Московский государственный технический университет
имени Н. Э. Баумана

Р. С. Самарев
К. В. Кучеров

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

Методические указания к выполнению лабораторных работ № 1 и 2 по дисциплине «Языки интернет-программирования»

Редакция от 29.08.2019

УДК 681.3.06
ББК 22.18
С17

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

С17

Самарев, Р. С.

Создание простейших HTML-страниц, устаревшие элементы разметки. 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

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.

Лабораторная работа № 1
СОЗДАНИЕ ПРОСТЕЙШИХ HTML-СТРАНИЦ, УСТАРЕВШИЕ ЭЛЕМЕНТЫ РАЗМЕТКИ, 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 без привязки создаваемых документов к средствам отображения. Иными словами, главной целью при создании этого языка было обеспечение отображения документа без стилистических и структурных искажений 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).

К тому моменту уже существовал язык SGML, решавший сходные задачи, однако этот язык был сложен для использования людьми, не являющимися специалистами в области компьютерной верстки. HTML успешно справился с проблемой сложности языка SGML. Помимо того, что по сравнению с SGML было упрощено описание структуры документа, в HTML внесена поддержка гипертекста.

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 — 28 октября 2014 года — современный стандарт HTML, рекомендованный к использованию.

Помимо HTML существует семейство языков разметки веб-страниц XHTML (англ. eXtensible HyperText Markup Language — расширяемый язык разметки гипертекста), повторяющих и расширяющих возможности HTML 4. Спецификация HTML 5 основана на XHTML.

Структура HTML-разметки

Язык HTML произошел от SGML, поэтому унаследовал многие особенности языка SGML и его терминологию. Так, SGML-страница (как и HTML-страница), называется документом. Схематично разметка любого html-документа представлена на рисунке 1 (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>
```

Рис. 1. Структура HTML-документа

Документ состоит из элементов, например, элемента html, head, body. Следует отметить, что элемент html является корневым и обязательно должен присутствовать в документе.

Помимо перечисленных элементов существует целый ряд других, так, [<Надпись заголовка>], [<Стили>], [<Скрипты>] и [<Разметка страницы>] на самом деле тоже являются элементами, но для простоты пока заменим их текстом.

Структура элемента head из примера выше выглядит следующим образом:

```
<head (Заголовок)>
    [<Надпись заголовка>]
    [<Стили>]
    [<Скрипты>]
</head>
```

В данном случае в состав элемента входят все вложенные в него элементы (надпись заголовка, стили, скрипты). Таким образом, элемент может состоять из других элементов, однако это не обязательно.

Для ограничения элементов используются теги. Теги выделяются специальными знаками «<» и «>», причем открывающий тег выглядит как <имя-элемента>, а закрывающий — как </имя-элемента>.

У элемента head из примера выше есть два тега – открывающий <head> и закрывающий </head>. Как правило, у элемента есть и открывающий, и закрывающий теги, благодаря чему в этот элемент может быть вложен один или несколько дочерних. Однако существуют элементы, которые состоят всего из одного тега. В случае XHTML для таких элементов используется тег вида <имя-элемента />.

Тег может содержать атрибуты. Атрибут размещается внутри открывающего тега. Он характеризует элемент и имеет формат <имя-элемента атрибут1=”123”>. Атрибуты широко используются в языке HTML: с помощью атрибутов можно задать путь к подключаемым к документу стилям и скриптам, ширину и высоту некоторых элементов, поставить автофокус браузера на поле ввода и многое другое. Также существуют два наиболее часто используемых атрибута:

- атрибут **id** позволяет установить элементу уникальный в пределах страницы идентификатор и используется для того, чтобы

ссылаться на единственный конкретный элемент из CSS или JS (конкретные способы достижения этой цели будут рассмотрены в методических указаниях к следующим лабораторным работам);

- атрибут **class**, так же как **id**, позволяет установить уникальный в пределах страницы идентификатор, но сделать это можно как одному, так и нескольким элементам сразу, иными словами, class — это способ группировать элементы;

Анализ и отображение html-разметки на экране компьютера осуществляется браузером в соответствии со спецификацией html-документа. 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 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>
```

Актуальной спецификацией HTML по состоянию на 2019 год является HTML 5.2, принятая 14 декабря 2017. Поэтому рекомендуется использовать именно последнюю из приведённых выше деклараций, а использование более ранних версии спецификации HTML целесообразно только в тех случаях, когда

разработчикам предъявляются жесткие требования по совместимости со старыми браузерами.

Основные элементы разметки

Рассмотрим основные элементы разметки: html, head, body, а также некоторые элементы разметки текста.

Элемент html

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

Элемент head

Элемент head представляет собой контейнер, в котором размещаются элементы заголовка документа. Такие элементы содержат метаданные, то есть информацию о самой HTML-странице.

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

Элемент body

Элемент body 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.

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

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

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.

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

Отметим, что для разметки списков в тексте используются специальные элементы, которые, в частности, позволяют автоматизировать нумерацию (см. табл. 3).

Таблица 3

Тег	Описание

	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">`
	`<li>Кофе</li>`
	`<li>Чай</li>`
	`<li>Молоко</li>`
	`</ul>`
	Будет выведено:
	• Кофе
	• Чай
	• Молоко

	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

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 использования элемента 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).

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>
```

Исторически элемент table может иметь атрибут **width**, который указывает ширину таблицы в пикселах или процентах ширины устройства отображения, а также атрибут **border**, который указывает толщину линии границы. В HTML 5 эти использование этих атрибутов не рекомендуется, для задания стилей таблиц следует использовать CSS. Хотя CSS будет рассмотрен более подробно в следующей лабораторной работе, приведем пример задания стиля границ таблицы с помощью CSS.

Необходимо добавить в элемент <head> HTML-документа элемент <style> со следующим содержимым:

```
<style>
  table, tr, th, td {
    border: 1px solid black;
  }
</style>
```

Результат отображения браузерами таблицы представлен на рис. 2.

Наименование	**Количество**
Ручка	10
Карандаш	20

Рис. 2. Пример отображения таблицы

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. 3.

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

Рис. 3. Пример отображения таблицы с объединением столбцов и строк

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

Верстка с использованием таблиц часто называется “табличной верстkой”. Такой подход к верстке был единственно возможным до появления каскадных таблиц стилей (CSS), а также оставался популярным на протяжении некоторого промежутка времени после появления CSS, пока окончательно не был им вытеснен. Сейчас верстка таблицами относится к устаревшим методам в связи с тем, что не позволяет использовать современные

возможности CSS, поэтому ее использование в настоящее время может быть оправдано только в образовательных целях

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. 4. (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. 4. 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>

   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. 5.

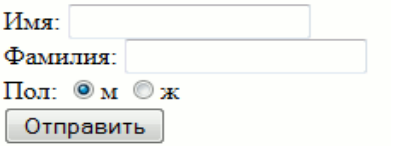

Рис. 5. Пример отображения формы

Атрибуты **action** и **method** формы позволяют указать, на какой URL (атрибут **action**) и каким HTTP-методом (атрибут **method**) будут отправлены данные формы после ее подтверждения (submit).

URL (uniform resource locator) является стандартным единообразным определителем местонахождения ресурса (в данном случае веб-страницы) и состоит из следующих частей:

```
scheme:[//authority]path[?query][#fragment]
```

или

```
http://www.example.com/path/to/resource?query=1#fragment
```

Если атрибут **action** содержит только path, а не URL целиком, недостающие части (а именно scheme и authority) будут взяты браузером из URL текущей страницы. Иными словами, если форма из примера выше отображается на странице с URL *http://example.com/form* то после нажатия на кнопку “Отправить” пользователь будет направлен на url *http://example.com/form_submit*, так как action=”/form_submit” будет добавлен к *http://example.com* (/form отбрасывается, т.к. является составной частью следующей за authority части path).

Основным элементом внутри формы является <input>, причем вид поля ввода определяется атрибутом type. Некоторые возможные значения атрибута type приведены в табл. 4.

Таблица 4

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

Данные, содержащиеся в форме, по умолчанию передаются строкой запроса («query» из состава URL) следующим образом:

```
?[имя1]=[значение1]&[имя2]=[значение2]&...
```

Знак & используется в URL как разделитель. Значения берутся браузером из того, что пользователь ввел в элементы <input> формы. Для задания имен используется атрибут name тега <input>. Так, к примеру, для такой разметки

```
<input type="text" id=”fname” name="fname" /><br />
<input type="text" id=”lname” name="lname" /><br />
```

(при условии, что пользователь ввел “Ivan” в input с id=”fname” и “Ivanov” в input с id=”lname”) будет сгенерирована следующая строка с query-параметрами

```
?fname=Ivan&lname=Ivanov
```

Таким образом, после отправки формы браузер перейдет на URL

```
http://example.com/form_submit?fname=Ivan&lname=Ivanov
```

Следует сделать несколько важных уточнений:

1. Символы всех алфавитов, кроме английского, а также специальные символы, например, кавычки, пробел, восклицательный знак, не могут быть записаны в URL и

кодируются специальным образом со знаком процента (так называемая percent encoding).

2. Символ пробел не может присутствовать в «query», поэтому его следует или кодировать (см. п.1), или заменять знаком плюс.

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. 5.

Таблица 5

Некоторые 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>`
<strike>	Перечеркнутый текст
<nobr>	Запрет переноса строк для текста внутри элемента
<listing> <plaintext>	Вывод текста без интерпретации HTML

Элементы семантической разметки текста

Стандарт HTML5 позволяет использовать так называемые семантические элементы разметки, позволяющие на уровне использования языка разметки определять смысл, структуру или стиль текста (или какой-либо его части). В таблице 6 приводятся некоторые такие элементы и примеры их использования.

Таблица 6

Элементы семантической разметки текста

Тег	Описание
`<em>`	Акцентирует внимание на выделенном тексте. По-умолчанию используется шрифт курсив. Пример: `Здесь <em>нет</em> сложностей.` Здесь *нет* сложностей
`<strong>`	Особенно важный текст. По-умолчанию отображается жирным шрифтом. Пример: `Это <strong>важная часть</strong>.` Это **важная часть**.
`<mark>`	Выделение текста цветом в справочных целях, например, на странице с результатами поиска, на которой выделяется каждый экземпляр искомого слова. Пример: `Мы работаем с <mark>HTML 5</mark>.` Мы работаем с HTML 5.
`<s>`	Означает, что текст не актуален. Отображается как зачеркнутый прямой линией текст. Пример: `Сделаем пример на <s>SGML</s> HTML 5.` Сделаем пример на ~~SGML~~ HTML 5.
`<small>`	Фрагмент текста, представляющий собой комментарий, вспомогательный текст и пр. Отображается уменьшенным шрифтом
`<var>`	Выделяет переменную в математическом выражении. Пример: `<var>l</var> × <var>w</var> × <var>h</var>` $l \times w \times h$,
`<sub>`	Нижний индекс. Пример: `C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>` $C_8H_{10}N_4O_2$
`<sup>`	Верхний индекс. Пример: `<p><var>a<sup>2</sup></var>+<var>b<sup>2</sup></var> =<var>c<sup>2</sup></var></p>` $a^2 + b^2 = c^2$

<code>	Используется для отображения одной строки кода в тексте без переноса строки. По-умолчанию применяется моноширинный шрифт. В отличии от элемента <pre>, здесь не учитываются дополнительные пробелы и переносы строк. Использование <code> для многострочного кода возможно, но необходимо использовать тег или оборачивать каждую строку текста элементом <p>. Примеры: `<p>Метод **<code>push()</code>** добавляет элемент</p>` `**<code>**` `#include **<**stdio.h**> **` `int main (void)** **` `{** **` `  puts("Hello, World!");** **` `  return 0;** **` `}` `**</code>**`
<samp>	Пример вывода программы. Отображается моноширинным шрифтом

<cite>	(*от англ. Citation*) Используется для указания ссылки на источник цитаты. Должен содержать внутри себя название произведения или URL.
	Пример:
	`<p><cite>http://www.asimovonline.com</cite></p>`
	
	`<cite><a href="http://www.george-orwell.org/1984/0.html">`
	`<i>Nineteen Eighty-Four</i></a></cite> by George Orwell (Part 1, Chapter 1).`

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 https://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. https://addons.mozilla.org/ru/firefox/addon/html-validator/). 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. 6 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>

<html>
<head>
    <title>Проверка разметки</title>
    <meta 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>
```

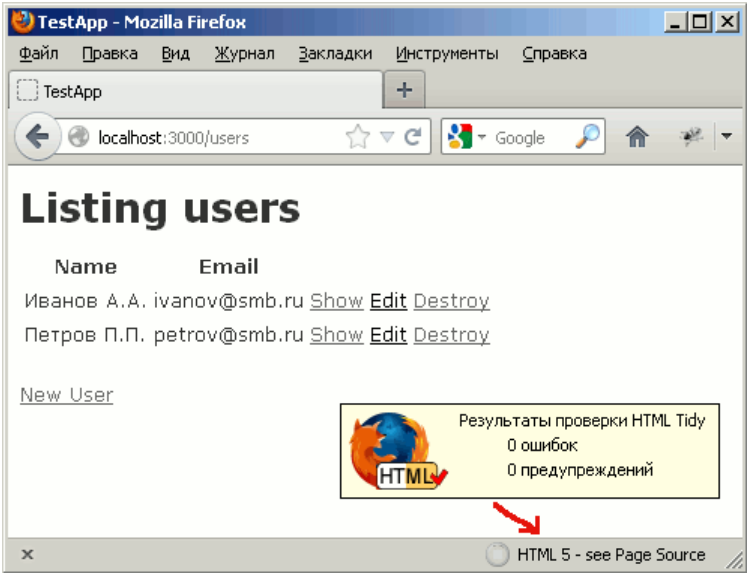

Рис. 6. Сообщение валидатора о корректной странице

При визуальном просмотре страницы ошибки могут быть не обнаружены. Однако валидатор показывает, что существует одна ошибка (рис. 7). В этом случае необходимо дважды щелкнуть мышью по выданному сообщению об ошибке, после чего откроется окно просмотра исходного кода страницы (рис. 8).

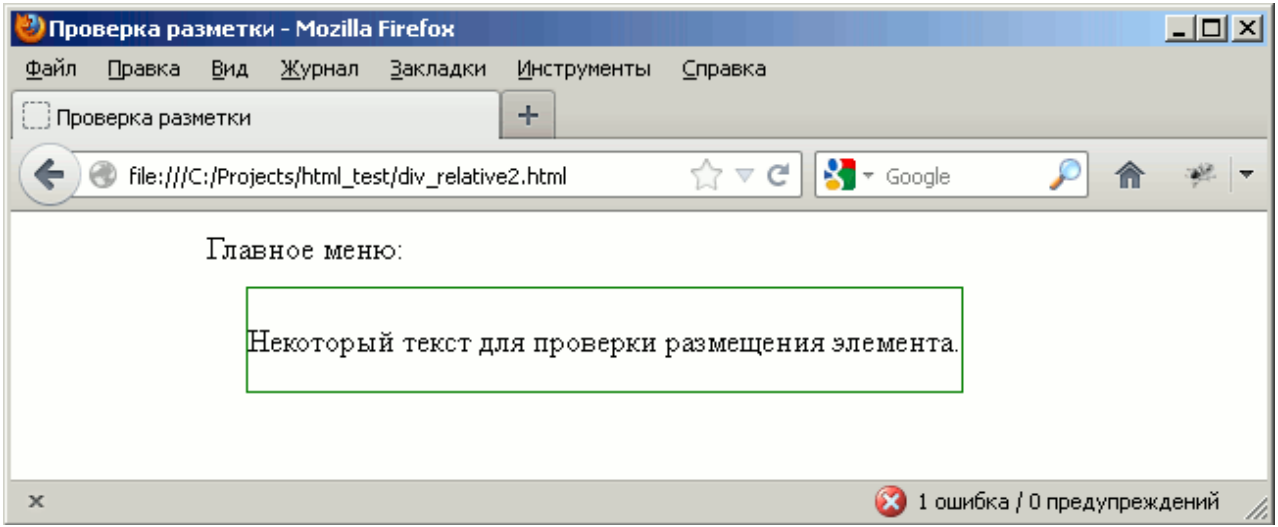

Рис. 7. Сообщение валидатора о некорректной странице

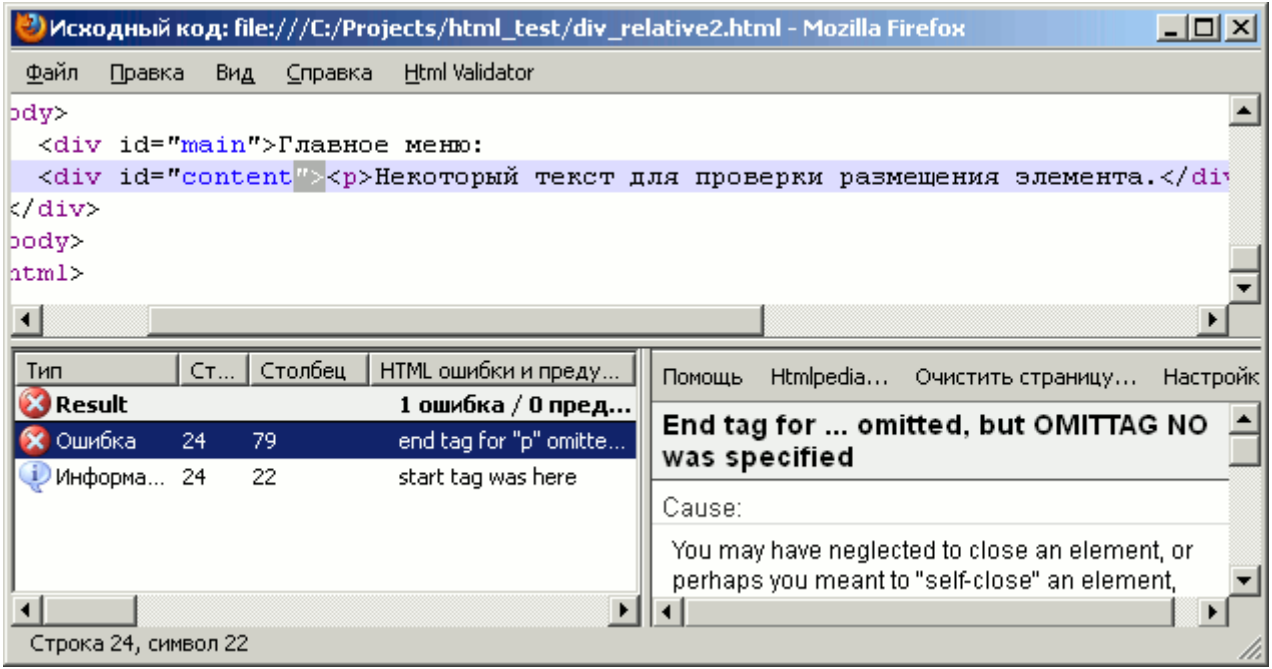

Рис. 8. Просмотр кода и сообщение об ошибке

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. 9.

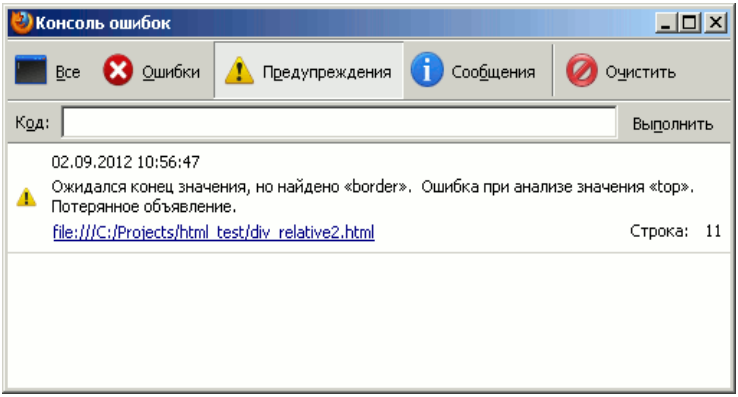

Рис. 9. Просмотр ошибок отображения браузера

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

1. Выполните разметку HTML-страницы с использованием элементов семантической разметки текста (см. таблицу 6). Напишите не менее 10 строк произвольного текста. Выделите в каждой строке несколько слов как более важные, значительно важные, добавьте математические формулы и пр. Также, с использованием семантической разметки, добавьте к тексту произвольные строки кода на любом языке программирования.

Подготовьте второй вариант разметки, отличающийся выделенными словами. Замените фрагменты, которые ранее отображались курсивом на фрагменты, отображаемые жирным шрифтом.

2. Сформируйте HTML-страницу с фрагментом расписания занятий (используйте элемент table). Выберите фрагмент расписания таким образом, чтобы хотя бы в одном месте возникала необходимость объединения ячеек таблицы.

3. Подготовьте разметку формы регистрации на произвольном сайте. Для расположения элементов используйте табличную верстку.

4. Используя валидаторы HTML (предустановленный в браузере или https://validator.w3.org/), проверьте полученные HTML-страницы на наличие ошибок. Составьте таблицу выявленных ошибок, в которую внесите все ошибки валидации и их фактические проявления в браузере. Устраните все найденные ошибки.

Отчет по выполненной работе должен содержать все тексты HTML по каждому пункту и должен быть оформлен в виде pdf-файла.

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. Приведите примеры элементов семантической разметки.
5. Какими средствами может быть проверена разметка HTML?

Лабораторная работа № 2
ТАБЛИЦЫ СТИЛЕЙ, СЕЛЕКТОРЫ, БЛОЧАЯ МОДЕЛЬ РАЗМЕТКИ, СТРАНИЦА С КНОПКАМИ И ССЫЛКАМИ, BOOTSTRAP

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/).

Изначально в языке HTML не было возможности манипулировать отображением нескольких элементов сразу (к примеру, сделать текст всех заголовков страницы красным, не изменяя каждый из них по отдельности). По мере усложнения разметки веб-страниц эта возможность становилась все более востребована, что в конечном итоге привело к идее разделения разметки элементов и её внешнего вида. В то время как языки разметки уже существовали и использовались, средства манипулирования внешним видом разметки имели место только внутри языков разметки. Шагом к разделению разметки и ее внешнего вида стала разработка языка CSS. Исторически CSS появился вместе с HTML версии 4.01 и привнес в верстку возможность 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.

Структура HTML-разметки

В методических указаниях по лабораторной работе 1 была приведена следующая условная схема содержимого HTML-страницы:

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

CSS добавляется на HTML-страницу в секцию [<Стили>] одним из двух способов:

1. внедрить CSS-правила прямо в HTML-код (этот способ также называют инлайн-CSS), для этого необходимо написать их внутри элемента style;

Пример:

```
<!Спецификация DOCTYPE>
<html>
<head (Заголовок)>
 [<Надпись заголовка>]
 <style>
        CSS, который необходимо добавить
 </style>
 [<Скрипты>]
</head>
<body (Тело страницы)>
 [<Разметка страницы>]
</body>
</html>
```

2. подключить внешний CSS-файл с помощью элемента link;

Пример:

```
<!Спецификация DOCTYPE>
<html>
<head (Заголовок)>
 [<Надпись заголовка>]
 <link href="css/style.css" type="text/css"/>
 [<Скрипты>]
</head>
<body (Тело страницы)>
 [<Разметка страницы>]
</body>
</html>
```

Элементов link может быть столько, сколько файлов со стилями необходимо подключить.

Стиль элемента

Стиль — это набор свойств элемента, например, ширина, высота, цвет текста, цвет фона, фоновый рисунок и т.д.

Стиль элемента может быть задан несколькими способами:

- как значение атрибута style внутри любого открывающего тега непосредственно внутри разметки, например <div style=”width: 1px;”>; это наименее предпочтительный подход, так как при его использовании нет возможности задать стиль для группы элементов;
- с помощью селекторов внутри элемента style в заголовке страницы или во внешнем файле с расширением *.css.

Селекторы

Для того, чтобы применить стиль к одному или нескольким элементам, в CSS существуют селекторы. Любой CSS-файл построен по следующему принципу:

```
селектор1 {
        свойство1;
        свойство2;
        ...
}

селектор2 {
        свойство3;
        свойство4;
        ...
}

...
```

Селектор — ключевое понятие CSS — представляет собой правило, в соответствии с которым браузер определяет, к каким элементам будет применен стиль.

Различают несколько видов селекторов, рассмотрим их далее.

Простые селекторы указывают браузеру элемент, к которому необходимо применить стиль, 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 предыдущему фрагменту, 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" */
```

Класс в CSS — это способ группировать элементы. Каждый элемент может одновременно принадлежать к разным классам.

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" */
```

Помните, что на странице может быть только один элемент с заданным id. Идентификатор всегда уникален!

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 */
```

К стандартным атрибутам относятся атрибуты id, class, src и т.п., то есть атрибуты, указанные в спецификации HTML. В HTML 5 появилась возможность вводить нестандартные (или пользовательские) атрибуты. Имена этих атрибутов должны начинаться с префикса data-, поэтому их неформальное название — data-атрибуты. Пример:

```
<h1 id="chapter1" data-my-attribute="my-value">…</h1>
```

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 */
```

В заключение проиллюстрируем некоторые возможности CSS:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <style>

     /*Зададим ширину и высоту элементов html и body в процентах относительно размера окна браузера, доступной для просмотра веб-страниц, но без учета панели меню. Такие правила часто используются при верстке, так как по умолчанию элементы занимают как можно меньше места (в соответствии с размером содержимого), в то время как часто бывает необходимо растянуть страницу на все окно просмотра.*/
        html, body {
            height: 100%;
            width: 100%;
        }

    /*селектор стандартного атрибута с заданным значением*/
        main[role=”main”] {
            height: 100%;
        }

       /*селектор класса. Используются, часто, из-за гибкости управления, даже в тех ситуациях, когда на странице есть только один элемент указанного класса.*/
        .red {
            color: red;
        }

      /*селектор id. Используется реже, чем классы. Обычно id элемента используется при манипулировании разметкой конкретного элемента при помощи JavaScript.*/
        #green {
            color: green;
        }

     /*правило иллюстрирует способ использования data-атрибута. Обратите внимание, что значение атрибута никак не используется в CSS-селекторе. Таким образом, каким бы ни было значение атрибута, стили будут применены к элементу.*/
        p[data-my-font] {
            font-style: italic;
        }
    </style>
</head>
```

```
<body>
    <main role="main">
        <p class="red">red</p>
        <p id="green">green</p>
        <p data-my-font>italic</p>
    </main>
</body>
</html>
```

Блочная модель

Вместе с CSS появилась блочная модель визуализации элементов html-документа. Блочная модель 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 каждой области может быть задан размер. Наличие внутреннего поля позволяет сформировать рамку на заданном расстоянии от содержимого элемента; наличие внешнего поля — установить отступ между расположенными рядом элементами. На рис. 10 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.

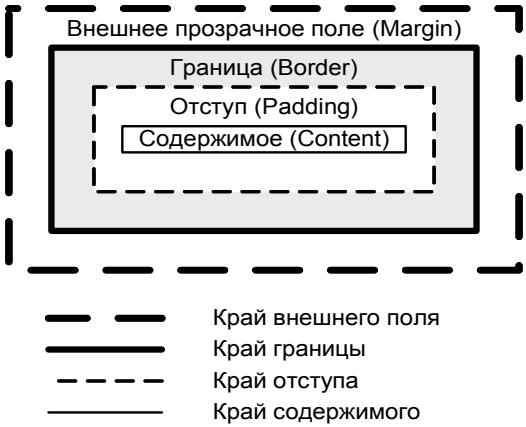

Рис. 10. Блочная модель CSS:
край внешнего поля; край границы;
край отступа; край содержимого

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. 6. 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 7

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:

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

Важным аспектом CSS является позиционирование элементов на странице. Материал, посвященный данной теме, приведен в Приложении А.

Основы библиотеки Bootstrap

Несмотря на то, что CSS существенно облегчает манипулирование разметкой, создание современного интерактивного дизайна «с нуля» является трудоемкой задачей, решать которую в каждом новом проекте неэффективно даже для крупных IT-компаний.

Развитие веб-дизайна, отсутствие развития CSS (с 1998 по 2011 год CSS велась разработка CSS 2.1) и желание разработчиков повторно использовать уже написанный код привело к формированию библиотек с CSS-стилями. Среди них можно найти узкоспециализированные библиотеки, библиотеки, разработанные исключительно для упрощения размещения элементов на странице, и многофункциональные библиотеки, изменяющие отображение стандартных html-элементов и добавляющие интерактивность на страницу.

Одной из многофункциональных библиотек является Bootstrap, разработанный компанией Twitter. Эта библиотека за время своего существования стала очень популярной, Bootstrap оказал большое влияние на более современные библиотеки и сильно изменил подход к разработке пользовательских интерфейсов. Так, к примеру, способы именования CSS-классов и подходы к их организации стали де-факто стандартными и используются за пределами этой библиотеки. Эта библиотека до сих пор активно развивается, текущей «мажорной» (по состоянию на 2019 год) является версия 4.

Рассмотрим основные концепции, которые привносит в верстку библиотека Bootstrap.

Основные классы блоков

Любая разметка строится по иерархическому принципу. Этому же принципу следуют вводимые Bootstrap классы блоков. Так как в данном разделе речь идет о позиционировании элементов, все рассматриваемые классы должны применяться к элементу div.

Иерархия классов разметки выглядит следующим образом (см. рис. 11):

Рис. 11. Иерархия классов разметки Bootstrap

Класс «контейнер» представляет собой контейнер для элементов и обеспечивает их корректное отображение с учетом изменения размеров окна браузера. Если проводить параллели с табличной верстkой, то класс “контейнер” соответствует элементу table.

В контейнер могут быть вложены «строки». Элементы этого класса можно воспринимать как строки таблицы (Bootstrap-класс **row**).

Внутри «строки» могут располагаться «столбцы». Элементы этого класса можно воспринимать как столбцы таблицы (Bootstrap -класс **col**).

Таким образом, разметка на Bootstrap строится по следующему иерархическому принципу

```
<div class=”container”>
    <div class=”row”>
        <div class=”col”>
        </div>
    </div>
</div>
```

Сетка

Для упрощения размещения элементов на странице в Bootstrap используется так называемая сетка (grid). Окно просмотра браузера условно делится на 12 равных по ширине столбцов (изменение ширины столбцов и отступов между ними в зависимости от ширины окна браузера Bootstrap берет на себя). Таким образом, веб-разработчик может представлять себе окно просмотра браузера так, как это представлено на рис. 12.

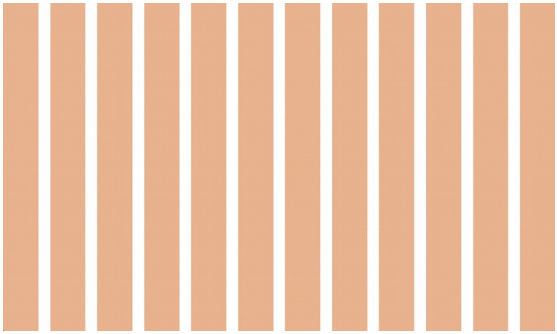

Рис. 12. Сетка Bootstrap

Столбцы можно объединять между собой (см. рис. 13, где числами на горизонтальных блоках обозначена относительная ширина столбца в виде количества используемых элементарных столбцов).

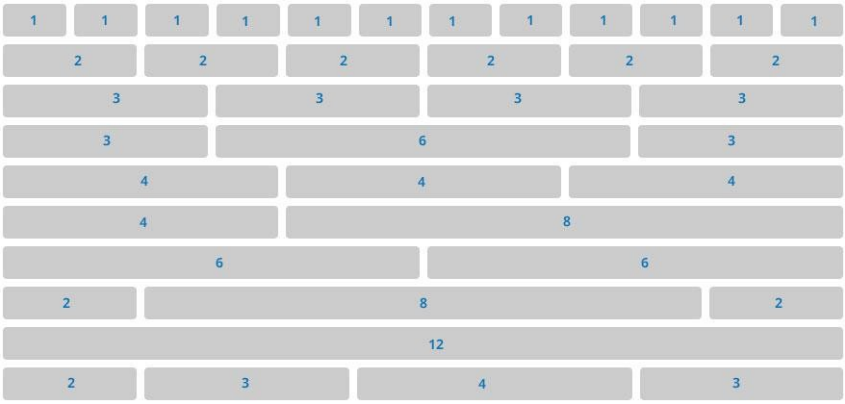

Рис. 13. Схема объединения столбцов

Используя эту модель, можно гибко манипулировать размещением блоков на странице.

Для манипулирования шириной в Bootstrap используются классы столбцов: «col-1», «col-2», …, «col-12». Следует отметить, что числа в названиях классов могут быть только целыми с 1 до 12 и соответствуют относительной ширине столбцов, показанных на иллюстрации выше.

Для того, чтобы разделить страницу на две части с соотношением областей 1:2 , необходимо написать:

```
<div class="container">
    <div class="row">
        <div class="col-4">
            <p>Это часть 1 - слева</p>
        </div>
        <div class="col-8">
            <p>Это часть 2 - справа</p>
        </div>
    </div>
</div>
```

В данном случае часть 1 будет занимать ⅓ ширины окна просмотра браузера (4 единицы ширины из 12), а часть 2 — ⅔ (8 единиц ширины из 12). Если необходимо разделить страницу на несколько равных частей, можно использовать класс col без суффиксов — Bootstrap автоматически подберет ширину в зависимости от количества элементов с этим классом (один элемент div с классом col будет занимать 12 единиц ширины из 12, два — по 6 единиц из 12 каждый, три — по 4 и т.д.).

Адаптивность

Одной из главных причин, по которым Bootstrap стал так популярен, является встроенная поддержка адаптивности. Адаптивность — это свойство веб-страницы подстраиваться под изменяющиеся размеры экрана, благодаря чему на мобильных устройствах, планшетах и ПК страница выглядит по-разному при том, что создается только один вариант разметки (но все изменения отображения в этой разметке, безусловно, прописываются).

В данных методических указаниях мы не будем подробно рассматривать адаптивную верстку, однако отметим, что в Bootstrap существуют (аналогично именованию классов сетки) несколько суффиксов названий классов: xs (extra small, небольшие

смартфоны), sm (small, смартфоны), md (medium, планшеты), lg (large, ноутбуки), xl (extra large, мониторы с большой диагональю). При необходимости это позволяет использовать или классы вида col-sm, col-md (если нужен один “столбец” или все столбцы равного размера) или col-sm-12, col-md-6 (если нужно явно указать размер столбцов) и т.п., благодаря чему легко можно задать различное расположение элементов на странице в зависимости от размеров экрана.

Пример разметки с помощью Bootstsrap

Рассмотрим пошаговый пример разметки страницы профиля пользователя с помощью Bootstrap. Создаваемая страница будет состоять из трех областей:

- верхней части (header) с навигационной панелью (navbar);
- основного содержимого (main) с аватаром пользователя и текстом;
- подвала (footer).

Макет такой страницы приведен на рис. 14.

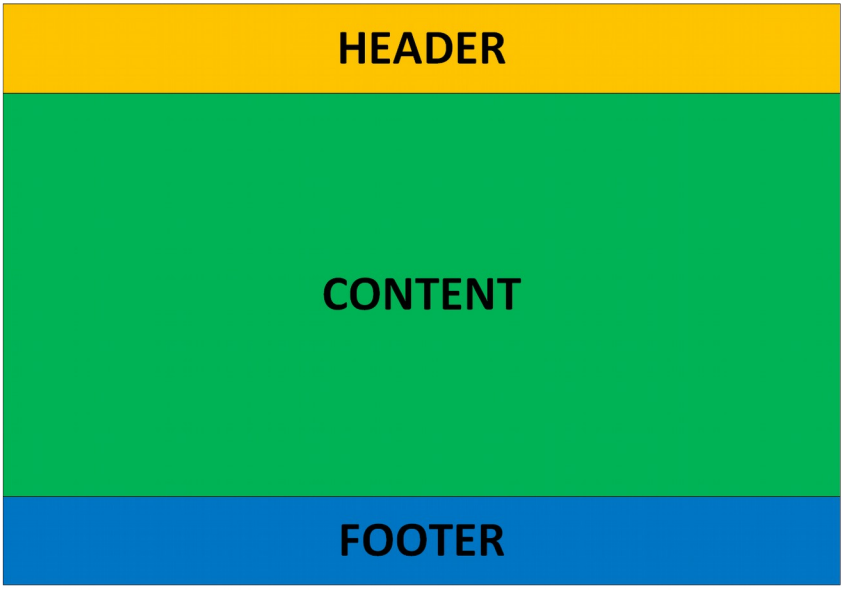

Рис. 14. Макет разметки

Рассмотрим процесс разметки по шагам.

Пример создания Bootstrap-разметки

Возьмём следующий шаблон, записанный в файле с расширением html:

```
<!DOCTYPE html>
<html lang=”ru”>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport"
          content="width=device-width, initial-scale=1">
    <title>ICS6 Bootstrap</title>
    <!-- <style></style> -->
</head>
<body>
    <!-- <header></header> -->
    <!-- <main></main> -->
    <!-- <footer></footer> -->
</body>
</html>
```

Если открыть данный файл в браузере, будет отображаться пустая страница с заголовком окна “Bootstrap”.

Обратите внимание на выделенные полужирным шрифтом html-комментарии. Они обозначают места, куда в дальнейшем будет добавлена разметка верхней части страницы (header), основного содержимого (main), подвала (footer), а также стили (style). В дальнейшем мы не будем повторять приведенную выше шаблон, а будем использовать комментарии для того, чтобы ссылаться на нужные места этого шаблона.

Подключение Bootstrap

Существует несколько способов подключения Bootstrap к веб-странице: можно скачать архив с определенной версией библиотеки с официального сайта или воспользоваться CDN (Content Delivery Network). CDN — это географически распределенная по планете сеть прокси-серверов, которая может отдавать какие-либо статические файлы (в данном случае CSS-файлы Bootstrap) с наиболее близкого клиенту сервера (а значит быстрее всего). Единственное ограничение CDN — для его работы необходимо подключение к интернету.

Мы будем рассматривать подключение Bootstrap с сайта кафедры ИУ-6. В данной работе нам понадобится исключительно CSS, поэтому для подключения Bootstrap достаточно добавить только один элемент в head созданной страницы:

```
<link rel="stylesheet"
      href="http://e-learning.bmstu.ru/moodle/pluginfile.php/7546/
mod_folder/content/0/bootstrap.min.css">
```

Структура страницы

Добавим в элемент body страницы с подключенным Bootstrap следующую разметку:

```
<!-- <header></header> -->
<header>
</header>

<!-- <main></main> -->
<main class="container" role="main">
</main>

<!-- <footer></footer> -->
<footer>
</footer>
```

Элементы header, footer и main по отображению и машинному представлению ничем не отличаются от div, но для программиста, который формирует разметку, названия элементов, несущие явный смысл, позволяют минимизировать ошибки. Их использование рекомендуется HTML 5 и делает разметку “семантической”.

Верхняя часть страницы (header)

В верхней части страницы чаще всего размещают навигационную панель (navbar). Добавим в header навигационную панель.

```
<!-- <header></header> -->
<header>
    <nav class="navbar navbar-expand-lg navbar-dark my-primary">
        <a class="navbar-brand" href="#">Navbar</a>
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="#">Home
                    <span class="sr-only">(current)</span>
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Link</a>
            </li>
```

```
        </ul>
    </nav>
</header>
```

Разметка представляет собой упрощенный пример с сайта Bootstrap, элемент nav, как и main, footer и header, создан для упрощения понимания разметки, и ведет себя аналогично div.

Добавим в элемент head элемент style и определим класс my-primary:

```
<!-- <style></style> -->
<style>
    .my-primary {
        background-color: #1A237E;
        color: white;
    }
</style>
```

После выполнения указанных действий страница будет выглядеть так, как это показано на рис. 15.

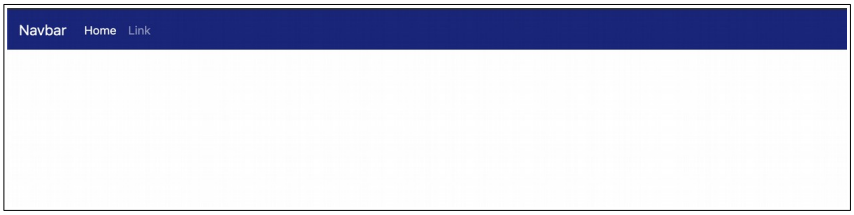

Рис. 15. Внешний вид верхней части страницы

Основное содержимое страницы (main)

Пусть страница пользователя разделена на две неравных части: левая (⅓ страницы) содержит «аватар», правая — информацию о пользователе. Добавим следующую разметку в элемент main:

```
<!-- <main></main> -->
<main class="container" role="main">
    <div class="row">
        <div class="col-4">в
              <img src=" http://e-learning.bmstu.ru/moodle/plugin-
file.php/7546/mod_folder/content/0/avatar.jpg"
                  alt="avatar" width="150" height="150">
        </div>
        <div class="col-8">
            <p>Любой текст</p>
        </div>
    </div>
</main>
```

Обратите внимание на использование атрибута alt элемента img и ограничение ширины и высоты изображения с помощью атрибутов width и height.

Для того, чтобы между навигационной панелью и содержимым оставалось свободное место, можно добавить в элемент style следующее правило:

```
main {
    margin-top: 15px;
}
```

После выполнения указанных действий страница будет выглядеть так, как это показано на рис. 16.

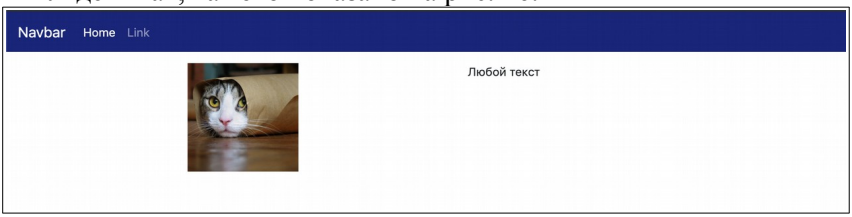

Рис. 16. Внешний вид основной и верхней частей страницы

Подвал (footer)

Подвалом сайта называют нижнюю его часть, на которой обычно размещают ссылки на второстепенную информацию, адреса и телефоны организации и т.п. Добавьте в элемент footer следующую разметку:

```
<!-- <footer></footer> -->
<footer class="my-primary">
        <div class="container">
            <p class="footer-text">Footer</p>
        </div>
</footer>
```

В элемент style необходимо добавить следующие стили:

```
html {
    position: relative;
    min-height: 100%;
}
body {
    margin-bottom: 40px;
}
footer {
    height: 40px;
    position: absolute;
    bottom: 0;
    width: 100%;
}
```

Благодаря наличию класса my-primary подвал сайта будет иметь тот же цвет фона и текста, что и навигационная панель. Стиль элемента footer задает расположение подвала всегда внизу страницы. Правило position: absolute говорит браузеру, что footer должен быть вне нормального потока, а правило bottom: 0 — что нижняя граница обрамляющего его прямоугольника должна совпадать с нижней границей окна просмотра браузера. Ширина и высота заданы явно потому, что браузер будет пытаться задать наименьшее возможное значение по умолчанию, в то время как нам требуется растянуть подвал на всю ширину страницы.

Дополнительные стили для html и body позволяют одинаково корректно отображать футер внизу страницы как при наличии, так и при отсутствии прокрутки в браузере.

Применение рассмотренной разметки приведет к тому, что браузер будет показывать следующую HTML-страницу (см. рис. 17).

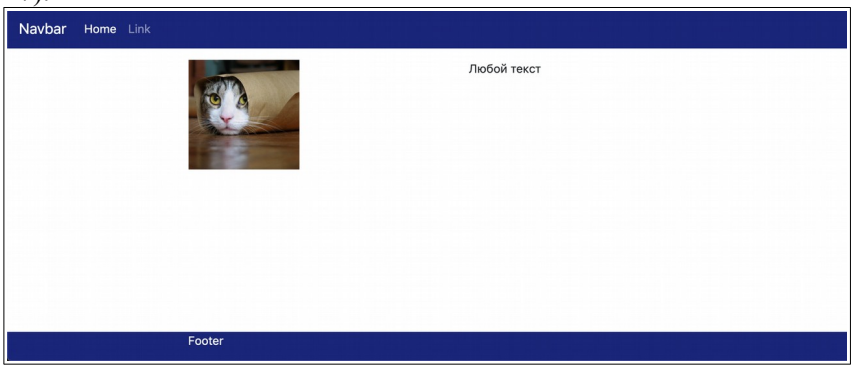

Рис. 17. Внешний вид размеченной страницы

Некоторые специфичные классы Bootstrap

Кратко рассмотрим некоторые дополнительные примеры разметки с помощью Bootstrap. Наиболее полную информацию о всех доступных возможностях библиотеки можно получить из официальной документации:
https://getbootstrap.com/docs/4.0/getting-started/introduction/.

Кнопки

Для разметки кнопок используются классы btn в комбинации со спецификаторами внешнего вида: btn-primary, btn-danger, btn-info. Пример:

```
<button class=”btn btn-primary”>Кнопка</button>
```

Все возможные варианты внешнего вида кнопок представлены на рис. 18. Обратите внимание, что имя класса отображения соответствует обозначению на кнопке. Например btn-primary, btn-secondary, btn-danger….

Рис. 18. Различные варианты отображения кнопок

Также приведенные классы могут быть применены при разметке ссылок и для <input type=”submit”/>. Более подробная информация доступна по ссылке
https://getbootstrap.com/docs/4.0/components/buttons/.

Поля ввода формы

В Bootstrap предусмотрены специальные классы для разметки полей ввода формы (класс form-group для блока и класс form-control для элемента формы):

```
<div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control"
           id="exampleInputEmail1" placeholder="Enter email">
</div>
```

Радиокнопка может быть добавлена следующим образом (класс form-check-input для элемента input):

```
<div class="form-check">
     <input class="form-check-input" type="radio"
            name="exampleRadios" id="exampleRadios1"
            value="option1" checked>
     <label class="form-check-label" for="exampleRadios1">
        Default radio
     </label>
</div>
```

Более подробная информация доступна по ссылке https://getbootstrap.com/docs/4.0/components/forms/.

Классы для таблицы

Верстка таблиц не отличается от рассмотренного ранее подхода. Отметим, для элемента table доступны классы «table», «table-bordered», «table-striped», «table-hover» и пр. Полное описание всех доступных возможностей Bootstrap доступно по ссылке https://getbootstrap.com/docs/4.0/content/tables/.

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 для Mozilla Firefox. Ранее этот отладчик был реализован как модуль-дополнение. В современных версиях Firefox, он встроен по-умолчанию. 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. 19). В Firebug доступны следующие закладки:

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

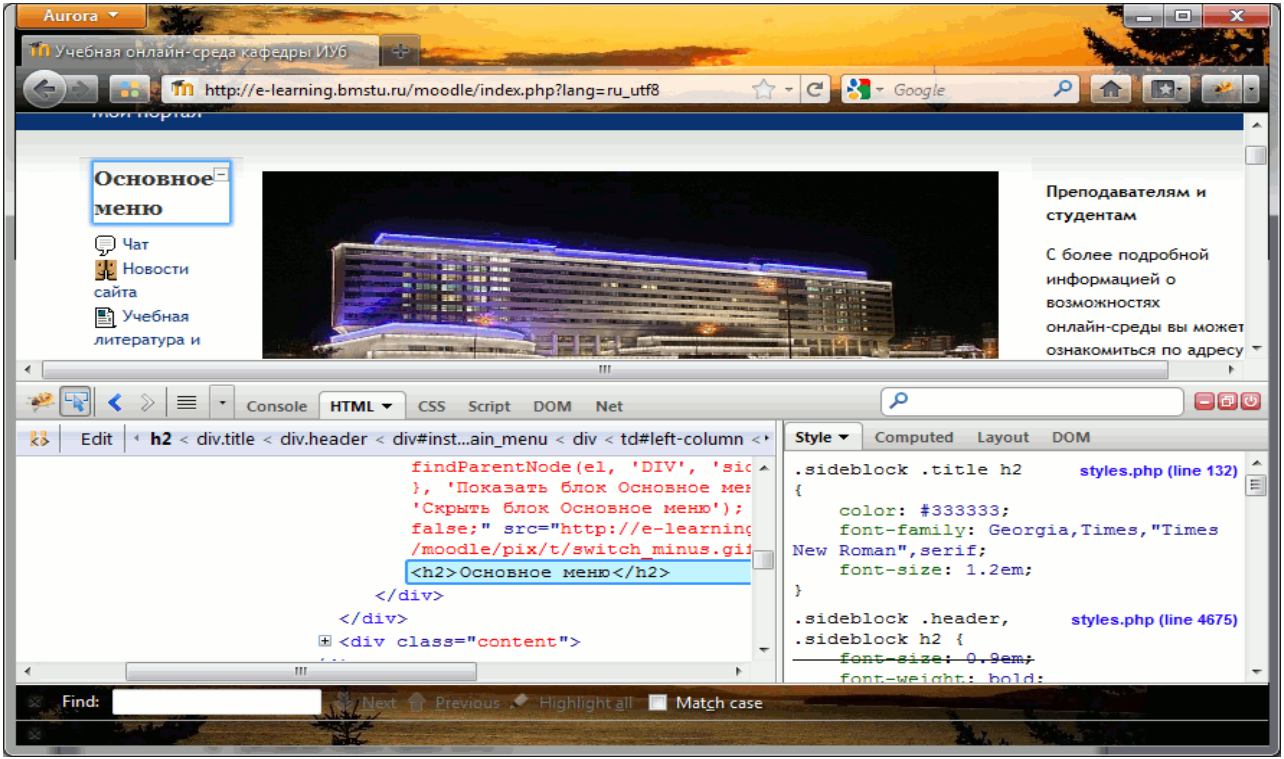

Рис. 19. Окно просмотра структуры и стилей в Firebug

• 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

Часть 1

1. Подготовьте разметку произвольного текста, содержащего не менее 10 строк (могут быть использованы материалы из лабораторной работы № 1) с использованием таблицы стилей. Продемонстрируйте выделение отдельных слов с помощью стилей, цвета и шрифта.
2. С использованием элементов div подготовьте разметку таблицы, например, содержащей фрагмент расписания.

Часть 2

3. Возьмите шаблон страницы Bootstrap (см. Приложение Б). Измените цвет фона навигационной панели и подвала сайта на свое усмотрение.
4. Вставьте:
   - на место текста “Вставьте сюда форму” форму из лабораторной работы 1;
   - на место текста “Вставьте сюда таблицу” произвольную таблицу (на основе элементов table/tr/td);
   - на место текста “Вставьте сюда текст” блок разметки текста (из пункта 1).

Добавьте классы Bootstrap в элементы формы, и заголовки таблицы.

5. Проверьте полученные HTML-страницы на наличие ошибок. Составьте таблицу выявленных ошибок, в которую внесите все ошибки валидации и их фактические проявления в браузере. Устраните все найденные ошибки.

Отчет по выполненной работе должен содержать все тексты HTML по каждому пункту и должен быть оформлен в виде pdf-файла.

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. HTML и CSS. Разработка и дизайн веб-сайтов. М.: Эксмо, 2017.

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.

Приложение А. 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>
<html>
<head>
    <meta 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>
<html>
<head>
    <meta 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 charset="utf-8">
    <style type="text/css">
        #content {
            position: relative;
            left: 50px;
            top: 20px;
            border: 1px solid green;
        }
    </style>
</head>
<body>
    <p>В ночные сборки Firefox добавлена реализация новой
       служебной страницы "about:certificate", предлагающей
       расширенный интерфейс для просмотра применяемых для
       шифрования сертификатов. </p>
    <div id="content"><p>Реализация интерфейса просмотра
сертификатов полностью переписана с использованием JavaScript и
стандартных web-технологий, а также приведена в соответствие со
стилевым оформлением Firefox Quantum.</p></div>
    <p>Новый режим просмотра опционально станет доступен через
страницу about:certificate в выпуске Firefox 70.</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 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. А.1.

Таблица А.1

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

Окончание табл А.1

Значение	Описание
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="ru">
<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.

Приложение Б. Полный текст разметки с помощью Bootstrap

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0">
    <title>ICS6 Bootstrap</title>
    <link
        href="http://e-learning.bmstu.ru/moodle/pluginfile.php/754
6/mod_folder/content/0/bootstrap.min.css" rel="stylesheet">
    <style>
        html {
            position: relative;
            min-height: 100%;
        }

        body {
            margin-bottom: 40px;
        }

        main {
            margin-top: 15px;
        }

        footer {
            height: 40px;
            position: absolute;
            bottom: 0;
            width: 100%;
        }

        .my-primary {
            background-color: #1A237E;
            color: white;
        }
    </style>
</head>
<body>
    <header>
      <nav class="navbar navbar-expand-lg navbar-dark my-primary">
            <a class="navbar-brand" href="#">Navbar</a>

            <ul class="navbar-nav mr-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="#">Home
                    <span class="sr-only">(current)</span>
                </a>
            </li>
            <li class="nav-item">
```

```
                <a class="nav-link" href="#">Link</a>
            </li>
        </ul>
    </nav>
    </header>

    <main class="container">
        <div class="row">
            <div class="col-4">
            <img
      src=" http://e-learning.bmstu.ru/moodle/pluginfile.php/7546/
mod_folder/content/0/avatar.jpg"
                    alt="avatar" width="150" height="150">
            </div>

            <div class="col-8">
                <p>Любой текст</p>
            </div>
        </div>

        <div class="row">
            <div class="col-4">
                Вставьте сюда форму
            </div>

            <div class="col-4">
                Вставьте сюда таблицу
            </div>
        </div>

        <div class="row">
            <div class="col">
                Вставьте сюда текст
            </div>
        </div>
    </main>

    <footer class="my-primary">
        <div class="container">
            <p class="footer-text">Footer</p>
        </div>
    </footer>
</body>
</html>
```

Содержание

Учебное издание

Самарев Роман Станиславович
Кучеров Кирилл Владимирович

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

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК