

**Государственное образовательное учреждение высшего
профессионального образования
Волго-Вятская академия государственной службы**
Кафедра информатики и информационных технологий

Разработка web-приложений

Часть 1.

Логика функционирования Web приложений.
Конструирование Web страниц с использованием языка HTML
и
каскадных таблиц стилей (CSS)

Окулич В.И., Трубилов Н.М.

Н.Новгород
2011 год

Оглавление

Введение	5
Глава 1. Основные подходы к разработке Web-приложений: языки, модели и технологии	7
1.1. Эволюция языков создания (разметки) страницы Web	7
1.2. Разработка Web приложений.....	10
1.3. Реализация принципов функционирования Web-технологий	29
1.3.1. Некоторые детали сетевого взаимодействия.....	29
1.3.2. Типы данных и их обработка Web сервере	34
1.3.3. О некоторых других Web технологиях.....	39
1.3.4. Заключение. О навыках разработки серьёзных Web-- приложений....	40
1.4. Контрольные вопросы к главе 1	42
Глава 2. Основы HTML – языка гипертекстовой разметки	44
2.1 Введение в HTML.	44
2.1.1 История развития HTML.....	44
2.1.2. Основные положения гипертекстовой разметки	47
2.1.3. Назначение и состав контейнера <HTML> ... </HTML>	51
2.2 Структура HTML – документа и используемые для этого тэги	52
2.2.1. Назначение раздела 1- дать информацию о версии HTML	52
2.2.2. Назначение и состав контейнера <HTML> ... </HTML>	54
2.3. Разметка, обеспечивающая форматирование HTML – документа.....	66
2.3.1. Понятие форматирования.....	66
2.3.2. Форматирование страницы и текста	66
2.3.3. Форматирование абзаца.....	69
2.3.4. Форматирование произвольного фрагмента текста.	75
2.4. Форматирование гипертекстовых ссылок и закладок	79
2.4.1. Гиперссылки	79
2.4.2. Закладки	81
2.4.3. Открытие html страниц, вызываемой гиперссылкой, в новом окне ...	83
2.4.4. Форматирование гиперссылок.....	84
2.5. Таблицы в HTML: создание и форматирование.....	85

2.5.1. Использование таблиц в HTML документе.....	85
2.5.2. Создание простейших HTML-таблиц	85
2.5.3. Форматирование таблиц.....	88
2.6. Графика в HTML.....	99
2.6.1. Общие соображения.....	99
2.6.2. Способы хранения изображений.	100
2.6.3. Использование изображения в качестве фона HTML-документа (атрибут BACKGROUND тега <BODY>).....	101
2.6.4. Встраивание изображений в HTML-документы (элемент) ...	101
2.6.5. Гиперкарты – важный элемент гипертекста.....	106
2.7. Формы в HTML.....	117
2.7.1. Основы создания форм.	117
2.7.2. Элементы форм	121
2.8. Фреймы	143
2.8.1. Сфера применения фреймов	143
2.8.2. Создание Web-страниц с фреймами.....	144
Глава 3. Основы использования CSS – каскадных таблиц стилей.....	149
3.1. Понятие CSS.....	149
3.2. Подключение CSS.....	150
3.2.1. Таблица связанных стилей	150
3.2.2. Таблица глобальных стилей.....	152
3.2.3. Внутренние стили.....	153
3.3. Базовый синтаксис	154
3.4. Селекторы тегов.....	156
3.5. Классы.....	157
3.6. Идентификаторы.....	162
3.7. Правила создания стиля	166
3.8. Основы CSS-верстки. Создание «резинового» шаблона.	170
3.8.1. Создание макета из одной колонки.....	170
3.8.2. Макет из двух колонок	172
3.8.3. Макет из трех колонок.....	174

Задания для самостоятельной работы.	177
Задание 1.....	177
Задание 2.....	178
Задание 3.....	179
Задание 4.....	181
Задание 5.....	182
Задание 6.....	184
Задание 7.....	184

Введение

В настоящее время известны и используются более десяти тысяч языков программирования. Язык программирования – это совокупность слов, правил орфографии и синтаксиса, по которым можно составлять предложения, немного похожие на обычный язык. Совокупность этих предложений (программы) заставляет электронные устройства компьютеров выполнять различные действия: вычислять, рисовать, печатать, разговаривать и т. п.. Эти программы могут работать на разных операционных системах, но цель у них одна – заставить электронные схемы процессоров и микросхем различных вспомогательных устройств (принтеров, сканеров, мониторов, дисков, флэш-памяти и т.п.) выполнять нужные человеку действия.

Обзоры по языкам программирования можно посмотреть в Интернете¹.

Почему появилось их так много, и разве нельзя было обойтись меньшим количеством? Дело в том, что идёт постоянное совершенствование электронных схем процессоров ЭВМ и, как следствие, появляются новые машинные команды (отражаемые на электронном уровне совокупностью условных единиц и нулей, а точнее разными уровнями напряжений на контактах микросхем). Поэтому возникают новые возможности работы компьютеров. Их надо отображать в виде новых конструкций языка, и это ведёт к появлению новых и более мощных по своим возможностям языков программирования.

Интернет является одним из наиболее динамично развивающихся областей информационных технологий. Поэтому в этой области созданы и продолжают развиваться языки программирования. Появилась целая область (и даже отрасль) знаний, называемая web-программированием.

¹ <http://www.abcpnb.ru/RUS/Resour/PLaTMe/pltmT1.pdf>
<http://www.forth.org.ru/~byka/download/Forth/kg93113.html>
<http://progopedia.ru>

Web программирование – это разработка программных продуктов, предназначенных для работы на сайтах Word Wide Web. Даже разработка web страниц на чистом HTML является web-программированием, так как при просмотре страницы в браузере соответствующая программа исполняет код HTML, форматируя изображение согласно инструкциям этого языка. В настоящее время web-программирование включает очень большое число различных технологий.

В данном пособии мы рассмотрим небольшую часть этой обширной области – язык текстовой разметки HTML, использование так называемых таблиц стилей (CSS) для ускоренного формирования интернет страниц и основы языка web-программирования PHP.

Предварительно в Главе 1 проведён обзор понятий и моделей Web взаимодействия и Web разработки.

Глава 1. Основные подходы к разработке Web-приложений: языки, модели и технологии

1.1. Эволюция языков создания (разметки) страницы Web

В 1989 году Тим Бернерс-Ли предложил руководству европейского центра ядерных исследований (CERN) проект распределенной гипертекстовой системы, которую он назвал World Wide Web (WWW), Всемирная паутина. Первоначально идея системы состояла в том, чтобы при помощи гипертекстовой навигационной системы объединить все множество информационных ресурсов CERN в единую информационную систему. Технология оказалась настолько удачной, что дала толчок к развитию одной из самых популярных в мире глобальных информационных систем. Практически в сознании большинства пользователей глобальной компьютерной сети Internet сама эта сеть ассоциируется с тремя основными службами:

- ✓ World Wide Web.
- ✓ электронная почта (e-mail);
- ✓ файловые архивы FTP;

Успех World Wide Web определен двумя основными факторами: простотой и использованием протоколов межсетевого обмена семейства TCP/IP , (Transmission Control Protocol, /Internet Protocol или на русском языке - протокол управления передачей/ протокол Internet) которые являются основой Internet. Немного подробнее о протоколах будет рассказано ниже. Практически все пользователи Сети одновременно получили возможность попробовать себя в качестве создателей и читателей информационных материалов, опубликованных во Всемирной паутине. Но и популярность самого Internet во многом вызвана появлением World Wide Web, так как это первая сетевая технология, которая предоставила пользователю простой современный интерфейс для доступа

к разнообразным сетевым ресурсам.

Простота и удобство применения привели к росту числа пользователей WWW и привлекли внимание коммерческих структур. Далее процесс роста числа пользователей стал лавинообразным, и так продолжается до сих пор. Сегодня создание страницы Web является не слишком трудной задачей. Многие стандартные программные пакеты персональных компьютеров обладают встроенными средствами для преобразования документов текстовых процессоров, электронных таблиц, баз данных и т.д. в специально кодированные документы, которые могут быть доступны в Web. Специальные пакеты для создания страниц Web, такие, как Microsoft FrontPage и Macromedia Dreamweaver, позволяют легко создавать страницы Web. В большинстве таких случаев даже не нужно знать о существовании специального языка кодирования HTML (язык разметки гипертекста), который неявно все это обеспечивает.

Существует язык XHTML². XHTML - это основанный на XML язык разметки гипертекста, максимально приближенный к текущим стандартам HTML. XHTML отличается от HTML строгостью написания кода. С его помощью страницы Web можно создавать с помощью простого текстового редактора, получая в этом случае значительно больший контроль над их структурой и форматированием.

Безошибочная разметка XHTML повышает вероятность корректного отображения вашей страницы в большинстве браузеров и других пользовательских агентах в ближайшие несколько лет. Четкость, простота в написании и гибкость XHTML позволяют создавать компактный быстро загружающийся код, понятный при его редактировании в будущем, и подготовить содержимое для просмотра в различных пользовательских агентах.

² <http://htmlweb.ru/html/xhtml.php>

Определить, соответствует ли сайт Web-стандартам очень легко: если вы используете только допустимый код XHTML (и правила CSS), - сайт совместим со стандартами.

Допустимый XHTML³ означает, что вы используете только XHTML, без бессмысленных, незакрытых или устаревших тегов HTML. Вы можете проверить код с помощью валидатора - Web-программы проверки кода (<http://validator.w3.org>), в которой необходимо ввести адрес вашей страницы. Если все в порядке, вы увидите сообщение This page Is Valid XHTML!. CSS можно проверить по адресу <http://jigsaw.w3.org/css-validator> тем же образом.

В свою очередь XML4 (eXtensible Markup Language) - расширяемый язык разметки. Основное внимание в XML сосредоточено на данных. В XML структурная разметка данных и представление данных строго разделены. XML — это обобщенный язык разметки. В отличие от HTML, XML позволяет создавать собственные теги и таким образом формировать собственную структуру документа.

Основные причины создания XML:

- попытка предоставить мощные средства форматирования и структурирования данных всем желающим;
- необходимость в стабильной реализации языка структурирования документов, для которого

Наконец, надо упомянуть о последней «на сегодняшний день» (2010 год) пятой версии языка HTML – HTML5.

HTML5⁵ (англ. HyperText Markup Language 5) — пятая версия одного из главных языков разметки Интернета, HTML. Версия языка, полностью соответствующая стандарту XML, называется XHTML5. HTML 5 вводит несколько новых элементов и атрибутов.

Основные отличия HTML 5 от HTML 4 составляют:

³ <http://www.linkex.ru/css/markimg.php>

⁴ В данном учебнике нет описания этих языков.

⁵ http://ru.wikipedia.org/wiki/HTML_5

- Новые правила лексического разбора;
- Новые элементы — header, footer, section, article, video, audio, progress, nav, meter, time, aside, canvas;
- Новые типы input-элементов;
- Новые атрибуты;
- Глобальные атрибуты — id, tabIndex, repeat;
- Устаревшие элементы убраны — center, font, strike.

1.2. Разработка Web приложений

Рассмотрим понятие "Разработка" Web. Оно в противоположность "созданию" страниц Web, выходит далеко за пределы использования кодов разметки и нескольких подключаемых модулей или метода сценариев для создания привлекательных или информативных страниц Web. Этот термин относится к использованию **специальных стратегий, инструментов и методов** для создания страниц Web и сайтов Web, характеризуемых как **трех-уровневые, клиент/серверные системы** обработки информации. Рассмотрим эти термины более подробно, чтобы понять разнообразие задач, для которых разрабатываются страницы и сайты Web.

1.2.1. Системы обработки информации: интранет, интернет и экстранет.

Технологии Web используются не только для создания персональных или рекламных сайтов Web, содержащих информативный, интересный или развлекательный материал для публичного потребления. Прежде всего они стали важным средством поддержки фундаментальных "бизнес-процессов" современных организаций, а именно операционных и управленческих функций.

Технические инфраструктуры поддержки этих задач упрощенно делятся на три типа систем на основе Web, называемых системами интранет, интернет и экстранет.

Системы интранет

Системы интранет являются внутренними системами, помогающими выполнять повседневную обработку информации, обеспечивая управленческо-информационную и производственную деятельность организаций. Системы интранет на основе Web **обслуживают стандартные внутренние функции бизнеса, оказывая тем самым влияние на основные организационные системы, такие, как бухгалтерский учет и финансовая отчетность, маркетинг и отдел продаж, системы закупок и сбыта, производственные системы, системы трудовых ресурсов и другие.**

Со временем системы интранет на основе Web станут основными техническими средствами, посредством которых будет осуществляться внутренняя деятельность организаций по выполнению бизнес-процессов.

Системы интернет

Системы интернет являются публичными информационными системами. Они включают в себя публичные сайты, которые предоставляют новости, информацию, и развлечения; сайты электронной коммерции для маркетинга и продажи продуктов и услуг; правительственные сайты для информирования или обслуживания широкой публики; и образовательные сайты для предоставления локального и удаленного доступа к образованию и знаниям. Всем частям общества публичные системы интернет предоставляют товары, услуги и информацию посредством Всемирной паутины WWW и связанных с ней сетей и услуг.

Системы экстранет

Системы экстранет являются системами бизнес-для-бизнеса (B2B), **которые управляют электронным обменом данными (EDI) между деловыми предприятиями.** Эти системы обеспечивают информационный поток между организациями – между компанией и ее поставщиками и между компанией и ее сбытовыми организациями – чтобы помочь в координации последовательности закупки, производства и распространения. Электронный обмен данными помогает **исключить бумажный поток**, сопровождающий

бизнес-транзакции⁶, используя технологии Web для пересылки электронных документов между компьютерами, а не между людьми.

Так как эти системы основаны на Web приложениях, то это автоматически⁷ устраняет трудности передачи информации между различными программными и аппаратными платформами с **изначально различными информационными форматами и различными протоколами обмена информацией**, так как для Web сетей такое взаимодействие планировалось изначально.

Web становится основным технологическим базисом, электронной магистралью для сбора информации, обработки и распространения во всех типах организаций – в коммерческих и финансовых предприятиях, образовательных учреждениях, правительственных агентствах, учреждениях здравоохранения, агентствах новостей и отрасли развлечений и в большинстве других формальных организаций, как больших, так и маленьких. Это всепроникающая технология для разработки систем работы с информацией во всех частях общества.

1.2.2. Что означает термин «системы на основе Web»

Термин "на основе Web" относится к тому факту, что системы обработки информации полагаются на технологию Интернет, в частности, на так называемую Всемирную паутину (WWW). Поэтому системы на основе Web действуют **в технологических рамках** со следующими характеристиками.

1. Системы действуют в публичных, а не в частных сетях данных. Они осуществляют коммуникацию через Интернет, т.е. через распространенные по всему миру, взаимосвязанные сети компьютеров, которые являются публично доступными.

⁶ Под бизнес транзакцией (business transaction) обычно понимается элементарное взаимодействие между двумя сторонами любого бизнес-процесса (например, производителем товаров или услуг и их клиентами), которое будучи начатым, обязательно должно быть закончено

⁷ Поскольку Web технология изначально разрабатывалась с целью преодоления именно этих трудностей

2. Коммуникационные сети основываются на открытых и публичных технических стандартах, таких, как архитектуры Ethernet, протоколы передачи TCP/IP и протоколы **приложений HTTP и FTP**. Они не являются частными или патентованными стандартами, но являются принципиально открытыми и свободными для публичного использования.
3. Системы обработки на основе Web используют широко распространенное, часто бесплатное, программное обеспечение для разработки и работы. Деятельность по обработке происходит с помощью браузеров Web, а не специально написанного программного обеспечения для интерфейса пользователя и для внешнего сбора данных и обработки. Браузеры Microsoft Internet Explorer, Mozilla, Firefox, Opera, Netscape Navigator и другие являются средством взаимодействия пользователей с системами обработки информации. Также широко распространенные компьютеры **серверов Web** выполняют основные функции бизнес-обработки, а **серверы баз данных** обеспечивают хранение информации, доступ к ней и извлечение.

Поэтому общедоступные, не являющиеся специализированными, не являющиеся патентованными оборудование и системы программного обеспечения предоставляют техническую среду для разработки систем обработки информации и для управления этой деятельностью.

1.2.3. Концепция клиент/серверной архитектуры

В предыдущем разделе были упомянуты термины серверы Web и серверы баз данных. Общее понятие сервера означает либо программу (комплекс программ), выполняющую определенную и достаточно сложную задачу управления каким-либо компьютерным процессом, представляющим интерес для пользователя, либо компьютер, на котором работает эта программа.

В этом случае можно дать такое определение: сервер - компьютер, подключенный к локальной или глобальной сети с установленным на нем соот-

ветствующим ПО, позволяющим отвечать на запросы и обслуживать другие серверы и клиентские компьютеры.

В принципе на одном компьютере может работать несколько «серверных» программ, но чаще крупные поставщики компьютерных услуг разделяют функции по разным аппаратам.

Под Web-сервером понимают основную программу, которая обеспечивает работу веб-сайта. Главная задача такого сервера - передача страниц сайта браузеру (программе на компьютере клиента) по протоколу HTTP. При необходимости сервер запускает скрипты для динамического создания страниц сайта. Действия сервера обычно протоколируются в логах⁸ (лог-файлах) и служат основанием для подсчета статистики сайта.

Под сервером баз данных понимают программу, позволяющую пользователю получать доступ к данным базы, манипулировать ими и структурой базы на специальном языке запросов к данным SQL.

Противоположным понятию сервера (и неразрывно с ним связанного) является понятие программы-клиента или компьютера-клиента.

Википедия⁹ даёт следующее определения этому понятию:

«Клиент — это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу».

Программа, являющаяся клиентом, взаимодействует с сервером, используя определённый протокол (стандартизированные правила передачи и приёма). Она может:

- Запрашивать с сервера какие-либо данные,
- Манипулировать данными непосредственно на сервере,
- Запускать на сервере новые процессы и т. п.

⁸ Для учета всех действий, совершаемых как с web-сервером, так и с сайтом, используются лог-файлы (или же просто логи). Лог – это текстовый файл, понятный даже новичку, в котором каждому событию соответствует одна строка с временем и некоторыми дополнительными сведениями. Для удобства пользователей лог-файлы часто группируются по датам, что облегчает поиск необходимых сведений.

⁹ [http://ru.wikipedia.org/wiki/Клиент_\(информатика\)](http://ru.wikipedia.org/wiki/Клиент_(информатика))

Полученные от сервера данные клиентская программа может предоставлять пользователю или использовать как-либо иначе, в зависимости от назначения программы.

Программа-клиент и программа-сервер могут работать как на одном и том же компьютере, так и на разных. Во втором случае для обмена информацией между ними используется сетевое соединение.

Разновидностью клиентов являются терминалы — рабочие места на многопользовательских ЭВМ, оснащённые монитором с клавиатурой, и не способные работать без сервера.

Тем не менее, не всегда под клиентом подразумевается компьютер со слабыми вычислительными ресурсами. Чаще всего понятия «клиент» и «сервер» описывают распределение ролей при выполнении конкретной задачи, а не вычислительные мощности. На одном и том же компьютере могут одновременно работать программы, выполняющие как клиентские, так и серверные функции. Например, веб-сервер может в качестве клиента получать данные для формирования страниц от SQL-сервера.

Понятия сервер и клиент и закрепленные за ними роли образуют программную концепцию «клиент-сервер».

Для взаимодействия с клиентом (или клиентами, если поддерживается одновременная работа с несколькими клиентами) сервер выделяет необходимые ресурсы межпроцессного взаимодействия (например, разделяемая память) и ожидает запросов на открытие соединения (или, собственно, запросов на предоставляемый сервис). В зависимости от типа такого ресурса, сервер может обслуживать процессы в пределах одной компьютерной системы или процессы на других машинах через каналы передачи данных (например СОММ-порт) или сетевые соединения.

Формат запросов клиента и ответов сервера определяется протоколом.

Более предметное понимание концепции «клиент-сервер» может быть достигнуто через рассмотрение моделей клиент-серверного взаимодействия.

1.2.4. Обобщенная модель взаимодействия клиент-сервер

В настоящее время существуют двух- и трёхзвенные модели взаимодействия (иногда термин «звено» заменяют на «слой»). Компанией Gartner Group¹⁰, специализирующейся в области исследования информационных технологий, предложена следующая классификация двухзвенных моделей взаимодействия клиент-сервер (двухзвенными эти модели называются потому, что три компонента приложения различным образом распределяются между двумя компьютерными узлами) – Рис. 1-1:

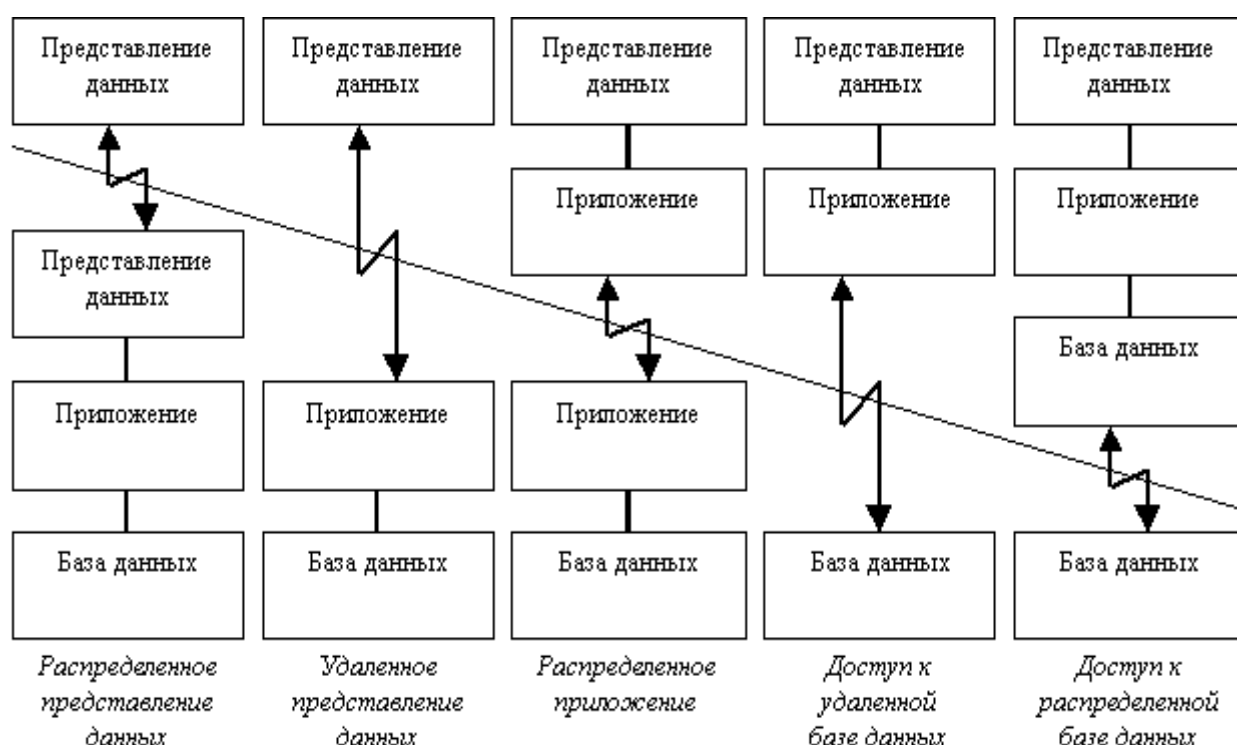


Рис. 1-1. Схематическое представление возможного распределения разных элементов обработки данных на компьютере пользователя (над диагональю) и компьютере-сервере (под диагональю)

Показанная схема модифицировалась параллельно с развитием вычислительной техники, в том числе и технологий межкомпьютерной связи.

Исторически первой появилась модель распределенного представления данных, которая реализовывалась на универсальной ЭВМ с подключенными к ней неинтеллектуальными терминалами. **Управление данными и взаимо-**

¹⁰ <http://www.gartner.com/technology/home.jsp>

действие с пользователем при этом объединялись в одной программе, на терминал передавалась только "картинка", сформированная на центральном компьютере.

Затем, с появлением персональных компьютеров (ПК) и локальных сетей, были реализованы модели доступа к удаленной базе данных. Некоторое время базовой для сетей ПК была архитектура **файлового сервера**. При этом один из компьютеров является **файловым сервером**, на **клиентах выполняются приложения**, в которых совмещены компонент представления и прикладной компонент (СУБД и прикладная программа). Протокол обмена при этом представляет набор низкоуровневых вызовов операций файловой системы. Такая архитектура, реализуемая, как правило, с помощью персональных СУБД, имеет очевидные недостатки - высокий сетевой трафик и отсутствие унифицированного доступа к ресурсам.

С появлением первых **специализированных серверов баз данных** появилась возможность другой реализации модели доступа к удаленной базе данных. В этом случае ядро СУБД функционирует на сервере, протокол обмена обеспечивается с помощью языка SQL. Такой подход по сравнению с файловым сервером ведет к уменьшению загрузки сети и унификации интерфейса "клиент-сервер". Однако, сетевой трафик остается достаточно высоким, кроме того, по-прежнему невозможно удовлетворительное администрирование приложений, поскольку в одной программе совмещаются различные функции.

Позже была разработана концепция активного сервера, который использовал механизм хранимых процедур. Это позволило часть прикладного компонента перенести на сервер (модель распределенного приложения). Процедуры хранятся в словаре базы данных, разделяются между несколькими клиентами и выполняются на том же компьютере, что и SQL-сервер. Преимущества такого подхода: возможно централизованное администрирование прикладных функций, значительно снижается сетевой трафик (т.к. передаются не SQL-запросы, а вызовы хранимых процедур). Недостаток - ограничен-

ность средств разработки хранимых процедур по сравнению с языками общего назначения (C, C++, C#, PHP и т.д.).

На практике сейчас обычно используются смешанный подход:

- простейшие прикладные функции выполняются хранимыми процедурами на сервере,
- более сложные реализуются на клиенте непосредственно в прикладной программе,

В последние годы начала реализовываться концепция «тонкого» клиента, функцией которого остается только отображение данных (модель удаленного представления данных).

В компьютерных технологиях терминал или тонкий клиент (англ. thin client) — это **компьютер-клиент сети с клиент-серверной архитектурой, который переносит большинство задач по обработке информации на сервер.**

Таким образом, сервер **необходим для нормальной работы терминала.** Этим терминал (тонкий клиент) отличается от **толстого клиента**, который, напротив, производит обработку информации независимо от сервера, используя последний в основном лишь для хранения данных. **Примером подобного терминала (тонкого клиента) может служить компьютер с браузером, использующийся для работы с веб-приложениями.**

Терминальный сервер — также сервер терминалов, предоставляющий клиентам вычислительные ресурсы (процессорное время, память, дисковое пространство) для решения задач. Технически, терминальный сервер представляет собой очень мощный компьютер (либо кластер), соединенный по сети с терминальными клиентами — которые, как правило, представляют собой маломощные или устаревшие рабочие станции. Построение сетевой инфраструктуры на базе терминалов (тонких клиентов) осуществляется с использованием терминального сервера, на котором и происходит обработка информации.

В последнее время также наблюдается тенденция к большему использованию модели распределенного приложения. Характерной чертой таких приложений является **логическое разделение приложения на две и более частей**, каждая из которых может выполняться на отдельном компьютере. Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате. В этом случае двухзвенная архитектура клиент-сервер становится трехзвенной, а в некоторых случаях, она может включать и больше звеньев (смотри Рис. 1-2).



Рис. 1-2. Схематическое изображение трёхзвенной архитектуры

1.2.5. Трехслойная модель клиент/серверная архитектура для Web приложений и задачи его разработки

Термин "клиент/сервер" относится к применению сетей на основе серверов (то есть программ) для управления общим доступом к ресурсам и для распределения задач между аппаратными и программными компонентами. В клиент-серверных сетях на основе Web распределение задач обработки происходит в трех слоях, которые соответствуют трем основным компонентам оборудования и программного обеспечения системы (Рис. 1-3).

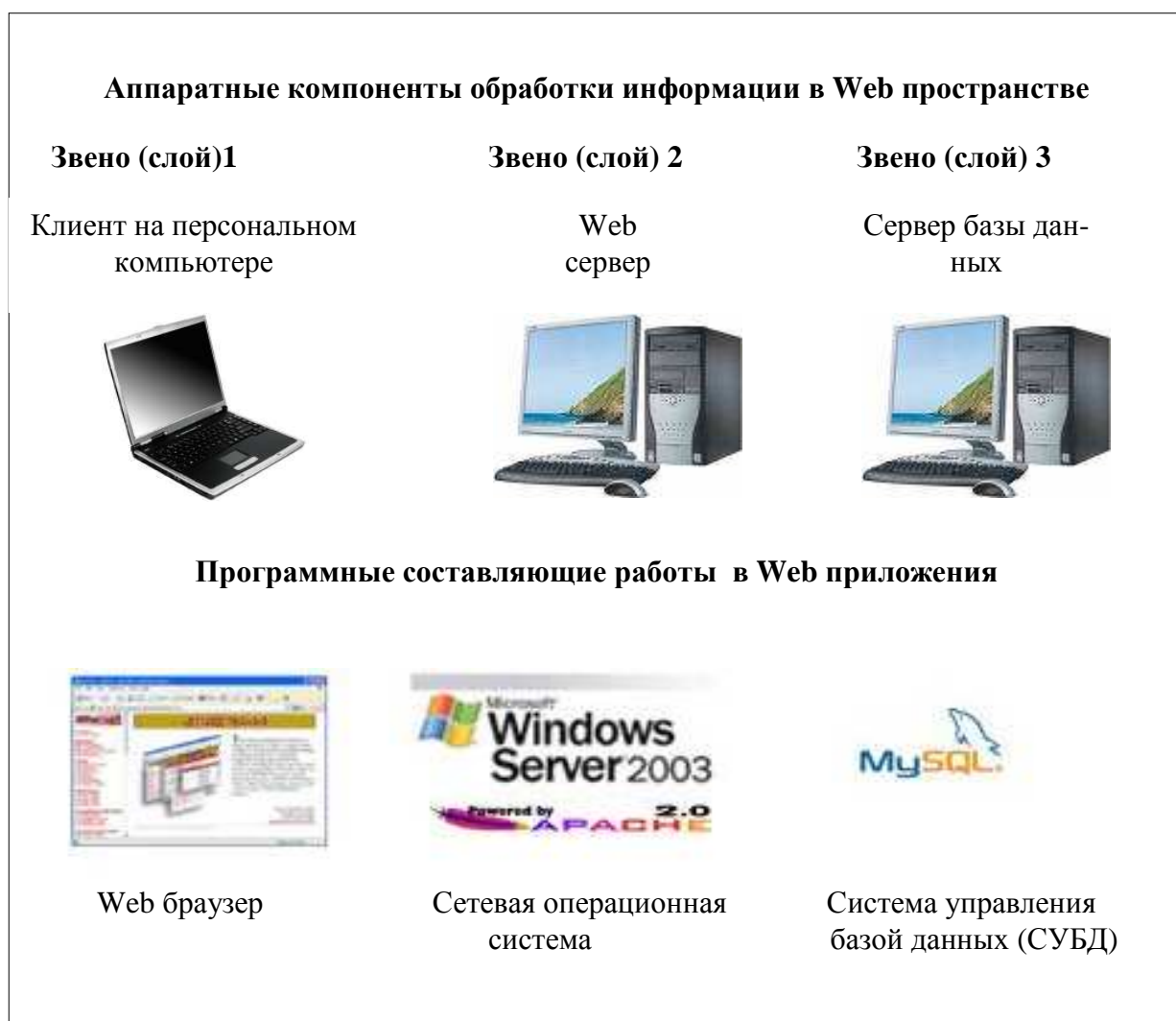


Рис. 1-3. Аппаратные и программные слои трехслойной системы обработки информации

В первом слое клиентский настольный ПК выполняет работу интерфейса пользователя системы; во втором слое сервер Web выполняет основные функции системы по обработке; и в третьем слое сервер базы данных, и в некоторых случаях медиа-сервер, осуществляет требуемые системе функции хранения и извлечения информации.

В свою очередь, каждый из трех аппаратных компонентов содержит соответствующее программное обеспечение. Клиентским программным обеспечением является браузер Web. Сервер Web выполняет сетевую операционную систему (NOS), такую, как Windows Server, Unix Server или Linux Server, и с помощью дополнительного программного обеспечения, напри-

мер, Internet Information Server или Apache Web Server, реализует службы Интернет, — WWW, FTP, SMTP (Simple Mail Transfer Protocol) и другие. Сервер базы данных выполняет команды размещённой на нём системы управления базой данных, таких, как MySQL, Oracle, Microsoft SQL Server, Access и других популярных пакетов.

Таким образом, отдельные компоненты выполняют отдельные задачи обработки, которые интегрируются с помощью Web в законченную систему обработки информации.

Рассмотрим, например, посещение Web-сайта электронной коммерции (е-коммерции). Браузер Web является интерфейсом работы с сайтом. В ответ на различные "входящие" запросы, которые вы отправляете при просмотре продаваемых товаров, создаются различные страницы "вывода". Запросы вводятся в систему через ссылки Web и посылаемые формы, **ответы системы создают страницы HTML, передаваемые назад браузеру для вывода на экране.** Браузер выполняет действия по вводу и выводу, необходимые для взаимодействия с сайтом.

При этом на Web-сервере решаются специальные задачи по обработке информации. При запросе клиента по поиску товара, выполняются программы поиска в базах данных для извлечения подходящего для клиента товара и для форматирования вывода для доставки в браузер Web. При просмотре корзины покупателя другие процедуры извлекают выбранные товары и вычисляют общую стоимость заказа. При оплате заказа исполняются специальные программы для соединения с системой проверки кредитной карты и банковскими системами, так что соответствующие счета дебетуются и кредитуются.

Множество задач обработки, связанных с перемещением в сети и покупкой, происходят на серверах Web **скрыто от пользователя**, но они критически важны для осуществления покупки и для осуществления бизнес-транзакций, которые с этим связаны.

Большая часть информации, которая собирается и генерируется во время покупки, хранится в базах данных, которые находятся на отдельных серверах баз данных. Вся информация, которая выводится на экран, извлекается из таблиц базы данных. Выбранные товары хранятся в таблицах базы данных. **Практически каждый фрагмент информации о просматриваемых продуктах и транзакциях при покупке сохраняется в больших базах данных в самой системе е-коммерции или в связанных базах данных, которые находятся в центре окружающих ее систем бухгалтерского учета, закупок и дистрибуции.**

Даже в самых маленьких коммерческих системах на основе Web присутствуют такие же функции. Главное состоит в том, что в системах на основе Web любого размера существуют **три основных слоя функциональности**. Поэтому, с точки зрения разработчика Web, задача разработки полноценного приложения состоит в создании:

- интерфейса пользователя,
- процедур обработки бизнес-операций,
- компонентов поддержки базы данных и, наконец,
- последующей интеграции в полностью функциональную систему обработки информации.

1.2.6. Модели реализации функций Web приложений

Рассмотрев концепцию многозвенной архитектуры Web приложения, перейдём к более подробному рассмотрению реализации различных функций этих приложений. Это удобно сделать, представив эти функции в составе двух моделей:

- Модели доставки информации и
- Модели обработки информации

Исторически Всемирная паутина WWW начинала функционирование просто как "система доставки информации". К настоящему времени Web стала чем-то большим, чем просто электронной библиотекой информации.

Она стала платформой коммуникации, информации и транзакций, на которой реализуется экономическая, социальная, политическая, образовательная и культурная деятельность.

1.2.6.1. Модель доставки информации

При функционировании в качестве системы доставки информации деятельность по разработке Web — достаточно простая и прямолинейная.

Прежде всего, информационное содержимое вводится в документ, который со временем станет страницей Web. Это содержимое окружается специальными кодами компоновки и форматирования Языка разметки гипертекста (HTML) — в последнее время Расширяемого языка разметки гипертекста (XHTML) — для управления его структурой и представлением в браузере Web.

Затем документ сохраняют на сайте, расположенном на **компьютере сервера Web** для ожидания публичного доступа. Пользователи обращаются к документу, вводя в окне своего браузера адрес Web-документа. Этот адрес, называемый **URL (Uniform Resource Locator** или Единообразный локатор ресурса), определяет сайт, где хранится страница, и расположение ее каталога на сервере Web. Этот сервер, в свою очередь, извлекает страницу и посылает ее браузеру, который интерпретирует код HTML и выводит документ на экране компьютера.

URL включает в себя:

- метод доступа к ресурсу, т.е. протокол доступа (http, gopher, WAIS, ftp, file, telnet и др.)
- сетевой адрес ресурса (имя хост-машины и домена)
- полный путь к файлу на сервере

В общем виде формат URL выглядит так:

method://host.domain[:port]/path/filename

где **method** имеет одно из значений, перечисленных ниже:

- file файл на вашей локальной системе, или файл на anonymous FTP сервере
- http файл на World Wide Web сервере
- gopher файл на Gopher сервере
- WAIS файл на WAIS (Wide Area Information Server) сервере
- news группа новостей телеконференции Usenet
- telnet выход на ресурсы сети Telnet

Параметр **host.domain** - адрес ресурса в сети Internet.

Параметр **port** - число, которое необходимо указывать, если метод требует номер порта (отдельные сервера могут иметь свой отличительный номер порта).

В рамках данной модели, информационное содержимое страницы Web "фиксируется" или "замораживается" в определенном месте. Оно становится встроенным и тесно связанным с кодами форматирования XHTML, которые его окружают. В связи с этим становится трудно изменять содержимое страницы, не переписывая и не редактируя его форматы представления. Поэтому затрудняется сохранение актуальности страниц, особенно если содержимое постоянно изменяется.

В то самое время авторам страниц Web зачастую необходимо быть знакомым с кодированием XHTML. Даже при использовании визуальных инструментов, таких, как FrontPage или Dreamweaver, автору может понадобиться специалист по кодированию, чтобы страница выглядела требуемым образом. **"Эксперту" Web часто также бывает необходимо работать в тесном контакте с поставщиком контента, обеспечивая технические навыки для сопровождения страниц.**

Для пользователей также имеются ограниченные возможности взаимодействия с традиционными страницами Web. Они могут быть лишь пассивными читателями контента, для которых сервер Web действует в качестве простого электронного "переворачивателя страниц".

Поэтому сайт Web, создаваемый вокруг модели доставки информации, может стать не источником актуальной информации, а статическим, пассивным историческим архивом.

1.2.6.2. Модель обработки информации

Чтобы преодолеть статическое, пассивное использование Web, возникает необходимость рассматривать Web не просто как систему доставки информации, но как полнофункциональную систему обработки информации

Основные функции обработки информации отображены на Рис. 1-4.

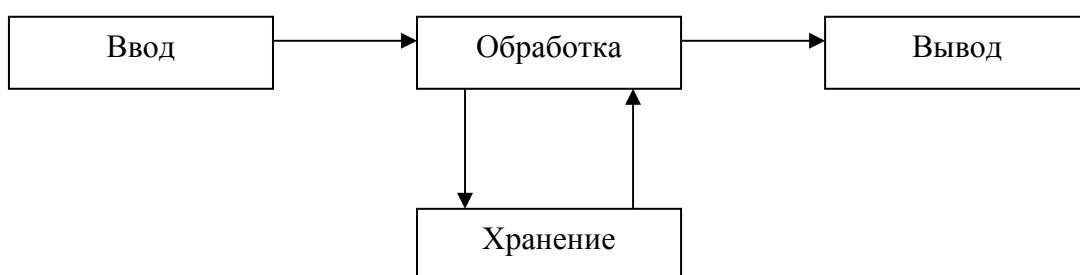


Рис. 1-4. Основные функции обработки информации

В системе Web четыре базовые функции ввода, обработки, вывода и хранения имеют специфические значения:

- Функция ввода позволяет пользователям взаимодействовать с системой, запрашивая параметры обработки, управляя информационным доступом и определяя методы доставки. Кроме того, пользователь может стать источником данных, которые обрабатывает система и которые она поддерживает в своих репозиториях¹¹ хранимой информации.
- Функция обработки относится к деятельности по манипуляции данными и логике обработки, необходимых для выполнения работы системы. Этот термин предполагает, что система может "программироваться" для выполнения арифметических и логических операций, необходимых

¹¹Репозиторий — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети

для манипуляции данными ввода и для создания выводимой информации.

- Функция вывода доставляет результаты обработки пользователю в правильном, своевременном и соответствующем образом форматированном виде.
- Функция хранения гарантирует продолжительность существования и целостность обрабатываемой информации, поддерживая ее в течение длительного периода времени и позволяя добавлять, изменять или удалять систематическим образом. В конечном счете, хранимая информация становится основным контентом страниц Web, отражая самую современную и точную информацию, появляющуюся на этих страницах.

С точки зрения обработки информации сама сеть Web функционирует как гигантская открытая компьютерная система, и фактически такой и является. Деятельность по обработке информации происходит на различных **аппаратных** и **программных** компонентах, расположенных в **одном месте** или **разбросанных** по всему миру.

На сайтах, реализующих полную модель обработки информации, информационное содержание имеет следующие характеристики:

- всегда актуально, поскольку пользователь может взаимодействовать со страницей Web, добавляя или изменяя информацию в системе,
- персонализировано в соответствии с потребностями пользователя,
- автоматически изменяется в ответ на запросы пользователя, и когда изменяется сама информация.

1.2.7. Присваивание функций обработки информации компонентам трёхзвенной системы Web

Вернёмся снова к понятию трехслойной, клиент/серверной системы, и посмотрим, как **аппаратные компоненты, программные компоненты, и функции обработки** связаны друг с другом с точки зрения обработки информации. Для этого придётся соединить два рисунка: Рис. 1-3. Аппаратные и программные слои трехслойной системы обработки информации

и Рис. 1-4. В результате получим **Рис. 1-5.**

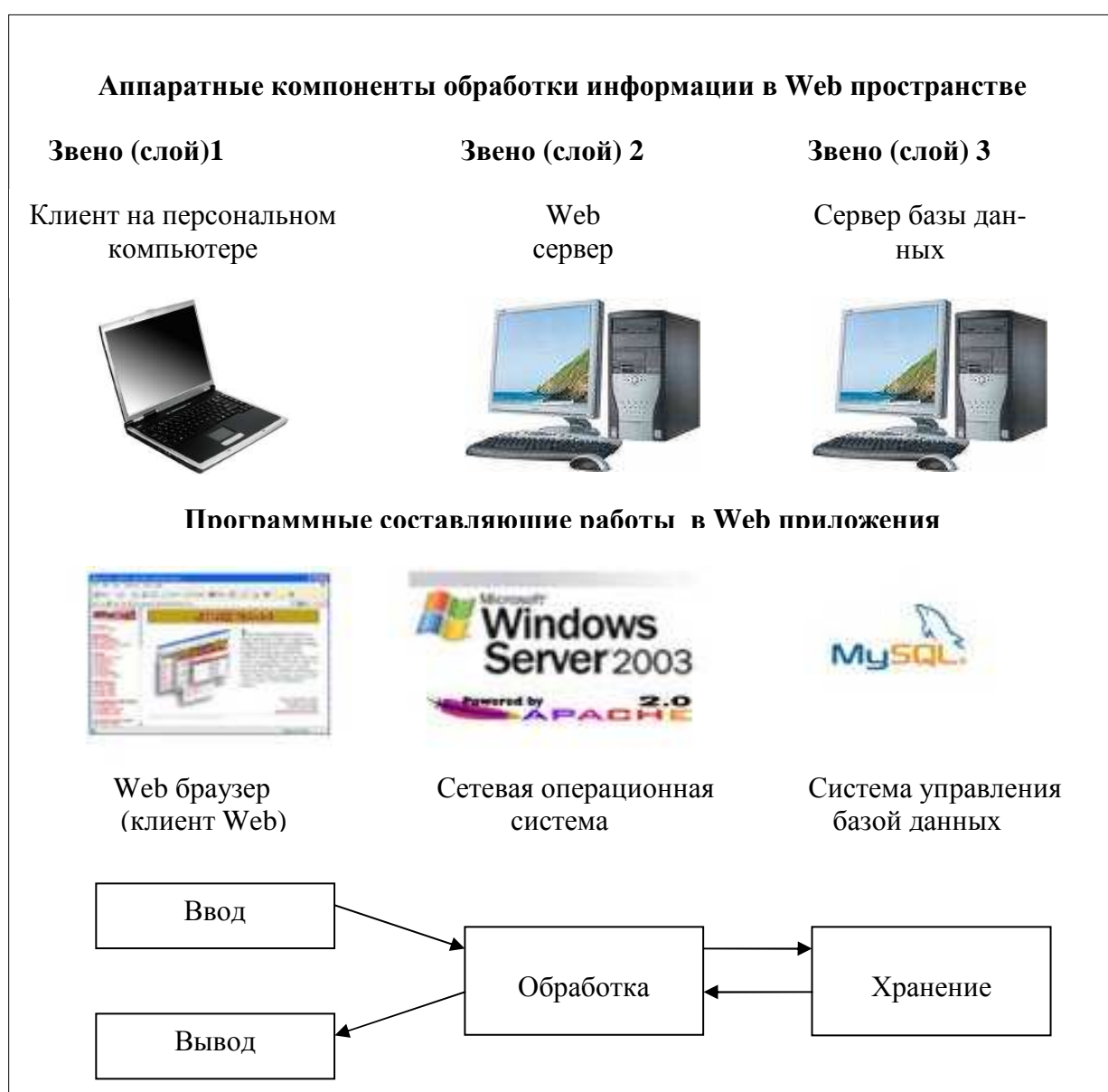


Рис. 1-5. Функции системы обработки информации, отображенные в трехслойную систему клиент/сервер

Все функции ввода и вывода **в основном** выполняются на клиентской машине Web. Такая деятельность интерфейса пользователя, как ввод данных, проверка данных, управление обработкой и форматирование вывода, выполняются на клиенте Web.

Основная деятельность по обработке информации падает на сервер Web, называемый иногда **сервером приложений Web**. Здесь программируются арифметические и логические процедуры для выполнения задач системы по бизнес обработке.

Наконец, хранение данных и функции доступа выполняются на сервере базы данных. Этот компонент управляет хранением информации и функциями извлечения, необходимыми для выполнения бизнес - обработки, и позволяет осуществлять долгосрочное обслуживание хранимой информации посредством добавления, изменения и удаления файлов и баз данных.

Основная идея подобного разделения функций заключается в том, что специализированные компоненты выполняют специализированную работу, для которой они лучше всего подходят.

Справедливо также то, что системные функции «не привязаны к месту». То есть деятельность ввода, обработки, вывода и хранения могут происходить там, где расположены компоненты. Они могут быть:

- заключены в одной машине на одном рабочем столе,
- распределены между двумя или несколькими машинами в отделе или компании либо
- широко разбросаны по всему земному шару.

Во всех этих случаях используемые технологии и методы являются практически одинаковыми, делая достаточно рутинной (простой, однотипной) разработку приложений Web для любой физической или географической среды, с которой столкнется разработчик.

1.3. Реализация принципов функционирования Web-технологий

1.3.1. Некоторые детали сетевого взаимодействия

Очевидно, что реализация изложенных выше принципов функционирования Web приложений требует согласованного взаимодействия многих программных компонент и аппаратных комплексов. Это касается, прежде всего, деталей сетевого взаимодействия между клиентами и серверами. Достаточно давно разработаны модели межкомпьютерного обмена информацией (модель OSI¹² и её модификации), а также протоколы (точные и чёткие правила или спецификации), по которым осуществляются эти взаимодействия¹³.

Детальное рассмотрение принципов и технической реализации межкомпьютерного взаимодействия по сети не являются предметом рассмотрения данного пособия. Поэтому ограничимся рассмотрением лишь некоторых процессов, происходящих незаметно для пользователя при нажатии мышки по ссылке, вводе URL-адреса или при осуществлении любого другого запроса к Web-странице. Такое описание необходимо для получения целостной картины о многообразии компонентов, используемых для создания всемирной сети, а также для выбора инструментов, которые используются при создании WWW.

Для того чтобы читатели получили целостную картину о принципах взаимодействия компонентов, используемых при создании Web-проектов, на Рис. 1-6 представлена структура Web. Обычно доступ к Web-ресурсам осуществляется при помощи браузеров, наиболее известные из которых Netscape/Mozilla и Internet Explorer. Однако существует достаточно много альтернативных браузеров, например Galeon Konquerer (для Linux-систем),

¹² Сетевая модель OSI (базовая эталонная модель взаимодействия открытых систем, англ. Open Systems Interconnection Basic Reference Model, 1978 г.) — абстрактная сетевая модель для коммуникаций и разработки сетевых протоколов. Представляет уровневый подход к компьютерной сети. Каждый уровень обслуживает свою часть процесса взаимодействия. Благодаря такой структуре совместная работа сетевого оборудования и программного обеспечения становится гораздо проще и прозрачнее.

¹³ http://ru.wikipedia.org/wiki/Сетевая_модель_OSI

OmniWeb и Opera а также браузеров, работающих в текстовом режиме Lynx, links и w3g.

Когда пользователь щелкает на ссылке или вводит URL-адрес (например, www.example.com) браузер устанавливает соединение (используется также термин сетевое соединение (network connection)) с сервером www. example.com.

Имя (сервера www.example.com присвоено Internet-адресу, который представляет собой четыре числа, разделенные точками (например, 1.2 .3 .4), и называется IP-адресом. Браузер осуществляет соединение с сервером www.example.com через **порт 80**¹⁴, который используется операционной системой сервера для работы с **HTTP-запросами**. Номера портов стандартизированы.

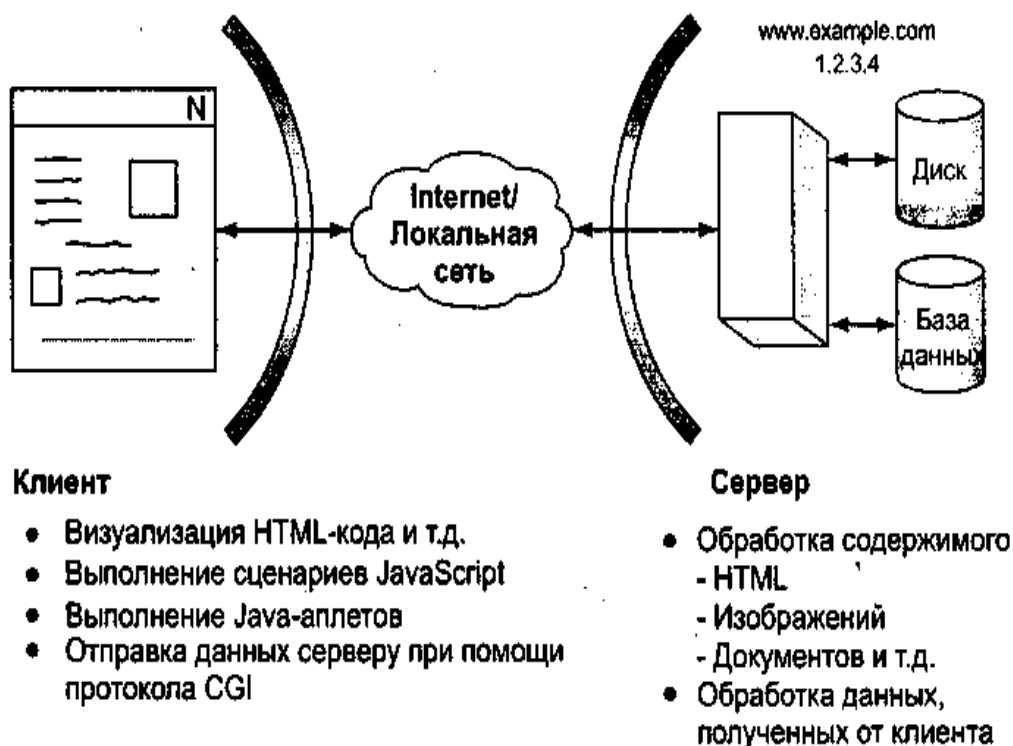


Рис. 1-6 Функциональная структура Web

¹⁴ Сетевой порт представляет собой число от 1 до 65535, указанное и известное обоим приложениям, между которыми устанавливается связь.

В зависимости от типа клиентского запроса сервер передает или получает информацию от клиента. Типы данных, передаваемых от сервера к клиенту, содержат текст (включая HTML-данные), изображения, Java-апплеты документы типов, файлы PDF и т.д. **Данные, передаваемые от сервера, могут генерироваться несколькими способами: статически, динамически или внедряться** (каждый из этих способов рассмотрен в последующих главах).

В задачи клиента входит получение от сервера потока текста, изображений, Java-апплетов, документов и т.д., а также визуализация изображения.

Кроме того, клиенту необходимо выполнять код сценариев JavaScript и Java-апплетов¹⁵, которые переданы от сервера¹⁶. **Для отправки данных серверу клиент может использовать протокол CGI¹⁷ (Common Gateway Interface).**

Эти данные могут обрабатываться любым способом, который определяются сервером. Обработка может осуществляться как на стороне клиента, так и на стороне сервера. Одним из преимуществ использования подобного метода, так же как и при использовании многих других Web-компонентов, является то, что после установки правил, события, происходящие на стороне клиента, не зависят от событий на стороне сервера. Например, если клиенту необходимо заблокировать всплыв окна с рекламой, то лишь немногие серверы

¹⁵ Апплет (англ. applet от application — приложение и -let — уменьшительный суффикс) — это несамостоятельный компонент программного обеспечения, работающий в контексте другого, полновесного приложения, предназначенный для одной узкой задачи и не имеющий ценности в отрыве от базового приложения.

¹⁶ Не следует путать язык сценариев JavaScript с языком программирования Java; это совершенно разные языки. Язык сценариев JavaScript, первоначально называвшийся Live Script, был создан в компании Netscape (www.netscape.com). Программы, написанные на этом языке, выполняются в браузере пользователя и позволяют создавать новые всплывающие окна, перемещать рисунки и выполнять ; менее оригинальные действия. Язык же Java, созданный в компании Sun Microsystems (перешедшей с 2010 г. под контроль фирмы Oracle является платформенно -независимым языком программирования: он часто используется для создания, апплетов, загружаемых и выполняющихся в браузере пользователя.

¹⁷ CGI (Common Gateway Interface, общий интерфейс шлюза) -- протокол для запуска внешних по отношению к web-серверу приложений (CGI-скриптов), регулирует коммуникацию между HTTP-сервером и CGI-скриптами.

позволят ему это. При создании приложения, работающего на стороне сервера, или при доработке старого приложения разработчики не заботятся о работе приложения у клиента, поскольку приложение разрабатывается согласно строго заданным правилам.

Подробнее о сетевых протоколах

В протоколах семейства TCP/IP (Transport Control Protocol) и UDP(User Datagram Protocol — протокол пользовательских датаграмм¹⁸) порт — идентифицируемый номером системный ресурс, выделяемый приложению, выполняемому на некотором сетевом хосте, для связи с приложениями, выполняемыми на других сетевых хостах (в том числе с другими приложениями на этом же хосте).

Для каждого из протоколов TCP и UDP стандарт определяет возможность одновременного выделения на хосте до 65536 уникальных портов, идентифицирующихся номерами от 1 до 65535. При передаче по сети **номер порта в заголовке пакета используется** (вместе с IP-адресом хоста) для адресации конкретного приложения (и конкретного, принадлежащего ему, сетевого соединения).

В обычной клиент-серверной модели приложение либо ожидает входящих данных (или запроса на соединение; «слушает порт»; роль сервера), либо посылает данные (или запрос на соединение) на известный порт, открытый приложением-сервером (роль клиента).

По умолчанию приложению выдается порт с произвольным (например, ближайшим свободным, большим 1024) номером. При необходимости приложение может запросить конкретный (предопределённый) номер порта. Так, веб-серверы обычно открывают для ожидания соединения предопределённый

¹⁸ Дейтаграмма (англ. datagram), также датаграмма — блок информации, посланный как пакет сетевого уровня через передающую среду без предварительного установления соединения и создания виртуального канала. Датаграмма представляет собой единицу информации в протоколе (protocol data unit, PDU) для обмена информацией на сетевом (в случае протокола IP, IP-датаграммы) и транспортном (в случае протокола UDP, UDP-датаграммы) уровнях эталонной модели OSI

порт 80 протокола TCP.

Для других Internet-соединений используются другие номера портов: 22 для SSH, 23 для службы Telnet и т.д. Следует отметить, что номера IP-портов **не присваиваются физическим портам компьютера** (COM1 и COM2, USB, параллельный порт принтера и т.д.).

Порты TCP не пересекаются с портами UDP. То есть, порт 1234 протокола TCP не будет мешать обмену по UDP через порт 1234.

Ряд номеров портов стандартизован (смотри краткий список портов TCP и UDP). Список поддерживается некоммерческой организацией IANA.

В большинстве операционных систем прослушивание портов с номерами 0—1023 (почти все из которых зарегистрированы) требует особых привилегий. Каждый из остальных портов может быть захвачен первым запрошившим его процессом. Однако, зарегистрировано номеров намного больше, чем 1023.

Краткий список номеров портов для разных протоколов передачи информации

Подразумевается использование протокола TCP там, где не оговорено иное.

FTP: 21 для команд, 20 для данных

SSH: 22 (remote access)

telnet: 23 (remote access)

SMTP: 25

DNS: 53 (обычно UDP)

DHCP: 67/68

TFTP: 69/UDP

HTTP: 80, 8080

POP3: 110

NTP: 123 (time server)(UDP)

IMAP: 143

HTTPS: 443

MySQL: 3306

XMPP: 5222/5223 - клиент-сервер, 5269 - сервер-сервер

BitTorrent: 6969, 6881—6889

Подробнее о протоколе TCP

TCP — «гарантированный» транспортный механизм с предварительным установлением соединения, предоставляющий приложению надёжный поток данных, дающий уверенность в безошибочности получаемых данных, перезапрашивающий данные в случае потери и устраняющий дублирование данных. TCP позволяет регулировать нагрузку на сеть, а также уменьшать время ожидания данных при передаче на большие расстояния. Более того, TCP гарантирует, что полученные данные были отправлены точно в такой же последовательности. В этом его главное отличие от UDP.

Протокол TCP – свод правил и процедур, описывающий соответствующий транспортный механизм.

Подробнее о протоколе UDP

UDP **протокол передачи** датаграмм без установления соединения. Также его называют протоколом «ненадёжной» передачи, в смысле невозможности удостовериться в доставке сообщения адресату, а также возможного перемешивания пакетов. В приложениях, требующих гарантированной передачи данных, используется протокол TCP.

UDP обычно используется в таких приложениях, как потоковое видео и компьютерные игры, где допускается потеря пакетов, а повторный запрос затруднён или не оправдан, либо в приложениях вида запрос-ответ, где создание соединения занимает больше ресурсов, чем повторная отправка.

1.3.2. Типы данных и их обработка Web сервере

1.3.2.1. Обработка статических данных

С точки зрения сервера проще всего осуществлять обработку *статических данных* (static data), или, другими словами, данных, которые передаются в **неизменном виде всем клиентам и изменяются только в том слу-**

чае, если программист внесет изменения в исходный HTML-файл. Исходный HTML-файл (или файл рисунка, или PDF-файл) хранится на локальном жестком диске сервера, и при запросе содержимое файла в неизменном виде передается клиенту. Для этой передачи даже не требуется дополнительного программного обеспечения на сервере — всю работу способен выполнить Web-сервер Apache.

Обработка статических данных показана на Рис. 1-7. Допустим, пользователь ввел в поле адреса Web-браузера адрес `www.example.com`. Браузер соединяется с сервером `www.example.com` и производится запрос локального файла сервера (`/var/www/html/index.html`) на локальном жестком диске. После того как файл найден, он отсылается к отправителю запроса, т.е. к клиенту. Дополнительно сервер подсоединяет к файлу `index.html` дополнительный блок информации, называемый заголовком.

Обработка статических может быть рассмотрена при настройке и использовании Web-сервера Apache. Создание и управление большим количеством статических Web-страниц может осуществляться при помощи языка Website META Language (WML).

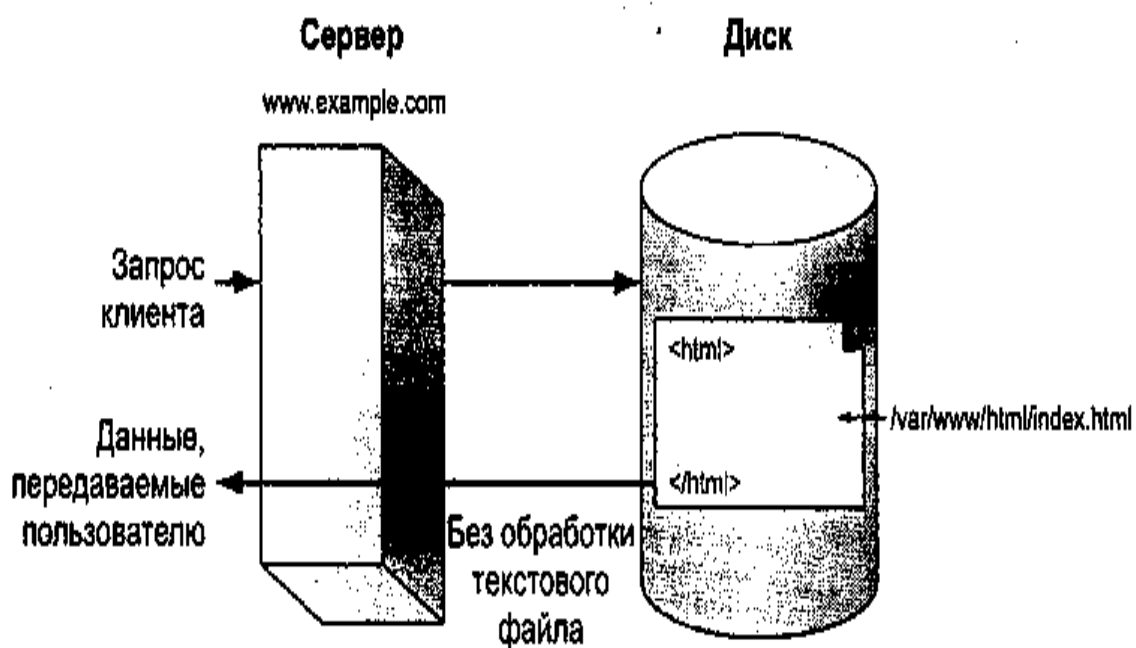


Рис. 1-7. Обработка статических данных

1.3.2.4.Обработка динамических данных

Более сложным способом генерации HTML-кода является использование программ, работающих на сервере, которые динамически создают код, посылаемый браузеру клиента. Применение *программирования на стороне сервера* (server-side programming) дает неоспоримые преимущества при создании Web-страниц, включая хорошо зарекомендовавший себя на практике интерфейс CGI, а также более гибкий и более мощный инструмент.

Существует несколько типов программ, выполняемых на стороне сервера. Некоторые из них считывают данные из баз данных, другие же запускают программы, выполняемые на стороне сервера.

Не следует путать динамические Web-страницы с динамическим HTML-кодом. Последний термин обычно используется при рассмотрении Web-страниц с динамическими компонентами, такими как всплывающие окна, изменяемые изображения, динамические элементы меню и другие зрелищные эффекты. Довольно часто в динамическом HTML-коде используются сценарии JavaScript и Document Object Model (DOM)¹⁹.

Различают два типа динамических данных: представленных в виде двоичных, так называемых, **bin-файлов** и код, внедрённый (вставленный) в HTML документ. Соответственно можно говорить о двух методах обработки динамических данных. Рассмотрим их подробнее.

А.Обработка bin-файлов

На рис. Рис. 1-8. **Обработка динамических данных** проиллюстрировано взаимодействие клиента с сервером при использовании динамических данных. Если пользователь введет в поле адреса браузера строку `www. example.com/cgi-bin/a.cgi`, то серверу (`www. example. com`) будет передан запрос на выполнение программы `a.cgi` (**сервер "знает", что запраши-**

¹⁹ http://ru.wikipedia.org/wiki/Document_Object_Model : DOM (от англ. Document Object Model — «объектная модель документа») — это не зависящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML, XHTML и XML-документов, а также изменять содержимое, структуру и оформление таких документов

вается исполняемый файл, поскольку в адресе присутствует строка `cgi-bin`).

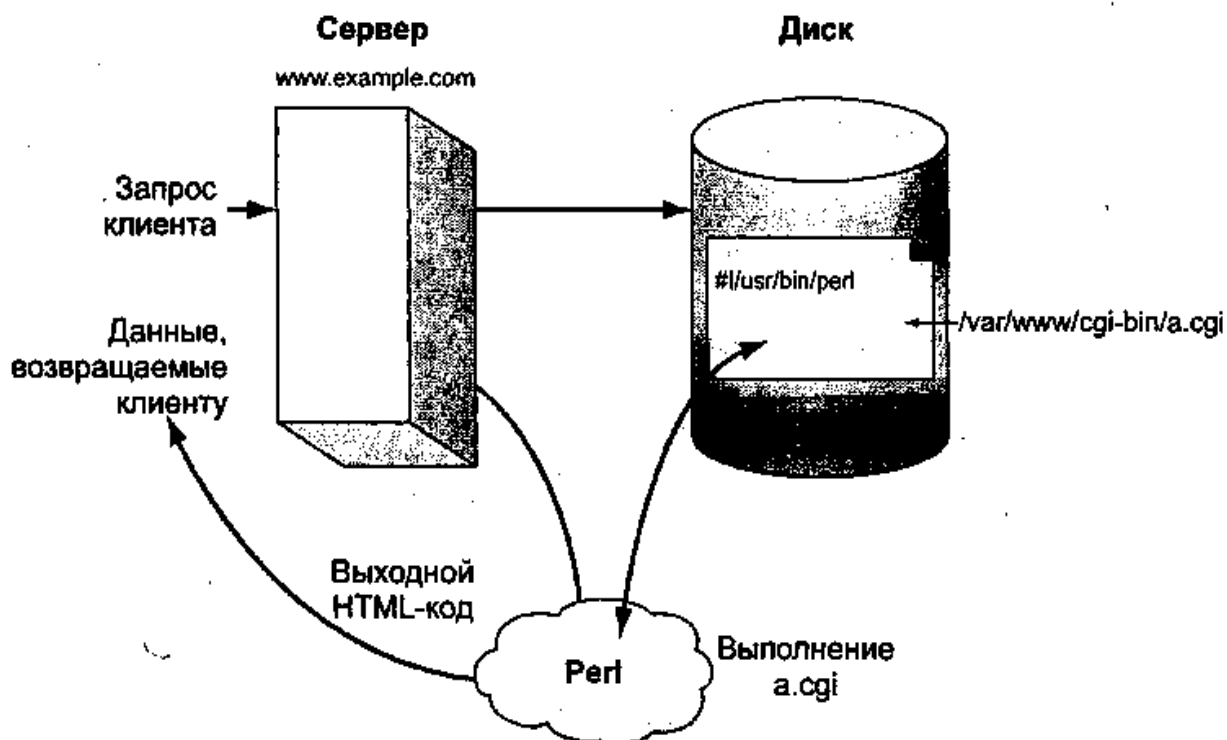


Рис. 1-8. Обработка динамических данных

Сервер устанавливает месторасположение файла на жестком диске, например, расположенного в каталоге `/var/www/cgi-bin/a.cgi`, после чего выполняет эту программу. **Результатом работы программы является HTML-код** (в ходе этого может производиться чтение из баз данных, отправка электронных писем или запись в файл системных событий), который передается клиенту.

Б. Обработка кода, внедренного в HTML

Другим, более гибким решением создания динамических Web-страниц является использование HTML с внедренным кодом или исполняемого кода внедренного в HTML -файл.

К примеру, если человек, знающий HTML, однако не умеющий программировать, создаст шаблон для Web-страницы, то далее программист

сможет добавить исполняемый код непосредственно в HTML-файл. **Не статические, но и не совсем динамические Web-страницы с внедренным кодом представляют собой очень удобное и гибкое решение.**

Внедрение кода в HTML-файл позволяет разработчику создавать удобные и отлично выглядящие Web-страницы, в отличие от посредственных, не слишком удобных страниц, создаваемых на чистом HTML²⁰. После создания структуры Web-страницы, программист добавляет исполняемый код в HTML-файл, изменяя тем самым статическое содержимое на динамическое, что придает странице более живой вид.

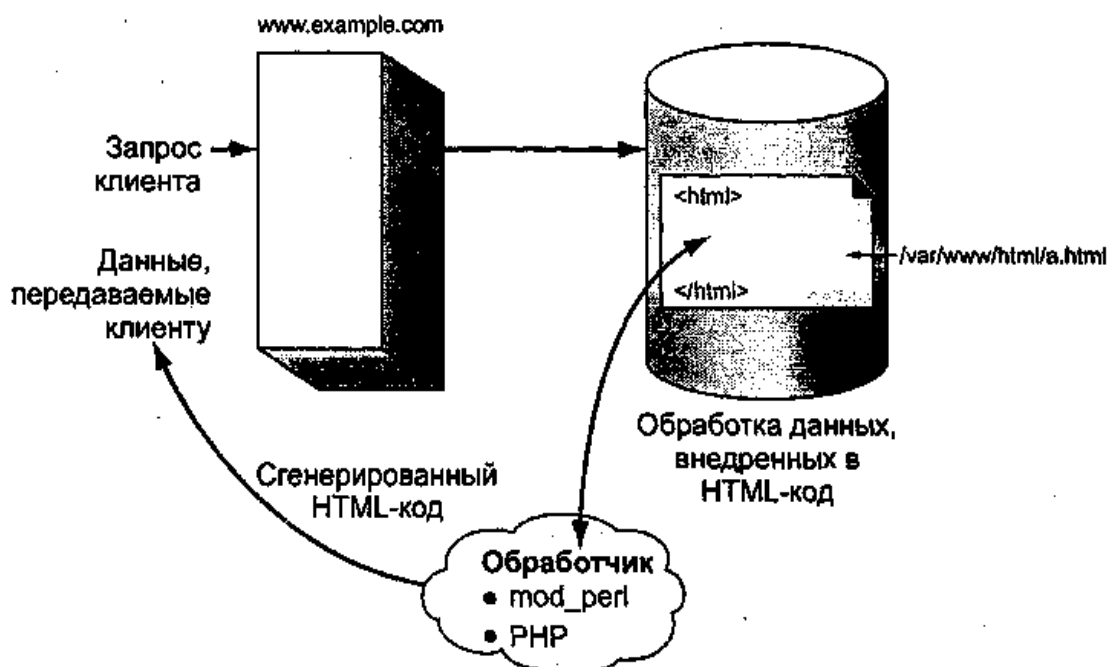


Рис. 1-9. Обработка динамических данных

На Рис. 1-9 проиллюстрирована обработка данных, внедренных в HTML-файл.

Давайте представим, что пользователь ввел в поле адреса браузера www.example.com/a.html, который имеет вставленный код на каком-либо языке программирования. Web-сервер находит HTML-файл, например, рас-

²⁰ В качестве примера можно привести Web-узел, посвященный этой книге (www.opensourceweb-book.com) разработанный командой BDGI (www.dgi.com). Более поздние примеры можно найти по адресу www.ifkr.com.

положенный в каталоге /var/www/html/, производит **его предварительную обработку**, выполняя исполняемый код оригинального HTML-файла, создает новый HTML-код, который и передается браузеру клиента.

В принципе возможны несколько вариантов внедрения кода в HTML-файлы. Упомянем всего 4:

- SSI (Server Side Includes) — простое, уже встроенное в Web-сервер Apache решение, в котором используется уникальный синтаксис.
- Embperl — модуль **Perl**, который позволяет внедрять Perl-код в HTML-файлы.
- Mason — другой модуль **Perl**, который, так же, как и Embperl, позволяет внедрять Perl-код в HTML-файлы.
- PHP — язык программирования, имеющий синтаксис, похожий на Perl и обеспечивающий большое количество встроенных функций для выполнения различных задач.

Механизм SSI проще в использовании, однако имеет ограниченные возможности. Средства Embperl, Mason и PHP предоставляют гораздо более широкие возможности.

При использовании этих инструментов появляется возможность доступа из HTML-страницы к **отправленным по почте данным формы, соединения с базами данных, записи и чтения файлов, а также выполнения любых задач, которые можно выполнить средствами программ.**

Это позволяет создавать не только привлекательные, динамичные Web-страницы, но и приложения, выполняющие разнообразные задачи.

1.3.3. О некоторых других Web технологиях

Информация о множестве существующих на сегодняшний день Web-технологий не вошла в эту книгу, поскольку в одной книге невозможно охватить все современные Web-технологии; к тому же некоторые из популярных технологий не являются продуктами с открытым исходным текстом. Ниже

описаны перспективные Web-технологии, информация о которых не вошла в книгу, однако о которых необходимо иметь представление.

- XML (Extensible Markup Language) — достаточно новый перспективный язык, который, возможно, в один прекрасный день заменит HTML. Кроме того, язык XML широко используется для представления и передачи данных в сетях.
- Java — объектно-ориентированный язык программирования, который используется для создания апплетов (программ, выполняемых браузером) и самостоятельных приложений. В языке Java *имеется* богатый программный интерфейс приложений (API), который определяет методы создания GUI-приложений, программ для работы с сокетами и др. Кроме того, язык Java имеет надстройку JSP (JavaServer Pages). Язык Java в своей основе имеет программный продукт с закрытым исходным кодом, язык программирования C++.
- *JavaScript* —Java-подобный язык, *который* имеет возможности для совершения различных действий с браузером пользователя. Язык JavaScript используется для создания новых всплывающих окон браузера (вы, очевидно, посещали Web-страницы, на которых создавались новые окна браузера с надоедливой рекламой; такие вещи проделываются именно при помощи JavaScript), перемещения изображений в окне браузера, работы с уровнями и для других остроумных действий на машине клиента.

1.3.4. Заключение. О навыках разработки серьёзных Web-- приложений

Понятие «Разработка Web приложений» относится к деятельности по использованию технологий Web для создания клиентских и серверных компонентов обработки, к интеграции их в качестве приложений в системах обработки интранет, интернет или экстранет, а также к развертыванию их в Web для реализации частной и публичной деятельности организаций.

Набор навыков для реализации этих задач простирается далеко за пределы способности сохранить страницы Web из программы текстового про-

цессора, за пределы создания простого сайта Web с помощью соответствующего программного пакета настольного компьютера и даже за пределы жестко закодированных с помощью XHTML страниц с вкраплениями подключаемых модулей.

При окончательном анализе разработчику Web необходимо проникновение в сущность операционных и управленческих процессов организации, понимание того, каким образом производственные процессы создаются и опираются на информационные потоки при производстве товаров и услуг, и способность абстрагироваться и моделировать эти бизнес системы на доступном оборудовании и программных технологиях, а также навыки использования этих технологий для создания систем на основе Web, которые реализуют эти модели.

Один курс не может вместить все это. В настоящем пособии освещаются основы гипертекстового языка разметки документа и основы программирования на языке PHP.

1.4. Контрольные вопросы к главе 1

1. Определение процесса Web программирования и его составляющие;
2. Почему World Wide Web характеризуется как проект распределённой гипертекстовой системы?
3. В чём причина популярности WWW и на чём основа эта популярность?
4. Назначение и основные характеристики современных языков гипертекстовой разметки;
5. Приведите краткие характеристики и назначение существующих технических инфраструктур обработки информации в Web – пространстве;
6. Каковы основные характеристики технологической среды, в которой функционируют Web системы?
7. Расшифруйте определения понятий сервера, клиента и концепции «клиент-сервер», используемые в описании архитектур информационных систем в целом, и Web-систем, в частности;
8. Перечислите типы двухзвенных моделей клиент-серверного взаимодействия «по Gatner Group» и охарактеризуйте их особенности. Назовите причину отнесения этих моделей к двухзвенной архитектуре.
9. В чём заключается роль сервера данных?
10. Дайте определение тонкого клиента и его места в компьютерных технологиях.
11. Основные признаки и составные части трёхзвенной системы обработки информации.
12. Опишите схему передачи и обработки информации при работе трёхуровневой Web-ориентированной информационной системы;
13. Каковы задачи разработки Web приложения?
14. Что описывает и как функционирует модель доставки информации Web приложения?
15. Опишите модель обработки информации в Web приложении, используя понятия функциональности;

16. Отобразите функции обработки информации на двух- и трёхслойные клиент-серверные системы;
17. Назначение протокола CGI;
18. Определение протоколов и характеристики основных протоколов, используемых в WWW;
19. К чему относится и как понимается “Номер порта” в информационной технологии?
20. Особенности статической и динамической информации и её обработки в информационных системах?
21. Сформулируйте различия между двумя способами представления информации в Web приложениях.

Глава 2. Основы HTML – языка гипертекстовой разметки

2.1 Введение в HTML.

2.1.1 История развития HTML

При этом сама технология создания и использования материалов на начальном этапе была чрезвычайно проста. Дело в том, что при разработке различных компонентов технологии (языка гипертекстовой разметки HTML (HyperText Markup Language, язык разметки гипертекста), протокола обмена гипертекстовой информацией HTTP, спецификации разработки прикладного программного обеспечения CGI и др.) предполагалось, что квалификация авторов информационных ресурсов и их оснащенность средствами вычислительной техники будут минимальными.

Одним из компонентов технологии создания распределенной гипертекстовой системы World Wide Web стал язык гипертекстовой разметки HTML, разработанный Тимом Бернерсом-Ли на основе стандарта языка разметки печатных документов — SGML (Standard Generalised Markup Language, стандартный обобщенный язык разметки). Дэниел В. Конноли написал для него Document Type Definition — формальное описание синтаксиса HTML в терминах SGML.

Разработчики HTML смогли решить две задачи:

- предоставить дизайнерам гипертекстовых баз данных простое средство создания документов;
- сделать это средство достаточно мощным, чтобы отразить имевшиеся на тот момент представления об интерфейсе пользователя гипертекстовых баз данных.

Первая задача была решена за счет выбора теговой модели описания документа. Такая модель широко применяется в системах подготовки

документов для печати.

Язык HTML позволяет размечать электронный документ, который отображается на экране с полиграфическим уровнем оформления; результирующий документ может содержать самые разнообразные метки, иллюстрации, аудио- и видеофрагменты и так далее. В состав языка вошли развитые средства для создания различных уровней заголовков, шрифтовых выделений, различные списки, таблицы и многое другое.

Вторым важным моментом, повлиявшим на судьбу HTML, стало то, что в качестве основы был выбран обычный текстовый файл. Выбор был сделан под влиянием следующих факторов:

- В качестве средства создания документа может выступать любой (самый простой) текстовый редактор на любой аппаратно-программной платформе. Для дальнейшей работы с помощью браузера достаточно сохранить такой документ в виде файла с расширением .htm (.html);
- К моменту разработки HTML существовал американский стандарт для разработки сетевых информационных систем — Z39.50, в котором в качестве единицы хранения указывался простой текстовый файл в кодировке LATIN1, что соответствует US ASCII.

Таким образом, гипертекстовая база данных в концепции **WWW** — это набор текстовых файлов, размеченных на языке HTML, который определяет форму представления информации (разметка) и структуру связей между этими файлами и другими информационными ресурсами (гипертекстовые ссылки). Гипертекстовые ссылки, устанавливающие связи между текстовыми документами, постепенно стали объединять самые различные информационные ресурсы, в том числе звук и видео; в результате возникло новое понятие — гипермедиа.

Такой подход предполагает наличие еще одного компонента технологии —

интерпретатора языка. Для HTML функции интерпретатора выполняет интерфейс пользователя (браузер), который осуществляет интерпретацию конструкций языка, связанных с представлением информации.

Язык HTML прошёл длительную историю развития. Первая версия языка была направлена на представление языка как такового, где описание его возможностей носило скорее рекомендательный характер. Вторая версия языка (HTML 2.0) фиксировала практику использования его конструкций. Версия HTML++ представляла новые возможности, расширяя набор тегов HTML в сторону отображения научной информации и таблиц, а также улучшения стиля компоновки изображений и текста. Версия 3.2 смогла упорядочить все нововведения и согласовать их с существующей практикой. HTML 3.2 позволяет реализовать использование таблиц, выполнение кодов языка Java, обтекание графики текстом, а также отображение верхних и нижних индексов.

Сейчас World Wide Web Consortium (W3C) — международная организация, которая занимается подготовкой и распространением документации на описание новых версий HTML — уже опубликовала материалы спецификации HTML 4.01. Кроме возможностей разметки текста, включения мультимедиа и формирования гипертекстовых связей, уже существовавших в предыдущих версиях HTML, в версию 4.01 включены дополнительные средства работы с мультимедиа, языки программирования, таблицы стилей, упрощенные средства печати изображений и документов. Для управления сценариями просмотра страниц Website (гипертекстовой базы данных, выполненной в технологии World Wide Web) можно использовать языки программирования этих сценариев, например, JavaScript, Java и VBScript.

Усложнение HTML и появление языков программирования привело к тому, что разработка Web-узлов стала делом высокопрофессиональным, требующим специализации по направлениям деятельности и постоянного изучения новых Web-технологий. Но возможности Internet позволяют пользователям,

владеющим основами HTML, создавать и размещать собственные Web-узлы без больших затрат.

2.1.2. Основные положения гипертекстовой разметки

HTML является описательным языком разметки документов, в нем используются **указатели разметки** (теги). Теговая модель описывает документ как совокупность контейнеров, каждый из которых начинается и заканчивается тегами, то есть документ HTML представляет собой не что иное, как обычный ASCII-файл, с добавленными в него управляющими **HTML-кодами** (тегами). В HTML разрешено использовать только три управляющих символа: горизонтальную табуляцию, перевод каретки и перевод строки. Это облегчает взаимодействие с различными операционными системами.

Теги HTML-документов в большинстве своем просты и понятны, так как они образованы с помощью общеупотребительных слов английского языка, понятных сокращений и обозначений.

HTML-тег состоит из **имени**, за которым может следовать **необязательный список атрибутов тега**. Текст тега заключается в угловые скобки ("<" и ">"). Простейший вариант тега — имя, заключенное в угловые скобки, например, <HEAD> или <I>. Для ряда тегов характерно наличие атрибутов, которые могут иметь конкретные значения, устанавливаемые автором для изменения функции тега.

Например, при описании таблицы открывающий тег с атрибутами может выглядеть так:

***<TABLE WIDTH=570 ALIGN=center CELLPADDING=10
CELLSPACING=2 BORDER=16>***

Эта запись означает следующее: таблица шириной 570 пикселей, выровнена по центру, поле между рамкой и содержимым ячеек 10 пикселей, поле рамки 2 пикселя, ширина бордюра 16 пикселей.

Атрибуты тега следуют за именем и отделяются друг от друга одним или несколькими знаками табуляции, пробелами или символами возврата к началу строки. Порядок записи атрибутов в теге значения не имеет.

Значение атрибута, если таковое имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута — одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в одинарные или двойные кавычки, особенно если они содержат несколько разделенных пробелами слов. Длина значения атрибута ограничена 1024 символами. **Регистр символов в именах тегов и атрибутов не учитывается.** , Однако при анализе значений атрибутов регистр символов учитывается. Например, особенно важно использовать нужный регистр при вводе URL (Uniform Resource Locator, унифицированный указатель ресурса).

Чаще всего элементы разметки HTML или HTML-контейнеры состоят из начального и конечного компонентов, между которыми размещаются текст и другие элементы документа. Имя конечного тега идентично имени начального, но перед именем конечного тега ставится косая черта (/) (например, для тега стиля шрифта — курсив *<I>* закрывающая пара представляет собой *</I>*, для тега заголовка *<TITLE>* закрывающей парой будет *</TITLE>*). Конечные теги никогда не содержат атрибутов.

По своему значению теги близки к понятию скобок "begin/end" или

фигурных скобок {... } в универсальных языках программирования, которые задают области действия имен локальных переменных и т.п. Теги определяют область действия правил интерпретации текстовых документов.

При использовании вложенных элементов разметки в документе следует соблюдать особую аккуратность. Вложенные теги нужно закрывать, начиная с последнего. Некоторые элементы разметки не имеют конечного компонента, поскольку являются автономными элементами. Например, тег изображения , который служит для вставки в документ графического изображения, конечного компонента не требует. К автономным элементам разметки также относятся разрыв строки (
), горизонтальная линейка (<HR>) и теги, содержащие такую информацию о документе, которая не влияет на его отображаемое содержимое, например, теги <META> и <BASE>.

В некоторых случаях конечные теги в документе можно опускать. Большинство браузеров устроено так, что при обработке текста документа начальный тег воспринимается как конечный тег предыдущего. Самый распространенный тег такого типа — тег абзаца <P>. Поскольку он используется в документе очень часто, его обычно ставят только в начале каждого абзаца. Когда один абзац заканчивается, следующий тег <P> сигнализирует браузеру о том, что нужно завершить данный абзац и начать следующий. Большинство авторов тегом конца абзаца не пользуются.

Есть и другие конечные теги, без которых браузеры отлично работают, например, конечный тег </HTML>. Тем не менее, рекомендуется включать по возможности больше конечных тегов, чтобы избежать путаницы и ошибок при воспроизведении документа.

Для краткости и образности мы будем в ряде случаев вместо словосочетания "элемент разметки" применять термин "контейнер".

Общая схема построения контейнера в формате HTML может быть записана в следующем виде:

```

"контейнер"=
<"имя тега" "список атрибутов">
содержание контейнера
</"имя тега">

```

Следует отметить, что в литературе кроме термина "контейнер" еще используется и термин "элемент". Следует быть внимательным, чтобы не путать контейнер (например, BODY) и тег (BODY), используемый при формировании контейнера.

Кроме тегов, элементами HTML являются подстановочные коды (CER или Character Entity Reference). Они предназначены для представления специальных символов в документе HTML, которые могут быть неверно обработаны браузером. Предположим, создается документ HTML, речь в котором идет об элементах данного языка. Если указать имя тега (к примеру -
) просто в документе, браузер может воспринять его как непосредственно старт-тег. Для вывода таких символов и используются такие ссылки.

CER легко обнаружить, если посмотреть на структуру любого документа HTML, поскольку каждый из них начинается с амперсанда "&". В отличие от наименований тегов HTML, наименования CER чувствительны к регистру символов. Также наименования CER могут задаваться не в виде имени, а с помощью трехзначных кодов символов в виде &#nnn;. Далее в таблице 2.1 приведены наиболее часто используемые CER и соответствующие им числовые коды.

Таблица 2.1.

Наиболее часто используемые CER и соответствующие им числовые коды.

Числовой код	Именная замена	Символ	Описание
"	"	"	Кавычка
&	&	&	Амперсанд
<	<	<	Меньше

Числовой код	Именная замена	Символ	Описание
>	>	>	Больше
¡	¡	¡	Перевернутый восклицательный знак
¢	¢	¢	Цент
£	£	£	Фунт
¤	¤	¤	Валюта
¥	¥	¥	Йена
¨	¨	¨	Умляют
©	©	©	Копирайт
«	«	«	Левая угловая кавычка
®	®	®	Зарегистрированная торговая марка
±	±	±	Плюс или минус
»	»	»	Правая угловая кавычка

2.1.3. Назначение и состав контейнера <HTML> ... </HTML>

Тэги <HTML> и </HTML> определяют границы **собственно HTML документа**. Все остальные элементы, описывающие HTML - документ, находятся внутри данного контейнера.

Гипертекстовый документ (или Контейнер HTML) состоит из двух других вложенных контейнеров: заголовочной части (или заголовка документа) HEAD и тела документа (BODY). Рассмотрим правила их составления подробнее.

2.2 Структура HTML – документа и используемые для этого тэги

Документ в формате HTML 4.0 состоит из трех частей:

- строки, содержащей информацию о версии HTML ;
- раздела заголовков (определяемого элементом [HEAD](#));
- тела, которое включает собственно содержимое документа. Тело

может вводиться элементом [BODY](#).

Перед каждым элементом или после каждого элемента может находиться пустое пространство (пробелы, переход на новую строку, табуляции и комментарии).

Вот пример простого документа HTML:

Раздел 1	<code><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd"></code>
Начало контейнера HTML	<code><HTML></code>
Раздел 2- заголовочная часть документа	<code><HEAD></code> <code><TITLE>Мой первый документ HTML</TITLE></code> <code></HEAD></code>
Раздел 3 -	<code><BODY></code> <code><P>Всем привет!</code> <code></BODY></code>
Конец контейнера HTML	<code></HTML></code>

Следует обратить внимание, что разделы 2 и 3 входят в состав контейнера `<HTML>...</HTML>`.

Рассмотрим подробнее части документа и теги, формирующие их структуру

2.2.1. Назначение раздела 1- дать информацию о версии HTML

При подготовке документов HTML в качестве первой строки используется

идентификатор текста DTD (Document Type Declaration - определение типа документа). Это позволяет браузеру идентифицировать документ как соответствующий стандарту HTML. Обычно (но не обязательно) каждый документ HTML начинается со строки типа:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">

В этой строке содержится информация о том, что документ соответствует версии HTML 4.0; разработанной W3C; используемый язык — английский.

Возможные варианты идентификатора текста DTD:

- [HTML 4.0 Strict DTD \(строгое определение\)](#) включает все элементы и атрибуты, не являющиеся [нежелательными](#) и не используемые в документах с кадрами. Для документов, использующих это DTD, используйте такое объявление типа документа:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"

- [HTML 4.0 Transitional DTD \(переходное определение\)](#) включает все, что включено в строгое DTD, а также нежелательные элементы и атрибуты (большинство из которых относится к визуальному представлению). Для документов, использующих это DTD, используйте такое объявление типа документа:

***<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN"***

- [HTML 4.0 Frameset DTD \(определение для кадров\)](#) включает все, что включено в переходное DTD, а также кадры. Для документов, использующих это DTD, используйте такое объявление типа:

***<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Frameset//EN"***

Этот элемент структуры HTML документа обязательным не является. По

умолчанию (если DTD явным образом не определен) в качестве версии используется последний вариант.

2.2.2. Назначение и состав контейнера <HTML> ... </HTML>

Тэги <HTML> и </HTML> определяют границы **собственно HTML документа**. Все остальные элементы, описывающие HTML - документ, находятся внутри данного контейнера.

Гипертекстовый документ (или Контейнер HTML) состоит из двух других вложенных контейнеров: заголовочной части (или заголовка документа) HEAD и тела документа (BODY). Рассмотрим правила их составления подробнее.

2.2.2.1. Заголовочная часть HTML – документа

Заголовок HTML-документа является необязательным элементом разметки. В HTML 2.0 предлагалось вообще отказаться от деления элементов документа на элементы заголовка и элементы тела документа, так как в то время в HTML не было элементов, которые использовались одновременно и в заголовке, и в теле документа. Современная практика HTML-разметки такова, что почти в каждом документе есть HTML-заголовок. Элементы заголовка HTML-документа содержатся внутри контейнера HEAD.

Основными элементами заголовка HTML – документа являются:

- TITLE (заглавие документа);
- BASE (база URL);
- ISINDEX (поисковый шаблон);
- META (метаинформация);
- LINK (общие ссылки);
- STYLE (описатели стилей);
- SCRIPT (скрипты).

Чаще всего применяются элементы TITLE, SCRIPT, STYLE.

Использование элемента META говорит об осведомленности автора о правилах индексирования документов в поисковых системах и возможности управления HTTP-обменом данными. BASE и ISINDEX в последнее время практически не применяются. LINK указывают только при использовании внешних относительно данного документа описателей стилей.

Элемент разметки TITLE

Элемент разметки TITLE служит для именования документа в World Wide Web. Более прозаическое его назначение — именование окна браузера, в котором просматривается документ. Состоит контейнер из тега начала, содержания и тега конца. Наличие тега конца обязательно. Тег начала элемента не имеет специфических атрибутов.

Синтаксис контейнера TITLE в общем виде выглядит следующим образом:

<TITLE>название документа</TITLE>

Заглавие не является обязательным контейнером документа. Его можно опустить. Однако следует отметить, что роботы многих поисковых систем используют содержание элемента TITLE для создания поискового образа документа. Слова из TITLE попадают в индекс поисковой системы. Из этих соображений элемент TITLE всегда рекомендуется использовать на страницах внешнего Web-узла.

Элемент разметки BASE

Элемент разметки BASE служит для определения базового URL для гипертекстовых ссылок документа, заданных в неполной (частичной) форме. Кроме того, BASE позволяет определить мишень (окно) загрузки документа по

умолчанию при выборе гипертекстовой ссылки текущего документа. Подробнее об использовании этого элемента рассказывается в лекции “Гиперссылки”

Элемент разметки META

Это наиболее популярный элемент разметки заголовка, более распространен только элемент TITLE. Такое положение дел объясняется назначением данного элемента разметки. META содержит управляющую информацию, которую браузер использует для правильного отображения и обработки содержания тела документа.

Остановимся на наиболее ярких примерах использования данного контейнера:

- 1. Описание поискового образа документа.** Собственно, для описания документа используется два META-тега. Один определяет список ключевых слов, а второй – реферат (краткое содержание документа), который отображается в качестве пояснения к ссылке на документ в отчете поисковой машины о выполненном запросе. Контейнер TITLE здесь также используется в качестве названия документа.

```
<TITLE>Основы Web-технологий</TITLE>
<META NAME="description"
http-equiv="description"
content="Учебный курс Разработка WEB-приложений.
Тема: Структура HTML-документа. Элемент
разметки META. Дается краткое описание
основных способов применения контейнера META
в заголовке HTML-документа. Рассматривается
управление HTTP-обменом и индексирование
документов.">
<META NAME="keywords" HTTP-EQUIV="keywords"
CONTENT="учебный курс; Web-технология; web;
технология; HTML; язык гипертекстовой разметки;
заголовок HTML-документа; заголовок; HTML;
документ; контейнер; META; элемент; HEAD;
пример; разметка; методика">
```


При индексировании такого документа содержимое контейнера TITLE и атрибутов CONTENT контейнеров META после фильтрации попадет в индекс поисковой машины и может быть использовано для составления запросов. Процесс фильтрации отбракует так называемые stop-слова и общие слова. Они не попадут в индекс поисковой машины. В частности, будут отбракованы предлоги или, если речь идет о тематическом поисковом индексе, например по технологиям World Wide Web, то в него не попадут: web, Web-технология и т.п.

- 2. Указание типа кодировки документа.** В Windows 95 и Windows NT 4.0 с поддержкой таблиц UNICODE появилась возможность указывать тип кодировки документа - CHARSET. К сожалению, на многих Unix-платформах этот механизм не работает, что часто приводит к ошибкам, например в IRIX версий 6.2-6.4. Скептическое отношение поклонников Unix к этой возможности ничем не подкреплено, так как основная масса пользователей российской части Internet просматривает документы World Wide Web в Windows. Для перекодировки на стороне клиента (документ подготовлен в кодировке cp1251) в заголовок документа необходимо включить META-тег следующего вида:

```
<META HTTP-EQUIV="Content-type"  
CONTENT="text/html;  
CHARSET=windows-1251">
```

Элемент разметки LINK

Элемент разметки LINK – это результат давно предпринятой попытки придать HTML академический вид. Согласно теории гипертекстовых систем, все гипертекстовые связи разделяют на два типа: контекстные и общие. Такое деление чисто условное и определяется тем, что контекстную связь можно привязать к определенному месту документа, а общую — отнести только ко всему документу целиком. Если взглянуть на проблему связи чуть шире, то

очевидной становится аналогия с отношениями. Гипертекстовая связь задает отношение на множестве информационных узлов.

Контекстная связь определяет отношение на паре узлов. При этом в модели World Wide Web один из узлов является источником, а второй — мишенью. Собственно, это и отражено в названии элемента разметки A (anchor), который определяет гипертекстовую ссылку (не путать с гипертекстовой связью). При этом в контекстной связи один и тот же термин может идентифицировать разные связи. Например, в контексте содержания конспекта данной темы слово "HEAD" определяет документ head.htm, который описывает контейнер HEAD и особенности его применения, а в контексте справочника по данной теме слово "HEAD" будет означать ссылку на описание синтаксиса этого контейнера.

Общие ссылки нельзя привязать по контексту. Например, два информационных узла находятся в отношении следования, т.е. при "линейном" просмотре одна Web-страница является следующей для другой Web-страницы. В этом случае речь идет о страницах целиком, а не об отдельных их частях. Такой же общей связью является принадлежность к Web-узлу, который ассоциируется со своей домашней страницей.

Осмысленное применение контейнер LINK поучил после реализации поддержки описателей стилей в Netscape Navigator и Internet Explorer четвертых версий. CSS (Cascade STYLE Sheets, каскадные таблицы стилей). Он позволил загружать внешние описатели стилей:

```
<LINK REL=stylesheet href='../css/style.css'  
      TYPE="text/css">
```

В данном случае речь идет о загрузке стилей из файла style.css. При этом стили задаются в нотации W3C. В сущности, атрибут REL определяет тип гипертекстовой связи, HREF (Hypertext REference) указывает адрес документа,

идентифицирующего связь, а атрибут TYPE определяет тип содержания этого документа.

Элемент разметки STYLE

Элемент разметки STYLE, также как и предыдущий, предназначен для размещения описателей стилей. При этом описание стиля из данного элемента разметки, если оно совпадает по имени класса и/или идентификатору подкласса со стилем, описанным во внешнем файле, заменяет описание стиля из внешнего файла. С точки зрения влияния на весь документ, описатели стилей задают правила отображения контейнеров HTML-документа для всей страницы.

В настоящее время контейнер используется только с одним атрибутом TYPE, который задает тип описателя стиля. Это может быть либо text/css , либо text/javascript. Если элемент разметки открыт тегом начала, то он должен быть закрыт тегом конца. В общем виде запись элемента STYLE выглядит так:

```
<STYLE TYPE=тип_описания_стилей>  
описание_стиля/стилей  
</STYLE>
```

Применению стилей в HTML-разметке, а также проектированию Web-узлов с применением CSS посвящена отдельная лекция "Применение каскадных таблиц и стилей".

Элемент разметки SCRIPT

Элемент разметки SCRIPT служит для размещения внутри HTML-документа кода JavaScript, VBScript или Jscript. Соответствующие скрипты придают документу определенную интерактивность, делают страницу динамической. Вообще говоря, SCRIPT можно использовать не только в заголовке документа, но и в его теле. В отличие от контейнера STYLE, ему не требуется дополнительный контейнер LINK для загрузки внешних файлов

кодов. Это можно сделать непосредственно в самом контейнере SCRIPT:

***<SCRIPT LANGUAGE="JavaScript"
SRC=script.code>***

Если открыт тег начала, то нужно обязательно использовать тег конца контейнера. В противном случае, браузер может отобразить только символ "]". Если код не помещен в HTML-комментарии, то старые версии браузеров (до Mozilla 2) отображают программу перед текстом страницы. В ряде случаев страница вообще может не отображаться.

В общем виде запись контейнера выглядит следующим образом:

***<SCRIPT [TYPE=тип_языка_программирования]
[SRC=URL]>
JavaScript/VBScript-код
</SCRIPT>***

Существует несколько скриптовых языков: JavaScript, VBScript, JScript. По умолчанию подразумевается JavaScript. Подробнее с JavaScript и контейнером SCRIPT можно ознакомиться в лекциях раздела "Введение в JavaScript".

2.2.2.2. Тело HTML документа: структурные элементы и описывающие их тэги

Тело гипертекстового документа содержит теги, описывающие структурные части обычного документа: заголовки, абзацы, таблицы, рисунки и т.д..

Описание тегов тела документа следует начать с тега BODY.

Заголовки в тексте

Заголовок обозначает начало раздела документа. В стандарте определено 6 уровней заголовков: от H1 до H6. Текст, окруженный тегами <H1></H1>, получается большим — это основной заголовок. Если текст окружен тегами <H2></H2>, то он выглядит несколько меньше (подзаголовок); текст внутри <H3></H3> еще меньше и так далее до <H6></H6>. Некоторые программы позволяют использовать большее число заголовков, однако реально более трех уровней встречается редко, а более 5 — крайне редко.

Ниже на рисунке показан код и результат использования следующих заголовков <H1> и <H2>:

<H1>Заголовок 1</H1>

<H2>Заголовок 2</H2>

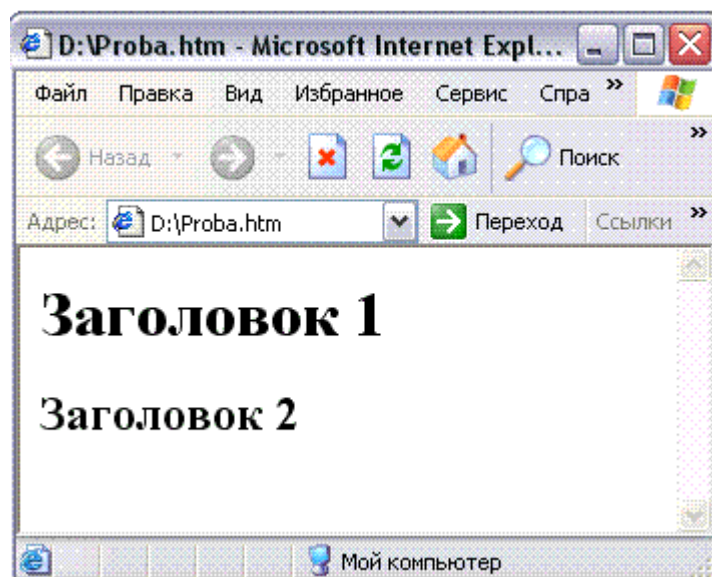


Рис. 2.1. Результат использования элементов <h1> и <h

Абзацы (параграфы)

Элемент <P> применяется для разделения текста на абзацы. К каждому абзацу можно применить собственные параметры форматирования:

- выравнивание (по левому краю, по правому краю, по центру, по ширине);
- отступы (слева, справа, первой строки);
- интервалы (перед абзацем, после абзаца, между строк внутри абзаца);
- маркированный список;
- нумерованный список.

Таблицы

Таблицы — едва ли не самый противоречивый с точки зрения их включения в структуру документа элемент. Они могут использоваться как для разметки соответствующего документа, так и для оформления определенного элемента текста. Для описания таблиц используется элемент <TABLE>.

Рисунки

Для того чтобы вставить в Web-страницу изображение, необходимо либо нарисовать его, либо взять уже готовое. В любой программе рисования можно создать простое изображение и сохранить его в нужном формате. Если программа этот формат не поддерживает, необходимо преобразовать файл в требуемый формат. Существует множество программ, предназначенных для преобразования одного графического формата в другой. Позаимствовать же картинки можно из различных программных пакетов или с других Web-страниц в Internet, содержащих библиотеки свободного доступа художественных изображений. Когда браузер выводит Web-страницу с изображением, соответствующий графический файл временно хранится в памяти компьютера. В большинстве браузеров есть команда, позволяющая сохранить файл на

локальном диске. Существует также множество других вариантов получения графических файлов.

Изображения могут нести определенную информацию, да и просто придают Web-странице привлекательный вид. Приведем наиболее распространенные случаи применения изображений:

- логотип компании на деловой странице;
- графика для рекламного объявления;
- различные рисунки;
- диаграммы и графики;
- художественные шрифты;
- подпись автора страницы;
- применение графической строки в качестве горизонтальной разделительной линии;
- применение графических маркеров для создания красивых маркированных списков.

Тегом HTML, который заставляет браузер выводить изображение, является `` с обязательным атрибутом `SRC` (SouRCe, источник). Имя файла представляет собой имя выводимого графического файла. Замыкающего тега не требуется.

Пример

вставки

изображения:

``

Гиперссылки

Гиперссылка представляет собой соединение одного Web-ресурса с другим, в качестве которого может выступать другой документ, другое место в данном документе, внешний файл любого формата (изображение, видеоклип, аудиофрагмент, программа и др.). Несмотря на простоту концепции, введение гиперссылки в практику явилось одним из фундаментальных факторов, приведших к успешному развитию Web.

Приведенный ниже пример демонстрирует гиперссылку на начало файла primer.htm, который находится внутри базовой папки (вводится с помощью элемента <BASE>), привязанную к тексту «Пример ссылки»

Пример ссылки

Если же необходимо осуществить переход к конкретному месту в документе, то это место должно быть помечено с помощью закладки («якоря»):

***Текст к которому должен осуществиться
переход***

где «yacor» - имя закладки. И гиперссылка в этом случае будет выглядеть следующим образом:

Пример ссылки

HTML формы

Формы были созданы и используются в WWW для получения отклика пользователя на предоставленную информацию и сбора данных о пользователе. После заполнения пользователем формы и запуска процесса ее обработки информация из нее попадает к программе, работающей на сервере. Простота

использования тега `<MAILTO:>` в формах позволяет даже владельцам небольших страниц получать отклик от своих читателей. Для обработки большого количества откликов используются программы, поддерживающие Common Gateway Interface (CGI) и расположенные на сервере, в адрес которого поступают отклики. Таким образом пользователь может интерактивно взаимодействовать с Web-сервером через Internet.

Для определения документа (или части документа) как формы используется элемент `<FORM>` внутри которого с помощью соответствующих тегов описываются элементы управления формы. Подробнее об этом будет говориться в лекции «Формы HTML»/

Пользовательские элементы структуры.

Контейнеры `` и `<DIV></DIV>` используются для выделения произвольного фрагмента текста как элемента структуры HTML-документа. К таким элементам могут быть применены все атрибуты, характерные стандартным элементам структуры (`<H1>`, `<P>`, `<TABLE>`, `<FORM>`, `` и др.), включая возможность присвоения идентификатора (атрибут `ID`) и пользовательского стиля из CSS-таблицы (атрибут `CLASS`).

2.3. Разметка, обеспечивающая форматирование HTML – документа

2.3.1. Понятие форматирования

Для отображения не отформатированного текст достаточно просто ввести его между тегами начала и конца соответствующего структурного элемента тела документа. При обработке такой страницы браузер найдет и выведет весь этот текст. **Если необходимо чтобы к тексту было применено какое-либо форматирование, например, выделение полужирным или курсивом, необходимо использовать соответствующие атрибуты и теги, отвечающие за форматирование.**

Можно не использовать атрибутов, определяющих форматирование. В этом случае браузер будет использовать для такого атрибута значение по умолчанию. Однако, поскольку для различных браузеров такое значение может отличаться, внешний вид созданного документа на компьютере клиента в этом случае становится непредсказуемым.

2.3.2. Форматирование страницы и текста

Применение параметров форматирования для страницы в целом осуществляется путем указания **определенных атрибутов для тега <BODY>**. Некоторые из них описаны ниже.

2.3.2.1. Цвет фона страницы (атрибут BGCOLOR)

Атрибут BGCOLOR определяет цвет фона, на котором отображается текст документа. Формальные обозначения для наиболее часто используемых цветов приведены в таблице 2.2. Следует отметить, что в качестве значения атрибута может выступать как код цвета, так и его название:

<BODY BGCOLOR=#FFFFFF>

или

<BODY BGCOLOR=white>

Таблица 2.2.

Принятые в HTML формальные обозначения некоторых цветов.

Название	Код	Название	Код
aqua	#00FFFF	navy	#000080
black	#000000	olive	#808000
blue	#0000FF	purple	#800080
fuchsia	#FF00FF	red	#FF0000
gray	#808080	silver	#C0C0C0
green	#008000	teal	#008080
lime	#00FF00	white	#FFFFFF
maroon	#800000	yellow	#FFFF00

2.3.2.2. Использование в качестве фона страницы изображения (атрибут BACKGROUND)

Атрибут **BACKGROUND** позволяет определить в качестве фона, на котором отображается текст документа изображение, предварительно созданное с помощью любого графического редактора и сохраненное в виде файла с расширением .gif или .jpg :

<BODY BACKGROUND="image.gif">

Как видно из этого примера, в качестве значения данного атрибута используется адрес в сокращенной форме URL. Напоминаем, что для хранения информации о базовом URL (начиная с которого идет поиск требуемого файла) используется элемент BASE заголовочной части документа. Если базовый URL не будет явным образом определен, то в качестве такового будет восприниматься папка, в которой сохранен сам файл HTML. Разумеется, никто не мешает указать адрес соответствующего файла полностью:

<BODY BACKGROUND=file:///E:/Foto/image.gif>

или

<BODY BACKGROUND=http://informnn.com/Foto/image.gif>

однако это не всегда удобно, если идет речь о перемещении соответствующей страницы.

2.3.2.3. Цвет символов текста (атрибут TEXT)

Атрибут TEXT определяет цвет символов текста в HTML - документе. Формальные обозначения для наиболее часто используемых цветов приведены в таблице 4.1.

<BODY TEXT=#000000>

или

<BODY TEXT=black>

2.3.2.4. Цвет символов гиперссылки (атрибут LINK)

Атрибут LINK определяет цвет символов для гиперссылки до момента первого перехода по ней с данного компьютера. Формальные обозначения для наиболее часто используемых цветов приведены в таблице 4.1.

<BODY LINK=#0000FF>

или

<BODY LINK=blue>

2.3.2.5. Цвет символов активной гиперссылки (атрибут ALINK)

Активной называют гиперссылку на которую наведен указатель мыши. Цвет символов для такой гиперссылки определяет **атрибут ALINK**. Формальные обозначения для наиболее часто используемых цветов приведены в таблице 4.1.

<BODY ALINK=#FF0000>

или

<BODY ALINK=red>

2.3.2.6. Цвет символов гиперссылки после перехода (атрибут VLINK)

Атрибут VLINK определяет цвет символов для гиперссылки после первого перехода по ней с данного компьютера. Формальные обозначения для наиболее часто используемых цветов приведены в таблице 3.1.

<BODY VLINK=#800080>

или

<BODY VLINK=purple>

2.3.3. Форматирование абзаца

Уточним, что под абзацем понимается законченная со смысловой точки зрения часть текста (а не часть текста между тегами <P></P>). Именно поэтому речь здесь пойдет не только об атрибутах тега <P>, но и о некоторых дополнительных тегах, позволяющих оформлять специфические по своей сути абзацы.

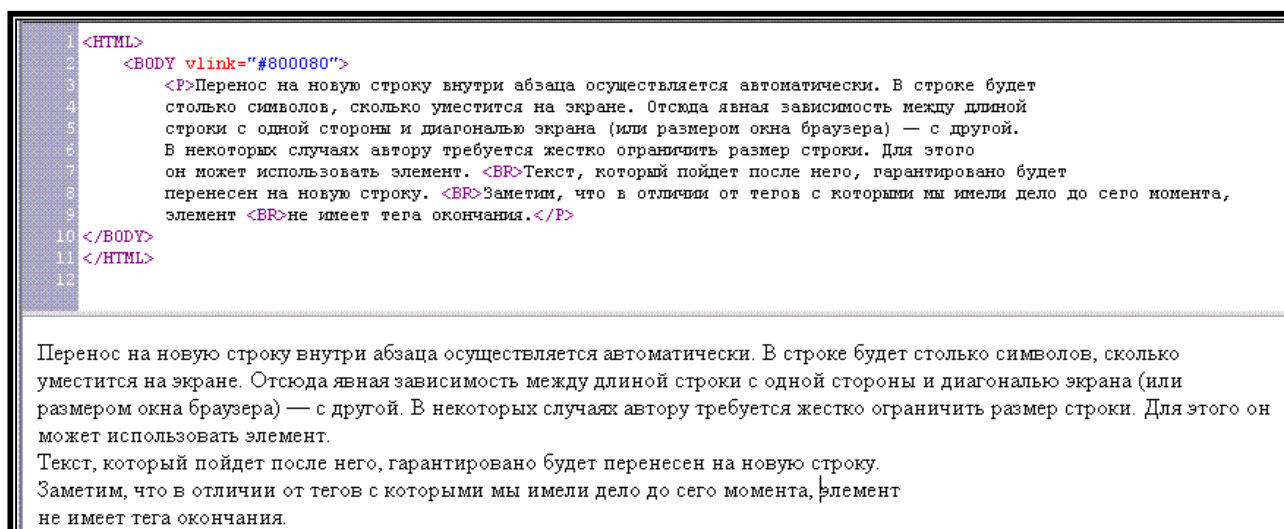
2.3.3.1. Выравнивание (атрибут ALIGN)

Атрибут ALIGN тега <P> позволяет установить для соответствующего абзаца горизонтальное выравнивание. Возможные варианты значения:

- ✓ left – по левому краю;
- ✓ center – по центру;
- ✓ right – по правому краю;
- ✓ justify – по ширине.

2.3.3.2. Принудительный перенос на новую строку (элемент
)

Перенос на новую строку внутри абзаца осуществляется автоматически. В строке будет столько символов, сколько уместится на экране. Отсюда явная зависимость между длиной строки с одной стороны и диагональю экрана (или размером окна браузера) — с другой. В некоторых случаях автору требуется жестко ограничить размер строки. Для этого он может использовать элемент
. Текст, который пойдет после него, гарантировано будет перенесен на новую строку. Заметим, что в отличие от тегов с которыми мы имели дело до сего момента, элемент
 не имеет тега окончания. На рисунке 2.2 Вы можете увидеть пример применения этого тега (в верхней части окна — код HTML, в нижней — вид документа в браузере).



**Рис. 2.2. Пример использования элемента
**

2.3.3.3. Оформление цитат (элемент <BLOCKQUOTE>)

Элемент <BLOCKQUOTE> обычно используется при цитировании текста, взятого из другого источника. Большинство браузеров выделяют такой фрагмент текста от остального пустыми строками и отступами. Пример использования данного элемента — рисунок 2.2.

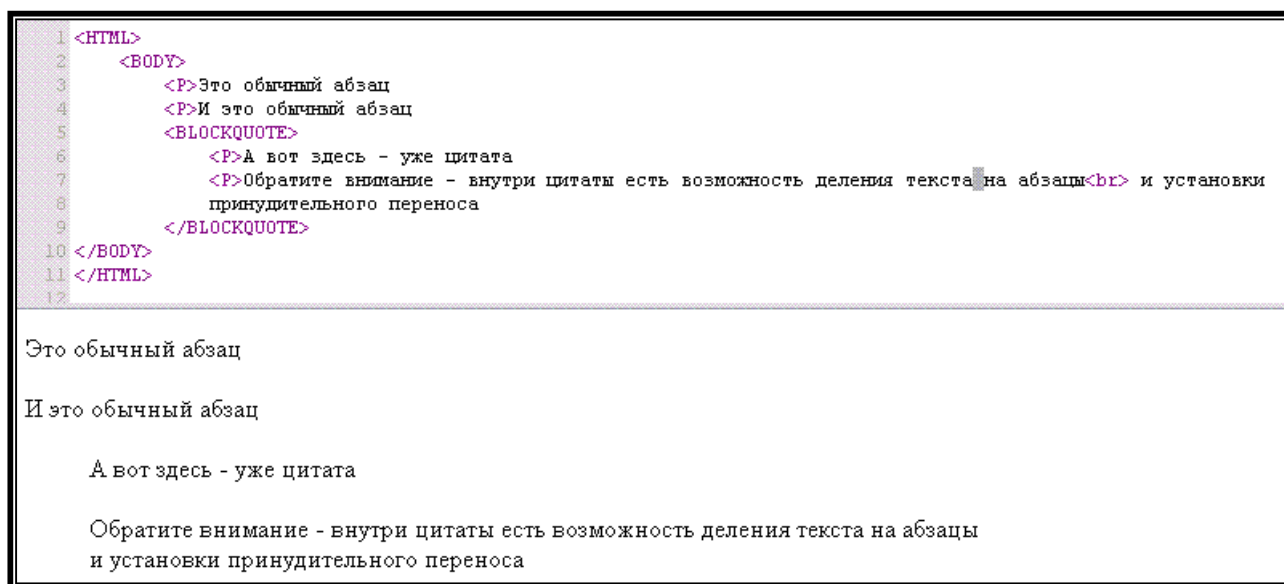


Рис. 2.3. Пример использования элемента <BLOCKQUOTE>

2.3.3.4. Маркированные списки (элемент)

Совокупность абзацев, каждый из которых должен быть оформлен как

элемент маркированного списка, помещается внутрь контейнера ``. С помощью атрибута `TYPE` тега `` можно выбрать стиль маркера для создаваемого списка.

Возможные варианты значения атрибута:

- `disc` — маркер-диск (по умолчанию) (●)
- `circle` — маркер-круг (○)
- `square` — маркер-квадрат (■)

Каждый элемент списка помещается в контейнер `` (рис. 2.4).

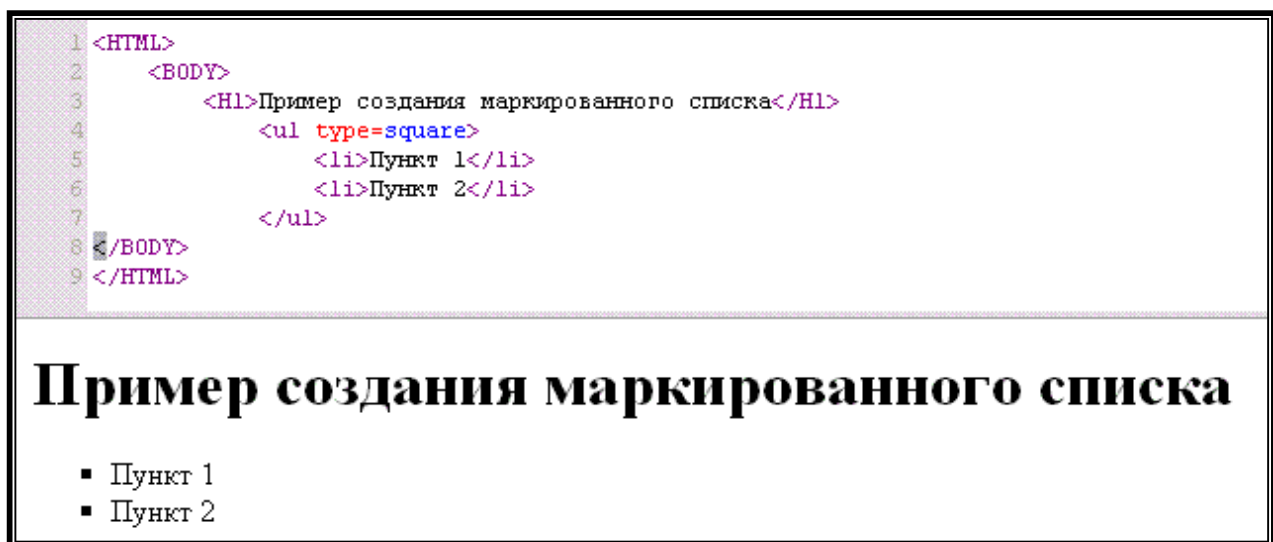


Рис. 2.4. Пример создания маркированного списка.

2.3.3.5. Нумерованные списки (элемент ``)

Совокупность абзацев, каждый из которых должен быть оформлен как элемент нумерованного списка, помещается внутрь контейнера ``. С помощью атрибута `TYPE` тега `` можно выбрать стиль маркера для создаваемого списка.

Возможные варианты значения атрибута:

- **A** - устанавливает маркер в виде прописных букв
- **a** - устанавливает маркер в виде строчных букв
- **I** - устанавливает маркер в виде больших римских цифр
- **i** - устанавливает маркер в виде маленьких римских цифр
- **1** - устанавливает маркер в виде арабских цифр (по умолчанию)

С помощью следующих атрибутов можно дополнительно управлять отображением списка

- **START = n** - устанавливает начальный маркер в текущем списке; n - номер, с которого начинается нумерация в списке
- **COMPACT** - Представляет список в более компактном виде

Каждый элемент списка помещается в контейнер `` (рис. 2.5).

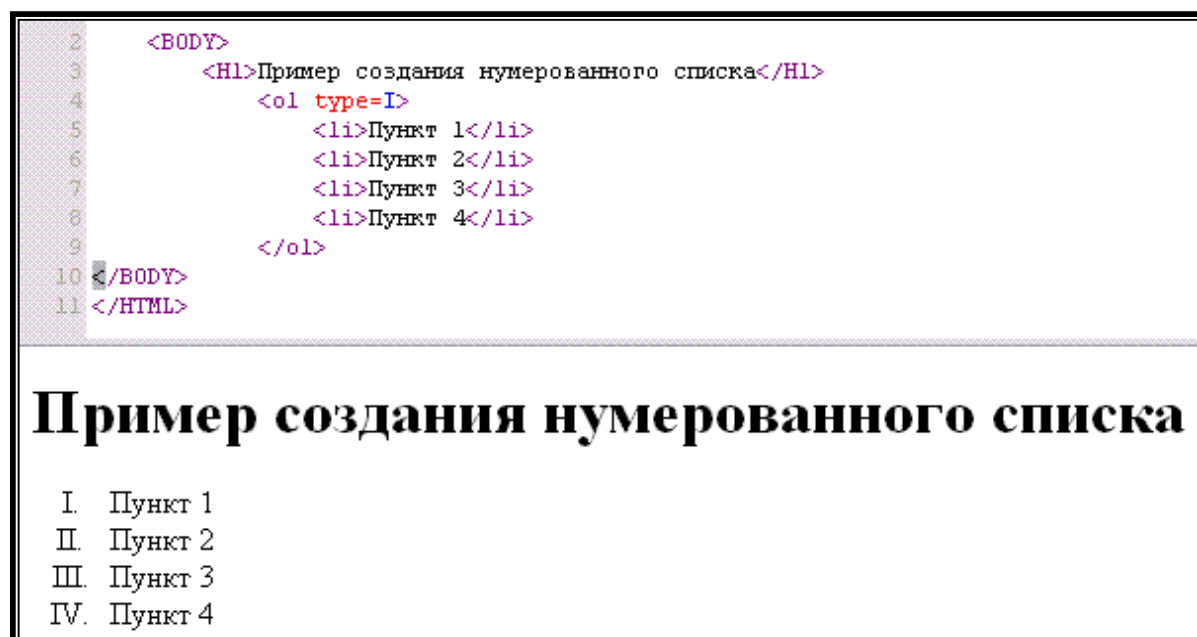


Рис. 2.5. Пример создания нумерованного списка

2.3.3.6. Многоуровневые списки

Для оформления части текста как многоуровневого списка достаточно лишь включить контейнер для создания списка второго уровня внутри контейнера для создания списка первого уровня (рис. 2.6).

```

1 <HTML>
2   <BODY>
3     <H1>Пример создания многоуровневого списка</H1>
4     <OL Type=I>
5       <LI>Пункт 1</LI>
6       <LI>Пункт 2</LI>
7         <UL Type=disc>
8           <LI>Пункт 2.1</LI>
9           <LI>Пункт 2.2</LI>
10        </UL>
11      <LI>Пункт 3</LI>
12    </ol>
13  </BODY>
14 </HTML>

```

Пример создания многоуровневого списка

- I. Пункт 1
- II. Пункт 2
 - Пункт 2.1
 - Пункт 2.2
- III. Пункт 3

Рис. 2.6. Пример создания нумерованного списка

2.3.4. Форматирование произвольного фрагмента текста.

Параметры форматирования, описываемые в этом параграфе, можно применить к любой последовательности символов. Необходимо лишь заключить ее в соответствующий контейнер.

2.3.4.1. Параметры шрифта (элемент)

Элемент позволяет управлять параметрами шрифтов.

Гарнитура шрифта устанавливается с помощью атрибута FACE, значение которого — название желаемой гарнитуры шрифта (Arial, Tahoma, Vernada). При необходимости использования нестандартной гарнитуры шрифта рекомендуется указывать через запятую несколько альтернативных вариантов — на случай если определенный шрифт не будет установлен на компьютере клиента. В этом случае будет применяться первый из найденных на компьютере клиента шрифтов.

Размер шрифта задается с помощью атрибута SIZE. В качестве значения атрибута указывается порядковый номер размера (1 — 7). Примерное соответствие порядкового номера шрифта и размера шрифта в пт (единица измерения принятая во всех текстовых редакторах) для браузера Internet Explorer приведено в таблице 2.3.

Таблица 2.3.

Размер шрифта

Номер размера	Размер в пт
1	7
2	10
3	12
4	14
5	18
6	24
7	36

Еще одним атрибутом тега , который можно использовать на практике является COLOR, позволяющий указать цвет шрифта для выделяемого фрагмента.

2.3.4.2. Размер шрифта (элементы <BIG> и <SMALL>)

Достаточно удобным вариантом выделения размером определенного фрагмента текста является использование элементов разметки <BIG> и <SMALL>.

Элемент <BIG> увеличивает размер шрифта на единицу по сравнению с обычным текстом (3). Допускается применение вложенных тегов <BIG>, при этом размер шрифта будет больше с каждым уровнем, но не может быть больше, чем 7.

Элемент **<SMALL>** уменьшает размер шрифта на единицу по сравнению с обычным текстом (3). Допускается применение вложенных тегов **<SMALL>**, при этом размер шрифта будет меньше с каждым уровнем, при этом размер шрифта будет меньше с каждым вложенным уровнем, но не может быть меньше, чем 1.

2.3.4.3. Выделение курсивом (элемент **<I>)**

Курсивное (наклонное) начертание символов осуществляется с помощью элемента разметки **<I>**.

2.3.4.4. Выделение жирным (элемент **)**

Помещенный внутри контейнера **** текст отображается браузерами жирным начертанием.

2.3.4.5. Верхние и нижние индексы (элементы **<SUB> и **<SUP>**)**

Элемент **<SUB>** позволяет отобразить последовательность символов в виде нижнего индекса. Текст при этом располагается ниже базовой линии остальных символов строки и уменьшенного размера.

Элемент **<SUP>** позволяет отобразить последовательность символов в виде верхнего индекса. Текст при этом располагается выше базовой линии остальных символов строки и уменьшенного размера.

2.3.4.6. Подчеркивание и перечеркивание (элементы **<U> и **<STRIKE>**)**

Элемент **<U>** позволяет отобразить текст как подчеркнутый, а элемент **<STRIKE>** - как перечеркнутый.

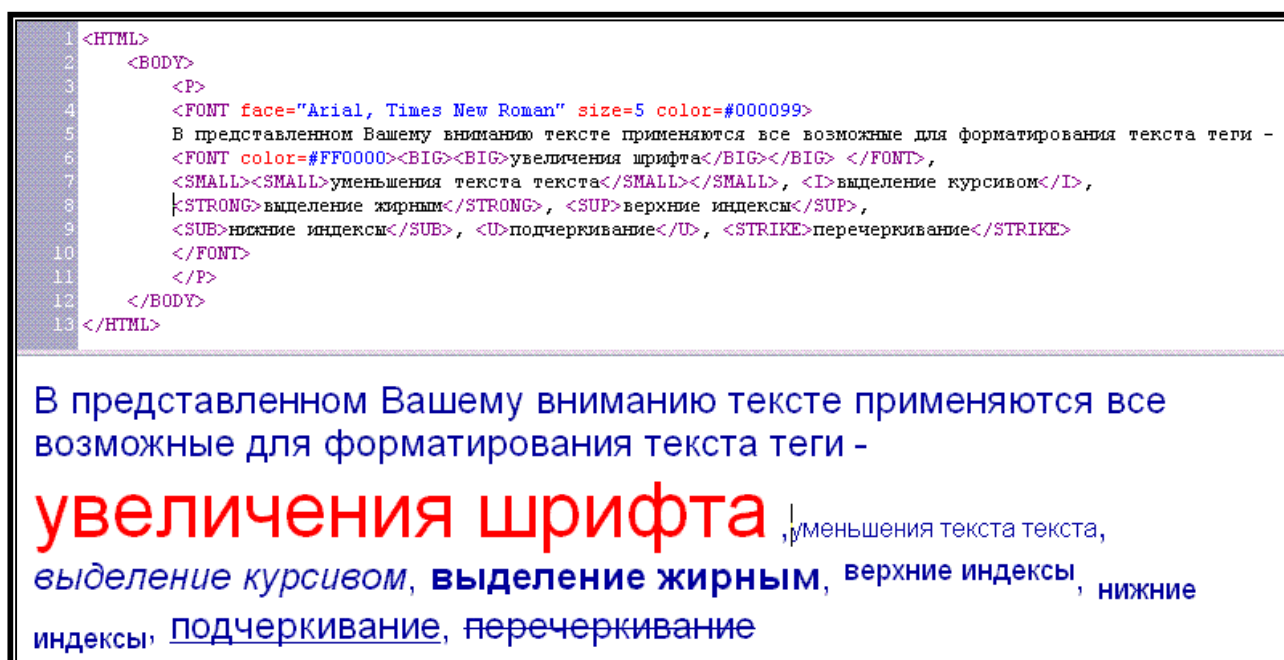


Рис. 2.7. Пример форматирования произвольных фрагментов текста

2.3.4.7. Авторское форматирование (элемент <PRE>)

В общем случае, текст документа формируется браузером, игнорируя пробелы и переносы строк. Используя элемент разметки <PRE> можно описать в тексте заданный авторский стиль (то есть пробелы и пустые строки существующие в исходном HTML-коде будут интерпретироваться браузером именно так).

Внутри контейнера <PRE> Можно применять все описанные выше параметры форматирования произвольных фрагментов текста.

```

1 <HTML>
2   <BODY>
3     <H1>Без применения авторского форматирования</H1>
4     <P> <font face=arial size=4>
5       Мой дядя <u>самых честных правил</u>
6
7       Когда не в шутку занемог
8
9       Он уважать себя заставил
10    </font></P>
11    <H1>С авторским форматированием</H1>
12    <P> <PRE> <font face=arial size=4>
13      Мой дядя <u>самых честных правил</u>
14
15      Когда не в шутку занемог
16
17      Он уважать себя заставил
18    </font></PRE></P>
19
20  </BODY>
21 </HTML>

```

Без применения авторского форматирования

Мой дядя самых честных правил Когда не в шутку занемог Он уважать себя заставил

С авторским форматированием

Мой дядя самых честных правил

Когда не в шутку занемог

Он уважать себя заставил

Рис. 2.8. Применение авторского форматирования

2.4. Форматирование гипертекстовых ссылок и закладок

2.4.1. Гиперссылки

Как отмечалось выше (п. 2.2.2) под гипертекстовой ссылкой (гиперссылкой) понимают соединение одного Web-ресурса с другим. Для создания гиперссылки используется элемент разметки <A>, называемый также «якорем». Заметим, что при создании гиперссылки зачастую говорят о «якорях» двух видов :

- **начальном** (ставится в месте с которого необходимо осуществить переход) и

- конечном (помечающим место к которому необходимо осуществить переход).

Последний также часто называют **закладкой**. Конечный якорь не является обязательным. Если речь не идет о переходе к определенному месту в документе — его можно опустить.

Направление гиперссылки задается с помощью атрибута HREF начального якоря. В качестве значения атрибута выступает URI ресурса на который необходимо осуществить переход. Приведенный ниже пример содержит две ссылки, одна из которых указывает на документ HTML с названием "chapter2.html", а другая - на GIF-изображение в файле "forest.gif":

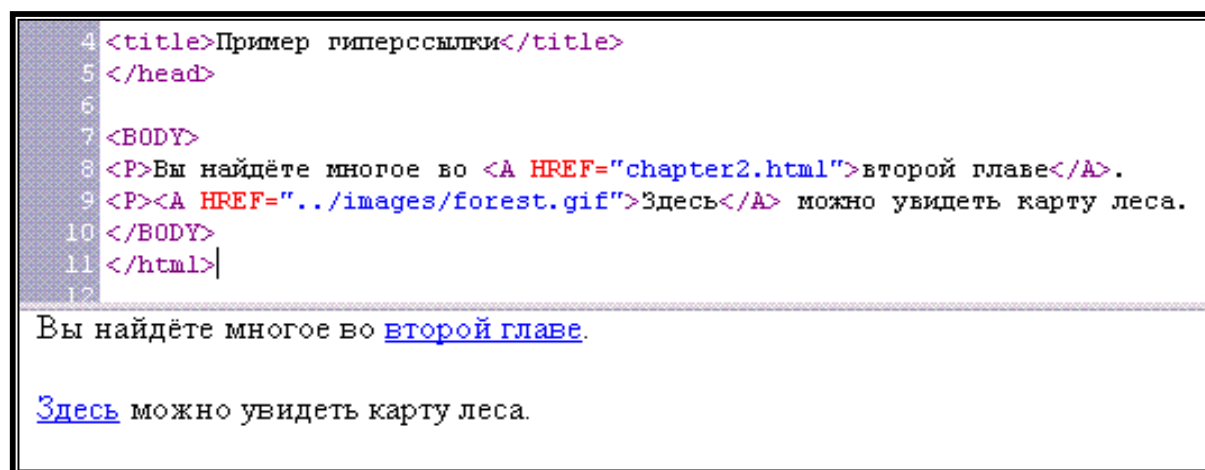


Рис. 2.9.Пример гиперссылки

Активировав эту ссылку (к примеру — щелчком мыши), пользователь может посетить этот ресурс. Заметим, что URI ресурса к которому должен осуществляться переход, может указываться в относительном виде (как это показано в данном примере), так и в абсолютном (<http://MySite.ru/image/forest.gif>).

Относительный стиль адресации предполагает, что базовый URI (тот, который помечается символом «..») указан в заголовочной части документа в элементе BASE:


```

<HTML>
<HEAD>
  <TITLE>Пример гиперссылки</TITLE>
  <BASE href="http://www.MySite.ru/index.html">
</HEAD>
<BODY>
  <P>Вы видели наши <A href=" ../Prod/birds.gif">товары</A>?
</BODY>
</HTML>

```

С учетом базового относительный URI "../cages/birds.gif" будет восприниматься браузером как «http://www.MySite.ru/cages/birds.gif»:

Что же касается варианта «chapter2.html» - то в данном случае в качестве базового URI воспринимается папка, в которой находится страница с начальным якорем.

Стоит также обратить внимание на прикладные протоколы по котрым HTML позволяет осуществлять доступ к ресурсам. Помимо традиционного http:// это:

- ✓ ftp://... - создает ссылку на ftp-сайт или расположенный на нем файл;
- ✓ mailto: - запускает почтовую программу-клиент с заполненным полем имени получателя. Если после адреса поставить знак вопроса, то можно указать дополнительные атрибуты, разделенные знаком "&";
- ✓ news:... - создает ссылку на конференцию сервера новостей;
- ✓ telnet://... - создает ссылку на telnet-сессию с удаленной машиной;
- ✓ file:/// - создает ссылку на место в файловой системе локального компьютера.

2.4.2. Закладки

Атрибут NAME позволяет присвоить соответствующему якорю имя по которому к нему в дальнейшем можно обращаться. В частности, механизм именования якорей является принципиальным, если речь идет о переходе не

просто к определенному ресурсу, а к определенному месту в этом ресурсе. В этом случае соответствующий якорь может выступать в качестве закладки, по имени которой к ней можно осуществить переход.

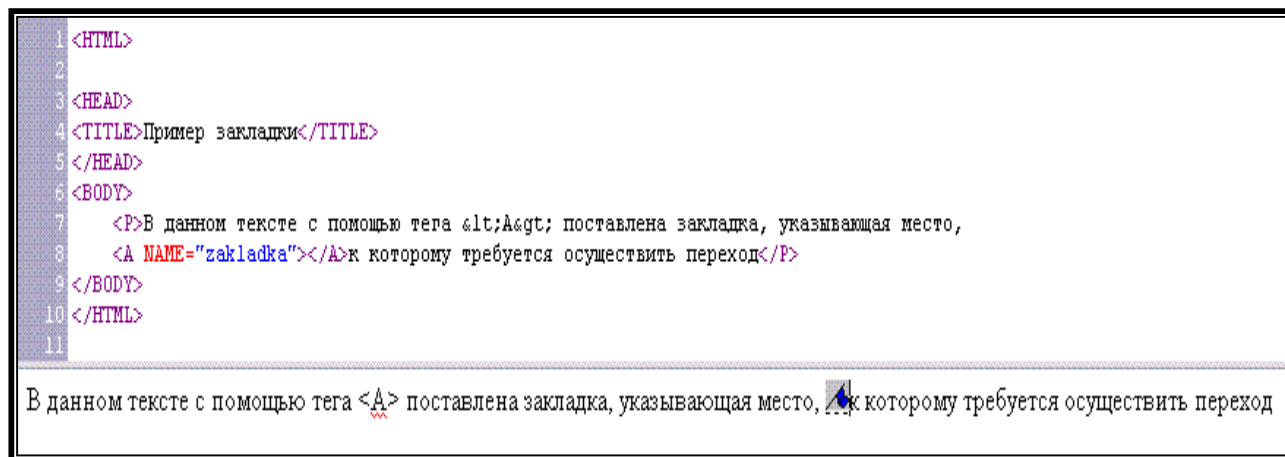


Рис. 2.10. Пример использования имени якоря в качестве закладки

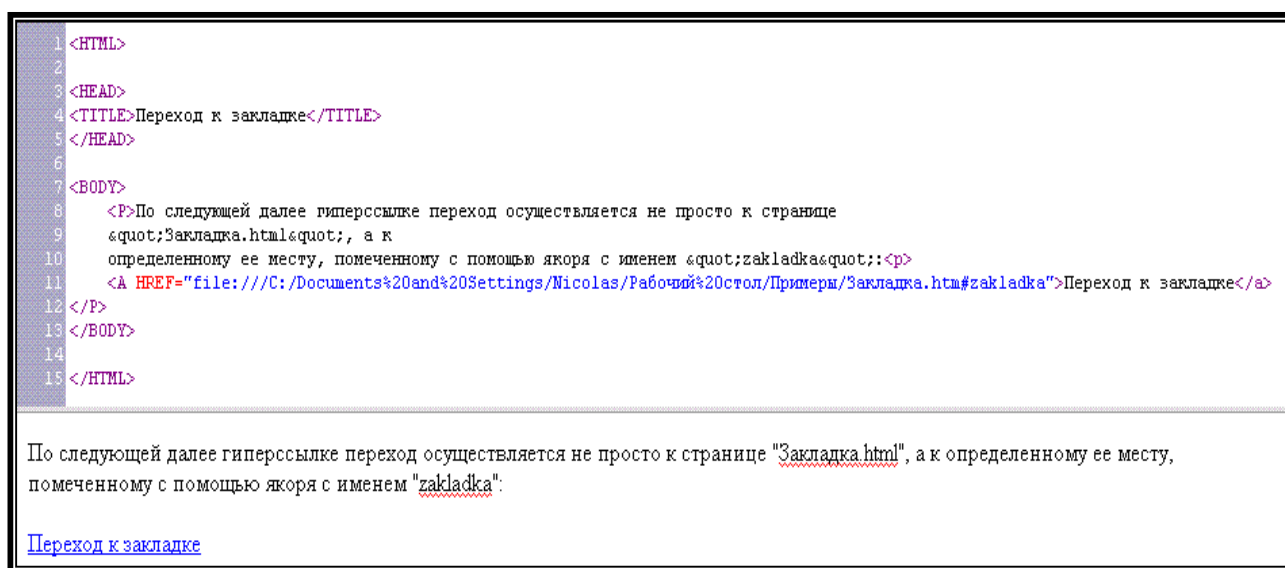


Рис. 2.11. Пример гиперссылки, позволяющей перейти к закладке

Требований к имени якоря лишь два — уникальность в рамках конкретного документа и соблюдение регистра при обращении по имени. Для обеспечения перехода к закладке ее имя добавляется после символа «#» в конце URI, используемого в качестве значения атрибута атрибута HREF.

Следует также отметить, что имя закладки в качестве элемента направления

перехода по гиперссылке может быть использовано и при обеспечении возможности гипертекстового перехода в рамках одного документа. В данном случае последовательность символов типа «#zakladka» и будет являться значением атрибута HREF.

Еще одной возможностью идентификации закладки, переход к которой должен осуществиться после активации соответствующей гиперссылки, является использование универсального атрибута ID, который можно использовать для начального тега любого элемента, описывающего структуру тела документа (<H1>, <H2>, <P>, <TABLE>, и др.).

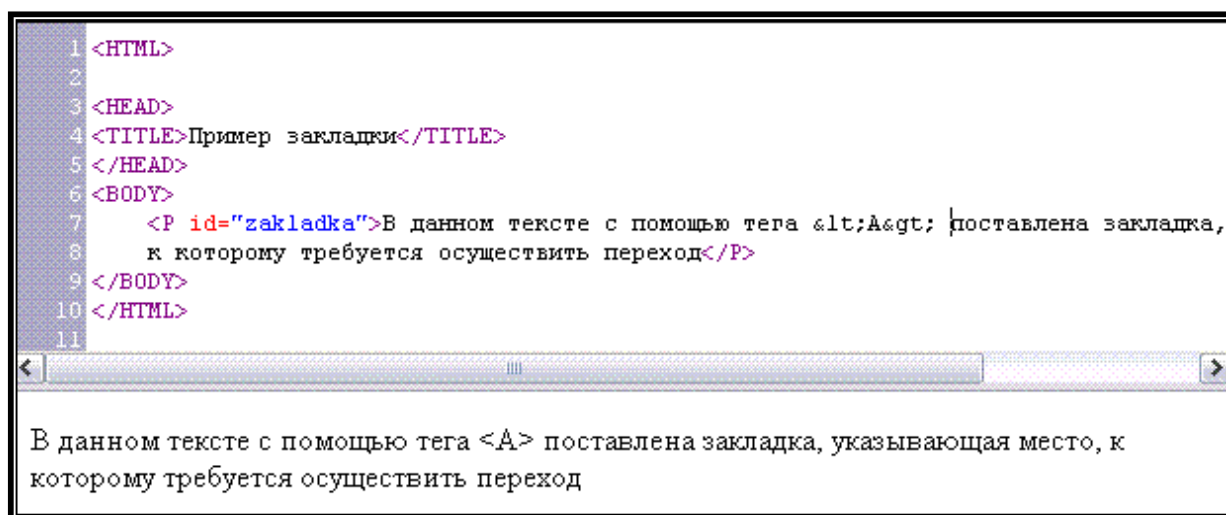


Рис. 2.12. Пример использования качества закладки ID структурного элемента HTML документа.

Синтаксис же для начального якоря при таком варианте идентификации закладки ничем не будет отличаться от рассмотренного ранее.

2.4.3. Открытие html страниц, вызываемой гиперссылкой, в новом окне

При помощи атрибута TARGET можно предусмотреть возможность загрузки страницы, открываемой при активации гиперссылки, в новом окне браузера. Этот атрибут предназначен для задания наименования окна, в котором должна открываться страница. Чтобы открыть страницу в новом окне надо

использовать значение `_blank`²¹.

```
1 <HTML>
2
3 <HEAD>
4 <TITLE>Переход к закладке</TITLE>
5 </HEAD>
6
7 <BODY>
8   <P>По следующей далее гиперссылке переход осуществляется не просто к странице
9   «Закладка.html», а к
10  определенному ее месту, помеченному с помощью якоря с именем «zakladka»:<p>
11  <a href="Закладка.htm#zakladka" target=_blank>Переход к закладке </a>
12 </P>
13 </BODY>
14
15 </HTML>
```

По следующей далее гиперссылке переход осуществляется не просто к странице "Закладка.html", а к определенному ее месту, помеченному с помощью якоря с именем "zakladka".

[Переход к закладке](#)

Рис. 2.13. Открытие документа в новом окне браузера.

2.4.4. Форматирование гиперссылок

Форматирование гиперссылок мало чем отличается от форматирования обычного текста — вопрос, которому была посвящена лекция 3, а именно:

- Пункты 1.3 — 1.5 — цветовое оформление гиперссылок;
- Пункт 3.1 — гарнитура шрифта и размер;
- Пункты 3.3, 3.4, 3.6 — параметры начертания символов (курсив, жирность, подчеркивание)

²¹ Использование данного атрибута является принципиальным, если речь идет о фреймовой разметке структуры страницы. Подробнее об этом будет говориться в лекции, посвященной фреймам.

2.5. Таблицы в HTML: создание и форматирование

2.5.1. Использование таблиц в HTML документе

Одним из наиболее мощных и широко применяемых в HTML средств являются таблицы. Понятие табличного представления данных не нуждается в дополнительном пояснении. В HTML таблицы используются не только традиционно, как метод представления данных, но и как средство форматирования Web-страниц.

Первая версия языка HTML не предусматривала специальных средств для отображения таблиц, так как была в основном предназначена для написания простого текста. С развитием сфер применения HTML-документов стала актуальной задача представления данных, для которых типично наличие ряда строк и столбцов. Создание документов, содержащих выровненные по колонкам данные, на первых порах осуществлялось использованием элемента `<PRE>`, внутри которого необходимое выравнивание обеспечивалось введением нужного количества пробелов. Напомним, что при использовании пары тэгов `<PRE>` и `</PRE>` все пробелы и символы табуляции являются значащими. Работа по выравниванию такого текста выполнялась вручную, что существенно замедляло создание документов.

Поддержка табличного представления данных стала стандартом де-факто, поскольку изначально была реализована во всех ведущих браузерах и лишь по прошествии значительного времени была закреплена в спецификации HTML 3.2.

2.5.2. Создание простейших HTML-таблиц

Каждая таблица должна начинаться тегом `<TABLE>` и завершаться тегом `</TABLE>`. Внутри этого контейнера располагается описание содержимого таблицы. Любая таблица состоит из одной или нескольких строк, в каждой из которых задаются данные для отдельных ячеек.

Каждая строка начинается тегом `<TR>` (Table Row) и завершается тегом

</TR>. Отдельная ячейка в строке обрамляется парой тегов <TD> и </TD> (Table Data) или <TH> и </TH> (Table Header). Тэг <TH> используется обычно для ячеек-заголовков таблицы, а <TD> — для ячеек-данных. Различие в использовании заключается лишь в типе шрифта, используемого по умолчанию для отображения содержимого ячеек, а также расположению данных внутри ячейки. Содержимое ячеек типа <TH> отображается полужирным (Bold) шрифтом и располагается по центру (ALIGN=CENTER, VALIGN=MIDDLE). Ячейки, определенные тегом <TD> по умолчанию отображают данные, выровненные влево (ALIGN=LEFT) и посередине (VALIGN=MIDDLE) в вертикальном направлении.

Теги <TD> и <TH> не могут появляться вне описания строки таблицы <TR>. Завершающие коды </TR>, </TD> и </TH> могут быть опущены. В этом случае концом описания строки или ячейки является начало следующей строки или ячейки, или конец таблицы.

Количество строк в таблице определяется числом открывающих тегов <TR>, а количество столбцов — максимальным количеством <TD> или <TH> среди всех строк. Часть ячеек могут не содержать никаких данных, такие ячейки описываются парой следующих подряд тегов — <TD>, </TD>. Если одна или несколько ячеек, располагающихся в конце какой-либо строки, не содержат данных, то их описание может быть опущено, а браузер автоматически добавит требуемое количество пустых ячеек. Отсюда следует, что построение таблиц, в которых в разных строчках располагается различное количество столбцов одного и того же размера, не разрешается.

Таблица может иметь заголовок, который заключается в пару тегов <CAPTION> и </CAPTION>. Описание заголовка таблицы должно располагаться внутри тегов <TABLE> и </TABLE> в любом месте, однако вне области описания любого из тегов <TD>, <TH> или <TR>. Согласно спецификации языка HTML расположение описания заголовка

регламентировано более строго: оно должно располагаться сразу же после тэга `<TABLE>` и до первого `<TR>`.

По умолчанию текст заголовка таблицы располагается над ней (`ALIGN=TOP`) и центрируется в горизонтальном направлении.

Перечисленные теги могут иметь параметры, число и значения которых различны. Однако в простейшем случае тэги используются без параметров, которые принимают значения по умолчанию.

Этих сведений вполне достаточно для построения элементарных таблиц. Приведем пример простейшей таблицы, состоящей из трех строк (одна из которых — строка заголовка) и двух столбцов (рис. 2.14):

```
2 <HEAD>
3 <TITLE>Простейшая таблица из трех строк и двух столбцов</TITLE>
4 </HEAD>
5 <BODY>
6     <TABLE BORDER>
7         <CAPTION>Пример таблицы</CAPTION>
8         <TR>
9             <TH>Столбец 1</TH>
10            <TH>Столбец 2</TH>
11        </TR>
12        <TR>
13            <TD>Строка 1 ячейка 1</TD>
14            <TD>Строка 1 ячейка 2</TD>
15        </TR>
16        <TR>
17            <TD>Строка 2 ячейка 1</TD>
18            <TD>Строка 2 Ячейка 2</TD>
19        </TR>
20    </TABLE>
21 </BODY>
22 </HTML>
```

Пример таблицы

Столбец 1	Столбец 2
Строка 1 ячейка 1	Строка 1 ячейка 2
Строка 2 ячейка 1	Строка 2 Ячейка 2

Рис. 2.14. Пример таблицы

2.5.3. Форматирование таблиц

Рассмотрим назначение различных атрибутов тега `<TABLE>`, отвечающих за внешний вид таблицы.

2.5.3.1. Расположение заголовка таблицы

Тег заголовка таблицы (`<CAPTION>`) имеет единственный допустимый параметр `ALIGN`, принимающий значения `TOP` (заголовок над таблицей) или `BOTTOM` (заголовок под таблицей). Параметр `ALIGN` может быть опущен, что соответствует значению `ALIGN=TOP`. В горизонтальном направлении заголовок таблицы всегда располагается по ее центру. Таблица может не иметь заголовка. В качестве заголовка таблицы в большинстве случаев используется простой текст, что регламентируется спецификацией HTML, однако в действительности между тегами `<CAPTION>` и `</CAPTION>` допустимо записывать любые HTML-элементы, употребляемые в разделе `<BODY>`.

2.5.3.2. Рамка таблицы. Атрибуты **BORDER** и **BORDERCOLOR**

Атрибут `BORDER` тега `<TABLE>` управляет изображением рамки вокруг каждой ячейки, которые, по сути, дают линии сетки таблицы, и вокруг всей таблицы. По умолчанию рамки не рисуются, и на экране пользователь увидит лишь ровно расположенный текст ячеек таблицы. Существует немало ситуаций, когда использование таблиц без рамок вполне оправданно, например, для многоколонных списков, реализованных при помощи таблиц, или задания точного взаимного расположения рисунков и текста. Однако в большинстве случаев для традиционного использования таблиц ее ячейки полезно отделить друг от друга линиями сетки, что облегчает восприятие и понимание информации, содержащейся в таблице.

Для добавления в таблицу рамок необходимо включить в код `<TABLE>` параметр `BORDER`, который может иметь численное значение. Например:

<TABLE BORDER>

или

<TABLE BORDER=10>.

Численное значение параметра определяет толщину рамки в пикселях, рисуемую вокруг всей таблицы, однако на толщину рамок вокруг каждой ячейки это значение не влияет. При отсутствии численного значения обычно оно принимается равным минимальному значению (1), хотя для различных браузеров стиль показа рамок может отличаться.

Атрибут BORDERCOLOR тега <TABLE> позволяет указать цвет рамки таблицы. Формальные обозначения для наиболее часто используемых цветов приведены в таблице 2.3.

Пример таблицы с рамкой красного цвета толщиной 10 пикселей приведен на рис. 2.15.

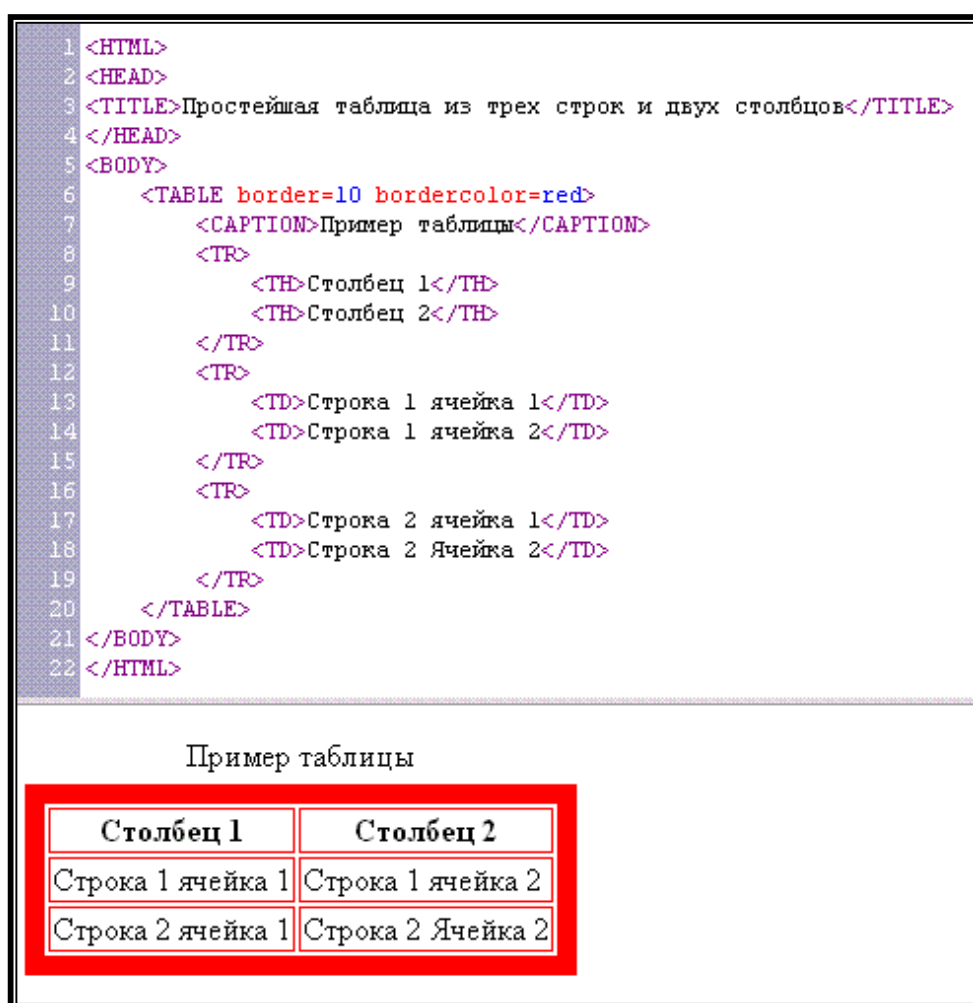


Рис. 2.15. Таблица с рамкой толщиной 10 пикселей красного цвета

2.5.3.3 Цвет фона таблицы и фоновый рисунок. Атрибуты BGCOLOR и BACKGROUND

Применение данных атрибутов тега <TABLE> полностью аналогично одноименным атрибутам тега <BODY>, рассмотренным в пунктах 1.1 и 1.2 лекции 4.

2.5.3.4 Расстояние между ячейками. Атрибут CELLSPACING

Форма записи атрибута: CELLSPACING=num, где num — численное значение в пикселах, которое не может быть опущено. Величина num определяет расстояние между смежными ячейками (точнее между рамками ячеек) как по горизонтали, так и по вертикали. По умолчанию значение принимается равным двум. Заметим, что традиционно в издательских системах смежные ячейки таблицы имеют общую границу. В HTML-таблицах по умолчанию между ними оставляется место, что хорошо видно на приведенном выше рисунке (рис. 2.15). При задании CELLSPACING=0 рамки смежных ячеек сольются и создадут впечатление единой сетки таблицы (рис. 2.16).

```

1 <HTML>
2 <HEAD>
3 <TITLE>Простейшая таблица из трех строк и двух столбцов</TITLE>
4 </HEAD>
5 <BODY>
6     <TABLE BORDER=10 cellspacing=0>
7         <CAPTION align=bottom>Пример таблицы</CAPTION>
8         <TR>
9             <TH>Столбец 1</TH>
10            <TH>Столбец 2</TH>
11        </TR>
12        <TR>
13            <TD>Строка 1 ячейка 1</TD>
14            <TD>Строка 1 ячейка 2</TD>
15        </TR>
16        <TR>
17            <TD>Строка 2 ячейка 1</TD>
18            <TD>Строка 2 Ячейка 2</TD>
19        </TR>
20    </TABLE>
21 </BODY>
22 </HTML>

```

Пример таблицы

Столбец 1	Столбец 2
Строка 1 ячейка 1	Строка 1 ячейка 2
Строка 2 ячейка 1	Строка 2 Ячейка 2

Рис.2.16. Таблица со значением CELLSPACING=0

2.5.3.5. Отступ между рамкой таблицы и содержимым ячейки. Атрибут CELLPADDING

Форма записи атрибута аналогична CELLSPACING. Величина num определяет размер свободного пространства (отступа) между рамкой ячейки и данными внутри ячейки. По умолчанию значение принимается равным единице. Установка атрибута CELLPADDING равным нулю может привести к тому, что некоторые части текста ячейки могут касаться ее рамки, что выглядит не очень эстетично.

На рис. 2.17 показан пример таблицы со значением CELLPADDING=10.

```

1 <HTML>
2 <HEAD>
3 <TITLE>Простейшая таблица из трех строк и двух столбцов</TITLE>
4 </HEAD>
5 <BODY>
6     <TABLE BORDER=10 cellspacing=0 cellpadding=10>
7         <CAPTION align=bottom>Пример таблицы</CAPTION>
8         <TR>
9             <TH>Столбец 1</TH>
10            <TH>Столбец 2</TH>
11        </TR>
12        <TR>
13            <TD>Строка 1 ячейка 1</TD>
14            <TD>Строка 1 ячейка 2</TD>
15        </TR>
16        <TR>
17            <TD>Строка 2 ячейка 1</TD>
18            <TD>Строка 2 Ячейка 2</TD>
19        </TR>
20    </TABLE>
21 </BODY>
22 </HTML>

```

Пример таблицы

Столбец 1	Столбец 2
Строка 1 ячейка 1	Строка 1 ячейка 2
Строка 2 ячейка 1	Строка 2 Ячейка 2

Рис. 2.17. Таблица со значением CELLPADDING=10

2.5.3.6. Ширина и высота таблицы. Атрибуты WIDTH и HEIGHT

При отображении таблиц их ширина и высота автоматически вычисляются браузером и зависят от многих факторов: значений параметров, заданных в описании всего документа, данной таблицы, отдельных ее строк и ячеек, содержимого ячеек, а также параметров, задаваемых при просмотре документа в том или ином браузере, например, типа и размеров шрифта, размеров окна просмотра и др. При отображении расчет размеров таблиц выполняется автоматически с учетом этих факторов, при этом делается попытка представить

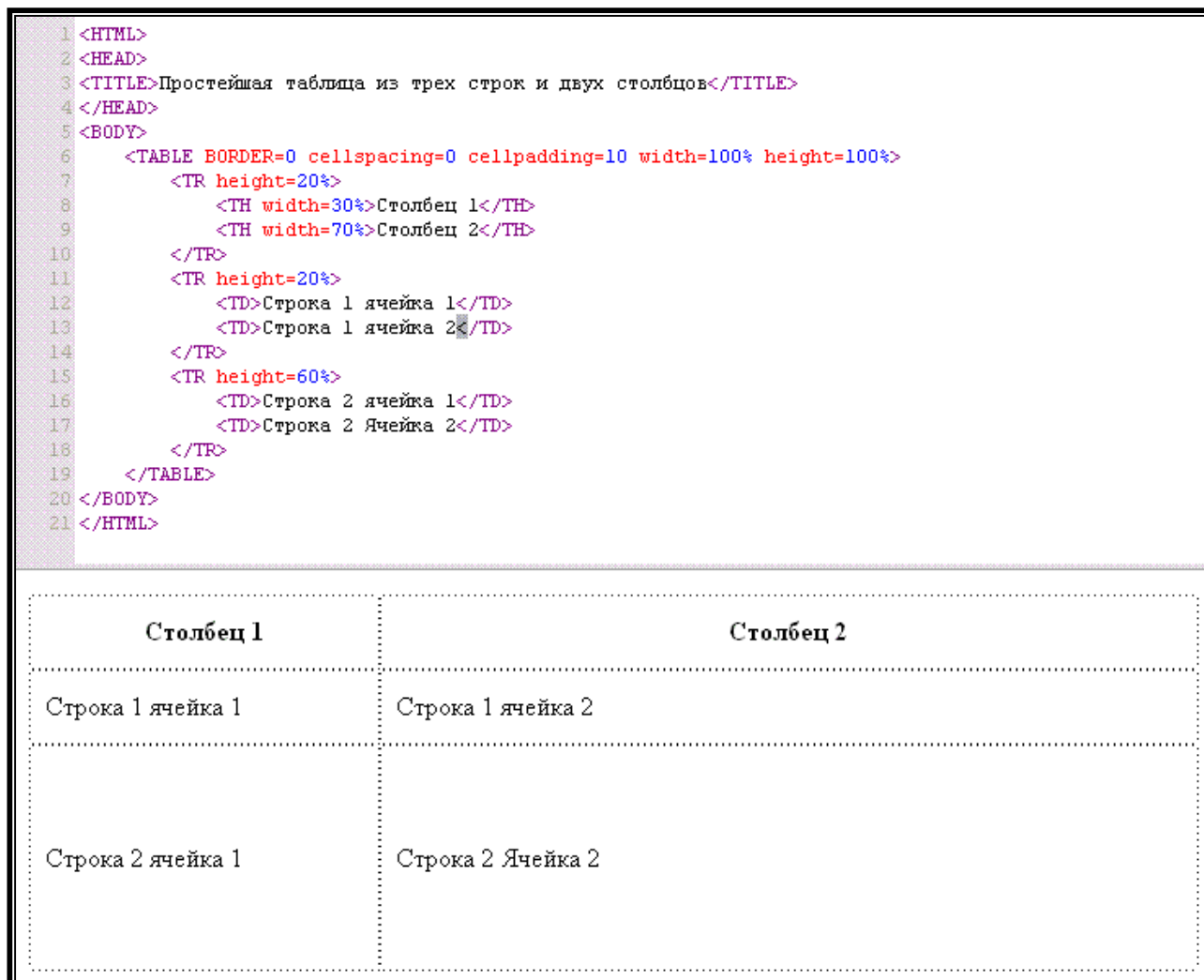
таблицу в наиболее удобном виде — расположить таблицу так, чтобы она помещалась и окне просмотра. Общая схема просмотра больших документов, как правило, сводится к линейной прокрутке содержимого документа по вертикали и чтении текста, перемежаемого различными таблицами, изображениями и т. п. Это относится как к HTML-документам, так и к обычным документам, создаваемым в любых текстовых редакторах. Большинство как текстовых редакторов (например, Microsoft Word), так и HTML-браузеров автоматически форматируют текст так (если возможно), чтобы длина строк не превосходила ширину окна просмотра. Это позволяет избежать необходимости горизонтальной прокрутки документа. Аналогичные действия предпринимаются браузерами с таблицами — по возможности форматировать их таким образом, чтобы ширина таблицы не превосходила ширины окна просмотра. Можно сделать вывод, что ширина таблиц является более важным, первостепенным параметром, расчет которого выполняется в первую очередь по сравнению с высотой.

В большинстве случаев динамическое определение размеров таблицы дает в результате эстетически соразмерное изображение с эффективным использованием реальной площади окна просмотра. Однако бывает необходимо принудительно указывать ширину или высоту таблицы. Для этой цели используются атрибуты WIDTH (ширина таблицы) и HEIGHT (высота таблицы) тега <TABLE>. Форма записи: WIDTH=num или WIDTH=num%, где num — численное значение ширины всей таблицы в пикселах или в процентах от всего размера окна браузера. Последний вариант указания высоты и ширины (в %) активно используется, к примеру, при применении таблиц для разметки страницы:

<TABLE WIDTH=100%>.

Аналогичные атрибуты могут задаваться и для отдельных строк и столбцов (ячеек). В примере, приведенном на рисунке 2.18 ширина таблицы составляет

100% от размера окна браузера. Из них 30% приходится на столбец 1 и 70% - на столбец 2. Высота таблицы составляет 100% от размера окна браузера. Из них 20% - первая строка, 20% - вторая строка и 60% - третья строка.



```

1 <HTML>
2 <HEAD>
3 <TITLE>Простейшая таблица из трех строк и двух столбцов</TITLE>
4 </HEAD>
5 <BODY>
6 <TABLE BORDER=0 cellspacing=0 cellpadding=10 width=100% height=100%>
7 <TR height=20%>
8 <TH width=30%>Столбец 1</TH>
9 <TH width=70%>Столбец 2</TH>
10 </TR>
11 <TR height=20%>
12 <TD>Строка 1 ячейка 1</TD>
13 <TD>Строка 1 ячейка 2</TD>
14 </TR>
15 <TR height=60%>
16 <TD>Строка 2 ячейка 1</TD>
17 <TD>Строка 2 Ячейка 2</TD>
18 </TR>
19 </TABLE>
20 </BODY>
21 </HTML>

```

Столбец 1	Столбец 2
Строка 1 ячейка 1	Строка 1 ячейка 2
Строка 2 ячейка 1	Строка 2 Ячейка 2

Рис. 2.18. Таблица с указанием параметров ширины и высоты в % от размера окна

2.5.3.7. Горизонтальное выравнивание таблицы. Атрибут ALIGN

Данный атрибут тега <TABLE> определяет горизонтальное расположение таблицы в области просмотра. Допустимые значения — LEFT (выравнивание влево), RIGHT (выравнивание вправо) и CENTER (выравнивание по центру). По умолчанию таблицы выровнены по левому краю.

На рисунке 2.19 представлена таблица, выравненная по центру страницы.

```
1 <HTML>
2 <HEAD>
3 <TITLE>Простейшая таблица из трех строк и двух столбцов</TITLE>
4 </HEAD>
5 <BODY>
6 <TABLE BORDER align=center>
7 <CAPTION>Пример таблицы</CAPTION>
8 <TR>
9 <TH>Столбец 1</TH>
10 <TH>Столбец 2</TH>
11 </TR>
12 <TR>
13 <TD>Строка 1 ячейка 1</TD>
14 <TD>Строка 1 ячейка 2</TD>
15 </TR>
16 <TR>
17 <TD>Строка 2 ячейка 1</TD>
18 <TD>Строка 2 Ячейка 2</TD>
19 </TR>
20 </TABLE>
21 </BODY>
22 </HTML>
```

Пример таблицы

Столбец 1	Столбец 2
Строка 1 ячейка 1	Строка 1 ячейка 2
Строка 2 ячейка 1	Строка 2 Ячейка 2

Рис. 2.19. Таблица, выровненная по центру страницы.

2.5.3.8. Объединение ячеек. Атрибуты COLSPAN и ROWSPAN

Для сложных таблиц характерна потребность в объединении нескольких смежных ячеек по горизонтали или по вертикали в одну. Данная возможность реализуется с помощью атрибутов COLSPAN (COLiimn SPANning) и ROWSPAN (ROW SPANning), задаваемых в кодах <TD> или <TH>. Форма записи: COLSPAN=num, где num — числовое значение, определяющее, на сколько столбцов следует расширить текущую ячейку по горизонтали. Применение атрибута ROWSPAN аналогично, только здесь указывается количество строк, которые должна захватить текущая ячейка по вертикали. По

умолчанию для этих параметров устанавливается значение, равное единице. Допустимо одновременное задание значений обоих параметров для одной ячейки. Правильная установка значений этих параметров может оказаться не очень простой задачей, тем более, что большинство HTML-редакторов позволяют визуально конструировать с последующей генерацией HTML-кодов лишь простейшие таблицы.

На рисунке 2.20 представлен типичный пример выполнения задачи. Методом объединения ячеек создан общий заголовок для двух столбцов и общий заголовок для трех строк.

```

1 <HTML>
2 <HEAD>
3   <TITLE>Пример сложной таблицы</TITLE>
4 </HEAD>
5 <BODY>
6   <TABLE border="1" width="100%">
7     <TR>
8       <TD>&nbsp;</TD>
9       <TD colspan="2">Объединение ячеек - общий заголовок для двух столбцов</TD>
10    </TR>
11    <TR>
12      <TD rowspan="3">Объединение ячеек - общий заголовок для трех строк</TD>
13      <TD>&nbsp;</TD>
14      <TD>&nbsp;</TD>
15    </TR>
16    <TR>
17      <TD>&nbsp;</TD>
18      <TD>&nbsp;</TD>
19    </TR>
20    <TR>
21      <TD>&nbsp;</TD>
22      <TD>&nbsp;</TD>
23    </TR>
24  </TABLE>
25 </BODY>
26 </HTML>
27

```

	Объединение ячеек - общий заголовок для двух столбцов	
Объединение ячеек - общий заголовок для трех строк		

Рис. 2.20. Пример таблицы с объединенными ячейками.

2.5.3.9. Форматирование данных внутри таблицы

Каждую отдельную ячейку внутри таблицы можно рассматривать как область для независимого форматирования. Все правила, которые действуют для управления отображением текста, могут быть использованы для форматирования текста внутри ячейки. Внутри ячейки допустимо использование практически всех элементов HTML, которые могут появляться внутри тела документа `<BODY>`, в том числе тэги, управляющие расположением текста — `<P>`, `
`, `<HR>`, коды заголовков — от `<H1>` до `<H6>`, теги форматирования символов — ``, `<I>`, ``, `<BIG>`, ``, ``, теги вставки графических изображений ``, гипертекстовых ссылок `<A>` и т. д. Сразу же подчеркнем, что область действия тегов, заданных внутри отдельной ячейки, ограничивается пределами этой ячейки независимо от наличия завершающего тега. Например, если внутри ячейки определен цвет текста — ``, то даже при отсутствии завершающего кода `` или расположения его через несколько ячеек или строк таблицы, текст следующей ячейки будет отражен цветом по умолчанию.

Помимо стандартны, при работе с таблицами можно использовать и специфические элементы форматирования. В частности, выравнивание текста внутри соответствующей ячейки можно установить с помощью атрибутов `ALIGN` и `VALIGN`, которые могут применяться в тегах `<TR>`, `<TD>` и `<TH>`.

Атрибут для горизонтального выравнивания `ALIGN` может принимать значения `LEFT`, `RIGHT` и `CENTER` (по умолчанию `LEFT` для `<TD>` и `CENTER` для `<TH>`). Атрибут для вертикального выравнивания `VALIGN` может принимать значения `TOP` (по верхнему краю), `BOTTOM` (по нижнему краю), `MIDDLE` (посередине), `BASELINE` (по базовой линии). По умолчанию — `MIDDLE`. Выравнивание по базовой линии обеспечивает привязку текста отдельной строки во всех ячейках к единой линии. Задание параметров

выравнивания на уровне кода <TR> определяет выравнивание для всех ячеек данной строки, при этом в каждой отдельной ячейки строки может быть определены свои параметры, переопределяющие действие параметров, заданных в <TR>.

Некоторые варианты выравнивания текста внутри таблицы представлены на рисунке 2.21. Следует обратить внимание, что если параметры выравнивания соответствуют установленным по умолчанию можно не указывать.

```
1 <HTML>
2 <HEAD>
3   <TITLE>Выравнивание элементов таблицы</TITLE>
4 </HEAD>
5 <BODY>
6   <TABLE border width=100% height="180">
7     <TR>
8       <TD align=right valign=top>Ячейка 1</TD>
9       <TD align=center>Ячейка 2</TD>
10      <TD valign=bottom>Ячейка 3</TD>
11    </TR>
12    <TR>
13      <TD align=right valign=bottom>Ячейка 4</TD>
14      <TD align=center>Ячейка 5</TD>
15      <TD valign=top>Ячейка 6</TD>
16    </TR>
17  </TABLE>
18 </BODY>
19 </HTML>
```

Ячейка 1	Ячейка 2	Ячейка 3
Ячейка 4	Ячейка 5	Ячейка 6

Рис.

Рис. 2.21. Выравнивание текста в таблице

2.6. Графика в HTML.

Без иллюстраций документ однообразен, вял и скучен. Расчетливо подобранная и правильно размещенная графика делает его визуально привлекательнее и, что самое важное, передает одну из основных идей документа. Вспомните старое правило - лучше один раз увидеть, чем сто раз услышать (данном случае прочитать).

2.6.1. Общие соображения.

Принимая решение о целесообразности включения в документ тех или иных иллюстраций необходимо учитывать следующие моменты:

- Графические файлы могут иметь значительные размеры, что требует времени для их загрузки. Насыщенность графикой может привести к недопустимо большим затратам времени, требуемым для получения документов, особенно, если используется соединение с помощью модема на небольших скоростях. С другой стороны, одновременная работа нескольких пользователей с большими документами, размещенными на вашем сервере, может также привести к его перегрузке.
- Многие пользователи работают в режиме отключения приема графических изображений для увеличения скорости передачи данных. В этом случае от полученных документов остается только текстовая часть, которая должна давать информацию о содержательной стороне документа.
- Поисковые системы не могут индексировать графику. Поэтому если на ваших страницах расположены только иллюстрации без текстовых пояснений, то читатели, использующие современные методы поиска, такие страницы не обнаружат.
- Следует помнить, что пользователи могут работать с различными расширениями экрана монитора и различной глубиной цвета. Страницы хорошо смотрящиеся при одном расширении, могут выглядеть

совершенно по иному при другом расширении.

2.6.2. Способы хранения изображений.

Существует значительное количество форматов хранения графических файлов, - порядка нескольких десятков.

При создании web-страниц чаще всего используется графика в двух форматах: GIF и JPG. Эти два формата поддерживаются всеми современными браузерами.

Формат BMP является стандартом MS Windows и поддерживается браузером Internet Explorer, однако его употребление не может быть рекомендовано, так как данный формат не поддерживает сжатие данных.

Разработанный недавно формат PNG был призван заменить формат GIF, однако, несмотря на его очевидные преимущества, должного распространения на настоящий момент не получил.

Иные графические форматы (кроме GIF и JPG) в HTML-документах на WWW-серверах практически не встречаются, хотя принципиально это возможно. Использование других форматов не рекомендуется по следующим причинам: во-первых, только для GIF и JPG осуществляется встроенная поддержка в большинстве браузеров, тогда как для иных файлов необходимо подключение внешних программ отображения, во-вторых, структура файлов GIF и JPG наиболее подходит для передачи данных по сети и является независимой от платформы.

В общем, изображения на Web-страницах могут использоваться двумя способами: в качестве фонового изображения, на котором располагаются элементы основного документа, и изображения, встраиваемые в документ.

2.6.3. Использование изображения в качестве фона HTML-документа (атрибут BACKGROUND тега <BODY>)

Техническая сторона вопроса относительно оформления фона страницы достаточно подробно был рассмотрен в рамках пункта 1.2 лекции 4. Остановимся лишь на некоторых замечаниях относительно эстетической стороны вопроса.

К фоновому рисунку HTML-документа, безусловно, предъявляется ряд требований. В зависимости от того, какой вид хочет придать своей странице автор, выбирается направление конструирования фона. До недавнего времени классическим решением было создание бледно-серого фона с таким же бледным, но рельефным рисунком. Здесь очень многое зависит от художника, но современные графические редакторы позволяют создавать похожие эффекты и автоматически. Такой фон не должен ощутимо снижать контрастность страницы и мешать чтению текста.

В последнее время в моде на фоновые рисунки произошли изменения. Все чаще можно встретить белый фон. Белый цвет вне конкуренции, и тут не надо что-либо объяснять. Другое направление — использование бледного фона, но другого цвета, например, бежевого, голубого или зеленого. Сам рисунок выглядит как произвольный узор из точек, напоминая поверхность кожи или камня. Здесь опять таки, полет фантазии художника ограничивается необходимостью обеспечения контрастности.

2.6.4. Встраивание изображений в HTML-документы (элемент)

Для встраивания изображений в HTML-документы следует использовать тег , имеющий единственный обязательный атрибут SRC, определяющий URL-адрес файла с изображением. Простейший пример встраивания изображения:

Что касается остальных атрибутов, то, хотя они и не являются обязательными, их использование может существенно повлиять на качество отображение графики на странице.

2.6.4.1. Выравнивание изображений (атрибут ALIGN).

При включении графического изображения в документ можно указать его расположение относительно текста или других элементов страницы. Способ выравнивания изображения задается значением параметра align тега . Возможные значения этого параметра:

Значение параметра align	Действие параметра
top	Верхняя граница изображения выравнивается по самому высокому элементу текущей строки
texttop	Верхняя граница изображения выравнивается по самому высокому текстовому элементу текущей строки
middle	Выравнивание середины изображения по базовой линии текущей строки
absmiddle	Выравнивание середины изображения посередине текущей строки
baseline или bottom	Выравнивание нижней границы изображения по базовой линии текущей строки
absbottom	Выравнивание нижней границы изображения по нижней границе текущей строки
left	Изображение прижимается к левому полю окна. Текст обтекает изображение с правой стороны
right	Изображение прижимается к правому полю окна. Текст обтекает изображение с левой стороны

Сразу же оговоримся, что все значения параметров выравнивания изображений можно условно разделить на две группы по их принципу действия. К одной группе относятся два значения параметра - `left` и `right` . При использовании любого из этих параметров мы получаем так называемое "плавающие" изображение. В этом случае изображение прижимается к соответствующему краю окна просмотра браузера, а последующий текст (или другие элементы) "обтекают" изображение с противоположной стороны. Здесь текст, размещаемый рядом с изображением может занимать несколько строк.

К другой группе значений параметров относятся все остальные. При их использовании изображение как бы встраивается в строчку текста, а параметры выравнивания задают расположение изображения относительно строки текста. Таким образом, в отличие от плавающих изображений, здесь изображения являются обычным элементом строки. Это легко понять, если представить, что изображение является просто одной буквой текста, правда, достаточно большой (типа буквицы).

Приведём пример кода, в котором используются изображения:

```
<html>  
<head>  
<title>Выравнивание изображений</title> </head>  
<body> Выравнивание<IMG src='../images/cat.jpg' align='top'>no  
верхнему краю <p> Выравнивание<IMG src='../images/cat.jpg'  
align='baseline'>по базовой линии </body>  
</html>
```

Возникает вопрос, в чем разница между базовой линией и нижней границей строки? Или различие между самым высоким элементом строки и самым высоким текстовым элементом строки? Результат действия этих параметров может отличаться в зависимости от содержимого рассматриваемой строки.

Базовая линия - это нижняя часть линии текста, которая проводится без учета нижней части некоторых символов, например, букв типа j, q, y. В отличие от выравнивания по базовой линии, при задании значения `absbottom` выравнивание выполняется по нижней части самого низкого элемента в строке, т.е. по одному из символов строки, имеющему элементы, лежащие ниже базовой линии. Аналогично обстоит дело с различием между параметрами `top` и `texttop`.

2.6.4.2. Задание размеров выводимого изображения (атрибуты WIDTH и HEIGHT).

Тег встраивания изображений имеет два необязательных атрибута, указывающих размеры изображения при отображении - `WIDTH` и `HEIGHT`. Значения параметров могут указываться как в пикселях, так и в процентах от размеров окна просмотра.

Значения параметров ширины и высоты изображения могут не совпадать с истинными размерами изображения. В этом случае браузер автоматически при загрузке изображения выполняет его перемасштабирование. Заметим, что неаккуратное задание параметров может привести к изменению пропорций рисунка и, как следствие, к его искажению.

Любой из этих параметров может быть опущен. Если задан только один из параметров, то при загрузке рисунка второй параметр будет вычисляться автоматически из условий сохранения пропорций. Изменение размеров изображений при помощи задания параметров ширины и высоты может использоваться для просмотра иллюстраций в уменьшенном виде, однако такой подход не сокращает время загрузки изображения.

Если не требуется изменить размеры изображения, настоятельно рекомендуется указывать их реальные размеры в пикселях. Указание действительных размеров:

- ✓ позволяет читателю, работающему в режиме отключения загрузки

изображений, иметь представление о размерах иллюстрации по пустому прямоугольнику, выдаваемому на экран вместо изображения (если размеры не будут указаны, то браузер, не зная их, выведет маленькую пиктограмму и форматирование будет нарушено).

- ✓ позволяет ускорить верстку документа на экране. Обычно браузеры должны загрузить все встроенные изображения прежде, чем отформатировать текст на экране. Указание размеров изображений позволяет выполнить форматирование документа до полной загрузки файлов и с изображениями.

2.6.4.3. Отделение изображений от текста (атрибуты HSPACE и VSPACE).

Для тега можно задать атрибуты HSPACE и VSPACE, значения которых определяют отступы от изображения, оставляемые пустыми, соответственно по горизонтали и вертикали. Это гарантирует, что между текстом и изображением останется пространство, необходимое для нормального восприятия.

2.6.4.4. Рамки вокруг изображений (атрибут BORDER).

Изображение, встраиваемое на страницу, можно поместить в рамку различной ширины. Для этого служит атрибут BORDER тега . В качестве значения атрибута используется число, означающее толщину рамки в пикселях. По умолчанию рамка вокруг изображения не рисуется. Исключением из этого правила является случай, когда изображение является ссылкой.

2.6.4.5. Альтернативный текст (атрибут ALT).

Одним из атрибутов тега является ALT, определяющий альтернативный текст. Его указание дает возможность пользователям работающих в режиме отключенной загрузки изображений, получить некоторую текстовую информацию о встроенных изображениях.

При отключенном изображении вместо них на экране появится

альтернативный текст, определенный значением атрибут ALT . Значение этого параметра имеет смысл и для случаев, когда загрузка изображений будет выполняться. Поскольку загрузка изображений выполняется на втором проходе после отображения текстовой информации, то изначально на экране на месте изображения появится альтернативный текст, который по мере загрузки будет сменяться изображением.

Современные браузеры также будут отображать альтернативный текст в качестве подсказки (tooltip) при помещении курсора мыши в область изображения.

2.6.5. Гиперкарты – важный элемент гипертекста

2.6.5.1. Использование гиперкарт

В последнее время многие Web-страницы для организации ссылок используют так называемые гиперкарты. Реализация этой возможности предусмотрена языком HTML и **позволяет привязывать гипертекстовые ссылки к различным областям изображения.**

Такой подход нагляднее, чем применение обыкновенных текстовых связей, поскольку пользователь может не читать словесное описание связи, а сразу понять ее смысл по графическому образу.

На рис. 6.1 показана Web-страница одной из крупнейших компьютерных фирм России. Основное меню на этой странице представляет собой гиперкарту с соответствующими ссылками.



Рис. 2.22. Главное меню сайта компьютерной фирмы ФЦЕНТР.

2.6.5.2. Терминология, связанная с гиперкартами

Imagemap, Image Map, Area Map, Clickable Map, Sensitive Map — все эти англоязычные термины используются в справочной литературе для обозначения одной и той же возможности — использование встроенного в HTML-документ изображения, для которого определены "горячие" (или активные) точки или области, имеющие ссылки на различные URL-адреса. Будем описывать эту возможность термином «гиперкарта», подразумевая под этим совокупность нескольких компонентов, обеспечивающих реализацию данной концепции. Основными компонентами являются: само изображение, которое будем называть опорным для данной гиперкарты; описание конфигурации активных областей; а также соответствующее программное обеспечение.

2.6.5.3. Преимущества и недостатки гиперкарт

В использовании гиперкарт есть как положительные, так и отрицательные моменты. Большинство из них носит эстетический характер, но некоторые имеют и технические аспекты.

Гиперкарты наиболее удобно использовать в следующих ситуациях:

- Для представления пространственных связей, например географических координат, которые было бы трудно задать отдельными кнопками или

текстом. В качестве примера может быть приведена карта Северной Америки, на которой выбор каждого из штатов ведет к переходу на соответствующую страницу.

- В качестве меню верхнего уровня, появляющегося на каждой странице. Наличие такого меню предоставляет возможность перехода в интересующий раздел сервера с любой страницы и в любой момент. Создание общего графического меню позволит сократить время разработки HTML-документов, поскольку будет использоваться один и тот же файл описания ссылок. Вместо того, чтобы на каждой странице устанавливать связи с различными частями начальной страницы, достаточно сослаться на общее меню. Такое меню также облегчит навигацию для пользователя.

Несмотря на то, что гиперкарты стали необычайно популярны, очевидно, что они не являются неотъемлемым атрибутом Web-страниц и используются далеко не на всех страницах. Есть ситуации, когда их использование не допустимо.

К недостаткам гиперкарт можно отнести следующие:

- Если не предусмотрено альтернативное текстовое меню, то не остается никаких средств навигации для пользователей, которые не могут загрузить графику или отключили ее загрузку.
- Картам-изображениям свойственны общие недостатки, присущие использованию изображений на Web-страницах, а именно, значительное увеличение времени загрузки по сравнению с чисто текстовыми документами.
- Неудачно спроектированные изображения могут внести путаницу. Иногда бывает трудно определить области, являющиеся активными на изображении.

- При использовании гиперкарт браузер не имеет возможности отмечать другим цветом уже пройденные ссылки так, как это делается для текстовых ссылок.

2.6.5.4. Графическое представление гиперкарты

Карта-изображение фактически представляет собой обычное встроенное графическое изображение на Web-странице. Эти изображения могут иметь любой допустимый формат (GIF или JPG). Для того чтобы изображение могло использоваться в качестве опорного для карты-изображения, формально не накладывается никаких дополнительных ограничений.

2.6.5.5. Описание конфигурации карты-изображения

Конфигурация гиперкарты записывается в виде обычного текста, который в зависимости от используемого формата может быть сохранен в отдельном файле или являться частью HTML-документа. Описание конфигурации содержит координаты для каждой из активных областей изображения, а также URL-адреса, связанные с каждой из этих областей. Активные области могут иметь форму прямоугольников, кругов и многоугольников. Допускается любая комбинация этих фигур. Также может задаваться одно значение URL-адреса, определенное для случая, когда пользователь выполняет щелчок в пределах изображения, но вне любой из заданных активных областей.

Элемент разметки гиперкарт

Для описания конфигурации областей карты-изображения используется элемент разметки <MAP>, единственным параметром которого является NAME. Значение параметра NAME определяет имя, которое должно соответствовать имени в USEMAP. Закрывающий тег - обязателен. Внутри контейнера должны располагаться описания активных областей карты, для чего используется элемент разметки <AREA>.

Элемент разметки <AREA>

Каждый элемент <AREA> задает одну активную область. Завершающий тег не требуется. Активные области могут перекрываться. В случае если некоторая точка относится одновременно к нескольким активным областям, то будет реализована та ссылка, описание которой располагается первым в списке областей.

Атрибутами тега <AREA> являются SHAPE, COORDS, HREF, NOHREF, TARGET, и ALT. Рассмотрим их назначение.

Атрибут SHAPE

Атрибут SHAPE определяет форму активной области. Допустимыми значениями являются rect, circle, poly. Эти значения задают области в виде прямоугольника, круга, многоугольника. Если атрибут SHAPE опущен, то по умолчанию предполагается значение rect, т. е. область в виде прямоугольника.

Атрибут COORDS

Параметр COORDS задает координаты отдельной активной области. Значением параметра является список координат точек, определяющих активную область, разделенных запятыми. Координаты записываются в виде целых неотрицательных чисел. Начало координат располагается в верхнем левом углу изображения, которому соответствует значение 0,0. Первое число определяет координату по горизонтали, второе — по вертикали. Список координат зависит от типа области.

Для области типа rect задаются координаты верхнего левого и правого нижнего углов прямоугольника.

Для области типа circle задаются три числа — координаты центра круга и радиус.

Для области типа poly задаются координаты вершин многоугольника в нужном порядке. Заметим, что последняя точка в списке координат не

обязательно должна совпадать с первой. Если они не совпадают, то при интерпретации данных для этой формы области браузер автоматически соединит последнюю точку с первой. Различные редакторы карт-изображений в этом отношении работают по-разному — одни добавляют первую точку в конец списка, а другие — нет. Количественные ограничения на число вершин довольно велики и покрывают практически все мыслимые потребности. По крайней мере многоугольник, имеющий 100 вершин, уверенно обрабатывается всеми ведущими браузерами. Есть ограничение, связанное с самим языком HTML, согласно которому список не может содержать более 1024 значений. Многоугольник вполне может быть невыпуклым.

Атрибуты HREF и NOHREF

Атрибуты HREF и NOHREF являются взаимоисключающими. Если не задан ни один из этих параметров, то считается, что для данной области не имеется ссылки. То же самое явно определяет атрибут NOHREF, не требующий значения. Атрибут HREF определяет адрес ссылки, который может записываться в абсолютной или относительной форме. Правила записи полностью совпадают с правилами записи ссылок в теге <A>.

Атрибут NOHREF полезно использовать для исключения части активной области. Пусть, например, необходимо создать активную область в виде кольца. Такой тип области не предусмотрен в списке возможных областей, однако он может быть реализован путем задания двух круговых областей. Для этого сначала следует задать область меньшего радиуса и указать в качестве параметра NOHREF. Далее нужно задать область большего радиуса с центром в той же точке и указать нужную ссылку. Тогда область внутри кольца, определенная двумя окружностями различного радиуса, будет иметь необходимую ссылку. Использование подхода, основанного на взаимном перекрытии областей, позволит строить области весьма разнообразной формы.

Атрибут TARGET

Атрибут TARGET употребляется при работе с фреймами. Его назначение — указать имя фрейма, в который будет размещен документ, загружаемый по данной ссылке. Более подробную информацию об использовании этого параметра можно получить из лекции, посвященной работе с фреймами.

Атрибут ALT

Атрибут ALT позволяет записать альтернативный текст для каждой из активных областей изображения. По существу этот текст будет играть лишь роль комментария для создателя документа. Если альтернативный текст, записанный для всего изображения (в теге), служит для выдачи его на экран при работе с отключенной загрузкой изображений, то альтернативный текст для активных областей никогда на экране не появится.

На рисунке 7.2 приведен пример гиперкарты, состоящей из 5 прямоугольных областей.

2.6.5.6. Алгоритм создания гиперкарт

Создание гиперкарт требует двух шагов: подготовки опорного изображения, на котором впоследствии будут заданы активные области, и разработки конфигурационных строк, описывающих геометрические параметры активных областей. Подготовка изображения, которое будет являться основой для карты-изображения, ничем не отличается от работы по подготовке обычных изображений, встраиваемых на Web-страницах. Для этого можно воспользоваться любым графическим редактором или использовать готовое изображение.



Рис. 2.23. Пример гиперкарты.

На втором шаге необходимо отметить активные области на изображении и сопоставить им соответствующие адреса ссылок. Подготовка файла конфигурации является наиболее трудным шагом в создании карт-изображений. В принципе возможен ручной способ задания границ активных областей на изображении. Например, при работе в графическом редакторе можно отмечать отдельные точки, записывать их координаты и затем создавать описания геометрических параметров выбранных областей. Однако такой подход крайне

неудобен и громоздок.

Для автоматизации процесса разметки областей на изображении существует ряд программ, большинство из которых очень похожи друг на друга. Они позволяют создавать и изменять конфигурационные элементы, работая непосредственно с изображением на экране. Большинство программ представляют собой отдельные утилиты, работающие автономно и, по существу, являющиеся дополнением к HTML-редакторам. Эти программы позволяют сохранять создаваемый конфигурационный файл либо в буфере обмена Windows, либо в файле на диске. В первом случае типичным вариантом является совместная работа программы редактирования карты-изображения и какого-либо HTML-редактора или обычного текстового редактора. Если же программа позволяет сохранять конфигурационный файл на диске, то она может использоваться полностью автономно. Все программы позволяют размечать на изображении области трех основных типов — rect, circle и poly. Пожалуй, единственным критерием выбора программы редактирования карт-изображений является удобство ее использования, так как по функциональным свойствам все программы очень близки. Если интерфейс программы покажется вам неудобным, можно отказаться от ее использования и выбрать другую.

Рассмотрим некоторые из существующих программ.

2.6.5.7. Средства создания гиперкарт

MapEdit

Одной из наиболее простых и известных программ редактирования конфигурационных файлов является утилита MapEdit, разработанная Томасом Бутеллом (Thomas Boutell). Эта программа существует уже на протяжении нескольких лет и реализована для различных платформ (Windows, MacOS). Как и для большинства программ, существовал ряд версий данной утилиты. На текущий момент последней доступной версией для Windows является версия 4.05 (декабрь 2007г.). Информацию о программе можно получить по адресу:

<http://www.boutell.com/mapedit/>

Программа MapEdit является условно-бесплатной (shareware) и имеет 30-дневный оценочный период, по истечении которого необходима ее регистрация. Программа невелика по размеру — дистрибутив занимает около 1 Мб, и при этом обладает практически всеми необходимыми возможностями.

CoffeeCup Mapper

Эта утилита разработана компанией [CoffeeCup Software](http://www.coffeecup.com/), известной такими программами, как HTML Editor Pro и рядом FTP-клиентов. Работать с этой программой очень просто: запускаете Map Wizard, загружаете графическое изображение для будущей гиперкарты, определяете адрес, по которому браузер перейдет в случае нажатия на область, не имеющей значений координат, и вариант открытия окна (в этом же окне, в новом и т.д.). После появления файла в окне программы нужно выбрать тип области, которому соответствует своя пиктограмма на панели инструментов (прямоугольник, круг, многоугольник) и выделить область необходимого размера на заготовке. После выделения появится окно, в котором следует указать адрес ссылки, альтернативный текст и вариант открытия документа (здесь же, в виде нового документа и т.д.). Возможно параллельно просмотреть готовую или текущую КИ в установленном по умолчанию браузере. HTML-код созданной карты отображается в нижнем окне программы. CoffeeCup Mapper позволяет создавать КИ только в клиентском варианте.

Существует триальная версия программы (ограничение по времени — 15 дней). Получить информацию о программе можно по адресу: <http://www.coffeecup.com/image-mapper/>

HTML Map Designer

"HTML Map Designer" - простая утилита, которая поможет упростить создание гиперкарт, содержащих много ссылок в области картинки на web-страницах. Используя данную утилиту создать собственные гипер-изображения

за несколько минут. Встроенный простой HTML редактор поможет редактировать и управлять полученным кодом HTML.

Особенности:

- Поддержка BMP, JPEG и GIF форматов.
- Вы можете корректировать интерфейс (цвета активных и пассивных областей).
- Простой доступ к любой отмеченной области со справкой списка областей.
- Отметки и события: ALT, TARGET...

Программа условно-бесплатная (ограничение по времени — 20 дней). Актуальная версия — 2.2. Получить информацию о программе можно по адресу: <http://www.imagecure.com/index.html>

В заключении хотелось бы отметить, что достаточно мощные с функциональной точки зрения средства разработки web-приложений, относящиеся к категории WYSIWYG редакторов (MS FrontPage, Macromedia Dreamweaver и др.) обладают встроенными возможностями создания гиперкарт которых более чем достаточно для решения большинства задач, описанных в рамках данной лекции.

Также встроенными возможностями по созданию гиперкарт обладают мощные растровые графические редакторы типа GIMP, относящиеся к программам Open Source-типа (то есть программ с открытым исходным кодом с и свободным доступом).

2.7. Формы в HTML

2.7.1. Основы создания форм.

Формы являются одним из важных элементов любого сайта и предназначены для обмена данными между пользователем и сервером. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.

Любая форма характеризуется следующими параметрами.

- ✓ Элементы формы, которые представляют собой стандартные поля для ввода информации.
- ✓ Кнопку отправки данных формы на сервер (кнопка SUBMIT).
- ✓ Адрес программы на веб-сервере, которая будет обрабатывать содержимое данных формы.

Перед отправкой данных браузер подготавливает информацию в виде пары «имя=значение», где имя определяется атрибутом NAME тега `<INPUT>` или другим, допустимым в форме, а значение введено пользователем или установлено в поле формы по умолчанию. После нажатия пользователем кнопки SUBMIT, происходит запуск обработчика формы, которая получает введенную в форме информацию, а дальше делает с ней то, что предполагал разработчик. В качестве обработчика формы обычно выступает CGI-программа, заданная атрибутом ACTION тега `<FORM>`. Аббревиатурой CGI (Common Gateway Interface, общий шлюзовый интерфейс) обозначается протокол, с помощью которого программы взаимодействуют с веб-сервером. С помощью CGI на сервере можно выполнять программы на любом языке программирования и результат их действия выводить в виде веб-страницы. Наиболее популярны следующие языки — PHP, Perl, C++.

Для указания браузеру где начинается и заканчивается форма, используется контейнер `<FORM>`. Между открывающим и закрывающим тегами `<FORM>` и

</FORM> можно помещать любые необходимые теги HTML. Это позволяет добавить элементы формы в ячейки таблицы для их форматирования, а также использовать изображения. Документ может содержать несколько форм, но они не должны быть вложены одна в другую (рисунок 2.24).

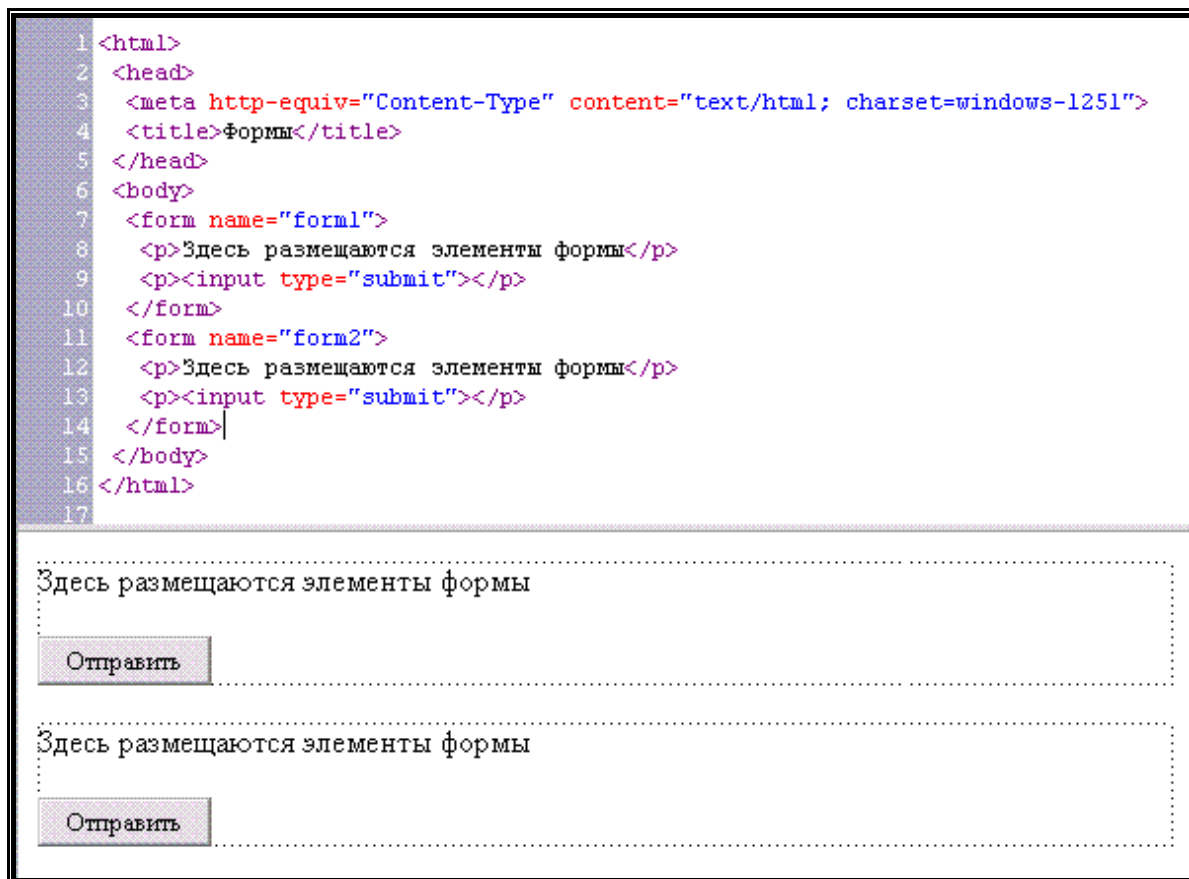


Рис. 2.24. Добавление формы в документ

В данном примере показано добавление двух разных форм. При нажатии на кнопку SUBMIT данные текущей формы отправляются на сервер, а остальные формы на веб-странице никак не будут обработаны.

Каждая форма характеризуется некоторыми атрибутами, которые указываются в теге <FORM> и позволяют указать: имя формы, ее обработчик и метод отправки данных на сервер, а также некоторые другие характеристики.

2.7.1.1. Атрибут ACTION

Указывает обработчик, к которому обращаются данные формы при их отправке на сервер (рис. 2.25). В качестве обработчика может выступать CGI-программа или HTML-документ, который включает в себя серверные сценарии. После выполнения обработчиком действий по работе с данными формы он возвращает новый HTML-документ.

Если атрибут ACTION отсутствует, текущая страница перезагружается, возвращая все элементы формы к их значениям по умолчанию.

```
1 <html>
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
4   <title>Формы</title>
5 </head>
6 <body>
7   <form action="http://informnn.com/files/file.php" id="Forma">
8     <h1 align="center">Введите Ваше имя:</h1>
9     <table border="0" width="30%" align="center">
10      <tr>
11        <td width="120">Имя:</td>
12        <td><input type="text" id="Imya" size="38"></td>
13      </tr>
14      <tr>
15        <td width="120">&nbsp;</td>
16        <td>&nbsp;</td>
17      </tr>
18      <tr>
19        <td width="120"><input type="submit"></td>
20        <td><input type="reset" align="right"></td>
21      </tr>
22    </table>
23  </form>
24 </body>
25 </html>
```

Введите Ваше имя:

Имя:	<input type="text"/>
<input type="submit" value="Отправить"/>	<input type="reset" value="Сброс"/>

Рис

Рис. 2.25. Указание обработчика формы

В качестве обработчика можно указать также адрес электронной почты, начиная его с ключевого слова `mailto`. При отправке формы будет запущена почтовая программа установленная по умолчанию.

2.7.1.2. Атрибут METHOD

Метод сообщает серверу о цели запроса. Различают два метода — GET и POST. Существуют и другие методы, но они пока мало используются.

Метод GET

Этот метод является одним из самых распространенных и предназначен для получения требуемой информации и передачи данных в адресной строке. Пары «имя=значение» присоединяются в этом случае к адресу после вопросительного знака и разделяются между собой амперсандом (символ `&`). Удобство использования метода GET заключается в том, что адрес со всеми параметрами можно использовать неоднократно, сохранив его, например, в «Избранное» браузера, а также менять значения параметров прямо в адресной строке.

Метод POST

Метод POST посылает на сервер данные в запросе браузера. Это позволяет отправлять большее количество данных, чем доступно методу GET, поскольку у него установлено ограничение в 4 Кб. Большие объемы данных используются в форумах, почтовых службах, заполнении базы данных и т.д.

2.7.1.3. Атрибут TARGET

Атрибут TARGET употребляется при работе с фреймами. Его назначение — указать имя фрейма, в который будет размещен результат обработки формы (в виде HTML-документа). Более подробную информацию об использовании этого параметра можно получить из лекции, посвященной работе с фреймами.

2.7.2. Элементы форм

Форма представляет собой лишь контейнер для размещения объектов, которые дублируют элементы интерфейса операционной системы: кнопки, поле со списком, переключатели, флажки и т.д.

2.7.2.1. Однострочное текстовое поле

Однострочное текстовое поле предназначено для ввода строки символов с помощью клавиатуры. Синтаксис создания такого поля следующий.

<INPUT TYPE="text">

При создании текстового поля с помощью соответствующих атрибутов тега **<INPUT>** можно указывать следующие параметры:

Табл. 2.4.

Основные атрибуты однострочного текстового поля

Атрибут	Описание
SIZE	Ширина текстового поля, которое определяется числом символов моноширинного шрифта. Иными словами, ширина задается количеством близстоящих букв одинаковой ширины по горизонтали.
MAXLENGTH	Устанавливает максимальное число символов, которое может быть введено пользователем в текстовом поле. Когда это количество достигается при наборе, дальнейший ввод становится невозможным. Если этот параметр опустить, то можно вводить строку длинее самого поля.
ID	Идентификатор поля, предназначенный для того, чтобы обработчик формы мог его идентифицировать.
NAME	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
VALUE	Начальный текст, отображаемый в поле.

Создание текстового поля с разными параметрами показано на рис. 2.26.

```
1 <html>
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
4   <title>Текстовое поле</title>
5 </head>
6 <body>
7   <form id=Forma>
8     <p><b>Как ваше имя?</b></p>
9     <p><input type="text" maxlength="25" size="40"></p>
10  </form>
11 </body>
12 </html>
```

Как ваше имя?

Рис .

Рис. 2.26. Текстовое поле

2.7.2.2. Поле для пароля

Поле для пароля — обычное текстовое поле, но отличается от него тем, что все символы отображаются звездочками. Предназначено для того, чтобы никто не подглядел вводимый пароль. Синтаксис создания следующий.

<INPUT TYPE="password">

Атрибуты поля полностью аналогичны перечисленным для текстового поля. Создание поля для пароля показано на рисунке 2.27.

```

1 <html>
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
4   <title>Ввод пароля</title>
5 </head>
6 <body>
7   <form>
8     <p><b>Логин:</b> <input type="text" maxlength="25" size="40" name="text" value="Imya"></p>
9     <p><b>Пароль:</b> <input type="password" maxlength="15" size="40" name="pass" value="Pass"></p>
10    <p><input type="submit"></p>
11  </form>
12 </body>
13 </html>

```

Рис. 2.27. Поле для пароля

2.7.2.3. Многострочный текст

Этот элемент формы предназначен для создания области, в которой можно вводить несколько строк текста. В таком текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер.

Синтаксис создания следующий.

<TEXTAREA>текст</TEXTAREA>

Между тегами `<textarea>` и `</textarea>` можно поместить любой текст, который будет отображаться внутри поля.

Допустимые атрибуты перечислены в табл. 2.5.

Табл. 2.5.

Атрибуты многострочного текста

Атрибут	Описание
COLS	Ширина поля в символах.
DISABLED	Блокирует доступ и изменение элемента.
ID	Идентификатор поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
NAME	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
READONLY	Устанавливает, что поле не может изменяться пользователем.
ROWS	Высота поля в строках текста.
WRAP	Параметры переноса строк.

Создание поля многострочного текста показано на рисунке 2.28.

```

1 <html>
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
4   <title>Текстовое поле</title>
5 </head>
6 <body>
7   <form>
8     <p><b>Введите ваш отзыв:</b></p>
9     <p><textarea rows="10" cols="45" id="comment"></textarea></p>
10    <p><input type="submit"></p>
11  </form>
12 </body>
13 </html>

```

Введите ваш отзыв:

Отправить

Рис. 2.28. Многострочный текст

2.7.2.4. Поле со списком

Поле со списком, называемое раскрывающимся списком или ниспадающим меню, один из гибких и удобных элементов формы. В зависимости от настроек, в списке можно выбирать одно или несколько значений. Преимущество списка в его компактности, он может занимать всего одну строку, а чтобы просмотреть весь список нужно на него нажать. Однако это является и недостатком, ведь пользователю сразу не виден весь выбор.

Поле со списком создается следующим образом.

```
<SELECT>  
<OPTION>Пункт 1</OPTION>  
<OPTION>Пункт 2</OPTION>  
<OPTION>Пункт 3</OPTION>  
</select>
```

Тег `<SELECT>` позволяет создать элемент интерфейса в виде раскрывающегося списка, а также список с одним или множественным выбором. Конечный вид зависит от использования параметра `SIZE` тега `<SELECT>`, который устанавливает высоту списка. Ширина списка определяется самым широким текстом, указанным в теге `<OPTION>`, а также может изменяться с помощью стилей. Каждый пункт создается с помощью тега `<OPTION>`, который должен быть вложен в контейнер `<SELECT>`.

Атрибуты тега `<SELECT>`

Рассмотрим атрибуты тега `<SELECT>`, с помощью которых можно изменять вид и представление списка:

- **MULTIPLE.** Наличие данного атрибута сообщает браузеру отображать содержимое элемента `<SELECT>` как список множественного выбора. Конечный вид списка зависит от использования атрибута `SIZE`. Если он отсутствует, то высота списка равна количеству пунктов, если значение `SIZE` меньше числа пунктов, то появляется вертикальная полоса прокрутки.

Для выбора нескольких значений списка применяются клавиши <Ctrl> и <Shift> совместно с курсором мыши.



Рис. 2.29. Список множественного выбора

- **ID** определяет уникальный идентификатор элемента <SELECT>. Как правило, это имя используется для доступа к данным через скрипты или для получения выбранного значения серверным обработчиком.
- **SIZE** устанавливает высоту списка. Если значение атрибута SIZE равно единице, то список превращается в раскрывающийся. При добавлении параметра multiple к тегу <SELECT> при size=1 список отображается как «крутилка». Во всех остальных случаях получается список с одним или множественным выбором. Значение по умолчанию зависит от параметра multiple. Если он присутствует, то размер списка равен количеству элементов. Когда параметра multiple нет, то значение параметра size равно 1.

Атрибуты тега <OPTION>

Тег <OPTION> также имеет атрибуты, влияющие на вид списка, они представлены далее.

- **SELECTED** делает текущий пункт списка выделенным. Если у тега `<SELECT>` добавлен атрибут `MULTIPLE`, то можно выделять более одного пункта.
- **VALUE** определяет значение пункта списка, которое будет отправлено на сервер. На сервер отправляется пара «имя/значение», где имя задается атрибутом `ID` тега `<SELECT>`, а значение — атрибутом `VALUE` выделенных пунктов списка. Значение может, как совпадать с текстом пункта, так быть и самостоятельным.

Создание списка показано на рис. 2.30.

The image shows a screenshot of an HTML editor with two parts: a code editor at the top and a preview window at the bottom.

Code Editor:

```

1 <html>
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
4   <title>Список</title>
5 </head>
6 <body>
7   <form>
8     <p><b>Выбери персонажа</b></p>
9     <p><select id="hero">
10      <option value="s1">Чебурашка</option>
11      <option value="s2" selected>Крокодил Гена</option>
12      <option value="s3">Шапокляк</option>
13      <option value="s3">Крыса Лариса</option>
14    </select>
15    <input type="submit" value="Отправить"></p>
16  </form>
17 </body>
18 </html>

```

Preview Window:

The preview shows a web form titled "Выбери персонажа". It contains a dropdown menu with four options: "Крокодил Гена" (selected), "Чебурашка", "Шапокляк", and "Крыса Лариса". Below the dropdown is a button labeled "Отправить".

Рис. 2.30. Использование списка

В примере создается список из четырех пунктов с именем `hero`, причем второй пункт из них предварительно выделен через параметр `selected` тега `<OPTION>`. Результат примера показан ниже.

Группировка элементов списка

При достаточно обширном списке имеет смысл сгруппировать его элементы по блокам, чтобы обеспечить наглядность списка и удобство работы с ним. Для этой цели применяется тег `<OPTGROUP>`. Он представляет собой контейнер, внутри которого располагаются теги `<OPTION>` объединенные в одну группу. Особенностью тега `<OPTGROUP>` является то, что он не выделяется как обычный элемент списка, акцентируется с помощью жирного начертания, а все элементы, входящие в этот контейнер, смещаются вправо от своего исходного положения. Для добавления заголовка группы используется Атрибут LABEL, как показано на рис. 2.31.

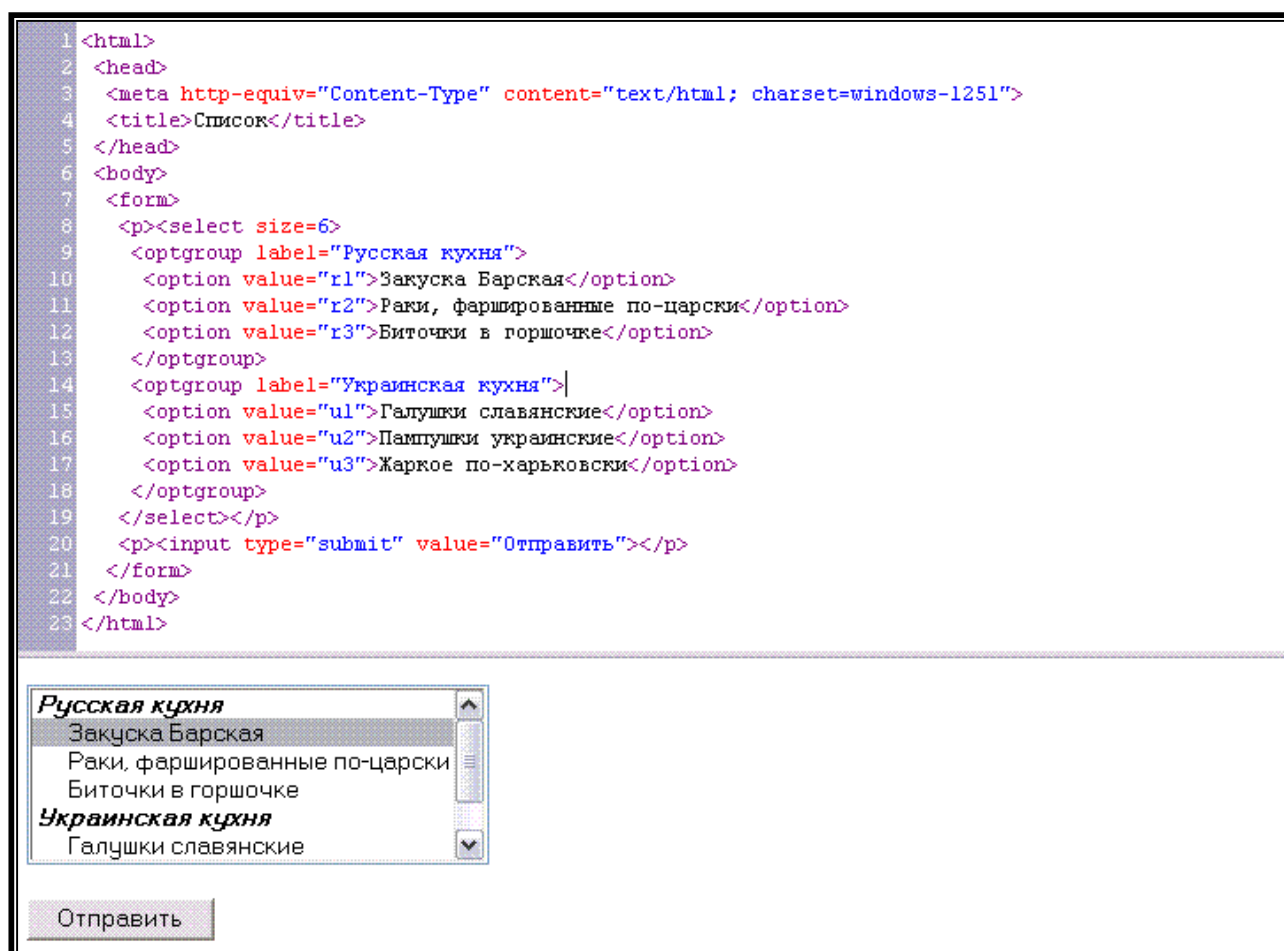


Рис. 2.31. Группирование элементов списка

2.7.2.5. Переключатели

Переключатели используют, когда необходимо выбрать один единственный вариант из нескольких предложенных. Создаются следующим образом.

`<INPUT TYPE="radio" NAME="Primer">`

Возможные для переключателей атрибуты перечислены в табл. 2.6

Табл. 2.6.

Параметры переключателей

Атрибут	Описание
CHECKED	Предварительное выделение переключателя. По определению, набор переключателей может иметь только один выделенный пункт, поэтому добавление checked сразу к нескольким полям не даст особого результата. В любом случае, будет отмечен элемент, находящийся в коде HTML последним.
NAME	Имя группы переключателей для идентификации поля. Поскольку переключатели являются групповыми элементами, то имя у всех элементов группы должно быть одинаковым. Атрибут является обязательным.
ID	Идентификатор группы переключателей по которому к ней можно обращаться из сценариев на обработку событий. Поскольку переключатели являются групповыми элементами, то имя у всех элементов группы должно быть одинаковым.
VALUE	Задаёт, какое значение будет отправлено на сервер. Здесь уже каждый элемент должен иметь свое уникальное значение, чтобы можно было идентифицировать, какой пункт был выбран пользователем.

Создание группы переключателей показано на рис. 2.32.

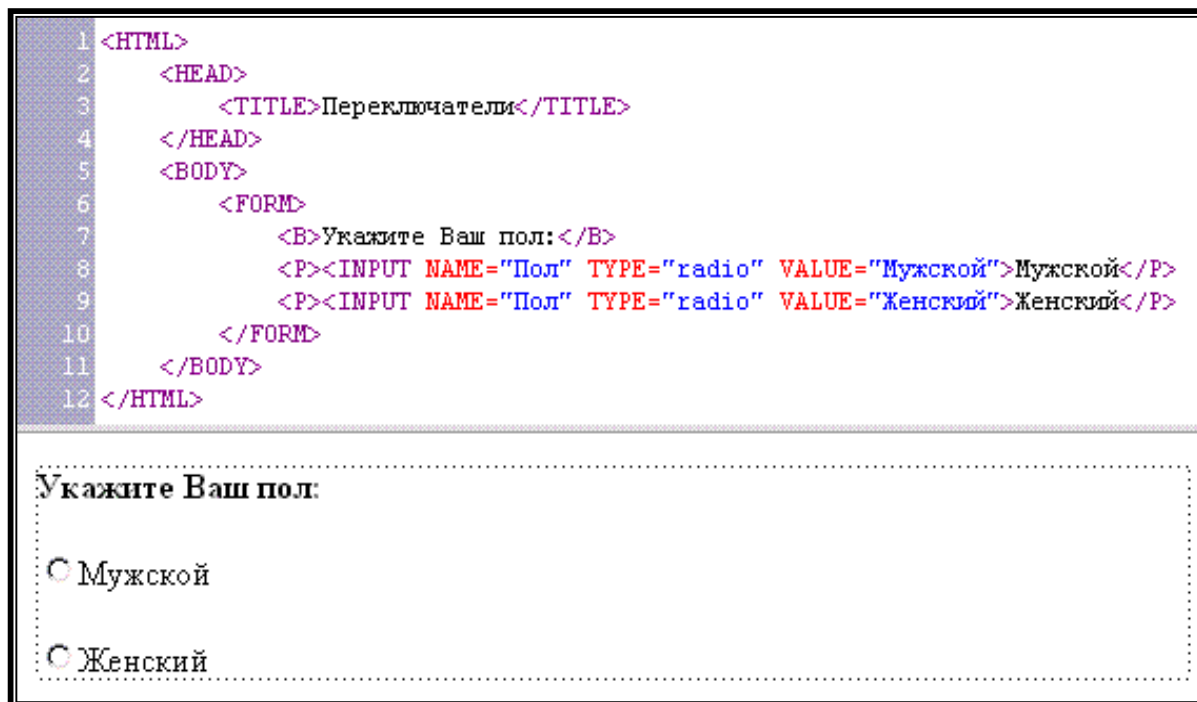


Рис. 2.32. Создание переключателей

Заметим, что, атрибут NAME для всех переключателей обязателен и имеет одинаковое значение. Только в этом случае браузер будет понимать, что переключатели связаны между собой и помечать только один пункт из предложенных. Значение параметра VALUE же должно различаться, чтобы обработчик формы мог понять, какой вариант выбран пользователем.

2.7.2.6. Флажки

Флажки используют, когда существует возможность выбора нескольких вариантов значения из предложенного списка.

Флажок создается следующим образом.

<INPUT TYPE="checkbox">

Атрибуты для флажков полностью аналогичны перечисленным для переключателей. Использование флажков показано на рис. 2.33.

```

1 <HTML>
2   <HEAD>
3     <TITLE>Флажки</TITLE>
4   </HEAD>
5   <BODY>
6     <FORM ACTION="handler.php">
7       <P><B>С какими операционными системами вы знакомы?</B></P>
8       <P><INPUT TYPE="checkbox" NAME="option1" VALUE="a1" CHECKED> Windows 95/98<BR>
9       <INPUT TYPE="checkbox" NAME="option2" VALUE="a2"> Windows 2000<BR>
10      <INPUT TYPE="checkbox" NAME="option3" VALUE="a3"> Windows XP<BR>
11      <INPUT TYPE="checkbox" NAME="option4" VALUE="a4"> Windows 2000<BR>
12      <INPUT TYPE="checkbox" NAME="option5" VALUE="a5"> OS/2<BR>
13      <INPUT TYPE="checkbox" NAME="option6" VALUE="a6"> Linux<BR>
14      <INPUT TYPE="checkbox" NAME="option7" VALUE="a7"> MS-DOS</P>
15      <P><INPUT TYPE="submit" VALUE="Выбрать"></P>
16    </FORM>
17  </BODY>
18 </HTML>

```

С какими операционными системами вы знакомы?

☒ Windows 95/98

☐ Windows 2000

☐ Windows XP

☐ Windows 2000

☐ OS/2

☐ Linux

☐ MS-DOS

Выбрать

Рис. 2.33. Создание флажков

2.7.2.7. Скрытое поле

Скрытое поле не показывается на странице и прячет свое содержимое от пользователя. Посетитель не может ничего в него внести или напечатать. Цель создания скрытых полей — в передаче технической информации на сервер. В большинстве случаев это необходимо для передачи данных формы от страницы к странице.

Синтаксис создания скрытого поля.

<INPUT TYPE="hidden">

Возможные для скрытого поля атрибуты перечислены в табл. 7.4.

Атрибуты скрытого поля

Атрибут	Описание
ID	Идентификатор поля, предназначенный для того, чтобы обработчик формы мог его идентифицировать.
NAME	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
VALUE	Начальный текст, отображаемый в поле.

Пример использования скрытых полей приведен на рис. 2.34.

The image shows a screenshot of a web browser displaying an HTML form. The form is enclosed in a dashed border and contains the following elements:

- A title bar at the top: "Напишите любимое слово (никакие данные не будут передаваться на сервер!):"
- A text input field below the title bar.
- A submit button labeled "Отправить" at the bottom left of the form.

Below the form, the corresponding HTML code is displayed, showing the use of hidden fields:

```

1 <HTML>
2   <HEAD>
3     <TITLE>Скрытое поле</TITLE>
4   </HEAD>
5   <BODY>
6     <FORM>
7       <P><B>Напишите любимое слово (никакие данные не будут передаваться на сервер!):</B></P>
8       <P><INPUT TYPE="text" SIZE="25" NAME="word">
9       <INPUT TYPE="hidden" NAME="UserName" VALUE="Vasya">
10      <INPUT TYPE="hidden" NAME="password" VALUE="pupkin"></P>
11      <P><INPUT TYPE="submit" VALUE="Отправить"></P>
12    </FORM>
13  </BODY>
14 </HTML>
15

```

Рис. 2.34. Использование скрытого поля

В данном примере показано создание двух скрытых полей, одно из них носит имя «UserName» и получает значение Vasya, а второе именуется «password» со значением pupkin. В результате отправки формы программа может легко прочитать эти данные и использовать их по усмотрению разработчика.

2.7.2.8. Кнопки

Кнопки являются одним из самых понятных и интуитивных элементов интерфейса. По их виду сразу становится понятно, что единственное действие, которое с ними можно производить — это нажимать на них. Этому событию чаще всего и ставят в соответствие выполнение сценариев, написанных на соответствующих языках программирования, если такие сценарии должны выполняться при отправке данных на сервер..

Кнопку на веб-странице можно создать двумя способами — с помощью тега `<INPUT>` и тега `<BUTTON>`.

`<INPUT TYPE="button">`

Параметры кнопки перечислены в табл. 2.8.

Табл. 2.8.

Параметры кнопок	
Атрибут	Описание
NAME	Имя кнопки, предназначено для того, чтобы обработчик формы мог его идентифицировать.
VALUE	Текст надписи на кнопке.
ID	Идентификатор кнопки, предназначено для того, чтобы обработчик формы мог его идентифицировать.

В отличие от тега `<INPUT>`, `<BUTTON>` предлагает расширенные возможности по созданию кнопок. Например, на подобной кнопке можно размещать любые элементы HTML, в том числе изображения и таблицы. Ниже показаны разные виды кнопок, полученные с помощью указанных тегов.



Рис. 2.35. Создание кнопок

Помимо кнопок общего назначения, в HTML можно создавать кнопки, которым уже заранее поставлены в соответствие жестко определенные действия: отправка данных на сервер или очистка формы.

Кнопка SUBMIT

Для отправки данных на сервер предназначена специальная кнопка SUBMIT. Ее вид ничем не отличается от других кнопок, но при нажатии на нее происходит выполнение серверной программы, указанной параметром ACTION тега <FORM>. Эта программа, называемая еще обработчиком формы, получает данные, введенные пользователем в полях формы, производит с ними необходимые манипуляции, после чего возвращает результат в виде HTML-документа. Что именно делает обработчик, зависит от автора сайта, например, подобная технология применяется при создании опросов, форумов, гостевых книг, тестов и многих других вещей.

Синтаксис создания кнопки SUBMIT зависит от используемого тега <INPUT> или <BUTTON>.

<INPUT TYPE="submit">

<BUTTON TYPE="submit">Надпись на кнопке</button>

Параметры такие же, что и у обычных кнопок.

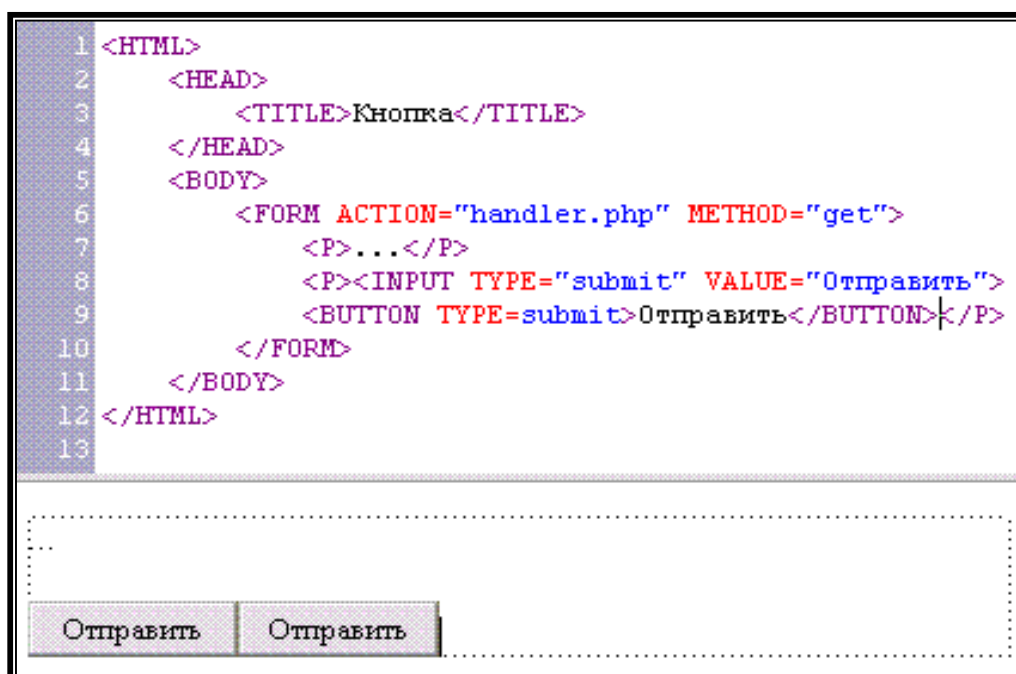


Рис. 2.36. Кнопка для отправки данных на сервер

Отметим также, что параметр NAME для этого типа кнопки может быть опущен. Для кнопки, созданной с помощью тега `<INPUT>` можно также не указывать атрибут VALUE: на кнопке автоматически появится надпись «Подача запроса» для русской версии браузера Internet Explorer, «Отправить запрос» для русской версии Firefox.

Кнопка RESET

При нажатии на кнопку RESET, данные формы возвращаются в первоначальное значение. Как правило, эту кнопку применяют для очистки введенной в полях формы информации.

Синтаксис создания указанной кнопки прост и похож на другие кнопки.

`<INPUT TYPE="reset">`

`<BUTTON TYPE="reset">Надпись на кнопке</button>`

В примере на рис. 2.37 показана форма с одним текстовым полем, которое уже содержит предварительно введенный текст с помощью параметра value тега `<INPUT>`. После изменения текста и нажатия на кнопку «Очистить», значение поля будет восстановлено и в нем снова появится надпись «Введите текст».

```
1 <HTML>
2   <HEAD>
3     <TITLE>Кнопка</TITLE>
4   </HEAD>
5
6   <BODY>
7     <FORM ACTION="handler.php">
8       <P><INPUT TYPE="text" VALUE="Введите текст"></P>
9       <P><INPUT TYPE="submit" VALUE="Отправить"><INPUT TYPE="reset" VALUE="Очистить"></P>
10    </FORM>
11  </BODY>
12 </HTML>
13
```

Введите текст

Отправить Очистить

Рис. 2.37. Кнопка для очистки формы

Если на кнопке, созданной с помощью тега `<INPUT>` опустить атрибут VALUE, на кнопке по умолчанию будет добавлен текст «Сброс» или «Reset».

Кнопка с изображением

Кнопка с изображением аналогично по действию кнопке SUBMIT, но представляет собой рисунок. Это расширяет возможности дизайнерских изысков по оформлению формы. Когда пользователь нажимает на рисунок, данные формы отправляются на сервер и обрабатываются программой, заданной параметром ACTION тега <FORM>.

Изображение в форме создается следующим образом.

<INPUT TYPE="image">

Параметры перечислены в табл. 2.8.

Табл. 2.8.

Параметры поля с изображением

Параметр	Описание
ALIGN	Определяет выравнивание изображения.
ALT	Альтернативный текст для кнопки с изображением.
BORDER	Толщина рамки вокруг изображения в пикселах.
NAME	Имя поля для обращения к нему из скриптов или для получения значения обработчиком формы.
SRC	URL-адрес графического файла, изображение которого должно использоваться в качестве фона

На рис. 2.38 показано использование поля с изображением.

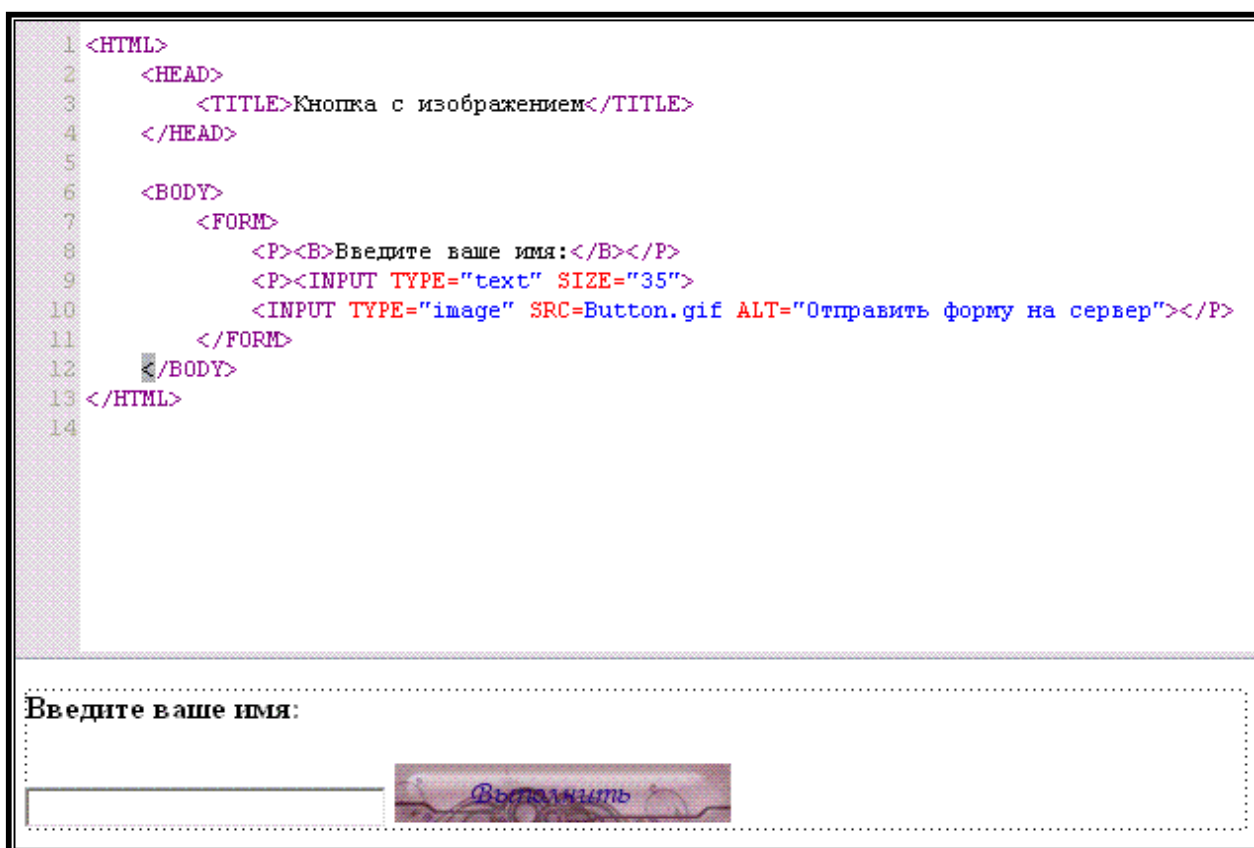


Рис. 2.38. Кнопка с изображением

2.7.2.9. Поле отправки файла

Для того чтобы можно было отправить на сервер файл, используется специальное поле. Такой элемент формы отображается как текстовое поле, рядом с которой располагается кнопка Обзор. При нажатии на эту кнопку открывается окно для выбора файла, где можно указать, какой файл пользователь желает использовать.

Синтаксис поля для отправки файла следующий.

<input type="file" параметры>

Атрибуты для элемента управления перечислены в табл. 2.9.

Атрибуты поля для отправки файла

Атрибут	Описание
SIZE	Ширина текстового поля, которое определяется числом символов моноширинного шрифта.
MAXLENGTH	Устанавливает максимальное число символов, которое может быть введено пользователем в текстовом поле.
NAME	Имя поля, используется для его идентификации обработчиком формы.
ID	Идентификатор элемента управления для обращения к нему из скриптов или для получения значения обработчиком формы.

Создание поля для отправки файла показано на рис. 2.39.

The image shows a screenshot of an HTML editor with a code window and a preview window. The code window displays the following HTML code:

```

1 <HTML>
2   <HEAD>
3     <TITLE>Отправка файла на сервер</TITLE>
4   </HEAD>
5   <BODY>
6     <FORM ENCTYPE="multipart/form-data" METHOD="post">
7       <P><B>Укажите файл для отправки на сервер:</B> <INPUT TYPE="file" SIZE="30"></P>
8       <P><INPUT TYPE="submit" VALUE="Отправить"></P>
9     </FORM>
10  </BODY>
11 </HTML>
12

```

The preview window below the code shows the rendered HTML form. It contains a text input field with the placeholder text "Укажите файл для отправки на сервер:" followed by a file selection button labeled "Обзор...". Below the input field is a submit button labeled "Отправить".

Рис. 2.39. Создание поля для отправки файла

Результат данного примера показан ниже.

Параметр формы `enctype` со значением `multipart/form-data` нужен для корректной передачи файла. Если его не указать, будет передан лишь путь к файлу.

Поскольку графические файлы занимают относительно большой объем данных, их следует отправлять на сервер с помощью метода `POST`, как показано в примере.

2.7.2.10. Группировка элементов формы.

При создании сложной формы не обойтись без визуального отделения одного логического блока от другого. Этого можно добиться, используя внутри тега `<FORM>` сочетания тегов и стилей. Например, элементы формы можно выделить, если использовать для них фоновый цвет или рамку, задавая их через CSS. Кроме того, существует и другой подход, который состоит в применении тега `<FIELDSET>`. Этот контейнер группирует элементы формы, отображая вокруг них рамку (рис 2.40).

The image shows a screenshot of an HTML editor. The top part displays the HTML code for a form, with line numbers 1 through 17 on the left. The code uses the `<FIELDSET>` tag to group form elements, including a `<LEGEND>` tag for a title and three `<INPUT type="checkbox">` elements with labels. The bottom part shows the rendered output of this code, which is a form with a title 'Работа со временем' and three checkboxes with their corresponding labels.

```
1 <HTML>
2   <HEAD>
3     <TITLE>Fieldset</TITLE>
4   </HEAD>
5   <BODY>
6     <FORM>
7       <FIELDSET>
8         <LEGEND>Работа со временем</LEGEND>
9         <INPUT TYPE="checkbox" VALUE="t1"> Создание пунктуальности (никогда не будете
10        никуда опаздывать).<BR>
11        <INPUT TYPE="checkbox" VALUE="t2"> Излечение от пунктуальности (никогда никуда
12        не будете торопиться).<BR>
13        <INPUT TYPE="checkbox" VALUE="t3"> Изменение восприятия времени. </FIELDSET>
14      </FORM>
15    </BODY>
16  </HTML>
17
```

Работа со временем

☐ Создание пунктуальности (никогда не будете никуда опаздывать).

☐ Излечение от пунктуальности (никогда никуда не будете торопиться).

☐ Изменение восприятия времени.

Рис. 2.40. Использование тега `<FIELDSET>`

Чтобы добавить к рамке, визуально группирующей элементы, специальный заголовок применяется контейнер `<LEGEND>`, который должен располагаться в теге `<FIELDSET>`. Внутри тега `<LEGEND>` допустимо использовать текст и теги форматирования, вроде ``, `<I>`, `<SUP>`, `<SUB>`, а также стили.

2.7.2.11. Переход между полями с помощью табуляции

При достаточно большом количестве полей формы, которые необходимо

заполнить, переходить между ними с помощью курсора мыши становится утомительно. При этом требуется навести курсор на соответствующее поле, нажать кнопку мыши, и только после этого вводить нужное значение. Как альтернатива, используется клавиша <Tab>, которая позволяет быстро переключать фокус с одного поля на другое. Атрибут TABINDEX определяет последовательность перехода между полями при нажатии на <Tab>.

2.7.2.12. Ограничение доступа к элементам формы.

У любого элемента формы есть два состояния, которые ограничивают доступ к элементу или ввод данных, — блокирование (disabled) и только для чтения (readonly).

Блокирование элемента не позволяет вообще производить с ним каких-либо действий. Максимум на что будет способен пользователь (да и то не во всех браузерах) - выделять и копировать содержимое заблокированного текстового поля. Заблокированное поле помечается обычно серым цветом

Блокирование элементов форм обычно используется для того, чтобы динамически с помощью скриптов изменять значение поля. Пользователь же не должен в подобном случае иметь доступ к полю, поэтому оно блокируется.

Установка блокирования соответствующего элемента управления осуществляется с помощью параметра DISABLED, который можно указывать для любого тега, описывающего элемент управления, отвечающий за ввод данных.

Поля формы можно не только блокировать, но и делать их только для чтения. В этом случае доступ к ним сохраняется, но изменять значения, заданные по умолчанию, нельзя. Разумеется, речь идет только о текстовых полях, где требуется вводить текст. Выделять и копировать его можно, но

изменить не получится. Для установки режима «только для чтения» используется параметр `readonly`.

2.8. Фреймы

Фреймы позволяют разбивать окно просмотра браузера на несколько прямоугольных областей, располагающихся рядом друг с другом. В каждую из таких областей можно загрузить отдельный HTML-документ, просмотр которого осуществляется независимо от других. Между фреймами можно организовать взаимодействие заключающееся в том, что выбор ссылки в одном из фреймов может привести к загрузке документа в другой фрейм. При создании страниц с фреймами разрабатываются несколько HTML-файлов. Документы раскладки (layout), используются для создания структуры Web-страниц, то есть разделения их на несколько областей. Документы содержания (content) предназначены для заполнения информацией каждой из областей.

2.8.1. Сфера применения фреймов

Выбор фреймовой структуры отображения информации на WWW оправдан в следующих случаях:

- при необходимости организовать управление загрузкой документов в одну из областей окна просмотра браузера при работе в другой области;
- для расположения в определённом месте окна просмотра информации, которая должна постоянно находиться на экране вне зависимости от содержания других областей экрана.
- для представления информации, которую удобно расположить в нескольких смежных областях окна, каждая из которых может просматриваться независимо.

Приведённый список не исчерпывает все возможные случаи, где можно применить фреймы, а носит рекомендательный характер.

2.8.2. Создание Web-страниц с фреймами

Итак, как же создать Web-страницу с фреймами? Сначала необходимо продумать разбиение экрана на области. Вторым этапом является подготовка HTML-файлов для каждой области. Они создаются по тем же правилам, что и другие гипертекстовые документы. Нужно только учитывать размер области, в которой будет демонстрироваться каждый из них.

2.8.2.1. Элемент <FRAMESET>

Фреймы определяются в структуре <FRAMESET> , которая используется для страниц, содержащих фреймы, вместо раздела <BODY> обычного документа. Web-страницы составленные из фреймов, не могут содержать раздел <BODY> в своём HTML-коде. В свою очередь, страницы с разделом <BODY> не могут использовать фреймы.

Контейнер из тегов <FRAMESET> и </FRAMESET> обрамляет каждый блок определений фрейма. Внутри контейнера <FRAMESET> могут содержаться только теги <FRAME> и вложенные тэги <FRAMESET> .

Тэг <FRAMESET> имеет два параметра: rows и cols . Горизонтальное деление экрана задаётся при помощи атрибута rows , а вертикальное - при помощи атрибута cols . Значение атрибута могут быть выражены в пикселах или процентах. Кроме того используется символ *, для обозначения оставшейся части экрана.

Приведём несколько примеров:

cols= 50%, 50%	деление области просмотра по вертикали пополам (принцип программы Norton Commander);
rows=150, 30%, *	для верхней горизонтальной области отведено 150 пикселей, для средней - 30% доступного пространства, а для нижней всё что остаётся;
cols=*, 4*	стиль для любителей головоломок, правая вертикальная область в четыре раза шире левой; эту формулу можно записать так: cols=20%, 80%

2.8.2.2. Элемент <FRAME>

Элемент разметки <FRAME> определяет одиночный фрейм. Он должен располагаться внутри пары тегов <FRAMESET> и </FRAMESET> . Этот контейнер не имеет завершающего тега. Всё определение одиночного фрейма выполняется одной строчкой HTML-кода.

Тег <FRAME> имеет шесть атрибутов: SRC, NAME, MARGINWIDTH, MARGINHEIGHT, SCROLLING и NORESIZE.

На практике в теге <FRAME> редко используются одновременно все атрибуты. Наиболее важный атрибут - SRC (сокращение от слова source). Значение этого атрибута определяет URL-адрес документа, который будет загружён изначально в данный фрейм. Обычно в качестве такого адреса записывается имя HTML-файла, расположенного в том же самом каталоге, что и основной документ. Тогда строка определения фрейма будет выглядеть, например, так:

<FRAME src="sample.htm">

Разумеется, в качестве значения SRC может быть задан любой допустимый URL-адрес.

Атрибут NAME определяет имя фрейма, которое может использоваться для ссылки к данному фрейму. Обычно ссылка задаётся из другого фрейма, располагающегося на той же самой странице. Например:

<FRAME src="sample.htm" name="frame_1">

Такая запись создаёт фрейм с именем "frame_1", на который может быть выполнена ссылка. Например:

*** Кликните здесь для загрузки документа other.htm во фрейм с именем frame_1 ***

Обратите внимание на атрибут TARGET , который ссылается на имя фрейма. Если для фрейма не задано имя, то будет создан фрейм без имени, и не будет возможности использовать ссылки на него из другого фрейма. Имена фреймов должны начинаться с алфавитно-цифрового символа.

Атрибуты `MARGINWIDTH` и `MARGINHEIGHT` дают возможность устанавливать ширину полей фрейма. Например:

`<FRAME src="sample.htm" marginwidth="5" marginheight="7">`

Данный фрейм имеет поля сверху и снизу по 5 пикселей, а слева и справа по 7 пикселей. Не забудьте, что здесь идёт речь о полях, а не о рамках. Параметры `MARGINWIDTH` и `MARGINHEIGHT` определяют пространство внутри фрейма, в пределах которого не будет располагаться никакая информация. Минимально допустимое значение этих параметров равно единице.

Для фреймов будут автоматически создаваться и отображаться полосы прокрутки, если содержимое фрейма не помещается полностью в отведённом пространстве. Иногда это нарушает дизайн страницы, поэтому было бы удобно иметь возможность управлять отображением полос прокрутки. Для этих целей используется атрибут `SCROLLING`. Форма записи:

`<FRAME src="sample.htm" scrolling="yes/no/auto">`

Атрибут `SCROLLING` может принимать три значения: `yes`, `no` или `auto`. Значение `auto` действует также, как и в случае отсутствия атрибута. Значение `yes` вызывает появление полос прокрутки вне зависимости от необходимости этого, а `no` - запрещает их появление.

Обычно пользователь может изменять размер фреймов при просмотре страницы. Если установить курсор мыши на рамки фрейма, то курсор примет форму, указывающую на возможность изменения размеров, и позволит выполнить перемещение рамки в нужное место. Это иногда нарушает структуру красиво спроектированных фреймов. Для предотвращения возможности изменения пользователем размера фреймов следует воспользоваться параметром `noresize`. Этот параметр не требует никаких значений. Естественно, когда задан параметр `noresize` для одного из фреймов, то размер любого из смежных фреймов также не сможет быть изменён.

2.8.2.3. Элемент <NOFRAMES>

Этот элемент используется, чтобы предусмотреть ситуацию, когда браузер не поддерживает фреймы. В этом случае нужно вывести на экран предупреждающее сообщение или адресовать клиента к другой странице. Фрагмент кода может быть записан следующим образом:

```
<NOFRAMES>  
<P> Для просмотра этой страницы необходим, браузер  
поддерживающий фреймы </p>  
Вы можете просмотреть  
<A href="без_фреймов.htm"> упрощенную версию </a> страницы  
</NOFRAMES>
```

Разумеется, браузеры поддерживающие фреймы не станут воспроизводить этот код.

Пример страницы с фреймами приведен на рис. 8.1. Обратите внимание на логику вложения фреймовой структуры, состоящей из двух вертикальных фреймов, внутрь среднего фрейма фреймовой структуры, состоящей из трех горизонтальных фреймов.

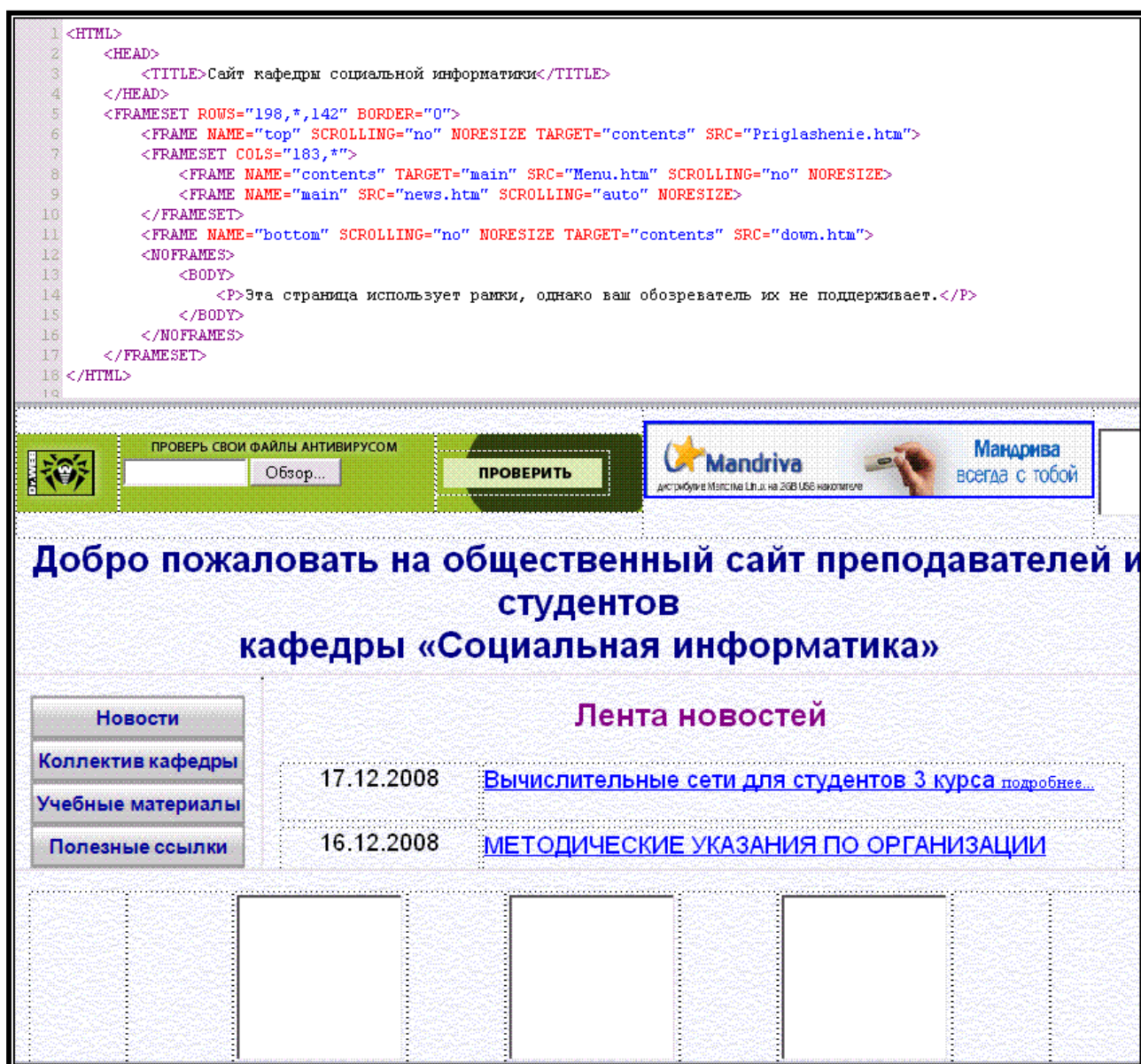


Рис. 2.41. Пример страницы с фреймами

Глава 3. Основы использования CSS – каскадных таблиц стилей

3.1. Понятие CSS

CSS (Cascading Style Sheets, каскадные таблицы стилей) — это набор параметров форматирования, который применяется к элементам веб-страницы для управления их видом и положением.

Стили являются удобным, практичным и эффективным инструментом при верстке веб-страниц и оформлении текста, ссылок, изображений и других элементов.

Использование при верстке HTML-документа CSS дает следующие преимущества:

- **Разделение оформления и содержания.** Идея о том, чтобы код HTML был свободен от элементов оформления вроде установки цвета, размера шрифта и других параметров, стара как мир. В идеале, веб-страница должна содержать только теги логического форматирования, а вид элементов задается через стили. При подобном разделении формирование дизайна и верстка сайта могут вестись параллельно.
- **Единое оформление документов.** Сайт это не просто набор связанных между собой документов, но и единое расположение основных блоков и их оформление. Применение единообразного оформления заголовков, основного текста и других элементов создает преемственность между страницами и облегчает пользователям работу с сайтом и его восприятие в целом. Разработчикам же использование стилей существенно упрощает проектирование дизайна.
- **Централизованное хранение.** Стили, как правило, хранятся в одном или нескольких специальных файлах, ссылка на которые указывается во всех документах сайта. Благодаря этому удобно править стиль в одном месте, при этом оформление элементов автоматически меняется на всех

страницах, которые связаны с указанным файлом. Вместо того чтобы модифицировать десятки HTML-файлов, достаточно отредактировать один файл со стилем и оформление нужных документов сразу же поменяется.

- **Расширенные возможности.** В отличие от HTML стили имеют гораздо больше возможностей по оформлению элементов веб-страниц. Простыми средствами можно изменить цвет фона элемента, добавить рамку, установить шрифт, определить размеры, положение и многое другое.
- **Быстрая работа.** При хранении стилей в отдельном файле, он кэшируется и при повторном обращении к нему извлекается из кэша¹ браузера. За счет кэширования и того, что стили хранятся в отдельном файле, уменьшается код веб-страниц и снижается время загрузки документов.

3.2. Подключение CSS

Для добавления стилей на веб-страницу существует несколько способов, которые различаются своими возможностями и назначением. Далее рассмотрим способы подключения CSS.

3.2.1. Таблица связанных стилей

При использовании таблицы связанных стилей описание селекторов и их свойств располагается в отдельном файле, как правило, с расширением `css`, а для связывания документа с этим файлом применяется тег `<LINK>`. Данный тег помещается в контейнер `<HEAD>`, как показано в следующем примере.

¹ Кэшем здесь и далее называется специальное место на локальном компьютере пользователя, куда браузер сохраняет файлы сайта при первом обращении к нему. При следующем обращении к сайту такие файлы уже не скачиваются по сети, а берутся с локального диска. Такой подход позволяет существенно повысить скорость загрузки веб-страниц.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
    charset=windows-1251">
    <title>Стили</title>
    <link rel="stylesheet" type="text/css" href="mysite.css">
    <link rel="stylesheet" type="text/css"
    href="http://www.htmlbook.ru/main.css">
  </head>
  <body>
    <h1>Заголовок</h1>
    <p>Текст</p>
  </body>
</html>

```

Значения атрибутов тега <LINK> — rel и type остаются неизменными, как показано в данном примере. Параметр href задает путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Заметьте, что таким образом можно подключать таблицу стилей, которая находится на другом сайте.

Содержимое файла mysite.css подключаемого посредством тега <LINK> приведено в следующем примере:

```

H1 {
  color: navy;
  font-size: 200%;
  font-family: Arial, Verdana, sans-serif;
  text-align: center;
}

```

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

Как видно из данного примера, файл со стилем не хранит никаких данных, кроме синтаксиса CSS. В свою очередь и HTML-документ содержит только ссылку на файл со стилем, т.е. таким способом в полной мере реализуется принцип разделения содержимого и оформления сайта. Поэтому использование таблицы связанных стилей является наиболее универсальным и удобным методом добавления стиля на сайт. Ведь стили хранятся в одном файле, а в HTML-документах указывается только ссылка на него.

3.2.2. Таблица глобальных стилей

При использовании таблицы глобальных стилей свойства CSS описываются в самом документе и обычно располагаются в заголовке веб-страницы. По своей гибкости и возможностям этот способ добавления стиля уступает предыдущему, но также позволяет размещать все стили в одном месте. В данном случае, прямо в теле документа, с помощью контейнера <STYLE>, как показано в следующем примере.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=windows-  
1251">  
<title>Глобальные стили</title>  
<style type="text/css">  
H1 {  
font-size: 120%;
```



```

    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #336;
}
</style>
</head>
<body>
    <h1>Hello, world!</h1>
</body>
</html>

```

В данном примере определен стиль тега <H1>, который затем можно повсеместно использовать на данной веб-странице.

Отметим, что таблица глобальных стилей в принципе может размещаться не только внутри контейнера <HEAD>, но также в любом месте кода HTML-документа.

3.2.3. Внутренние стили

Внутренний стиль является по существу расширением для одиночного тега используемого на веб-странице. Для определения стиля используется атрибут тега style, а его значения указываются с помощью языка таблицы стилей:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
    <title>Внутренние стили</title>
</head>
<body>

```

```
<h1 style='font-size: 120%; font-family: Verdana, Arial, Helvetica,  
sans-serif;  
color: #3366''>Заголовок</h1>  
</body>  
</html>
```

В данном примере стиль тега <H1> задается с помощью параметра style, в котором через точку с запятой перечисляются стилевые атрибуты.

3.3. Базовый синтаксис

Способ записи CSS отличается от формы использования тегов HTML и в общем виде имеет следующий синтаксис.

Селектор { свойство1: значение; свойство2: значение; ... }

Селектором называется имя стиля, в котором указаны параметры форматирования. Селекторы делятся на несколько типов: селекторы тегов, идентификаторы и классы, они подробно описаны далее. После указания селектора идут фигурные скобки, в которых записывается необходимое стилевое свойство, а его значение указывается после двоеточия. Параметры разделяются между собой точкой с запятой, в конце этот символ можно опустить.

CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от желания разработчика. Так, в следующем примере показаны две разновидности оформления селекторов и их правил:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
    <title>Использование стилей</title>
    <style type="text/css">
      h1 { color: #a6780a; font-weight: normal; }
      H2 {
        color: olive;
        border-bottom: 2px solid black;
      }
    </style>
  </head>
  <body>

    <h1>Заголовок 1</h1>
    <h2>Заголовок 2</h2>

  </body>
</html>

```

В данном примере свойства селектора H1 записаны в одну строку, а для селектора H2 каждый параметр находится на отдельной строке. Во втором случае легче отыскивать параметры и править их по необходимости, но при этом незначительно возрастает объем данных за счет активного использования пробелов и переносов строк.

Отметим, что имена селекторов обязательно должны начинаться с латинского символа (a-z, A-Z) и могут содержать в себе цифры, например — terminator4, Titanic2, extra.

3.4. Селекторы тегов

В качестве селектора может выступать любой тег HTML для которого определяются правила форматирования, такие как: цвет, фон, размер и т.д. Правила задаются в следующем виде.

Тег { свойство1: значение; свойство2: значение; ... }

Вначале указывается имя тега, оформление которого будет переопределено, заглавными или прописными символами не имеет значения. Внутри фигурных скобок пишется свойство CSS, а после двоеточия — его значение. Набор параметров разделяется между собой точкой с запятой и может располагаться как в одну строку, так и в несколько:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
  <title>Селекторы тегов</title>
  <style type="text/css">
    P {
      text-align: justify; /* Выравнивание по ширине */
      color: green /* Зеленый цвет текста */
    }
  </style>
</head>
<body>
```

```
<p>Более эффективным способом ловли льва в пустыне  
является метод золотого сечения. При его использовании пустыня  
делится  
на две неравные части, размер которых подчиняется правилу  
золотого  
сечения.</p>  
</body>  
</html>
```

В данном примере изменяется цвет текста и выравнивание текста параграфа. Стилль будет применяться только к тексту, который располагается внутри контейнера <P>.

Следует понимать, что стилль можно добавить только для тегов, которые непосредственно отображаются в контейнере <BODY>.

3.5. Классы

Классы применяют, когда необходимо определить стилль для индивидуального элемента веб-страницы или задать разные стили для одного тега. При использовании совместно с тегами синтаксис для классов будет следующий.

Тег.Имя класса { свойство1: значение; свойство2: значение; ... }

Внутри стилевой таблицы вначале пишется желаемый тег, а затем, через точку пользовательское имя класса. Чтобы указать в коде HTML, что тег используется с определенным стилем, к тегу добавляется параметр class="Имя класса":

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
  <title>Классы</title>
  <style type="text/css">
    P { /* Обычный параграф */
      text-align: justify; /* Выравнивание текста по ширине */
    }
    P.cite { /* Параграф с классом cite */
      text-align: justify; /* Выравнивание текста по ширине */
      color: navy; /* Синий цвет текста */
      font-style: italic; /* Курсивное начертание */
    }
  </style>
</head>
<body>

```

<p>При работе на вычислительной технике необходимо сесть так, чтобы руки с предплечьями образовывали прямой угол, глаза поставить на расстояние 30-40 см от рабочей поверхности монитора. Набирать на клавиатуре следует подушечками пальцев короткими отрывистыми ударами.</p>

<p class="cite">Предельно допустимая длина ногтей для женщин составляет 12-15 мм, для мужчин 3-5 мм. При выходе длины ногтей за нормы регламентируемые ГОСТом, во избежании риска

*поцарапать поверхность дорогостоящей техники, оператор
допускается к работе на компьютере только в верхонках.</p>*

</body>

</html>

Результат данного примера показан на рис. 3.1.

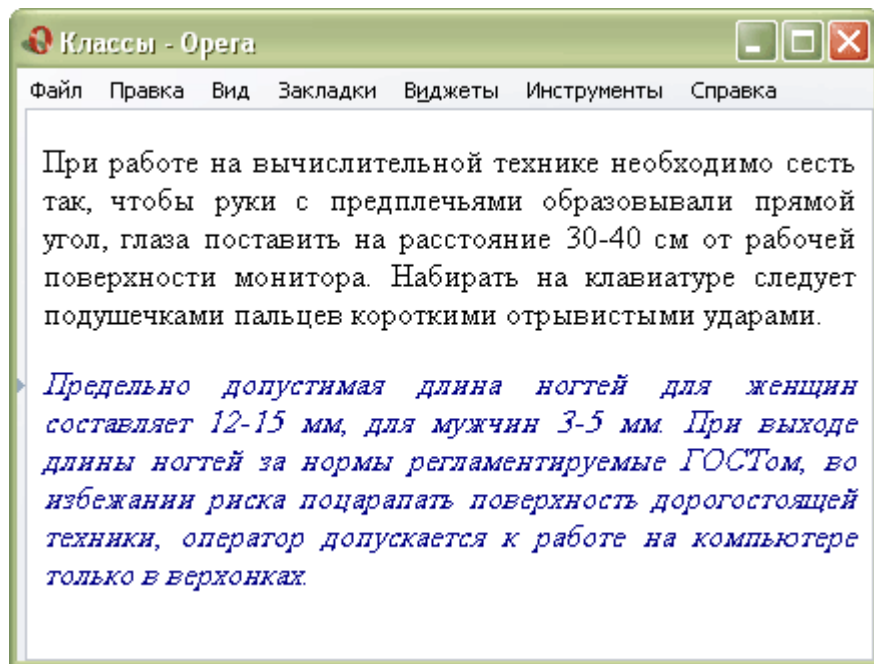


Рис. 3.1. Вид текста, оформленного с помощью стилей

Первый абзац выровнен по ширине с текстом черного цвета, а следующий, к которому применен класс cite — написан курсивом синего цвета.

Имена классов выбираются по желанию, главное, чтобы они были понятны и соответствовали их использованию, при этом имя должно всегда начинаться с символа. Можно, также, использовать классы и без указания тега. Синтаксис в этом случае будет следующий.

.Имя класса { свойство1: значение; свойство2: значение; ... }

При такой записи, класс можно применять к любому тегу:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
  <title>Классы</title>
  <style type="text/css">
    .cite {
      color: navy; /* Синий цвет текста */
      font-style: italic; /* Курсивное начертание */
    }
  </style>
</head>
<body>

  <p>Следует тщательно позаботиться о своем рабочем месте.
  Освещение в помещении отрегулировать таким образом, чтобы
  источник света находился сбоку или сзади оператора. Во избежании
  медицинских осложнений
  <b class="cite">стул рекомендуется выбирать с мягким
  сидением</b>.</p>

</body>
</html>

```

Результат применения селектора с именем cite к тегу показан на рис. 3.2.

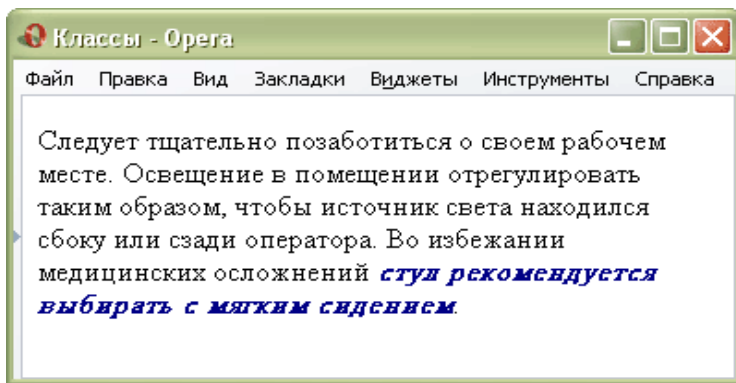


Рис. 3.2. Вид тега , оформленного с помощью стилей

Классы удобно использовать, когда нужно применить стиль к разным тегам веб-страницы: ячейкам таблицы, ссылкам, параграфам и др. Для изменения отдельных слов или даже букв, а также блоков, содержащих в себе разные элементы, употребляются теги и <DIV>. В примере 6.3 показано изменение стиля первой буквы предложения путем использования класса совместно с тегом .

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
  <title>Классы</title>
  <style type="text/css">
    .inicial {
      color: red; /* Красный цвет текста */
      font-size: 200%; /* Размер текста */
    }
  </style>
</head>
```

<body>

<p>Набирать на клавиатуре следует подушечками пальцев короткими отрывистыми ударами.</p>

</body>

</html>

Результат данного примера показан на рис. 3.3.

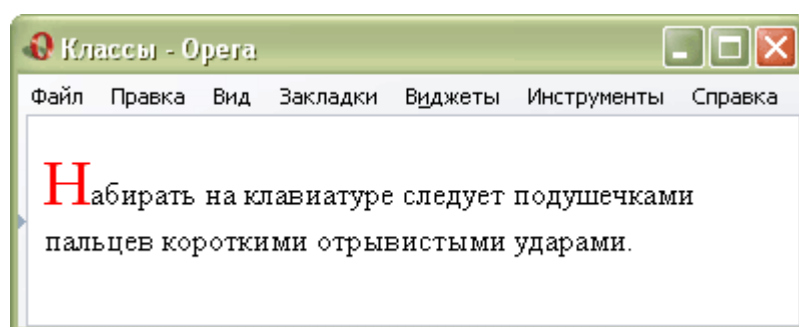


Рис. 3.3. Результат использования классов

3.6. Идентификаторы

Идентификатор (называемый также «ID селектор») определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты, что позволяет управлять стилем элемента динамически.

Синтаксис использования идентификатора следующий.

**#Имя идентификатора { свойство1: значение; свойство2: значение;
... }**

В отличие от классов идентификаторы должны быть уникальны, иными словами, встречаться в коде документа только один раз.

Обращение к идентификатору происходит аналогично классам, но в качестве ключевого слова у тега используется параметр `id`, значением которого выступает имя идентификатора (пример 7.1). Символ решетки при этом уже не

указывается.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
  <title>Идентификаторы</title>
  <style type="text/css">
    #help {
      position: absolute; /* Абсолютное позиционирование */
      left: 160px; /* Положение элемента от левого края */
      top: 50px; /* Положение от верхнего края */
      width: 225px; /* Ширина блока */
      height: 180px; /* Высота блока */
      background: #f0f0f0; /* Цвет фона */
    }
  </style>
</head>
<body>
  <div id="help">
    Этот элемент помогает в случае, когда вы находитесь в осознании
    того факта, что совершенно не понимаете, кто и как вам может
    помочь. Именно в этот момент мы и подсказываем, что помочь вам
    никто не сможет.
  </div>
</body>
</html>
```

В данном примере определяется стиль тега <DIV>, для которого указан селектор help через параметр id (рис. 3.4).

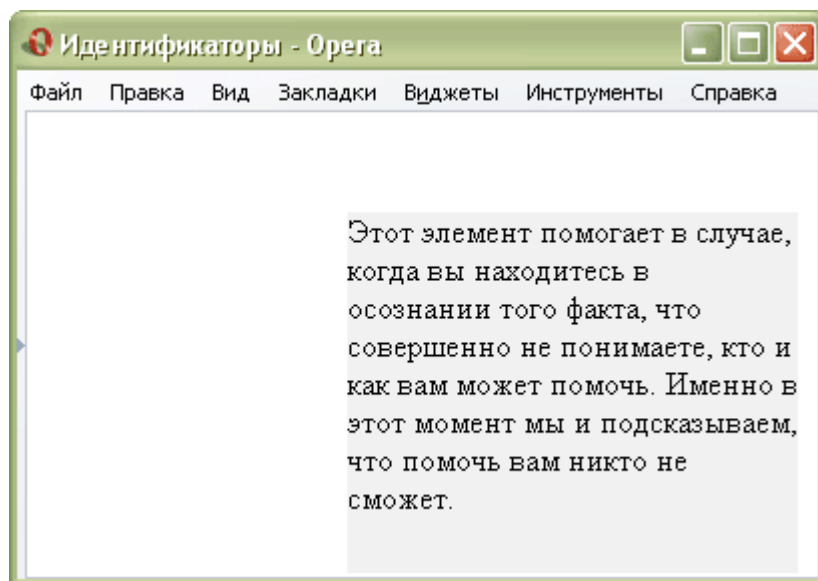


Рис. 3.4. Результат применения идентификатора

Как и при использовании классов, идентификаторы можно применять к конкретному тегу. Синтаксис при этом будет следующий.

Тег#Имя идентификатора { свойство1: значение; свойство2: значение; ... }

Вначале указывается имя тега, затем без пробелов символ решетки и название идентификатора. В следующем примере показано использование идентификатора применительно к тегу <P>.

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

"http://www.w3.org/TR/html4/strict.dtd">

<html>

<head>

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

<title>Идентификаторы</title>

```

<style type="text/css">
P {
  color: green; /* Зеленый цвет текста */
  font-style: italic; /* Курсивное начертание текста */
}
P#opa {
  color: red; /* Красный цвет текста */
  border: 1px solid #666; /* Параметры рамки */
  background: #eee; /* Цвет фона */
  padding: 5px; /* Поля вокруг текста */
}
</style>
</head>
<body>
<p>Обычный параграф</p>
<p id="opa">Параграф необычный</p>
</body>
</html>

```

Результат данного примера показан на рис. 3.5.

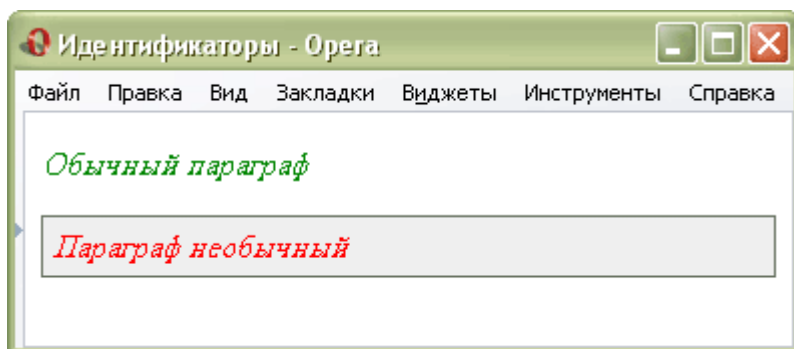


Рис. 3.5. Вид параграфов после применения стиля

В данном примере вводится стиль для тега `<P>` и для такого же тега, но с указанием идентификатора «opa».

3.7. Правила создания стиля

При написании достаточно объемного CSS-файла следует придерживаться некоторых общих рекомендаций, которые помогут избежать ошибок, сделать код понятным и удобным:

- **Пишите все правила для каждого селектора в одном месте.** Допускается для каждого селектора добавлять каждый стилевой параметр и его значение по отдельности, как это показано в следующем примере:

```
TD { background: olive; }  
TD { color: white; }  
TD { border: 1px solid black; }
```

Однако такая запись не очень удобна. Приходится повторять несколько раз один и тот же селектор, да и легко запутаться в их количестве. Поэтому пишите аргументы для каждого селектора вместе. Указанный набор записей в таком случае получит следующий вид:

```
TD {  
    background: olive;  
    color: white;  
    border: 1px solid black;  
}
```

Форма написания — в одну строку или несколько — зависит от воли автора. Заметим только что, когда каждый параметр занимает отдельную строку, проще отыскивать взглядом нужное значение. Соответственно, быстрее и удобнее происходит редактирование кода CSS.

- **Имеет приоритет значение, указанное в коде ниже.** Если для селектора вначале задается параметр с одним значением, а затем тот же параметр, но уже с другим значением, то применяться будет значение, которое в коде установлено ниже:

P { color: green; }

P { color: red; }

В данном примере для селектора P цвет текста вначале устанавливается зеленым, а затем красным. Поскольку значение red расположено ниже, то оно в итоге и будет применяться к параметру color.

На самом деле такой записи лучше вообще избегать и удалять повторяющиеся параметры селекторов. Но подобное может произойти не явно, например, в случае подключения разных стилевых файлов, в которых содержатся одинаковые селекторы.

- **Начинайте с селекторов верхнего уровня.** Учитывая, что многие стилевые свойства элементов наследуются от своих родителей, начинать таблицу стилей лучше именно с селекторов, которые выступают контейнерами для других элементов. В частности, это BODY, TABLE, UL, OL и т.д. Если требуется задать гарнитуру шрифта для всего текста веб-страницы, то надо применить параметр font-family к селектору BODY, как показано в следующем примере.

BODY {

font-family: "Times New Roman", Times, serif; /* Гарнитура

шрифта */

```
font-size: 110%; /* Размер шрифта */  
}
```

Наследование свойств позволяет не повторять многократно одни и те же параметры, если они заданы для селекторов верхнего уровня. Только учтите, что не все атрибуты наследуются и к некоторым из них вроде border, все же приходится обращаться несколько раз.

- **Группируйте селекторы с одинаковыми параметрами и значениями.**

Достоинство и удобство группирования состоит в применении набора параметров сразу к нескольким селекторам одновременно. Так, в следующем примере показано добавление одинаковых атрибутов для трех разных идентификаторов. Но поскольку для них требуется различный цвет фона, то он устанавливается уже ниже.

```
#col1, #col2, #col3 {  
font-family: Arial, Verdana, sans-serif; /* Гарнитура шрифта */  
font-size: 90%; /* Размер шрифта */  
font-weight: bold; /* Нормальное начертание */  
color: white; /* Цвет текста */  
}
```

```
/* Цвет фона для каждого слоя */  
#col1 { background: #ebe0c5; }  
#col2 { background: #bbe1c4; }  
#col3 { background: #add0d9; }
```


- **Используйте идентификаторы и классы.** Классы или идентификаторы удобно использовать, когда нужно применить один стиль к разным элементам веб-страницы: ячейкам таблицы, ссылкам, абзацам и т.д. Если тег перед именем класса не установлен, то он может добавлять к любому тегу:

```
.new {  
    ... /* Этот стиль можно использовать с любыми тегами */  
}
```

```
P.new {  
    ... /* Этот стиль работает только для тега <P> */  
}
```

Класс new в данном примере хотя и один, но стиль определяет для разных элементов, поэтому он различается.

- **Применяйте универсальные стилевые параметры.** Универсальные параметры задают одновременно сразу несколько значений. Примеры таких свойств: border (вид границы), padding (поля вокруг элемента), margin (отступы вокруг элемента).

Так, параметр padding определяет поля одновременно для всех четырех сторон элемента. Поэтому использование padding: 10px короче и понятнее, чем последовательное добавление аргументов padding-left, padding-top, padding-right и padding-bottom, определяющих поля для каждой стороны. Через padding также можно задать разные значения сверху, справа, снизу и слева. В следующем примере показано использование универсального стилевого атрибута margin.

```

P {
    margin-top: 10px;
    margin-right: 30px;
    margin-bottom: 5px;
    margin-left: 0;
}

P {
    margin: 10px 30px 5px 0;
}

```

В данном примере приведены две записи, дающие одинаковый результат, но запись с `margin` выглядит нагляднее и короче.

3.8. Основы CSS-верстки. Создание «резинового» шаблона.

3.8.1. Создание макета из одной колонки

Попробуем создать макет страницы, состоящий всего из одной колонки и сделать эту колонку «резиновой».

```

body {
    margin-left: 20%;
    margin-right: 20%;
}

```

Мы задали значения левого и правого полей элемента **BODY** в процентах, что позволяет содержимому занимать всю ширину окна браузера (за исключением полей естественно) и располагаться при этом точно в центре. Если мы хотим задать нашей колонке фиксированную ширину- поступим вот следующим образом:

```
body {  
    width:600px;  
}
```

При этом колонка будет выровнена по левому краю окна браузера. И если нам необходимо вновь отцентрировать ее, то без дополнительного блока не обойтись.

```
<div id="content">  
... содержимое блока ...  
</div>
```

и далее вот такой CSS код:

```
body {  
    width:600px;  
    padding-left:50%;  
}  
#content {  
    width:600px;  
    margin-left:-300px;  
}
```

Чтобы понять суть этого приема, надо хорошо понимать основы модели визуального форматирования. В частности точно представлять, из каких именно величин складывается общая ширина (высота) блока. При этом еще необходимо учитывать различия в реализации блочной модели конкретных браузеров. Особенно того, название которого Вы и так знаете.

В элементе **body**, который имеет ширину 600px мы задали левый отступ в

50%, добавив таким образом к ширине элемента половину ширины окна браузера. При этом левый край блока **content** расположился бы точно по центру окна, но мы предусмотрительно сдвигаем его ровно на половину ширины, на 300px влево, задавая отрицательное значение левого поля.

Теперь блок **content** расположен точно в центре экрана и ширина его фиксирована. Этот прием работает во всех популярных браузерах.

3.8.2. Макет из двух колонок

Попробуем теперь создать макет страницы с двумя колонками, изменяющими свою ширину в зависимости от ширины окна браузера, т.е. «резиновый» макет. И сначала создадим HTML-разметку, форматировать которую будем с помощью стилевых таблиц.

```
<div id="header">
... шапка страницы ...
</div>
<div id="content">
... основное содержимое страницы ...
</div>
<div id="sidebar">
... колонка меню ...
</div>
<div id="footer">
... подвал ...
</div>
```

А теперь CSS код, который сформирует нужное нам визуальное представление:

```
#content {  
  float:left;  
  width:67%;  
  background-color:#CCC;  
}  
#sidebar {  
  margin-left:67%;  
  background-color:#FFC;  
}  
#header {  
  background-color:#FCF;  
  text-align:center;  
}  
#footer {  
  clear:both;  
  text-align:center;  
  background-color:#CFF;  
}
```

Мы используем свойство **float** для того, чтобы задать положение блока **content** на странице и задаем ширину этого блока равной двум третям окна браузера. В соответствии с правилами CSS, последующее содержимое включенное в «плавающую модель», а в нашем случае это блок **sidebar**, будет «обтекать» блок **content** со стороны, противоположной направлению выравнивания, т.е. Справа.

Свойство **float** работает примерно как атрибут **align**, который применяется в HTML, но дает больше возможностей управления отображением элементов за счет использования полей, рамок и отступов.

Разный цвет фона блоков задан лишь для того, чтобы Вам легче было наблюдать границы блоков, если Вы вдруг захотите испытать этот пример.

И наконец, в блоке **footer** используется свойство **clear** для того, чтобы гарантировано обеспечить его отображение ниже любого элемента из «плавающей модели».

Правда такой вариант имеет недостаток. При наличии в блоке **sidebar** содержимого большего объема, чем в блоке **content**, «лишнее» содержимое перестает обтекать блок content и занимает всю ширину окна браузера.

Устранить недоразумение возможно, если подкорректировать значения левого отступа или левого поля элемента **sidebar**. Для этого мы используем свойство `margin-left:67%`.

3.8.3. Макет из трех колонок

Следующий шаг – создание «резинового» макета страницы из трех колонок. Как и в случае макета с двумя колонками сначала создадим HTML-разметку, которую будем в дальнейшем форматировать.

```

<div id="header">
... шапка страницы ...
</div>
<div id="sidebar">
... колонка меню ...
</div>
<div id="content">
... основное содержимое страницы ...
</div>
<div id="otherbar">
... колонка меню ...
</div>
<div id="footer">
... подвал ...
</div>

```

и отформатируем ее с помощью следующей таблицы стилей:

```

#content {
  float:left;
  width:50%;
  background-color:#CCC;
}
#sidebar {
  float:left;
  width:30%;
  background-color:#FFC;
}
#otherbar {
  float:left;
  width:20%;
  background-color:#FFC;
}
#header {
  background-color:#FCF;
  text-align:center;
}
#footer {
  clear:both;
  text-align:center;
  background-color:#CFF;
}

```

В этом решении все три колонки включены в плавающую модель, выравниваются по левому краю и суммарная их ширина не превышает 100%.

Однако следует очень внимательно относиться к полям, рамкам и отступам этих элементов, поскольку, если суммарная ширина колонок превысит 100%, то одна или две колонки будут расположены друг под другом.

Задания для самостоятельной работы.

Задание 1.

- Прodelайте все примеры, приведенные в качестве иллюстраций в пункте 2.3 (рис. 2.2 – 2.8). Каждый пример должен быть реализован в виде отдельной страницы.
- По приведенному ниже образцу, используя, где необходимо, элемент <PRE> создайте свое резюме. Рекомендуемые параметры:
 - ✓ **Цвет фона** для документа – голубой.
 - ✓ **Фамилия, имя** – заголовок первого уровня, выравнивание по центру.
 - ✓ Слова «**Цель**», «**Опыт работы**», «**Образование**», «**Увлечения**» - заголовки второго уровня.
 - ✓ **Для остального текста** – выравнивание – по левому краю, шрифт Arial, размер 12пт, параметры начертания (жирный, курсив, подчеркнутый) и отступы от левого края – по образцу.

Сергей Алексеев

ЦЕЛЬ:

Получение высокооплачиваемой работы

ОПЫТ РАБОТЫ:

1990-1994 ТОО «Башмачок» Москва

Руководитель планового отдела

- Введена новая система планирования.
- Увеличены объемы продаж на 13%.
- Уменьшены издержки производства на 23%.

1985-1990 ТОО «Башмачок» Москва

Заместитель руководителя планового отдела

- Увеличены объемы продаж на 7%.

- Организована единая компьютерная сеть.
- Введены в строй 4 филиала предприятия.

ОБРАЗОВАНИЕ:

1971-1975 Институт легкой промышленности Москва

- Факультет: Экономика легкой промышленности.
- Специальность: *Экономист*.

УВЛЕЧЕНИЯ

Компьютеры, автомобили, теннис, чтение.

Задание 2.

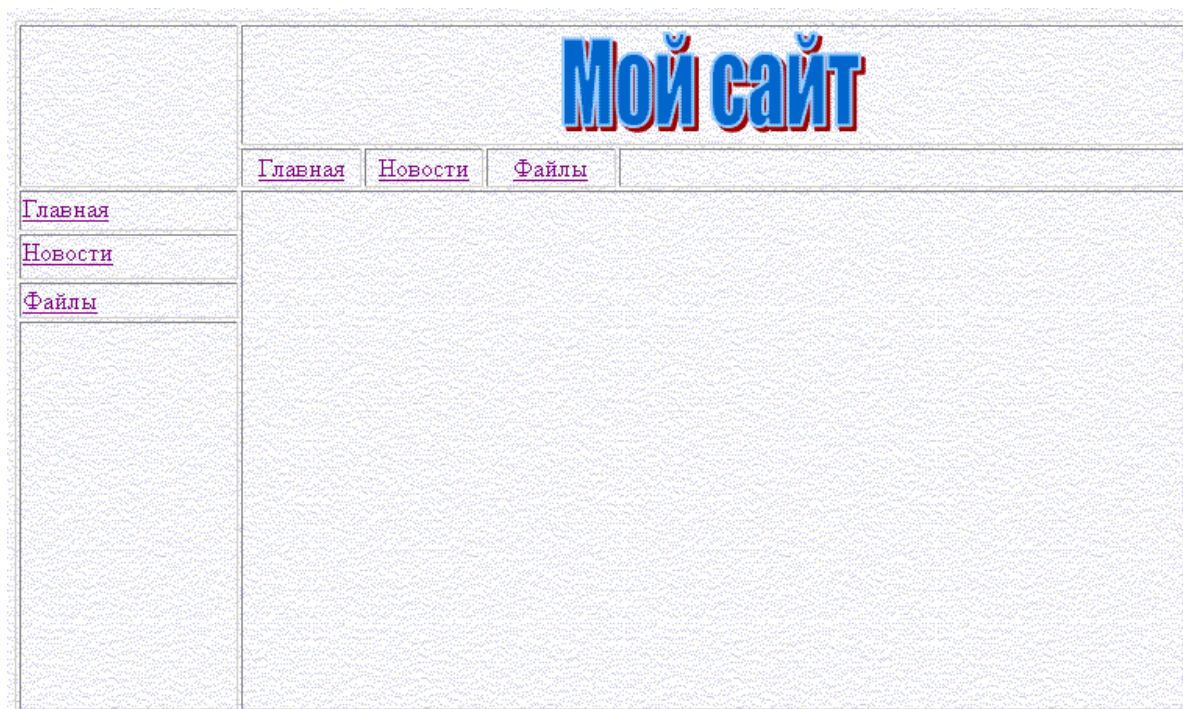
- Прodelайте все примеры, приведенные в качестве иллюстраций в пункте 2.4 (рис. 2.9 – 2.13). Каждый пример должен быть реализован в виде отдельной страницы.
- Взяв за основу текст пункта 2.2 (электронный вариант прилагается), создать серию html-страниц, отвечающих следующим требованиям:
 - ✓ Страница 1 (index.html) содержит заголовок лекции и оглавление, включающее заголовки параграфов;
 - ✓ Каждый параграф должен быть реализован в виде отдельной html-страницы (название — произвольно);
 - ✓ Если параграф содержит более мелкие элементы структуры (параграф 2), в начале страницы должно быть создано оглавление, содержащее заголовки этих элементов структуры;
 - ✓ Форматирование в html-документе должно соответствовать приложенному электронному варианту лекции;

- ✓ Внедряемые в html-документы рисунки должны храниться в папке с названием img.

Задание 3.

1. Прodelайте все примеры, приведенные в качестве иллюстраций в пункте 2.5 (рис. 2.14 – 2.21). Каждый пример должен быть реализован в виде отдельной страницы.
2. Создайте страницы со следующими заголовками:
 - Главная;
 - Новости;
 - Файлы.

Структура каждой из страниц должна соответствовать примеру, изображенному на рисунке:



Требования к страницам:

- ✓ Высота и ширина таблицы — 100% от размера окна браузера.
- ✓ Толщина линий сетки — 0.
- ✓ Для каждой из гиперссылок явным образом прописать цвет ссылки при наведении указателя мыши, цвет ссылки до перехода, цвет ссылки после перехода.
- ✓ Для текста на странице: шрифт — Arial, размер — 12пт, цвет — черный.
- ✓ Для фона и графического объекта использовать рисунки, располагающиеся в папке «ris_zadanie».
- ✓ В рабочей области страницы «Главная» должен располагаться текст следующего содержания: «Добро пожаловать на сайт, созданный Ивановым Иваном».
- ✓ В рабочей области страницы «Новости» должна содержаться таблица следующей структуры:

--	--

- ✓ В первом столбце — дата (выравнивание по центру), во втором столбце — текст новости (выравнивание по ширине).
- ✓ В рабочей области страницы «Файлы» должны располагаться гиперссылки на документы MS Word с текстами лекций по курсу «Разработка Web-приложений», оформленные в виде нумерованного списка (нумерация — арабскими цифрами).

Задание 4.

Все необходимые для выполнения задания графические объекты находятся в папке «ris_zad»

- С помощью программы GIMP сделайте пример, приведенный в качестве иллюстраций в пункте 2.6 – рис. 2.23 (каждая из страниц к которым осуществляется переход должна быть создана предварительно).
- С помощью программы Map Designer в отдельном файле создайте графическое меню на основе изображения, находящегося в файле Menu.jpg (ссылки для соответствующих активных областей должны указывать на страницы сайта, созданного в рамках выполнения пункта задания к лекции 5).
- Замените созданным графическим меню гипертекстовые ссылки, расположенные в левой области каждой из страниц сайта, созданного в рамках выполнения пункта 2 задания к лекции 5).

Задание 5.

1. Прodelайте все примеры, приведенные в качестве иллюстраций в пункте 2.7 (рис. 2.24 – 2.40). Каждый пример должен быть реализован в виде отдельной страницы.
2. По приведенному ниже образцу создайте страницу с формой регистрации.

Заполните приведенную ниже форму и нажмите ОТПРАВИТЬ. Знаком «*» отмечены поля, обязательные для заполнения.

Login*	<input type="text"/>	
Пароль*	<input type="password"/>	
Повторите пароль*	<input type="password"/>	
Фамилия*	<input type="text"/>	
Имя*	<input type="text"/>	
Отчество	<input type="text"/>	
Город*	<input type="text" value="Нижний Новгород"/>	
e-mail*	<input type="text"/>	
Пол	<input type="radio"/> мужской <input type="radio"/> женский	
Какими браузерами пользуетесь	<input type="checkbox"/> Internet Explorer	<input type="checkbox"/> Mozilla Firefox
	<input type="checkbox"/> Google Chrom	<input type="checkbox"/> Opera
	<input type="checkbox"/> Другой <input type="text"/>	
<input type="button" value="Отправить"/>		
<input type="button" value="Очистить"/>		

Обратите внимание, что выравнивание элементов управления и подписей к ним лучше всего проводить с помощью таблицы.

Для раскрывающегося списка «Город» используйте следующие варианты значений: Арзамас, Дзержинск, Кстово, Нижний Новгород.

- По приведенному ниже образцу создайте форму авторизации. Обратите внимание, что при заполнении поля «Пароль» формы вводимый текст должен заменяться символом «*».
- Свяжите страницы между собой с помощью гиперссылки, поставленной в соответствие слову «зарегистрируйтесь» на странице с формой авторизации.

**Для продолжения введите LOGIN и ПАРОЛЬ
после чего нажмите кнопку ВОЙТИ.**

Login

Пароль

Войти

Если Вы не являетесь зарегистрированным пользователем, [зарегистрируйтесь](#).

Задание 6.

С помощью системы фреймов создайте сайт структуры, аналогичной предложенной в пункте 2 задания 3.

Задание 7

1. Прodelать описанные в текущей лекции примеры
2. С помощью описанной в данной лекции технологии «резиновых» макетов создайте сайт структуры, аналогичной предложенной в пункте 2 задания 3.