Лекции по современным веб-технологиям

2-е издание, исправленное

Кузнецова Л.В.

Национальный Открытый Университет "ИНТУИТ"
2016

Лекции по современным веб-технологиям

2-е издание, исправленное

Кузнецова Л.В.

Национальный Открытый Университет "ИНТУИТ"
2016

Лекции по современным веб-технологиям/ Л.В. Кузнецова - М.: Национальный Открытый Университет "ИНТУИТ", 2016

В курсе излагаются история, стандарты и базовые принципы организации Всемирной паутины. Теоретически и практически рассмотрены основные возможности языка гипертекстовой разметки HTML и каскадных таблиц стилей CSS. Описаны основные ограничения полиграфии и цветового оформления веб-сайта.

Данный курс представляет собой пособие по теоретическому и практическому освоению технологий создания веб-страниц – языка разметки гипертекста HTML и каскадных таблиц стилей CSS. В лекциях приводится история развития Интернета и создания Всемирной паутины. Рассказывается о том, что такое современный Интернет, как он работает, рассматриваются основные технологии, которые обеспечивают деятельность Всемирной паутины. Особое внимание уделено стандартам HTML и CSS и способам проверки согласованности с данными стандартами. Слушатели узнают об основных возможностях языка разметки гипертекста и каскадных таблиц стилей. В лекциях приведены возможности и средства по оформлению текста и изменению его вида. Описаны основные графические форматы, которые используются на сайтах. Освещены вопросы работы с рисунками, ссылками, списками и многое другое. Раскрыты возможности использования таблиц и управления их видом. Приведены способы выравнивания рисунков, слоев и текста, использования отступов и полей и другие приемы оформления веб-страниц. Подробные примеры и многочисленные скриншоты позволяют легко воспроизвести приведенные технологии на практике и делают представленный материал более наглядным. Множество советов, посвященных разным аспектам создания веб-страниц, помогут закрепить уже имеющиеся навыки и знания или сделать первые шаги в этом направлении. Также освещены вопросы проектирования и дизайна сайтов: рассказывается о начальных этапах планирования и общих принципах создания веб-сайта и об основных элементах сайта. Кроме того, описаны основные ограничения полиграфии и цветового оформления веб-сайта и даны некоторые рекомендации по выбору шрифтов, длины и высоты строк, цветовых схем и др.

Что такое современный Интернет?

В лекции приводятся некоторые сведения о сети Интернет и Всемирной паутине.

В 1962 г. Дж. Ликлайдером, руководителем исследовательского компьютерного проекта экспериментальной сети передачи пакетов в Управлении перспективных исследований и разработок Министерства обороны США (Defense Advanced Research Project Agency, DARPA), была опубликована серия заметок, в которых обсуждалась концепция "Галактической сети" ("Galactic Network"). "Галактическая сеть" представлялась как глобальная сеть взаимосвязанных компьютеров, позволяющая любому пользователю получить доступ к данным и программам на компьютерах, объединенных данной сетью. Можно сказать, что эта идея положила начало развитию сети Интернет.

Уже через несколько лет специалисты DARPA начали работу над крупной децентрализованной компьютерной сетью ARPANet (Advanced Research Project Agency Network), днем рождения которой считается 29 октября 1969 г., когда была предпринята первая удачная попытка удаленного соединения между двумя компьютерами, находившимися в исследовательском центре Стэнфордского университета и Калифорнийском университете в Лос-Анджелесе. Эти компьютеры и стали первыми узлами будущей сети ARPANet.

С момента появления ARPANet по сегодняшний день Интернет прошел долгий путь, основные вехи которого вкратце представлены ниже. Более подробную информацию об истории возникновения и развития Интернета можно почерпнуть из многочисленных публикаций в Сети.

1970-е годы	Разработана первая программа для отправки электронной почты по сети, появились первые списки почтовой рассылки, новостные группы и доски объявлений. К сети подключились первые международные сетевые узлы, расположенные в Великобритании и Норвегии, ARPANet вышла на международный уровень. Начали развиваться протоколы передачи данных TCP/IP.
1980-	Стандартизированы протоколы передачи данных TCP/IP. Сеть

е годы	ARPANet перешла с протокола NCP на TCP/IP. Разработана система доменных имен, или DNS. Создана магистраль NSFNet. Термин "Интернет" закрепился за сетью ARPANet.
1990-е годы	Сеть ARPANet прекратила свое существование, уступив NSFNet. Всемирная паутина стала доступна в Интернете. Разработаны протокол HTTP, язык HTML и идентификаторы URI. Создан первый графический браузер Mosaic. Образован Консорциум всемирной паутины (W3C). Всемирная паутина полностью подменила собой понятие "Интернет". Число зарегистрированных доменных имен превысило 2 млн.

Современный Интернет (Internet, Interconnected Networks - соединенные сети) представляет собой "сеть сетей", узлами которой являются не отдельные компьютеры, а целые компьютерные сети, каждая из которых управляется независимыми операторами. Она не имеет центра управления, однако работает по единым правилам и предоставляет пользователям единые услуги.

В качестве наиболее общего определения термина "Интернет" можно привести следующее определение, взятое из книги "Doctor Bob’s Guide to Offline Internet Access" ("Доступ к Интернет через электронную почту", 1995 г.), в переводе Вадима Федорова: "Internet (сущ.) - бурно разросшаяся совокупность компьютерных сетей, опутывающих земной шар, связывающих правительственные, военные, образовательные и коммерческие институты, а также отдельных граждан, с широким выбором компьютерных услуг, ресурсов, информации. Комплекс сетевых соглашений и общедоступных инструментов Сети разработан с целью создания одной большой сети, в которой компьютеры, соединенные воедино, взаимодействуют, имея множество различных программных и аппаратных платформ".

Основные протоколы сети Интернет

Основными протоколами сети Интернет являются протоколы стека TCP/IP. Термин TCP/IP характеризует все, что связано с протоколами взаимодействия между компьютерами в сети Интернет. Протокол TCP/IP получил свое название от названия двух коммуникационных

протоколов:

- Transmission Control Protocol - TCP (протокол контроля передачи данных)
- Internet Protocol - IP (протокол передачи данных между сетями Интернет).

Протокол IP отвечает за адресацию в сети и доставку пакетов данных между компьютерами без установления соединения и гарантий доставки. Каждому компьютеру в сети присваивается уникальный IP-адрес, который представляется как четыре десятичных числа (октеты), разделенных точками. Значение любого октета может изменяться от 0 до 255, например, 149.76.12.4. В IP-адресе выделяют две части: сетевую часть (адрес локальной сети) и адрес компьютера в этой локальной сети. Сетевая часть адреса может иметь переменную длину, которая зависит от класса IP-адреса и некоторых других параметров. Выделяют несколько классов IP-адресов.

Класс A	Сети с адресами от 1.0.0.0 до 127.0.0.0. Сетевой номер содержится в первом октете (1-127), что предусматривает 126 сетей по 1.6 миллионов компьютеров в каждой сети каласса A.
Класс B	Сети с адресами от 128.0.0.0 до 191.255.0.0. Сетевой номер находится в первых двух октетах (128.0 – 191.255), что предусматривает 16320 сетей с 65024 компьютерами в каждой.
Класс C	Сети с адресами от 192.0.0.0 до 223.255.255.0. Сетевой номер содержится в первых трех октетах (192.0.0 - 223.255.255). Это предполагает почти 2 миллиона сетей по 254 компьютеров в каждой.
Классы D	Сети с адресами от 224.0.0.0 до 239.255.255.0. Адреса являются групповыми (multicast). Адреса зарезервированы для организации теле- и радиовещания на группы компьютеров.
Классы E и F	Сети с адресами от 240.0.0.0 до 254.0.0.0. Являются экспериментальными и не определяют какую-либо сеть.

IP-адреса могут назначаться вручную или динамически. Для динамической настройки сети используется специальный протокол DHCP (Dynamic Host Configuration Protocol). С его помощью можно

настраивать компьютер пользователя несколькими способами. При ручном способе настройки администратор должен настроить соответствие IP-адресов физическим адресам. При использовании статического способа администратор указывает DHCP-серверу диапазон допустимых IP-адресов. При первом соединении клиент получает адрес из этого диапазона, а сервер устанавливает соответствие выданному IP-адресу физического адреса устройства-клиента. В случае динамического способа настройки IP-адрес выдается из допустимого диапазона, но на определенное время. В этом случае можно построить сеть, в которой количество клиентов значительно превышает количество допустимых IP-адресов.

Протокол TCP позволяет устанавливать виртуальный канал передачи данных между компьютерами. В функции данного протокола входит деление данных на пакеты, подтверждение факта получения пакетов принимающей стороной и повторная передача пакетов в случае необходимости. Кроме того, в протоколе TCP реализованы сложные механизмы регулирования загрузки сети и устранения заторов.

Система доменных имен DNS

Несмотря на то, что адресация в рамках сетей TCP/IP происходит строго по IP-адресам, для пользователя более удобно использование символьных или доменных имен.

Доменное имя – это символьный адрес, имеющий строгую иерархическую структуру, например, ссылка: www.somewhere.com - http://www.somewhere.com. В доменном адресе справа указывается домен верхнего уровня, состоящий из двух, трех или четырех букв. Двухбуквенный домен указывает на географическое расположение ресурса, например, ru - Россия, us - США и т.д. Трех- и четырехбуквенные домены используются для обозначения принадлежности организации к различным видам. Например, com - коммерческая организация, edu - образовательное учреждение и т.д.

В сетях TCP/IP соответствие между доменными именами и IP-адресами определяется централизованной службой DNS (Domain Name Service), использующей распределенную базу отображений "доменное имя – IP-

использующей распределенную базу отображений "доменное имя – IP-адрес". Под распределенностью базы подразумевается то, что DNS-серверы распределены по всему миру, на каждом из которых находится какая-то часть от этой базы.

Алгоритм работы DNS можно описать следующим образом. Пользователь в окне браузера вводит доменное имя определенного ресурса. Компьютер пользователя отправляет запрос об установлении IP-адреса по введенному доменному имени на первый DNS-сервер, IP-адрес которого обычно устанавливается провайдером. Если в базе данных сервера имеется соответствующая запись "доменное имя – IP-адрес", то IP-адрес возвращается компьютеру пользователя. Если же в базе данных такая информация отсутствует, то запрос передается на DNS-сервер более высокого уровня, а в случае необходимости, на DNS-сервер, отвечающий за данную зону доменных имен. Ответ от сервера по цепочке вернется к компьютеру пользователя. Такая схема наиболее распространена, однако возможна и другая. Если в базе данных сервера отсутствует запрашиваемая запись "доменное имя - IP-адрес", то пользователю будет возвращен IP-адрес DNS-сервера более высокого уровня, и компьютер пользователя впоследствии сам выполнит запросы к последующим DNS-серверам.

Всемирная паутина (World Wide Web)

С появлением Интернета стал возможным свободный обмен информацией пользователями во всем мире. Однако долгое время Интернет позволял лишь обмениваться файлами и неформатированным текстом. Лишь после возникновения Всемирной паутины в конце 80-х гг. XX века появилась универсальная среда, с помощью которой стало возможно обмениваться информацией любого типа. Тремя главными компонентами Всемирной паутины стали язык разметки гипертекста HTML (HyperText Markup Language), универсальный идентификатора ресурса URL (Uniform Resource Locator) и протокол обмена гипертекстовой информацией HTTP (HyperText Transfer Protocol).

Всемирную паутину можно определить как распределенную информационную систему, основанную на гипертексте. В распределенных системах информация хранится на так называемых

обеспечением, являющихся узлами сети. Информация во Всемирной паутине представляется в виде веб-страниц, которые хранятся на веб-серверах в виде связанных наборов, называемых сайтами. Пользователи сети получают доступ к этой информации с помощью браузеров, специальных программ-клиентов для просмотра HTML-документов. Браузер обеспечивает взаимодействие с веб-серверами по протоколу HTTP и, получив данные в формате HTML, правильно отображают их на экране.

Браузеры

Несмотря на многообразие существующих браузеров, все они обладают общими чертами. Интерфейс браузеров прост и понятен всем пользователям, знакомым с Microsoft Windows. Вид окна браузера Internet Explorer 8 представлен на рисунке 1.1.

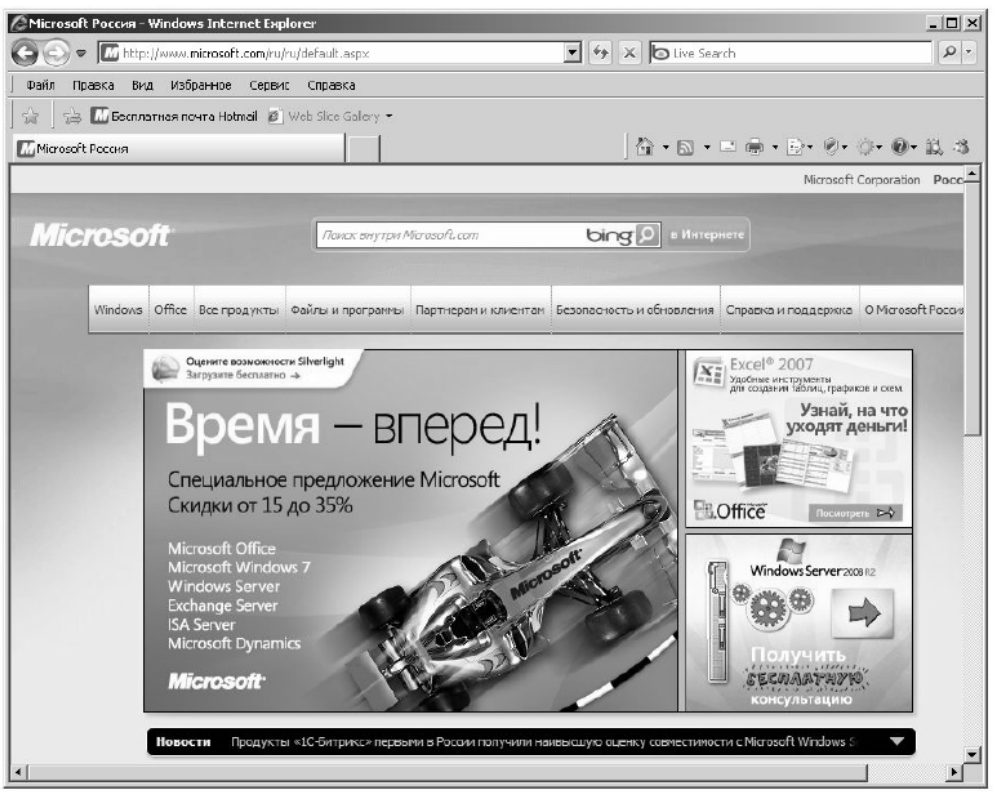


Рис. 1.1. Вид окна браузера Internet Explorer 8

В строке заголовка, которая располагается вдоль верхней границы окна, отображается название используемого браузера и текущего документа. Ниже расположены основные элементы управления программой. Большую часть окна занимает область, в которой отображается просматриваемая веб-страница. В нижней части окна располагается строка состояния, в которой отображается некоторая дополнительная информация.

Согласно статистике сайта "Сайты Рунета" (http://www.liveinternet.ru/stat/ru/browsers.html) тройка наиболее популярных на сегодняшний день браузеров выглядит следующим образом: Firefox 3, Internet Explorer 7 и Opera 10. Динамика использования данных браузеров представлена на рисунке 1.2.

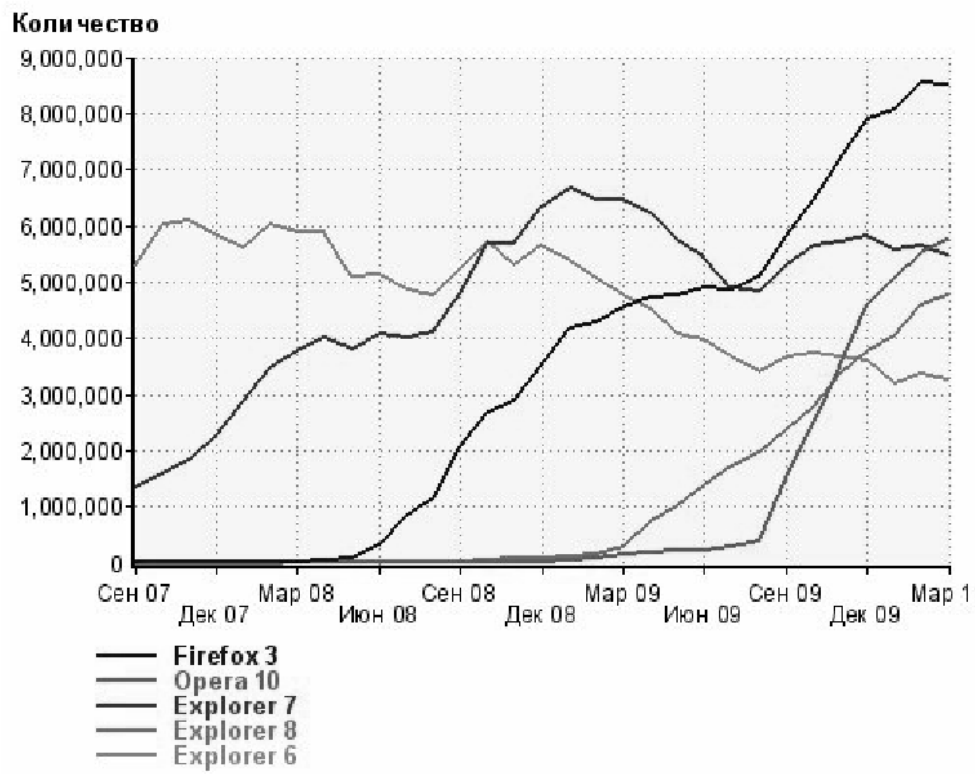


Рис. 1.2. Статистика использования браузеров в Рунете (http://www.liveinternet.ru/)

Internet Explorer 8

К лидерам уверенно приближается новая версия браузера Internet Explorer 8, выпущенная корпорацией Microsoft. Данный браузер имеет массу нововведений, которые способны заинтересовать не только рядовых пользователей, но и опытных разработчиков. Новшества коснулись как удобства работы с браузером и поиска информации, так и некоторых аспектов обеспечения безопасности. К первой группе можно отнести улучшенную навигацию и управление журналом и Избранным, механизмы, реализующие мгновенный поиск и веб-фрагменты, а также механизм ускорителей, позволяющих "посмотреть адрес на карте, выполнить поиск, перевести фразу, прочесть определение незнакомого слова, отправить текст по электронной почте или опубликовать в блоге" и многое другое всего несколькими кликами мыши.

Наиболее заметными нововведениями, относящимися ко второй группе, являются функция восстановления после сбоев, фильтр SmartScreen, позволяющий "защититься от скрытой установки вредоносных программ", а также режим просмотра InPrivate, не позволяющий браузеру сохранять информацию о просмотренных страницах.

Однако наиболее значительным и долгожданным приобретением Internet Explorer 8 являются средства разработчика (Developer Tools), дающие возможность разработчикам веб-сайтов исследовать поведение веб-страниц в браузере, инспектировать и отлаживать HTML-код и стилевые спецификации страницы, отлаживать сценарии JScript и многое другое.

Начать работу со средствами разработчика можно, выбрав в меню "Сервис" команду "Средства разработчика", или нажав клавишу F12. Средства разработчика открываются в отдельном окне для каждой веб-страницы и представляют собой двухпанельный WYSWYG-редактор. При отладке HTML-кода на левой панели отображается документо-объектная модель сайта, а правая панель отображает расширенное содержание выбранного фрагмента кода (см. рисунок 1.3). Команды, расположенные на правой панели, позволяют быстро включать и отключать правила CSS, устанавливая или снимая флажок рядом с правилом (команда "CSS"), получать сведения о расположении

элемента на странице, его высоте и ширине (команда "Раскладка") и исследовать, изменять, добавлять и удалять атрибуты выбранного элемента (команда "Атрибуты").

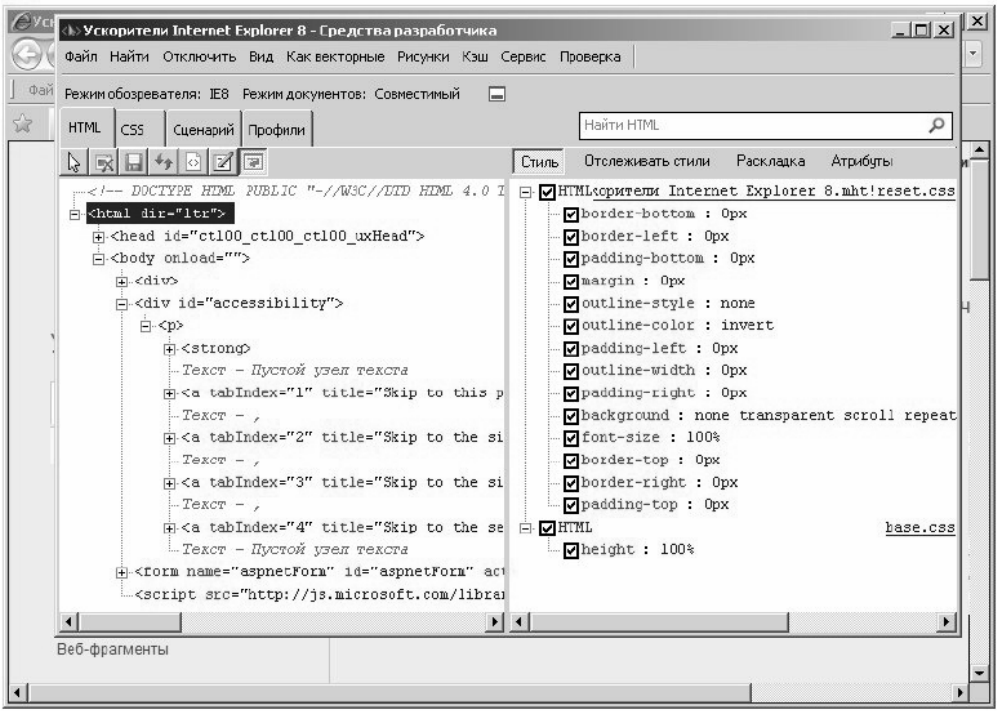


Рис. 1.3. Отладка HTML-кода средствами разработчика

Чтобы просмотреть список всех файлов CSS для данного сайта, можно перейти на вкладку "CSS". Пользователь во вкладке "CSS" выбирает нужный шаблон, описывающий оформление страницы, после чего в левой части окна появляется иерархический список всех атрибутов. Отключение и включение свойства производится простой установкой или снятием галочки на соответствующем пункте (рисунок 1.4).

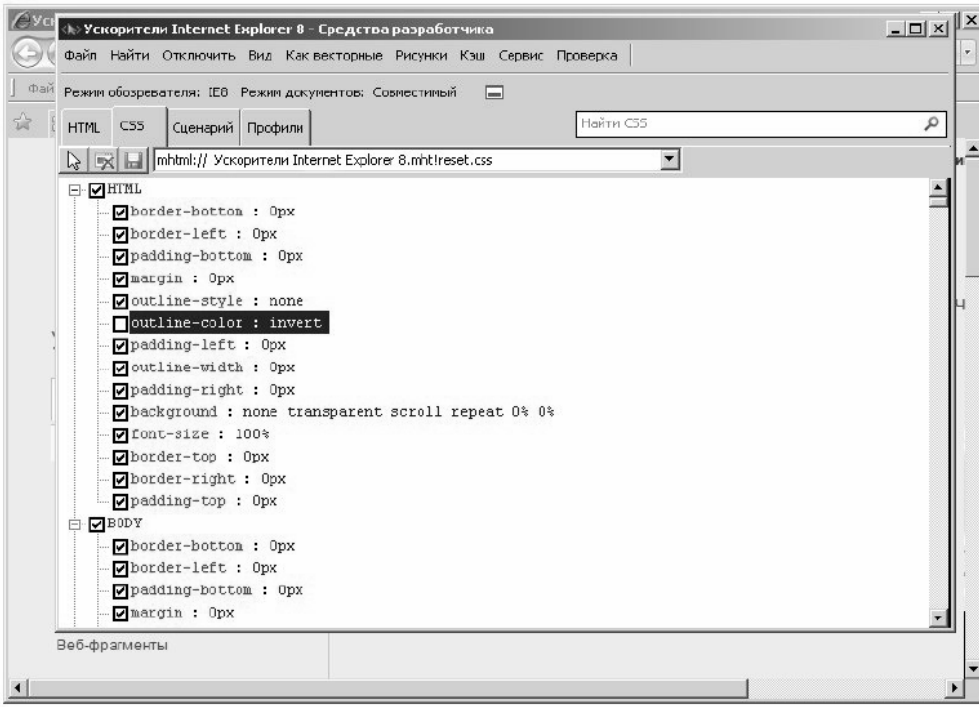


Рис. 1.4. Отладка CSS средствами разработчика

Средства разработчика позволяют редактировать не только любые атрибуты HTML или свойства CSS, но и изменять, добавлять и удалять целые элементы. Внесение изменений осуществляется простой правкой кода или установкой других значений атрибутов, после чего требуется нажать клавишу "Enter". Отмена всех изменений производится или обновлением страницы, или нажатием на кнопку "ESC", если требуется отменять правки по частям. Полученные в ходе редактирования веб-документы можно сохранять на локальном диске компьютера, нажав на вкладках "CSS" и "HTML" кнопку "Сохранить". Чтобы избежать случайной перезаписи кода, выходные данные сохраняются в виде текста, а в начало файла добавляется комментарий.

В лекции рассмотрены лишь некоторые возможности средств разработчика. Получить подробную информацию об Internet Explorer 8, средствах разработчика и многом другом можно на официальном сайте корпорации Microsoft (ссылка: http://www.microsoft.ru/ - http://www.microsoft.ru/).

Введение в стандарты Веб

В лекции рассказывается о веб-стандартах. Введено понятие валидации и описан процесс проверки веб-страниц с помощью валидатора W3C HTML.

Всемирная паутина задумывалась как общее пространство, в котором пользователи могли бы получить доступ к любой представленной информации, общаться, работать над совместными проектами и т.д. Однако на просторах Сети до сих пор можно встретить сайты, оптимизированные только для конкретного вида браузеров, например, Internet Explorer. Посетители таких сайтов, использующие браузеры других производителей, не могут в полной мере получить доступ к ресурсам сайта. В настоящее время такие сайты встречаются все реже и реже, однако встает другая задача: обеспечить доступ к веб-ресурсам пользователям "нестандартных" устройств вывода, например, голосовых, браузеров Брайля, браузеров портативных устройств и др. Чтобы гарантировать каждому доступ к предоставляемой в Интернете информации и были предложены веб-стандарты.

W3C и WaSP

Двумя наиболее авторитетными организациями в области стандартизации Всемирной паутины являются Консорциум Всемирной паутины (World Wide Web Consortium, W3C) и группа специалистов, называющая себя Проект по поддержанию веб-стандартов (Web Standards Project, WaSP).

Консорциум Всемирной паутины был основан 1994 году Тимом Бернерсом-Ли, автором множества разработок в сфере информационных технологий. Создание Консорциума стало закономерной реакцией сообщества веб-разработчиков на так называемые "браузерные войны", бушевавшие в 90-х годах двадцатого века. Миссию W3C можно сформулировать следующим образом: "Полностью раскрыть потенциал Всемирной паутины путем создания протоколов и принципов, гарантирующих долгосрочное развитие Сети". Двумя другими важнейшими задачами Консорциума являются полная "интернационализация Сети" и доступность ее сервисов для

людей с ограниченными возможностями.

Консорциум состоит из частных лиц и представителей различных академических институтов, правительственных организаций и частных компаний. Он объединяет производителей оборудования и программного обеспечения, поставщиков контента и телекоммуникационные компании, такие как Microsoft, Netscape Communications, Apple Computer, Adobe, Sun Microsystems и многие другие. Подразделения Консорциума расположены в трех исследовательских институтах – Массачусетском технологическом институте (MIT) в США, Национальном институте исследований в области компьютерной обработки данных и автоматики (INRIA) в Европе и Университетом Кейо (Keio University) в Японии.

Консорциум W3C разрабатывает для Всемирной паутины единые функциональные требования, называемые рекомендациями и спецификациями, которые и являются веб-стандартами. Рекомендации и спецификации W3C не защищены патентами и доступны для использования всем желающим. Благодаря нескольким степеням внедрения, разработчики могут следовать им лишь частично, не нарушая при этом общих стандартов. W3C не имеет программ сертификации на соответствие своим рекомендациям и спецификациям, поэтому соблюдение стандартов в настоящее время отдается на усмотрение разработчика.

Выработкой Рекомендаций W3C занимаются рабочие группы, состоящие из членов Консорциума и приглашенных экспертов. Любой стандарт проходит четыре стадии согласования: от рабочего проекта до предлагаемой рекомендации, которая представляется членам и директору W3C для формального одобрения и придания ей официального статуса. Более подробную информацию об этой процедуре и этапах рассмотрения рекомендаций можно получить на сайте W3C (ссылка: http://www.w3c.org/ - http://www.w3c.org/).

За почти двадцатилетнюю историю существования Консорциум Всемирной паутины проделал огромную работу, разработав и утвердив более 80 технических спецификаций и рекомендаций. В числе одобренных Консорциумом технологий – язык разметки гипертекста HTML (HyperText Markup Language), расширяемый язык разметки

гипертекста XHTML (Extensible HyperText Markup Language), каскадные таблицы стилей CSS (Cascading Style Sheets), объектная модель документов DOM (Document Object Model) и многие другие, получившие общее название "веб-стандарты".

Большую роль в развитии и популяризации веб-стандартов играет добровольная организация, называемая Проект по поддержанию веб-стандартов (Web Standards Project, WaSP), созданная в конце 90-х годов двадцатого века независимой группой профессиональных веб-разработчиков. Главная цель членов WaSP – "сделать Интернет лучше и для разработчиков, и для конечных пользователей, поощряя создателей и редакторов браузеров и веб-страниц следовать стандартам"; разработчикам рекомендуется взять "для себя за правило придерживаться стандартов при создании веб-страниц". WaSP активно пропагандирует современные веб-стандарты и концепции веб-технологий, способствует оптимизации политики Консорциума путем конструктивной критики и т.д.

Проверка согласованности со стандартами

Правильно разработанные и соответствующие стандартам сайты внешне не отличаются от сайтов, созданных без учета рекомендаций Консорциума W3C. Однако исходный код таких сайтов будет выглядеть по-разному: сайт, созданный в соответствии со стандартами, имеет "чистый" и удобочитаемый код. Для того чтобы проверить, соответствует ли рассматриваемый сайт или какой-либо другой документ веб-стандартам, не вдаваясь в детали оформления кода, можно прибегнуть к валидации.

Валидацией называется проверка соответствия кода документа формальным правилам веб-стандартов. Документ, прошедший процедуру валидации и не имеющий замечаний по коду, считается валидным. Для проверки документов на соответствие популярным веб-стандартам предназначены программы-валидаторы. Консорциум W3C предоставляет пользователям два основных валидатора MarkUp Validator (ссылка: http://validator.w3.org/ - http://validator.w3.org/) и W3C CSS Validator (ссылка: http://jigsaw.w3.org/css-validator/ - http://jigsaw.w3.org/css-validator/). Оба валидатора доступны в Сети и позволяют за несколько

секунд проверить соответствие документов HTML и CSS соответствующим Спецификациям.

W3C MarkUp Validator

Валидатор W3C MarkUp Validator позволяет проверить любой сайт в сети Интернет, локальный HTML-файл или введенный в форму HTML-код. В зависимости от расположения проверяемого документа необходимо выбрать соответствующую вкладку интерфейса валидатора "Validate by URI", "Validate by File Upload" или "Validate by Direct Input". В зависимости от выбранной вкладки пользователю необходимо указать URI-адрес проверяемого сайта, путь к проверяемому документу или скопировать HTML-код, как показано на рисунках 2.1 – 2.3. После нажатия кнопки Check (Проверить), можно получить сообщение о том, соответствует проверяемый сайт стандартам или нет.

Рис. 2.1. Форма для ввода адреса документа

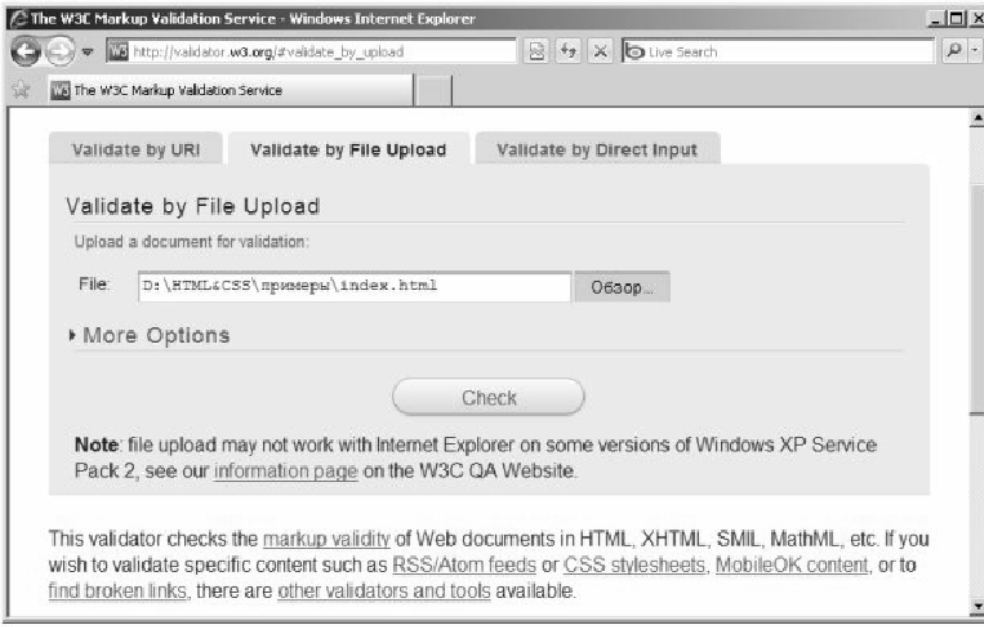


Рис. 2.2. Форма ввода пути к локальному файлу для его проверки

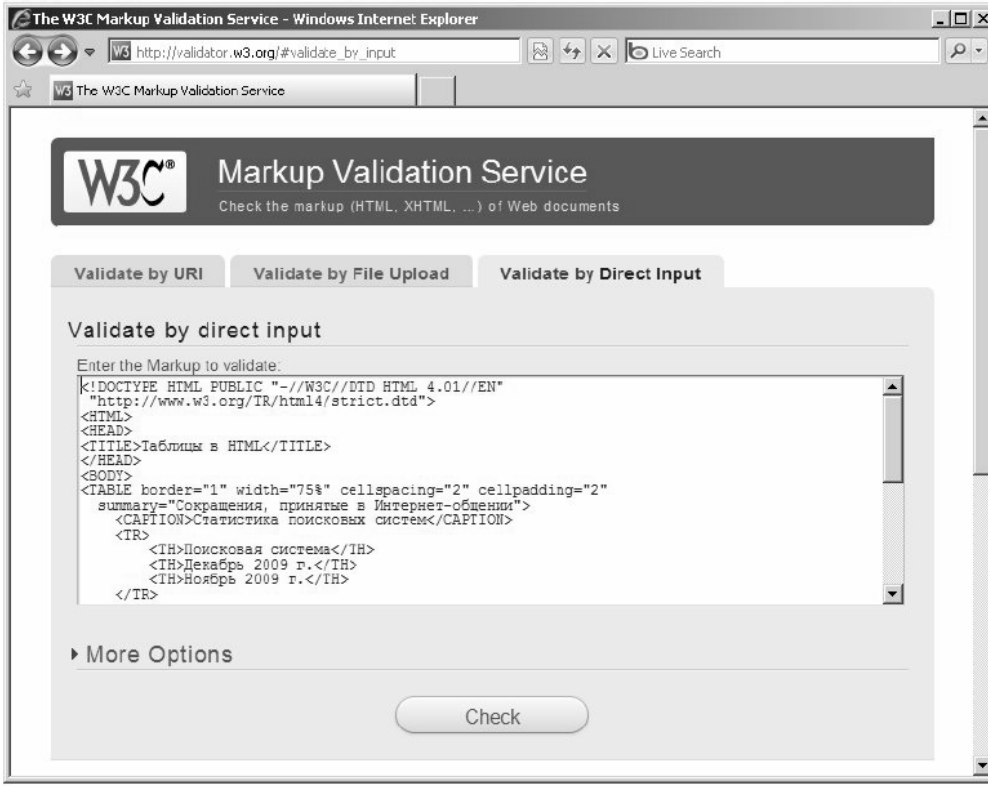


Рис. 2.3. Форма для ввода HTML-кода

Большинство размещенных в Интернете страниц не соответствуют веб-стандартам. По данным компании-разработчика браузера Opera всего около 5% всех страниц в Интернете являются валидными. Многие разработчики считают, что для создания успешного сайта совсем не обязательно строго соблюдать веб-стандарты. Действительно, существует множество успешных проектов, код которых не проходит проверки валидатором. Например, проверка главной страницы самого популярного почтового портала Рунета – Mail.ru дает 270 ошибок (см. рисунок 2.4). На главной странице всеми любимого портала ВКонтакте – 69 ошибок (см. рисунок 2.5).

А вот самый популярный поисковик Рунета – Яндекс успешно прошел валидацию, результаты которой представлены на рисунке 2.6.

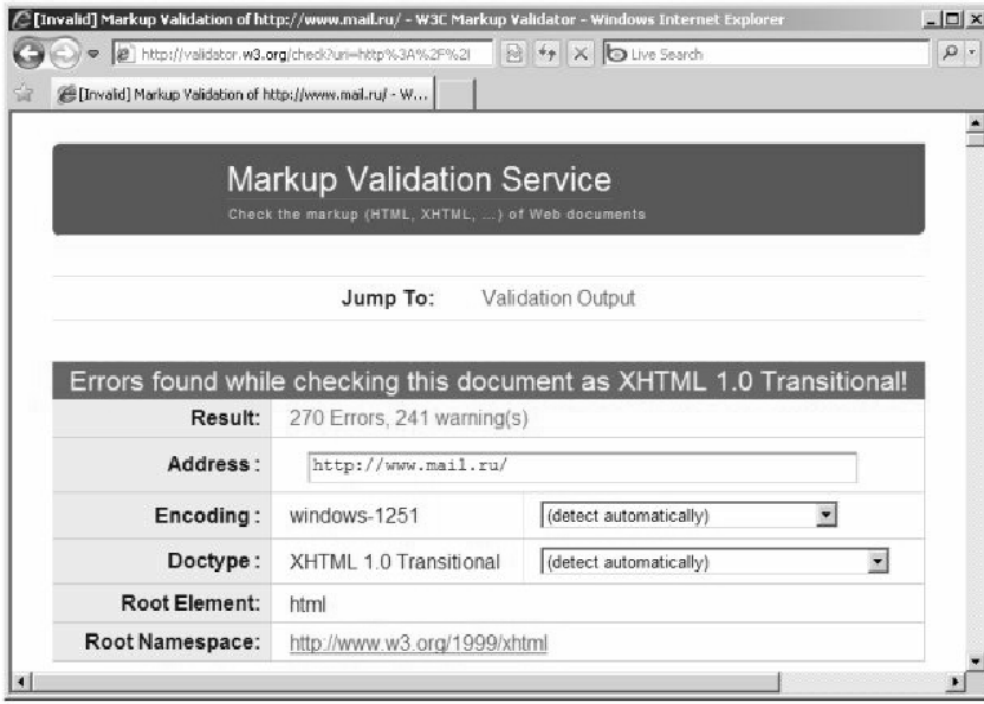


Рис. 2.4. Результаты проверки главной страницы почтового портала Mail.ru

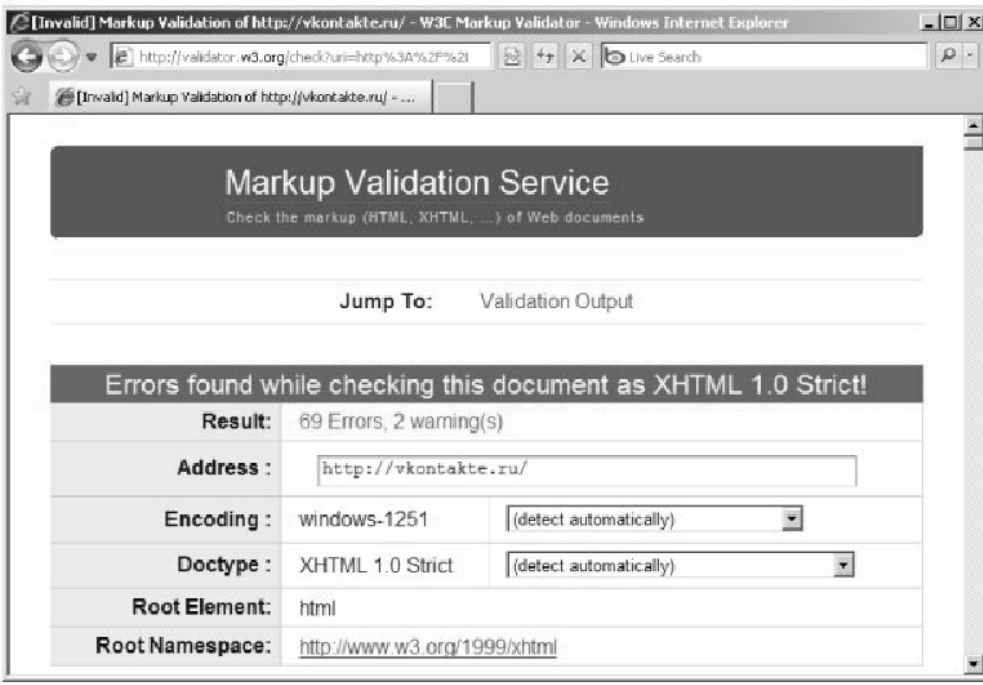


Рис. 2.5. Результаты проверки главной страницы портала ВКонтакте

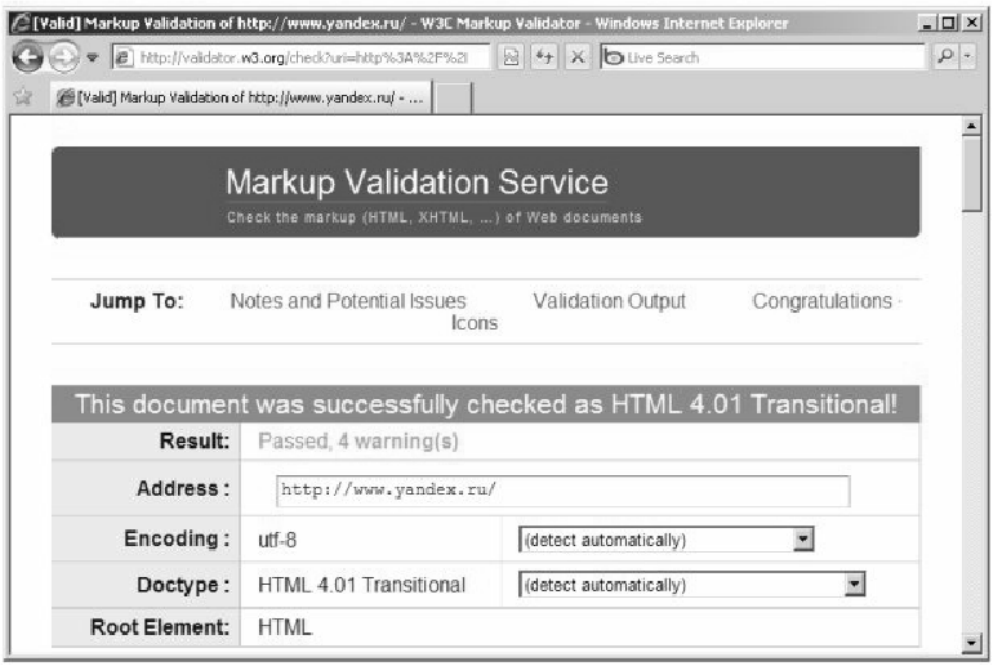


Рис. 2.6. Результаты проверки главной страницы портала Яндекс

Таким образом, полное соответствие кода формальным требованиям стандартов не является обязательным условием для создания хорошего сайта. Однако разработка валидного кода имеет ряд преимуществ как для самого разработчика, так и для конечного пользователя.

Одним из важнейших преимуществ является доступность. Под доступностью понимается обеспечение доступа к ресурсам сети Интернет пользователям "нестандартных" браузеров, в том числе, голосовых, браузеров Брайля, браузеров различных портативных устройств и др. Стандартизация сайтов позволяет гарантировать правильность отображения сайтов большинством этих "нестандартных" устройств.

Другим аспектом доступности является возможность доступа к ресурсам Сети различных поисковых машин и автоматических процессов. Структурная информация, которая содержится в HTML-документах, соответствующих стандартам, эффективнее распознается серверным и клиентским программным обеспечением, что упрощает применение на сайтах поисковых машин и обеспечивает более точные результаты поиска и индексации.

Документы, разработанные в соответствии со стандартами, проще конвертировать в другие форматы, что облегчает их использование и упрощает адаптацию данных к новым системам оборудования или программного обеспечения.

Сайты, выполненные в соответствии со стандартами, более удобны в разработке и сопровождении. Разнесение в различные файлы информации об оформлении и структуре сайта приводит к значительному сокращению объема HTML-файла, а, следовательно, и его "веса". Такие файлы намного быстрее передаются по сети и выводятся на экран браузерами.

Почти все стандарты разрабатываются с расчетом как на предыдущие версии браузеров, так и на те, которые могут появиться в будущем, чтобы документы, созданные в соответствии с устаревшими стандартами отображались в более современных браузерах, а страницы, разработанные с применением новых стандартов, упрощались для

обеспечения приемлемого отображения в старых версиях браузеров.

Таким образом веб-сайт, созданный с соблюдением стандартов, имеет больше шансов для создания доступного, совместимого и оптимизированного под поисковые системы содержимого.

Что нужно хорошей веб-странице?

В лекции рассказывается об общих принципах создания веб-сайта, его основных элементах, а также о некоторых особенностях дизайна.

Планирование веб-сайта

Веб-сайт представляет собой совокупность веб-страниц с повторяющимся дизайном, объединенных по смыслу и физически находящихся на одном веб-сервере. Таким образом, повторяющийся дизайн, общность смысла или концепции и физическое расположение являются основными условиями при создании веб-сайта.

Планирование является важнейшим этапом в разработке любого сайта, будь то простенькая домашняя страничка или гигантский сайт транснациональной корпорации. Для планирования сайта можно использовать различные специализированные программы (например, Microsoft Visio), но обычно достаточно карандаша и бумаги или какого-нибудь текстового редактора.

Перед началом работы над сайтом прежде всего необходимо определить основные задачи сайта, т.е. решить, для чего и для кого предназначается сайт: создается ли он для решения какой-либо проблемы, должен ли сайт привлекать потенциальных клиентов, рассказывать о чем-либо и т.д. Определившись с назначением сайта, необходимо определить, какая именно информация должна на нем размещаться. Данный этап должен включать также и сбор всей необходимой информации.

Собрав все необходимое, можно задуматься над дизайном и решить, в каком ключе будет выполнен сайт: будет ли он консервативным, строгим или затейливым. Например, рекламный сайт лучше сделать повеселее, а новостной сайт должен обладать скромным, консервативным оформлением, чтобы не отвлекать от основного содержания. Для проектирования сайта требуется также разработать (и, желательно, нарисовать) логическую структуру сайта: нужно подумать о перемещении между страницами, о взаимодействии ссылок и т.д.

Продумав логическую структуру, необходимо позаботиться и о физической структуре сайта, т.е. определить, как отдельные файлы,

составляющие сайт, будут размещены в папках. Как правило, файлы организуются по типу: веб-страницы - в одной папке, графические файлы - в другой, мультимедийные файлы - в третьей и т.д. Названия папок должны отражать их содержимое. Например, файлы веб-страниц должны храниться в каталоге HTML, файлы с графическими изображениями – в каталоге с именем IMAGES или PICS и т.д. Файл главной страницы практически всегда помещают в корневой каталог (так называется папка, в которой помещается сайт).

Основные составляющие сайта

Домашняя (или главная) страница является обязательным элементом любого сайта. Именно с нее начинается знакомство посетителя с сайтом. Она отображается, когда пользователь набирает адрес сайта без указания имени файла какой-либо конкретной страницы.

Домашняя страница должна давать посетителю достаточно информации о сайте, при этом не перегружая его излишними сведениями. Обычно она содержит краткую вводную информацию о сайте, новости и меню навигации, позволяющее пользователям перейти на другие страницы сайта. Иногда на домашней странице помещаются сведения о разработчиках, их контактная информация и сведения об авторских правах.

Новости сайта представляют собой хронологический список всех дополнений и обновлений, сделанных на сайте. Новости могут располагаться на главной странице (в этом случае посетитель сайта сразу видит, что на нем изменилось), но чаще помещаются на отдельной странице. Как правило, выводятся только новости за некоторый период. Устаревшая информация помещается в архив новостей, на который должна вести специальная гиперссылка. Новости можно не предусматривать, если сайт обновляется редко или имеет небольшой объем.

Полезное содержимое сайта (или основной контент) - это та информация, ради которой он был создан. Структурируется она так же, как в книге: отдельные абзацы, посвященные какой-либо теме, объединяются в главы, а главы, в свою очередь, в разделы. Таким образом, посетитель сайта сразу сможет найти нужную информацию,

двигаясь от разделов к главам, а от глав к абзацам, пока не найдет то, ради чего сюда пришел.

Организация перемещения по сайту является одним из наиболее важных аспектов реализации. Необходимо определить наиболее важные места сайта и задать их в системе навигации. Подробнее об организации навигации будет рассказано в лекции 8.

Существуют другие элементы сайта, которые повторяются на страницах. Большинство сайтов используют логотип или какие-либо титульные данные, чтобы продемонстрировать право собственности. Заголовок (в верхней части страницы) может содержать логотип и средства навигации.

Полезным дополнением является поле поиска, позволяющее пользователям искать на сайте, а не перемещаться по сайту с помощью меню и ссылок.

Нижний колонтитул (последняя область страницы) должен содержать дополнительную информацию, такую как сведения о разработчиках, указание на авторские права и ссылки на полезные вспомогательные страницы на сайте, если эта информация не вынесена на отдельную страницу.

Юзабилити и доступность

Проектируя сайт, необходимо также учитывать требования юзабилити и доступности.

Термин "юзабилити" можно рассматривать как "конечную суммарную степень удобства, меру интеллектуального усилия, необходимого для получения полезных качеств этой вещи, и скорость достижения положительного результата при управлении ею". Юзабилити сайта является всеобъемлющим термином, определяющим комплекс мер, результатом которого является создание удобного и понятного сайта.

Существуют типовые ошибки, которые обычно приводят к тому, что пользователи сайта не могут найти интересующую их информацию. К таким ошибкам относятся плохая визуализация основных разделов

сайта, сложная система навигации, отсутствие единого стиля оформления сайта, несоответствие содержимого ожиданиям посетителей и т.д. Все эти ошибки юзабилити обусловлены непониманием потребностей пользователей сайта, которые являются важнейшим элементом при проектировании сайта.

Под доступностью понимается обеспечение доступа к сети Интернет не только людям с ограниченными физическими возможностями, но и пользователям "нестандартных" браузеров, в том числе, голосовых, которые читают страницы вслух людям с ослабленным зрением, браузеров Брайля, которые переводят текст на язык Брайля, браузеров портативных устройств с маленькими мониторами, дисплеев с телетекстом и других необычных устройств вывода. Понимание проблем, с которыми могут столкнуться пользователи, является важным при разработке хороших веб-страниц. Создавая сайт, необходимо предусмотреть альтернативный контент для разных групп пользователей, чтобы увеличить потенциальную аудиторию пользователей создаваемого сайта.

Цветовые решения для сайта

При разработке веб-сайта особое внимание следует уделить цветовому оформлению. Цвет привлекает внимание, создает настроение, посылает сообщение. Цвета не существуют сами по себе, они всегда воспринимаются совместно с другими цветами. Для того чтобы понять, как цвета взаимодействуют друг с другом и создать их гармоничное сочетание, используется цветовой круг, пример которого представлен на рисунке 3.1.

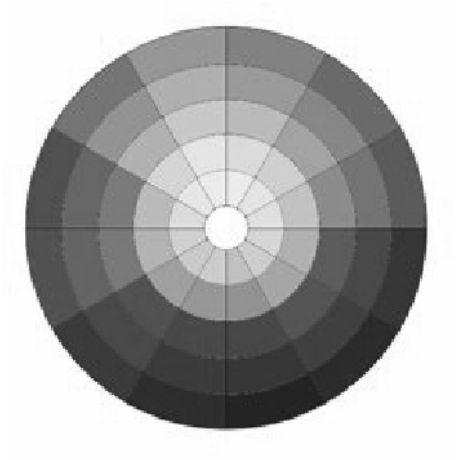


Рис. 3.1. Пример цветового круга: круг естественных цветов по Гете

Практика художников показывает, что многие цвета и оттенки можно получить смешением других цветов в разных пропорциях. Стремление человечества все разложить "на элементы", привело к выделению основных цветов, в качестве которых можно рассматривать красный, желтый и синий. Основные цвета являются "родителями" цветового круга - это единственные цвета, не образованные из других. Основные цвета находятся в третях цветового круга.

Понятие "дополнительный цвет" (или вторичный цвет) было введено по аналогии с "основным цветом". Смешивая соседние основные цвета попарно в равной пропорции можно получить дополнительные цвета: например, смешение красного и синего цветов даст фиолетовый цвет. Так, к триаде основных цветов красный-желтый-синий дополнительными являются оранжевый-зеленый-фиолетовый. Вторичные цвета находятся на цветовом круге строго между основными.

Рассматривают обычно и третичные цвета (или цвета 3-го порядка), которые образуются при смешивании основного цвета с соседним дополнительным цветом, например, смешивая красный (основной) и фиолетовый (дополнительный) можно получить красно-фиолетовый цвет и т.п.

Кроме того, цвета условно делят на теплые и холодные. Теплые цвета, содержащие желтый и красный, более динамичные, выступающие и

объемные. Холодные цвета, расположенные от фиолетовой до зеленой зоны цветового круга, кажутся удаляющимися по мере усиления тона. Эффект движения, вызванный сочетанием холодных и теплых цветов, широко применяется дизайнерами.

Цветовые схемы

Важнейшей задачей веб-дизайнера является поиск гармоничного цветового сочетания. Некоторые цвета отлично смотрятся вместе и дополняют друг друга. Другие же, создают раздражающие глаз сочетания. Чтобы не проводить эксперименты на собственном сайте, необходимо иметь представление об основных цветовых схемах, т.е. о гармоничных комбинациях цветов, составленных по определенным правилам. Ниже будут рассмотрены четыре цветовые схемы, которые являются самыми простыми для понимания и реализации, но далеко не единственными.

Монохроматическая цветовая схема. Монохроматическая цветовая схема соответствует одному базовому цвету и всем его оттенкам, тональностям и теням (см. рисунок 3.2,а). Схема не обладает цветовой насыщенностью, однако она обеспечивает контраст между различными оттенками одного цвета, что очень важно для хорошего дизайна.

Дополнительная цветовая схема. Дополнительная схема образуется сочетаниями противоположных в цветовом круге цветов (см. рисунок 3.2,б). При выборе одного цвета и его противоположного используют также все оттенки, тональности и тени обоих цветов. Такая схема обеспечивает более широкий диапазон выбора. Такое сочетание смотрится довольно грубо, однако в некоторых случаях именно контрастное сочетание способно придать дизайну неповторимый стиль. Обычно, дополнение используется в небольших количествах, как акцент.

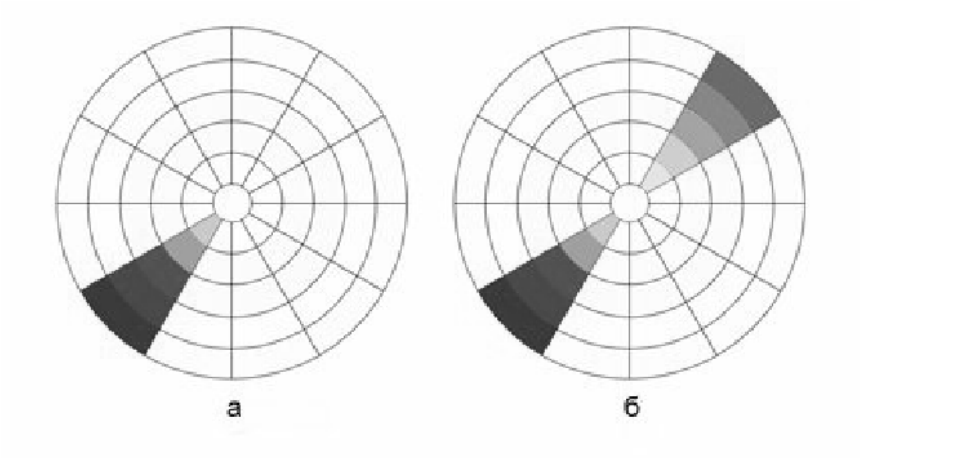


Рис. 3.2. Пример монохроматической (а) и дополнительной (б) цветовых схем

Триадические цветовые схемы. Триадическая цветовая схема создается при выборе одного цвета и добавлении к нему двух других цветов, расположенных на одинаковых расстояниях друг от друга на цветовом круге. Такая схема содержит как теплые, так и холодные цвета, но одна "температура" обязательно должна преобладать. Обычно температура, которая будет преобладать над другими, выбирается для переднего плана. Пример триадической схемы представлен на рисунке 3.3, а .

Тетрадические цветовые схемы. Чем больше цветов выбирается, тем более сложной будет цветовая схема. Данная цветовая схема использует сочетание четырех цветов. Эта схема, представленная на рисунке 3.3, б похожа на дополнительную схему, только используется две пары дополнительных цветов, расположенных на равном расстоянии друг от друга.

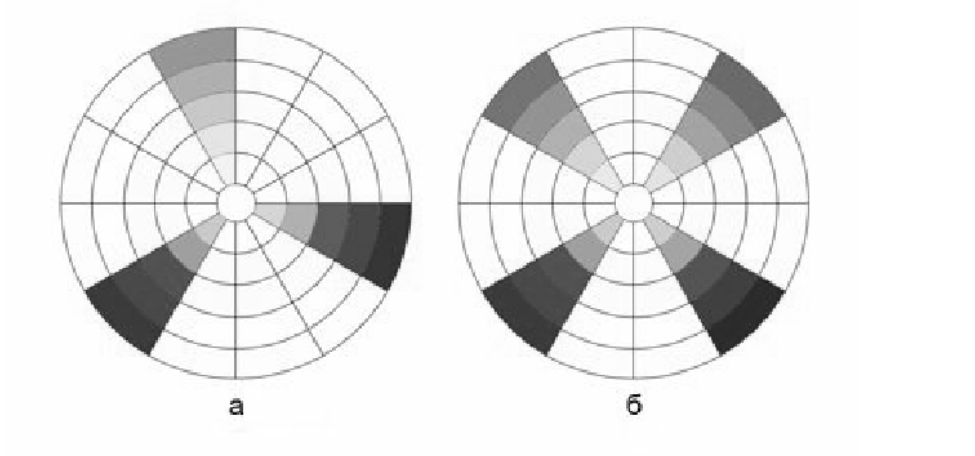


Рис. 3.3. Пример триадической (а) и тетрадической (б) цветовой схемы

В настоящее время существует множество сайтов, с помощью которых можно выбрать цветовую схему не прибегая к помощи цветового круга. Многие из этих ресурсов позволяют пользователям загружать уже готовые цветовые схемы и дорабатывать их. Большинство ресурсов позволяют искать и сортировать цветовые схемы по определенным оттенкам или ключевым словам. Это может быть полезно, если основной цвет уже выбран, и есть необходимость подобрать к нему другие цвета. К одним из лучших сайтов для поиска цветовых схем относятся Color Scheme Designer (ссылка: http://colorschemedesigner.com/ - http://colorschemedesigner.com/), Toucan (ссылка: http://aviary.com/tools/toucan - http://aviary.com/tools/toucan), ColoRotate (ссылка: http://www.colorotate.org/ - http://www.colorotate.org/) и Adobe Kuler (ссылка: http://kuler.adobe.com/ - http://kuler.adobe.com/).

Полиграфия в сети Интернет

Для того, чтобы пользователь заинтересовался ресурсом, создается красивый дизайн, придумывается интересное цветовое оформление и разрабатывается удобная навигация. Но большинство сайтов ставит своей целью донести до аудитории необходимую информацию, будь то реклама, новости или тематические статьи. Именно поэтому основой любого сайта был и остается текст. А для оформления текста придумана полиграфия.

Дизайнеры традиционной печати имеют в своем распоряжении огромный инструментарий, развивавшийся годами: огромное число доступных шрифтов, гибкие возможности позиционирования текста и др. Веб-полиграфия же более ограничена. Основные ее ограничения включают ограниченный выбор шрифтов, отсутствие переноса слов и слабый контроль за кернингом. Рассмотрим эти ограничения подробнее.

Ограниченный выбор шрифтов. С этим ограничением создатели и посетители веб-сайтов встречаются в первую очередь. Хотя в настоящее время можно определить любой шрифт, посетители оценят замысел автора, только если такой шрифт уже установлен на их компьютере. В противном случае, браузер воспользуется значением шрифта по умолчанию, которым обычно является Times New Roman. В связи с этим большинство веб-дизайнеров ограничиваются наиболее общедоступными шрифтами, к которым относятся Times New Roman, Georgia, Verdana, Arial, Courier и некоторые другие.

Переносы слов. Когда речь идет о выравнивании текста, имеется четыре варианта: выравнивание по левому или правому краям, выравнивание по центру и выравнивание по ширине. Текст, выровненный по ширине, выглядит более эстетично, чем текст с "рванными" краями. Такой способ выравнивания можно видеть в большинстве журналов и книг. Однако в Интернет это связано с некоторыми трудностями, в связи с отсутствием автоматического переноса слов, который разбивает слова в подходящем месте, чтобы лучше разместить их на строке. Чтобы полностью выровнять текст, браузеры могут только изменять интервалы между словами, что зачастую приводит к многочисленным пустым пространствам, образованным несколькими идущими подряд пробелами.

Кернинг. Под кернингом понимается процесс настройки пробелов между определенными символами пропорционального шрифта. В пропорциональном шрифте (например, Times New Roman) расстояние между отдельными символами изменяется от символа к символу, в отличие от моноширинного шрифта (например, Courier), где расстояние между символами всегда одинаково. Кернинг используется при печати для уменьшения пространства между буквами, которые размещаются естественным образом. Большинство профессиональных шрифтов

поставляются со встроенными командами кернинга, чтобы предоставить информацию о пробелах в системе воспроизведения текста.

Для текста, размещаемого на сайтах, кернинг с таким уровнем точности фактически недоступен. Единственной возможностью является настройка расстояния между символами в основном тексте, независимо от самих символов. Поэтому при уменьшении интервала между определенными символами будет изменяться интервал и между всеми остальными.

Описав основные ограничения, будет полезным привести некоторые рекомендации, призванные улучшить внешний вид и восприятие текста, размещенного на веб-странице:

- Для глаз утомительны как короткие, так и длинные строки, т.к. читателю приходится напрягать мышцы глаз при переходе от одной строки к другой. Наиболее комфортной является длина строки в 50-65 символов.
- Расстояние между строками текста (интерлиньяж) является важным фактором для удобочитаемости и эстетики текста: слишком малый интерлиньяж затрудняет чтение, слишком большой можно спутать с разделением абзацев.
- Висячие строки, образующиеся в случае, когда последняя строка абзаца слишком короткая или состоит из одного слова, также отрицательно влияют на читабельность текста, т.к. прерывают взгляд читателя, нарушая прямоугольную форму текста.
- В блоках с выравниванием по левому или правому краю необходимо особое внимание уделять "рваным" краям, которые образуются в результате заметной разницы в длине строк и могут сбивать читателя. Край должен быть равномерным, без слишком длинных и слишком коротких строк.
- Выделять слова в тексте нужно так, чтобы не отвлекать читателя. Существует несколько форм выделения: курсивное начертание, полужирное начертание, заглавные буквы, размер шрифта, цвет. Лучше не комбинировать несколько способов, а использовать только один. Так как подчеркнутый текст прочно ассоциируется со ссылкой, недопустимо выделять текст подобным образом.
- Правильное оформление знаков позволяет глазу беспрепятственно

скользить по тексту, облегчая чтение и восприятие текста. Плохо оформленная пунктуация отвлекает внимание даже от хорошего дизайна сайта. Особенно часто путают дефис и тире, кавычки, принятые в русском языке и "симметричные кавычки".

Введение в HTML

В лекции представлены основные сведения о языке HTML: его история, структура HTML-документа, рассмотрен синтаксис элементов HTML. Также в лекции вводится понятие редактора для верстки веб-страниц.

Краткая история HTML

Начало истории HTML можно отнести к 1986 году, когда Международная организация по стандартизации (ISO) приняла стандарт ISO-8879, озаглавленный "Standard Generalized Markup Language (SGML)". Этот стандарт описывал обобщенный метаязык, позволяющий строить системы структурной разметки любых разновидностей текстов. Управляющие элементы (так называемые теги), вносимые в текст при такой разметке, не несли никакой информации о внешнем виде документа, а лишь задавали его логическую структуру, т.е. указывали границы и соподчинение составных частей документа. Размеченный таким образом текст могла интерпретировать любая программа, работающая на какой угодно компьютерной платформе с любым устройством вывода.

Несмотря на всю значительность принципов, лежащих в основе языка SGML, он не имел заметного распространения до тех пор, пока в 1991 г. сотрудник Европейского института физики элементарных частиц Тим Бернерс-Ли не выбрал его в качестве основы для нового языка разметки гипертекстовых документов. Этот язык, ставший самым известным и широко используемым приложением SGML, был назван HTML – HyperText Markup Language, что переводится как язык разметки гипертекста.

Первые версии HTML разделяли все особенности идеологии своего прародителя. Вся разметка была чисто логической, а из более чем сорока тегов HTML версии 1.2, вышедшей в свет в июне 1993 г., только три тега отвечали за физические параметры представления документа.

В сентябре 1993 года группа программистов Национального центра суперкомпьютерных приложений США (NCSA) выпустила первый (и на долгое время единственный) графический браузер Mosaic, благодаря которому язык HTML получил широкое распространение. Браузер

покорил виртуальное пространство с поразительной быстротой – всего за год около двух миллионов пользователей установили Mosaic на свои компьютеры.

В апреле 1994 г. под эгидой созданного в том же году Консорциума Всемирной паутины (World Wide Web Consortium, W3C) началась подготовка новой версии языка HTML 2.0, ставшей официальной рекомендацией W3C лишь в сентябре 1995 г. Стандарт HTML 2.0 вобрал в себя всю сложившуюся к 1994 году практику создания веб-сайтов.

В марте 1995 г. началась работа над проектом HTML 3. К этому времени уже достаточно очевидно стало противоречие между идеологий чисто логической разметки существовавших версий HTML и потребностями пользователей, заинтересованных в расширении средств визуального оформления. Чтобы разрешить это противоречие, не отказываясь от парадигмы структурной разметки, авторы HTML 3 ввели поддержку нового средства - так называемых иерархических стилевых спецификаций (Cascading Style Sheets, CSS). Стилевые спецификации представляли собой отдельный по отношению к структурной разметке "информационный слой" и были предназначены только для визуального форматирования структурных элементов документа. К сожалению, работа над этой версией была прервана в связи с отсутствием поддержки со стороны производителей браузеров. Принятая чуть позже рекомендация HTML 3.2 потеряла многие новые свойства 3.0, однако закрепила разработки популярных в то время браузеров Netscape Navigator и Internet Explorer.

В 1997 году консорциум W3C опубликовал версию HTML 4.0 в качестве рекомендации, которая включила дополнительные специальные расширения браузеров, но попыталась также рационализировать и очистить HTML. Многие элементы и атрибуты (в основном касающиеся визуального оформления) были помечены как не рекомендуемые. Это должно было стимулировать более "правильное" использование HTML.

Версия HTML 4.01 была опубликована в 1999 г. Это самая последняя версия HTML, хотя W3C уже опубликовал черновой вариант спецификации пятой версии языка HTML. Работа над черновой версией спецификации HTML 5 началась в марте 2008 года. Для этого была сформирована специальная группа, объединившая порядка пятисот

участников, среди которых специалисты таких компаний, как Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera, BEA Systems, Cisco, France Telecom и Hewlett-Packard.

Среди наиболее заметных и важных нововведений в HTML 5 Консорциум W3C выделяет программные интерфейсы для работы с двумерной графикой, средства внедрения в веб-страницы видео- и аудиоматериалов, а также инструменты, позволяющие посетителям самостоятельно редактировать сайты. Ознакомиться с черновым вариантом спецификации HTML 5 можно на официальном сайте Консорциума (ссылка: http://www.w3.org/TR/2008/WD-html5-20080122/ - http://www.w3.org/TR/2008/WD-html5-20080122/).

Семантические требования HTML

Ключевыми моментами в Спецификации HTML являются синтаксические правила HTML и определение семантики его элементов.

Прежде чем дать определение термина "семантика" применительно к языку гипертекстовой разметки, необходимо разобраться с понятием структурной разметки.

В общем случае структура документа представляет собой определенное расположение и связь его частей, составляющих единое целое. Согласно идеологии HTML, вносимые в документ управляющие конструкции не должны нести какой-либо информации о визуальном представлении документа. Они предназначены только для указания границ и соподчинения составных частей HTML-документа. Другими словами, при разметке структуры документа разработчик должен полностью абстрагироваться от вопросов представления и всего лишь создать "каркас" будущего HTML-документа.

Естественно, что для правильного описания структуры документа необходимо четко представлять себе назначение каждого конкретного элемента HTML, что как раз и является семантикой элемента. Таким образом, семантическая верстка представляет собой выбор элементов HTML исходя из их смыслового назначения, а не на основе того, как он выглядит в браузере.

Ярким примером нарушения правил семантической верстки является так называемая табличная верстка, которая представляет собой метод верстки документов, при котором структурный элемент `TABLE` используется для управления визуальным расположением других элементов, а не для представления табличных данных, что является его смысловым назначением. Поэтому использование элемента `TABLE` для представления нетабличных данных является нарушением его семантики.

Семантическая верстка неотделима от концепции разделения структуры и представления, согласно которой язык разметки гипертекста HTML должен использоваться только для описания структуры документа (т.е. его содержания), а для визуального представления этой структуры (т.е. его оформления) предлагается другой официально утвержденный W3C стандарт – каскадные таблицы стилей CSS (Cascading Style Sheets).

Составные элементы HTML-документа

Элементы

Любой HTML-документ представляет собой набор элементов, описывающих отдельные составляющие документа, такие как заголовки, списки, абзацы текста, таблицы и др. Имена большинства элементов представляют собой общеупотребительные слова английского языка, понятные сокращения и обозначения.

Чаще всего элемент разметки состоит из трех частей: начального и конечного компонентов, между которыми размещаются текст или другие элементы документа. Эти компоненты представляют собой специальные управляющие конструкции для разметки содержимого HTML-документа и называются тегами. Начальный тег состоит из имени элемента, заключенного в угловые скобки (< и >). Конечный тег отличается от начального тем, что перед именем элемента в нем ставится косая черта (/). Например, элемент EM, используемый для акцентирования текста, будет иметь следующий вид:

```
<EM>акцентируемый текст</EM>
```

Следует также заметить, что имена элементов не чувствительны к регистру, т.е. записи `<EM>` и `<em>` равнозначны.

Элементы должны либо следовать друг за другом, либо быть вложены один в другой. Так, если начальный тег `<STRONG>` расположен внутри элемента `<P>…</P>` , то и конечный тег `</STRONG>` должен быть расположен внутри этого элемента.

Конечные теги некоторых элементов в документе можно опускать, т.к. большинство браузеров устроено так, что при обработке текста документа начальный тег воспринимается как конечный тег предыдущего. Например, конечный тег элемента `LI`, используемого для создания пункта списка, не обязателен, поскольку начало очередного пункта списка означает конец предыдущего. Есть и другие конечные теги, без которых браузеры отлично работают, например, конечный тег `</HTML>`. Тем не менее, рекомендуется не опускать конечные теги, чтобы избежать путаницы и ошибок при воспроизведении документа. Кроме того, некоторые элементы, такие, как, например, элемент IMG, используемый для вставки в HTML-документ изображения, не имеют конечного тега, поскольку не имеют содержимого.

Все элементы HTML делятся на две категории: блочные элементы и строковые элементы. Блочный элемент обычно информирует о структуре документа. По умолчанию блоки начинаются с новой строки и отделяются от предыдущего и последующего блока пустой строкой определенной ширины. Блоки могут быть вложены друг в друга, а блочные элементы могут содержать строковые элементы. К основным блочным элементам относятся параграфы, заголовки, предварительно форматированный текст, цитаты, разделитель, таблицы и списки.

Строковые элементы содержатся внутри структурных блочных элементов и охватывают не целые области, а только части текста документа. Строковые элементы обычно появляются в параграфах текста и не приводят к появлению в документе новой строки. Распространенными строковыми элементами являются гипертекстовые ссылки, изображения, выделенные слова или фразы и краткие цитаты.

Атрибуты

Элементы могут также иметь атрибуты, которые определяют дополнительную информацию о них. Многие атрибуты в HTML являются общими для всех элементов, однако большинство из них являются специфическими для данного элемента или группы элементов.

Атрибуты размещаются в начальном теге после имени элемента и отделяются друг от друга одним или несколькими пробелами, причем порядок следования атрибутов в теге значения не имеет. Значение атрибута (если таковое имеется) следует за знаком равенства, стоящим после имени атрибута. Значение должно быть помещено в одиночные или двойные кавычки. Несмотря на то, что в некоторых ситуациях кавычки можно опустить, Спецификация HTML рекомендует всегда заключать значения атрибутов в кавычки. Имена атрибутов могут быть набраны в любом регистре, однако их значения могут зависеть от регистра.

Все атрибуты и их возможные значения определяются в основном Спецификациями HTML (ссылка: http://www.w3.org/TR/html401/index/attributes.html - http://www.w3.org/TR/html401/index/attributes.html). Пользователь не может создавать свои собственные атрибуты или использовать значения, не определенные Спецификацией, так как это может вызывать проблемы правильной интерпретации веб-страницы.

Как было отмечено, ряд атрибутов применим практически ко всем элементам HTML. Наиболее часто используемыми атрибутами являются базовые атрибуты (`class`, `id`, `style` и `title`), определяющие общие свойства элементов, и локализующие атрибуты (`dir` и `lang`), которые указывают на свойства языка написания содержимого элемента. Кратко рассмотрим эти атрибуты.

Атрибут `id`, называемый также идентификатором элемента, присваивает элементу имя, уникальное в пределах данного документа, т.е. никакие два элемента не могут иметь одинаковых значений `id`. В основном идентификатор применяется в качестве селектора стилей отдельных элементов, в качестве закладки для гиперссылок, а также для указания на конкретный элемент из сценария. В следующем примере атрибут `id` использован для идентификации параграфа:

```
<P id="footer">Знание – сила! </P>
```

Атрибут `class` задает принадлежность элемента определенному классу. В отличие от атрибута `id`, к одному классу может относиться любое количество элементов. Кроме того, элемент может принадлежать к нескольким классам: в этом случае значением атрибута является список имен классов, разделенных пробелами. Однако большинство браузеров не поддерживают списки классов. Атрибут `class` в основном используется в качестве селектора стилей. В следующем примере созданный параграф включается в класс элементов с именем `header`:

```
<P class="header">Заглавный текст</P>
```

Атрибут `style` позволяет задать стиль элемента. Однако для использования этого атрибута в заголовок документа должен быть включен метаописатель. Следующий пример демонстрирует способ изменения цвета шрифта для заданного параграфа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<META http-equiv="Content-Style-Type" content="text/css">
...
</HEAD>
<BODY>
...
<P style="color: red">Мир, труд, май! </P>
...
</BODY>
</HTML>
```

В большинстве случаев употребление атрибутов `class` и `id` предпочтительнее, т.к. они обеспечивают разделение содержимого документа и стиля его отображения, что в большей степени согласуется с концепцией разделения структуры и представления.

Все три описанных выше атрибута можно применить ко всем

элементам, кроме элементов BASE, HEAD, HTML, META, SCRIPT, STYLE и TITLE.

Атрибут title определяет заголовок элемента и часто используется обозревателями в качестве подсказки, которая выводится на экран, когда курсор помещается на данный элемент. Приведенный ниже пример демонстрирует использование атрибута title совместно с элементом A, создающим ссылку на веб-страницу:

```
<A href="http://www.microsoft.com"
title="Официальный сайт Microsoft">Microsoft</A>
```

Атрибут lang определяет язык, на котором написаны значения остальных атрибутов данного элемента и его содержимое, а также всех вложенных элементов, не имеющих аналогичного атрибута. Использование атрибута позволяет авторам менять стиль текста в зависимости от языка. Например, текст двуязычного документа может быть оформлен следующим образом:

```
...
<P lang="en">This paragraph is in English. </P>
<P lang="ru">Этот абзац на русском языке. </P>
...
```

Атрибут dir определяет направление вывода текста: слева направо (dir="ltr", по умолчанию) или справа налево (dir="rtl"). Для всех символов в кодировке Unicode в целях правильного отображения текста определено направление. Так, латинские и русские буквы выводятся слева направо, а еврейские и арабские – справа налево.

Стандарт Unicode определяет двунаправленный алгоритм, который должен применяться всякий раз, когда документ содержит символы, выводимые справа налево. Хотя обычно этот алгоритм дает правильное изображение текста, существуют ситуации, когда направление вывода текста приходится задавать явно с помощью атрибута dir.

Специальные символы и ссылки-мнемоники

В документ HTML также могут быть включены ссылки-мнемоники (или символьные ссылки). Данные ссылки предназначены для ввода в документ нестандартных и специальных символов. К нестандартным относятся символы, которые трудно или невозможно ввести с помощью клавиатуры или в кодировке конкретного документа. К специальным символам относятся символы, которые начинают и заканчивают части документа HTML, а не представляют соответствующие символы (<, >, & и ").

Ссылки-мнемоники начинаются знаком & и заканчиваются точкой с запятой (;). Однако многие браузеры могут быть достаточно лояльны к ошибкам HTML, таким как отсутствие точки с запятой.

Ссылки могут быть либо числами (числовые ссылки), либо сокращенными словами (объектные ссылки). Например, амперсанд может быть введен в документ как &, что является объектной ссылкой символа, или как &, что является числовой ссылкой. Полную таблицу символьных ссылок можно найти на сайте evolt.org (ссылка: http://www.evolt.org/article/A_Simple_Character_Entity_Chart/17/21234 - http://www.evolt.org/article/A_Simple_Character_Entity_Chart/17/21234).

Комментарии в HTML

HTML-документы могут содержать комментарии, которые являются удобным средством для описания кода, поиска разделов документа или объяснения мотивов, которыми руководствовался разработчик кода. Комментарии не влияют на отображение документа, а только поясняют его содержимое при просмотре HTML-кода. Текст комментариев располагается между двумя группами символов: открывающим ограничителем (<!--) и закрывающим ограничителем (-->). В тексте комментариев запрещается использовать два или более подряд идущих дефиса.

Комментарии в HTML могут выглядеть следующим образом:

```
<!-- это комментарий -->
<!-- это комментарий,
```

```
    занимающий две строки -->
```

Структура HTML-документа

Пример HTML-документа выглядит следующим образом:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
   <TITLE>Пример простого HTML-документа</TITLE>
</HEAD>
<BODY>
   <P>Это самый простой HTML-документ.</P>
</BODY>
</HTML>
```

Пример выполнения этого кода браузером представлен на рисунке 4.1.

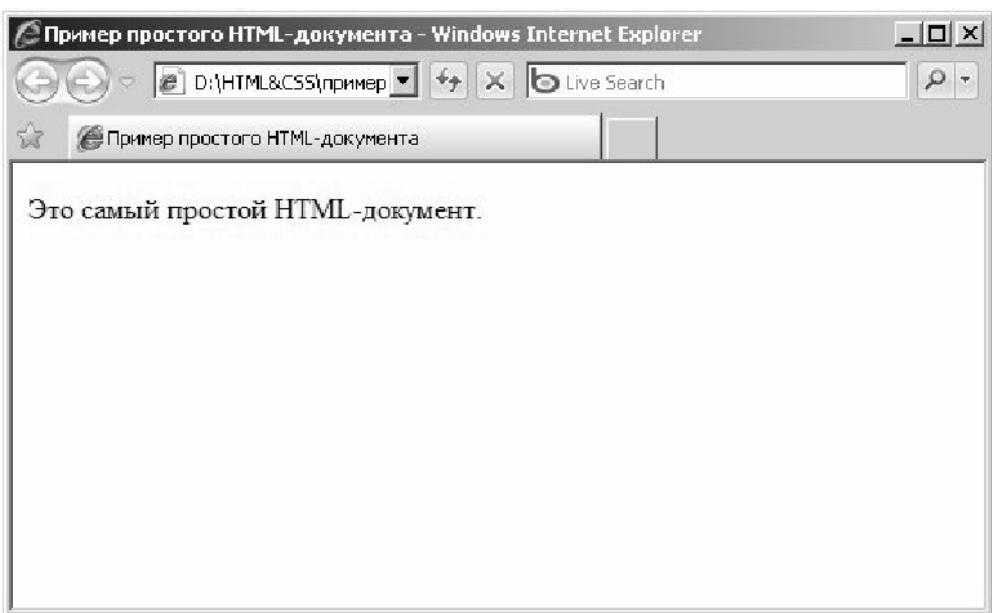


Рис. 4.1. Вид в браузере простого HTML-документа

Каждый документ рекомендуется начинать с элемента декларации типа документа, или DOCTYPE, который описывает используемый тип и версию HTML. Данная информация предназначена прежде всего для браузеров и программ-валидаторов разметки. DOCTYPE указывает браузеру основные синтаксические элементы для каждой версии HTML, а также режим воспроизведения. Валидаторы разметки просматривают DOCTYPE, чтобы определить, согласно каким правилам они должны проверять данный документ (более подробно DOCTYPE рассматривается в лекции 5).

За декларацией типа документа следует элемент <HTML>…</HTML>, который указывает, что последующий документ является HTML-документом и включает в себя все остальное содержимое документа. Данный элемент называется корневым элементом документа. Он может содержать только элементы HEAD и BODY, причем каждый из этих элементов может встречаться внутри элемента HTML не более одного раза. Данный элемент имеет два необязательных атрибута: lang и dir.

Внутри элемента HTML располагается заголовок. Заголовок документа, создаваемый элементом <HEAD>…</HEAD>, содержит информацию об общих свойствах документа. Данный элемент в обязательном порядке должен содержать элемент TITLE и может содержать необязательные элементы BASE, SCRIPT, STYLE, META, LINK и OBJECT. Элемент HEAD имеет три необязательных атрибута: lang, dir и profile ; последний указывает местонахождение словаря метаданных. Предполагается, что такой словарь должен содержать имена метапеременных, значения которых определяются элементами META и LINK в заголовке документа.

После элемента HEAD следует элемент <BODY>…</BODY>, который является оболочкой, содержащей реальный контент веб-страницы. Элемент BODY обязательно должен содержать хотя бы один блочный элемент: в данном случае это элемент P, создающий параграф, который содержит текст "Это самый простой HTML-документ.". Элемент BODY имеет довольно много атрибутов, однако рекомендуемыми к применению являются только id, class, style, title, lang и dir.

Редакторы для верстки веб-страниц

Редактор для верстки веб-страниц или HTML-редактор - это компьютерная программа, позволяющая составлять и изменять документы в формате HTML. Сегодня на запрос в сети Интернет по ключевому словосочетанию "HTML-редактор" можно получить перечень из сотни разнообразных программ. Однако все редакторы делятся на два класса: визуальные и текстовые.

Визуальные редакторы не требуют от разработчика знаний HTML, CSS и других технологий разметки документов. Пользователь просто располагает различные элементы будущей страницы в окне редактирования, а редактор сам генерирует соответствующий код. Именно поэтому визуальные редакторы еще называют WYSIWYG-редакторами, что означает What You See Is What You Get - что видишь, то и получаешь.

Визуальные редакторы обычно располагают массой функций, призванных облегчить работу разработчику. Так, хороший редактор должен обладать подсветкой синтаксиса, возможностью предварительного просмотра созданного документа, менеджером проектов, библиотеками различных элементов и др.

К популярным визуальным редакторам относятся продукты компании Adobe - GoLive и Dreamweaver, а также Microsoft FrontPage и Microsoft Expression Web. Данные редакторы являются проприетарными, хотя производители предлагают всем желающим ознакомительные версии данных программ. Так, ознакомительную версию Microsoft Expression Web можно получить, перейдя по ссылке ссылка: http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=0a73a3a7-3e06-4125-b3c6-f9c10387e9cc - http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=0a73a3a7-3e06-4125-b3c6-f9c10387e9cc. Среди свободно распространяемых редакторов стоит выделить редактор Nvu, который хоть и уступает в функциональности упомянутым выше редакторам, однако бесплатное распространение делает его привлекательным для многих разработчиков.

Однако ни один визуальный редактор не совершенен, и все они так или

иначе ограничены в своих возможностях. Поэтому профессиональные разработчики часто пользуются небольшими текстовыми редакторами. Текстовые редакторы, как правило, содержат набор функций, облегчающих разработчику написание кода. К наиболее распространенным функциям относятся подсветка кода, различные горячие клавиши и т.д. К наиболее популярным текстовым редакторам относятся Macromedia HomeSite и HTML Pad. Однако лидером среди текстовых редакторов является Notepad (он же Блокнот). В этой программе нет никаких специальных функций, но зато она есть у каждого пользователя Windows, т.к. входит в число стандартных. С нее можно начать свои первые шаги в написании кода, а затем уже сменить на более понравившийся редактор.

DOCTYPE и раздел документа HEAD

В лекции рассматривается разметка внутри элемента head, рассказывается о различных частях этого раздела и их предназначении. Также описаны функции объекта DOCTYPE и порядок выбора DOCTYPE для HTML-документа.

Объявление DOCTYPE

Согласно Спецификации HTML 4.01, директива `DOCTYPE` предназначена для объявления типа документа - так называемого DTD (Document Type Definition, определение типа документа). DTD представляет собой формальную конструкцию, в которой выражена вся специфика HTML как одного из приложений языка SGML. Определение типа документа представляет собой перечень различных конструкций SGML, которые определяют, например, какие элементы в каком порядке могут встречаться внутри каждого из элементов; полный список допустимых элементов с указанием на обязательность для каждого из них начального и конечного тегов; полный список атрибутов для каждого элемента и значениями по умолчанию и другие правила синтаксиса.

Объявление `DOCTYPE` выглядит следующим образом:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Наиболее важной частью `DOCTYPE` являются две строки, взятые в кавычки. Первая строка "-//W3C//DTD HTML 4.01//EN" утверждает, что данный документ является документом DTD, использует английский язык текста для описания объекта, описывает HTML версии 4.01 и опубликован организацией W3C. Вторая строка "http://www.w3.org/TR/html4/strict.dtd" указывает на размещение самого документа DTD, используемого для этого `DOCTYPE`.

Режимы представления

Все современные браузеры ориентируются на DOCTYPE (или его отсутствие) в начале страницы, выбирая режим отображения для HTML-документов. В Internet Explorer версии 8 режим определяется также и другими факторами, например, HTTP-заголовком, периодически получаемыми от Microsoft данными, пользовательскими настройками и др. Но по умолчанию даже в Internet Explorer 8 режим зависит от DOCTYPE.

Основными режимами современных браузеров являются режим обратной совместимости, стандартный режим и почти стандартный режим. Приведем некоторые характеристики названных режимов.

При режиме обратной совместимости (Quirks Mode) в браузерах нарушаются рекомендации W3C для обеспечения нормального отображения страниц. Если в коде HTML-документа используется неполный или устаревший вид DOCTYPE или DOCTYPE вообще отсутствует, то браузер перейдет в данный режим и будет исходить из предположения, что код страницы написан с ошибками, не соответствует принятым стандартам или документ написан для "старых" браузеров. В этом режиме браузер пытается разобрать страницу по правилам обратной совместимости и выводит ее на экран так, как вывел бы ее браузер более ранней версии. Для разных браузеров существуют различные варианты совместимости с предыдущими версиями. Так, Internet Explorer версий 6, 7 и 8 в режиме обратной совместимости фактически воспроизводят установки, типичные для Internet Explorer 5. У других браузеров режим обратной совместимости представляет собой набор отклонений от почти стандартного режима. К сожалению, режим обратной совместимости лидирует с большим отрывом. Статистика использования типов DTD от 10 апреля 2008 г., опубликованная на сайте Qindex.info (ссылка: www.qindex.info/Q_get.php?g_clss=forum&g_prcss=thrd&g_tmplt=&g_brd=5&g_thrd=128 - http://www.qindex.info/Q_get.php?g_clss=forum&g_prcss=thrd&g_tmplt=&g_brd=5&g_thrd=128) показывает, что более 50% сайтов работают в режиме обратной совместимости.

В стандартном режиме (Standards Mode) браузеры пытаются обращаться с правильно составленными документами в полном соответствии со спецификацией CSS, настолько, насколько конкретный браузер поддерживает стандарты. Поскольку уровень поддержки стандартов

разными браузерами различен, то даже стандартный режим пока не может гарантировать полностью одинакового отображения и поведения страниц.

Некоторые браузеры, такие как Firefox, Safari, Opera (начиная с 7.5) и Internet Explorer 8 поддерживают и третий режим - почти стандартный (Almost Standards Mode), который не достаточно строго следует рекомендациям W3C. Internet Explorer 6 и 7 для Windows, Opera ниже версии 7.5 и некоторые другие браузеры не нуждаются в таком режиме, поскольку даже в своих стандартных режимах не соблюдают все спецификации CSS.

Internet Explorer 8 также поддерживает режим, в основном воспроизводящий стандартный режим Internet Explorer 7. У других браузеров подобных режимов нет.

Выбор DOCTYPE

Спецификация допускает использование трех различных версий DTD, которые отличаются друг от друга поддержкой различных элементов и атрибутов. Опубликованный W3C список DTD, рекомендованных для использования в веб-документах, представлен на официальном сайте Консорциума (ссылка: http://www.w3.org/QA/2002/04/valid-dtd-list.html - http://www.w3.org/QA/2002/04/valid-dtd-list.html).

Любое из перечисленных ниже объявлений `DOCTYPE` гарантированно включает браузеры в стандартный режим.

Объявление строгого DTD:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
                "http://www.w3.org/TR/html4/strict.dtd">
```

Строгое DTD (строгий синтаксис) не допускает использования в документе различных не рекомендуемых элементов и атрибутов, большинство из которых предназначаются не для логической разметки, а для визуального форматирования. Данный `DOCTYPE` гарантирует, что при работе с данным HTML документом браузер будет использовать

свой стандартный режим. Наиболее заметный эффект состоит в том, что будут получены более согласованные результаты при оформлении документа с помощью CSS.

Объявление переходного DTD:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```

Переходное DTD (переходный синтаксис) включает в себя все элементы и атрибуты строгого DTD в совокупности с не рекомендуемыми элементами и атрибутами.

Объявление DTD "Набор фреймов":

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
```

Объявление DTD "Набор фреймов" аналогично переходному синтаксису, но содержит также элементы для создания фреймов.

Раздел документа HEAD

Раздел документа `HEAD` определяет его заголовок и не является обязательным элементом, однако правильно составленный заголовок может быть очень полезен. Элементы, находящиеся в разделе `HEAD`, определяют большую часть инструкций для браузера, а также дополнительную информацию о HTML-документе. Содержимое элемента `HEAD` не отображается напрямую на веб-странице, за исключением элемента TITLE, устанавливающего заголовок веб-страницы. Раздел `HEAD` может содержать в себе следующие элементы: `TITLE`, `META`, `BASE`, `LINK`, `STYLE`, `SCRIPT` и `OBJECT`. В лекции мы дадим лишь основные подходы к работе с перечисленными элементами. Более подробно ознакомиться с ними можно в Спецификации HTML.

Элемент TITLE

Одним из наиболее важных элементов заголовка является элемент TITLE, который задает название HTML-документа. Этот элемент разметки не является обязательным, однако его использование настоятельно рекомендуется. Текст, содержащийся в TITLE, выводится почти всеми браузерами в панели заголовка браузера, а также отображается в виде ссылки на странице результатов поиска в поисковых системах, нажав на которую, пользователи переходят из поисковой системы на страницу сайта.

Оптимальная длина содержимого элемента не должна превышать 80 символов, т.к. более длинный текст заголовка уходит за видимую область окна браузера, а также может быть воспринят поисковой машиной как спам.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
  <TITLE>Пример использования элемента TITLE</TITLE>
</HEAD>
<BODY>
  <P>Обратите внимание на панель заголовков!</P>
</BODY>
</HTML>
```

Результат выполнения данного кода представлен на рисунке 5.1. Текст из элемента TITLE выведен в панели заголовка выше средств навигации браузера.

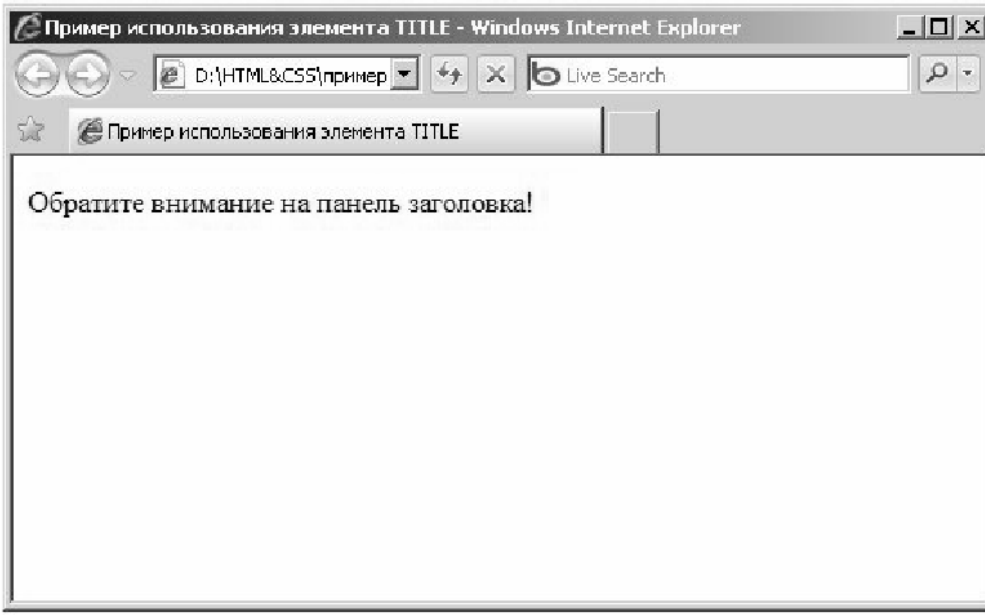


Рис. 5.1. Пример использования элемента TITLE

Элемент META

Элемент `META` содержит метаописатели некоторых свойств документа, которые предназначены для браузеров и поисковых систем. Эти свойства могут идентифицировать авторство HTML-документа, его адрес, периодичность обновления и т.п. Поисковые системы используют данные свойства для индексации и формирования заголовков HTML-документов, они могут влиять на режим отображения HTML-документов, хотя сами на экран не выводятся. Этот элемент, вместе с элементом `TITLE` является наиболее используемым при задании заголовка.

Количество доступных свойств приближается к нескольким десяткам, но все они делятся на две группы: `NAME` и `HTTP-EQUIV`. Элементы группы `NAME` содержат текстовую информацию о документе, его авторе и некоторые рекомендации для поисковых машин (например, Robots, Description, Keywords, Author, Copyright и др.). Элементы группы `HTTP-EQUIV` влияют на формирование HTML-заголовка и определяют режим его обработки (Content-Language, Content-Type, Refresh и др.).

Следует отметить, что Спецификация HTML 4.01 не содержит стандартного списка этих свойств, поэтому авторы пока свободны в их определении. Мы рассмотрим лишь несколько свойств.

Каждый элемент `META` содержит в себе пару атрибутов: для указания названия свойства используются атрибуты `name` или `http-equiv` (в зависимости от используемого свойства), а значение этого свойства устанавливается атрибутом content. Например, следующий метаописатель задает имя автора документа:

```
<META name="Author" content="Bazil Snowman">
```

Дополнительно он может содержать атрибут `lang`, указывающий язык, на котором написано значение свойства:

```
<META name="Author" lang="en" content="Bazil Snowman">
```

Наибольший интерес из группы свойств `NAME` представляют свойства `Description` и `Keywords`. Эти свойства, наряду с элементом `TITLE`, широко используются для оптимизации и продвижения сайтов.

Свойство `Description` предназначено для краткого описания веб-страницы, которое используется поисковыми машинами для индексации и в качестве краткой аннотации, сопровождающей ссылку, в ответе на запрос поисковыми системами. По содержанию этого свойства пользователь поисковой системы будет оценивать, соответствует сайт его ожиданиям или нет. Если свойство `Description` отсутствует, то в качестве описания поисковые системы используют первую строку текста или отрывок из текста с найденным ключевым словом. Поисковые системы устанавливают разные нормы по ограничению длины описания и в зависимости от этого воспринимают только заданное количество символов. Однако оптимальная длина описания не должна превышать 150 символов.

Свойство `Keywords` позволяет определить ключевые слова веб-страницы. При формировании списка ключевых слов необходимо использовать слова, содержащиеся в тексте документа. Не стоит включать в `Keywords` служебные слова, записывать слова во множественном числе или повторять их несколько раз. Очередность

ключевых слов составляется по степени важности в порядке убывания. Оптимальная длина последовательности ключевых слов не должна превышать 250 символов.

```
<META name="Description" content="Интернет-магазин Всякая всячина – 
<META name="Keywords" content="Интернет-магазин, игрушки, книги, л
  музыка, диски, видео, DVD, двд, кино, софт, программы, игры,
   ПО, игрушки, books, video, music, software, toys">
```

Среди свойств группы `HTTP-EQUIV` наиболее важными являются свойства `Content-Type`, `Content-Style-Type` и `Content-Script-Type`.

Свойство `Content-Type` отвечает за указание типа документа и кодировки символов. Данный метаописатель устанавливает в качестве кодировки HTML-страницы Кириллицу (Windows):

```
<META http-equiv="Content-Type" content="text/html;
      charset=windows-1251">
```

Свойство `Content-Style-Type` служит для указания языка таблицы стилей. Если язык таблиц стилей не задан, по умолчанию используется язык text/css. Приведенная ниже запись явно задает в качестве языка таблицы стилей CSS:

```
<META http-equiv="Content-Style-Type" content="text/css">
```

Свойство `Content-Script-Type` определяет язык программирования сценариев. Возможны несколько значений данного параметра, однако по умолчанию используется JavaScript. Язык программирования сценариев JavaScript можно задать с помощью следующего метаописателя:

```
<META http-equiv="Content-Script-Type"
      content="text/javascript">
```

Метаописатели могут также содержать атрибут *`scheme`*, задающий формат значения свойства и предусмотренный для использования совместно со словарями метаданных. В настоящее время этот атрибут не поддерживается обозревателями.

Элемент BASE

Элемент `BASE` используется для явного задания полного URL-адреса документа. Это бывает полезно ввиду того, что общепринятым стилем задания гипертекстовых ссылок является их относительная адресация. То есть при задании ссылки на документ указывается не полный его URL-адрес, а его месторасположение относительно текущего адреса. Элемент `BASE` как раз и задает адрес, относительно которого и будут браться относительные ссылки.

Использование элемента `BASE` позволяет поддерживать относительные ссылки в том случае, когда HTML-документ перемещен, а все остальные документы, на которые он ссылается, остались на прежнем месте. Адрес документа поменялся, однако при активизации относительной ссылки, она будет взята браузером относительно исходного адреса, прописанного в элементе `BASE`.

Основным атрибутом элемента является `href`, который задает полный URL-адрес документа. Пример иллюстрирует применение элемента `BASE` и относительных ссылок:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HEAD>
<TITLE>Пример использования элемента BASE</TITLE>
<BASE href= "http://www.somewhere.ru">
</HEAD>
<BODY>
...текст документа...
<a href= "/images/someimage.jpg">Относительная ссылка на изображение<
...текст документа...
</BODY>
</HTML>
```

В этом примере переход по относительной ссылке задается относительно URL-адреса http://www.somewhere.ru. Таким образом, заданная в этом документе ссылка в абсолютном варианте независимо от месторасположения документа будет записана как ссылка:

http://www.somewhere.ru/images/someimage.jpg - http://www.somewhere.ru/images/someimage.jpg. Если базовый адрес не задан, то все относительные ссылки интерпретируются относительно каталога, в котором находится данный HTML-документ.

Элемент `BASE` можно использовать и в заголовке, и в теле документа, причем несколько раз. Область действия элемента определяется от места его задания и до конца документа или до следующего объявления элемента `BASE`, если таковой имеется.

Элемент LINK

Элемент `LINK` задает вид взаимоотношений между содержащим его документом и другим ресурсом Сети и устанавливает между ними логическую связь. Один элемент `LINK` устанавливает связь только с одним внешним документом. Однако в HTML-документе может присутствовать несколько таких элементов.

Основными атрибутами элемента `LINK` являются `href`, указывающий URL-адрес документа, с которым задается взаимоотношение, и `rel` или *`rev`*, задающие прямую и обратную ссылку и определяющие тип ссылки, который показывает, чем документ, указанный в ссылке, является по отношению к текущему документу. Типы ссылок определены в Спецификации HTML.

Например, запись

```
<LINK rel="Copyright" href="copyright.html">
```

означает, что документ copyright.html является документом, содержащим сведения об авторском праве для текущего документа (прямая ссылка), а запись

```
<LINK rev="Chapter" href="main.html">
```

означает, что текущий документ является главой документа main.html (обратная ссылка).

На данный момент информация о взаимоотношениях документов, задаваемых элементом LINK, браузерами практически не отображается. Пользователь может увидеть эти сведения, только просмотрев HTML-код документа.

Заглавные ссылки учитываются поисковыми машинами в своей работе, поэтому рекомендуется их задавать. Например, чтобы указать поисковому роботу на расположение иноязычных версий данного документа, достаточно использовать следующее задание элемента LINK:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<TITLE>Документ на русском языке</TITLE>
<LINK title ="Этот же документ на английском языке"
type="text/html" rel="alternate"
href="http://www.somewhere.ru/english.html"
hreflang="en">
...
</HEAD>
<BODY>
...текст документа на русском языке...
</BODY>
</HTML>
```

Также важным применением элемента LINK является подключение к документу внешней таблицы стилей. В этом случае элемент LINK имеет вид:

```
<LINK rel="StyleSheet" href="style.css" type="text/css">
```

Элемент STYLE

Элемент STYLE позволяет включать в документ внутренние таблицы стилей. Заголовок документа может содержать любое количество этих

элементов. Обязательный атрибут `type` указывает на тип таблицы стилей, т.е. на язык, на котором описываются стили. Для каскадных таблиц стилей этот атрибут всегда должен иметь значение `text/css`.

Ниже представлен пример включения каскадных таблиц стилей в HTML-документ:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
  <TITLE>Пример использования элемента STYLE</TYTLE>
  <STYLE type="text/css">
    body{
      background:#000;
      color:#ccc;
      font-family: helvetica, arial, sans-serif;
    }
  </STYLE>
</HEAD>
<BODY>
<P>Информация к размышлению...</P>
</BODY>
</HTML>
```

HTML позволяет авторам создавать документы, учитывающие особенности устройств, на которых будет отображаться HTML-документ. К таким устройствам относятся принтеры, проекторы, устройства, предназначенные для людей с ограниченными физическими возможностями и др. Для того, чтобы использовать разные таблицы стилей для различных устройств, используется атрибут `media`. По умолчанию, значением данного атрибута является `screen` (т.е. предполагается, что документ будет отображаться на экране монитора).

В описанном выше примере можно переопределить используемые цвета и размер шрифта, чтобы страница лучше выглядела при печати. Для этого можно добавить другой блок `STYLE` с атрибутом `media` со значением `print`, как показано ниже:

```
<STYLE type="text/css" media="print">
  body{
    background:#fff;
    color:#000;
    font-family: helvetica, arial, sans-serif;
    font-size:300%;
  }
</STYLE>
```

Результат предыдущего примера представлен на рисунке 5.2, где применяется стиль с параметром media="screen". На рисунке 5.3 представлен результат предварительного просмотра той же страницы, однако при этом уже действует стиль для печати, заданный параметром `media="print"` .

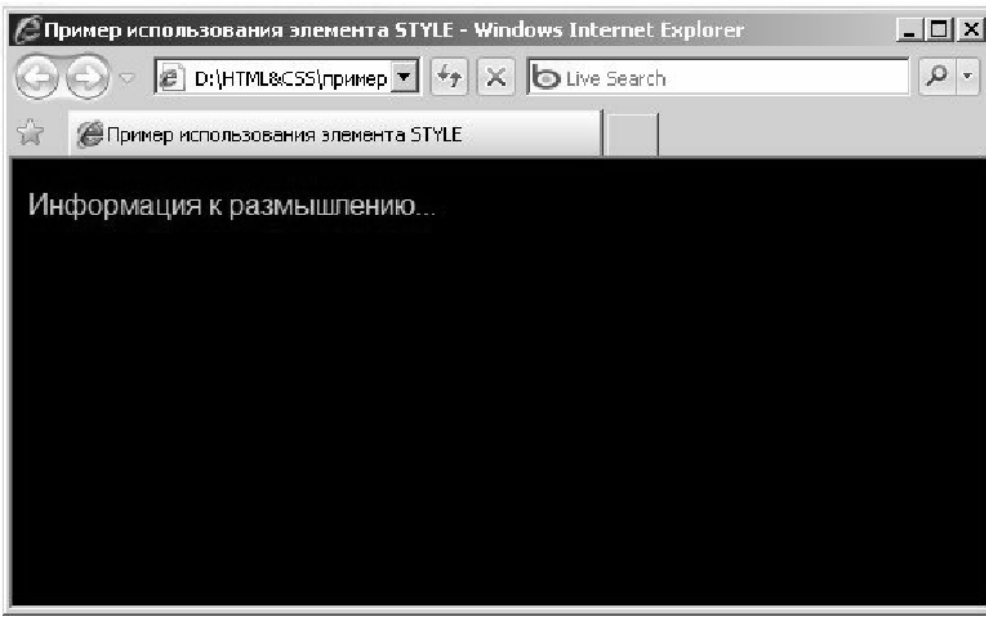


Рис. 5.2. Страница со стилем для просмотра на мониторе

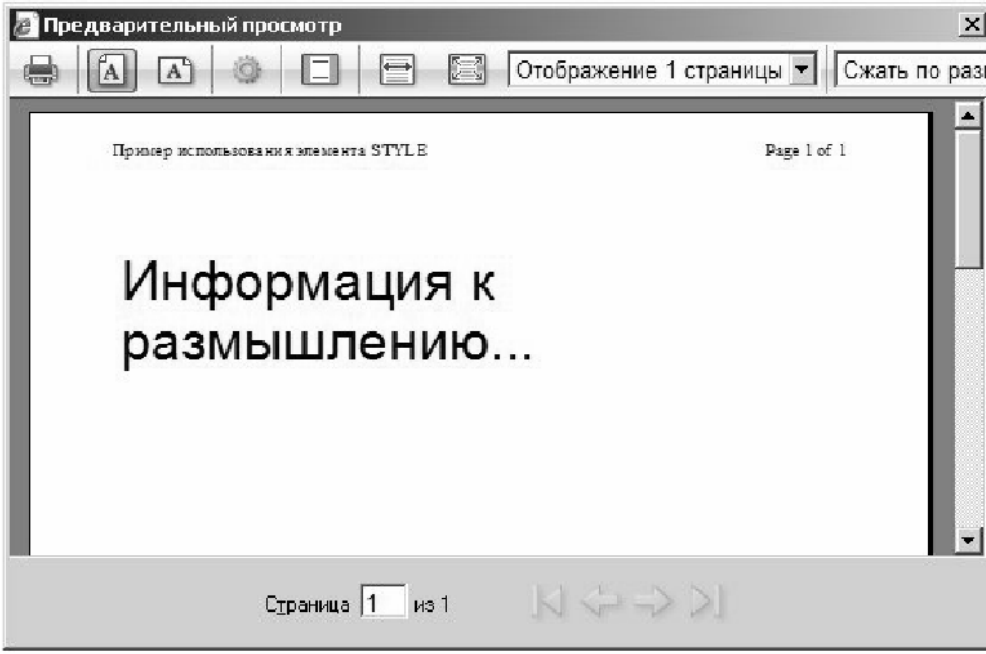


Рис. 5.3. Страница со стилем для вывода на печать

Элемент SCRIPT

Элемент `SCRIPT` предназначен для добавления так называемых клиентских сценариев - сценариев, которые выполняются браузером и написаны на языке JavaScript. JavaScript добавляет динамическое поведение в статические документы HTML, например, эффекты анимации, или проверку данных формы, или другие функции, которые запускаются, когда пользователь выполняет определенные действия.

Элемент `SCRIPT` может располагаться в заголовке или теле HTML-документа в неограниченном количестве. Когда браузер встречает такой сценарий, он прекращает разбор оставшегося документа, пока не выполнит код сценария. Поэтому, чтобы гарантировать, что код JavaScript будет доступен до загрузки основного документа, его необходимо поместить в раздел `HEAD` .

Например, можно предупредить посетителя с помощью следующего сценария, что определенная ссылка отправит его на другой сервер:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
  <TITLE>Пример использования элемента SCRIPT</TITLE>
  <SCRIPT>
    function crossing(){
      return confirm("Вы готовы перейти на другой сервер?")
    }
  </SCRIPT>
</HEAD>
<BODY>
    <A href="http://www.somewhere.com" onclick="return crossing()"> Ссылка н
</BODY>
</HTML>
```

Если открыть этот пример в браузере и щелкнуть на ссылке, то код попросит подтвердить действие пользователя, как представлено на рисунке 5.4.

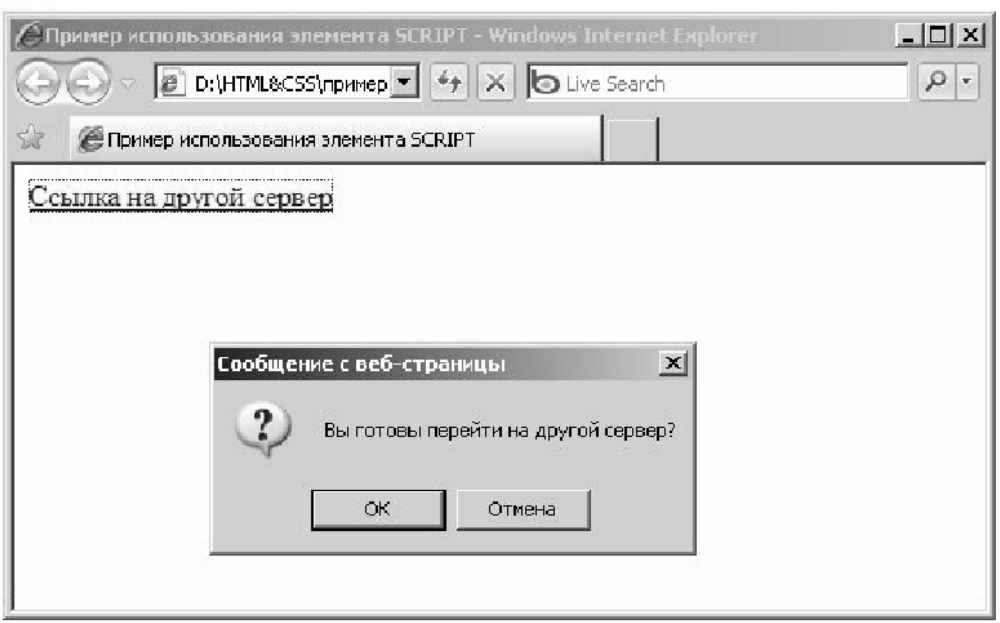


Рис. 5.4. Пример использования элемента SCRIPT

Разметка текста в HTML

В лекции рассматриваются основные блочные и строковые элементы HTML, используемые для структурирования и форматирования текста.

Структурирование текста

Как было отмечено в лекции 4, все элементы HTML делятся на блочные и строковые. Блочный элемент обычно информирует о структуре документа, начинается с новой строки и отделяется от предыдущего и последующего блока пустой строкой определенной ширины. Блочные элементы могут содержать строковые элементы. В данном разделе будут рассмотрены основные блочные элементы, предназначенные для структурирования текста.

Заголовки разделов страниц: элементы H1…H6

HTML предлагает шесть заголовков разного уровня, для задания которых используются элементы `H1`, `H2`, …, `H6`. Заголовки различных уровней призваны показать относительную важность секции, расположенной после заголовка и, по умолчанию, отображаются браузером различным размером шрифта. Так, элемент H1 представляет собой наиболее важный заголовок первого уровня, который, по умолчанию, отображается самым крупным шрифтом жирного начертания. А элемент H6 служит для обозначения заголовка шестого уровня, является наименее значительным и отображается самым мелким шрифтом. Таким образом, использование заголовков разного уровня позволяет структурировать документ по разделам, главам, параграфам и т.п., облегчая чтение, делая документ более понятным для считывателей экрана, а также для некоторых автоматических процессов. Следующий пример показывает создание заголовков различных уровней:

```
<H1>Заголовок 1-го уровня</H1>
<H2>Заголовок 2-го уровня</H2>
<H3>Заголовок 3-го уровня</H3>
<H4>Заголовок 4-го уровня</H4>
```

```
<H5>Заголовок 5-го уровня</H5>
<H6>Заголовок 6-го уровня</H6>
```

А на рисунке 6.1 представлен вид заголовков различного уровня в браузере.

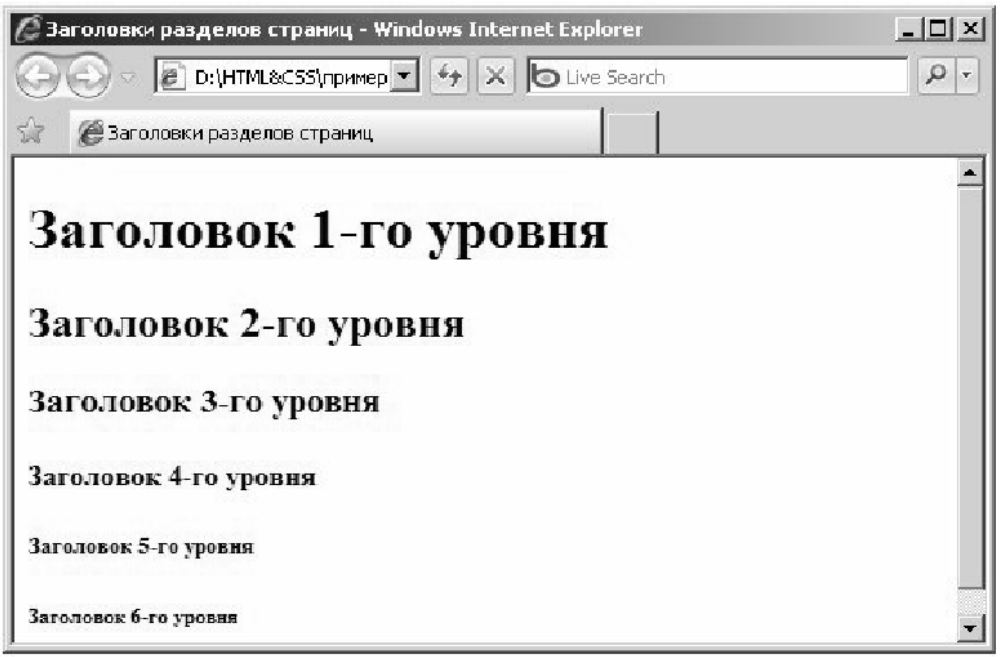


Рис. 6.1. Вид различных заголовков в браузере

Стандартные параграфы: элемент P

Неприятной особенность браузеров является то, что они не обрабатывают символы перевода строки. Т.е. текст, который разработчик набрал на своем компьютере, одновременно форматируя его, разделяя на абзацы и придавая ему какое-то логическое деление, браузер сначала преобразует, удалив все подряд идущие пробелы и переводы строк, а затем выведет на экран. При этом информация об абзацах будет утеряна. Чтобы этого не произошло, для логической организации абзацев используется специальный элемент `P`. Если закрывающего тега нет, считается, что конец параграфа совпадает с началом следующего блочного элемента. Например, оформить анекдот,

приписываемый Даниилу Хармсу, в виде параграфа можно следующим образом:

```
<P>Гоголь только под конец жизни о душе задумался,
 а смолоду у него вовсе совести не было.
  Однажды невесту в карты проиграл. И не отдал.</P>
```

Предварительно форматированный текст: элемент PRE

Элемент `PRE` определяет блок предварительно форматированного текста. По умолчанию, любое количество пробелов, идущих в коде подряд, на веб-странице показывается как один. Элемент `PRE` позволяет обойти эту особенность и отображать текст так, как требуется разработчику. В большинстве браузеров текст, помеченный как предварительно форматированный, будет выводиться с помощью моноширинного шрифта (т.е. шрифта фиксированной ширины), что придает тексту вид как бы отпечатанного на машинке. Это является наследием программистов, которые использовали ранее шрифты фиксированной ширины для представления предварительно форматированного текста. Например, представленный ниже фрагмент кода выводит отрывок из стихотворения Владимира Маяковского "Блек энд Уайт" с помощью элементов `P` и *`SAMP`*. Результат представлен на рисунке 6.2.

```
Если
    Гаванну
          окинуть мигом -
рай-страна,
     страна что надо.



<PRE>
Если
    Гаванну
          окинуть мигом -
рай-страна,
     страна что надо.
```

```
</PRE>
```

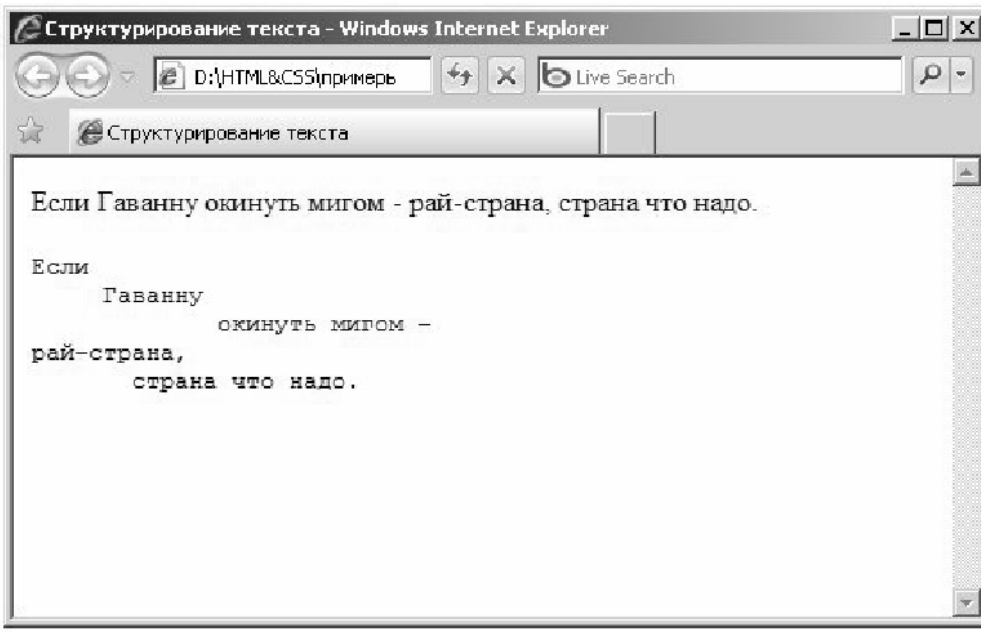


Рис. 6.2. Пример использования элементов P и SAMP

Внутри контейнера `PRE` допустимо применять любые теги, кроме тегов `IMG`, `OBJECT`, `SMALL`, `SUB` и `SUP`.

Цитирование других источников: элемент BLOCKQUOTE

Часто на веб-странице необходимо целиком или частично процитировать информацию из статей, публикаций блога, справочных документов и т.д. В HTML для выделения длинных цитат внутри документа предназначен элемент `BLOCKQUOTE`. Текст, обозначенный этим элементом, традиционно отображается как выровненный блок с отступами слева и справа. Единственный параметр данного элемента `cite` указывает полный или относительный адрес первоисточника, откуда была позаимствована приведенная цитата. Однако параметр `cite` не поддерживают как Internet Explorer, так и некоторые другие

распространенные браузеры. Рассматриваемый выше анекдот можно оформить в виде цитаты следующим образом:

```
<BLOCKQUOTE>Гоголь только под конец жизни о душе задумался,
  а смолоду у него вовсе совести не было.
   Однажды невесту в карты проиграл. И не отдал.</BLOCKQUOTE>
```

Результат применения элементов `P` и `BLOCKQUOTE` к оформлению текста представлен на рисунке 6.3.

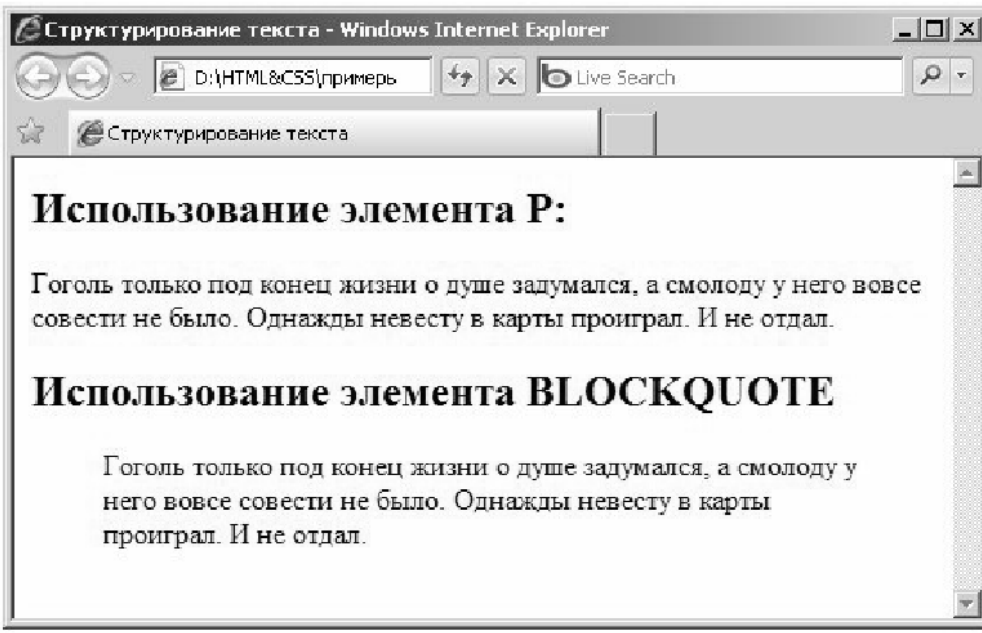


Рис. 6.3. Пример использования элементов P и BLOCKQUOTE

Горизонтальные линии: элемент HR

Горизонтальная линия создается в HTML с помощью элемента `HR`. Вид линии в основном зависит от браузера. Элемент `HR` имеет несколько параметров, управляющих выравниванием, цветом, длиной и шириной линии. Однако все эти параметры не рекомендуются Спецификацией HTML. Поэтому, если обычного представления будет недостаточно, то вид линии должен оформляться с помощью CSS.

Разрыв строки: элемент BR

В силу специфики определения разделителей в HTML невозможно управлять разрывом строк текста простым нажатием клавиши ENTER во время записи текста. Разрыв строки можно ввести в документ с помощью элемента `BR`. Однако он должен применяться только для принудительного разрыва строки, а не для увеличения вертикального интервала между элементами в документе. Для этих целей необходимо использовать свойства CSS.

Форматирование текста

Для форматирования текста HTML-документа предусмотрена целая группа элементов, которую условно можно разделить на элементы логического и физического форматирования.

Элементы логического форматирования своими именами обозначают структурные типы своих текстовых фрагментов, такие, например, как цитата (элемент `CITE`), аббревиатура (элемент *`ABBR`*) и др. Логическая разметка не влияет на конкретное экранное представление фрагмента текста браузером. Фрагменты с логическим форматированием браузеры отображают на экране определенным образом, заданным по умолчанию. Вид отображения никак не связан со структурным типом фрагмента (т.е. именем элемента логического форматирования), но может быть легко переопределен.

Элементы физического форматирования определяют формат отображения указанного в них фрагмента текста в окне браузера. Например, для отображения фрагмента курсивом можно использовать элемент `I`.

Между разработчиками документов HTML долгое время шли споры о преимуществах и недостатках того или иного подхода. С выходом спецификации HTML 4.0 эти споры завершились в пользу применения логического форматирования, поскольку был провозглашен принцип отделения структуры документа от его представления (о котором рассказывалось в лекции 4). Действительно, только на базе логического форматирования можно гибко управлять представлением документа,

используя современные методы, основанные, например, на таблицах стилей.

Тем не менее, на настоящий момент может свободно использоваться и физическое форматирование. В спецификации HTML 4.01 некоторые элементы физического форматирования не рекомендуются для применения, однако пока они все еще поддерживаются всеми браузерами. Некоторые элементы логического форматирования, призванные заменить отдельные элементы физического форматирования, распознаются не всеми браузерами, что делает их применение неудобным. Примером может служить логический элемент `DEL`, который рекомендуется использовать вместо физического элемента `STRIKE`.

Элементы логического форматирования

Аббревиатуры: элементы ABBR и ACRONYM

Элемент *`ABBR`* указывает, что последовательность символов, заключенная между его начальным и конечным тегом, является аббревиатурой. По умолчанию, такой текст подчеркивается пунктирной линией. Браузер Internet Explorer до 7 версии включительно не поддерживает элемент *`ABBR`*, рекомендуя использовать элемент *`ACRONYM`*.

Элемент *`ACRONYM`*, являясь разновидностью аббревиатуры, указывает на то, что текст является акронимом. В отличие от аббревиатуры, акроним - это устоявшееся сокращение, которое применяется как самостоятельное слово. К акронимам, например, можно отнести слова ликбез, соцстрах и т.д.

Для указания полной формы записи аббревиатуры или акронима удобно использовать атрибут title. Тогда визуальные браузеры при наведении курсора на текст, размеченный элементом *`ACRONYM`* или *`ABBR`*, будут выдавать полное наименование в виде появляющейся подсказки. Кроме того, поисковые системы индексируют полнотекстовый вариант сокращения, что может использоваться для повышения рейтинга

документа. Ниже представлен пример использования элемента ACRONYM. Результат выполнения данного фрагмента представлен на рисунке 6.4.

```
<P>Действующим президентом <ACRONYM title="Соединенные Штат
```

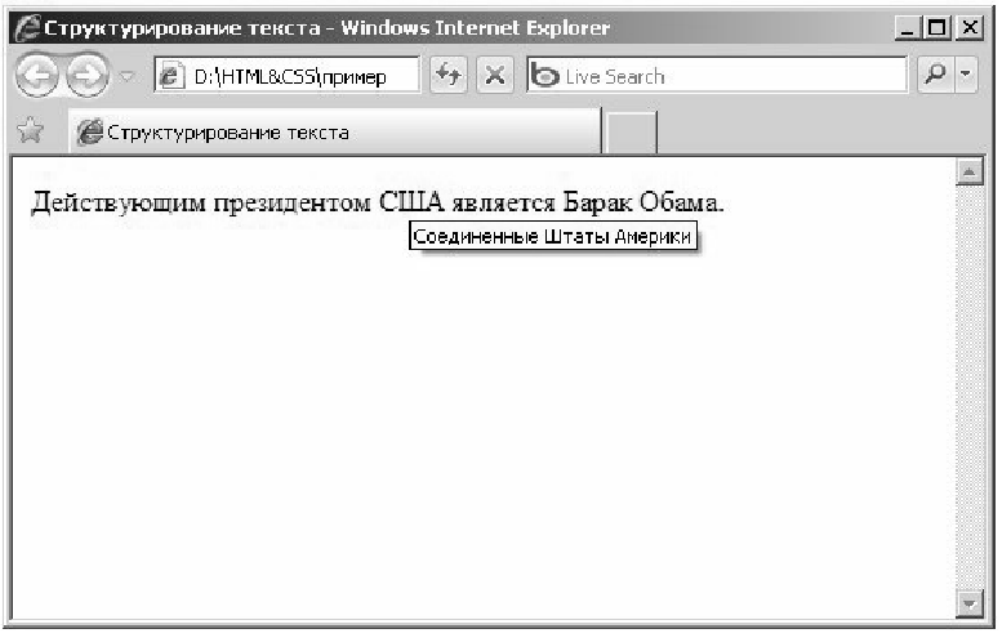


Рис. 6.4. Пример использования элемента ACRONYM

Выделение контактной информации: элемент ADDRESS

Элемент `ADDRESS` предназначен для хранения информации об авторе и может включать в себя любые элементы HTML вроде ссылок, текста, выделений и т.д. Планируется, что поисковые системы будут анализировать содержимое этого тега для сбора информации об авторах сайтов. По умолчанию, текст внутри контейнера `ADDRESS` отображается курсивным начертанием. Информация об авторе может быть задана следующим образом:

```
<ADDRESS>Сделано в СССР.</ADDRESS>
```

Цитаты: элементы CITE и Q

Элемент `CITE` используется для отметки цитат или названий книг и статей, ссылок на другие источники и т.д. Браузерами такой текст обычно выводится курсивом. Если необходимо отметить короткие цитаты в строке текста, то рекомендуется использовать элемент `Q`. В отличие от блочного элемента `BLOCKQUOTE`, при отображении не выполняется отделение размеченного текста пустыми строками. Например, цитата Оскара Уайльда может быть оформлена следующим образом:

```
<CITE>Нет греха, кроме глупости.</CITE>
```

Изменения в документах: элементы DEL и INS

Изменения, возникшие в документе с момента, когда он стал доступен в первый раз, можно пометить, чтобы посетители могли сразу определить, что и когда изменилось.

Для выделения текста, который был удален в новой версии документа, предназначен элемент `DEL`. Подобное форматирование позволяет отследить изменения, сделанные в тексте документа. Текст, помеченный элементом `DEL`, браузеры обычно отображают как перечеркнутый. В спецификации HTML 4.01 этому элементу отдается предпочтение перед элементами физического форматирования `STRIKE` или `S`, обозначающих перечеркнутый текст.

Для выделения текста, который был добавлен в новую версию, используется элемент `INS`. Его полезно использовать для отметки изменений, вносимых в документ от версии к версии. Текст, помеченный элементом `INS`, обычно отображается подчеркнутым текстом.

Элементы `DEL` и `INS` имеют два необязательных атрибута: `cite` и `datetime`. Значение атрибута `cite` должно представлять собой URL-адрес документа, поясняющего причины удаления данного фрагмента или подробности внесенных дополнений, а атрибут `datetime`

указывает дату удаления или вставки.

Программный код: элемент CODE

Элемент `CODE` предназначен для отображения одной или нескольких строк текста, который представляет собой программный код: имена переменных, ключевые слова, тексты функции и т.д. Браузеры обычно отображают содержимое контейнера `CODE` моноширинным текстом уменьшенного размера. Этот элемент не следует путать с элементом `PRE`, являющимся элементом блочного уровня, который следует использовать для отметки больших фрагментов кода. Кроме того, в отличие от элемента `PRE`, дополнительные пробелы внутри контейнера `CODE` не учитываются так же, как и переносы текста.

Определения: элемент DFN

Как правило, новый термин, упоминающийся в документе, выделяется курсивом, также дается его определение. Тогда при использовании этого термина в дальнейшем, он считается уже известным читателю. Для выделения таких терминов при их первом появлении в тексте используется элемент `DFN`. Браузеры, по умолчанию, отображают содержимое контейнера `DFN` с помощью курсивного начертания.

Вывод взаимодействия с компьютером: элементы SAMP и KBD

Элемент *`SAMP`* используется для отображения текста, который является результатом вывода компьютерной программы или скрипта. Элемент *`SAMP`* используется также для выделения нескольких символов моноширинным шрифтом. Применение данного элемента предпочтительнее применения элемента физического форматирования `TT`.

Элемент `KBD` используется для обозначения текста, который набирается

на клавиатуре, или для названия клавиш. Браузеры, по умолчанию, помечают текст в контейнере KBD и *SAMP* моноширинным шрифтом.

Выделение важных фрагментов текста: элементы STRONG и EM

Для выделения важных фрагментов текста, как правило, используются элементы EM и STRONG. Браузеры обычно отображают текст, отмеченный элементом EM и STRONG, курсивом и полужирным шрифтом соответственно. Элементом STRONG обычно размечают более важные фрагменты текста, чем те, что размечены элементом EM.

Следует отметить, что элементы I и EM, так же как B и STRONG, несмотря на сходство результата, являются не совсем эквивалентными и взаимозаменяемыми. Например, элемент I является элементом физической разметки и устанавливает курсивный текст, а элемент EM - элементом логической разметки и определяет важность помеченного текста. Такое разделение тегов на логическое и физическое форматирование изначально должно было сделать HTML универсальным, в том числе не зависящим от устройства вывода информации. Теоретически, если воспользоваться, например, речевым браузером, то текст, оформленный с помощью тегов I и EM, будет отмечен по-разному. Однако получилось так, что в популярных браузерах результат использования этих тегов равнозначен.

Переменные: элемент VAR

Элемент VAR используется для указания переменных в текстовом контенте. Он может включать алгебраические математические выражения или находиться в программном коде. Браузеры обычно помечают такой текст курсивом. Например, элемент VAR можно использовать следующим образом:

```
<VAR>x+y=z</VAR>
```

Элементы физического форматирования

В данном разделе приведено краткое описание некоторых элементов физического форматирования. Часть из них не рекомендуется к использованию спецификацией HTML 4.01 и заменена более новыми методами достижения аналогичного результата, однако они продолжают поддерживаться браузерами и поэтому представляют определенный интерес.

Элементы стиля шрифта: TT, I, B, BIG, SMALL, U, STRIKE и S

Элемент B отображает текст полужирным шрифтом. В большинстве случаев рекомендуется вместо него использовать элементы логического форматирования `STRONG` или `EM`.

Элемент I отображает текст курсивом. Для большинства случаев вместо данного элемента рекомендуется использовать элементы `EM`, `DFN` или `CITE`, поскольку последние лучше отражают назначение выделяемого текста.

Элемент `TT` отображает текст моноширинным шрифтом. Для большинства случаев вместо этого элемента лучше использовать элементы `CODE`, *`SAMP`* или `KBD`.

Элемент U отображает текст подчеркнутым. Данный элемент отмечен как не рекомендуемый, вместо него предлагается использовать элементы `STRONG` или `CITE`.

Элементы `STRIKE` и `S` отображают текст, перечеркнутый горизонтальной линией. Оба элемента отменены, и вместо них рекомендуется использовать элемент DEL.

Элементы `BIG` и `SMALL` выводят текст шрифтом большего и меньшего (чем непомеченная часть текста) размера. Вместо элемента `BIG` предпочтительнее использовать элемент `STRONG`, а размер устанавливать с помощью соответствующих свойств CSS.

Верхние и нижние индексы: элементы SUB и SUP

Чтобы пометить часть некоторого текста как верхний или нижний индекс, используют элементы `SUB` и `SUP`. Элемент `SUB` сдвигает текст ниже уровня строки и выводит его шрифтом меньшего размера. Элемент SUP сдвигает текст выше уровня строки и выводит его шрифтом меньшего размера. Эти элементы удобно применять для разметки несложных математических выражений, не обращаясь к использованию специального языка математической разметки MathML. Например, формула воды может быть записана следующим образом:

```
<P>H<SUB>2</SUB>O</P>
```

Элементы модификатора шрифта: FONT и BASEFONT

Элемент `FONT` позволяет задать такие параметры шрифта, как тип (или семейство шрифтов), размер и цвет. Элемент `BASEFONT` применяется для задания базового шрифта сразу для всей страницы. Обычно он указывается сразу после элемента `BODY`. В Спецификации HTML 4.01 эти элементы отнесены к отмененным, и их дальнейшее применение не рекомендуется.

Базовые контейнеры: элементы DIV и SPAN

Большинство элементов в HTML существует для описания контента, такого, как изображения, списки, заголовки, или помогают в настройке документа (`HEAD`, `LINK` и т.д.). Однако имеется два элемента, которые не имеют заданного значения. Спецификация HTML декларирует, что элементы DIV и SPAN совместно с атрибутами id и class образуют базовый механизм для добавления в документы структуры. Эти два элемента можно считать некоторой несущей опорой HTML.

Элемент `DIV` является базовым контейнером блочного уровня. Он используется для объединения различных частей контента в логически цельную единицу или логический блок. Этот элемент, эффективно взаимодействуя с таблицами стилей, позволяет форматировать разделы отдельной страницы или даже целого сайта. Текстовый блок,

отделенный логически, легко потом выделить при отображении любым способом, например, шрифтом, цветом, межстрочным интервалом, центрированием и т.п. Содержимое элемента DIV может быть расположено произвольным образом, выделено другим фоном и т.д. Как правило, вид блока управляется с помощью стилей. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить параметр class или id с именем селектора.

Аналогом `DIV` на уровне текста является элемент `SPAN`. Элемент `SPAN` является базовым контейнером строкового уровня. Он также помогает представить структуру документа, но используется для объединения в группу или создания оболочки вокруг других строковых элементов или текста.

Различие между двумя этими элементами состоит в типе контента, а также в его представлении при записи без какого-либо стилевого оформления. Элемент `DIV` помещается вокруг группы элементов блочного уровня (например, создать оболочку вокруг заголовка и списка ссылок, чтобы создать навигационное меню). Элемент SPAN создает оболочку вокруг группы строковых элементов или (чаще всего) обычного текста.

Следующий фрагмент кода демонстрирует применение элементов `DIV` и `SPAN`:

```
<BODY>
  <DIV id="Content">
    <H1>Заголовок страницы </H1>
    <P>Это первый параграф.</P>
    <IMG src="image.gif" alt="Это какое-то изображение">
    <P>Это второй параграф, который содержит <SPAN id="Special"> эле
  </DIV>
</BODY>
```

Теперь можно выделить содержимое элементов `DIV` и `SPAN` с помощью их атрибутов `id` и применить к ним специальное стилевое оформление и позиционирование с помощью свойств CSS.

Списки и изображения в HTML

В лекции рассматриваются особенности работы со списками и изображениями в HTML.

Списки в HTML

Списком называется взаимосвязанный набор данных, которые начинаются с маркера или цифры. Списки используются для систематизации разных данных и представления их в наглядном и удобном для пользователя виде.

В HTML используется три типа списков:

- маркированный (или неупорядоченный) список,
- нумерованный (или упорядоченный) список,
- список определений.

Маркированный список используется для объединения в группу множества связанных объектов, причем порядок размещения данных объектов не важен. Представленный выше список является маркированным.

Нумерованный (или упорядоченный) список используется для вывода списка объектов, которые необходимо разместить в определенном порядке. Нумерованный список выглядит аналогично маркированному, но его элементы нумеруются:

1. Нарезать картошку.
2. Посолить.
3. Поставить аквариум на огонь.

Список определений используется для вывода пар термин-определение, объединяя определенные объекты и их определения в списке. Например, следующий список определений используется для связи расширений файла и программ, с помощью которых данные файлы создаются:

*.doc
файлы, созданные Microsoft Word
*.xls
файлы, созданные Microsoft Excel
*.ppt
файлы, созданные Microsoft PowerPoint

Списки могут вкладываться друг в друга, причем допускается вложение списков одного типа в списки другого типа. Следующий рецепт приготовления ухи демонстрирует список определений, в который вложены маркированный и нумерованный списки:

Ингредиенты:

- рыбка,
- картошка,
- соль по вкусу.

Способ приготовления:

1. Нарезать картошку
2. Посолить.
3. Поставить аквариум на огонь.

Маркированный список

Маркированный список определяется с помощью элемента `UL`. Каждый элемент списка должен начинаться с элемента `LI` и быть вложенным в элемент `UL`. Если к элементу `UL` применяется свойства CSS, то элементы `LI` наследуют эти свойства. Приведенный выше маркированный список выглядит на языке HTML следующим образом:

```
<UL>
  <LI>маркированный (или неупорядоченный) список,</LI>
  <LI>нумерованный (или упорядоченный) список,</LI>
  <LI>список определений.</LI>
</UL>
```

Маркированные списки могут выводиться с использованием одного из трех видов маркеров: закрашенного кружка (по умолчанию), не закрашенного кружка и закрашенного квадрата. Вид маркера можно изменить с помощью атрибута `type`, установив его значение `disk`, `circle` и `square` соответственно. Однако этот атрибут не рекомендуется Спецификацией HTML 4.01. Изменить маркер на один из нескольких используемых по умолчанию стилей, использовать свое собственное изображение или даже вывести список без маркеров можно с помощью соответствующих свойств CSS *`list-style`* или *`list-style-type`*.

Нумерованный список

Нумерованный список устанавливается с помощью элемента `OL`. Каждый элемент списка должен начинаться с элемента `LI`, как и в случае маркированного списка. Если к элементу `OL` применяется свойства CSS, то элементы `LI` наследуют эти свойства. Приведенный выше нумерованный список выглядит на языке HTML следующим образом:

```
<OL>
  <LI>Нарезать картошку.</LI>
  <LI>Посолить.</LI>
  <LI>Поставить аквариум на огонь.</LI>
</OL>
```

Нумерованные списки могут выводиться с помощью одной из нескольких цифровых или алфавитных систем. По умолчанию, в большинстве браузеров используются десятичные числа, но имеется большее количество возможностей, например, заглавные и строчные латинские буквы, заглавные и строчные римские цифры и многое другое. Выбор маркера и в данном случае осуществляется с помощью параметра `type` элемента `OL`. Однако, как было отмечено выше, данный параметр является не рекомендуемым, а для осуществления аналогичной функциональности рекомендуется использовать свойства CSS *`list-style`* или *`list-style-type`*.

Начинать список допускается с любого номера. Для этой цели

применяется атрибут `start` элемента `OL` или атрибут `value` элемента LI. В качестве значения атрибутов `start` и `value` задается любое целое положительное число. При этом неважно, какой тип нумерации установлен, даже если в качестве списка используются латинские буквы. Если одновременно для списка применяются атрибуты `start` и `value`, то последний имеет преимущество, и нумерация отображается с числа (или символа), указанного аргументом `value`. Следуем отметить, что эти атрибуты являются фактически исключенными в Спецификации HTML 4.01. Это может показаться странным, так как эти атрибуты имеют смысл и не имеют эквивалента в CSS. Поэтому, использовать данные атрибуты или нет, решает только разработчик.

Список определений

Для создания списка определений используются три элемента: `DL`, `DT` и `DD`. Каждый такой список начинается с контейнера `DL`, куда входит элемент `DT`, создающий термин, и элемент DD, задающий определение этого термина. Закрывающий тег `</DT>` не обязателен, поскольку следующий тег сообщает о завершении предыдущего элемента. Тем не менее, хорошим стилем является закрывать все теги. Приведенный выше список определений может быть реализован следующим образом:

```
<DL>
  <DT>*.doc</DT>
  <DD>файлы, созданные Microsoft Word</DD>
  <DT>*.xls</DT>
  <DD>файлы, созданные Microsoft Excel</DD>
  <DT>*.ppt</DT>
  <DD>файлы, созданные Microsoft PowerPoint</DD>
</DL>
```

Стандарт не исключает возможности давать несколько определений одного термина или одно определение нескольким терминам, как показано в следующих примерах:

```
кофе
        напиток, приготовленный из кофейных зерен
        один из оттенков цвета коричневого цвета
```

Кока-Кола
Фанта
Колокольчик
: Сладкий, насыщенный углекислым газом напиток

Обычно не принято связывать несколько терминов с одним определением, но полезно знать, что это возможно, если возникнет такая необходимость.

Изображения в HTML

Изображения придают сайту уникальный вид, делают его более выразительным и привлекательным, а разработчику сайта позволяют расширить арсенал средств для управления дизайном. Определенную информацию, вроде графиков и схем, проще и нагляднее передать через рисунки, чем долго описывать, что к чему. Так что правильное использование изображений на сайте только улучшит его.

Форматы графических файлов

Большинство браузеров поддерживает только три графических формата: GIF, JPEG и PNG. Наиболее широкое распространение получили GIF и JPEG. Долгое время они являлись практически стандартами веб-изображений. Формат PNG появился достаточно недавно, и его популярность значительно уступает форматам GIF и JPEG. Распространение формата PNG сдерживается старыми версиями браузеров, а также недостаточной и неполной поддержкой возможностей PNG в новых версиях.

Каждый из этих трех форматов имеет свои преимущества и недостатки, так что вопрос выбора оптимального формата должен определяться в каждом конкретном случае индивидуально. Рассмотрим основные особенности данных графических форматов.

Формат GIF (Graphics Interchange Format, формат обмена графическими данными) – это растровый формат, использующий для хранения информации о цвете только 8 битов. Таким образом, цветовой диапазон

ограничен 256 цветами, а для уменьшения размера графических файлов возможно сократить количество используемых цветов до 2. Формат GIF поддерживает один уровень прозрачности, однако поддерживает анимацию, что делает его популярным для создания баннеров и простой анимации.

Еще одно преимущество формата GIF заключается в чересстрочной развертке. При включении этой опции изображение будет постепенно увеличивать четкость по мере его загрузки. Таким образом, изображение появляется на экране почти сразу после начала загрузки страницы и, не дожидаясь полной загрузки, можно понять, что представлено на картинке.

Формат GIF целесообразно использовать в том случае, если цветовой диапазон исходных изображений не превышает 256 цветов, либо количество цветов может быть уменьшено без существенного ухудшения качества. Это, как правило, логотипы, иллюстрации с четкими краями, анимированные рисунки, изображения с большими площадями одноцветных областей, баннеры. Для полноцветных изображений, в том числе для фотографий, формат GIF мало применим. В этом случае лучше использовать другие форматы сжатия.

Формат JPEG (Joint Photographic Experts Group - объединенная группа экспертов в области фотографии) поддерживает 24-битные цвета (т.е. около 16 миллионов цветов) и сохраняет яркость и оттенки цветов изображений. Формат предполагает сжатие с потерями: JPEG-сжатие основано на разложении изображений на составляющие, близкие к тем, которые используются в человеческом зрении при отбрасывании информации, не сказывающейся на зрительном восприятии образа. За счет этого достигается высокое сжатие изображений при незначительном ухудшении качества. Степень сжатия и качество изображений находятся в обратной зависимости: чем сильнее сжато изображение, тем ниже его качество (и, соответственно, размер графического файла). В отличие от форматов GIF и PNG формат JPEG не поддерживает прозрачность: при сохранении изображения в данном формате, прозрачные пикселы заполняются определенным цветом. Формат JPEG поддерживает технологию, известную под названием прогрессивный JPEG, при которой версия рисунка с низким разрешением появляется в окне просмотра до полной загрузки самого

изображения.

JPEG хорошо подходит для изображений с богатой цветовой гаммой, плавным переходом цветов, для фотографий и изображений с градиентными областями. В силу особенностей алгоритма сжатия JPEG лучше не использовать для сжатия изображений, цветовая гамма которых ограничена несколькими цветами, изображений с мелким текстом, изображений, которые должны сохранить четкие границы или содержат мелкие детали.

Формат PNG (Portable Network Graphics - переносимая сетевая графика), призванный заменить формат GIF, является относительно новым. Формат PNG существует в двух вариантах: PNG-8 и PNG-24.

PNG-8 практически полностью аналогичен формату GIF, за исключением улучшенного сжатия и отсутствия возможности анимации. PNG-8 хорошо подходит для текста, логотипов, иллюстраций с четкими краями и изображений с градиентной прозрачностью.

PNG-24 аналогичен PNG-8, но использует 24-битную палитру цвета и, подобно формату JPEG, сохраняет яркость и оттенки цветов в фотографиях. А подобно GIF и формату PNG-8, сохраняет детали изображения как, например, в логотипах или иллюстрациях. Формат PNG-24 обладает рядом дополнительных преимуществ, таких как применение улучшенного сжатия, наличие альфа-прозрачности и гамма-коррекции и др.

PNG-24 рекомендуется выбирать для полноцветных изображений с четкими краями и мелкими деталями, изображений с мелким текстом, а также для изображений с прозрачными областями. При использовании формата PNG-24 для сжатия полноцветных изображений, он проигрывает формату JPEG в размере созданного файла, т.к. использует сжатие без потерь.

Включение изображений в HTML

Для вставки изображения в HTML-документ используется элемент IMG, имеющий два обязательных атрибута src и alt.

Атрибут `alt` содержит так называемый альтернативный текст, который будет отображаться, если по каким-либо причинам изображение недоступно. Кроме того, люди с недостатками зрения используют вспомогательные технологии для чтения веб-страниц. Эти технологии считывают содержимое атрибутов `alt` элементов IMG. Поэтому важно написать хороший альтернативный текст для описания содержимого изображения и поместить его в атрибут `alt`. Такой текст позволяет также получить текстовую информацию о рисунке при отключенной в браузере загрузке изображений. Поскольку загрузка изображения происходит после получения браузером информации о нем, то замещающий рисунок текст появляется раньше. По мере же загрузки текст будет сменяться изображением. Браузеры также отображают альтернативный текст в виде подсказки, появляющейся при наведении курсора мыши на изображение. В качестве значения параметра `alt` служит любая подходящая текстовая строка. Ее обязательно надо брать в двойные или одинарные кавычки. Этот атрибут крайне полезен как для доступности, так и для оптимизации поисковых машин.

Атрибут src задает адрес графического файла, который будет отображаться на веб-странице. В качестве значения принимается абсолютный или относительный адрес файла. К абсолютному адресу относится полный путь к ресурсу, включая протокол передачи данных, наименование сервера, а также имена всех каталогов, ведущих к файлу, например http://www.somewhere.com/images/image.jpg. Относительные адреса описывают местоположение файла относительно текущего каталога (например, "../../images/image.jpg") или корня каталогов сервера (например, "/images/image.jpg"). Если требуемый графический файл находится в том же каталоге, что и HTML-документ, его использующий, то в качестве значения аргумента src допустимо указать просто имя требуемого графического файла.

Фрагмент кода, использующий описанные аргументы, представлен ниже:

```
<IMG src="http://www.somewhere.com/images/beatles.jpg"
  alt="Группа Beatles: Пол, Ринго, Джордж и Джон">
```

Результат выполнения данного кода представлен на рисунке 7.

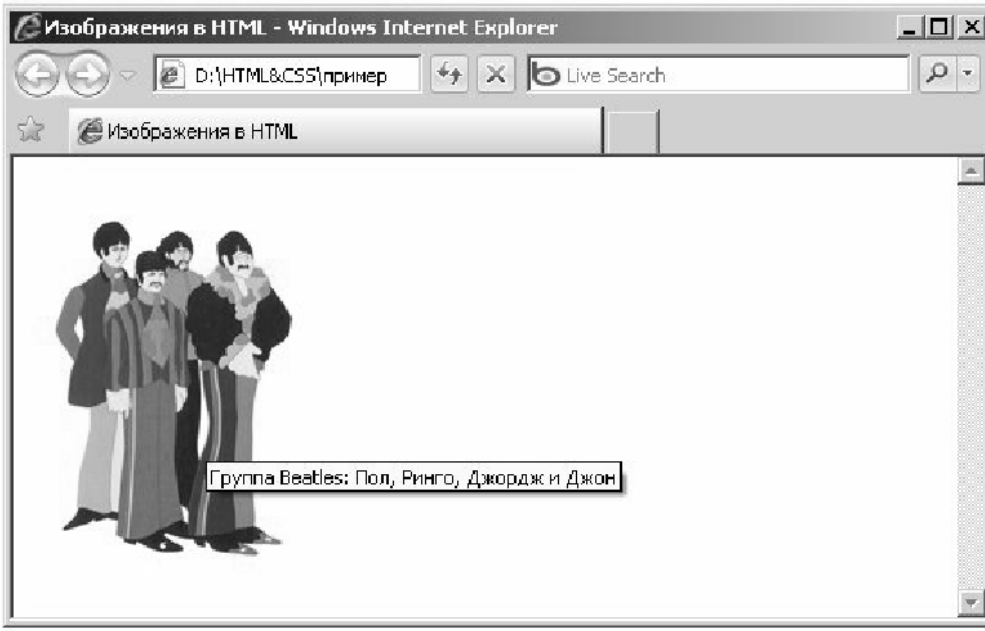


Рис. 7.1. Вид изображения в браузере и всплывающая подсказка

Кратко рассмотрим некоторые другие атрибуты элемента IMG, отвечающие за отображение изображений и их характеристики. С атрибутами, не вошедшими в лекцию, можно самостоятельно ознакомиться в Спецификации HTML 4.01.

Описание изображения: атрибут longdesc

Если изображение является очень сложным, как, например, график, можно предложить более длинное его описание, используя атрибут `longdesc`, чтобы люди, использующие считыватель экрана или при отключенном выводе изображений, могли, тем не менее, получить доступ к информации, содержащейся в изображении. Этот атрибут содержит URL, который указывает на документ, содержащий эту же самую информацию. Например, если имеется график, показывающий множество данных, можно соединить его с таблицей данных с той же информацией с помощью `longdesc`. Атрибут longdesc может быть задан следующим образом:

```
<img src="diagramma.jpg" alt="Диаграмма использования поисковых сист
```

Размеры изображения: атрибуты height и width

Атрибуты `height` и `width` дают указание браузеру выделить необходимое пространство для изображений до их загрузки. В этом случае выводится свободное пространство с заданными размерами, пока изображение не загрузится и не займет свое место. Кроме того, данные атрибуты используются для изменения размеров изображения средствами HTML. Ширину и высоту изображения можно менять как в меньшую, так и в большую сторону. Допускается использовать значения в пикселах или процентах относительно родительского элемента (контейнера, в который помещен элемент IMG), или окна браузера. Добавление только одного атрибута `width` или `height` сохраняет пропорции изображения. Браузер при этом ожидает полной загрузки рисунка, чтобы определить его первоначальную высоту и ширину. Задание размеров изображения несколько ускоряет загрузку страницы, поскольку браузеру нет нужды вычислять размер каждого рисунка после его получения.

В следующем примере ширина и высота изображения задаются равными 300 и 200 пикселов соответственно.

```
<IMG src="http://www.somewhere.com/images/image.jpg"
  alt="Какая-то фотография" width="300" height="200">
```

Карты-изображения: атрибуты ismap и usemap

Атрибут `ismap` сигнализирует браузеру о том, что рисунок является серверной картой-изображением. Карты-изображения позволяют привязывать ссылки к разным областям одного изображения. Параметр реализуется в двух различных вариантах - серверном и клиентском. В случае применения серверного варианта браузер посылает запрос на сервер для получения адреса выбранной ссылки и ждет ответа с нужной информацией. Такой подход требует дополнительного времени на ожидание результата и отдельные файлы для каждой карты-изображения. Кроме того, координаты имеют смысл только в

графической среде, поэтому ссылки, оформленные таким образом, доступны только пользователям графических браузеров, что отрицательно сказывается на доступности сайта.

Атрибут `usemap` указывает на клиентскую карту ссылок, определенную элементом `MAP`. Его значением должно быть имя закладки, заданное атрибутом name соответствующего элемента `MAP`. Применение данного атрибута будет более подробно рассмотрено в следующей лекции.

Практическое занятие 1

Целью практического занятия является закрепление материала лекций 6 и 7. В ходе практического занятия формируются навыки структурирования и форматирования текста на страницах HTML, оформления информации в виде списков, работы с графическими изображениями.

Слушателю предлагается создать HTML-документ (например, с именем page1.html) и оформить фрагмент текста, представленный в Приложении 1, в виде веб-страницы. При верстке данного текста необходимо максимально придерживаться представленной структуры и оформления. Необходимо обратить особое внимание на элементы HTML, предназначенные для создания заголовков различного уровня, стандартных параграфов, цитирования, выделения контактной информации, выделения важных фрагментов текста.

К предложенному тексту необходимо добавить несколько уместных графических изображений (например, портрет А. Эйнштейна, автора представленной формулы), снабдив каждое из них осмысленными описаниями и определенными размерами.

Слушателю предлагается создать новый HTML-документ (например, с именем page2.html) и разместить в нем фрагмент текста, представленный в Приложении 2. При верстке необходимо обратить внимание на "ЗАКОНЫ ПРИКЛАДНОЙ ПУТАНИЦЫ", в котором используется маркированный список, вложенный в нумерованный. Слушателю также рекомендуется самостоятельно продумать и организовать список определений, с вложенными в него маркированным и нумерованным списками, как описано в лекции 7.

Ссылки в HTML

В лекции описываются основные способы работы со ссылками. Кратко рассматриваются некоторые способы реализации навигации по сайту.

Хотя HTML содержит большое количество средств для форматирования текста и структурирования документов, его основной особенностью является возможность создания гипертекстовых документов. Гипертекстовыми являются документы, которые содержат гиперссылки на другие ресурсы сети Интернет. Особенность гиперссылки заключается в том, что она может указывать не только на другой HTML-документ или любой другой ресурс (текстовые файлы, файлы PDF, изображения, звуковые файлы и т.д.), но и на определенный раздел текущего HTML-документа.

Текст гиперссылки, как правило, помечается подчеркиванием и цветом, чтобы его было легко визуально отличить от обычного текста. При наведении курсора мыши на ссылку, курсор превращается в "руку", а в статусной строке браузера отображается путь к ресурсу, на который указывает ссылка.

Любая ссылка на веб-странице может находиться в одном из следующих пяти состояний:

- непосещенная ссылка (link). Такое состояние характерно для ссылок, которые еще не открывали. По умолчанию, непосещенные ссылки имеют синий цвет и отображаются шрифтом с подчеркиванием;
- активная ссылка (active). Ссылка помечается как активная в момент ее открытия, т.е. в тот короткий промежуток времени между нажатием на ссылку и началом загрузки нового документа. В некоторых браузерах этот стиль применяется, когда ссылка открыта в другом окне или вкладке. Цвет такой ссылки по умолчанию красный;
- посещенная ссылка (visited). Как только пользователь открывает документ, на который указывает ссылка, она помечается как посещенная и меняет свой цвет на фиолетовый, установленный по умолчанию;
- фокус (focus). Ссылка находится "в фокусе", когда, например, она

выделена с помощью клавиатуры пользователя. Браузер Internet Explorer не поддерживает в настоящее время состояние `focus`, и использует вместо него состояние active;

- под указателем (hover). Состояние применяется, когда пользователь удерживает над ссылкой указатель мыши.

Следует помнить, что различные состояния ссылок в различных браузерах реализуются и оформляются по-разному.

Создание гиперссылок

Любой строковый элемент или изображение можно превратить в гиперссылку. Для этого необходимо сообщить браузеру, какой элемент является ссылкой, а также указать адрес ресурса, на который следует сделать ссылку. Оба эти действия выполняются с помощью элемента A.

Элемент `A` имеет несколько атрибутов, главным из которых являются `href`, задающий адрес ресурса, на который указывает ссылка. Адрес ссылки может быть абсолютным и относительным. Как правило, абсолютные адреса применяются для перехода на другой ресурс, а внутри текущего сервера применяются относительные адреса. Ниже представлены некоторые примеры создания ссылок:

```
<A href="../images/image.jpg">Изображение</A>
<A href="http://www.microsoft.ru">сайт Microsoft</A>
<A href="mailto:myname@gmail.com" title="Пишите письма">
myname@gmail.com</A>
```

Иногда оказывается полезным организовать ссылки не на другие HTML-документы, а на определенные места того же самого документа. Подобные ссылки еще называют закладками. Вначале следует задать в соответствующем месте HTML-документа закладку и установить ее имя с помощью атрибута name элемента `A`. Имя ссылки на закладку начинается с символа `#`, после чего идет название закладки. Можно также делать ссылку на закладку, находящуюся на другой веб-странице и даже другом сайте. Для этого в адресе ссылки надо указать ее адрес и в конце добавить символ решетки `#` и имя закладки. Между тегами `<A name=..>` и `</A>` текст писать необязательно, т.к. требуется лишь

указать местоположение перехода по ссылке.

Приведенный фрагмент кода создает закладку и ссылку на нее:

```
<A name="top"></A>
  ...
  <A href="#top">Вверх!</A>
```

Ссылкой может быть не только строковый элемент, но и изображение. Изображение в этом случае надо поместить между тегами `<A href=...>` и `</A>`, как показано в примере:

```
<A href="sample.html"><img src="sample.gif" width="50" height="50" alt="Изоб
```

Атрибут `href` элемента `A` задает путь к документу, на который указывает ссылка, а атрибут `src` элемента IMG - путь к графическому файлу, который используется в качестве ссылки.

Вокруг изображения-ссылки автоматически добавляется рамка толщиной один пиксел и цветом, совпадающим с цветом текстовых ссылок. Можно воспользоваться CSS, чтобы убрать рамку для всех изображений, которые являются ссылками. Для этого применяется свойство `border` со значением none. На рисунке 8.1 представлены текстовая и графическая ссылки.

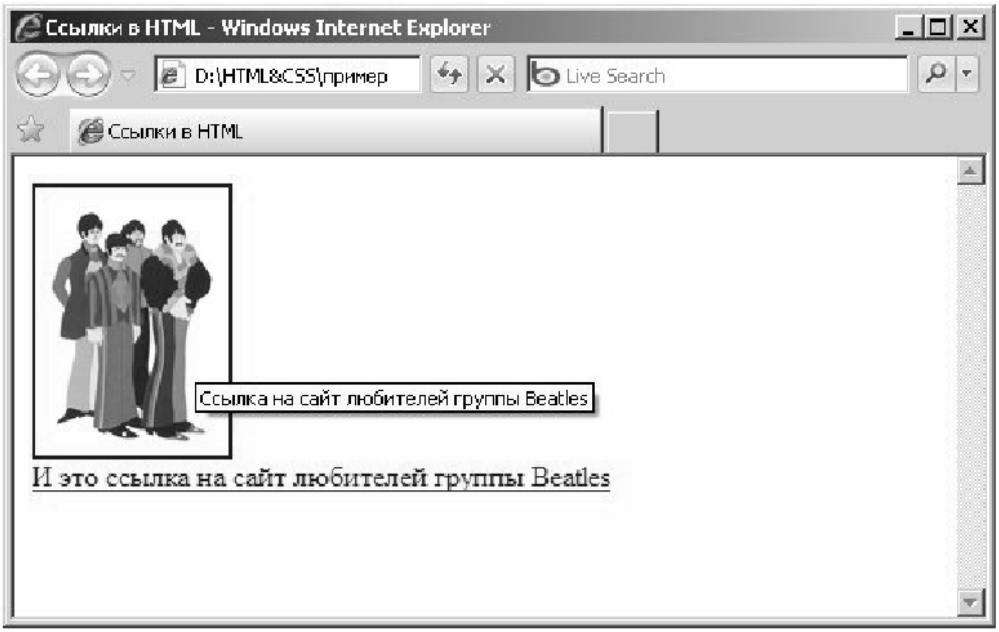


Рис. 8.1. Вид текстовой и графической ссылок в браузере

Как и большинство элементов HTML, элемент A может иметь атрибут title, который представляет некоторую дополнительную информацию. Браузеры показывают так называемую всплывающую подсказку, когда посетители проводят курсором своей мыши над ссылкой. Эта всплывающая подсказка сообщает информацию о ссылке. Например, можно дать небольшое введение в содержание и расположение присоединенного по ссылке документа.

Создание навигационного меню

Отличительной особенностью веб-сайтов является нелинейность их содержимого. А для перемещения и ориентации в этом нелинейном пространстве служит навигационная система сайта. Основными инструментами для создания навигационной системы в HTML являются ссылки, закладки и списки. Чтобы показать пользователю, что закладки и ссылки выполняют функцию навигационного меню, а не являются случайным набором ссылок, их необходимо структурировать и стилистически оформить. Если порядок страниц не имеет значения, можно использовать неупорядоченный список, как показано в примере

ниже:

```
<ul id="mainmenu">
    <li><a href="home.html">Домашняя страница</a></li>
    <li><a href="projects.html">Наши проекты</a></li>
    <li><a href="services.html">Наши услуги </a></li>
    <li><a href="contact.html">Контакты</a></li>
  </ul>
```

Если порядок, в котором посетители просматривают все документы, важен, то необходимо использовать упорядоченный список. Если, например, имеется онлайновый курс, состоящий из нескольких лекций, и учащиеся должны изучать его в определенном порядке, то можно было бы использовать упорядоченный список. Использование списков для создания меню очень удобно, т.к. весь код HTML содержится в одном элементе списка, что позволяет использовать CSS для различного оформления каждого элемента. Кроме того, списки могут быть вложенными, что позволяет легко создавать несколько уровней иерархии навигации. Даже без какого-либо стилевого оформления списка, представление в браузере кода HTML имеет смысл, и посетителю легче понять, что эти ссылки объединены и составляют одну логическую единицу.

Различные типы меню

Существует несколько типов меню. Большинство из них можно создавать с помощью списков. Меню, созданные на основе списков, чаще всего применяются для организации навигации по странице и по сайту, контекстной навигации, для организации схемы сайта и нумерации страниц. Ниже приведено краткое описание различных типов меню.

Примером организации навигации по странице может служить оглавление для одной страницы со ссылками, указывающими на различные разделы страницы.

Навигация по сайту является наиболее распространенным типом меню, в том или ином виде присутствующим практически на каждом сайте.

Это меню для всего сайта, со ссылками, указывающими на различные страницы на этом сайте. Данное меню представляет пользователю как доступные варианты перехода, так и иерархию сайта. Важным вопросом при организации навигации по сайту является вопрос выделения текущего документа в меню, чтобы создать у пользователя ощущение присутствия в определенном месте.

Одно из золотых правил разработки и навигации в Интернете состоит в том, что текущий документ никогда не должен ссылаться на себя самого, однако должен явно отличаться от других записей в меню. Это важно, так как дает посетителям точку опоры и говорит им, где они находятся в своем путешествии по сайту.

Если пользователю предлагается ссылка, которая указывает на текущий документ, активация ее приведет к перезагрузке документа. По этой причине ссылка на текущую страницу никогда не должна присутствовать в меню ссылок. Данную ссылку лучше удалить или деактивировать.

Контекстная навигация представляет собой список ссылок, которые указывают на HTML-документы, связанные с темой просматриваемой страницы. Контекстные меню являются ссылками, которые создаются на основе содержимого текущего документа, и предлагают дополнительную информацию, связанную с текущей страницей. Классическим примером являются ссылки на связанные статьи, которые можно видеть в начале новостных лент.

Схема сайта является схемой всех страниц сайта или основных разделов крупных сайтов. Она позволяет посетителям сайта получить представление обо всей структуре сайта и быстро перейти в требуемое место, даже если оно находится где-то в иерархии страниц. Как схема сайта, наряду с поиском по сайту, дает возможность пользователю быстро и эффективно перемещаться по сайту.

Нумерация страниц необходима для организации быстрого перемещения в большом документе, разбитом на отдельные страницы. Можно встретить нумерацию страниц в больших архивах изображений или на страницах с результатами поиска, такими, как поисковые системы Яндекс, Google и т.п. Нумерация страниц отличается от других типов навигации тем, что она обычно соединяется с тем же

документом, но с помощью ссылки, которая содержит дополнительную информацию, такую как номер страницы, с которой начинать. Меню нумерации страниц позволяют посетителям просматривать множество данных без потери представления о том, где они находятся.

Карты-изображения

Для организации большинства навигационных меню вполне достаточно использовать упорядоченные или неупорядоченные списки. Существуют, однако, ситуации, которые могут потребовать другие методы организации меню.

Одним из таких методов являются клиентские карты-изображения. Карты-изображения превращают изображение в меню, задавая в нем интерактивные области, которые можно соединить с различными документами. Например, следующий ниже код превращает изображение легендарной ливерпульской четверки в меню, на котором можно щелкать мышью.

Рис. 8.2. Изображения участников группы "Beatles" являются ссылками

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
```

```
 "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<TITLE>Карта-изображение</TITLE>
</HEAD>
<BODY>
<img src="map.gif" width="400" height="100" usemap="#map">
<MAP name="map">
<AREA shape="poly" title="Пол" coords="57,47, 40,29, 53,10, 74,5, 95,9, 10
<AREA shape="poly" title="Ринго" coords="144,87, 120,68, 122,45, 155,38,
<AREA shape="poly" title="Джордж" coords="212,62, 192,56, 176,41, 177,2
<AREA shape="poly" title="Джон" coords="332,71, 300,63, 280,40, 298,17,
</MAP>
</BODY>
</HTML>
```

Любое изображение можно превратить в меню, определяя карту с различными областями, называемыми также горячими точками. Для указания браузеру на то, что изображение является картой, используется атрибут `usemap`. Он является ссылкой на описание конфигурации карты, которая задается элементом `MAP`. Значение атрибута `name` данного элемента должно соответствовать имени в `usemap`. Для задания активной области, являющейся ссылкой на документ HTML, используется элемент AREA.

Каждая область должна иметь несколько атрибутов.

Атрибут `href` определяет URL-адрес ресурса, с которым должна соединяться ссылка.

Атрибут shape определяет форму активной области. Область может быть задана в виде окружности (значение `circle`), прямоугольника (значение `rect`) или многоугольника (значение `poly`).

Атрибут `coords` определяет координаты в изображении, которые должны стать горячими точками. Значения координат отсчитываются от верхнего левого угла изображения и могут измеряться в пикселях или процентах. Для прямоугольных областей необходимо определить только верхний левый и нижний правый углы; для окружностей необходимо

определить центр окружности и радиус; для многоугольников необходимо предоставить наиболее полный список всех угловых точек.

Использование карт-изображений наглядней, чем обычные текстовые ссылки, и позволяет применять всего один графический файл для организации ссылок. Однако не нужно считать, что карты-изображения следует включать везде, где требуются графические ссылки. Прежде всего, следует оценить все доводы за и против, а также просмотреть альтернативные варианты.

Практическое занятие 3

Целью практического занятия является закрепление материала лекций 7 и 8. В ходе практического занятия слушатель осваивает основные приемы организации навигации на сайте.

Слушателю предлагается организовать навигацию по странице page2.html и создаваемому сайту с помощью списков и ссылок. В качестве навигации по странице page2.html можно создать иерархическое меню, содержащее разделы "Дела конструкторские" и "Дела машинные" с соответствующими подразделами, позволяющее пользователю перемещаться не только к данным разделам, но и к содержащимся в них законам. Для реализации навигации по сайту предлагается создать меню, позволяющее пользователю перемещаться между созданными на предыдущих практических занятиях страницами page1.html, page2.html и page3.html. Для реализации навигации по сайту предлагается также создать карту ссылок. Изображение-основа карты ссылок может быть создано в любом графическом редакторе.

Таблицы

В лекции рассматриваются основные приемы работы с таблицами в HTML. Описываются некоторые способы улучшения представления данных в таблице, способы дополнительного структурирования таблиц и др.

Таблицы позволяют представить большой объем информации в компактном и наглядном виде, а также сравнивать и сопоставлять различные данные. Они часто встречаются на сайтах, где служат для представления разнообразной статистики, рейтингов, сравнения цен и т.д.

Создание таблиц

Для добавления таблицы на веб-страницу используется элемент `TABLE`, который указывает браузеру, что содержимое необходимо организовать в табличном виде. Этот элемент служит контейнером для элементов, определяющих содержимое таблицы: внутри элемента `<TABLE>…</TABLE>` допустимо использовать такие элементы HTML, как `CAPTION`, `COL`, `COLGROUP`, `THEAD`, `TBODY`, `TFOOT`, `TH`, `TD` и `TR`.

Любая таблица состоит из строк и ячеек, которые задаются с помощью элементов TR и TD соответственно.

Для определения строки в уже созданной таблице используется элемент `TR`, который позволяет браузеру организовать содержимое между тегами `<TR>` и `</TR>` в горизонтальном виде: между ними должны размещаться все данные, которые требуется поместить в одну строку. Внутри строки таблицы обычно размещаются ячейки с данными. Для определения ячейки таблицы используется элемент `TD`. Число элементов `TD` в строке определяет число ячеек. Ячеек может быть произвольное количество, однако таблица должна содержать хотя бы одну ячейку.

Вместо элемента TD допускается использовать элемент `TH`. Элемент `TH` определяет содержимое ячейки как заголовок для каждого столбца. Это

помогает не только семантически описать содержимое, но также представляет его более аккуратно в различных браузерах и устройствах. Текст в такой ячейке обычно отображается браузерами жирным шрифтом и выравнивается по центру. В остальном разницы между ячейками, созданными с использованием элементов `TD` и `TH`, нет.

Фрагмент кода, представленного ниже, создает таблицу, состоящую из трех столбцов и четырех строк, одна из которых задает строку заголовков столбцов:

```
<TABLE>
  <TR>
    <TH>Поисковая система</TH>
    <TH>Декабрь 2009 г.</TH>
    <TH>Ноябрь 2009 г.</TH>
  </TR>
  <TR>
    <TD>Яндекс</TD>
    <TD>48.0%</TD>
    <TD>47.9%</TD>
  </TR>
  <TR>
    <TD>Google</TD>
    <TD>34.9%</TD>
    <TD>34.7%</TD>
  </TR>
  <TR>
    <TD>Search.Mail.ru</TD>
    <TD>8.6%</TD>
    <TD>8.6%</TD>
  </TR>
</TABLE>
```

Созданная таблица представлена на рисунке 9.1.

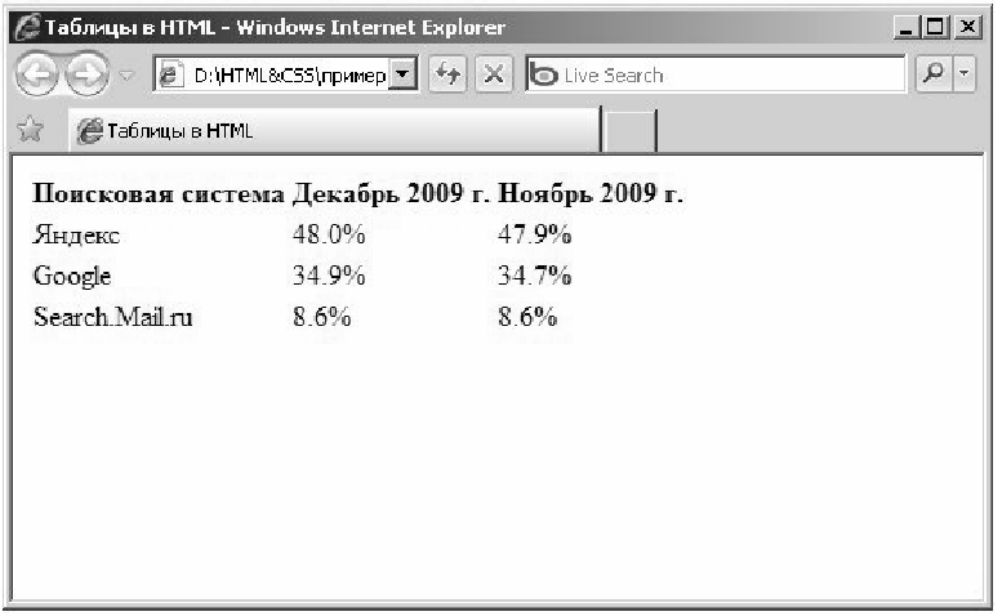


Рис. 9.1. Пример простой таблицы со строкой заголовков

Заголовок таблицы

Таблицам на странице удобно задать заголовок, содержащий название таблицы и ее описание. Для этой цели в HTML используется специальный элемент `CAPTION`, который помещается внутри элемента `<TABLE>…</TABLE>`. Удобство использования элемента `CAPTION` состоит в том, что заголовок, созданный с его помощью, оказывается привязанным к таблице и не выходит за условные рамки, ограниченные ее шириной.

По умолчанию, заголовок таблицы помещается сверху таблицы по центру, его ширина не превышает ширины таблицы, и в случае длинного текста он автоматически переносится на новую строку. Для изменения положения заголовка предусмотрен атрибут `align`. Однако этот атрибут помечен в Спецификации HTML 4.01 как не рекомендуемый, и с его помощью получить код, одинаково работающий в разных браузерах, довольно сложно. Для выравнивания заголовка таблицы рекомендуется использовать CSS, а именно, свойство `text-align`.

Добавление некоторых свойств

Тот факт, что таблицы применяются достаточно часто, обязан не только их гибкости и универсальности, но и обилию атрибутов элементов `TABLE`, `TR` и `TD`, управляющих различными свойствами таблицы. Рассмотрим наиболее часто используемые атрибуты подробнее.

Описание таблицы: атрибут summary

Пользователи с нормальным зрением могут сами для себя решить, стоит им изучать таблицу или нет. Быстрый взгляд на нее и на ее заголовок позволит сказать, велика ли таблица и что она содержит. Пользователи с программами для чтения с экрана не смогут этого сделать, пока разработчик не добавит к элементу `TABLE` атрибут `summary`. Этот атрибут позволяет написать более развернутое описание, чем то, которое подходит для элемента `CAPTION`. Содержание атрибута summary не будет отражено визуальными браузерами, поэтому можно сделать описание достаточно длинным для того, чтобы те, кто его услышат, смогли понять, что именно представлено в таблице. Данный атрибут рекомендуется использовать только в случае необходимости, например, для больших и сложных таблиц. Для созданной выше таблицы можно добавить описание следующим образом:

```
<TABLE summary="Статистика переходов с основных поисковых систем
...
</TABLE>
```

Толщина рамки таблицы: атрибут border

Атрибут устанавливает толщину рамки в пикселах, т.е. использует в качестве значения любое положительное число. По умолчанию, рамка изображается трехмерной, однако вид рамки меняется в зависимости от браузера. При использовании атрибута `border` без аргументов, браузер отображает рамку толщиной один пиксел. Толщину рамки возможно также установить с помощью одноименного свойства CSS. Следующий фрагмент кода добавляет к таблице рамку толщиной 2 пиксела:

```
<TABLE border="2">
...
</TABLE>
```

Ширина таблицы: атрибут width

Ширину таблицы можно задать с помощью атрибута `width`. Ширину можно задавать в пикселах или процентах от доступного пространства. Если общая ширина содержимого превышает указанную ширину таблицы, то браузер будет пытаться форматировать текст, чтобы подогнать его к заданным размерам. Если это невозможно, атрибут `width` будет проигнорирован, и новая ширина таблицы будет вычислена на основе ее содержимого. Если ширина явно не указана, то она также будет вычисляться на основе содержимого таблицы. Аналогом данного атрибута является одноименное свойство CSS. В следующем примере ширина таблицы устанавливается равной 75% от ширины окна браузера:

```
<TABLE width="75%">
...
</TABLE>
```

Расстояние между ячейками таблицы: атрибут cellspacing

Атрибут `cellspacing` задает расстояние между внешними границами ячеек. Если установлен атрибут `border`, то толщина границы принимается в расчет и входит в общее значение. Значением атрибута может быть любое целое положительное число. По умолчанию, атрибут `cellspacing` принимает значения 0 или 2, в зависимости от того, установлен атрибут `border` или нет. Следующий фрагмент кода устанавливает расстояние между ячейками 2 пиксела:

```
<TABLE cellspacing="2">
...
</TABLE>
```

Расстояние внутри ячеек: атрибут cellpadding

Атрибут `cellpadding` определяет расстояние между границей ячейки и ее содержимым: он добавляет пустое пространство к ячейке, увеличивая тем самым ее размеры. Добавление `cellpadding` позволяет улучшить читабельность текста. При отсутствии границ этот атрибут не имеет особого значения. Значение данного атрибута может быть задано в пикселах или процентах от доступного пространства и по умолчанию равно нулю. Задать расстояние между границей ячейки и ее содержимым можно следующим образом:

```
<TABLE cellpadding="2">
...
</TABLE>
```

Пример совместного применения описанных выше атрибутов представлен в следующем листинге:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<TITLE>Таблицы в HTML</TITLE>
</HEAD>
<BODY>
<TABLE border="1" width="75%" cellspacing="2"
        summary="Статистика поисковых систем Рунета">
   <CAPTION>Статистика поисковых систем</CAPTION>
   <TR>
      <TH>Поисковая система</TH>
      <TH>Декабрь 2009 г.</TH>
      <TH>Ноябрь 2009 г.</TH>
   </TR>
   <TR>
      <TD>Яндекс</TD>
      <TD>48.0%</TD>
      <TD>47.9%</TD>
   </TR>
```

```
    <TR>
      <TD>Google</TD>
      <TD>34.9%</TD>
      <TD>34.7%</TD>
    </TR>
    <TR>
      <TD>Search.Mail.ru</TD>
      <TD>8.6%</TD>
      <TD>8.6%</TD>
    </TR>
  </TABLE>
  </BODY>
  </HTML>
```

Результат выполнения данного кода представлен на рисунке 9.2.

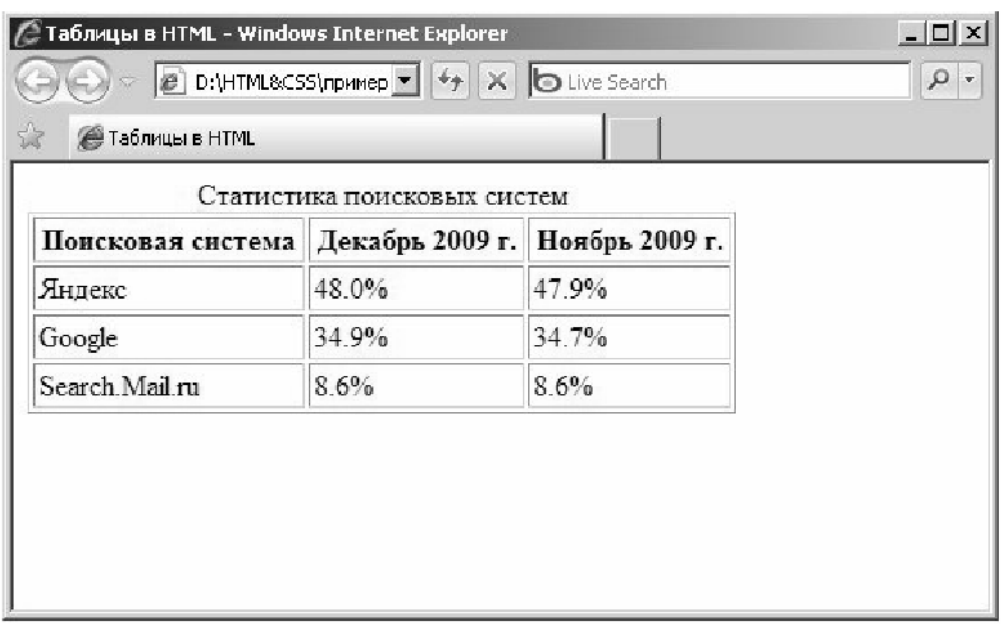


Рис. 9.2. Результат применения описанных выше атрибутов

Каждая ячейка таблицы, задаваемая элементом `TD` или `TH`, тоже имеет свои атрибуты, часть из которых совпадает с атрибутами элемента `TABLE`. Рассмотрим наиболее часто используемые атрибуты элемента `TD`.

Объединение ячеек: атрибуты colspan и rowspan

Атрибут `colspan` устанавливает число ячеек, которые должны быть объединены по горизонтали. Например, в первой таблице, показанной на рисунке 9.3, содержатся две строки и две колонки, причем верхние горизонтальные ячейки объединены с помощью параметра `colspan`. Пример кода представлен ниже:

```
<TABLE width="200" border="1" cellpadding="4">
  <TR>
    <TD colspan="2">Ячейка 1</TD>
  </TR>
  <TR>
    <TD>Ячейка 2</TD>
    <TD>Ячейка 3</TD>
  </TR>
</TABLE>
```

Атрибут rowspan устанавливает число ячеек, которые должны быть объединены по вертикали. Например, во второй таблице, представленной на рисунке 9.3, содержатся две строки и две колонки, причем левые вертикальные ячейки объединены с помощью параметра `rowspan`. Такого результата можно достичь с помощью следующего фрагмента кода:

```
<TABLE width="200" border="1" cellpadding="4">
    <TR>
      <TD rowspan="2">Ячейка 1</TD>
      <TD>Ячейка 2</TD>
    </TR>
    <TR>
      <TD>Ячейка 3</TD>
    </TR>
  </TABLE>
```

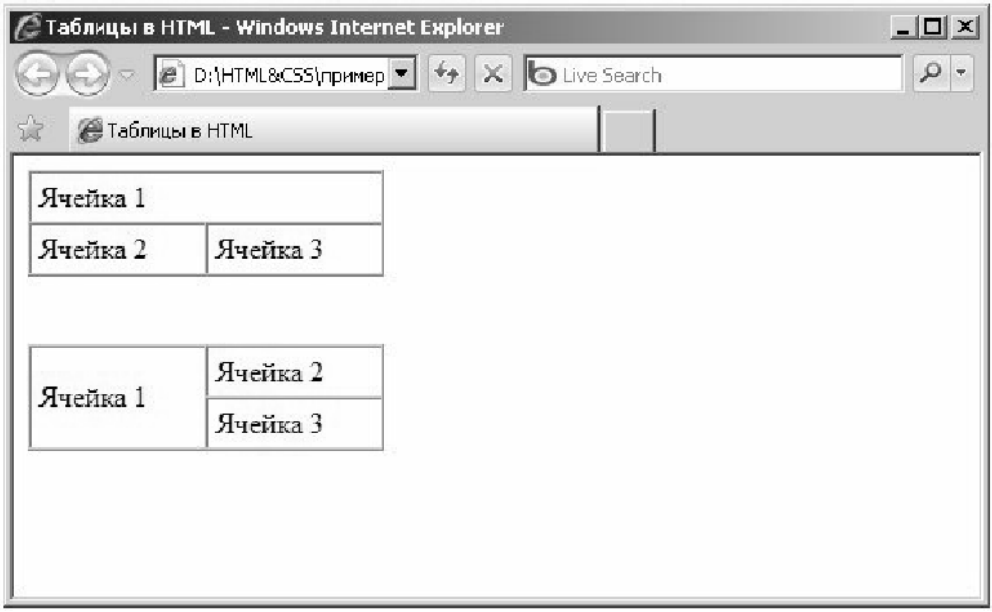


Рис. 9.3. Примеры таблиц, демонстрирующих горизонтальное и вертикальное объединение ячеек

Перенос слов в ячейках: атрибут nowrap

Добавление атрибута `nowrap` к элементу `TD` заставляет браузер отображать текст внутри ячейки без переносов. Неправильное использование этого атрибута может привести к тому, что таблица будет слишком широкой и не поместится целиком на веб-странице. Как следствие, появится горизонтальная полоса прокрутки, и пользоваться подобной таблицей будет неудобно. В Спецификации HTML 4.01 данный атрибут помечен как не рекомендуемый, и его применение осуждается. Для достижения аналогичной функциональности рекомендуется использовать свойство CSS *`white-space`*.

Ширина и высота ячейки: атрибуты width и height

С помощью атрибутов `width` и `height` можно задать ширину и высоту ячейки. Суммарное значение ширины всех ячеек может

превышать общую ширину таблицы только в том случае, если содержимое ячейки превышает указанную ширину. Высоту таблицы и ее ячеек браузер устанавливает сам, исходя из их содержимого. Однако при использовании атрибута `height` высота ячеек будет изменена. Здесь возможны два варианта. Если значение `height` меньше, чем содержимое ячейки, то этот атрибут будет проигнорирован. В случае, когда установлена высота ячейки, превышающая ее содержимое, добавляется пустое пространство по вертикали. Применение атрибутов `width` и `height` также осуждаются в Спецификации HTML. Вместо них рекомендуется использовать одноименные свойства CSS.

В заключение этого раздела следует отметить, что официальная спецификация HTML не рекомендует все атрибуты, касающиеся оформления таблиц и ячеек. Устанавливать такие свойства таблиц и ячеек, как выравнивание самой таблицы относительно окна браузера, текста в ячейках, цвет и стиль рамки вокруг ячеек таблицы, фона и многое другое рекомендуется только с использованием свойств CSS. Подробно эти вопросы будут освещены в лекции 15.

Дополнительная структуризация таблицы

Сложные таблицы со множеством столбцов и строк возможно дополнительно структурировать, определив верхний колонтитул, тело и нижний колонтитул таблицы. В сложных таблицах использование этих элементов позволит структурировать содержимое таблицы не только для разработчика, но и для браузеров и других устройств. Для добавления данной структуры к таблице используются элементы `THEAD`, `TBODY` и `TFOOT` соответственно.

Элемент `THEAD` предназначен для хранения одной или нескольких строк, представленных вверху таблицы. Допустимо использовать не более одного элемента `THEAD` в пределах одной таблицы, и он должен идти в исходном коде сразу после элемента `TABLE`.

Элемент `TBODY` позволяет создавать структурные блоки внутри таблицы, к которым можно применять единое оформление через стили. Допускается применять несколько элементов `TBODY` внутри контейнера `TABLE`. Элемент `TBODY` не должен перекрываться с другими видами

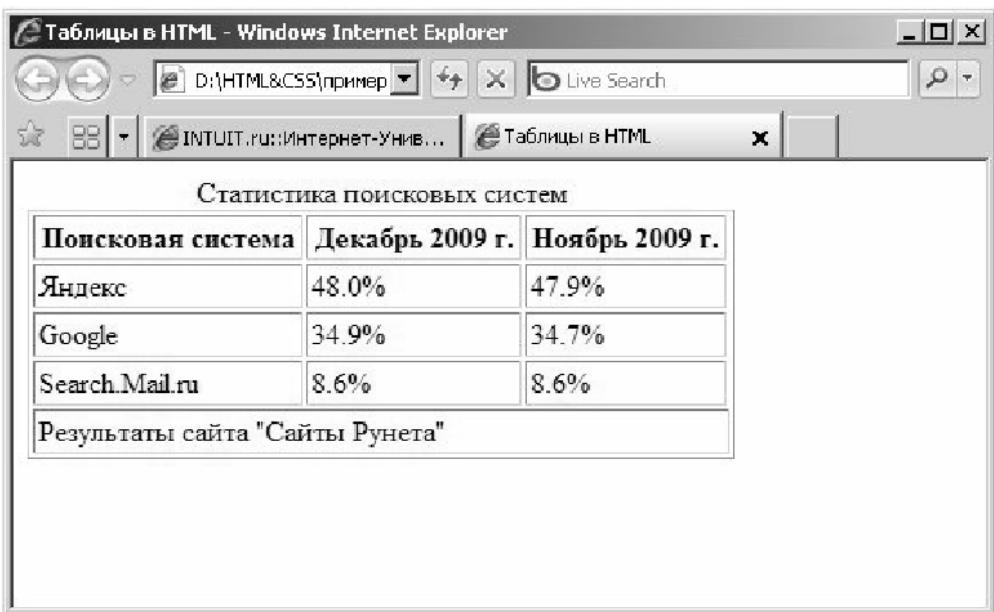
группировок строк (т.е. с TFOOT и THEAD).

Элемент TFOOT предназначен для хранения одной или нескольких строк, которые представлены внизу таблицы, и служит для создания нижнего колонтитула таблицы. Хотя этот элемент в исходном коде должен быть определен до элемента TBODY, браузеры отображают его в самом низу таблицы. В пределах таблицы разрешается использовать только один элемент TFOOT.

Пример использования элементов THEAD, TBODY и TFOOT, представлен в следующем ниже листинге. Результат выполнения данного кода иллюстрирует рисунок 9.4.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<TITLE>Таблицы в HTML</TITLE>
</HEAD>
<BODY>
<TABLE border="1" width="75%" cellspacing="2"
        summary="Статистика поисковых систем Рунета">
  <CAPTION>Статистика поисковых систем</CAPTION>
  <THEAD>
   <TR>
     <TH>Поисковая система</TH>
     <TH>Декабрь 2009 г.</TH>
     <TH>Ноябрь 2009 г.</TH>
   </TR>
  </THEAD>
  <TFOOT>
   <TR>
     <TD colspan="4">Результаты сайта "Сайты Рунета"</TD>
   </TR>
  </TFOOT>
  <TBODY>
   <TR>
     <TD>Яндекс</TD>
     <TD>48.0%</TD>
```

```
        <TD>47.9%</TD>
      </TR>
      <TR>
        <TD>Google</TD>
        <TD>34.9%</TD>
        <TD>34.7%</TD>
      </TR>
      <TR>
        <TD>Search.Mail.ru</TD>
        <TD>8.6%</TD>
        <TD>8.6%</TD>
      </TR>
    </TBODY>
  </TABLE>
  </BODY>
  </HTML>
```



Статистика поисковых систем

Поисковая система	Декабрь 2009 г.	Ноябрь 2009 г.
Яндекс	48.0%	47.9%
Google	34.9%	34.7%
Search.Mail.ru	8.6%	8.6%
Результаты сайта "Сайты Рунета"		

Рис. 9.4. Результат дополнительной структуризации таблицы

Практическое занятие 2

Целью практического занятия является закрепление материла, представленного в лекциях 8 и 9. В ходе практического занятия формируются навыки работы с таблицами и ссылками в HTML.

Слушателю предлагается оформить в виде новой веб-страницы (например, page3.html) текст, представленный в Приложении 3. В тексте слово "СССР" необходимо оформить как аббревиатуру, снабдив ее расшифровкой, т.е. указанием полной формы записи. К таблице, представленной в данной статье, необходимо добавить заголовок, соответствующий содержанию таблицы, рамку определенной толщины, задать ширину и выделить строку заголовков. Слушателю также предлагается создать несколько таблиц с различной структурой, примеры которых даны в Приложении 4.

В части работы со ссылками рекомендуется создать несколько текстовых ссылок на любимые сайты, а также различные ресурсы Сети (документ Microsoft Word, графический файл и т.д.). В HTML-документе page2.html, созданном на первом практическом занятии, необходимо создать несколько закладок (т.е. ссылок на различные места того же документа) и организовать переходы к этим закладкам. Например, переходы можно организовать на различные законы, после которых можно предусмотреть ссылку, возвращающую пользователя к началу страницы.

Что такое CSS?

В лекции даются основные сведения о синтаксисе CSS, рассматриваются основные способы внедрения CSS в HTML. Также описываются основные селекторы, псевдоэлементы и псевдоклассы, затрагиваются такие фундаментальные концепции CSS как наследование и каскадирование.

Синтаксис CSS

Таблица стилей документа представляет собой набор правил, ассоциированных с документом HTML и определяющих его отображение. Каждое правило в таблице стилей состоит из селектора и блока объявлений. Блок объявлений всегда отделяется от селектора пробелом и заключается в фигурные скобки. Селектор определяет область применения стилевого правила, а блок объявлений содержит одно или несколько объявлений, которые отделяются друг от друга точкой с запятой.

Каждое объявление включает в себя свойство и соответствующее этому свойству значение. Любое свойство является обобщенным параметром оформления и должно отделяться от значения двоеточием. Свойства могут быть достаточно разнообразны: они могут определять цвет элемента и фона, поля, заполнение, тип и размер шрифта и т.п. Набор допустимых значений для каждого конкретного свойства должен быть определен индивидуально. Правило может быть задано следующим образом:

```
P {font-family: arial; color: red}
```

В приведенном примере свойству `font-family`, определяющему вид шрифта, присваивается значение `arial`, а свойству `color`, определяющему цвет шрифта, – значение red. Поскольку блок объявлений соотносится с селектором `P`, то областью применения данного стилевого правила будут являться все элементы `<P>…</P>`.

Существует несколько различных селекторов, каждый из которых соответствуют различным частям разметки. Четырьмя наиболее общими селекторами являются универсальные селекторы, селекторы элементов,

селекторы классов и селекторы идентификаторов.

Универсальные селекторы. Универсальные селекторы позволяют применить стилевое оформление сразу ко всем элементам на странице. Например, следующее правило устанавливает сплошную границу толщиной 1 пиксел для каждого элемента на странице:

```
* {border: 1px solid;}
```

Селекторы элементов. Для определения информации о представлении элементов HTML большинство стилевых правил используют в качестве селекторов имена этих элементов. Например,

```
P {font-size: 1.2em; margin: 10px 20px;}
```

Приведенное выше правило будет применяться ко всем элементам `<P>...</P>`.

Селекторы классов. Напомним, что Спецификация HTML 4.01 предусматривает для всех элементов специальный атрибут `class`, который позволяет особым образом отметить некоторые структурные элементы в документе, например

```
<P class="important">...</P>
```

Каскадные таблицы стилей позволяют использовать в качестве селекторов значения атрибутов `class`. Названия классов в селекторах предваряются точкой и указываются после имен элементов:

```
P.important {padding-left: 20px;}
```

В приведенном примере стилевое правило будет применяться только к тем элементам `<P>...</P>`, атрибут `class` которых имеет значение `important`.

Каскадные таблицы стилей также позволяют использовать в селекторах названия классов без указания имен элементов:

```
.important {padding-left: 20px;}
```

Подобные стилевые правила будут применяться ко всем элементам,

которые имеют атрибут `class` с указанным значением.

Селекторы идентификаторов. Помимо классов, Спецификация HTML предусматривает для всех элементов еще один специальный атрибут `id`, с помощью которого любому элементу можно назначить уникальный идентификатор, например:

```
<DIV id="header">...</DIV>
```

Главное отличие идентификаторов от классов заключается в том, что каждое значение атрибута `id` должно быть уникальным. Идентификаторы в селекторах указываются после имен элементов и предваряются символом `#`:

```
DIV#header {width: 1000px;}
```

Данное стилевое правило будет применяться к единственному экземпляру элемента `<DIV>…</DIV>`, которому назначен атрибут `id` со значением `header`. Если в селекторе используется идентификатор без указания конкретного элемента, то такое стилевое правило будет применяться к единственному экземпляру любого элемента, значение `id` которого равно `header`:

```
#header {width: 1000px;}
```

Можно соединять несколько селекторов, чтобы определить еще более конкретные правила. Например, запись

```
P.warning{}
```

соответствует всем параграфам со значением `class` равным *`warning`* ;

```
DIV#example{}
```

соответствует элементу `DIV` со значением атрибута `id` равным `example` ;

```
P.info, LI.highlight{}
```

соответствует параграфам со значением `class` равным `info` и элементам списка со значением `class` равным *highlight*

Группировка селекторов

Если нескольким разным селекторам необходимо сопоставить одинаковые объявления, такие селекторы можно сгруппировать в список. Имена селекторов в данном случае должны разделяться запятыми:

```
P, UL {font-size: 1.2em; line-height: 1.5em;}
```

В приведенном примере для двух элементов `P` и `UL` назначается одинаковый размер шрифта и межстрочный интервал. Синтаксис каскадных таблиц стилей позволяет легко группировать не только простейшие селекторы с именами элементов, но также и другие типы селекторов. В некоторых случаях подобная группировка позволяет значительно сократить размер CSS-файла.

Дополнительные селекторы CSS

Кроме рассмотренных выше селекторов существуют и другие, которые позволяют выбирать элементы для стилевого оформления на основе более специальных критериев. Ниже будут рассмотрены некоторые из них.

Селекторы атрибутов. Селекторы атрибутов позволяют выбирать элементы на основе содержащихся в них атрибутов. Например, можно выбрать каждый элемент `IMG` с атрибутом `alt` с помощью следующего селектора:

```
IMG[alt] {border: 1px solid;}
```

Элементы можно выбирать не только на основе содержащихся в них атрибутов, но и по значениям этих атрибутов. Следующее правило будет применяться ко всем изображениям со значением атрибута `src` равным image.jpg:

```
IMG[src="image.jpg"] {border: 1px solid;}
```

Селекторы потомков. Селекторы потомков используются для выбора только определенных элементов, которые являются потомками других определенных элементов. Например, следующее правило будет применяться ко всем элементам `STRONG`, которые являются потомками элементов `H1`. Другими словами, оно применяется только к тем элементам `STRONG`, которые находятся внутри элемента `H1`, и никакого промежуточного элемента между ними нет:

```
H1 > STRONG {color: blue;}
```

Селекторы нижележащих элементов. Селекторы нижележащих элементов выбирают все подходящие элементы в любом месте иерархии элементов. Рассмотрим следующий фрагмент кода HTML:

```
<DIV>
  <STRONG>Первый потомок элемента DIV</STRONG>
  <P>Второй потомок элемента DIV
    <STRONG>Единственный потомок элемента P</STRONG>.
  </P>
</DIV>
```

В этом фрагменте элемент `DIV` является предком всех других элементов: двумя его потомками являются элементы `STRONG` и `P`. Элемент `P` имеет один элемент-потомок - еще один `STRONG`. Можно использовать селектор потомков для выбора только первого элемента `STRONG`, находящегося непосредственно внутри `DIV` следующим образом:

```
DIV > STRONG { ... }
```

Если вместо этого использовать селектор нижележащих элементов, то будут выбраны оба элемента `STRONG`:

```
DIV STRONG { ... }
```

Селекторы смежных одноуровневых элементов. Эти селекторы позволяют выбирать определенный элемент, который следует непосредственно после другого определенного элемента на том же уровне в иерархии элементов. Например, если необходимо выбрать

только элементы P, следующие непосредственно за элементами H1, можно воспользоваться следующим правилом:

```
H1 + P { ... }
```

Псевдоклассы

Псевдоклассы используются для обеспечения стилевого оформления различных состояний элементов. Наиболее часто псевдоклассы применяются для оформления состояний ссылок.

Псевдоклассы :link и :visited. Современные браузеры по-разному отображают посещенные и непосещенные ссылки. Для того, чтобы можно было применить к этим типам ссылок различные стилевые правила, в CSS предусмотрены специальные псевдоклассы :link и :visited.

```
a:link {color: red;}
a:visited {color: green;}
```

В приведенном выше примере первое стилевое правило будет применяться к непосещенным ссылкам в документе, а второе - к посещенным.

Псевдоклассы :hover, :active и :focus. Стиль отображения некоторых элементов может динамически изменяться в результате определенных действий пользователя. Для этого в CSS используются псевдоклассы :hover, :active и :focus.

Псевдокласс :hover применяется к соответствующему элементу в случае, когда пользователь навел курсор мыши на этот элемент, но не активировал его щелчком мыши.

Псевдокласс :active применяется к соответствующему элементу, когда пользователь нажимает кнопку мыши и до тех пор, пока он ее не отпустит. Как правило, это довольно короткий промежуток времени.

Псевдокласс :focus применяется к соответствующему элементу, когда

он получает фокус (в результате нажатия определенных клавиш).

CSS не определяет, к каким именно элементам могут применяться указанные псевдоклассы. Однако современные браузеры поддерживают их только применительно к HTML-элементам `A`, т.е. к ссылкам, например:

```
A:link    {color: red}
A:visited {color: blue}
A:hover   {color: yellow}
A:active  {color: green}
```

Псевдокласс `:first-child`. Данный псевдокласс выбирает все экземпляры элемента, который является первым элементом-потомком своего предка, поэтому, например, следующее правило выбирает первый объект списка любого вида и делает его текст жирным:

```
LI:first-child {font-weight: bold;}
```

Псевдокласс `:lang`. Псевдокласс `:lang` позволяет выбирать элементы, язык которых был задан как определенный язык с помощью атрибута `lang`. Например, следующий элемент

```
<P lang="en-US">London is a capital of Great Britan.<P>
```

можно было бы выбрать с помощью кода

```
p:lang(en-US) { ... }
```

Псевдоэлементы

Одним из назначений псевдоэлементов является установка стиля первой буквы или первой строки текста в заданном элементе. Чтобы легко создать буквицу в начале каждого параграфа документа, можно использовать следующее правило:

```
P:first-letter {
  font-weight: bold;
  font-size: 200%
```

```
}
```

Первая буква каждого параграфа будет теперь жирной и на 200% больше остального текста параграфа.

Чтобы сделать первую строку каждого параграфа жирной, можно использовать следующее правило:

```
P:first-line {font-weight: bold;}
```

Другим применением псевдоэлементов является вставка автоматически генерируемого содержимого перед или после указанного элемента. За эти действия отвечают псевдоэлементы `:before` и `:after` соответственно. Подробно ознакомиться с особенностями применения данных пседоклассов можно в Спецификации CSS.

Комментарии

Комментарии в CSS начинаются с группы символов `/*` и заканчиваются символами `*/`. Например:

```
/* Так выглядит комментарий в CSS */
```

Использование комментариев может сэкономить немало времени при поиске определенного стилевого правила в CSS-файлах. Комментарии в CSS не влияют на интерпретацию таблиц стилей браузерами. Однако не рекомендуется использовать комментарии на русском языке, т.к. в некоторых случаях использование кириллицы в CSS-комментариях может приводить к неправильной интерпретации таблиц стилей некоторыми популярными браузерами.

Включение таблиц стилей в HTML-документ

Имеется три способа задания стилей в HTML-документе. Перечислим их в порядке предпочтения.

Внешние таблицы стилей

Для подключения к документу внешней таблицы стилей (т. е. таблицы стилей, хранящейся в отдельном файле) следует поместить в заголовок документа HTML элемент `LINK`, например:

```
...
<HEAD>
  <LINK rel="stylesheet" href="style.css" type="text/css">
</HEAD>
...
```

В элементе `LINK` можно дополнительно указать типы устройств, на которые распространяется данная таблица стилей, например:

```
<LINK rel="stylesheet" href="style.css" type="text/css"
 media="screen, print">
```

Внешние таблицы стилей рекомендуется использовать в том случае, когда несколько HTML-документов пользуются единой таблицей стилей.

Внутренние таблицы стилей

Для включения в документ внутренней таблицы стилей следует поместить в заголовок документа элемент `STYLE`. Например:

```
...
<HEAD>
  <STYLE type="text/css">
     H1 {text-align: center}
  </STYLE>
</HEAD>
<BODY>
  <H1>Этот заголовок имеет указанный выше стиль</H1>
</BODY>
```

Внутренние таблицы стилей рекомендуется использовать в том случае, когда данная таблица стилей используется только в данном HTML-документе.

Таблицы стилей элементов

Последним способом задания стилей является определение таблицы стилей отдельного элемента путем задания его атрибута `style`. Например, стиль элемента `H1` из предыдущего примера мог бы быть задан и так:

```
<H1 style="text-align: center">
```

Подобной практики следует избегать. Она приемлема только в том случае, когда документ HTML содержит единственный элемент с данным набором стилей.

Наследование

Наследование в CSS является механизмом, с помощью которого определенные свойства передаются от элемента предка его элементам потомкам. Наследуются не все свойства CSS: например, поля не наследуются, так как маловероятно, что элементу-потомку могут понадобиться такие же поля, как и его предку. В большинстве случаев здравый смысл подскажет, какие свойства наследуются, а какие нет, но для абсолютной уверенности необходимо проверить свойство в итоговой таблице свойств в Спецификации CSS (ссылка: http://www.w3.org/TR/CSS21/propidx.html - http://www.w3.org/TR/CSS21/propidx.html). Однако следует помнить, что значения, заданные в виде процентных величин, не наследуются никогда.

Пусть, например, элемент `H1` содержит элемент `EM`:

```
<H1>Этот заголовок <EM>очень важен</EM>!</H1>
```

Если элементу `EM` не присвоен цвет, то он унаследует цвет своего предка, т.е. элемента `H1`. Для задания стиля отображения элементов по умолчанию, достаточно задать стиль элемента `BODY`. Все остальные элементы являются потомками этих элементов, поэтому они будут наследовать их свойства.

Для свойств, которые не наследуются по умолчанию, можно определить принудительное наследование, используя ключевое слово `inherit`. Например, следующее правило заставит все параграфы наследовать все свойства фона от своих предков:

```
P {background: inherit;}
```

Принудительное наследование не предназначено для постоянного использования. Оно может быть полезно для отмены объявления в конфликтующем правиле. Однако данный вид наследования необходимо использовать с осторожностью, так как, например, Internet Explorer, включая версию 7, не поддерживает это ключевое слово.

Каскадирование

Сам термин CSS означает Каскадные таблицы стилей (Cascading Style Sheets), поэтому нет ничего удивительного, что каскадирование является его важной характеристикой. Это механизм, который управляет конечным результатом, когда несколько конфликтующих объявлений CSS применяется к одному элементу. Например, поверх стилевых спецификаций, примененных к какой-нибудь отдельной веб-странице, может действовать стилевой файл, общий для всех страниц веб-сайта.

Имеется три основных показателя, которые управляют порядком, в котором применяются объявления CSS. К таким показателям относятся важность, специфичность и порядок исходного кода.

Важность объявления CSS зависит от того, где оно определено. Таблицы стилей могут иметь три источника происхождения: автор, пользователь и агент пользователя.

Таблица стилей агента пользователя является встроенной таблицей стилей браузера. Каждый браузер имеет свои используемые по умолчанию правила, определяющие, как выводить различные элементы HTML, если никакой стиль не определен пользователем или создателем страницы. Типичным примером является оформление ссылок: непосещенные ссылки обычно выводятся синим цветом и

подчеркнутыми.

Таблица стилей пользователя является таблицей стилей, которую определил пользователь. Не все браузеры поддерживают таблицы стилей пользователя, но они могут быть очень полезны, особенно для пользователей с некоторыми типами функциональных недостатков.

Таблица стилей автора является таблицей стилей, которую автор документа (или, более вероятно, дизайнер сайта) написал и присоединил к соответствующему документу HTML или включил в него.

Как правило, вес правил таблицы автора больше, чем вес правил таблицы пользователя, а вес правил таблиц автора и пользователя больше, чем вес правил таблицы агента пользователя.

Для того, чтобы правила пользовательской таблицы стилей могли перекрывать авторскую, CSS содержит атрибут !important. Правило пользовательской таблицы стилей, имеющее такой атрибут, имеет больший вес, чем соответствующее правило авторской таблицы стилей. Например, если в пользовательской таблице определено следующее ниже правило, то не имеет значения, что определил автор веб-страницы, и не имеет значение, какое семейство шрифтов задано по умолчанию в браузере - все будет выводиться шрифтом Comic Sans MS.

```
* {
  font-family: "Comic Sans MS" !important;
}
```

Специфичность можно представить как меру того, насколько конкретным является селектор некоторого правила. Селектор с низкой специфичностью может соответствовать многим элементам, в то время как селектор с высокой специфичностью может соответствовать только одному элементу на страницу. Если два или больше объявлений конфликтуют за данный элемент, и все они имеют одинаковую важность, то победителем в правиле будет объявление с наиболее специфичным селектором. В общем случае класс элементов является более специфичным, чем просто элемент, а идентификатор элемента более специфичен, чем класс.

Если два объявления, влияющие на один и тот же элемент, имеют одинаковую важность и специфичность, то окончательное решение зависит от порядка исходного кода. Объявление, которое появляется позже в таблицах стилей, будет выигрывать у тех, которые встречаются раньше. Например, если имеется единственная внешняя таблица стилей, то при возникновении конфликта объявления в конце файла будут переопределять объявления, которые встречаются раньше в файле. Конфликтующие объявления могут также возникать в различных таблицах стилей. В этом случае порядок, в котором присоединяются или включаются таблицы стилей, будет определять, какое объявление будет применяться.

Оформление текста с помощью CSS

В лекции рассматриваются основные свойства CSS, позволяющие управлять стилем, размером, семейством, гарнитурой шрифта, а также свойства, отвечающие за выравнивания и отступы текста, регулировки просвета и высоты строки.

Большая часть сайтов, несмотря на их разнородную направленность, имеет нечто общее. Это интересная, привлекающая посетителей информация, а также интерактивная возможность пообщаться с другими людьми. И в том, и другом случае дело не обходится без текста. Именно текст служит основным составляющим практически любого сайта. Красиво и элегантно оформленный текст может лучше передать замысел автора и привлечь к себе внимание. К тому же с таким текстом приятнее работать, он лучше воспринимается, и пользователи это ценят.

Задание свойств шрифтов

Существуют тысячи шрифтов, которые предназначены для оформления текстов. Однако, число шрифтов, применяемых для набора текста на сайтах, существенно ниже. Конечно, можно задать, например, для заголовка, вычурный шрифт, установленный на компьютере разработчика. Но если такого шрифта на компьютере пользователя нет, то текст будет отображаться шрифтом, установленным в браузере по умолчанию. Получается, что труд дизайнеров и разработчиков пропал даром.

Одной из возможностей обойти эту проблему является новая концепция стандарта CSS2. В основе новой концепции лежит понятие загружаемых шрифтов, т.е. шрифтов, отсутствующих на компьютере пользователя, но доступных для загрузки из сети Интернет. В дополнение к этому CSS2 предусматривает наличие базы данных о шрифтах, содержащей их разнообразные характеристики и позволяющей по мере необходимости синтезировать недостающие шрифты на основе шрифтов, доступных обозревателю. Однако, несмотря на то, что CSS поддерживает эту возможность, в реальности она используется очень редко, т.к. далеко не все браузеры поддерживают данную технологию, а

пользователи не любят загружать лишнюю информацию.

Поэтому самым распространенным способом гарантировать правильное отображение шрифтов в браузере пользователя является использование стандартных шрифтов, встроенных в браузер и операционную систему.

Семейство шрифтов: свойство font-family

Свойство `font-family` используется для задания списка имен семейств шрифтов для отображения содержимого элемента. Список шрифтов может включать одно или несколько названий, разделенных запятыми. Если в имени шрифта содержатся пробелы, например, Times New Roman, оно должно заключаться в двойные или одинарные кавычки. Гарнитуры должны указываться в порядке возрастающей вероятности доступности или предпочтения. В качестве защиты от отказа значение свойства `font-family` всегда должно заканчиваться ключевым словом, ссылающимся на родовое имя шрифта. Таким образом, последовательность шрифтов лучше начинать с экзотических типов и заканчивать обобщенным именем, которое задает вид начертания.

Например, следующее ниже свойство следует понимать как указание браузеру пользователя использовать шрифт Verdana; если его нет, то использовать шрифт Arial; если его нет, то использовать родовой шрифт sans-serif:

```
font-family: Verdana, Arial, sans-serif;
```

Такой список необходим, поскольку разработчику заранее не известно, какие именно шрифты установлены на компьютерах пользователей.

Имя семейства шрифтов может быть задано как название семейства шрифтов (например, Times New Roman, Arial и т.д.) или как родовое имя. Родовые имена шрифтов были разработаны на тот случай, если на компьютере пользователя не установлен ни один из шрифтов, заданных разработчиком. В этом случае браузер использует родовой шрифт, начертание которого напоминает шрифт, который планировал

использовать разработчик. Спецификацией определено пять родовых имен, изображения которых представлены на рисунке 11.1.

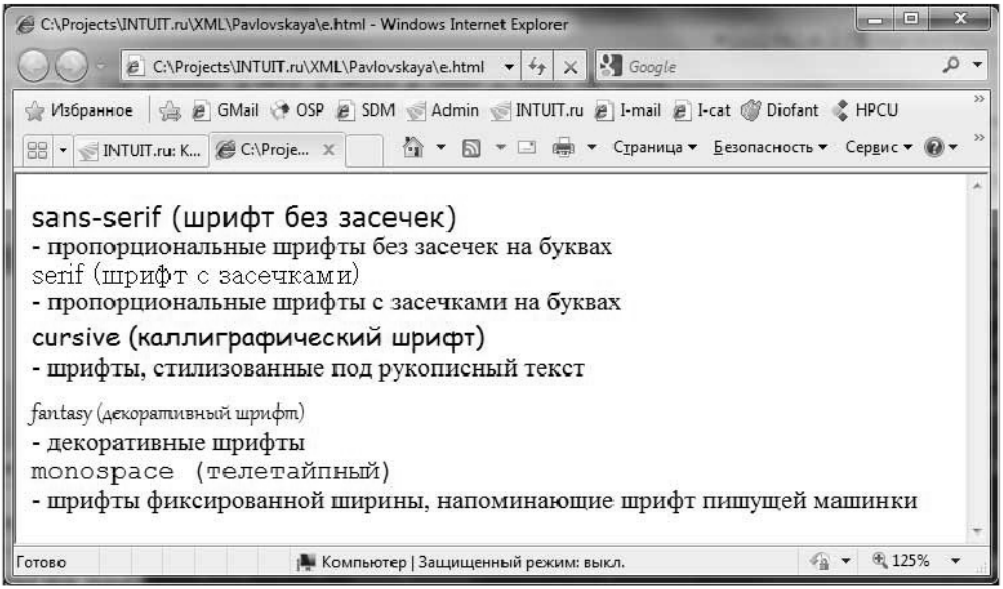


Рис. 11.1. Пример различных семейств шрифтов

Т.к. список шрифтов на компьютерах пользователей может сильно различаться в зависимости от операционной системы и собственных предпочтений, необходимо пользоваться наиболее распространенными шрифтами, к которым относятся Arial, Comic Sans MS, Courier, Courier New, Lucida Console, Tahoma, Times, Times New Roman, Trebuchet MS, Verdana. Однако следует помнить, что шрифты с одинаковыми именами в разных браузерах и системах могут незначительно отличаться друг от друга по форме или по размеру.

Размер шрифтов: свойство font-size

Размер шрифта может быть установлен несколькими способами. Набор констант `xx-small`, `x-small`, `small`, *`medium`*, `large`, `x-large`, `xx-large` задает размер, который называется абсолютным. По правде говоря, он не совсем абсолютный, поскольку зависит от настроек браузера и операционной системы. На рисунке 11.2 представлены варианты размеров шрифтов, соответствующих данным

константам.

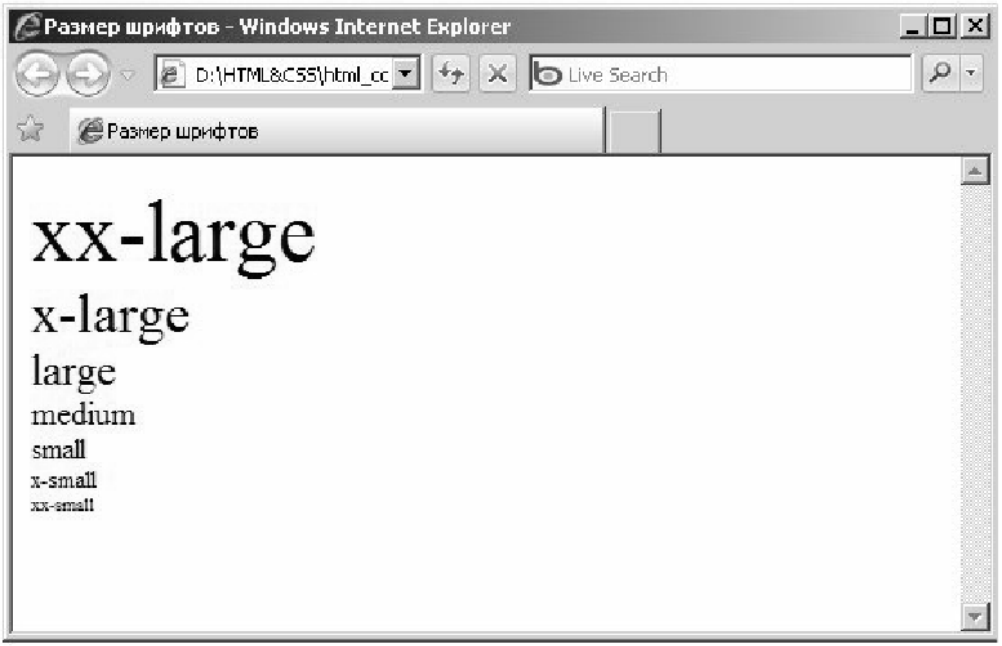


Рис. 11.2. Пример использования различных значений свойства font-size

Другой набор констант larger, smaller устанавливает относительные размеры шрифта. Поскольку размер унаследован от родительского элемента, эти относительные размеры применяются к родительскому элементу, чтобы определить размер шрифта текущего элемента.

Также разрешается использовать любые допустимые единицы CSS: em (высота шрифта элемента), ex (высота символа x), пункты (pt), пикселы (px), проценты (%) и др. При использовании процентной записи за 100% принимается размер шрифта родительского элемента. Если размер шрифта задается в пунктах или пикселах, то изменить эту величину с помощью специальной опции браузера "Размер шрифта" нельзя. Если шрифт установлен слишком мелким, то исправить этот недостаток пользователю простыми средствами не удастся. Поэтому лучше использовать другие единицы размеров шрифта, например, проценты.

Насыщенность шрифтов: свойство font-weight

Насыщенность шрифта (или жирность) управляется с помощью свойства *font-weight*. Значениями этого свойства могут быть ключевые слова bold, bolder, lighter и normal, которые устанавливают полужирное, жирное, светлое и нормальное начертание шрифта. Также можно использовать условные единицы от 100 до 900 с шагом 100, причем чем больше значение, тем выше жирность. Установленное по умолчанию нормальное начертание шрифта эквивалентно значению 400, а стандартный полужирный текст - 700. Задание насыщенности шрифта может выглядеть следующим образом:

```
P {font-weight: 900;}
```

Стиль шрифта: свойство font-style

Свойство *font-style* определяет начертание шрифта как обычное, курсивное или наклонное. Данным начертаниям соответствуют значения свойства normal, italic и *oblique*. Когда для текста установлено курсивное или наклонное начертание, браузер обращается к системе для поиска подходящего шрифта. Если заданный шрифт не найден, браузер использует специальный алгоритм для имитации нужного вида текста. Результат и качество при этом могут получиться неудовлетворительными, особенно при печати документа.

Капитель: свойство font-variant

Капителью называется текст, набранный прописными буквами уменьшенного размера. Для создания такого эффекта используется свойство *font-variant* со значением small-caps. Особенность капители заключается в том, что заглавные и строчные буквы при ее использовании сохраняются. Браузер Internet Explorer до шестой версии отображает текст неправильно, заменяя все символы прописными. Остальные браузеры преобразуют символы вполне корректно.

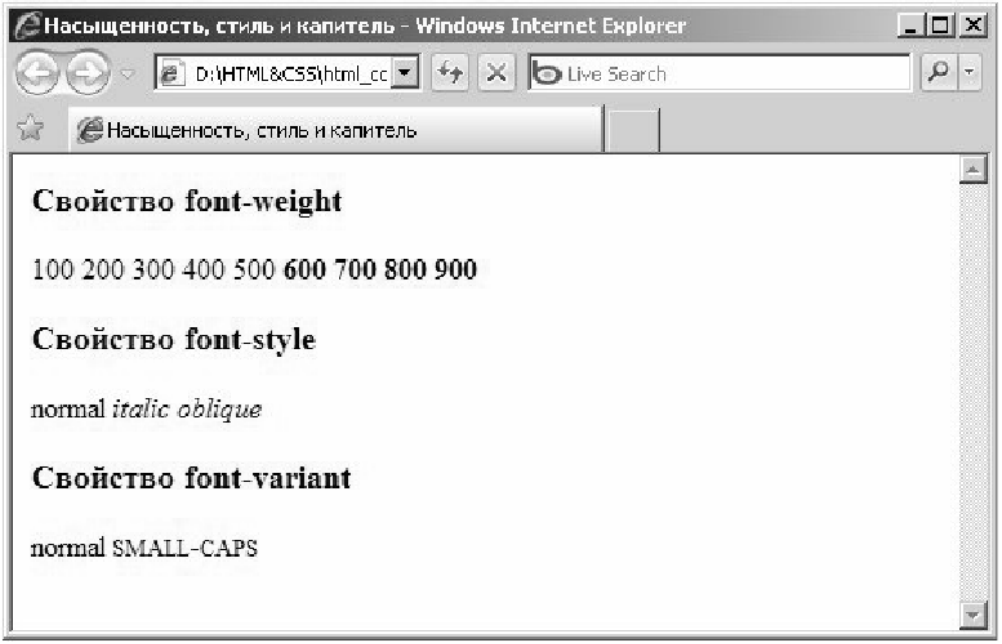


Рис. 11.3. Пример использования различных значений свойств font-weight, font-style и font-variant

Задание свойств текста

Преобразование текста: свойство text-transform

Обычно создатель сайта сам решает, какие буквы будут прописными, а какие строчными, исходя из правил правописания и собственных предпочтений. Тем не менее, процесс изменения регистров символов можно автоматизировать, используя свойство `text-transform`. Данное свойство может принимать четыре значения:

- `none` - текст пишется без изменений;
- `capitalize` - каждое слово будет начинаться с заглавного символа;
- `lowercase` - все символы становятся строчными (нижний регистр);
- `uppercase` - все символы становятся прописными (верхний

регистр).

Например, следующее правило указывает, что заголовок `H1` должен выводиться прописными буквами:

```
H1 {text-transform: uppercase;}
```

Автоматическое изменение регистра символов удобно задавать для аббревиатур, названий элементов, заголовков и других элементов текста, где требуется писать прописными или строчными символами.

Украшение текста: свойство text-decoration

Свойство *`text-decoration`* позволяет задать тексту дополнительное оформление. Значениями данного свойства являются константы `none`, `underline`, `overline`, `line-through` и *`blink`*, позволяющие отобразить обычный текст, провести линию над, под или через текст, а также сделать текст мигающим. Пример использования различных значений данного свойства приведен на рисунке 11.4.

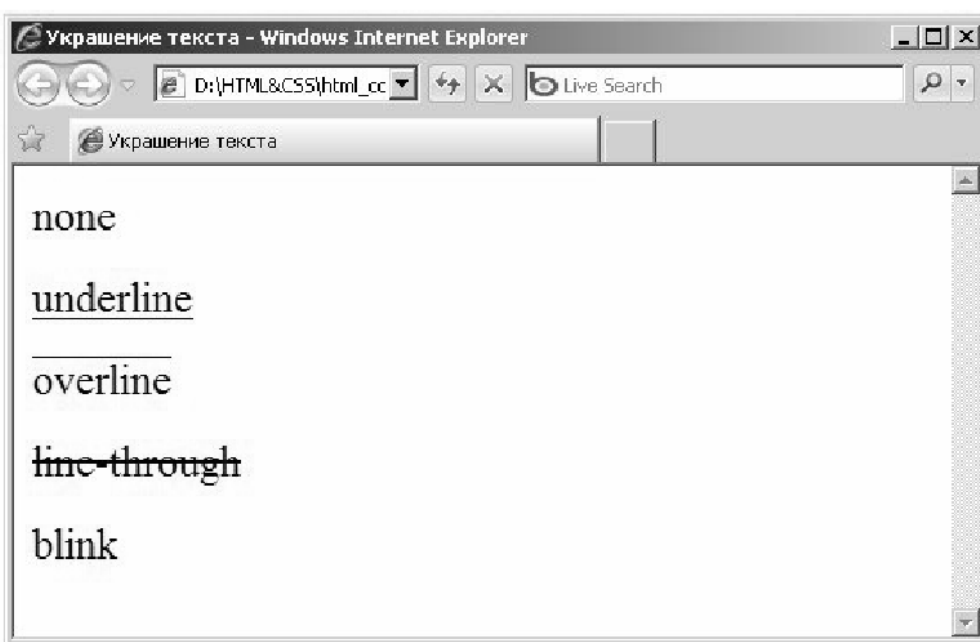


Рис. 11.4. Пример использования различных значений свойства text-decoration

Наиболее распространенным применением свойства *text-decoration* является изменение используемого по умолчанию подчеркивания ссылок. Например, следующее правило указывает, что ссылки должны подчеркиваться:

```
A:link {text-decoration: underline;}
```

Интервал между словами: свойство word-spacing

Чтобы задать интервал между словами в тексте, используется свойство `word-spacing`. Значения данного свойства можно задать с помощью ключевого слова `normal`, которое используется по умолчанию и задает стандартный интервал для текущего шрифта. Для задания интервала в дополнение к стандартному можно указать значение в любых доступных единицах CSS, причем значение может быть и отрицательным.

Так, следующее правило увеличивает интервал между словами в заголовке `H1` на `1em`:

```
H1 {word-spacing: 1em;}
```

Выравнивание текста: свойство text-align

Выравниванием называется размещение левого или правого края блока текста вдоль невидимой вертикальной линии. Для выравнивания текста используется свойство `text-align`. Допустимыми значениями данного свойства являются `left`, `right`, `center` и `justify`, задающие выравнивание по левому краю, по правому краю, по центру и по ширине соответственно.

Следующее правило устанавливает выравнивание по центру всех элементов, содержащихся в элементе `DIV`:

```
DIV {text-align: center;}
```

Интерлиньяж: свойство line-height

Интерлиньяжем называется расстояние между базовыми линиями близких друг к другу строк. При обычных обстоятельствах расстояние между строками зависит от вида и размера шрифта и автоматически определяется браузером. Но это значение может быть изменено с помощью свойства `line-height`. Заданное по умолчанию значение normal заставляет браузер вычислять расстояние между строками автоматически. Любое число больше нуля воспринимается как множитель от размера шрифта текущего текста. В качестве значений данного свойства допустимо также использовать любые единицы длины, принятые в CSS. Разрешается также использовать процентную запись, причем в этом случае за 100% принимается высота шрифта. Отрицательное значение межстрочного расстояния не допускается.

Интервал между буквами: свойство letter-spacing

Браузер автоматически подбирает интервалы между символами, исходя из размера и типа шрифта. В некоторых случаях необходимо подкорректировать расстояние между буквами. Для управления межбуквенным интервалом используется свойство `letter-spacing`. В качестве значений данного свойства могут использоваться любые единицы длины, принятые в CSS, однако рекомендуется использовать относительные единицы, основанные на размере шрифта (em и ex). В отличие от межстрочного интервала, свойство `letter-spacing` допускает использование отрицательного значения, однако в этом случае надо убедиться, что сохраняется читабельность текста.

Следующее правило увеличивает интервал между символами в заголовке `H1` на `0.5em`:

```
H1 {letter-spacing: 0.5em;}
```

Цвет и фоновые изображения CSS

В лекции рассматриваются основные свойства, управляющие цветом и характеристиками фона на веб-страницах.

Для оформления фона в CSS предусмотрено несколько свойств, позволяющих задавать цвет фона, фоновое изображение, позиционирование и укладку фонового изображения и многое другое. Однако, перед тем, как описывать основные свойства CSS для работы с фоном, опишем способ задания цвета переднего плана или цвета текста элемента.

Управление цветом переднего плана: свойство color

Цвет текста задается свойством color. Значения данного свойства можно задавать несколькими способами. Можно задать явное название цвета (например, red, yellow и др.), указать шестнадцатеричное значение или значение RGB. Шестнадцатеричное значение состоит из символа #, за которым следует шесть символов. Первая пара указывает уровень красного цвета, а вторая и третья – уровни зеленого и синего цветов соответственно, например, #FF0000. Можно определить цвет, используя значения уровня красной, зеленой и синей составляющей в десятичном исчислении, например, RGB(49, 151, 116). Также можно задавать цвет в процентном отношении. Например, следующее свойство делает все заголовки документа красными, а для задания свойства используется шестнадцатеричное значение:

```
H1 {color: #FF0000;}
```

Некоторые программы позволяют выбрать оттенок цвета, а затем определить его шестнадцатеричное или RGB-значение, как это показано на рисунке 12.1.

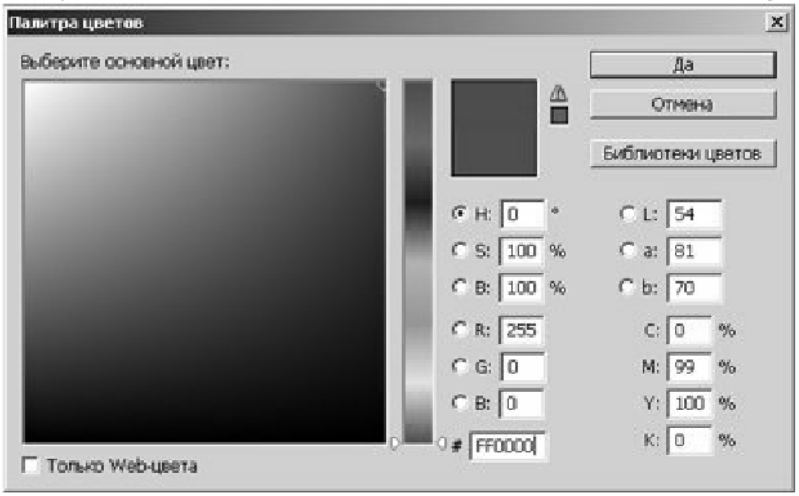


Рис. 12.1. Пример выбора значения цвета в программе Adobe Photoshop

Далее представлены основные свойства CSS, предназначенные для оформления фона элемента.

Управление цветом фона: свойство background-color

Для описания цвета фона элемента используется свойство `background-color`. Для изменения цвета фона всей веб-страницы свойство `background-color` нужно применить к элементу `BODY`. Это свойство можно применять и к другим элементам, в том числе к заголовкам и тексту. В следующем примере различные цвета фона применяются к элементам `BODY` и `H1`:

```
BODY {background-color: #8798C3;}

H1 {color: #1A284D;
    background-color: #4F659E;}
```

Результат применения данных правил представлен на рисунке 12.2.

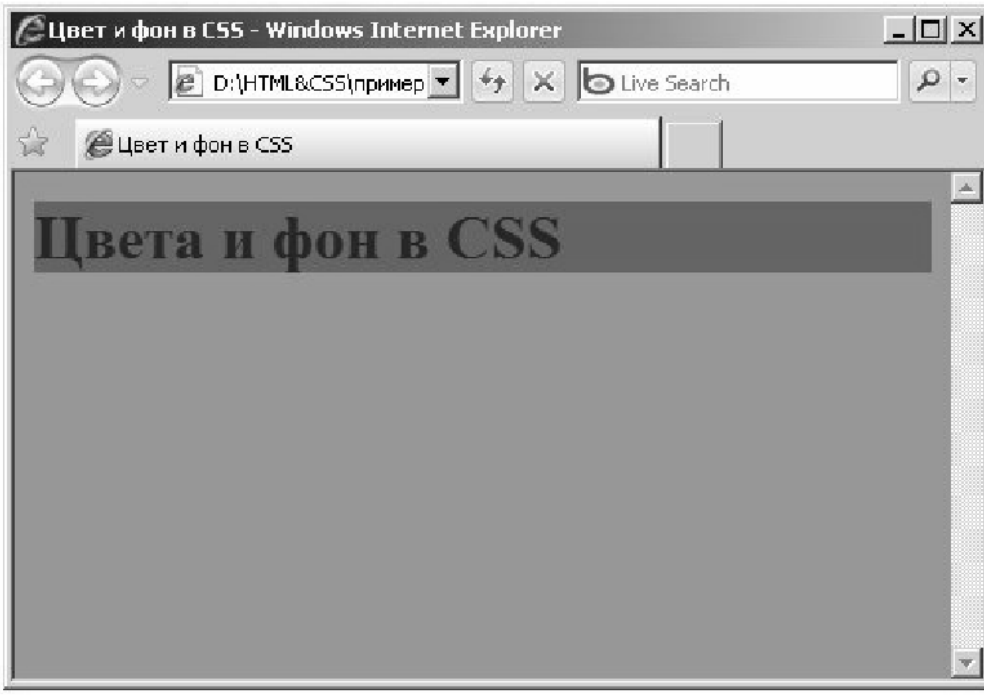


Рис. 12.2. Применение цвета фона для элементов BODY и H1

Применение фонового изображения: свойство background-image

Свойство `background-image` позволяет установить фоновое изображение или графический образ для элемента. В качестве значения данного свойства используется путь к графическому файлу, который указывается внутри конструкции `url()`. Например, следующее свойство задает в качестве фона страницы графическое изображение image.jpg:

```
BODY {background-image: url("image.jpg")}
```

Когда фоновое изображение не требуется, аргумент может принимать значение none.

Если одновременно для элемента задан цвет фона и фоновое изображение, то цвет фона будет отображаться до тех пор, пока фоновое

изображение не загрузится полностью или в случае, если изображение по какой-либо причине не доступно. В случае наличия в рисунке прозрачных областей, через них будет проглядывать фоновый цвет.

Повторение фонового изображения: свойство background-repeat

Если фоновое изображение меньше области элемента, то по умолчанию оно будет повторяться по горизонтали и по вертикали, стремясь заполнить всю выделенную область. Однако CSS представляет возможность управлять повторением фонового изображения, т.е. выбрать, в каком направлении оно должно повторяться. Данный выбор можно осуществить с помощью свойства *background-repeat*, которое может принимать значения r `epeat-x`, `repeat-y` и `repeat`, которое соответствуют повторению изображения по горизонтали, по вертикали и в обоих направлениях соответственно. Еще одно значение no-repeat является значением, которое требуется использовать, чтобы изображение не повторялось.

Например, горизонтальное повторение изображения может быть определено с помощью следующего свойства:

```
BODY {background-image: url("image.jpg");
    background-repeat: repeat-x;}
```

Результат применения данного свойства представлен на рисунке 12.3.

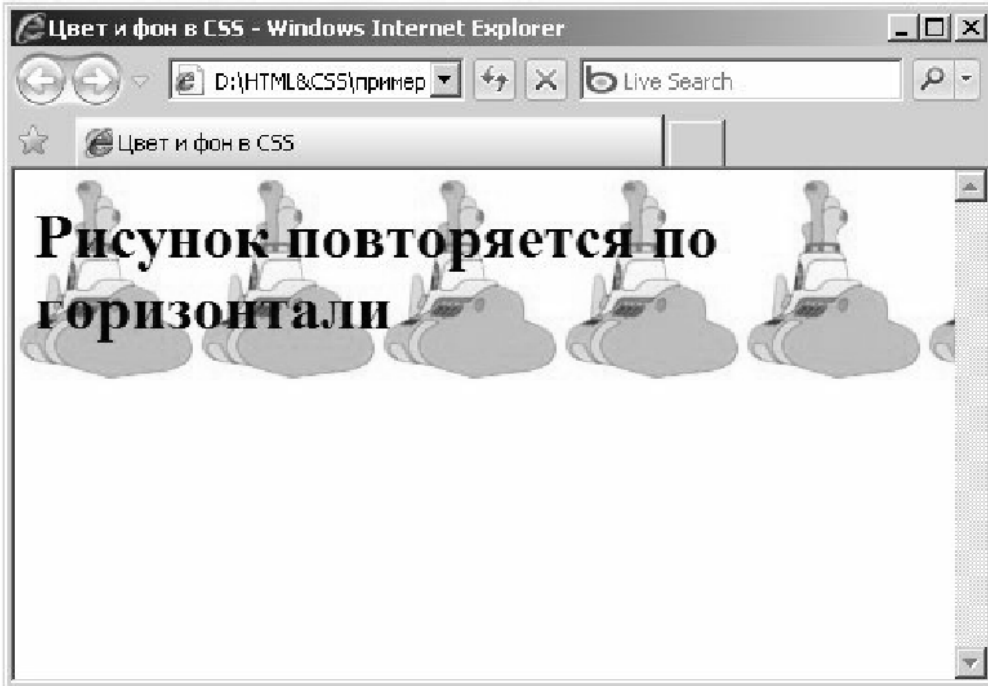


Рис. 12.3. Повторение фонового рисунка по горизонтали

Присоединение: свойство background-attachment

Свойство *background-attachment* определяет, фиксируется ли фоновый рисунок или прокручивается вместе с содержимым страницы. Данное свойство имеет два значения `scroll` и `fixed`. Значением по умолчанию является scroll, которое заставляет фоновое изображение прокручиваться вместе с содержимым элемента. Значение `fixed` фиксирует изображение, а содержимое страницы прокручивается. Например, следующий код фиксирует изображение:

```
BODY {
    background-color: #FFCC66;
    background-image: url("image.jpg");
    background-repeat: no-repeat;
    background-attachment: fixed;
}
```

Положение фонового изображения: свойство background-position

По умолчанию, фоновый рисунок располагается в левом верхнем углу экрана. Свойство background-position позволяет располагать фоновое изображение в любом месте. Есть много способов установить значение *background-position*. Тем не менее, все они представляют собой набор координат. Например, значение 200px 200px располагает фоновый рисунок на 200 пикселей слева и на 200 пикселей сверху в окне браузера. Координаты можно указывать в процентах от ширины экрана, в фиксированных единицах (пикселы, сантиметры, и т. п.), либо использовать символьные константы `top`, `bottom`, `center`, `left` и `right`. На рисунке 12.4 представлена иллюстрация использования различных координат. В данном примере фоновое изображение располагается в правом нижнем углу:

```
BODY {
    background-image: url("image.jpg");
    background-position: right bottom;
    }
```

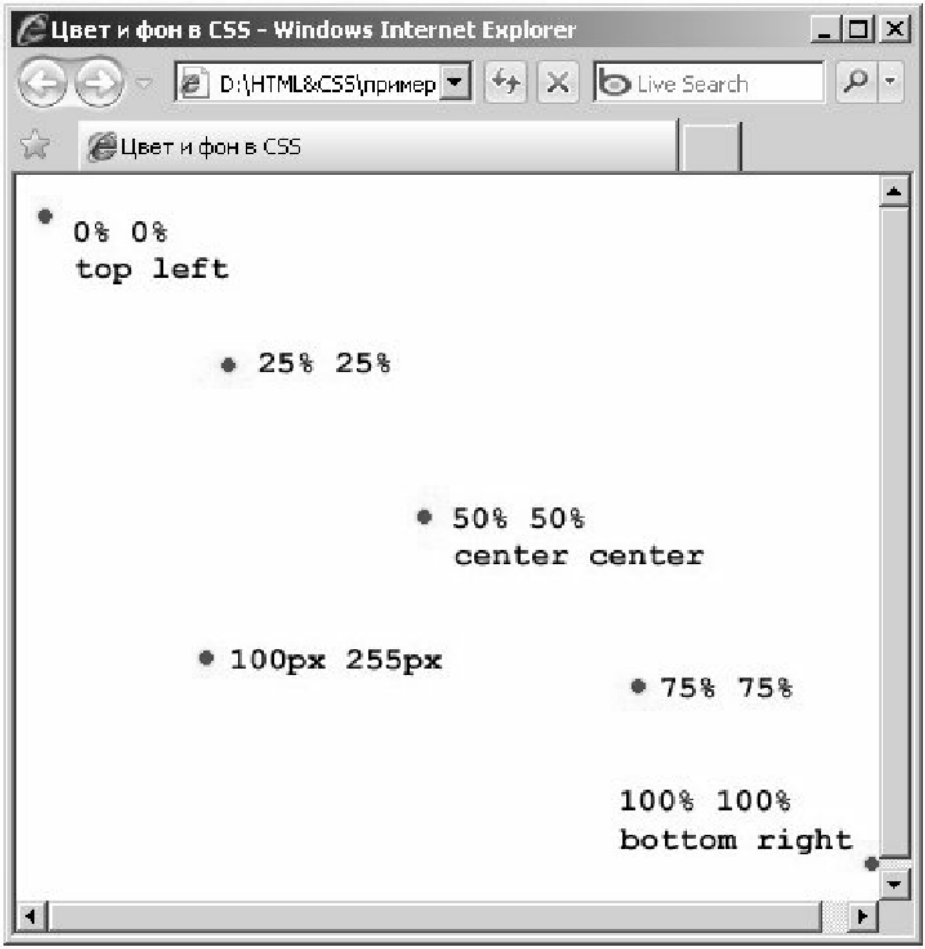


Рис. 12.4. Различные примеры позиций фонового изображения, использующие ключевые слова, проценты и пиксели

Свойство background

С помощью свойства `background` можно объединить несколько свойств и записывать стили в сокращённом виде, что облегчает чтение таблиц. Например, приведенные ниже строки позволяют установить цвет фона, фоновое изображение, вид повторения, присоединение и позицию изображения:

```
background-color: #FFCC66;
```

```
background-image: url("image.jpg");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right top;
```

Используя свойство `background`, того же результата можно достичь одной строкой кода:

```
background: #FFCC66 url("image.jpg") no-repeat fixed right top;
```

Для обеспечения межбраузерной совместимости и для организации и обслуживания таблицы стилей при объединении отдельных свойств фона в группу рекомендуется размещать свойства в следующем порядке: `background-color`, *`background-image`*, *`background-repeat`*, *`background-attachment`*, *`background-position`*.

Если какое-либо свойство отсутствует, то оно автоматически получает значение по умолчанию. Например, в приведенном ниже примере не заданы свойства *`background-attachment`* и *`background-position`*:

```
background: #FFCC66 url("image.jpg") no-repeat;
```

Поэтому этим свойствам будут присвоены значения по умолчанию - `scroll` и `top left`.

Практическое занятие 4

Целью занятия является закрепление материала лекций 11 и 12. В ходе практического занятия слушатель осваивает основные приемы оформления текстов с использованием CSS, а также методы работы с фоном и фоновыми изображениями.

Слушателю предлагается оформить созданные на предыдущих практических занятиях страницы, добавив цвет фона и фоновое изображение. Для фонового изображения необходимо предусмотреть такие свойства как направление повторения фонового изображения и поведение изображения при прокрутке. Также будет полезно оформить

фоном не только саму страницу (т.е. элемент BODY), но и некоторые структурные элементы, например, заголовки или параграфы.

При оформлении текста необходимо установить такие свойства как семейство шрифтов, размер, насыщенность и стиль, капитель и другие свойства, описанные в лекции 11. Данные свойства необходимо применять как к блокам текста, так и к группам символов. Слушателю предлагается поработать над цветовым оформлением текста и отдельных его символов. Выбор объектов применения данных свойств оставляется на усмотрение пользователя.

Задание потребует предварительной группировки элементов созданного HTML-документа в блоки с использованием элементов DIV и SPAN.

Модель компоновки CSS

В лекции вводится понятие модели компоновки CSS (боксовой модели) и рассматриваются основные ее возможности. Рассматриваются свойства CSS, которые управляют компоновкой элементов HTML, определяют их границы, поля, заполнение, ширину, высоту и др.

Модель компоновки CSS

Одним из самых важных инструментов дизайна является точное управление свободным пространством. Любое пустое пространство вокруг элемента невольно притягивает к нему взгляд, а для текста еще и обеспечивает его оптимальное восприятие. Пустой промежуток вокруг элемента выделяет его на веб-странице и позволяет отделить один элемент от другого. Однако таблицы стилей браузера, используемые по умолчанию, не решают задачу управления свободным пространством с достаточной точностью, поэтому разработчикам часто приходится использовать поля, границы, заполнение и другие свойства компоновки CSS. Все свойства компоновки CSS объединены в модель компоновки CSS, которая также называется боксовой моделью. Боксовая модель имеет детальные опции для определения полей, границ, заполнения и содержимого каждого элемента. Однако прежде чем рассматривать свойства для оформления боксов, необходимо немного поговорить о них самих.

Документ HTML состоит из множества перемешанных элементов. Когда такой документ изображается на экране компьютера или печатается на бумаге, эти элементы генерируют прямоугольные боксы. По умолчанию, встроенная таблица стилей в браузере заставляет элементы HTML блочного уровня (такие, как `P` и `DIV`) генерировать блочные боксы, в то время как строковые элементы (такие, как `STRONG` и `SPAN`) генерируют строковые боксы. Типом генерируемого бокса можно управлять, используя свойство `display`, которое будет рассмотрено ниже.

На рисунке показано, как построена боксовая модель:

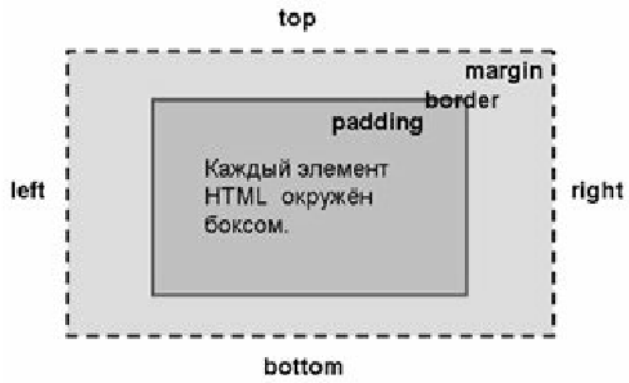


Рис. 13.1. Иллюстрация различных частей бокса элемента, помеченных соответствующими свойствами CSS

Поля элемента: свойство margin

Для управления значениями полей элементов предназначено свойство `margin`. Это универсальный параметр, в зависимости от числа значений, он устанавливает поля со всех сторон элемента или для каждой его стороны отдельно. Например, указание одного значения задаст равные поля вокруг элемента.

Допустимые значения обычно определяют в единицах измерения px или em (пикселях или em). В таблицах стилей, предназначенных для печати, в качестве единиц измерения можно использовать дюймы (in), сантиметры (cm) или пункты pt (пункты).

Для задания полей с разных сторон элемента предназначены производные от свойства `margin` - `margin-left`, `margin-right`, `margin-top` и `margin-bottom`, задающие значения левого, правого, верхнего и нижнего поля соответственно. Например, ниже представлен пример задания полей документа, т.е. элемента `BODY`. На рисунке 13.2 показано, какие поля необходимо определить и какие значения необходимо им придать.

```
BODY {
    margin-top: 100px;
```

```
margin-right: 70px;
margin-bottom: 40px;
margin-left: 40px;
}
```

Это же правило можно записать в следующем виде:

```
BODY {margin: 100px 70px 40px 40px;}
```

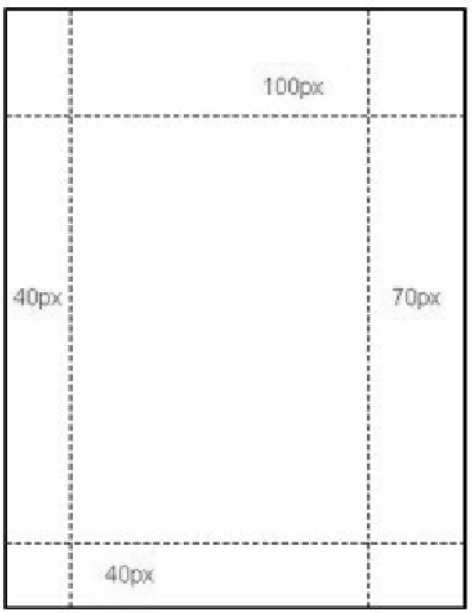


Рис. 13.2. Иллюстрация применения свойства margin

Таким же образом можно установить поля почти для любого элемента. Например, можно определить поля для всех параграфов на веб-странице:

```
P {margin: 5px 50px 5px 50px;}
```

Добавление границы: свойства border, border-width, border-style и border-color

Границы имеют разнообразное применение, например, как

декоративный элемент или для отделения двух объектов. Для задания границ применяется несколько способов, один из которых основан на использовании свойства border и его производных. Это свойство позволяет одновременно установить толщину, стиль и цвет границы вокруг элемента. Значения разделяются пробелами и могут идти в любом порядке. Браузер сам определит, какое значение соответствует нужному атрибуту:

```
P {border: 2px solid black;}
```

Данное правило позволяет создать вокруг прямоугольной области сплошную рамку черного цвета толщиной 2 пиксела. Первый аргумент в данном случае определяет толщину, второй - тип линии, а третий - ее цвет.

Когда значение в border отсутствует, выводимый элемент будет использовать значения по умолчанию: толщина границы будет определяться браузером, стиль границы будет `solid`, а цвет границы будет совпадать с цветом, используемым для рассматриваемого элемента.

Можно задать толщину, стиль и цвет любой из четырех сторон элемента, используя свойства `border-top`, *`border-bottom`*, *`border-left`* и `border-right`. Например, следующий пример создает нижнюю границу для элемента `H1` в виде красной сплошной линии толщиной 1 пиксел:

```
H1 {border-bottom: 1px solid red);}
```

Толщину, стиль и цвет также можно задать отдельно, используя соответствующие свойства.

Толщина границы: свойство border-width

Это свойство задает толщину одной или нескольких сторон границы. Сокращенное свойство *`border-width`* принимает значения в той же нотации, что и сокращенное свойство `margin`, за исключением того, что процентные значения не поддерживаются. Например, свойство

`border-width` может быть задано следующим образом:

```
TD {border-width: 1px 0 0 1px;}
```

Стиль границы: свойство border-style

Свойство border-style задает стиль линии и может принимать одно из восьми значений, представленных на рисунке 13.3.

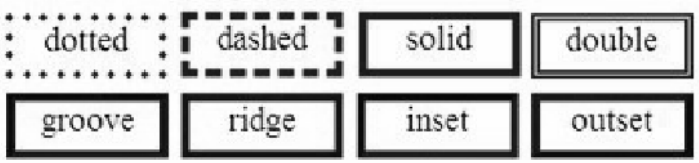


Рис. 13.3. Возможные значения свойства border-style

Цвет границы: свойства border-color

Для каждой границы можно задать любой цвет с помощью сокращенного свойства `border-color` или его уточнения. Например

```
TD {border-color: ##FF0000;}
```

Заполнение элемента: свойство padding

Заполнение определяет внутреннее расстояние между границей и содержимым элемента. Для изменения этой характеристики предназначено свойство `padding`. Оно позволяет задать расстояние между границей и содержанием для всех или определенных сторон элемента. Это свойство действует аналогично margin, поэтому итоговый результат зависит от числа аргументов. Для указания расстояний от разных сторон элемента можно воспользоваться свойствами `padding-left`, `padding-right`, `padding-top` и `padding-bottom`, которые управляют величиной расстояния слева, справа, сверху и снизу соответственно.

Основное предназначение заполнения - создать пустое пространство вокруг содержимого блочного элемента, например, текста, чтобы он не прилегал плотно к границе элемента. Использование заполения повышает читабельность текста и улучшает внешний вид страницы. В следующем примере показано использование заполнения для оформления текста:

```
DIV.first {padding: 20px;}

DIV.second {padding: 10px;
        padding-left: 50;}
```

В данном примере создается два блока с разными характеристиками. В первом блоке вокруг текста по вертикали и горизонтали с помощью свойства `padding` задается одинаковое поле со значением 20 пикселов. Во втором блоке поле слева увеличено через свойство *`padding-left`*.

Установка высоты и ширины элемента

Установить высоту и ширину элемента можно с помощью свойств `height` и `width` соответственно. Однако при применении данных свойств существуют некоторые особенности. Например, данные свойства не могут применяться к строковым элементам HTML, таким, как, например, `SPAN`, `STRONG` или `EM`.

Работа с потоком элементов

Типы блоков: свойство display

Каждый элемент в Рекомендациях HTML 4.01, который связан с основным контентом, имеет соответствующий строковый или блочный тип. Каждый тип определяет поведение компоновки по умолчанию различным образом. Например, последовательно идущие строковые элементы изображаются на общей базовой линии, в то время как блочные элементы всегда отделяются друг от друга и выводятся с

предшествующим и последующим разрывом строки.

Свойство `display` имеет три наиболее часто используемых значения - `block`, `inline` и `none` - два из которых имеют прямое отношение к соответствующим типам элементов. Данное свойство позволяет изменить поведение элементов (например, строковый элемент будет вести себя как блочный или наоборот). Свойство `display` со значением none может изменять представление данного элемента в документе. Например, с помощью следующего правила можно удалить посторонний фрагмент из заголовка:

```
.sectionNote {display: none;}
```

"Всплывающие" элементы: свойства float и clear

Элемент может "всплывать" вправо или влево с помощью свойства `float`. То есть бокс с его содержимым может смещаться к правому или левому краю в окне документа (или содержащего бокса). Если необходимо, например, чтобы текст окружал рисунок, как показано на рисунке 13.4, то фрагмент кода должен выглядеть следующим образом:

```
...
<STYLE type="text/css">
  #picture {
          float:left;
          width: 130px;}
</STYLE>
...
<DIV id="picture">
<IMG src="Beatles.jpg" alt="The Beatles">
</DIV>
<P>Far away, 80000 leagues below the sea, ... </P>
...
```

Чтобы рисунок смещался влево, а текст его окружал, необходимо определить ширину бокса, окружающего рисунок, и установить в свойстве `float` значение `left`.

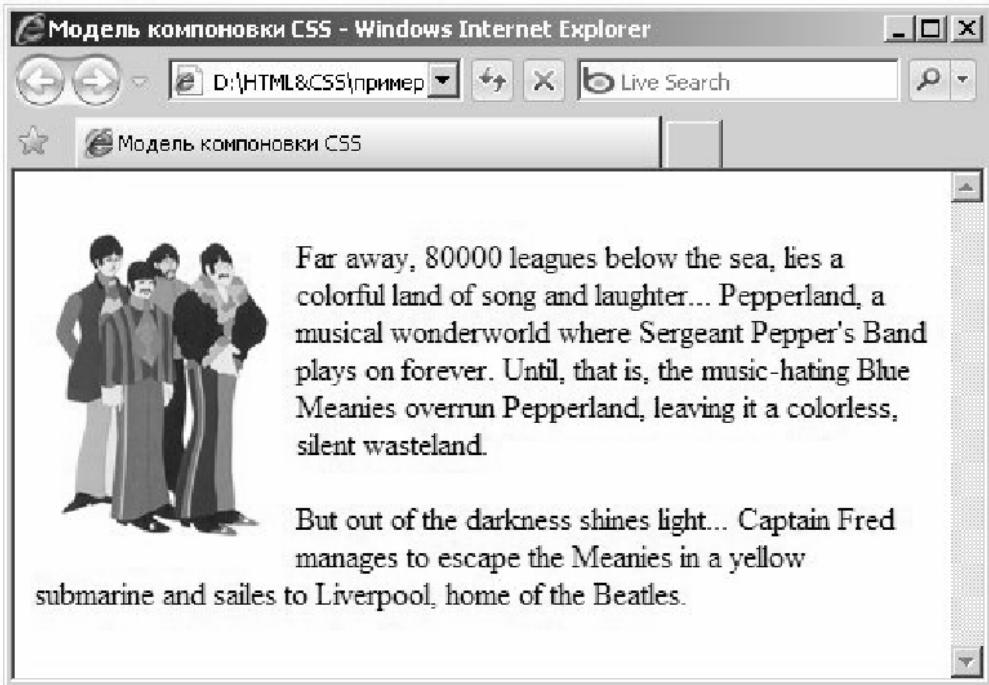


Рис. 13.4. Пример обтекания рисунка текстом

Свойство `clear` управляет поведением последовательности всплывающих элементов документа. По умолчанию, последовательные элементы смещаются вверх, заполняя доступное пространство, которое освобождается, если бокс смещается в сторону. Например, в предыдущем примере текст автоматически смещается вверх вдоль изображения. Свойство `clear` может иметь значения `left`, `right`, `both` или `none`.

Оформление списков и ссылок с помощью CSS

В лекции рассматриваются основные подходы для оформления списков и ссылок средствами CSS.

Оформление списков

Базовые маркеры и числа: свойство list-style-type

Для управления видом маркера используется свойство `list-style-type`. Данное свойство применяется и к маркированному (элемент HTML UL) и нумерованному (элемент HTML OL) спискам, однако аргументы для этих двух видов списка различаются. Для маркированного списка используются аргументы circle, disc и square, которые устанавливают маркер в виде незакрашенного кружка, закрашенного кружка и квадрата соответственно. Для нумерованного списка аргументов свойства `list-style-type` намного больше, и с ними можно самостоятельно ознакомиться в Спецификации CSS2.1. На рисунке 14.1 представлены несколько распространенных типов маркированного и нумерованного списков. Аргумент `none` устанавливает тип маркера, как у родительского элемента. По умолчанию данное свойство принимает значение `disk` для маркированного списка и `decimal` для нумерованного списка.

Например, следующие правила задают для всех маркированных списков на сайте квадратные маркеры, а для нумерованных - десятичные числа:

```
ul li {list-style-type: square;}
ol li {list-style-type: decimal;}
```

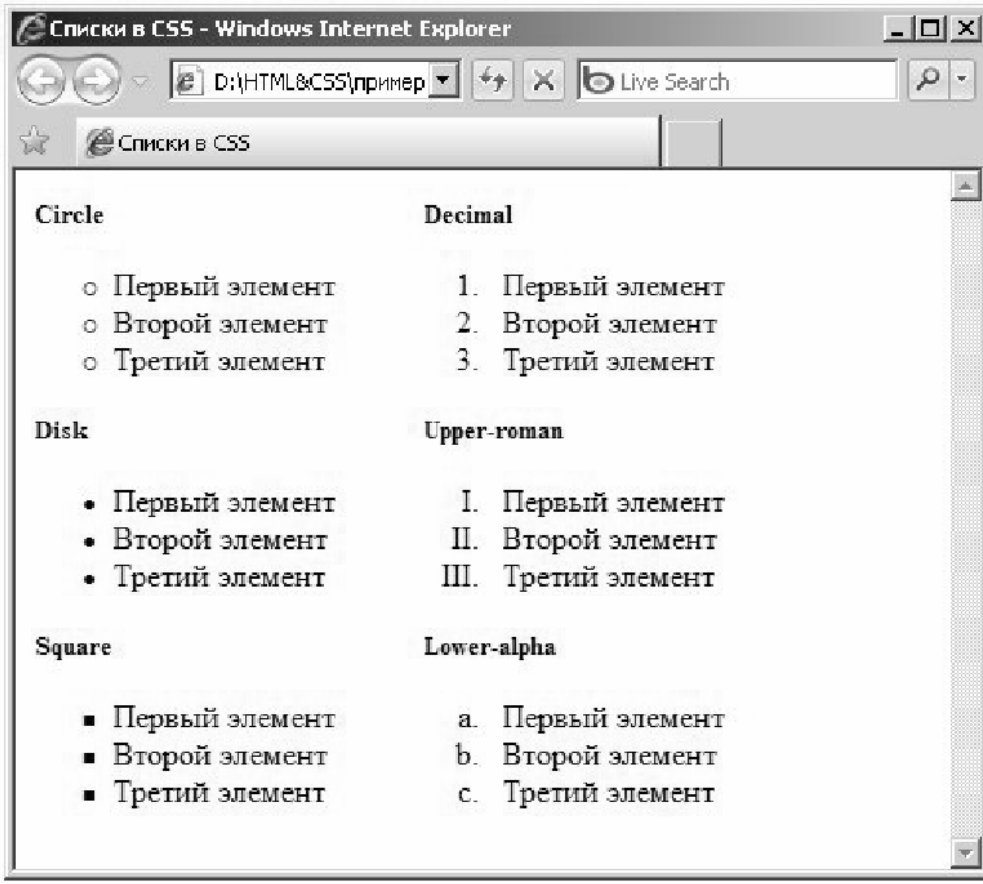


Рис. 14.1. Распространенные стили списков

Маркеры-изображения: свойство list-style-image

Хотя количество значений атрибута свойства `list-style-type` для элемента `UL`, т.е. для маркированного списка, ограничено тремя, это не значит, что в распоряжении разработчика или дизайнера всего три вида маркера. CSS позволяет установить в качестве маркера любое подходящее изображение с помощью свойства `list-style-image`. В качестве аргумента используется относительный или абсолютный адрес графического файла, содержащего изображение, которое должно служить в качестве маркера. В следующем примере для каждого элемента списка в качестве маркера устанавливается изображение marker.png:

```
UL {list-style-image: url("marker.png");}
```

Этот атрибут наследуется, поэтому для отдельных элементов списка для восстановления первоначально вида маркера используется значение атрибута none, которое отменяет изображение в качестве маркера для родительского элемента.

Следует заметить, что это свойство имеет ограниченные возможности позиционирования для фонового изображения и в некоторых ситуациях вообще не работает в браузере Internet Explorer. Поэтому значительно более распространенной практикой является просто задание фонового изображения в пунктах списка.

Прежде всего, необходимо определить для списка отсутствие маркера и удалить поле и заполнение:

```
UL {
   margin: 0;
   padding: 0;
   list-style-type: none;
  }
```

Затем можно добавить фоновое изображение для каждого пункта списка, некоторое заполнение слева и снизу, чтобы сместить текст, позволяя вывести изображение, и растянуть пространство между пунктами списка:

```
UL LI {
      background: #fff url("icon.gif") 0 3px no-repeat;
      padding: 0 0 5px 15px;
     }
```

Размещение маркера: свойство list-style-position

Существует два способа размещения маркера относительно текста: маркер выносится за границу элементов списка или обтекается текстом. Чтобы управлять положением маркера относительно текста, применяется свойство *list-style-position*. Это свойство имеет

два значения: `outside` и *`inside`*. Значение `outside` (значение по умолчанию) размещает маркеры за пределами текстового блока. Значение inside включает маркеры в текстовый блок и отображает их в элементе списка. Пример использования этих значений показан на рисунке 14.2.

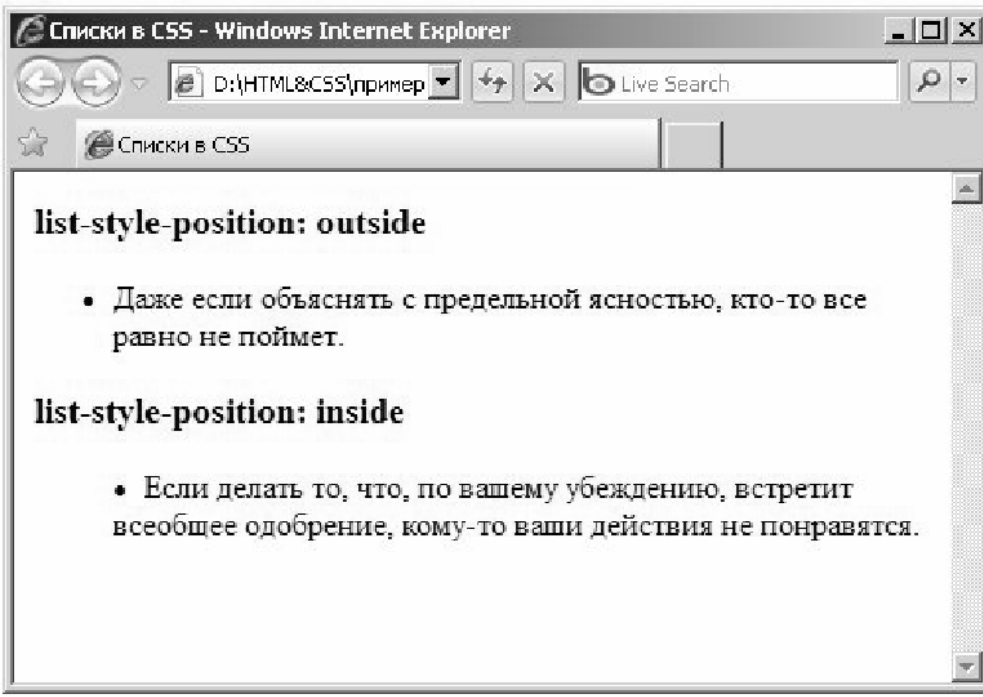


Рис. 14.2. Варианты размещения маркера

Оформление списков определений

Обычно списки определений не требуют большого внимания, за исключением задания стиля DT (обычно жирный текст) и управления отступом определений:

```
DT {font-weight: bold;}
DD {margin-left: 2em;}
```

В CSS предусмотрена возможность одновременно задать стиль маркера, его положение, а также изображение, которое будет использоваться в

качестве маркера. Для этой цели используется свойство `list-style`. В качестве аргументов данного свойства могут выступать любые комбинации трех значений, определяющих стиль маркеров, в произвольном порядке. Значения разделяются между собой пробелом. Ни один аргумент не является обязательным, поэтому неиспользуемые значения можно опустить.

Оформление ссылок

Существует несколько общих правил, которые разработчик должен учитывать при создании веб-страниц. Эти правила основываются на ожиданиях пользователей относительно оформления и действия ссылок:

- пользователи ожидают, что ссылки отличаются от остального текста, представленного на веб-странице, и что ссылкой является именно подчеркнутый текст;
- пользователи ожидают, что ссылки реагируют при наведении на них курсора и видоизменяются после того, как их посетили.

Таким образом, при оформлении ссылок разработчик должен задавать оформление для всех состояний ссылок и использовать подчеркивание только для ссылок

Оформление состояния ссылок

Стили ссылок всегда должны быть заданы в таблице стилей в следующем порядке: `link`, `visited`, `focus`, `hover` и `active`. Если стили ссылок будут размещены в другом порядке, то настройки будут переопределять друг друга, и состояния ссылок не будут работать.

Для оформления различных состояний ссылок используются псевдоклассы `:link`, `:visited`, `:focus`, `:hover` и `:active`, которые добавляют к селектору элемента `A`:

```
A:link{}
A:visited{}
A:focus{}
```

```
A:hover{}
A:active{}
```

Если необходимо задать оформление для всех ссылок во всех состояниях, то можно оформлять непосредственно элемент `A`. Однако базовое правило должно быть определено в первую очередь:

```
A {}
A:link{}
A:visited{}
A:focus{}
A:hover{}
A:active{}
```

Такая запись полезна, если необходимо убрать используемое по умолчанию подчеркивание ссылок.

Управление поведением по умолчанию

По умолчанию, большинство браузеров задает для всех ссылок подчеркивание, а состояние фокуса клавиатуры создает вокруг ссылок рамку. Данное оформление можно заменить или вообще отключить.

Подчеркивание задается с помощью свойства `text-decoration`, рассмотренного в лекции 11. Напомним, что подчеркивание задается с помощью значения свойства text-decoration, равного underline:

```
A {text-decoration: underline;}
```

Можно отключить подчеркивание с помощью следующего правила:

```
A {text-decoration: none;}
```

Установленное по умолчанию подчеркивание является толстоватым и пересекает нижние выносные элементы строчных букв. Если необходимо сохранить стиль подчеркивания ссылок, но сделать подчеркивание тоньше и запретить пересечение нижних выносных элементов, можно использовать ложное подчеркивание.

Прежде чем создавать ложное подчеркивание, необходимо отключить подчеркивание всех состояний ссылок:

```
A {text-decoration: none;}
```

Отключив заданное по умолчанию подчеркивание, можно задать свое подчеркивание с помощью свойства `border-bottom`:

```
A:link {border-bottom: 1px solid #00c;}
```

Результат применения описанных выше свойств к состоянию ссылки представлен на рисунке 14.3. Для сравнения представлена также ссылка, оформленная по умолчанию.

Рис. 14.3. Ложное подчеркивание в действии

При использовании метода ложного подчеркивания необходимо следить за тем, чтобы было задано достаточно большое значение `line-height`, чтобы избежать наложения подчеркивания на следующую строку текста.

Ложное подчеркивание позволяет создавать дизайн, в котором состояния ссылок можно отличать не только по цветам. Задавая различный стиль подчеркивания, можно гарантировать, что пользователь сможет различить состояния ссылок даже в черно-белом представлении.

Изображения возле ссылок

Некоторые сайты используют изображения и символы для добавления информации о своих ссылках. Например, можно использовать стрелку для указания, что ссылка позволяет перейти на внешний сайт, или применить какой-либо символ, чтобы отметить посещенные ссылки. Такие эффекты легко создаются с помощью фоновых изображений.

Чтобы добавить изображение к внешним ссылкам, вначале необходимо определить принадлежность такой ссылки к некоторому классу, в приведенном ниже примере это класс external:

```
<A href="http://www.somewhere.com /"
 class="external">external link</A>
```

Затем необходимо задать фоновое изображение для этого класса:

```
A.external {
  background: #fff url("arrow.gif") center right no-repeat;
  padding-right: 30px;
}
```

Этот пример будет применять выбранное изображение ко всем экземплярам посещенных ссылок во всех состояниях. Если необходимо выделять с помощью изображений только непосещенные внешние ссылки, то можно объединить классы и псевдоклассы состояний ссылок следующим образом:

```
A.external:link{
  background: #fff url("arrow.gif") center right no-repeat;
  padding-right: 30px;
}
```

Объединение классов и состояний открывает широкие возможности для ссылок, в чем слушателю предлагается убедиться самостоятельно.

Практическое занятие 5

Целью практического занятия является закрепление материала лекции 14. В ходе практического занятия слушатель осваивает основные методы оформления ссылок и списков с помощью CSS.

Слушателю предлагается средствами CSS установить базовые маркеры для нумерованных и маркированных списков, а также специальные маркеры, использующие графические изображения. Также рекомендуется создать горизонтальный список и список без маркеров.

Используя псевдоклассы, предлагается оформить различные состояния ссылок с помощью различного вида подчеркиваний, а также установить индивидуальные цвета. Необходимо установить для ссылок, указывающих на ресурсы Сети, определенное графическое изображение и предусмотреть смену изображения для уже посещенной ссылки. Также необходимо предусмотреть изменения вида ссылки при наведении на нее курсора: сделать ссылку перечеркнутой и изменить цвет фона ссылки.

Оформление таблиц с помощью CSS

В лекции рассматриваются свойства CSS для управления основными параметрами таблицы.

Таблицы обеспечивают возможность расположения многомерных данных по строкам и столбцам. Структура и содержание таблицы описываются с помощью элементов HTML, а ее оформление задается с помощью правил CSS. Визуальное оформление таблиц состоит в задании правил отображения заголовков и ячеек таблицы, их выравнивания относительно друг друга, изображения рамок вокруг них и т.д. Далее будут рассмотрены основные приемы визуального оформления таблиц.

Для дальнейшего изложения будет использоваться следующая нестилизованная таблица, представленная на рисунке 15.1.

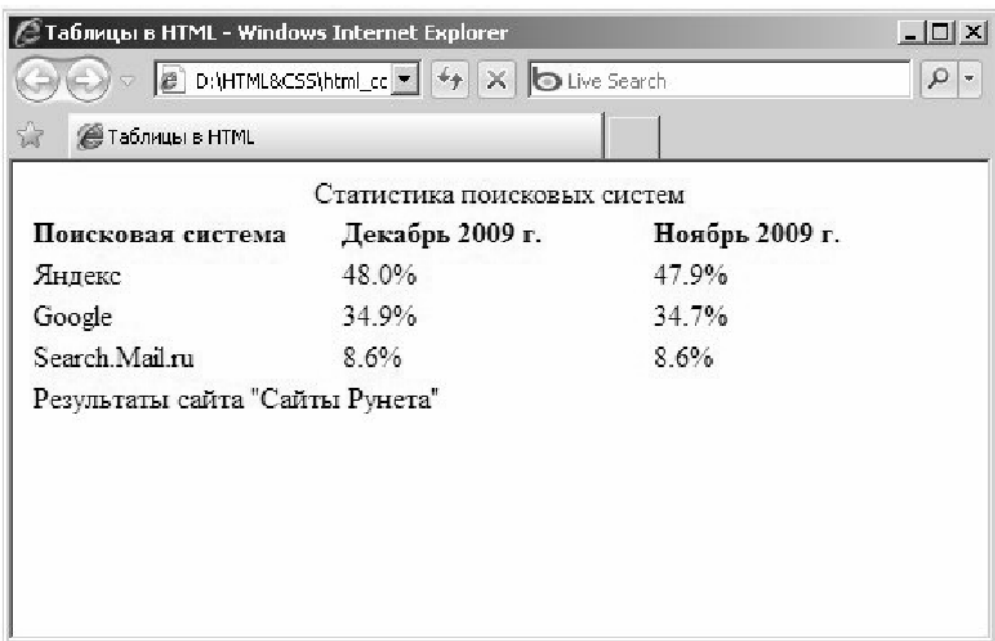


Рис. 15.1. Нестилизованная таблица

Ширина таблицы и ячеек

Для определения ширины таблицы и ячеек используется свойство width, в качестве значения которого принимаются любые единицы длины, принятые в CSS (пикселы, дюймы, пункты и др.) При использовании процентной записи ширина элемента вычисляется в зависимости от ширины родительского элемента, либо окна браузера. По умолчанию, браузер использует настройку `TABLE {width: auto;}`, что приводит к выводу таблицы во всю ширину окна браузера.

Следующие правила задают ширину таблицы в 100% доступной ширины, и ширину ячеек таблицы по 33% для каждой:

```
TABLE {width: 100%;}
TH, TD {width: 33%;}
```

Браузеры неодинаково работают с шириной, кроме того, результат отображения зависит от используемого `DOCTYPE`. Так, в Internet Explorer при использовании переходного `DOCTYPE` или при его отсутствии, если содержимое превышает заданную ширину, то блок изменяет свои размеры, подстраиваясь под содержимое. В противном случае ширина блока равна значению width. В то время, как для строгого `DOCTYPE` ширина формируется путем сложения значений `width`, `padding`, `margin` и `border`. Если содержимое блока не помещается в заданные размеры, оно отображается поверх.

Выравнивание в таблице

Часто необходимо менять установленные по умолчанию настройки выравнивания текста в таблице. CSS позволяет задавать выравнивание текста в таблице по горизонтали и вертикали. Для этой цели служат свойства `text-align` и *`vertical-align`*. Два данных свойства подробно описаны в лекции 13, поэтому ниже приведен лишь пример применения данных свойств к оформлению текста в ячейках таблицы:

```
TH, TD {
  width: 33%;
  text-align: left;
  vertical-align: top;
}
```

Отображение границ

Чтобы четко отделить содержимое одной ячейки от другой, к ним добавляются границы. За создание границ отвечает атрибут `border` элемента `TABLE`, который определяет толщину рамки. Однако, рамки, созданные с помощью данного атрибута, получаются разными по своему виду в каждом браузере. Чтобы этого избежать, рекомендуется пользоваться CSS-свойством `border`, применяя его к таблице или ее ячейкам (элементам `TD` или `TH`).

Составное свойство `border` позволяет задать толщину, цвет и стиль границы всей таблицы и отдельных ее ячеек. Данное свойство и его производные подробно описаны в лекции 13, поэтому ниже приведен лишь пример использования данных свойств для оформления таблиц:

```
TABLE {border: 1px solid #000;}
TH, TD {border-left: 1px dashed #000;}
```

Поскольку границы создаются для каждой ячейки, то в местах соприкосновения ячеек получается граница удвоенной толщины. Самым простым способом устранения указанной особенности является применение свойства *`border-collapse`*, которое устанавливает, как именно отображать границы вокруг ячеек. Данное свойство имеет два значения: *`collapse`* и `separate`. Значение *`collapse`* заставляет браузер анализировать места соприкосновения ячеек в таблице и убирать в ней двойные линии. При этом между ячейками остается только одна граница, одновременно принадлежащая обеим ячейкам. То же правило соблюдается и для внешних границ, когда вокруг самой таблицы добавляется рамка. При использовании значения `separate`, которое устанавливается по умолчанию, вокруг каждой ячейки отображается своя собственная рамка, а в местах соприкосновения ячеек показываются сразу две линии.

Различия между двумя значениями свойства *`border-collapse`* представлены на рисунке 15.2. Первая таблица соответствует границам удвоенной толщины и задается правилом

```
TABLE {
```

```
    border: 1px solid #000;
    border-collapse: separate;}
```

Вторая таблица соответствует схлопнувшимся границам и задается правилом

```
TABLE {
    border: 1px solid #000;
    border-collapse: collapse;}
```

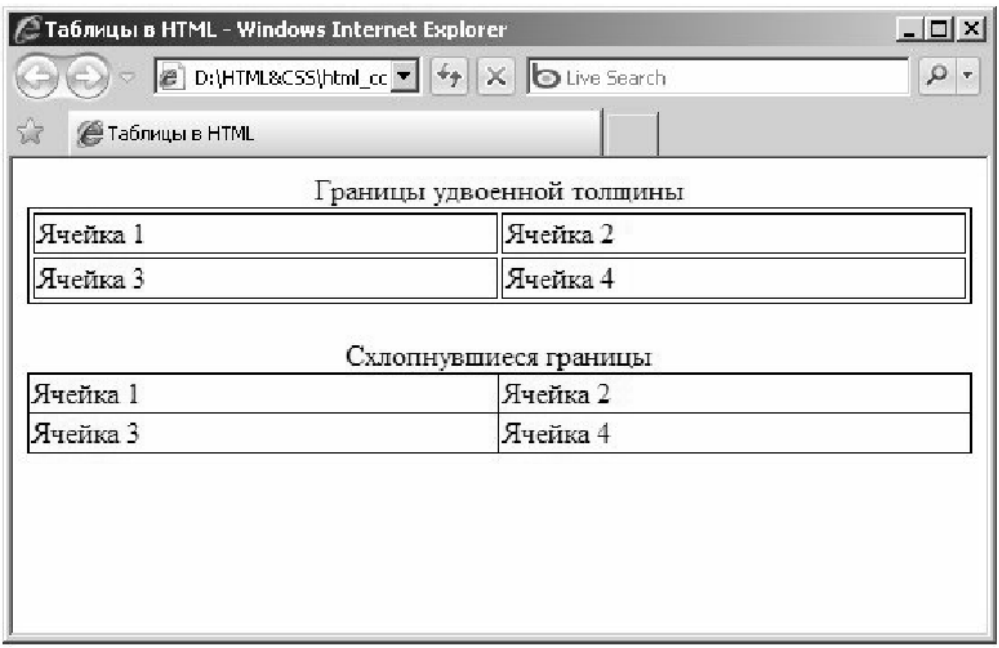


Рис. 15.2. Результат применения различных значений свойства border-collapse

Когда задается схлопывание границ, необходимо помнить, что это может создавать проблемы, если к границам смежных ячеек были применены различные стили оформления. Схлопывание границ с различными стилями приводит к конфликтам, которые разрешаются согласно правилам разрешения конфликтов границ таблиц спецификации CSS2 (ссылка: http://www.w3.org/TR/REC-CSS2/tables.html#border-conflict-resolution - http://www.w3.org/TR/REC-CSS2/tables.html#border-conflict-resolution). Данные правила определяют, какие стили "выигрывают" при схлопывании границ.

Расстояние между ячейками

CSS позволяет также увеличивать расстояния между ячейками, используя свойство *border-spacing*. Данное свойство задает расстояние между границами ячеек в таблице и не действует в случае, когда для таблицы установлен параметр *border-collapse* со значением *collapse*. Одно значение устанавливает одновременно расстояние по вертикали и горизонтали между границами ячеек. Если значений два, то первое определяет горизонтальное расстояние, а второе - вертикальное.

Следующие правило позволяет установить расстояние между границами ячеек 150 пикселей по горизонтали и 20 пикселей по вертикали:

```
TABLE {
    border-collapse: separate;
    border-spacing: 150px 20px;}
```

Браузер Internet Explorer до восьмой версии не обрабатывает свойство *border-spacing*, поэтому пользоваться данным свойством надо с осторожностью.

Заполнение

Для ячеек, которые имеют границу, можно добавить свободное пространство между границами ячеек и их содержимым. Для этого используется свойство `padding`.

Данное свойство рассмотрено в лекции 13, поэтому ниже будет приведен пример применения данного свойства к оформлению ячеек:

```
TH, TD {
  border: 1px solid #000;
  border-collapse: collapse;
  padding: 0.3em;
}
```

Размещение заголовка таблицы

По умолчанию, заголовок таблицы размещается вверху таблицы. Однако в некоторых браузерах возможно переместить заголовок таблицы в другое место с помощью свойства *caption-side*. Значениями данного свойства являются top, bottom, left и right, размещающие заголовок вверху, внизу, слева или справа таблицы. В некоторых браузерах, как, например, Internet Explorer, доступны только два значения top и bottom. Следующее правило поместить заголовок таблицы под ней:

```
CAPTION {caption-side: bottom;}
```

Аналогичного результата можно достичь, используя свойство

```
TABLE {caption-side: bottom;}
```

Шаблоны таблиц

К таблицам рекомендуется применять определенное оформление, что поможет не только вписать их в общий дизайн веб-страницы, но и представить информацию в более наглядном виде. Далее представлены несколько приемов оформления таблиц с помощью стилей.

Простой дизайн

Широко используемым вариантом дизайна для таблиц является выделение ячеек и строк с помощью фона. Так, в представленной ниже таблице заголовок таблицы выделен путем использования белого текста на темном фоне (так называемая выворотка), а для заголовков столбцов таблицы задан серый фон:

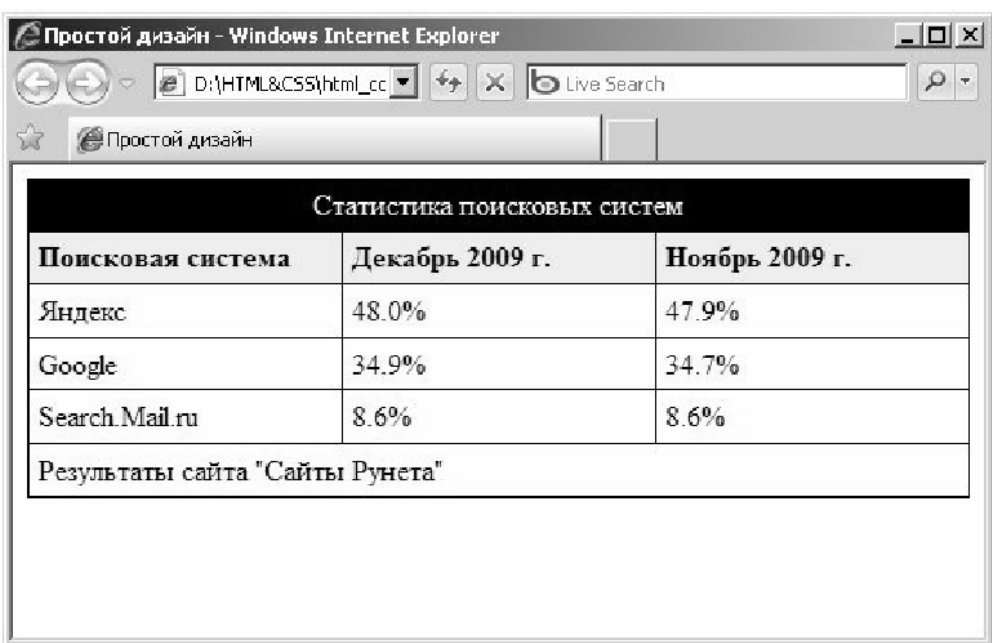
```
TABLE {
  width: 100%;
  border: 1px solid #000;
  border-collapse: collapse;
}
```

```
TH, TD {
  width: 33%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  padding: 0.3em;
  caption-side: bottom;
}

CAPTION {
  padding: 0.3em;
  color: #fff;
  background: #000;
}

TH {
  background: #eee;
}
```

Полученный результат представлен на рисунке 15.3.

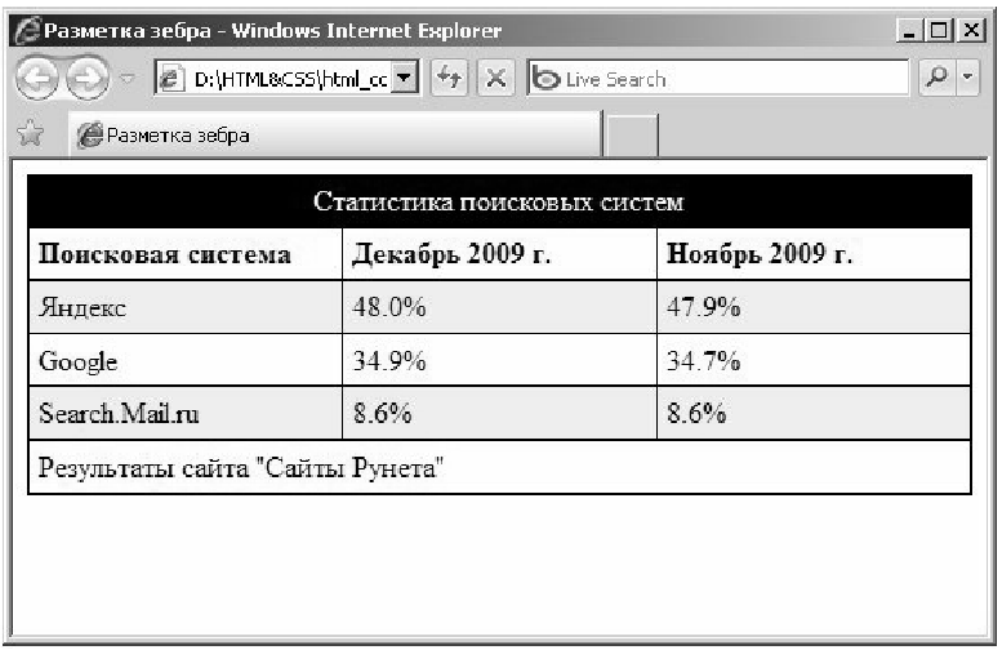


Статистика поисковых систем		
Поисковая система	**Декабрь 2009 г.**	**Ноябрь 2009 г.**
Яндекс	48.0%	47.9%
Google	34.9%	34.7%
Search.Mail.ru	8.6%	8.6%
Результаты сайта "Сайты Рунета"		

Рис. 15.3. Таблица с простым дизайном

Разметка "зебры"

Для удобного представления данных в таблице можно сделать строки таблицы чередующимися, чтобы цвет фона четных и нечетных строк различался. Такая разметка обычно называется "зеброй". Хотя и существуют сомнения в отношении того, насколько данная разметка действительно облегчает восприятие информации, она является популярным стилем оформления. Пример использования данного оформления представлен на рисунке 15.4.



Статистика поисковых систем		
Поисковая система	**Декабрь 2009 г.**	**Ноябрь 2009 г.**
Яндекс	48.0%	47.9%
Google	34.9%	34.7%
Search.Mail.ru	8.6%	8.6%
Результаты сайта "Сайты Рунета"		

Рис. 15.4. Таблица с разметкой "зебры"

Для изменения цвета фона у определенных строк прежде всего необходимо ввести классы для четных и нечетных строк:

```
...
<TABLE>
  <TR class="odd">
    ...
```

```
    <TR class="even">
      ...
</TABLE>
```

Затем необходимо добавить селектор для задания фона всех ячеек в строках, относящихся к заданным классам, например:

```
.odd th, .odd td {background: #eee;}
```

Неполные сетки

Для представления некоторых данных хорошо подходят менее структурированные таблицы. Простым вариантом является удаление вертикальных границ и заливки фона заголовка таблицы, как показано на рисунке 15.5.

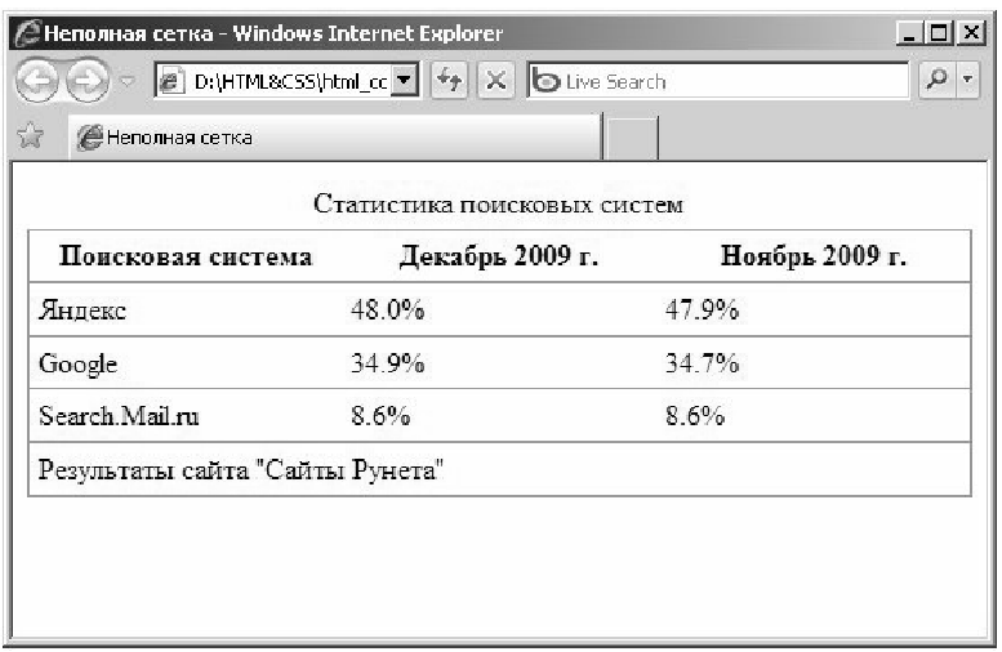


Рис. 15.5. Таблица с границами только на внешних краях и по нижнему краю каждой ячейки

Код CSS для этого представления может иметь следующий вид:

```
TABLE {
  width: 100%;
  border: 1px solid #999;
  text-align: left;
  border-collapse: collapse;
  margin: 0 0 1em 0;
  caption-side: top;
}

CAPTION, TD, TH {
  padding: 0.3em;
}

TH, TD {
  border-bottom: 1px solid #999;
  width: 25%;
}
```

Можно удалить все границы, за исключением верхней и нижней, чтобы определить только основное содержимое таблицы. Пример такого оформления представлен на рисунке 15.6.

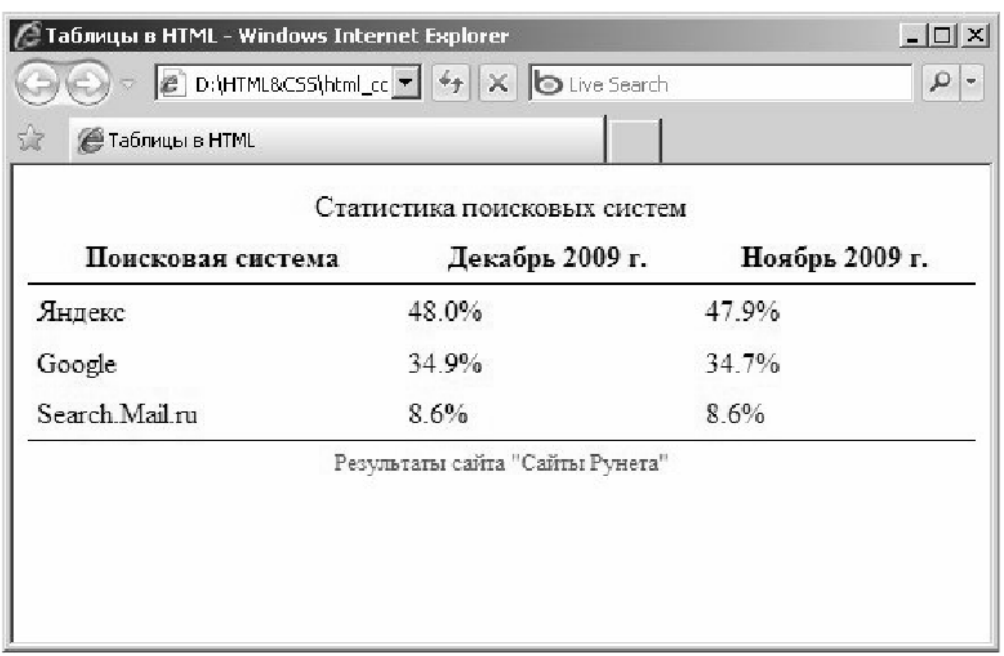


Рис. 15.6. Таблица с границами только вверху и внизу тела таблицы

Код CSS для такой таблицы будет следующим:

```
TABLE {
  width: 100%;
  text-align: left;
  border-collapse: collapse;
  margin: 0 0 1em 0;
  caption-side: top;
}

CAPTION, TD, TH {
  padding: 0.3em;
}

TBODY {
  border-top: 1px solid #000;
  border-bottom: 1px solid #000;
}

TBODY TH, TFOOT TH {
  border: 0;
}

TFOOT {
  text-align: center;
  color: #555;
  font-size: 0.8em;
}
```

Практическое занятие 6

Целью практического занятия является закрепление материала лекции 15. В ходе практического занятия слушатель осваивает основные приемы оформления таблиц с помощью CSS.

Слушателю рекомендуется последовательно применить к созданной на практическом занятии 2 таблице оформление, добавляя ко всей таблице

и ячейкам индивидуальное оформление, включающее толщину и стиль границы, заполнение и выравнивание текста в ячейках, высоту и ширину и т.д.

Слушателю рекомендуется также применить к созданной таблице основные шаблоны оформлений, описанные в лекции.

Позиционирование в CSS

В лекции рассматриваются основные виды позиционирования элементов на странице, а также возможности CSS по созданию слоев.

Прежде чем рассматривать позиционирование, в двух словах напомним, что с точки зрения боксовой модели каждый элемент HTML представляет собой прямоугольник (бокс), для которого можно задать такие параметры как поля, границы и заполнения. Позиционирование определяет, где должен располагаться этот прямоугольник, а также как он должен влиять на элементы вокруг себя. При помощи позиционирования можно разместить любой элемент точно в нужном месте страницы. Вместе со всплывающими элементами, рассмотренными в лекции 13, позиционирование дает большие возможности для создания оригинального дизайна.

В основе позиционирования лежит представление окна браузера как системы координат. Любой бокс возможно расположить в этой системе координат где угодно. Например, представленное ниже правило позволяет расположить заголовок на расстоянии 100 пикселей от верхней границы документа и на 200 пикселей от левой границы документа:

```
H1 {
position:absolute;
top: 100px;
left: 200px;
}
```

Результат выполнения данного кода представлен на рисунке 16.1.

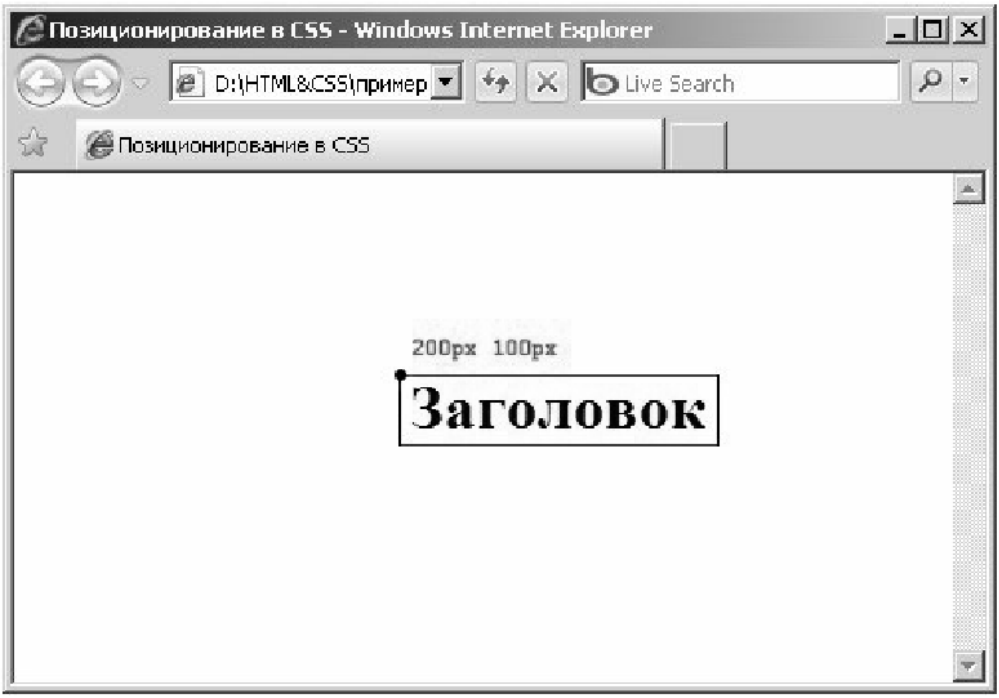


Рис. 16.1. Пример позиционирования заголовка

Из приведенного примера понятно, что для позиционирования элементов используется свойство `position`. Свойство `position` имеет четыре значения `static`, `relative`, `absolute` и `fixed`, которые определяют тип позиционирования и влияют на расположение элемента.

Статическое позиционирование

Значение `static` свойства `position` используется по умолчанию. Любой элемент со статическим позиционированием находится в общем потоке документа. Правила для его размещения определяются боксовой моделью. Блочные и строковые элементы размещаются по разным правилам, четко определенным в Спецификации CSS2.1.

По умолчанию, блочные боксы выкладываются вертикально сверху вниз в порядке появления их в разметке. Каждый бокс обычно занимает всю ширину документа и имеет разрыв строки перед и после себя.

Вертикальное расстояние между двумя блочными боксами управляется свойством *margin-bottom* первого блока и свойством *margin-top* второго бокса, причем вертикальные поля между двумя последовательными блочными боксами будут перекрываться таким образом, что расстояние между ними будет определяться не суммой двух полей, а большим из них.

Строковые боксы выстраиваются по горизонтали в том порядке, в котором они появляются в разметке, переходя на новую строку, только если исчерпано доступное горизонтальное пространство. В зависимости от свойства direction, строковые боксы будут располагаться либо слева направо (`direction: ltr`), либо справа налево (`direction: rtl`).

Множество строковых боксов, которые составляют строку на экране, заключаются еще в один прямоугольник, называемый линейным боксом. Линейные боксы выкладываются вертикально в своих предках блочного уровня без дополнительного пробела между ними. Высотой линейных боксов можно управлять с помощью свойства `line-height`. Для строковых боксов нельзя определить размеры и вертикальные поля.

Относительное позиционирование

Элемент со значением свойства `position`, равным `relative`, сначала размещается по правилам статического позиционирования. Но затем сгенерированный бокс смещается относительно своего положения в потоке, согласно значениям свойств `top`, `bottom`, `left` и `right`. Но при этом из потока он не исключается, а продолжает занимать там свое место. То есть сдвигается со своего места он только визуально, а положение всех боксов вокруг него никак не меняется. Это означает, что смещенный бокс может перекрывать боксы других элементов, так как они по-прежнему действуют, как если бы относительно позиционированный элемент остался там, где он должен был быть перед применением позиционирования. Как пример относительного позиционирования, попробуем сместить текст, заключенный в элемент `<SPAN>…</SPAN>` относительно его

начального положения на странице с помощью следующего фрагмента кода:

```
SPAN {
  position: relative;
  top: 10px;
  left: 10px;
  background-color: lime;
}
...

<P> В представленном <SPAN>фрагменте</SPAN> текста некоторые
    слова находятся не на своем месте и перекрывают другие слова.</P>
```

Результат выполнения данного кода представлен на рисунке 16.2. Блок теперь перекрывает следующую строку текста, а на его прежнем месте появилось пустое пространство.

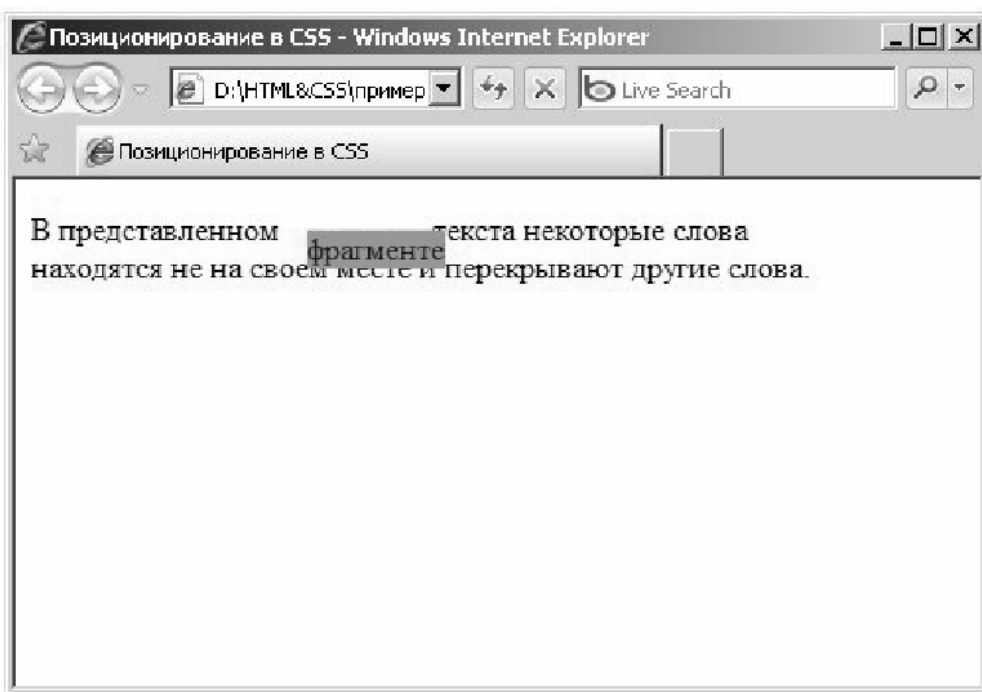


Рис. 16.2. Пример относительно позиционирования

Абсолютное и фиксированное позиционирование

Бокс с абсолютным позиционированием располагается по заданным координатам, а из того места, где он должен был бы быть, он удаляется, и в этом месте сразу начинают размещаться следующие боксы. Считается, что бокс исключается из потока.

Для абсолютного позиционирования элемента свойство position должно принимать значение `absolute`. А для задания положения размещения блока используются значения `left`, `right`, `top` и `bottom`. Все четыре свойства можно использовать одновременно для определения расстояния от каждого края позиционируемого элемента до соответствующего края браузера или родительского блока. Можно определить также позицию одного из углов абсолютно позиционируемого (например, используя `top` и `left`), а затем определить размеры бокса, используя `width` и `height`.

Представленный ниже код позволяет разместить четыре бокса в разных углах HTML-документа:

```
div#yellow1 {
  position:absolute;
  top: 10px;
  left: 10px;
}

div# yellow2 {
  position:absolute;
  top: 50px;
  right: 50px;
}

div# yellow3 {
  position:absolute;
  bottom: 10px;
  right: 10px;
}
```

```
div#yellow4 {
  position:absolute;
  bottom: 10px;
  left: 10px;
}
```

Результат применения описанных выше свойств представлен на рисунке 16.3.

Рис. 16.3. Пример абсолютного позиционирования

Фиксированное позиционирование действует подобно абсолютному, однако элемент с фиксированным позиционированием всегда располагается только относительно окна браузера и никогда не смещается при прокручивании веб-страницы. Отметим, что Internet Explorer версии 6 и более ранних версий не поддерживает фиксированное позиционирование.

Третье измерение веб-страницы

Страница сайта двумерна: для нее заданы ширина и высота. CSS позволяет добавить к веб-станице глубину (третье измерение) с помощью свойства `z-index`. Данное свойство позволяет создавать слои и располагать одни элементы поверх других.

Для создания слоев необходимо для каждого элемента задать значение свойства `z-index`, которое является своеобразным порядковым номером слоя, в котором находится данный элемент. Это значение может быть целым числом (которое может быть отрицательным) или одним из ключевых слов auto или `inherit`. Значением по умолчанию является `auto` или `0`. Элемент с большим значением свойства `z-index` перекрывает элемент с меньшим значением данного свойства. Ниже представлен код, располагающий игральные карты в порядке возрастания. Результат выполнения данного кода представлен на рисунке 16.4.

```
div#card1 {
    position:absolute;
    top: 50px;
    left: 150px;
    z-index: 1;
}

div#card2 {
    position:absolute;
    top: 70px;
    left: 170px;
    z-index: 2;
}

div#card3 {
    position:absolute;
    top: 90px;
    left: 190px;
    z-index: 3;
}
```

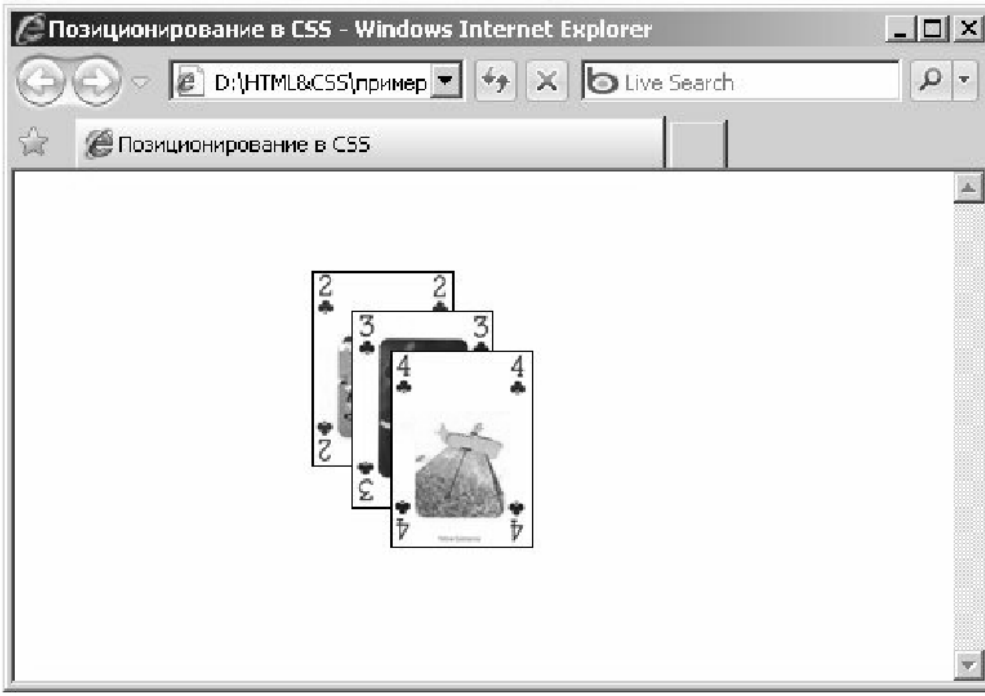


Рис. 16.4. Пример использования свойства z-index

Практическое занятие 7

Целью практического занятия является закрепление материала лекций 13 и 16. В ходе практического занятия слушатель осваивает основные свойства CSS, отвечающие за управление размещением структурных блоков и их содержимого.

Слушателю необходимо, используя свойства, управляющие компоновкой и позиционированием элементов, облагородить вид созданных веб-страниц: установить расстояния между блоками, оформить их границами, установить высоту и ширину блоков, обтекание графических изображений на странице текстом, используя "всплывающие" элементы, и т.д. С помощью различных типов позиционирования предлагается реализовать виды разметки веб-страниц, представленных в Приложении 5.

Практическое занятие 8

Целью практического занятия является освоение способов проверки HTML-документов и CSS-кодов.

Слушателю предлагается проверить созданные на предыдущих практических занятиях веб-страницы с помощью валидаторов W3C MarkUp Validator и W3C CSS Validator. В случае обнаружения ошибок, разобрать их и исправить.

Приложения

ПРИЛОЖЕНИЕ 1. Статья из книги "Физики смеются. Но смеются не только физики", М.: Совпадение, 206.

ФИЗИКИ СМЕЮТСЯ

Эффект Чизхолма

Основные закономерности срывов, неудач и затяжек

Ф. Чизхолм

Можно быть уверенным только в одном: ни в чем нельзя быть уверенным. Если это утверждение истинно, оно тем самым и ложно.

Древний парадокс

Подобно большинству научных открытий, общие принципы, сформулированные в настоящей работе, покоятся на экспериментальных данных, в болезненном процессе накопления которых участвовало несколько поколений наблюдателей. Мой приятный долг поблагодарить их за объемистые записки, в которых зарегистрировано все, что касается разного рода проволочек и провалов. Это целая гора данных, и до сих пор не было строгой теории, которая связала бы их в цельную науку.

Я не хочу сказать, что ощущался недостаток в попытках объяснить, что именно происходит, когда люди стараются довести какое-то дело до конца. Уже в средние века фортуну считали капризной богиней, и Шекспир был близок к сути дела, когда называл ее "непостоянной". Строго научное объяснение рассматриваемого феномена стало возможным только в наше время. Разница между ожидаемыми и получаемыми результатами, как оказалось, может быть записана в виде точного соотношения, называемого уравнением Снэйфу и содержащего постоянную Финэйгла. Организация под названием "Международная

организация инженеров-философов" уже опубликовала некоторые свои наблюдения: "Какой бы расчет вы ни делали, любая ошибка, которая может в него вкрасться, - вкрадется" и "Любое устройство, требующее наладки и регулировки, с максимальным трудом поддается и тому, и другому".

Остается только обобщить эти и многие другие наблюдения, сделанные в различных специальных областях, и записать стоящий за ними совершенно общий, всеобъемлющий принцип, которому подчиняется во всех случаях целенаправленная человеческая деятельность. Это обобщение я называю первым законом Чисхолма: ВСЕ, ЧТО МОЖЕТ ИСПОРТИТЬСЯ, - ПОРТИТСЯ.

Дальнейшие исследования показывают, что логика, которой подчиняются рассматриваемые нами явления, не Аристотелева, поскольку следствие первого закона Чисхолма имеет следующий вид: ВСЕ, ЧТО НЕ МОЖЕТ ИСПОРТИТЬСЯ, - ПОРТИТСЯ ТОЖЕ.

Все, кому приходится иметь дело с планами, проектами и программами, сразу заметят, какой порядок наводят эти простые утверждения в хаосе их собственных неудач. Действительно, эти обобщения отличаются той классической простотой, по которой мы сразу узнаем фундаментальные открытия типа

$E=mc^2$.

Администраторы, футбольные тренеры, генералы и жены, пытающиеся перевоспитать своих мужей, сразу вынуждены будут признать (каждый для своего поля деятельности) справедливость первого закона.

Давно известно, что в физических системах энтропия (мера беспорядка) стремится к увеличению, и что системы с большой энтропией теряют ее в борьбе с менее организованным окружением. Аналог этого второго закона термодинамики действует и в жизни. Достаточно вспомнить, как с течением времени нарастает беспорядок на письменном столе после новогодней уборки. Поэтому я формулирую в самом общем виде второй закон Чисхолма: КОГДА ДЕЛА ИДУТ ХОРОШО, ЧТО-ТО ДОЛЖНО ИСПОРТИТЬСЯ В САМОМ БЛИЖАЙШЕМ БУДУЩЕМ.

У этого закона также есть очевидное следствие: КОГДА ДЕЛА ИДУТ

ХУЖЕ НЕКУДА, В САМОМ БЛИЖАЙШЕМ БУДУЩЕМ ОНИ ПОЙДУТ ЕЩЕ ХУЖЕ.

Без труда можно получить и второе следствие: ЕСЛИ ВАМ КАЖЕТСЯ, ЧТО СИТУАЦИЯ УЛУЧШАЕТСЯ, ЗНАЧИТ, ВЫ ЧЕГО-ТО НЕ ЗАМЕТИЛИ.

По традиции фундаментальные научные законы объединяются по три, поэтому я поспешу сформулировать третий закон Чисхолма. Предварительная работа в этой области проведена многими лекторами, писателями, председателями комиссий и влюбленными, которые часто замечают, что люди слышат от вас вещи, которые вы им не говорили. Итак, обобщая: ЛЮБУЮ ЦЕЛЬ ЛЮДИ ПОНИМАЮТ ИНАЧЕ, ЧЕМ ЧЕЛОВЕК, ЕЕ УКАЗЫВАЮЩИЙ.

Следствие первое: ЕСЛИ ЯСНОСТЬ ВАШЕГО ОБЪЯСНЕНИЯ ИСКЛЮЧАЕТ ЛОЖНОЕ ТОЛКОВАНИЕ, ВСЕ РАВНО КТО-ТО ПОЙМЕТ ВАС НЕПРАВИЛЬНО.

Следствие второе: ЕСЛИ ВЫ УВЕРЕНЫ, ЧТО ВАШ ПОСТУПОК ВСТРЕТИТ ВСЕОБЩЕЕ ОДОБРЕНИЕ, КОМУ-ТО ОН НЕ ПОНРАВИТСЯ.

Учет законов Чисхолма как решающих факторов при планировании любого процесса должен понизить всеобщее нервное напряжение и решить национальную проблему перепроизводства адреналина.

Из книги "A Stress Analysis of a Strapless Evening Gown"

Englewood Cliffs, 1963.

ПРИЛОЖЕНИЕ 2. Фрагмент из книги "Полное собрание законов Мерфи", Мн.: ООО "Попурри", 2005

Законы Мерфи. Книга 1.

ДЕЛА КОНСТРУКТОРСКИЕ

ЗАКОН ТЕХНИЧЕСКИХ УСЛОВИЙ КЛИНШТЕЙНА

В технических условиях Закон Мерфи отменяет закон Ома.

ЗАКОН УТЕРЯННОГО ДЮЙМА

При разработке любого типа конструкции в пятницу после 16.40 ни один габаритный размер нельзя подсчитать точно.

Следствия.

1. При тех же условиях, если малые размеры даны с точностью до шестнадцатой дюйма, даже приблизительное определение габаритного размера невозможно.
2. В понедельник в 9.01 точное значение станет очевидным само по себе.

ЗАКОНЫ ПРИКЛАДНОЙ ПУТАНИЦЫ

1. Деталь, которую завод забыл поставить, составляет 75% поставок.

   Следствие.

   Завод не только забыл ее поставить, но и не изготовлял ее в течение 50% рабочего времени.

2. Доставка, которая обычно занимает один день, длится пять, если вы ждете товара.
3. Если вы добавили к графику две недели на непредвиденные задержки, не забудьте еще две на непредвиденные непредвиденные задержки.
4. В любой конструкции на самой важной детали, скорее всего, стоит неправильная метка.

   Следствия.

   - В любом наборе деталей с одинаковой меткой для сборки

найдется деталь, на которой эта метка лишняя.
- Это не обнаруживается до тех пор, пока вы не попытаетесь вставить ее туда, куда указывает метка.
- Никогда не спорьте с заводом-производителем об ошибке. Они очень тщательно проверяют акты о приемке продукции.

ДЕЛА МАШИННЫЕ

ЗАКОН МАСТЕРСКИХ ЭНТОНИ

Любой упавший инструмент закатывается в самый недоступный уголок мастерской.

Следствие.

По пути в этот уголок инструмент сперва непременно попадет вам по ноге.

ПЕРВЫЙ ЗАКОН ДЖОНСОНА

Любое механическое устройство отказывает в самый неподходящий момент.

ЗАКОН РАЗДРАЖЕНИЯ

Если во время работы вы убрали инструмент, который, вы уверены, больше не понадобится, в нем тут же возникает срочная потребность.

ЗАКОН САТТИНГЕРА

Если вставить вилку в розетку, прибор будет работать лучше.

ЗАКОН ШМИДТА

Любая вещь, если с ней долго возиться, обязательно сломается.

Артур Блох.

ПРИЛОЖЕНИЕ 3. Статья из книги "Физики смеются. Но смеются не только физики", М.: Совпадение, 206.

Но смеются не только физики…

Чем заняты физики

Шагая в ногу со временем, редакция стенгазеты "Импульс" Физического института АН СССР сформировала отдел социологических исследований. Сотрудники этого отдела провели опрос населения Москвы на тему "Чем заняты физики".

Группа населения	Опрошено	Ответили	Не знают	Ответы
Писатели-реалисты	11	7	4	Спорят до хрипоты в прокуренных комнатах. Неизвестно зачем ставят непонятные и опасные опыты на огромных установках
Писатели-фантасты	58	58	0	Работают на громадных электронных машинах, именуемых электронным мозгом. Работают преимущественно в космосе
Студенты первого курса	65	65	0	Очень много размышляют. Делают открытия не реже раза в месяц

Студенты-дипломники	30	10	20	Паяют схемы
Младшие научные сотрудники-экспериментаторы	53	40	13	Просят старших найти течь. Пишут статьи. Бегают в отдел снабжения. Моют форвакуумные насосы. Хлопают ушами на семинарах.
Младшие научные сотрудники-теоретики	19	19	0	Разговаривают в коридорах, надеясь сделать великое открытие. Пишут множество формул, большая часть которых окажется неверной
Старшие научные сотрудники	7	6	1	Спят на заседаниях. Помогают младшим научным сотрудникам искать течь.
Сотрудники отдела кадров	5	5	0	Экспериментаторы должны приходить в 8.25, чтобы в 8.30 уже молча сидеть возле включенных установок. Теоретики вовсе не работают, их на месте не застанешь
Сотрудники охраны	6	6	0	Ходят взад-вперед. Предъявляют пропуск вверх ногами
Сотрудники Министерства финансов	18	18	0	Тратят деньги впустую

ПРИЛОЖЕНИЕ 4.

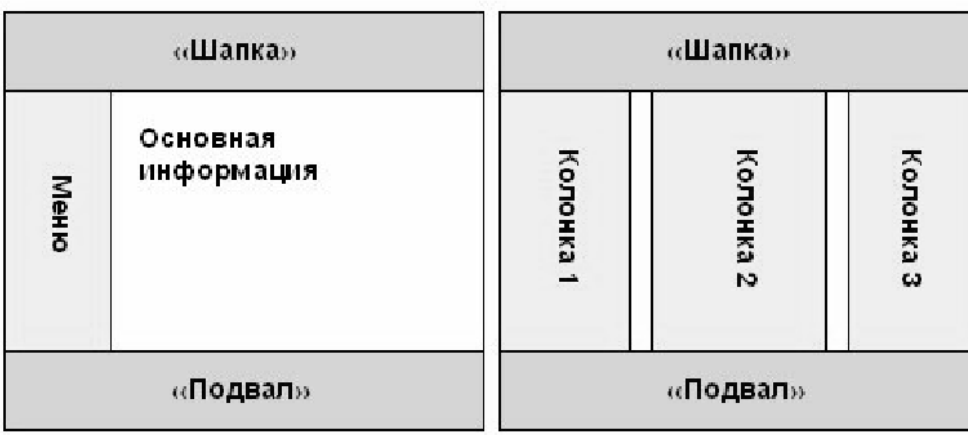
Таблица 1.

Таблица 2.

Таблица 3.

Примеры таблиц с различной структурой

ПРИЛОЖЕНИЕ 5.



Примеры разметки веб-страниц

Содержание