Московский государственный технический университет
имени Н. Э. Баумана

Р. С. Самарев

Создание простейших HTML-страниц, валидаторы кода. Каскадные таблицы стилей CSS

Методические указания к выполнению практикума № 1 и лабораторной работы № 1 по дисциплинам «Языки интернет-программирования» и «Практикум по интернет-программированию»

Москва
ИЗДАТЕЛЬСТВО
МГТУ им. Н. Э. Баумана
2015

УДК 681.3.06
ББК 22.18
С17

Издание доступно в электронном виде на портале *ebooks.bmstu.ru*
по адресу: http://ebooks.bmstu.ru/catalog/255/book1272.html

Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

Рекомендовано Редакционно-издательским советом
МГТУ им. Н.Э. Баумана в качестве методических указаний

Рецензент
канд. техн. наук, доцент *В.А. Марынюк*

Самарев, Р. С.

С17 Создание простейших HTML-страниц, валидаторы кода. Каскадные таблицы стилей CSS : методические указания к выполнению практикума № 1 и лабораторной работы № 1 по дисциплинам «Языки интернет-программирования» и «Практикум по интернет-программированию» / Р. С. Самарев. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2015. — 39, [3] с. : ил.

ISBN 978-5-7038-4220-1

Приведены основные теоретические сведения о языке разметки HTML и о таблицах стилей CSS, необходимые для выполнения лабораторной работы и практикума по созданию HTML-страниц с помощью стилей CSS. Рассмотрены примеры. Также приведена литература по курсу и даны ссылки на интернет-источники.

Для студентов МГТУ им. Н.Э. Баумана, обучающихся по направлению «Информатика и вычислительная техника».

УДК 681.3.06
ББК 22.18

ISBN 978-5-7038-4220-1

Предисловие

Данные методические указания содержат сведения, необходимые для выполнения практической работы по теме HTML и лабораторной работы по теме стилей CSS. Для понимания этапов развития языка учащимся предлагается выполнить разметку страниц как с использованием старых и уже не применяющихся элементов HTML, так и с использованием новых элементов разметки.

Приведены минимальные сведения о языке разметки HTML, необходимые для впервые приступивших к изучению этого языка.

Даны сведения о таблицах стилей CSS, которые обеспечивают гибкость управления отображением элементов разметки и широко применяются в современном веб-программировании.

Подробно рассмотрены примеры, достаточные для понимания того, как следует выполнять практикум и лабораторную работу. Приведены порядок выполнения работ и контрольные вопросы.

Вопросы и замечания по данной работе просьба присылать автору на адрес: samarev@acm.org.

Практикум № 1
СОЗДАНИЕ ПРОСТЕЙШИХ HTML-СТРАНИЦ, ВАЛИДАТОРЫ КОДА

Цель работы — знакомство с языком разметки HTML, получение и закрепление практических навыков создания и форматирования HTML-страниц с использованием устаревших и современных средств разметки.

Объем работы — 4 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Язык HTML был разработан британским ученым Т. Бернерсом-Ли приблизительно в 1989—1991 гг. в Европейском совете по ядерным исследованиям в Женеве (Швейцария). Язык HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области верстки. HTML успешно справился с проблемой сложности языка SGML, определяя ограниченный набор структурных и семантических элементов — дескрипторов (или тегов). Помимо того, что по сравнению с SGML упрощено описание структуры документа, в HTML внесена поддержка гипертекста. Мультимедийные возможности были добавлены позже.

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

Ниже приведены ключевые спецификации HTML и даты их принятия:

• RFC 1866, HTML 2.0 — 22 сентября 1995 года;

• HTML 3.2 — 14 января 1997 года (создан консорциумом W3C — World Wide Web Consortium);

• HTML 4.0 — 18 декабря 1997 года;

• HTML 4.01 — 24 декабря 1999 года (внесены изменения, причем более значительные, чем кажется на первый взгляд);

• ISO/IEC 15445:2000 — 15 мая 2000 года (так называемый ISO HTML, основан на HTML 4.01 Strict);

• HTML 5 — по состоянию на 2014 год принят в черновом варианте.

В настоящее время язык HTML заменяется более строгим языком XHTML. Спецификация HTML 5 основана 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.

Структура HTML-разметки

Схематично разметку любой страницы можно представить следующим образом (см. сайты http://www.w3schools.com/, http://www.w3.org/):

```
<!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! Ниже приведены примеры современных спецификаций DOCTYPE.

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">
```

HTML 5:

```
<!DOCTYPE html>
```

В соответствии с терминологией языка 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 «< >», причем 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 размещается внутри открывающего тега. Он 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: html, head, body и некоторые другие элементы разметки текста.

Элемент html

Элемент html является обязательным для любой страницы. Он определяет границы html-документа.

Элемент head

Элемент head не является обязательным. Он представляет собой контейнер, в котором размещаются элементы заголовка документа.

Внутри элемента head могут быть определены другие элементы, представленные в табл. 1.

Таблица 1

Допустимые элементы внутри head

Тег	Описание
<title>	Заглавие документа. Обычно браузеры отображают этот заголовок в заголовке окна
<base/>	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/>	Связи между документом и внешними ресурсами, например, подключение стилей отображения
<meta/>	Метаданные о документе. Среди них могут быть как дополнительные ключевые слова, используемые для поиска, так и указание кодировки страницы
<script>	Скрипты, выполняемые на стороне браузера
<style>	Информация о стилях отображения в документе. В отличие от <link> эти стили будут внедрены в документ

Примечание. Различная форма записи тегов <title> и <base/> означает, что в XHTML непарные теги должны быть завершены символами «/>». В то же время теги типа <title> имеют парный закрывающий тег </title>.

Следует отметить, что для страниц, сформированных на русском языке, 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" />
```

Для HTML 5 допустимо использование короткой формы:

```
<meta charset="utf-8">
```

В настоящее время целесообразно использовать только UNICODE-кодировки, в частности utf-8. Кроме того, для HTML 5 существует короткая форма декларации:

```
<meta charset="utf-8">
```

Элемент body

Элемент body не является обязательным. Он 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 в настоящее время запрещено.

Элемент body содержит ряд стандартных атрибутов, таких как класс, стиль, заголовок элемента, язык, а также атрибуты-события, среди которых onload (загрузка страницы), onunload (закрытие страницы), события мыши и клавиатуры. В данном пособии не будем останавливаться на них подробно, поскольку они имеют отношение к программированию на Javascript.

Некоторые элементы разметки текста

Некоторые часто используемые элементы, которые применяются для разметки текста, приведены в табл. 2.

Таблица 2

Некоторые элементы разметки текста

Тег	Описание
	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. Пример: Некоторый текст. Будет выведено: Некоторый текст.
 	Перевод строки в любом месте текста, включая параграфы. Параграф при этом не разрывается. Пример: Некоторый текст. Будет выведено: Некоторый текст.

<pre>	Устаревший, но широко применяемый элемент, позволяющий обеспечить вывод текста, включая все переводы строк.
	Пример:
	<pre>Некоторый
	текст.</pre>
	Будет выведено:
	Некоторый
	текст.

	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 … . Пример: <ul type=”disk”> - Кофе - Чай - Молоко Будет выведено: • Кофе • Чай • Молоко

Окончание табл. 2

Тег	Описание

	Ordered list. Позволяет разметить список, элементы которого будут отмечены порядковым номером. В разметке каждый элемент списка указывается тегами … . Имеется атрибут (устаревший) start, позволяющий указать стартовый номер, а также атрибут type, позволяющий указать тип нумерации: 1 (арабские числа); A, a (буквы латинского алфавита); I, i (римские числа).
	Пример:
	`<ol>`
	`  <li>Кофе</li>`
	`  <li>Чай</li>`
	`  <li>Молоко</li>`
	`</ol>`
	Будет выведено:
	1. Кофе
	2. Чай
	3. Молоко

<h1> … <h6>	Устаревшие теги, предназначенные для того, чтобы выделить заголовки соответствующего уровня. Рекомендуется использовать стили

Вставка изображений

Для вставки изображений применяется элемент <img />.

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.

Другим обязательным атрибутом является **alt**. Его применение необходимо для того, чтобы в браузерах, которые не могут вывести 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. Они необходимы 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 программист при создании страницы не учитывает скорость загрузки страниц, пользователи, просматривающие такие страницы, будут видеть постоянно меняющееся расположение блоков и текста.

Пример img:

```
<img src=”picture.png” width=”100” height=”200” alt=”Диаграмма” />
```

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. Обязательным атрибутом является href, который содержит URL.

Примеры:

```
<a href=”http://www.somesite.ru/”>
<img src=”picture.png” width=”100” height=”200” alt=”Диаграмма” />
</a>

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

Разметка таблиц

Для разметки таблиц служит 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, но и 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 отображения браузерами таблицы представлен на рис. 1.

Наименование	**Количество**
Ручка	10
Карандаш	20

Рис. 1. Пример отображения таблицы

Элемент **table** может иметь 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 объединять ячейки таблицы по строкам и столбцам соответственно.

Пример:

```
<table width="100%" border="1">
   <tr>
     <th colspan="2">Назв. &amp;amp; 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 отображения браузером таблицы с объединением столбцов и строк представлен на рис. 2.

Назв. & Кол.		**Всего**
Ручка	10	30
Карандаш	20	

Рис. 2. 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 должен быть заменен блочной разметкой, а именно элементами и .

Элемент служит для выделения фрагментов текста, например цветом внутри параграфа. Обратите внимание на то, что параграф при этом остается целым.

Пример:

```
<p>Некоторый текст с <span style="color:red">красным</span>.</p>
```

При этом на экране отображается некоторый текст, в котором слово «красный» будет выделено красным цветом.

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

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. &amp;amp; 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 отображения браузером таблицы внутри блока представлен на рис. 3. (Более подробно см. раздел блочной модели http://www.w3.org/TR/CSS2/box.html)

Отображение некоторого текста, часть которого выделена красным

Назв. & Кол.		Всего
Ручка	10	30
Карандаш	20	

Рис. 3. 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: <input type="radio" name="gender" value="м"/>м
<input type="radio" name="gender" value="ж"/>ж<br />
<input type="submit" value="Отправить" />
</form>
```

Результат отображения браузером формы представлен на рис. 4.

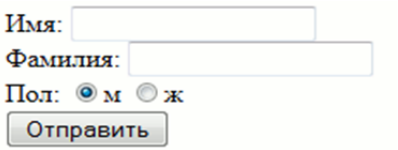

Рис. 4. Пример отображения формы

Основным элементом внутри формы является <input>, причем вид поля ввода определяется атрибутом type. Некоторые возможные значения атрибута type приведены в табл. 3.

Таблица 3

Значения атрибута type тега input

Значение type	Описание
type="text"	Текстовое поле
type="password"	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 приведенном выше примере **name="gender"**)
type="checkbox"	Селектор в виде квадрата. В отличие от типа radio, предполагает только два состояния: «пункт выделен» или «пункт 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 применялись 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. Некоторые из таких элементов приведены в табл. 4.

Таблица 4

Устаревшие элементы разметки

Тег	Описание
	Выравнивание по центру. Применим для любых элементов
<font>	Гарнитура текста, размер и цвет. Пример: `<font face="verdana" color="green">This is some text!</font>`

Окончание табл. 4

Тег	Описание
<i>	Шрифт курсив
<b>	Жирный шрифт

Использование валидаторов HTML и CSS

Для контроля правильности набора html-страницы и таблиц стилей CSS (сайтов) существуют специализированные средства — валидаторы.

Эталонный валидатор — Validator.W3C — доступен бесплатно в сети Интернет по адресу http://validator.w3.org/. Однако его использование предполагает, что проверяемый сайт полностью доступен в Интернете либо проверить можно будет только одиночные файлы html-страниц и таблиц стилей CSS.

Локальную проверку правильности разметки можно осуществить с помощью плагинов для браузера Mozilla Firefox, например HTML VALIDATOR (см. 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 (см. http://www.htmlvalidator.com/).

Рассмотрим принципы работы с HTML VALIDATOR. На рис. 5 представлен пример сообщения валидатора о корректной странице для браузера Mozilla Firefox 14, HTML VALIDATOR 0.9.5.1.

Для примера внесем 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>
```

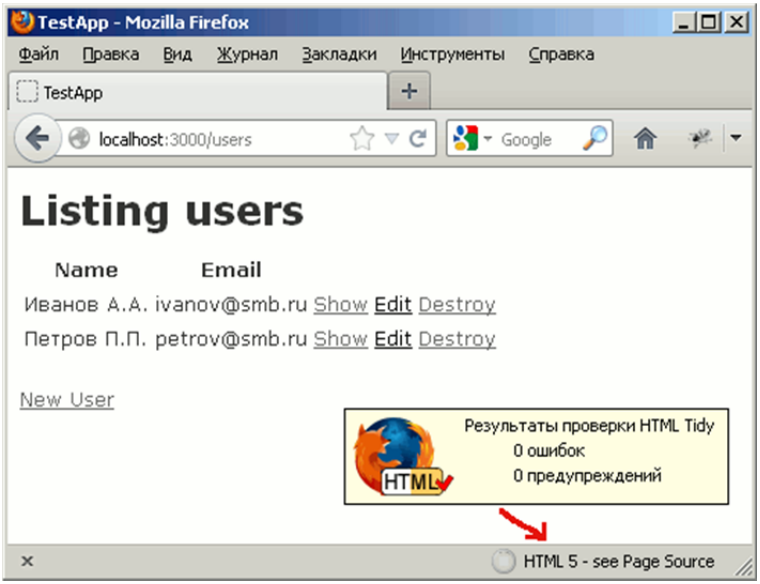

Рис. 5. Сообщение валидатора о корректной странице

При визуальном просмотре страницы ошибки могут быть не обнаружены. Однако валидатор показывает, что существует одна ошибка (рис. 6). В этом случае необходимо дважды щелкнуть мышью по выданному сообщению об ошибке, после чего откроется окно просмотра исходного кода страницы (рис. 7).

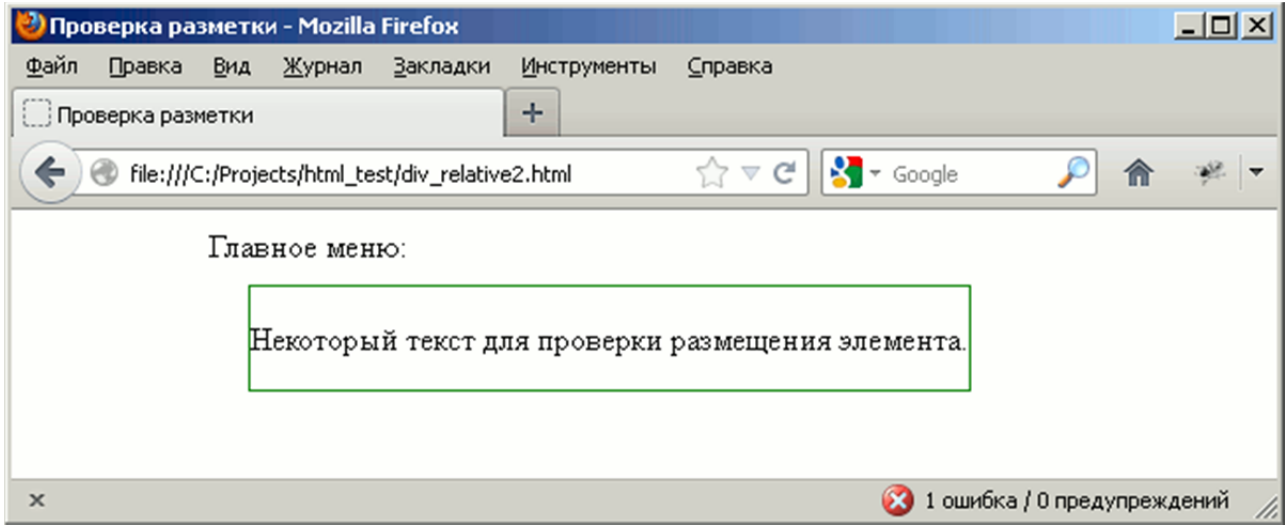

Рис. 6. Сообщение валидатора о некорректной странице

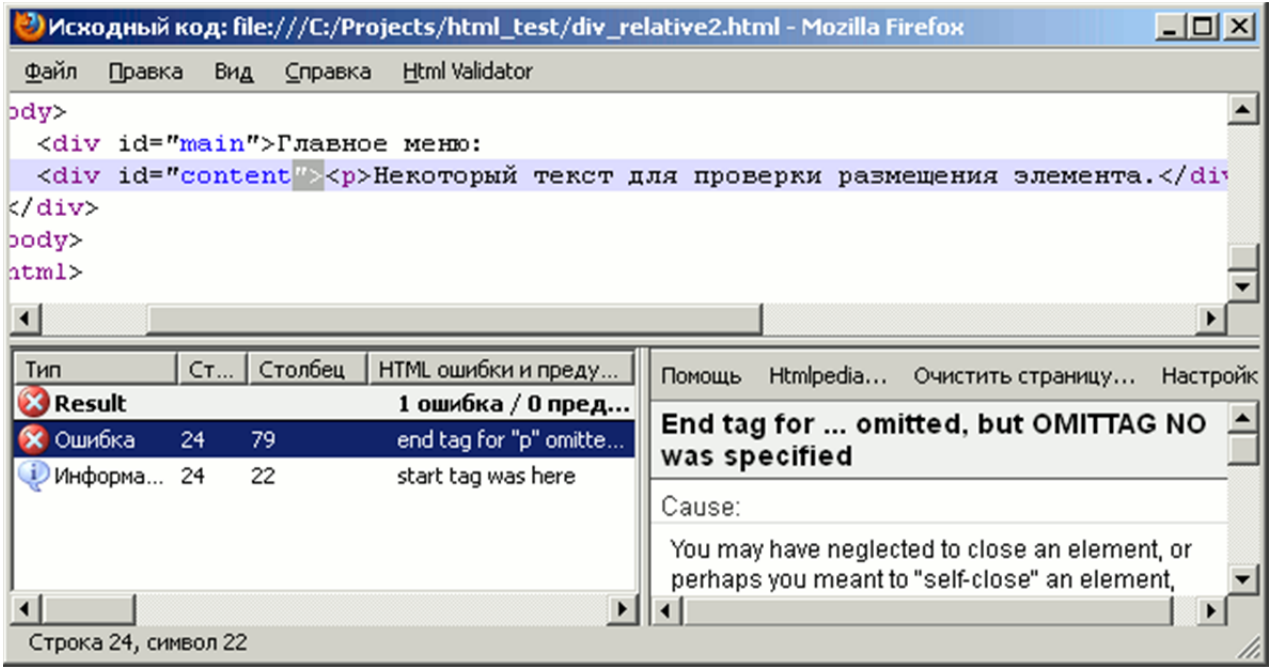

Рис. 7. Просмотр кода и сообщение об ошибке

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.

Для рассматриваемой страницы получим сообщение, представленное на рис. 8.

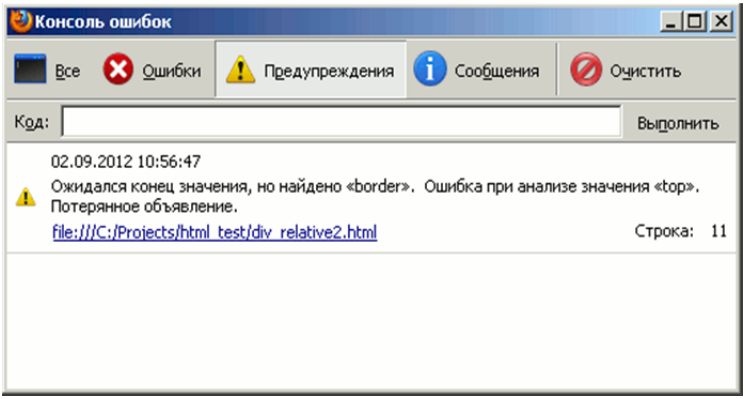

Рис. 8. Просмотр ошибок отображения браузера

По результатам проверки выявлено следующее:

1) в разметке страницы отсутствует закрывающий тег </p> в строке

```
<div id="content"><p>Некоторый текст для проверки...</p></div>
```

2) нарушены таблицы стилей, поскольку отсутствует разделитель «;» между селекторами:

```
top: 50px;
border: 1px solid black;
```

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ

Выполнить разметку HTML-страницы на основе устаревших элементов.

Задание выполнить в указанном ниже порядке.

1. Создать html-документ и указать старую спецификацию для совместимости со старыми элементами разметки:

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

Написать не менее 10 строк произвольного текста. Указать шрифт текста. Выделить в каждой строке одно из слов жирным шрифтом и красным цветом.

Подготовить второй вариант разметки, отличающийся шрифтом текста, а также выделенными словами. Заменить жирный шрифт курсивом, а красный цвет — синим.

2. Сформировать фрагмент расписания занятий с использованием элемента table.

3. Создать страницу, содержащую форму для ввода минимальных сведений о человеке. На странице предусмотреть поля для ввода значений с клавиатуры, а также селекторы выбора готовых значений.

4. Проверить корректность всех разработанных страниц и устранить ошибки. Составить таблицу выявленных ошибок и пояснить, как они отображаются в браузере.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен включать:

1) ФИО студента и номер группы;

2) наименование лабораторной работы;

3) названия выполненных пунктов и тексты реализованных HTML и CSS-разметки с указанием имен файлов.

Отчет предоставляется в электронном виде в формате pdf или odt.

Зачет ставится при условии выполнения всех пунктов задания, демонстрации работы программы и при наличии отчета и устных ответов на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каков основной принцип формирования разметки в HTML?

2. Какие примеры спецификаций DOCTYPE вы можете привести? Каковы причины их различий?

3. Каково назначение элементов div и span?

4. Какие примеры устаревших элементов разметки вы можете привести?

5. Какими средствами может быть проверена разметка HTML?

Лабораторная работа № 1
СОЗДАНИЕ HTML-СТРАНИЦ С ИСПОЛЬЗОВАНИЕМ КАСКАДНЫХ ТАБЛИЦ СТИЛЕЙ CSS

Цель работы — знакомство с языками HTML и CSS, а также получение практических навыков применения каскадных таблиц стилей для формирования отображения страниц HTML.

Объем работы — 2 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

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 (см. сайт http://www.w3.org/Style/CSS/).

Исторически CSS появился вместе с HTML версии 4.01 для упрощения манипулирования разметкой, поскольку именно применение стилей позволяет осуществлять групповую замену шрифта, цвета, размера и взаимного расположения элементов, разделив разметку и внешний вид как таковой.

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 — представляет собой правило для использования стиля. Браузер для каждого элемента пытается применить стиль в соответствии с заданным правилом. Стиль содержит набор свойств.

Различают простые селекторы, которые будут использованы для указанного элемента, в приведенном ниже примере — к любому заголовку h1, h2, h3:

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

Селекторы групп:

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

Этот фрагмент эквивалентен предыдущемуфрагменту, состоящему из трех простых фрагментов.

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 динамических атрибутов, которые изменяются 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 (family) шрифтов:

• Serif — шрифт с засечками. Обычно используется при бумажной печати. Наиболее используемый шрифт — Times;

• Sans-serif — шрифт без засечек. Подходит для заголовков. Наиболее часто применяемые шрифты этого семейства — Arial, Helvetica, Verdana;

• Monospace — шрифт, который обеспечивает равную ширину символов. Служит для вывода примеров кода, поскольку внешний вид этого текста будет соответствовать текстовой консоли. Наиболее распространен шрифт Courier;

• Fantasy, Cursive — декоративные и курсивные шрифты. Не рекомендуются к применению, поскольку шрифты этой группы необязательно присутствуют в компьютере, на котором будут просматривать html-страницу.

Выбор шрифта осуществляется свойством font-family. Пример использования 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 рис. 9 и 10.

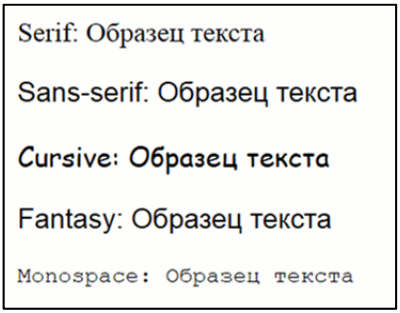

Рис. 9. Пример отображения шрифта в Mozilla Firefox 5

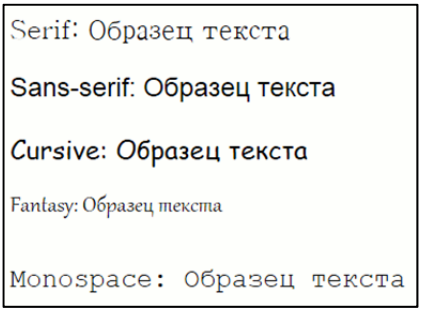

Рис. 10. Пример отображения шрифта в MS Internet Explorer 9

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. Для каждого семейства шрифтов различные браузеры в различных операционных системах могут иметь разные конкретные шрифты, соответствующие 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 применен шрифт, назначенный для serif-семейства по умолчанию.

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

```
font-style: normal | italic | oblique | inherit
```

где normal — обычное начертание; italic — курсив (имитация рукописного шрифта); oblique — наклонный шрифт (образован наклоном обычного).

Размер шрифта задается с помощью свойства font-size. Его возможные значения:

• larger и smaller — константы, определяющие относительный размер;

• xx-small, x-small, small, medium, large, x-large, xx-large — константы, определяющие абсолютный размер;

• ЧИСЛО % — число, определяющее размер в процентах от шрифта родительского элемента;

• ЧИСЛО px — число, определяющее размер в пикселах. Кроме того, размер шрифта указывается в специальных единицах: em (высота элемента, равная размеру текущего шрифта), ex (высота символа x), пункты (pt).

Толщина шрифта регулируется с помощью свойства font-weight:

```
font-weight: normal | bold | bolder | lighter | 100,  200 .. 900
```

Здесь normal — обычная толщина; bold — жирный шрифт; bolder — предельно жирный шрифт; lighter — тонкий шрифт;

100 — тонкий шрифт; 400 — соответствует нормальному; 700 — жирному.

Пример страницы с шрифтом разной толщины:

```
<!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 задает цвет шрифта. Можно указать цвет по его названию на английском языке (red, green, lime) или задать точное значение цвета в системах RGB, HSL, а также в RGBA, HSLA, для которых добавлен канал прозрачности. Полный перечень допустимых значений свойства color приведен в спецификации (см. http://www.w3.org/TR/css3-color/).

Пример:

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

Выбор цветовой схемы является очень важным этапом создания дизайна сайта. Существуют определенные методики подбора совместимых цветов. Имеются бесплатные средства в Интернете (например, см. http://colorscheme.ru/).

Фон

Фон элементов может быть задан однородным цветом, одиночным или мозаично расположенным изображением (возможности применения фона более подробно изложены на сайте http://www.w3schools.com/css/css_background.asp).

Используются следующие свойства фона:

• background-color — однородный цвет константой или кодом в одной из допустимых 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 — указание на то, будет ли изображение смещаться при скроллировании или будет оставаться на месте. Пример:

```
background-attachment:fixed;
```

• background-position — свойство, определяющее позицию размещения изображения на устройстве отображения. Пример:

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

Преобразование текста

В соответствии со спецификацией (см. http://www.w3.org/TR/css3-text/) текст может быть подвергнут преобразованиям при отображении, например, таким как:

• 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 средств преобразования текста:

```
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 из приведенного выше примера будет отображен так:

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

Если окно станет еще у́же, то текст будет выглядеть следующим образом:

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

Но ни при каких условиях текст не будет отображен в виде

```
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. (Для более подробного изучения блочной модели следует обратиться на сайт http://www.w3.org/TR/CSS2/box.html.)

Отступы и границы

В соответствии с блочной моделью для любого элемента имеются область самого элемента (content), внутренние поля (padding), рамка или граница (border), внешние границы (margin). Для каж-

дой области может быть задан размер. Наличие внутреннего поля позволяет сформировать рамку на заданном расстоянии от содержимого элемента; наличие внешнего поля — установить отступ между рядом расположенными элементами. На рис. 11 представлена схема расположения этих составных частей — элементов блочной модели. Обратите внимание на то, что рамка может иметь толщину, задаваемую соответствующим свойством CSS, и также участвует в расчете внешнего размера элемента.

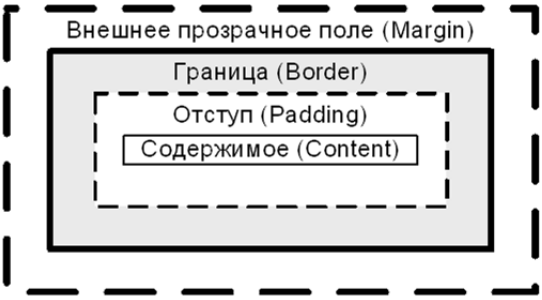

Рис. 11. Блочная модель CSS:
край внешнего поля; край границы;
край отступа; край содержимого

В CSS любой элемент имеет свойства width и height, которые устанавливают размер «содержимого» элемента в процентах, пикселах. Кроме того, значения этих свойств могут быть вычислены.

Размер отступа задается 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.

Пример:

```
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'. Типы границ представлены в табл. 5. Указываются толщина, тип начертания границы и цвет.

Пример:

```
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; }
```

При расчете размеров блока необходимо помнить про толщину границы.

Таблица 5

Типы границ

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

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

Пример:

```
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).

В литературе используются следующие термины, относящиеся к разделу позиционирования:

• нормальный поток — обычное поведение браузера при отображении данных;

• окно просмотра браузера — окно браузера, в котором отображается содержимое документа.

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

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

Абсолютное позиционирование

Абсолютное позиционирование подразумевает указание расположения элемента 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.

Приведем 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 , а зависит только от заданной позиции в стиле.

Относительное позиционирование

Смещение элемента в относительном позиционировании 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>
<html>
<head>
	<meta http-equiv="Content-Type" content="text/html; char-
set=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 серьезных оши-
	бок.</p>
  <div id="content"><p>В ближайшие дни на стадию бета-тестирования
перейдет ветка Firefox 16 и будет отделена aurora-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>
```

Чтобы понять различие во влиянии на нормальный поток абсолютного и относительного позиционирования, необходимо заменить в этом примере **position** на **absolute**.

Часто применяется комбинирование абсолютного и относительного позиционирования. Приведем пример, иллюстрирующий относительное позиционирование внутри блока с абсолютным позиционированием (обратите внимание на то, каким образом браузер отображает внешний блок при изменении размера окна):

```
<!DOCTYPE html>
<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 позиционирование позволяет закрепить элемент относительно окна просмотра, а не элемента-контейнера. Это дает возможность 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 свойства стиля обеспечат отображение элементов-изображений img в правой части страницы, а все остальные элементы будут размещены 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. Некоторые значения свойства display приведены в табл. 6.

Таблица 6

Некоторые значения свойства display

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

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

```
<!DOCTYPE html>
<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>
```

Кроме того, формирование табличного представления блоков возможно средствами позиционирования без использования специальных значений свойства display.

С помощью CSS можно размещать элементы в любом порядке, в том числе соответствующем табличному представлению, но без использования таблиц.

Существуют генераторы HTML/CSS-шаблонов (например, см. сайт http://csstemplater.com/). Однако отметим, что аналогичную работу можно проделать самостоятельно или с помощью собственных шаблонов.

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 (рис. 12). В Firebug доступны следующие закладки:

• Console — сообщения браузера об ошибках, предупреждения и отладочный вывод Javascript-программ;

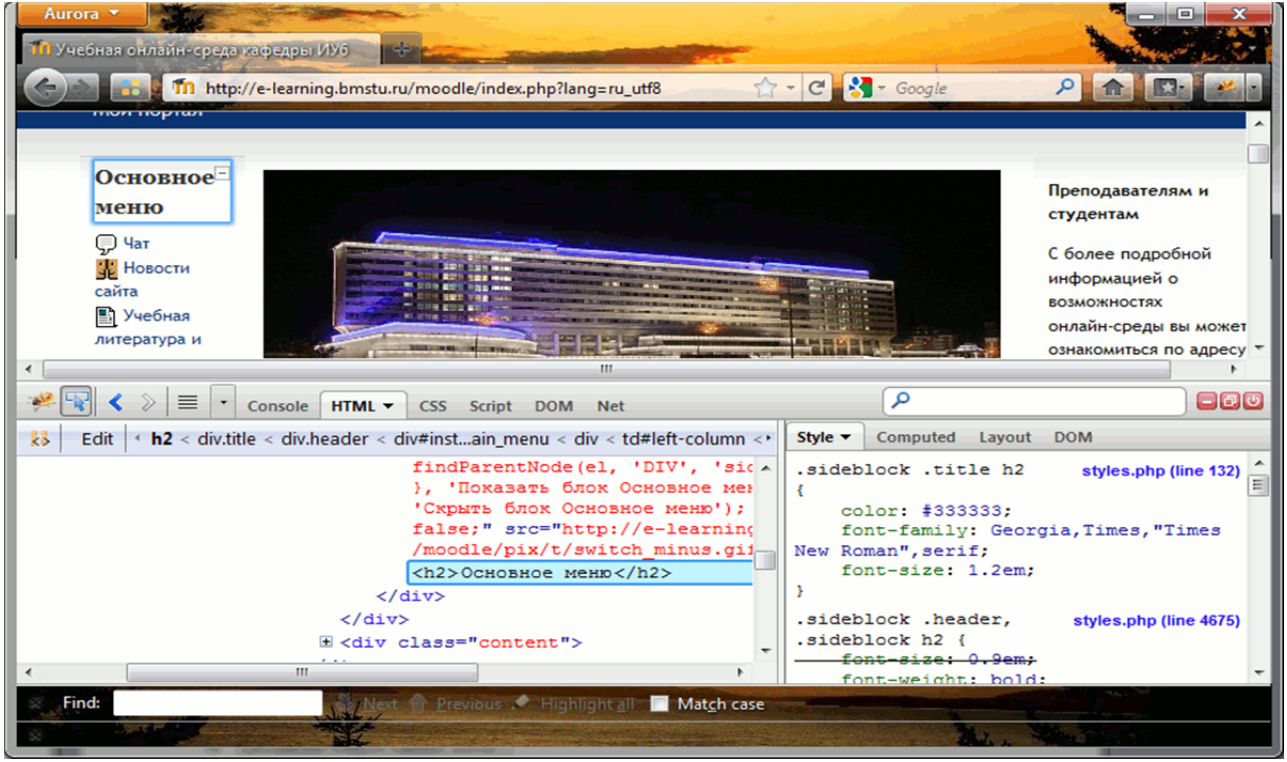

Рис. 12. Окно просмотра структуры и стилей в Firebug

• HTML — структура разметки страницы. Позволяет найти по отображенному элементу его соответствие в разметке и наоборот. Возможно интерактивное изменение стилей любых элементов;

• CSS — изменение таблиц стилей загруженной страницы;

• Script — Javascript-код страниц. Позволяет просматривать код, устанавливать точки останова, выполнять пошаговую отладку и трассировку значений переменных. По умолчанию отладка выключена;

• DOM — просмотр и изменение значения документа по модели DOM;

• Net — просмотр данных, передаваемых между браузером и сервером. Предоставляет возможность просмотра заголовков HTTP, переданных данных, а также порядка передачи данных. По умолчанию мониторинг сети выключен.

Обратите внимание на то, что изменения, внесенные в отладчик, 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, в котором они были созданы. Различаться могут 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 необходимо редактирование одного конкретного элемента, следует использовать закладку HTML. В закладке HTML доступны следующие вложенные закладки:

• Style (стиль) служит для редактирования стиля одного конкретного элемента, причем Firebug группирует стили, назначенные конкретно для выделенного элемента, и стили, унаследованные для этого элемента;

• 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.

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ

Выполнить разметку страницы с помощью стилей и блоков.

Задание выполнить в указанном ниже порядке.

1. Подготовить разметку произвольного текста, содержащего не менее 10 строк (могут быть использованы материалы из практикума № 1) с использованием таблицы стилей. Продемонстрировать выделение отдельных слов с помощью стилей, цвета и шрифта.

2. Сформировать фрагмент расписания занятий, используя элемент div и стили.

3. Создать страницу, содержащую форму с набором кнопок. На этой странице добавить список ссылок, соответствующих кнопкам. Модифицировать с помощью стилей отображение ссылок так, чтобы они стали похожи на кнопки.

4. Проверить корректность всех разработанных страниц и устранить ошибки. Составить таблицу выявленных ошибок и пояснить, как они отображаются в браузере.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен включать:

1) ФИО студента и номер группы;

2) наименование лабораторной работы;

3) названия выполненных пунктов и тексты реализованных HTML-страниц и стилей CSS с указанием имен файлов.

Отчет представляется в электронном виде в формате pdf или odt.

Зачет ставится при условии выполнения всех пунктов задания, демонстрации работы программы и при наличии отчета и устных ответов на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначен язык CSS?

2. Какие примеры селекторов CSS вы можете привести?

3. Какие примеры описания шрифтов с помощью CSS вы можете привести?

4. Что представляет собой блочная модель?

5. В чем различие абсолютного и относительного позиционирования?

6. Какие средства существуют для отладки стилей?

Литература

Дакетт Дж. Основы веб-программирования с использованием HTML, XHTML и CSS. М.: Эксмо, 2010.

Мейер Э. CSS-каскадные таблицы стилей. Подробное руководство: пер. с англ. СПб.: Символ-Плюс, 2008.

Хольцшлаг М. Языки HTML и CSS для создания web-сайтов: пер. с англ. А. Климович. М.: Триумф, 2007.

Содержание

Учебное издание

Самарев Роман Станиславович

Создание простейших HTML-страниц, валидаторы кода. Каскадные таблицы стилей CSS

Редактор *О.М. Королева*
Художник *А.С. Ключева*
Корректор *Н.В. Савельева*
Компьютерная верстка *С.А. Серебряковой*

В оформлении использованы шрифты
Студии Артемия Лебедева.

Оригинал-макет подготовлен
в Издательстве МГТУ им. Н.Э. Баумана.

Подписано в печать 07.07.2015. Формат 60×90/16.
Усл. печ. л. 2,75. Тираж 50 экз. Изд. № ЛР-0058-2015. Заказ

Издательство МГТУ им. Н.Э. Баумана.
105005, Москва, 2-я Бауманская ул., д. 5, стр. 1.
press@bmstu.ru
www.baumanpress.ru

Отпечатано в типографии МГТУ им. Н.Э. Баумана.
105005, Москва, 2-я Бауманская ул., д. 5, стр. 1.
baumanprint@gmail.com

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК