

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. И.РАЗЗАКОВА**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Программное обеспечение компьютерных систем»

Основы web-проектирования

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ
ЛАБОРАТОРНЫХ РАБОТ ПО РАЗДЕЛУ HTML, CSS**

**Для студентов направления
710400 «Программная инженерия»**

Бишкек – 2015

«Рассмотрено»
На заседании кафедры
«Программное обеспечение
компьютерных систем»
Прот.№20 от 20.06.2015г

«Одобрено»
Учебно-методическим советом
ИДО и ПК
Прот. № 6 от 09.09.2015г.

УДК 681.3.01

Составители: Турсалиева Э.Н., Садралиева Р.А.

Основы web-проектирования. Методические указания к выполнению лабораторных работ по разделу HTML, CSS /КГТУ им. И. Раззакова; Сост.: Турсалиева Э.Н., Садралиева Р.А. / - Б.: ИЦ «Текник», 2015. - 40 с.

Представлены краткие теоретические сведения
Предназначено для студентов специальности «Программная инженерия»
всех форм обучения.

Ил. 12. Библиогр. 5 названий.

Рецензент: доцент кафедры «ПОКС», к.т.н. Раматов К.С.

Введение

Веб дизайн-это создание сайта с нуля под ключ, то есть выполнение всех шагов, которые требуются для создания полностью функционирующего сайта:

- **техническое задание**, а именно утверждённый план, по которому будет создаваться сайт, для чего и кого будет сайт, его объёмы, функционал, структура, вид

- **юзабилити сайта**, удобство его использования пользователями, структура веб страниц, решается, как сайт должен работать

- **оформление сайта**, производится в графическом редакторе, создаются элементы украшения и оформления сайта

- **верстка**, готовые картинки нарезаются, и встраиваются в каркас сайта с помощью HTML и CSS

- **вебмастеринг**, внедрение сайта в интернет сеть, покупка хостинга, домена, продвижение и оптимизация его в поисковых системах

Как создать сайт самому:

чтобы создать сайт самому, вам понадобится:

- браузер
- обязательное знание основ HTML и CSS, основы PHP - MySQL, Java Script.

- умение пользоваться каким либо веб редактором (например Notepad++ или Adobe Dreamweaver)

- умение работать в графическом редакторе, (например Paint или Photoshop)

- знание и умение работать с Ftp менеджером

- реальное желание **создать свой сайт**, хорошо развитая фантазия и чувство вкуса, готовность много писать и учиться на ошибках, а не думать, что всё будет быстро и сразу

Какой сайт можно создать:

- Сайт визитка (краткая информация о вашей компании, услугах, которые вы предоставляете, товарах, которые вы продаёте, биография и контакты)

- Интернет магазин (информация о товарах или услугах, которые можно заказать через интернет)

- Информационный портал (веб сайт с большим количеством разнообразных ресурсов, сервисов уникальным контентом и сложной структурой)

- Блог (интернет дневник, автор постоянно добавляет свои статьи, на разнообразные темы, а гости сайта могут комментировать и обсуждать их)

- Социальная сеть (сайт с регистрацией, заполнением анкетных данных, по которым вас могут найти знакомые).

- При создании Web-страниц следует придерживаться современных интернет-стандартов. При этом нужно полностью отказаться от устаревших и закрытых интернет-технологий, как не укладывающихся в современную парадигму Web-дизайна и зачастую не поддерживаемых всеми Web-обозревателями.

- Особое внимание нужно обратить на структуру и наполнение Web-страниц. Структура Web-страниц должна быть максимально простой, а наполнение достаточно богатым, чтобы посетитель быстро нашел нужную ему информацию. Кроме того, необходимо создавать Web-страницы так, чтобы дизайн не мешал восприятию информации.

- Web-страницы обязательно следует делать максимально доступными на любых

устройствах. Web-страницы должны быстро загружаться и выводиться на экран.

Также Web-страницы не должны требовать для отображения никакого дополнительного программного обеспечения.

Концепция Web 2.0 предусматривает четыре принципа, являющиеся "передним краем" Web-дизайна.

Принцип первый — разделение содержимого, представления и поведения Web-страницы. Здесь содержимое — это информация, которая выводится на Web-странице, представление описывает формат вывода этой информации, а поведение — реакцию Web-страницы или отдельных ее элементов на действия посетителя. Благодаря их разделению мы сможем править, скажем, содержимое, не затрагивая представление и поведение, или поручать создание содержимого, представления и поведения разным людям.

Принцип второй — подгружаемое содержимое. Вместо того чтобы обновлять всю Web-страницу в ответ на щелчок на гиперссылке, мы можем подгружать только ее часть, содержащую необходимую информацию. Это сильно уменьшит объем передаваемой по сети информации (сетевой трафик) и позволит выполнять какие-либо действия с данными после их подгрузки.

Принцип третий — генерируемое содержимое. Какая-то часть Web-страницы может не загружаться по сети, а генерироваться прямо на месте, в Web-обозревателе. Так еще сильнее сократится сетевой трафик.

Принцип четвертый — семантическая разметка данных. Она позволит нам связать выводимые на Web-страницу данные согласно каким-либо правилам. Например, мы можем семантически связать страницы справочника по HTML, и посетитель, загрузив какую-либо страницу, сможет сразу же перейти на связанные с ней страницы, содержащие дополнительные или родственные сведения.

Обзор систем управления контентом (Content Management System)

Сегодня большинство вебмастеров во время создания собственных сайтов используют те или иные системы управления контентом – CMS. Главная задача, которая стоит перед каждым из них, - это выбор именно той CMS, которая будет максимально подходить для достижения целей и задач

конкретного проекта. Правильный выбор движка для сайта, портала или интернет магазина может обеспечить его динамическое развитие. Современная система управления сайтом обязана обеспечить возможность быстро нарастить функционал площадки, к примеру, это может быть форум, специальные фото или видео галерея и многое другое. Процесс добавления на сайт новых материалов должен быть простым и быстрым.

Движок **WordPress**, бесспорно, является самой популярной бесплатной CMS во всем мире. Исторически сложилось, что она создавалась как движок для блогов, однако современные версии позволяют существенно расширить функционал сайтов, созданных на основе данной платформы. Процесс установки WordPress достаточно прост. Его очень легко освоить и применять по своему усмотрению в дальнейшем. Существенным плюсом данного движка является наличие большого количества тем, которые можно использовать для оформления внешнего вида и функционального наполнения сайта. Создание любого проекта на базе данной системы не вызывает особых трудностей и не требует от пользователя серьезных специальных знаний. Она может считаться идеальным вариантом для начинающих вебмастеров. Популярнейшая CMS **Joomla** является одной из самых гибких бесплатных систем. С ее помощью могут строиться небольшие сайты и огромные многофункциональные проекты. Процесс установки данной системы полностью автоматизирован и максимально упрощен. Количество разных тем для оформления сайта достаточно велико и разнообразно. Весь спектр необходимых настроек можно осуществить из административной панели. Большим плюсом данной CMS является наличие большого количества расширений, которые могут быть установлены с использованием специального инсталлятора. Еще одним большим достоинством системы является ее гибкость, которая достигается благодаря наличию большого количества расширений. На Joomla создаются большие новостные порталы, качественные фото и видео галереи, интернет магазины, специализированные каталоги и доски объявлений, различные социальные сети и многое другое.

Последнее время пользуется серьезной популярностью система **Drupal**. Процесс ее установки достаточно прост. Однако интеграция новых шаблонов потребует начальных знаний и навыков программирования. Для администрирования сайтов, созданных на базе Drupal, также потребуются определенные знания и навыки. Добавление материалов можно осуществлять при использовании визуального редактора. Гибкость системы обеспечивает большое число различных плагинов, которые позволяют создавать разноплановые сайты разных уровней сложности. Три описанные бесплатные CMS системы являются самыми распространенными среди вебмастеров рунета.

World Wide Web (WWW, Web или Веб) – информационное пространство, в котором элементы, называемые ресурсами, адресуются глобальными идентификаторами, которые называются унифицированными идентификаторами ресурсов - Uniform Resource Identifiers (URI).

- параметры - строка запроса с передаваемыми на сервер (методом GET) параметрами. Разделитель параметров — знак &. Пример: ?параметр_1=значение_1&параметр_2=значение_2&параметр3=знач_3
- якорь - идентификатор «якоря», ссылающегося на некоторую часть (раздел) открываемого документа.

Взаимодействие. Веб-агенты взаимодействуют посредством стандартизованных протоколов, обеспечивающих передачу сообщений и данных. Сетевые протоколы Веб, такие как HTTP, FTP, SOAP, NNTP, и SMTP, основаны на обмене сообщениями. Сообщение может включать метаданные о ресурсе (например заголовки HTTP), данные сообщения и сообщение как таковое (например в разделе "Transfer-encoding" заголовка HTTP). Веб-агенты могут использовать URI для доступа к ресурсу, на который он ссылается, и этот процесс называется разыменованием URI. Далее возможен доступ различными способами в зависимости от протокола. Например для протокола HTTP это может быть запрос представления ресурса (команды HTTP GET или HEAD), добавление или модификация представления ресурса (команды HTTP POST или PUT) или удаление некоторых или всех представлений ресурса (команда HTTP DELETE). Поскольку URI содержит указание протокола

Форматы. Форматы данных (например XHTML, RDF/XML, SMIL, XLink, CSS, PNG) реализуют соглашение о представлении данных. Первым форматом данных в Веб был HTML. Со времени появления Веб число форматов существенно увеличилось, однако не существует формального ограничения на количество форматов. Любой разработчик имеет возможность создать свой формат, но это должно быть тщательно обосновано.

Некоторые сведения о SGML, XML

Формат SGML(Standard Generalized Markup Language), разработан организацией Open GIS Consortium и стандартизован ISO: «ISO 8879:1986 Information processing—Text and office systems—Standard Generalized Markup Language (SGML)»

Изначально SGML был разработан для совместного использования машинно-читаемых (т.е. чётко структурированных) документов в больших правительственных и аэрокосмических проектах. Он широко использовался в печатной и издательской сфере, но его сложность затруднила его широкое распространение для повседневного использования.

Основные части документа SGML:

- SGML-декларация — определяет, какие символы и ограничители могут появляться в приложении;
- Document Type Definition — определяет синтаксис конструкций разметки. DTD может включать дополнительные определения, такие, как символьные ссылки-мнемоники;
- Спецификация семантики, относится к разметке — также даёт ограничения синтаксиса, которые не могут быть выражены внутри DTD;
- Содержимое SGML-документа — по крайней мере, должен быть корневой элемент.

DTD (Document Type Definition — определение типа документа) — язык описания структуры SGML- (и в частности XML)-документа. Разработан для того, чтобы обеспечить возможность формальной проверки правильности структуры любого составленного документа в соответствии с заданной схемой.

Спецификация в формате DTD считается устаревшей по ряду причин: 1. используется отличный от XML синтаксис.

2. отсутствие типизации узлов.

Несмотря на то, что DTD еще применяется, ему на смену пришел стандарт консорциума **W3C XML Schema**.

Пример очень простого XML DTD, описывающего список людей:

```
<!ELEMENT people_list (person*)>
  <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA) >
  <!ELEMENT birthdate (#PCDATA) >
  <!ELEMENT gender (#PCDATA) >

  <!ELEMENT socialsecuritynumber (#PCDATA) >
```

Начиная с первой строки:

Элемент `<people_list>` элемент содержит любое число элементов `<person>`. Знак `<*>` означает что возможно 0, 1 или более элементов `<person>` внутри элемента `<people_list>`.

Элемент `<person>` содержит элементы `<name>`, `<birthdate>`, `<gender>` и `<socialsecuritynumber>`. Знак `<?>` означает что элемент необязателен. Элемент `<name>` не содержит `<?>`, что означает что элемент `<person>` обязательно должен содержать элемент `<name>`.

Элемент `<name>` содержит данные. Элемент `<birthdate>` содержит данные. Элемент `<gender>` содержит данные.

Элемент `<socialsecuritynumber>` содержит данные.

Пример XML-документа использующего этот DTD: `<?xml version="1.0" encoding="UTF-8"?>`

```
<![DOCTYPE] people_list SYSTEM "example.dtd">
```

```
<people_list> <person>
  <name>Fred Bloggs</name> <birthdate>27/11/2008</birthdate>
  <gender>Male</gender>
  </person> </people_list>
```

Если будет составлен XML-документ, который в описании персоны, например не содержит поле `name`, этот документ можно отбросить по причине несоответствия схеме.

Отметим, что сейчас DTD считается устаревшей и подлежащей замене на XML Schema. Среди основных недостатков DTD называется несоответствие формата языку XML, а также отсутствие типизации элементов и атрибутов.

Общие сведения о HTML, XHTML

Основные элементы разметки

Язык HTML был разработан британским учёным Тимом Бернерсом-Ли приблизительно в 1989—1991 годах в стенах Европейского совета по ядерным исследованиям в Женеве (Швейцария). HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки. HTML успешно справлялся с проблемой сложности SGML путём определения ограниченного набора структурных и семантических элементов — дескрипторов (или тегов). Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. Мультимедийные возможности были добавлены позже.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов).

Ключевые даты и стандарты:

- RFC 1866 — HTML 2.0, одобренный как стандарт 22 сентября 1995 года;
- HTML 3.2[1] — 14 января 1997 года - W3C;
- HTML 4.0[2] — 18 декабря 1997 года;
- HTML 4.01[3] (изменения, причём более значительные, чем кажется на первый взгляд) — 24 декабря 1999 года;
- ISO/IEC 15445:2000[4] (так называемый ISO HTML, основан на HTML 4.01 Strict) — 15 мая 2000 года.
- HTML 5

В настоящее время заменяется более строгим XHTML. HTML 5 будет основан на XHTML.

XHTML (англ. Extensible Hypertext Markup Language — расширяемый язык разметки гипертекста) — семейство языков разметки веб-страниц на основе XML, повторяющих и расширяющих возможности HTML 4.

XHTML 1.0 Strict:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
XHTML 1.0 Frameset:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.0 Mobile:
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile
1.0//EN" "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
XHTML 1.1:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Структура HTML разметки

Схематично разметку любой страницы можно представить следующим образом:

```
<!Спецификация DOCTYPE> <html>
<head (Заголовок)> [<Надпись заголовка>] [<Стили>] [<Скрипты>]
</head>
<body (Тело страницы)> [<Разметка страницы>]
</body> </html>
```

Спецификация документа должна быть указана, поскольку в противном случае браузер не будет иметь возможности корректно интерпретировать разметку. В различных браузерах по умолчанию отображение различно!

В соответствии с терминологией SGML, html-страница является документом, а элемент html – корневым элементом.

Каждый элемент может быть представлен одним или двумя тегами и содержать или не содержать внутри себя другие элементы.

Теги выделяются специальными знаками < >, причём открывающий тег выглядит как <имя элемента>, а закрывающий как </ имя элемента>. В случае XHTML, используется тег вида <имя элемента /> для указания элементов без закрывающих тегов.

Атрибут характеризует элемент и имеет формат <имя элемента атрибут1="123">. Существуют атрибуты специального вида: id – уникальный идентификатор элемента для всего документа, class – имя класса элемента, которое используется для разметки таблицами стилей.

Основные элементы разметки

Рассмотрим некоторые наиболее часто используемые элементы разметки.

Элемент html

Элемент html является обязательным для любой страницы. Определяет границы html-документа.

Элемент head

Не является обязательным. Представляет собой контейнер, в котором размещаются элементы заголовка документа.

Внутри элемента head могут быть определены следующие элементы:

Таблица 1 допустимые элементы внутри head.

Тэг	Описание
<title>	Определяет заглавие документа. Обычно браузеры отображают этот заголовок в заголовке окна.
<base />	Определяет базовый адрес или адрес по умолчанию для всех ссылок на странице. Ссылки на странице, содержащие относительный путь будут дополняться базовым адресом. При перемещении такого документа все ссылки будут сохранены.
<link />	Определяет связи между документом и внешними ресурсами. Например подключение стилей отображения.
<meta />	Определяет метаданные о документе. Среди них могут быть как дополнительные ключевые слова, используемые для поиска, так и указание кодировки страницы.
<script>	Определяет скрипты, выполняемые на стороне браузера.
<style>	Определяет информацию о стилях отображения в документе. В отличии от <link> эти стили будут внедрены в документ.

Следует отметить, что для страниц на русском языке обязательно необходимо указывать кодировку текста. Примеры:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

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

```
/> <meta http-equiv="Content-Type" content="text/html; charset=koi8-r" />
```

В настоящее время целесообразно использовать только UNICODE-кодировки, например utf-8.

Различная форма записи тэгов <title> и <base /> означает то, что в XHTML не парные теги должны быть завершены символами “/>”. В то же время теги типа <title> имеют парный закрывающий тег </title>.

HTML-5 допустимо использование короткой формы: <meta charset="utf-8">

Элемент body

Не является обязательным. Представляет собой контейнер, в котором размещаются элементы, описывающие тело (или содержимое) документа.

Исторически этот элемент имел дополнительные атрибуты, позволявшие выбрать цвет фона, шрифта, фоновый рисунок, однако эти атрибуты устарели и их использование запрещено.

Элемент содержит ряд стандартных атрибутов, таких как класс, стиль, заголовок элемента, язык. А также атрибуты-события, среди которых onload (загрузка страницы), onunload (закрытие страницы), а также события мыши и клавиатуры.

Некоторые элементы разметки текста

Рассмотрим еще несколько часто используемых элементов, которые применяются для разметки текста.

Таблица 2 Некоторые элементы разметки текста.

Тэг	Описание
<p>	<p>Параграф. Позволяет указать область текста, относящуюся к одному параграфу. Параграфы при отображении обычно отделяются увеличенным отступом. В рамках параграфа игнорируются переводы строк в исходном тексте. Формат переноса может быть задан стилями, но не имеет отношения к исходной разметке</p> <p><i>Пример:</i> <p>Некоторый текст.</p> <i>Будет выведено:</i> Некоторый текст.</p>
 	<p>Перевод строки в любом месте текста, включая параграфы. Параграф при этом не разрывается.</p> <p><i>Пример:</i> <p>Некоторый
текст.</p> <i>Будет выведено:</i> Некоторый текст.</p>
<pre>	<p>Устаревший, но широко применяемый элемент, позволяющий обеспечить вывод текста, включая все переводы строк.</p> <p><pre>Некоторый текст.</pre> <i>Будет выведено:</i> Некоторый текст.</p>
	<p>Unordered list. Позволяет разметить список, элементы которого в зависимости от значения атрибута type (устаревшего) будут отмечены как круг, квадрат или окружность. В разметке каждый элемент списка указывается тегами</p> <p><i>Пример:</i> type="disk"> Кофе Чай Молоко </p> <p><i>Будет выведено:</i> <input type="checkbox"/> <input type="checkbox"/> Кофе <input type="checkbox"/> <input type="checkbox"/> Чай <input type="checkbox"/> <input type="checkbox"/> Молоко</p>

	<p>Ordered list. Позволяет разметить список, элементы которого будут отмечены порядковым номером. В разметке каждый элемент списка указывается тегами Имеются атрибуты (устаревшие) start, позволяющий указать стартовый номер, а также type, позволяющий указать тип нумерации: 1 (арабские числа), A, a (буквы латинского алфавита), I, i (римские числа).</p> <p><i>Пример:</i> Кофе Чай Молоко </p> <p>Выведено: <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Кофе <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Чай <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Молоко</p>
<h1>...<h6>	<p>Устаревшие теги, предназначенные для того, чтобы выделить заголовки соответствующего уровня. Рекомендуется использовать стили.</p>

Вставка изображений

Для вставки изображений применяется элемент .

Элемент img имеет обязательный атрибут **src**, который должен содержать URL графического файла. Любой современный браузер поддерживает отображение форматов png, jpg, gif. Рекомендуется использование формата png по причине отсутствия лицензионных ограничений.

Другой обязательный атрибут – **alt**. Его использование необходимо для того, чтобы в браузерах, которые не могут отобразить изображение, вместо изображения был выведен текст.

Обратите внимание на то, что значение атрибута **alt** также используется поисковыми системами для того, чтобы ассоциировать изображение, указанное в атрибуте **src** с текстом.

Необязательные атрибуты width и height указывают ширину и высоту изображения в пикселах или процентах. Их использование необходимо для корректного отображения страницы до того, как браузер загрузил изображение. Помните о том, что каналы связи не идеальны, а скорости порядка 10-100 килобит до сих пор встречаются в условиях плохого покрытия у беспроводных модемов, включая технологии 3G и LTE! Если не учитывать скорость загрузки страниц, пользователи, просматривающие такие страницы, будут видеть постоянно меняющуюся разметку.

Пример:

```

```

Вставка ссылок

Для вставки ссылок применяется так называемый якорный элемент <a> (anchor). Любой текст или изображение, помещенные между тегами <a> и будут отображены в браузере как элементы, чувствительные к нажатию. Обязательный атрибут – href, который содержит URL.

Примеры:

```
<a href="http://www.somesite.ru/">  
 </a>  
<a href="http://www.somesite.ru/">Некоторый текст</a>
```

Разметка таблиц

Для разметки таблиц используется элемент <table>, который является контейнером для элементов <tr> (строка), <th> (заголовок), и <td> (колонка).

Следует отметить, что элемент table часто использовался не только для оформления таблиц как таковых, а для разграничения областей отображения на странице.

Пример:

```
<table border="1"> <tr>  
<th>Наименование</th> <th>Количество</th>  
</tr> <tr>  
<td>ручка</td> <td>10</td>  
</tr> <tr>  
<td>карандаш</td> <td>20</td>  
</tr> </table>
```

Результат представлен на рисунке 1.

Наименование	Количество
Ручка	10
Карандаш	20

Таблица 3 Пример отображения таблицы

Следует обратить внимание на атрибут width, который указывает ширину таблицы в пикселах или процентах ширины устройства отображения. Атрибут border указывает толщину линии.

Элементы <td> и <th> обеспечивают разметку колонок и имеют два важных опциональных атрибута: rowspan и colspan. С помощью этих атрибутов можно указать объединение ячеек таблицы по строкам и столбцам соответственно.

Пример:

```
<table width="100%" border="1"> <tr>
  <th colspan="2">Назв. & Кол.</th> <th>Всего</th>
</tr> <tr>
  <td>Ручка</td> <td>10</td>
  <td rowspan="2">30</td> </tr>
<tr> <td>Карандаш</td> <td>20</td>
</tr> </table>
```

Результат представлен на рисунке 1.

Назв. & Кол.		Всего
Ручка	10	30
Карандаш	20	

Рис. 1 Пример отображения таблицы с объединением столбцов и строк

Элементы разметки блоков

Как уже упоминалось ранее, элемент `<table>` исторически используется не только для того, чтобы отображать таблицы как таковые, но и для разметки блоков. Этот подход применялся до того, как были разработаны таблицы стилей и в настоящее время должны быть заменены блочной разметкой, а именно элементами `<div>` и ``.

Элемент `` используется для выделения фрагментов текста, например для того, чтобы выделить цветом фрагмент внутри параграфа. Обратите внимание на то, что параграф при этом остаётся целым.

Пример:

```
<p>Некоторый текст, часть которого <span style="color:red">выделена красным</span>.</p>
```

Отображение:

Некоторый текст, часть которого **выделена красным**.

Элемент `<div>` используется для разметки блока, содержащего произвольные элементы, а не только текст. Более того, элемент `<div>` может содержать вложенные элементы `<div>`.

Пример:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html>
<head>
<style type="text/css"> #div1 {float: right;}
#div2 {width: 100px;float: left;} </style>
</head> <body>
<div id="div1"> <table border="1">
<tr>
<th colspan="2">Назв. & Кол.</th> <th>Всего</th>
</tr> <tr>
<td>Ручка</td> <td>10</td>
<td rowspan="2">30</td> </tr>
<tr> <td>Карандаш</td> <td>20</td>
</tr> </table>
</div>
<div id="div2"> <p>
Некоторый текст, часть которого <span
style="color:red">выделена красным</span>.
</p> </div>
</body> </html>
```

Результат представлен на рисунке 2.

Некоторый
текст, часть
которого
выделена
красным.

Назв. & Кол.		Всего
Ручка	10	30
Карандаш	20	

Рис. 2 Пример отображения таблицы внутри блока

Формы

Для того, чтобы пользователь имел возможность ввода данных и отправки их на сервер, разработан элемент `<form>`, который является контейнером для элементов ввода.

Пример:

```
<form action="/form_submit" method="get">
```

```
Имя: <input type="text" name="fname" /><br /> Фамилия: <input type="text" name="lname" /><br />
```

```
Пол:</form><input type="radio" name="gender" value="м"/>м <input type="radio" name="gender" value="ж"/>ж<br /> <input type="submit" value="Отправить" />
</form>
```

Результат представлен на рисунке 3.

Имя:
Фамилия:
Пол: м ж

Рис. 3 Пример отображения формы

Основным элементом внутри формы является `<input>`, причем вид поля ввода определяется атрибутом `type`. Некоторые возможные значения атрибута `type` приводятся в следующей таблице.

Таблица 4. Значения атрибута `type` тэга `input`.

Тип	Описание
<code>type="text"</code>	Текстовое поле.
<code>type="password"</code>	Текстовое поле для ввода пароля. Введенные символы маскируются символом *.

type="radio"	Селектор в виде круглой кнопки. Возможные позиции выбора ввода реализуются несколькими элементами с этим типом и одинаковым значением атрибута name. См. в примере name="gender"
type="checkbox"	Селектор в виде квадрата. В отличие от типа radio, предполагает только два состояния: пункт выделен или не выделен
type="submit"	Кнопка выполнения стандартного действия подтверждения.
type="button"	Произвольная кнопка. Форма может иметь несколько кнопок.

Устаревшие элементы форматирования

До появления таблиц стилей использовались специальные элементы модификации текста и выравнивания других элементов. В настоящее время их использование ЗАПРЕЩЕНО, однако они могут встретиться в существующей разметке и продолжают поддерживаться браузерами.

Таблица 5 Устаревшие элементы разметки

Тэг	Описание
<center>	Устанавливает выравнивание по центру. Применим для любым элементам
	Определяет гарнитуру текста, размер и цвет. Пример: This is some text!
<i>	Определяет шрифт курсив.
	Определяет жирный шрифт.

Использование валидаторов HTML и CSS

Для контроля правильности набора html-страницы и таблиц стилей CSS существуют специализированные средства.

Эталонный валидатор Validator. W3C доступен бесплатно по адресу <http://validator.w3.org/>, однако его использование предполагает, что проверяемый сайт полностью доступен в Интернет, либо проверить можно будет только одиночные файлы.

Локальную проверку правильности разметки можно осуществить при помощи плагинов для браузера Mozilla Firefox, например HTML VALIDATOR (<http://users.skynet.be/mgueury/mozilla/>). Отметим также, что Mozilla Firefox имеет встроенную консоль ошибок, в которой отображаются ошибки разметки CSS. Существуют также коммерческие продукты для проверки правильности разметки, например CSE HTML Validator for Windows - <http://www.htmlvalidator.com/>.

Рассмотрим принципы работы с HTML VALIDATOR. Браузер Mozilla Firefox 14, HTML VALIDATOR 0.9.5.1.

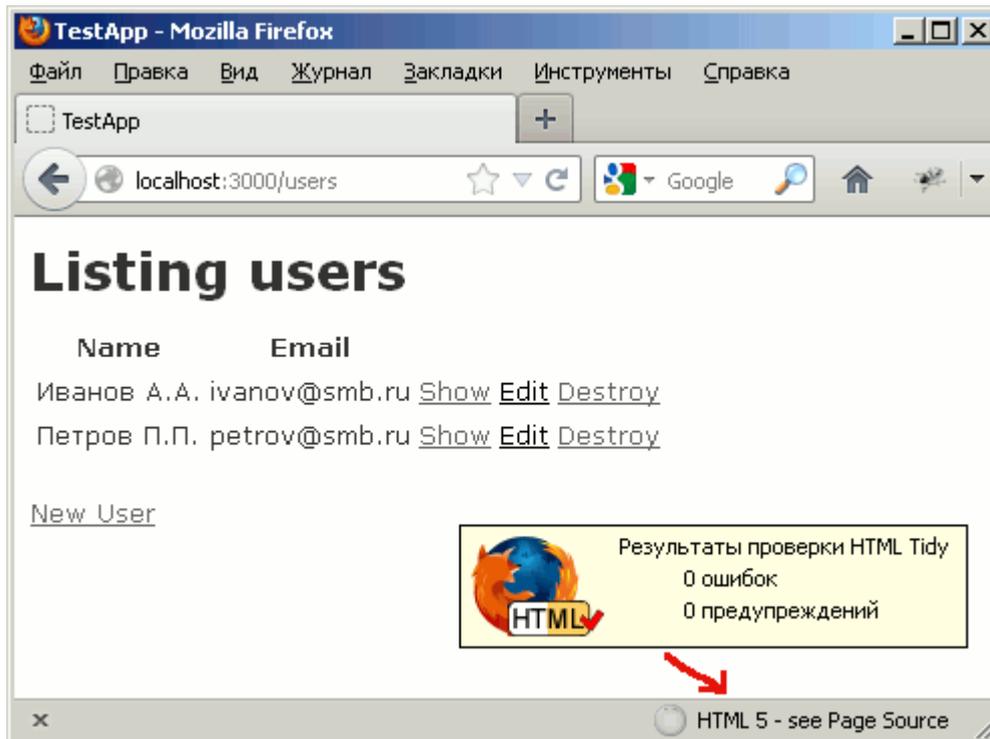


Рис.4 Информация от валидатора о корректной странице

Для примера внесём несколько ошибок в разметку HTML и CSS.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html>
<head>
<title>Проверка разметки</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/> <style
  type="text/css">
#main {
position: absolute; left: 100px;
top: 50px
border: 1px solid black; padding: 0 100px 100px 0 }
#content {
position: relative; left: 20px;
top: 10px;
border: 1px solid green; }
</style> </head> <body>
<div id="main">Главное меню:
<div id="content"><p>Некоторый текст для проверки размещения
  элемента.</div> </div>
</body> </html>
```

При визуальном просмотре страницы ошибки могут быть не обнаружены. Однако валидатор показывает, что существует одна ошибка (рисунок 10).

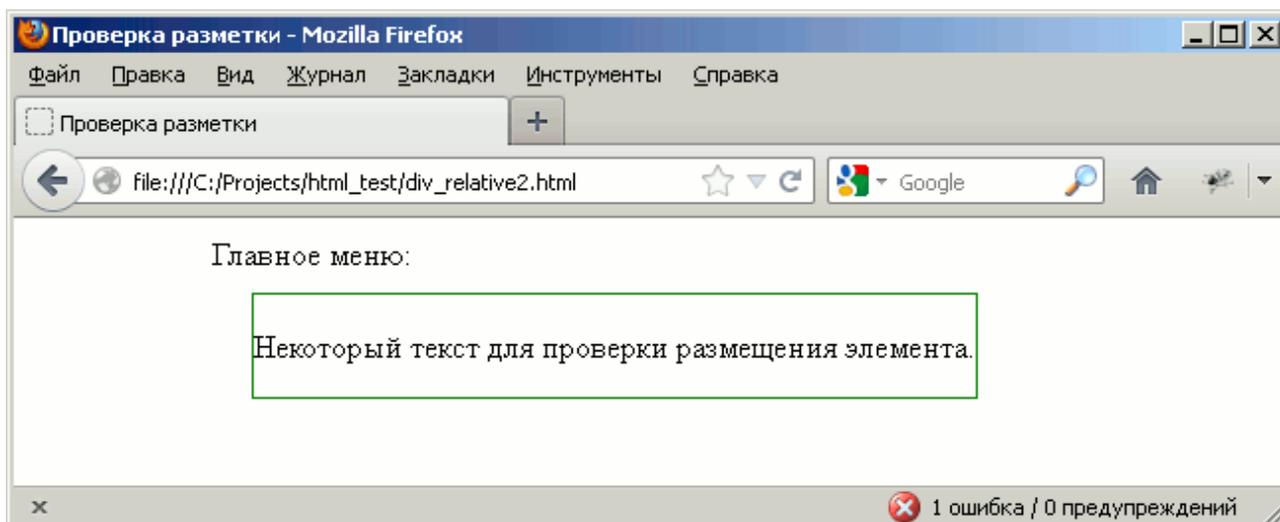


Рис. 5 Предупреждение валидатора о некорректной странице.

Необходимо выполнить двойной щелчок мышью на выданное сообщение об ошибке, после чего откроется окно просмотра исходного кода страницы.

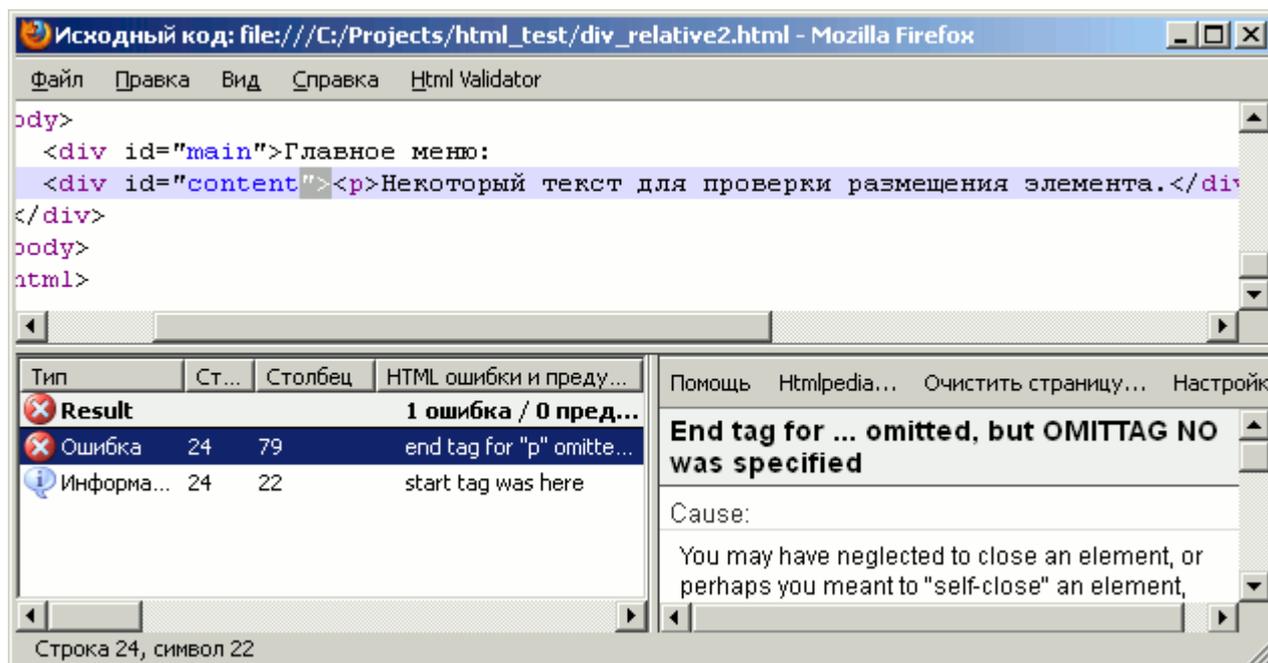


Рис. 6 Просмотр кода и сообщение об ошибке.

Помимо ошибок разметки всегда следует проверять правильность CSS. В браузере Firefox существует консоль ошибок, которая доступна через меню Инструменты/Веб-разработка/Консоль ошибок или через нажатие комбинации клавиш Ctrl+Shift+J.

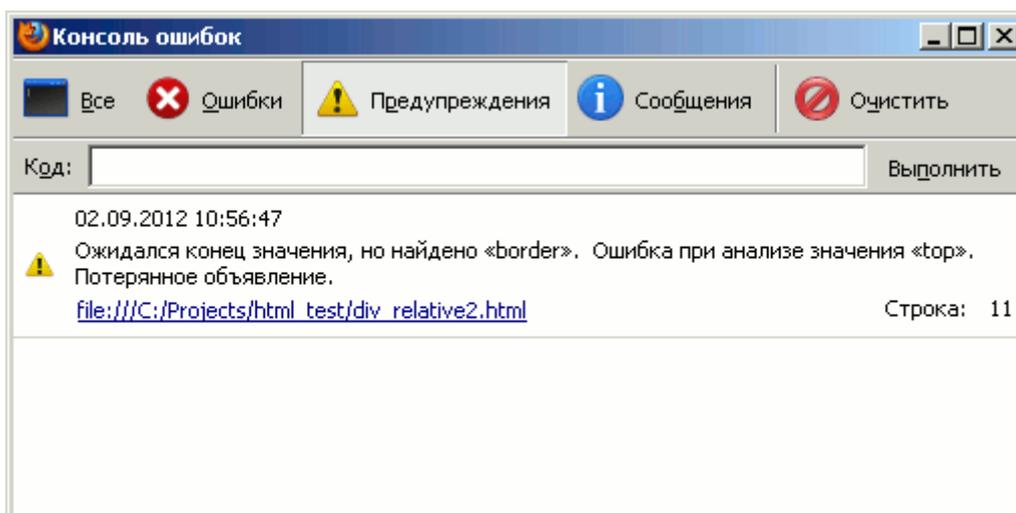


Рис. 7 Просмотр ошибок отображения браузера.

Итого, по результатам проверки выявлено:

- В разметке страницы отсутствует закрывающий тег `</p>` в строке `<div id="content"><p>Некоторый текст для проверки размещения элемента.</p></div>`
- Нарушены таблицы стилей, поскольку отсутствует разделитель `‘;` между селекторами:
`top: 50px;`
`border: 1px solid black;`

Cascading Style Sheets (CSS)

CSS (англ. Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, который написан с использованием языка разметки HTML.

Подготовкой и выпуском спецификации занимается консорциум W3C - <http://www.w3.org/Style/CSS/>

Исторически CSS появился вместе с HTML4.01 для упрощения манипулирования разметкой, поскольку именно использование стилей позволяет осуществлять групповую замену шрифта, цвета, размера и взаимного расположения элементов.

CSS используется применительно к языкам разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

Селекторы

Ключевое понятие в CSS – селектор, которое представляет собой правило для применения стиля. Браузер для каждого элемента пытается применить стиль в соответствии с заданным правилом. Стиль содержит набор свойств.

Различают простые селекторы, которые будут применены к указанному элементу (в примере к любому заголовку h1, h2, h3):

```
h1 { font-family: sans-serif } h2 { font-family: sans-serif } h3 { font-family: sans-serif }
```

Группы селекторов (эквивалентно выше приведённому фрагменту): h1, h2, h3 { font-family: sans-serif }

Селекторы класса:

.pastoral { color: green } / все элементы, имеющие class=pastoral */ Или

.pastoral { color: green } /* все элементы, имеющие class=pastoral */ А

также:

H1.pastoral { color: green } /* только элементы H1, имеющие class=pastoral */

Селекторы идентификатора ID:

h1#chapter1 { font-family: sans-serif } /* для <h1 id="chapter1">...</h1> */

#chapter1 { font-family: sans-serif } /* для любого элемента с id="chapter1" */

Селекторы атрибутов:

h1[class] { font-family: sans-serif } /* элемент имеет class */

h1[class="fancy"] { font-family: sans-serif } /* элемент имеет class="fancy" */

[title] { font-family: sans-serif } / любой элемент, имеющий заголовок */

Селектор потомков (устанавливает иерархию применения):

tr h1 { font-family: sans-serif } /* <tr><td><h1>...</h1></td></tr>*/

Псевдоклассы (особый вид динамически атрибутов, который меняются в зависимости от определенных действий):

a:link /* ссылки, которые не были посещены */ a:visited /* посещенные ссылки */

a:hover /* выделенная в данный момент ссылка */ a:active /* активные ссылки */

Шрифты

При оформлении страницы доступны следующие семейства шрифтов:

- Serif - шрифты с засечками. Обычно используются при бумажной печати. Наиболее используемый шрифт – Times.
- Sans-serif - шрифты без засечек. Подходят для заголовков. Наиболее используемые шрифты этого семейства – Arial, Helvetica, Verdana.
- Monospace - шрифт, обеспечивающий равную ширину символов. Используется для вывода примеров кода, поскольку внешний вид этого текста будет соответствовать текстовой консоли. Наиболее используемый шрифт – Courier.
- Fantasy, Cursive – декоративные и курсивные шрифты. Не рекомендуются к использованию, поскольку шрифты этой группы не обязательно присутствуют на компьютере, на котором будут просматривать html-страницу.

Выбор семейства шрифта осуществляется свойством font-family. Пример отображения семейств стилей:

```
<p style="font-family: serif">Serif: Образец текста</p>  
<p style="font-family: sans-serif">Sans-serif: Образец текста</p> <p style="font-  
family: cursive">Cursive: Образец текста</p>  
<p style="font-family: fantasy">Fantasy: Образец текста</p>  
<p style="font-family: monospace">Monospace: Образец текста</p>
```

Результаты отображения в разных браузерах представлены на рисунках 5 и 6.

Serif: Образец текста

Sans-serif: Образец текста

Cursive: Образец текста

Fantasy: Образец текста

Monospace: Образец текста

Рис. 8 Пример отображения шрифта в Mozilla Firefox 5

Serif: Образец текста

Sans-serif: Образец текста

Cursive: Образец текста

Fantasy: Образец текста

Monospace: Образец текста

Рис. 9 Пример отображения шрифта в MS Internet Explorer 9

Обратите внимание на то, что семейство Fantasy не совпадает по отображению. Кроме того, различен размер шрифта при отображении по умолчанию, который зависит от системных настроек браузера. Более того, для каждого семейства шрифтов, различные браузеры в различных операционных системах могут иметь различные конкретные шрифты, соответствующим этим семействам.

Свойство font-family может содержать перечисление шрифтов: p{font-family:"Times New Roman", Times, serif;}

В этом случае браузер последовательно будет пытаться найти соответствующий шрифт в системе. Если конкретный шрифт не будет найден

(в примере "Times New Roman", Times), то будет использован шрифт, назначенный для указанного (serif) семейства по-умолчанию.

Возможно указать начертание шрифта с использованием свойства font-style. Допустимые значения:

font-style: normal | italic | oblique | inherit

normal – обычное начертание, italic – курсив (имитация рукописного шрифта), oblique – наклонный шрифт (образован наклоном обычного).

Размер шрифта задаётся с помощью свойства font-size. Возможные значения:

- Относительный размер – константы: larger и smaller.
- Абсолютный размер – константы xx-small, x-small, small, medium, large, x-large, xx-large.
- Размер в процентах от шрифта родительского элемента. Указывается со знаком %).
- Размер в пикселах. Указывается с суффиксом px.
- Размер в специальных единицах: em (высота элемента, равная размеру текущего шрифта), ex (высота символа x), пункты (pt).

Толщина шрифта регулируется при помощи свойства: font-weight: normal | bold | bolder | lighter | 100, 200 .. 900 normal – обычная толщина, bold — жирный шрифт, bolder — предельно жирный шрифт, lighter — тонкий шрифт, 100 — тонкий шрифт, 400 — соответствует нормальному, 700 — жирному.

Пример:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html>
<head>
<style type="text/css"> #p1 {font-size: 100% }
#p2 {font-size: smaller} #p3 {font-size: larger} #p4 {font-size: 12px} #p5 {font-
size: 12pt} #p6 {font-size: 0.5em} #p7 {font-size: 12x}
</style> </head> <body>
<p id="p1">Образец текста</p> <p id="p2">Образец текста</p> <p
id="p3">Образец текста</p> <p id="p4">Образец текста</p> <p
id="p5">Образец текста</p> <p id="p6">Образец текста</p>
<p id="p7">Образец текста</p>
</body> </html>
```

Цвет

Свойство `color` задаёт цвет шрифта. Возможно указание по названию цвета (`red`, `green`, `lime`), так и указание точного его значения в системах `RGB`, `HSL`, а также `RGBA`, `HSLA` (добавлен канал прозрачности). Полный перечень допустимых значений см. <http://www.w3.org/TR/css3-color/>

Пример:

```
body {color:blue;} h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

Отметим, что выбор цветов, цветовой схемы является очень важным элементом создания дизайна сайта. Существуют определенные методики подбора совместимых цветов. Имеются бесплатные средства в Интернете, например <http://colorscheme.ru/>.

Фон

Фон элементов может быть задан однородным цветом, одиночным или мозаично расположенным изображением. Подробнее см. http://www.w3schools.com/css/css_background.asp

Используются следующие свойства:

`background-color` – однородный цвет константой или кодом в одной из допустимым систем цветности.

Пример: `div {background-color:#b0c4de;}`

`background-image` – фоновое изображение. Пример: `body {background-image:url('paper.gif');}`

`background-repeat` – флаг мозаичного размножения изображения. Пример:
`body {`
`background-image:url('gradient2.png');` `background-repeat:repeat-x;`
`}`

`background-attachment` – указывает, будет ли изображение смещаться при скроллинге или будет оставаться на месте.

Пример: `background-attachment:fixed;`

`background-position` – указывает позицию размещение изображения на устройстве отображения.

Пример: `body`

```
{
background-image:url('img_tree.png'); background-repeat:no-repeat; background-
position:right top;
}
```

Преобразование текста

В соответствии с <http://www.w3.org/TR/css3-text/> текст может быть подвергнут преобразованию при отображении, среди которых:

- изменение регистра букв (capitalize | uppercase | lowercase);
- изменение пробелов (collapse | preserve | preserve-breaks);
- ограничение длины строки;
- формирование переносов слов;
- форматирование текста;
- выравнивание и разреживание;
- отступы;
- декорирование.

Некоторые примеры использования:

```
footer { text-wrap: avoid; /* inherits to all descendants */ } ...
```

```
<footer>
```

```
<venue>27th Internationalization and Unicode Conference</venue> &#8226;
```

```
<date>April 7, 2005</date> &#8226;
```

```
<place>Berlin, Germany</place> </footer>
```

В узком окне текст будет отображен как:

27th Internationalization and Unicode Conference • April 7, 2005 • Berlin, Germany

Если окно еще больше сужено, то текст будет выглядеть: 27th Internationalization and Unicode

Conference • April 7, 2005 • Berlin, Germany

Но ни при каких условиях текст не будет выглядеть как: 27th Internationalization and Unicode Conference • April 7, 2005 • Berlin, Germany

Блочная модель

Блочная модель лежит в основе модели визуализации элементов и описывает прямоугольники, формирующиеся вокруг всех элементов в соответствии с их иерархией в дереве элементов документа. См. подробнее в <http://www.w3.org/TR/CSS2/box.html>

Отступы и границы

В соответствии с блочной моделью, для любого элемента имеется область самого элемента (content), внутренние поля (padding), рамка или граница (border), внешние границы (margin). Для каждой области может быть задан размер. Наличие внутреннего поля позволяет сформировать рамку на заданном расстоянии от содержимого элемента. Наличие внешнего поля позволяет установить отступ между рядом расположенными элементами. На рисунке 7 представлена схема расположения этих составных частей элемента. Обратите

внимание на то, что рамка может иметь толщину, задаваемую соответствующим свойством CSS и также участвует в расчёте внешнего размера элемента.

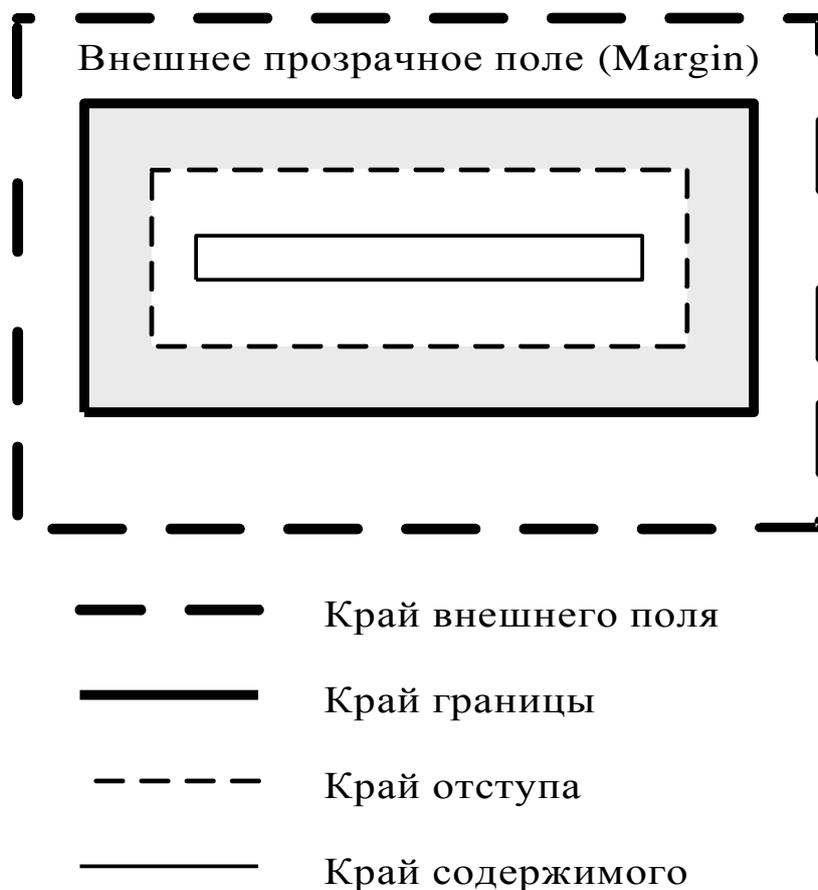


Рис. 10 Блочная модель CSS

В CSS любой элемент имеет свойства `width` и `height`, которые устанавливают размер «содержимого» в процентах, пикселах или автоматически вычисляемые.

Размер отступа задаётся свойствами `'padding-top'`, `'padding-right'`, `'padding-bottom'`, `'padding-left'` или единственным свойством `padding`, которому указывается один общий размер отступа или последовательно отступ сверху, справа, снизу, слева.

```
div { padding: 10px }  
blockquote { padding-top: 0.3em }
```

```
h1 {  
background: white; padding: 1em 2em;  
}
```

Граница представляет собой видимое обрамление элемента с указанным начертанием, цветом и толщиной. Граница может быть задана единственным

свойством `border` или отдельно для каждой стороны блока: `'border-top'`, `'border-right'`, `'border-bottom'`, `'border-left'`. Указываются толщина, тип начертания границы и цвет.

```
h1 { border: thick solid red }
#content { border-style: solid dotted }/* horiz: 'solid' vertical: 'dotted'*/ h1 { border-
width: thin } /* thin thin thin thin */
h1 { border-width: thin thick } /* thin thick thin thick */ h1 { border-
width: thin thick medium } /* thin thick medium thick */ h1 {
border-bottom: 10px; border-color: red; }
```

Таблица 6 Обозначение различных способов начертания границ.

Стиль	Эффект
<code>solid</code>	Сплошная линия
<code>dotted</code>	Ряд точек
<code>dashed</code>	Ряд штрихов
<code>double</code>	Две сплошные линии
<code>groove</code>	Углубление на холсте
<code>ridge</code>	Выступ на холсте
<code>inset</code>	Врезанный вид
<code>outset</code>	Рельефный вид
<code>hidden</code>	Скрытая граница, которая может быть отображена скриптом
<code>none</code>	Граница никогда не будет отображена

При расчёте размеров блока необходимо помнить про толщину границы.

Внешнее прозрачное поле также может быть задано либо единственным свойством `'margin'` с указанием одинакового размера границы для всех сторон, либо перечислением размеров по каждой из сторон. А также с использованием свойств `'margin-top'`, `'margin-right'`, `'margin-bottom'`, `'margin-left'` для каждой из сторон в отдельности.

```
body { margin: 2em } /* all margins set to 2em */
body { margin: 1em 2em } /* top & bottom = 1em, right & left = 2em */ body {
margin: 1em 2em 3em } /* top=1em, right=2em, bottom=3em, left=2em */ body {
margin-top: 1em; margin-right: 2em;
}
```

Позиционирование

CSS поддерживает 4 вида позиционирования:

- Статическое (`static`)
- Абсолютное (`absolute`)
- Относительное (`relative`) □□ Фиксированное (`fixed`)

Используются следующие термины:

- Нормальный поток – который означает обычное поведение браузера при отображении данных.
- Окно просмотра браузера – окно браузера, в котором отображается содержимое документа.

Элементы контейнеры могут быть размечены с использованием позиционирования. В качестве элементов-контейнеров может быть любой элемент, однако обычно используется специальный элемент div. Все элементы, включенные в элемент-контейнер, будут размещены в его границах.

Статическое позиционирование назначается у всех элементов по умолчанию и означает нормальное следование элементов. В явном виде спецификатор static применяется для перекрытия унаследованных стилей.

Абсолютное позиционирование подразумевает указание расположения относительно его блока-контейнера или корневого элемента html. При этом, как только появляется абсолютное позиционирование, элемент выпадает из нормального потока и всегда будет позиционироваться относительно контейнера, независимо от остального содержимого страницы.

Рассмотрим простейший пример позиционирования. В данном случае позиция будет совпадать с левым верхним отступом от окна отображения.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <style
type="text/css">
#content {
position: absolute; left: 200px;
top: 100px;
border: 1px solid green; }
</style> </head> <body>
<div id="content"><p>Некоторый текст для проверки размещения элемента.
</p></div>
</body> </html>
```

Рассмотрим другой пример, включающий абсолютное позиционирование относительно другого блока.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <style
type="text/css">
#main {
```

```

position: absolute; left: 100px;
top: 50px;
border: 1px solid black;
padding: 0 100px 100px 0 } #content {
position: absolute; left: 50px;
top: 20px;
border: 1px solid green; }
</style> </head>

```

```

<body>
<div id="main">Главное меню:
  <div id="content"><p>Некоторый текст для проверки размещения
элемента.</p></div>
</div> </body> </html>

```

В данном случае элемент с идентификатором content смещен относительно элемента с идентификатором main. Обратите внимание на то, что его смещение не зависит от текста, который непосредственно помещен в <div id="main">, а зависит только от заданной позиции в стиле.

Относительное позиционирование. Несмотря на то, что позиционирование контрастирует? в названии с абсолютным позиционированием, следует помнить, что смещение в этом случае вычисляется не относительно соседних элементов, а относительно нормального потока.

В следующем примере блок с идентификатором content смещен относительно нормального потока, но элемент, расположенный за ним, будет отображен так, как будто никаких изменений в потоке не было. Обратите внимание на то, что этот блок будет менять положение при изменении размеров окна браузера. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

<html> <head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <style
type="text/css">

```

```

#content {
position: relative; left: 50px;
top: 20px;
border: 1px solid green; }
</style> </head> <body>

```

```

<p>Проект Mozilla официально выпустил релиз web-браузера Firefox 15.
Кроме того, выпущен корректирующий релиз ветки с длительным сроком
поддержки - Firefox 10.0.7, в котором отмечается только исправление
уязвимостей и серьёзных ошибок.</p>

```

```

<div id="content"><p>В ближайшие дни на стадию бета-тестирования
перейдет ветка Firefox 16 и будет отделена аурора-ветка Firefox 17.</p></div>
<p>В соответствии с шестинедельным циклом разработки, релиз Firefox 16
намечен на 9 октября, а Firefox 17 на 20 ноября.</p>
</body> </html>

```

Представленный пример также позволяет легко понять различие во влиянии на нормальный поток абсолютного и относительного позиционирования. Замените здесь position на absolute.

Часто применяется комбинирование абсолютного и относительного позиционирования. Следующий пример иллюстрирует относительное позиционирование внутри блока с абсолютным позиционированием. Обратите внимание на то, каким образом браузер отображает внешний блок при изменении размера окна.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <style
type="text/css">
#main {
position: absolute; left: 100px;
top: 50px;
border: 1px solid black; padding: 0 100px 100px 0 }
#content {
position: relative; left: 20px;
top: 10px;
border: 1px solid green; }
</style> </head> <body>
<div id="main">Главное меню:
<div id="content"><p>Некоторый текст для проверки размещения
элемента.</p></div>
</div> </body> </html>

```

Фиксированное позиционирование. В отличие от абсолютного позиционирования, фиксированное позволяет закрепить элемент относительно окна просмотра, а не элемента-контейнера. Это позволяет разместить элементы, которые не будут подвергаться прокрутке, например постоянно отображаемый блок меню.

Плавающие элементы

Плавающее размещение не является схемой позиционирования. Оно было введено как средство, позволяющее получить обтекание элементов, но не для создания макета страницы.

Например следующие: стиль обеспечит отображение изображений в правой части страницы, а все остальные элементы будут размещать в свободном пространстве слева.

```
img {
float: right; padding: 15px;
}
```

Плавающее размещение иногда применяют к блокам, содержащим меню и прочие средства навигации.

Управление отображением элемента

Для управления отображением элемента используется свойство `display`.

Таблица 7 Некоторые значения свойства `display`.

Значение	Описание
<code>none</code>	Элемент не будет отображен
<code>block</code>	Элемент отображается как блок (например как <code><p></code> или как <code><h..></code>). Блок имеет отступы над и под собой, а также отделяется от следующих за ним HTML элементов.
<code>inline</code>	Режим по умолчанию. Встроенный элемент отображается как часть строки (подобно <code>span</code>) и не разрывает строку перед и после себя. Отделяется от следующих за ним элементов.
<code>inline-table</code>	Элемент является таблицей, встраиваемой в строку.
<code>list-item</code>	Элемент отображается как список и получает соответствующую метку.
<code>table</code>	Элемент отображается как таблица
<code>table-caption</code>	Элемент отображается как заголовок таблицы
<code>table-cell</code>	Элемент отображается как ячейка таблицы
<code>table-column</code>	Элемент отображается как колонка таблицы
<code>table-column-</code>	Элемент отображается как группа колонок (как <code><colgroup></code>)
<code>table-row</code>	Элемент отображается как строка таблицы
<code>table-row-group</code>	Элемент отображается как группа строк
<code>inherit</code>	Значение будет унаследовано у родительского элемента

Режимы отображения применяют в том числе для отображения блоков как таблиц:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> <html
xmlns="http://www.w3.org/1999/xhtml" lang="en-US"> <head><title>table using
divs</title>
<style type="text/css">
.div-table{display:table; border:1px solid #003399;}
.div-table-caption{display:table-caption; background:#009999;} .div-table-
row{display:table-row;}
    .div-table-col{display:table-cell; padding: 5px; border: 1px solid #003399;}
</style>
</head> <body>
<div class="div-table">
<div class="div-table-caption">This is a caption</div> <div class="div-table-row">
    <div class="div-table-col">1st Column</div> <div class="div-table-col">2nd
    Column</div>
    </div> </div>
<a href="http://linuxandfriends.com/2009/04/04/how-to-style-div-elements-as-
tables/">See source...</a>
</body> </html>
```

При этом отметим, что формирование табличного представления блоков возможно и средствами позиционирования без использования специальных значений свойства display.

Макеты. Размещение элементов по колонкам, замена фреймов и таблиц

С использованием CSS возможно размещение элементов в любом порядке, в том числе соответствующем табличному представлению, но без использования таблиц.

Существуют генераторы HTML+CSS шаблонов, например <http://csstemplater.com/>. Однако заметим, что аналогичную работу можно проделать самостоятельно или с использованием собственных шаблонов. Ниже приводится пример генерации шаблона <http://csstemplater.com/>.

```
* {
margin: 0; padding: 0;
}
html {
    height: 100%; }
body {
font: 12px/18px Arial, Tahoma, Verdana, sans-serif; width: 100%;
    height: 100%; }
a {
```

```

color: blue; outline: none;
text-decoration: underline;

}
a:hover {
    text-decoration: none; }
p {
    margin: 0 0 18px }
img {
    border: none; }
input {
    vertical-align: middle; }
#wrapper {
width: 1000px; margin: 0 auto; min-height: 100%;
height: auto !important; height: 100%;
}

/* Header -----*/ #header {
height: 150px; background: #FFE680;
}

/* Middle -----*/ #middle {
width: 100%; padding: 0 0 100px; height: 1%; position: relative;
}
#middle:after { content: '.'; display: block; clear: both;
visibility: hidden; height: 0;
}
#container { width: 100%; float: left;
overflow: hidden; }
#content {
padding: 0 520px 0 0; }

/* Sidebar Left -----*/ #sideLeft {
float: left;
margin-left: -500px; width: 250px; position: relative; background: #B5E3FF;
}

/* Sidebar Right -----*/ #sideRight {
float: left;
margin-left: -250px; margin-right: -3px; width: 250px; position: relative;
background: #FFACAA;
}

/* Footer -----*/ #footer {
width: 1000px;

```

```
margin: -100px auto 0; height: 100px; background: #BFF08E;
}
```

```
-----
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> <html
xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" /> <title></title>
<meta name="title" content="" /> <meta name="keywords" content="" />
<meta name="description" content="" />
    <link rel="stylesheet" href="style.css" type="text/css" media="screen,
projection" />
</head>

<body>

<div id="wrapper">

    <div id="header"> <strong>Header:</strong> Text.
        </div><!-- #header-->

        <div id="middle">

<div id="container"> <div id="content">
                <strong>Content:</strong> Text </div><!-- #content-
->
        </div><!-- #container-->

<div class="sidebar" id="sideLeft"> <strong>Left Sidebar:</strong> Text.
        </div><!-- .sidebar#sideLeft -->

<div class="sidebar" id="sideRight"> <strong>Right Sidebar:</strong> Text.
        </div><!-- .sidebar#sideRight -->

        </div><!-- #middle-->

        </div><!-- #wrapper -->

<div id="footer"> <strong>Footer:</strong> Text.
        </div><!-- #footer -->

</body> </html>
```

Результат представлен на рисунке 11.

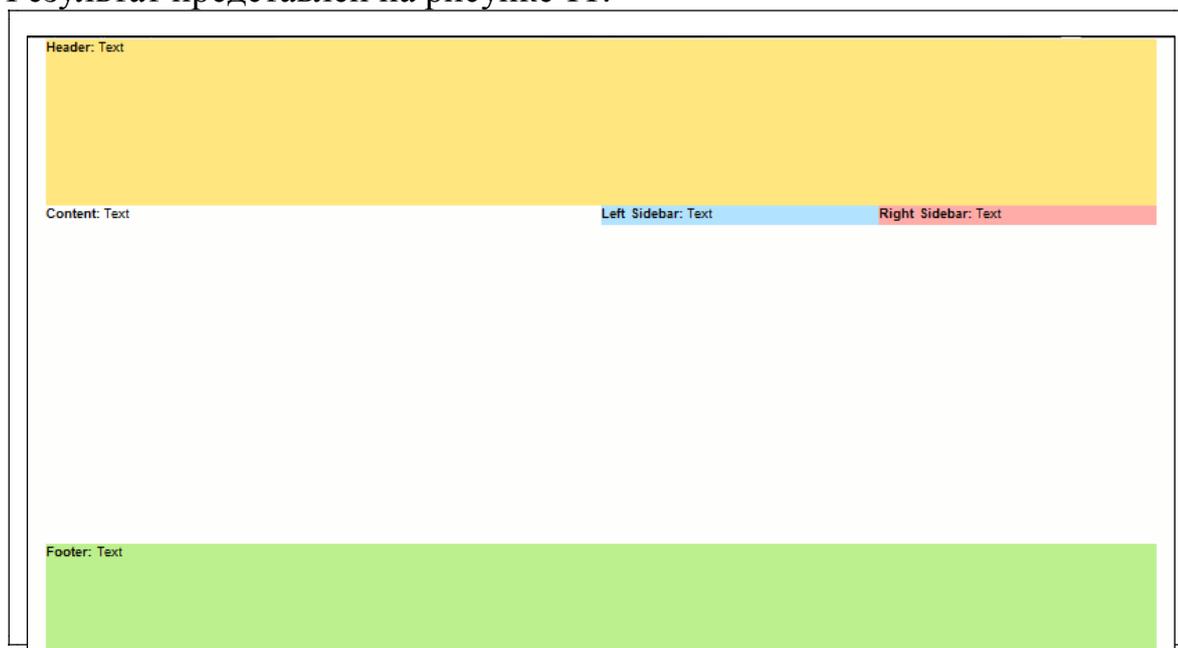


Рис. 11 Пример отображения страницы с блоками

Отладка CSS в браузере Firefox

Все современные браузеры предоставляют средства для просмотра разметки HTML, CSS и визуальной отладки таблиц стилей. Рассмотрим более подробно отладчик Firebug (реализован как дополнение) для Mozilla Firefox. Если Firebug установлен, то для того, чтобы открыть отладчик на требуемой странице, необходимо нажать клавишу F12.

В нижней части окна браузера появятся дополнительные окна. В Firebug доступны следующие закладки:

- Console – сообщения браузера об ошибках, предупреждения и отладочный вывод Javascript-программ.
- HTML – структура разметки страницы. Позволяет найти по отображенному элементу его соответствие в разметке и наоборот. Возможно интерактивное изменение стилей любых элементов.
- CSS – позволяет изменить таблицы стилей загруженной страницы.
- Script – Javascript код страниц. Позволяет просматривать код, устанавливать точки останова, выполнять пошаговую отладку и трассировку значений переменных. По-умолчанию выключено.
- DOM – просмотр и изменение значение документа по модели DOM.
- Net – просмотр данных, передаваемых между браузером и сервером.
- Предоставляется возможность просмотра заголовков HTTP, переданных данных, а также порядка передачи данных. По-умолчанию выключено.

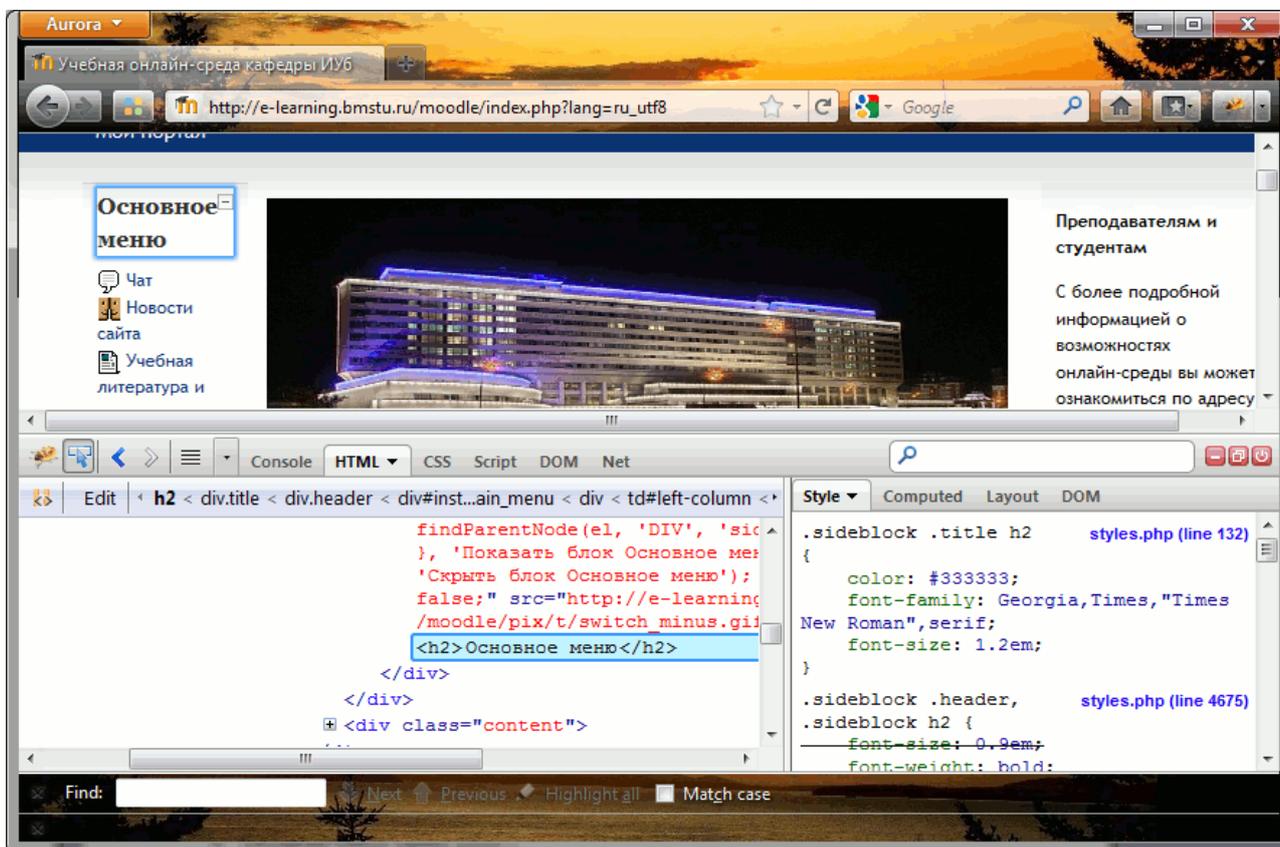


Рис. 12 Просмотр структуры и стилей в Firebug.

Обратите внимание на то, что изменения, внесенные в отладчике, не сохраняются в исходных файлах. Для того, чтобы сбросить внесенные в CSS или в HTML-разметку изменения, необходимо перезагрузить страницу.

Закладка CSS отображает файлы стилей примерно в том виде, как они были созданы. Отличаться может форматирование и способ отображения кода цвета. Этот режим удобен в том случае, если редактируется стиль для нескольких элементов сразу. Изменения такого стиля будут сразу же отображены на открытой странице.

Если же необходимо редактирование одного конкретного элемента, следует использовать закладку HTML. В закладке HTML доступны следующие вложенные закладки:

- Style (Стиль) для редактирования стиля одного конкретного элемента. Причем Firebug группирует стили, назначенные конкретно на выделенный элемент и стили, унаследованные для этого элемента.
- Закладка Computed (Скомпилированный стиль) отображает результат применения к элементу всех назначенных правил. В отличие от закладки Style, здесь приводятся конкретные значения измененных свойств.
- Layout (Макет) позволяет работать с блочной моделью разметки и отображает размеры элемента, его полей и границ в пикселах. Возможно интерактивное изменение отображаемых размеров.

Литература

1. Языки HTML и CSS для создания Web-сайтов./М. Хольцшлаг; [пер. с англ. А. Климович].–М. Триумф, 2007.–304с, ил.
2. Джон Дакетт. Основы веб-программирования с использованием HTML, XHTML и CSS.– М.: Эксмо, 2010.
3. Мейер Э. CSS - каскадные таблицы стилей. Подробное руководство, 3-е издание. - Пер. с англ. - СПб: Символ-Плюс, 2008. - 576с., ил.
4. Методические указания к выполнению лабораторной работы по курсу «Языки Интернет-программирования» , Самарев Роман Станиславович, 2013-46 с.
5. Дронов В. А. , HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. — СПб.: БХВ-Петербург, 2011. — 416 с.: ил. — (Профессиональное программирование)

Корректор *Эркинбек к. Ж.*
Редактор *Турдукулова А.К.*
Тех.редактор *Кочоров А.Д.*

Подписано к печати 23.10.2015 г. Формат бумаги 60x84¹/₁₆.
Бумага офс. Печать офс. Объем 2,5 п.л. Тираж 50 экз. Заказ 418. Цена 42,75с.
Бишкек, ул. Сухомлинова, 20. ИЦ “Текник” КГТУ им. И.Раззакова, т.: 54-29-43
е-mail: beknur@mail.ru

