МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО КЕМЕРОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ
ИНСТИТУТ ПИЩЕВОЙ ПРОМЫШЛЕННОСТИ
(УНИВЕРСИТЕТ)

А.В. Шафрай

ВЕБ-ДИЗАЙН В МАРКЕТИНГЕ УПАКОВКИ

Учебное пособие

для студентов направления подготовки 29.03.03.
«Технология полиграфического и упаковочного производства»

Кемерово 2016

УДК 658.512.2(07)
ББК 30.182я7
Ш30

Рецензенты:

Н.А. Русакова, доцент кафедры ЮНЕСКО по информационным вычислительным технологиям, Кемеровского государственного университета, канд. пед. наук;

А.С. Пфайф, технический директор Web-студии Origami, ИП «Пфайф Е.Д.»

Рекомендовано редакционно-издательским советом Кемеровского технологического института пищевой промышленности (университета)

Ш00 **Шафрай, А.В.**

Веб-дизайн в маркетинге упаковки: учебное пособие / А.В. Шафрай; Кемеровский технологический институт пищевой промышленности (университет). – Кемерово, 2016. – 106 с.

ISBN 978-5-89289-964-2

Рассматриваются технологии верстки шаблонов интернет-сайтов, такие как: HTML, CSS. Изложены основные принципы и методы разработки HTML шаблона и CSS файла. Приведена информация по способам размещения сайтов в интернете и выбору доменного имени.

Предназначено для студентов, обучающихся по направлению подготовки 29.03.03 «Технология полиграфического и упаковочного производства».

УДК 658.512.2(07)
ББК 30.182я7

ISBN 978-5-89289-964-2

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Маркетинг упаковки, как и маркетинг любого товара или услуги, понятие сложное и многогранное, которое можно рассматривать с разных сторон. В связи с повсеместным распространением интернет-технологий в маркетинге можно выделить отдельную часть: интернет-маркетинг (веб-маркетинг). В интернет-маркетинге важнейшим средством продвижения является интернет-сайт, посвященный товару или услуге, бренду или компании. Сайт является неотъемлемой частью брендбука упаковки, товара или компании. Именно поэтому в курсе «Веб-дизайн в маркетинге упаковки» рассматриваются технологии, позволяющие создать сайт, посвященный упаковке или товару.

Данный курс является продолжением курса «Веб-дизайн», в котором студенты направления 29.03.03. «Технология полиграфического и упаковочного производства» изучают основы создания дизайна сайта в графических редакторах. Курс «Веб-дизайн в маркетинге упаковки» рассматривает следующую ступень создания сайта – верстку шаблона, изучаются технологии верстки и основные навыки профессии – верстальщика. Исследуются основные принципы проектирования сайта с позиции маркетинга.

В данном учебном пособии рассматриваются важнейшие технологии, позволяющие создать сайт.

В первом разделе рассматривается HTML – язык разметки гипертекста. С помощью этого языка создано большинство сайтов в интернете. HTML позволяет перевести дизайн сайта из графического файла в файл формата html, который просматривается с помощью браузеров. Полученные файлы будут содержать основные структурные части дизайна, но без графического оформления. В разделе изучаются элементы и конструкции HTML.

Второй раздел посвящен технологии CSS – каскадным листам стилей. Они позволяют графически оформить страницы в формате html. Применение данной технологии позволяет полностью перевести дизайн сайта из графического файла.

В третьем разделе приведены основные термины и способы, необходимые для размещения сайта в сети интернет. Рассмотрены виды хостинга и рекомендации по его выбору.

Умения и навыки, полученные в результате изучения курса, а также умения и навыки, полученные в ходе изучения дисциплин «Веб-дизайн» и «Маркетинг», позволят создать сайт об упаковке или товаре, для которого проектируется упаковка, в ходе дипломной работы. Данный сайт может быть включен в брендбук, создаваемый студентами в рамках выпускной квалификационной работы.

ВВЕДЕНИЕ

В настоящее время существует огромное количество разнообразных сайтов. Классифицировать их можно различными способами. Приведем классификацию на основе размера и назначения сайта:

— одностраничный сайт (лэндинг пэйдж);

— сайт – визитка;

— корпоративный сайт;

— интернет магазин;

— порталы.

Чем больше ресурс, тем, соответственно, сложнее его разработка. Но этапы разработки будут всегда одинаковы: разработка технического задания, веб-дизайн, верстка, разработка и сопровождение сайта. Каждый из этапов может выполняться отдельным специалистом (специалистами) и, по сути, требует отдельную профессию (веб-дизайнер, верстальщик, программист, администратор, seo-специалист и т.д.).

Данное учебное пособие посвящено третьему этапу создания сайта, а именно – верстке HTML-шаблона сайта.

Верстка веб-страниц (шаблона) – процесс формирования веб-страниц (шаблона) в текстовом редакторе, либо веб-редакторе, следующий этап после веб-дизайна; а также результат этого процесса, то есть собственно веб-страницы.

На данный момент выделяют несколько типов верстки, возникших в хронологическом порядке.

Табличная верстка ранее была основным методом верстки, но и сейчас широко применяются в самых разных случаях. С помощью таблиц делают рамки, задают модульные сетки, создают цветной фон, выравнивают элементы и т. д.

Блочная верстка. Верстка при помощи блоков (тэг ** **) и описывающих их таблиц стилей (CSS). Реализует концепцию семантичной верстки.

До недавнего времени в качестве основных инструментов верстки выступали таблицы и фреймы (фреймы, ввиду их некоторых проблем, уходят в прошлое: например, стандарт HTML 5 более не включает в себя поддержку фреймов). В настоящее

время блочная используется гораздо шире. Однако табличная верстка в исполнении гораздо проще блочной верстки.

Все сайты, по макету верстки, можно разделить на три принципиальные группы:

— жестко фиксированные (Rigid fixed);

— адаптивные резиновые (Adaptable fluid);

— расширяемые эластичные (Expandable elastic) макеты.

Фиксированный тип макета – дизайн (табличный либо блочный), в котором ширина столбца/рисунка задана в пикселях, т.е. указана точно.

Резиновый тип макета – дизайн, в котором ширина столбца/рисунка задана в процентах от текущего разрешения экрана.

Адаптивная верстка/тип макета – дизайн, который подстраивается (адаптируется) под размер экрана, в том числе может происходить перестройка блоков с одного места на другое, или их замена блоками, отображаемыми только при определенном разрешении. Адаптивная верстка убрала необходимость специальных мобильных версий сайта, размещенных на отдельных поддоменах (например, m.wikipedia.org).

Преимущества и недостатки основных видов дизайна приведены в табл. 1.1.

Каждый вид дизайна имеет свои сильные и слабые стороны, поэтому выбор определенного дизайна целиком зависит от решаемой задачи. Также имеет место смешанный дизайн (например, некоторые столбцы табличного дизайна задать в процентах, а другие в пикселях).

Специализировано версткой веб-страниц занимаются верстальщики. В общем случае в задачу верстальщика входит:

— создание кода веб-страницы с помощью соответствующего языка разметки (HTML, XHTML, XML);

— оформление ранее созданного кода страницы с помощью встроенных средств языка разметки, либо же с помощью каскадных таблиц стилей CSS.

В ходе работы верстальщик обязательно использует следующее программное обеспечение:

— текстовый редактор или редактор HTML для написания и редактирования кода страницы;

— графическая программа для так называемой «нарезки» графического макета, полученного верстальщиком от веб-дизайнера.

Таблица 1.1

Виды дизайна и их характеристики

Название	Преимущества	Недостатки
Фиксированный дизайн	Такой дизайн гораздо легче разрабатывать, можно заранее предугадать, как будет выглядеть сайт. У дизайнера есть возможность следить за размером строки.	На различных разрешениях может появиться горизонтальный скроллинг, у сайта существует только одно оптимальное разрешение экрана
Резиновый дизайн	Сайт будет заполнять все пространство браузера, что значительно улучшает его вид. Сайт с таким дизайном будет одинаково смотреться на разных разрешениях.	Разработка такого дизайна сплошной эксперимент, не знаешь, как поведет себя вся структура сайта в тот или иной момент. На больших разрешениях возможно появление слишком длинных строк, что значительно утруждает читателя.
Адаптивная верстка	Сайт будет отображаться при разных размерах экрана так, как это наиболее удобно пользователю	Требует тщательной проработки нескольких макетов, для различных размеров экранов.

В некоторых случаях также могут быть использованы:

— WYSIWYG редакторы, в которых пользователь располагает все элементы, которые должны были быть получены с помощью HTML, используя графический интерфейс пользователя. После чего программа преобразует визуальное представ-

ление в HTML код. В данном случае автору не обязательно обладать исчерпывающим знанием HTML;

— программы автоматической верстки сайтов, которые извлекают слои с изображениями и текстом из дизайн-макета, созданного в графическом редакторе Adobe Photoshop или ему подобном, и формируют из этих слоев HTML код. Таким образом, создается базовый каркас веб страницы, готовый к последующей доработке.

Результатом работы верстальщика является HTML-шаблон веб-страницы, который передается программисту для дальнейшей работы над сайтом. К верстке HTML-шаблона выдвигается ряд требований.

Во-первых, верстка должна обладать *валидностью* – это ее соответствие стандартам организации The World Wide Web Consortium (W3C). Отсутствие ошибок в верстке документа – один из основных показателей качества верстки. Автоматическая проверка верстки на ошибки может быть проведена как с помощью онлайн сервиса W3C, так и различными программами-»валидаторами». Разные версии спецификации HTML предполагают различный синтаксис, поэтому тест верстки на валидность должен обязательно учитывать это.

Во-вторых, *кроссбраузерность* – близкое к исходному дизайнерскому и функциональному виду отображение параметров страниц при использовании разных браузеров и их различных версий и модификаций. Веб-разработчики принимают всевозможные усилия по унификации гипертекстовой разметки, направленные на разработку единого стандарта отображения в браузерах, но реализация этого – сложный технологический процесс из-за ряда конфронтаций среди разработчиков.

Таким образом, верстка HTML-шаблона – это сложный специфический процесс, требующий отдельной специализации. Основой этой специализации является язык разметки гипертекста – HTML, с изучения которого необходимо начать.

1. HTML

HTML можно назвать одним из основ всей сети интернет. HTML расшифровывается как HyperText Markup Language (в переводе означает Язык Разметки ГиперТекста). HTML используется для создания веб-страниц. HTML достаточно легок в изучении.

Для изучения HTML не требуется какое-то дополнительное программное обеспечение. Начать работать можно в Блокноте, стандартной программе, включаемой в любую версию Windows.

Любой HTML документ состоит из обычного текста и разметочных тэгов. Текст, как и в любых других файлах, используется для передачи информации, а разметочные тэги используются для ее группировки и оформления. Разметочные тэги HTML представляют собой специальные слова, которые окружены с обеих сторон угловыми скобками, например, **<html>**.

HTML тэги обычно используются в паре, например, ****. Первый тэг называется начальным тэгом, а второй конечным тэгом. Текст, находящийся между начальным и конечным тэгом подвергается «разметке». Например, ****Привет**** будет отображено браузером как *Привет* (слово 'Привет', написанное курсивом).

Также существуют непарные тэги, в этом случае символ «/» добавляется в конец тэга, перед угловой скобкой «>«, например, **
**

Любой HTML документ можно открыть в любом текстовом редакторе и любом браузере. Если HTML документ открывается в редакторе, то он будет отображен как обычный текстовый файл, если в браузере, то он будет отображен в соответствии с разметочными тэгами и будет называться веб-страницей. Главная цель любого веб-браузера (такого как Internet Explorer, Firefox, Chrome и т.д.) прочитать HTML документ и отобразить его как веб-страницу.

Чтобы создать HTML документ необходимо:

— открыть любой текстовый редактор (например, блокнот, встроенный в Windows);

— набрать произвольный текст и разметить его HTML тэгами;

— сохранить файл с расширением .htm или .html (команды Файл-Сохранить как…, далее необходимо в поле «Тип файла» выбрать «все файлы» и сохранить);

— все современные браузеры одинаково понимают .htm и .html, поэтому выбор расширения не имеет принципиального значения;

— созданный файл можно открыть с помощью веб-браузера, и он будет отображен как веб-страница.

Для редактирования HTML документов создано множество специальных редакторов, оснащенных специальными возможностями, позволяющими значительно упростить работу с ними (например, подсветка тэгов). К таким редакторам можно отнести:

— Notepad ++;

— Microsoft FrontPage;

— Adobe Dreamweaver.

Некоторые редакторы можно скачать в интернете в свободном доступе. Для данного курса рекомендуется использовать *Notepad ++*.

Каждый HTML документ начинается с тэга **<html>** (самый первый тэг в документе) и заканчивается парным тэгом **</html>** (самый последний тэг в документе). Это основополагающий тэг HTML документа, указывающий браузеру, что дальше идет код в формате HTML.

```
<html>
Другие тэги и текст документа
</html>
```

1.1.1. Основные тэги

Знакомство с тэгами лучше всего начать с основных тэгов, которые используются наиболее часто.

Любой HTML документ включает в себя *Заголовки*, которые определяются тэгами **<h1>-<h6>** (**h1** определяет самый крупный заголовок, а **h6** - самый мелкий).

```
<h1> Пример </h1>
```

Основной текст любой страницы оформляется в абзацы. Абзацы используются для логической группировки текста. Перед и после текста абзаца браузер автоматически отступает одну строку.

```
<p> Текст абзаца </p>
```

Основа всего HTML – это гиперссылка (ссылка), именно с помощью этого инструмента происходит связь между всеми интернет страницами. Ссылка определяется с помощью тэга **<a>.** Нажав на ссылку, пользователь перемещается на другой HTML документ, *url* (адрес страницы) которого указан в атрибуте ***href***.

```
<a href="yandex.ru"> ссылка на yandex.ru </a>
```

Также к основным тэгам можно отнести **<img/>**, с помощью которого можно вставить в HTML документ произвольное изображение. Ширина и высота картинки может задаваться с помощью атрибутов ***width*** и ***height***.

```
<img src="image1.jpg" width="70" height="50"/>
```

1.1.2. Элементы и их атрибуты

HTML *элементом* называется совокупность начального тэга, конечного тэга и содержимого (в случае непарного тэга – сам тэг). Большинство элементов могут иметь *атрибуты* – это своего рода свойства элемента, которые можно настраивать под выполняемые задачи.

Браузер верно отобразит HTML элемент даже если не указан конечный тэг. Это связано с тем, что HTML является языком «прощающим ошибки», однако, существует и более строгая

разновидность языка (XHTML), в которой пропуск конечного тэга будет считаться ошибкой.

Большинство элементов могут быть вложены друг в друга, как матрешки, т.е. в содержимом одного элемента может располагаться другой элемент.

Например, вложив элемент **** в элемент ****, текст будет одновременно жирным и курсивным. В XHTML элементы всегда должны быть вложены правильно. Неправильно: ****, правильно: ****, т.е. должна соблюдаться последовательность открытия и закрытия элементов. Нельзя закрывать внутренний элемент, пока не закрыт внешний. В HTML неправильно вложенные элементы не считаются ошибкой.

Элементы, которые не имеют содержимое, называются пустыми элементами, например, **
, **<hr/>, являются пустыми элементами.

HTML не чувствителен к регистру, это значит, что тэг **** и тэг **<B>** будут отображаться одинаково, в XHTML тэги могут быть написаны только в нижней раскладке.

Несмотря на то, что HTML не имеет строгий синтаксис, рекомендуется использовать нижний регистр. Код, не следующий никаким правилам, неприятно читать и сложно понимать.

Все HTML элементы могут иметь атрибуты. Атрибуты задаются в начальном тэге элементов и состоят из имени и значения, например, в атрибуте ***href=«http://www.yandex.ru/» href*** является именем, а ***http://www.yandex.ru/*** значением.

Для некоторых элементов существуют обязательные атрибуты (как ***href*** – адрес ссылки для **<a>**), но большинство атрибутов являются необязательными и используются по необходимости.

В XHTML значение атрибута обязательно должно помещаться в кавычки, в HTML наличие кавычек не обязательно, но желательно. Значение атрибута может заключаться как в двойные («), так и в одинарные (') кавычки.

Если в значении атрибута имеется слово, заключенное в кавычки, необходимо использовать комбинацию из двойных и одинарных кавычек («*Александр 'Наше Все' Пушкин*»).

В HTML регистр, в котором написано имя атрибута и его значения, не важен (т.е. ***имя=значение*** равнозначно ***ИМЯ=ЗНАЧЕНИЕ***), в XHTML имя и значение атрибута всегда должно быть написано в нижнем регистре.

Список атрибутов, которые имеют практически все элементы, приведен в табл. 1.2.

Таблица 1.2

Наиболее распространенные атрибуты

Атрибут	Описание
accesskey	определяет сочетание клавиш для доступа к данному элементу
class	определяет имя класса для элемента
id	определяет уникальный идентификатор для элемента
style	определяет стиль элемента
title	содержит дополнительную информацию об элементе (значение данного атрибута отображается при наведении курсора мыши на элемент)

1.2. Секция body

Любой HTML документ делится на секции. Секция – это парный HTML тэг, который содержит внутри себя большое количество других тэгов, объединенных функционально.

Тэг **<body>** является секцией, он содержит в себе все элементы, которые отображаются на странице в браузере, т.е. видимые части HTML документа. По-другому секция **<body>** называется телом документа.

В свою очередь секция **<head>** содержит в себе все элементы, которые не отображаются на странице в браузере (служебная информация, скрипты, стили и т.д.).

Также существуют межсекционные элементы, о которых будет написано позже.

Секция **<body>** является важнейшим структурным элементом и должна присутствовать во всех HTML документах.

1.2.1. Заголовки

При заходе на страницу, первым делом, пользователи зачастую пробегают глазами ее содержимое, проверяя, содержит ли она интересующую их информацию. Обычно, заголовки это первое (и часто единственное) на что они обращают внимание, поэтому неправильное использование заголовков может привести к потере посетителей.

Заголовок всегда должен кратко и точно описывать содержание, которое он озаглавливает (т.е. отрывок текста, описывающий леса России, должен быть озаглавлен как «*Леса России*»).

Наиболее важная информация страницы должна располагаться под заголовками большего размера, а наименее важная под заголовками меньшего размера.

Поисковые системы используют заголовки для индексации структуры и содержимого веб-страниц, поэтому, если есть желание, чтобы страница о «*Горнолыжном курорте в Сибири*» выводилась в поисковой системе по соответствующему запросу, необходимо использовать подходящие заголовки.

Заголовки в HTML определяются тэгами **<h1>-<h6>**. **<h1>** определяет заголовок наибольшего размера, а **<h6>** наименьшего.

```
<h1> Это самый большой заголовок </h1>
<h3> Это заголовок поменьше </h3>
<h6> Это самый маленький заголовок </h6>
```

Использовать данные тэги нужно только для создания заголовков. Никогда не надо пользоваться данными тэгами для создания БОЛЬШОГО или **жирного** текста.

1.2.2. Форматирование текста

Необходимо всегда визуально отделять не связанные разделы страницы друг от друга. Одним из вариантов такого отделения может являться вставка горизонтальной линии с помощью HTML тэга **<hr />**.

При создании веб-страниц желательно оставлять пояснительные заметки с помощью комментариев. Комментарии полностью игнорируются браузером и не отображаются при просмотре страницы.

<!– это комментарий -->

Комментарии могут использоваться при отладке кода, если нет уверенности в необходимости видеть данный заголовок, абзац и т.д. В итоговой версии страницы можно обернуть код в тэг **<!-- -- >** на время принятия решения.

Комментарии также полезны при разборе чужого кода. Поскольку над созданием сайта зачастую работают несколько человек, то каждый из них может оставить или прочитать необходимую служебную информацию с помощью комментариев.

Содержимое любых страниц в интернете может быть просмотрено. Исследование кода чужих страниц помогает быстрее освоить HTML.

Для того чтобы посмотреть исходный код, необходимо при просмотре страницы нажать правую кнопку мыши и выбрать пункт «*Просмотреть исходный код страницы*» (текст может незначительно отличаться в различных браузерах).

Большие объемы сплошного текста сложно читать, надо всегда разбивать сплошной текст на абзацы. Для создания абзацев используется тэг ** **. Перед и после данного тэга браузеры автоматически отступают одну строку.

 Пример абзаца 1

 Пример абзаца 2

С помощью тэгов форматирования можно выделять «важный» текст на страницах, создавать тексты «рукописного начертания», добавлять в HTML документы формулы, а также увеличивать и уменьшать размер шрифта.

Среди тэгов форматирования имеются тэги одинаковые по видимому результату, но разные по функциональному значению. Например, **** и **** задумывались как специальные тэги для выделения «важного» содержимого, а тэги **<b>** и **<i>** просто для создания жирного и курсивного текста. Несмотря на то, что браузеры отображают их одинаково, поисковые си-

стемы могут придавать им различное значение при анализе страницы.

Тэг **
** используется для перевода текста на новую строку.

Тэг **<pre>** отображает предформатированный текст. Все, что находится внутри тэга **<pre>,** будет отображено точно так, как написано. Браузер не будет удалять идущие подряд пробелы и символы перевода строки.

1.2.3. Стили

Вместе с выходом HTML версии 4 был опубликован новый инструмент – CSS. CSS предоставляет более удобный способ оформления HTML документов, полную свободу при оформлении документов. CSS позволяет:

— устанавливать размер, начертание и цвет шрифта;

— изменять местоположение элементов;

— оформлять фон элементов;

— выравнивать текст;

— оформлять таблицы, списки и многое другое.

CSS представляет из себя набор *стилей*, которые задают оформление элементов.

```
<p style='font-size:30px; display:inline; color:white;'> Это абзац, оформленный с помощью CSS </p>
<p style='display:inline; color:white; font-family:Verdana; border-style:solid;'> Это еще один абзац, оформленный с помощью CSS </p>
```

После введения CSS некоторые тэги и атрибуты в HTML считаются устаревшими. Они не будут поддерживаться в будущих версиях HTML и XHTML. Список тэгов и атрибутов, которых необходимо избегать приведен в табл. 1.3.

Таблица 1.3

Устаревшие тэги и атрибуты

Атрибуты	Описание
color	Определяет цвет текста
bgcolor	Определяет цвет фона
align	Определяет выравнивание текста
Тэги	**Описание**
<strike>	Определяет зачеркнутый текст
<font> и **<basefont>**	Устанавливает шрифт для текста
** **	Выравнивает содержимое по центру
<menu>	Определяет список меню

1.2.4. Гиперссылки

Гиперссылки являются основой HTML и всемирной паутины. Они представляют собой текст (или картинку), нажав на который, пользователь будет перемещен на другой документ или другое место в данном документе.

Как правило, гиперссылки выделяются на страницах подчеркнутым шрифтом синего цвета.

Различают два вида гиперссылок:

— *внешние гиперссылки* перемещают пользователя, нажавшего на них, на другой HTML документ;

— *внутренние гиперссылки* перемещают пользователя на предварительно созданную закладку в текущем документе.

Термины «гиперссылка» и «ссылка» являются синонимами и далее могут употребляться оба.

Пример внешней гиперссылки:

```
<a href=«адрес»>Текст ссылки </a>
```

В примере «адрес» указывает url адрес документа, на который будет перемещен пользователь после нажатия на ссылку.

Пример внутренней гиперссылки:

```
<!-- Создание гиперссылки на закладку -->
<a href=«#bookmark»>Текст ссылки </a>
 <!-- Создание закладки -->
 <a id=«bookmark»>Текст закладки. </a>
```

В примере ***«bookmark»*** – имя закладки (может быть произвольным). Внутренняя гиперссылка состоит из двух частей: ссылки – объекта, на который нажимает пользователь и закладки – места в документе, куда перемещается пользователь.

Примеры:

Внешние

```
<a href=«http://www.yandex.ru» target=”top”> www.yandex.ru </a>
<a href=«http://www.yandex.ru» target=”_blank”> <img src= 'image1.jpg' width='200' height='60' /> </a>
```

Внутренние

```
<a href='#chapter3'>Перейти на главу 3</a>
<a href='#chapter5'>Перейти на главу 5</a>
```

Атрибут ***target*** регулирует способ, каким будет открываться гиперссылка. Данный атрибут может иметь следующие значения:

— ***top*** документ откроется в текущем окне браузера;

— ***_blank*** документ откроется в новом окне браузера;

— ***_self*** документ откроется в текущем фрейме;

— ***_parent*** – документ откроется в родительском фрейме.

1.2.5. Изображения

Полноценный HTML документ получается путем комбинирования изображений и обычного текста. Таким образом, информация представляется пользователям максимально эффективно.

Для добавления в HTML документ изображений используется тэг **<img/>**. Атрибут ***src*** тэга **<img/>** содержит адрес, по которому находится изображение, которое должно быть отображено.

Загрузка изображений занимает время, так что прежде чем размещать на одной странице большое количество изображений, нужно быть уверенным, что оно необходимо.

```
<img src=«image1.jpg» />
```

В атрибуте ***alt*** задается альтернативный текст, который будет отображен, если браузер пользователя не сможет по каким-либо причинам отобразить изображение.

Роботы поисковых систем, которые анализируют страницы, не могут просматривать изображения, поэтому единственный способ указать им о содержимом изображения - написать об этом в атрибуте ***alt***.

```
<img src=«boat.gif» alt=«На картинке изображена плывущая лодка» />
```

1.2.6. Карты

HTML позволяет создавать несколько ссылок в одном изображении, где некоторые области изображения отвечают за определенные ссылки, такие конструкции называются *картами*.

Карты в HTML создаются с помощью тэга **<map>**, а области-ссылки в них с помощью тэга **<area>**. Форма и координаты областей-ссылок задаются с помощью атрибутов ***shape*** и ***coords*** тэга **<area>**, а место, на которое они ссылаются, с помощью атрибута ***href***.

В качестве ссылок может использоваться несколько форм, они перечислены в табл. 1.4.

Таблица 1.4

Формы, используемые в картах

Форма	Атрибуты
Прямоугольная область-ссылка	***shape=«rect»***, ***coords=«x1,y1,x2,y2»***, где x1,y1 координаты верхнего левого угла прямоугольной области, а x2,y2 координаты нижнего правого угла области.
Круглая область-ссылка	***shape=«circle»***, ***coords=«x,y,радиус»***, где x,y координаты центра области.
Область-ссылка многоугольник	***shape=«poly»***, ***coords=«x1,y1,x2,y2..xz,yz»***, где x1,y2 координаты первого угла многоугольника, x2,y2 второго и т.д. Если координаты первого и последнего угла не совпадают, браузер автоматически добавит координату последнего угла, чтобы завершить многоугольник.

После создания карты необходимо привязать ее к какому-нибудь изображению. Для этого необходимо добавить к изображению атрибут ***usemap=«#имя_карты»***, а к карте добавить атрибут ***name=«имя_карты»***.

```
<img src=«figures.gif» usemap=«#figuremap» />
<map name=«figuremap»>
<area shape=«rect» coords=«3,8,72,75»
href=«bluesquare.html» target=«_blank» />
<area shape=«circle» coords=«125,50,38»
href=«greencircle.html» target=«_blank» />
<area shape='poly'
coords='180,27,210,6,248,30,233,74,190,72'
href=«redpolygon.html» target=«_blank» /> </map>
```

1.2.7. Таблицы

Для создания таблиц в HTML используется тэг **<table>**. Тэг **<tr>** создает в таблице строку, а тэг **<td>** добавляет в эту строку ячейку с данными.

По умолчанию таблицы отображаются без границ. Для того чтобы отобразить границу, необходимо использовать атрибут ***border=«1»***, где «*1*» – ширина границы в пикселях.

В ячейках таблицы может располагаться не только обычный текст, но и картинки, ссылки и другие таблицы.

```
<table border=«1»>
<tr>
<td>Россия</td> <td>Москва</td>
</tr><tr>
<td>США</td> <td>Вашингтон</td>
</tr><tr>
<td>Китай</td> <td>Пекин</td>
</tr><tr>
<td>Великобритания</td> <td>Лондон</td>
</tr>
</table>
```

Табличный заголовок создается с помощью тэга **<th>**. Текст этого элемента центрируется и выделяется жирным шрифтом.

Ячейки могут растягиваться на несколько столбцов, с помощью атрибута ***colspan***.

```
<td colspan=«2»> Ячейка, растянутая на 2 столбца </td>
```

С помощью атрибута ***rowspan*** можно растянуть ячейку на несколько строк.

```
<td rowspan=«2»> Ячейка, растянутая на две строки </td>
```

1.2.8. Списки

Упорядочивание однородной информации происходит за счет создания списков. Различают два вида списков: упорядоченные (немаркированные) и неупорядоченные (маркированные).

Неупорядоченные списки определяются тэгом **<ul>**, а упорядоченные тэгом **<ol>**. Элементы (пункты) в списках определяются с помощью тэга **<li>**.

Пункты неупорядоченного списка отмечаются слева маленьким черным кружочком (маркером), а пункты упорядоченного списка нумеруются.

```
<p>Упорядоченный список: </p>
<ol>
<li>Меркурий</li>
<li>Венера</li>
<li>Земля</li>
</ol>
<p>Неупорядоченный список: </p>
<ul>
<li>Зебра</li>
<li>Слон</li>
<li>Жираф</li>
</ul>
```

Списки, как и таблицы, могут быть вложенными.

```
<p> Крупнейшие города различных стран: </p>
```

```
<ol>
<li>Россия
<ol><li>Москва</li><li>Санкт-Петербург</li></ol>
</li>
<li>США
<ol> <li>Нью-Йорк</li> <li>Лос-Анджелес</li> </ol>
</li>
</ol>
```

Пункты упорядоченных и неупорядоченных списков помимо обычного текста могут также содержать изображения и ссылки.

1.2.9. Формы

Для обратной связи с посетителями сайта в HTML предусмотрены *формы*. HTML формы могут содержать такие элементы ввода как:

- текстовые поля;
- флажки;
- радио-кнопки;
- кнопки отправления и др.

Формы также могут содержать списки выбора, многострочные текстовые поля, метки и др.

Для того чтобы создать форму в HTML используется элемент **<form>**.

```
<form>
<p>Введите ваше имя:</p>
<input type=«text» />
<p>Введите пароль: </p>
<input type=«password» />
</form>
```

Элементы ввода используются для приема пользовательских данных. Элементы ввода сильно отличаются друг от друга в зависимости от значения атрибута ***type***.

Самым распространенным элементом является однострочное текстовое поле, в которое пользователь может вводить различную информацию **<input type=«text» />**. По умолчанию текстовое поле вмещает 20 знаков.

```
<form>
<p> Введите ФИО в поля ниже: </p>
<br />
Имя: <input type=«text» name=«firstname» /><br />
Фамилия: <input type=«text» name=«lastname» /><br />
Отчество: <input type=«text» name=«lastname» />
</form>
```

Поле пароля **<input type=«password» />** определяет поле для ввода пароля. Содержимое, вводимое в данное поле, закрывается черными кружками, позволяя вводить пароли даже в присутствии посторонних.

```
<form> Введите пароль:
<input type=«password» name=«pass» />
</form>
```

Флажки **<input type=«checkbox» />** позволяют пользователям выбирать несколько пунктов с предварительно заполненной информацией из группы.

```
<form>
<p> Как вы относитесь к полетам в космос? </p>
<input type=«checkbox» name=«space» value=«1» /> Положительно, всегда хотел полететь в космос<br />
<input type=«checkbox» name=«space» value=«2» /> Безразлично, никогда не думал об этом серьезно <br />
<input type=«checkbox» name=«space» value=«3» /> Отрицательно, меня с детства отталкивают мысли о космосе <br />
</form>
```

Для группировки нескольких флажков в одну группу используется атрибут ***name***, который должен иметь одинаковое значение для всех элементов группы. В свою очередь, уникальные значения в пределах одной группы должны иметь атрибуты ***value***.

Элемент **<input type=«radio» />** определяет радио-кнопку. Радио-кнопки позволяют пользователям выбрать только один пункт с предварительно заполненной информацией из группы.

Радио-кнопки называются так, потому что их поведение напоминает поведение кнопок старых радио, у которых только одна кнопка могла быть включена одновременно.

```
<form>
<p> Укажите Ваш пол: </p>
<input type=«radio» name=«s» value=«m» /> Мужской<br />
<input type=«radio» name=«s» value=«f» /> Женский
</form>
```

У радио-кнопок атрибут ***name*** также должен иметь одинаковое, а атрибут ***value*** уникальное значение в пределах одной группы.

Элемент **<input type=«submit» />** определяет кнопку отправления. После нажатия на данную кнопку данные, введенные пользователем, будут отправлены на сервер.

Адрес, на который будут пересылаться данные формы, указывается в атрибуте тэга **form** - ***action***. Если данный атрибут отсутствует, данные будут отправлены на текущую страницу.

Обработка данных, полученных сервером в результате отправки форм, будет производиться с помощью серверных языков (таких как PHP, Ruby или Python) и поэтому в данном пособии не рассматривается.

```
<form name=«input» action=«form.php» method=«get»>
Введите Ваше имя:
<input type=«text» name=«name» />
<input type=«submit» value=«Отправить» />
</form>
```

Для создания выпадающих списков используется тэг **<select>**, а элементы выпадающего списка определяются с помощью тэга **<option>**.

Следует учитывать, что пользователи не любят вводить данные вручную, поэтому при возможности следует заменять обычное текстовое поле выпадающим списком, радио-кнопками или флажками.

```
<p> Выберите ваш пол </p>
```

```
<form>
<select name=«sex» >
<option value=«m»> Мужской </option>
<option value=«f»> Женский </option>
</select>
</form>
```

С помощью атрибута ***multiple*** можно указать, что в выпадающем списке могут быть выбраны одновременно несколько элементов.

```
<p> В данном списке может быть одновременно выбрано несколько значений (для этого зажмите клавишу Ctrl и щелкайте на необходимые элементы): </p>
<form>
<select name='city' multiple='multiple'>
<option value='london'>Лондон</option>
<option value='moscow'>Москва</option>
<option value='newyork'>Нью Йорк</option>
</select>
</form>
```

Для того чтобы озаглавить группу элементов формы, необходимо с помощью тэга **<fieldset>** сгруппировать желаемую часть формы и затем с помощью тэга **<legend>** установить желаемое заглавие.

```
<form> <fieldset>
<legend>Данные о пользователе</legend>
Имя: <br /><input type=«text» name=«firstname» /><br />
Фамилия:<br /><input type=«text» name=«lastname»/><br />
Отчество:<br /><input type=«text» name=«lastname»/><br />
<p>Укажите Ваш пол:</p>
<input type=«radio» name=«s» value=«m» /> Мужской<br />
<input type=«radio» name=«s» value=«f» /> Женский
</fieldset>
Элементы ввода
<fieldset> <legend> Анкета: </legend>
```

Как Вы относитесь к полетам в космос:
<input type=«checkbox» name=«space» value=«1» /> Положительно, всегда хотел полететь в космос
<input type=«checkbox» name=«space» value=«2» /> Безразлично, никогда не думал об этом серьезно
<input type=«checkbox» name=«space» value=«3» /> Отрицательно, меня с детства отталкивают мысли о космосе
</fieldset> </form>

1.2.10. Спецсимволы

Некоторые символы невозможно отобразить в HTML документах обычным образом, они называются спецсимволами. Спецсимвол можно отобразить, только используя соответствующую ему мнемонику или специальный код.

Синтаксис:

&мнемоника_символа;

<!-- или -->

&#код_символа;

Самый часто используемый спецсимвол - это *неразрывный пробел* (его мнемоника *** ***).

Данный символ используется для контролирования переноса строки (после данного символа автоматический перевод строки невозможен) и для вставки в текст подряд идущих пробелов (по умолчанию, при вставке в текст пяти подряд идущих пробелов, браузер вырежет четыре из них и отобразит лишь один).

Текст ниже содержит шесть пробелов:

Первое слово   второе слово

Также в HTML коде часто используются спецсимволы < (мнемоника ***<***) и > (мнемоника ***>***). Данные символы невозможно напрямую использовать в коде, потому что браузер путает их с началами тэгов.

Текст ниже содержит HTML тэги:

<head>, <p>, <html>

В табл. 1.5 приведены наиболее часто встречающиеся спецсимволы.

Таблица 1.5

Часто встречающиеся спецсимволы

Символ	Описание	Мнемоника	Код
	Пробел		
<	Меньше чем	&lt;	&#60;
>	Больше чем	&gt;	&#62;
©	Знак охраны авторского права	&copy;	&#169;
®	Зарегистрированный товарный знак	&reg;	&#174;
&	Знак амперсанда	&amp;	&#38;

1.3. Межсекционные элементы

1.3.1. Фреймы

Фреймы позволяют отображать в одном окне браузера несколько HTML документов одновременно. Каждый из документов будет находиться в своей области окна, в своем фрейме (рамке).

На данный момент фреймы считаются устаревшей технологией и не будут поддерживаться в HTML 5 (это касается лишь обычных фреймов, строковые фреймы не считаются устаревшими).

Фреймы являются межсекционными элементами и вкладываются только в элемент **<html>**.

С помощью HTML тэга **<frameset>** описывается количество и взаимное расположение фреймов в окне браузера.

Тэг **<frame>** описывает один отдельный фрейм. В его атрибуте ***src*** указываться адрес документа, который будет отображен в данном фрейме.

<html>

```
<frameset cols=«25%,50%,25%»>
<frame src=«frame_a.html» />
<frame src=«frame_b.html» />
<frame src=«frame_c.html» />
</frameset>
</html>
```

Размеры фреймов устанавливаются с помощью атрибутов тэга **<frameset>**. Размеры могут задаваться в пикселях (px) и процентах (%).

С помощью атрибута ***rows*** устанавливается высота фрейма. Размеры для отдельных фреймов должны отделяться запятой. С помощью этого атрибута фреймы принимают форму строк.

С помощью атрибута ***cols*** задается ширина отдельного фрейма. С помощью этого атрибута фреймы принимают форму колонок.

```
<frameset rows=«35%,35%,30%»>
<frameset cols=«25%,75%»>
```

Фрейм по умолчанию имеет видимую границу, поэтому пользователь может изменять его размер, перетаскивая ее. Чтобы запретить перетаскивание границ необходимо добавить атрибут ***noresize=«noresize»*** к тэгу **<frame>**.

Если браузер пользователя не поддерживает фреймы, то используется тэг **<noframes>**. Значение этого тэга будет отображено только тем пользователям, браузер которых не может отобразить фреймы.

При использовании тэга **noframe** указывается тэг **body**, в остальных случаях, при использовании фреймов, тэг **body** отсутствует.

```
<frameset cols=«25%,75%»>
<frame src=«frame_a.html» />
<frame src=«frame_b.html» />
</frameset>
<noframes>
<body><p> Ваш браузер не поддерживает фреймы. </p>
</body>
```

```
</noframes>
```

Тэг **<iframe>** позволяет вставлять фрейм в любое место обычного HTML документа. Данный тэг часто используют для отображения рекламы на сайтах. Ширина строкового фрейма задается с помощью атрибута ***width***, а высота - с помощью атрибута ***height***.

Если браузер пользователя не сможет отобразить строковый фрейм, то он просто его пропустит.

```
<iframe src='frame_a.html' width='400' height='150'>
</iframe>
<iframe src='frame_b.html' width='700' height='250'>
</iframe>
```

1.3.2. DTD

DTD – это не HTML тэг, а инструкция браузеру о версии языка разметки данной страницы. DTD указывается перед тэгом **<html>**. DTD расшифровывается как Document Type Definition (объявление типа документа).

DTD нужен для того, чтобы браузер правильно понимал, как отображать HTML документ. В HTML существует несколько видов DTD.

HTML 4.01 Strict (строгий) – HTML документы со строгим DTD могут содержать все HTML элементы и атрибуты, кроме презентационных и устаревших. Использование фреймов запрещено.

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

Вместо презентационных тэгов на данный момент используются CSS.

HTML 4.01 Transitional (переходный) – HTML документы с переходным DTD могут содержать все HTML элементы и атрибуты, включая презентационные и устаревшие. Использование фреймов запрещено.

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

HTML 4.01 Frameset (фреймовый) – HTML документы с фреймовым DTD могут содержать все HTML элементы и атрибуты, включая презентационные и устаревшие. Использование фреймов разрешено.

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

HTML 5 Doctype - в HTML 5 вместо трех различных Doctype введен один универсальный.

<!DOCTYPE html>

Для того чтобы узнать, соответствует ли код на странице выбранному DTD, необходимо проверить страницу на программе-валидаторе. Такие программы проверяют *валидность* (т.е. корректность кода). Одну из таких программ можно найти по адресу: ***http://validator.w3.org/***.

1.4. Секция head

В секции **head** могут располагаться скрипты, инструкции об оформлении страницы и различная мета-информация о данном HTML документе.

Элементы, располагающиеся в секции head, не отображаются явно на странице и используются для служебных функций.

Следующие элементы должны обязательно располагаться в секции **head**: **<link>**, **<title>**, **<base>**, **<style>**, **<script>** и **<meta>**.

1.4.1. Заголовок документа

С помощью элемента **<title>** задается заголовок HTML документа. Элемент **title** обязательно должен присутствовать во всех HTML документах.

Элемент **title** обладает следующими свойствами:

— определяет заголовок окна браузера;

— используется как заголовок страницы в результатах выдачи поисковых систем;

— используется как заголовок страницы при добавлении сайта в избранное.

```
<head>
<title>
Главная страница сайта
</title>
</head>
```

1.4.2. Метаданные

Метаданные – это информация о данных, находящихся в HTML документе. Пример метаданных: кодировка страницы, краткое описание содержимого, ключевые слова, имя автора, дата последней модификации.

Метаданные не отображаются явно на странице, но активно используются браузерами и поисковыми системами.

Метаданные HTML документов определяются с помощью тэга **<meta>**. Тэг **<meta>** всегда должен располагаться в секции **head**.

Кодировка

Для того чтобы указать браузеру пользователя какая кодировка используется на данной странице, необходимо использовать атрибут ***charset*** тэга **meta**.

Если явно не указать кодировку, браузер при отображении страницы будет определять ее автоматически. Если кодировка при этом будет определена не верно, пользователь увидит страницу, содержащую бессмысленные символы, поэтому кодировка обязательно должна указываться к каждому HTML документу.

```
<!-- Текст данного HTML документа набран в типичной для
windows кириллической кодировке windows-1251 -->
<meta http-equiv=«Content-Type» content=«text/html; char-
set=windows-1251»>
```

```
<!-- Текст данного HTML документа набран во всемирной кодировке UTF-8 -->
<meta http-equiv=«Content-Type» content=«text/html; charset=UTF-8»>
```

Мета элементы и поисковые системы

Большинство поисковых систем, во время индексации страницы, обращаются к мета элементам. Это обстоятельство следует учитывать при вводе метаданных.

Мета элемент с атрибутом ***name=”description”*** определяет описание для HTML документа (данное описание может использоваться поисковыми системами при отображении документа в поисковой выдаче):

```
<meta name=«description» content=“изучение HTML» />
```

Мета элемент с атрибутом ***name=”keywords”*** задает ключевые слова для документа:

```
<meta name=«keywords» content=«HTML, ТД, КемТИПП» />
```

В связи с тем, что многие веб-мастера для поднятия рейтинга выдачи сайта указывали огромное количество ключевых слов, не имеющих отношения к их сайту, но пользующихся популярностью, поисковые системы были вынуждены сократить до минимума использование ***keywords*** для определения ключевых слов страницы.

1.4.3. Связь с внешними файлами

С помощью тэга **<link>** можно связать HTML документ с внешним CSS файлом.

```
<head>
<link rel=«stylesheet» type=«text/css» href=«style.css» />
</head>
```

1.4.4. Корневой адрес

С помощью тэга **<base>** задается корневой адрес (адрес, который будет подставляться к относительным ссылкам) и способ открытия ссылок по умолчанию.

Атрибут ***href*** используется для задания корневого адреса по умолчанию. В большинстве случаев, корневой адрес должен иметь значение доменного имени сайта.

С помощью атрибута ***target*** указывается способ открытия ссылок. По умолчанию ссылки открываются в том же окне браузера.

1.4.5. Стили

Элемент **<style>** используется для оформления HTML документов. Подробное описание стилей приведено в следующем разделе учебного пособия.

```
<html>
<head>
<style type=«text/css»>
body {background-color:yellow}
p {color:blue}
</style>
</head>
<body>
<p> Это абзац синего цвета. </p>
</body>
</html>
```

В HTML5 атрибут ***type*** тэга **style** стал необязательным.

1.4.6. Скрипты

HTML тэг **<script>** используется для определения скриптов, выполняющихся на стороне клиента. Большинство скриптов создаются с помощью языка программирования JavaScript.

Он является самым популярным языком программирования этого типа.

JavaScript используется для создания веб-приложений и динамических сайтов, способных взаимодействовать с пользователем.

Тэг **script** может либо содержать код скрипта, либо ссылаться на внешний скриптовый файл через атрибут ***src***.

```
<head>
<script type=«text/javascript»>
function showMessage() { alert('Вы нажали на кнопку.') }
</script>
</head>
<body>
<p>Нажмите на кнопку ниже, чтобы заданный скрипт выполнился. </p>
<input type='button' value='Нажмите на меня' onclick='showMessage()' />
</body>
<script type=«text/javascript» src=«hello.js»></script>
```

Тэг **<noscript>** используется для отображения альтернативного текста пользователям, браузер которых не поддерживает JavaScript.

```
<script type=«text/javascript»>
document.write(«<p>Привет, привет!</p>«)
</script>
<noscript>
<p>Вы видите это сообщение, потому что Ваш браузер не поддерживает JavaScript</p>
</noscript>
```

1.5. Создание секций

Элемент ** ** позволяет создать секции внутри секции **<body>**. Данные секции используются для логической разметки HTML документа. К стандартным секциям относятся:

— *header* – заголовок HTML страницы, который обычно содержит название сайта, логотип и общие элементы сайта;

— *navigation* или *menu* – секция, включающая в себя меню сайта (навигацию). Зачастую выделяется основное меню и второстепенное меню;

— *content* – основное содержание страницы;

— *footer* – заключительная часть документа, содержит общую информацию, контактную информацию, дублирующее меню, сведения об авторских правах и т.п.

Примерная структура HTML-страницы представлена на рис. 1.1.

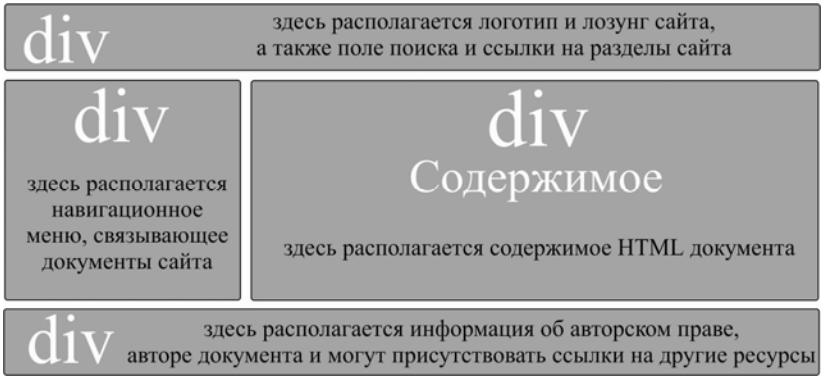

Рис. 1.1. Структура HTML-страницы

Оформление и размещение секции происходит с помощью CSS.

```
<div style=«color:green;font-size:1.2em;»>
 <p>Это оформляемый абзац</p>
 <h4> Это оформляемый заголовок </h4>
</div>
```

1.6. HTML 5

HTML 5 – это новая версия HTML. Предыдущая версия HTML была выпущена в 1999 году, за это время интернет сильно изменился, и поэтому изменения требовались и от HTML.

В HTML 5 добавлено много нового, поэтому некоторые возможности, которые были доступны в HTML 4 только с использованием внешних плагинов (таких как Adobe Flash) или клиентских скриптов теперь доступны с помощью обычных разметочных тэгов.

Разработка HTML 5 завершена в 2014 году, на данный момент его поддержку имеют все современные браузеры.

К наиболее интересным нововведениям HTML 5 относятся:

— поддержка видео и аудио (элементы **video** и **audio**);

— возможности рисования на веб-страницах произвольных объектов (элемент **canvas**);

— улучшение форм (новые значения ***type*** для **<input>** и множество новых элементов и атрибутов);

— добавление семантических тэгов, позволяющих сделать веб-страницы более понятными для поисковых систем, браузеров и других программ и устройств, читающих веб-страницы (элементы **footer, header, nav, article**);

— DOM хранилища - более функциональная альтернатива cookie.

В данном разделе будут рассмотрены некоторые из нововведений, которые являются наиболее интересными для рассматриваемого курса.

Все HTML 5 документы должны начинаться с объявления **DOCTYPE**. Предыдущие версии HTML имели несколько типов **DOCTYPE,** HTML 5 имеет только один:

```
<!DOCTYPE html>
```

Данное объявление переводит все браузеры в нормальный режим. Браузеры, не поддерживающие HTML 5, в данном режиме будут интерпретировать старые тэги и игнорировать новые, которые они не поддерживают.

1.6.1. Видео

Одна минута видео иногда может рассказать пользователям больше, чем десятки страниц текста. HTML 5 вводит специальный тэг **<video>**, позволяющий встраивать в веб-страницы видео файлы.

Поддержку видео имеют браузеры: Internet Explorer 9+, Opera 10.50+, Firefox 3.5+, Chrome 3.0+, Safari 3.1+.

Атрибут ***src*** тэга **<video>** позволяет указать путь к видео файлу, который будет воспроизведен. Атрибут ***controls*** отображает в плеере кнопки управления видео, а атрибуты ***height*** (высота) и ***width*** (ширина) задают размеры плеера.

```
<video src=«video1.mp4» width='300' height='200' controls='controls'></video>
```

Не все браузеры поддерживают определенные форматы видео, поэтому, для того чтобы видео нормально воспроизводилось во всех браузерах, необходимо добавлять источники в формате Ogg и MPEG4 (или WebM и MPEG4).

С помощью тэга **<source>** имеется возможность предоставить несколько источников видео в разных форматах для воспроизведения. Браузер будет использовать первый поддерживаемый им формат.

```
<video width=«300» height=«200» controls=«controls»>
<source src=«video1.ogv» type=«video/ogg» />
<source src=«video1.mp4» type=«video/mp4» />
<source src=«video1.webm» type=«video/webm» />
</video>
```

С помощью атрибута ***type*** необходимо указать *mime-тип* видео файла. Видео в формате ogg должно иметь тип ***'video/ogg'***, mp4 - ***'video/mp4'***, а webm - ***'video/webm'***.

Текст, помещенный в содержимое элемента **video**, будет отображен в случае, если браузер пользователя не поддерживает тэг видео.

```
<video src='video1.ogv' width=«300» height=«200» controls=«controls»>Ваш браузер не поддерживает элемент video.</video>
```

Для быстрой конвертации видео в необходимые форматы (Ogg, MPEG4 и WebM) можно использовать бесплатную программу Miro Video Converter.

1.6.2. Аудио

С помощью нового HTML 5 элемента **<audio>** можно добавить на веб-сайт музыкальный трек с помощью обычного разметочного тэга.

Поддержку аудио имеют следующие браузеры: Internet Explorer 9+, Opera 10.50+, Firefox 3.5+, Chrome 3.0+, Safari 3.0+.

Атрибут ***src*** тэга **<audio>** позволяет указать путь к аудио файлу, а атрибут ***controls*** добавляет кнопки управления.

```
<audio src=«track1.ogv» controls='controls'></audio>
```

Не все браузеры поддерживают определенные форматы аудио, поэтому, для того чтобы файл нормально проигрывался во всех браузерах, необходимо добавить источник в формате MP3.

С помощью элемента **<source>** можно предоставить несколько источников в разных форматах для воспроизведения. Браузер будет использовать первый, поддерживаемый им формат.

Текст, находящийся в содержимом элемента, будет отображен в случае, если браузер пользователя не поддерживает элемент **audio**.

```
<audio controls=«controls»>
<source src=«track1.ogv» type=«audio/ogg» />
<source src=«track1.mp3» type=«audio/mpeg» />
Данный текст будет выведен, если браузер пользователя
не поддерживает элемент audio.
</audio>
```

С помощью атрибута ***autoplay*** можно заставить трек автоматически проигрываться после загрузки страницы.

```
<audio autoplay='autoplay'>
<source src=«track1.ogv» type=«audio/ogg» />
<source src=«track1.mp3» type=«audio/mpeg» />
```

Данный текст будет выведен, если браузер пользователя не поддерживает элемент audio.
</audio>

С помощью представленной ранее программы Miro Video Converter можно также конвертировать в необходимые форматы (MP3 и Ogg) аудио файлы.

1.6.3. Семантические тэги

В HTML 4 благодаря использованию CSS необходимо было создавать страницы с хорошо понятной для пользователей визуальной структурой, но эти страницы были непонятны для поисковых систем и браузеров.

Например, поисковый робот не может отличить содержимое документа от навигационного меню, если они размечены с помощью одинаковых **div** элементов.

Для того чтобы разрешить эту проблему, в HTML 5 были введены семантические тэги. С помощью семантических тэгов можно сделать страницы сайтов более понятными для поисковых систем и браузеров.

В табл. 1.6 приведены новые семантические тэги.

Таблица 1.6

Семантические тэги

Тэги	Описание
<footer>	Определяет футер
<header>	Определяет заголовочный блок сайта
<nav>	Определяет навигационное меню

На рис. 1.2 представлена структура документа в HTML 5 с использованием новых тэгов разметки.

Группировка содержимого. С помощью тэга **<section>** можно группировать логически связанные элементы в документе.

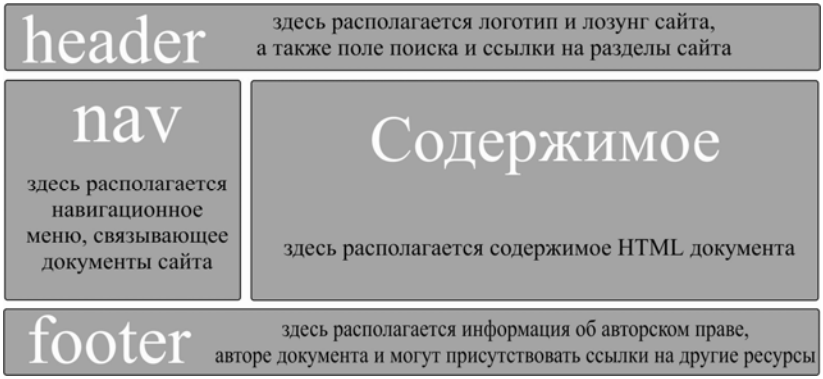

Рис. 1.2. Структура HTML-страницы в формате HTML 5

Если логически связанное содержимое является автономным (может использоваться в других документах независимо от остального содержимого на странице) необходимо использовать вместо **<section>** тэг **<article>.**

```
<article>
<h1>Титаник</h1>
<p>«Титаник» – британский пароход компании «Уайт Стар Лайн». Во время первого рейса 14 апреля 1912 года столкнулся с айсбергом и через 2 часа 40 минут затонул. На борту находилось 1316 пассажиров и 908 членов экипажа, всего 2224 человека. </p>
<section>
<h2>Постройка</h2>
<p>Заложен 31 марта 1909 года на верфях судостроительной компании «Харланд энд Вольф» в Куинс-Айленд (Белфаст, Северная Ирландия), спущен на воду 31 мая 1911 года, прошел ходовые испытания 2 апреля 1912 года.</p>
</section>
<section>
<h2>Местонахождение</h2>
```

```
<p>1 сентября 1985 года экспедиция под руководством директора Института океанологии города Вудс-Холл, штат Массачусетс, доктора Роберта Балларда (Robert D. Ballard) обнаружила место залегания «Титаника» на дне Атлантического океана на глубине 3750 метров.</p>
</section>
</article>
```

Предположим сайт, на котором была опубликована данная статья, содержит также несколько других статей. Так как текст этой статьи является автономным, т.е. можно понять его смысл, не читая остальные статьи, опубликованные на этом сайте, и даже без проблем опубликовать его в любом другом месте, выделяется статья о Титанике целиком тэгом **<article>**. Подразделы данной статьи, однако, не могут являться автономными, т.к. они логически связаны между собой и в данном случае уже необходимо использовать тэг **<section>**.

Выделение элементов. Тэг **aside** используется для выделения элементов, которые не являются частью содержимого, но косвенно с ним связаны. Данным тэгом могут выделяться: цитаты, дополнительная информация к статье, словарь с терминами, список ссылок и т.д.

```
<article>
<h1>Титаник</h1>
<p>«Титаник» – британский пароход компании «Уайт Стар Лайн». Во время первого рейса 14 апреля 1912 года столкнулся с айсбергом и через 2 часа 40 минут затонул. На борту находилось 1316 пассажиров и 908 членов экипажа, всего 2224 человека. </p>
<aside>Гибель Титаника - это одно из самых крупных кораблекрушений в истории человечества.</aside>
</article>
```

Выделение текста. С помощью тэга **<mark>** можно выделить «важную» часть в тексте.

```
<p>Я считаю, что <mark>HTML 5</mark> облегчает жизнь веб-разработчиков.</p>
```

Подписи для иллюстраций. В журналах и газетах иллюстрации часто сопровождаются подписями. В HTML 4 невоз-

можно было создавать подписи, не прибегая к использованию CSS. В HTML 5 это проблема решена добавлением новых тэгов: **<figure>** и **<figcaption>**.

```
<figure>
<img src='image1.jpg' width='300' height='230' />
<figcaption>Картинка с именем image1 </figcaption>
</figure>
```

1.6.4. Формы

В HTML 5 у элемента **input** появились **новые типы**. Использовать их нужно с осторожностью, т.к. старые версии браузеров новые типы не поддерживают.

input *type=email* определяет поле, которое должно содержать email адрес. Значение, введенное в поле, автоматически проверяется перед отправкой на сервер.

input *type=url* определяет поле, которое должно содержать url адрес. Значение, введенное в поле, автоматически проверяется перед отправкой на сервер.

input *type=tel* определяет поле для ввода телефонного номера. С помощью атрибута ***pattern*** можно установить формат принимаемого телефонного номера. Формат задается с помощью регулярных выражений.

input *type=number* определяет поле, которое должно содержать числа. Можно ограничивать диапазон принимаемых чисел с помощью атрибутов ***min*** (минимальное допустимое число) и ***max*** (максимальное допустимое число). С помощью атрибута ***step*** можно задать шаг допустимых чисел (к примеру, если шаг равен 2, то в поле могут вводиться числа 0,2,4,6 и т.д.)

input *type=range* определяет поле, которое может содержать значения в определенном интервале. Отображается как ползунок, который можно перетаскивать мышкой. Ограничивать диапазон принимаемых чисел можно с помощью атрибутов ***min*** (минимальное допустимое число) и ***max*** (максимальное допустимое число). С помощью атрибута ***step*** можно задать шаг допустимых чисел (к примеру, если шаг равен 2, то в поле могут вводиться числа 0,2,4,6 и т.д.)

input *type=search* определяет поле поиска (может использоваться, например, для создания поиска по сайту).

```
<form action='html5.php'>
Введите email: <input name='email' type='email' value='He email' />
<input type='submit' value='Отправить' />
</form><br />
<form action='html5.php'>
Введите url: <input name='url' type='url' value='He url' />
<input type='submit' value='Отправить' />
</form><br />
<form action='html5.php'>
Введите сотовый телефон (<i>в формате 8XXXXXXXXXX</i>):
<input name='tel1' type='tel' pattern='8[0-9]{10}'/>
<input type='submit' value='Отправить' />
</form><br />
<form action='html5.php'>
Введите домашний телефон (<i>в формате XXX-XX-XX или XX-XX-XX</i>):
<input name='tel2' type='tel' pattern='[0-9]{2,3}-[0-9]{2}-[0-9]{2}'/>
<input type='submit' value='Отправить' />
</form><br />
<form action='html5.php'>
Введите число: <input name='number' type='number' value='10' />
<input type='submit' value='Отправить' />
</form><br />
<form action='html5.php'>
Выберите число: <input name='range' type='range' min='1' max='5' />
<input type='submit' value='Отправить' />
</form><br />
<form action='html5.php'>
<input name='search' type='search' value='Поиск по сайту' />
```

```
<input type='submit' value='Отправить' />
</form><br />
<form action='html5.php'>
Выберите цвет: <input name='color' type='color' />
<input type='submit' value='Отправить' />
</form>
```

Новые элементы в формах. Помимо типов **input** у форм появились новые элементы.

datalist позволяет привязать список к полям формы. Значения списка будут выводиться как поисковые подсказки во время ввода информации в поле, связанное с ним.

keygen позволяет генерировать открытые и закрытые ключи, которые используются для безопасной связи с сервером.

output может использоваться для вывода различной информации. С помощью атрибута ***for*** можно указать связанные поля.

```
<form action='html5.php'>
Введите название города:
<input name=«city» list=«city» />
<datalist id=«city»>
<option label=«Москва» value=«Москва, Россия» />
<option label=«Санкт-Петербург» value=«Санкт-Петербург, Россия» />
<option label=«Кемерово» value=«Кемерово, Кемеровская область, Россия» />
</datalist>
<input type='submit' value='Отправить'/>
</form>
<hr />
<form action='html5.php'>
Выберите тип шифрования:
<keygen name='keygen' />
<input type='submit' value='Отправить'/>
</form>
<hr />
<form action='html5.php' id='form1'>
<input name='first' type='text' value='10'/>
```

X<input name='second' type='text' value='10' />=

<output name='result' for='first second'>100</output>

<input type='button' value='Посчитать' onclick='result.value=first.value*second.value' />

</form>

Новые атрибуты в формах. Помимо новых элементов в HTML 5 добавлены новые атрибуты.

autofocus делает поле активным после загрузки страницы (может использоваться со всеми типами **input**).

form указывает форму, которой принадлежит данное поле (может использоваться со всеми типами **input**).

multiple указывает, что данное поле может принимать несколько значений одновременно (может использоваться с **input** типов ***email*** и ***file***).

novalidate указывает, что данное поле не должно проверяться перед отправкой (может использоваться с **form** и **input**).

placeholder отображает текст-подсказку в поле (может использоваться с **input** следующих типов: ***text, search, url, tel, email*** и ***password***).

required указывает, что данное поле должно быть обязательно заполнено перед отправкой.

<form action='html5.php' id='form1'>

1. Поле автоматически будет активно после загрузки страницы

<input type='text' autofocus=«autofocus» />

2. Вы можете выбрать несколько файлов для отправки одновременно

<input type='file' multiple='multiple' />

</form>

3. Поле принадлежит форме form1 хотя находится за ее пределами

<input type='text' form='form1' />

<form action='html5.php' novalidate='novalidate'>

4. Поля этой формы не будут проверяться перед отправкой

Email: <input type='email' value='wisdomweb.ru' />

URL: <input type='url' value='admin@wisdomweb.ru' />
<input type='submit' value='Отправить' />
</form>
<form action='html5.php'>
5. У этого поля будет отображаться сообщение подсказка
<input type='email' placeholder='Введите Ваш email' />

6. Данное поле обязательно должно быть заполнено перед отправкой
<input type='text' required=«required» />
<input type='submit' value='Отправить' />
</form>

Выбор даты. В HTML5 были добавлены новые элементы ввода, позволяющие удобно выбирать дату и время.

date позволяет выбрать дату в формате *год-месяц-день_месяца*.

time позволяет выбрать время.

datetime позволяет выбрать дату в формате *год-месяц-день_месяцаTвремяZ* (отчет ведется по глобальному времени).

datetime-local позволяет выбрать дату в формате *год-месяц-день_месяцаTвремя* (отчет ведется по местному времени).

month позволяет выбрать дату в формате *год-месяц*.

week позволяет выбрать дату в формате *год-Wнеделя*.

<form action='html5.php'>
Выберите дату: <input type='date' />
Выберите время: <input type='time' />
Выберите глобальное время и дату: <input type='datetime' />
Выберите местное время и дату: <input type='datetime-local' />
Выберите месяц: <input type='month' />
Выберите неделю: <input type='week' />
</form>

2. CSS

На данный момент для оформления сайтов повсеместно используется технология CSS. CSS расшифровывается как Cascading Style Sheets (Каскадные Листы Стилей). С помощью CSS возможно оформить как любой HTML элемент, так и документ в целом.

CSS были представлены вместе с HTML 4.0 и создавались для разрешения проблем с оформлением, возникающих в предыдущих версиях HTML. Во время создания HTML в 1991 году не предполагалось, что он будет содержать тэги, позволяющие оформлять документы. Изначально HTML был создан только для группировки текста

В HTML 3.2 были добавлены такие тэги как **<font>** (данный тэг использовался для изменения шрифта элементов) оформление и разметка смешались в единое целое, и оформление элементов стало занимать огромное количество времени.

Для решения этой проблемы в конце 1996 года W3C (Консорциум Всемирной Паутины – занимается разработкой веб-стандартов) представил CSS.

CSS позволяют хранить информацию об оформлении HMTL документа в отдельном внешнем файле с расширением .css. Редактируя лишь один этот файл, можно изменить оформление веб-сайта в целом. На данный момент CSS являются стандартом оформления HTML документов и поддерживаются всеми современными браузерами. В данном учебном пособии рассмотрен наиболее популярный сейчас стандарт CSS 2.1.

2.1. Синтаксис

Таблицы стилей состоят из набора правил (1). Каждое правило состоит из одного или нескольких селекторов (3) и блока определения (2), выделяющегося фигурными скобками (рис. 2.1).

Блок определения может содержать одно или несколько свойств (4), отделенных точкой с запятой (после последнего

свойства точка с запятой необязательна). Каждое свойство должно иметь значение (5), отделенное двоеточием.

Рис. 2.1. Синтаксис CSS

Таблицы стилей могут содержать комментарии, которые используются для создания пояснений. Комментарии полностью игнорируются браузером при разборе таблиц стилей.

В CSS комментарии начинаются с «/*», и заканчиваются «*/».

```
/* Правило перекрасит абзацы HTML документа в зеленый
цвет */
p {color:green;}
```

Необходимо всегда вставлять комментарии, потому что разбираться в коде с пояснениями намного проще.

2.2. Селекторы

С помощью селекторов выбираются элементы на странице, которые необходимо оформить.

В CSS существуют несколько видов селекторов.

Селекторы тэгов – используются для оформления элементов страницы по названию тэгов.

```
p {color:green;}
h2 {color:red;}
```

Селектор id – данный вид селекторов позволяет производить более точную выборку и используется, когда необходимо выбрать только один определенный элемент на странице, с предварительно заданным идентификатором. Идентификатор для элемента задается с помощью атрибута ***id*** (**<p *id=«идентификатор»*>**текст**</p>**).

Для того чтобы оформить данный элемент, необходимо обратиться к идентификатору в таблицах стилей, добавив перед ним символ «#» (**#идентификатор {*color:red*}**).

```
/* Оформим элемент с id=«test1» */
#test1
{
 color:green;
 font-family:verdana;
 font-size:1.2em;
}
```

Селектор class – данный вид селекторов позволяет выбирать для оформления не единственный элемент, а группу элементов. С помощью атрибута ***class*** можно указать, что элемент относится к некоторой группе (**<p *class= «имя_группы»*>** текст **</p>**).

Для того, чтобы затем оформить эту группу, необходимо в таблицах стилей обратится к имени группы, добавив перед ней символ «.» (**.имя_группы {*color:red*}**).

Имя группы и идентификатор может состоять только из латинских букв и не может начинаться с цифр.

```
/* Свойства будут применены ко всем элементам с
class=«test1» */
.test1 {
 color:green;
 font-family:verdana;
 font-size:1.2em;}
```

Селекторы атрибутов - используются для оформления элементов страницы по названию атрибутов.

```
/* Оформит все элементы имеющие атрибут src */
 [src]
```

```
{
 border:solid green 3px;
}

/* Оформит все элементы, атрибут href которых имеет
значение http://www.kemtipp.ru */
[href=«http://www.kemtipp.ru»]
{
 color:green;
}
```

Комбинирование селекторов – используется для более точного выбора элементов в CSS. Например, можно комбинировать селекторы тэгов с селекторами class:

```
/* Свойства будут применены только к тем элементам с
class=«test1», которые являются заголовками h2 */
h2.test1
{
 color:green;
 font-family:verdana;
 font-size:1.2em;
}
```

Также возможно комбинирование селекторов тэгов.

```
/* Свойства будут применены только к тем элементам p,
которые находятся внутри элементов div */
div p
{
 color:green;
 font-family:verdana;
 font-size:1.2em;
}
```

Символ «+» позволяет выбирать элементы, которые идут сразу после указанного.

```
/* Свойства будут применены только к тем элементам p,
которые идут сразу после элементов div */
div+p
{
```

```
 color:green;
 font-family:verdana;
 font-size:1.2em;
}
```

Группировка селекторов – используется, когда при оформлении HTML документов с помощью CSS приходится применять одинаковые свойства для разных селекторов.

```
h1 { font-family:verdana; color:green; }
h2 { font-family:verdana; color:green; }
p { font-family:verdana; color:green; }
```

Для того, чтобы сократить размер кода, необходимо сгруппировать селекторы с одинаковыми свойствами, разделяя их запятой.

```
h1,h2,p
{
 font-family:verdana;
 color:green;
}
```

2.3. Включения

Листы стилей можно подключать различными способами, каждый из них удобен в конкретном случае.

Внешнее объявление стилей используется в случаях, когда оформление задается для группы связанных HTML документов (например, для целого веб-сайта).

В этом случае все оформление выносится в один внешний файл, на который должны ссылаться все документы веб-сайта. Внешнее объявление стилей очень удобно, так как позволяет, редактируя лишь один файл, изменять оформление целого веб-сайта.

Для подключения внешнего файла стилей, необходимо в секции **head** каждой страницы веб-сайта указать ссылку на него с помощью элемента **<link>**.

```
<head>
```

```
<link rel=«stylesheet» type=«text/css» href= «адрес_внешнего_файла_стилей» />
</head>
```

Атрибут ***type*** тэга **style** является необязательным в HTML 5.

Внешний файл стилей является обычным текстовым файлом с расширением .css.

Внутреннее объявление стилей используется в случаях, когда стиль нужно задать только для одного отдельного HTML документа.

В этом случае оформление определяется в секции **head** с помощью тэга **style**.

```
<head>
<style type='text/css'>
h1 {
color:red;
}
p {
margin-right:38px;
}
div {
float:left;}
</style>
</head>
```

Строковое объявление стилей используется в случаях, когда необходимо оформить только один определенный элемент в HTML документе.

Для этого необходимо воспользоваться атрибутом ***style*** соответствующего элемента. Атрибут ***style*** может содержать любые CSS свойства.

```
<p style=«font-size:1.3em»> Абзац, оформленный с помощью CSS.</p>
```

Приоритет стилей

Стили, подключенные разным способом, имеют разный приоритет. Перечислим их, в порядке увеличения приоритета:

— внешние стили;

— внутренние стили;

— строковые стили.

Если один HTML документ имеет несколько стилей, подключенных разным способом, и в них заданы разные свойства оформления для одного и того же элемента, то в итоге элемент будет оформлен согласно содержимому стиля с более высоким приоритетом.

```
/* Внутренние стили: */
h1 {color:red}
/* Строковые стили: */
<h1 style=”color:green;”> Привет </h1>
/* В результате заголовок будет зеленого цвета, так как строковые стили имеют более высокий приоритет */
```

2.4. Текст

CSS имеют большие возможности для оформления текста.

Цвет текста. С помощью CSS свойства ***color*** можно изменять цвет текста HTML элементов.

Цвет может быть задан следующими способами:

— с помощью названия цвета (например, ***'red'*** – задаст красный цвет);

— с помощью RGB значения (например, ***'rgb(255,255,255)'*** -–задаст белый цвет);

— с помощью шестнадцатеричного значения (например, ***'#00ff00'*** – задаст зеленый цвет).

При задании цвета первым способом используются названия цветов – предопределенные слова, каждое из которых соответствует определенному цвету. Существуют специальные таблицы с перечнями существующих в HTML цветов.

При втором способе – цвет формируется по трем составляющим: ***rgb(красный, зеленый, синий)***, где:

— ***красный*** – число от 0 до 255, указывающее на насыщенность красного цвета в итоговом оттенке;

— ***зеленый*** – число от 0 до 255, указывающее на насыщенность зеленого цвета в итоговом оттенке;

— ***синий*** – число от 0 до 255, указывающее на насыщенность синего цвета в итоговом оттенке.

Учитывая величину всех трех цветов, браузер формирует итоговый цвет.

Например, ***rgb(255,0,0)*** задаст красный цвет, ***rgb(0,255,0)*** – зеленый. Смешивая красный с зеленым ***rgb(255,255,0)***, получим – желтый.

Третий способ задания цвета похож на второй, с тем лишь отличием, что насыщенность каждого цвета задается числом в шестнадцатеричном формате (сами числа находятся в промежутке от 0 до 255): ***#красныйзеленыйсиний***, где:

— ***красный*** – шестнадцатеричное число от 0 до ff, указывающее на насыщенность красного цвета в итоговом оттенке;

— ***зеленый*** – шестнадцатеричное число от 0 до ff, указывающее на насыщенность зеленого цвета в итоговом оттенке;

— ***синий*** – шестнадцатеричное число от 0 до ff, указывающее на насыщенность синего цвета в итоговом оттенке.

Например, ***#ff0000*** задаст красный цвет, а ***#00ff00*** зеленый. Смешивая красный с зеленым ***#ffff00***, получим - желтый.

```
p {color:green;}
p {color:rgb(0,255,0);}
p {color:#00ff00;}
```

Выравнивание текста. С помощью CSS свойства ***text-align*** можно выровнять текст элемента по горизонтали. Текст может быть выравнен:

— по центру (значение ***center***);

— по левому краю (***left***);

— по правому краю (***right***);

— по ширине (***justify***).

По ширине (***justify***) текст выравнивается путем растягивания всех строчек до одинаковой длины. Этот метод выравнивания используется в газетах, журналах, научных и студенческих работах (курсовые, дипломные и т.д.).

```
p.ta1 {text-align:center;}
p.ta2 {text-align:left;}
p.ta3 {text-align:right;}
```

```
p.ta4 {text-align:justify;}
```

Декорирование текста. С помощью CSS свойства ***text-decoration*** можно добавить тексту HTML элемента следующие эффекты:

— подчеркивание (значение ***underline***);

— перечеркивание (***line-through***);

— отображение над текстом элемента линии (***overline***).

```
p.td1 {text-decoration:underline;}
p.td2 {text-decoration:line-through;}
p.td3 {text-decoration:overline;}
```

Таблица 2.1

Свойства текста

Свойство	Описание	Значения
direction	Направление текста	ltr rtl
line-height	Высота строки	normal пиксели %
text-indent	Величину отступа первого символа текста	пиксели %
text-transform	Регистр букв текста элемента	none capitalize uppercase lowercase
vertical-align	Вертикальное выравнивание элемента	sub supper top middle bottom пиксели %
white-space	Обработка пробелов внутри элементов	normal pre nowrap

Свойство ***text-decoration*** со значением ***none*** «очищает» текст от всех вышеперечисленных эффектов. Это может использоваться для создания неподчеркнутых ссылок.

Подчеркивать обычный текст не рекомендуется, так как пользователи могут перепутать его со ссылкой.

Отступ между словами и буквами в тексте. С помощью CSS свойства ***letter-spacing*** можно увеличивать или уменьшать отступ между буквами в тексте HTML элементов.

```
p.ls1 {letter-spacing:10px;}
```

С помощью свойства ***word-spacing*** можно увеличивать или уменьшать отступ между словами в тексте HTML элементов.

```
p.ws1 {word-spacing:15px;}
```

В табл. 2.1 приведены некоторые свойства текста.

2.5. Шрифт

Свойство ***font-family*** позволяет устанавливать шрифт для текста HTML элементов. Если название шрифта состоит из одного слова, то кавычки не ставятся, если название шрифта состоит из нескольких слов, то оно обязательно должно заключаться в кавычки.

```
p.f1 {font-family:Arial;}
p.f2 {font-family:”Times New Roman”;}
```

Все шрифты делятся на безопасные и небезопасные. Безопасными шрифтами называют шрифты, вероятность поддержки которых на любом компьютере с любой установленной операционной системой близка к 100%.

К безопасным шрифтам относятся:

— Arial;
— Arial Black;
— Courier New;
— Comic Sans MS;
— Georgia;
— Impact;
— Times New Roman;
— Trebuchet MS;

— Verdana.

В CSS 3 добавлено новое свойство ***@font-face***, позволяющее использовать на веб-страницах небезопасные шрифты.

Если есть необходимость использовать небезопасные шрифты, без использования свойства ***@font-face***, то можно использовать *резервные шрифты*.

Резервный шрифт будет использован в случае, если основной шрифт не установлен на компьютере пользователя. Элементы могут иметь несколько резервных шрифтов, которые будут поочередно перебираться, пока не найдется установленный шрифт.

```
/* Если на компьютере пользователя нет шрифта Verdana,
то будет использован Times New Roman, если на компью-
тере нет Times New Roman, то будет использован Arial */
p {font-family:Verdana,»Times New Roman»,Arial;}
```

В качестве замыкающего резервного шрифта часто указывают семейство, к которому принадлежит основной шрифт, чтобы, в крайнем случае, браузер автоматически выбрал один из шрифтов данного семейства, установленных на компьютере пользователя.

В CSS шрифты делятся на следующие «семейства»:

— **Serif** – шрифты данного семейства имеют небольшие засечки на концах символов. Примеры шрифтов данного семейства: ***Times New Roman, Georgia***.

— **Sans-Serif** – шрифты данного семейства не имеют засечек. Примеры: ***Arial, Arial Black, Trebuchet MS, Verdana***.

— **Monospace** – все символы шрифтов данного семейства занимают одинаковую ширину. Примеры: ***Courier New***.

— **Fantasy** – семейство «причудливых» шрифтов, использующихся в основном для создания красочных заголовков. Примеры: ***Impact***.

— **Cursive** – семейство шрифтов рукописного начертания. Примеры: ***Comic Sans MS***.

```
/* Если на компьютере пользователя нет шрифта Verdana,
то будет использован Arial, если на компьютере нет Arial,
то будет использоваться один из шрифтов семейства Sans-
Serif, имеющихся на компьютере */
```

```
p {font-family:Verdana,Arial,Sans-serif;}
```

Размер шрифта. CSS свойство ***font-size*** устанавливает размер шрифта HTML элементов. Размер шрифта можно задать двумя способами.

Абсолютный способ – этот способ по сравнению со следующим удобнее, но его использование может привести к несоответствию при отображении одной и той же страницы в различных браузерах. Данный способ позволяет задавать размер шрифта в абсолютных единицах, таких как: пиксели (px) или проценты (%).

Относительный способ – данный способ помогает избежать проблем с несоответствием при отображении страницы в разных браузерах, так как все размеры устанавливаются относительно. Для задания размера шрифта данным способом используются единицы ***em***. ***1em*** эквивалентна размеру шрифта в браузере по умолчанию и равна 16px. W3C рекомендует для задания шрифта использовать именно этот способ.

```
p.fz1 {font-size:1.2em;}
p.fz2 {font-size:1.5em;}
p.fz3 {font-size:1.15em;}
```

Не следует использовать свойство ***font-size*** для того, чтобы оформить абзац как заголовок. Для определения абзацев всегда нужно использовать тэг **p**, а для заголовков **h1-h6**.

Стиль шрифта. Свойство ***font-style*** позволяет сделать шрифт HTML элемента курсивным.

Свойство ***font-weight*** позволяет изменять толщину шрифта.

```
p.italic {font-style:italic;}
p.fz1 {font-weight:bold;}
```

2.6. Фон

В CSS существует группа свойств для оформления фона HTML элементов:

— background-attachment;

— background-color;

— background-image;

— background-position;

— background-repeat.

Цвет фона. CSS свойство ***background-color*** позволяет установить цвет фона для выбранного элемента.

```
body {background-color:green;}
```

Цвет может быть задан следующими способами:

— с помощью названия цвета (например, ***'red'*** – задаст красный цвет);

— с помощью RGB значения (например, ***'rgb(255,255,255)'*** – задаст белый цвет);

— с помощью шестнадцатеричного значения (например, ***'#00ff00'*** – задаст зеленый цвет).

Изображение в качестве фона. С помощью CSS свойства ***background-image*** можно вставить произвольное изображение в качестве фона. По умолчанию, изображение будет повторяться, пока не заполнит все содержимое элемента.

```
body {background-image:url('image1.jpg');}
```

Необходимо всегда тщательно подбирать цвета. Фоновое изображение не должно сливаться с текстом.

Зачастую повторяющееся изображение в качестве фона не требуется. CSS свойство ***background-repeat*** позволяет определить, как должно повторяться фоновое изображение при вставке:

— ***background-repeat:repeat-x*** – изображение будет повторяться только по горизонтали;

— ***background-repeat:repeat-y*** – изображение будет повторяться только по вертикали;

— ***background-repeat:no-repeat*** – изображение не будет повторяться.

CSS свойство ***background-position*** позволяет задавать местоположение фонового изображения. В качестве первого значения данного свойства должна задаваться величина смещения изображения по горизонтали, а в качестве второго величина смещения по вертикали. Смещение происходит от левого верхнего угла страницы, по горизонтали – вправо, по вертикали – вниз.

Величина смещения может быть указана как с помощью пикселей (px), процентов (%) и сантиметров (cm) (***background-***

position:50px 30px;), так и с помощью предопределенных ключевых слов (***background-position:right top;***).

CSS свойство ***background-attachment*** позволяет прикреплять фоновые изображения к определенным местам. Прикрепленное изображение будет оставаться на данном месте даже при прокрутке страницы.

```
body {background-attachment:fixed;}
```

Краткая форма записи. Для того чтобы сократить итоговый размер кода, в CSS предусмотрена краткая (стенографическая) форма записи некоторых свойств.

Для краткой записи оформления фона элементов в CSS предусмотрено свойство ***background***.

```
body
{background:url('kartinka.jpg') no-repeat fixed right top;}
```

При использовании стенографической формы записи соблюдается следующий порядок следования свойств:

1. background-color;
2. background-image;
3. background-repeat;
4. background-attachment;
5. background-position.

Если какое-либо свойство не требуется изменить, то его можно пропустить.

2.7. Ссылки

Каждая ссылка имеет четыре состояния, каждое из этих состояний может быть отдельно оформлено с помощью специальных псевдо-классов.

Псевдо-класс должен добавляться к селектору элемента, отделяясь от него двоеточием (:).

Оформление для псевдо-классов должно задаваться в следующем порядке:

1. ***a:link*** – определяет оформление обычной, не посещенной ссылки;

2. ***a:visited*** – определяет оформление посещенной пользователем ссылки;

3. ***a:hover*** – определяет оформление ссылки, на которую наведен курсор мыши;

4. ***a:active*** – определяет оформление ссылки, на которую щелкнули мышкой.

```
a:link {text-decoration:none; color:green;}
a:visited {text-decoration:none; color:green;}
a:hover
{text-decoration:underline; color:red; font-size:1.1em;}
a:active
{ text-decoration:none; color:red; font-size:1.1em;}
```

2.8. Списки

CSS свойство ***list-style-type*** позволяет оформлять списки. Всего существует два вида списков:

— *неупорядоченные* – маркер таких списков имеет вид закрашенного круга;

— *упорядоченные* – элементы таких списков нумеруются арабскими цифрами.

Перед каждым элементом списка (упорядоченным и неупорядоченным) ставится метка, которая называется маркер. У неупорядоченных списков маркер по умолчанию является закрашенным кругом, у упорядоченных – арабской цифрой. Маркер можно менять, варианты маркеров приведены в табл. 2.2.

Использование маркера-картинки. Свойство ***list-style-image*** позволяет заменить маркер списка на произвольное изображение.

```
ul.lis1 {list-style-image:url('marimg.gif'); }
```

Таблица 2.2

Виды маркеров списков

Значение	Описание
Значения для неупорядоченных списков	
none	Нет маркера
disc	Закрашенный черный круг. *Значение по умолчанию*
circle	Не закрашенный круг
square	Закрашенный квадрат
Значения для упорядоченных списков	
armenian	Армянские числа
decimal	Арабские числа. *Значение по умолчанию*
decimal-leading-zero	Арабские числа, начинающиеся с нуля (01, 02, 03 и т.д.)
georgian	Грузинские числа
lower-greek	Традиционная греческая нумерация (alpha, beta, gamma)
lower-latin	Маленькие латинские буквы (a, b, c, d)
lower-roman	Маленькие римские цифры (i, ii, iii, iv)
upper-latin	Большие латинские буквы (A, B, C, D)
upper-roman	Большие римские цифры (I, II, III, IV)

2.9. Таблицы

Ширина и высота таблицы являются регулируемыми величинами. С помощью CSS свойства ***width*** можно устанавливать ширину таблицы. В основном ширина устанавливается в пикселях или %, но можно также использовать cm и em.

CSS свойство ***height*** позволяет установить высоту таблицы. Высота в основном указывается в пикселях, но можно также использовать cm и em.

```
.tab1 {width:100%;}
.tab2 {width:70%;}
.tab3 {width:300px;}
.tab1 {height:200px;}
.tab2 {height:7cm;}
```

Оформление границ. Для оформления табличных границ в CSS используется свойство ***border***. Свойство ***border*** не является уникальным свойством таблиц, оно может использоваться с любыми элементами.

```
table, th, td
{
border-style:solid;
border-color:green;
border-width:1px;
}
```

Сплошные границы. Таблица в примере выше имеет двойную границу, потому что и сама таблица и ее ячейки (элементы **th** и **td**) имеют собственные границы. Свойство ***border-collapse*** позволяет соединить границы таблицы и ячеек. Соединенные границы обычно смотрятся аккуратнее.

```
table, th, td
{
border-style:solid;
border-color:green;
border-width:1px;
border-collapse:collapse;
}
```

Выравнивание табличного текста. С помощью свойства ***text-align*** можно выравнивать текст табличных ячеек по горизонтали. Текст может быть выравнен:

— по левому краю (значение ***left***);

— по правому краю (***right***);

— по центру (***center***).

```
.tab1 {text-align:right;}
.tab2 {text-align:left;}
.tab3 {text-align:center;}
```

С помощью свойства ***vertical-align*** можно выравнивать текст табличных ячеек по вертикали. Текст может быть выравнен:

— по верхней границе (***top***);

— по центру (***middle***);

— по нижней границе (***bottom***).

```
.top {vertical-align:top;}
.mid {vertical-align:middle;}
.bot {vertical-align:bottom;}
```

Отступ от границы. С помощью свойства ***padding*** можно устанавливать величину отступа между границей ячейки и ее содержимым.

```
#tab1 td {padding:10px;}
#tab2 td {padding:5px;}
#tab3 td {padding:0px;}
```

2.10. Блоковая модель

Все элементы в CSS являются прямоугольными блоками. Каждый такой блок имеет зону ***content***, в которой располагается содержимое элемента (т.е. текст, изображения и т.д.). Вокруг зоны ***content*** могут располагаться необязательные зоны: ***padding, border*** и ***margin***. Схема блоковой модели представлена на рис. 2.2.

Рис. 2.2. Схема блоковой модели

Зона ***padding*** окружает зону ***content***. Данная зона используется для задания величины отступа содержимого элемента (зона ***content***) от его границы (зона ***border***). Зона может быть разбита на четыре части, которые могут оформляться независи-

мо друг от друга: ***padding-top, padding-right, padding-bottom, padding-left***.

Зона ***border*** окружает зону ***padding***. Данная зона позволяет задать элементу границу произвольной ширины, стиля и цвета. Зона может быть разбита на: ***border-top, border-right, border-bottom, border-left***.

Зона ***margin*** окружает зону ***border***. Данная зона позволяет задать величину внешнего отступа данного элемента от окружающих. Зона может быть разбита на: ***margin-top, margin-right, margin-bottom, margin-left***.

```
.ex1
{border:10px red solid;
margin-left:50px; margin-right:10px;
padding:15px;}
.ex2
{border:5px brown solid;
margin-top:30px; margin-left:250px; margin-right:70px;
padding:15px;}
```

Свойства ***width*** и ***height*** устанавливают ширину и высоту только блока ***content***, а не элемента целиком. Итоговый размер элемента, помимо размеров ***content***, будет включать в себя еще размеры ***padding***, ***border*** и ***margin***.

```
.ex1 {width:400px;}
.ex2
{width:250px;
padding-left:85px; padding-right:85px;
border-left:5px brown solid;
border-right:5px brown solid;}
```

2.10.1. Границы

Границы каждого элемента могут отображаться различными стилями. CSS свойство ***border-style*** позволяет установить стиль для границ HTML элемента. Существуют следующие возможные значения для установки стилей границ:

— ***solid*** границы будут нарисованы сплошной линией;

— ***dashed*** границы будут нарисованы пунктирной линией;

— ***dotted*** границы будут нарисованы точками;

— ***double*** границы будут нарисованы двойной сплошной линией.

```
.bor1 {border-style:solid;}
.bor2 {border-style:dashed;}
.bor3 {border-style:dotted;}
```

По умолчанию граница элементов всегда невидима (значение ***none***).

Цвет границы. С помощью CSS свойства ***border-color*** можно задать цвет границы HTML элемента. Цвет может быть задан следующими способами:

— с помощью названия цвета (например, ***'red'*** – задаст красный цвет);

— с помощью RGB значения (например, ***'rgb(255,255,255)'*** – задаст белый цвет);

— с помощью шестнадцатеричного значения (например, ***'#00ff00'*** – задаст зеленый цвет).

Толщина границы. С помощью CSS свойства ***border-width*** можно задать толщину границы HTML элемента. Толщина может быть указана либо в пикселях, либо с помощью предопределенных значений: ***thin, medium, thick*** (тонкая, средняя, толстая).

```
.bor1 {border-style:solid; border-width:4px;}
.bor2 {border-style:solid; border-width:2px;}
.bor3 {border-style:solid; border-width:thin;}
```

Задание стилей для отдельных сторон. Рассмотренные ранее свойства могут применяться к отдельным сторонам границы.

Существует способ быстрого задания стилей границ. Например, в результате применения ***border-style:dashed double solid groove***; к верхней границе будет применено ***dashed***, к правой ***double***, к нижней ***solid***, а к левой ***groove***.

Если указать только два значения, например, ***border-style:dashed double***, то верхняя и нижняя границы будут оформлены как ***dashed***, а левая и правая границы будут оформлены как ***double***.

```
.bor1 {border-top-style:solid; border-width:2px;}
.bor2 {border-bottom-style:solid; border-width:2px;}
.bor3
{border-left-style:solid; border-right-style:solid;
 border-width:2px;}
```

Стенографическое задание свойств. Для того чтобы сократить длину кода, в CSS предусмотрен стенографический способ записи. Стенографическая форма записи объединяет все свойства оформления границ в одном свойстве ***border***.

```
/* Вокруг элемента с классом bor1 будет отображена
сплошная граница зеленого цвета, толщиной 2 пикселя */
.bor1 { border:2px solid green;}
```

Порядок следования свойств в стенографической форме записи следующий:

1. border-width;
2. border-style;
3. border-color.

Если какое-либо свойство не требуется изменить, то его можно пропустить.

```
.bor1 {border:1px solid;}
```

Внешняя граница. Помимо обычной границы элементы могут иметь еще и внешнюю границу, которая задается с помощью CSS свойства ***outline***.

```
.out1
{outline-style:dashed;
/* определяет стиль внешней границы */
outline-color:green;
/* определяет цвет внешней границы */
outline-width:4px;
/* определяет ширину внешней границы */
border-style:solid;}
```

2.10.2. Отступы

В CSS существует два вида отступов: внутренние и внешние. *Внутренний отступ* – это расстояние между содержимым элемента и его границей. *Внешний отступ* – это расстояние, отступаемое с внешней стороны границы элемента.

Величина отступов может определяться в пикселях (px), сантиметрах (cm), процентах (%), em.

Внутренний отступ. CSS свойство ***padding*** позволяет установить величину внутреннего отступа. Величина отступа может быть независимо определена для каждой стороны элемента.

```
.pad1 {border-style:solid; padding:20px;}
.pad2 {padding-top:30px; padding-left:20px;
padding-bottom:10px; padding-right:40px;}
```

Внешний отступ. С помощью CSS свойства ***margin*** можно задать величину внешнего отступа. Величина внешнего отступа может быть задана отдельно для каждой стороны элемента.

```
.mar1 { margin:25px; }
.mar2 {margin-top:20px; margin-bottom:40px;
 margin-left:70px; margin-right:10px;}
```

Краткая форма записи. Для того, чтобы сократить размер кода, можно определять величину отступа для каждой стороны элемента, используя краткую форму записи свойств.

```
.pad1
 {border:2px solid;
 padding:60px 20px 40px 50px;}
.mar1
 {margin:50px 20px 40px 50px;}
```

2.11. Отображение

Любой видимый элемент в HTML можно скрыть, сделать это можно двумя способами:

— с помощью свойства ***visibility:hidden***;

— с помощью ***display:none***.

Элемент скрытый первым способом будет невидим, но будет по-прежнему занимать место на странице. Второй способ позволяет полностью скрыть элемент, чтобы он больше не занимал места на странице.

```
/* Абзац стал невидим, но все еще занимает место на странице */
p.dis1 {visibility:hidden;}
/* Абзац стал невидим и не занимает места */
p.dis2 {display:none;}
```

Блочные и строковые элементы. В CSS встречаются два типа элементов по способу отображения:

— *Блочные элементы* полностью занимают всю ширину их родительского элемента. Примеры блочных элементов: **p, h1-h6, div**.

— *Строковые элементы* занимают только необходимую им ширину. Примеры строковых элементов: **a, span**.

С помощью CSS можно изменять способ отображения элементов.

```
/* Превратим блочный элемент p в строковый */
#dis1 {display:inline;}
/* Превратим строковый элемент a в блочный */
a.dis2 {display:block;}
```

2.12. Размещение

Размещение всех элементов на HTML странице происходит за счет применения свойств позиционирования. Местоположение элементов задается с помощью следующих CSS свойств:

— ***top*** устанавливает величину смещения текущего элемента от верхнего края родительского элемента;

— ***bottom*** устанавливает величину смещения текущего элемента от нижнего края родительского элемента;

— ***left*** устанавливает величину смещения текущего элемента от левого края родительского элемента;

— ***right*** устанавливает величину смещения текущего элемента от правого края родительского элемента.

Описанные выше свойства не вступят в силу, пока не будет задан способ размещения. Также способ размещения определяет поведение данных свойств.

Статическое размещение. Статичные элементы всегда отображаются там, где они были объявлены. CSS свойства ***top, bottom, left*** и ***right*** не работают со статичными элементами. Все элементы по умолчанию размещаются данным способом, но возможно явно объявить элемент статичным с помощью ***position:static***.

```
#pos1
{position:static; top:40px; left:17px;}
```

Фиксированное размещение. Фиксировано размещенные элементы не изменяют своего местоположения даже при прокрутке окна браузера. К фиксировано размещенным элементам могут применяться CSS свойства ***top, bottom, left, right***. Элемент объявляется фиксировано размещенным с помощью ***position:fixed***.

```
#pos1
{position:fixed; right:40px; top:17px;}
```

Относительное размещение. Относительно размещенные элементы размещаются относительно их обычной позиции. Элемент объявляется относительно размещенным с помощью ***position:relative***. Относительно размещенные элементы (в отличие от абсолютно- и фиксировано-размещенных) занимают место, на котором они изначально были определены и место, на которое они были смещены, путем применения свойств позиционирования.

```
#pos1
{position:relative; top:40px; left:170px;}
```

Абсолютное размещение. Абсолютно размещенные элементы располагаются относительно первого родительского элемента, способ размещения которого отличен от статического. Если такие элементы не были найдены, то элемент будет расположен относительно корневого элемента (**html**). Иногда, для того чтобы добиться желаемого эффекта, родительский элемент специально определяется как относительно размещенный с нулевым смещением. Объявление элемента абсолютно размещенным происходит с помощью ***position:absolute***.

```
#pos1
{
position:absolute; top:10px; left:200px;
}
#pos2
{
position:absolute; top:1px; left:0px;
}
#pos3
{
position:absolute; top:100px; right:70px;
}
```

Наложение элементов. При применении свойств позиционирования элементы могут накладываться друг на друга. Свойство ***z-index*** позволяет установить какой элемент в случае наложения будет сверху, а какой снизу.

Элементы с большим значением свойства ***z-index*** располагаются выше других. Свойство ***z-index*** может принимать отрицательные значения.

```
#pos1 {
z-index:10;
}
#pos2 {
z-index:5;
}
```

```
#pos3 {
z-index:-1;
}
```

В табл. 2.3 приведены связанные с размещением CSS свойства.

Таблица 2.3

Свойства CSS, связанные с размещением

Свойство	Описание	Значения
clip	Обрезает элемент, размещенный абсолютно	rect auto inherit
cursor	Задает вид, который будет принимать курсор при наведении на элемент	auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help
overflow	Устанавливает, как должно быть отображено содержимое, вышедшее за границы элемента	auto hidden scroll visible inherit

2.13. Выравнивание

Центрирование с помощью внешних отступов. Блочные элементы могут быть выравнены по центру установкой ***margin*** с левой и правой стороны значения ***auto***. Если ширина блочного элемента будет равна 100%, то он не будет выравнен (т.к. доступного ***margin*** нет).

```
.ali1 {margin-left:auto; margin-right:auto; width:50%;}
```

Выравнивание с помощью свойств позиционирования. Имеется возможность выравнивать элементы с помощью свойств позиционирования.

```
.pos
{position:absolute; width:400px; left:0px;
background-color:green;}
```

Свойство обтекания. Элементы в CSS также могут быть выравнены с помощью свойства ***float***. Элемент, выравненный с помощью ***float***, будет прижат к левой или правой границе родительского элемента (в зависимости от заданного значения) и заставит следующие за ним элементы «обтекать» его с противоположной стороны.

Свойство ***float*** наиболее часто используется с изображениями, но оно также бывает полезно при обычном выравнивании.

```
.fl1 {float:right;}
```

HTML элементы, следующие за элементом с заданным ***float***, будут его «обтекать», чтобы избежать этого, необходимо использовать свойство ***clear***.

Значения, переданные данному свойству (***left*** или ***right***), указывают, с какой стороны находится элемент с ***float***. Если нет уверенности или элементы с ***float*** находятся с двух сторон, то нужно воспользоваться свойством ***both***.

```
.fl2 {clear:both;}
```

2.14. Псевдо-классы

Псевдо-класс :first-child позволяет выбрать элемент, который является первым потомком в его родительском элементе.

```
p:first-child {color:red;}
```

Псевдо-класс :first-letter позволяет оформить первую букву указанного элемента.

```
p:first-letter
{font-size:1.8em; color:green; font-family:»Comic Sans MS»;
border:1px solid black; padding:2px; }
```

Псевдо-класс :focus позволяет выбрать активные **input** элементы формы.

```
:focus {background-color:yellow;}
```

Псевдо-класс :first-line позволяет оформить первую строчку указанного элемента.

```
p:first-line {font-size:1.2em; font-weight:bold; color:green;}
```

С помощью **псевдо-классов :before и :after** можно вставить произвольное содержимое до и после указанных элементов.

```
p:before {content:»CSS - «; font-weight:bold;}
p:after {content:» HTML документы.»;}
```

2.15. Условные комментарии

При создании сайта веб-разработчикам часто приходится сталкиваться с проблемами совместимости. Дело в том, что не все браузеры одинаково отображают одни и те же страницы.

Основные браузеры «доставляющие неудобства» это Internet Explorer 8 и Internet Explorer 7, доля использования которых на сегодняшний день достаточно велика (на 1 июля 2016 года Internet Explorer 8 используют ~4,5%, а Internet Explorer 7 ~1,8 % пользователей России).

Для того чтобы избежать проблем с некорректным отображением страниц в браузерах, веб-разработчики часто прибегают к использованию *условных комментариев*.

Условные комментарии могут использоваться для выбора определенных версий Internet Explorer.

```
<!--[if IE 8]>
Стили расположенные здесь будут применены только для
Internet Explorer 8
<![endif]-->

<!--[if IE 7]>
Стили расположенные здесь будут применены только для
Internet Explorer 7
<![endif]-->
```

```
<!--[if lt IE 8]>
Стили расположенные здесь будут применены только для браузеров Internet Explorer, версия которых ниже 8
<![endif]-->
<!--[if lte IE 8]>
Стили расположенные здесь будут применены для браузера Internet Explorer 8 и браузеров IE, версия которых ниже 8
<![endif]-->

<!--[if gt IE 8]>
Стили расположенные здесь будут применены только для браузеров Internet Explorer, версия которых выше 8
<![endif]-->

<!--[if gte IE 8]>
Стили расположенные здесь будут применены для браузера Internet Explorer 8 и браузеров IE, версия которых выше 8
<![endif]-->
```

Чтобы избежать проблем с оформлением, разработчики создают три внешних CSS файла с различным оформлением для:

— Internet Explorer 8 (***ie8.css***);

— Internet Explorer 7 (***ie7.css***);

— для всех остальных браузеров Firefox, IE8+, Opera, Safari, Chrome (***all.css***).

```
<link rel=«stylesheet» type=«text/css» href=«all.css»>
<!--[if IE 8]>
<link rel=«stylesheet» type=«text/css» href=«ie8.css» />
<![endif]-->
<!--[if IE 7]>
<link rel=«stylesheet» type=«text/css» href=«ie7.css» />
<![endif]-->
```

Данный код должен располагаться в секции **head**.

Так как доля использования браузера Internet Explorer 7 среди пользователей заметно упала (его используют ~1.8 %) и продолжает стремительно падать, возникает справедливый вопрос: «А стоит ли вообще добавлять его поддержку на сайт?»

Выбор должен совершаться, исходя из личных соображений и анализа трафика сайта. Если доля использования IE7 среди посетителей сайта мала, то можно отказаться от его поддержки.

Некоторые веб-разработчики сейчас прибегают к побудительным мерам, которые призывают пользователей со старыми версиями браузеров обновить их.

С помощью, например, PHP определяется версия браузера, и, если это IE7 (или любой другой устаревший браузер), пользователю вместо запрашиваемой страницы выдается страница, на которой его просят обновить браузер и приводят ссылки на соответствующие ресурсы.

2.16. CSS 3

CSS 3 – это новый стандарт оформления HTML документов, значительно расширяющий возможности предыдущего стандарта CSS 2.1. Многие возможности, которые были труднодоступны в CSS 2.1, то есть требовали использования дополнительных внешних программ (таких как Adobe Photoshop), скриптов (таких как JavaScript) или специальных «хитростей» могут легко достигаться в CSS 3 за счет использования новых свойств оформления.

В CSS 3 появились следующие возможности:

— создавать элементы со сглаженными углами;

— создавать линейные и сферические градиенты;

— создавать гибкое оформление фоновой картинки элементов;

— добавлять к элементам и к тексту элементов тени;

— использовать небезопасные шрифты;

— создавать анимацию и различные эффекты переходов;

— задавать цвета несколькими новыми способами и многое другое.

```
<html>
<head>
<style type=«text/css»>
#wrap {
position:relative;
border:1px #000 solid;
float:left;
width:330px;
height:220px;
margin-right:30px;
background:-webkit-linear-gradient(top,#E567B1,#84004D);
background:-moz-linear-gradient(top,#CB0077,black);
background:-o-linear-gradient(top,#CB0077,black);
box-shadow:3px 3px 10px 1px #000000;}
#inner {
position:absolute;
top:80px;
left:0px;
padding:10px 0px;
background-color:#dfc51e;
width:330px;
border-radius:20px;
color:white;
font-family:Arial;
text-align:center;
font-size:2em;
text-shadow:2px 2px 0px #000000;}
</style>
</head>
<body>
<div id=«wrap»>
<div id=«inner»>KEMTIPP</div>
</div>
```

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

CSS 3 свойства, разобранные в данном учебном пособии, поддерживаются только в современных браузерах: IE9+, Firefox 3.6+, Opera 10+, Chrome 12+, Safari 5+. Спецификация CSS 3 все еще находится в разработке, поэтому поведение некоторых свойств рассмотренных в данном разделе может измениться. К тому же, уже начата разработка спецификации CSS 4, которая будет логичным дополнением CSS 3.

2.16.1. Фон

В CSS 3 имеется возможность устанавливать размер фоновых изображений с помощью свойства ***background-size***. Размер фоновых изображений может быть указан в пикселях или в процентах.

```
#wrap1 {
background-image:url(«image.gif»);
background-size:150px 250px;}
#wrap2 {
background-image:url(«image.gif»);
background-size:70% 70%;}
```

Теперь один элемент может иметь несколько фоновых изображений одновременно, реализовать это можно с помощью свойства ***background-image***.

```
#wrap1 {
background-image:url(wislink.gif),url(mountimg3.jpg);
background-position:bottom right, center;
background-size:150px 40px,100% 100%;
background-repeat:no-repeat,no-repeat;}
```

С помощью нового CSS 3 свойства ***background-origin*** можно установить, как должно вычисляться положение элемента относительно границ его родительского элемента. Данное свойство может иметь следующие значения:

— ***border-box*** положение элемента вычисляется относительно верхнего левого угла границы элемента;

— ***padding-box*** положение элемента вычисляется относительно верхнего левого угла блока padding;

— ***content-box*** положение элемента вычисляется относительно верхнего левого угла содержимого.

```
#wrap1 {
background-origin:border-box;
background-image:url(«border-box.png»);}
#wrap2 {
background-origin:padding-box;
background-image:url(«padding-box.png»);}
#wrap3 {
background-origin:content-box;
background-image:url(«content-box.png»);}
```

2.16.2. Цвет

В CSS 3 цвет может задаваться с помощью **HSL**, то есть оттенка, насыщенности и яркости. Для того чтобы задать цвет этим способом, необходимо указать:

— ***оттенок цвета*** указывается в градусах поворота цветового круга (0 градусов - красный, 120 градусов - зеленый, 240 градусов - голубой и т.д.);

— ***насыщенность цвета*** указывается в процентах (по мере понижения процентов цвет будет блекнуть);

— ***яркость цвета*** также указывается в процентах (0% - темный, 100% - светлый).

```
<html>
<style type='text/css'>
#wrap1,#wrap2,#wrap3 {
border:1px #000 solid;
width:230px; height:120px;
margin:10px;
float:left;
padding:10px;
font-size:1.5em;
```

```
color:#000;
text-decoration:underline;}
#wrap1 {
background-color:hsl(0,30%,50%);}
#wrap2 {
background-color:hsl(120,100%,80%);}
#wrap3 {
background-color:hsl(240,100%,50%);}
</style>
<body>
<div id=«wrap1»>hsl(0,30%,50%)</div>
<div id=«wrap2»>hsl(120,100%,80%)</div>
<div id=«wrap3»>hsl(240,100%,50%)</div>
</body>
</html>
```

Цвет может задаваться с помощью **RGBA**. Данный способ позволяет определять цвет и прозрачность одновременно. Вначале необходимо указать значения RGB, а затем значение прозрачности (0 - максимальная прозрачность, 1 - минимальная прозрачность).

Задание прозрачности с помощью RGBA отличается от действия свойства ***opacity*** тем, что ***opacity*** делает прозрачным сам элемент и все его элементы потомки, а RGBA делает прозрачным только сам элемент.

```
<html>
<style type='text/css'>
#exbody {
background-color:#ff0000;
background-repeat:no-repeat;
background-size:500px 370px;}
#wrap1,#wrap2 {
width:200px; height:120px;
margin:10px; padding:10px;
float:left;
font-size:1.5em;
background-color:black;
```

```
color:white;
z-index:100;}
#wrap1 {
background-color:rgba(0,0,0,0.6);}
#wrap2 {
opacity:0.6;}
</style>
<body id=«exbody»>
<div id=«wrap1»>Прозрачность задана с помощью RGBA.</div>
<div id=«wrap2»>Прозрачность задана с помощью opacity.</div>
</body>
</html>
```

Подобно RGBA цвет сразу вместе с прозрачностью можно задавать, используя **HSLA**.

```
#wrap1
{background-color:hsla(0,100%,0%,0.6);}
```

2.16.3. Границы

Создание элементов со сглаженными углами. С помощью нового CSS 3 свойства ***border-radius*** можно сглаживать углы элементов.

```
<html>
<head>
<style type='text/css'>
#el1,#el2,#el3,#el4 {
float:left;
border:2px #000000 solid;
padding:10px;
width:300px;
height:100px;
margin:20px;
background-color:#85004B;
```

```
color:#FFF;
font-weight:bold;}
#el1 {
border-radius:5px;}
#el2 {
border-radius:10px;}
#el3 {
border-radius:20px;}
#el4 {
border-radius:15px;}
</style>
</head>
<body>
<p id='el1'>Я - первый элемент со сглаженными углами
<span style='text-decoration:underline'> border-
radius:5px</span></p>
<p id='el2'>Я - второй элемент со сглаженными углами
<span style='text-decoration:underline'> border-
radius:10px</span></p>
<p id='el3'>Я - третий элемент со сглаженными углами
<span style='text-decoration:underline'> border-
radius:20px</span></p>
<p id='el4'>Я - четвертый элемент со сглаженными углами
<span style='text-decoration:underline'> border-
radius:15px</span></p>
</body>
</html>
```

Данное свойство может применяться не ко всем углам элемента, а только к определенным:

— ***border-top-left-radius*** делает сглаженным только верхний левый угол элемента;

— ***border-top-right-radius*** делает сглаженным только верхний правый угол элемента;

— ***border-bottom-left-radius*** делает сглаженным только нижний левый угол элемента;

— ***border-bottom-right-radius*** делает сглаженным только нижний правый угол элемента.

```
#el1 {
border-top-left-radius:20px;}
#el2 {
border-top-right-radius:20px;}
#el3 {
border-bottom-left-radius:20px;}
#el4 {
border-bottom-right-radius:20px;}
```

Добавление тени. С помощью свойства ***box-shadow*** можно добавить к элементам страницы тени. Добавляя тени к элементам, дизайн страницы становится более «естественным» (то есть имитирующим реальный мир, так как объекты в нем отбрасывают тени).

Тень может быть внешней и внутренней. Внешние тени создают эффект приподнятости элемента над остальным содержимым, а внутренние создают эффект вдавленности элемента.

```
#el1 {
box-shadow:4px 4px black;}
#el2 {
box-shadow:6px 6px 6px 2px black;}
#el3 {
box-shadow:0px 0px 6px 2px black inset;}
```

Цвет границы. С помощью свойства ***border-colors*** можно регулировать цвет каждого пикселя границы.

```
#el1
{
border:8px solid;
-moz-border-top-colors: #FF0000 #EB1010 #D22E2E #B03E3E;
-moz-border-right-colors: #FF0000 #EB1010 #D22E2E
#B03E3E;
-moz-border-bottom-colors: #FF0000 #EB1010 #D22E2E
#B03E3E;
-moz-border-left-colors: #FF0000 #EB1010 #D22E2E #B03E3E;
}
```

2.16.4. Шрифт

В предыдущих версиях CSS разработчики были вынуждены использовать только те шрифты, которые гарантированно установлены на компьютере пользователя, в CSS 3 разработчики могут использовать любые шрифты, которые они захотят.

Для этого необходимый шрифт нужно просто разместить на веб-сервере и подключить его с помощью свойства ***@font-face***. Подключенный шрифт будет загружен и отображен автоматически при посещении страницы пользователем.

Необходимо помнить, что браузеры IE9+, Chrome, Firefox, Opera и Safari поддерживают шрифты в формате .woff (Web Open Font Format - Открытый Формат Шрифтов Всемирной Паутины). Браузеры Chrome, Firefox, Opera и Safari также поддерживают шрифты в формате TTF и OTF, а IE в формате EOT.

```
@font-face {
font-family:opensans; /* Задаем имя шрифта */
src:url('opensans.woff')
/* Указываем местонахождение шрифта */
}
```

Для того чтобы использовать подключенный шрифт, необходимо просто добавить к желаемому элементу свойство ***font-family***, содержащее имя этого шрифта.

```
<html>
<head>
<style type='text/css'>
@font-face {
font-family:»opensans»; /* Имя шрифта */
src:url('opensans.woff') /* Местонахождение шрифта */
}
div
{
font-family:»opensans»;
font-size:1em;}
</style>
</head>
```

```
<body>
<div>Текст данного элемента написан шрифтом opensans.
</div>
</body>
</html>
```

Также имеется возможность добавить курсивное или жирное начертание к подключенному шрифту. Для этого необходимо дополнительно добавить в ***@font-face*** свойство ***font-style:italic*** или ***font-weight:bold*** и указать путь к шрифту в соответствующем начертании.

```
/* Подключаем курсивную версию шрифта */
@font-face {
font-family:»opensans»;
font-style:italic;
src:url('opensans-italic.woff');
}
/* Подключаем жирную версию шрифта */
@font-face {
font-family:»opensans»;
font-style:bold;
src:url('opensans-bold.woff');
}
```

2.16.5. Текст

С помощью свойства ***text-shadow*** можно добавить к тексту элементов тени (к тексту одного элемента может быть добавлено одновременно несколько теней). При задании тени для текста необходимо указать: величину смещения тени от текста по горизонтали и вертикали (может быть отрицательной), а также радиус размытия и цвет тени.

```
#shadow1 {
text-shadow:3px 2px #FFAE00;}
#shadow2 {
text-shadow:1px 1px 10px #FFAE00;}
```

```
#shadow3 {
text-shadow:2px 2px 2px #FFAE00, 2px 2px 15px #1435AD;}
```

В CSS 3 было добавлено новое свойство ***text-overflow***, которое позволяет указать, что должно случиться с текстом, вышедшем за пределы границ элемента.

```
#wrap1 {
text-overflow:ellipsis;
overflow:hidden;
}
#wrap2 {
text-overflow:clip;
overflow:hidden;
}
```

С помощью свойства ***word-wrap*** можно установить, что бы длинные слова, выходящие за пределы границ элемента, разделялись и переносились на новую строку.

```
#wrap2 {
word-wrap:break-word;
}
```

2.16.6. Прозрачность

С помощью CSS можно создавать прозрачные элементы и картинки. Для создания прозрачных элементов во всех браузерах кроме Internet Explorer используется свойство ***opacity:x***, где ***x*** значение которое может изменяться от 0.0 (полностью прозрачный элемент) до 1.0 (полностью непрозрачный элемент).

Для создания прозрачных элементов в Internet Explorer используется свойство ***filter:alpha(opacity=x)***, где ***x*** значение которое может изменяться от 0 (полностью прозрачный элемент) до 100 (полностью непрозрачный элемент).

```
.op1
{
opacity:0.8;
filter:alpha(opacity=80);
```

```
}
.op2
{
opacity:0.2;
filter:alpha(opacity=20);
}
.op3
{
opacity:0.5;
filter:alpha(opacity=50);
}
```

2.16.7. Столбцы

В CSS 3 появились свойства, позволяющие разбивать текст на столбцы. Данные свойства поддерживаются в браузерах IE10+ и Opera. Для браузеров Chrome и Safari перед свойством требуется добавить префикс ***-webkit***, а для Firefox префикс ***-moz***.

С помощью CSS3 свойства ***column-count*** можно указать количество столбцов, на которое необходимо разбить текст выбранного элемента.

```
<html>
<style type='text/css'>
h1 {
border-bottom:1px solid;
}
div {
column-count:3;
-moz-column-count:3;
-webkit-column-count:3;}
</style>
<body>
<h1>Титаник</h1>
<div> На Титанике имелось 8 стальных палуб, расположенных друг над другом на расстоянии 2,5– 3,2 м.
```

Самая верхняя была шлюпочная, под ней находилось семь остальных, обозначенных сверху вниз буквами от A до G. Только палубы C, D, E и F проходили по всей длине судна. Шлюпочная палуба и палуба A не доходили ни до носовой части, ни до кормы, а палуба G располагалась только в передней части лайнера – от котельных отделений до носа и в кормовой части – от машинного отделения до среза кормы. На открытой шлюпочной палубе размещались 20 спасательных шлюпок, вдоль бортов находились прогулочные палубы.

```
</div>
</body>
</html>
```

С помощью свойства ***column-gap*** можно установить величину отступа между столбцами текста.

```
div {
column-count:4;
column-gap:50px;
}
```

С помощью свойства ***column-rule*** можно задать ширину, цвет и стиль оформления пространства между столбцами.

```
div {
column-count:4;
column-rule:2px dotted #7F0055;
}
```

Свойство ***column-width*** позволяет указывать ширину столбцов текста.

```
div {
column-width:150px;
}
```

2.16.8. Трансформирование

С помощью свойства ***transform*** имеется возможность трансформировать элементы. В качестве значения данного

свойства должна указываться одна из функций трансформирования. На данный момент в современных браузерах поддерживаются только 2D трансформации, но в будущем будут также доступны и 3D трансформации.

Данное и последующие свойства работают во всех современных браузерах (IE9+, Safari, Chrome, Firefox, Opera), но для некоторых браузеров требуется добавление специальных префиксов. Для браузеров Chrome и Safari требуется префикс ***-webkit***, для браузера IE версии 9 требуется префикс ***-ms*** (для IE10 данный префикс не требуется).

С помощью функции ***translate(x,y)*** можно сместить элемент на указанное количество пикселей по горизонтали и вертикали.

```
<html>
<head>
<style type='text/css'>
#el1,#el2 {
position:absolute;
top:10px;
left:10px;
background-color:#7A005C;
color:white;
width:200px;
height:150px;
font-size:1.5em;
border:1px #000 solid;}
#el2 {
transform: translate(180px,180px);
-webkit-transform: translate(180px,180px);
/* для Chrome и Safari */
-ms-transform: translate(180px,180px); /* для IE */
}
</style>
</head>
<body>
<div id='el1'>Изначальная позиция</div>
```

```
<div id='el2'> Позиция после применения translate(180px,180px) </div>
</body>
</html>
```

С помощью функции ***rotate(градусы)*** можно повернуть элемент на указанное количество градусов по часовой стрелке.

```
#el3 {
transform: rotate(45deg);
}
#el4 {
transform: rotate(120deg);
}
```

С помощью метода ***scale(x,y)*** элемент растягивается в ширину или высоту.

```
#el6 {
transform:scale(1.3,1);
}
```

С помощью метода ***skew(x,y)*** можно скосить элемент на указанное количество градусов по горизонтали и вертикали.

```
#el7 {
transform:skew(40deg,20deg);
}
```

В табл. 2.4 приведены функции трансформирования.

Таблица 2.4

Функции трансформирования CSS 3

Функция	**Описание**
translate(x,y)	Смещает элемент от изначальной позиции по горизонтали и вертикали.
translateX(x)	Смещает элемент по горизонтали.
translateY(y)	Смещает элемент по вертикали.
scale(x,y)	Растягивает элемент по вертикали и горизонтали.
scaleX(x)	Растягивает элемент по горизонтали.
scaleY(y)	Растягивает элемент по вертикали.
rotate(градусы)	Поворачивает элемент по часовой стрелке.

Окончание табл. 2.4

Функция	Описание
skew(x,y)	Скашивает элемент по горизонтали и вертикали.
skewX(x)	Скашивает элемент по горизонтали.
skewY(y)	Скашивает элемент по вертикали.
matrix(x,x,x,x,x,x)	Совмещает все перечисленные выше методы в один.

2.16.9. Градиент

В предыдущей версии CSS градиенты можно было реализовать в виде отдельных картинок, вставляющихся как фоновые. В CSS 3 имеются встроенные свойства для создания градиентов. Так как в CSS 3 сам браузер отрисовывает градиенты, вследствие этого необходимость дополнительных запросов градиентных картинок у сервера отпадает и это позволяет увеличить скорость загрузки страниц.

Градиенты поддерживается во всех современных браузерах, но требуют добавления специального префикса. Для браузера IE10+ требуется префикс ***-ms***, для Chrome и Safari префикс ***-webkit***, для Opera префикс ***-o*** и для Firefox префикс ***–moz***.

Линейные градиенты создаются с помощью метода ***linear-gradient***, который должен указываться в значение свойства ***background***.

Для того, чтобы создать линейный градиент необходимо указать его направление (может задаваться с помощью ключевых слов или градусов) и цвета перехода.

```
<html>
<head>
<style type='text/css'>
#wrap1,#wrap2,#wrap3,#wrap4 {
float:left;
margin:5px;
width:300px;
```

```
height:150px;
padding:10px;
border:1px #000 solid;
text-decoration:underline;
}
#wrap1 {
background:linear-gradient(top,white,black);
background:-webkit-linear-gradient(top,white,black);
/* для Safari */
}
#wrap2 {
background:linear-gradient(left,white,black);
background:-webkit-linear-gradient(left,white,black);
/* для Safari */
}
#wrap3 {
background:linear-gradient(0deg,white,black);
background:-webkit-linear-gradient(0deg,white,black);
/* для Safari */
}
#wrap4 {
background:linear-gradient(270deg,white,black);
background:-webkit-linear-gradient(270deg,white,black);
/* для Safari */
}
</style>
</head>
<body>
<p>1. Примеры задания направлений градиентов с помощью ключевых слов: </p>
<div id=«wrap1»>linear-gradient(top,white,black)</div>
<div id=«wrap2»>linear-gradient(left,white,black)</div>
<br style=«clear:both;» />
<p>2. Примеры задания направлений градиентов с помощью градусов: </p>
```

```
<div id=«wrap3»>linear-gradient(0deg,white,black)</div>
<div id=«wrap4»>linear-gradient(270deg,white,black)</div>
</body>
</html>
```

Цвета перехода – это цвета, которые принимает градиент в определенных его точках, например, градиент, который плавно изменяет цвет с белого на черный, имеет белый цвет перехода в начальной точке и черный в конечной.

Линейные градиенты могут иметь неограниченное количество цветов перехода. Необходимо указывать координаты местоположения цветов с помощью % (0% подразумевает начало градиента, 100 % конец).

```
#wrap1 {
background:linear-gradient(top, white 0%, green 50%, black 100%);
}
#wrap2 {
background:linear-gradient(left, #8F04A8 0%, #7CE700 60%, #FFE100 100%);
}
```

С помощью метода ***radial-gradient*** создаются сферические градиенты. Синтаксис определения сферических градиентов очень похож на синтаксис линейных, но требует также задания формы градиента (может быть сферической или эллипсоидной).

```
#wrap1 {
background:radial-gradient(white 20%, black 40%);
}
#wrap2 {
background:radial-gradient(circle, #8F04A8 0%, #5D016D 40%, black 60%);
}
```

Повторяющиеся градиенты задаются с помощью методов ***repeating-linear-gradient*** (создает повторяющийся линейный градиент) и ***repeating-radial-gradient*** (создает повторяющийся сферический градиент). Для того чтобы создать повторяющийся

градиент, необходимо указать направление градиента, а также цвета перехода и расстояние, которое они должны занимать.

```
#wrap1 {
background:repeating-linear-gradient(50deg,white,white
5px,black 5px,black 10px);
}
#wrap2 {
background:repeating-radial-gradient(circle,#8F04A8
0%,#5D016D 40%,black 60%);
}
```

2.16.10. Переходы

С помощью ***transition*** можно создавать эффекты перехода. Данное свойство поддерживается в браузерах IE 10+, Chrome, Firefox и Opera. Для браузера Safari требуется добавить префикс ***-webkit***.

Для создания переходов необходимо указать какое свойство будет изменяться и скорость выполнения этих изменений в секундах.

```
<html>
<style type='text/css'>
#wrap1 {
border:1px #000 solid;
width:200px;
padding:10px;
font-size:1.5em;
transition: width 4s;
-webkit-transition: width 4s; /* Safari*/
}
#wrap1:hover {
width:500px;
}
</style>
<body>
```

```
<div id=«wrap1»>Наведите на меня курсор мыши.</div>
</body>
</html>
```

Для того, чтобы добавить эффект перехода к нескольким свойствам, необходимо перечислить их названия через запятую.

```
#wrap1 {
background-color:#E869AA;
color:#000;
width:200px;
transition: color 4s, width 4s, background-color 4s;
}
#wrap1:hover {
color:#FFFFFF;
width:400px;
background-color:#880045;
}
```

Плавность выполнения переходов контролируется с помощью функций смягчения. Существует несколько видов таких функций:

— linear;
— ease (функция смягчения по умолчанию);
— ease-in;
— ease-out;
— ease-in-out;
— cubic-bezier(x,x,x,x) (поведение функции контролируется переданными параметрами).

```
div {
width:230px;
transition:width 4s;
}
div:hover {
width:600px;
}

#el1 {
transition-timing-function:linear;
```

```
}
#el2 {
transition-timing-function:ease;
}
#el3 {
transition-timing-function:ease-in;
}
#el4 {
transition-timing-function:ease-out;
}
#el5 {
transition-timing-function:ease-in-out;
}
#el6 {
transition-timing-function:cubic-bezier(0.6,0.2,0.5,0.6);
}
```

В табл. 2.5 приведены свойства переходов.

Таблица 2.5

CSS 3 свойства переходов

Свойство	Описание
transition	Позволяет задать значения четырех различных свойств перехода в одном определении.
transition-property	Позволяет указать имя CSS свойства, к которому будет применен эффект перехода.
transition-duration	Позволяет указать время выполнения перехода (*по умолчанию имеет значение 0*).
transition-timing-function	Позволяет задать функцию смягчения, отвечающую за плавность выполнения перехода (*по умолчанию имеет значение 'ease'*).
transition-delay	Позволяет задать задержку перед началом выполнения перехода (*по умолчанию имеет значение 0*).

3. Размещение сайта в интернете

3.1. Веб-хостинг

После того как сайт сверстан, его передают программистам, которые «оживляют» сайт, делая его работоспособным. Когда же и этот этап пройден, сайт размещают в интернете. Для того чтобы разместить сайт в интернете, потребуется услуга, которая называется веб-хостинг.

Веб-хостинг – это услуга, позволяющая заинтересованным лицам размещать свои веб-сайты в интернете. Данный вид услуг предоставляется специальными компаниями, которые называются хостинг-провайдерами.

Оплатив веб-хостинг, провайдер предоставляет в аренду на определенный срок компьютер (или часть ресурсов компьютера), который располагается в дата-центре компании и специально настроен для размещения веб-сайтов.

На данный момент существует большое количество видов хостинга, которые подходят под те или иные сайты. Наиболее часто виды хостинга классифицируют по цене.

Бесплатный хостинг иногда предоставляется некоторыми хостинг-провайдерами (в основном зарубежными). Данный вид хостинга подходит для маленьких сайтов с низкой посещаемостью. Данный вид хостинга не подходит для коммерческих проектов даже на ранних стадиях развития. Бесплатный хостинг часто обладает рядом ограничений:

— отсутствие возможности использовать собственное доменное имя;

— отсутствие поддержки баз данных;

— отсутствие некоторого программного обеспечения;

— отсутствие технической поддержки;

— безопасность сильно ограничена.

Разделяемый хостинг является самым экономичным видом хостинга. На разделяемом хостинге веб-сайт имеет собственное доменное имя и располагается вместе с еще примерно сотней других веб-сайтов на мощном веб-сервере, чьи ресурсы

они делят между собой. Данный вид хостинга подходит для сайтов со средней посещаемостью.

VPS хостинг по цене занимает нишу между разделяемым и выделенным хостингом. Данный вид хостинга похож на предыдущий. Веб-сайт также располагается вместе с другими сайтами на мощном веб-сервере, ресурсы которого разделяются между ними. Но если на разделяемом хостинге ресурсы компьютера делятся между сайтами случайным образом (в зависимости от нагрузки веб-сайтов), то в данном случае арендуется необходимая вычислительная мощность, которая гарантированно будет предоставляться.

Выделенный хостинг является самым дорогим видом хостинга. Данный вид хостинга подходит для крупных сайтов с высокой посещаемостью. Веб-сайт, размещенный на выделенном хостинге, будет располагаться на отдельном высокопроизводительном веб-сервере и может использовать все его доступные ресурсы.

Размещение сайта в интернете может быть реализовано без услуг хостинг-провайдеров. Можно создать домашний веб-сервер и расположить на нем веб-сайт. Обычно затраты на создание работоспособного домашнего сервера намного превышают затраты на оплату веб-хостинга, поэтому данный вариант используется редко. Он обладает следующими недостатками:

— необходима установка и настройка необходимого программного обеспечения;

— необходимо следить за обновлениями ПО;

— необходимо, чтобы компьютер работал 24 часа в сутки и 7 дней в неделю;

— необходимо предусмотреть аварийное питание на случай отключения электричества;

— необходимо, чтобы компьютер был достаточно мощным;

— необходимо, обеспечить компьютер высокоскоростным соединением с интернетом.

Прежде чем приступить к выбору конкретного хостинг-провайдера, необходимо рассчитать какие характеристики от него потребуются для реализации проекта.

Объем предоставляемого места. Часто хостинг-провайдеры ограничивают объем места на жестком диске. На российском рынке хостинг-провайдеры часто предоставляют следующие объемы:

— 1 GB;
— 3 GB;
— 6 GB;
— 10 GB.

На зарубежном рынке почти все крупные хостинг-провайдеры предоставляют неограниченное место на жестком диске.

Пропускная способность. Рассчитать необходимую пропускную способность можно по следующей формуле: *средний_вес_страницы*ожидаемое_количество_просмотров_за_месяц*. Например, если средняя страница веб-сайта имеет размер 20Kb и ожидается, что страницы будут просмотрены за месяц 150000 раз, то необходимая пропускная способность будет равна: *0,02Mb*150000=3GB*.

Количество сайтов для размещения. Хостинг-провайдер может ограничить количество веб-сайтов, которые могут быть размещены на хостинге.

Базы данных. Хостинг-провайдер может ограничить количество доступных для создания баз данных. Необходимо удостовериться, что количество предоставляемых баз данных удовлетворяет нуждам сайта.

Скорость и качество соединения. Прежде чем приобретать хостинг, нужно оценить качество соединения с сайтами, которые уже пользуются услугами данного хостинг-провайдера. Также необходимо обратить внимание на доступность этих сайтов в разное время суток. Каждая минута недоступности сайта лишает сайт посетителей.

Зарубежные хостинг-провайдеры часто дают гарантию, что 99.9 % времени сервера будут находиться в рабочем состоянии.

Поддержка языков программирования. Обязательно нужно, чтобы хостинг поддерживал языки программирования, которые используются в проекте. Примеры языков программи-

рования, которые могут использоваться для реализации проектов: CGI, Fast CGI, PHP, Ruby on Rails, Perl, Python, SSI.

3.2. Доменное имя

Доменное имя – это уникальное имя, которое идентифицирует веб-сайт во всемирной паутине (например, доменное имя КемТИППа – kemtipp.ru). Доменное имя – это визитная карточка сайта и от его качества частично зависит и посещаемость ресурса.

Факторы, на которые необходимо обратить внимание при выборе доменного имени:

— длина – необходимо, чтобы длина доменного имени не превышала 10 символов.

— доменная зона – для русскоязычных сайтов наиболее подходящей является зона .ru (также может использоваться .рф и международные зоны .com, .net, .org).

— ясность - доменное имя должно давать пользователю хотя бы примерное представление о том, чему посвящен ресурс.

— созвучность – необходимо, чтобы пользователь мог легко читать и произносить доменное имя.

Регистрация доменного имени. Зарегистрировать доменное имя можно у регистраторов доменных имен. Рекомендуемые регистраторы: Reg.ru или Nic.ru. Это крупнейшие доменные регистраторы в стране.

Регистрация доменного имени предполагает аренду, а не покупку доменного имени, так что каждый год необходимо будет продлевать регистрацию (регистрация доменного имени в зоне .ru стоит: ~600руб/год, а продление стоит: ~450руб./год).

3.3. Выбор хостинг-провайдера

Имеется ощутимая разница цен на российском и зарубежном рынке (цены на зарубежном рынке значительно ниже из-за более жесткой конкуренции и более низких закупочных цен на

серверное оборудование, но из-за повышения курса доллара эта разница заметно снизилась).

В качестве примера приведены характеристики хостинга от компании Hostgator. Hostgator - это Американский хостинг-провайдер, функционирующий с 2002 года. Hostgator входит в десятку крупнейших хостинг-провайдеров мира.

— неограниченное дисковое пространство под веб-сайт;

— неограниченная пропускная способность;

— управление хостингом через Cpanel, которая имеет русский интерфейс;

— неограниченное количество доменов и сайтов (кроме тарифа Hatchkling);

— поддержка: CGI, Fast CGI, PHP, Ruby on Rails, SSH, Perl, Python, SSI, Cron, FrontPage, Curl;

— неограниченное количество почтовых ящиков;

— неограниченное количество баз данных;

— возврат денег в течении 45 дней в случае, неудовлетворенности качеством предоставляемых услуг.

3.4. Шаблоны сайтов

Шаблоном сайта обычно называют готовый HTML шаблон (CSS шаблон), т.е. набор веб-страниц, выполненных в едином стиле, созданном при помощи CSS, также шаблоном могут называть дизайн сайта в графическом формате (например, cdr или psd).

Шаблонов огромное множество, их можно скачать бесплатно, можно купить, а можно изготовить свои на продажу.

Вот список наиболее популярных ресурсов с набором бесплатных HTML шаблоном:

— bestfreetemplates.info/allfree.php;

— os-templates.com/;

— websitetemplatesonline.com/free-templates.html;

— templates.arcsin.se/demo/emplode-website-template/.

При написании шаблона с нуля, удобно пользоваться генераторами CSS и HTML кода, которые автоматически создают начальный каркас сайта:

— wdevblog.net.ru;
— inknoise.com;
— csscreator.com;
— blog.html.it;
— maxdesign.com.au;
— cssplay.co.uk;
— thenoodleincident.com;
— developer.yahoo.com/yui/grids;
— dynamicdrive.com;
— code-sucks.com;
— free-css.com;
— csseasy.com;
— blueprintcss;
— csstemplater.com;
— yvoschaap.com.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт. – М: Эксмо, 2015. – 465 с.

2. Дронов, В.В. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В.В. Дронов. – СПб.: БХВ-Петербург, 2014. – 430 с.

3. Кисленко, Н.П. HTML. Самое необходимое / Н.П. Кисленко. – СПб.: БХВ-Петербург, 2012. – 360 с.

4. Комолова, Н.М. HTML, XHTML и CSS / Н.М. Комолова. – СПб.: Изд-во «Питер», 2012. – 513 с.

5. Ташков, П.А. Веб-мастеринг на 100 %: HTML, CSS, JavaScript, PHP, CMS, графика, раскрутка / П.А. Ташков. – СПб.: Питер, 2010. – 650 с.

УЧЕБНОЕ ИЗДАНИЕ

Шафрай Антон Валерьевич

ВЕБ-ДИЗАЙН В МАРКЕТИНГЕ УПАКОВКИ

Учебное пособие

для студентов направления подготовки 29.03.03.
«Технология полиграфического и упаковочного производства»

Редактор *А.В. Дюмина*
Технический редактор *О.П. Долгополова*
Художественный редактор *О.П. Долгополова*

ЛР № 020524 от 02.06.97
Подписано в печать 31.10.2016. Формат 60x84$^{1/16}$
Бумага типографская. Гарнитура Times New Roman
Уч.-изд.л. 6,6. Тираж 100 экз.
Заказ № 77.

Оригинал-макет изготовлен в лаборатории множительной техники
Кемеровского технологического института пищевой промышленности (университета)
650002, г. Кемерово, ул. Институтская, 7

ПЛД № 44-09 от 10.10.99
Отпечатано в лаборатории множительной техники
Кемеровского технологического института пищевой промышленности (университета)
650002, г. Кемерово, ул. Институтская, 7