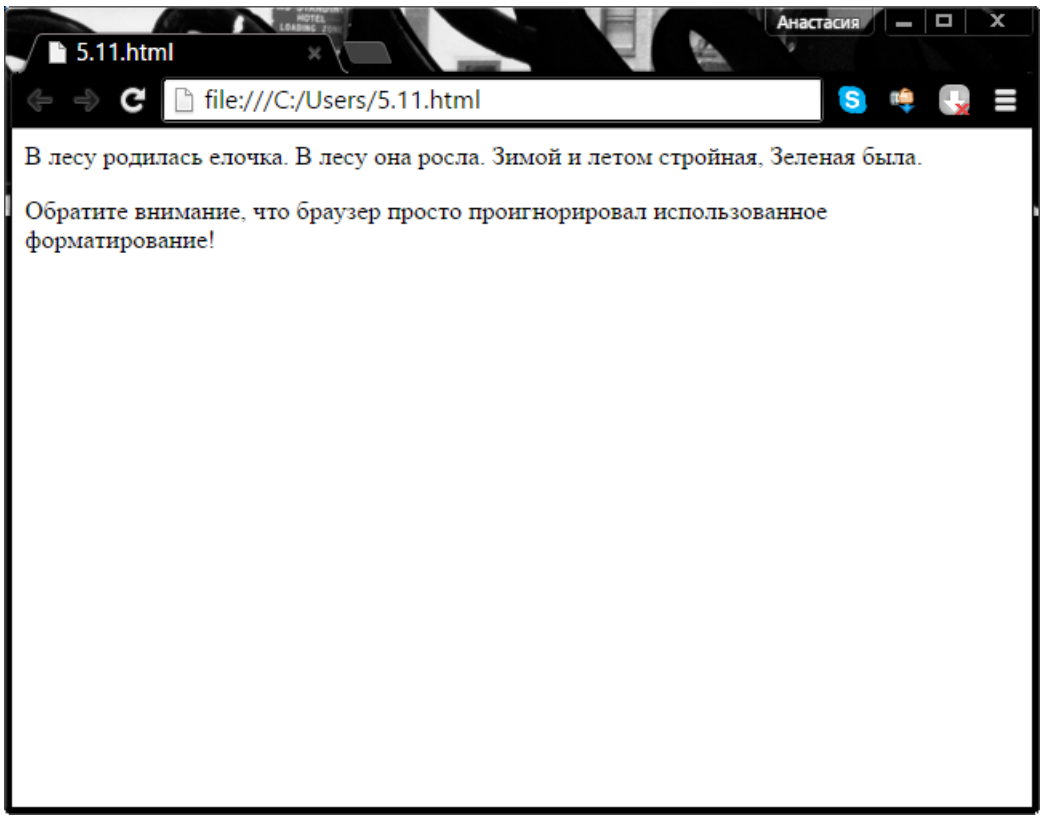


Рис. 5.11 Пример выполнения данного HTML-кода

Заголовки

Этот пример демонстрирует теги, которые выводят заголовки с атрибутом выравнивания в документе HTML.

```
<html>
<body>
<h1 align="center">
Это заголовок 1, он выровнен по центру страницы.
</h1>
<p>
Используйте теги заголовков только для заголовков.
Не используйте их просто для того, чтобы выделить что-то жирным шрифтом.
Используйте для этого другие теги.
</p>
</body>
</html>
```

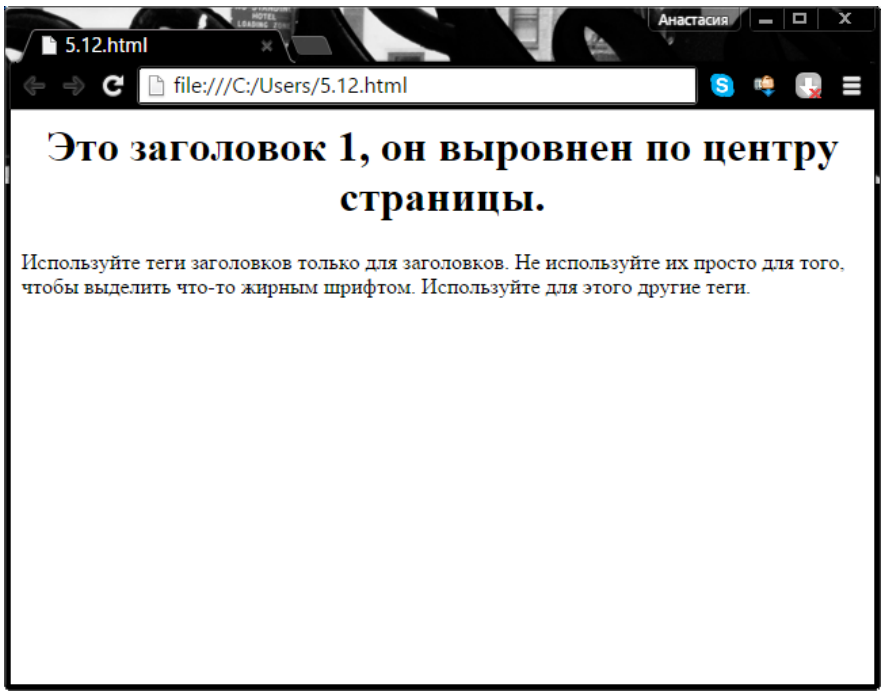


Рис. 5.12 Пример выполнения данного HTML-кода

Фоновый цвет

Этот пример демонстрирует использование цветного фона на странице HTML. При выборе фона всегда проверяйте, чтобы текст был хорошо читаем!

```
<html>
<body bgcolor="yellow">
<h2>Смотри: Цветной фон!</h2>
</body>
</html>
```

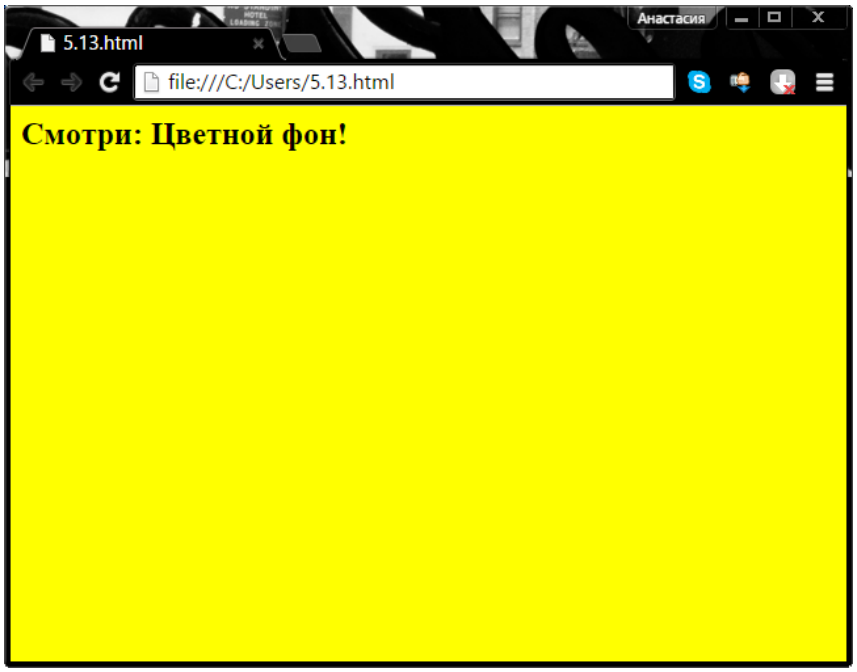


Рис. 5.13 Пример выполнения данного HTML-кода

Таблица основных тегов HTML

Тег	Описание
<html>	Определяет документ HTML
<body>	Определяет основную часть или тело документа
<h1> -- <h6>	Определяет заголовки с 1 по 6
	Определяет параграф
 	Вставляет единичный перенос строки
<hr>	Определяет горизонтальную линейку
<!-->	Определяет комментарий

Чтобы отобразить неотформатированный текст, достаточно просто ввести его между тегами начала и конца документа <body> </body>. При обработке такой страницы браузер найдет и выведет весь этот текст. Если необходимо, чтобы к тексту было применено какое-либо форматирование, например, выделение полужирным или курсивом, необходимо использовать соответствующие теги форматирования. При этом форматируемый текст помещается между тегами.

Форматирование текста
<html>
<body>

Если необходимо чтобы к тексту было применено какое-либо форматирование, например, выделение полужирным или курсивом, необходимо использовать соответствующие теги форматирования.
При этом форматируемый текст помещается между тегами.

</body>
</html>

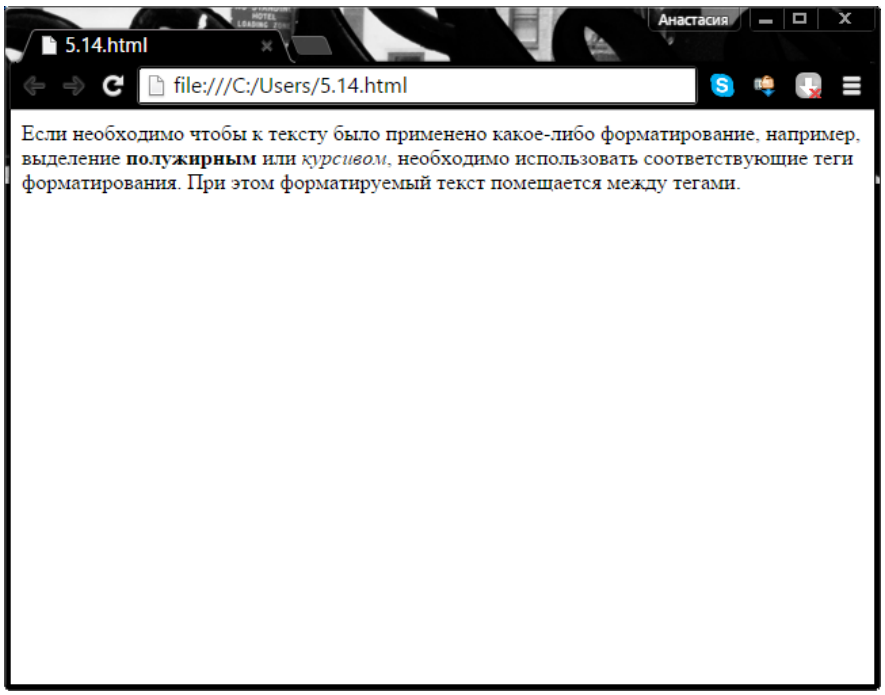


Рис. 5.14 Пример выполнения данного HTML-кода

Также для выделения текста используются теги и , данные теги являются контейнерами и требуют закрывающегося тега. Тег сообщает браузеру, что на заключенном в нем тексте необходимо сделать сильное ударение. Обычно визуальные браузеры отображают содержимое данного тега полужирным шрифтом, но это может быть легко изменено с помощью таблицы стилей. Так как данный тег является структурным, он несет смысловую нагрузку, в отличие от тега <b>, который лишь форматирует текст полужирным, то его использование предпочтительней. Тег тоже акцентирует внимание на заключенном внутри тексте, но он считается менее сильным ударением. Отображается он, как правило, курсивом. По тем же причинам использование предпочтительней чем применение <i>. Сравните как будет отформатирован следующий текст.

```
<html>
<body>
<p><strong>Данный параграф отформатирован тегом strong</strong></p>
<p><b>А этот тегом b, внешне они не отличаются.</b></p>
<p><em>Данный параграф отформатирован тегом em</em></p>
<p><i>А этот тегом i, внешне они не отличаются.</i></p>
</body>
</html>
```

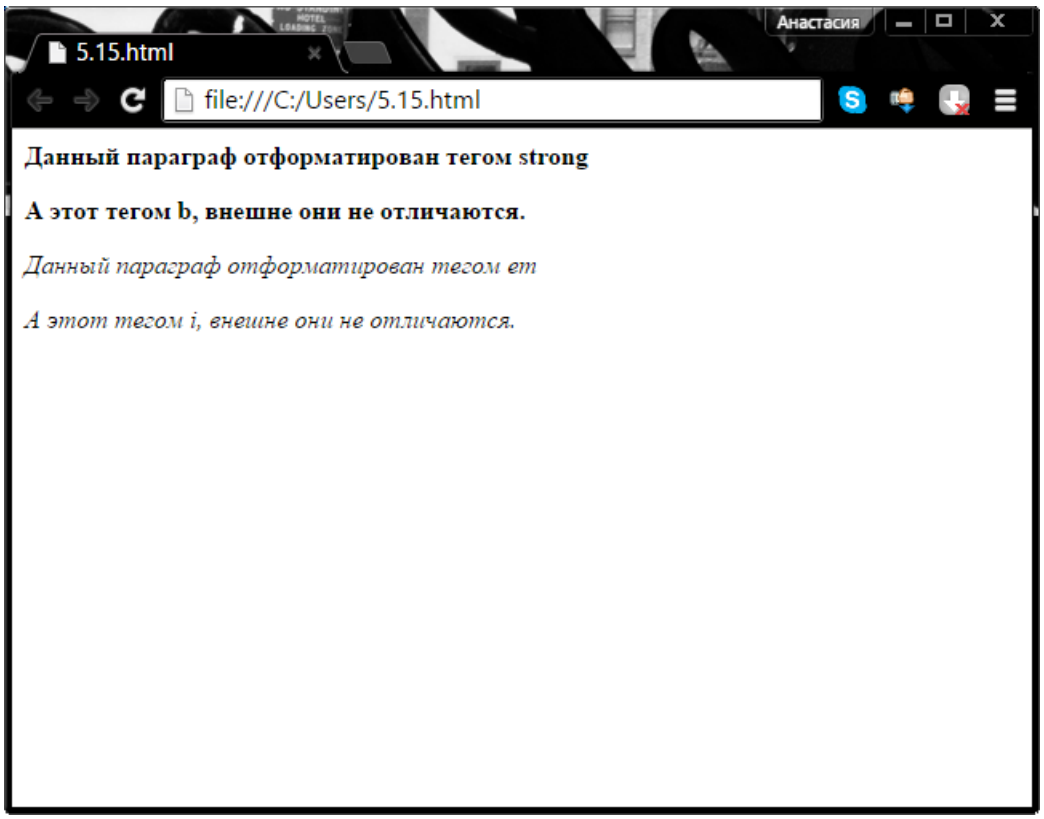


Рис. 5.15 Пример выполнения данного HTML-кода

Еще одной парой тегов форматирования являются теги <big> и <small>. Первый выводит текст увеличенным по сравнению со стандартным, а второй уменьшенным. Вместо тега <big> рекомендуется использовать теги или теги заголовков, так как они несут и структурную нагрузку. Тег <small> является тегом по смыслу противоположным и , он деакцентирует внимание на тексте. По поводу применения этого тега следует сделать одно замечание: на разных компьютерах установлено различное разрешение экрана и в браузере может быть установлен различный размер основного шрифта и сильное уменьшение текста может сделать текст не читаемым. Следующий пример демонстрирует вид текста, отформатированного с помощью этих тегов.

```
<html>
<body>
<p><big>Данный параграф отформатирован тегом big </big></p>
<p><small>Данный параграф отформатирован тегом small </small></p>
<p>А в данном параграфе теги не применяются</p>
</body>
</html>
```

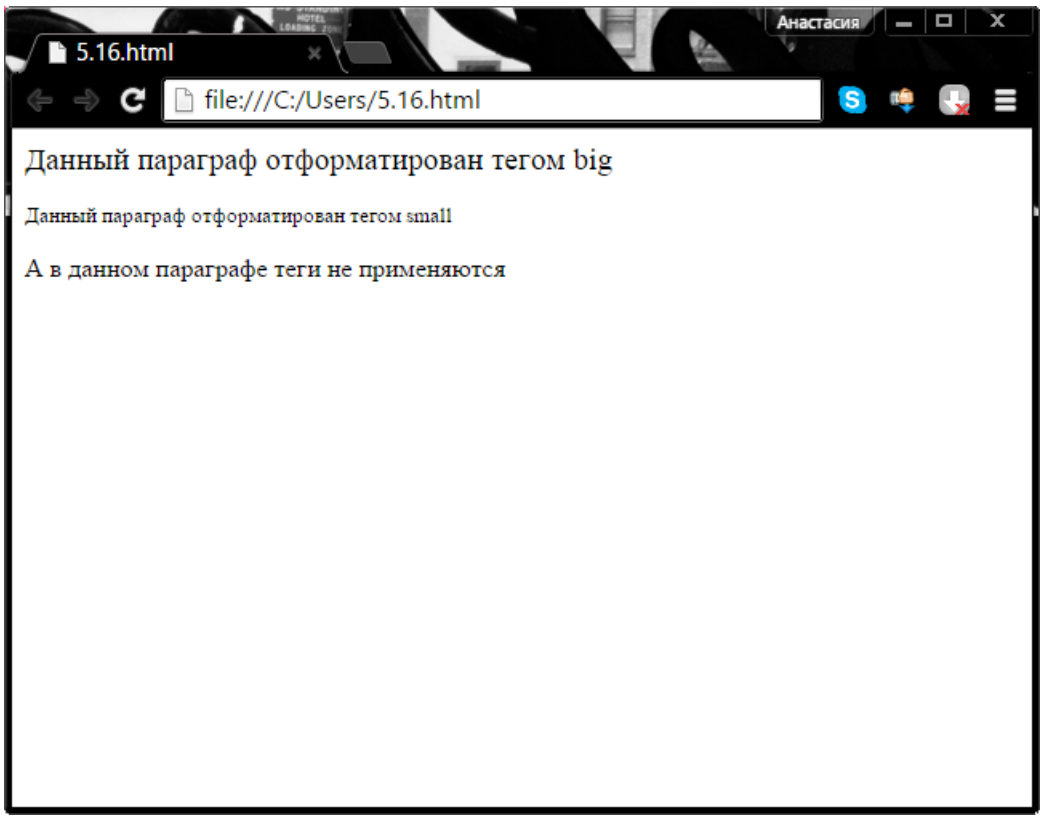


Рис. 5.16 Пример выполнения данного HTML-кода

Другой парой полезных тегов являются <sup> - верхний индекс и <sub> - нижний индекс, которые могут быть полезными при написании математических и химических формул. Сравните формулы, набранные различным способом:

<html>
<body>
Формула воды H2O. В данном параграфе формула набрана без использования тега sub
Формула воды H₂O.
В данном параграфе формула набрана с использованием тега sub Формула выглядит более привычно.
2^4=16.
В данном параграфе формула набрана без использования тега sup
2⁴=16. В данном параграфе формула набрана с использованием тега sup.
Формула выглядит более привычно.
</body>
</html>

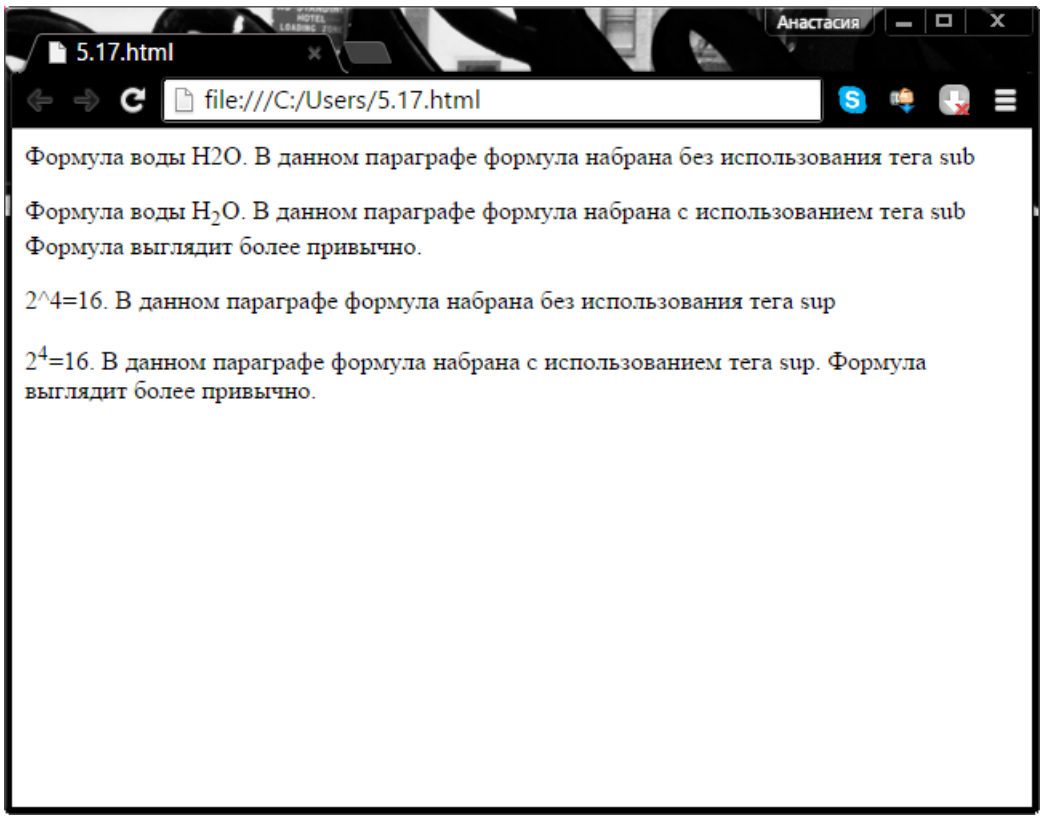


Рис. 5.17 Пример выполнения данного HTML-кода

В некоторых случаях, например для вывода кода программ, полезным будет использование тега <pre>, который сообщает браузеру, что находящийся внутри текст должен быть выведен как есть. При этом будут сохранены все пробелы, переносы строк и прочие символы, которые обычно при выводе браузером не отображаются.

Этот пример показывает, как можно управлять переносами строк и пробелами с помощью тега <pre>.

```
<html>
<body>
<pre>
Это
предварительно форматированный текст.
Он сохраняет     как пробелы,
так и переносы строк.
</pre>
</body>
</html>
```

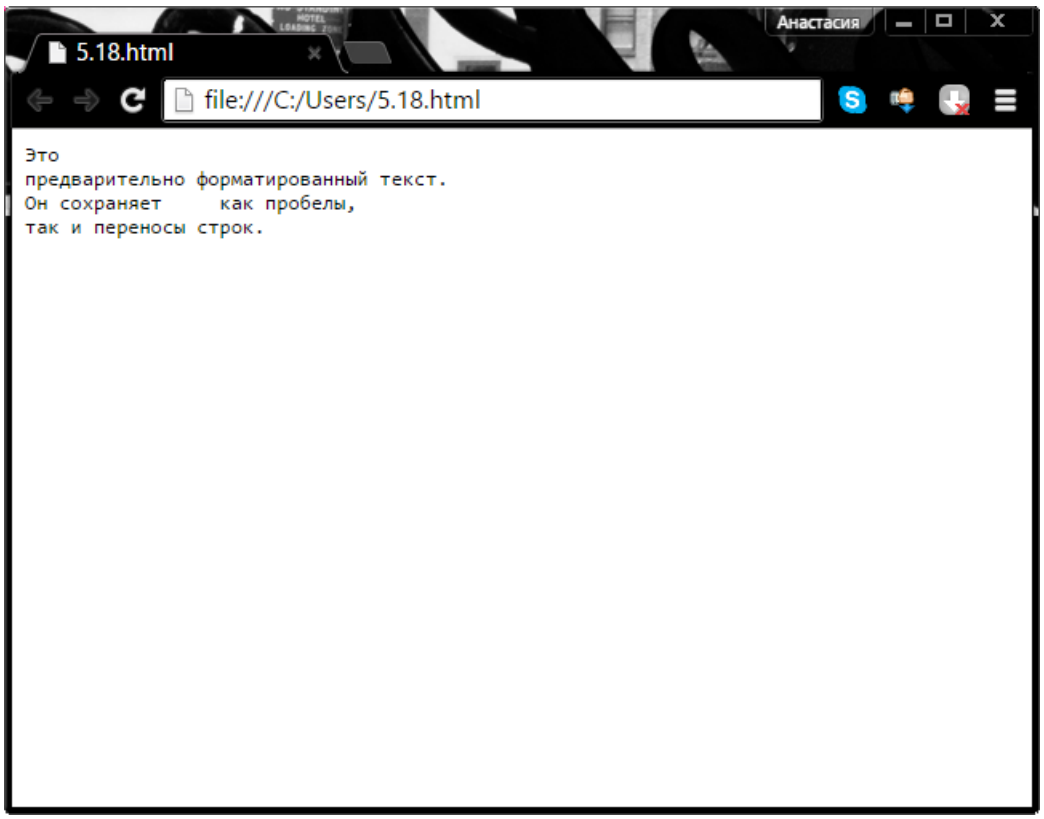


Рис. 5.18 Пример выполнения данного HTML-кода

Сравните написание кода программы с применением тега <pre> и без его использования.

```
<html>
<body>
<pre>
// Данный фрагмент набран с использованием тега pre
for (int i = 1; i < 10; i++)
   {
     printf ("i=%i\n", i);
   }
</pre>

<p>
// Здесь тег pre не применялся
for (int i = 1; i < 10; i++)
   {
     printf ("i=%i\n", i);
   }
</p>
</body>
</html>
```

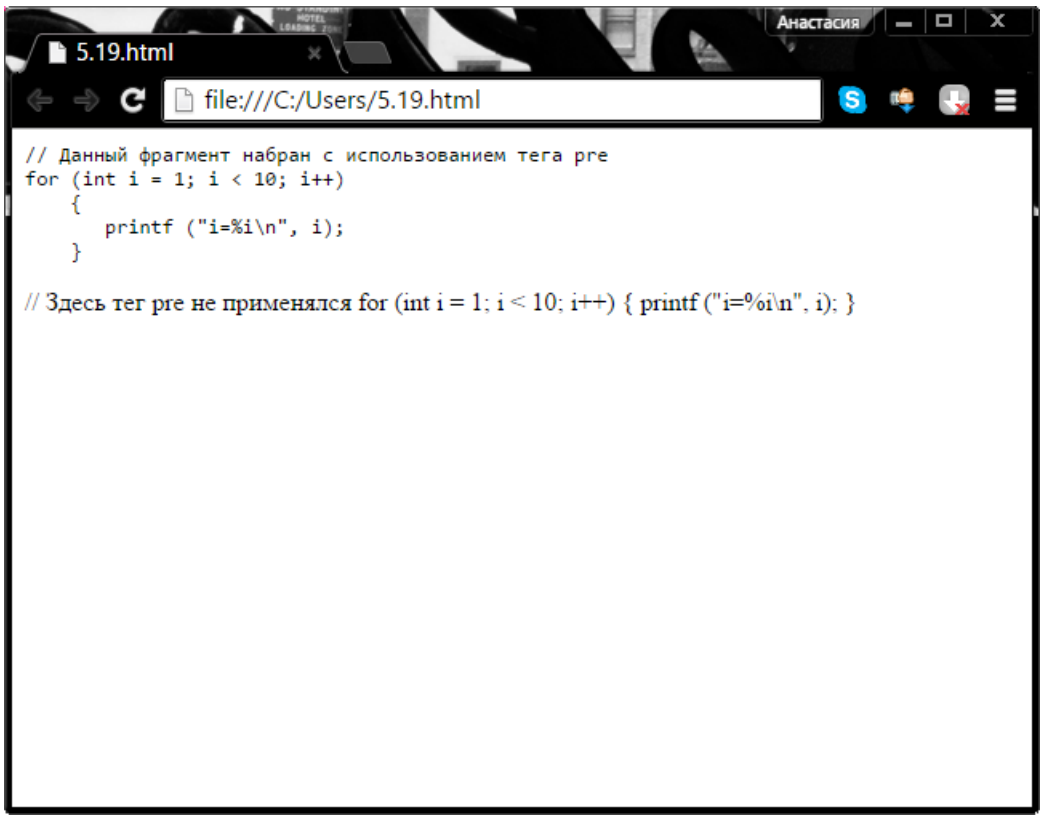


Рис. 5.19 Пример выполнения данного HTML-кода

Тег <code> маркирует код компьютерной программы. Браузеры обычно отображают его моноширинным шрифтом. Этот элемент является структурным, поэтому его использование предпочтительней использования тега <tt>, который маркирует моноширинный текст. Также следует отметить, что тег <code> лучше использовать внутри тега <pre>, так как в противном случае, все множественные пробелы будут заменены одним. Тег <kbd> маркирует текст, введенный с клавиатуры. Отображается он также, как и текст внутри тега <code>

<html>
<body>

Данные примеры демонстрируют то, как будет представлен текст при использовании разных тегов

<pre><code>
// отформатировано с помощью pre и code
class helloworld {
  public static void main(string[] args) {
    system.out.println("hello world!");
  }
}
</code></pre>

```
<pre><tt>
// отформатировано с помощью pre и tt
class helloworld {
  public static void main(string[] args) {
    system.out.println("hello world!");
  }
}
</tt></pre>

<code>
// отформатировано с помощью code
class helloworld {
  public static void main(string[] args) {
    system.out.println("hello world!");
  }
}
</code>
<h3>Использование тега kbd</h3>
<p>Сохранить результат <kbd>Да/Нет</kbd></p>
</body>
</html>
```

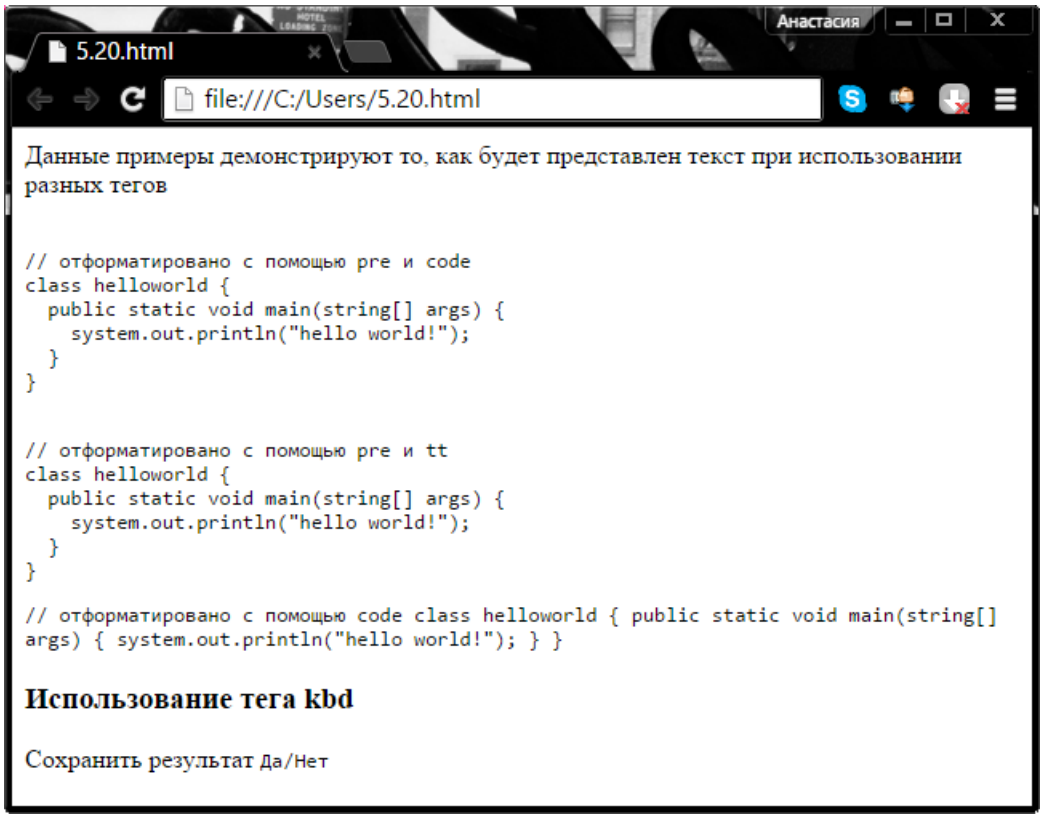


Рис. 5.20 Пример выполнения данного HTML-кода

Для маркировки примера вывода программы или скрипта используется тег <samp>.

```
<html>
```

```
<body>
<p>Если в HTML коде встретится ошибка, то будет выдано следующее:
</p><p><samp>c:\sp\bin\nsgmlsu.exe:test.html:4:7:e: element
"foobar" undefined</samp></p>
</body>
</html>
```

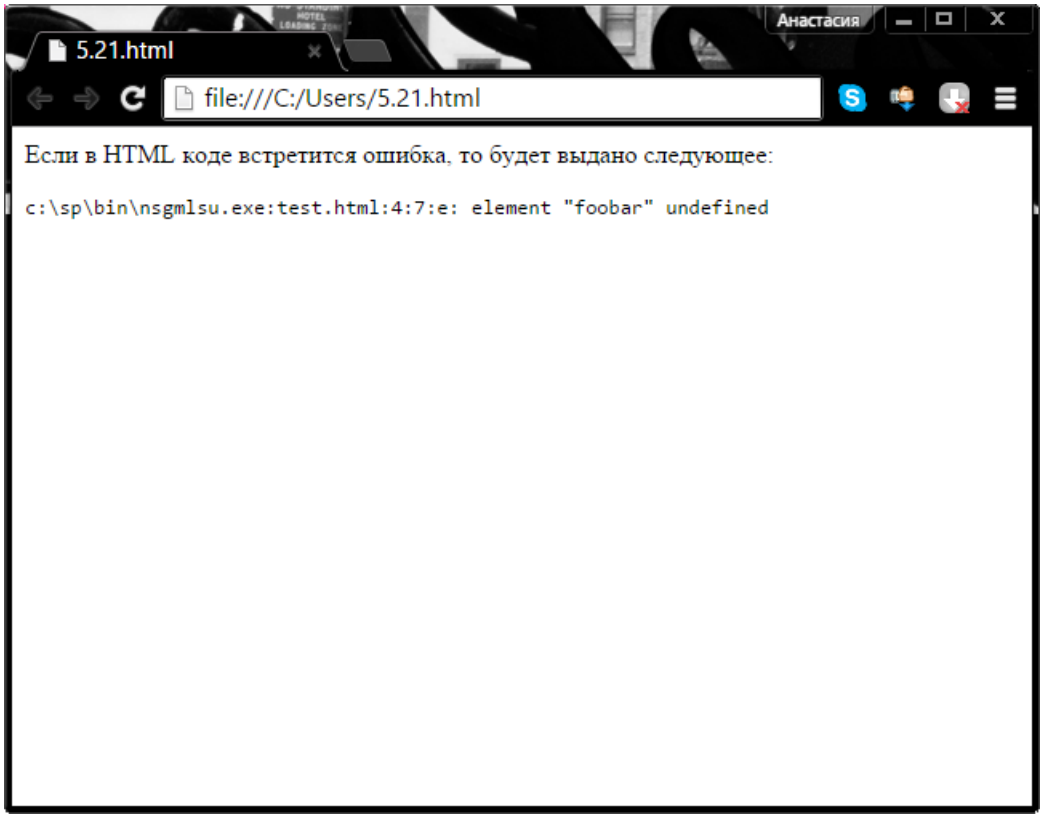


Рис. 5.21 Пример выполнения данного HTML-кода

Для маркировки переменных используется тег <var>, который обычно отображается курсивом.

```
<html>
<body>
<p>Версии стандарта HTML обычно маркируются следующим образом
<var>x</var>.<var>y</var>.</p>
</body>
</html>
```

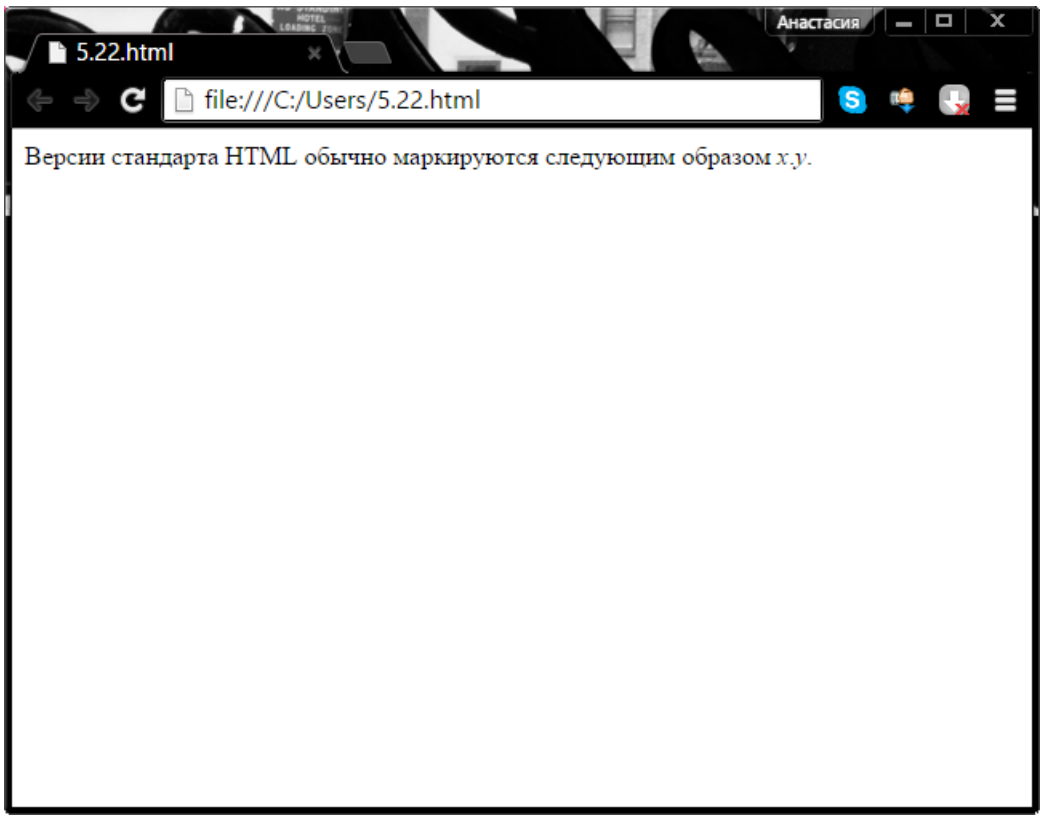


Рис. 5.22 Пример выполнения данного HTML-кода

Тег <address> маркирует контактную информацию для всего документа или его части. Он может включать имена людей осуществляющих поддержку документа, ссылки на страницы, адреса электронной почты, телефоны и прочее. Использование данного тега для маркировки почтовых адресов организаций является не совсем корректным. Его следует использовать только для пометки адресов для связи по поводу документа. Также допустимо использование тега <address> для выделения контактной информации по поводу части документа, обычно формы.

```
<html>
<body>
<form method=post action="/cgi-bin/order.cgi">
 <fieldset>
  <legend accesskey=c>Информация о кредитной карте<br></legend>
  <p>
   <label accesskey=v>
    <input type="radio" name="card" value="visa"> Visa
   </label>
   <label accesskey=m>
    <input type="radio" name="card" value="mc"> Mastercard
   </label>
   <br>
   <label accesskey=n>
    Номер: <input type="text" name="number">
   </label>
```

```
    <label accesskey=e>
      Срок действия: <input type="text" name="expiry">
    </label>
  </p>
</fieldset>

<p>
  <input type="submit" value="Отправить заказ" accesskey=s>
</p>

<address>
  Если у вас имеются вопросы по поводу заказа, свяжитесь с нами
  по адресу <a href="mailto:orders@intuit.ru">orders@intuit.ru</a>,
  Или телефону в офисе (+7 499) 253-9312.
</address>

</form>
</body>
</html>
```

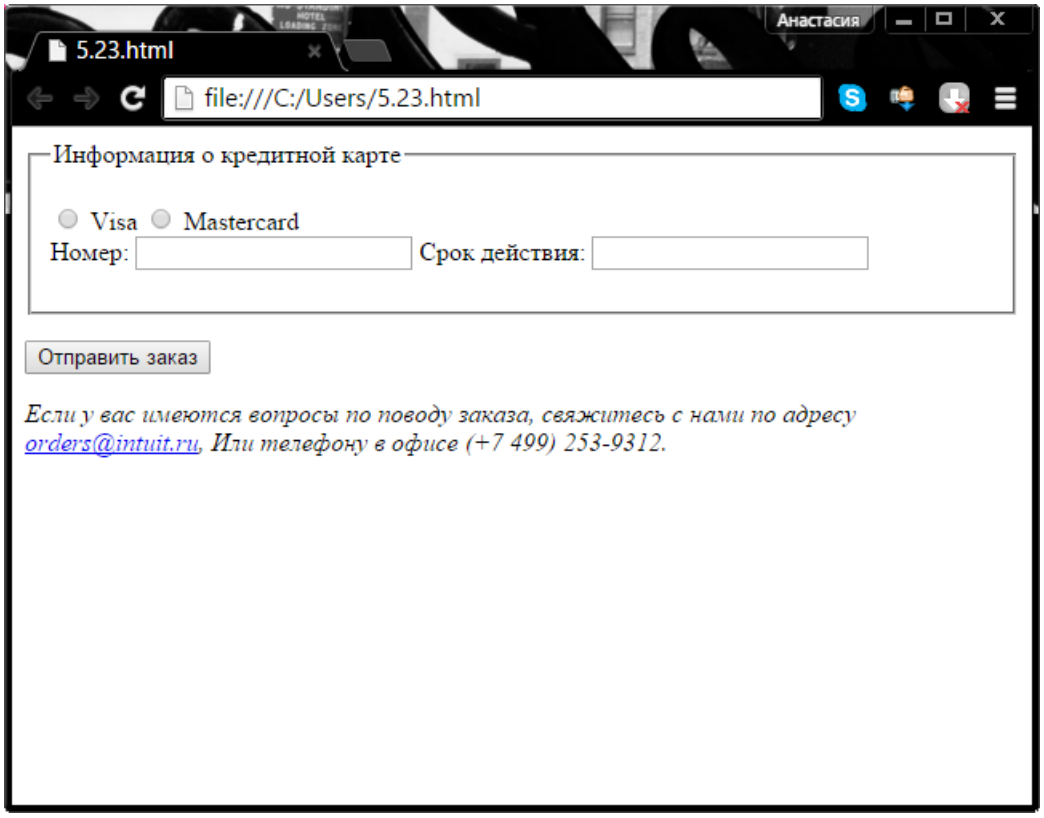


Рис. 5.23 Пример выполнения данного HTML-кода

Этот пример показывает, как работать с сокращениями или акронимами.

```
<html>
<body>
<abbr title="Содружество Независимых Государств">СНГ</abbr>
```

```
<br>
<acronym title="World Wide Web">WWW</acronym>
<p>При наведении указателя мыши на акроним или сокращение показывается атрибут title.</p>
<p>Это работает по-разному в разных браузерах.</p>
</body>
</html>
```

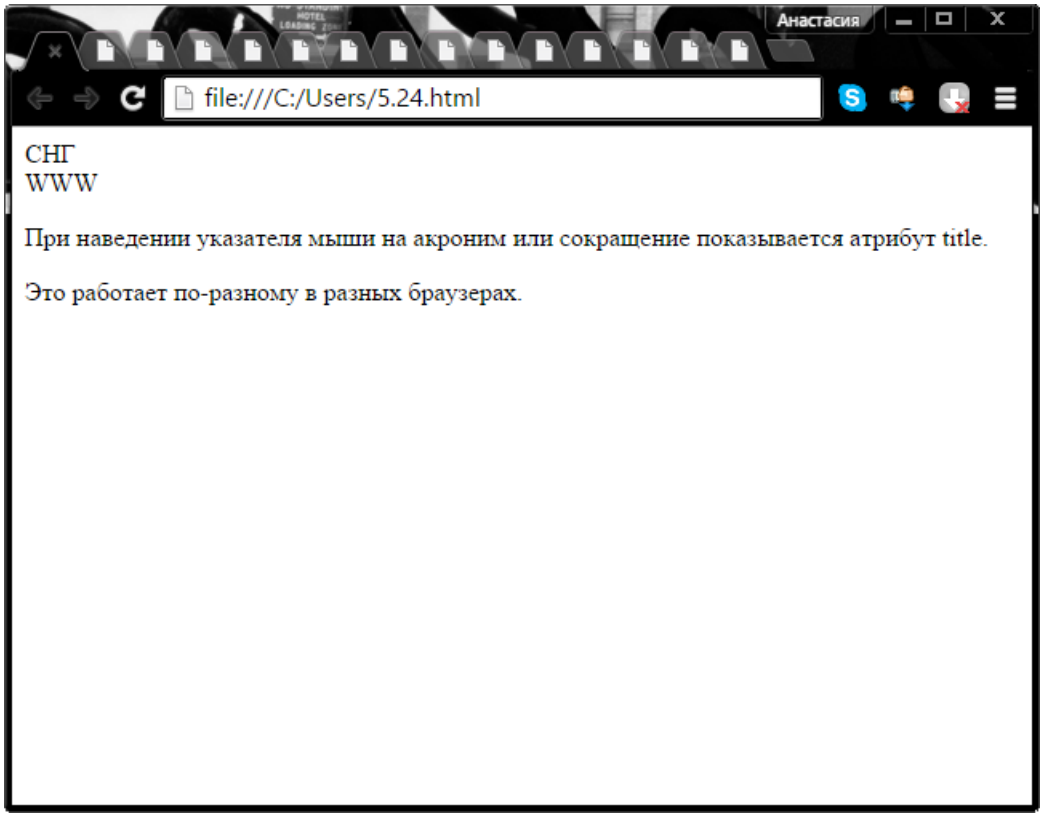


Рис. 5.24Пример выполнения данного HTML-кода

Этот пример показывает, как изменять направление вывода текста.

```
<html>
<body>
<p>
```

Если используемый браузер поддерживает двунаправленное представление (bdo), то следующая строка будет записана справа налево (rtl):

```
</p>
<bdo dir="rtl">
Здесь какой-то арабский текст
</bdo>
</body>
</html>
```

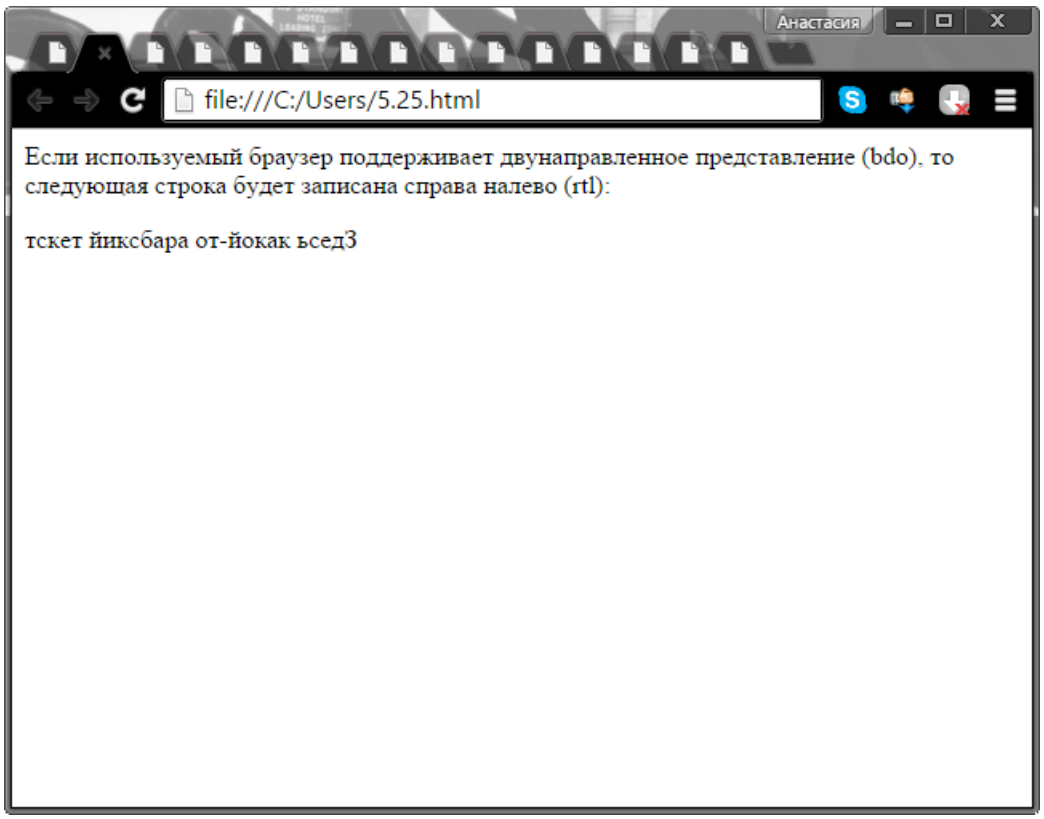


Рис. 5.25 Пример выполнения данного HTML-кода

Этот пример показывает, как использовать длинные и короткие цитаты.

```
<html>
<body>
Здесь представлена длинная цитата:
<blockquote>
Это длинная цитата. Это длинная цитата. Это длинная цитата.
Это длинная цитата. Это длинная цитата.
</blockquote>
Здесь представлена короткая цитата:
<q>
Это короткая цитата
</q>
<p>
Для элемента blockquote браузер вставляет дополнительные
переносы строки, пустые строки и поля, но элемент q
не изображается каким-то специальным образом.
</p>
</body>
</html>
```

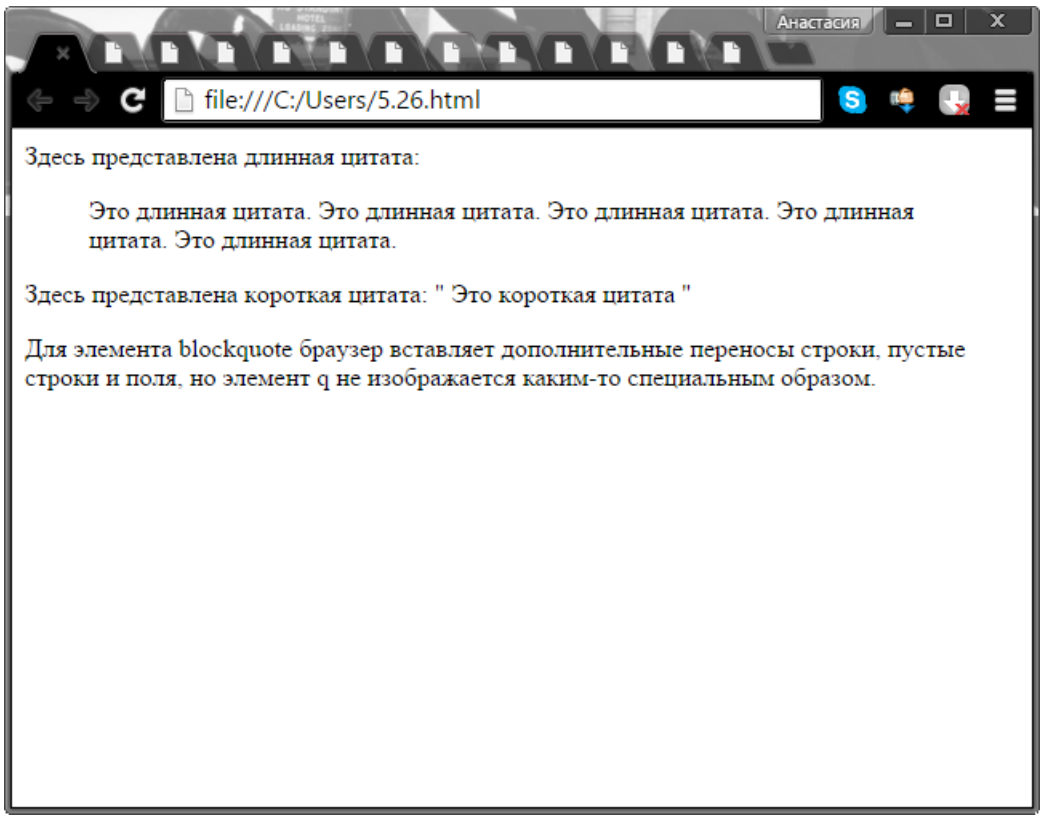


Рис. 5.26 Пример выполнения данного HTML-кода

Этот пример показывает, как пометить текст, который удален или вставлен в документ.

```
<html>
<body>
<p>
дюжина означает
<del>двадцать</del>
<ins>двенадцать</ins>
частей
</p>
<p>
```

Большинство браузеров будет зачеркивать удаленный текст и подчеркивать вставленный текст.

```
</p>
<p>
```

Некоторые старые браузеры будут выводить удаленный или вставленный текст как обычный текст.

```
</p>
</body>
</html>
```

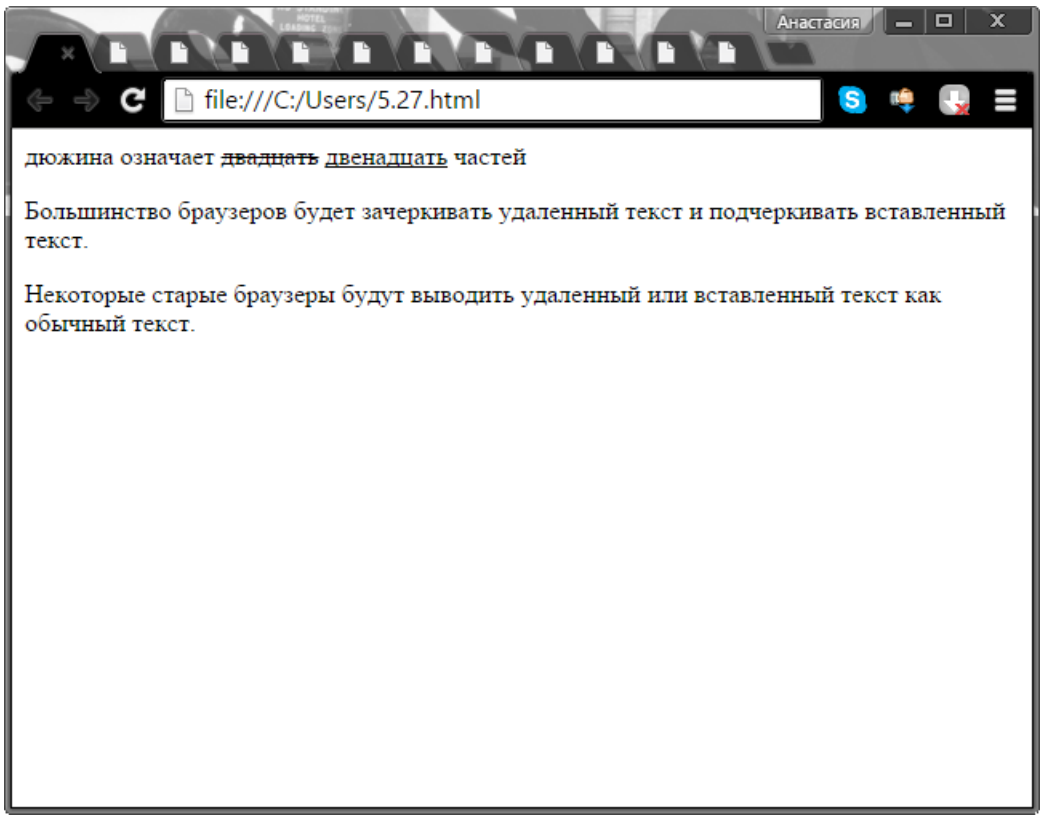


Рис. 5.27 Пример выполнения данного HTML-кода

Исходный код HTML-документа

Полезно изучать код Web-страниц, сделанных другими. Такую возможность предоставляют все популярные браузеры. Для этого в меню View ("Вид") браузера следует выбрать Source ("Исходный код") или Page Source ("Код страницы"). Откроется окно, в котором будет показан фактический код HTML страницы.

Таблица тегов управления формой отображения

Тег	Описание
<b>	Задает жирный текст
<big>	Относительное увеличение текста
	Выделяет текст (обычно курсив)
<i>	Задает курсив
<small>	Относительное уменьшение текста
	Акцентирует текст (обычно жирный)
<sub>	Определяет нижний индекс
<sup>	Определяет верхний индекс
<ins>	Вставленный текст
<del>	Удаленный текст
<s>	Не рекомендуется. Используйте вместо этого <del>
<strike>	Не рекомендуется. Используйте вместо этого <del>
<u>	Не рекомендуется. Используйте вместо этого стили (style)

Таблица тегов управления типом информации

Тег	Описание
<code>	Определяет текст кода программы
<kbd>	Определяет текст клавиатуры
<samp>	Определяет образец кода программы
<tt>	Определяет текст телетайпа
<var>	Определяет переменную
<pre>	Определяет заранее отформатированный текст
<listing>	Не рекомендуется. Используйте вместо этого <pre>
<plaintext>	Не рекомендуется. Используйте вместо этого <pre>
<xmp>	Не рекомендуется. Используйте вместо этого <pre>

Таблица тегов цитирования, сносок и определения

Тег	Описание
<abbr>	Определяет сокращение
<acronym>	Определяет акроним
<address>	Определяет элемент address (адрес)
<bdo>	Определяет направление вывода текста
<blockquote>	Определяет длинную цитату
<q>	Определяет короткую цитату
<cite>	Определяет сноску на другой материал
<dfn>	Используется для определения термина

Символьные элементы

Некоторые символы, такие как символ <, имеют в HTML специальное значение. Поэтому их нельзя использовать в тексте в явном виде. Для их отображения используются символьные элементы **CER** (**Character Entity Reference**).

Для представления символа "<" в теле документа HTML используется <, а для символа ">" используется >. Символьный элемент отображается в виде:

&имя_символа;

или

&#номер_символа;

Например, знак меньше (<) изображается в виде: < или <.

Преимущество использования имени вместо номера состоит в том, что имя легче запомнить. Недостаток состоит в том, что не все браузеры поддерживают самые новые имена объектов, в то время как поддержка номеров объектов реализована очень хорошо почти во всех браузерах. Обратите внимание, что символьные объекты зависят от регистра символов.

Следующий пример позволит поэкспериментировать с символьными объектами.

Символьные объекты. Работает только в Internet Explorer.

<html>
<body>
Это символьный объект: &#000;

Попробуйте заменить номер (000) на другой номер (например, 169), сохраните измененный текст и перезагрузите страницу в браузере, чтобы увидеть результат.

</body>
</html>

Пример выполнения данного HTML-кода

Неразрывный пробел

Наиболее часто используемым символьным объектом является неразрывный пробел –   . Обычно браузер удаляет лишние пробелы и вместо нескольких использует один, если необходимо вставить в текст пробелы, используется символьный объект . Еще одно частое применение неразрывного пробела заполнение пустых ячеек в таблице, так как большинство браузеров не отображает ячейки, в которых ничего нет.

Таблица наиболее часто используемых символьных объектов

Результат	Описание	Имя объекта	Номер объекта
	неразрывный пробел		
<	меньше	&lt;	&#60;
>	больше	&gt;	&#62;
&	амперсанд	&amp;	&#38;
"	двойная кавычка	&quot;	&#34;
'	апостроф	&apos;	&#39;
¢	цент	&cent;	&#162;
£	фунт стерлингов	&pound;	&#163;
¥	йена	&yen;	&#165;
§	параграф	&sect;	&#167;
©	авторское право	&copy;	&#169;
®	зарегистрированная торговая марка	&reg;	&#174;
x	умножение	&times;	&#215;
÷	деление	&divide;	&#247;

Полный список символьных объектов HTML представлен в Справочнике объектов HTML.

Гипертекстовые ссылки необходимы для соединения с другими документами в Web. Для их записи используется тег <a>, который называют "якорь" (anchor).

Создание гиперссылок

Этот пример показывает, как создавать ссылки в документе HTML.

```
<html>
<body>
<p>
<a href="page1.htm">
Этот текст</a> является ссылкой на страницу на этом Web-сайте.
</p>
<p>
<a href="http://www.intuit.ru/">
Этот текст </a> является ссылкой на страницу во Всемирной Паутине.
</p>
</body>
</html>
```

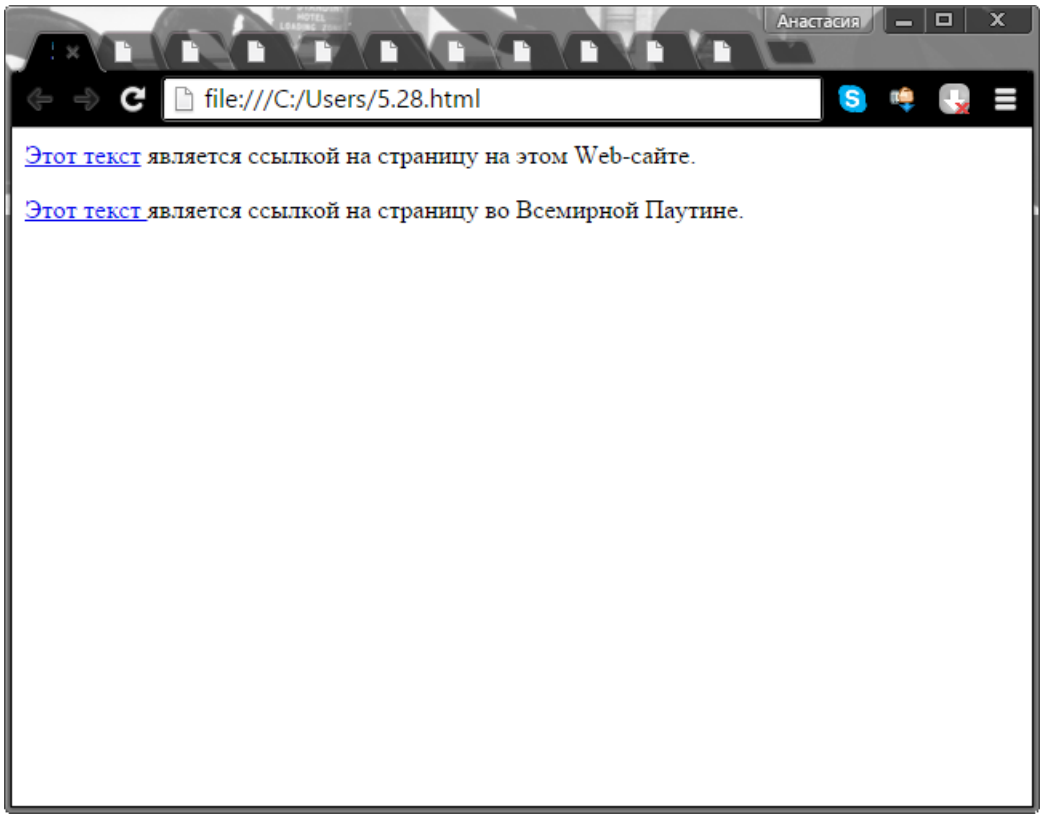


Рис. 5.28 Пример выполнения данного HTML-кода

Изображение в качестве ссылки

Этот пример показывает, как использовать в качестве ссылки изображение.

```
<html>
<body>
<p>
Можно также использовать в качестве ссылки изображение:
<a href="http://www.tuit.uz/">
<img border="0" src="logo.gif" width="285" height="102">
</a>
```

```
</p>
</body>
</html>
```

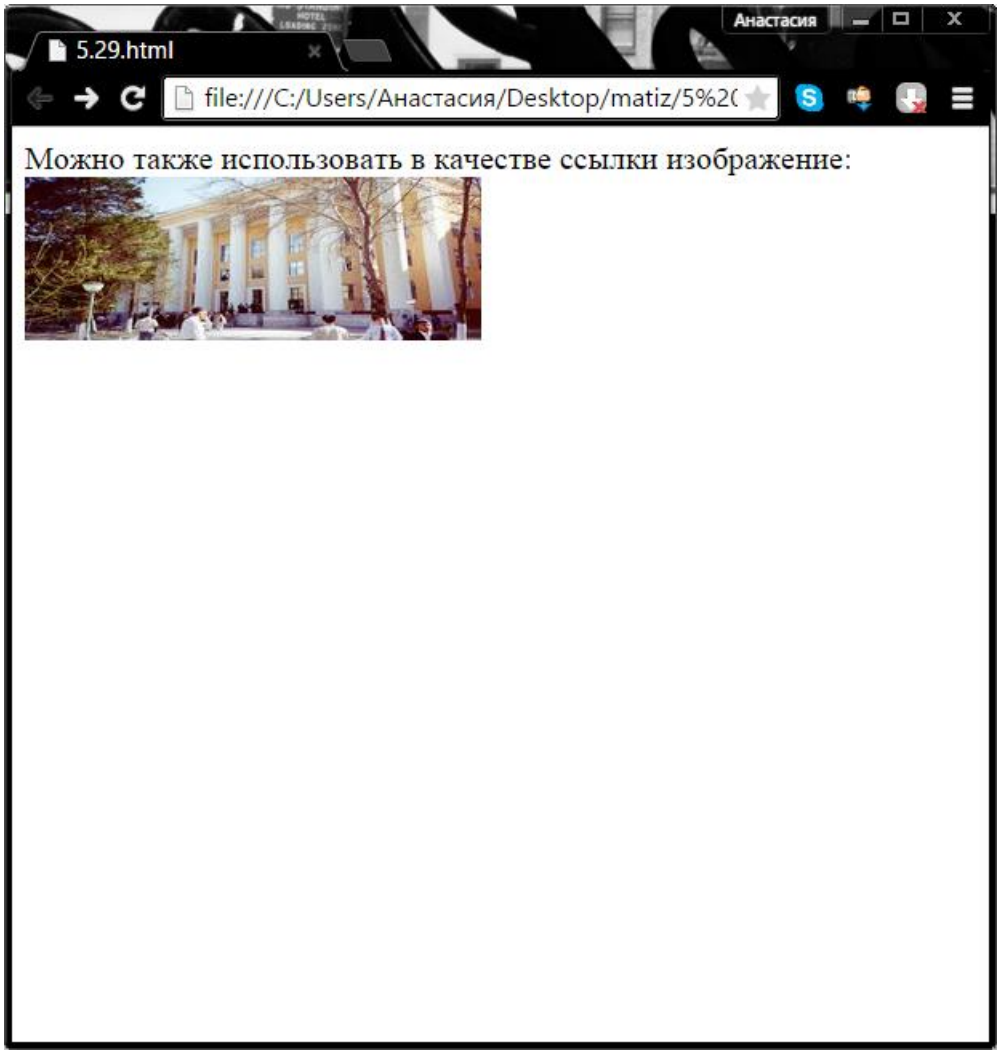


Рис. 5.29 Пример выполнения данного HTML-кода

Описание тега гипертекстовой ссылки

Якорь (anchor) позволяет устанавливать связь с любыми ресурсами во Всемирной Сети: другую страницу HTML, изображение, звуковой файл, видеоклип, и т.д.

Синтаксис тега гипертекстовой ссылки:

текст ссылки

При помощи атрибута href передается адрес документа, на который указывает ссылка.

Добро пожаловать в ТУИТ!

Такой же синтаксис используется для перехода во вложенную папку, например в папку "VOS TUIT" на сайте Ташкентского Университета Информационных Технологий используется код:

Виртуальная образовательная система ТУИТ!

Всегда добавляйте завершающую косую черту к ссылкам на вложенные папки. Если ссылка имеет следующий вид: href="http://www.etuit.uz/dl", то будет создано два запроса HTTP к серверу, так как сервер добавит косую черту к адресу и создаст новый запрос следующего вида: href="http://www.etuit.uz/dl/". При этом вызове обращение обычно происходит к странице index.htm внутри папки dl или к другому файлу – это зависит от настроек web-сервера.

С помощью данного атрибута можно задавать ссылки не только по протоколу HTTP, но и по другим:

- http://... - создает ссылку на www-документ;
- ftp://... - создает ссылку на ftp-сайт или расположенный на нем файл;
- mailto:... - запускает почтовую программу-клиент с заполненным полем имени получателя. Если после адреса поставить знак вопроса, то можно указать дополнительные атрибуты, разделенные знаком "&";
- news:.. - создает ссылку на конференцию сервера новостей;
- telnet://... - создает ссылку на telnet-сессию с удаленной машиной;
- wais://... - создает ссылку на WAIS - сервер;
- gopher://... - создает ссылку на Gopher - сервер;

С помощью атрибута target можно определить, где будет открыт документ, на который указывает ссылка. В качестве значения необходимо задать либо имя одного из существующих фреймов, либо одно из следующих зарезервированных имен:

- _self - указывает, что определенный в параметре href документ должен отображаться в текущем фрейме;
- _parent - указывает, что документ должен отображаться во фрейме-родителе текущего фрейма. Иначе говоря, _parent ссылается на окно, содержащее frameset, включающий текущий фрейм;
- _top - указывает, что документ должен отображаться в окне-родителе всей текущей фрэймовой структуры;
- _blank - указывает, что документ должен отображаться в новом окне.

```
<html>
<body>
<a href="http://www.tuit.uz/" target="_blank">Добро пожаловать в ТУИТ!</a>
<a href="http://www.tuit.uz/" target="_parent">Добро пожаловать в ТУИТ!</a>
<a href="http://www.tuit.uz/" target="_top">Добро пожаловать в ТУИТ!</a>
<a href="http://www.tuit.uz/" target="_self">Добро пожаловать в ТУИТ!</a>
</body>
</html>
```

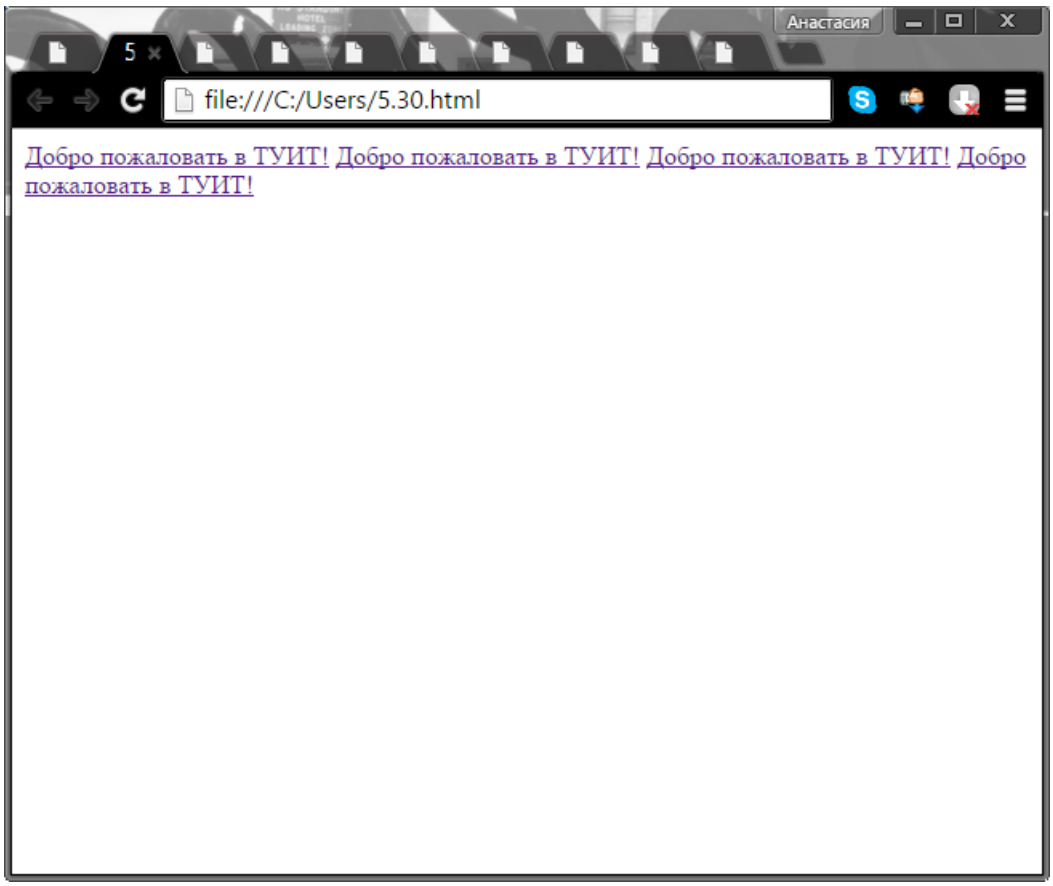


Рис. 5.30 Пример выполнения данного HTML-кода

Для перехода внутри страницы к конкретному разделу используют атрибут name тега гипертекстовой ссылки a, при помощи которого создается именованный якорь.

ссылка на именованный якорь

В данном примере внутри документа создается своеобразная метка, при этом именованный якорь никак не отображается в окне браузера.

Для перехода на место, которое отмечено именованным якорем используется знак # и имя якоря в конце URL следующим образом:

переход на именованный якорь

Переход внутри файла "page1.html" на именованный якорь выглядит следующим образом:

переход на именованный якорь

Если браузер не сможет найти указанный именованный якорь на вызываемой странице, то переход произойдет на начало документа, при этом сообщения об ошибке доступа не возникает.

Дополнительные примеры

Открытие ссылки в новом окне браузера

Этот пример показывает, как открыть ссылку на другую страницу в новом окне, чтобы посетителю не нужно было покидать ваш Web-сайт.

<html>
<body>
Последняя страница
<p>

Если задать атрибут target ссылки как "_blank", то ссылка будет открыта в новом окне.

</p>
</body>
</html>

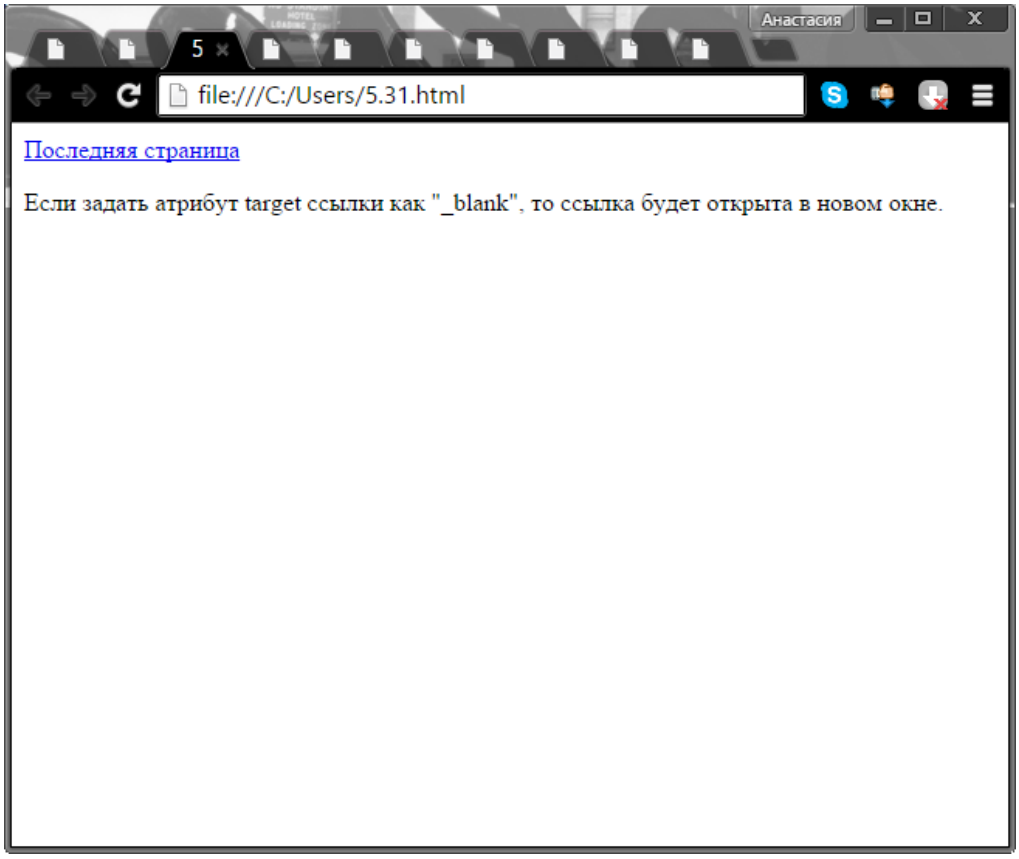


Рис. 5.31 Пример выполнения данного HTML-кода

Ссылка в определенное место на той же странице

Этот пример показывает, как использовать ссылку на другую часть документа.

```
<html>
<body>

<p>
<a href="#part5">Переход на часть 5.</a>
</p>

<h1>Лекция 1</h1>

<h2>Часть 1</h2>
<p>Это первая часть ... </p>

<h2> Часть 2</h2>
<p>Это вторая часть ...</p>
```

```
<h2> Часть 3</h2>
<p>Это третья часть ...</p>

<h2> Часть 4</h2>
<p>Это четвертая часть ...</p>

<h2><a name="part5"> Часть 5</a></h2>
<p>Это пятая часть ...</p>

<h2> Часть 6</h2>
<p>Это шестая часть ...</p>

<h2> Часть 7</h2>
<p>Это седьмая часть ...</p>

<h2> Часть 8</h2>
<p>Это восьмая часть ...</p>

<h2> Часть 9</h2>
<p>Это девятая часть ...</p>

<h2> Часть 10</h2>
<p>Это десятая часть ...</p>

</body>
</html>
```

Рис.5. 32 Пример выполнения данного HTML-кода

Создание ссылки mailto

Этот пример показывает, как соединиться с сообщением mail (будет работать только, если установлена служба mail).

```
<html>
<body>
<p>
Это ссылка на почтовые адреса:
<a
href="mailto:help@intuit.ru?cc=orders@intuit.ru&bcc=admin@intuit.ru?subject=
Тестовый%20запрос!">
отправить запрос
</a>
</p>
<p>
```

При помощи этой ссылки отправляется письмо по адресу электронной почты help@intuit.ru, его копия на адрес orders@intuit.ru и скрытая копия на адрес admin@intuit.ru. Для корректной обработки заголовка письма необходимо пробелы заменять на %20.

```
</p>
</body>
</html>
```

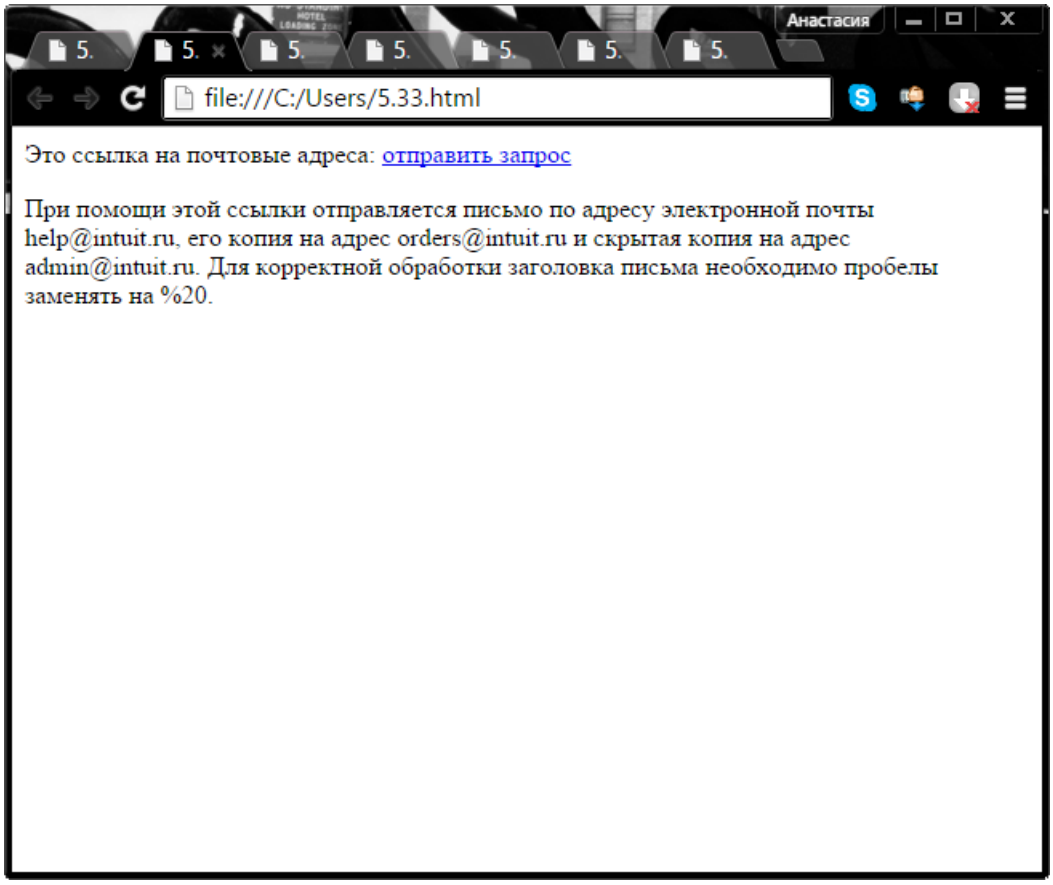


Рис. 5.33 Пример выполнения данного HTML-кода

Фреймы HTML

Фреймы используются для разбивки окна браузера на несколько независимых частей, каждая из которых представляет собой отдельный HTML-документ. Наиболее частое применение фреймов – это отделение меню от основного информационного наполнения. Хотя фреймы достаточно удобная вещь, использование их **не рекомендуется**. Следует внимательно следить за тем, чтобы теги, формирующие фреймовую структуру, располагались вне тегов <body>, так как они не относятся к телу документа.

Набор вертикальных фреймов

Этот пример показывает, как создать набор вертикальных фреймов с тремя различными документами.

```
<html>
<frameset cols="50%,25%,25%">
<frame src="frame_1.htm">
<frame src="frame_2.htm">
<frame src="frame_3.htm">
</frameset>
</html>
```

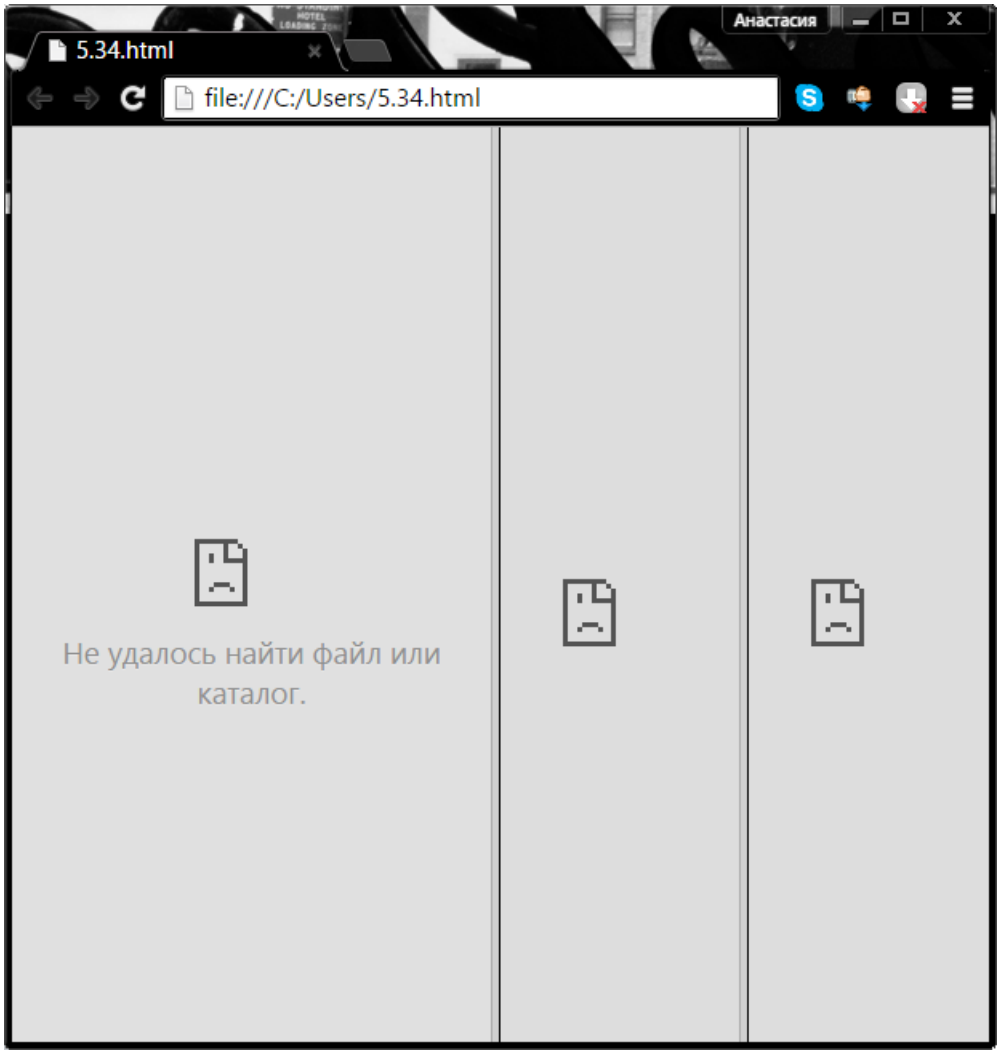


Рис. 5.34 Пример выполнения данного HTML-кода

Набор горизонтальных фреймов

Этот пример показывает, как создать набор горизонтальных фреймов с тремя различными документами.

```
<html>
<frameset rows="25%,50%,25%">
<frame src="frame_1.htm">
<frame src="frame_2.htm">
<frame src="frame_3.htm">
</frameset>
</html>
```

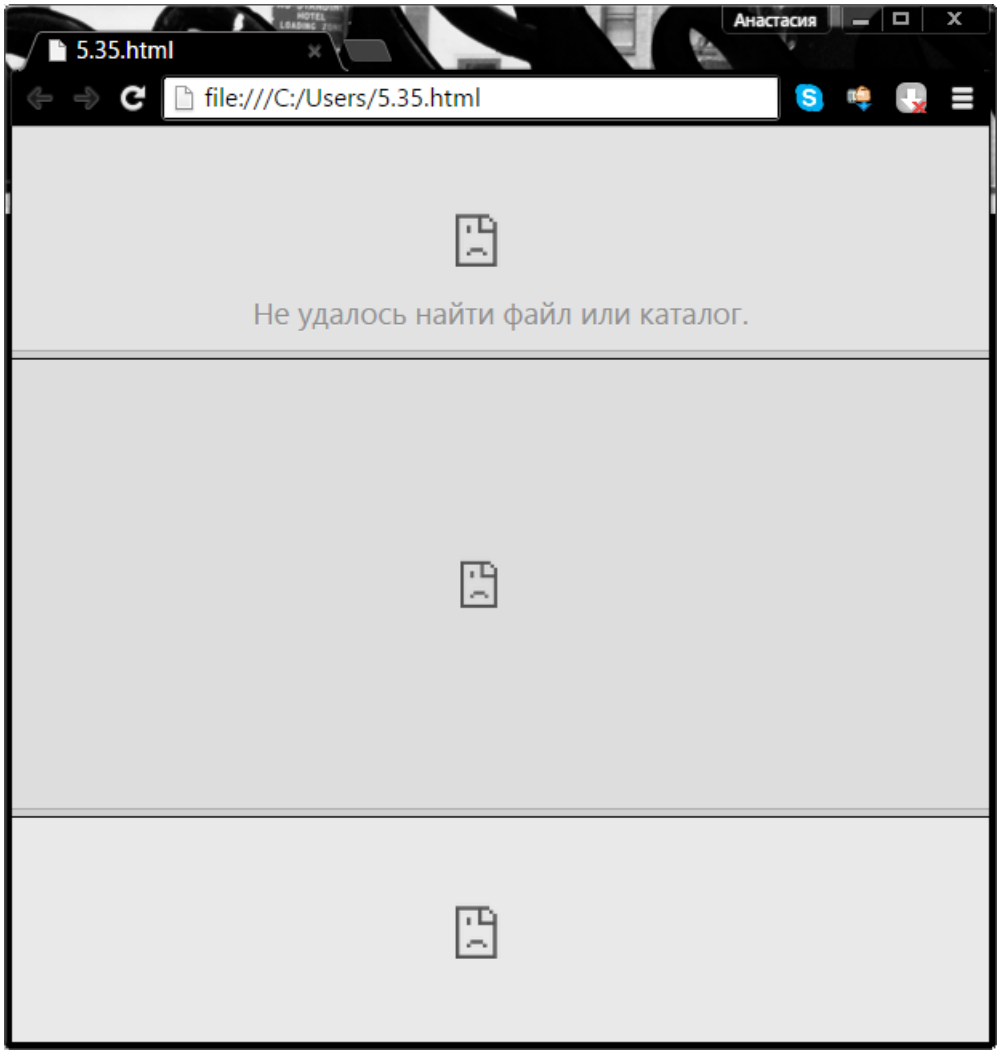


Рис. 5.35 Пример выполнения данного HTML-кода

Теги для работы с фреймами

Тег <frameset> определяет, как разделить окно на фреймы. Каждый frameset определяет набор строк или столбцов. Значения rows/cols указывают величину области экрана, которую будут занимать каждая строка/столбец. Атрибут framespacing – определяет расстояние между фреймами в пикселах.

Тег <frameset> является контейнером для всех остальных тегов фреймов. Между начальным и конечным тэгами кроме элементов <frame> и <noframes> могут находиться другие элементы <frameset>. То есть элемент <frameset> поддерживает вложенные конструкции фреймов.

Тег <frame> определяет, какой документ HTML поместить в каждый фрейм. Тег <frame> имеет атрибут noresize="noresize", который запрещает изменять размер фрейма пользователю.

В примере ниже задана фреймовая структура с двумя столбцами. Для первого столбца задано 25% ширины окна браузера. Для второго фрейма задано 75% ширины окна браузера. В первый столбец помещается документ HTML "frame_1.htm", а во второй столбец помещается документ HTML "frame_2.htm":

```
<frameset cols="25%,75%">
  <frame src="frame_1.htm">
  <frame src="frame_2.htm">
</frameset>
```

Необходимо учитывать, что браузер может не поддерживать фреймы, в этом случае следует использовать тег <noframes>.

При использовании фреймов теги <body> </body> не используются. Однако, если добавить тег <noframes>, содержащий некоторый текст для браузеров, которые не поддерживают фреймы, необходимо будет поместить этот текст между тегами <body> </body>. Например,

```
<html>
<frameset cols="25%,50%,25%">
 <frame src="frame_1.htm">
 <frame src="frame_2.htm">
 <frame src="frame_3.htm">
<noframes>
<body>Ваш браузер не поддерживает фреймы!</body>
</noframes>
</frameset>
</html>
```

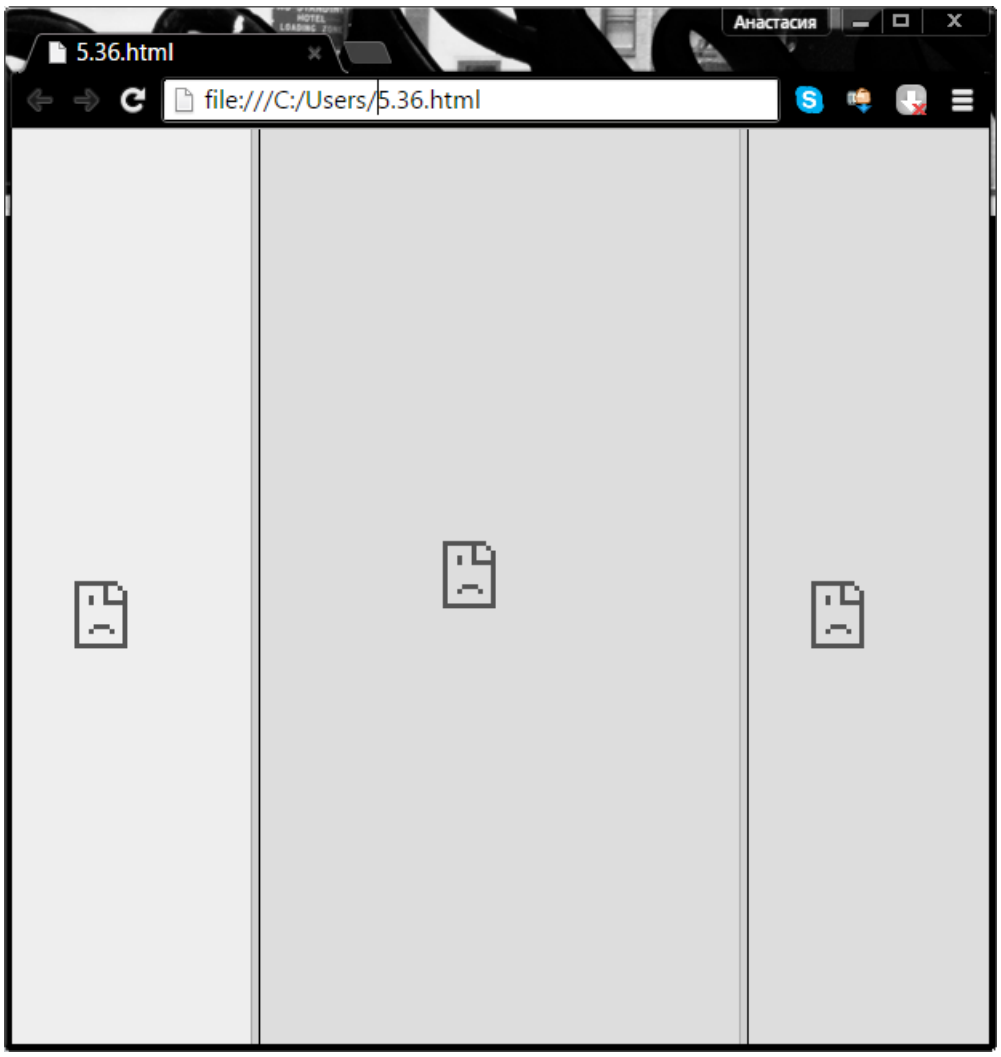


Рис. 5.36 Пример выполнения данного HTML-кода

Горизонтальные и вертикальные фреймы

Этот пример показывает, как сделать набор фреймов с тремя документами, и как разделить их на строки и столбцы. При этом не все границы фреймов можно передвинуть.

```
<html>
<frameset rows="50%,50%">
<frame src="frame_1.htm">
<frameset cols="25%,75%">
<frame noresize="noresize" src="frame_2.htm">
<frame src="frame_3.htm">
</frameset>
</frameset>
</html>
```

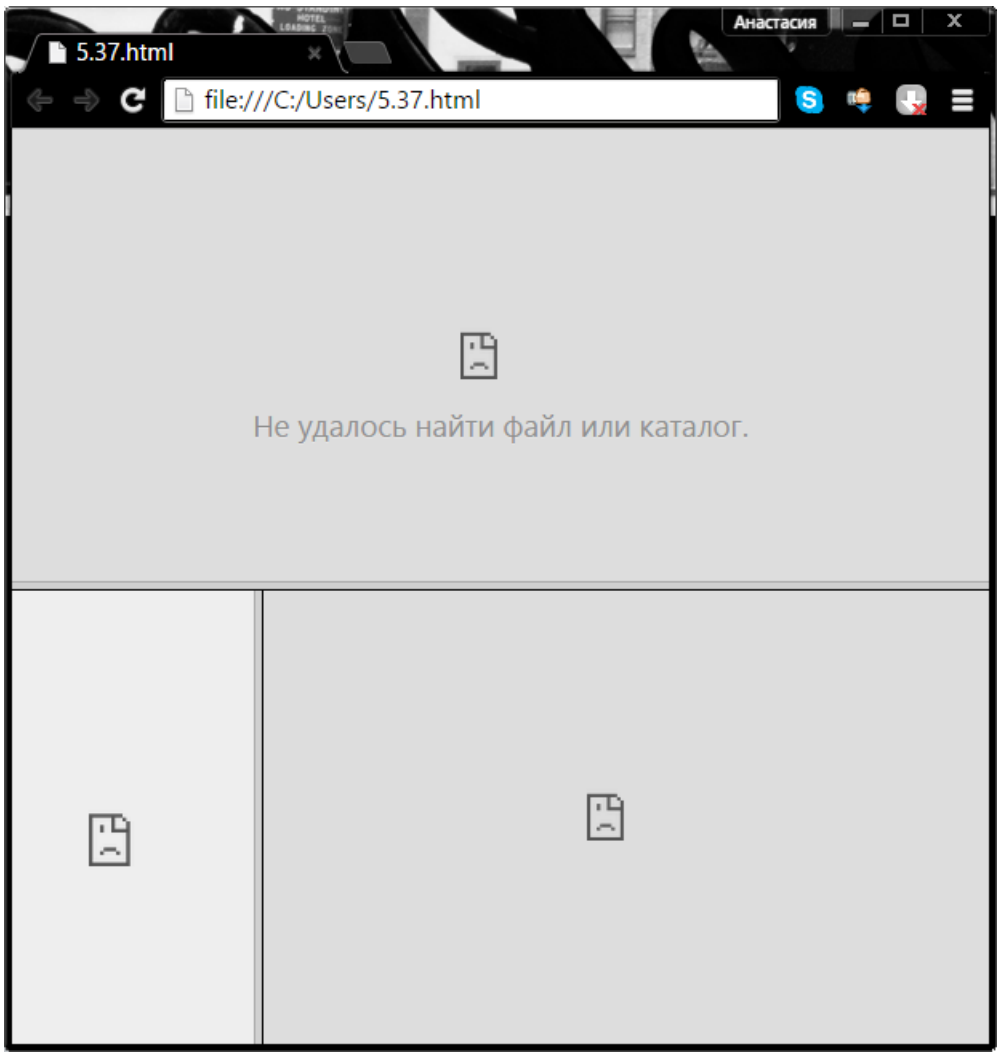


Рис. 5.37 Пример выполнения данного HTML-кода

Фрейм навигации

Этот пример показывает, как сделать фрейм навигации. Фрейм навигации содержит список ссылок, указывающих на второй фрейм.

```
<html>
<frameset cols="120,*">
<frame src="menu.htm">
<frame src="frame_1.htm" name="frame_1">
</frameset>
</html>
```

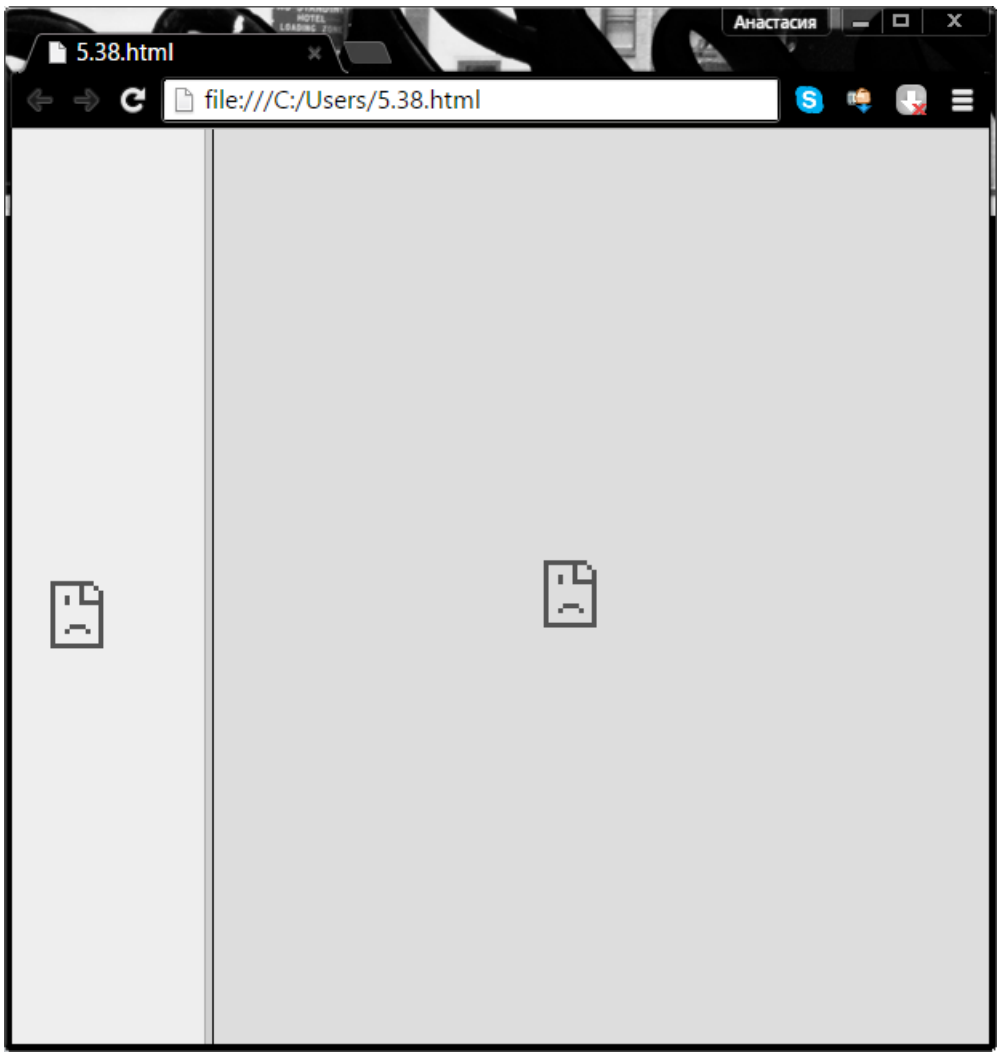


Рис. 5.38 Пример выполнения данного HTML-кода

Файл с именем "menu.htm" содержит ссылки на другие документы HTML, которые будут загружаться в frame_1. Исходный код для ссылок:

```
<a href ="file_1.htm" target ="frame_1">Файл 1</a><br>
<a href ="file_2.htm" target ="frame_1">Файл 2</a><br>
<a href ="file_3.htm" target ="frame_1">Файл 3</a>
```

Встроенный фрейм

Этот пример показывает, как создать встроенный фрейм (фрейм внутри страницы HTML).

```
<html>
<body>
<iframe src="menu.html"></iframe>

<p>Некоторые старые браузеры не поддерживают iframes.</p>
<p>В этом случае встроенный фрейм (iframe) не будет виден.</p>
</body>
</html>
```

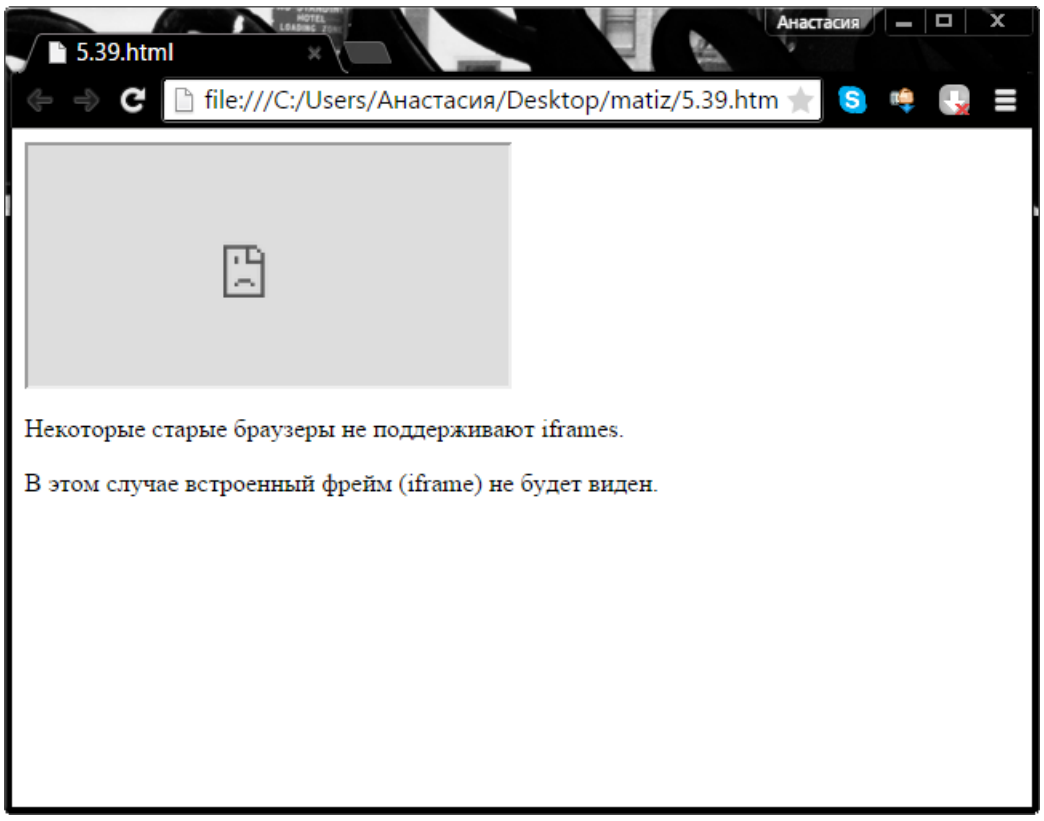


Рис. 5.39 Пример выполнения данного HTML-кода

Новые элементы (теги) в HTML5

Свойства HTML5

Язык HTML5 содержит много новых свойств, что делает HTML значительно более мощным и удобным для создания приложений Web. В приведенном ниже списке суммируются основные свойства, о которых действительно необходимо знать.

Некоторые из перечисленных ниже свойств не являются на самом деле частью самой спецификации HTML5, но определены в тесно связанных спецификациях, поэтому они все еще являются допустимыми частями нового движения в направлении современных Web-приложений, и о них полезно знать.

Новые семантические элементы: Как вы возможно уже знаете, семантика является очень важной в HTML — мы всегда должны использовать для работы подходящий элемент. В HTML 4.01 мы имеем проблему — да, существует много элементов для определения специальных средств, таких как таблицы, списки, заголовки, и т.д., но существует также много общих свойств web-страницы, которые не имеют элемента для их определения. Представьте верхние и нижние колонтитулы сайта, навигационные меню, и т.д. — до сих пор мы определяли их с помощью , которые мы можем понять, но машины не могут, кроме того, различные разработчики web будут использовать различные ID и

классы. К счастью, HTML5 содержит новые семантические элементы, такие как <nav>, <header>, <footer> и <article>.

Новые свойства форм: HTML 4.01 уже позволяет создавать удобные, доступные web-формы, но некоторые общие свойства форм являются не слишком удобными и требуют специальных усилий для реализации. HTML5 предоставляет стандартизованный, простой способ реализации таких свойств, как выбор даты, ползунки и клиентская проверка. Формы HTML5 подробно рассматриваются в Главе 7 «Программирование форм и элементов форм».

Собственная поддержка видео и аудио: В течение многих лет видео и аудио в Web делалось, вообще говоря, с помощью Flash. Фактическая причина, почему технология Flash стала так популярна на заре 21 века, состояла в том, что открытые стандарты не смогли предоставить для различных браузеров совместимый механизм реализации таких вещей, поэтому различные браузеры реализовали различные конкурирующие способы выполнения одних и тех же вещей (например, <object> и <embed>), делая тем самым весь процесс действительно сложным. Flash предоставлял высококачественный, легкий способ реализации работы видео в различных браузерах.

HTML5 содержит элементы <video> и <audio> для простой реализации собственных видео и аудио плееров с помощью только открытых стандартов, и также содержит API, позволяющий легко реализовать индивидуальные элементы управления плеером.

API рисования на холсте: Элемент <canvas> и соответствующий API позволяют определить на странице область для рисования, и использовать команды JavaScript для рисования линий, фигур и текста, импорта и манипуляций с изображениями и видео, экспорта в различные форматы изображений, и многих других вещей.

Сокеты Web: Этот API (определенный в спецификации Сокеты Web - http://www.w3.org/TR/websockets/, отдельно от спецификации HTML5) позволяет открывать постоянное соединение между сервером и клиентом на определенном порте, и посылать данные в обоих направлениях, пока порт не будет закрыт. Это существенно улучшает эффективность приложений web, так как данные могут непрерывно и аккуратно передаваться между клиентом и сервером без постоянной перезагрузки страницы, и без постоянного опроса сервера, чтобы проверить, нет ли доступных обновлений.

Автономные приложения web: HTML5 предоставляет ряд свойств, позволяющих приложениям web выполняться в автономном режиме. Кэши приложений (http://dev.opera.com/articles/view/offline-applications-html5-appcache/) позволяют сохранить копию всех ресурсов и других файлов, необходимых для локального выполнения приложения web, и базы данных Web SQL позволяют сохранить локальную копию данных приложения web. Совместно они позволяют продолжать использовать приложение, когда отсутствует соединение с сетью, и затем синхронизируют изменения с основной версией на сервере, когда сеть снова становится доступной.

Хранилище Web: Cookies предоставляют в какой-то степени локальное хранилище данных, но их использование довольно ограничено. Web хранилище HTML5 позволяет хранить значительно больше данных, и делать с ними значительно больше.

Web workers: Общая проблема, встающая перед приложениями web, состоит в том, что их производительность страдает, когда требуется обработать много данных, в связи с тем, что все происходит в одной нити/процессе (только одна последовательность обработки может выполняться в текущий момент). Web Workers смягчают эту проблему, позволяя создавать фоновые процессы для выполнения значительного объема вычислений, позволяя основному процессу продолжить выполнение других задач.

Геолокация: Спецификация геолокации (http://dev.w3.org/geo/api/spec-source.html) (также не являющаяся частью спецификации HTML5) определяет API, который позволяет приложению web легко получить доступ к данным в любом местоположении, которое стало доступным, например, с помощью средств GPS устройства. Это позволяет добавлять в приложения различные полезные свойства, связанные с местоположением, например, выделить контент, который больше подходит для местоположения. Чтобы получить общее представление о возможностях геолокации, прочтите статью "Как использовать W3C Geolocation API" (http://dev.opera.com/articles/view/how-to-use-the-w3c-geolocation-api/).

Примечание: CSS3 СОВЕРШЕННО ОПРЕДЕЛЕННО НЕ является HTML5, и никогда не будет. Не позволяйте отделу маркетинга говорить вам обратное.

Краткий обзор новинок HTML5

HTML5 содержит две новые вещи: новые API, которые добавляют существенно важные новые свойства к открытым стандартам модели разработки web, и новые структурные элементы, которые определяют специальные свойства страницы Web со значительно более точной семантикой, чем это было доступно в HTML 4. Статьи, посвященные многим новым API, можно найти на Dev.Opera (http://dev.opera.com/articles/tags/html5/) с меткой HTML5.

Данный раздел рассматривает другую составляющую – мы кратко опишем, как были выбраны новые семантические элементы, какие новые свойства имеются, и как они используются, как в HTML5 работают заголовки, и поддержку в браузерах новых элементов, включая их поддержку в старых браузерах.

Введение в структурные элементы HTML5

HTML4 имеет множество семантических элементов, позволяющих четко определять различные свойства страницы Web, такие как формы, списки, параграфы, таблицы и т.д. Однако он имеет и свои недостатки. Существенно используются элементы и с различными

атрибутами id и class для определения различных других свойств, таких как навигационные меню, верхние и нижние колонтитулы, основной контент, окна предупреждения, боковые панели, и т.д. Что-нибудь типа будет понятно разработчикам и дизайнерам, знающим, для чего это предназначено, и умеющих использовать CSS и JavaScript для применения собственных стилей и поведения, чтобы сделать это понятным для конечных пользователей.

Все могло бы быть значительно лучше, но с такими настройками по-прежнему существуют проблемы:

- Люди могут понять разницу между различным контентом, но машины не могут — браузер не видит различные div как верхние и нижние колонтитулы и т.д. Он видит их как различные div. Не будет ли полезнее, если браузеры и считыватели экрана смогут явно идентифицировать, скажем, навигационное меню, чтобы пользователям с недостатками зрения было легче его найти, или различные элементы новостей в пакете блогов, чтобы их было легче объединить в ленту RSS без дополнительного программирования?
- Даже если для решения некоторых из этих проблем используется дополнительный код, вы можете сделать это надежно только для собственных web-сайтов, так как другие разработчики web будут использовать другие имена классов и ID, особенно, если рассмотреть международную аудиторию — различные разработчики web в разных странах будут использовать различные языки для записи имен своих классов и id.

Поэтому имеет смысл определить согласованное множество элементов для общих структурных блоков, которые часто появляются на большинстве Web-сайтов. В данной статье будут рассмотрены следующие новые элементы HTML5:

- <header>: Используется для верхнего колонтитула сайта.
- <footer>: Используется для нижнего колонтитула сайта.
- <nav>: Содержит навигационные функции страницы.
- <article>: Содержит автономный фрагмент контента, который будет иметь смысл, если используется как позиция RSS, например, новостное сообщение.
- <section>: Используется либо для объединения в группу различных статей с различной целью или по различным темам, или для определения различных разделов одной статьи.
- <time>: Используется для разметки времени и даты.
- <aside>: Определяет блок контента, который связан с основным контентом, но не входит в его основной поток.
- <hgroup>: Используется в качестве оболочки скрытия более одного заголовка, если требуется, чтобы учитывался только один заголовок в структуре заголовков страницы.

- <figure> и <figcaption>: Используется для инкапсуляции рисунка как единого элемента, и содержит, соответственно, подпись для рисунка.

Как были выбраны названия элементов?

Во время создания HTML5 редактор Ян Хиксон использовал инструменты Google для извлечения данных более чем миллиарда web-страниц, определяя, какие имена ID и class использовались наиболее часто в реальной сети web. Можно посмотреть один из подготовленных обзоров, опубликованный на сайте Google Code (http://code.google.com/webstats/2005-12/classes.html). Выражаясь короче, имена элементов, указанные в этой статье, были взяты из 20 наиболее популярных имен ID и class, представленных в обзорах Хиксона.

Примечание: Компания Opera выполнила аналогичное исследование 3.5 миллионов URL, назвав его MAMA. MAMA использует меньшее множество URL, но рассматривает более широкий набор статистики Web-страниц. Дополнительную информацию можно найти на домашней странице проекта MAMA (http://dev.opera.com/articles/view/mama/).

Почему нет элемента <content>?

Хотя это может казаться кричащим упущением, на самом деле это не так. Основной контент будет блоком верхнего уровня контента, который не является <header>, <nav> или <footer>, и в зависимости от конкретных обстоятельств, может иметь больше смысла пометить контент с помощью <article>, <section>, или даже . Брюс Лоусон называет это "алгоритмом Scooby Doo", но чтобы понять почему, вы должны спросить его самого в Twitter или на конференции.

Представляем пример страницы HTML5

Рассмотрев в некоторой степени предпосылки и увидев, какие новые элементы предлагаются, давайте теперь перейдем к примеру и посмотрим детально, как использовать их в контексте реальной страницы.

Посмотрите на мою страницу A history of Pop Will Eat Itself (http://dev.opera.com/articles/view/new-structural-elements-in-html5/pwei_sample_html5.html) – историю и дискографию одной из моих любимых музыкальных групп 80/90-х (если вам нравится альтернативная музыка, пожалуйста, подтвердите это). Я использовал исходную разметку страницы Wikipedia Pop will Eat Itself (http://en.wikipedia.org/wiki/Pop_Will_Eat_Itself), почистил ее, и превратил в HTML5. Давайте внимательно посмотрим на то, что было сделано.

Держите пример страницы открытым в отдельной вкладке во время чтения статьи – вы можете захотеть вернуться к ней.

Пример использует традиционную проверенную оболочку для размещения контента по центру, но Крок Камен опубликовал интересную статью о создании центрированного дизайна без оболочки (http://camendesign.com/code/developpeurs_sans_frontieres), поэтому я подумал, что покажу ее здесь также. Также полезно порекомендовать не

использовать элементы HTML5 <section> в качестве оболочки на страницах HTML5 – это просто совершенно неверно!

Некоторые мета-различия

Первое, что вы заметите, состоит в том, что doctype значительно проще, чем в более старых версиях HTML:

<!DOCTYPE html>

Создатели HTML5 выбрали самую короткую возможную строку doctype для этой цели — в конце концов, почему разработчик должен помнить длинную строку, содержащую множество URL, когда в действительности doctype присутствует здесь только для того, чтобы задать для браузера стандартный режим (в противоположность необычному режиму)?

Затем, я хотел бы привлечь ваше внимание к кажущимся "слабым синтаксическим требованиям" HTML5. Я добавил кавычки вокруг всех значений атрибутов, и написал все элементы строчными буквами, но это, на самом деле, связано с привычкой писать по правилам XHTML. Но для кого-то может показаться удивительным открытие, что в HTML5 можно при желании игнорировать эти правила. Фактически не нужно даже беспокоиться о том, чтобы использовать элементы <head>, <body>, или <html>, все будет по-прежнему допустимо!

Примечание: Это не так, если вы переключаетесь на использование XHTML (HTML работающий с doctype XHTML— application/xhtml+xml)

Дело в том, что такие элементы подразумеваются браузером в любом случае. Если создать пример страницы HTML5 без этих элементов, загрузить ее в браузер и посмотреть исходный код загруженной страницы, то можно заметить, что они будут автоматически вставлены браузером. Иначе можно воспользоваться утилитой-просмотрщиком Яна Хиксона Live DOM (http://software.hixie.ch/utilities/js/live-dom-viewer/), чтобы увидеть состояние DOM.

Примечание: На самом деле можно также отправить на проверку документ HTML4, не содержащий <head>, <body>, или <html>, но это, тем не менее, стоит здесь отметить.

Необходимо также упомянуть, что спецификация HTML5 строго определяет, как обрабатывать плохо сформированную разметку (например, неправильно вложенные элементы, или незакрытые элементы), впервые определяя алгоритм синтаксического разбора. Это означает, что даже если разметка выполнена неправильно, DOM будет согласован с поддерживающими HTML5 браузерами.

Не означает ли это, что нам не нужно больше беспокоиться о проверке правильности и эффективных методах работы? ВОВСЕ НЕТ! Проверка правильности по-прежнему является очень полезным инструментом для создания страниц насколько возможно хорошими. Даже если DOM

совместим с браузерами, он по-прежнему может вести себя, прежде всего не так, как требуется, создавая проблемы в CSS и JavaScript! И как вы увидите при дальнейшем изучении HTML5, существуют очень серьезные причины для проверки, что такие свойства документа как <html> объявляются явно. Например, вы можете захотеть объявить язык документа в элементе <html> для интернационализации и улучшения доступности, и это требуют также некоторые связанные технологии. Хорошим примером является AppCache (http://dev.opera.com/articles/view/offline-applications-html5-appcache/).

Для проверки документов HTML5 можно использовать валидатор W3C (http://validator.w3.org/), который может проверять HTML5, а также большой спектр других разновидностей языков разметки. Или использовать специальный валидатор HTML5 (+ WAI-ARIA и MathML) по адресу validator.nu (http://validator.nu/).

В конце этого раздела следует привлечь ваше внимание к следующей строке:

<meta charset="utf-8" />

Вам необходимо объявить множество символов документа среди первых 512 байтов, чтобы защититься от серьезной угрозы безопасности. Если нет действительно серьезной причины не делать это, то рекомендуется использовать множество символов UTF-8.

<header>

Верхний колонтитул документа выглядит следующим образом:

```
<header>
  <hgroup>
    <h1>A history of Pop Will Eat Itself</h1>
    <h2>Introducing the legendary Grebo Gurus!</h2>
  </hgroup>
</header>
```

Назначение элемента <header> состоит в создании оболочки вокруг раздела контента, который формирует верхний колонтитул страницы, содержащий обычно логотип компании/графику, заголовок основной страницы, и т.д.

<hgroup>

Вы можете заметить, что в приведенном выше коде, единственным контентом колонтитула является элемент <hgroup>, содержащий два заголовка. Я хочу здесь определить заголовок документа верхнего уровня, плюс подзаголовок/ключевую фразу. Мне нужно, чтобы только заголовок верхнего уровня учитывался в иерархии заголовков документа, и именно это делает элемент <hgroup> - он позволяет учитывать группу заголовков как один заголовок в структуре документа. Подробнее о работе иерархии заголовков в HTML5 будет написано ниже в разделе "Разбивка HTML5 и алгоритм заголовков HTML5".

<footer>

Внизу документа можно увидеть следующий код:

```
<footer>
  <h3 id="copyright">Copyright and attribution</h3>
</footer>
```

<footer> необходимо использовать для размещения контента нижнего колонтитула сайта — если просмотреть нижнюю часть какого-то количества сайтов, то легко увидеть, что нижние колонтитулы могут содержать различные вещи, от уведомления об авторских правах и контактной информации, до заявления о доступности, информации о лицензировании и множества других второстепенных ссылок.

Примечание: Не существует ограничения только на один верхний и нижний колонтитул на страницу — страница может содержать несколько статей, и иметь для каждой статьи верхний и нижний колонтитул.

<nav>

Двигаясь дальше по документу, мы встречаемся со следующей структурой:

```
<nav>
  <h2>Contents</h2>
    <ul>
      <li><a href="#Intro">Introduction</a></li>
      <li><a href="#History">History</a>

      <!—другие навигационные ссылки ... -->

    </ul>
</nav>
```

Элемент <nav> предназначен для разметки навигационных ссылок или других конструкций (например, формы поиска), которые направляют вас на различные страницы текущего сайта, или различные области текущей страницы. Другие ссылки, такие как рекламные ссылки, не учитываются. Можно, конечно, включать заголовки и другие структурные элементы внутрь <nav>, но это не обязательно.

<aside>

Сразу под заголовком документа имеется следующий код:

```
<aside>
  <table>
    <!—- множество кратких фактов в этом месте -->
  </table>
</aside>
```

Элемент <aside> предназначен для разметки фрагментов контента, которые имеют отношение к основному контенту, но не вписываются явно в основной поток изложения. Например, в данном случае мы имеем пакет

кратких интересных фактов и статистики о музыкальной группе, которые не очень хорошо подходят для основного контента. Другими подходящими кандидатами для элементов <aside> являются списки ссылок на внешний связанный контент, справочная информация, цитаты, и боковые панели.

<figure> и <figcaption>

Динамический дуэт из <figure> и <figcaption> был создан для решения достаточно специального множества проблем. Прежде всего, не казалось ли всегда немного семантически сомнительно и неясно размечать рисунок и его подпись как два параграфа, или как пару списка определений, или как-то иначе? И второе, что делать, когда требуется рисунок, состоящий из изображения, или двух изображений, или двух изображений и некоторого текста? Элемент <figure> хорошо подходит, чтобы объединить весь контент, из которого вы хотите составить один рисунок, будет ли это текст, изображения, SVG, видео, или что-то другое. Элемент <figcaption> затем помещается внутри элемента <figure>, и содержит описательный заголовок для этого рисунка. В мой пример включен простой рисунок, чтобы просто показать, как это используется:

```
<figure>
  <img src="pwei.png" alt="Old poppies logo" />
  <figcaption>
    The old poppies logo, circa 1987.<br /> <a
href="http://www.flickr.com/photos/bobcatrock/317261648/">
      Original picture on Flickr</a>, taken by bobcatrock.
  </figcaption>
</figure>
```

<time>

Элемент <time> позволяет определить точно выраженное значение даты и времени, которое одновременно понятно человеку и машине. Например, даты выпуска синглов музыкальной группы помечены следующим образом:

```
<time datetime="1989-03-13">1989</time>
```

Текст между открывающим и закрывающим тегами может быть любым, подходящим для людей, посещающих сайт. При желании можно сделать это следующим образом:

```
<time datetime="1989-03-13">13th March 1989</time>
<time datetime="1989-03-13">March 13 1989</time>
<time datetime="1989-03-13">My nineteenth birthday</time>
```

С другой стороны дата внутри атрибута datetime представлена в стандарте ISO (см. дополнительные сведения в "Советы W3C: Используйте международный формат даты (ISO)" - http://www.w3.org/QA/Tips/iso-date), и является машинно-читаемой датой, поэтому мы получаем двойную выгоду. Можно также добавить время в конце стандартного представления ISO следующим образом:

<time datetime="1989-03-13T13:00">One o'clock in the afternoon, on the 13th of March 1989</time>

Можно добавить также настройку часового пояса, поэтому, например, чтобы представить последний пример в стандартном тихоокеанском времени, можно сделать следующее:

<time datetime="1989-03-13T13:00Z-08:00">One o'clock in the afternoon, on the 13th of March 1989</time>

<article> и <section>

Теперь обратим наше внимание к двум, возможно, наиболее неправильно понимаемым элементам HTML5 - <article> и <section>. При первом знакомстве различие может показаться нечетким, но на самом деле не все так плохо.

В основном элемент <article> предназначен для независимых фрагментов контента, которые будут иметь смысл вне контекста текущей страницы, и могут хорошо объединяться. Такие фрагменты контента включают публикации в блоге, видео и его текстовая запись, новостная история, или одна часть серийной истории.

Элемент <section>, с другой стороны, предназначен для разбиения контента страницы на различные функциональные или тематические области, или разбиения статьи или истории на различные части. Поэтому, например, в моей истории PWEI, структура выглядит следующим образом:

```
<article>
  <section id="Intro">
    <h2>Introduction</h2>
  </section>
  <section id="History">
    <h2>History</h2>
  </section>
  <section id="Discography">
    <h2>Discography</h2>
  </section>
</article>
```

Но можно также структурировать следующим образом:

```
<section id="rock">
  <h2>Rock bands</h2>
  <!--multiple article elements could go in here -->
</section>
<section id="jazz">
  <h2>Jazz bands</h2>
  <!--multiple article elements could go in here -->
</section>
<section id="hip-hop">
  <h2>Hip hop bands</h2>
  <!--multiple article elements could go in here -->
```

</section>

Где остается ?

Итак, со всеми этими прекрасными новыми элементами, которые можно использовать на страницах Web, дни простого элемента сочтены, не так ли? НЕТ. Фактически, по-прежнему имеет совершенно законное применение. Его следует использовать, когда не существует другого более подходящего доступного элемента для объединения области контента, что часто происходит, когда вы используете элемент только для объединения контента в группу с целью стилевого или визуального оформления. Примером в моей истории PWEI является элемент , который использован для создания оболочки вокруг всего контента. Единственная причина его использования здесь в том, чтобы я мог использовать CSS для выравнивания контента по центру в браузере:

```
#wrapper {
  background-color: #ffffff;
  width: 800px;
  margin: 0 auto;
}
```

<mark>

Элемент <mark> предназначен для выделения терминов, значимых в данный момент, или выделения частей контента, к которым вы просто хотите привлечь внимание, но не хотите изменять семантическое значение. Это похоже на то, как при просмотре напечатанной статьи вы выделяете важные для вас строки с помощью цветного маркера. Поэтому, например, вы можете захотеть использовать этот элемент для выделения строк в wiki, которые требуют редакторской правки, или выделения экземпляров термина поиска, который пользователь только что искал на странице, и задание затем для них подходящего оформления в CSS.

Атрибут hidden

Атрибут hidden, когда применяется к любому элементу, скрывает его полностью от любых форм представления/медиа, и должен использоваться, если вы намерены показать контент позже (например, используя JavaScript для удаления этого атрибута), но не хотите, чтобы он отображался в данный момент. Он не должен использоваться для скрытия такого контента, как скрытые вкладки интерфейса с вкладками, так как это совершенно другой способ представления контента в меньшем пространстве, а не скрытие контента вообще.

Разбивка HTML5 и алгоритм заголовков HTML5

Прежде чем продолжить путешествие в направлении более глубокого овладения HTML5, необходимо обсудить одно важное отличие, которое существует между HTML5 и предыдущими версиями спецификации. В HTML мы имеем концепцию разбивки документа, которая является в основном разбивкой документа по его заголовкам, и их иерархии

относительно друг друга, точно таким же образом, как при записи документа в текстовом процессоре вы просматриваете документ в виде плана-конспекта. На самом деле я фактически создал план-конспект этого документа, вкладывая списки друг в друга для создания оглавления в начале статьи. План-конспект этой статьи выглядит примерно следующим образом:

- Введение в структурные элементы HTML5
  - Как были выбраны названия элементов?
  - Почему нет элемента <content>?
- Представляем пример страницы HTML5
  - Некоторые мета-различия
  - <header>
  - <hgroup>
  - <footer>
  - <nav>
  - <aside>
  - <figure> и <figcaption>
  - <time>
  - <article> и <section>
    - Где остается ?
    - <mark>
    - Атрибут hidden
- Разбивка HTML5 и алгоритм заголовков HTML5
- Как заставить это работать в старых браузерах
- Заключение

Поэтому "Новые структурные элементы в HTML5" будет <h1>, "Введение в структурные элементы HTML5" будет <h2>, и т.д. В HTML4 мы привыкли к тому факту, что имеется шесть возможных уровней заголовков, и каждый уровень заголовков диктуется реально используемым элементом, что означает, что вполне возможно получить совершенно искаженную иерархию заголовков, если использовать неправильные уровни заголовков, или даже если часть вашего контента объединяется в другой CMS (Системе управления контентом).

HTML5 решает эту проблему, создавая иерархию заголовков на основе относительного вложения различных разделов документа. Новый раздел документа создается, когда вы используете так называемый разделяющий контент – элементы <article>, <section>, <nav>, и <aside>. Возьмем следующий пример.

```
<h1>My title</h1>
<div>
  <h2>My subtitle</h2>
</div>
```

Посмотрите первый пример разбивки в действии (http://dev.opera.com/articles/view/new-structural-elements-in-html5/outlining1.html).

HTML 4 будет считать это заголовком первого уровня, за которым следует заголовок второго уровня, но HTML5 будет считать это как два заголовка первого уровня. Почему? Так как не является разделяющим элементом, то новый раздел в иерархии не создается. Чтобы исправить это, необходимо заменить на разделяющий элемент:

```
<h1>My title</h1>
<section>
  <h2>My subtitle</h2>
</section>
```

Посмотрите на второй пример разбивки в действии (http://dev.opera.com/articles/view/new-structural-elements-in-html5/outlining2.html).

Ни один из браузеров в настоящее время не реализовал алгоритм HTML5 для разбивки, но вы уже можете получить представление о том, как это работает, используя Расширение Opera HTML5 Outliner (https://addons.opera.com/addons/extensions/details/html5-outliner/1.0/?display=en), сетевой HTML5 outliner Джефри Шеддона (http://gsnedders.html5.org/outliner/), или Google HTML5 outliner (http://code.google.com/p/h5o/). Попробуйте пропустить приведенные выше примеры через один из этих инструментов, если вы мне не верите на слово. И в будущем в действительности не нужно будет беспокоиться об иерархии h1, h2, h3, и т.д., так как независимо от реально используемых элементов заголовков, алгоритм будет создавать одну и ту же иерархию на основе вложенности разделов документа. Но пока об этом необходимо волноваться, так как ни один браузер (или считыватель экрана) этого не поддерживает!

Поэтому возникает естественный вопрос – "Зачем вообще об этом беспокоиться"? На самом деле этот новый способ создания план-конспекта документа/иерархии заголовков имеет два основных преимущества по отношению к старому:

1. Можно иметь любое количество уровней заголовков — количество не ограничивается шестью.
2. Если контент переносится в какую-то другую CMS (Систему управления контентом), что приводит к тому, что уровни заголовков h1, h2, h3, и т.д.становятся неправильными, алгоритм будет по-прежнему создавать правильную иерархию несмотря ни на что.

Примечание: Иерархия заголовков HTML5 является в действительности достаточно старой идеей, первоначально намеченной Тимом Бернерс-Ли еще в 1991 г. (http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html):

На самом деле, я бы предпочел вместо <H1>, <H2> и т.д. для заголовков [которые возникают из AAP DTD] иметь допускающий вложение друг в друга элемент <SECTION>..</SECTION>, и базовый <H>..</H>, который на любом уровне в разделах создавал бы заголовок требуемого уровня.

Как заставить это работать в старых браузерах

Старые браузеры: всегда бедствие для самого нашего существования, когда пытаешься справиться с новыми блестящими игрушками в Web! Фактически проблема здесь со всеми браузерами – ни один браузер в настоящее время по существу не распознает и не поддерживает новые структурные элементы HTML5. Но не надо бояться, вы можете, тем не менее, заставить их работать сегодня в разных браузерах с минимальными усилиями.

Прежде всего, если вы помещаете неизвестный элемент на web-страницу, по умолчанию браузер будет интерпретировать его просто как , т.е. анонимный встроенный элемент. Предполагается, что большинство рассмотренных в этой статье элементов HTML5, ведут себя как блочные элементы, поэтому простейший способ заставить их вести себя правильно в старых браузерах, состоит в задании для них display:block; в коде CSS:

```
article, section, aside, hgroup, nav, header, footer, figure, figcaption {
  display: block;
}
```

Это решает все проблемы для всех браузеров, за исключением одного. Хотите угадать какого? … Удивительно, не правда ли, что IE оказался сложнее других браузеров, и отказывается оформлять элементы, которые не распознает? Исправление для IE кажется нелогичным, но к счастью достаточно простым. Для каждого используемого элемента HTML5 необходимо вставить строку JavaScript в заголовок документа следующим образом:

```
<script>
   document.createElement('article');
   document.createElement('section');
   document.createElement('aside');
   document.createElement('hgroup');
   document.createElement('nav');
   document.createElement('header');
   document.createElement('footer');
   document.createElement('figure');
   document.createElement('figcaption');
</script>
```

Теперь IE будет магическим образом применять стили для этих элементов. Печально, что приходиться использовать JavaScript, чтобы заставить работать CSS, но, по крайней мере, мы продвинулись вперед. Почему это все-таки работает? Никто, с кем я говорил об этом, в действительности не знает. Существует также проблема с этими стилями, которые не выводятся на принтер при печати документов HTML5 из IE.

Примечание: Проблема с печатью в IE может быть решена с помощью библиотеки JavaScript HTML5 Shiv (http://code.google.com/p/html5shiv/), которая справляется также с добавлением строк document.createElement. Необходимо поместить ее в Условные комментарии (http://dev.opera.com/articles/view/supporting-ie-with-conditional-comments/) для IE меньше IE9, чтобы современные браузеры не выполняли JS, который им не нужен.

Заключение

На этом завершается обсуждение новых структурных элементов в HTML5. Если вам требуется дополнительная помощь с HTML5, многое можно найти на сайте dev.opera.com, и можно также проконсультироваться на сайте HTML5 doctors (http://html5doctor.com/).

Ключевые слова: *HTML-документ, тег, HTML-код, фрейм, заголовок, параграф, гиперссылка, геолокация, атрибуты, контент, сайт, Javascript, браузер, маркировка,web-страница.*

Контрольные вопросы

1. Из каких двух частей состоит HTML-программа?
2. Какими тегами задаётся элемент, содержащий головную часть программы?
3. Сколько уровней заголовков поддерживает HTML?
4. Для чего в основном используются фреймы?
5. Новые элементы в HTML5 и их функции.

ГЛАВА 6. ТАБЛИЧНАЯ И КОНТЕЙНЕРНАЯ РАЗМЕТКИ

Таблицы

В web-дизайне таблицы являются одним из основных методов структурирования информации. Они представляют данные в виде удобных для восприятия колонок и строк, что значительно упрощает анализ информации. С их помощью можно легко отделить одну часть страницы от другой.

Основным тегом для обозначения таблицы является <table>. Элемент **TABLE** представляет собой тег-контейнер, в котором размещается содержимое таблицы. Построение таблицы осуществляется по строкам, для обозначения которых применяется контейнер TR. Внутри контейнера строк помещаются контейнеры для обозначения ячеек. Стандарт HTML определяет два типа контейнеров для обозначения ячеек <th> и <td>. Первый предназначен для обозначения заголовков, а второй для данных в ячейках. Отличие этих двух элементов заключается лишь в том как их содержимое отображается браузером, заголовки большинство браузеров выделяют полужирным шрифтом и центрируют в своих ячейках. Ячейка данных может содержать текст, изображения, списки, параграфы, формы, горизонтальные линейки, таблицы и т. д.

Примеры

Рассмотрим вышеизложенное на примерах различного вида таблиц.

```
<html>
<body>
<p>
Каждая таблица начинается с тега table.
Каждая строка таблицы начинается с тега tr.
Каждый элемент данных таблицы начинается с тега td.
</p>
<h1>Это пример простейшей таблицы, содержащей одну строку и одну ячейку.</h1>
<table border="1">
<tr>
  <td>Одна строка и одна ячейка</td>
</tr>
</table>
<h1>Одна строка и три столбца:</h1>
<table border="1">
<tr>
  <td> столбец 1</td>
  <td> столбец 2</td>
  <td> столбец 3</td>
</tr>
</table>
<h1>Две строки и три столбца:</h1>
<table border="1">
<tr>
  <td>1.1</td>
  <td>1.2</td>
  <td>1.3</td>
</tr>
<tr>
  <td>2.1</td>
  <td>2.2</td>
  <td>2.3</td>
</tr>
</table>
<h1>Рамка таблицы</h1>
<h1>Обычная рамка:</h1>
<table border="1">
<tr>
  <td>Первая</td>
  <td>строка</td>
</tr>
```

```
<tr>
  <td>Вторая </td>
  <td>строка</td>
</tr>
</table>
<h1>Толстая рамка:</h1>
<table border="10">
<tr>
  <td>Первая</td>
  <td>строка</td>
</tr>
<tr>
  <td>Вторая </td>
  <td>строка</td>
</tr>
</table>

<table border="1">
<tr>
<td>строка 1, ячейка 1</td>
<td>строка 1, ячейка 2</td>
</tr>
<tr>
<td>строка 2, ячейка 1</td>
<td>строка 2, ячейка 2</td>
</tr>
</table>
</body>
</html>
```

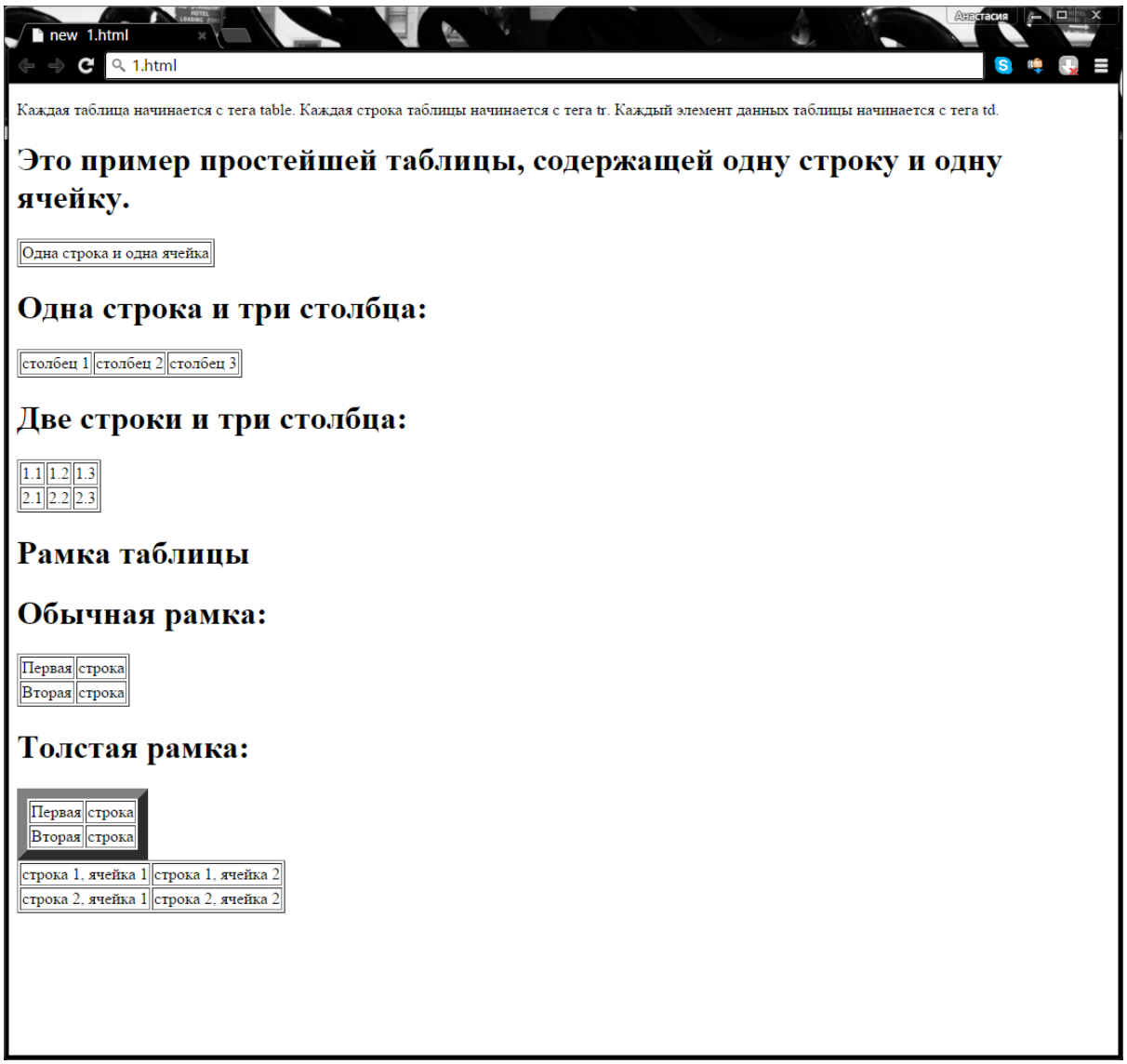


Рис. 6.1 Пример выполнения данного HTML-кода

Для тега table определены следующие атрибуты.

- align - определяет способ горизонтального выравнивания таблицы на странице. Возможные значения: left, center, right. Значение по умолчанию - left.
- valign - должен определять способ вертикального выравнивания для содержимого таблицы. Возможные значения: top, bottom, middle.
- border - определяет ширину внешней рамки таблицы (в пикселах). При BORDER="0" или при отсутствии этого параметра рамка отображаться не будет.
- cellpadding - определяет расстояние (в пикселах) между рамкой каждой ячейки таблицы и содержащимся в ней материалом.
- cellspacing - определяет расстояние (в пикселах) между границами соседних ячеек.

Дополнительные примеры

Таблица без рамки

```
<html>
<body>

<h4>Эта таблица не имеет рамки:</h4>
<table>
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>

<h4>И эта таблица не имеет рамки:</h4>
<table border="0">
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>600</td>
</tr>
</table>

</body>
</html>
```

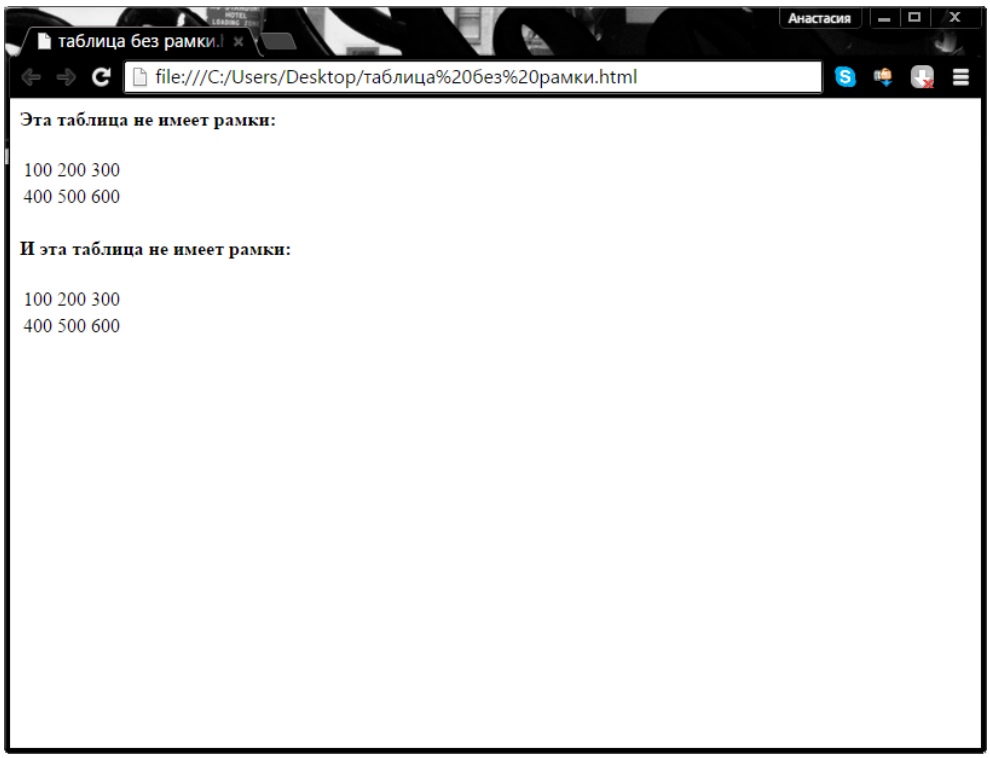


Рис. 6.2 Пример выполнения данного HTML-кода

Заголовки в таблице

```
<html>
<body>

<h4>Заголовки таблицы:</h4>
<table border="1">
<tr>
  <th>Имя </th>
  <th>Телефон </th>
  <th>Телефон </th>
</tr>
<tr>
  <td>Калиткин</td>
  <td>202 55 55 </td>
  <td>456 77 84 </td>
</tr>
</table>

<h4>Вертикальные заголовки:</h4>
<table border="1">
<tr>
  <th>Фамилия:</th>
  <td>Калиткин</td>
</tr>
<tr>
```

```
  <th>Телефон :</th>
  <td>202 55 55</td>
</tr>
<tr>
  <th>Телефон:</th>
  <td>456 77 84</td>
</tr>
</table>
</body>
</html>
```

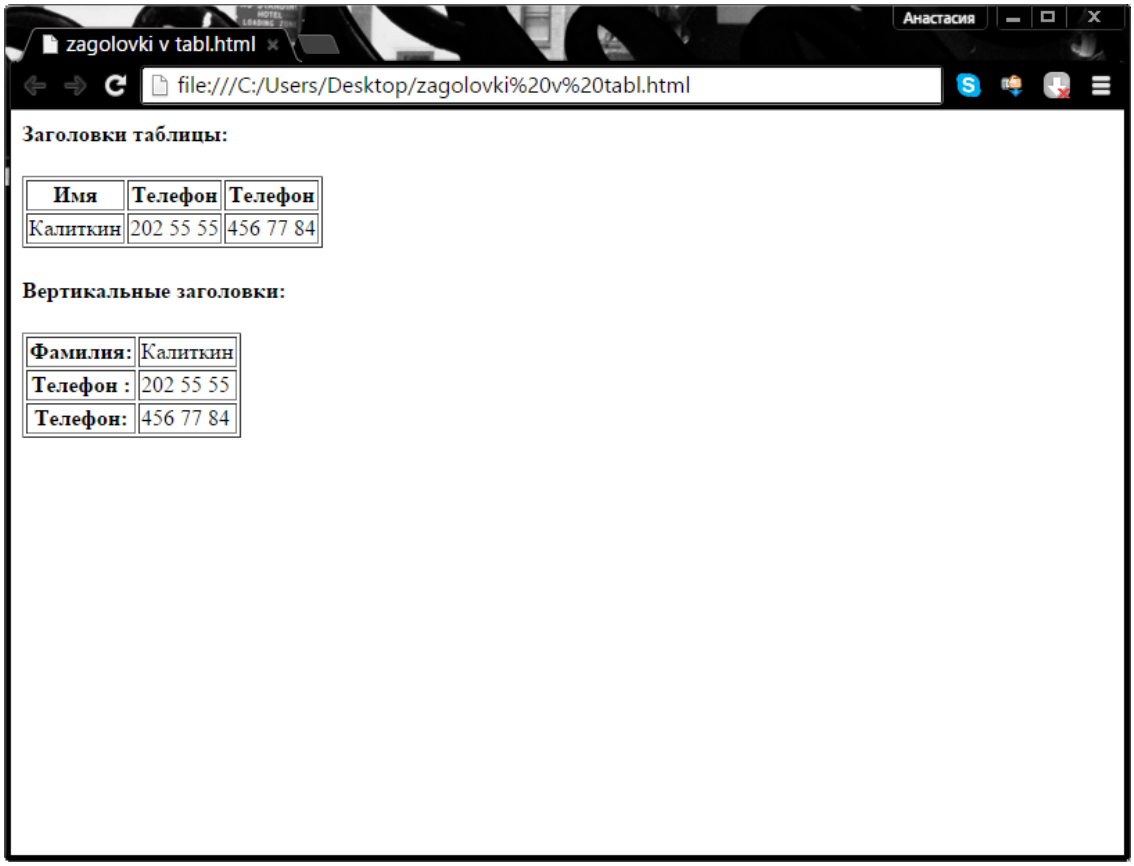


Рис. 6.3 Пример выполнения данного HTML-кода

Пустые ячейки

```
<html>
<body>

<table border="1">
<tr>
  <td>Некоторый текст </td>
  <td>Некоторый текст </td>
</tr>
<tr>
  <td></td>
```

```
  <td>Некоторый текст </td>
</tr>
</table>

<p>
Как можно видеть, одна из ячеек не имеет рамки.
Это потому, что она пустая. Попробуйте вставить в ячейку пробел.
Она по-прежнему не будет иметь рамки.
</p>

<p>
Хитрость состоит в том, чтобы вставить в ячейку неразрывный пробел.
</p>

<p>Неразрывный пробел является символьным объектом.
</p>

<p>Объект неразрывного пробела начинается с амперсанда ("&"),
затем следуют буквы "nbsp", и в конце стоит точка с запятой (";")
</p>

<p>
</p>

</body>
</html>
```

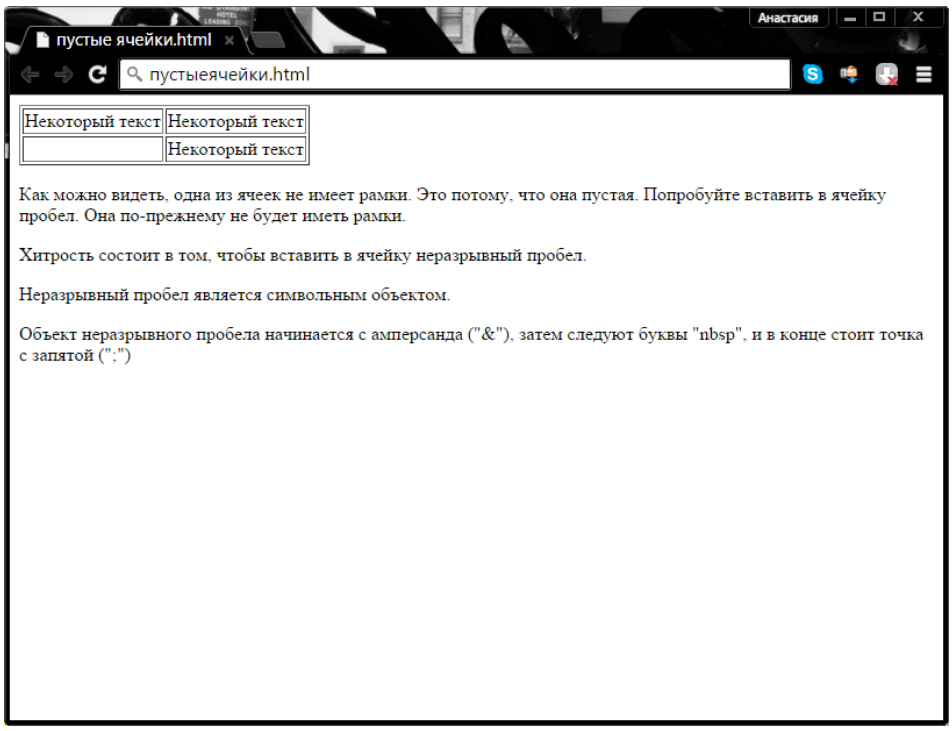


Рис. 6.4 Пример выполнения данного HTML-кода

Заглавие таблицы

```
<html>
<body>

<h4>
Эта таблица имеет заглавие
и толстую рамку:
</h4>

<table border="6">
<caption>Заглавие</caption>
<tr>
  <td>100</td>
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>

</body>
</html>
```

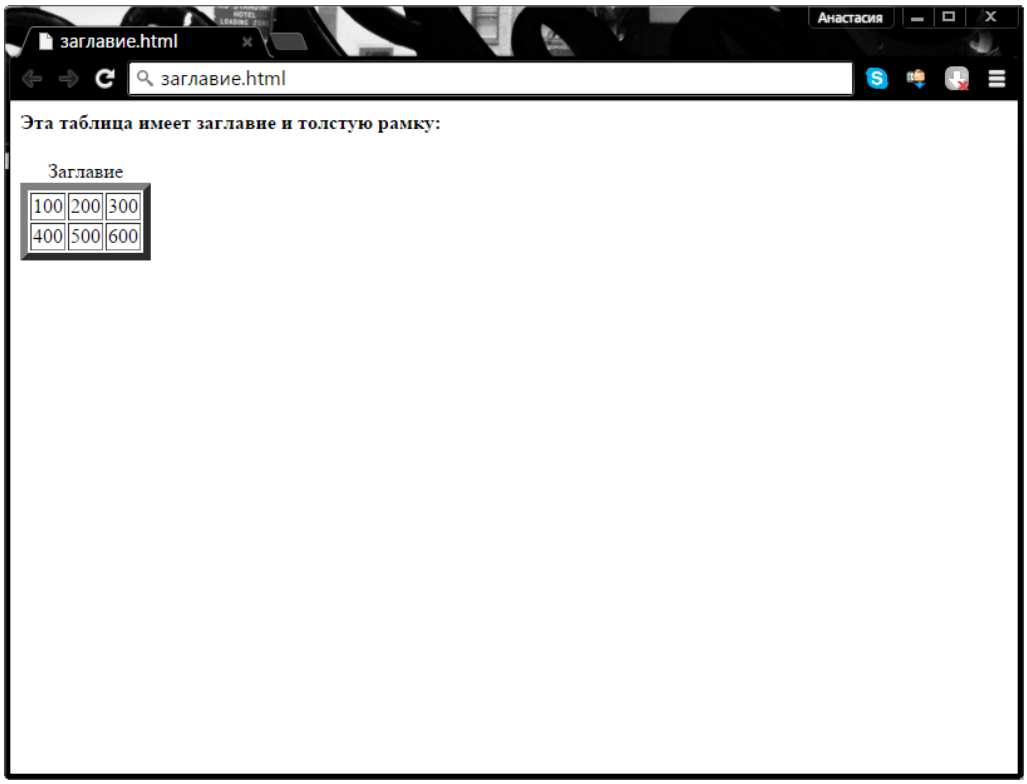


Рис. 6.5 Пример выполнения данного HTML-кода

Ячейки таблицы, которые охватывают более одной строки/столбца

```
<html>
<body>

<h4>Ячейка, которая охватывает два столбца:</h4>
<table border="1">
<tr>
  <th>Организация</th>
  <th colspan="2">Телефон</th>
</tr>
<tr>
  <td>Интернет-Университет Информационных Технологий</td>
  <td>253-9312</td>
  <td>253-9313</td>
</tr>
</table>

<h4>Ячейка, которая охватывает две строки:</h4>
<table border="1">
<tr>
  <th>Организация:</th>
  <td> Интернет-Университет Информационных Технологий </td>
</tr>
<tr>
  <th rowspan="2">Телефон:</th>
  <td>253-9312</td>
</tr>
<tr>
  <td>253-9313</td>
</tr>
</table>

</body>
</html>
```

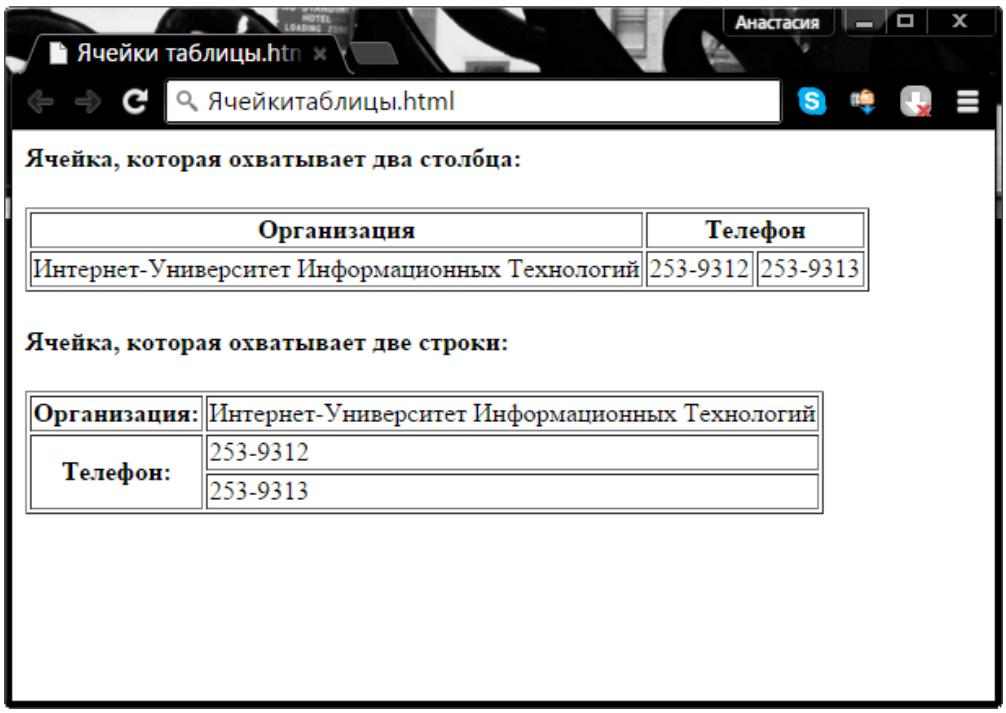


Рис. 6.6 Пример выполнения данного HTML-кода

Теги внутри таблицы

```
<html>
<body>
<table border="1">
<tr>
 <td>
  <p>Это параграф </p>
  <p>Это другой параграф </p>
 </td>
 <td>Эта ячейка содержит таблицу:
  <table border="1">
  <tr>
    <td>A</td>
    <td>B</td>
  </tr>
  <tr>
    <td>C</td>
    <td>D</td>
  </tr>
  </table>
 </td>
</tr>
<tr>
 <td>Эта ячейка содержит список
  <ul>
   <li>элемент 1</li>
```

```
   <li>элемент 2</li>
   <li>элемент 3</li>
  </ul>
 </td>
 <td>КОНЕЦ</td>
</tr>
</table>

</body>
</html>
```

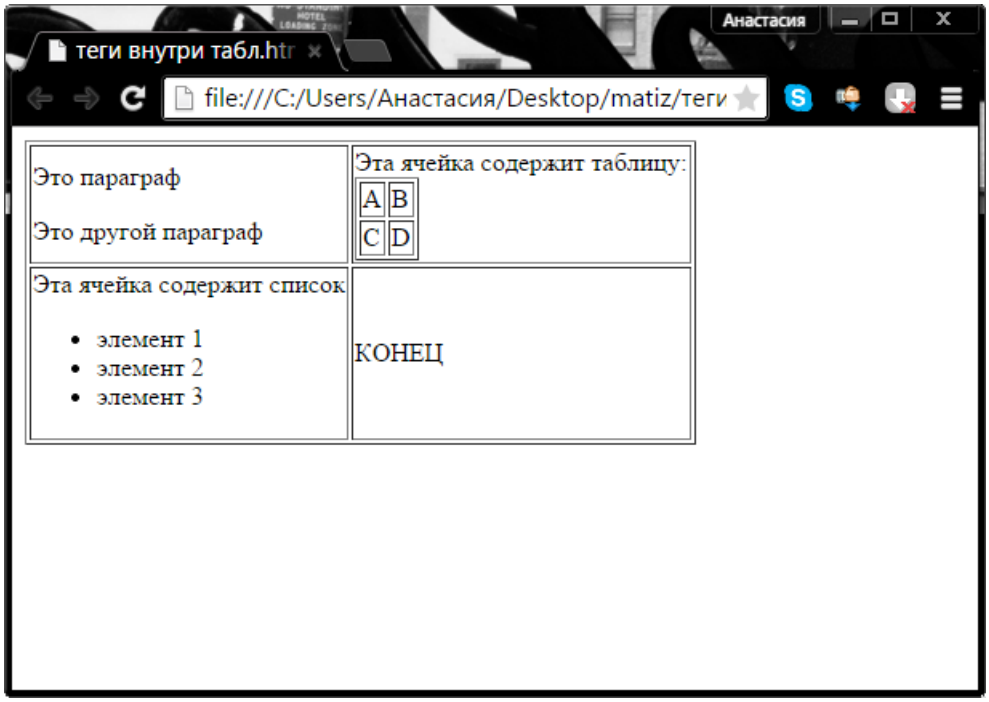


Рис. 6.7 Пример выполнения данного HTML-кода

Отступ от рамки до содержимого ячейки

```
<html>
<body>

<h4>Без отступа:</h4>
<table border="1">
<tr>
 <td>Первая </td>
 <td>Строка</td>
</tr>
<tr>
 <td>Вторая </td>
 <td>Строка</td>
</tr>
</table>
```

```
<h4>С отступом (cellpadding) :</h4>
<table border="1"
cellpadding="10">
<tr>
  <td>Первая</td>
  <td>Строка</td>
</tr>
<tr>
  <td>Вторая</td>
  <td>Строка</td>
</tr>
</table>

</body>
</html>
```

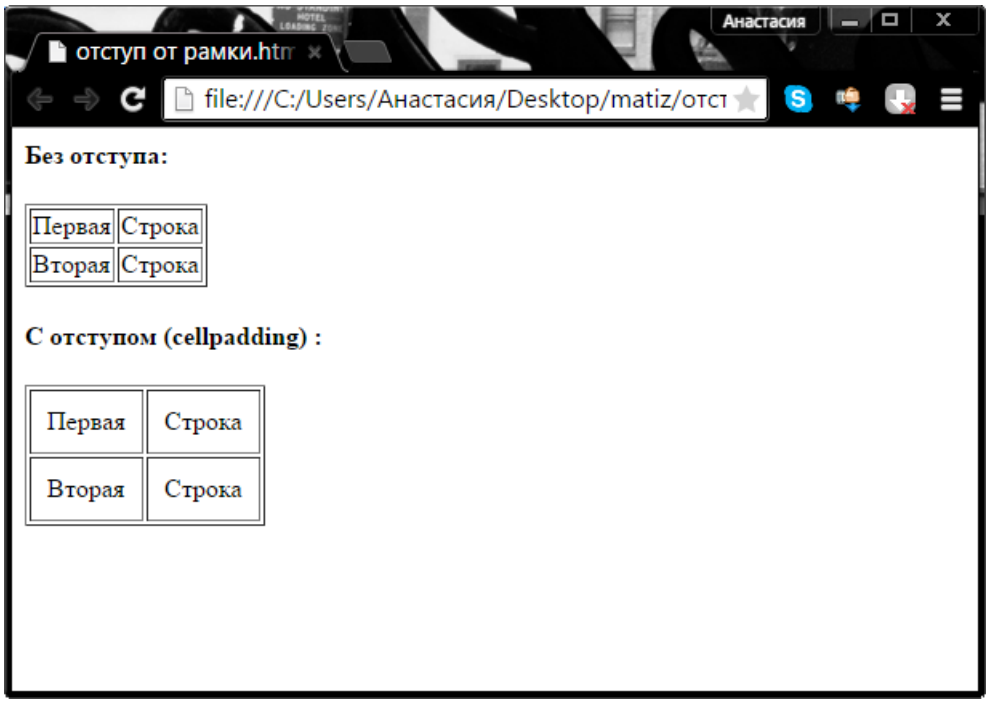


Рис. 6.8 Пример выполнения данного HTML-кода

Компоновка документа в HTML

Выше были рассмотрены основные теги HTML. Используя их, уже можно создавать свои страницы. Но создание страницы это не только верстка материалов, но и компоновка всех элементов (меню, заголовок страницы, основное информационное наполнение, баннеры и др.) на странице HTML документа. Если все элементы должны располагаться один за другим, тогда понятно как поступить - размещаем все блоки последовательно друг за другом и получаем результат. А как быть, если два элемента должны размещаться на одном уровне по горизонтали? Например, меню должно находиться слева или справа от текста? Ответ очевиден, использовать свойство HTML тегов, позволяющее вкладывать один тег в другой.

Наиболее распространенным способом компоновки страницы является использование таблиц. Ниже будет рассмотрен этот способ.

Компоновка HTML - использование таблиц

Как известно тег <table> используется для отображения таблиц в HTML документах. У этого тега имеется атрибут border, указывающий толщину границы. И если его значение равно нулю, то граница не видна, видно лишь содержимое ячеек. Это и используется для компоновки страниц. В распоряжении дизайнера появляется не одна большая область для вставки элементов, а бесконечное число более мелких зон.

Наберите следующий пример.

```
<html>
<body>
<p>Часть этой страницы отформатирована с помощью двух столбцов, как газетная страница. Все, что находится ниже этого текста, располагается в двух ячейках таблицы. Как можно видеть, есть левый столбец и правый столбец.</p>
<table border="0" width="100%">
<tr><td width="50%" valign="top">Этот текст выводится в левом столбце.</td><td width="50%" valign="top">А этот текст выводится в правом столбце.</td></tr>
</table>
</body>
</html>
```

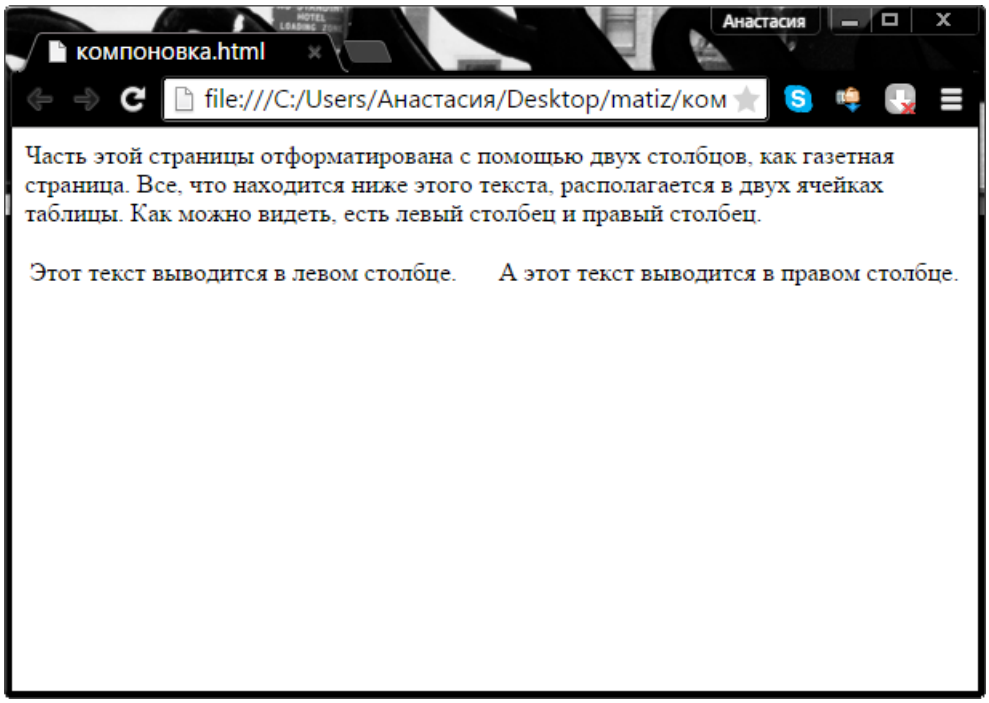


Рис. 6.9 Пример выполнения данного HTML-кода

В этом примере тег HTML <table> используется для деления части Web-страницы на два столбца, причем в данном случае ширина столбцов

одинакова. Однако это не является обязательным условием, что показано в следующем примере.

```
<html>
<body>
<p>Часть этой страницы отформатирована с помощью двух столбцов, как
газетная страница. Все что находится ниже этого текста располагается в двух
ячейках таблицы. Как можно видеть, есть левый столбец и правый столбец.</p>
<table border="0" width="100%">
<tr><td width="20%" valign="top">Этот текст выводится в левом
столбце. Ширина этого столбца 20%</td><td width="80%"
valign="top">А этот текст выводится в правом столбце. Ширина столбца
80%</td></tr>
</table>
</body>
</html>
```

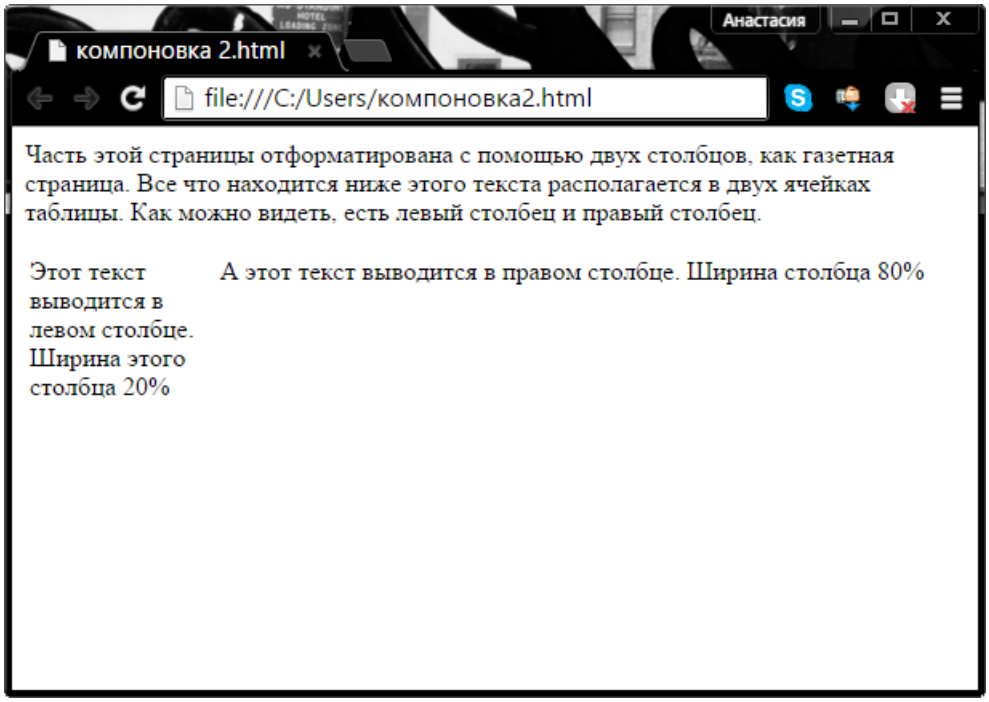


Рис. 6.10 Пример выполнения данного HTML-кода

Особенность состоит в использовании таблицы без видимой рамки и возможно небольшого дополнительного отступа внутри ячеек таблицы.

Не имеет значения, сколько текста будет размещено на этой странице, он будет оставаться в пределах своего столбца.

После добавления таблицы для разметки страницы мы получаем несколько независимых областей, с которыми мы можем работать независимо, задавать цвет фона, шрифты, добавлять вложенные таблицы для

разбиения ячеек полученной части еще на более мелкие элементы. Однако создавать большую вложенность таблиц не рекомендуется - так увеличивается размер страницы. Гораздо рациональнее использовать объединение ячеек.

```
<html>
<body>
<p>Часть этой страницы отформатирована с помощью двух столбцов, как
газетная страница, и общего заголовка. Все что находится ниже этого текста
располагается в трех ячейках таблицы, причем для верхней ячейки указан атрибут
colspan="2". Как можно видеть, есть левый столбец и правый столбец, а также
общий заголовок. Для каждой ячейки указан свой цвет фона.</p>
<table border="0" width="100%" cellpadding="0" cellspacing="0">
<tr height="150px"><td width="100%" valign="top" colspan="2" bgcolor="#cccccc">
Этот текст выводится в верхней ячейке таблицы.</td></tr>
<tr height="500px"><td width="20%" valign="top" bgcolor="#dddddd">
Этот текст выводится в левом столбце.
Ширина этого столбца 20%</td><td width="80%" valign="top" bgcolor="#eeeeee">A
этот текст выводится в правом столбце. Ширина столбца 80%</td></tr>
</table>
</body>
</html>
```

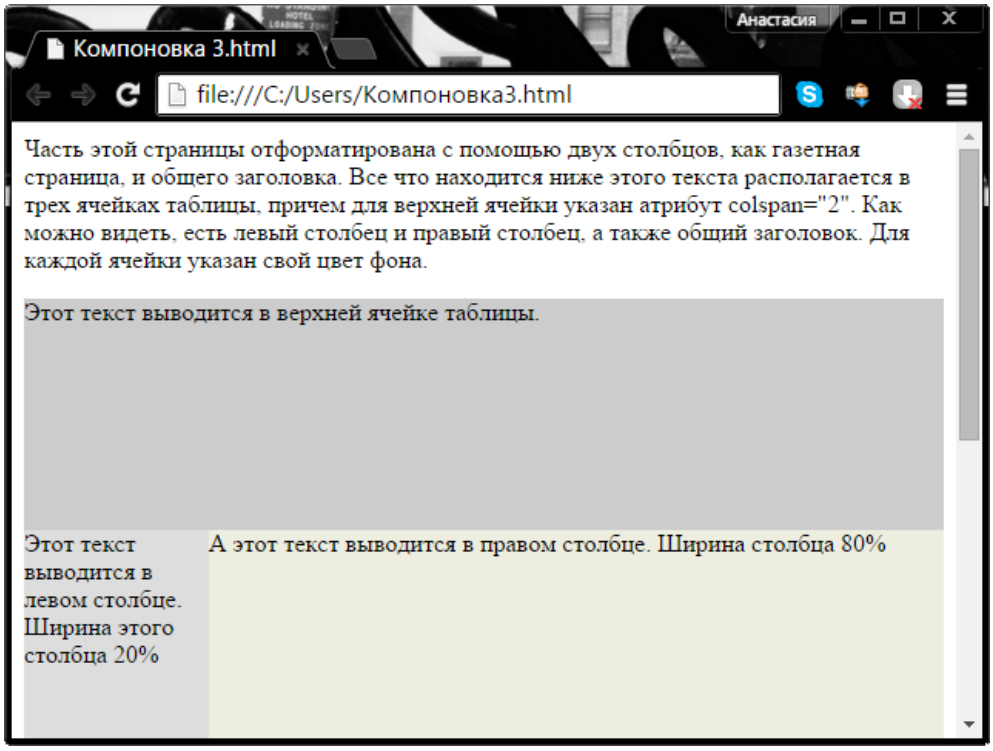


Рис. 6.11 Пример выполнения данного HTML-кода

Этот же эффект можно получить и применив две таблицы вложенных одна в другую.

```
<html>
<body>

<p>Часть этой страницы отформатирована с помощью двух столбцов, как
газетная страница, и общего заголовка. Все что находится ниже этого текста
располагается в ячейках таблицы, причем здесь применяется вложенность одной
таблицы в другую. Как можно видеть, есть левый столбец и правый столбец, а
также общий заголовок. Для каждой ячейки указан свой цвет фона.</p>

<table border="0" width="100%" cellpadding="0" cellspacing="0">
<tr height="150px"><td width="100%" valign="top" bgcolor="#cccccc">
Этот текст выводится в верхней ячейке таблицы.</td></tr>

<tr height="500px"><td width="100%">

<table border="0" width="100%" height="100%" cellpadding="0" cellspacing="0">

<tr><td width="20%" valign="top" bgcolor="#dddddd">

Этот текст выводится в левом столбце.
Ширина этого столбца 20% </td><td width="80%" valign="top" bgcolor="#eeeeeee">

А этот текст выводится в правом столбце. Ширина столбца 80%</td></tr></table>

</td></tr>

</table>

</body>

</html>
```

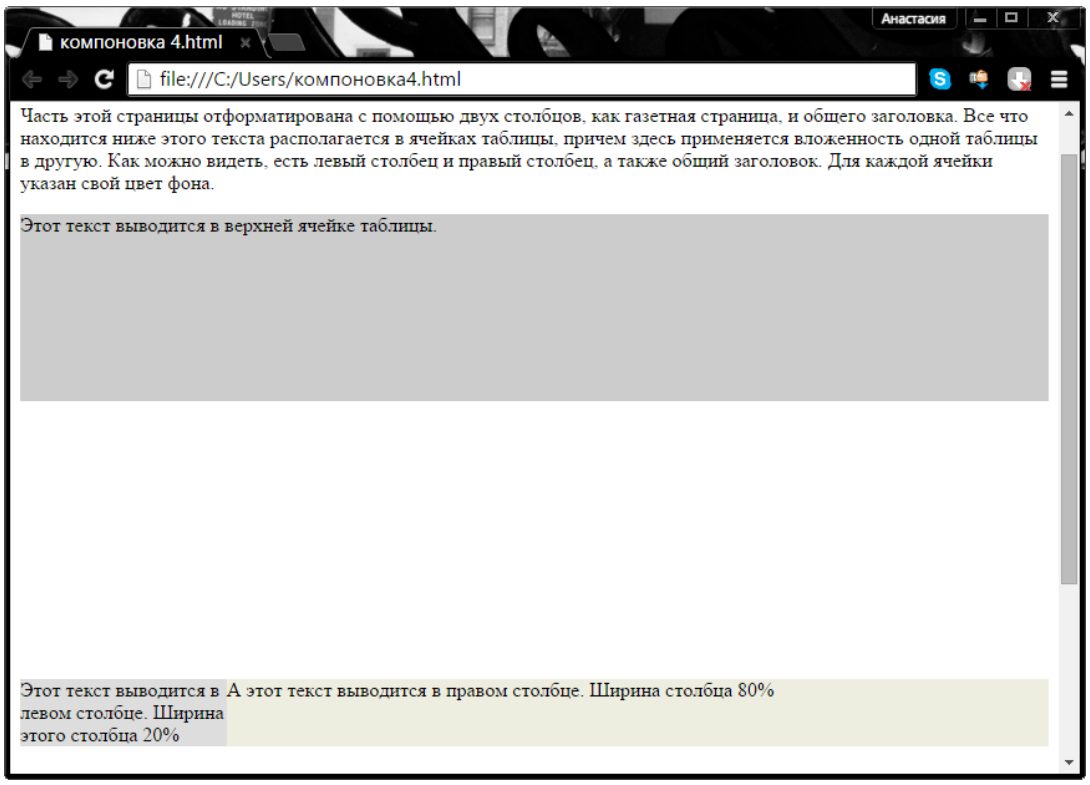


Рис. 6.12 Пример выполнения данного HTML-кода

Как можно видеть мы получили одинаковый результат, однако в первом случае размер кода получился значительно меньше, и кроме того сам исходный код более понятен.

Ключевые слова: *таблица, рамка, столбец, разметка страницы, ячейка, тег-контейнер, компоновка, отступ, граница, выравнивание таблицы.*

Контрольные вопросы:

1. Какие атрибуты определены для тега table?
2. Почему рациональнее использовать объединение ячеек?
3. Что указывает атрибут border?
4. Что такое тег-контейнер?
5. Каково отличие между элементами <th> и <td>?

ГЛАВА 7. ПРОГРАММИРОВАНИЕ ФОРМ И ЭЛЕМЕНТОВ ФОРМ

Формы HTML предназначены для организации взаимодействия с пользователем. Они позволяют вводить текст, осуществлять выбор из предложенных значений при помощи списков или кнопок. С помощью форм можно организовать интерактивный обмен информацией между Web-страницей и сервером. Можно определить формы как электронные бланки для заполнения различных данных таких как, например, имя, возраст, выбор страны проживания и других. Как правило, форма работает совместно с

установленным на сервере сценарным приложением, обрабатывающим введенную информацию.

Примеры форм:

Форма поиска

Одна из наиболее распространенных форм. Пользователь получает возможность ввести искомый запрос, определить область поиска и отправить запрос на сервер для обработки.

```
<html>
<body>
<form name="input" action="html_form_action.asp" method="get">
 <table border=1 bgcolor="#ddffdd">
  <tr>
   <td align="center">
    <input type="text" name="search" size="50" value="Строка для поиска">
    <input type="submit" value="Поиск">
    <br>
    <input type="checkbox" name="news">Искать в новостях
    <input type="checkbox" name="arhive">Искать в архивах
   </td>
  </tr>
 </table>
</form>
</body>
</html>
```

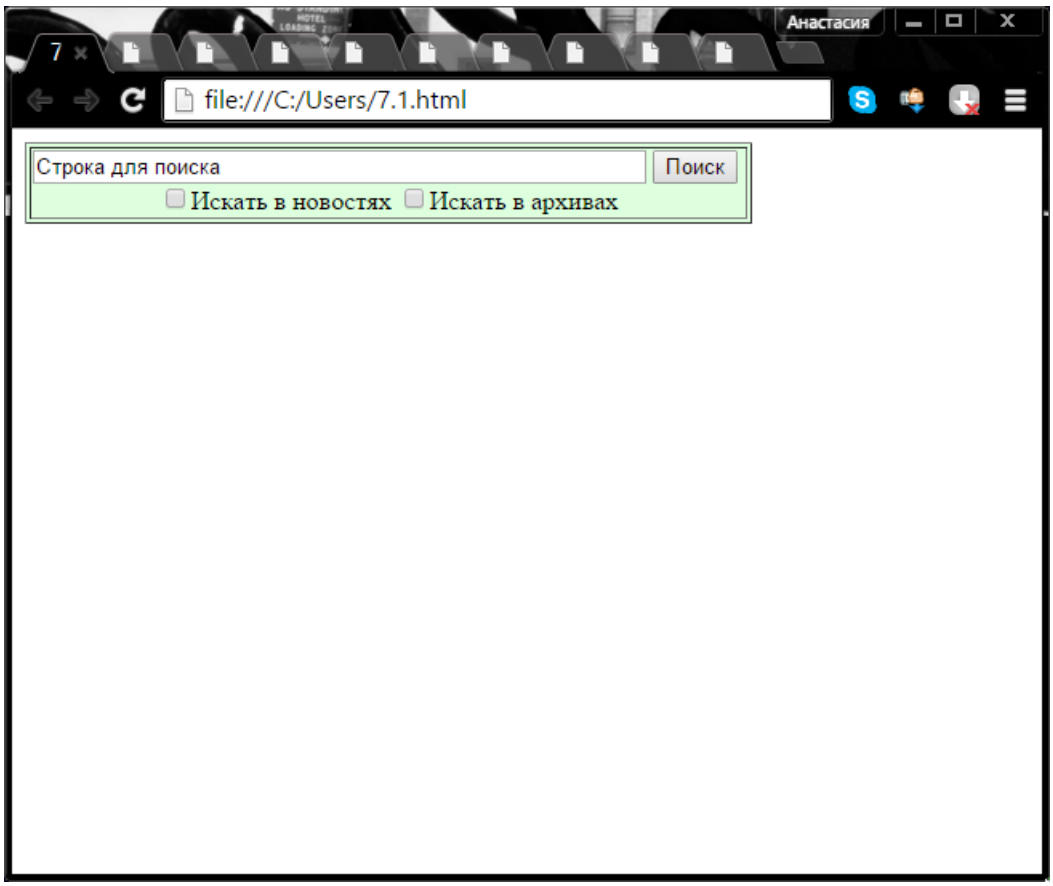


Рис. 7.1 Пример выполнения данного HTML-кода

Отправка e-mail из формы

Вторая форма представляет собой форму заказа электронной рассылки ТУИТа, заполненная форма отправляется по электронной почте для последующей обработки.

```
<html>
<body>
<form action="mailto:info@tuit.uz" method="post" enctype="text/plain">
  <h3>Подписка на новостную рассылку ИНТУИТ</h3>
  Имя:<br>
  <input type="text" name="name" value="ваше имя" size="20">
  <br>
  Mail:<br>
  <input type="text" name="mail" value="ваш e-mail" size="20">
  <br>
  Я хочу получать:
  <br>
  <input type="checkbox" name="news"> Новости о жизни ТУИТа
  <br>
  <input type="checkbox" name="courses"> Информацию о новых курсах
  <br>
  <input type="checkbox" name="books"> Информацию о новых изданиях
```

```
  <br>
  Комментарий:<br>
  <textarea rows="5" cols="30">
  </textarea>
  <hr>
  <input type="submit" value="Послать">
  <input type="reset" value="Сброс">
</form>
</body>
</html>
```

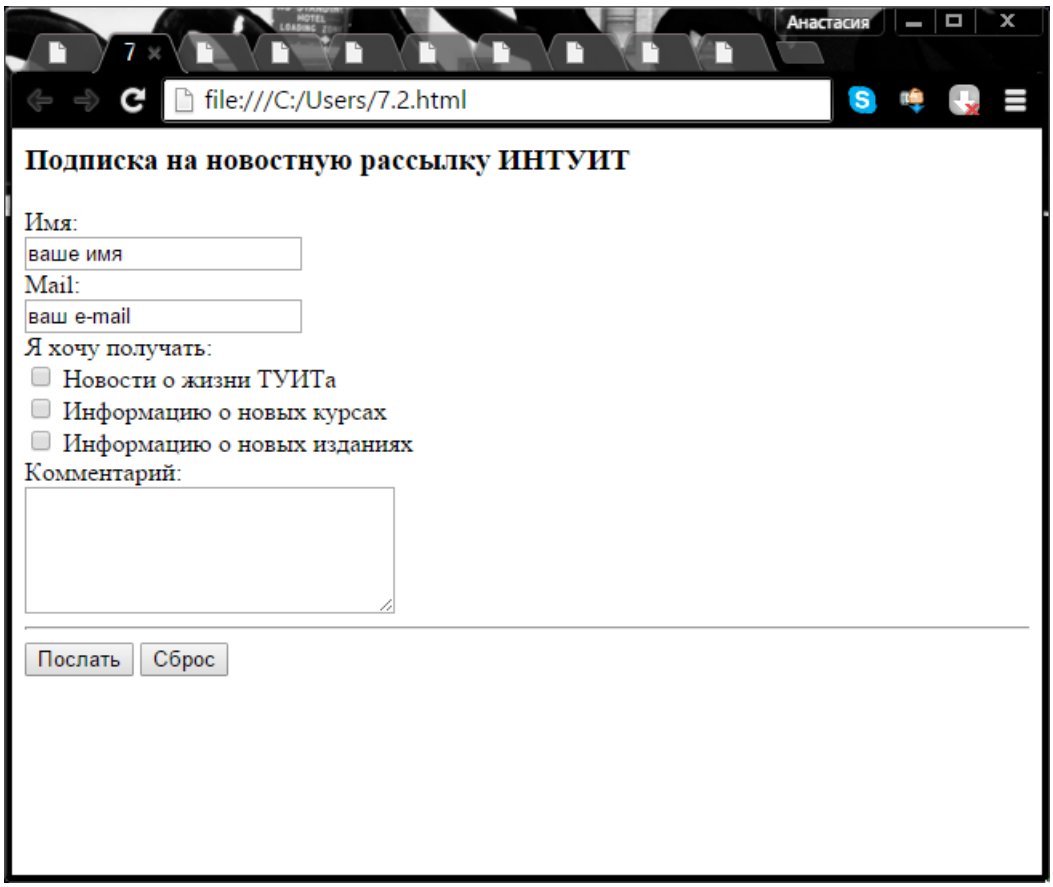


Рис. 7.2 Пример выполнения данного HTML-кода

Формы

Форма является областью, которая может содержать элементы, позволяющие пользователю вводить информацию (такие как текстовые поля, поля многострочного текста, раскрывающиеся меню, переключатели, флажки, и т.д.).

Форма определяется с помощью тегов <form> </form>, между которыми располагаются поля ввода, кнопки, а также все необходимые элементы оформления формы.

Тег <form> имеет ряд атрибутов, из которых необходимо выделить атрибуты action и method. Без этих атрибутов форма не сможет передать информацию от пользователя на сервер.

<form action="html_form_action.asp" method=get>

Атрибут Action указывает URL-адрес объекта, который должен получить данные формы.

Атрибут method может иметь два значения: get и post.

Значение атрибута method=get заставляет Web-браузер передать все данные формуляра по URL-адресу, заданному в action. При этом введенные при заполнении формы данные просто добавляются в адресную строку с использованием разделителя – знака вопроса. Этот метод удобен для небольших форм.

Значение атрибута method=post заставляет Web-браузер, прежде всего, связаться с сервером, обрабатывающим форму, и только после установки связи приступить к передаче данных, для обработки которых будут использоваться специальные сценарии.

Поля ввода

Большинство элементов ввода и управления в форме можно описать при помощи тега <input>, обязательными для которого являются атрибуты name (приписывает данному элементу ввода уникальное имя, использующееся для дальнейшей обработки формы) и type (определяет тип элемента управления или ввода).

Текстовые поля

Текстовое поле (type=text) определяет однострочное поле ввода и используется, когда необходимо, чтобы пользователь ввел в форму данные в произвольной форме, но ограниченные по объему (слова, словосочетания, числа и т.д.).Следующий пример демонстрирует простейшую форму для ввода имени и фамилии.

```
<html>
<body>
<form>
  Имя:
  <input type="text" name="firstname">
  <br>
  Фамилия:
  <input type="text" name="lastname">
</form>
</body>
</html>
```

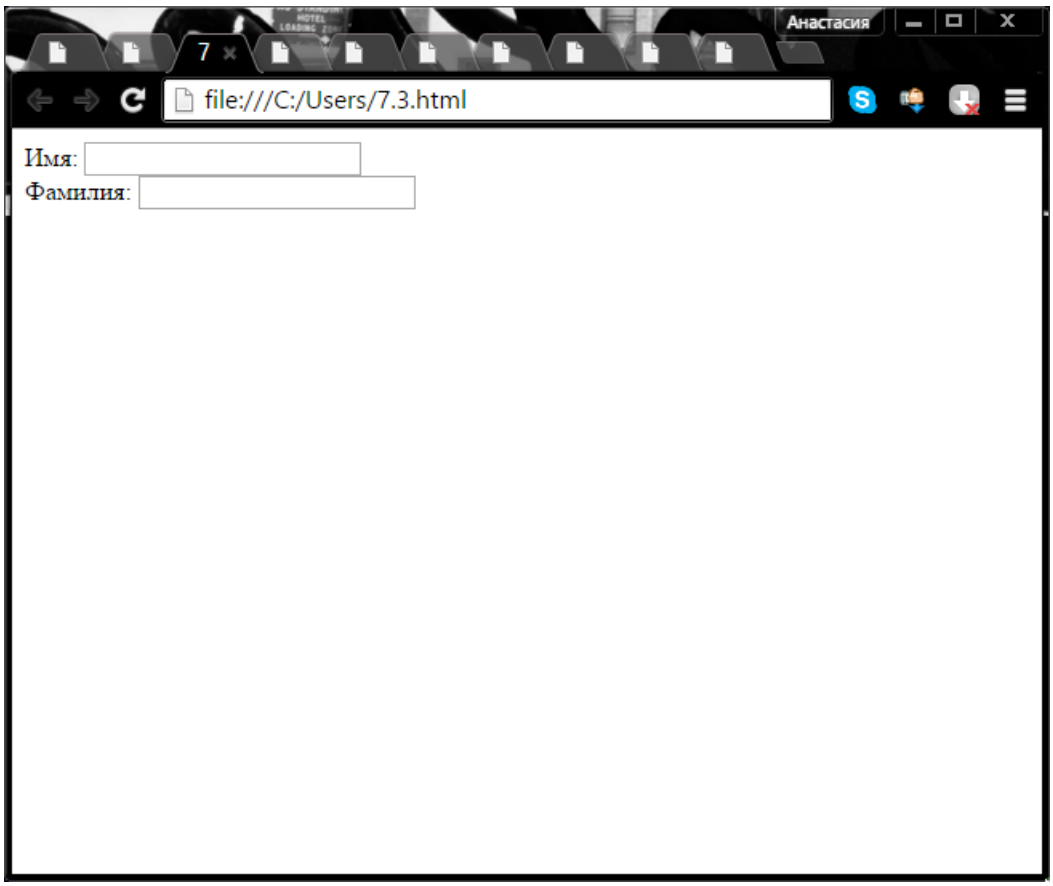


Рис. 7.3 Пример выполнения данного HTML-кода

В большинстве браузеров по умолчанию ширина текстового поля равна 20 символам. Для изменения этого значения используется атрибут size. В следующем примере поле для ввода имени ограничено 10 символами.

```
<html>
<body>
<form>
  Имя:
  <input type="text" name="firstname" size="10">
  <br>
  Фамилия:
  <input type="text" name="lastname">
</form>
</body>
</html>
```

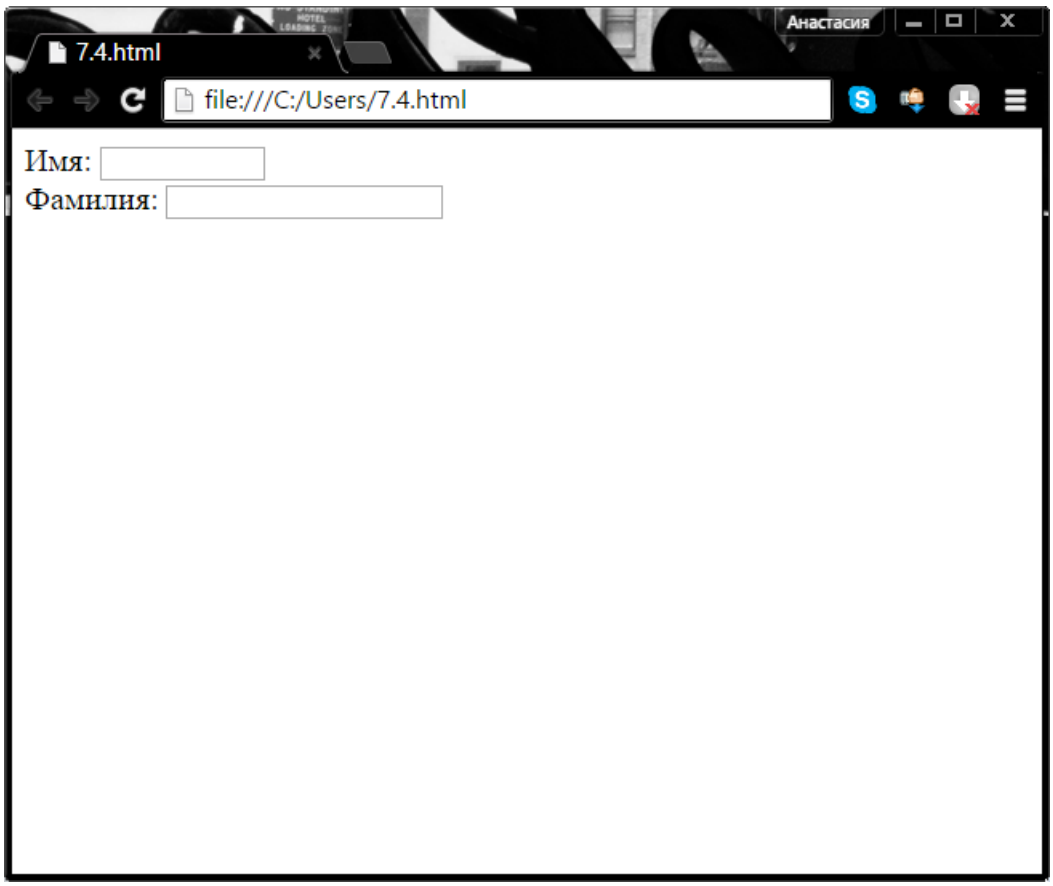


Рис. 7.4 Пример выполнения данного HTML-кода

Здесь необходимо отметить, что атрибут size ограничивает только видимую область ввода данных, а не длину вводимой строки. Для этой цели используется атрибут maxlength. В следующем примере максимальное количество символов для ввода имени ограничено тремя.

```
<html>
<body>
<form>
 Имя:
 <input type="text" name="firstname" maxlength="3">
 <br>
 Фамилия:
 <input type="text" name="lastname">
</form>
</body>
</html>
```

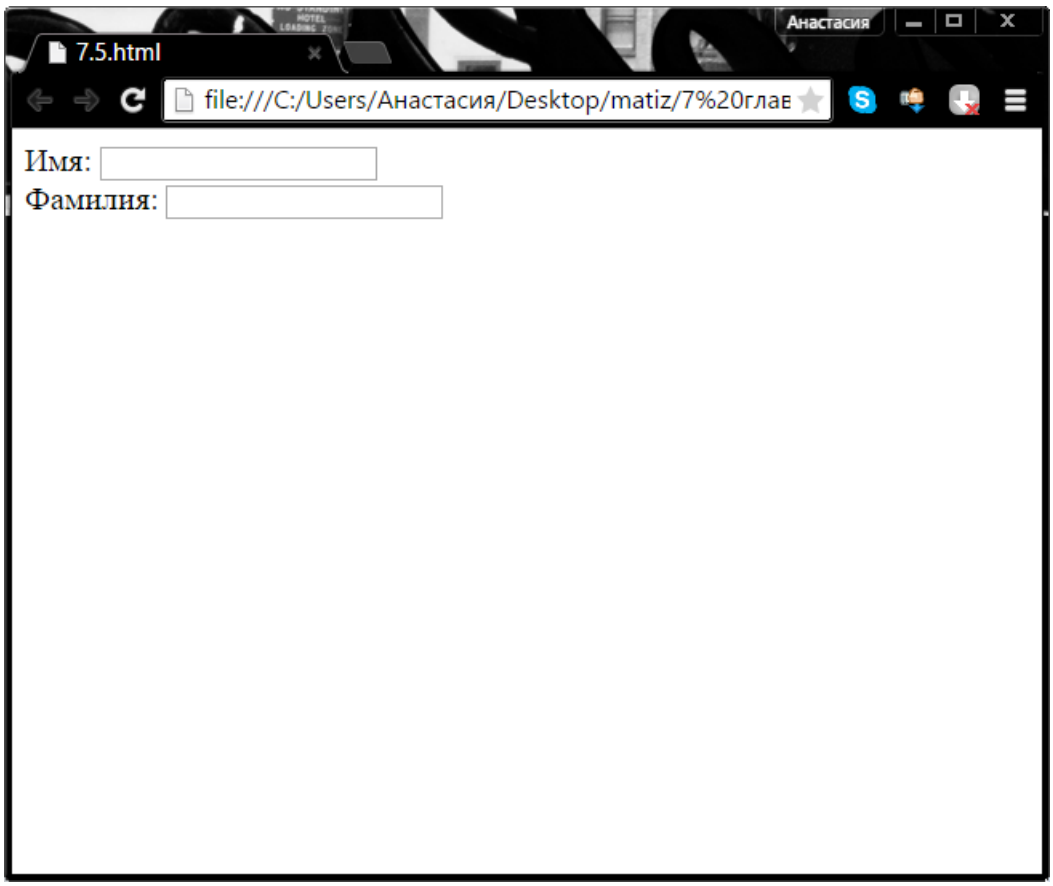


Рис. 7.5 Пример выполнения данного HTML-кода

При вводе в поле имени слова "Елена" форма воспримет только первые три символа имени.

Кроме того, при использовании текстовых полей возможно задать значения по умолчанию. Для этого используется атрибут value. Применение этого атрибута показано на следующем примере:

```
<html>
<body>
<form>
  Имя:
  <input type="text" name="firstname" value="Билл">
  <br>
  Фамилия:
  <input type="text" name="lastname" value="Гейтс">
</form>
</body>
</html>
```

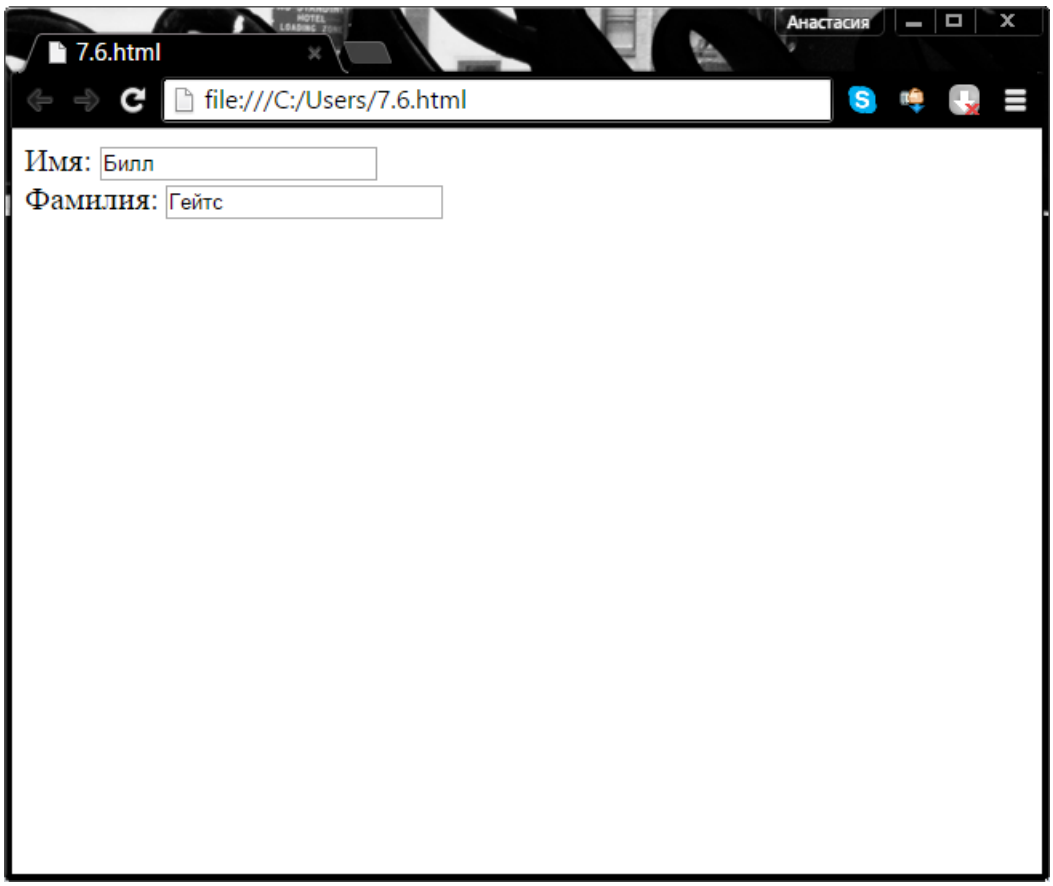


Рис. 7.6 Пример выполнения данного HTML-кода

Необходимо отметить, что применение атрибутов для различных типов полей вывода и управления может отличаться. Так, например, атрибут size для текстовых полей ввода (type="text" или type="password") указывает максимальное количество символов, отображаемых в поле, а для прочих элементов – занимаемый по горизонтали размер в пикселях.

Поле пароля

Поле пароля (type=password) создает защищенное поле ввода, которое дает возможность пользователю, заполняющему форму ввести текст, но в отличие от обычного текстового поля, вводимые данные при отображении на мониторе заменяются звездочками или точками.

Следующий пример демонстрирует простейшую форму для ввода имени пользователя и пароля.

```
<html>
<body>
<form>
 Имя пользователя:
 <input type="text" name="user" value="Елена">
 <br>
 Пароль:
 <input type="password" name="password" value="tktyf">
```

```
</form>
</body>
</html>
```

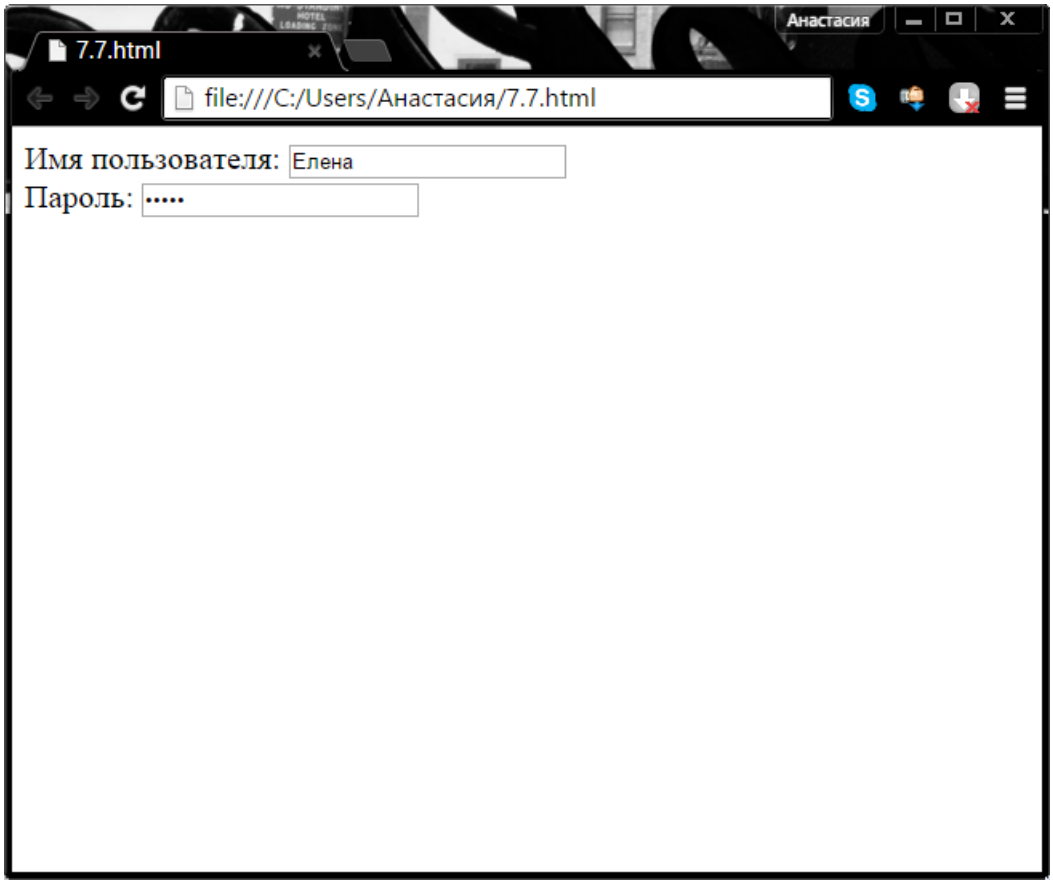


Рис. 7.7 Пример выполнения данного HTML-кода

Необходимо обратить внимание, что хотя значение пароля и задано по умолчанию, при отображении браузер выводит вместо символов звездочки.

Переключатели

Переключатели или радиокнопки (type=radio) определяют поля выбора одного значения из нескольких доступных. Поля этого типа часто используются в диалоговых окнах. Для каждой позиции переключателя создается свой тег <input type=radio>. Группируются переключатели при помощи одинакового имени, задаваемого атрибутом name.

```
<html>
<body>
<form>
  Укажите Ваш пол:
  <br>
  <input type="radio" name="sex" value="male"> мужчина
  <br>
  <input type="radio" name="sex" value="female"> женщина
</form>
</body>
</html>
```

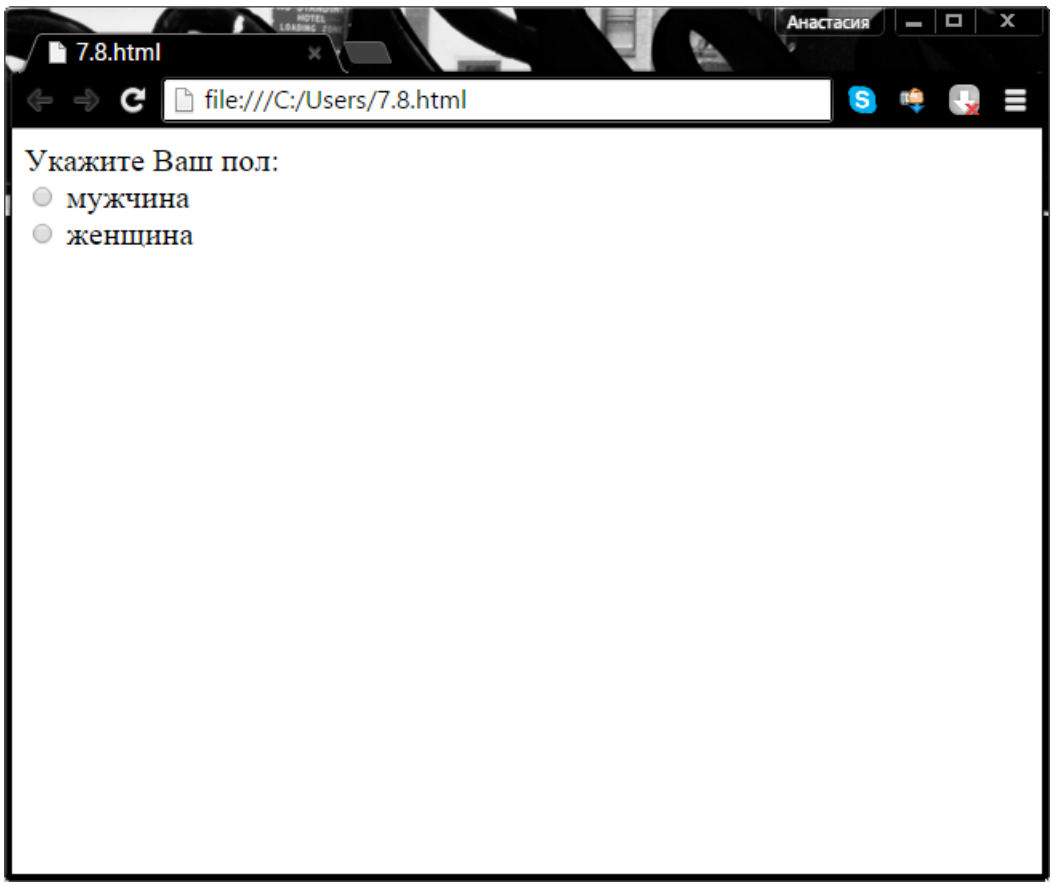


Рис. 7.8 Пример выполнения данного HTML-кода

Необходимо отметить, что в отличие от текстового поля и поля пароля атрибут value задает значение, которое будет передано серверу для дальнейшей обработки в случае выбора данного переключателя. Для выбора по умолчанию одного из возможных значений группы переключателей используется атрибут checked.

```
<html>
<body>
<form>
  Укажите Ваш пол:
  <br>
  <input type="radio" name="sex" value="male" checked="checked"> мужчина
  <br>
  <input type="radio" name="sex" value="female"> женщина
</form>
</body>
</html>
```

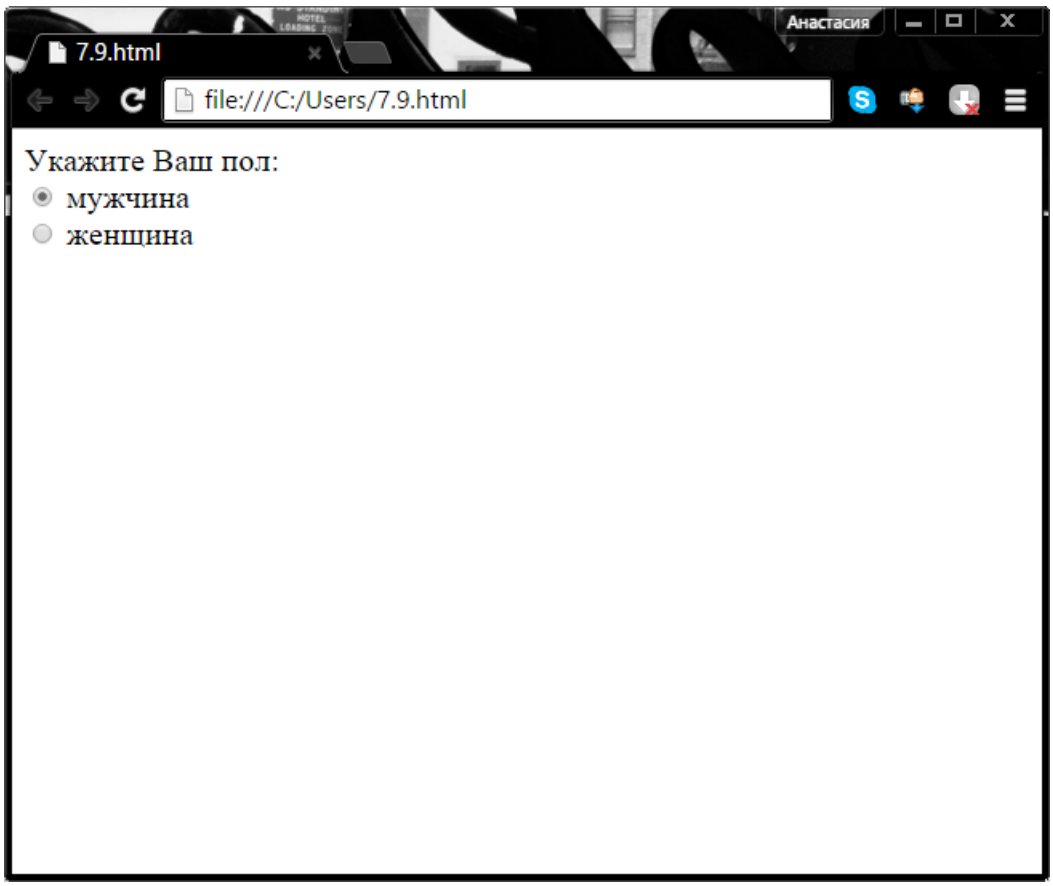


Рис. 7.9 Пример выполнения данного HTML-кода

В данном примере по умолчанию выбран мужской пол.

Флажки

Флажки (type=checkbox) используются, когда необходимо, чтобы пользователь выбрал один или несколько вариантов из ограниченного числа вариантов выбора. Флажки в форме не зависят друг от друга, их можно установить или сбросить в любой комбинации. Для каждого флажка необходимо задать свое уникальное имя при помощи атрибута name. Создание двух флажков с одним именем не вызовет ошибки при отображении формы, но не позволит сценарию обработки на сервере корректно обработать передаваемые с формы данные.

```
<html>
<body>
<form>
 В этом году я собираюсь приобрести:
 <br>
 <input type="checkbox" name="computer">
 Компьютер
 <br>
 <input type="checkbox" name="notebook">
 Ноутбук
```

```
  <br>
  <input type="checkbox" name="printer">
  Принтер
  <br>
  <input type="checkbox" name="scanner">
  Сканер
</form>
</body>
</html>
```

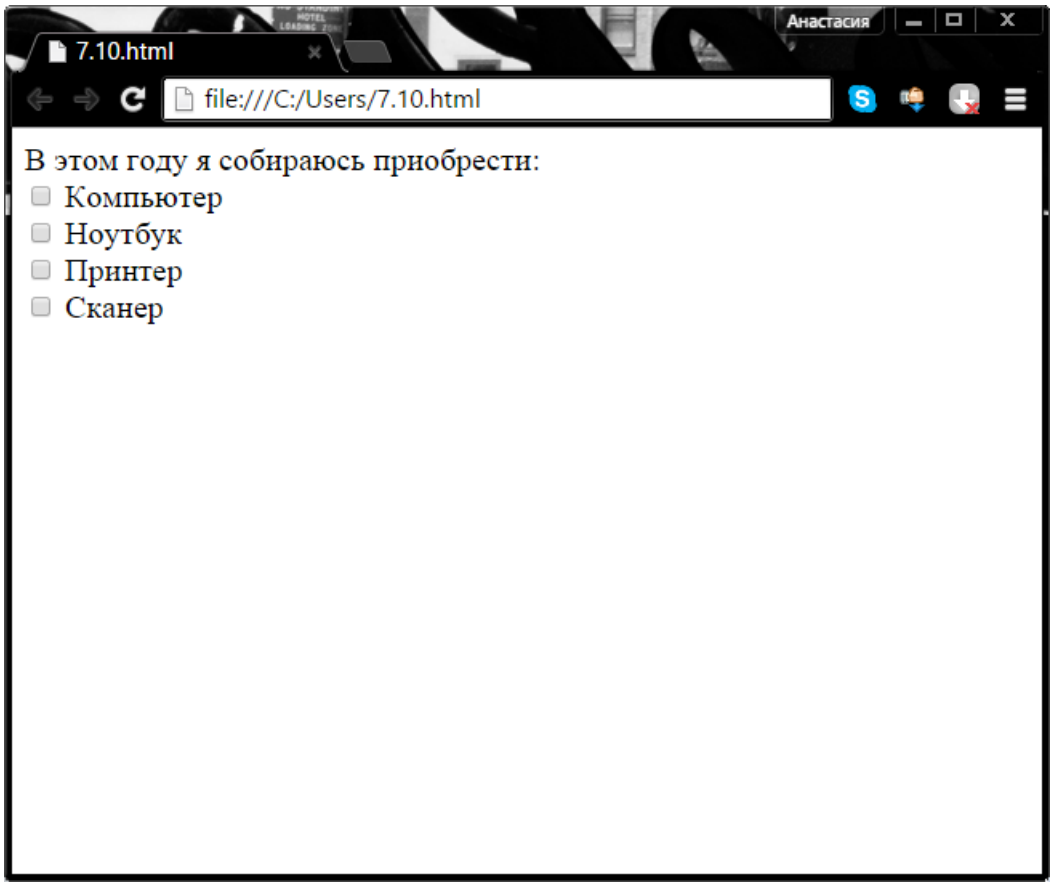


Рис. 7.10 Пример выполнения данного HTML-кода

При помощи атрибута checked можно установить, какие из флажков будут выбраны по умолчанию при загрузке страницы. Отличие от переключателей заключается только в том, что для флажков можно отметить сразу несколько вариантов.

```
<html>
<body>
<form>
  В этом году я собираюсь приобрести:
  <br>
  <input type="checkbox" name="computer" checked="checked">
  Компьютер
  <br>
  <input type="checkbox" name="notebook">
```

```
  Ноутбук
  <br>
  <input type="checkbox" name="printer">
  Принтер
  <br>
  <input type="checkbox" name="scanner" checked="checked">
  Сканер
</form>
</body>
</html>
```

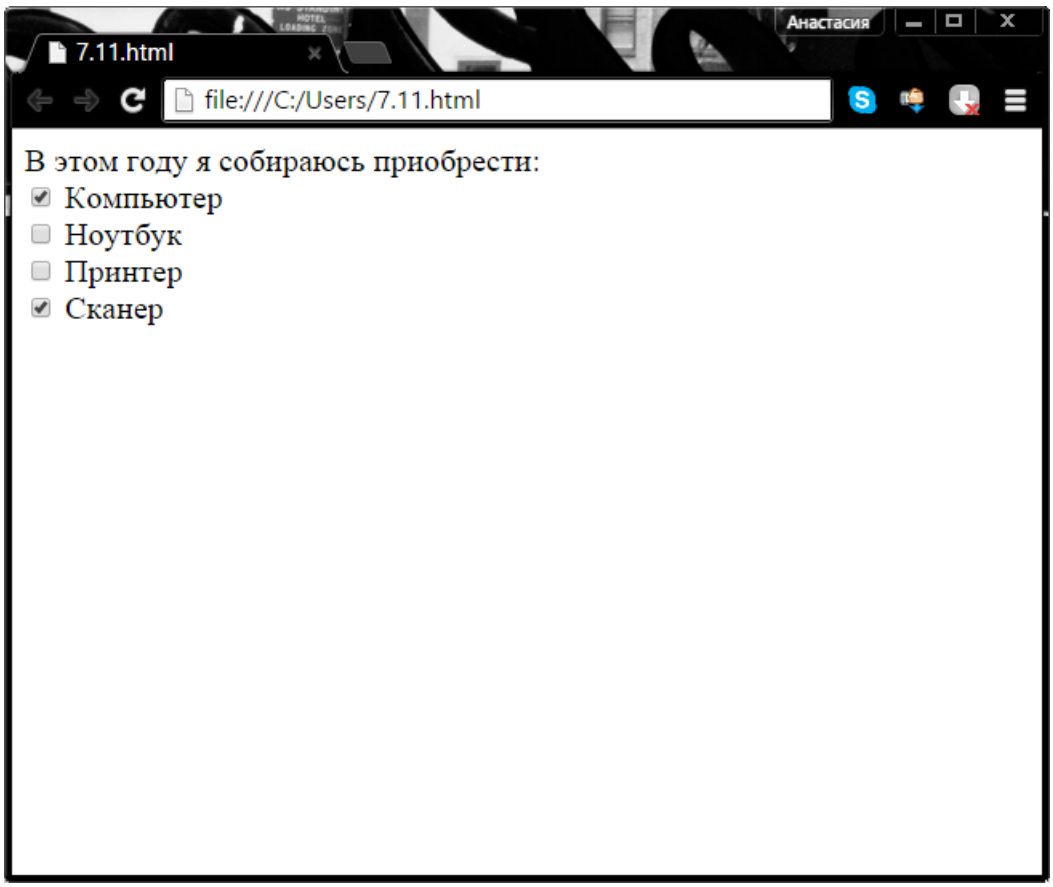


Рис. 7.11 Пример выполнения данного HTML-кода

В данном примере по умолчанию выбраны флажки "Компьютер" и "Сканер":

При отправке данных формы с флажками на сервер выбранным флажкам присваивается значение по умолчанию "on". Как правило, этого достаточно для корректной обработки данных, но в некоторых случаях удобнее задать каждому флажку свое значение при помощи атрибута value.

<input type="checkbox" name="printer" value="Принтер">

Командные кнопки

Командная кнопка отправки (type=submit) используется для выполнения пересылки данных формы на сервер. Метод отправки и адрес

файла, обрабатывающего полученную информацию, задаются в теге <form> при помощи атрибутов method и action. Командная кнопка сброса (type=reset) возвращает форму к исходному состоянию (очищает форму). При этом данные не передаются.

В следующем примере показана форма поиска с двумя кнопками: отправки и сброса.

```
<html>
<body>
 <form name="input" action="html_form_action.asp" method="get">
 Найти:
 <input type="text" name="search" size=25>
 <br>
 <input type="submit">
 <input type="reset">
</form>
</body>
</html>
```

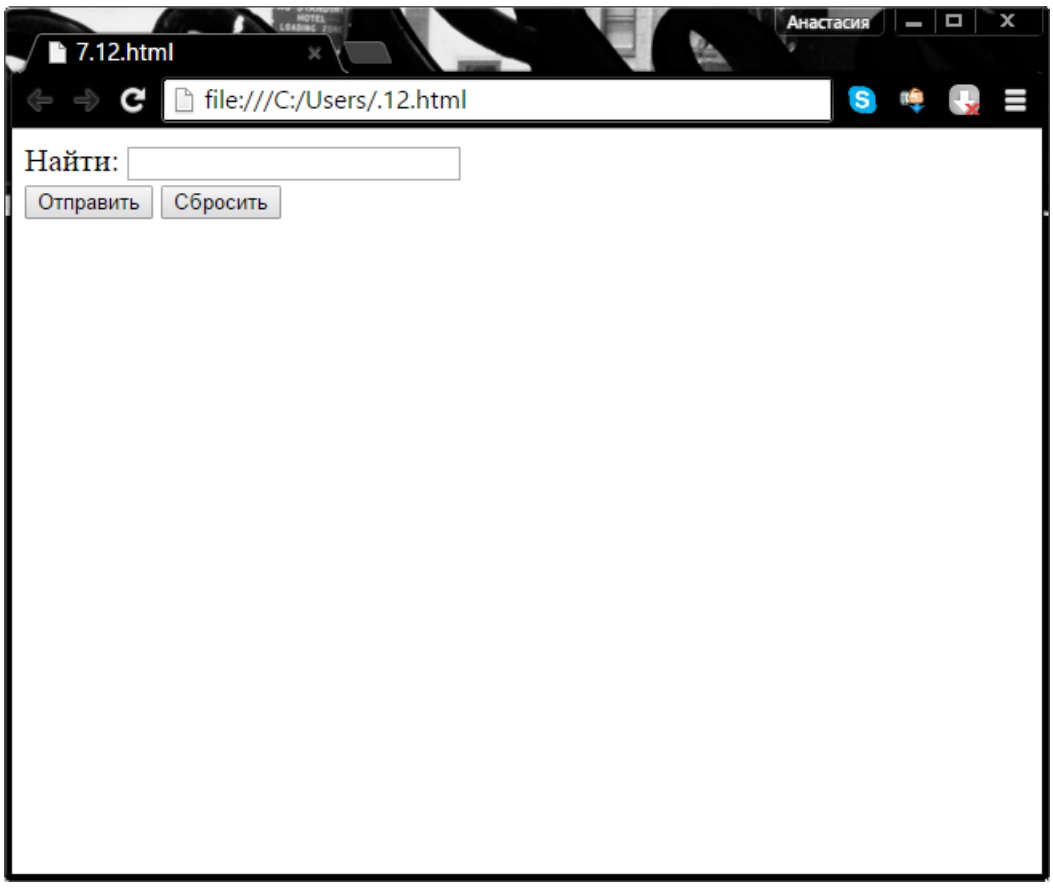


Рис. 7.12 Пример выполнения данного HTML-кода

Если ввести в текстовое поле какие-то символы и нажать кнопку "Подача запроса", то введенная информация будет послана на страницу с именем "html_form_action.asp". При нажатии на кнопку "Сброс" текстовое поле очистится.

Надписи на кнопках "Подача запроса" и "Сброс" установлены по умолчанию. Для их изменения необходимо использовать атрибут value.

Кроме кнопок отправки и сброса существует также возможность добавлять пользовательские кнопки (type=button), которые могут использоваться для выполнения процедур (скриптов) непосредственно на Web-странице.

Поле выбора файла

Поле выбора файла (type=file) создает поле для выбора файла, который будет загружен на сервер вместе с информацией формы. Рядом с полем ввода отображается командная кнопка "Обзор…", открывающая стандартное диалоговое окно выбора файла.

```
<html>
<body>
<form>
 Прикрепить файл:
 <br>
 <input type="file" size="50">
</form>
</body>
</html>
```

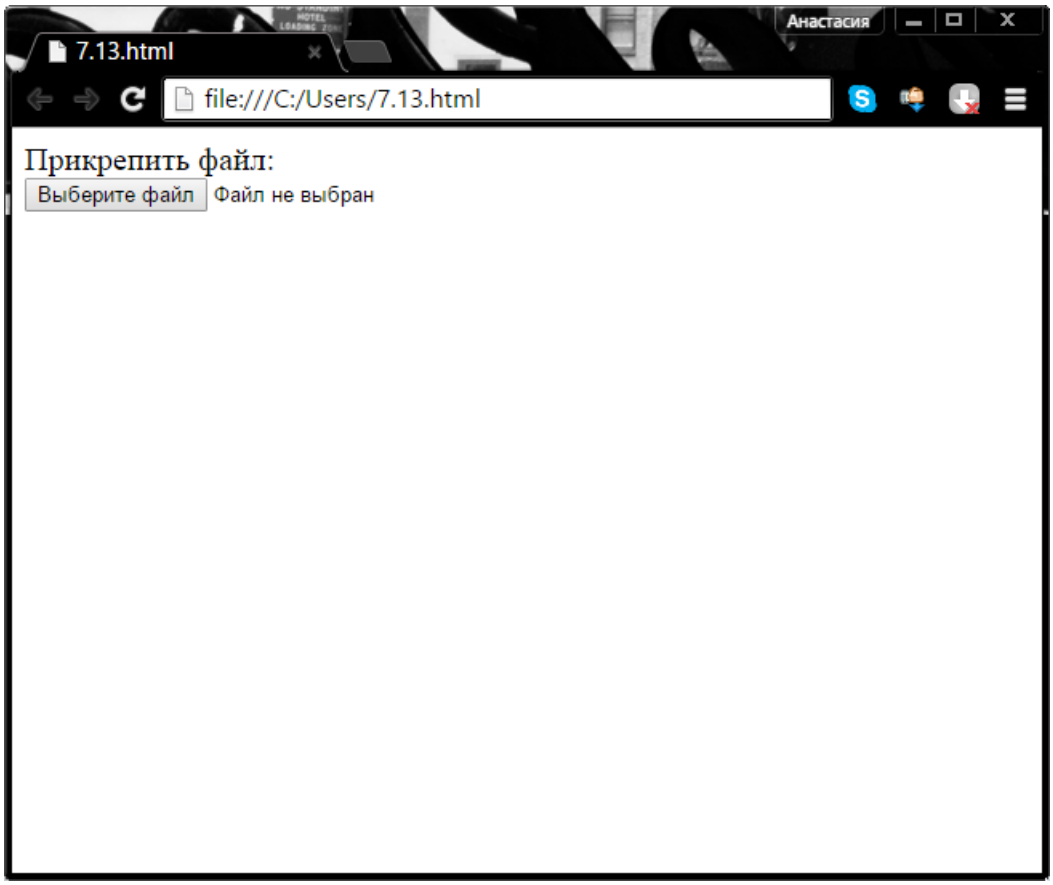


Рис. 7.13 Пример выполнения данного HTML-кода

Для поля выбора файла по аналогии с текстовым полем можно использовать атрибуты size, maxlength, value.

Списки выбора

Списки выбора бывают двух типов: раскрывающиеся списки (выпадающие меню) и списки с множественным выбором. Независимо от типов списков описываются они одинаково с помощью пары тегов <select> </select>. Отдельные элементы списка задаются с использованием тега <option>. Тип списка определяется при помощи атрибутов тега <select>.

Атрибут name задает имя поля для отправки выбранных пунктов списка на сервер. Атрибут multiple разрешает множественный выбор. Атрибут size определяет, какое количество пунктов списка будет одновременно отображено на экране. При этом, если атрибут multiple не задан и size=1, то на экране отображается раскрывающийся список, если же задан атрибут Multiple или значение size больше 1, то список отображается развернутым

.

Раскрывающийся список выбора

Раскрывающиеся списки выбора по своему назначению соответствуют переключателям, в то же время их использование предпочтительно в тех случаях, когда количество вариантов выбора достаточно велико. Как правило, при выборе более чем из трех вариантов желательно вместо переключателей использовать раскрывающиеся списки.

В следующем примере создан простой раскрывающийся список выбора ноутбука по производителю.

```
<html>
<body>
<form>
  Выбор ноутбука по производителю:
  <select name="notebook">
    <option value="acer">Acer
    <option value="asus">Asus
    <option value="compaq">Compaq
    <option value="hp">HP
    <option value="sony">Sony
    <option value="toshiba">Toshiba
  </select>
</form>
</body>
</html>
```

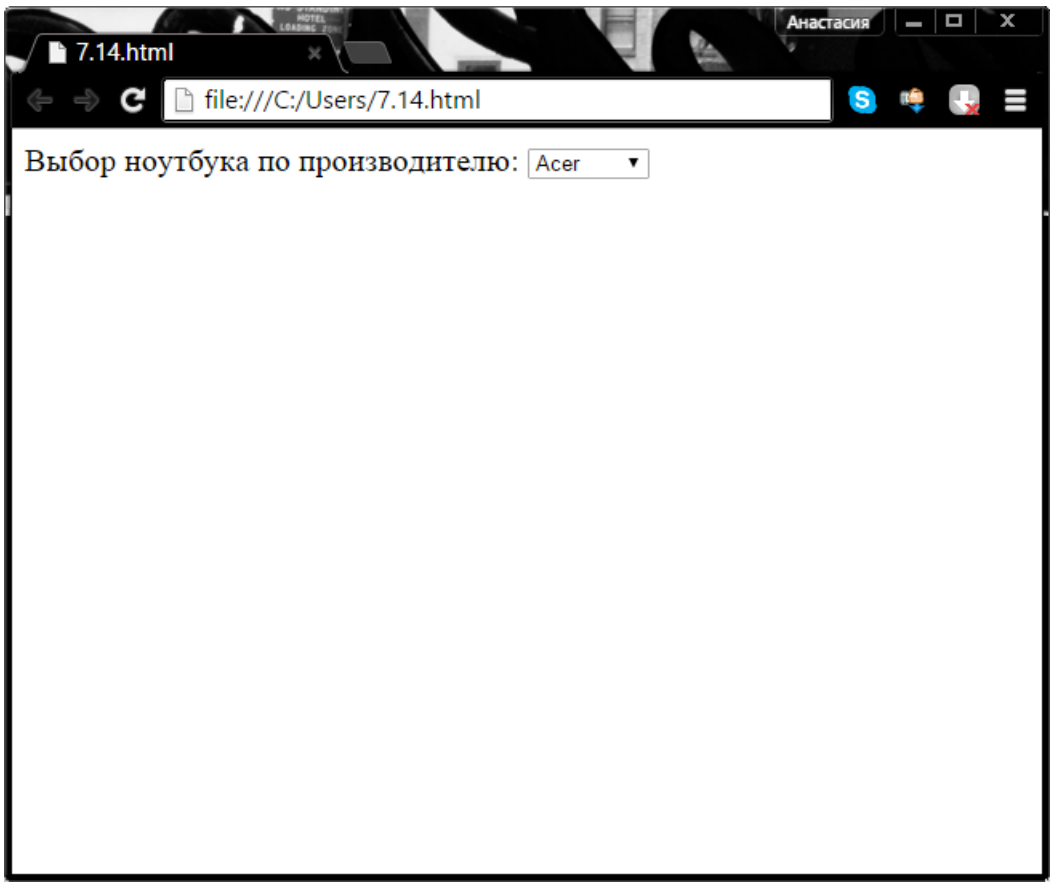


Рис. 7.14 Пример выполнения данного HTML-кода

Необходимо заметить, что по умолчанию выбирается первое значение из списка. При помощи атрибута selected тега <option> это значение можно изменить. Следующий пример показывает раскрывающийся список выбора размера экрана ноутбука с предварительно установленным значением "15.4".

```
<html>
<body>
<form>
  Выбор размера экрана ноутбука
  <select name="tft">
    <option value="tft-12">12"
    <option value="tft-13">13"
    <option value="tft-14">14"
    <option value="tft-15">15"
    <option value="tft-15-4" selected="selected">15.4"
    <option value="tft-17">17"
  </select>
</form>
</body>
</html>
```

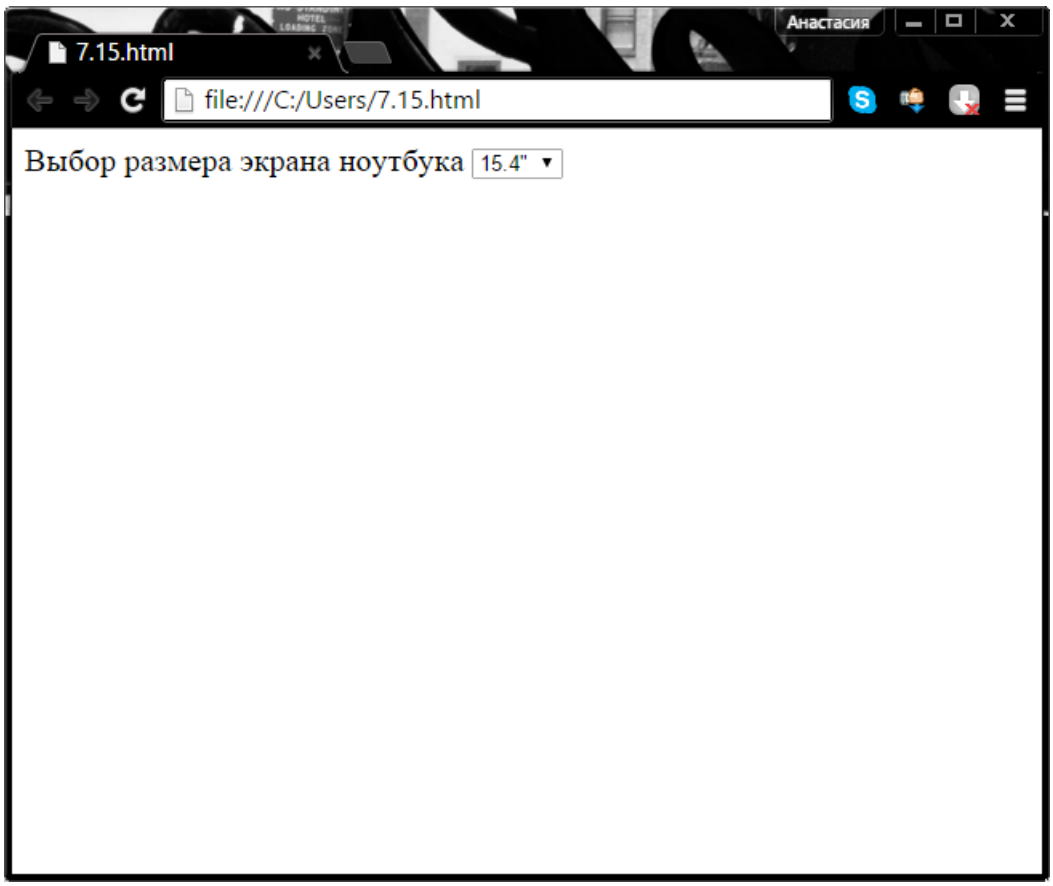


Рис. 7.15 Пример выполнения данного HTML-кода

Развернутый список выбора

Развернутые списки используются, как правило, при необходимости множественного выбора значений. В следующем примере раскрывающийся список выбора ноутбука по производителю преобразован в развернутый список с возможностью множественного выбора.

```
<html>
<body>
<form>
 Выбор ноутбука по производителю:
 <select name="notebook" multiple>
   <option value="acer">Acer
   <option value="asus">Asus
   <option value="compaq">Compaq
   <option value="hp">HP
   <option value="sony">Sony
   <option value="toshiba">Toshiba
 </select>
</form>
</body>
</html>
```

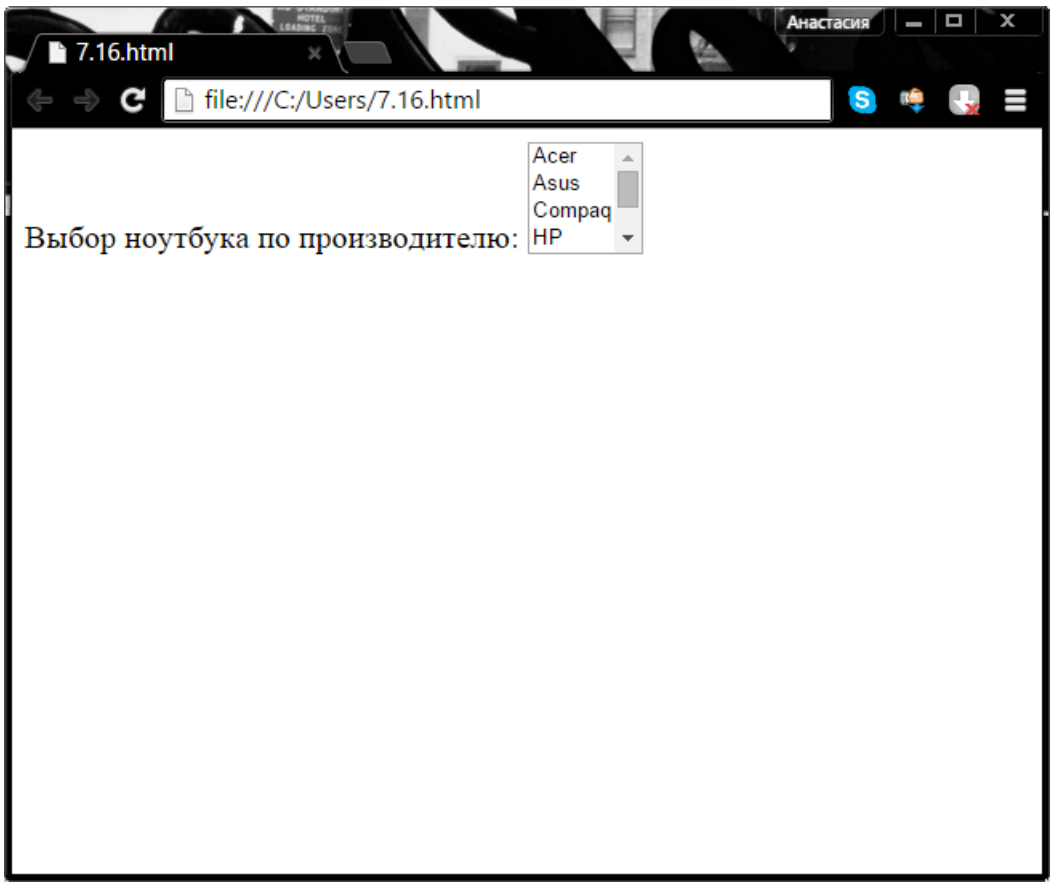


Рис. 7.16 Пример выполнения данного HTML-кода

Необходимо отметить, что для списков с множественным выбором значение атрибута size по умолчанию устанавливается равным 4, кроме того, ни одно из значений не является выбранным по умолчанию. В следующем примере показана форма с развернутым списком высотой в 6 пунктов с предварительно заданными значениями "Compaq" и "Sony".

```
<html>
<body>
<form>
  Выбор ноутбука по производителю:
  <select name="notebook" multiple size=6>
    <option value="acer">Acer
    <option value="asus">Asus
    <option value="compaq" selected="selected">Compaq
    <option value="hp">HP
    <option value="sony" selected="selected">Sony
    <option value="toshiba">Toshiba
  </select>
</form>
</body>
</html>
```

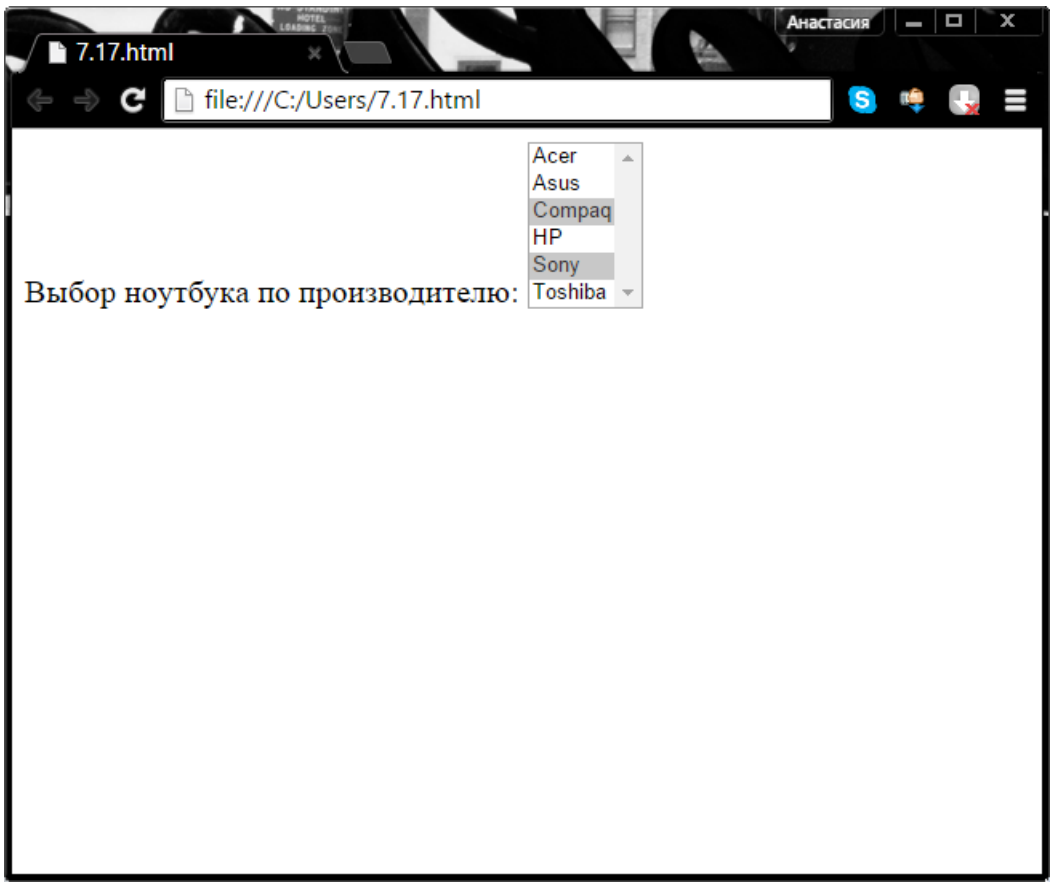


Рис. 7.17 Пример выполнения данного HTML-кода

Текстовая область

В отличие от текстового поля <input type=text> текстовая область позволяет вводить многострочный текст большого объема. Такие области часто используются при вводе сообщений, комментариев.

Текстовая область описывается при помощи тегов <textarea> </textarea>, между которыми можно разместить предварительно отформатированный стандартный текст. Атрибуты cols и rows задают размер видимой области текстового поля.

В следующем примере создана текстовая область с предварительно введенным текстом.

```
<html>
<body>
  <textarea rows="7" cols="30">
```

В данном примере мы создали текстовую область с шириной в 30 символов и высотой в 7 строк.

Заданное значение высоты не ограничивает общий объем вводимого текста, а влияет только на размер отображаемой на экране текстовой области.

Для просмотра всего текста необходимо воспользоваться полосой прокрутки.

```
  </textarea>
</body>
```

</html>

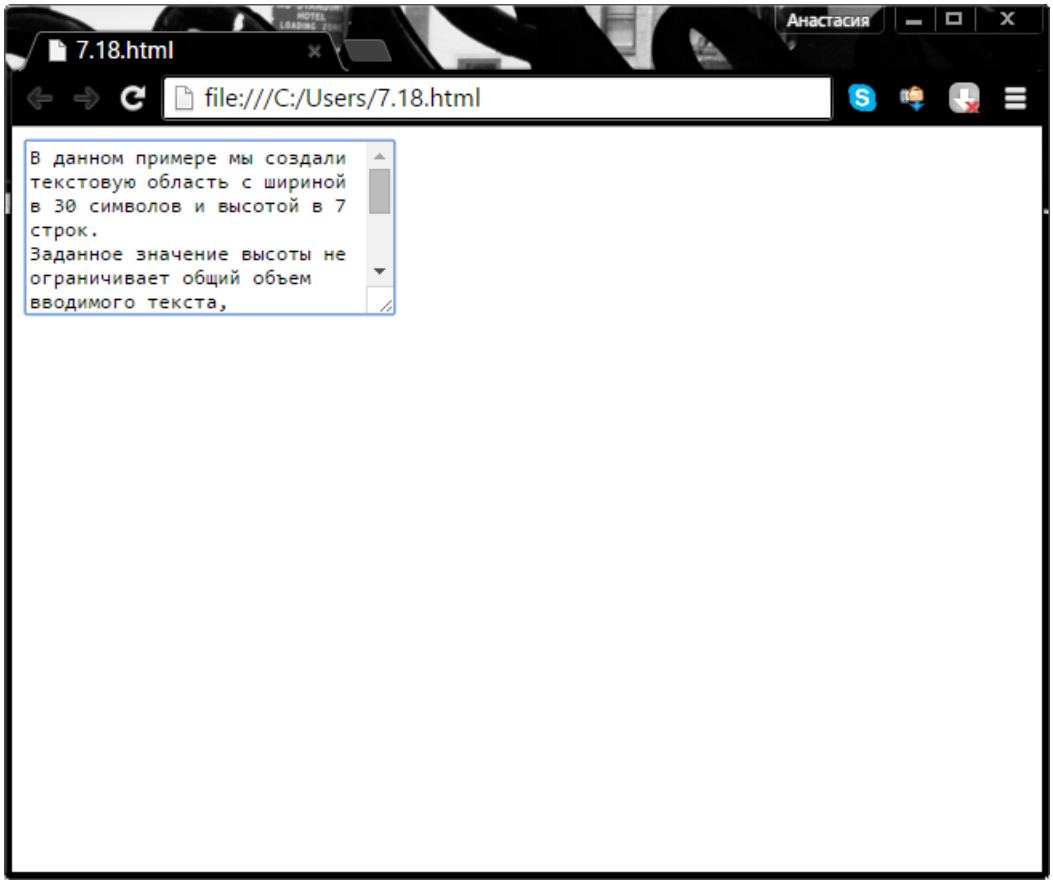


Рис. 7.18 Пример выполнения данного HTML-кода

Внешний вид форм

Существуют различные способы оформления внешнего вида форм. Один из них основан на использовании пары тегов <fieldset> </fieldset>, которые позволяют объединить тематически близкие и рядом расположенные поля формы в группу и выделить ее визуально. Другой способ основан на применении стандартных средств форматирования HTML, включая цветовое и шрифтовое оформление, графику, таблицы и т.п.

Группировка полей формы

Теги <fieldset> </fieldset> объединяют поля формы в группы, выделяют их визуально, упрощают навигацию в форме с помощью клавиши [Tab] (в первую очередь обходятся поля в пределах группы). Для того чтобы объединить несколько элементов ввода или управления в группу достаточно поместить их внутри тегов <fieldset> </fieldset>. Закрывающий тег </fieldset> обязателен.

В следующем примере создается форма, содержащая поля для ввода имени и фамилии, объединенные в группу, а также кнопку отправки

<html>
<body>

```
<form name="input" action="html_form_action.asp" method="get">
  <fieldset>
    Имя:
    <input type="text" name="firstname" value="Билл">
    <br>
    Фамилия:
    <input type="text" name="lastname" value="Гейтс">
  </fieldset>
  <input type="submit" value="отправить">
</form>
</body>
</html>
```

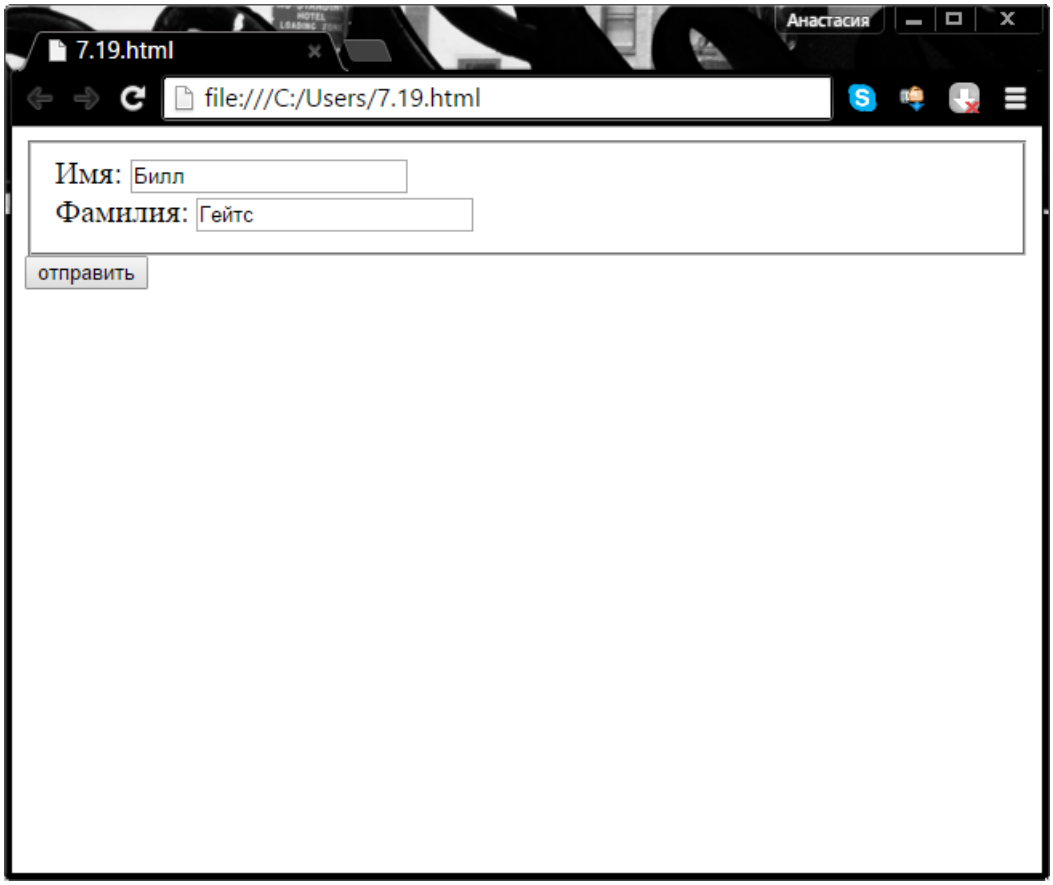


Рис. 7.19 Пример выполнения данного HTML-кода

При помощи пары тегов <legend> </legend> образованной группе полей можно присвоить имя. В следующем примере форма ввода дополнена полями для ввода информации о месте работы и занимаемой должности, объединенными во вторую группу. Каждой группе полей присвоен свой заголовок.

```
<html>
<body>
<form name="input" action="html_form_action.asp" method="get">
  <fieldset>
    <legend>Личные данные</legend>
```

```
Имя:
<input type="text" name="firstname" value="Билл">
<br>
Фамилия:
<input type="text" name="lastname" value="Гейтс">
<br>
Пол:
<input type="radio" name="Sex" value="Male" checked="checked">
Мужской
<input type="radio" name="Sex" value="Female"> Женский
</fieldset>
<fieldset>
<legend>Данные о работе</legend>
Место работы:
<input type="text" name="work" value="Microsoft">
<br>
Должность:
<input type="text" name="status" value="Президент">
</fieldset>
<input type="submit" value="отправить">
</form>
</body>
</html>
```

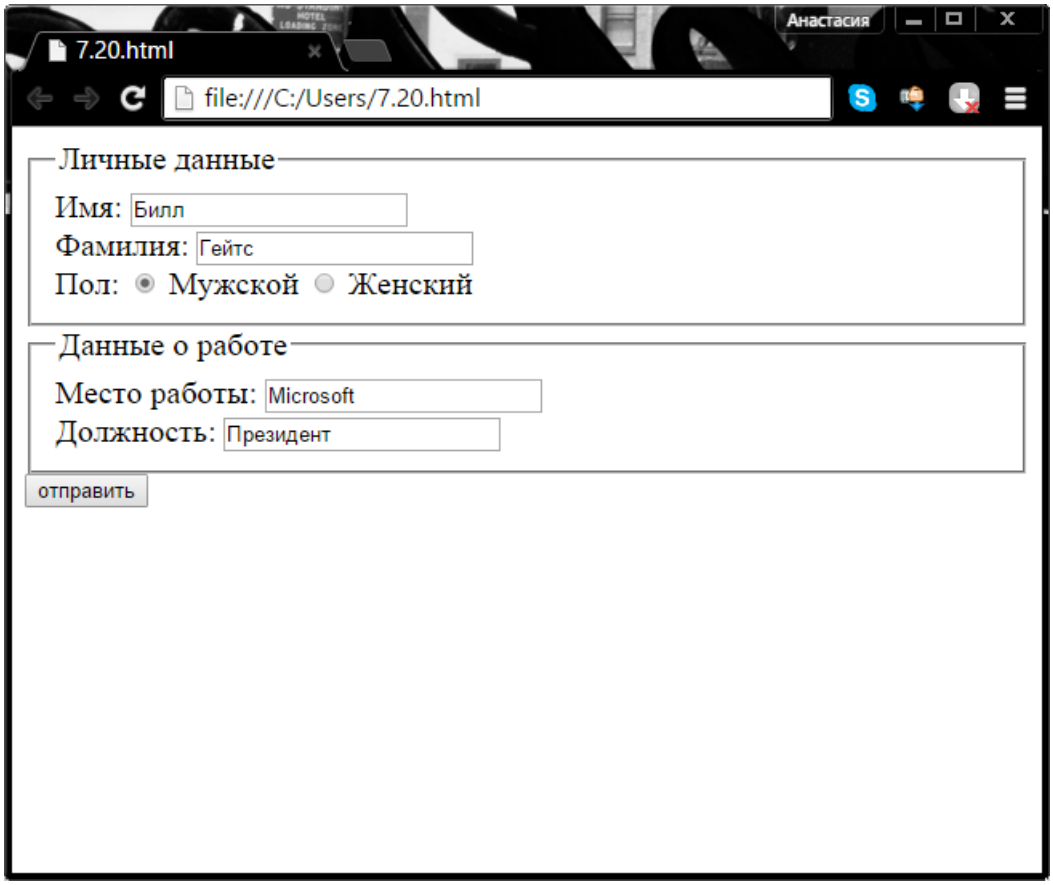


Рис. 7.20 Пример выполнения данного HTML-кода

Оформление форм стандартными средствами HTML

Несмотря на простоту использования тегов группировки <fieldset>, их возможности по оформлению форм весьма ограничены. Чаще всего для оформления внешнего вида форм используются стандартные средства HTML. В следующем примере показано, как с помощью таблицы поля формы выровнены друг под другом, а заголовок формы и кнопка отправки размещены по центру формы. Сама форма визуально выделена желтым цветом.

```
<html>
<body>
<form name="input" action="html_form_action.asp" method="get">
  <table bgcolor="#FFFF99">
    <tr>
      <th colspan="2" align="center">Личные данные:</th>
    </tr>
    <tr>
      <td align="right">Имя:</td>
      <td align="left"><input type="text" name="firstname" value="Билл"></td>
    </tr>
    <tr>
      <td align="right">Фамилия:</td>
      <td align="left"><input type="text" name="lastname" value="Гейтс"></td>
    </tr>
    <tr>
      <td colspan="2" align="center"><input type="submit" value="отправить"></td>
    </tr>
  </table>
</form>
</body>
</html>
```

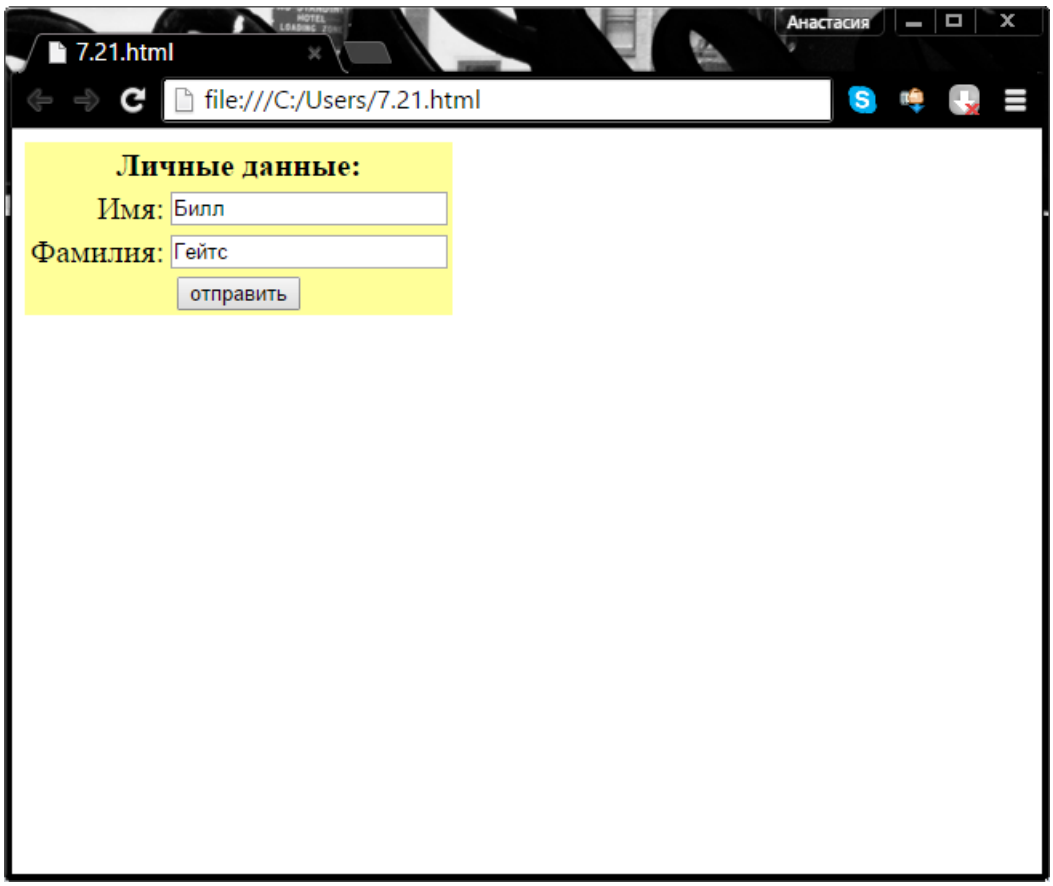


Рис. 7.21 Пример выполнения данного HTML-кода

Примеры форм

Регистрация в системе

```
<html>
<body>
<form name="input" action="html_form_action.asp" method="post">
  <table bgcolor="#9999FF">
    <tr>
      <th colspan="2" align="center">Регистрация в системе</th>
    </tr>
    <tr>
      <td align="right">Имя:</td>
      <td align="left"><input type="text" name="name"></td>
    </tr>
    <tr>
      <td align="right">Пароль:</td>
      <td align="left"><input type="password" name="password"></td>
    </tr>
    <tr>
      <td colspan="2" align="center"><input type="checkbox">Запомнить пароль</td>
    </tr>
```

```
        <tr>
          <td colspan="2" align="center"><input type="submit"
value="Вход"></td>
        </tr>
      </table>
    </form>
    </body>
    </html>
```

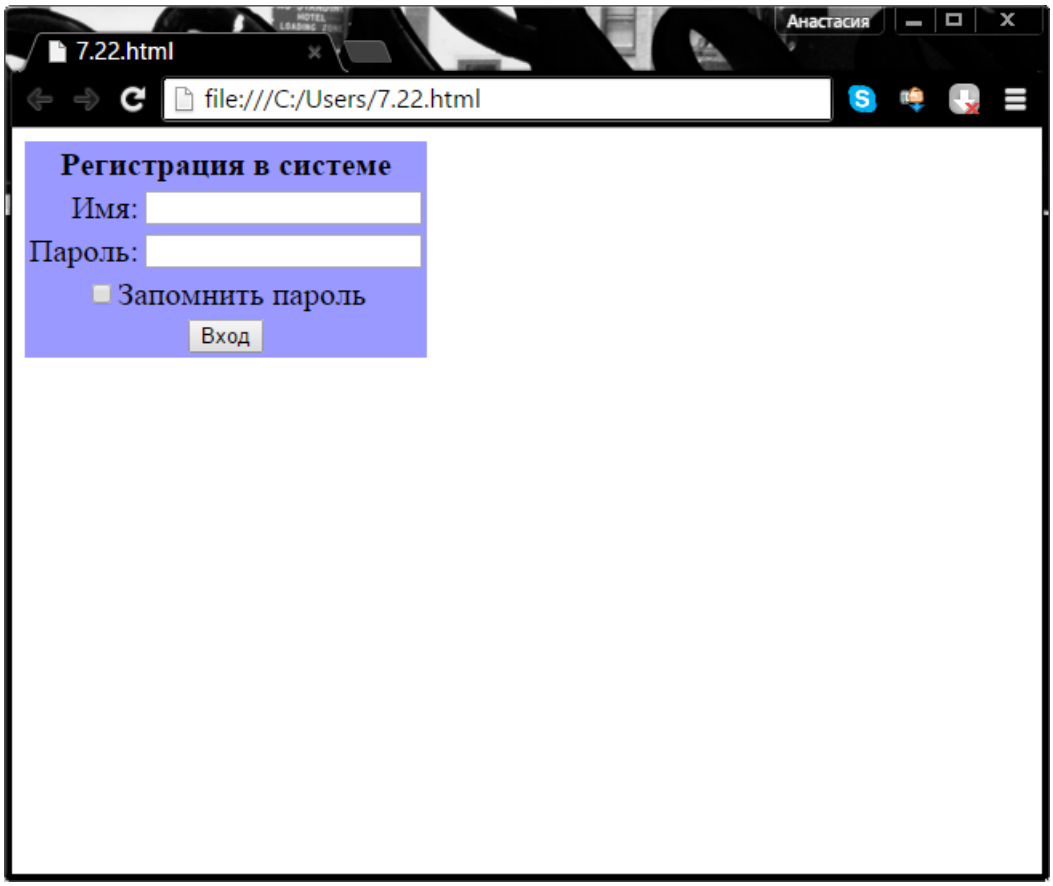


Рис. 7.22 Пример выполнения данного HTML-кода

В этом примере представлена типовая форма входа в систему авторизованного пользователя. Форма состоит из поля ввода имени, поля ввода пароля, флажка сохранения пароля и кнопки входа.

Необходимо отметить, что при работе с паролями следует использовать только метод post. При использовании метода get пароль будет передаваться для обработки в открытом виде. Для проверки этого достаточно в данном примере заменить метод post на get, запустить пример, заполнить поля (например, имя пользователя: Bill, пароль superstar) и нажать кнопку "Вход". В этом случае в адресной строке браузера появится адрес: {ваш рабочий каталог} /html_form_action.asp?name=Bill&password=superstar.

Примечание: необходимо, чтобы в Вашем рабочем каталоге находился файл с именем "html_form_action.asp" (это может быть даже пустой файл).

Запрос документа

```
<html>
```

```
<body>

<form name="input" action="html_form_action.asp" method="get">
 <table bgcolor="#dddddd">
  <tr>
   <th colspan="2" align="center">Запрос документа</th>
  </tr>
  <tr>
   <td align="right">Тип документа:</td>
   <td align="left">
    <select name="do_type">
     <option value="mail">Письмо
     <option value="order">Приказ
     <option value="message">Служебная записка
    </select>
   </td>
  </tr>
  <tr>
   <td align="right">Дата регистрации с</td>
   <td align="left"><input type="text" name="date_begin" size="21"
value="дд.мм.гггг"></td>
  </tr>
  <tr>
   <td align="right">по</td>
   <td align="left"><input type="text" name="date_end" size="21"
value="дд.мм.гггг"></td>
  </tr>
  <tr>
   <td align="right">Регистрационный номер:</td>
   <td align="left"><input type="text" name="number" size="21"></td>
  </tr>
  <tr>
   <td colspan="2"><hr></td>
  </tr>
  <tr>
   <td align="right" valign="top">Ключевые слова:</td>
   <td align="left">
    <textarea cols="17" rows="5">
    </textarea>
   </td>
  </tr>
  <tr>
   <td colspan="2"><hr></td>
  </tr>
  <tr>
```

```
      <td colspan="2" align="center">
       <input type="submit" value="Отправить">
       <input type="reset">
      </td>
     </tr>
   </table>
</form>

</body>
</html>
```

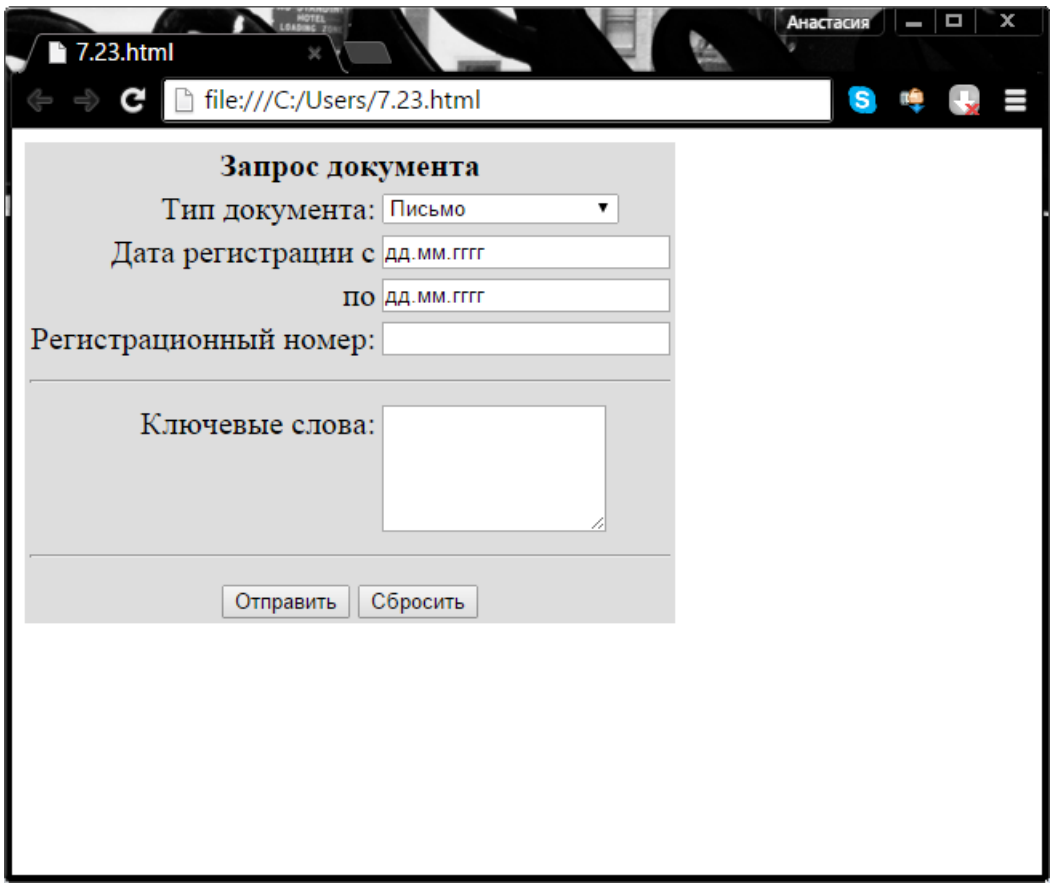


Рис. 7.23 Пример выполнения данного HTML-кода

Это достаточно типовая форма, содержащая поля ввода текста, текстовую область и выпадающий список. Для пояснения формата ввода в полях даты регистрации по умолчанию показан образец заполнения. Но при этом необходимо отметить, что форма не проверяет корректность введенной информации.

Регистрация нового пользователя

```
<html>
<body>

<form name="input" action="html_form_action.asp" method="post">
 <table bgcolor="#dddddd">
```

```
<tr>
 <th colspan="2" align="center">Регистрация нового пользователя</th>
</tr>
<tr>
 <td colspan="2" align="center">Обязательные поля выделены красным цветом<br><br></td>
</tr>
<tr>
 <td align="right"><font color="red">Ник (псевдоним):</td>
 <td align="left"><input type="text" name="nick" size="27"></td>
</tr>
<tr>
 <td align="right"><font color="red">E-mail</td>
 <td align="left"><input type="text" name="e_mail" size="27"></td>
</tr>
<tr>
 <td align="right"><font color="red">Пароль (не менее 6 сим.):</td>
 <td align="left"><input type="password" name="password" size="27"></td>
</tr>
<tr>
 <td align="right"><font color="red">Подтвердите пароль:</td>
 <td align="left"><input type="password" name="f_password" size="27"></td>
</tr>
<tr>
 <td colspan="2"><hr></td>
</tr>
<tr>
 <td align="right">Фамилия</td>
 <td align="left"><input type="text" name="name_f" size="27"></td>
</tr>
<tr>
 <td align="right">Имя</td>
 <td align="left"><input type="text" name="name_i" size="27"></td>
</tr>
<tr>
 <td align="right">Отчество</td>
 <td align="left"><input type="text" name="name_o" size="27"></td>
</tr>
<tr>
 <td align="right">Дата рождения:</td>
 <td align="left">
  <select name="day">
```

```
  <option>1
  <option>2
  <option>3
  <option>4
  <option>5
  <option>6
  <option>7
  <option>8
  <option>9
  <option>10
  <option>11
  <option>12
  <option>13
  <option>14
  <option>15
  <option>16
  <option>17
  <option>18
  <option>19
  <option>20
  <option>21
  <option>22
  <option>23
  <option>24
  <option>25
  <option>26
  <option>27
  <option>28
  <option>29
  <option>30
  <option>31
</select>
<select name="month">
  <option>января
  <option>февраля
  <option>марта
  <option>апреля
  <option>мая
  <option>июня
  <option>июля
  <option>августа
  <option>сентября
  <option>октября
  <option>ноября
  <option>декабря
```

```
        </select>
        <select name="year">
          <option>1980
          <option>1981
          <option>1982
          <option>1983
          <option>1984
          <option>1985
          <option>1986
          <option>1987
          <option>1988
          <option>1989
          <option>1990
        </select>
      </td>
    </tr>
    <tr>
      <td align="right">Пол:</td>
      <td align="left">
        <input type="radio" name="sex" value="male">Мужской
        <input type="radio" name="sex" value="female">Женский
      </td>
    </tr>
    <tr>
      <td align="right">Город проживания:</td>
      <td align="left"><input type="text" name="state" size="27"></td>
    </tr>
    <tr>
      <td align="right" valign="top">Образование:</td>
      <td align="left">
        <input type="radio" name="edu" value="h">Высшее<br>
        <input type="radio" name="edu" value="hh">Незаконченное
высшее<br>
        <input type="radio" name="edu" value="ss">Среднее
специальное<br>
        <input type="radio" name="edu" value="s">Среднее
      </td>
    </tr>
    <tr>
      <td colspan="2"><hr></td>
    </tr>
    <tr>
      <td colspan="2" align="center">
        <input type="submit" value="Зарегистрироваться">
        <input type="reset" value="Очистить форму">
```

```
    </td>
   </tr>
  </table>
</form>

</body>
</html>
```

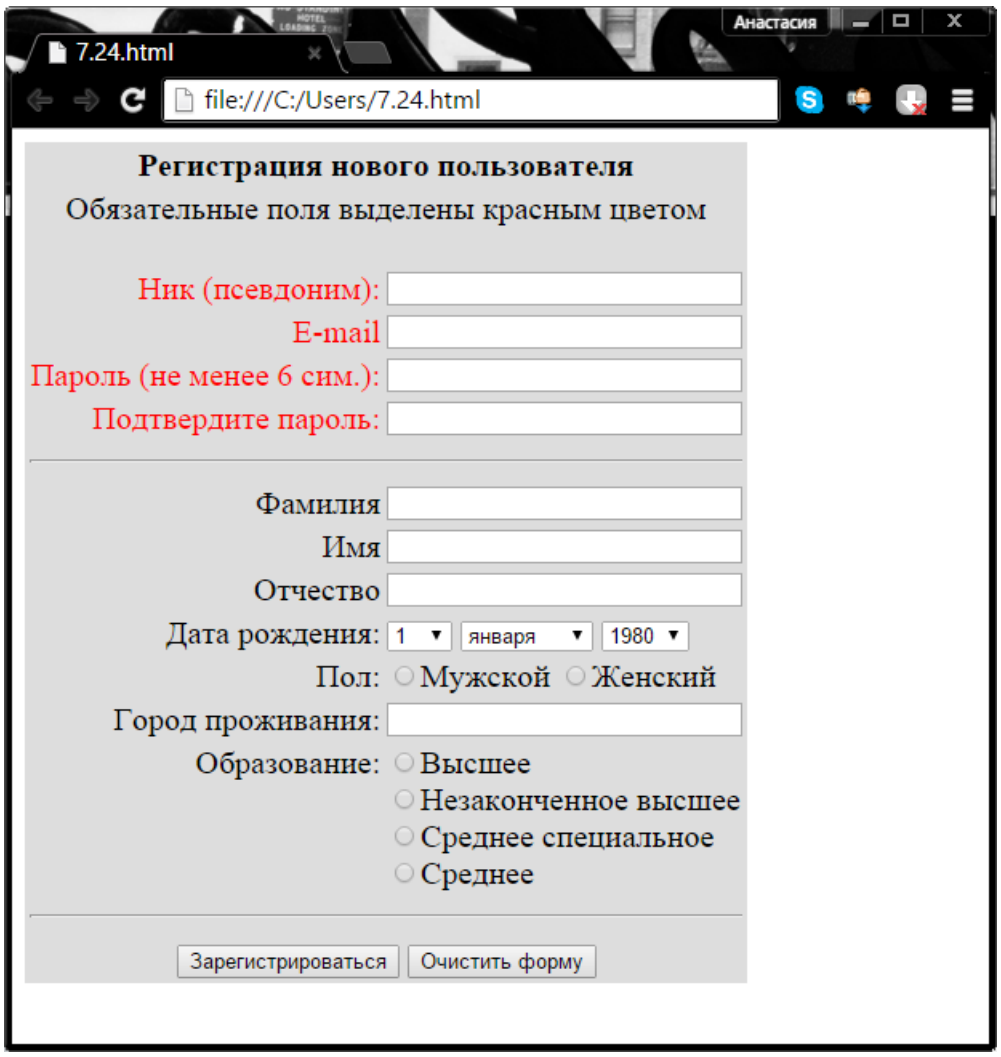


Рис. 7.24 Пример выполнения данного HTML-кода

В данном примере представлена форма регистрации нового пользователя. Отличительной особенностью данной формы является организация ввода даты рождения, построенная с использованием трех выпадающих списков. Хотя реализация такого ввода дат на первый взгляд более трудоемка в процессе создания Web-страницы (для сравнения, в предыдущем примере для организации поля ввода даты потребовалась всего одна строка), но она оправдывает себя за счет сведения к минимуму ошибок при вводе информации. Такая схема организации ввода широко используется при вводе и других данных, таких как, например, страна и город проживания (в данном примере не реализовано).

Следует обратить внимание и на присутствие в форме двух групп переключателей, которые отвечают за ввод пола и образования пользователя.

Теги форм

Тег	**Описание**
<form>	Определяет форму для ввода пользователя
<input>	Определяет поле ввода
<textarea>	Определяет текстовую область (элемент управления для ввода многострочного текста)
<label>	Определяет метку для элемента управления
<fieldset>	Определяет набор полей
<legend>	Определяет заглавие для набора полей
<select>	Определяет список выбора (раскрывающееся поле)
<optgroup>	Определяет группу вариантов выбора
<option>	Определяет вариант в раскрывающемся поле
<button>	Определяет кнопку
<isindex>	Не рекомендуется. Используйте вместо этого <input>

Ключевые слова: *форма, список, метка, кнопка, группировка, текстовое поле, переключатели, флажки, раскрывающийся список выбора, развернутый список выбора.*

Контрольные вопросы:

1. Для чего нужны формы в HTML?
2. Какие атрибуты имеет тег <form> и для чего они предназначены?
3. Как реализовать раскрывающийся список выбора? Опишите код программы.
4. Какой атрибут ограничивает только видимую область ввода данных, а не длину вводимой строки?
5. Как объединить несколько элементов ввода или управления в группу?

ГЛАВА 8. ОБЪЕКТНАЯ МОДЕЛЬ ДОКУМЕНТА

DOM (Document Object Model) – представляет собой стандарт консорциума W3C для программного доступа к документам HTML или XML. Фактически это платформо- и языково-нейтральный интерфейс, позволяющий программам и сценариям динамически обращаться и обновлять содержимое, структуру и стиль документа.

В рамках данного стандарта можно выделить 3 части:

- Core DOM - стандартная модель любого структурированного документа
- XML DOM - стандартная модель XML документа
- HTML DOM - стандартная модель HTML документа

DOM определяет объекты и свойства всех элементов документа и методы (интерфейс) для доступа к ним.

HTML DOM определяет объекты и свойства всех HTML элементов и методы (интерфейс) для доступа к ним. Иначе говоря, HTML DOM описывает каким образом необходимо получать, изменять, добавлять и удалять HTML элементы.

В соответствии с моделью DOM все, что содержится внутри HTML документа - является узлом. То есть HTML документ представляется в виде дерева узлов, которыми являются элементы, атрибуты и текст.

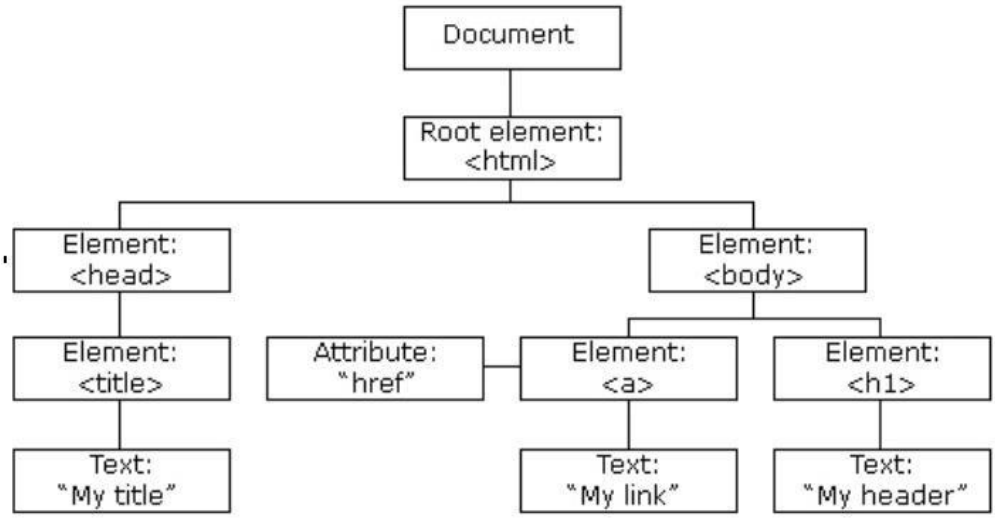


Рис.8.1.Древовидная модель HTML документа

Узлы дерева HTML документа

Согласно модели DOM:

- Весь документ представляется узлом документа;
- Каждый HTML тэг является узлом элемента;
- Текст внутри HTML элементов представляется текстовыми узлами;
- Каждому HTML атрибуту соответствует узел атрибута;
- Комментарии являются узлами комментариев.

```
<html>
<head>
<title>HTML документ</title>
</head>
<body>
<h1>Заголовок </h1>
<p>Просто текст</p>
</body>
</html>
```

В этом примере корневым узлом является тэг <html>. Все остальные узлы содержатся внутри <html>. У этого узла имеется два дочерних узла: <head> и <body>. Узел <head> содержит узел <title>, а узел <body> содержит узлы <h1> и <p>.

Следует обратить особое внимание на то, что текст, расположенный в узле элемента соответствует текстовому узлу. В примере <title>HTML документ </title> узел элемента <title> содержит текстовый узел "HTML документ", то есть "HTML документ" не является значением элемента <title>. Тем не менее, в рамках TML DOM значение текстового узла может быть доступно посредством свойства innerHTML.

Все узлы HTML документа могут быть доступны посредством дерева, при этом их содержимое может быть изменено или удалено, а также можно добавить новые элементы.

Все узлы дерева находятся в иерархических отношениях между собой. Для описания этих отношений используются термины родитель, дочерний элемент и потомок. Родительские узлы имеют дочерние узлы, а дочерние элементы одного уровня называются потомками (братьями или сестрами).

В отношении узлов дерева соблюдаются следующие принципы:

- Самый верхний узел дерева называется корневым;
- Каждый узел, за исключением корневого, имеет ровно один родительский узел;
- Узел может иметь любое число дочерних узлов;
- Конечный узел дерева не имеет дочерних узлов;
- Потомки имеют общего родителя.

Программный интерфейс HTML DOM

Как было указано выше, в рамках DOM модели HTML документ можно рассматривать как множество узловых объектов. Доступ к ним осуществляется с помощью JavaScript или других языков программирования. Программный интерфейс DOM включает в себя набор стандартных свойств и методов.

Свойства представляют некоторые сущности (например, <h1>), а методы – действия над ними (например, добавить <a>).

К типичным свойствам DOM относятся следующие:

- x.innerHTML - внутреннее текстовое значение HTML элемента x;
- x.nodeName - имя x;
- x.nodeValue - значение x;
- x.parentNode - родительский узел для x;
- x.childNodes - дочерний узел для x;
- x.attributes - узлы атрибутов x.

Узловой объект, соответствующий HTML элементу поддерживает следующие методы:

- x.getElementById(id) - получить элемент с указанным id;
- x.getElementsByTagName(name) - получить все элементы с указанным именем тэга (name);
- x.appendChild(node) - вставить дочерний узел для x;
- x.removeChild(node) - удалить дочерний узел для x.

Пример 3.

Для получения текста из элемента <p> со значением атрибута id "demo" в HTML документе можно использовать следующий код:

txt = document.getElementById("demo").innerHTML

Тот же самый результат может быть получен по-другому:

txt=document.getElementById("demo").childNodes[0].nodeValue

В рамках DOM возможны 3 способа доступа к узлам:

1. С помощью метода getElementById(ID) . При этом возвращается элемент с указанным ID.
2. С помощью метода getElementsByTagName(name). При этом возвращаются все узлы с указанным именем тэга (в виде индексированного списка). Первый элемент в списке имеет нулевой индекс.
3. Путем перемещения по дереву с использованием отношений между узлами.

Для определения длины списка узлов используется свойство length.

```
x = document.getElementsByTagName("p");
for (i = 0; i < x.length; i++) {
document.write(x[i].innerHTML);
document.write("<br/>");
}
```

В данном примере внутрь HTML документа вставляется в виде списка текстовое содержимое всех элементов соответствующих тэгу <p> .

Для навигации по дереву в ближайших окрестностях текущего узла можно использовать следующие свойства:

- parentNode;
- firstChild;
- lastChild.

Для непосредственного доступа к тэгам можно использовать 2 специальных свойства:

- document.documentElement – для доступа к корневому узлу документа;
- document.body – для доступа к тэгу <body> .

Свойства узлов

В HTML DOM каждый узел является объектом, который может иметь методы (функции) и свойства. Наиболее важными являются следующие свойства:

- nodeName;
- nodeValue;
- nodeType.

Свойство nodeName указывает на имя узла. Это свойство имеет следующие особенности:

- Свойство nodeName предназначено только для чтения;
- Свойство nodeName узла элемента точно соответствует имени тэга;
- Свойство nodeName узла атрибута соответствует имени атрибута;
- Свойство nodeName текстового узла всегда равно #text
- Свойство nodeName узла документа всегда равно #document

Замечание: nodeName всегда содержит имя тэга HTML элемента в верхнем регистре.

Свойство nodeValue указывает на значение узла. Это свойство имеет следующие особенности:

- Свойство nodeValue узла элемента не определено;
- Свойство nodeValue текстового узла указывает на сам текст;
- Свойство nodeValue узла атрибута указывает на значение атрибута.

Свойство nodeType возвращает тип узла. Это свойство предназначено только для чтения:

Наиболее важными типами узлов являются следующие:

Тип элемента	Тип узла
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Изменение HTML элементов

HTML элементы могут быть изменены с посредством использования JavaScript, HTML DOM и событий.

В следующем примере показано, как можно динамически изменять текстовое содержимое тэга <p> :

```
<html>
<body>
<p id="p1">Hello World!</p>
<script type="text/javascript">
document.getElementById("p1").innerHTML="New text!";
</script>
</body>
</html>
```

Ключевые слова: *XML DOM, HTML DOM, узел, дерево, атрибут, комментарий, дочерний узел, родитель, дочерний элемент, потомок, корневой узел.*

Контрольные вопросы

1. Что из себя представляет Document Object Model?
2. Что такое узел?
3. Посредством чего могут быть доступны все узлы HTML документа?
4. Какие принципы соблюдаются в отношении узлов дерева?
5. Каким образом осуществляется доступ к узловым объектам?
6. Основные свойства узлов.

ГЛАВА 9. CSS3 – КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ

Немного о CSS

Так что это такое – CSS? Cascading Style Sheets (Таблицы Каскадных Стилей) – это язык, содержащий набор свойств для описания внешнего вида любых HTML документов. С его помощью дизайнер имеет полный контроль над стилем и расположением каждого элемента веб страницы, что проще и гораздо функциональнее использования обычного набора HTML тегов. Приведу пример: Вам нужно создать жирный красный подчеркнутый текст.

ПРИМЕР HTML:

<u> Какой-то текст </u>

А если подобный стиль нужно использовать несколько раз? Хорошо если раз 5, а если 10-20? Вот тут нам и поможет CSS. Существует три вида таблиц стилей: **Внутренние таблицы стилей, Глобальные таблицы стилей** и **Связанные таблицы стилей**. Внутренние таблицы стилей (Inline Style Sheets) при помощи специального атрибута помещаются прямо в HTML теги. Глобальные (Global Style Sheets) определяют стиль элементов во всем документе. Связанные (Linked Style Sheets) могут быть использованы для нескольких документов сразу и хранятся во внешнем файле. Подробнее обо всем этом написано ниже.

Структура и правила

Селекторы (Selectors):

Синтаксисис:

селектор {свойства}

Любой элемент HTML – это возможный CSS селектор. Свойства селектора определяют стиль элемента, для которого он определен.

ПРИМЕР:

H1 {color:red; size:20pt;}

Все элементы H1 в документе будут красного цвета, размером в 20 точек (pt, point).

Классовые селекторы (Class Selectors):

Синтаксис:

селектор.класс {свойства}

CLASS - атрибут элемента в HTML, определяющий его класс. В CSS можно описать собственные стили для различных классов одних и тех же элементов.

ПРИМЕР:

H1.blue {color:blue; size:20pt;}

Все элементы H1 с атрибутом CLASS="blue" станут синими.

Классы могут так же быть описаны без явного привязывания их к определенным элементам.

Синтаксис:

.класс {свойства}

ПРИМЕР:

.green {color:green;}

В данном случае все элементы с атрибутом CLASS="green" станут зелеными.

ID селекторы (ID Selectors):

Синтакс:

#id {свойства}

ID – индивидуально именованный стиль. С его помощью можно создавать стилистические исключения среди элементов одного класса.

Идентификаторы используются в основном для придания одному или нескольким элементам одного класса индивидуальных свойств. Скажем, Вы создали класс blue – синий курсив. Но Вам понадобился жирный подчеркнутый текст синим курсивом. Конечно, можно создать новый класс, но зачем? Проще описать ID. Например "boldunderline". И все элементы класса blue с значением ID "boldunderline" станут жирным подчеркнутым синим курсивом. Произойдет как бы синтез свойств класса blue и идентификатора boldunderline.

ПРИМЕР:

```
<html>
<head>
<title> Пример CSS </title>
<style>
.blue {color:blue; font-style:italic}
#boldunderline {text-decoration:underline; font-weight:bold}
</style>
</head>
<body>
<p class="blue"> Здравствуйте, это моя домашняя страница. </p>
<p class="blue" id="boldunderline"> Пока еще в стадии разработки ... </p>
<p id="boldunderline">... Но скоро откроется </p>
</body>
</html>
```

Как видно из примера, атрибут ID может быть использован без указания класса (последний параграф примера. Тогда параграф будет обладать только свойствами ID "boldunderline" (в примере - жирный, подчеркнутый текст).

Контекстуальные селекторы (Contextual Selectors):

Контекстные селекторы – это сочетания нескольких обыкновенных селекторов. Стиль задается только элементам в заданной последовательности в зависимости от каскадного порядка.

ПРИМЕР:

```
P EM {color:silver;}
```

В данном примере все элементы EM внутри элементов P будут иметь заданный стиль.

Придание нескольким элементам одинаковых свойств:

Скажем Вам нужно придать нескольким элементам Вашей веб страницы одинаковых свойств. В этом случае при определении селекторы перечисляются через запятую перед блоком свойств.

ПРИМЕР:

h1,h2,h3,p,strong {color:green; font-style:italic;}

Все элементы h1, h2, h3, p и strong будут зелеными.

Псевдоклассы и псевдоэлементы :

Синтаксис:

селектор:псевдокласс { свойства }

селектор.класс:псевдокласс { свойства }

селектор:псевдоэлемент { свойства }

селектор.класс:псевдоэлемент { свойства }

Псевдоклассы и псевдоэлементы – это особые классы и элементы, присущие CSS и автоматически определяемые поддерживающими CSS браузерами. Псевдоклассы различают разные типы одного элемента, создавая при определении собственные стили для каждого из них. Псевдоэлементы являются частями других элементов, задавая этим частям отличный от элемента в целом стиль.

Список псевдоклассов и псевдоэлементов :

Anchor Pseudo Classes - эти псевдоклассы элемента <a href=" ">, обозначающего ссылку. Псевдоклассы этого элемента: (ссылка), active (активная ссылка), visited (посещенный ранее URL), hover (псевдокласс, возникающий при поднесении курсора к ссылке, не работает в Нетскейпе).

First Line Pseudo-element - first-line. Этот псевдоэлемент может быть использован с block-level элементами (p, h1 и т.д.). Он изменяет стиль первой строки этих элементов.

First Letter Pseudo-element - first-letter. Похож на first-line, но влияет не на всю строку, а только на первый символ.

ПРИМЕР:

a:link,a:visited {color:blue}

a:active {color:red}

a:hover {text-decoration:none}

В данном примере все элементы Anchor (ссылки) будут синими. При нажатии (в активном состоянии) поменяют цвет на красный. И при подведении курсора мышки исчезнет подчеркивание.

Примечание: описания нескольких свойств для одного селектора, контекстуального селектора, класса, индивидуально именованного стиля или группы объединенных селекторов отделяются друг от друга точкой с запятой ";".

Внутренние Таблицы Стилей

Как уже говорилось, использование Внутренних стилей мало чем отличается от использования обычных HTML тегов. Они задают стиль только одному элементу документа при помощи атрибута STYLE в HTML теге.

ПРИМЕР HTML:

 Вперед в будущее

ПРИМЕР INLINE STYLE SHEET:

 Вперед в будущее

Как можно заметить, код Inline Style Sheet получился больше чем HTML. Поэтому ISS следует использовать только если необходимо задать определенному элементу свой индивидуальный стиль, существующий в классификации CSS и нереализованный в HTML. Или же при необходимости абсолютно позиционировать данный элемент.

Глобальные Таблицы Стилей

Глобальные стили задают вид элементов всего документа. Для этого используется тег <STYLE type="text/css">. Он размещается в заголовке документа.

ПРИМЕР:

```
<html>
<head> <title> Пример Глобальных Таблиц Стилей </title>
<style type="text/css">
h1{color:red; font-style:italic; font-size:32px}
.blue{color:blue}
#bold{font-weight:bold}
</style>
</head>
<body>
<h1> Этот заголовок написан крупным красным курсивом </h1>
Вот <font class="blue"> это </font> слово - синие, а <font id="bold"> это</font> - жирное.
</body>
</html>
```

В данном примере все элементы H1 будут написаны крупным красным курсивом, все элементы с указанным классом BLUE будут синими , а все элементы с идентификатором ID="Bold" станут жирными. Для простоты вместо <STYLE type="text/css"> можно использовать просто тег <STYLE>, что менее грамотно.

Связанные Таблицы Стилей

Связанные таблицы стилей используются для придания нескольким документам одного стиля и хранятся в отдельном файле. Это очень привлекательно, когда нужно выдержать сайт в одном стиле, не утруждая себя составлением таблиц для каждого документа.

ПРИМЕР:

Файл styles.css

```
<STYLE type="text/css">
body {background:black; font-size:9pt; color:red; font-family:Arial Black}
.base{color:blue; font-style:italic}
h1 {color:white}
```

#bold {font-weight:bold}
</STYLE>

В самих же HTML документах делается ссылка на этот файл при помощи тега <LINK>. Выглядит это так: <LINK rel="STYLESHEET" TYPE="text/css" HREF="путь до файла">

ПРИМЕР:

Файл Index.html

<html>
<head>
<title> Просто еще один пример </title>
<LINK rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
Содержание Документа
</body>
</html>

Часть Il. Свойства CSS

Свойства Font

font-family

Возможные значения:

[1] любой шрифт

*Применимо для: всех элементов

Описание: это свойство определяет используемый элементом шрифт. Если указать URL, то шрифт автоматически установится на компьютер пользователя

ПРИМЕР:

font-family:Arial Black URL('arialblack.ttf')

font-style

Возможные значения:

[1] normal - без изменений

[2] italic - курсив

*Применимо для: всех элементов

Описание: стиль элемента. Курсивный или обычный

ПРИМЕР:

font-style:italic

font-variant

Возможные значения:

[1] normal - без изменений

[2] small-caps - заменяет все маленькие буквы на большие

*Применимо для: всех элементов

Описание: варианты отображения шрифта. Нетскейп не поддерживает это свойство

ПРИМЕР:

font-variant:small-caps

font-weight

Возможные значения:

[1] normal - без изменений

[2] bold - жирный

[3] bolder - очень жирный (в MSIE не отличается от bold, в Нетскейпе от нормал)

[4] lighter - тонкий (не отличается от normal)

[5] любое значение от 100 до 900

*Применимо для: всех элементов

Описание: выделение (жирность) элемента

ПРИМЕР:

font-weight:bold

font-size

Возможные значения:

[1] размер (+)

[2] xx-small, x-small, small, medium, large, x-large, xx-large - любое из этих значений

[3] smaller, larger - любое из этих значений

*Применимо для: всех элементов

Описание: размер шрифта

ПРИМЕР:

font-size:30pt

font

Возможные значения:

[1] font-family

[2] font-style

[3] font-variant

[4] font-weight

[5] font-size

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства

ПРИМЕР:

font: italic bolder Arial 12pt

Свойства Text

word-spacing

Возможные значения:

[1] длина (+)

[2] normal - без изменений

*Применимо для: всех элементов

Описание: расстояние между словами. Не работает ни в Нетскейпе, ни в MSIE

ПРИМЕР:

word-spacing:0.4em

<UL

· text-decoration

Возможные значения:

[1] none - нет

[2] underline - подчеркнутый

[3] overline - надчеркнутый (не поддерживается в Нетскейпе)

[4] line-through - перечеркнутый

[5] blink - мигающий (не поддерживается в IE)

*Применимо для: всех элементов

Описание: "украшение" текста

ПРИМЕР:

text-decoration:line-through

letter-spacing

Возможные значения:

[1] длина (+)

[2] normal - без изменений

*Применимо для: всех элементов

Описание: расстояние между буквами. Не работает в Нетскейпе

ПРИМЕР:

letter-spacing:100

vertical-align

Возможные значения:

[1] baseline

[2] sub

[3] super

[4] top-text

[5] top

[6] middle

[7] bottom

[8] bottom-text

[9] процент

*Применимо для: inline элементов

Описание: позиционирование элементов по отношению к другим элементам стоящих в одном ряду. Не работает в Нетскейпе

ПРИМЕР:

vertical-align:top-text

text-transform

Возможные значения:

[1] none - нет

[2] Capitalize - каждое слово начинается с большой буквы

[3] UPPERCASE - каждая буква текста становится заглавной

[4] lowercase - каждая буква текста становится маленькой

*Применимо для: inline элементов

Описание: изменение текста. Не работает в Нетскейпе

ПРИМЕР:

text-transform:Capitalize

text-align
Возможные значения:
[1] left - текст слева
[2] right - текст справа
[3] center - текст по центру
[3] justify - текст "растянут"
*Применимо для: block-level элементов
Описание: положение текста
ПРИМЕР:
text-align:right
text-indent
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: block-level элементов
Описание: отступ
ПРИМЕР:
text-indent:30 em
line-height
Возможные значения:
[1] normal - без изменений
[2] длина (+)
[3] процент
*Применимо для: всех элементов
Описание: отступ сверху
ПРИМЕР:
line-height:100%
Свойства Color и Background
color
Возможные значения:
[1] цвет (+)
*Применимо для: всех элементов
Описание: цвет
ПРИМЕР:
color:#f00000
backgroung-color
Возможные значения:
[1] цвет (+)
*Применимо для: всех элементов
Описание: цвет фона элемента
ПРИМЕР:
background-color:#f00000
background-image
Возможные значения:
[1] none - нет

[2] URL (+)
*Применимо для: всех элементов
Описание: фоновое изображение
ПРИМЕР:
background-image:URL(cool.gif)
background-repeat
Возможные значения:
[1] repeat - размножает фоновое изображение во всех направлениях
[2] repeat-x - размножает фоновое изображение горизонтально
[3] repeat-y - размножает фоновое изображение вертикально
[4] no-repeat - не повторяющиеся изображение
*Применимо для: всех элементов
Описание: повторения фонового изображения
ПРИМЕР:
background-repeat:no-repeat
background-attachment
Возможные значения:
[1] scroll - фоновое изображение скролится вместе с содержанием документа
[2] fixed - не скролится. Фиксируется в одном месте. Не работает в Нетскейпе
*Применимо для: всех элементов
Описание: возможность прокрутки фонового изображения
ПРИМЕР:
background-attachment:fixed
background-position
Возможные значения:
[1] процент от ширины + процент от высоты (+)
[2] top, middle, bottom - одно из значений
[3] left, center, right - одно из начений
[4] расстояние от левого края + расстояние от вершины
*Применимо для: block-level и replaced элементов
Описание: положение фонового изображения (работает с background-repeat равным repeat-x, repeat-y или no-repeat)
ПРИМЕР:
background-position:50%0%
background
Возможные значения:
[1] background-color
[2] background-image
[3] backgroun-position
[4] background-attachment
[5] background-repeat
*Применимо для: всех элементов
Описание: обобщает вышеперечисленные свойства

ПРИМЕР:
background:no-repeat black fixed 50%0%
Свойства Box
margin-top
Возможные значения:
[1] длина (+)
[2] процент (+)
[3] auto - автоматически
*Применимо для: всех элементов
Описание: определяет отступ сверху
ПРИМЕР:
margin-top:100
margin-right
Возможные значения:
[1] длина (+)
[2] процент (+)
[3] auto - автоматически
*Применимо для: всех элементов
Описание: определяет отступ справа
ПРИМЕР:
margin-right:100%
margin-bottom
Возможные значения:
[1] длина (+)
[2] процент (+)
[3] auto - автоматически
*Применимо для: всех элементов
Описание: определяет отступ снизу
ПРИМЕР:
margin-bottom:100em
margin-left
Возможные значения:
[1] длина (+)
[2] процент (+)
[3] auto - автоматически
*Применимо для: всех элементов
Описание: определяет отступ слева
ПРИМЕР:
margin-left:100pt
margin
Возможные значения:
[1] margin-top
[2] margin-right
[3] margin-left
[4] margin-bottom

*Применимо для: всех элементов
Описание: обобщает все вышеперечисленные свойства
ПРИМЕР:
background:100pt
padding-top
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: осех элементов
Описание: отступ от верхнего border'a
ПРИМЕР:
padding-top:100pt
padding-right
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: всех элементов
Описание: отступ от правого border'a
ПРИМЕР:
padding-right:100%
padding-bottom
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: всех элементов
Описание: отступ от нижнего border'a
ПРИМЕР:
padding-bottom:100em
padding-left
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: всех элементов
Описание: отступ от левого border'a
ПРИМЕР:
padding-top:100
padding
Возможные значения:
[1] padding-top
[2] padding-right
[3] padding-left
[4] paddung-bottom
*Применимо для: всех элементов
Описание: обобщает вышеперечисленные свойства. Можно задать несколько значений одновременно (до четырех) для разных сторон. Если

установлено одно значение - задается единый отступ для всех сторон, если два - то задаются различные отступы для прилежащих сторон, а если четыре - то задаются индивидуальные отступы для всех сторон.

ПРИМЕР:
padding:100px
border-top-width
Возможные значения:
[1] длина (+)
[2] thin, medium или thick
*Применимо для: всех элементов
Описание: толщина верхнего border'a
ПРИМЕР:
border-top-width:100pt
border-right-width
Возможные значения:
[1] длина (+)
[2] thin, medium или thick
*Применимо для: всех элементов
Описание: толщина правого border'a
ПРИМЕР:
border-right-width:thick
border-bottom-width
Возможные значения:
[1] длина (+)
[2] thin, medium или thick
*Применимо для: всех элементов
Описание: толщина нижнего border'a
ПРИМЕР:
border-bottom-width:100em
border-left-width
Возможные значения:
[1] длина (+)
[2] thin, medium или thick
*Применимо для: всех элементов
Описание: толщина левого border'a
ПРИМЕР:
border-left-width:medium
border-width
Возможные значения:
[1] border-top-width
[2] border-right-width
[3] border-left-width
[4] border-bottom-width
*Применимо для: всех элементов

Описание: толщина border'ов. Можно задать несколько значений одновременно (до четырех) для разных border'ов. Если установлено одно значение - задается единая толщина для всех сторон, если два - то задаются различная толщина для прилежащих сторон, а если четыре - то задаются индивидуальная толщина для всех сторон

ПРИМЕР:

border-width: 15pt

border-color

Возможные значения:

[1] цвет (+)

*Применимо для: всех элементов

Описание: Цвет border'a. Не работает в Нетскейпе

ПРИМЕР:

border-color:green

border-style

Возможные значения:

[1] none

[2] dotted, dashed, solid, double, groove, ridge, inset, outset

*Применимо для: всех элементов

Описание: стиль border'ов. Можно задать несколько значений одновременно (до четырех) для разных border'ов. Если установлено одно значение - задается единый стиль для всех сторон, если два - то задаются различные стили для прилежащих сторон, а если четыре - то задаются индивидуальные стили для всех сторон

ПРИМЕР:

border-style: dotted groove

<UL

· border-top

Возможные значения:

[1] border-top-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства для верхнего border'a

ПРИМЕР:

border-top: 100em red groove

border-right

Возможные значения:

[1] border-right-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства для правого border'a

ПРИМЕР:

border-right: 5pt magenta solid

border-left

Возможные значения:

[1] border-left-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства для левого border'a

ПРИМЕР:

border-left: 15pc coral inset

border-bottom

Возможные значения:

[1] border-bottom-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства для нижнего border'a

ПРИМЕР:

border-bottom: 30 orange outset

border

Возможные значения:

[1] border-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства

ПРИМЕР:

border: thik black double

width

Возможные значения:

[1] длина (+)

[2] процент (+)

*Применимо для: block-level и replaced элементов

Описание: ширина элемента

ПРИМЕР:

width:10%

height

Возможные значения:

[1] длина (+)

[2] процент (+)

*Применимо для: block-level и replaced элементов

Описание: высота элемента

ПРИМЕР:

height:100pt

float

Возможные значения:

[1] left - слева

[2] right - справа

[3] none - по умолчанию

*Применимо для: всех элементов

Описание: расположение элемента

ПРИМЕР:

float:right

clear

Возможные значения:

[1] left - слева

[2] right - справа

[3] both - с двух сторон

[4] none - по умолчанию

*Применимо для: всех элементов

Описание: расположение других элементов вокруг данного

ПРИМЕР:

clear:both

Классификация

display

Возможные значения:

[1] none - не отображается

[2] block - разбивает строку до и после элемента (т.е. элемент не может находится на одной линии с другими элементами)

[3] inline - не разбивает строку

[4] list-item - разбивает линию строку до и после элемента + добавляет маркер как у list-item элементов

*Применимо для: всех элементов

Описание: определяет, как будет отображаться элемент

ПРИМЕР:

display:none

white-space

Возможные значения:

[1] normal - "сжимает" несколько подряд идущих пробелов в один

[2] pre - допускает отображение несколькольких подряд идущих пробелов

[3] nowrap - не допускает перенос строки без тега

*Применимо для: block-level элементов

Описание: оприделяет, как будут отображаться пробелы между элементами

ПРИМЕР:

white-space:nowrap

list-style-type

Возможные значения:

[1] disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha или upper-alpha

[2] none - никакой

*Применимо для: элементов со значением display равным list-item

Описание: определяет вид list-item маркера. Если значение list-style-image равно none или не уточнено

ПРИМЕР:

list-style-type:lower-alpha

list-style-image

Возможные значения:

[1] none - нет

[2] URL (+)

*Применимо для: элементов со значением display равным list-item

Описание: задает вид list-item маркера в виде картинки

ПРИМЕР:

list-style-image:URL(cool.gif)

list-style-position

Возможные значения:

[1] inside - при переносе следующие строки будут отображаться без отступа

[2] outside - по умолчанию

*Применимо для: элементов со значением display равным list-item

Описание: определяет положение маркера в зависимости от list item элемента

ПРИМЕР:

list-style-position:inside

list-style

Возможные значения:

[1] list-style-type

[2] list-style-position

[3] list-style-image

*Применимо для: элементов со значением display равным list-item

Описание: обобщает вышеперечисленные свойства

ПРИМЕР:

list-style:inside

Часть III. Дополнения

Меры длины

Синтаксис: "+" (можно опустить) или "-" затем [число] плюс [единица измерения] (без пропусков)

ПРИМЕР:

-566pt

Единицы длинны : ex - x-height, ширина буквы "x" используемого элементом шрифта

px - pixels, пикселы

in - inches, дюймы

cm - centimeters, сантиметры
mm - millimeters, миллиметры
pt - points, точка (1pt = 1/72in)
pc - picas (1pc = 12pt)
Процентные меры
Синтаксис : "+" или "-" затем [число] плюс "%" (без пропусков)
ПРИМЕР :
-566%
Цвета
Синтаксис: [color]
ПРИМЕР :
magenta
Значением цвета может быть его название (red , lightgreen, coral и т.д.) или RGB значение
Способы выразить цвет в RGB (red green blue) :
#rrggbb (например, #00cc00)
rgb(x,x,x) -- где "x" число от 0 до 255 (например, rgb(0,204,0))
#rgb (например, #0c0)
rgb (x%,x%,x%) -- где "x%" число от 0.0 до 100.0 (например, 0%,80%,0%)
Все примеры отображают один и тот же цвет
Ссылки
Синтаксис: "URL", потом в скобках приводится ссылка. Ссылку также можно, помимо скобок, заключить в одинарные или двойные кавычки (без пропусков)
ПРИМЕР:
URL('cool.gif')
Под MSIE и Нетскейпом подразумиваются Microsoft Internet Explorer 4.0+ и Netscape Navigator 4.0+ соответственно.

Ключевые слова: *CSS(каскадная таблица стилей), внутренние таблицы стилей, глобальные таблицы стилей, связанные таблицы стилей, селектор, CLASS, ID, идентификатор, псевдокласс, псевдоэлемент, ссылки.*

Контрольные вопросы

1. Что есть CSS?
2. Для чего нужны внутренние таблицы стилей, глобальные таблицы стилей и связанные таблицы стилей?
3. Что создают с помощью ID?
4. Что из себя представляют псевдоклассы и псевдоэлементы?
5. Что такое Anchor Pseudo Classes?

ГЛАВА 9. CSS3 – КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ

Часть I. Что есть CSS

Немного о CSS

Так что это такое – CSS? Cascading Style Sheets (Таблицы Каскадных Стилей) – это язык, содержащий набор свойств для описания внешнего вида любых HTML документов. С его помощью дизайнер имеет полный контроль над стилем и расположением каждого элемента веб страницы, что проще и гораздо функциональнее использования обычного набора HTML тегов. Приведу пример: Вам нужно создать жирный красный подчеркнутый текст.

ПРИМЕР HTML:

<u> Какой-то текст </u>

А если подобный стиль нужно использовать несколько раз? Хорошо если раз 5, а если 10-20? Вот тут нам и поможет CSS. Существует три вида таблиц стилей: **Внутренние таблицы стилей, Глобальные таблицы стилей** и **Связанные таблицы стилей**. Внутренние таблицы стилей (Inline Style Sheets) при помощи специального атрибута помещаются прямо в HTML теги. Глобальные (Global Style Sheets) определяют стиль элементов во всем документе. Связанные (Linked Style Sheets) могут быть использованы для нескольких документов сразу и хранятся во внешнем файле. Подробнее обо всем этом написано ниже.

Структура и правила

Селекторы (Selectors):

Синтаксисис:

селектор {свойства}

Любой элемент HTML – это возможный CSS селектор. Свойства селектора определяют стиль элемента, для которого он определен.

ПРИМЕР:

H1 {color:red; size:20pt;}

Все элементы H1 в документе будут красного цвета, размером в 20 точек (pt, point).

Классовые селекторы (Class Selectors):

Синтаксис:

селектор.класс {свойства}

CLASS - атрибут элемента в HTML, определяющий его класс. В CSS можно описать собственные стили для различных классов одних и тех же элементов.

ПРИМЕР:

H1.blue {color:blue; size:20pt;}

Все элементы H1 с атрибутом CLASS="blue" станут синими.

Классы могут так же быть описаны без явного привязывания их к определенным элементам.

Синтаксис:

.класс {свойства}

ПРИМЕР:

.green {color:green;}

В данном случае все элементы с атрибутом CLASS="green" станут зелеными.

ID селекторы (ID Selectors):

Синтакс:

#id {свойства}

ID – индивидуально именованный стиль. С его помощью можно создавать стилистические исключения среди элементов одного класса.

Идентификаторы используются в основном для придания одному или нескольким элементам одного класса индивидуальных свойств. Скажем, Вы создали класс blue – синий курсив. Но Вам понадобился жирный подчеркнутый текст синим курсивом. Конечно, можно создать новый класс, но зачем? Проще описать ID. Например "boldunderline". И все элементы класса blue с значением ID "boldunderline" станут жирным подчеркнутым синим курсивом. Произойдет как бы синтез свойств класса blue и идентификатора boldunderline.

ПРИМЕР:

```
<html>
<head>
<title> Пример CSS </title>
<style>
.blue {color:blue; font-style:italic}
#boldunderline {text-decoration:underline; font-weight:bold}
</style>
</head>
<body>
<p class="blue"> Здравствуйте, это моя домашняя страница. </p>
<p class="blue" id="boldunderline"> Пока еще в стадии разработки ... </p>
<p id="boldunderline">... Но скоро откроется </p>
</body>
</html>
```

Как видно из примера, атрибут ID может быть использован без указания класса (последний параграф примера. Тогда параграф будет обладать только свойствами ID "boldunderline" (в примере - жирный, подчеркнутый текст).

Контекстуальные селекторы (Contextual Selectors):

Контекстные селекторы – это сочетания нескольких обыкновенных селекторов. Стиль задается только элементам в заданной последовательности в зависимости от каскадного порядка.

ПРИМЕР:

P EM {color:silver;}

В данном примере все элементы EM внутри элементов P будут иметь заданный стиль.

Придание нескольким элементам одинаковых свойств:

Скажем Вам нужно придать нескольким элементам Вашей веб страницы одинаковых свойств. В этом случае при определении селекторы перечисляются через запятую перед блоком свойств.

ПРИМЕР:

h1,h2,h3,p,strong {color:green; font-style:italic;}

Все элементы h1, h2, h3, p и strong будут зелеными.

Псевдоклассы и псевдоэлементы :

Синтаксис:

селектор:псевдокласс { свойства }

селектор.класс:псевдокласс { свойства }

селектор:псевдоэлемент { свойства }

селектор.класс:псевдоэлемент { свойства }

Псевдоклассы и псевдоэлементы – это особые классы и элементы, присущие CSS и автоматически определяемые поддерживающими CSS браузерами. Псевдоклассы различают разные типы одного элемента, создавая при определении собственные стили для каждого из них. Псевдоэлементы являются частями других элементов, задавая этим частям отличный от элемента в целом стиль.

Список псевдоклассов и псевдоэлементов :

Anchor Pseudo Classes - эти псевдоклассы элемента <a href=" ">, обозначающего ссылку. Псевдоклассы этого элемента: (ссылка), active (активная ссылка), visited (посещенный ранее URL), hover (псевдокласс, возникающий при поднесении курсора к ссылке, не работает в Нетскейпе).

First Line Pseudo-element - first-line. Этот псевдоэлемент может быть использован с block-level элементами (p, h1 и т.д.). Он изменяет стиль первой строки этих элементов.

First Letter Pseudo-element - first-letter. Похож на first-line, но влияет не на всю строку, а только на первый символ.

ПРИМЕР:

a:link,a:visited {color:blue}

a:active {color:red}

a:hover {text-decoration:none}

В данном примере все элементы Anchor (ссылки) будут синими. При нажатии (в активном состоянии) поменяют цвет на красный. И при подведении курсора мышки исчезнет подчеркивание.

Примечание: описания нескольких свойств для одного селектора, контекстуального селектора, класса, индивидуально именованного стиля или группы объединенных селекторов отделяются друг от друга точкой с запятой ";".

Внутренние Таблицы Стилей

Как уже говорилось, использование Внутренних стилей мало чем отличается от использования обычных HTML тегов. Они задают стиль только одному элементу документа при помощи атрибута STYLE в HTML теге.

ПРИМЕР HTML:

 Вперед в будущее

ПРИМЕР INLINE STYLE SHEET:

 Вперед в будущее

Как можно заметить, код Inline Style Sheet получился больше чем HTML. Поэтому ISS следует использовать только если необходимо задать определенному элементу свой индивидуальный стиль, существующий в классификации CSS и нереализованный в HTML. Или же при необходимости абсолютно позиционировать данный элемент.

Глобальные Таблицы Стилей

Глобальные стили задают вид элементов всего документа. Для этого используется тег <STYLE type="text/css">. Он размещается в заголовке документа.

ПРИМЕР:

```
<html>
<head> <title> Пример Глобальных Таблиц Стилей </title>
<style type="text/css">
h1{color:red; font-style:italic; font-size:32px}
.blue{color:blue}
#bold{font-weight:bold}
</style>
</head>
<body>
<h1> Этот заголовок написан крупным красным курсивом </h1>
Вот <font class="blue"> это </font> слово - синие, а <font id="bold"> это</font> - жирное.
</body>
</html>
```

В данном примере все элементы H1 будут написаны крупным красным курсивом, все элементы с указанным классом BLUE будут синими , а все элементы с идентификатором ID="Bold" станут жирными. Для простоты вместо <STYLE type="text/css"> можно использовать просто тег <STYLE>, что менее грамотно.

Связанные Таблицы Стилей

Связанные таблицы стилей используются для придания нескольким документам одного стиля и хранятся в отдельном файле. Это очень привлекательно, когда нужно выдержать сайт в одном стиле, не утруждая себя составлением таблиц для каждого документа.

ПРИМЕР:

Файл styles.css

```
<STYLE type="text/css">
body {background:black; font-size:9pt; color:red; font-family:Arial Black}
.base{color:blue; font-style:italic}
h1 {color:white}
```

#bold {font-weight:bold}
</STYLE>

В самих же HTML документах делается ссылка на этот файл при помощи тега <LINK>. Выглядит это так: <LINK rel="STYLESHEET" TYPE="text/css" HREF="путь до файла">

ПРИМЕР:

Файл Index.html

<html>
<head>
<title> Просто еще один пример </title>
<LINK rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
Содержание Документа
</body>
</html>

Часть Il. Свойства CSS

Свойства Font

font-family

Возможные значения:

[1] любой шрифт

*Применимо для: всех элементов

Описание: это свойство определяет используемый элементом шрифт. Если указать URL, то шрифт автоматически установится на компьютер пользователя

ПРИМЕР:

font-family:Arial Black URL('arialblack.ttf')

font-style

Возможные значения:

[1] normal - без изменений

[2] italic - курсив

*Применимо для: всех элементов

Описание: стиль элемента. Курсивный или обычный

ПРИМЕР:

font-style:italic

font-variant

Возможные значения:

[1] normal - без изменений

[2] small-caps - заменяет все маленькие буквы на большие

*Применимо для: всех элементов

Описание: варианты отображения шрифта. Нетскейп не поддерживает это свойство

ПРИМЕР:

font-variant:small-caps

font-weight

Возможные значения:

[1] normal - без изменений

[2] bold - жирный

[3] bolder - очень жирный (в MSIE не отличается от bold, в Нетскейпе от нормал)

[4] lighter - тонкий (не отличается от normal)

[5] любое значение от 100 до 900

*Применимо для: всех элементов

Описание: выделение (жирность) элемента

ПРИМЕР:

font-weight:bold

font-size

Возможные значения:

[1] размер (+)

[2] xx-small, x-small, small, medium, large, x-large, xx-large - любое из этих значений

[3] smaller, larger - любое из этих значений

*Применимо для: всех элементов

Описание: размер шрифта

ПРИМЕР:

font-size:30pt

font

Возможные значения:

[1] font-family

[2] font-style

[3] font-variant

[4] font-weight

[5] font-size

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства

ПРИМЕР:

font: italic bolder Arial 12pt

Свойства Text

word-spacing

Возможные значения:

[1] длина (+)

[2] normal - без изменений

*Применимо для: всех элементов

Описание: расстояние между словами. Не работает ни в Нетскейпе, ни в MSIE

ПРИМЕР:

word-spacing:0.4em

<UL

· text-decoration

Возможные значения:

[1] none - нет

[2] underline - подчеркнутый

[3] overline - надчеркнутый (не поддерживается в Нетскейпе)

[4] line-through - перечеркнутый

[5] blink - мигающий (не поддерживается в IE)

*Применимо для: всех элементов

Описание: "украшение" текста

ПРИМЕР:

text-decoration:line-through

letter-spacing

Возможные значения:

[1] длина (+)

[2] normal - без изменений

*Применимо для: всех элементов

Описание: расстояние между буквами. Не работает в Нетскейпе

ПРИМЕР:

letter-spacing:100

vertical-align

Возможные значения:

[1] baseline

[2] sub

[3] super

[4] top-text

[5] top

[6] middle

[7] bottom

[8] bottom-text

[9] процент

*Применимо для: inline элементов

Описание: позиционирование элементов по отношению к другим элементам стоящих в одном ряду. Не работает в Нетскейпе

ПРИМЕР:

vertical-align:top-text

text-transform

Возможные значения:

[1] none - нет

[2] Capitalize - каждое слово начинается с большой буквы

[3] UPPERCASE - каждая буква текста становится заглавной

[4] lowercase - каждая буква текста становится маленькой

*Применимо для: inline элементов

Описание: изменение текста. Не работает в Нетскейпе

ПРИМЕР:

text-transform:Capitalize

text-align
Возможные значения:
[1] left - текст слева
[2] right - текст справа
[3] center - текст по центру
[3] justify - текст "растянут"
*Применимо для: block-level элементов
Описание: положение текста
ПРИМЕР:
text-align:right
text-indent
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: block-level элементов
Описание: отступ
ПРИМЕР:
text-indent:30 em
line-height
Возможные значения:
[1] normal - без изменений
[2] длина (+)
[3] процент
*Применимо для: всех элементов
Описание: отступ сверху
ПРИМЕР:
line-height:100%
Свойства Color и Background
color
Возможные значения:
[1] цвет (+)
*Применимо для: всех элементов
Описание: цвет
ПРИМЕР:
color:#f00000
backgroung-color
Возможные значения:
[1] цвет (+)
*Применимо для: всех элементов
Описание: цвет фона элемента
ПРИМЕР:
background-color:#f00000
background-image
Возможные значения:
[1] none - нет

[2] URL (+)
*Применимо для: всех элементов
Описание: фоновое изображение
ПРИМЕР:
background-image:URL(cool.gif)
background-repeat
Возможные значения:
[1] repeat - размножает фоновое изображение во всех направлениях
[2] repeat-x - размножает фоновое изображение горизонтально
[3] repeat-y - размножает фоновое изображение вертикально
[4] no-repeat - не повторяющиеся изображение
*Применимо для: всех элементов
Описание: повторения фонового изображения
ПРИМЕР:
background-repeat:no-repeat
background-attachment
Возможные значения:
[1] scroll - фоновое изображение скролится вместе с содержанием документа
[2] fixed - не скролится. Фиксируется в одном месте. Не работает в Нетскейпе
*Применимо для: всех элементов
Описание: возможность прокрутки фонового изображения
ПРИМЕР:
background-attachment:fixed
background-position
Возможные значения:
[1] процент от ширины + процент от высоты (+)
[2] top, middle, bottom - одно из значений
[3] left, center, right - одно из начений
[4] расстояние от левого края + расстояние от вершины
*Применимо для: block-level и replaced элементов
Описание: положение фонового изображения (работает с background-repeat равным repeat-x, repeat-y или no-repeat)
ПРИМЕР:
background-position:50%0%
background
Возможные значения:
[1] background-color
[2] background-image
[3] backgroun-position
[4] background-attachment
[5] background-repeat
*Применимо для: всех элементов
Описание: обобщает вышеперечисленные свойства

ПРИМЕР:
background:no-repeat black fixed 50%0%
Свойства Box
margin-top
Возможные значения:
[1] длина (+)
[2] процент (+)
[3] auto - автоматически
*Применимо для: всех элементов
Описание: определяет отступ сверху
ПРИМЕР:
margin-top:100
margin-right
Возможные значения:
[1] длина (+)
[2] процент (+)
[3] auto - автоматически
*Применимо для: всех элементов
Описание: определяет отступ справа
ПРИМЕР:
margin-right:100%
margin-bottom
Возможные значения:
[1] длина (+)
[2] процент (+)
[3] auto - автоматически
*Применимо для: всех элементов
Описание: определяет отступ снизу
ПРИМЕР:
margin-bottom:100em
margin-left
Возможные значения:
[1] длина (+)
[2] процент (+)
[3] auto - автоматически
*Применимо для: всех элементов
Описание: определяет отступ слева
ПРИМЕР:
margin-left:100pt
margin
Возможные значения:
[1] margin-top
[2] margin-right
[3] margin-left
[4] margin-bottom

*Применимо для: всех элементов
Описание: обобщает все вышеперечисленные свойства
ПРИМЕР:
background:100pt
padding-top
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: осех элементов
Описание: отступ от верхнего border'a
ПРИМЕР:
padding-top:100pt
padding-right
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: всех элементов
Описание: отступ от правого border'a
ПРИМЕР:
padding-right:100%
padding-bottom
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: всех элементов
Описание: отступ от нижнего border'a
ПРИМЕР:
padding-bottom:100em
padding-left
Возможные значения:
[1] длина (+)
[2] процент (+)
*Применимо для: всех элементов
Описание: отступ от левого border'a
ПРИМЕР:
padding-top:100
padding
Возможные значения:
[1] padding-top
[2] padding-right
[3] padding-left
[4] paddung-bottom
*Применимо для: всех элементов
Описание: обобщает вышеперечисленные свойства. Можно задать несколько значений одновременно (до четырех) для разных сторон. Если

установлено одно значение - задается единый отступ для всех сторон, если два - то задаются различные отступы для прилежащих сторон, а если четыре - то задаются индивидуальные отступы для всех сторон.

ПРИМЕР:
padding:100px
border-top-width
Возможные значения:
[1] длина (+)
[2] thin, medium или thick
*Применимо для: всех элементов
Описание: толщина верхнего border'a
ПРИМЕР:
border-top-width:100pt
border-right-width
Возможные значения:
[1] длина (+)
[2] thin, medium или thick
*Применимо для: всех элементов
Описание: толщина правого border'a
ПРИМЕР:
border-right-width:thick
border-bottom-width
Возможные значения:
[1] длина (+)
[2] thin, medium или thick
*Применимо для: всех элементов
Описание: толщина нижнего border'a
ПРИМЕР:
border-bottom-width:100em
border-left-width
Возможные значения:
[1] длина (+)
[2] thin, medium или thick
*Применимо для: всех элементов
Описание: толщина левого border'a
ПРИМЕР:
border-left-width:medium
border-width
Возможные значения:
[1] border-top-width
[2] border-right-width
[3] border-left-width
[4] border-bottom-width
*Применимо для: всех элементов

Описание: толщина border'ов. Можно задать несколько значений одновременно (до четырех) для разных border'ов. Если установлено одно значение - задается единая толщина для всех сторон, если два - то задаются различная толщина для прилежащих сторон, а если четыре - то задаются индивидуальная толщина для всех сторон

ПРИМЕР:

border-width: 15pt

border-color

Возможные значения:

[1] цвет (+)

*Применимо для: всех элементов

Описание: Цвет border'a. Не работает в Нетскейпе

ПРИМЕР:

border-color:green

border-style

Возможные значения:

[1] none

[2] dotted, dashed, solid, double, groove, ridge, inset, outset

*Применимо для: всех элементов

Описание: стиль border'ов. Можно задать несколько значений одновременно (до четырех) для разных border'ов. Если установлено одно значение - задается единый стиль для всех сторон, если два - то задаются различные стили для прилежащих сторон, а если четыре - то задаются индивидуальные стили для всех сторон

ПРИМЕР:

border-style: dotted groove

<UL

· border-top

Возможные значения:

[1] border-top-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства для верхнего border'a

ПРИМЕР:

border-top: 100em red groove

border-right

Возможные значения:

[1] border-right-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства для правого border'a

ПРИМЕР:

border-right: 5pt magenta solid

border-left

Возможные значения:

[1] border-left-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства для левого border'a

ПРИМЕР:

border-left: 15pc coral inset

border-bottom

Возможные значения:

[1] border-bottom-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства для нижнего border'a

ПРИМЕР:

border-bottom: 30 orange outset

border

Возможные значения:

[1] border-width

[2] border-style

[3] border-color

*Применимо для: всех элементов

Описание: обобщает вышеперечисленные свойства

ПРИМЕР:

border: thik black double

width

Возможные значения:

[1] длина (+)

[2] процент (+)

*Применимо для: block-level и replaced элементов

Описание: ширина элемента

ПРИМЕР:

width:10%

height

Возможные значения:

[1] длина (+)

[2] процент (+)

*Применимо для: block-level и replaced элементов

Описание: высота элемента

ПРИМЕР:

height:100pt

float

Возможные значения:

[1] left - слева

[2] right - справа

[3] none - по умолчанию

*Применимо для: всех элементов

Описание: расположение элемента

ПРИМЕР:

float:right

clear

Возможные значения:

[1] left - слева

[2] right - справа

[3] both - с двух сторон

[4] none - по умолчанию

*Применимо для: всех элементов

Описание: расположение других элементов вокруг данного

ПРИМЕР:

clear:both

Классификация

display

Возможные значения:

[1] none - не отображается

[2] block - разбивает строку до и после элемента (т.е. элемент не может находится на одной линии с другими элементами)

[3] inline - не разбивает строку

[4] list-item - разбивает линию строку до и после элемента + добавляет маркер как у list-item элементов

*Применимо для: всех элементов

Описание: определяет, как будет отображаться элемент

ПРИМЕР:

display:none

white-space

Возможные значения:

[1] normal - "сжимает" несколько подряд идущих пробелов в один

[2] pre - допускает отображение несколькольких подряд идущих пробелов

[3] nowrap - не допускает перенос строки без тега

*Применимо для: block-level элементов

Описание: оприделяет, как будут отображаться пробелы между элементами

ПРИМЕР:

white-space:nowrap

list-style-type

Возможные значения:

[1] disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha или upper-alpha

[2] none - никакой

*Применимо для: элементов со значением display равным list-item

Описание: определяет вид list-item маркера. Если значение list-style-image равно none или не уточнено

ПРИМЕР:

list-style-type:lower-alpha

list-style-image

Возможные значения:

[1] none - нет

[2] URL (+)

*Применимо для: элементов со значением display равным list-item

Описание: задает вид list-item маркера в виде картинки

ПРИМЕР:

list-style-image:URL(cool.gif)

list-style-position

Возможные значения:

[1] inside - при переносе следующие строки будут отображаться без отступа

[2] outside - по умолчанию

*Применимо для: элементов со значением display равным list-item

Описание: определяет положение маркера в зависимости от list item элемента

ПРИМЕР:

list-style-position:inside

list-style

Возможные значения:

[1] list-style-type

[2] list-style-position

[3] list-style-image

*Применимо для: элементов со значением display равным list-item

Описание: обобщает вышеперечисленные свойства

ПРИМЕР:

list-style:inside

Часть III. Дополнения

Меры длины

Синтаксис: "+" (можно опустить) или "-" затем [число] плюс [единица измерения] (без пропусков)

ПРИМЕР:

-566pt

Единицы длинны : ex - x-height, ширина буквы "x" используемого элементом шрифта

px - pixels, пикселы

in - inches, дюймы

cm - centimeters, сантиметры
mm - millimeters, миллиметры
pt - points, точка (1pt = 1/72in)
pc - picas (1pc = 12pt)
Процентные меры
Синтаксис : "+" или "-" затем [число] плюс "%" (без пропусков)
ПРИМЕР :
-566%
Цвета
Синтаксис: [color]
ПРИМЕР :
magenta
Значением цвета может быть его название (red , lightgreen, coral и т.д.) или RGB значение
Способы выразить цвет в RGB (red green blue) :
#rrggbb (например, #00cc00)
rgb(x,x,x) -- где "x" число от 0 до 255 (например, rgb(0,204,0))
#rgb (например, #0c0)
rgb (x%,x%,x%) -- где "x%" число от 0.0 до 100.0 (например, 0%,80%,0%)
Все примеры отображают один и тот же цвет
Ссылки
Синтаксис: "URL", потом в скобках приводится ссылка. Ссылку также можно, помимо скобок, заключить в одинарные или двойные кавычки (без пропусков)
ПРИМЕР:
URL('cool.gif')
Под MSIE и Нетскейпом подразумиваются Microsoft Internet Explorer 4.0+ и Netscape Navigator 4.0+ соответственно.

Ключевые слова: *CSS(каскадная таблица стилей), внутренние таблицы стилей, глобальные таблицы стилей, связанные таблицы стилей, селектор, CLASS, ID, идентификатор, псевдокласс, псевдоэлемент, ссылки.*

Контрольные вопросы

1. Что есть CSS?
2. Для чего нужны внутренние таблицы стилей, глобальные таблицы стилей и связанные таблицы стилей?
3. Что создают с помощью ID?
4. Что из себя представляют псевдоклассы и псевдоэлементы?
5. Что такое Anchor Pseudo Classes?

ГЛАВА 10. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ JAVASCRIPT

Элементы языка JavaScript

JavaScript позволяет "оживить" веб-страницу. Это реализуется путем добавления к статическому описанию фрагмент исполняемого кода. JavaScript-сценарий может взаимодействовать с любыми компонентами HTML-документа и реагировать на изменение их состояния.

JavaScript не является строго типизированным языком, в переменных могут храниться практически любые типы данных.

Как и программа на языке Java, сценарий JavaScript выполняется под управлением интерпретатора. Однако если Java-приложение или Java-аплет компилируется в байтовый код, то сценарий JavaScript интерпретируется на уровне исходного текста.

Следует отметить, что языковые конструкции JavaScript совпадают с соответствующими средствами C++ и Java.

Структура сценария

Сценарием JavaScript считается фрагмент кода, расположенный между дескрипторами <SCRIPT> и </SCRIPT> :

Текст HTML-документа

<SCRIPT>

Код сценария

</SCRIPT>

Текст HTML-документа

Переменные

В сценариях JavaScript переменные могут хранить данные любых типов: числа, строки текста, логические значения, ссылки на объекты, а также специальные величины, например "нулевое" значение null или значение NaN, которое сообщает о недопустимости операции.

Переменная в языке JavaScript объявляется с помощью ключевого слова var. Так, например, выражение

var selected = "first item";

создает переменную с именем select и присваивает ей в качестве значения строку символов "first item". Переменные могут объявляться также автоматически. Это происходит при присвоении значения переменной, не встречавшейся ранее в данном сценарии. Так, в следующем примере создается переменная с именем rating, которой присваивается числовое значение, равное 512.5:

rating = 512.5;

Объекты

В языке JavaScript не предусмотрены средства для работы с классами в том виде, в котором они реализованы в C++ или Java. Разработчик сценария не может создать подкласс на основе существующего класса, переопределить метод или выполнить какую-либо другую операцию с классом. Сценарию, написанному на языке JavaScript, в основном доступны лишь готовые

объекты. Построение нового объекта приходится выполнять лишь в редких случаях.

Объекты содержат свойства (свойства объектов можно сравнить с переменными) и методы. Объекты, а также их свойства и методы идентифицируются именами. Объектами являются формы, изображения, гипертекстовые ссылки и другие компоненты веб-страницы, HTML-документ, отображаемый в окне браузера, окно браузера и даже сам браузер. В процессе работы JavaScript сценарий обращается к этим объектам, получает информацию и управляет ими.

Кроме того, разработчику сценария на языке JavaScript доступны объекты, не связанные непосредственно с HTML-документом. Их называют предопределенными, или независимыми объектами. С помощью этих объектов можно реализовать массив, описать дату и время, выполнить математические вычисления и решить некоторые другие задачи.

Первый объект, с которым необходимо познакомиться, чтобы написать простейший сценарий, - это объект document, который описывает HTML документ, отображаемый в окне браузера. Ниже приведен исходный текст веб-страницы, содержащей сценарий, действия которого сводятся к выводу строки текста в окне браузера.

```
<html>
<head>
<titlе>первый сценарий javascript</title></head>
<body>
<script language="javascript">
document.write("проверка сценария javascript");
</script>
</body>
</html>
```

Имена чувствительны к регистрам символов, и если вы попытаетесь обратиться к текущему документу по имени Document, интерпретатор JavaScript отобразит сообщение об ошибке.

Основное назначение сценариев JavaScript – создавать динамически изменяющиеся объекты, корректировать содержимое HTML-документов в зависимости от особенностей окружения, осуществлять взаимодействие с пользователем и т.д.

Операции

Набор операторов в JavaScript, их назначение и правила использования в основном совпадают с принятыми в языке C++. Исключением является операция задаваемая символом "+".

В JavaScript символ "+" определяет как суммирование числовых значений, так и конкатенацию строк.

Так, например, в результате вычисления выражения

sum = 47 + 21;

переменной sum будет присвоено значение 68, а после выполнения операции

sum = "строка 1 " + "строка 2";

в переменную sum будет записана последовательность символов "строка 1 строка 2".

Рассмотрим еще один пример:

```
<html>
<body>
<H2>Числа и строки</H2><BR>
<SCRIPT LANGUAGE="JavaScript">
var a = 3;
var b = 8;
var c = " попугаев ";
document.write("a+b="); document.write(a + b); document.write("<BR>");
document.write( "a + c = "); document.write(a+c);
document.write("<BR>");
document.write("c + a = "); document.write (c + a);
document.write ("<BR>");
document.write ("a + b + c = "); document.write(a + b + c);
document.write("<BR>");
document.write("c + a + b = "); document.write(c + a + b);
document.write("<BR>");
</script>
</body>
</html>
```

В окне браузера приведенный выше HTML-код выглядит так, как показано на рис.

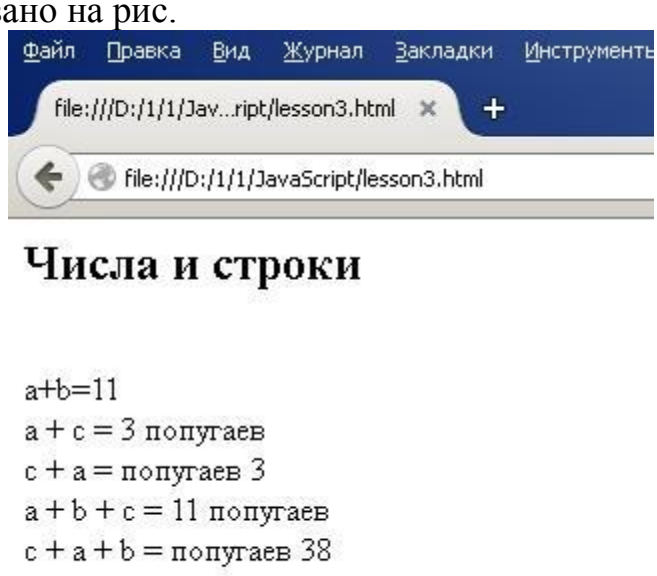


Рис.10.1. Скриншот веб-страницы с JavaScript-сценарием

Первая строка отображает результат суммирования двух числовых значений, вторая и третья - результат конкатенации строки и символьного представления числа. Если операция суммирования чисел предшествует конкатенации, JavaScript вычисляет сумму чисел, представляет ее в символьном виде, затем производит конкатенацию двух строк. Если же

первой в выражении указана операция конкатенации, то JavaScript сначала преобразует числовые значения в символьный вид, а затем выполняет конкатенацию строк.

Управляющие конструкции

Управляющие конструкции, используемые в языке C++, в основном применимы и в сценариях JavaScript.

В JavaScript дополнительно определены языковые конструкции, отсутствующие в C++, а именно: операторы for...in и with.

В примере 1 с помощью оператора цикла на веб-странице формируется таблица умножения чисел.

Пример 1.

```
<html>
<body>
<table>
<script language="JavaScript">
  document.write("<tr><td> </td>");
  for (i = 1; i < 10; i++) document.write("<td>"+i+" </td>");
  document.write("</tr>");
  for (i = 1; i < 10; i++) {
    document.write("<tr><td>" + i + " </td>");
    for (j = 1; j < 10; j++) {
      document.write("<td  bgcolor='#00ffa0'>" + (i*j) + " </td>");
    }
    document.write("</tr>");
  }
</script>
</table>
</body>
</html>
```

Отдельного внимания заслуживает оператор new. Несмотря на то, что большинство объектов уже созданы браузером и доступны сценарию, в некоторых случаях приходится создавать объекты в процессе работы. Это относится к предопределенным объектам и объектам, определяемым разработчиком сценария. Для создания объекта используется оператор new, который вызывается следующим образом:

переменная = new   тип_ объекта (параметры)

Функции

Формат объявления функции выглядит следующим образом:

function имя функции ([параметры]) тело функции

Объявление функции начинается с ключевого слова function. Так же, как и в языке C для идентификации функции используется имя, при вызове функции могут передаваться параметры, а по окончании выполнения возвращаться значение. Однако, в отличие от C, тип возвращаемого значения и типы параметров не задаются. Ниже показаны два способа вызова функции

- имя_функции ([параметры]);

- переменная = имя функции ([параметры]);

Во втором случае значение, возвращаемое функцией, присваивается указанной переменной.

Область видимости переменных

:Работа с переменными в теле функции подчиняется следующим правилам.

- Если переменная объявлена с помощью ключевого слова var, доступ к ней осуществляется по правилам, подобным тем, которые используются в языке C.
- Переменная, объявленная внутри функции, считается локальной. Область видимости такой переменной ограничивается телом функции, в которой она объявлена.
- Переменная, объявленная вне функции, считается глобальной. К ней можно обращаться из любой точки сценария.
- Если локальная и глобальная переменные имеют одинаковые имена, то в теле функции локальная переменная "маскирует" глобальную.
- Если переменная создается автоматически, т.е. если она не объявлена с помощью ключевого слова var, но присутствует в левой части оператора прямого присваивания, то она считается глобальной и становится доступной из любой точки сценария.

Диалоговые элементы

В JavaScript поддерживается работа со следующими диалоговыми элементами интерфейса:

1. **Alert**. Применяется для уведомления пользователя, работающего с веб-браузером.

Синтаксис:

alert("сообщение");

2. **Confirm.** Применяется для выбора пользователем одного из двух вариантов ответа "Да/Нет". Соответственно Confirm возвращает значение true/false.

Синтаксис:

confirm("вопрос");

3. **Prompt.** Применяется для ввода пользователем значения. При нажатии "OK" возвращается введенное значение, в случае "Cancel" возвращается значение null.

Синтаксис:

prompt("вопрос/запрос","значение по умолчанию");

Ниже приводится код веб-страницы, в которой пользователь имеет возможность выбрать цвет текста с помощью диалогового элемента

Пример 2

```
<html>
<body>
// здесь будет отображаться текст
<div id="c" style="color:blue">Вы выбрали цвет текста: черный</div>
<script language="JavaScript">
```

```
// пользователь выбирает цвет текста
  var tcolor = prompt("Выберите цвет текста: red, blue, green, yellow, black","black");
// задается текст
  document.getElementById("c").innerHTML = "Вы выбрали цвет текста: " + tcolor;
// задается цвет текста
  document.getElementById("c").style.color = tcolor;
</script>
</body>
</html>
```

Обработка событий в JavaScript

Популярность JavaScript во многом обусловлена именно тем, что написанный на нем сценарий может реагировать на действия пользователя и другие внешние события. Каждое из событий связано с тем или иным объектом: формой, гипертекстовой ссылкой или даже с окном, содержащим текущий документ.

В качестве примеров внешних событий, на которые могут реагировать объекты JavaScript, можно привести следующие.

- окончание загрузки документа в окно (или окончание загрузки документов во все фреймы окна). Это событие связано с объектом window;
- щелчок мышью на объекте. Это событие может быть связано с интерактивным элементом формы или с гипертекстовой ссылкой;
- получение объектом фокуса ввода. Это событие может быть связано с объектами типа Text, Password и с другими интерактивными элементами;
- передача на сервер данных, введенных пользователем с помощью интерактивных элементов. Связывается с формой.

Обработка события производится с помощью специально предназначенного для этого фрагмента кода, называемого обработчиком события. Для каждого события JavaScript предоставляет свой обработчик. Однако при построении сценария можно создавать собственный обработчик события и использовать его вместо обработчика, заданного по умолчанию.

Имя обработчика определяет, какое событие он должен обрабатывать. Так, для того чтобы сценарий нужным образом отреагировал на щелчок мышью, используется обработчик с именем onClick, для обработки события, заключающегося в получении фокуса ввода, – обработчик onFocus.

Для того чтобы указать интерпретатору JavaScript на то, что обработкой события должен заниматься обработчик, необходимо включить в HTML-дескриптор следующее выражение:

имя_обработчика="команды_обработчика"

Это выражение включается в тэг, описывающий объект, с которым связано событие.

Например, если необходимо обработать событие, заключающееся в получении фокуса полем ввода, дескриптор, описывающий этот интерактивный элемент, должен иметь примерно следующий вид:

<input type="text" name="Inform" onFocus="handleFocus();">

Имя обработчика является одним из атрибутов HTML-дескриптора, а команды, предназначенные для обработки события, выступают в роли значения этого атрибута. В данном случае обработка события производится в теле функции handleFocus(). В принципе, обработчиком может быть не только функция, но и любая последовательность команд JavaScript в виде составного оператора.

Следующий пример демонстрирует обработку события, связанного с наведением курсора мыши на гиперссылку:

<img src="my.gif" width="50" height="50">

Ниже приводится полный текст HTML документа с JavaScript сценарием, в котором обрабатывается событие нажатия кнопки мыши, и определяется, какая именно из них была нажата:

Пример 3.

```
<html>
<head>
<script language = "javascript">
function whichButton(event) {
if (event.button == 2) {
  alert("Вы щелкнули правой кнопкой мыши!");
  }
else {
  alert("Вы щелкнули левой кнопкой мыши!");
  }
}
</script>
</head>
<body onmousedown="whichButton(event)">
<p>Щелкните любой кнопкой мыши в любом месте документа</p>
</body>
</html>
```

Таким образом, для того чтобы обработать какое-либо стандартное событие в браузере, необходимо добавить в подходящий HTML тэг атрибут, соответствующий этому событию, указав в качестве значения атрибута имя JavaScript функции. Список атрибутов, которые определены для HTML тэгов приводится ниже:

IE: Internet Explorer, F: Firefox, O: Opera, W3C: стандарт

Атрибут	**Описание**	**IE**	**F**	**O**	W3C

onabort	Прерванная загрузка изображения	4	1	9	Да
onblur	утрата фокуса элементом	3	1	9	Да
onchange	Изменение содержимого в поле ввода	3	1	9	Да
onclick	Щелчок мыши на объекте	3	1	9	Да
ondblclick	Двойной щелчок мыши на объекте	4	1	9	Да
onerror	Ошибка при загрузке изображения или документа	4	1		Да
onfocus	Получение фокуса элементом	3	1	9	Да
onkeydown	Нажатие клавиши	3	1	Нет	Да
onkeypress	Клавиша нажата	3	1	9	Да
onkeyup	Отжатие клавиши	3	1	9	Да
onload	Завершение загрузки страницы или изображения	3	1	9	Да
onmousedown	Нажатие кнопки мыши	4	1	9	Да
onmousemove	Перемещение курсора мыши	3	1	9	Да
onmouseout	Смещение курсора мыши с объекта	4	1	9	Да
onmouseover	Наведение курсора мыши на объект	3	1	9	Да
onmouseup	Отжатие кнопки мыши	4	1	9	Да
onreset	Кнопка "Reset" нажата	4	1	9	Да
onresize	Изменение размера окна	4	1	9	Да
onselect	Выделение текста	3	1	9	Да
onsubmit	Кнопка "Submit" нажата	3	1	9	Да
onunload	Уход с веб-страницы	3	1	9	Да

Использование регулярных выражений в JavaScript

При поиске по тексту можно использовать шаблон, описывающий подстроку. В JavaScript такой шаблон может быть описан с помощью объекта RegExp. В простейшем случае такой шаблон описывает отдельный символ, однако имеет смысл его использовать для регулярных выражений.

Следующий ниже код описывает RegExp объект с именем pttn, содержащий регулярное выражение, описывающее целое десятичное число:

```
var pattn = new RegExp("/[0-9]+/");
```

Объект RegExp имеет 3 встроенных метода: test(), exec() и compile().

- Метод test() выполняет поиск по шаблону:

```
var pattn = new RegExp("[0-9]+");
document.write(pattn.test("38 попугаев"));
```

Результат:

```
true
```

- Метод exec() выполняет поиск подстроки по шаблону и возвращает найденные соответствия; если соответствий нет, возвращается значение null:

```
var pattn=new RegExp("[0-9]+");
document.write(pattn.exec("38 попугаев"));
```

Результат:

38

Если необходимо найти все соответствия, то при вызове конструктора RegExp следует указать дополнительный параметр "g", указывающий на необходимость глобального поиска.

Пример 8

```
var pattn = new RegExp("[0-9]+", "g");
do {
result  = pattn.exec("1 попугай, 2 попугая,…, 38 попугаев");
document.write(" " + result);
}
while (result != null)
```

Результат:

1 2 38 null

- Метод compile() применяется для изменения ранее созданного шаблона.

Пример 4

```
var pattn = new RegExp("[0-5]+");
document.write(pattn.exec("38 попугаев"));

pattn.compile("[6-9]+");
document.write(";" + pattn.exec("38 попугаев"));
```

Результат:

3;8

Ключевые слова: *JavaScript, переменная, функции, сценарий, объект, события, операция, управляющие конструкции, диалоговый элемент, регулярные выражения.*

Контрольные вопросы

1. Элементы языка JavaScript.
2. Что может храниться в сценариях JavaScript?
3. Какие свойства содержат объекты?
4. Основное назначение сценариев JavaScript.
5. С какого ключевого слова начинается объявление функции?
6. Диалоговые элементы интерфейса. Расскажите о них.
7. Что необходимо сделать для того чтобы обработать какое-либо стандартное событие в браузере?

ГЛАВА 11. ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКОВ JAVASCRIPT

Поскольку JavaScript развивается в течение около 20 лет (он появился в 1995 году), очевидно, что за это время накопилось множество полезного и эффективного кода, который был объединен в самые разнообразные библиотеки. Некоторые из этих библиотек мы рассмотрим в настоящем пособии, а с библиотекой jQuery поработаем более подробно и выполним несколько поучительных примеров. Поскольку применение этих библиотек обусловлено средой, то их иногда называют фреймворками. Среди известных JavaScript библиотек следует отметить Adobe life, Dojo, Toolkit, Extjs,jQuery, Mootools,Prototype, Qooxdoo, Underscore. Ниже приведем краткие характеристики этих библиотек, а потом рассмотрим более подробно jQuery

Adobe life

Dojo

Dojo (додзё) — свободная модульная библиотека JavaScript. Разработана с целью упростить ускоренную разработку основанных на JavaScript или AJAX приложений и сайтов. Разработка библиотеки была начата Алексом Русселом в 2004 году. Библиотека находится под двойной лицензией: BSD License и Academic Free License. Dojo Foundation — некоммерческая организация, созданная для продвижения Dojo. Dojo используется в Zend Framework, начиная с версии 1.6.0.

Toolkit

Extjs

Ext JS (Sencha Ext JS) — библиотека JavaScript для разработки веб-приложений и пользовательских интерфейсов, изначально задуманная как расширенная версия Yahoo! UI Library, однако преобразовавшаяся затем в отдельный фреймворк. До версии 4.0 использовала адаптеры для доступа к библиотекам Yahoo! UI Library, jQuery или Prototype/script.aculo.us, начиная с 4-й версии адаптеры отсутствуют. Поддерживает технологию AJAX, анимацию, работу с DOM, реализацию таблиц, вкладок, обработку событий и все остальные новшества Web 2.0.

С версии 2.1 библиотека Ext JS предусматривает двойное лицензирование: может быть поставлена по лицензии GPL v3, либо по коммерческой лицензии компании Sencha.

Начиная с версии Ext JS 3.0 библиотека разбивается на две части: Ext Core (набор JavaScript-функций, позволяющий создавать динамические веб-страницы с унификацией обработки в различных браузерах и распространяемый по MIT-лицензии) и Ext JS (собственно набор виджетов для создания пользовательских интерфейсов с двойным лицензированием по GPL v3 или по коммерческой лицензии).

Компания, разрабатывающая и поддерживающая фреймворк Ext JS, изначально носила наименование Ext, LLC. 14 июня 2010 года разработчики jQTouch и Raphaël присоединились к компании Ext LLC, и компания сменила

наименование на Sencha, Inc.[1], а Ext JS, сохранив своё название, стал одним из продуктов обновлённой компании.

Для разработки мобильных веб-приложений предназначен специальный фреймворк Sencha Touch, поддерживающий множество мобильных операционных систем.

jQuery

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX. Сейчас разработка jQuery ведется командой jQuery во главе с Джоном Резигом.

Mootools

MooTools — это свободный JavaScript-фреймворк для разработки кроссбраузерных веб-приложений и веб-сервисов.

MooTools является модульным, объектно-ориентированным фреймворком, созданным для помощи разработчикам JavaScript.

MooTools совместим и протестирован с браузерами: Safari 2+, Internet Explorer 6+, Firefox 2+ (и другими, основанными на движке Gecko), Opera 9+.

Фреймворк MooTools используется в CMS Contao, Joomla 1.5+, ZoneMinder, MODx.

Prototype

Prototype — JavaScript фреймворк, упрощающий работу с Ajax и некоторыми другими функциями. Несмотря на его доступность в виде отдельной библиотеки, он обычно используется программистами вместе с Ruby on Rails, Tapestry, script.aculo.us и Rico.

Заявлено, что данный фреймворк совместим со следующими браузерами: Internet Explorer (Windows) 6.0+, Mozilla Firefox 1.5+, Apple Safari 2.0.4+ и Opera 9.25+, Google Chrome 1.0+. Поддержка данных браузеров подразумевает, что фреймворк совместим также с Camino, Konqueror, IceWeasel, Netscape 7+, SeaMonkey, Яндекс.Браузер и др., которые принадлежат этим же семействам.

В Prototype присутствуют самые разные способы упрощения создания JavaScript приложений, от сокращённого вызова некоторых функций языка до сложных методов обращения к XMLHttpRequest.

Qooxdoo

Основная задача фреймворка — предоставить возможность проектировать многофункциональные кросс-браузерные веб-приложения. При этом, как сказано на qooxdoo.org, знания HTML, CSS, DOM не требуются.

Каких-то принципиальных новшеств в новом релизе лично я для себя не отметил. Хотя, безусловно, есть некоторые полезные плюшки, такие как Tri-state CheckBox.

В целом библиотека очень функциональна, хорошо продумана, реализована и задокументирована. Ниже приводятся краткие характеристики библиотеки.

Моделирование GUI

- Большое количество как типичных, так и специфичных элементов управления
- Проработанная система событий, как типичных (события мыши, клавиатуры) так и специфичных для определенных элементов и виджетов (выделение строки в таблице, изменение модели данных)
- DataBinding — привязка модели данных к элементу управления
- Визуальные темы
- Drag&Drop интерфейс

Транспортный уровень

- Возможна как работа на низком уровне (непосредственный AJAX) так и посредством RPC
- Предлагаются реализации RPC-серверов на Java, PHP, Python, Perl
- Отдельные коммуникационные решения для некоторых виджетов

Разработка

- «Компиляция» в debug и build версии
- Поддержка интернационализации
- Автоматизация создания документации
- Unit-тестирование

Лицензия

- LGPL (Lesser General Public License)
- EPL (Eclipse Public License)

Underscore

Underscore — библиотека JavaScript, реализующая дополнительную функциональность для работы с массивами, объектами и функциями, изначально отсутствующую в javascript, но имеющую аналоги в других языках. Библиотека умеет делегировать вызовы, если какая-то функциональность реализована разработчиками браузеров.

Списки функций

- Утилиты: noConflict, identity, times, mixin, uniqueId, escape, template, chain, value, random
- Функции: bind, bindAll, memoize, delay, defer, throttle, debounce, once, after, wrap, compose
- Массивы: first, initial, last, rest, compact, flatten, without, union, intersection, difference, uniq, zip, indexOf, lastIndexOf, range
- Коллекции: each, map, reduce, reduceRight, find, filter, reject, all, any, include, invoke, pluck, max, min, sortBy, groupBy, sortedIndex, shuffle, toArray, size, countBy, where

- Объекты: keys, values, functions, extend, defaults, clone, tap, has, isEqual, isEmpty, isElement, isArray, isArguments, isFunction, isString, isNumber, isBoolean, isDate, isRegExp, isNaN, isNull, isUndefined, pairs, invert, omit

Что такое jQuery?

jQuery - это библиотека, которая значительно упрощает и ускоряет написание JavaScript кода.

Девиз jQuery "write less, do more" (пиши меньше, делай больше) отражает ее главное предназначение.

jQuery позволяет создавать анимацию, обработчики событий, значительно облегчает выбор элементов в DOM и создание AJAX запросов.

Данная библиотека работает со всеми браузерами (IE 6.0+, FF 2.0+, Safari 3.0+, Opera 9.0+, Chrome). Это значит, что Вам больше не нужно будет беспокоиться о кроссбраузерной совместимости JavaScript кода.

Для jQuery написано огромное количество плагинов, которые позволяют расширить ее возможности еще больше.

Пример использования jQuery

```
$(document).ready(function(){

  $(":button").click(function(){
    $(":button").hide();
    $("#wrap1").addClass("add");
    $("#wrap1").animate({height:280},2000);
    $("#wrap1").animate({width:400},2000);
    $("#wrap1").animate({padding:20},2000,function(){
      $("#text1").hide(2000,function(){$("#text2").show(2000);});
    });
  });

});
```

Добавление jQuery на страницы

Для того, чтобы начать использовать jQuery необходимо:

1. Скачать ее с официального сайта. Существуют две версии jQuery: для использования в готовых приложениях (production) и для разработки (development). Версия для разработки содержит комментарии и структурированный код. В сокращенной версии нет комментариев и код в ней не структурирован зато она занимает меньше места и поэтому страницы с ней будут загружаться быстрее. После того, как Вы выберите подходящую версию нажмите на кнопку "Download (jQuery)"
2. Добавить ее на страницу. Для этого следующий код должен быть добавлен на страницу в секцию head:

```
<script type="text/javascript" src="путь_к_скачанному_файлу/jquery.js">
</script>
```

Для тех кто по каким-либо причинам не может (или не хочет) скачивать jQuery предусмотрен альтернативный способ удаленного использования библиотеки предоставленный компанией Google.

Для того, чтобы использовать jQuery удаленно просто добавьте на страницу в секцию head следующий код:

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.5/jquery.min.js">
</script>
```

Сделайте сами

Обратите внимание: при выполнении этого и следующих упражнений рекомендуем Вам выключить отладчики (такие как Firebug), которые могут выдавать подсказки. В ходе выполнения заданий Вы сами должны находить и исправлять ошибки в учебных целях.

Задание 1 на странице задан jQuery код, но он не работает потому, что к ней не подключен необходимый файл библиотеки jquery.js. Подключите jquery.js удаленно (воспользовавшись услугой Google), чтобы "починить" код.

Пример

```
$(document).ready(function(){

  $("input").click(function(event){
    alert("Поздравляем! Вы починили код!");
  });

});
```

Команды jQuery

Код jQuery как и код JavaScript состоит из последовательно идущих команд. Команды являются основной структурной единицей jQuery.

Стандартный синтаксис jQuery команд:

$(селектор).метод();

1. **Знак $** сообщает, что символы идущие после него являются jQuery кодом;
2. **Селектор** позволяет выбрать элемент на странице;
3. **Метод** задает действие, которое необходимо совершить над выбранным элементом. Методы в jQuery разделяются на следующие группы:
   - Методы для манипулирования DOM;
   - Методы для оформления элементов;
   - Методы для создания AJAX запросов;
   - Методы для создания эффектов;
   - Методы для привязки обработчиков событий.

Символ $

jQuery можно комбинировать с обычным JavaScript.

Если строка начинается с $ – это jQuery, если $ вначале строки отсутствует – это строка JavaScript кода.

```
function start() {
```

```
//Скроем абзац с id="hid" с помощью jQuery
  $("p#hid").hide();
//Изменим содержимое абзаца с id="change"
  document.getElementById("change").innerHTML="Данное содержимое было изменено с помощью JavaScript.";
}
```

Некоторые JavaScript библиотеки тоже используют $ для обозначения своего кода. Для того, чтобы избежать конфликта имен между ними в jQuery предусмотрена команда noConflict(). С помощью команды noConflict() Вы можете заменить знак $ на любой другой.

Синтаксис:

```
var произвольное_имя=jQuery.noConflict();
```

Пример

```
//Изменим стандартный символ $ на nsign
var nsign=jQuery.noConflict();
//Изменим цвет абзаца используя nsign вместо $
nsign(document).ready(function(){

  nsign("p").css("color","green");

});
```

Селекторы jQuery

С помощью селекторов Вы можете выбирать элементы на странице для применения к ним определенных действий.

Ниже располагается таблица с различными примерами использования селекторов для выбора желаемых элементов:

Пример	**Результат**
$("p")	Будут выбраны все элементы p, которые находятся на странице.
$(".par")	Будут выбраны все элементы на странице с class="par".
$("#par")	Будет выбран первый элемент на странице с id="par".
$(this)	Позволяет выбрать текущий HTML элемент. Щелкните на $(this) слева, чтобы посмотреть пример использования данного селектора в онлайн редакторе.
$("p.par")	Будут выбраны все элементы p на странице с class="par".
$("p#par")	Будут выбраны все элементы p на странице с id="par".
$(".par,.header,#heat")	Будут выбраны все элементы на странице со значениями атрибутов class="par", class="header" и id='heat'.
$("[src]")	Будут выбраны все элементы на странице, имеющие атрибут src.
$("[src='значение']")	Будут выбраны все элементы со значениям атрибута

	src="значение".
$("[src$='.gif']")	Будут выбраны все элементы со значениями атрибута src заканчивающимися на .gif.
$("div#wrap .par1")	Будут выбраны все элементы с class=par1, которые находятся внутри элементов div с id=wrap.
$(":input")	Будут выбраны все input элементы на странице.
$(":тип")	Будут выбраны все input элементы с <input type='тип' />. Например :button выберет все <input type='button' /> элементы, :text выберет все <input type='text' /> элементы и т.д.

Пример

```
$(document).ready(function(){
//Установим размер шрифта всех абзацев равным 20 пикселям
  $("p").css("fontSize","20px");
//Изменим на зеленый цвет шрифта элемента с id=el2
  $("#el2").css("color","green");
//Изменим на красный цвет шрифта элемента с class=el3
  $(".el3").css("color","red");
//Сделаем жирным шрифт элементов с id=el2 и class=el3
  $("#el2,.el3").css("fontWeight","bold");
//Изменим на синий цвет текста кнопки
  $(":input").css("color","blue");
//Установим размер шрифта всех элементов имеющих атрибут href равным 20 пикселям
  $("[href]").css("fontSize","20px");
//Изменим на зеленый цвет ссылки на www.wisdomweb.ru
  $("[href='http://www.wisdomweb.ru/']").css("color","green");
});
```

Полный список селекторов jQuery Вы можете найти в нашем jQuery справочнике.

Предотвращение преждевременного выполнения кода

Из учебника по JavaScript Вы наверно помните, что выполнение кода до полной загрузки документа часто приводит к ошибкам.

Дело в том, что скрипт может обратится к еще не загруженному содержимому, а это всегда приводит к ошибкам или неожиданным результатам.

Для того, чтобы избежать этого мы часто помещали код в функцию, которая начинала выполнение только после полной загрузки документа.

Предотвращение преждевременного выполнения кода в JavaScript:

```
<html>
<head>
<script type="text/javascript">
function start() {
  Код который будет выполнен после полной загрузки документа
```

```
}
</script>
</head>
<body onload="start()">
</body>
</html>
```

В jQuery можно избавиться от преждевременного выполнения кода следующими способами:

Предотвращение преждевременного выполнения кода в jQuery:

```
<script type='text/javascript'>
//Первый способ:
$(document).ready(function(){
  Код который будет выполнен после полной загрузки документа
});

//Второй способ:
$().ready(function(){
  Код который будет выполнен после полной загрузки документа
});

//Третий способ:
$(function(){
  Код который будет выполнен после полной загрузки документа
});
</script>
```

Существует еще один альтернативный способ также помогающий избежать преждевременное выполнение JavaScript и jQuery кода и позволяющий также ускорить загрузку страниц (благодарим за напоминание о нем пользователя Ghringo Americano).

Необходимо помещать код в самый конец тела документа (т.е. перед </body>) в данном случае интерпретатор JavaScript встроенный в браузер начнет разбирать код только после загрузки документа. В предыдущем же способе загрузка скриптов происходит одновременно с загрузкой документа, а выполняется этот код после загрузки документа.

```
<body>
<p>Содержимое тела документа</p>
<div>Содержимое тела документа</div>

<script type='text/javascript'>
  Код который будет выполнен после полной загрузки документа
</script>
</body>
```

Цепочки команд в jQuery

Для того, чтобы сократить размер кода Вы можете соединять команды jQuery в цепочки.

Команды в цепочке будут выполняться поочередно слева направо.

```
<script type='text/javascript'>
//Код без сокращения
$("p").css("color","green");
$("p").css("font-size","30px");
//Сокращенный код
$("p").css("color","green").css("font-size","30px");
</script>
```

Сделайте сами

Задание 1 измените цвет и размер шрифта, перечисленных ниже элементов, для того, чтобы выделить буквы кодового слова.

Измените цвет и размер шрифта:

- Элемента с id=meadow;
- Элемента с class=rainbow;
- Элемента с id=future находящегося внутри элемента с id=fut; (данный элемент необходимо выделить цветом отличным от красного);
- Элемента имеющего атрибут set;
- Элемента с атрибутом last=code.

Обратите внимание: для изменения цвета текста элемента используйте метод: css('color','новый_цвет_текста'), для изменения размера шрифта используйте метод: css('font-size','размер_шрифта_px').

Пример

```
$(document).ready(function(){

  $("div div").css("color","red");
//Пишите код здесь


});
```

Задание 2 измените оформление элементов согласно их содержимому (подробности в редакторе).

Пример

```
$(document).ready(function(){
//Пишите код здесь
});
```

Обработчики событий jQuery

Обработчики событий – это функции, код которых исполняется только после совершения определенных действий.

Обработчики событий присутствовали и в JavaScript, но jQuery облегчает их использование и расширяет их функциональность.

Примеры действий, после которых выполняются обработчики:

- Курсор мыши наведен на элемент;
- Веб-страница или картинка полностью загружена;
- Изменено содержимое поля формы;

- HTML форма отправлена;
- Нажата клавиша на клавиатуре;

Общий вид определения обработчиков jQuery:

$(селектор).обработчик_события(function(){код_обработчика_события});

Обработчики событий jQuery в действии

Код обработчика mouseover будет выполнен, когда курсор мыши будет наведен на элемент.

Код обработчика mouseout будет выполнен, когда курсор мыши будет выведен из границ элемента.

Пример

```
$(document).ready(function(){
  $("p").mouseover(function(){$("p").css("color","green")});
  $("p").mouseout(function(){$("p").css("color","black")});
});
```

Код обработчика click будет выполнен после одинарного щелчка мыши на элементе.

Код обработчика dblclick будет выполнен после двойного щелчка мыши на элементе

Пример

```
$(document).ready(function(){

  $("#but1").click(function(){alert("Вы нажали один раз на первую кнопку!");});
  $("#but2").dblclick(function(){alert("Вы нажали два раза на вторую кнопку!");});

});
```

Код обработчика focus() будет выполнен, когда элемент станет активным.

Код обработчика blur() будет выполнен, когда элемент перестанет быть активным.

Код обработчика change() будет выполнен, при изменении содержимого элемента.

Пример

```
$(document).ready(function(){

  $("#el1").focus(function(){$(this).attr("value","")});
  $("#el1").blur(function(){$(this).attr("value","Введите ФИО")});
  $("#el2").change(function(){ alert("Содержимое данного элемента было изменено.") });

});
```

С помощью jQuery Вы можете также вызывать обработчики событий привязанные к элементу.

Например: $('#test').blur() вызовет обработчик blur у элемента с id='test'.

Пример

```
$(document).ready(function(){

// Зададим обработчик события, который будет выводить сообщение при нажатии на кнопку с id=but1
  $("#but1").click(function(){alert("Вы нажали на кнопку с id=but1")});
// Вызовем обработчик click у элемента с id='but1'
  $("#but1").click();

});
```

Обратите внимание: полный список всех существующих событий в jQuery с примерами использования Вы найдете в нашем jQuery справочнике.

Объект event

Объект event содержит информацию о произошедшем событии.

Объект event создается для каждого произошедшего события, но для того, чтобы иметь возможность обращаться к его свойствам и методам его необходимо явно передать в обработчик события.

Синтаксис:

```
$(селектор).событие(function(event){код_обработчика_события});
```

После того, как объект event был передан обработчику события, Вы можете обращаться к его свойствам и методам.

В примере ниже с помощью объекта event мы будем выводить координаты, на которых находился курсор мыши во время того, как произошло событие.

Пример

```
$(document).ready(function(){

  $("#square").click(function(event){
    $("#coord").css("display","block");
    $("#x").html(event.pageX);
    $("#y").html(event.pageY);
  });

});
```

Обратите внимание: полный список всех существующих свойств и методов объекта event с примерами использования Вы найдете в нашем jQuery справочнике.

Управление обработчиками событий

Метод jQuery one() позволяет создавать обработчики, которые могут быть вызваны лишь один раз.

http://www.wisdomweb.ru/JQ/event.php

Ключевые слова:*JavaScript, фреймворк, jQuery, события, селекторы, библиотеки, моделирование, синтаксис, обработчик*

Контрольные вопросы

1.Опишите функции библиотек:
1) Dojo
2) Ext JS
3) jQuery
2.На какие части разбивается Ext JS 3.0?
3.Что такое Qooxdoo?
4.Для чего служит символ $?
5. Список функций **Underscore.**
6.В чем заключается предназначение селекторов jQuery?

ГЛАВА 12. ПРОГРАММИРОВАНИЕ НАСЫЩЕННЫХ ИНТЕРНЕТ ПРИЛОЖЕНИЙ (RIA)

Как правило, Веб-приложение – приложение, в котором клиентом выступает браузер, а сервером – веб-сервер.

Рассмотрим типы программ, обеспечивающих работу Веб и использующих HTTP-протокол.

Никакой HTTP-обмен невозможен без клиента и сервера. Однако помимо клиента и сервера в веб-сеансе могут участвовать и другие программы, которые и являются объектом веб-программирования.

Результатом работы веб-приложения является веб-страница, отображаемая в окне браузера. При этом само веб-приложение может выполняться как на компьютере клиента, так и на компьютере сервера.

Рассмотрим подробнее обе схемы.

Программы, выполняющиеся на клиент-машине

Одним из типов программ, предназначенных для выполнения на клиент-машине, являются сценарии, например, JavaScript (VBScript). Исходный текст сценария представляет собой часть веб-страницы, поэтому сценарий JavaScript передается клиенту вместе с документом, в состав которого он входит. Обрабатывая HTML-документ, браузер обнаруживает исходный текст сценария и запускает его на выполнение.

Ко всем программам, которые передаются с сервера на клиент-машины и запускаются на выполнение, предъявляется одно общее требование: эти программы должны быть лишены возможности обращаться к ресурсам компьютера, на котором они выполняются. Такое требование вполне обосновано. Ведь передача по сети и запуск Java-апплетов и JavaScript-сценариев происходит автоматически без участия пользователя, поэтому работа этих программ должна быть абсолютно безопасной для компьютера. Другими словами, языки, предназначенные для создания программ, выполняющихся на клиент-машине, должны быть абсолютно непригодны для написания вирусов и подобных программ.

Программы, выполняющиеся на сервере

Код программы, работающей на сервере, не передается клиенту. При получении от клиента специального запроса, предполагающего выполнение такой программы, сервер запускает ее и передает параметры, входящие в состав запроса. Средства для генерации подобного запроса обычно входят в состав HTML-документа.

Результаты своей работы программа оформляет в виде HTML-документа и передает их веб-серверу, а последний, в свою очередь, дополняет полученные данные HTTP-заголовком и передает их клиенту. Взаимодействие клиента и сервера в этом случае показано на рисунке 12.1.

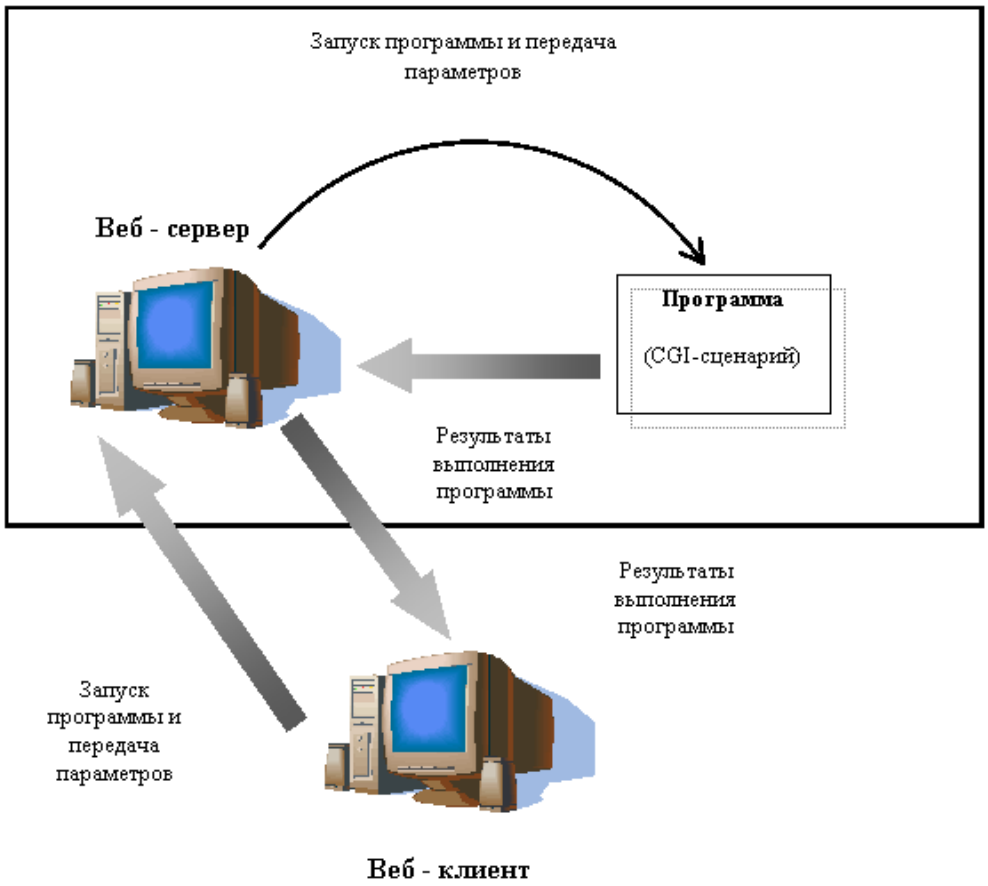


Рис. 12.1. Взаимодействие клиента с программой, выполняющейся на сервере

Насыщенные интернет-приложения

Насыщенное интернет-приложение (Rich Internet application) – еще один подход, который заключается в использовании Adobe Flash или Java-апплетов для полной или частичной реализации пользовательского интерфейса, поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов).

Возникновение данного подхода обусловлено тем, что в рамках веб-приложений с "тонким" клиентом взаимодействие пользователя с приложением реализуется в существенной степени через сервер, что требует отправки данных на сервер, получение ответа от сервера и перезагрузку страницы на стороне клиента.

При использовании Java-апплетов в состав HTML-документа включается специальный дескриптор, описывающий расположение файла, содержащего код апплета, на сервере. После того как клиент получает HTML-код документа, включающего апплет, он генерирует дополнительный запрос серверу. После того как сервер пересылает клиенту код апплета, сам апплет запускается на выполнение. Взаимодействие между клиентом и сервером при получении апплета показано на рисунке 12.2.

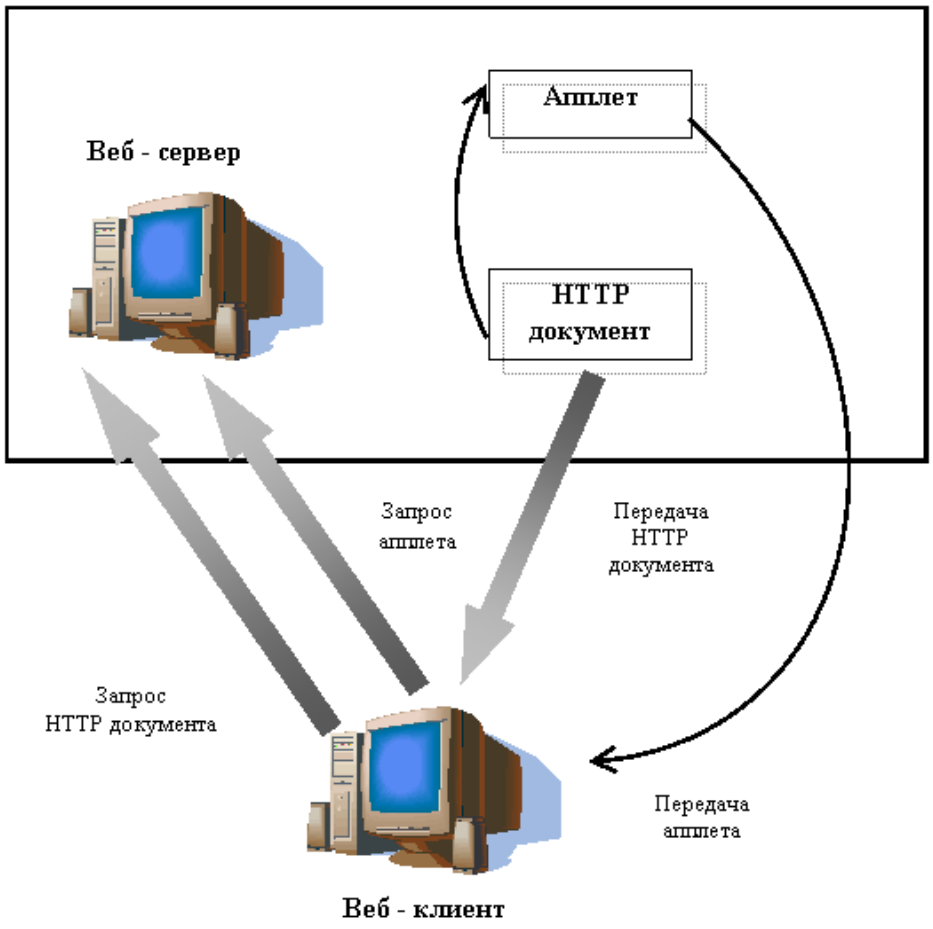


Рис. 12.1. Передача клиенту Java-апплета

При использовании насыщенных интернет-приложений приходится сталкиваться со следующими проблемами:

- необходимость обеспечения безопасной среды выполнения ("песочница");
- для исполнения кода должно быть разрешено исполнение сценариев;
- потеря в производительности (т.к. выполяется на клиентской стороне);
- требуется много времени на загрузку;

Для разработки насыщенных интернет-приложений используются пакеты HTML5, CSS3, JavaScript, Curl, JavaFX, Adobe Flex и Microsoft Silverlight.

Java-апплеты

Java-апплет – это программа, написанная на языке Java и откомпилированная в байт-код. Выполнется в браузере с использованием виртуальной Java-машины (JVM). Апплеты используются для предоставления интерактивных возможностей веб-приложений, которые не возможны в HTML. Так как байт-код Java платформо-независим, то Java-апплеты могут выполняться браузерами на многих операционных платформах.

Java-сервлеты являются серверными приложениями, но они отличаются от апплетов языком, функциями и другими характеристиками.

Предназначены Java-апплеты для выполнения в безопасной среде с целью предотвращения их доступа к локальным ресурсам клиентского компьютера.

Код апплета загружается с веб-сервера, и браузер

- либо вставляет апплет в веб-страницу;
- либо открывает отдельное окно с собственным пользовательским интерфейсом апплета.

Апплет может быть внедрен в веб-страницу с помощью использования HTML тэга <applet>, или (что рекомендуется) тэга <object>.

Можно назвать следующие преимущества Java-апплетов:

- работают практически на большинстве операционных платформ;
- поддерживаются большинством браузеров;
- кэшируются в большинстве браузеров, что существенно ускоряет их загрузку при возвращении на веб-страницу;
- после первого запуска апплета, когда Java-машина уже выполняется и быстро запускается, выполнение апплетов происходит существенно быстрее;
- загружаются со скоростью, сопоставимой с програмами на других компилируемых языках, например C++, но во много раз быстрее чем на JavaScript.

При этом у Java-апплетов имеются и недостатки:

- требуется установка Java-расширения, которые доступны по умолчанию не во всех браузерах;
- проблемы реализации Java-расширений для 64-разрядных процессоров;
- не могут запускаться до первой загрузки виртуальной Java-машины, что может занимать значительное время;
- разработка пользовательского интерфейса с использованием апплетов является более сложной задачей по сравнению с HTML;
- не имеют прямого доступа к локальным ресурсам клиентского компьютера;
- некоторые апплеты привязаны к использованию определенной среды времени выполнения Java (JRE).

ActionScript – общая характеристика

ActionScript — объектно-ориентированный язык программирования, один из диалектов EcmaScript, который добавляет интерактивность, обработку данных и многое другое в содержимое Flash-приложений. ActionScript исполняется виртуальной машиной (ActionScript Virtual Machine), которая является составной частью приложения Flash Player. ActionScript компилируется в байткод, который включается в SWF -файл.

SWF -файлы исполняются Flash Player. Сам Flash Player существует в виде плагина к веб-браузеру, а также как самостоятельное исполняемое приложение. Во втором случае возможно создание исполняемых exe -файлов, когда swf -файл включается во Flash Player.

С помощью ActionScript можно создавать интерактивные мультимедиа-приложения, игры, веб-сайты и многое другое.

XAML и Microsoft Silverlight

XAML (eXtensible Application Markup Language) - язык интерфейсов платформы Windows Vista.

Модель приложений Vista включает объект Application. Его набор свойств, методов и событий позволяет объединять веб-документы в связанное приложение. Объект Application контролирует выполнение программы и генерирует события для пользовательского кода. Документы приложения создаются с помощью языка XAML, который описывает, прежде всего, пользовательский интерфейс. Логика приложения управляется процедурным кодом (C#, VB и др.). XAML включает основные четыре категории элементов: панели, элементы управления, элементы, связанные с документом и графические фигуры.

Microsoft Silverlight является официальным названием основанной на XML и .NET технологии под кодовым именем WPF/E (Windows Presentation Foundation Everywhere), являющейся альтернативной Adobe Flash. Представляет собой подмножество Windows Presentation Foundation, в котором реализованы векторная графика, анимация и средства воспроизведения видео. В версии 1.1 включает в себя полную версию .NET CLR – называемую CoreCLR, что позволит разрабатывать Silverlight приложения на любом из языков .NET. Silverlight v.1.0 содержит подключаемый модуль браузера для обработки XAML и кодеки для воспроизведения мультимедийного содержимого в форматах WMV, WMA и MP3.

Microsoft Silverlight представляет браузеру внутреннюю модель DOM, управляемую из JavaScript кода. Поскольку язык XAML основан на XML, то документ, определяющий загружаемый клиенту пользовательский интерфейс - текстовый и потому вполне пригоден для индексирования поисковыми системами. Используя модель DOM, JavaScript может динамически обновлять содержимое Silverlight, аналогично DHTML.

Также можно вызывать методы управления презентацией (запуска анимации или приостановки воспроизведения видео, например).

Silverlight –приложение начинается с вызова объекта Silverlight из HTML страницы, загружающего XAML файл. XAML файл содержит объект

Canvas, выступающий подложкой для других элементов. Объекты XAML способны генерировать события, перехватываемые из JavaScript.

Понятие о DOM

Объектная модель подробно рассмотрена в Главе 8. Здесь напомним основные положения, поскольку это является одим из средств насыщенных интернет приложений.

DOM (Document Object Model) – объектная модель документа. Это независящий от платформы и языка программный интерфейс, позволяющий программам получать доступ к содержимому документов, а также изменять содержимое, структуру и вид документов.

В рамках DOM любой документ представляется в виде дерева узлов. Каждый узел представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы между собой находятся в отношении "родитель-потомок".

Изначально различные браузеры имели собственные модели DOM, не совместимые с остальными. Для того, чтобы обеспечить взаимную и обратную совместимость, консорциум W3C классифицировал эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу, носящую название W3C DOM:

- Уровень 0. Включает в себя все специфические модели DOM, которые существовали до появления Уровня 1, например document.images, document.forms. Эти модели формально не являются спецификациями DOM, опубликованными W3C, а скорее отражают то, что существовало до начала процесса стандартизации.
- Уровень 1. Базовые функциональные возможности DOM (HTML и XML) в документах, такие как получение дерева узлов документа, возможность изменять и добавлять данные.
- Уровень 2. Поддержка пространства имен XML, filtered views и событий.
- Уровень 3. Состоит из шести различных спецификаций:
  - DOM Level 3 Core ;
  - DOM Level 3 Load and Save ;
  - DOM Level 3 XPath ;
  - DOM Level 3 Views and Formatting ;
  - Level 3 Requirements ;
  - DOM Level 3 Validation.

Текущим уровнем спецификаций DOM является Уровень 2, но, тем не менее, некоторые части спецификаций Уровня 3 являются рекомендуемыми W3C.

DHTML

Динамический HTML или DHTML представляет собой набор технологий, которые совместно позволяют создавать интерактивные веб-сайты на основе статического языка разметки (HTML), языка создания клиентских сценариев (JavaScript), языка описания представления документа (CSS) и документной объектной модели (DOM).

DHTML позволяет сценарным языкам изменять переменные языка описания представления документа, таким образом, изменяя вид и поведение прежде статического содержимого HTML документа уже после полной загрузки документа и в процессе просмотра его пользователем. Таким образом, динамичность, привносимая DHTML, проявляет себя в процессе просмотра страницы, но не имеет никакого отношения к генерации содержимого страницы при каждой ее загрузке.

В противоположность DHTML, динамически генерируемая страница - более широкое понятие, подразумевающее, например генерацию содержимого веб-страницы индивидуально для каждого пользователя. Это достигается созданием страниц с помощью клиентских или серверных (например, на PHP или Perl) сценариев.

Регулярные выражения

Регулярные выражения — система поиска текстовых фрагментов в электронных документах, основанная на специальной системе записи образцов для поиска. Образец, задающий правило поиска, называется " шаблоном ". Применение регулярных выражений принципиально преобразило технологии электронной обработки текстов.

Многие языки программирования уже поддерживают регулярные выражения для работы со строками либо в виде отдельных функций, либо имеют уже встроенный в их синтаксис механизм обработки регулярных выражений, например, Perl и Tcl. Популяризации понятия регулярных выражений способствовали утилиты, поставляемые в дистрибутивах Unix.

С помощью регулярных выражений можно задавать структуру искомого шаблона и его позицию внутри строки (например, в начале или в конце строки, на границе или не на границе слова).

При описании структуры шаблона используются:

- гибкая система квантификаторов (операторов повторения);
- операторы описания наборов символов и их типа (числовые, нечисловые, специальные).

Ключевые слова: *Программирование, приложение, Интернет, сервер, клиент, веб-сайт, уровень, HTML,DOM, веб-страница*

Контрольные вопросы:

1.В чем заключается подход насыщенного интернет-приложения?
2. Что такое Java-апплеты?
3.Описание схемы передачи клиенту Java-апплета
4. Дайте характеристику ActionScript
5.Опишите все уровни DOM
6. Что такое регулярные выражения?

ЧАСТЬ 3. РАЗРАБОТКА НА СТОРОНЕ СЕРВЕРА

ГЛАВА 13. ПРОГРАММИРОВАНИЕ НА СТОРОНЕ СЕРВЕРА

Для расширения возможностей клиент-серверного взаимодействия в рамках протокола HTTP помимо создания на клиентской стороне расширений стандартных возможностей, предоставляемых языками разметки и браузерами, можно также разрабатывать на стороне веб-сервера приложения, плагины и сценарии, расширяющие возможности самого веб-сервера.

Плагин (plug-in) – независимо компилируемый программный модуль, динамически подключаемый к основной программе, предназначенный для расширения или использования ее возможностей. Обычно выполняются в виде разделяемых библиотек.

Сценарий (скрипт, script) – программа, автоматизирующая некоторую задачу, которую пользователь обычно выполняет вручную, используя интерфейсы программы.

Стандарт CGI

Круг задач, решаемых Web-сервером, ограничен. В основном он сводится к поддержке HTTP-взаимодействия и доставке клиенту Web-документов. Любые "нестандартные" действия реализуются с помощью специальной программы, которая взаимодействует с веб-сервером и клиентом. Это взаимодействие подчиняется определенным правилам.

Основной набор таких правил – стандарт **CGI** (Common Gateway Interface – интерфейс общего шлюза), который определяет порядок запуска программы на компьютере-сервере, способы передачи программе параметров и доставки результатов ее выполнения клиенту. Программа, написанная по правилам CGI, называется CGI-сценарием (script CGI), хотя это не означает, что на сервере не может выполняться двоичный файл.

Благодаря этому интерфейсу для разработки приложений можно использовать любой язык программирования, который располагает средствами взаимодействия со стандартными устройствами ввода/вывода. Такими возможностями обладают в также сценарии для встроенных командных интерпретаторов операционных систем.

Выполнение любой программы (в том числе CGI-сценария) можно условно разделить на пять этапов.

1. Запуск программы.
2. Инициализация и чтение выходных данных.
3. Обработка данных.
4. Вывод результатов выполнения.
5. Завершение программы.

Различия между CGI-сценарием и консольным приложением касаются первого, второго и четвертого этапов выполнения.

Каждый раз, когда веб-сервер получает запрос от клиента, он анализирует содержимое запроса и возвращает соответствующий ответ:

- Если запрос содержит указание на файл, находящийся на жестком диске, то сервер возвращает в составе ответа этот файл;
- Если запрос содержит указание на программу и необходимые для нее аргументы, то сервер исполняет программу и результат ее работы возвращает клиенту.

CGI определяет:

- каким образом информация о сервере и запросе клиента передается программе в форме аргументов и переменных окружения;
- каким образом программа может передавать назад дополнительную информацию о результатах (например о типе данных) в форме заголовков ответа сервера.

В подавляющем большинстве случаев запуск CGI-сценария осуществляется щелчком на кнопке Submit, сформированной с помощью дескриптора <input type = "submit">, который находится на HTML-странице между <form> и </form>. Не зная назначения атрибутов action и method, невозможно понять, как происходит вызов программы и передача параметров.

Значением атрибута action дескриптора <form> является URL файла, содержащего код CGI-сценария. Так, приведенное ниже выражение означает, что файл с кодом CGI-сценария находится на сервере www.myhp.edu в каталоге cgi-bin в файле script.pl.

<form action=”http://www.myhp.edu/cgi-bin/script.pl” method="post">

Как веб-сервер различает, что надо сделать с файлом, на который указывает URL, – передать его содержимое клиенту или запустить файл на выполнение? Существует два способа распознавания файлов, содержащих тексты CGI-сценариев.

- Первый способ заключается в том, что при установке веб-сервера один из каталогов специально выделяется для хранения сценариев. Обычно такой каталог получает имя cgi-bin (или Scripts для веб-сервера IIS). В этом случае, если клиент запрашивает файл из каталога cgi-bin, сервер воспринимает такой запрос как команду на запуск сценария. Файлы из других каталогов интерпретируются как HTML-документы.
- Второй способ использует расширение файла. При настройке сервера указывается, что файлы с определенными расширениями содержат коды сценариев.

Идентификация по расширению используется относительно редко. Чаще всего все сценарии помещаются в /cgi-bin, /Scripts или в другой каталог, специально выделенный для их хранения.

Вывод результатов выполнения CGI-сценария осуществляется чрезвычайно просто. Для того чтобы данные были переданы клиенту, достаточно вывести их в стандартный выходной поток. Однако, разрабатывая

CGI-сценарий, не следует забывать о том, что он все же отличается от консольной программы и имеет следующие особенности.

Информация, передаваемая клиенту, должна соответствовать протоколу HTTP, т.е. состоять из заголовка и тела ответа. Как правило, получив данные от сценария, сервер самостоятельно добавляет первую строку заголовка.

HTTP/1.0 200 OK

Формирование информационных полей, входящих в состав заголовка, – задача сценария. Чтобы данные, переданные сценарием, были правильно интерпретированы клиентом, необходимо, чтобы в заголовке присутствовало как минимум поле Content-type. За заголовком должна следовать пустая строка. При отсутствии полей заголовка реакция браузера будет непредсказуемой. В подобных случаях браузер обычно пытается отобразить полученную информацию как текстовый файл.

Самый естественный формат для браузера – формат HTML. Результаты работы сценария обычно оформляются в виде веб-страницы, т.е. возвращаемые данные следует дополнить дескрипторами HTML. Таким образом, ответ CGI-сценария клиенту обычно выглядит так:

```
Content-type: text/html

<html>
<head>
<title>ответ сценария</title>
</head>
<body>
……
</body>
</html>
```

Обратите внимание на пустую строку после выражения Content-type:text/html. Она обязательно должна присутствовать в ответе, в противном случае клиент воспримет все последующие данные как продолжение заголовка.

После компиляции программы необходимо скопировать исполняемый файл в каталог cgi-bin (или в другой каталог, предназначенный для размещения исполняемых файлов) из которого он может запускаться веб-сервером на выполнение по запросу клиента.

Для вызова данного сценария достаточно включить в веб-страницу следующий фрагмент HTML-кода:

```
<form method="post" action="/cgi-bin/hello.exe">
<input type="submit">
</form>
```

Если сценарий вызывается из формы, ему передаются те данные, которые пользователь ввел с помощью интерактивных элементов, отображаемых на веб-странице – передача информации CGI-сценарию

осуществляется в два этапа: сначала браузер передает данные веб-серверу, затем веб-сервер передает их сценарию.

В большинстве случаев кроме кнопки Submit форма содержит другие интерактивные элементы, каждый из которых имеет имя (атрибут NAME) и значение (атрибут VALUE, либо последовательность символов, введенная пользователем). Из имен элементов и их значений формируется строка параметров, которая имеет следующий формат.

имя=значение&имя=значение& . . . &имя=значение

Каждый параметр представляет собой имя управляющего элемента и его значение, разделенные знаком равенства, а несколько таких пар объединяют строку с помощью символа "&". Если в состав имени или значения входит символ "&" или "=", то подобные символы кодируются последовательностью знака процента "%", за которым следуют две шестнадцатеричные цифры, определяющие код символа. Так, например, последовательностью "%21" кодируется восклицательный знак "!". Как правило, при передаче параметров трехсимвольными последовательностями заменяются все знаки, кроме латинских букв, цифр и символа пробела (последний заменяется знаком "+").

Таким образом, перед использованием строки параметров ее надо декодировать. Алгоритм декодирования чрезвычайно прост и включает в себя следующие действия:

- Выделить из строки параметров пары имя = значение.
- Выделить из каждой пары имя и значение.
- В каждом имени и каждом значении заменить символы "+" пробелами.
- Каждую последовательность из символа "%" и двух шестнадцатеричных и преобразовать в ASCII-символ.

Атрибут method дескриптора <form> имеет либо значение "GET", либо значение "POST". Значения "GET" и "POST" определяют два различных метода передачи параметров сценарию:

- Если атрибут method имеет значение "GET", строка параметров передается вместе с URL вызываемого сценария. Разделителем между URL и строкой параметров является символ "?".
- Если атрибут method имеет значение "POST", строка параметров передается в теле HTTP-запроса.

Рассмотрим, как должен вести себя CGI-сценарий, чтобы правильно обработать данные в зависимости от метода, использованного при передаче данных, строка параметров доставляется CGI-сценарию различными способами.

Если атрибут METHOD дескриптора <FORM> имел значение "GET", строка параметр передается серверу в качестве значения переменной окружения QUERY_STRING.

При использовании метода POST данные доставляются сценарию по-другому. Они передаются через стандартный поток ввода (STDIN). Чтобы

сценарий смог определить, сколько символов следует читать из стандартного ввода, веб-сервер устанавливает значение переменной окружения CONTENT_LENGTH, равным длине строки параметров.

Получив управление, сценарий в первую очередь должен выяснить, с помощью какого метода выполнялась передача параметров. Эта информация содержится в переменной окружения REQUEST_METHOD.

Таким образом, в простейшем случае, чтобы выполнить обработку строки параметров, достаточно знать назначение трех переменных окружения: REQUEST_METHOD, QUERY_STRING и CONTENT_LENGTH.

Пример сценария на языке Perl, который возвращает клиенту строку параметров, приведен ниже. Сценарий определяет, какой метод использовался для передачи данных, читает строку параметров и передает ее клиенту, предварительно дополнив HTML-дескрипторами.

```
$method = $ENV{'REQUEST_METHOD'};

if ($method eq "GET")
{ $pars = $ENV{'QUERY_STRING'}; }
else
{ $length =$ENV{'CONTENT_LENGTH'}; }

read (STDIN, $pars, $ length);

print "Content-type: text/html\n\n";
print "<HTML><BODY>\n";
print "<P>METHOD = ", $method;
print "<P>String of parameters: <P>\n";
print $pars;
print "</HTML></BODY>\n";
```

При разработке более сложных сценариев может потребоваться дополнительная информация. Информация о типах сервера и браузера, адресе клиент-машины и многие другие сведения передаются с помощью переменных окружения. Некоторые из них перечислены ниже

REMOTE_ADDR	IP-адрес узла, с которого поступил запрос
REMOTE_HOST	Доменное имя узла, с которого поступил запрос
SERVER_PORT	Номер порта, который использовался при обращении к серверу
SERVER_SOFTWARE	Имя и версия сервера, посредством которого был запущен сценарий
SERVER_NAME	Имя или адрес узла, на котором выполняется сервер
SERVER_PROTOCOL	Название и версия протокола, с помощью которого был передан запрос
HTTP_USER_AGENT	Клиентская программа, отправившая запрос серверу
HTTP_REFERER	URL документа, отображаемого браузером при вызове сценария

Сценарии

К основным достоинствам разработки приложений на стороне веб-сервера в форме сценариев можно отнести следующие:

- поскольку сценарии не компилируются, а интерпретируются, то ошибки в сценарии вызовут только диагностическое сообщение, но не приведут к дестабилизации веб-сервера или операционной системы.
- лучшие выразительные возможности. Язык сценариев, как правило, имеет собственный проблемно-ориентированный набор команд, и одна строка сценария может делать то же, что несколько десятков строк на традиционном языке. Как следствие, на этом языке может писать программист низкой квалификации.
- Поддержка кроссплатформенности.

Поскольку сценарии интерпретируются из исходного кода динамически при каждом исполнении, они выполняются обычно значительно медленнее готовых программ, транслированных в машинный код на этапе компиляции.

В плане быстродействия сценарные языки можно разделить на:

- Языки динамического разбора (например command.com). Интерпретатор считывает инструкции из файла программы минимально требующимися блоками, и исполняет эти блоки, не читая дальнейший код.
- Предварительно компилируемые (например Perl). Вначале считывается вся программа, затем компилируется либо в машинный код, либо в один из внутренних форматов, после чего получившийся код исполняется.

В оставшейся части лекции рассмотрим кратко наиболее известные языки разработки сценариев для веб-приложений.

Python

Python – высокоуровневый многопарадигменный язык программирования общего назначения с акцентом на производительность и читаемость кода. Язык Python сочетает в себе минимализм синтаксиса ядра и большой объём полезных функций в стандартной библиотеке.

Python поддерживает структурную, объектно-ориентированную, функциональную, императивную и аспектно-ориентированную парадигмы.

Его основные архитектурные черты:

- динамическая типизация
- автоматическое управление памятью
- полная интроспекция
- механизм обработки исключений
- поддержка многопоточных вычислений
- удобные высокоуровневые структуры данных

Код в Python организовывается в функции и классы, которые могут объединяться в модули (которые в свою очередь могут быть объединены в пакеты).

Для всех основных платформ Python имеет поддержку характерных для данной платформы технологий (например, Microsoft COM/DCOM). Существует даже специальная версия Python для виртуальной машины Java – Jython, что позволяет интерпретатору выполняться на любой системе, поддерживающей Java, при этом классы Java могут непосредственно использоваться из Питона и даже быть написанными на Python. Несколько проектов обеспечивают интеграцию с платформой Microsoft.NET, основные из которых – IronPython и Python.Net.

Стандартная библиотека Python имеет средства для работы со многими сетевыми протоколами и форматами интернета, например, модули для написания HTTP-серверов и клиентов, для разбора и создания почтовых сообщений, для работы с XML и т.п. Набор модулей для работы с операционной системой позволяет писать кросс-платформенные приложения. Существуют также модули для работы с регулярными выражениями, текстовыми кодировками, мультимедийными форматами, криптографическими протоколами, архивами, сериализации данных, поддержка юнит-тестирования и др.

Помимо стандартной библиотеки существует множество библиотек, предоставляющих интерфейс ко всем системным вызовам на разных платформах; Имеется большое количество прикладных библиотек для Python в самых разных областях (веб, базы данных, обработка изображений, обработка текста, численные методы, приложения операционной системы и т. д.).

Ruby

Ruby – интерпретируемый язык высокого уровня для быстрого и удобного объектно-ориентированного программирования. Ruby обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, «сборщиком мусора» и многими другими возможностями. Многие особенности синтаксиса и семантики языка Perl заимствованы в Ruby.

Первая общедоступная версия Ruby появилась в 1995 г.

Ruby – полностью объектно-ориентированный язык:

- Все данные являются объектами, в отличие от многих других языков, где существуют примитивные типы.
- Каждая функция является методом.
- Переменные Ruby содержат не сами объекты, а ссылки на них.
- Присваивание – это не передача значения, а копирование ссылки на объект.
- В Ruby можно добавлять методы не только в любые классы, но и в любые объекты. Например, можно добавить к некоторой строке произвольный метод.

Массивы в Ruby могут автоматически изменять размер, могут содержать любые элементы и язык предоставляет мощные средства для их обработки.

Ruby поставляется с большой стандартной библиотекой. Это, прежде всего, библиотеки для работы с различными сетевыми протоколами на стороне сервера и клиента, средства для работы с различными форматами представления данных (XML, XSLT, YAML, PDF, RSS, CSV, WSDL). Также есть библиотеки для работы с архивами, датами, кодировками, матрицами, средства для системного администрирования, распределенных вычислений, поддержки многопоточности и т.д.

В языке Ruby также реализован простой и удобный механизм для расширения языка с помощью библиотек, написанных на Си, позволяющий легко разрабатывать дополнительные библиотеки. Для унифицированного доступа к базам данных разработана библиотека Ruby DBI.

К недостаткам интерпретатора Ruby можно отнести следующие:

- Невысокая скорость работы.
- Отсутствие поддержки потоков операционной системы (для Unix-подобных операционных систем есть поддержка процессов ОС), есть в экспериментальной версии 1.9.
- Отсутствие встроенной поддержки юникода (возможна работа с использованием дополнительных библиотек, есть в экспериментальной версии 1.9).
- Отсутствие компиляции в байткод. (При этом есть возможность компилировать Ruby в Java и .NET байткод, используя компилятор JRuby и Ruby.NET). В экспериментальную версию 2.0 входит виртуальная машина YARV, компилирующая Ruby в байткод и существенно ускоряющая исполнение.

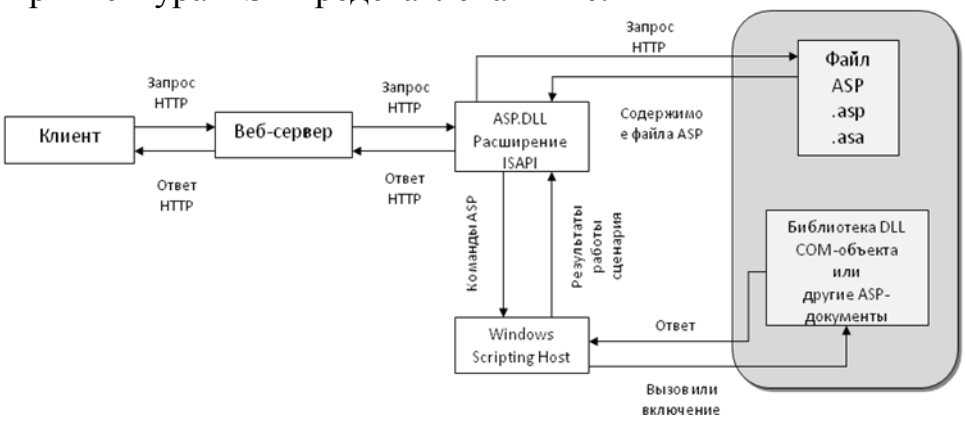
ASP

ASP (Active Server Pages) – технология, разработанная компанией Microsoft, позволяющая легко создавать приложения для Веб.

Программирование на ASP дает разработчикам доступ к интерфейсу программирования приложений Internet Information Server с помощью языка сценариев VBScript и JScript.

ASP работает на платформе операционных систем линии Windows NT и на веб-сервере Microsoft IIS.

Архитектура ASP представлена ниже.



Файлы ASP представляют собой сценарии, интерпретируемые по мере поступления запросов. ISAPI-расширение ASP.DLL связано в IIS с расширениями файлов .asp или .asa.

Порядок обработки таких файлов выглядит следующим образом:

- ASP.DLL просматривает файлы с указанными расширениями на наличие тегов, обозначающих внедренный код для выполнения на сервер и передает найденный код в Windows Script Host (WSH).
- WSH выполняет этот код и возвращает результат файлу ASP.DLL.
- ASP.DLL передает IIS этот результат и содержимое самого файла ASP.
- IIS возвращает ответ клиенту, от которого поступил запрос.

Рассмотрим основы синтаксиса ASP.

IIS различает код, выполняющийся на сервере, и содержимое, отправляемое клиенту с помощью ASP.DLL, анализируя файл ASP на наличие начального “<%” и конечного “%>” тегов и выполняя код, расположенный между ними, с помощью WSH.

Рассмотрим пример:

```
<% Language=VBScript %>
<HTML>
<BODY>
<%
Response.Write(“<p>Hello world!</p>”)
%>
</BODY>
</HTML>
```

В примере первая строка кода <% Language=VBScript %> сообщает о необходимости использовать интерпретатор языка VBScript. Для вставки строки в документ был использован метод Write стандартного объекта Response.

Событие веб-запроса в ASP обрабатывается с помощью следующих объектов:

- **Response.** Используется для записи данных в запрос HTTP, возвращаемый клиенту.
- **Application.** Содержит параметры и конфигурации по настройке работы ASP для данного веб-сайта.
- **Request.** Хранит содержимое HTTP-запроса и обеспечивает вспомогательные функции для обработки данных HTTP-запроса.
- **Server.** Содержит информацию о веб-сервере, веб-сайте, а также обеспечивает поддержку вызывающей программы.
- **Session.** Представляет собой состояние заданного веб-сеанса с заданным хостом клиентом.

ISAPI

Для веб-сервера IIS (Internet Information Server). был разработан специальный программный интерфейс для создания приложений расширяющих стандартные возможности веб-сервера.

ISAPI (Internet Server Application Programming Interface) – многозвенный API для IIS.

ISAPI также реализован в виде модуля mod_isapi для веб-сервера Apache. Таким образом, серверные приложения, разработанные для MS IIS могут также выполняться в Apache и других веб-серверах.

В противоположность CGI – ISAPI-приложение загружается в том же адресном пространстве, что и веб-сервер IIS. Это позволяет повысить производительность приложений благодаря сокращению издержек на запуск отдельных процессов. Однако сбой ISAPI-приложения может привести к неустойчивой работе самого веб-сервера. В 6 ой версии IIS имеется возможность запуска приложений в рамках отдельного процесса.

ISAPI включает в себя две компоненты: расширения и фильтры.

Таким образом, все многообразие разрабатываемых ISAPI-приложений сводится только к этим двум типам. И фильтры и расширения компилируются в DLL файлы динамически запускаемые веб-сервером.

ISAPI приложения могут разрабатываться с помощью любых языков, поддерживающих экспорт стандартных C-функций, например C, C++, Delphi Pascal. Для разработки имеется ограниченное число библиотек для разработки ISAPI приложений, например Intraweb-компоненты Delphi Pascal, специальные MFC-классы, специальная C++ библиотека серверных технологий ATL.

К наиболее важным особенностям ISAPI-расширений можно отнести следующие:

- ISAPI-расширения имеют доступ ко всем функциональным возможностям IIS.
- Реализуются в виде DLL-модулей, загружаемых в пространстве процесса, контролируемого IIS.
- Клиенты могут обращаться к ISAPI-расширениям также как к статическим HTML страницам.
- ISAPI-расширения могут быть ассоциированы с отдельными расширениями файлов, с целыми каталогами или сайтами.

ISAPI-фильтры необходимы для изменения или совершенствования функциональности IIS. Они обычно работают с IIS-сервером и фильтруют каждый запрос. Фильтры применяются для анализа и модификации входящих и исходящих потоков данных.

Фильтры также как и расширения реализуются в виде DLL файлов.

Обычно ISAPI-фильтры используются для решения следующих задач:

- Изменение данных в запросе клиента (URL или заголовков).
- Управление отображением URL в физические файлы.

- Управление именами и паролями пользователей при анонимной или базовой аутентификации.
- Анализ и модификация запросов по завершении аутентификации.
- Модификация ответа веб-сервера.
- Ведение журналов и анализ трафика.
- Реализация собственной аутентификации.
- Управление шифрацией и сжатием.

Стоит отметить, что существуют реализации в виде ISAPI-расширений для таких инструментальных средств как:

- ASP (Active Server Pages)
- ASP.NET
- ColdFusion
- Perl ISAPI (Perlis)
- PHP

Язык Perl

Язык Perl (Practical Extraction and Report Language) – это язык программирования, сильными сторонами которого считаются его богатые возможности для работы с текстом, в том числе реализованные при помощи регулярных выражений. Также язык известен тем, что имеет огромную коллекцию дополнительных модулей CPAN.

Чтобы запустить программу на языке Perl на выполнение, ее компиляция не требуется, она вполне может выполняться под управлением интерпретатора. Чтобы файл с исходным текстом Perl можно было запускать на выполнение, надо чтобы первая его строка выглядела так:

#!путь_к_интерпретатору_Perl

Основными типами данных в языке являются: скаляры, массивы (скалярные), хеш-таблицы (ассоциативные массивы), функции, файловые дескрипторы и константы.

Переменные разных типов отличаются знаком, который стоит перед именем переменной:

$a – скаляр или указатель
@b – скалярный массив
%c – ассоциативный массив (хеш-таблица)
&d – функция
F – дескриптор ввода-вывода или константа

Скалярные переменные используются для хранения одиночных значений. Они могут содержать числа, строки и ссылки на другие объекты. Перед именем скалярной переменной необходимо ставить знак доллара '$'. Тип скалярной переменной не фиксирован и определяется динамически в зависимости от контекста.

Скалярный массив является упорядоченным списком скаляров. Каждый элемент массива имеет порядковый номер (индекс), с помощью которого к нему можно получить доступ. Нумерация элементов начинается с нуля.

Перед именем переменной типа скалярный массив указывается знак @, а для доступа к определенному элементу массива необходимо ставить знак $, так как определенный элемент массива является скаляром:

```
@winter = ("декабрь", "январь", "февраль");
print "Второй месяц зимы ", $winter[1], "\n";
```

Хеш-таблица представляет собой ассоциативный массив, позволяющий ассоциировать строку (ключ) со скаляром (значение). Строка при этом называется ключом, а скаляр в хеш-таблице – значением. Перед именем переменной-списка необходимо ставить знак процента %, а для доступа к определенному элементу массива ставят знак $.

Фактически хеш-таблица представляет собой массив, где в нечетных позициях находятся ключи, а на четных – значения.

Использование ассоциативных массивов напоминает использование массивов скалярных значений, однако индексация производится не целыми числами, а ключевыми словами. Кроме того, индексы заключаются не в квадратные, а в фигурные скобки.

Так, например, для того чтобы присвоить значение трем элементам массива %dict с индексами first, second и third, можно воспользоваться одним из двух способов, указанных ниже.

```
$dict { 'first' } = "первый";
$dict { 'second' } = "второй";
$dict { 'third' } = "третий";
```

или

```
%dict { 'first', 'second', 'third' } = "первый", "второй", "третий";
```

Кроме того, существует способ одновременно записать в ассоциативный массив и ключевые слова, и их значения. Сделать это можно с помощью следующего выражения:

```
имя_массива = ( ключ 1, значение 1, ключ 2, значение 2, ... );
```

Для примера, приведенного выше, это выражение будет выглядеть так:

```
%dict = ("first", "первый", "second", "второй", "third", "третий");
```

Рассмотрим приведенный следующий фрагмент программы на языке Perl.

```
while (<STDIN>)
{ print; }
```

Несмотря на то, что формально программа составлена правильно, она на первый взгляд может показаться бессмысленной. Однако при запуске она ведет себя точно так же, как и одна из программ "эхо". В языке Perl существует предопределенная скалярная переменная $_, используемая по умолчанию. Именно в нее помещаются данные, прочитанные из стандартного ввода, и из нее берется значение для вывода в STDOUT.

Кроме $_ в Perl имеются и другие предопределенные переменные:

$] – номер версии Perl.

$. – номер строки, прочитанной из файла последней.

$! – сообщение об ошибке.

$$ – идентификатор текущего процесса.

$^T – время в секундах с начала 1970 года до запуска данной программы.

$O – имя файла, в котором содержится выполняемая программа.

$1...$9 – фрагменты текста, отмеченные при выполнении операции сопоставления с шаблоном.

Подобно предопределенным скалярным переменным, в Perl существуют массивы, имеющие специальное значение. Наиболее важный из них – ассоциативный массив %ENV, содержащий текущие значения переменных окружения. Чтобы получить значение переменной окружения, надо обратиться к элементу данного массива, указав в качестве индекса имя переменной окружения. Так, приведенное ниже выражение записывает в скалярную переменную $path_string значение переменной окружения PATH.

$path_string = $ENV { 'PATH' };

Одна из первых строк CGI-сценария на Perl, может выглядеть так

$method = $ENV { 'REQUEST_METHOD' };

Для работы с файлами и потоками в Perl предусмотрены специальные файловые дескрипторы.

Файловые дескрипторы представляет собой указатель на файл, устройство или PIPE канал, открытые для записи, чтения или для записи и чтения. Оператор “<>” в Perl называется бриллиантовым оператором (diamond operator). Он определяет операцию чтения строки из потока, дескриптор которого содержится в угловых скобках:

$str=<STDIN>; #чтение строки из дескриптора STDIN (стандартного потока ввода)

@lines=<F>; #чтение всех строк из связанного с дескриптором файла F.

print STDOUT $str; #печать в STDOUT (стандартный поток вывода)

Для связывания файла с файловым дескриптором используется функция open. Ниже приводятся варианты использования этой функции:

open дескриптор_потока > имя_файла	файл открывается для вывода данных. Если файл с указанным именем отсутствует, создается новый файл.
open дескриптор_потока >> имя_файла	файл открывается в режиме, позволяющем записывать данные в конец файла.
open дескриптор_потока +> имя_файла	открытый файл становится доступным для чтения и для записи.

Функция

close дескриптор_файла

закрывает файл, связанный с указанным дескриптором.

В состав языка Perl входят средства поиска и замены, причем, задавая шаблон для поиска, можно использовать регулярные выражения. Это значит, что сложные операции, встречающиеся в специализированных приложениях, можно легко реализовать в любой Perl-программе.

Оператор поиска m// записывается следующим образом:

m/шаблон/

Если значение переменной $_ содержит подстроку, соответствующую указанному шаблону, оператор поиска возвращает значение true.

Рассмотрим следующий пример:

```
$_ = <INPUT>;
if (m/Scripts/)
{ print "B URL есть каталог Scripts \n"; }
else
{ print " B URL нет каталога Scripts \n"; }
```

Оператор замены s/// записывается следующим образом:

s/шаблон поиска/выражение для замены/[набор модификаторов]

При выполнении оператора s/// производится поиск соответствия шаблону, и если поиск завершается успешно, найденная подстрока заменяется указанным выражением. Подобно оператору m//, оператор s/// использует переменную $_. Ниже приведен простейший пример применения оператора s///.

```
$_ = "CGI-сценарий написан на языке C";
s/C$/Perl/;
print;
```

В результате выполнения сценария на консоль будет выведена следующая строка:

CGI-сценарий написан на языке Perl

Поскольку символ C содержится в аббревиатуре CGI, поэтому в шаблоне поиска указано, что он должен быть последним в строке.

За последним разделителем в операторе s/// могут следовать один или несколько модификаторов. Назначение некоторых модификаторов приведено ниже.

- g – глобальный поиск. Если этот модификатор не указан, после обнаружения первого соответствия оператор s/// закончит свою работу. Поэтому при отсутствии модификатора g будет произведено не более одной замены.
- i – указывает, что при поиске следует игнорировать регистр символов.
- e – указывает, что последовательность символов для замены следует интерпретировать не как подстроку, а как выражение Perl.

В выражении для подстановки могут присутствовать переменные $1 – $9, и в этом случае необходимо указать модификатор e. Так, например, если требуется интерпретировать десятичное число как код символа, можно воспользоватъся следующим выражением:

```
s/([0-9]+)/chr($1)/e;
```

Чтобы поиск или замена производились в строке, содержащейся в нужной переменной, надо использовать следующее выражение:

Переменная =~ оператор_поиска_или_замены

Так, например, для преобразования шестнадцатеричных чисел, содержащихся в переменной $string, в десятичное представление можно использовать инструкцию:

$string =~ s/([0-9A-Fa-f]+)(H|h)/hex($1)/ge;

Язык PHP

Язык PHP (PHP:Hypertext Preprocessor) – один из наиболее популярных сценарных языков ввиду своей простоты, скорости выполнения, богатой функциональности и распространенности исходных кодов на основе лицензии PHP.

PHP состоит из ядра и набора подключаемых расширений: для работы с базами данных, сокетами, динамической графикой, криптографическими библиотеками, документами формата PDF и др. Возможна разработка своих собственных расширений с их последующим подключением. Хотя и существуют сотни расширений, однако в стандартную поставку входит лишь несколько десятков хорошо зарекомендовавших себя расширений.

Интерпретатор PHP подключается к веб-серверу либо через DLL модуль, созданный специально для этого сервера, либо в виде CGI-приложения.

В настоящее время PHP используется сотнями тысяч разработчиков. Порядка 20 миллионов сайтов сообщают о работе с PHP, что составляет более пятой доли доменов Интернета

Синтаксис PHP подобен синтаксису языка Си. При этом некоторые элементы, как например ассоциативные массивы и цикл foreach, заимствованы из языка Perl.

Для работы программы на PHP не требуется описывать какие-либо переменные, используемые модули, и т.п. Любая программа может начинаться непосредственно с оператора PHP.

```
<?php
echo 'Hello, world!';
?>
```

Помимо ограничителей <?php ?>, допускается использование дополнительных вариантов, таких как <? ?> и <script language="php"> </script>. Кроме того, до версии 6.0 допускается использование ограничителей языка программирования ASP <% %>.

Имена переменных начинаются с символа $, тип переменной объявлять не требуется. В отличие от имён функций и классов, имена переменных чувствительны к регистру. Переменные обрабатываются в строках, заключённых в двойные кавычки.

Инструкции завершаются точкой с запятой (;)

PHP поддерживает два типа комментариев:

- в стиле языка C (ограниченные /* */),
- C++ (начинающиеся с // и идущие до конца строки)

PHP является языком программирования с динамической типизацией, не требующим указания типа при объявлении переменных, равно как и самого объявления переменных. Преобразования между скалярными типами может осуществляться автоматически (хотя и имеются возможности для явного преобразования типов).

К **скалярным** типам данных относятся

- целый тип (integer),
- вещественный тип данных (float, double),
- логический тип (boolean),
- строковый тип (string)
- специальный тип NULL.

К **нескалярным** типам относится

- «ресурс» (resource),
- массив (array),
- объект (object).

Тип NULL предназначен для переменных без определённого значения. Значение NULL принимают неинициализированные переменные, переменные инициализированные константой NULL, а также переменные, удалённые при помощи конструкции unset().

Ссылки на внешние ресурсы имеют тип resource. Переменные данного типа, как правило, представляют собой дескриптор, позволяющий управлять внешними объектами, такими как файлы, динамические изображения, результирующие таблицы базы данных и т. п.

Массивы поддерживают числовые и строковые ключи и являются гетерогенными. Массивы могут содержать значения любых типов, включая другие массивы. Суперглобальными массивами (superglobal arrays) в PHP называются предопределённые массивы, которые видны в любом месте исходного кода без использования ключевого слова global.

- $GLOBALS – массив всех глобальных переменных (в том числе и пользовательских).
- $_SERVER – содержит множество информации о текущем запросе и сервере.
- $_ENV – текущие переменные среды. Их набор специфичен для каждой конкретной платформы, на которой выполняется сценарий.
- $_GET – ассоциативный массив с параметрами GET-запроса. В исходном виде эти параметры доступны в $_SERVER['QUERY_STRING'] и в $_SERVER['REQUEST_URI'] в составе URI.
- $_POST – ассоциативный массив значений полей HTML-формы при отправке методом POST.
- $_FILES – ассоциативный массив со сведениями об отправленных методом POST файлах. Каждый элемент имеет индекс идентичный значению атрибута «name» в форме и, в свою очередь, также является массивом со следующими элементами:
  - ['name'] – исходное имя файла на компьютере пользователя.
  - ['type'] – указанный агентом пользователя MIME-тип файла.
  - ['size'] – размер файла в байтах.
  - ['tmp_name'] – полный путь к файлу во временной папке.
  - ['error'] – код ошибки.
- $_COOKIE – ассоциативный массив с переданными агентом пользователя значениями cookie.

- $_REQUEST – общий массив вводных данных запроса пользователя как в массивах $_GET, $_POST, $_COOKIE. Начиная с версии PHP 4.1 включается и содержимое $_FILES.
- $_SESSION – информация о текущей сессии пользователя.

PHP поддерживает широкие объектно-ориентированные возможности, полная поддержка которых была введена в пятой версии языка. Класс в PHP объявляется с помощью ключевого слова class. Методы и поля класса могут быть общедоступными (**public**, по умолчанию), защищёнными (**protected**) и скрытыми (**private**). PHP поддерживает все три основные механизма ООП – инкапсуляцию, полиморфизм и наследование (родительский класс указывается с помощью ключевого слова **extends** после имени класса). Поддерживаются интерфейсы (ставятся в соответствие с помощью **implements**). Разрешается объявление финальных, абстрактных методов и классов. Множественное наследование классов не поддерживается, однако класс может реализовывать несколько интерфейсов. Для обращения к методам родительского класса используется ключевое слово **parent.** Экземпляры класса создаются с помощью ключевого слова **new**, обращение к полям и методам объекта производится с использованием символов ->. Для доступа к членам класса из его методов используется переменная **$this**.

Среди наиболее часто используемых возможностей PHP стоит отметить следующие:

- имеется большой набор функций для работы со строками;
- работа с регулярными выражениями PCRE.
- работа с базами данных, осуществляемая посредством модулей:
  - php5-mysql для MySQL,
  - php5-pgsql для PostgreSQL
  - и др.
- для PHP разработаны средства шаблонирования веб-страниц, позволяющие эффективно разделить представление от модели, например Smarty;
- имеется библиотека для работы с графическими изображениями GD, позволяющая производить преобразования с графическими файлами, и создавать изображения «на лету».

Архитектура веб-приложений ASP.NET. Разработка веб-приложений на платформе .NET.

Платформа .NET Framework предоставляет возможность разработки и интеграции веб-приложений. ASP.NET является одной из составляющих инфраструктуры .NET Framework и фактически является платформой для создания веб-приложений и веб-сервисов, работающих под управлением IIS.

ASP.NET внешне во многом напоминает более старую технологию ASP, но в то же время внутреннее устройство ASP.NET существенно отличается от ASP. Компания Майкрософт ASP.NET построила на базе CLR (Common Language Runtime), который является основой всех приложений

.NET. Разработчики могут создавать код для ASP.NET, используя языки программирования, входящие в .NET Framework: C#, Visual Basic.NET, JScript.NET и другие.

Рассмотрим более подробно, чем отличается ASP.NET от ASP.

Классический ASP имеет следующие недостатки:

- Используются только языки сценариев, которые дают большой проигрыш в производительности (из-за их интерпретируемости) и не поддерживают многие возможности объектно-ориентированного программирования.
- Логика представления (в виде кода HTML) не отделена от бизнес-логики (исполняемого кода), что приводит перемешиванию в одном файле кода HTML с кодом сценария.
- Невозможно повторно использовать готовые решения в других проектах (возможно только копирование кода сценариев).

В файлах ASP.NET включается код на таких языках программирования как C#, JScript.NET, VisualBasic.NET, что позволяет применять непосредственно в веб-приложениях возможности объектно-ориентированного программирования. Также существенно сокращается объем кода, написанного вручную за счет применения серверных объектов, автоматически генерирующих код элементов управления HTML. Возможно использование стандартной среды разработки Visual Studio.NET, т.е. ASP.NET имеет преимущество в скорости по сравнению со сценарными технологиями, так как при первом обращении код компилируется и помещается в специальный кеш, а впоследствии только исполняется, не требуя затрат времени на парсинг, оптимизацию, и т. д.

Несмотря на возможность совместной работы ASP и ASP.NET на одном веб-сервере, они не могут использовать общий сеанс. Файлы ASP.NET обрабатываются библиотекой aspnet_isapi.dll (а не asp.dll), которая, в свою очередь, использует для выполнения кода технологию .NET.

Библиотека базовых классов .NET содержит пространства имен 3 основных групп:

- элементы web-приложений (протоколы, безопасность и др.);
- элементы графического интерфейса (WebForms) ;
- web-службы.

Как уже указывалось ранее, ASP.NET использует возможности стандартной среды разработки Visual Studio.Net, и в частности классы библиотеки FCL (Framework Class Library).

Разработчику веб-приложений на ASP.NET доступны классы, входящие в следующие пространства имен:

Пространство имен	**Содержание**
System.Web	Организация взаимодействия web-клиента (браузера) с web-сервером (запрос-ответ, cookie и и др.)
System.Web.Caching	Поддержка кэширования при работе

Пространство имен	Содержание
	web-приложений
System.Web.Configuration	Настройка web-приложения в соответствии с файлами конфигурации проекта
System.Web.Security	Реализация системы безопасности web-приложений
System.Web.Services	Организация работы web-сервисов
System.Web.Services.Description	
System.Web.Services.Discovery	
System.Web.Services.Protocols	
System.Web.UI	Построение графического интерфейса пользователей web-приложений
System.Web.UI.WebControls	
System.Web.HtmlControls	

В свою очередь пространство имен System.Web включает в себя пространства имен, названия которых знакомы разработчикам веб-приложений на ASP:

Пространство имен	Содержание
HttpApplication	Данный класс определяет общие для всех web-приложений члены
HttpApplicationState	В данном классе содержится общая информация web-приложения для множества запросов, сеансов и каналов передачи данных
HttpBrowserCapabilities	Этот класс используется для получения информации о возможностях клиентского браузера, обращающегося к web-серверу
HttpCookie	Поддержка механизма безопасной работы с объектами HTTP cookie
HttpRequest	Предоставляет доступ к информации, переданной web-клиентом
HttpResponse	Используется для формирования HTTP-ответа сервера

В основу разработки веб-приложений на ASP.NET положена модель разделения кода представления и кода реализации, рекомендуемая Майкрософт при создании динамических документов с помощью программных кодов. Это делается путем размещения программного кода либо в отдельный файл, либо внутри специального тэга для сценариев. Файл такого рода обычно имеет расширение *.aspx.cs (*.aspx.vb) и имеет имя, совпадающее с именем основного ASPX файла. В принципе такой подход позволяет веб-дизайнеру сконцентрироваться на работе с кодом разметки документа с минимальными изменениями программного кода, в обычном ASP внедряемого непосредственно в код разметки.

Взаимодействие пользователя с веб-приложением, реализованном на ASP.NET включает в себя следующие процессы:

- При запросе страницы ASPX инициируется событие Page_Init, производящее начальную инициализацию страницы и ее объекта.
- Далее инициируется событие Page_Load, которое может быть использовано, например для установки начальных значений для элементов управления. При этом также можно определить была ли загружена страница впервые или обращение к ней осуществляется повторно в рамках обратной отсылки в ответ на события, связанные с элементами управления, размещенными на странице; т.е. проверить свойство Page.IsPostBack.
- Далее выполняется проверка валидности элементов страницы с точки зрения корректности введенных пользователем данных.
- И, наконец, следует обработка всех событий, связанных с действиями пользователя с момента последней обратной отсылки.

Для сохранения данных веб-страницы в промежутках между обращениями к ней в ASP.NET используются состояния отображения (view state).

Если данные, введенные в веб-форму, необходимо сделать доступными другим веб-формам того же приложения, эти данные необходимо сохранить в объектах Application и Session. Объекты Application доступны всем пользователям приложения и могут рассматриваться как глобальные переменные, обращение к которым возможно из любых сеансов. Объекты Session доступны только в рамках одного сеанса, и поэтому они оказываются доступными только одному пользователю.

Серверные элементы управления ASP.NET

Важной особенностью ASP.NET является использование серверных элементов управления на веб-странице (элементы WebForm), которые являются фактически тэгами, понятными веб-серверу. Эти элементы определены в пространстве имен System.Web.UI.WebControls.

Принято выделять три типа серверных элементов управления:

- Серверные элементы управления HTML – обычные HTML тэги.
- Элементы управления веб-сервера – новые тэги ASP.NET.
- Серверные элементы управления для проверки данных (валидации) – применяются для валидации входных данных от клиентского приложения (обычно веб-браузера).

Преимущества от использования таких элементов при разработке веб-приложений:

- Сокращается количество кода, написанного вручную (что особенно заметно в для сложных элементов документа). Элемент просто «перетаскивается» из панели инструментов, после чего выполняется настройка его параметров в специальном окне. При этом все изменения автоматически заносятся непосредственно в *.aspx файл.
- С программной точки зрения каждому из этих элементов управления соответствует определенный класс в библиотеке базовых классов .NET, что позволяет писать для них такой же код как и для любых других классов.

- Для любого элемента управления WebForm определен набор событий, обрабатываемых на веб-сервере.
- Для любого элемента управления WebForm предоставляется возможность для проверки ввода данных пользователем.

По умолчанию серверные элементы управления HTML в ASP.NET файлах рассматриваются как текст. Для их программирования требуется добавление атрибута runat="server" в соответствующий HTML элемент. Кроме того, все серверные элементы управления HTML должны быть размещены внутри области действия тэга <form>, также имеющего атрибут runat="server".

Подобно серверным элементам управления HTML элементы управления веб-сервера также создаются на веб-сервере и предполагают добавление атрибута runat="server". Однако они могут и не соответствовать конкретным элементам HTML, но представлять более сложные элементы.

Общий синтаксис для описания таких элементов:

<asp:тип_элемента id="идентификатор" runat="server"/>

Серверные элементы валидации применяются для проверки вводимых пользователем данных.

Имеют следующий синтаксис:

<asp:тип_элемента id="идентификатор" runat="server" />

Работа с источниками данных в ASP.NET

В ASP.NET используются два элемента управления WebForm для управления отображением данных, получаемых из источника данных:

- DataGrid – элемент управления, отображающий содержимое объекта ADO.NET DataSet в виде таблицы.
- DataList – элемент управления для выбора значений, заполняемых из источника данных.

Если необходимо отобразить данные, полученные по запросу пользователя из источника данных, в виде таблицы на веб-странице, то ASP.NET предоставляет в распоряжение веб-программиста удобный элемент управления DataGrid.

Интерфейсы взаимодействия веб-приложений с СУБД.

Сегодня большинство информационных систем в той или иной степени используют базы данных. Не составляют исключение и системы, основанные на веб-технологиях. Поэтому организация взаимодействия веб-приложений с СУБД является неотъемлемой составной частью веб-технологий.

До начала 90-х годов существовало несколько разных поставщиков баз данных, каждый из которых имел собственный интерфейс. Если приложению было необходимо обмениваться данными с несколькими источниками данных, для взаимодействия с каждой из баз данных было необходимо написать отдельный код. С целью решения этой проблемы Майкрософт и ряд других компаний создали стандартный интерфейс для получения и отправки данных источникам данных различных типов. Этот интерфейс получил название open database connectivity (ODBC).

С помощью ODBC прикладные программисты смогли разрабатывать приложения с использованием единого интерфейса доступа к данным, не учитывая тонкости взаимодействия с различными источниками данных. Это достигается благодаря тому, что поставщики различных баз данных разрабатывают драйверы, учитывающие специфику конкретных источников данных при реализации стандартных функций из ODBC API. При этом приложения используют функции такого API, реализованные в соответствующем конкретному источнику данных драйвере.

По-сути, интерфейс ODBC является обычным процедурным API. ODBC поддерживается большим количеством операционных систем.

Имеются также ODBC-драйверы и для нереляционных данных, таких как электронные таблицы, текст и XML файлы.

Типичный сценарий работы веб-приложения с источником данных выглядит следующим образом:

1. Установление соединение и подключение к источнику данных.
2. Выполнение запросов, необходимых для выборки, вставки или изменения наборов данных источника.
3. Отключение от источника данных.

Компанией Майкрософт был предложен интерфейс программирования приложений для доступа к данным, разработанный и основанный на технологии компонентов ActiveX – ADO (ActiveX Data Objects), который позволяет представлять данные из разнообразных источников (реляционных баз данных, текстовых файлов и т. д.) в объектно-ориентированном виде. Компоненты ADO нашли применение при разработке приложений на таких языках как VBScript в ASP и Visual Basic.

В рамках Microsoft .NET основной моделью доступа приложений к источникам данных является ADO.NET. Она не является развитием ADO и представляет собой совершенно самостоятельную технологию. Компоненты ADO.NET входят в поставку .NET Framework.

ADO.NET включает в себя две основные части:

- **Data provider –** набор классов для доступа к источникам данных. Каждый из источников данных имеет свой собственный набор объектов, однако все они имеют общее множество классов: Connection, Command, Parameter, DataAdapter, DataReader.
- **DataSets** объекты – группа классов, описывающих простые реляционные базы данных, размещаемы в памяти. Содержит иерархию таких классов как: DataTable, DataView, DataColumn, DataRow, DataRowView, DataRelation, Constraint.

Объект DataSet заполняется данными из БД с помощью объекта DataAdapter, у которого заданы свойства Connection и Command. DataSet может сохранять свое содержимое также в XML (опционально вместе с XSD схемой) или получать данные из XML.

ADO.NET поддерживает работу с отсоединенными наборами данных, что крайне важно при использовании масштабируемых веб-приложений.

Такая возможность реализуется с помощью класса DataSet совместно с классом DataAdapter.

Одной из важнейших составляющих технологии ADO.NET является поставщик данных. По-сути, это набор классов, предназначенных для взаимодействия с источником данных определенного типа. Использование разных поставщиков данных делает ADO.NET очень гибкой и расширяемой.

Ключевые слова: *Сервер, HTTP, интерфейс, программирование, СУБД, декодирование, сценарии, веб-приложение, идентификация,WEB.*

Контрольные вопросы:

1. Что такое плагин и для чего он используется?
2. На какие этапы делится выполнение программы?
3. Опишите , что такое скрипты и их задачи. Объясните пошагово схему типичного сценария работы веб-приложения
4. С помощью каких объектов обрабатывается событие веб-запроса в ASP?
5. Язык PHP.Синтаксис языка PHP.
6. Какие типы данных относятся к скалярным?
7. В чем разница между Data Provider и Data Sets?

ГЛАВА 14. ПРОГРАММИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ СИСТЕМ УПРАВЛЕНИЯ КОНТЕНТОМ И ФРЕЙМВОРКОВ

Современные сайты, в подавляющем большинстве, динамичны, т. е. реализуют идею динамического формирования отображаемых для посетителей данных, а потому для разработки таких сайтов чаще всего используются CMS – системы управления контентом. Однако есть и немного иной подход к разработке динамичных сайтов — использование CMF. Что такое CMF? Зачем нужны CMF? Какие самые популярные CMF?

CMF – Content Management Framework — это, согласно наиболее распространенному определению, фреймворк-система для управления содержимым сайта, а также инструментарий для создания систем управления контентом или же веб-приложений вообще.

Фреймворк (framework) — это, выражаясь простыми словами, некоторое программное обеспечение, позволяющие объединить большое количество разных по назначению компонентов большого программного проекта. Нередко в качестве синонима термину «фреймворк» употребляется термин «каркас».

Каркас может включать всевозможные библиотеки кодов, вспомогательные программы, язык программирования, язык сценариев. Объединение разных компонентов программного проекта обычно

происходит за счет использования единого API (application programming interface – интерфейс прикладного программирования).

Таким образом, CMF, как фреймворк-система, обладает присущими фреймворкам свойствами. Также можно сказать, что CMF — это понятие более широкое чем CMS, и каждая CMF является CMS, однако не каждая CMS — это CMF. Примеры CMF систем представлены в таблице:

Название	**Платформа**	**Поддерживаемые СУБД**
CakePHP	PHP	PostgreSQL, MySQL, SQLite, MS SQL, Oracle
Joomla!	PHP	MySQL
Catalyst	Perl	PostgreSQL, MySQL, SQLite, MS SQL, Oracle
ZendFramework	PHP	PostgreSQL,MariaDB, MySQL, SQLite, MS SQL, Oracle
Ruby on Rails	Ruby	MySQL, PostgreSQL, SQLite, Firebird, Oracle, SQL Server, DB2

Нередко, CMF системы обладающие достаточным функционалом для разработки полноценных веб-приложений и легких в управлении сайтов различных типов, называют CMF/CMS системами, поскольку они объединяют в себе возможности обеих видов систем. Примеры CMF/CMS и CMF систем представлены в таблице:

Название	**Платформа**	**Поддерживаемые СУБД**
MODx	PHP	MySQL
Drupal	PHP	MySQL, PostgreSQL
eZ publish	PHP	MySQL, PostgreSQL
TYPO3	PHP	MySQL, PostgreSQL

В CMF/CMS системах можно выделить несколько характерных особенностей.

Детерминированная внутренняя архитектура. В CMF/CMS системах внутренняя архитектура имеет развитые механизмы абстакции, не зависящие от CMS-образующих модулей. Это значит, что сопровождать проект, выполненный на основе CMF/CMS гораздо проще проекта, сделанного на «чистой» CMS.

Многофункциональность. Веб-сайты и веб-приложения, выполненные на основе CMF/CMS обладают высокой степенью индивидуализации: каждый проект может быть адаптирован применительно к конкретной ситуации. Многофункциональность CMF/CMS систем позволяет создавать на их основе любые интернет-проекты, от небольших сайтов-визиток до разветвленных порталов или интернет-магазинов.

Расширяемость и совместимость. Существующий функционал CMF/CMS систем может быть расширен за счет интеграции дополнительных модулей и программных кодов. Большинство CMF/CMS поддерживают работу с различными СУБД (MySQL, Oracle, PostgreSQL и др.), выполняют

трансляцию данных в любой требуемый формат (XHTML, JSON-структуры, PDF, XLS, RTF и т.д.).

Шаблонизация. В CMF/CMS поддерживается шаблонизация — удобное создание, а также интеграция шаблонов представления (дизайна) без необходимости затрагивать программное ядро или содержимое разрабатываемого сайта.

Удобство использования. CMF/CMS могут представлять собой как системы с произвольным, достаточно расширяемым и настраиваемым набором функциональных возможностей, так и дистрибутивы, удобные для использования непрофессиональными программистами.

Владельцы веб-сайтов на CMF/CMS получают систему настройки web-интерфейса, визуальный редактор для наполнения контентом, систему хранения и получения информации, систему упорядочивания информации, систему управления пользователями — т. е. все возможности, характерные для CMS.

Астоящем пособии мы покажем работу с системой управления контентом Drupal, которая может использоваться и как фреймворк.

Система управления контентом Drupal

Drupal (Друпал) — система управления содержимым, используемая также как каркас для веб-приложений (CMF), написанная на языке PHP и использующая в качестве хранилища данных реляционную базу данных (поддерживаются MySQL, PostgreSQL и др).

Исходя из многолетнего опыта практического применения различных CMS настоятельно рекомендую использовать для создания и управления сайтами, именно CMS Drupal.

Drupal настолько универсален, благодаря своей модульной архитектуре, что может являться основой, как визитки так и высоко нагруженного портала с сотней тысяч посетителей в сутки.

Это мощный инструмент, поддерживаемый профессиональным сообществом всего мира и являющийся свободным программным обеспечением. Это значит, что заказчик не платит за лицензию при использовании CMS Drupal и имеет поддержку мирового сообщества. В отличие от таких CMS как 1С-Битрикс и ей подобных, лицензия, на которую стоит в среднем $1000 на год.

Drupal — широко известная во всем мире система. Поэтому при передаче готового ресурса от разработчика к заказчику, последний легко найдет специалиста для сопровождения или научитесь сами, благо обучающих материалов в сети очень много.

Как работает Drupal

С концептуальной точки зрения стек Drupal выглядит, как показано на рис. 14.1. Drupal – это своего рода промежуточный уровень между внутренней реализацией (обеспечивающей работу Интернета) и интерфейсом (который видят посетители в окне веб-броузера).

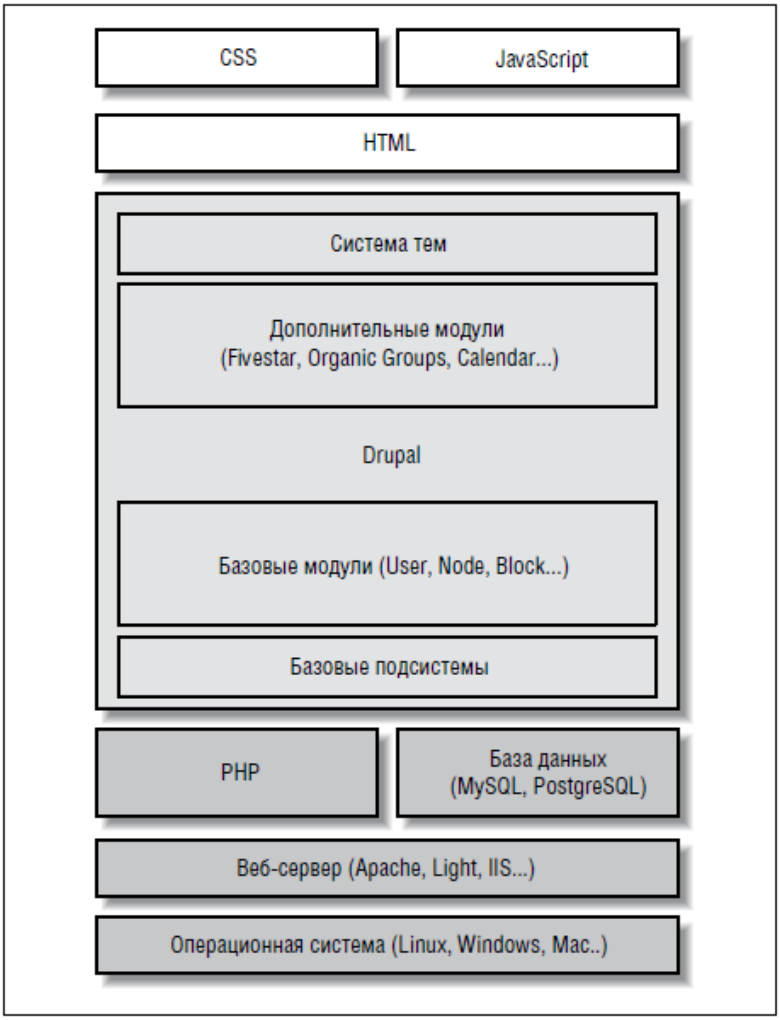


Рис. 14.1. Расположение концептуальных уровней системы Drupal относительно других уровней веб-сайта

На нижних уровнях находятся операционная система, веб-сервер, база данных и PHP. Операционная система обеспечивает работоспособность веб-сайта: здесь решаются такие низкоуровневые задачи, как обработка сетевых подключений, файлов и разграничение прав доступа к файлам. Веб-сервер обеспечивает доступность вашего компьютера в Интернете и служит для предоставления требуемой информации, когда посетитель выполняет переход по адресу http://www.example.com. База данных хранит данные: все содержимое веб-сайта, учетные записи пользователей и параметры настройки – в одном месте. А PHP – это язык программирования, который динамически генерирует страницы и перемещает информацию от базы данных к веб-серверу.

Сама система Drupal также состоит из множества слоев. На самом нижнем уровне она реализует дополнительные возможности поверх PHP, добавляя некоторые подсистемы, такие как обработка сеансов пользователей и механизм аутентификации, фильтр подсистемы безопасности и подсистема шаблонов отображения. Над этим слоем находится уровень дополнительных функциональных возможностей, называемых модулями.

Модули расширяют возможности Drupal и могут воспроизводить содержимое любой страницы. Но, прежде чем страница появится перед

пользователем, она пройдет через систему тем оформления, которая позволяет модифицировать оформление страниц и способна удовлетворить запросы самых взыскательных дизайнеров.

Система тем выводит содержимое страницы, обычно в формате XHTML, хотя поддерживаются и другие форматы. Для управления расположением элементов, цветом и шрифтами для заданной страницы используются каскадные таблицы стилей, а работоспособность динамических элементов интерфейса, таких как сворачиваемые наборы полей в формах и буксируемые строки таблиц в административном интерфейсе Drupal, обеспечивается с помощью JavaScript.

Модули

Практически все в системе Drupal вращается вокруг понятия модуля, который является файлом с программным кодом на языке PHP, реализующим функциональные возможности, с которыми система Drupal умеет работать. Все функциональные возможности Drupal, доступные администратору или конечному пользователю, от фундаментальных особенностей, таких как возможность выполнять вход в систему, добавлять новое содержимое в динамические фотогалереи и сложные ситемы голосования, – все они реализованы в виде модулей. Примерами модулей могут служить: модуль Contact, который реализует форму с контактной информацией, и модуль User, который реализует процедуру аутентификации пользователя и проверку прав доступа. В других системах управления содержимым модули также называются дополнениями или расширениями.

Существует два типа модулей: «базовые» модули, которые включены непосредственно в состав системы Drupal, и «сторонние» модули, которые создаются сообществом Drupal и могут загружаться и устанавливаться по отдельности. Кроме нескольких обязательных базовых модулей все остальные модули могут подключаться и отключаться, в зависимости от потребностей веб-сайта.

Несмотря на то, что существуют модули с функциональностью «под ключ», тем не менее разработчики из сообщества Drupal обычно отдают предпочтение разработке модулей, которые выполняют единственную функцию, но делают это хорошо и при этом допускают возможность комбинирования с другими модулями. Это означает, что у разработчика имеется практически неограниченный контроль над внешним видом и поведением веб-сайта. Возможности галереи изображений не ограничены представлениями разработчика о том, как она должна выглядеть и как действовать. Можно добавить систему рейтингов или возможность оставлять комментарии, при желании – сортировать изображения по типу фотокамеры, а не по дате. Однако для достижения этой гибкости разработчику придется «построить» функциональность в системе Drupal, объединив несколько различных модулей и выполнив настройку их параметров, а не просто щелкнуть на флажке «галерея изображений» и оставить его отмеченным. Мощь системы Drupal порождает сложность ее изучения, которой нет в

других пакетах CMS, а изобилие доступных модулей может отпугнуть при попытке определить, чем же стоит воспользоваться в конкретном случае.

Пользователи

Следующий строительный блок веб-сайта на базе системы Drupal – это понятие пользователя. В случае простого веб-сайта брошюрного типа, который будет сопровождаться единственным администратором и посещаться только потенциальными клиентами, можно создать единственную учетную запись пользователя – для самого администратора.

При создании сайтов, подразумевающих общение членов сообщества, вам придется настроить систему Drupal, чтобы позволить всем желающим пользоваться сайтом регистрироваться на нем и создавать свои учетные записи.

Каждому добавляемому пользователю может быть присвоена настраиваемая роль, такая как «редактор», «платный клиент» или «важная персона». Каждой роли могут быть присвоены права на выполнение различных действий на веб-сайте: посещение определенных страниц, просмотр содержимого определенного вида, разрешение оставлять комментарии к существующему содержимому, заполнять профиль пользователя и даже создавать новые учетные записи и управлять их правами доступа. По умолчанию в системе Drupal имеются две предопределенные роли: зарегистрированный пользователь и анонимный пользователь. Любой создаваемой учетной записи на сайте автоматически присваивается роль «зарегистрированный пользователь», а любому посетителю, для которого еще не создана учетная запись (или который еще не прошел процедуру аутентификации со своим именем пользователя и паролем), присваивается роль «анонимный пользователь».

Содержимое (узлы)

Узлы – это следующий строительный блок в системе Drupal, причем один из самых важных. Одна из важнейших составляющих проектирования любого сайта на базе Drupal заключается в выяснении, с какими конкретными видами содержимого (которые в Drupal называются «типами содержимого») вам предстоит работать. Практически в каждом конкретном случае будут присутствовать узлы различных видов.

Все узлы независимо от типов содержимого, хранящегося в них, обладают несколькими общими базовыми свойствами:

- Автор (пользователь сайта, создавший содержимое)
- Дата создания
- Заголовок
- Тело содержимого

Вам требуется создать страницу с конфиденциальными сведениями о вашей компании? Это узел. Вам требуется дать пользователям возможность оставлять сообщения в блоге? Каждое такое сообщение – это узел. Пользователи будут оставлять ссылки на интересные материалы, размещенные где-то в Интернете? Каждая такая ссылка, как и следовало ожидать, будет храниться как узел.

В дополнение к общим базовым свойствам все узлы могут обладать определенными особенностями, встроенными в систему Drupal, такими как флаги, указывающие, будут ли публиковаться данные узлы, и настройки, управляющие способом отображения узлов каждого типа.

Права на создание и редактирование содержимого узлов каждого типа также могут быть присвоены различным пользовательским ролям, например пользователи с ролью «blogger» могли бы создавать содержимое с типом «Blog entry» (Сообщение в блоге), но только пользователи с ролями «administrator» (администратор) или «editor» (редактор) могли бы создавать узлы с типом «News» (Новости).

В состав системы Drupal входят два предопределенных типа узлов: «Page» (Страница) и «Story» (Статья). Они не играют какую-то специальную роль – они просто предлагают стандартный набор особенностей для всех узлов и ничего больше. Единственное отличие между этими двумя типами узлов заключается в параметрах настройки по умолчанию. Узлы «Page» не отображают информации об авторе или дате создания. Они отлично подходят для размещения такого содержимого, как «About Us» (О компании) и «Terms of Service» (Условия обслуживания), у которого не может быть определенного автора. Узлы типа Story отображают эту информацию, а кроме того, настроены так, чтобы появляться на главной странице сайта в случае их публикации. Результатом является блог-подобный список последних статей на сайте.

Посредством административных инструментов Drupal управления содержимым можно создавать другие «простые» типы узлов. Многие администраторы создают типы «news» (новости) или «announcement» (объявление) – для публикации официальных объявлений, тогда как другие пользователи могут публиковать узлы, содержащие статьи. Однако как быть, если вам потребуется хранить больше информации об узле, чем его заголовок и тело? Дополнительные модули могут привносить в систему содержимого Drupal новые виды узлов, обладающих более широкими возможностями. Одним из таких модулей (поставляемым в составе Drupal) является модуль «Poll». Когда пользователь создает новый узел типа «Poll», вместо обычного «тела содержимого» он создает список вопросов для голосования. Узлы типа «Poll» при отображении выглядят как форма для голосования и автоматически отображают число голосов по каждому пункту.

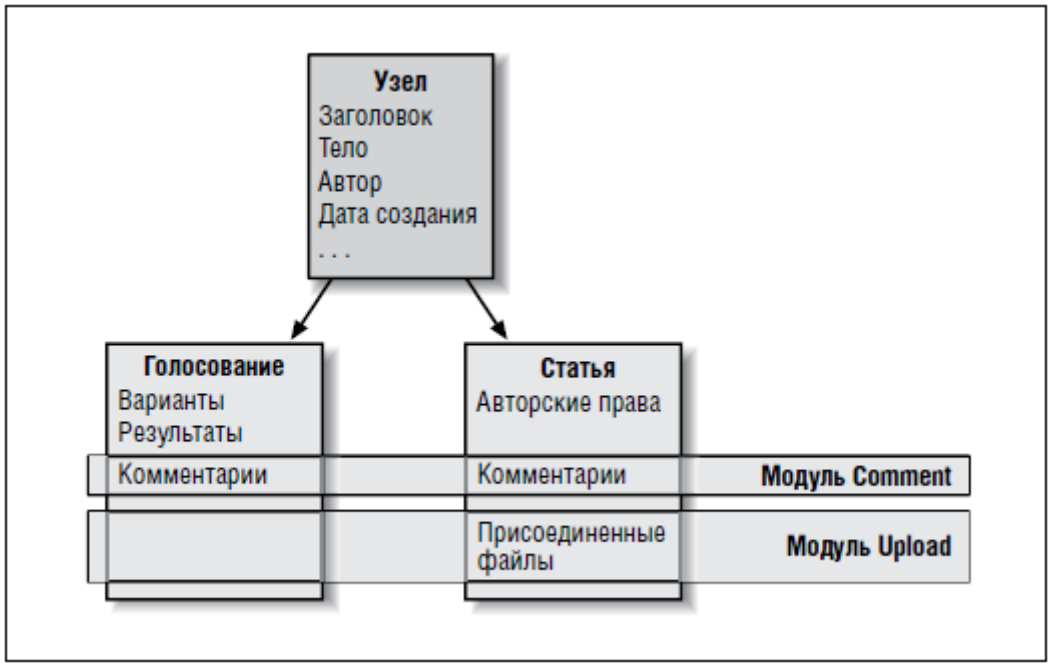


Рис. 14.2. Концепции особенности управления

Все узлы в системе обладают набором общих базовых свойств; узлы могут определять дополнительные поля, а модули могут придавать узлам дополнительные функциональные возможности

Кроме того, другие модули могут реализовывать новые свойства узлов, такие как комментарии, рейтинги, поля выгрузки файлов и другие. Из панели управления вы можете определить, узлы каких типов будут обладать этими особенностями. Эта концепция иллюстрируется на рис. 14.2.

Возможность добавления новых свойств с помощью новых модулей поверх существующей системы узлов означает, что все виды содержимого в системе Drupal относятся к одной базовой платформе, которая является одной из самых сильных сторон Drupal. Такие возможности, как поиск, рейтинг и комментарии, являются готовыми к использованию компонентами в любых новых типах узлов, которые вы только сможете определить, потому что за кулисами система Drupal всегда знает, как обращаться с базовыми элементами этих компонентов – узлами.

Использование дополнительных модулей для добавления новых типов узлов или для добавления новых полей к существующим типам является распространенной задачей в Drupal. Продвигаясь по книге, мы рассмотрим некоторые из имеющихся сотен дополнительных модулей, и вы узнаете, как создавать сложные типы содержимого, используя базовые инструменты.

Способы организации содержимого

Еще одним важным строительным блоком является весь комплекс инструментов и приемов организации узлов, составляющих информационное наполнение сайта. Веб-сайты первого поколения группировали страницы с помощью папок и каталогов. Сайты второго поколения для отображения содержимого различных типов использовали отдельные сценарии. В свою очередь, система Drupal практически все содержимое представляет как набор

узлов. Как можно разбить сайт на отдельные разделы по темам, на блоги конкретных пользователей или реализовать некоторую иную организационную схему?

Во-первых, каждый отдельный узел на сайте получает свой собственный адрес URL. По умолчанию этот адрес URL выглядит примерно так: http://www.example.com/node/1. Эти адреса URL можно преобразовать в более дружественный для пользователя вид, например: http://www.example.com/about, с помощью модуля Path, встроенного в систему Drupal. В организационных целях все эти узлы интерпретируются как единый «пул» содержимого. Любые другие страницы на сайте – тематические обзоры, последние новости и другие – создаются посредством извлечения списков узлов, соответствующих определенным критериям, и отображаются различными способами. Ниже приводится несколько примеров:

Главная страница

По умолчанию главная страница сайта, построенного на базе Drupal, представляет собой обзор 10 самых последних статей. Для его создания Drupal отыскивает в пуле содержимого узлы с флагами «Published» (Опубликовано) и «Promote to front page» (Представить на главной странице), установленными в значение true. Кроме того, этот список сортируется так, чтобы узлы с флагом «Sticky» (Закреплен) всегда находились на самом верху; эта возможность особенно удобна для публикации горячих новостей или важных объявлений, которые должны быть видны каждому пользователю.

Модуль Taxonomy

Выше упоминалось, что модули могут добавлять к узлам новые элементы информации; именно это и делает модуль Taxonomy. Он позволяет администратору сайта определять категории тем, с которыми могут быть связаны узлы при их создании, а также ключевые слова для облака тегов в стиле блогов. Этот модуль можно использовать для создания предопределенного набора «Regions» (Регионы), представленного в виде поля в новостях или поля «Tags» (Теги) для блогеров, которое они будут заполнять вручную при отправке сообщений. Все это в терминах модуля Taxonomy называется «рубриками» и позволяет отнести страницу к той или иной рубрике на сайте. Когда посетитель просматривает одну из таких страниц, система Drupal извлекает список всех узлов, отнесенных к этой рубрике.

Модуль Blog

Модуль Blog, встроенный в систему Drupal, реализует многопользовательскую систему блогов, делая всего три вещи. Во-первых, он добавляет новый тип узлов, который называется «Blog post» (Сообщение в блоге). Во-вторых, с адресом http://www.example.com/blog он ассоциирует страницу со списком, в котором отображаются узлы типа «Blog» (Блог), для которых флаг «Published» (Опубликовано) установлен в значение true. (Если сообщение в блоге имеет флаг «Published to front page» (Публиковать на главной странице) со значением true, оно будет также отображаться на

главной странице; система Drupal никогда не скрывает содержимое одной страницы только потому, что оно отображается на другой.) В-третьих, для каждого пользователя сайта он предоставляет отдельную страницу, где отображаются сообщения блога, принадлежащие только этому пользователю. Например, на странице http://www.example.com/blog/1 будут отображаться все узлы с записями в блоге, которые были созданы и опубликованы пользователем 1, то есть администратором.

В составе системы Drupal поставляется ряд других модулей, которые обеспечивают возможность организации узлов различными способами, а кроме того, существуют сотни дополнительных, доступных для загрузки, модулей, позволяющие организовать сайт разными способами. Важно запомнить, что практически все «страницы» в Drupal представляют собой одну из двух вещей: определенный узел содержимого или список узлов, обладающих определенным общим набором свойств.

Типы содержимого поддержки

Помимо содержимого и списков содержимого существуют также различные способы дополнения содержимого страницы. Двумя такими способами поддержки содержимого, включенными в ядро Drupal, являются комментарии и блоки.

Комментарии – это обычные отклики пользователей на некоторые фрагменты содержимого, они существуют только в привязке к этому содержимому. Пользователи могут отправлять комментарии, чтобы представить свои мысли относительно темы узла, как это часто делается, когда в сообщении блога или на форуме поднимается спорная тема. Подобно узлам, хотя и в меньшей степени, комментарии могут с помощью сторонних модулей расширяться дополнительными свойствами, такими как рейтинги или выгружаемые файлы.

Комментарии имеют большое число параметров настройки: комментарии могут отображаться в виде древовидной структуры или в виде плоского списка; комментарии могут сортироваться по дате и времени создания как в прямом, так и в обратном порядке; анонимным пользователям может быть предоставлена возможность оставлять комментарии, причем в последнем случае можно сделать предоставление контактной информации обязательным.

Блоки – это виджеты, которые присутствуют в таких областях страницы, как боковые меню, нижние колонтитулы и заголовки. Обычно они используются для отображения полезных ссылок или динамических списков, таких как «Most popular content» (Наиболее читаемые статьи), «Latest comments» (Последние комментарии), и похожих элементов. Блок пользователей управляет доступностью информации для посетителей вашего сайта; узлы отвечают за отображение содержимого; а блоки помогают встроить отдельный элемент содержимого в контекст структуры вашего сайта.

В большинстве случаев блоки отображают различное содержимое для разных зарегистрировавшихся пользователей: например, блок «Comments by

your buddies» (Комментарии ваших друзей) будет отображать список сообщений, оставленных теми пользователями, которых текущий пользователь добавил в список своих друзей. Обычно все пользователи, выполнившие вход, видят различные списки. Кроме того, блоки можно настроить так, что они будут появляться только на определенных страницах или, наоборот, будут скрыты только на определенных страницах.

Получение справки

Легко полагаться на ту функциональность, которую можно получить бесплатно, используя открытое программное обеспечение. Но не стоит забывать, что само сообщество пользователей Drupal может стать важным строительным блоком вашего веб-сайта!

По мере изучения практических примеров в этой книге вы можете столкнуться с некоторыми проблемами, характерными для вашего окружения. Проблемы могут также возникать при переходе на использование новых версий модулей. К счастью, у сообщества Drupal имеется богатый выбор ресурсов, где можно получить помощь в исследовании самых противных ошибок, с которыми вы только можете столкнуться:

- Руководства Drupal, расположенные по адресу http://drupal.org/handbooks, содержат массу информации обо всем, начиная от философии сообщества и заканчивая подробными сведениями, касающимися разработки Drupal. (Существует русскоязычный сайт Drupal.ru, где имеется похожий раздел http://drupal.ru/book)
- Руководство для начинающих, по адресу: http://drupal.org/gettingstarted, содержит некоторую информацию, которая будет вам особенно полезна в первые два часа знакомства с Drupal.
- Сборник вопросов и ответов по устранению проблем – по адресу: http://drupal.org/Troubleshooting-FAQ содержит полезные советы и рекомендации с расшифровкой сообщений об ошибках, с которыми вы можете столкнуться.
- За персональной помощью по любому вопросу, от подготовки к установке до проблем, связанных с обновлением, можно попробовать обратиться на форум «Support» (Поддержка), по адресу: http://drupal.org/forum/181.
- Если ваш вопрос касается определенного модуля, можно послать «запрос на поддержку» (или «отчет об ошибке», если это явная проблема) в очередь вопросов, который будет передан лицу, сопровождающему модуль. Полезный видеоурок о том, как использовать очередь вопросов на сайте Drupal.org, находится по адресу http://drupal.org/node/273658.
- На сайте irc.freenode.net имеется канал IRC #drupal-support для тех, кто предпочитает общаться в диалоговом режиме.

Установка и обновление Drupal

Первый шаг на пути к использованию Drupal, естественно, заключается в том, чтобы получить необходимое программное обеспечение и установить его. Система Drupal распространяется в комплекте со сценарием установки,

который проведет вас через несколько страниц сбора необходимой информации, затем создаст базу данных и файл с настройками сайта. Мы рассмотрим все операции, которые вам необходимо будет выполнить, чтобы установка прошла без сучка и задоринки, – вы увидите, что установка Drupal выполняется быстро и просто, как только будет собрана вся необходимая информация.

После установки и запуска системы Drupal очень важно обеспечить своевременное обновление программного обеспечения сайта. Периодически выходят новые версии сторонних модулей и ядра Drupal, в которых устраняются критические проблемы, связанные с безопасностью, поэтому очень важно своевременно выполнять обновления по мере их выхода. Мы рассмотрим модуль Update Status, встроенный в Drupal 6, который будет извещать вас о появлении обновлений для вашего сайта, а также поговорим о том, как выполнять обновление отдельных модулей и самого ядра Drupal одной версии к другой.

Перед установкой

Перед установкой Drupal важно убедиться, что вы действительно можете ее выполнить, для чего требуется иметь некоторое представление о структуре системы Drupal. В этом разделе мы рассмотрим перечень требований, предъявляемых системой Drupal, и остановимся на некоторых важных аспектах структуры дерева каталогов Drupal, которые желательно знать перед началом процесса установки.

Сбор необходимой информации

Перед установкой Drupal очень важно убедиться в соответствии определенным требованиям. Полный перечень требований приводится на странице http://drupal.org/requirements. Ниже приводится перечень основных требований, соответствие которым следует проверить перед установкой Drupal:

1. Наличие доступа к веб-хосту или к локальной среде разработки, со следующими возможностями:
   a. Веб-сервер, такой как Apache (http://httpd.apache.org), способный передавать страницы Drupal броузеру. Кроме того, наличие доступа к расширению mod_rewrite веб-сервера Apache позволит задействовать особенность Drupal «Clean URLs» (Чистые ссылки), которая выполняет приведение адресов URL вида http://www.example.com/index.php?q=contact к виду http://www.example.com/contact.
   b. PHP (http://php.net), динамический язык сценариев, являющийся движущей силой Drupal. Drupal 6 требует наличие поддержки PHP версии не ниже 4.3.5, при этом рекомендуется использовать версию PHP 5.2 или выше. На странице требований, на сайте Drupal.org, дается более полная информация об обязательных и рекомендуемых расширениях PHP, большая часть из которых включена по умолчанию.
   c. Сервер баз данных, такой как MySQL (http://mysql.com), где система Drupal будет хранить все содержимое сайта, свои данные и настройки, необходимые для нормальной работы.

2. Выпишите на листок бумаги следующую информацию о своем веб-хосте:

a. Ваши имя пользователя и пароль для (S)FTP или SSH, с помощью которых вы сможете записать файлы Drupal к себе.

b. Сведения о своем сервере баз данных, включая имя пользователя, пароль и название базы данных, – чтобы обеспечить Drupal возможностью соединения с базой данных. Кроме того, для некоторых веб-хостов необходимо указать дополнительные параметры соединения с базой данных, например указать имя удаленного хоста или порт подключения к серверу баз данных.

3. Прежде чем начинать установку Drupal, вам также потребуется создать базу данных, куда будет выполняться установка, – сама система Drupal не создает базу данных, так как для этой операции обычно требуются «повышенные» привилегии доступа к серверу баз данных. Систему Drupal можно установить либо в отдельную базу данных, либо в общую базу данных, которая используется также другими приложениями,, указав Drupal-префикс для имен таблиц; но вообще лучше, если для Drupal будет создана отдельная база данных. Обратитесь к своему поставщику услуг хостинга или к системному администратору, если вам необходима дополнительная информация о порядке создания новой базы данных, и выпишите ее название, так как оно потребуется немного позже. Кроме того, не забудьте выписать имя пользователя и пароль для доступа к базе данных.

Как только вы убедитесь, что все необходимое у вас уже имеется, можно приступать к установке.

Загрузка Drupal

Первый шаг перед установкой заключается в том, чтобы получить сам программный код Drupal. Вы можете получить исходный программный код Drupal на сайте книги http://usingdrupal.com/source_code или загрузить его непосредственно на сайте Drupal.org. Ниже описываются шаги, которые следует выполнить для загрузки программного кода с сайта Drupal.org:

1. Перейдите на страницу http://drupal.org и найдите несколько ссылок для загрузки Drupal. Они выделены на рис. 14.3. Щелкните на вкладке Download (Загрузить) в правом верхнем углу страницы.

2. На следующей странице перечислены все типы проектов, доступных для загрузки: модули, темы оформления, переводы и так далее. Щелкните на ссылке Drupal project (Проект Drupal), чтобы перейти на страницу ядра Drupal.

3. В таблице, изображенной на рис. 14.4, перечислены доступные версии в порядке от более новых к более старым. Если только вы не оказываете помощь в разработке Drupal, вам следует загружать версии, помеченные как Recommended (Рекомендуется). Это так называемые «стабильные версии». Чтобы иметь возможность использовать примеры из этой книги, щелкните на ссылке Download (Загрузить) для версии, помеченной как Recommended for 6.x (Рекомендуется для версии 6.x).

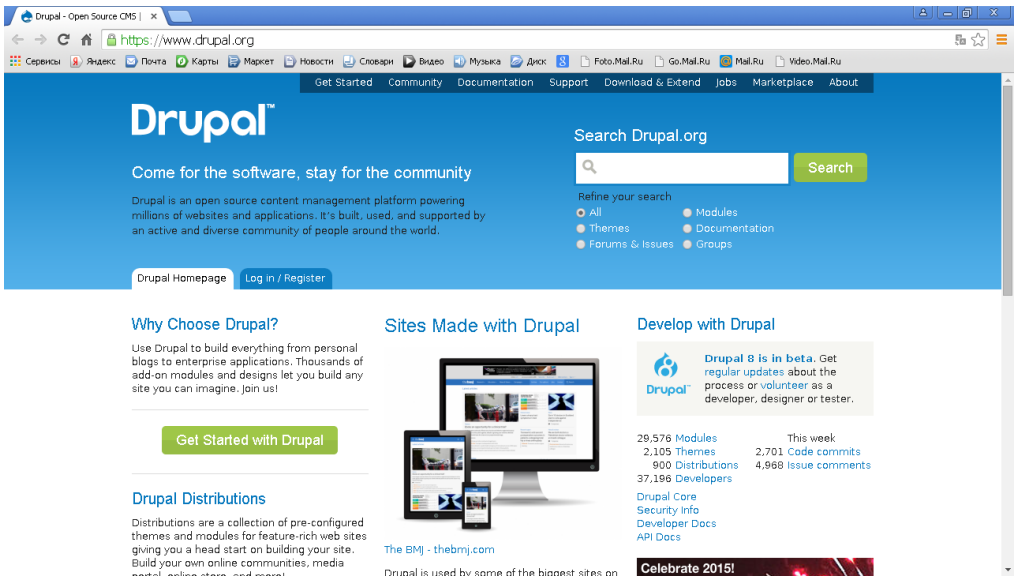


Рис. 14.3. Ссылки для загрузки на сайте Drupal.org

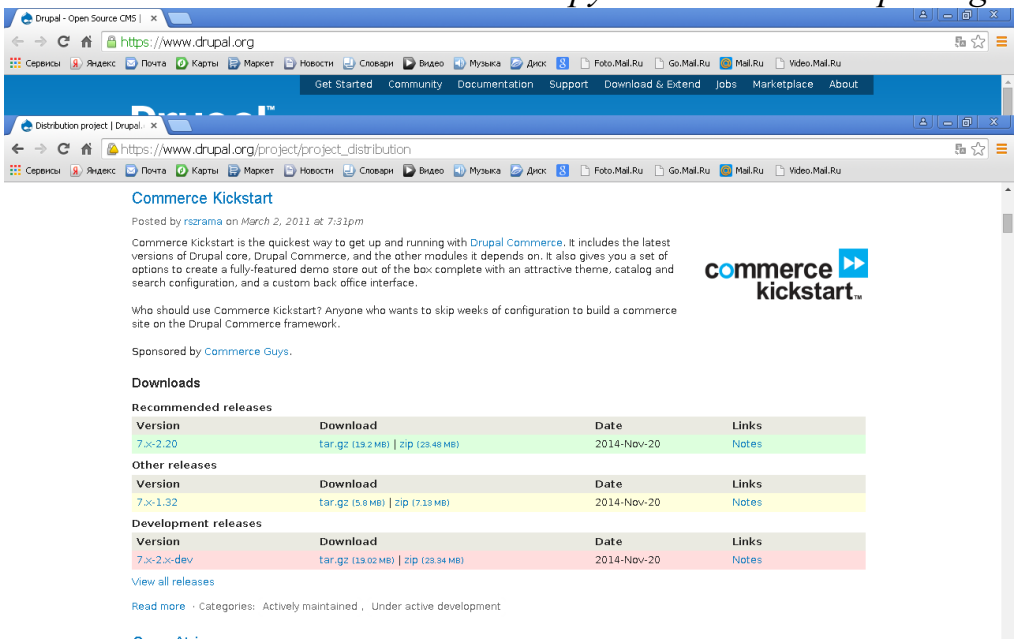


Рис. 14.4. Версии Drupal на странице загрузки

4. Файлы Drupal упакованы с помощью утилиты tar и сжаты с помощью утилиты gzip. В результате этого файл пакета имеет расширение tar.gz. Такие файлы очень напоминают архивы, созданные программой zip. Сохраните файл пакета и затем извлеките его содержимое с помощью предпочитаемого вами приложения для работы с архивами.

5. Поместите распакованные файлы на веб-сервер с помощью программы-клиента (S)FTP или зарегистрировавшись на сервере, и загрузите пакет непосредственно на сервер и распакуйте его там.

Файлы и каталоги Drupal

Теперь, когда вы загрузили пакет Drupal, следует потратить несколько минут, чтобы осмотреть его содержимое. Знакомство с основной структурой размещения важных файлов и каталогов поможет приоткрыть завесу тайны и получить некоторое представление о том, как все это действует. Когда вы откроете папку Drupal, вы увидите дерево каталогов, как показано на рис. A.5.

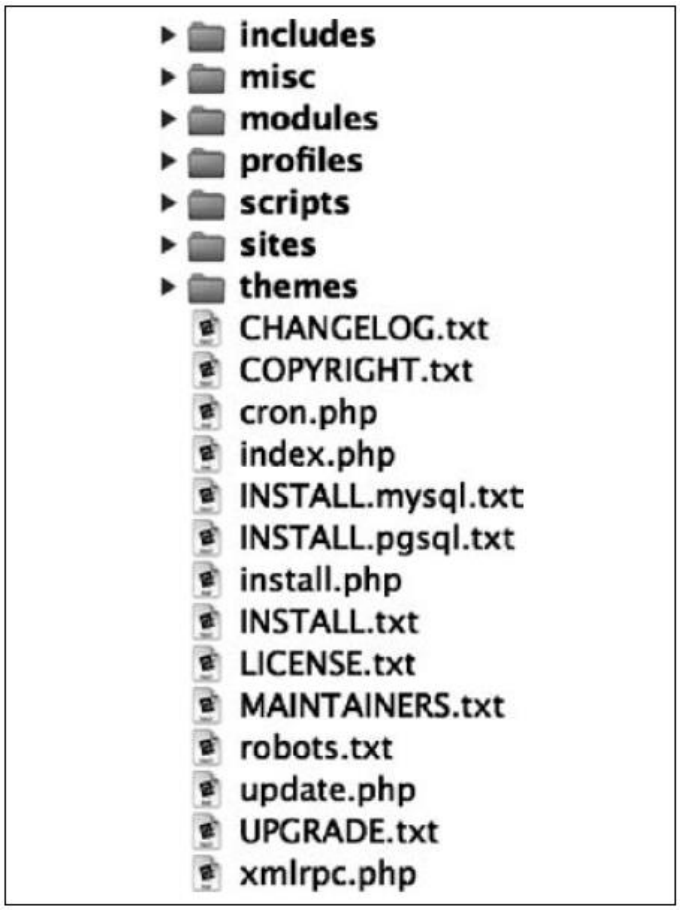


Рис. 14.5. Структура каталогов Drupal

Самыми важными для рассмотрения в этом разделе являются файлы установки и обновления, а также каталог sites. Файлы install.php и update.php – это два сценария, которые фактически выполняют операции, соответствующие их названиям. Так как они находятся в папке самого верхнего уровня, которая также называется корневым каталогом Drupal, вы можете обратиться к ним напрямую, введя в адресной строке броузера примерно такой адрес: http://example.com/install.php. Помимо сценариев здесь также присутствуют два текстовых файла, по одному для каждой операции: INSTALL.txt и UPGRADE.txt. Эти файлы, которые мы также будем рассматривать в этом приложении, содержат инструкции по использованию соответствующих им сценариев.

Большинство начинающих администраторов Drupal, взглянув на структуру каталогов, изображенную на рис. 14.5, решат размещать сторонние и собственные модули и темы оформления в каталоги modules и themes, находящиеся в корневом каталоге Drupal, соответственно.

Именно в них хранятся базовые модули и темы Drupal, поэтому их и нужно использовать, разве не так? Если вы поместите свои файлы в эти каталоги, они будут работать и Drupal будет в состоянии обнаружить их. Однако это приведет к появлению проблем при первой же попытке обновления до следующей версии с исправлениями в системе безопасности, так как перезаписывание этих каталогов новыми базовыми версиями приведет к уничтожению всех ваших изменений и модификаций. Лучше всего сохранять все сторонние и собственные модули и темы в каталоге sites.

Это означает, что вы должны создать новые каталоги modules и themes в каталоге sites/all, если только вы не создаете сложную архитектуру с несколькими сайтами (подробнее об этом рассказывается во врезке ниже), и помещать сторонние и собственные разработки в эти каталоги, как показано на рис.14.6. При таком подходе все файлы, имеющие значение только для вашего сайта, будут храниться отдельно и не будут смешиваться с файлами ядра. Это существенно упростит процедуру последующих обновлений.

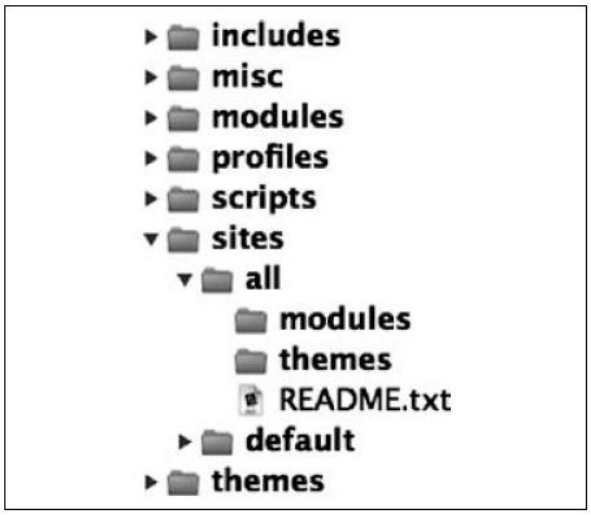


Рис. 14.6. Сторонние модули и темы оформления должны помещаться в каталог sites/all

Установка Drupal

Как только будут удовлетворены все требования и собрана вся необходимая информация, можно приступать к установке. Инструкции ниже предполагают, что вы уже создали базу данных, загрузили Drupal и поместили файлы, извлеченные из архива, на веб-сервер:

1. Для корректной установки Drupal необходимо создать файл с настройками. Для этого внутри каталога Drupal создайте копию файла sites/default/default.settings.php с именем sites/default/settings.php, то есть уберите префикс default. в начале имени файла. Вам также необходимо будет обеспечить доступность нового файла settings.php для записи, например, командой chmod 666 или chmod a+w sites/default/settings.php. На вашем веб-сервере должна иметься дополнительная информация о том, как обеспечить доступность файлов для записи. Загляните также в справочное руководство по системе Drupal, по адресу http://drupal.org/node/202483.
2. Теперь можно перейти по адресу http://www.example.com/install.php, чтобы начать процесс установки.
3. На первой странице мастера установки вы сможете выбрать язык, как показано на рис.14.7. По умолчанию доступен только английский язык. Однако вы можете загрузить другие переводы и выполнить установку Drupal на своем языке. В главе 8 вы найдете дополнительную информацию по установке и настройке многоязычных сайтов. Щелкните на ссылке Install Drupal in English (Установить Drupal на английском языке).

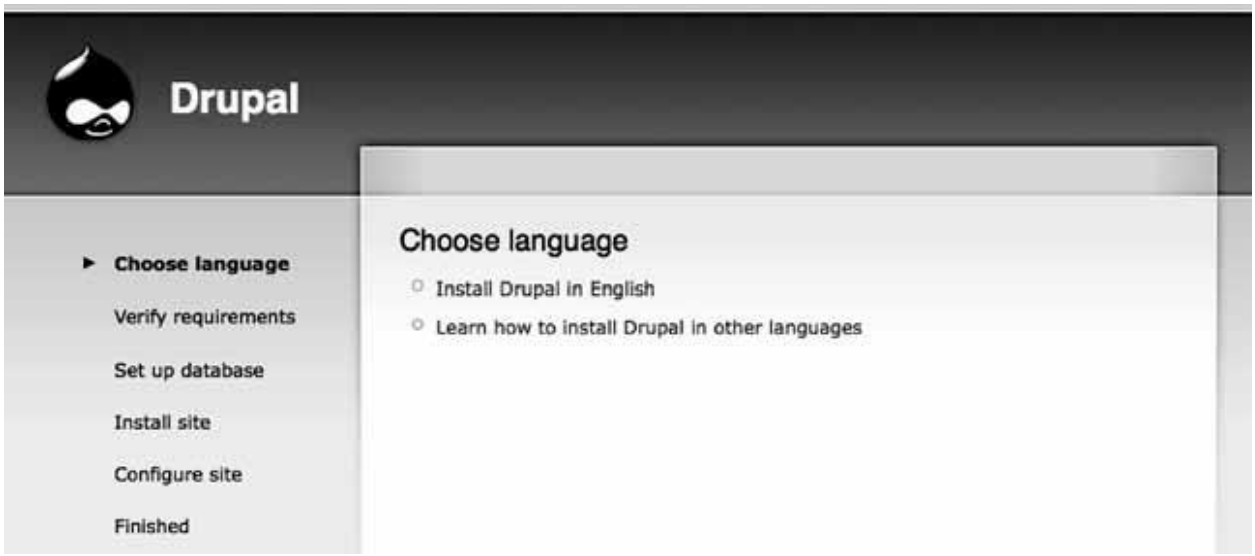


Рис. 14.7. Выбор языка в процессе установки Drupal

4. Если все идет, как надо, вы должны увидеть страницу, где вам будет предложено указать информацию о базе данных, как показано на рис.14.8.

5. Выше мы рекомендовали выписать на листе бумаги информацию о подключении к базе данных, включая имя пользователя и пароль. Теперь настало время использовать ее. Вам, как минимум, потребуется указать имя базы данных, имя пользователя базы данных и пароль. Если для вашего веб-сайта потребуется указать дополнительные сведения о базе данных, такие как имя хоста или порт подключения к серверу баз данных, разверните группу параметров Advanced options (Дополнительные настройки), чтобы ввести эти сведения. Как только вы введете всю необходимую информацию о базе данных, щелкните на кнопке Save and continue (Сохранить и продолжить).

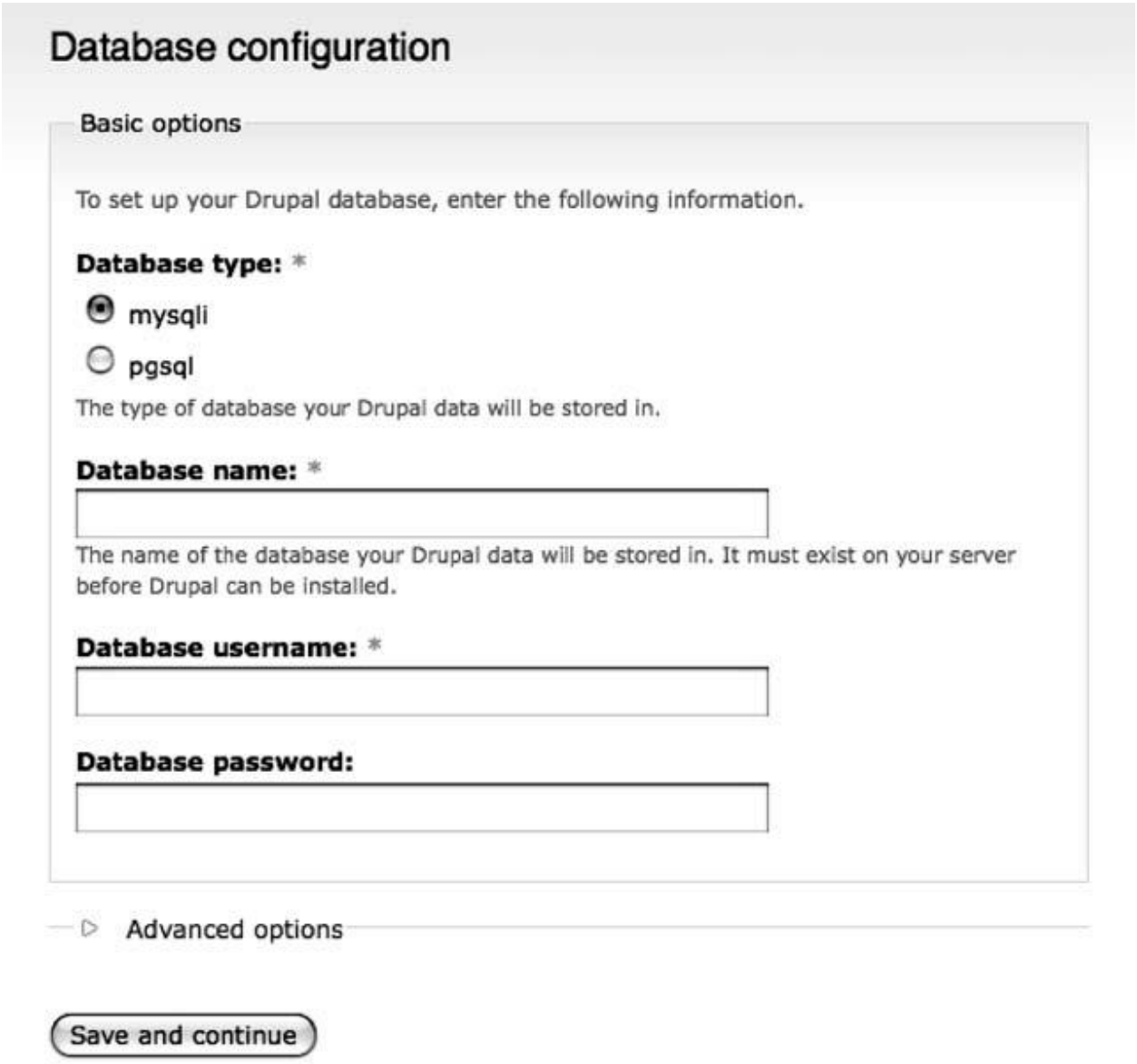


Рис.14.8. Настройка базы данных в процессе установки Drupal

6. Следующая страница, как показано на рис.14.9, содержит список основных параметров сайта, которые требуется настроить.

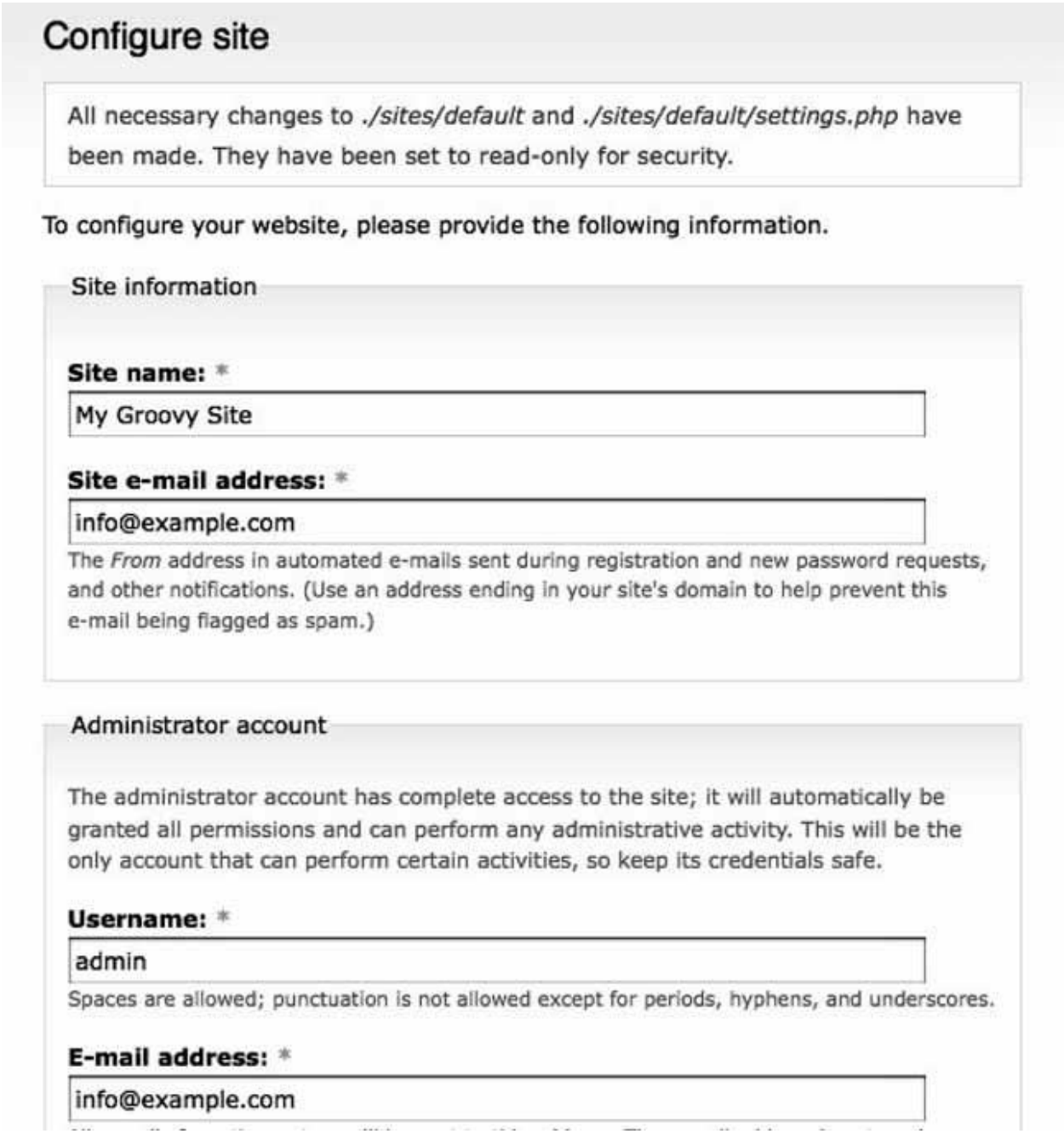


Рис.14.9. Настройка параметров в процессе установки Drupal

7. Сначала следует заполнить группу параметров Site information (О сайте). В этой группе присутствуют следующие важные глобальные параметры сайта:

Site name (Название сайта)

Это название по умолчанию будет отображаться в полосе заголовка на всех страницах, а также в верхнем левом углу всех страниц.

Site e-mail address (Адрес электронной почты сайта)

Все системы электронной почты будут отправлять сообщения с этого адреса, например, при регистрации нового пользователя.

8. Следующий шаг – создание учетной записи администратора. Администратор (также известен, как «пользователь № 1») – это «суперпользователь», который освобождается от любых проверок на право доступа и имеет неограниченную власть над сайтом. Поэтому для данной

учетной записи вы должны использовать пароль, который будет сложно подобрать (к счастью, Drupal попытается помочь вам, проверяя стойкость пароля по мере его ввода). Используйте эту учетную запись ответственно и только для решения административных задач. Для повседневного использования создайте вторую учетную запись с меньшим количеством привилегий.

9. Параметры в разделе Server settings (Настройки сервера) обычно можно оставить со значениями по умолчанию. В число этих параметров входят:

Default time zone (Часовой пояс по умолчанию)

Если пользователь не определит собственный часовой пояс в настройках своей учетной записи, все сообщения на сайте будут отображаться с указанием времени публикации в соответствии с указанным здесь часовым поясом. По умолчанию система Drupal установит часовой пояс, взяв его из настроек броузера, предполагая, что это совпадает с вашим желанием.

Clean URLs (Чистые ссылки)

Функция Clean URLs позволит вам иметь такие адреса URL, как http://example.com/about, вместо http://example.com/?q=about. Здесь будет выполнена проверка корректности настройки функции Clean URLs в веб-сервере, и если все будет в порядке, вы получите возможность включить этот параметр.

Update notifications (Сообщения об обновлениях)

Версия Drupal 6 поддерживает новую возможность – автоматически проверять наличие обновлений ядра системы, модулей и тем оформления и информировать вас об этом. Настоятельно рекомендуем включить этот параметр (включен по умолчанию), так как это позволит вовремя устанавливать обновления безопасности.

10. Как только будут введены все значения параметров, щелкните на кнопке Save and continue (Сохранить и продолжить).

11. Последняя страница сообщит вам, что установка завершена и можно приступать к настройке нового веб-сайта. Щелкните на ссылке your new site (ваш новый сайт), чтобы пуститься в путешествие по стране Drupal! На рис.14.10 показана начальная страница сайта Drupal сразу после установки.

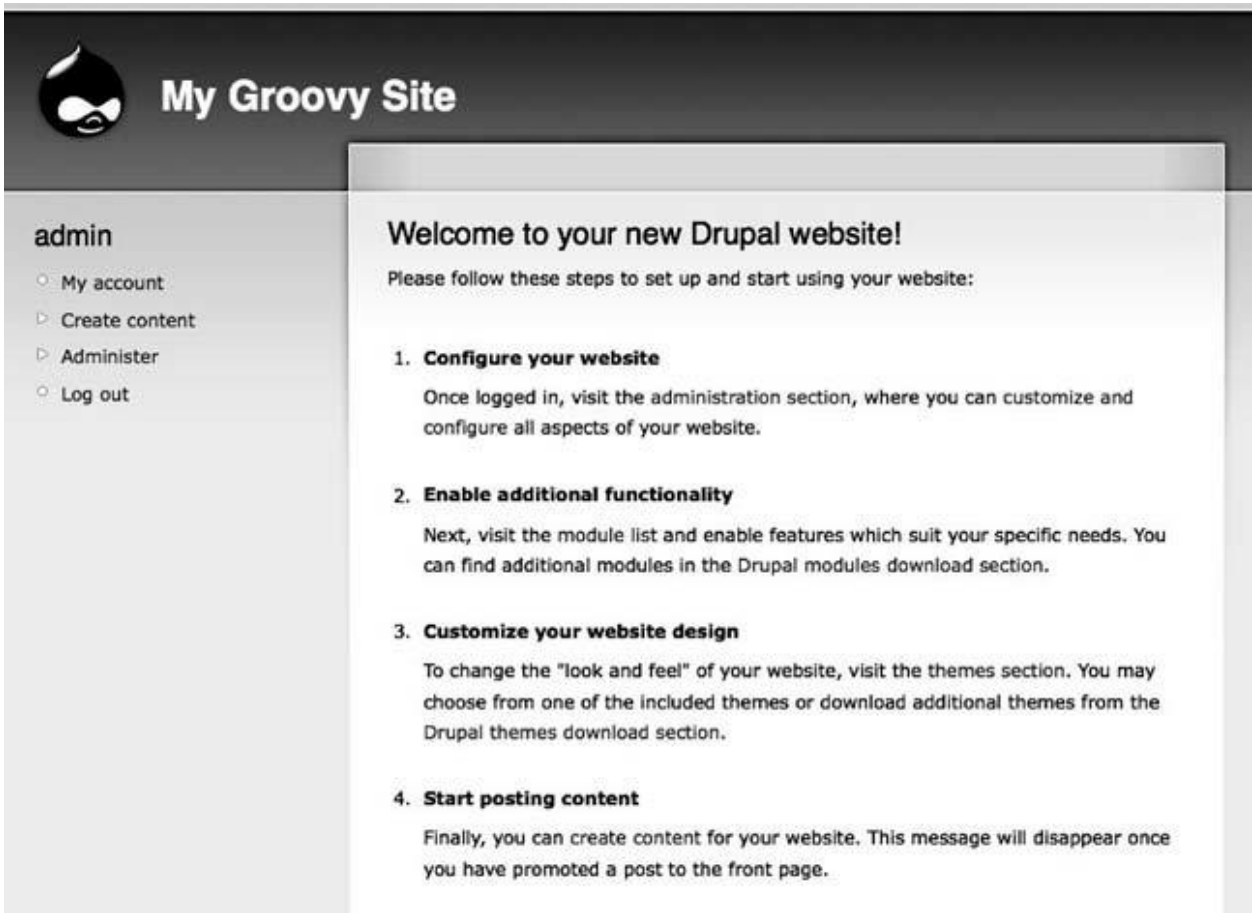


Рис.14.10. Вновь установленный сайт на базе Drupal

Обновление Drupal

Недостаточно просто установить Drupal, вам также потребуется своевременно обновлять его. Время от времени выходят новые версии модулей и ядра Drupal – чаще с исправлениями ошибок, иногда с реализацией дополнительных возможностей, а иногда с исправлениями критических проблем безопасности.

Нумерация версий

При обсуждении вопросов обновления системы совсем не лишним будет иметь некоторое представление о порядке нумерации версий Drupal.

Самые полные сведения по этой теме вы найдете на странице http://drupal.org/handbook/version-info, а в общих чертах порядок нумерации представлен на рис.14.11.

Каждый выпуск ядра Drupal «основной» версии получает новый номер: Drupal 5, Drupal 6, Drupal 7 и так далее. Новые основные версии Drupal

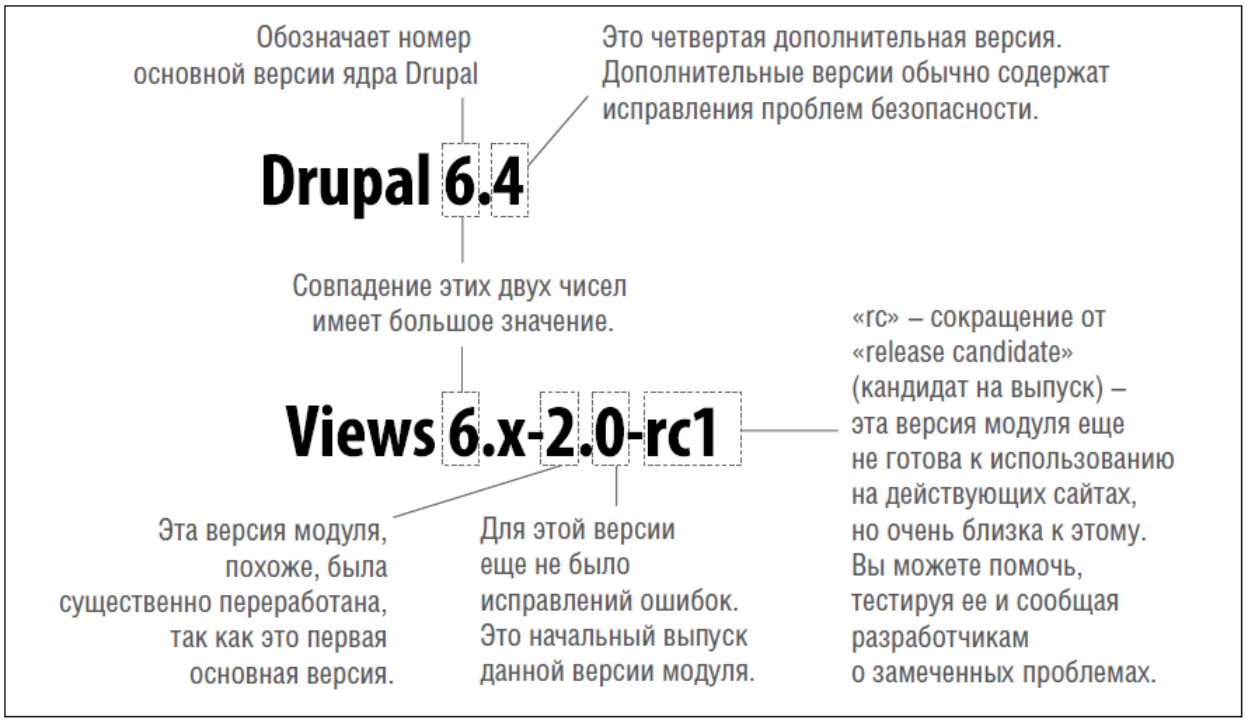


Рис.14.11. Порядок нумерации версий Drupal

Нумерация версий сторонних модулей, тем оформления и переводов производится в соответствии с такой схемой, как 6.x-1.3, где часть «6.x» обозначает основной номер версии Drupal, для которой предназначен этот модуль, тема или перевод – в данном случае Drupal 6. Число «1» обозначает «основной» номер версии стороннего модуля. А число «3» говорит о том, что это третий выпуск основной версии модуля с исправлениями ошибок.

Некоторые номера версий могут нести в себе «дополнительные» сведения о версии, например «-beta4» или «-rc2». В данном случае дополнения указывают, что модули пока находятся на этапе разработки, но их можно использовать для тестирования.

Обновление при переходе от одной дополнительной версии ядра Drupal или модуля к другой, например переход от версии Drupal 6.3 к версии 6.4 или от версии модуля Views 6.x-2.0 к версии 6.x-2.1 обычно проходит безболезненно, при условии, что обновления производятся регулярно. Но обновление при переходе от одной основной версии к другой, например от версии Drupal 6.3 к версии 7.0 или от версии модуля Organic Groups 6.x-1.0 к версии 6.x-2.0, и особенно к версии 7.x-1.0, требует особого внимания, так как обычно различия между версиями весьма существенны.

Запускаем Drupal

В этом разделе дается обзор возможностей системы, а также вводятся определения иногда не совсем понятных терминов и демонстрируется, как можно с помощью Drupal построить простой веб-сайт. К концу чтения вы будете знать, как выполнять в Drupal такие задачи администрирования, как настройка модулей, работа с типами содержимого и настройка системы навигации по сайту.

За помощью по этому вопросу обращайтесь к руководству для начинающих по адресу: http://drupal.org/gettingstarted.

Далее будут представлены следующие модули:

Node (базовый)

Позволяет добавлять новое содержимое и создавать свои типы содержимого.

Comment (базовый)

Дает пользователям возможность оставлять свои комментарии к элементу содержимого.

User (базовый)

Обеспечивает возможность регистрации пользователей, а также реализует надежные системы ролей и прав доступа

Block (базовый)

Добавляет динамические боковые врезки и другое дополнительное содержимое.

Menu (базовый)

Обслуживает систему навигации по веб-сайту на базе Drupal.

Path (базовый)

Обеспечивает поддержку дружественных адресов URL, таких как http://www.example.com/about , вместо http://www.example.com/node/

Administration Menu (http://drupal.org/project/admin_menu)

Реализует динамические раскрывающиеся меню для ускорения доступа к выполнению административных задач.

Contact (базовый)

Простая форма, которая может использоваться посетителями сайта для отправки своих вопросов владельцам веб-сайта.

Blog (базовый)

Реализует простой и быстрый многопользовательский блог

Taxonomy (базовый)

Мощная система классификации.

Filter (базовый)

Важный и часто недооцениваемый модуль, который является ключевым в системе безопасности Drupal.

FCKeditor (http://drupal.org/project/fckeditor)

Визуальный редактор («What You See Is What You Get», WYSIWYG – что вижу, то и получаю), который позволяет тем, кто не знаком с языком разметки HTML, создавать содержимое веб-сайта, применяя элементы форматирования.

IMCE (http://drupal.org/project/imce)

Дополнительный модуль, который может работать с такими редакторами, как FCKeditor, и позволяет легко добавлять изображения в содержимое веб-сайта.

Построенный веб-сайт будет выглядеть, как показано на рис.14.12 и по адресу: http://jumpstart.usingdrupal.com.

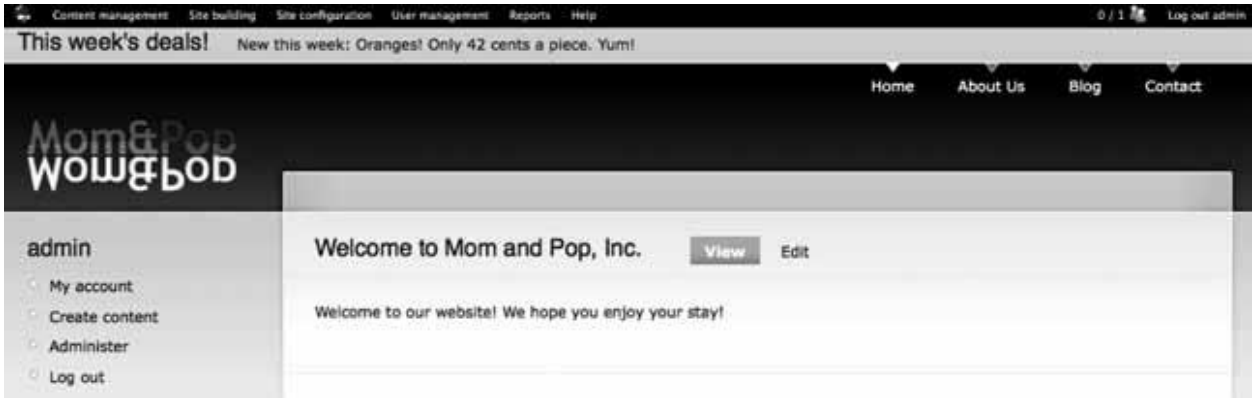


Рис.14.12. Законченный веб-сайт «Mom and Pop, Inc.»

Ключевые слова: *Веб-сайт, шаблонизация, контент, модуль, ссылки, Drupal, каталоги, файлы, CMF, узлы.*

Контрольные вопросы:

1. Что такое фреймворк? Дать описание CMF.
2. Система управления Drupal.Описать в виде схемы расположение концептуальных уровней системы Drupal относительно других уровней веб-сайта
3. Дать определение понятию модуль. Модули Taxonomy и Blog
4. Типы содержимого поддержки
5. Для чего служит функция Clean URLs?

ГЛАВА 15. ПУБЛИКАЦИИ ВЕБ-САЙТА, ВОПРОСЫ АДМИНИСТРИРОВАНИЯ И МОДЕРАЦИИ СОДЕРЖИМОГО

Варианты размещения Web-сайта в сети Интернет

Итак, ваш сайт готов, настала пора познакомить с ним обитателей сети Интернет. Как это сделать? Существует множество способов выложить ваш сайт в сети Интернет.

Виртуальный хостинг

Вашему сайту выделяется каталог на Web-сервере, из которого он и запускается. Название хостинг происходит от слова ХОСТ (от англ. Host – главный, хозяин). Хостом называется сервер, на котором размещается каталог с сайтом. Таких каталогов на сервере может быть несколько сотен и даже тысяч, но работать эта система будет так, будто для сайта выделен отдельный сервер, поэтому к слову хостинг часто добавляется слово «виртуальный».

Преимуществом такой системы является доступность. В зависимости от условий, предоставляемого дискового объема, ежемесячного Интернет-трафика и наличия различных дополнительных услуг виртуальный хостинг стоит от пяти долларов в месяц. Некоторые хостинг-провайдеры, т. е. фирмы,

занимающиеся предоставлением хостинга, предоставляют виртуальный хостинг и вовсе бесплатно.

Недостатки виртуального хостинга являются продолжением его достоинств. Вычислительную мощность сервера и пропускную способность канала, по которому он подключен к Интернету, вашему сайту придется делить еще с несколькими сайтами конкурентов. Поэтому скорость выполнения сценариев на вашем сайте и быстрота доступа к его страницам могут желать лучшего. В таких случаях лучше воспользоваться одним из других способов размещения сайта в Интернете.

Другим недостатком виртуального хостинга является зависимость от программного обеспечения, установленного на сервере. Вы не сможете поменять Web-сервер. установленный в системе, подключить поддержку дополнительных языков серверных сценариев и т.д.

Выделенный сервер

Этот вариант позволяет избавиться от большинства недостатков виртуального хостинга, для сайта выделяется отдельный сервер, на который устанавливается программное обеспечение, по выбору владельца сайта.

Основными недостатками этого способа размещения сайтов являются достаточно высокая цена такого решения и необходимость использовать только то оборудование, которое предоставляет хостинг-провайдер. Кроме того, свобода в выборе программного обеспечения сервера приносит с собой необходимость это программное обеспечение устанавливать и настраивать, так как хостинг-провайдер несет в данном случае ответственность только, аппаратную часть сервера.

Выделенные серверы используются, как правило, для размещения достаточно крупных коммерческих проектов, окупающих затраты на поддержание сервера и его администрирование. Еще одним возможным случаем использования выделенного сервера является размещение на нем нескольких Web-проектов, работающих одновременно. Использование отдельного сервера позволяет более гибко контролировать работу каждого из проектов по сравнению с использованием виртуального хостинга.

Совместное размещение

Совместное размещение (на сленге «**co-location**» - совместное размещение) это подключение сервера, принадлежащего заказчику, к Интернет-каналу хостинг провайдера. Содержимое сервера, как программное, так и аппаратное полностью определяется потребностями владельца сайта или сайтов, размещенных на сервере. Хостинг-провайдер предоставляет только выход в Интернет, электропитание сервера и физическое пространство размещения сервера.

Совместное размещение позволяет удовлетворить потребности любого, самого взыскательного сайтовладельца. Но этот вариант приносит с собой и некоторое количество проблем, связанных с тем, что обслуживание сервера полностью ложится на плечи владельца сайта, за работоспособность оборудования и программ хостинг-провайдер ответственности не несет.

Используется совместное размещение практически в тех же случаях, что и выделенный сервер: для крупных проектов С высокой посещаемостью либо для размещения нескольких Web-сайтов одного владельца,

Размещение сайта на своем компьютере

Одним из вариантов обеспечения доступа к своему сайту из Интернета, является размещение его на своем компьютере, подключенном к Сети.

Преимущества такого решения ясны. Не нужно платить за оборудование хостинг-провайдера и за размещение сервера, только за Интернет трафик. Можно устанавливать любое программное и аппаратное обеспечение без каких-либо ограничений. Сервер находится всегда под рукой, а не у хостинг-провайдера, и т. д.

Минусов у такого варианта тоже достаточно много. Необходимо быстрое и постоянное подключение к Интернету с выделенным IP-адресом. Нужно постоянно держать компьютер включенным. Нужно выделять часть ресурсов компьютера на обслуживание сервера и т.д.

На своем личном компьютере имеет смысл размещать только Web-сайты, для которых не будет критичным периодическое его пропадание из сети, при отключении электричества и перезагрузках, также не стоит размещать таким образом сайты с большой посещаемостью и сложными серверными сценариями, иначе нагрузка на компьютер будет слишком велика.

Особенности бесплатного хостинга

Самым простым и популярным способом размещения web-сайтов в сети был и остается виртуальный хостинг. Это решение подходит для большинства случаев, именно этот вариант мы и рассмотрим подробнее.

Как уже отмечалось ранее, виртуальный хостинг бывает платным и бесплатным.

Разумеется, ничего полностью бесплатного не существует и бесплатный хостинг на самом деле таким не является, Услуги бесплатных хостинг-провайдеров приходится оплачивать не деньгами, а чем-то другим.

Чаще всего за бесплатный хостинг приходится платить размещением рекламы на своем сайте. К тому же, хостинг-провайдер не отвечает ни за что, качество услуг может быть соответствующим.

Существует и другой вариант. когда бесплатный хостинг идет в дополнение к какой-нибудь другой услуге или товару. Например, большинство Интернет-провайдеров в числе прочих услуг предлагают своим клиентам бесплатно разместить Web-страницу. Этого, кстати, делать не стоит, поскольку в том случае, если вы поменяете Интернет-провайдера, вы потеряете и свою Web-страницу.

Кроме того, большинство хостинг-провайдеров предлагает на своих бесплатных тарифных планах очень маленький набор дополнительных услуг, маленький размер дискового пространства. ограничение на размер файлов и их типы. Эти ограничения вводятся, в том числе и для того, чтобы

пользователь перешел, со временем, на платный тарифный план, не меняя хостинг-провайдера.

Бесплатный хостинг идеально подходит для размещения ваших первых Web-сайтов, поскольку многочисленные эксперименты с сайтами и первые "блины комом", которых вы вряд ли избежите, не будут вам стоить ничего. В дальнейшем, когда эксперименты закончатся, вы сможете перейти на платный хостинг либо вообще поставить выделенный сервер, но для некоторых проектов можно так и ограничиться бесплатным хостингом. Если вы планируете впоследствии поменять хостинг-провайдера, то выбирайте хостинг с поддержкой сторонних доменов второго уровня, чтобы поменять сервер с минимальными проблемами.

Рекомендации по использованию бесплатного хостинга

Чтобы в дальнейшем у вас не возникало затруднений и сложностей, четко оговорим область применения бесплатного хостинга. Бесплатный хостинг можно применять для:

Экспериментов. Это как раз то, чем мы с вами будем заниматься в этой главе.

Домашних страниц. Если вы создаете страничку для неформальных целей, чтобы познакомить Интернет-сообщество с собой, своими увлечениями и пристрастиями и т.д., нет необходимости за это платить деньги. Если человек захотел узнать о вас побольше, он, скорее всего, не будет очень расстраиваться и не уйдет рассержено, если ваш сайт будет грузиться не слишком быстро, а на углу примостится рекламный баннер.

Некоммерческих проектов. Если проект не приносит денег, то желательно, чтобы он и тратил их как можно меньше, поэтому бесплатный хостинг в таком случае является хорошим решением. Хотя, если посещаемость вашего ресурса будет достаточно велика, можно вместо рекламы, вставляемой бесплатным хостингом, повесить на сайте рекламный баннер и оплачивать из вырученных денег услуги платного хостинга.

Фан-ресурсов на начальной стадии работы. Допустим, вы создали сайт, посвященный вашему увлечению, вы знаете, что ваше увлечение разделяют и другие люди, и им будет интересно посещать ваш сайт. Но пока о нем еще никто не знает и на уровень окупаемости, например, за счет показа рекламы сайт выйдет нескоро. В этом случае, можно сначала разместить его на сервере бесплатного хостинг-провайдера, а затем, после повышения популярности ресурса, перейти на платный хостинг. Хотя лучшим вариантом будет все-таки и начинать с платного хостинга.

Бесплатный хостинг не следует использовать для:

Солидных коммерческих и официальных сайтов представительств фирм, общественных организаций, государственных учреждений. Доменный адрес, принадлежащий бесплатному хостинг-провайдеру, наверняка подорвет доверие к солидной фирме, которая экономит несколько долларов на своем сайте. Кроме того, хозяева бесплатных хостингов, как правило, не очень хорошо относятся к размещению коммерческих сайтов на их территории.

Крупных проектов. Бесплатный сервер просто захлебнется от наплыва посетителей и одновременного запуска кучи серверных сценариев.

Файловых архивов. Хостинг-провайдеры не жалуют такой способ использования бесплатного Web-пространства и всячески противодействуют ему. Ограничивают максимальный размер файлов, запрещают закачивать файлы с определенными разрешениями и т.д.

Требования. предъявляемые к серверу бесплатного хостинга

Итак, для начала нам нужен бесплатный хостинг. Давайте разберемся, какие основные требования нужно предъявлять к бесплатным хостингам, на что следует обратить внимание.

Скорость работы сервера

Это требование понятно, чем быстрее сервер будет реагировать на запросы пользователя, тем ниже вероятность, что пользователь закроет Web-сайт. не дождавшись загрузки. Скорость работы определяется экспериментальным путем. Либо создайте тестовую страницу на сервере и проверьте ее работу, либо походите по страницам, уже размещенным на этом бесплатном хостинга.

Ограничения на типы и размеры файлов

Многие бесплатные хостинг-провайдеры устанавливают ограничение на размер размещаемых файлов и их тип. Чаще всего, Web-страницы содержат достаточно небольшие HTML файлы и картинки, поэтому в таких ограничениях нет ничего страшного.

Но не исключен вариант, что вам понадобится разместить на своем сайте несколько музыкальных композиций в формате **mp3**, видеоклипы или что-нибудь столь же объемистое. В таком случае при выборе бесплатного хостинга обратите внимание на условия размещения файлов.

Место. выделяемое ПОД сайт

Среднестатистический Web-сайт, особенно недавно начавший работу, занимает довольно мало места, и двадцати мегабайт минимального объема, предлагаемого бесплатными хостинг-провайдерами, вам, скорее всего, хватит с хорошим запасом. К тому же, на многих бесплатных хостингах имеется возможность впоследствии увеличить предоставляемый объем дискового пространства. Поэтому этот параметр не слишком важен, смотрите только, чтобы вам не предоставили объем порядка одного-двух мегабайт, этого места вам скорее всего не хватит.

Возможность работы с CGI и PHP

Если вы хотите разместить на бесплатном хостинге динамический Web-сайт, использующий серверные сценарии, то от хостинга вам потребуется поддержка технологий CGI и языка PHP. К сожалению, далеко не все бесплатные хостинг-провайдеры поддерживают такую возможность.

Чаще всего хостингом предоставляется некий предустановленный набор сценариев, создающих гостевую книгу, чат, набор счетчиков и т.д., без возможности добавить свои сценарии. Если вы размещаете на сервере

домашнюю страницу, то предустановленные сценарии вам еще могут быть полезны, если же вы делаете какой-то более-менее навороченный динамический сайт, то вам, безусловно, понадобится полноценная поддержка серверных сценариев.

Реклама на странице

Бесплатность размещения Web-сайтов хостинг-провайдер окупает, как правило, размещением своей рекламы на их страницах, хотя некоторые хостеры получают прибыль с бесплатных страниц другими способами, например, привлекая новых клиентов на свой платный хостинг или повышая с помощью бесплатных страниц популярность своего доменного имени.

Реклама, размещаемая хостинг-провайдерами, делится на два вида: рекламные баннеры на страницах сайтов и динамические рекламные вставки, pop-up'ы (от англ. pop-up - выскочить), закрывающие часть страницы, а затем автоматически исчезающие с экрана. Выбирайте из двух зол то, что кажется вам меньшим. Баннер можно органично вписать в дизайн вашей страницы, зато динамические вставки, исчезнув, не отвлекают посетителя и не уменьшают рабочий объем страницы.

Адрес сайта

Большинство бесплатных хостингов предлагают своим пользователям доменные имена третьего уровня **http://www.название_страницы.имя_хостинг_провайдера.географическая_доменная_зона,** например: **http://www.vasya.narod.ru** или **http://www.rot.front.ru**. Другим вариантом является представление сайта пользователя, как папки на сервере хостинг-провайдера, например, **http://www.chat.ru/-username**. Такое имя выглядит хуже, создается впечатление, будто ваш сайт не самостоятельный проект, а просто один из разделов сервера, да и запомнить такое название сложнее.

Следует отметить, что адрес сайта, указывающий на размещение его на одном из популярных серверов бесплатных страниц, негативно сказывается на рейтинге страницы. Дело в том, что большинство страниц, размещаемых на серверах бесплатных страниц, например, **narod.ru** или **by.ru,** - это слепленные по одному шаблону, скучные и некрасивые страницы, все содержание которых сводится к «здесь был Вася». Чтобы доменное Имя сервера не работало против вас, выберите хостинг-провайдера с не таким раскрученным именем. Но здесь вас поджидают другие подводные камни. Если доменное имя не популярно то, возможно, для этого есть объективные причины. Например, хостинг не очень хорош сам по себе или недавно открыт и еще не пережил «болезни роста», негативно сказывающиеся на стабильности и удобстве работы.

Некоторые бесплатные хостинг-провайдеры предлагают услугу по размещению сайта пользователя под произвольным доменом второго уровня. Но для этого этот домен нужно приобрести, заплатив некоторую сумму денег. Преимущество такого решения заключается в том, что даже перейдя в последствии на платный хостинг, вы сохраните свое доменное имя, а недостаток – за него нужно платить деньги. А, учитывая тот факт, что

заказывая платный хостинг у некоторых компаний, домен второго уровня вы получаете бесплатно, в таком случае имеет смысл подумать о платном хостинге. Если вы уже готовы платить за свой сайт, то почему бы не заплатить и за хостинг, избавившись от всех недостатков бесплатного сервера.

Способ обновления сайта

Некоторые бесплатные хостинг-провайдеры предлагают единственный способ обновления файлов на сайте - через специальную Web-форму, с помощью браузера. Безусловно, таким способом можно добавить 1-2 файла на сайт, но если вам придется таким образом обновлять несколько сотен файлов, вы недобрым словом помянете разработчиков этого инструмента. Единственно правильный способ работы с файлами на сервере - по протоколу FTP (File Transfer Protocol - Протокол передачи файлов), когда вы можете обращаться с файлами сайта, как с файлами в папке на вашем жестком диске. Т.е. просто и функционально. Обязательно обращайте внимание на этот параметр при выборе хостинга, благо возможность FTP доступа предоставляет большинство хостинг-провайдеров.

Сравнение различных серверов бесплатного хостинга

Найти сервера бесплатного хостинга можно просто введя в поисковую систему Яndex запрос «бесплатный хостинг». Сейчас же, мы перечислим несколько популярных бесплатных хостинг-провайдеров и отметим их основные особенности, см Табл. 8.1.

Табл. 8.1. Сравнительная характеристика бесплатных хостинг-провайдеров

Хостер	Дисковое пространство	Адрес сайта	Серверные сценарии	Способ обновления	Тип рекламы
www.ziyonet.uz	500 МБ	www.название_сайта.zn.uz	Поддержка CGI и PHP	Web-форма, FTP	Нет рекламы
www.narod.ru	Неогранич.	www.название_сайта. narod.ru	Только набор предустановленных сценариев	Web-форма, FTP	Всплывающее окно
by.ru	Неогранич.	www.название_сайта.by.ru поддержка произвольных доменов второго и третьего уровня	Поддержка CGI и PHP	Web-форма, FTP	Баннер
www.webser	С.	название_сайта.al.r	Поддерж	Web-	Баннер

Хостер	Дисковое пространство	Адрес сайта	Серверные сценарии	Способ обновления	Тип рекламы
www.ziyonet.uz	500 МБ	www.название_сайта.zn.uz	Поддержка CGI и PHP	Web-форма, FTP	Нет рекламы
vis.ru		u назв._сайта.bip.ru назв._сайта.dem.ru назв._сайта.fud.ru назв._сайта.hobl.ru	ка CGI и PHP, набор предустановленных сценариев	форма, FTP	
www.pochta.ru	20Mb	www.назв._сайта.pochta.ru возможны варианты fromru.com front.ru hotbox.ru krovatka.net land.ru mail15.com mail333.com pisem.net pochtamt.ru pop3.ru rbcmail.ru smtp.ru	нет	Web-форма, FTP	Нет рекламы
www.holm.ru	Неограни ч.	www.назв._сайта.h15.ru	Поддержка CGI, PHP	FTP	Баннер
www.hul.ru	100Мб, с возможностью дальнейшего увеличения	www.назв._сайта.hut.ru возможна поддержка произвольных доменов второго уровня	Поддержка CGI-сценариев на языке PERL, PHP	FTP	В верху страницы
www.mail.ru	50Мб, возможно увеличение	www.назв._сайта.mail.ru	сценарии не поддерживаются	Web-форма, FTP	Баннер в верху всех

Хостер	Дисковое пространство	Адрес сайта	Серверные сценарии	Способ обновления	Тип рекламы
www.ziyonet.uz	500 МБ	www.название_сайта.zn.uz	Поддержка CGI и PHP	Web-форма, FTP	Нет рекламы
					страниц
www.chat.ru	10 Мб	www.назв._сайта.chat.ru либо www. chat.ru/~название_сайта	Только предуста-новленные сценарии	Web-форма, FTP	Баннер

Регистрация на сервере бесплатного хостинга

Допустим, вы подобрали себе бесплатный хостинг по вкусу, теперь необходимо на нем зарегистрироваться. Процедура регистрации зависит от сервера хостинга, но в общем и целом они похожи. Рассмотрим эту процедуру на примере сервера http://www.ziyonet.uz.

- В адресной строке браузера введите адрес сайта хостинг-провайдера, http://www.ziyonet.uz. после чего откроется его главная страница (Рис. 8.1). На ней вы найдете упоминания о последних новостях, предоставляемых услугах и ссылку на страницу регистрации новых пользователей хостинга
- Щелкните мышью на ссылке Регистрация в правой части страницы. Откроется страница с регистрационной формой (Рис. 8.2).

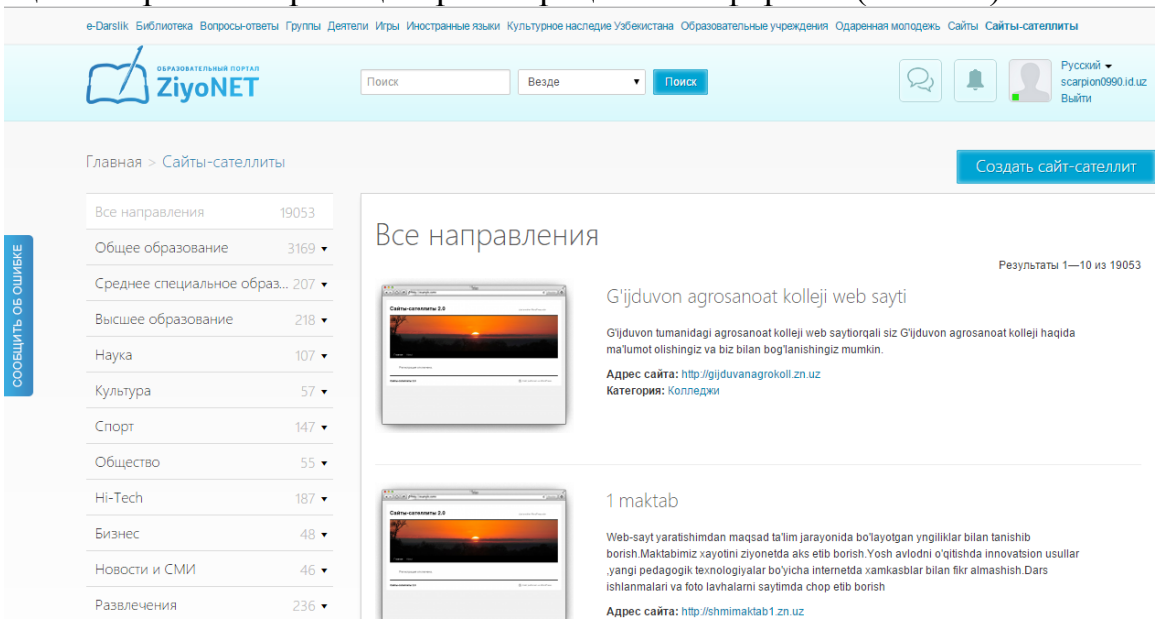


Рис. 8.2

- Щелкните мышью на ссылке **СОГЛАШЕНИЕ** о предоставлении услуг по бесплатному хостингу. Откроется страница, на которой будут приведены основные условия предоставления хостинга, ограничения на

размещение информации и файлов и правила поведения, которым вы должны следовать, пользуясь услугами этого хостинг-провайдера (Рис. 8.3).

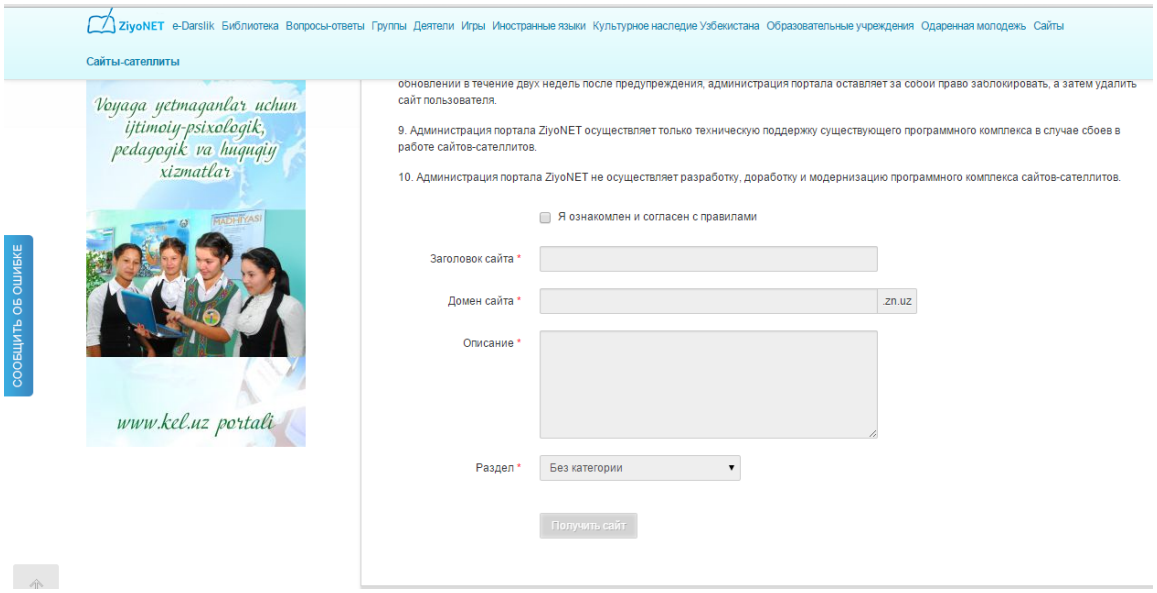


Рис. 8.3

- Если вам подходят условия соглашения, закрывайте эту страницу и приступайте к заполнению регистрационной формы, если же какие-то пункты вам категорически не подходят, значит, этот хостинг не для вас, выберите другого хостинг-провайдера.
- В регистрационной форме звездочками помечаются поля, обязательные к заполнению. Введите в соответствующие поля ввода свое имя, желаемое название домена третьего уровня. Выберите из выпадающего списка желаемый домен второго уровня. Напишите название сайта и краткое его описание. Эти данные будут использованы при размещении сайта в каталоге хостинг-провайдера. Наконец, укажите свой адрес электронной почты и пароль доступа.
- Щелкните мышью на кнопке **Зарегистрировать**. Если все данные введены правильно и выбранное вами доменное имя сайта еще не занято. вы будете зарегистрированы (Рис. 8.4). Если же в форме были ошибки либо выбранное доменное имя уже кем-то занято, вас попросят изменить часть введенных сведений.

Через некоторое время после выполнения процедуры регистрации, на ящик электронной почты, указанный в регистрационной форме, придет письмо, информирующее вас об окончании процесса создания вашей учетной записи и содержащее дальнейшие указания по работе с нашей учетной записью виртуального хостинга

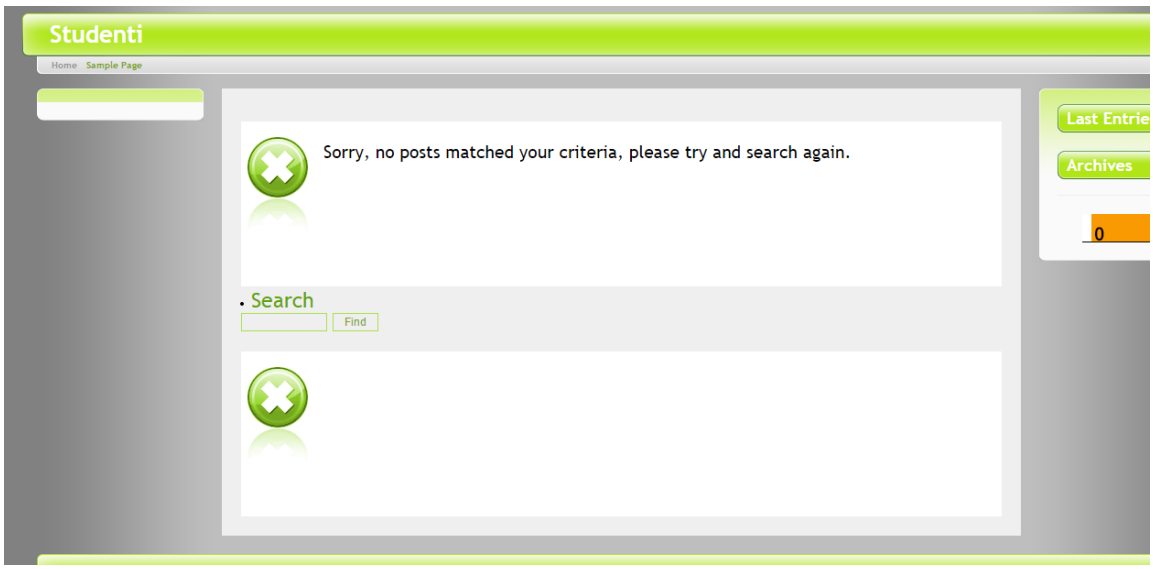


Рис. 8.4

Введите в адресной строке браузера адрес вашего сайта, приведенный в регистрационном письме. Вы увидите страницу, сообщающую о том, что сайт находится в разработке (Рис. 8.5). Эту страницу хостинг автоматически разместил на вашем виртуальном сервере, как главную страницу сайта, и она будет видна всем посетителям вашего сайта, пока вы не замените ее какой-либо другой страницей. О том, каким образом закачать свои собственные файлы на сервер, мы поговорим в следующем разделе.

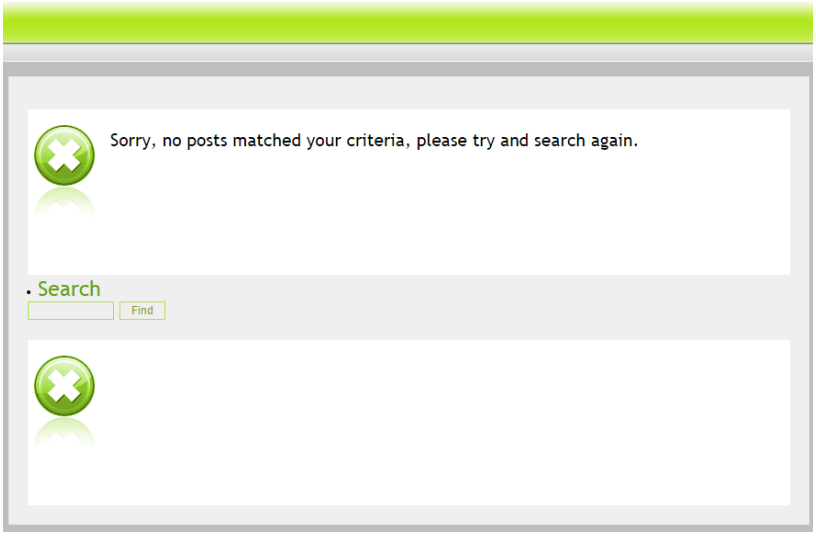


Рис. 8.5

Выгрузка сайта на сервер и его обновление

Итак, вы создали учетную запись на сервере бесплатного хостинга и теперь вам нужно выложить на него свой сайт. Для выполнения этой задачи существует несколько способов.

Использование Web-интерфейса

Первый и самый простой способ – воспользоваться специальным Web-интерфейсом. Такую услугу предлагает большинство хостинг-провайдеров. У разных хостингов Web-интерфейсы тоже разные, но в общих чертах они достаточно похожи. Рассмотрим Web-интерфейс управления файлами,

предлагаемый хостингом http://www.ziyonet.uz Получить доступ к Web-интерфейсу можно через страничку управления сайтом,

Введите в адресной строке браузера адрес главной страницы сайта хостинга, http://www.ziyonet.uz. в правой части открывшейся страницы будут два поля ввода для ID и пароля зарегистрировавшихся пользователей.

Введите в эти поля ID и пароль, полученные вами в регистрационном письме, и щелкните мышью на кнопке Перейти. Откроется страница с информацией для зарегистрированных пользователей (Рис. 8.6).

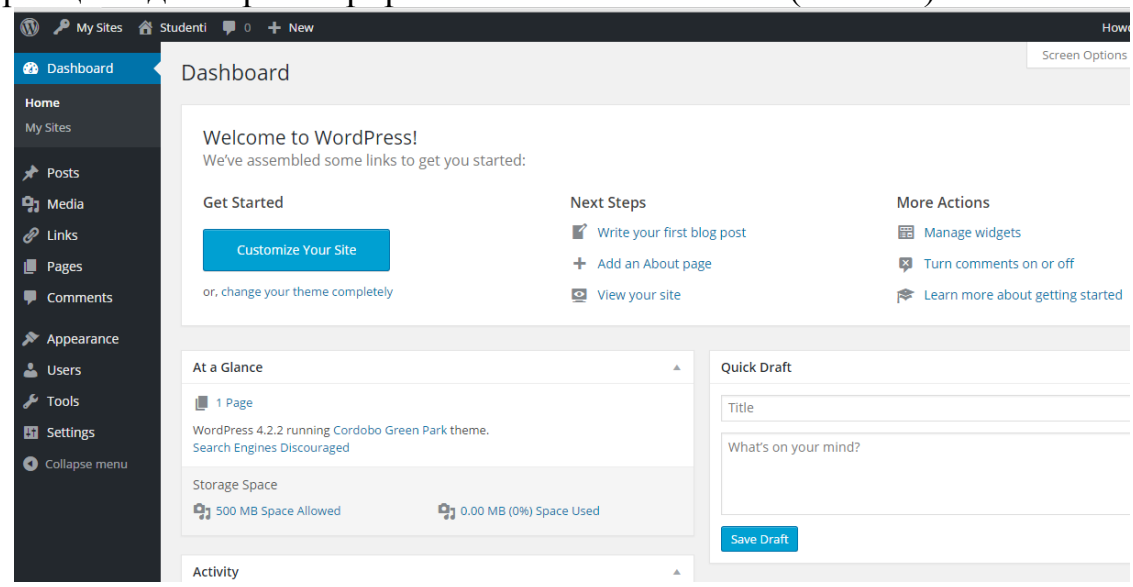


Рис. 8.6

В списке бесплатных услуг найдите ссылку Файл менеджер и щелкните на ней мышью, откроется страница Web-интерфейса управления файлами (Рис. 8.7).

Большую часть страницы занимает список папок и каталогов.

Если вы щелкнете мышью на названии папки, в окне откроется ее содержание. Чтобы вернуться к содержимому вышестоящей палки, щелкните мышью на двоеточии (..) в верхней части списка.

Чтобы переименовать файл либо папку, щелкните мышью на ссылке rename, справа от имени файла или папки. Появится окно переименования (Рис. 8.8).

Введите новое название и щелкните мышью на кнопке **переименовать.**

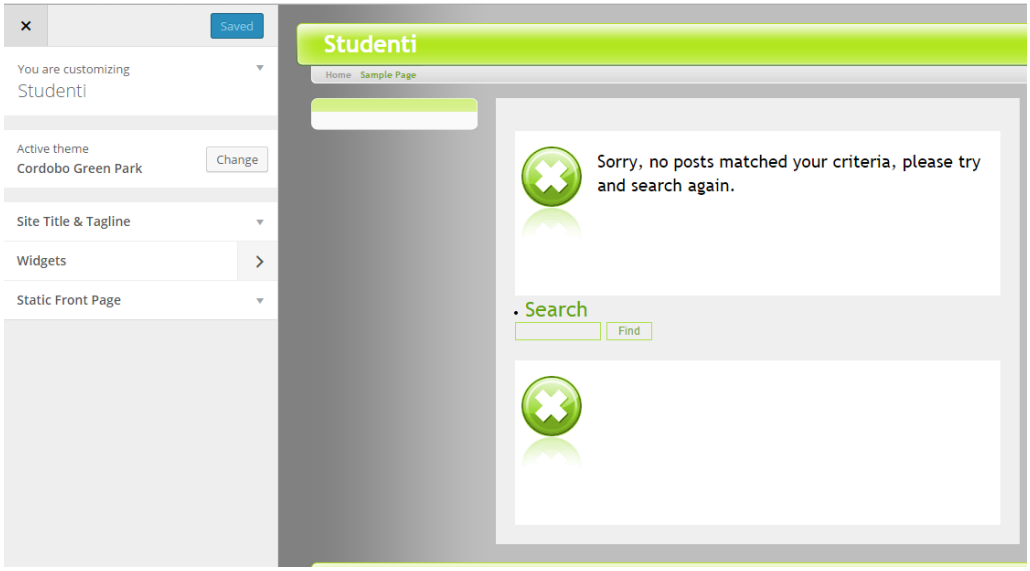


Рис. 8.7

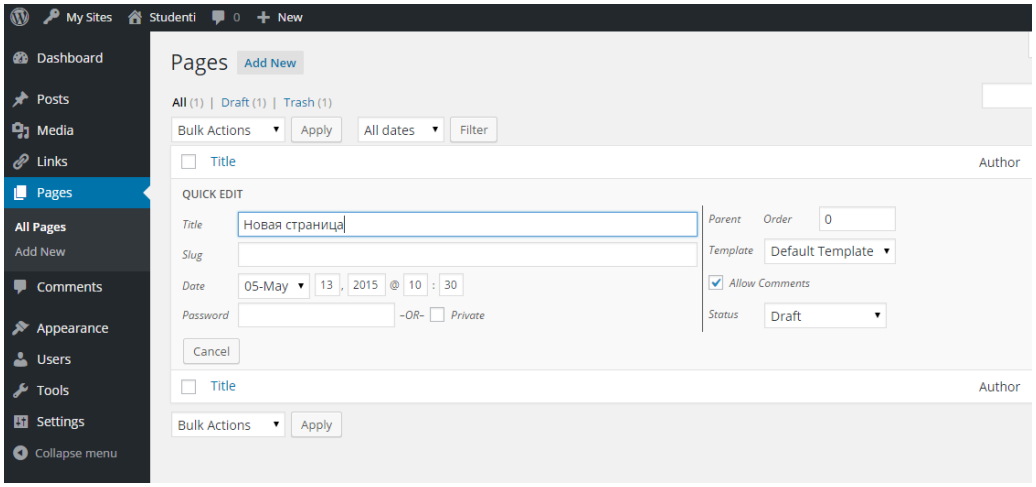


Рис. 8.8

- Если вы щелкнете на ссылке **edit** справа от названия файла, откроется страница редактирования файла, позволяющая немного подправить код той или иной страницы либо сценария (Рис. 8.9), Закончив редактирование, щелкните мышью на ссылке **Сохранить.**
- Чтобы удалить файлы или папки с сервера, установите флажки в столбце **Del** (Удалить) напротив имен удаляемых файлов и папок, а затем щелкните мышью на кнопке удалить отмеченные.
- Чтобы создать новую папку, введите название папки в поле ввода создать директорию, а затем щелкните мышью на кнопке создать справа от этого поля.
- Чтобы создать новый файл, введите его название в поле ввода создать файл, а затем щелкните мышью на кнопке создать, справа от этого поля

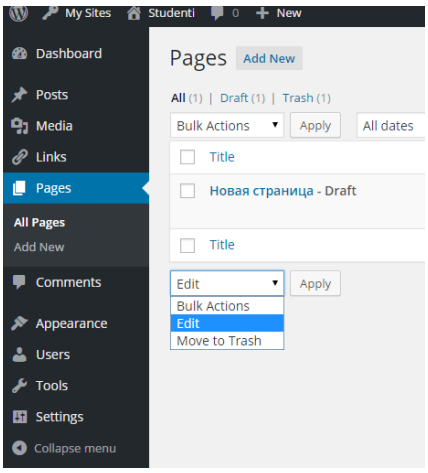


Рис. 8.9

• Чтобы выгрузить на Web-сервер файл из папки на вашем компьютере, в Web-форме предназначен раздел под названием загрузить файлы. Этот раздел состоит из четырех полей ввода файл #N с кнопками обзор справа от них и кнопкой загрузить. Щелкнув мышью на кнопке обзор, вы откроете диалог выбора файлов (Рис. 8.10). Выберите файл, который бы хотите отправить на сервер и щелкните мышью на кнопке открыть, путь к файлу добавится в поле ввода. Теперь, чтобы отправить файл на сервер, щелкните мышью на кнопке загрузить, файл будет выгружен на сервер. Если вам нужно выгрузить сразу несколько файлов, то перед нажатием кнопки загрузить заполните несколько полей файл #N и повторяйте процедуру до выгрузки всех нужных файлов. Причем учтите, что файлы загружаются в текущий каталог, чтобы загрузить файлы в другой каталог вашего виртуального сервера, предварительно перейдите в него, щелкнув мышью на его названии.

Как вы уже поняли, Web-интерфейс управления файлами лучше всего подходит для того, чтобы закачать на сервер 1-2 файла, немного подправить код страниц. Или произвести еще какую-нибудь косметическую правку. Эта система является незаменимой. если вам нужно произвести некоторые изменения на сайте, находясь вдалеке от дома. С помощью Web-интерфейса вы можете управлять вашим сайтом из гостей или вообще из любого Интернет-кафе.

Использование протокола FTP

Если же вам необходимо закачать на сервер несколько сотен файлов, лежащих к тому же в разных папках, работа с Web-интерфейсом превращается в сущий ад. В таком случае незаменимым является доступ к серверу через протокол FTP. Для работы с сервером по протоколу FTP нужна специальная программа FTP-клиент. FTP-клиентов существует великое множество, на любой вкус и любые потребности. Мы рассмотрим работу FTP-клиента на примере лишь одного из них - FilaZilla. Эта программа была выбрана по очень простым причинам: она бесплатна, имеет русифицированный интерфейс и обладает всей» необходимыми нам функциями.

Скачивание и установка FTP-клиента FileZilla

Сначала программу FileZilla необходимо скачать.

- Введите в адресной строке браузера адрес http://sourceforge.net/projects/filezilla/ Откроется главная страница сайта программы **FileZilla** (Рис. 8.11).
- Чтобы перейти к странице скачивания программы, щелкните мышью на пункте **Download** меню страницы. Откроется страница скачивания программы FileZilla (Рис. 8.12). Под заголовком FileZilla размещается список с файлами различных версий этой программы, последняя версия размещается в самом верху.
- Щелкните мышью на ссылке **Download FileZilia_X_X_XXX_setup.exe**, чтобы перейти на страницу выбора сервера, с которого будете скачивать файл, X_X_XXX – это номер версии программы (на момент написания этой книги, номер последней версии был 3_10_2_win32). На странице выбора сервера будет представлен список серверов, с которых можно скачать программу. с указанием их месторасположения (Рис. 8.13).
- щелкните мышью на иконке столбца Download той строки списка, в которой указан сервер, наиболее близко к вам расположенный. Например, можно выбрать сервер eitkit.ru, находящийся в Москве. После щелчка на иконке откроется страница, сообщающая о том, какой сервер вы выбрали, и начнется процесс скачивания.

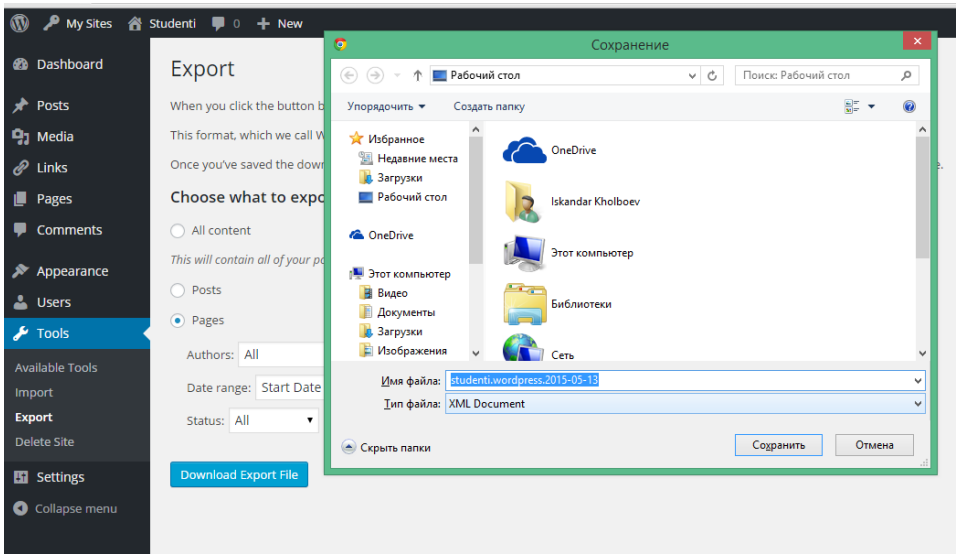


Рис. 8.10

Рис. 8.11

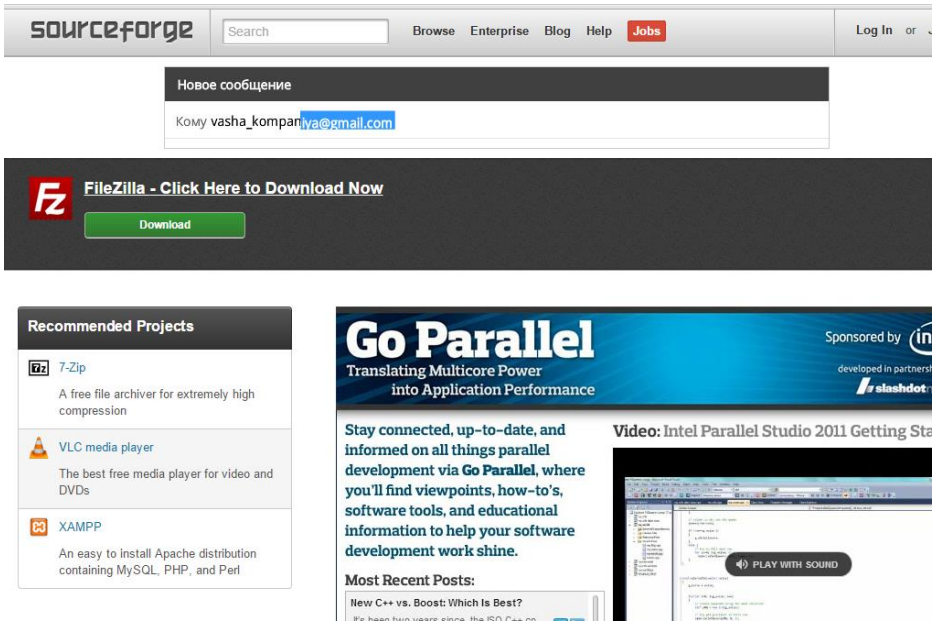


Рис. 8.12

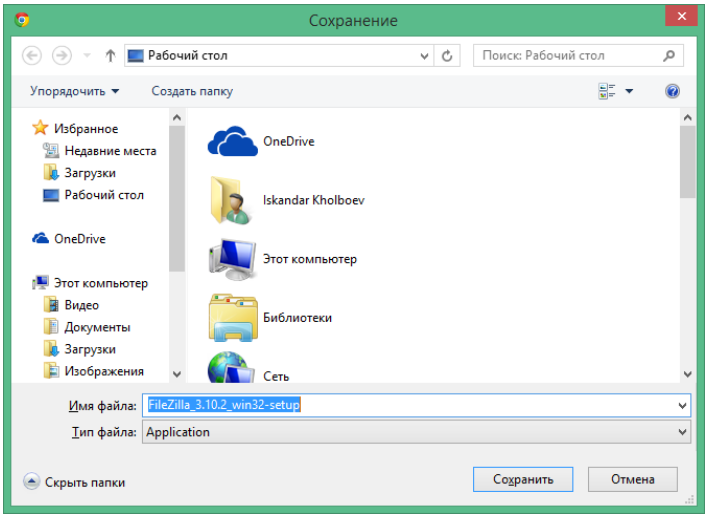


Рис. 8.13

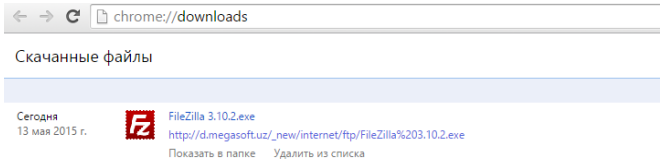


Рис. 8.14

После того как программа полностью загружена, необходимо ее установить на компьютер.

- Запустите файл **FileZilla_X_X_XXX_setup.exe**, чтобы начать процесс установки программы. Откроется: диалог Installer Language (Язык инсталляции), предлагающий выбрать язык программы и системы установки (Рис. 8.15),

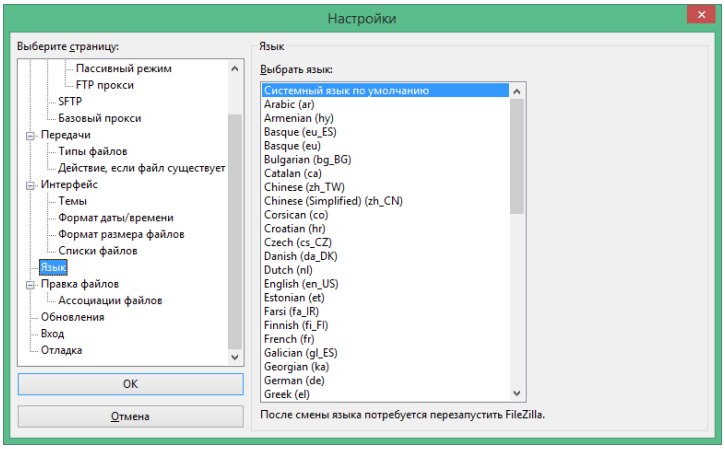


Рис. 8.15

- Выберите пункт Russian (Русский) из раскрывающегося списка, чтобы программа общалась по-русски.
- Щелкните мышью на кнопке OK, чтобы продолжить процесс установки. Откроется диалог с лицензионным соглашением программы **FileZilla** (Рис. 8.16)

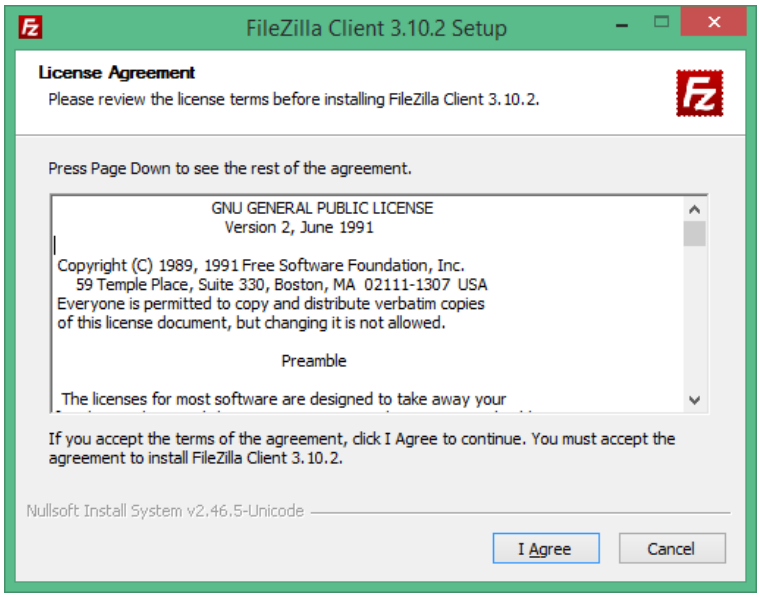


Рис. 8.16

- Щелкните мышью на кнопке **Согласен** (Accept), чтобы принять условия лицензионного соглашения. Появится диалог выбора Компонентов, которые необходимо устанавливать (Рис. 8.17).

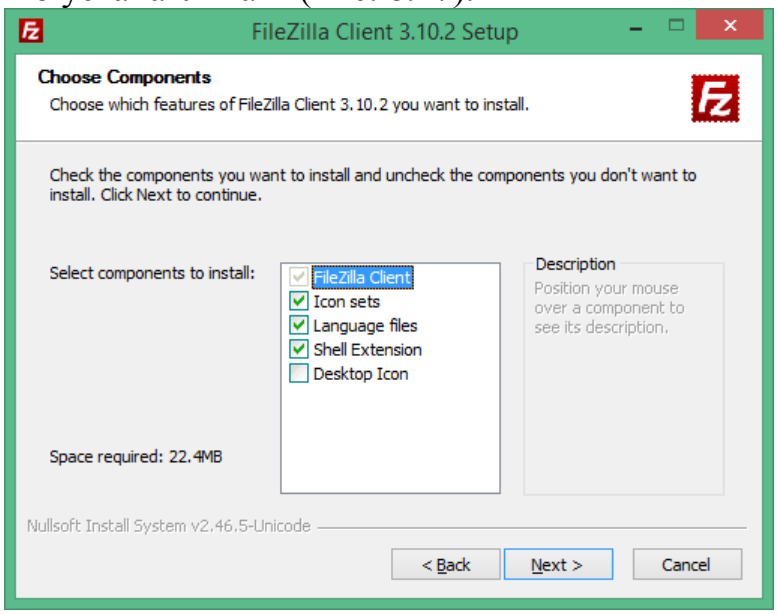


Рис. 8.17

- Выберите пункт **Стандартная** (Standard) из открывающегося списка Выберите тип установки (Choose installation type), чтобы выбрать тип установки по умолчанию.
- Щелкните мышью на кнопке **Далее** (Next), чтобы продолжить процесс установки. Откроется диалог выбора папки, в которую будет установлена программа **FileZilla** (Рис. 8.18).

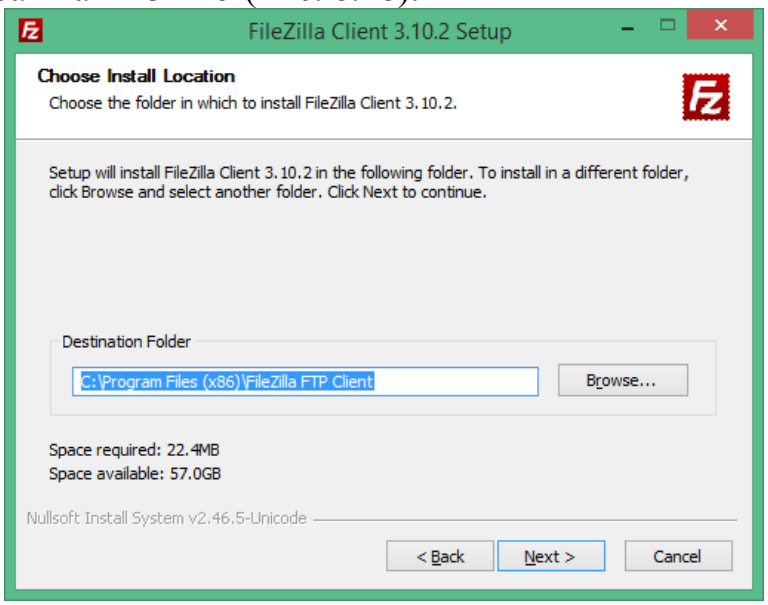


Рис. 8.18

- Щелкните мышью на кнопке **Далее** (Next), чтобы подтвердить установку программы в папку по умолчанию. Откроется диалог выбора папки в меню **Пуск** (Start), куда будут установлены иконки программы (Рис. 8.19).

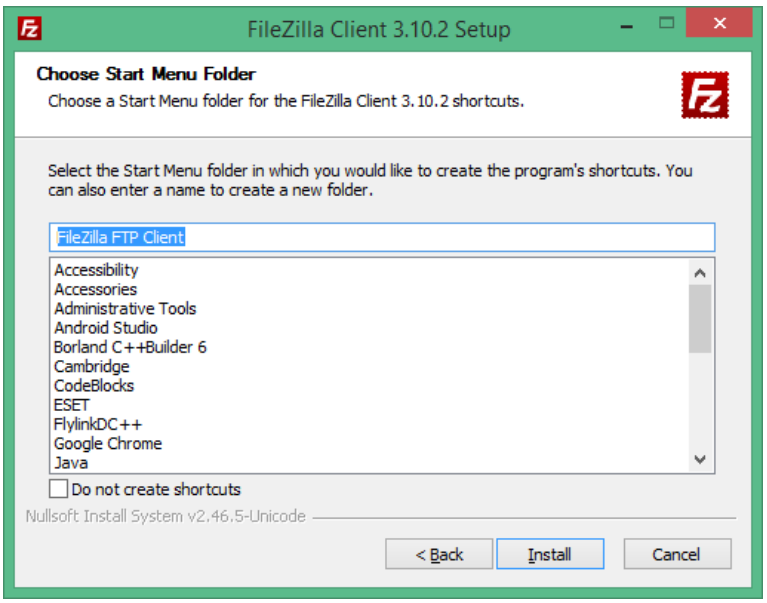


Рис. 8.19

- Щелкните мышью на кнопке **Далее** (Next), чтобы подтвердить установку иконок в папку меню **Пуск** (Start) по умолчанию. Откроется диалог с общими настройками программы (Рис.8.20).

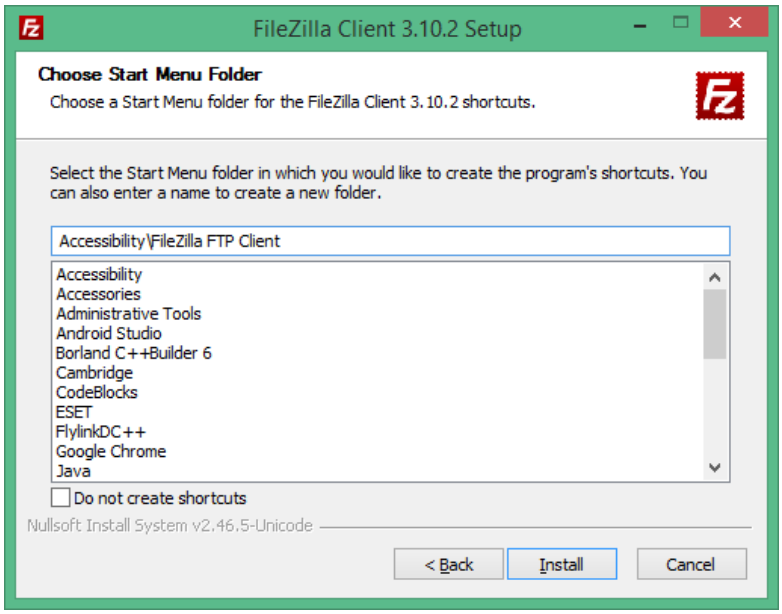


Рис. 8.20

- Щелкните мышью на кнопке **Установить** (Install), чтобы приступить к копированию файлов программы из установочного архива. Появится диалог копирования файлов, показывающий процесс установки файлов программы на компьютер (Рис. 8.21). Когда копирование файлов будет завершено, отобразится диалог завершения установки (Рис.8.22).

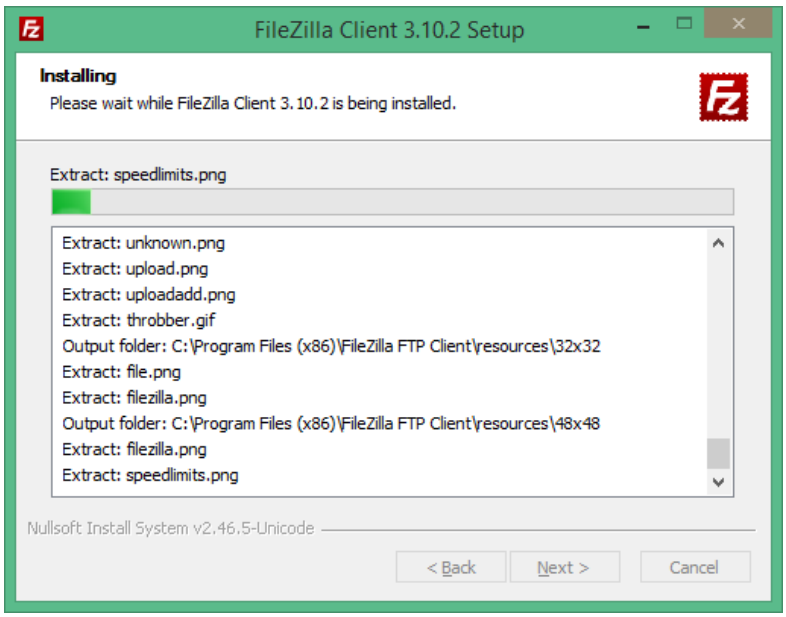


Рис. 8.21

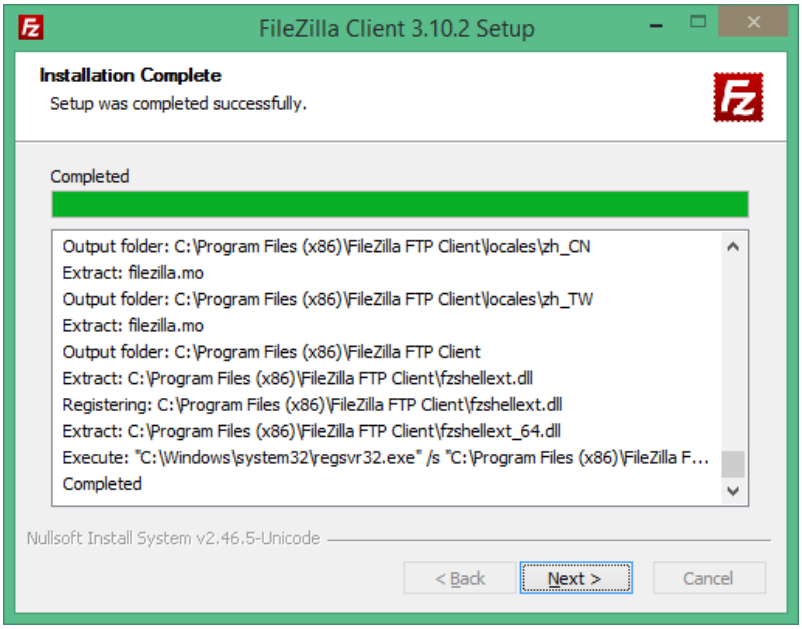


Рис. 8.22

- Щелкните мышью на кнопке **Следующий**(Next), чтобы завершить процесс установки и закрыть диалог.

Работа с FTP-клиентом FileZilla

Чтобы запустить программу FileZilla, выполните команду меню **Пуск** (Start) **Все программы → FileZilla → FileZilla (Programs → FileZilla → FileZilla)**

Главное окно программы **FileZilla** разделено на четыре части (Рис. 8.23). Внизу – область обработки задании, в ней отображается процесс передачи файлов с компьютера на удаленный сервер и наоборот. В верхней части окна, находится область сообщении, в ней отображаются команды, посылаемые серверу, и отклик на них.

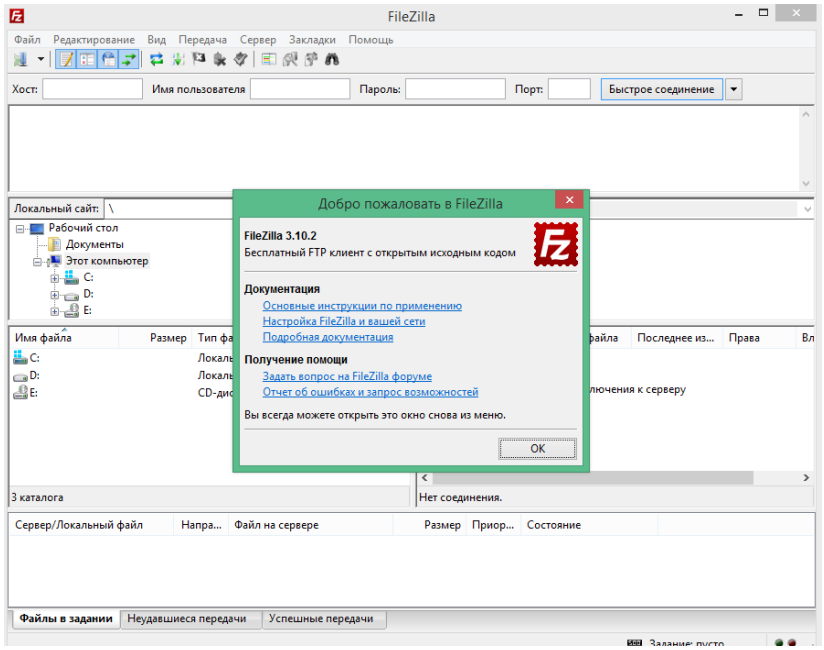


Рис.8.23

Середина окна делится на две части. В левой отображаются каталоги дисков вашего компьютера, а в правой, каталоги сервера. Чтобы передать файл, группу файлов – или каталог на сервер, достаточно просто перетащить их из левой части окна в правую. Точно так же, совершается и перенос файлов с сервера на компьютер, только переносить файлы нужно уже справа налево.

Настройка параметров подключения к серверу

Но чтобы получить доступ к файлам сервера, нужно к нему подсоединиться, для этого необходимо задать настройки сервера и менеджера сайтов программы FileSlla.

- Выполните команду меню **Файл → Менеджер сайтов** (File → Site manager). Откроется диалог Менеджер сайтов (Site manager), Рис. 8.24. В этом диалоге можно выбрать сервер, к которому вы будете подключаться, и добавить новый сервер к списку уже настроенных.
- Щелкните мышью на кнопке Новый (New). К списку Дерево FTP сайтов (рис. 8.25.)

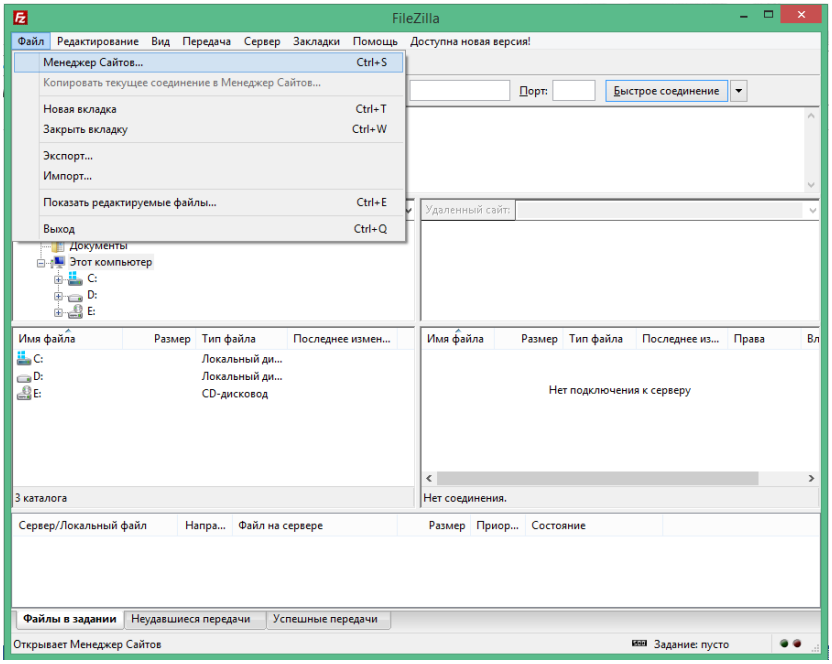


Рис. 8.24

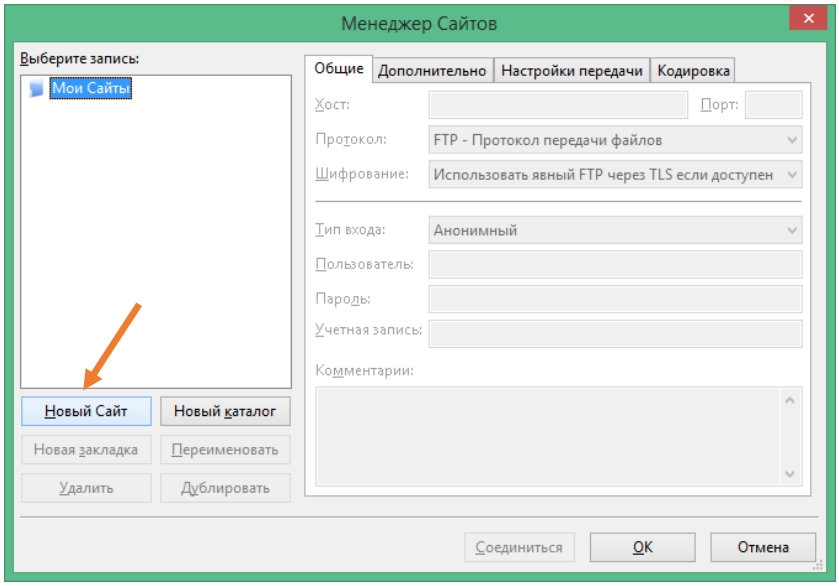


Рис. 8.25

Ключевые слова: *хостинг, виртуальный хостинг, хостинг-провайдер, выделенный сервер, совместное размещение.*

Контрольные вопросы:

1. Что такое виртуальный хостинг?
2. Недостатки выделенного сервера
3. Недостатки совместное размещение
4. Какие бесплатные хостинги вы знаете и какие возможности они предоставляют.
5. Способы обновления сайта?

ЛИТЕРАТУРА

Настоящее пособие предназначено в первую очередь для студентов ТУИТ, изучающих основные предметы информационно-коммуникационных технологий. Особенностью данного пособия является мультимедийное сопровождение практических занятий, которое будет играть роль электронного наставника, показывающего как научиться выполнять тот или иной прием процесса веб-разработки. В виду такой особенности, изучение основ веб-разработки лучше начать с самого начала, однако, студенты имеющие знания и навыки могут пропустить главы, изучение которых, по их мнению, будет напрасной тратой времени.