ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 1.

Тема: Язык гипертекстовой разметки HTML. Каскадные таблицы стилей CSS

Часть 1. Общие сведения о HTML, XHTML. Основные элементы разметки

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

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

Ключевые даты и стандарты:

- RFC 1866 — HTML 2.0, одобренный как стандарт 22 сентября 1995 года;
- HTML 3.2[1] — 14 января 1997 года - W3C;
- HTML 4.0[2] — 18 декабря 1997 года;
- HTML 4.01[3] (изменения, причём более значительные, чем кажется на первый взгляд) — 24 декабря 1999 года;
- ISO/IEC 15445:2000[4] (так называемый ISO HTML, основан на HTML 4.01 Strict) — 15 мая 2000 года.
- HTML 5— в разработке. Конец разработки запланирован на 2014 год.

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

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

XHTML 1.0 Strict:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

XHTML 1.0 Mobile:

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
```

XHTML 1.1:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Структура HTML разметки

Схематично разметку любой страницы можно представить следующим образом[2]:

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

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

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

Каждый элемент может быть представлен одним или двумя тегами и содержать или не содержать внутри себя другие элементы.
Теги выделяются специальными знаками < >, причём открывающий тег выглядит как <*имя элемента*>, а закрывающий как </ *имя элемента*>. В случае XHTML, используется тег вида <*имя элемента* /> для указания элементов без закрывающих тегов.

---

[2] Справочники по HTML см. http://www.w3schools.com/, http://www.w3.org/

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

Основные элементы разметки

Рассмотрим некоторые наиболее часто используемые элементы разметки.

Элемент html

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

Элемент head

Не является обязательным. Представляет собой контейнер, в котором размещаются элемента заголовка документа.
Внутри элемента head могут быть определены следующие элементы:

Таблица 1 допустимые элементы внутри head.

Тэг	Описание
<title>	Определяет заглавие документа. Обычно браузеры отображают этот заголовок в заголовке окна.
<base />[3]	Определяет базовый адрес или адрес по умолчанию для всех ссылок на странице. Ссылки на странице, содержащие относительный путь будут дополняться базовым адресом. При перемещении такого документа все ссылки будут сохранены.
<link />	Определяет связи между документом и внешними ресурсами. Например подключение стилей отображения.
<meta />	Определяет метаданные о документе. Среди них могут быть как дополнительные ключевые слова, используемые для поиска, так и указание кодировки страницы.
<script>	Определяет скрипты, выполняемые на стороне браузера.
<style>	Определяет информацию о стилях отображения в документе. В отличии от <link> эти стили будут внедрены в документ.

Следует отметить, что для страниц на русском языке обязательно необходимо указывать кодировку текста. Примеры[4]:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<meta http-equiv="Content-Type" content="text/html; charset=koi8-r" />
```

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

---

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

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

Элемент body

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

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

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

Некоторые элементы разметки текста

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

Таблица 2 Некоторые элементы разметки текста.

Тэг	Описание

<p>	Параграф. Позволяет указать область текста, относящуюся к одному параграфу. Параграфы при отображении обычно отделяются увеличенным отступом. В рамках параграфа игнорируются переводы строк в исходном тексте. Формат переноса может быть задан стилями, но не имеет отношения к исходной разметке
	
	Пример:
	`<p>Некоторый`
	`текст.</p>`
	Будет выведено:
	Некоторый текст.

 	Перевод строки в любом месте текста, включая параграфы. Параграф при этом не разрывается. *Пример:* `<p>Некоторый  текст.</p>` *Будет выведено:* Некоторый текст.

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

Таблица 2. Продолжение

Тэг	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 … . Имеются атрибуты (устаревшие) start, позволяющий указать стартовый номер, а также type, позволяющий указать тип нумерации: 1(арабские числа), A, a (буквы латинского алфавита), I,i (римские числа).
	
	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.
Другой обязательный атрибут – **alt**. Его использование необходимо для того, чтобы в браузерах, которые не могут отобразить изображение, вместо изображения был выведен текст.
Обратите внимание на то, что значение атрибута **alt** также используется поисковыми системами для того, чтобы ассоциировать изображение, указанное в атрибуте **src** с текстом.

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

Пример:
```
<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. Обязательный атрибут – 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 используется элемент <table>, который является контейнером для элементов <tr> (строка), <th> (заголовок), и <td> (колонка).

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

Пример:

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

Результат представлен на рисунке 1.

Наименование	**Количество**
Ручка	10
Карандаш	20

Рисунок 1 Пример отображения таблицы

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

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

Пример:

```
<table width="100%" border="1">
   <tr>
     <th colspan="2">Назв. &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 представлен на рисунке 2.

Назв. & Кол.		Всего
Ручка	10	30
Карандаш	20	

Рисунок 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 должны быть заменен блочной разметкой, а именно элементами и .
Элемент используется для выделения фрагментов текста, например для того, чтобы выделить цветом фрагмент внутри параграфа. Обратите внимание на то, что параграф при этом остаётся целым.
Пример:

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

Отображение:

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

Элемент 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; Кол.</th>
           <th>Всего</th>
         </tr>
         <tr>
           <td>Ручка</td>
           <td>10</td>
           <td rowspan="2">30</td>
         </tr>
```

```
          <tr>
            <td>Карандаш</td>
            <td>20</td>
          </tr>
        </table>
      </div>
      <div id="div2">
            <p>
                  Некоторый текст, часть которого <span
style="color:red">выделена красным</span>.
            </p>
 </div>
</body>
</html>
```

Результат представлен на рисунке 3.

Некоторый текст, часть которого выделена красным.

Назв. & Кол.		Всего
Ручка	10	30
Карандаш	20	

Рисунок 3 Пример отображения таблицы внутри блока

Более подробно см. раздел блочной модели http://www.w3.org/TR/CSS2/box.html.

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 представлен на рисунке 4.

Имя:
Фамилия:
Пол: м ж
Отправить

Рисунок 4 Пример отображения формы

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

Таблица 3. Значения атрибута type тэга input.

Тип	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 примере `name="gender"`
type="checkbox"	Селектор в виде квадрата. В отличии от типа radio, предполагает только два состояния: пункт выделен или не выделен
type="submit"	Кнопка выполнения стандартного действия подтверждения.
type="button"	Произвольная кнопка. Форма может иметь несколько кнопок.

Устаревшие элементы форматирования

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

Таблица 4 Устаревшие элементы разметки

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

Использование валидаторов HTML и CSS

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

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

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

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

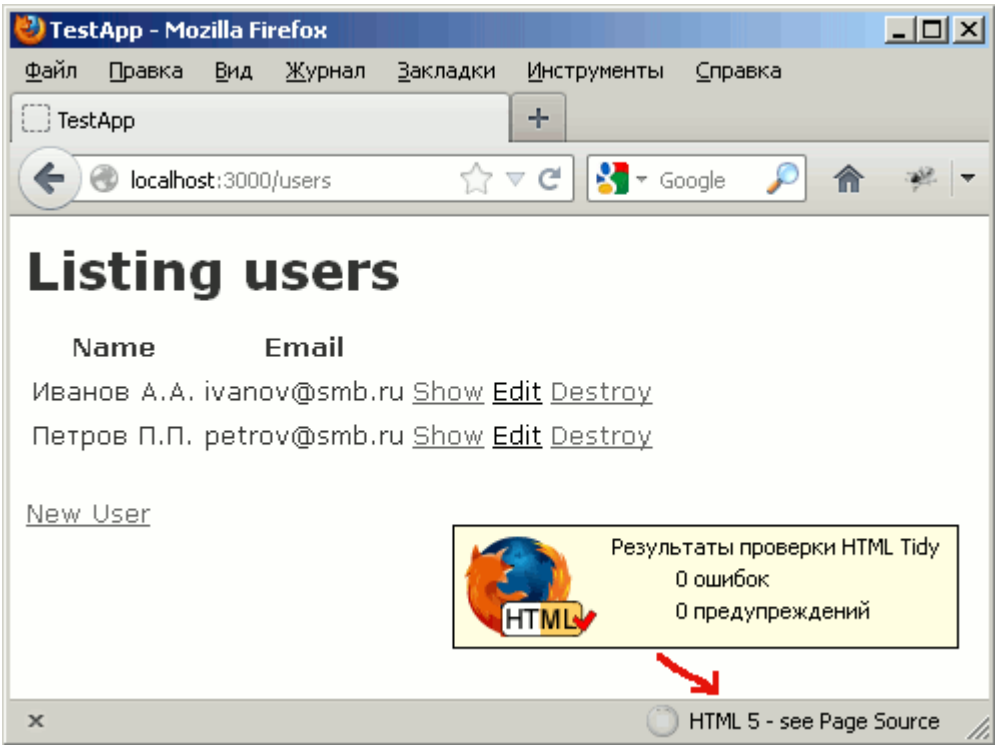


Рисунок 5 Информация от валидатора о корректной странице

Для примера внесём несколько ошибок в разметку HTML и CSS.

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

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

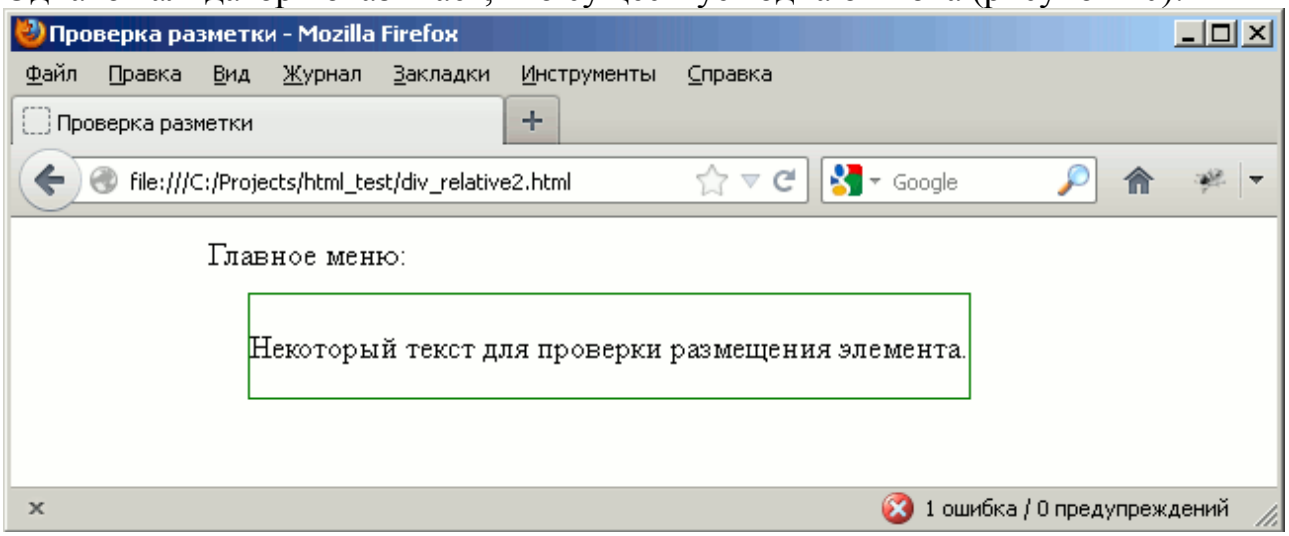


Рисунок 6 Предупреждение валидатора о некорректной странице.

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

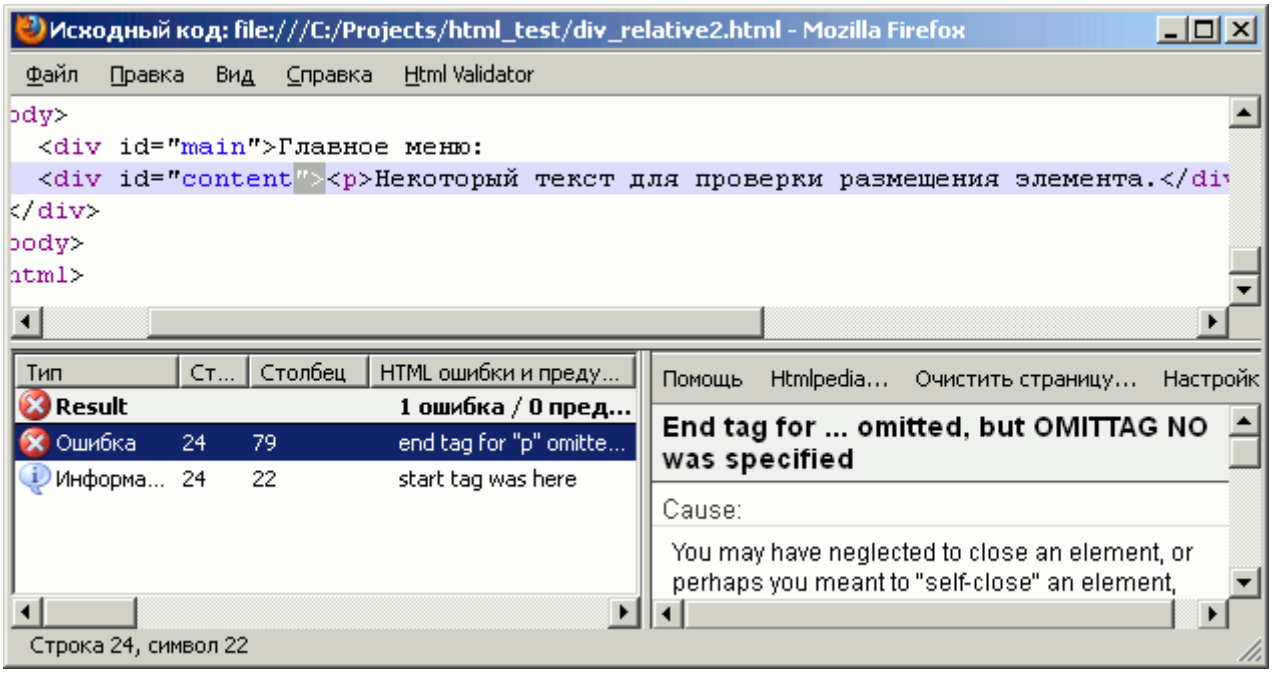


Рисунок 7 Просмотр кода и сообщение об ошибке.

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 получаем сообщение, представленное на рисунке 12.

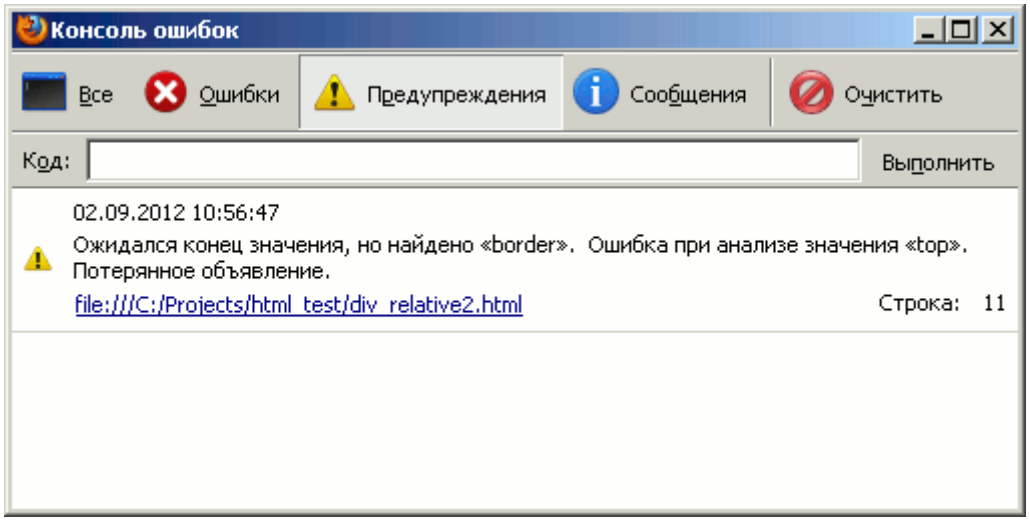


Рисунок 8 Просмотр ошибок отображения браузера.

Итого, по результатам проверки выявлено:

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

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

2. Нарушены таблицы стилей, поскольку отсутствует разделитель ‘;’ между селекторами:

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

Часть 2. Cascading Style Sheets (CSS)

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

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

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

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

Селекторы

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

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

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

Группы селекторов (эквивалентно выше приведённому фрагменту):

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

Селекторы класса:

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

Или

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

А также:

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

Селекторы идентификатора ID:

```
h1#chapter1 { font-family: sans-serif } /* для <h1 id="chapter1">…</h1> */
#chapter1 { font-family: sans-serif } /* для любого элемента с  id="chapter1" */
```

Селекторы атрибутов:

```
h1[class] { font-family: sans-serif } /* элемент имеет class */
h1[class=”fancy”] { font-family: sans-serif } /* элемент имеет class=”fancy” */
*[title] { font-family: sans-serif } /* любой элемент, имеющий заголовок */
```

Селектор потомков (устанавливает иерархию применения):

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

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

```
a:link    /* ссылки, которые не были посещены */
a:visited /* посещенные ссылки */
a:hover   /* выделенная в данный момент ссылка */
a:active  /* активные ссылки */
```

Шрифты

При оформлении страницы доступны следующие семейства шрифтов:

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

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

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

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

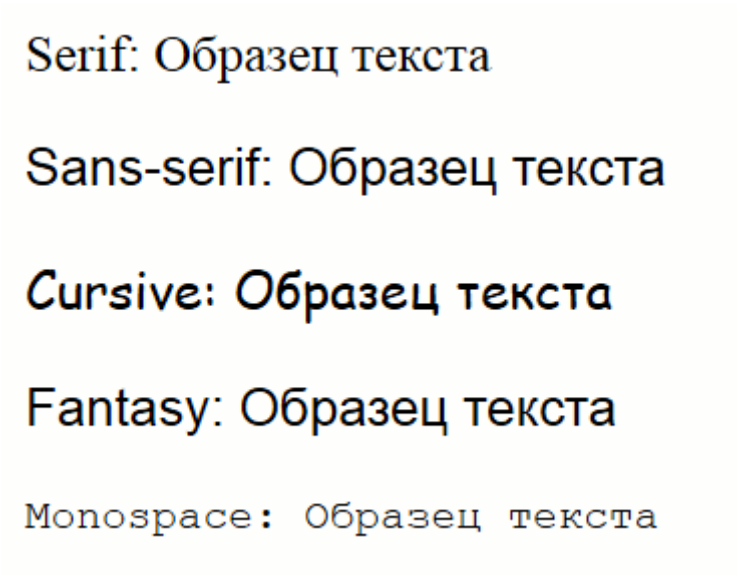


Рисунок 9 Пример отображения шрифта в Mozilla Firefox 5

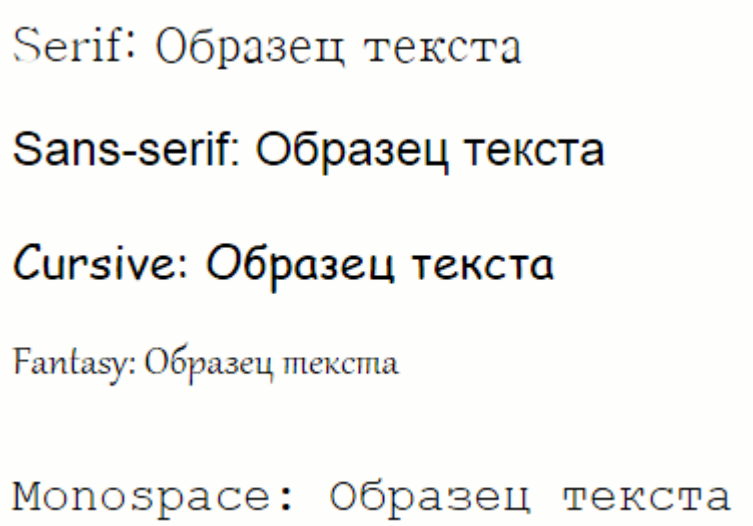


Рисунок 10 Пример отображения шрифта в MS Internet Explorer 9

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. Более того, для каждого семейства шрифтов, различные браузеры в различных операционных системах могут иметь различные конкретные шрифты, соответствующим этим семействам.

Свойство font-family может содержать перечисление шрифтов:

```
p{font-family:"Times New Roman", Times, serif;}
```

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

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

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

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

normal – обычное начертание, italic – курсив (имитация рукописного шрифта), oblique – наклонный шрифт (образован наклоном обычного).
Размер шрифта задаётся с помощью свойства font-size.
Возможные значения:

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

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

Пример:

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

```
      <p id="p7">Образец текста</p>

</body>
</html>
```

Цвет

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

Пример:

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

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

Фон

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

Используются следующие свойства:

background-color – однородный цвет константой или кодом в одной из допустимым систем цветности.
Пример: `div {background-color:#b0c4de;}`

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

background-repeat – флаг мозаичного размножения изображения.
Пример:

```
body {
   background-image:url('gradient2.png');
   background-repeat:repeat-x;
}
```

background-attachment – указывает, будет ли изображение смещаться при скроллировании или будет оставаться на месте.
Пример: `background-attachment:fixed;`

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

Пример:

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

Преобразование текста

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

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

Некоторые примеры использования:

```
footer { text-wrap: avoid; /* inherits to all descendants */ }
…
<footer>
  <venue>27th Internationalization and Unicode Conference</venue>
  &#8226; <date>April 7, 2005</date> &#8226;
  <place>Berlin, Germany</place>
</footer>
```

В узком окне текст будет отображен как:

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

Если окно еще больше сужено, то текст будет выглядеть:

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

Но ни при каких условиях текст не будет выглядеть как:

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

Блочная модель

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

Отступы и границы

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

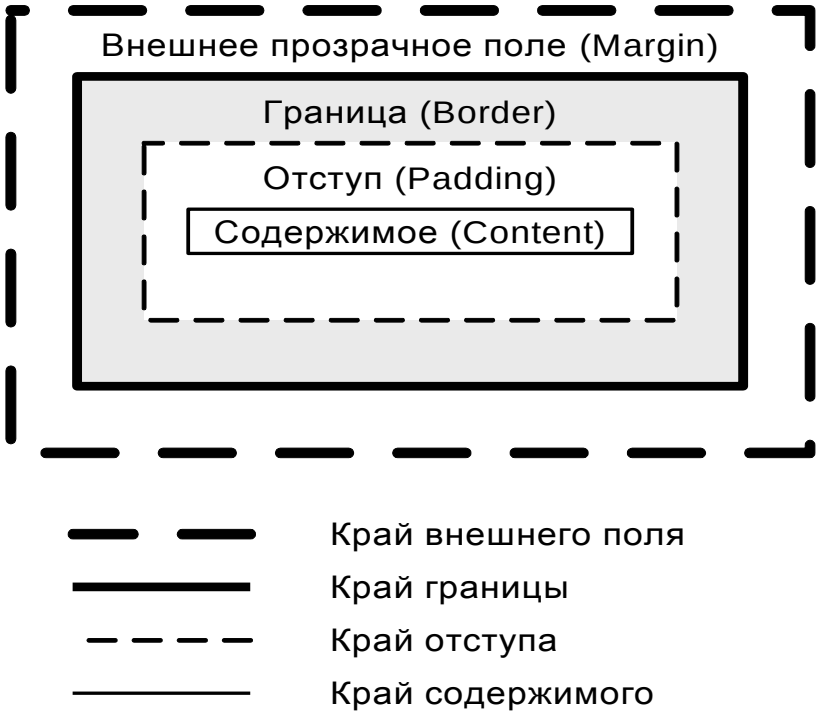


Рисунок 11 Блочная модель CSS

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

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

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

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

Граница представляет собой видимое обрамление элемента с указанным начертанием, цветом и толщиной. Граница может быть задана единственным свойством border или отдельно для каждой стороны блока: 'border-top', 'border-right', 'border-bottom', 'border-left'. Указываются толщина, тип начертания границы и цвет.

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

Таблица 5 Обозначение различных способов начертания границ.

Стиль	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

При расчёте размеров блока необходимо помнить про толщину границы.

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

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

Позиционирование

CSS поддерживает 4 вида позиционирования:

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

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

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

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

Абсолютное позиционирование
Абсолютное позиционирование подразумевает указание расположения относительно его блока-контейнера или корневого элемента html. При этом, как только появляется абсолютное позиционирование, элемент выпадает из нормального потока и всегда будет позиционироваться относительно контейнера, независимо от остального содержимого страницы.
Рассмотрим простейший пример позиционирования. В данном случае позиция будет совпадать с левым верхним отступом от окна отображения.

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

Рассмотрим другой пример, включающий абсолютное позиционирование относительно другого блока.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <style type="text/css">
            #main {
                  position: absolute;
                  left: 100px;
                  top: 50px;
                  border: 1px solid black;
                  padding: 0 100px 100px 0                  }
            #content {
                  position: absolute;
                  left: 50px;
                  top: 20px;
                  border: 1px solid green;
                  }
      </style>
</head>

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

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

Относительное позиционирование

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

В следующем примере блок с идентификатором content смещен относительно нормального потока, но элемент, расположенный за ним, будет отображен так, как будто никаких изменений в потоке не было. Обратите внимание на то, что этот блок будет менять положение при изменении размеров окна браузера.

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

```
            #content {
                  position: relative;
                  left: 50px;
                  top: 20px;
                  border: 1px solid green;
                  }
      </style>
</head>
<body>
      <p>Проект Mozilla официально выпустил релиз web-браузера Firefox 15. Кроме того, выпущен корректирующий релиз ветки с длительным сроком поддержки - Firefox 10.0.7, в котором отмечается только исправление уязвимостей и серьёзных ошибок.</p>
  <div id="content"><p>В ближайшие дни на стадию бета-тестирования перейдёт 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>
```

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

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

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

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

Плавающие элементы
Плавающее размещение не является схемой позиционирования. Оно было введено как средство, позволяющее получить обтекание элементов, но не для создания макета страницы.
Например следующие: стиль обеспечит отображение изображений в правой части страницы, а все остальные элементы будут размещать в свободном пространстве слева.

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

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

Управление отображением элемента
Для управления отображением элемента используется свойство display.

Таблица 6 Некоторые значения свойства diplay.

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

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

Макеты. Размещение элементов по колонкам, замена фреймов и таблиц

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

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

```
* {
      margin: 0;
      padding: 0;
}
html {
      height: 100%;
}
body {
      font: 12px/18px Arial, Tahoma, Verdana, sans-serif;
      width: 100%;
      height: 100%;
}
a {
      color: blue;
      outline: none;
      text-decoration: underline;
```

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


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


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


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

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


/* Footer -----------------------------------------------------------------------*/
#footer {
      width: 1000px;
      margin: -100px auto 0;
      height: 100px;
      background: #BFF08E;
}
```

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

<body>

<div id="wrapper">

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

      <div id="middle">

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

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

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

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

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

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

</body>
</html>
```

Результат представлен на рисунке 8.

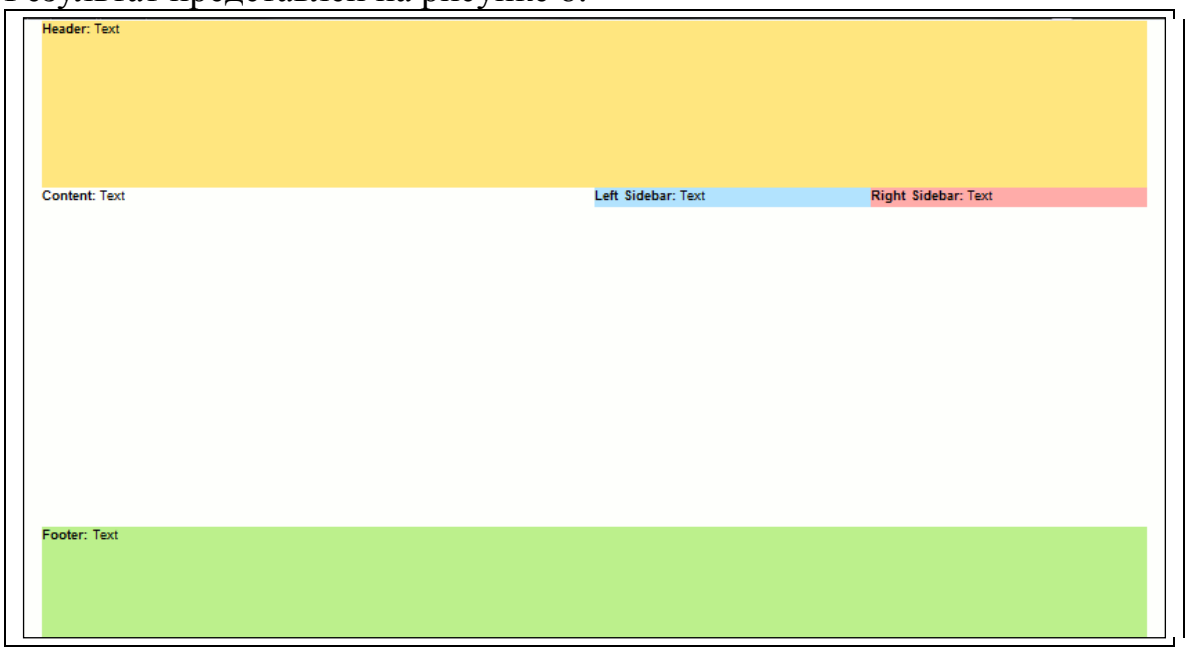


Рисунок 12 Пример отображения страницы с блоками

Отладка CSS в браузере Firefox

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

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

3. Console – сообщения браузера об ошибках, предупреждения и отладочный вывод Javascript-программ.
4. HTML – структура разметки страницы. Позволяет найти по отображенному элементу его соответствие в разметке и наоборот. Возможно интерактивное изменение стилей любых элементов.

5. CSS – позволяет изменить таблицы стилей загруженной страницы.
6. Script – Javascript код страниц. Позволяет просматривать код, устанавливать точки останова, выполнять пошаговую отладку и трассировку значений переменных. По-умолчанию выключено.
7. DOM – просмотр и изменение значение документа по модели DOM.
8. Net – просмотр данных, передаваемых между браузером и сервером. Предоставляется возможность просмотра заголовков HTTP, переданных данных, а также порядка передачи данных. По-умолчанию выключено.

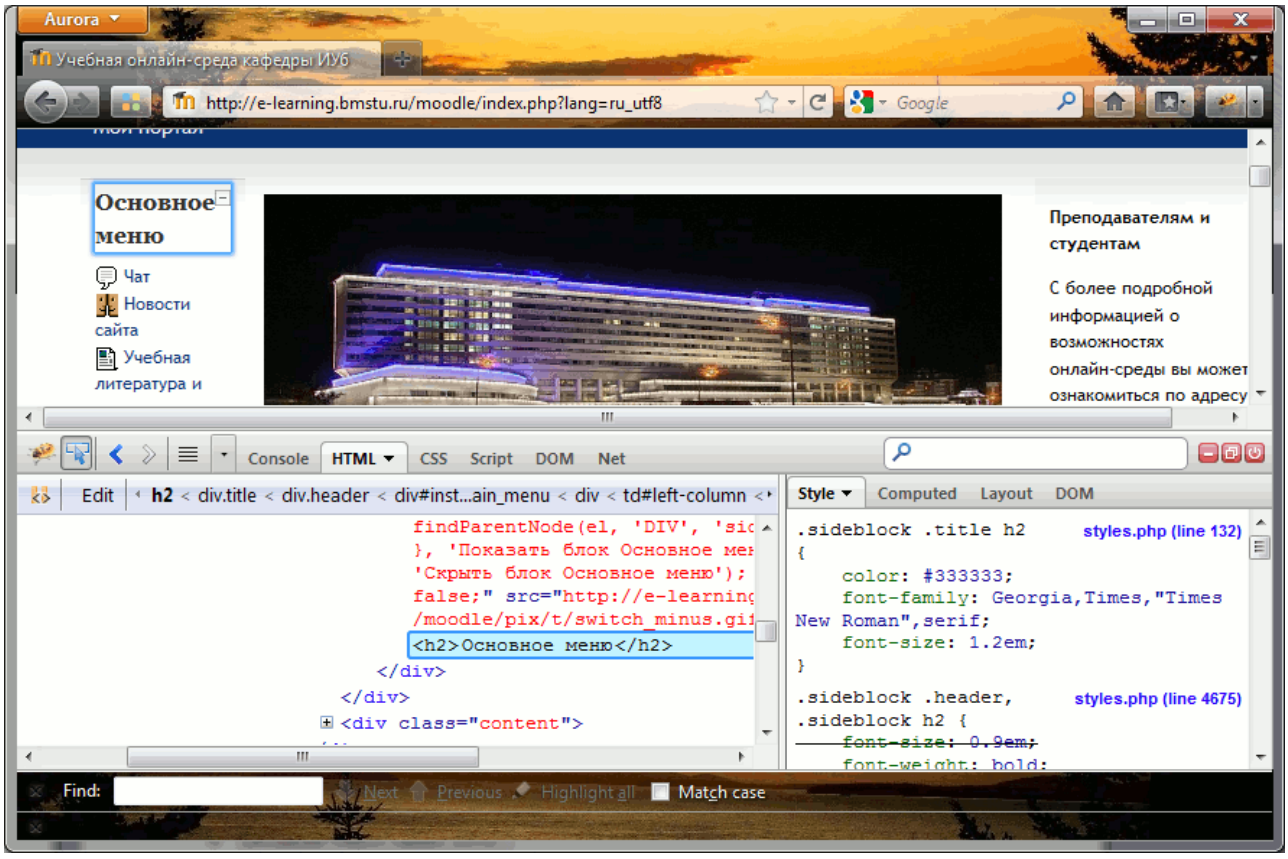


Рисунок 13 Просмотр структуры и стилей в Firebug.

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

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

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

- Style (Стиль) для редактирования стиля одного конкретного элемента. Причем Firebug группирует стили, назначенные конкретно на выделенный элемент и стили, унаследованные для этого элемента.
- Закладка Computed (Скомпилированный стиль) отображает результат применения к элементу всех назначенных правил. В отличии 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.

Задание к части 1

Разметка HTML-страницы на основе устаревших элементов.

1. Создать html-документ и указать старую спецификацию:
   <!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 на корректность всех разработанных страниц и устранить ошибки. Составить таблицу выявленных ошибок и их фактическое проявление в браузере.

Задание к части 2

Разметка страницы при помощи стилей и блоков.

1. Подготовить разметку произвольного текста, содержащего не менее 10 строк (взять из части 1) с использованием таблицы стилей. Продемонстрировать выделением стилями цвета и шрифта отдельных слов.
2. Сформировать фрагмент расписания занятий с использованием элемента div и стилей.
3. Сформировать страницу, содержащую форму с набором кнопок. На этой же странице добавить список ссылок, соответствующих кнопкам. Модифицировать стилями отображение ссылок так, чтобы они стали похожи на кнопки.
4. Проверить на корректность всех разработанных страниц и устранить ошибки. Составить таблицу выявленных ошибок и их фактическое проявление в браузере.

Контрольные вопросы

1. Назовите организацию, координирующую развитие стандартов HTML.
2. Укажите основной принцип формирования разметки в HTML.
3. Приведите примеры устаревших элементов разметки.
4. Приведите примеры селекторов CSS.
5. Приведите примеры описания шрифтов с помощью CSS.
6. Назовите основные виды вёрстки HTML и их отличия.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2-4.

Тема: Принципы проектирования и разработки веб-сайтов. Системы управления контентом CMS. Разработка веб-сайта на основе CMS.

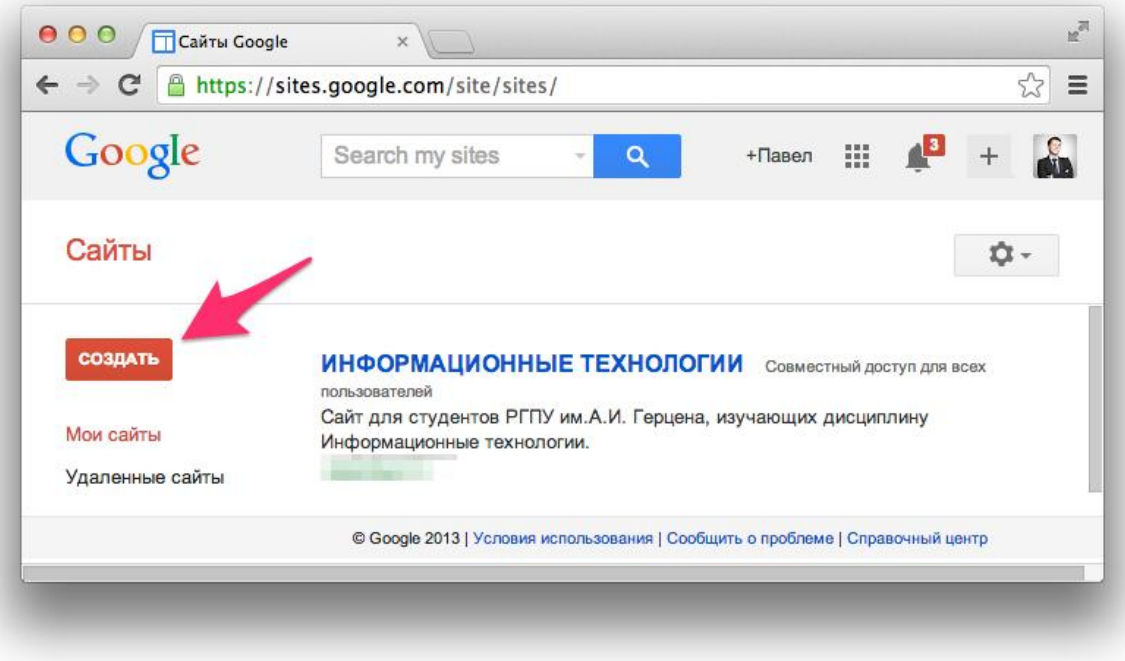
Ход выполнения работы

1) Создание сайта
2) Редактирование страниц
3) Добавление страниц, создание иерархической структуры
4) Настройка

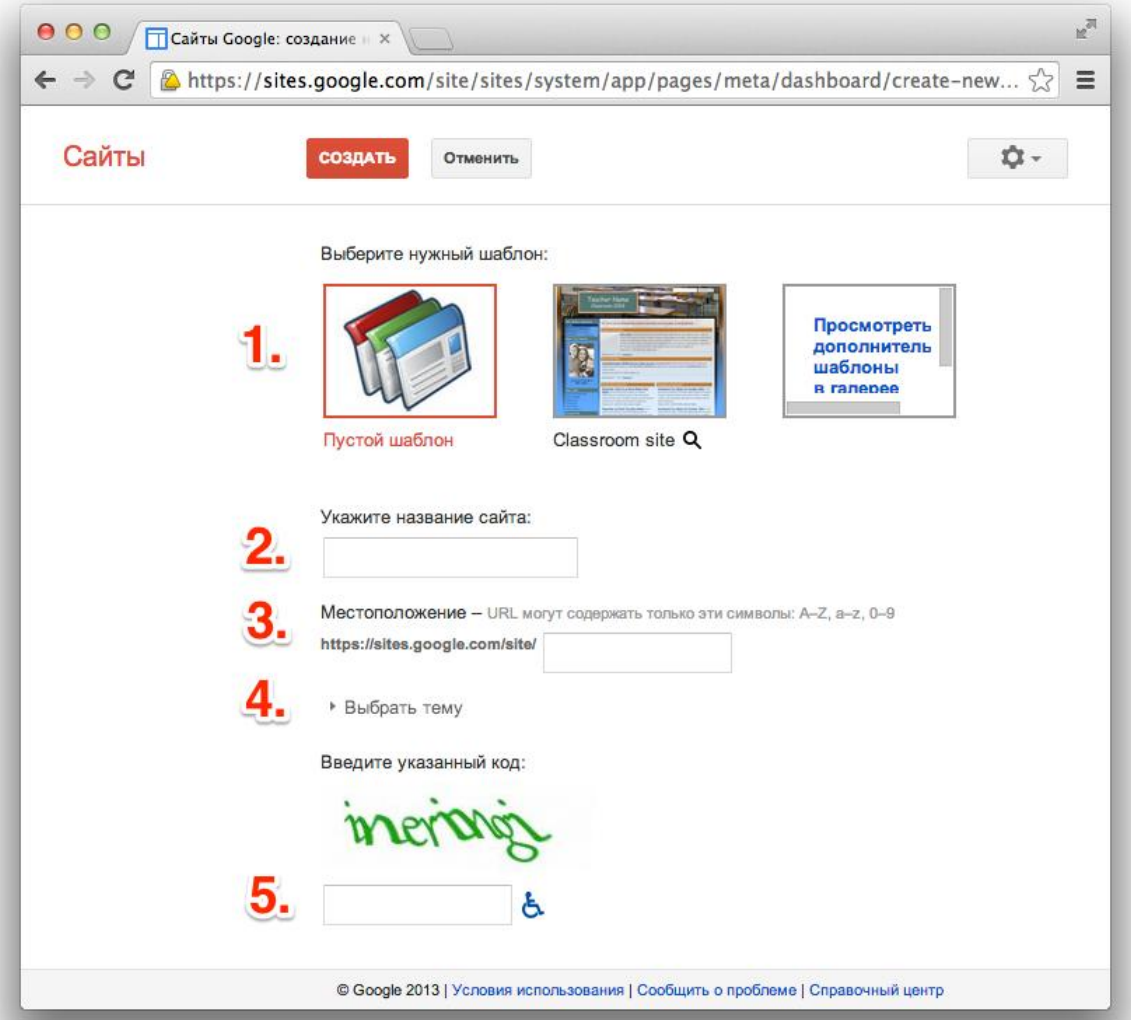
В данной лабораторной работе для создания сайта используется веб-приложение Google Сайты https://sites.google.com/. Перед началом работы необходимо войти в систему под учетной записью Google.

Текстовая инструкция:

Для создания нового сайта на главной странице веб-приложения щелкните по кнопке «Создать». Обратите внимание на то, что после создания сайта, его название будет указано в списке доступных для редактирования сайтов:



В форме создания нового сайта заполните все необходимые поля:



Комментарии к заполнению:

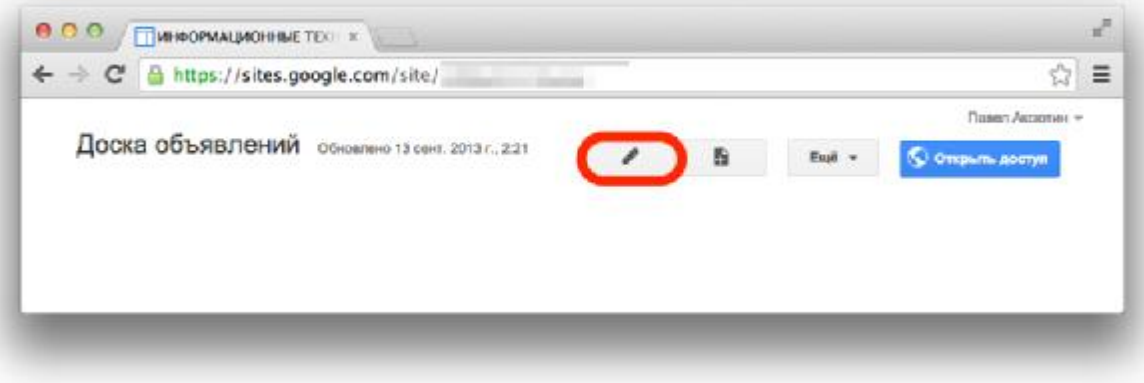
1. **Шаблон** (заданная структура и содержание сайта). Для ознакомления с различными вариантами шаблонов вы можете щелкнуть по ссылке «Просмотреть дополнительные шаблоны», однако в рамках данной работы выберите «**Пустой шаблон**».
2. **Название сайта**. Укажите краткое название сайта по-русски. Данный текст будет отображаться в верхней части сайта на всех страницах.
3. **Местоположение.** В данном поле необходимо указать последнюю (вариативную) часть адреса вашего будущего сайта. Будьте внимательны, скорее всего простые названия уже заняты, поэтому к адресу сайта нужно дописать фамилию и имя. Например: computervirusivanivanov, тогда полный адрес сайта будет выглядеть следующим образом: https://sites.google.com/site/computervirusivanivanov/
4. Выберите любое подходящее к тематике сайта оформление из списка в разделе «**выберите тему**»
5. Введите код **CAPTCHA.**

После нажатия на кнопку «Создать», Google создаст новый сайт и откроет его главную

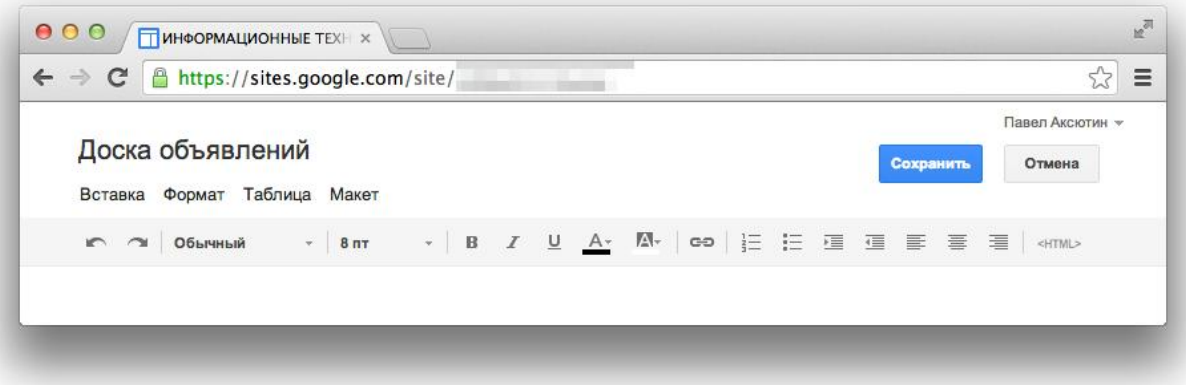
страницу.
В случае неправильного ввода одного из полей или занятости выбранного местоположения Google выдаст ошибку и после устранения проблемы неоходимо повторно нажать кнопку «Создать».

Редактирование страниц

В отличии от традиционной технологии создания сайтов на основе языков разметки и программирования, данное веб-приложение работает в режиме WYSIWYG (What YouSee Is What You Get - «Что видишь, то и получаешь»). Для создания страниц сайта необязательно использовать язык HTML, т.к. Google создаст всю необходимую разметку в автоматическом режиме на основе заданного пользователем представления.



Для редактирования страницы необходимо нажать на кнопку «Изменить страницу» в правом верхнем углу страницы). Панель инструментов редактора содержит базовые команды форматирования абзацев и текста, а основное меню позволяет вставить дополнительные материалы на страницу, управлять таблицами и макетом сайта.

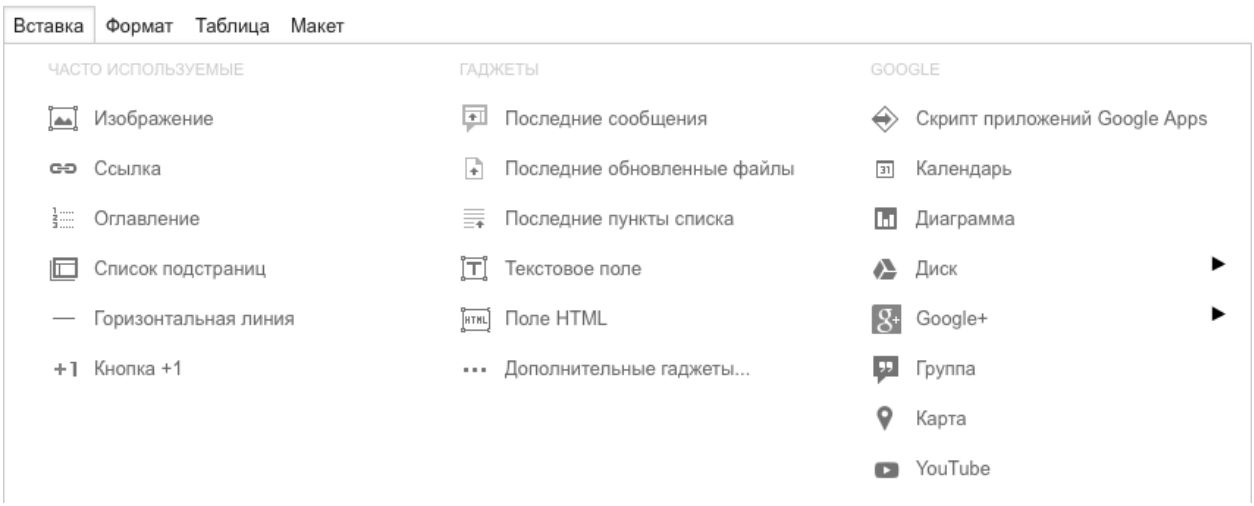


Изменение страницы в данном редакторе напоминает работу в текстовом редакторе, однако области для ввода текста строго подчиняются выбранному макету.

Для завершения работы с редактором и перехода к нормальному режиму сайта необходимо нажать кнопку «Сохранить». Google автоматически создает черновики страниц во время редактирования, поэтому потерять несохраненные изменения практически невозможно.

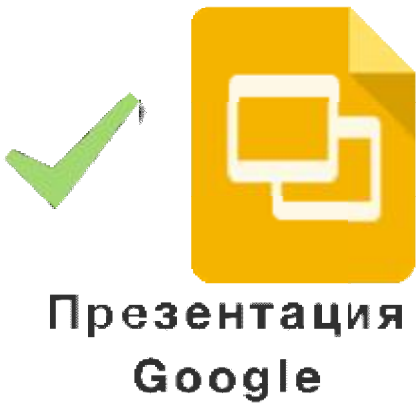
Вставка дополнительных материалов на страницу осуществляется через меню «Вставка».

Изображения и ссылки добавляются точно также, как и в блоге-портфолио Blogger.com. После выбора соответствующего пункта меню следуйте указаниям мастера.



Кроме того, в меню «вставка» доступны средства интеграции с другими сервисами Google: Календарь, карты GoogleMaps, Youtube и Google Диск. Интеграция с облачным хранилищем позволяет добавлять на страницы сайта тектовые документы, презентации, анкеты, электронные таблицы. В рамках данной лабораторной работы вам необходимо встроить презентацию и видеоролик. Создавать собственную презентацию и видеоролик не обязательно – можно найти готовые материалы в Интернете.

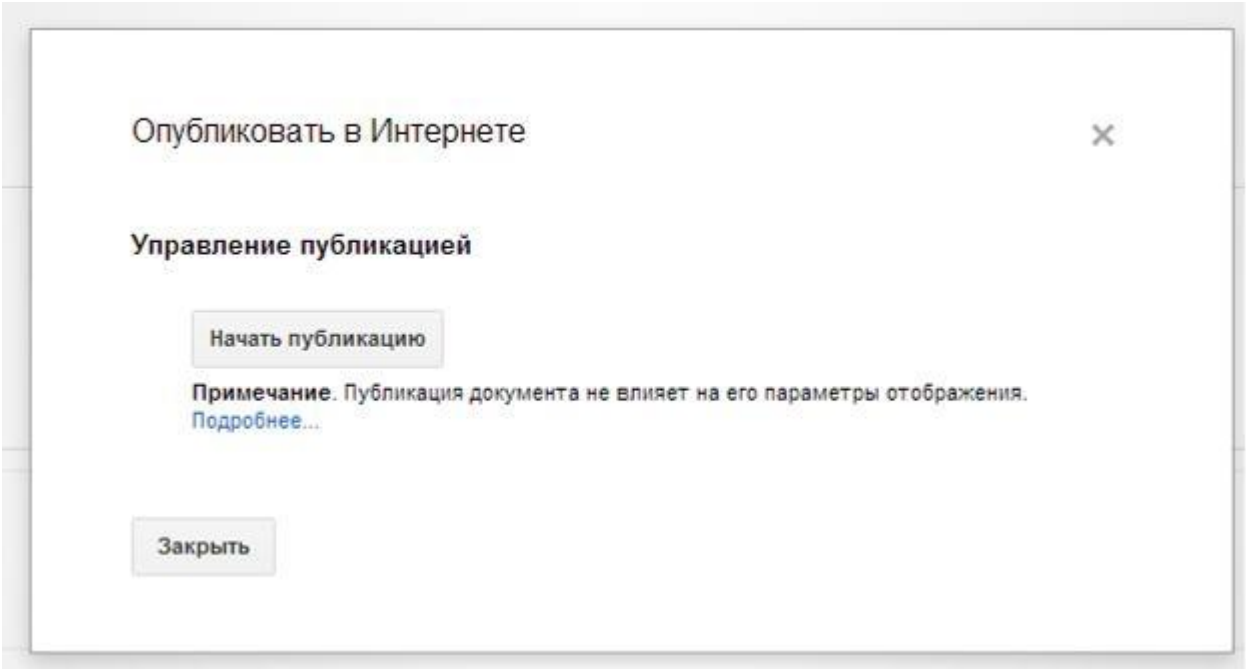
Для того, чтобы презентация была доступна для вставки на страницу, она должна быть изначально создана в редакторе GoogleDrive, либо презентация PowerPoint загружена и преобразована в документ Google.



Откройте Google Диск, убедитесь, что установлена галочка в меню «Настройки», «Настройки загрузки» напротив пункта «Преобразовывать загруженные файлы в формат Документов Google».

Загрузите файл в Google Диск, используя инструмент загрузки. Откройте презентацию и опубликуйте. Обратите внимание, что Google не сможет преобразовать презентацию, если её объем более 4Мб.

Перед встраиванием в сайт презентацию необходимо опубликовать: в Google Диске выполните команду: Файл – Опубликовать в Интернете. В открывшемся диалоговом окне нажмите на кнопку «Начать публикацию».



Получив «HTML-код для встраивания» вы можете использовать его для интеграции документа Google в любой веб-ресурс. Например, в блоге (blogger.com) вы можете перейти в режим редактирования HTML и, используя код <iframesrc="...", вставить документ непосредственно в сообщение блога.

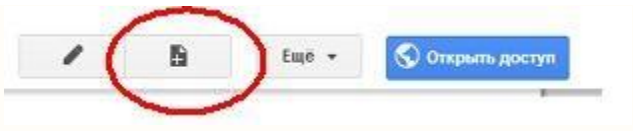
Для вставки презентации на страницу сайта Google выполните команду **Вставка – Диск – Презентация,** где в списке документов выберите необходимую презентацию.

Добавление страниц, создание иерархической структуры

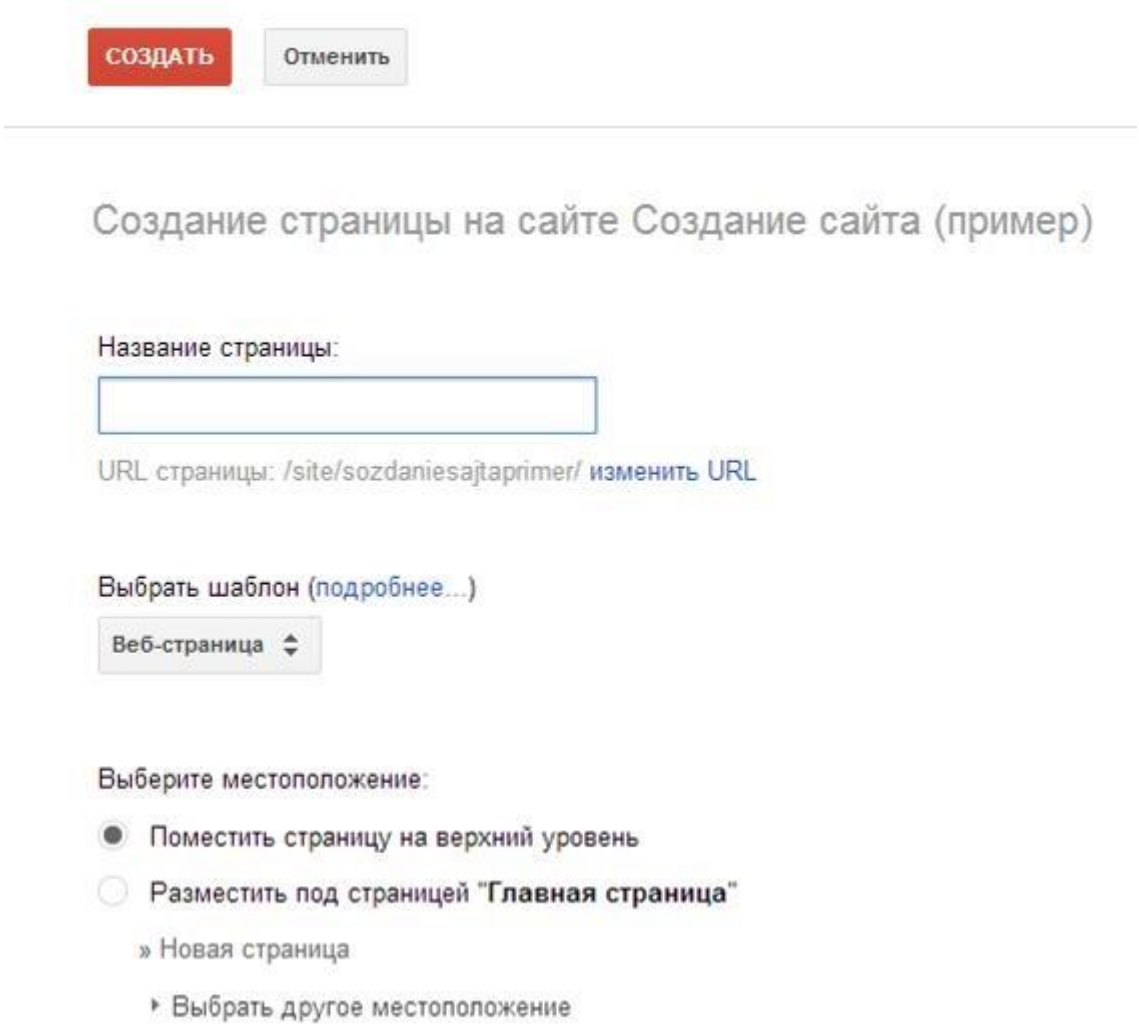
Современный веб-сайт - это набор связанных гиперссылками документов. Конечно, при более глубоком анализе поведения посетителей сайта можно заметить, что перемещения пользователя могут быть во многом случайны и более хаотичны, чем это предусмотрено картой сайта: например, перемещение к разделу «Контроль» сайта www.mediagnosis.ru возможно не только через главную страницу, но и с любой другой страницы через верхнее меню.

После добавления новых страниц Google автоматически изменяет боковое меню на вашем сайте, а также создает ссылки на подстраницы. Выбор местоположения страницы

осуществляется при создании новой страницы, либо в меню «Еще - Переместить страницу».



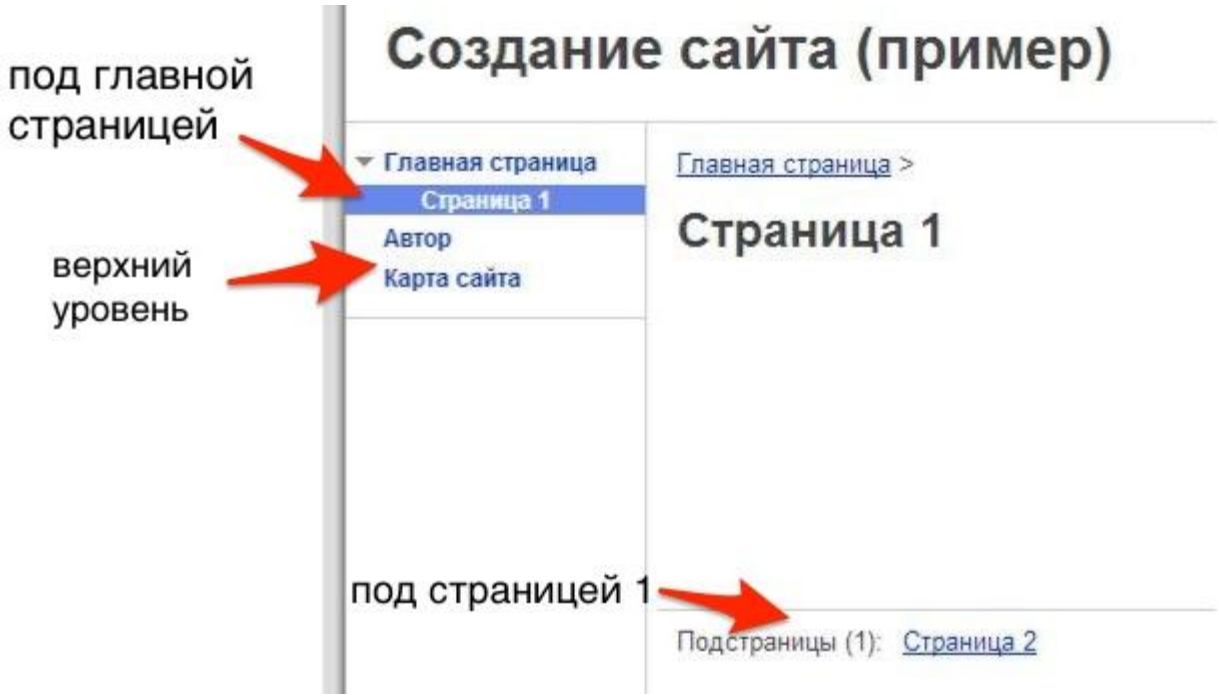
Для добавления новой страницы щелкните по кнопке «Создать страницу» и следуйте указаниям мастера.



1. В поле «Название страницы» введите название страницы. Выбранное название скорее всего попадет в меню, поэтому стоит воздержаться от использования длинных и непонятных названий.
2. В качестве шаблоны выберите «Веб-страница».
3. Выберите местоположение страницы на сайте. Для наглядности ниже также представлены несколько иллюстраций:
    - «Поместить страницу на верхний уровень». Страница будет размещена на том же уровне, что и «Главная страница». В примере ниже это страница «Автор».

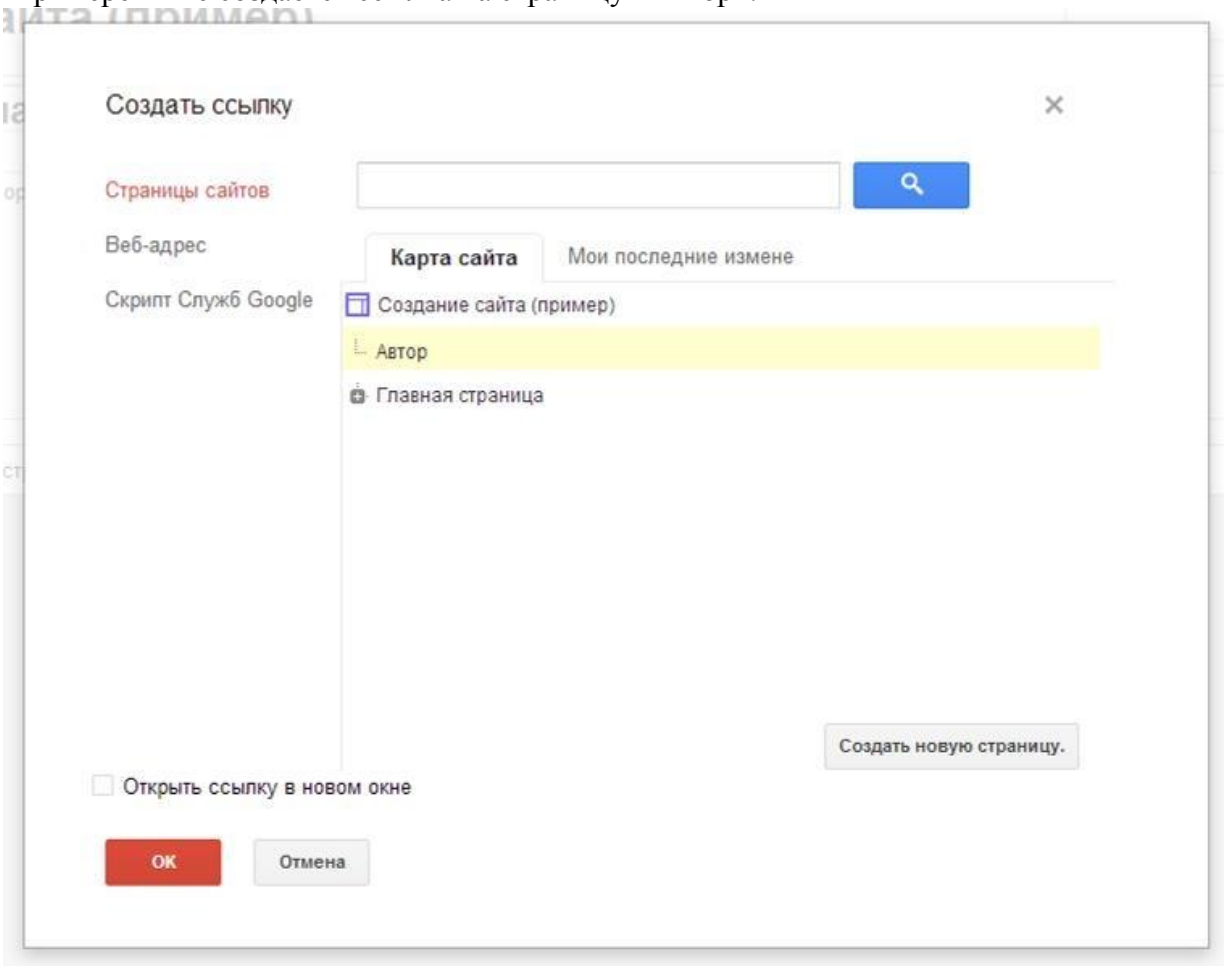
- «Разместить под страницей «Главная страница»». Страница будет размещена вторым уровнем, под главной страницей. «Страница 1» в примере ниже.
- «Выбрать другое расположение». Выбор данного пункта приведёт к открытию полной карты сайта с возможностью выбора любого местоположения в иерархической структуре сайта.

4. После нажатия кноки «Создать» автоматически откроется редактор только что созданной страницы.

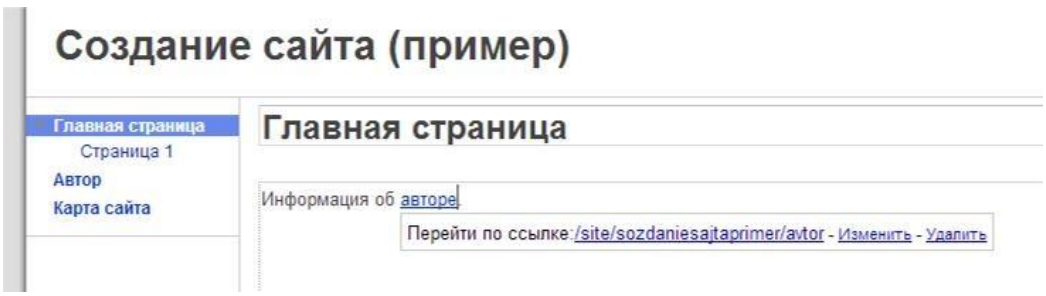


Вы также можете создать нелинейную навигацию на сайте, добавляя непосредственно в текст документа ссылки на другие страницы сайта или внешние ресурсы. Инструмент «Ссылка» работает точно также, как и в блоге, дополнительно позволяя выбрать внутренние страницы сайта из карты:

В примере ниже создается ссылка на страницу «Автор»:



Для посетителя ссылка будет работать в привычном режиме, но в редакторе она будет выглядеть следующим образом:



Расширенная версия руководства

Для ознакомления с инструментами «тонкой» настройки сайта Google перейдите к

расширенной версии инструкции :

В инструкции вы сможете узнать о том, как:

- изменить местоположение страницы: например, если страница создана, но неправильно был выбран уровень ее размещения;
- изменить название сайта;
- изменить форматирование (фон) сайта;
- изменить форматирование страниц сайта: цвет и размер заголовков страниц (в стандартном режиме редактирования страницы - форматировать заголовок нельзя); автоматическое форматирование новых страниц сайта (например, какой размер/цвет текста будет на только что созданной странице) и так далее;
- изменить параметры доступа к сайту: кто сможет видеть/читать (или редактировать) созданный сайт;
- изменить настройки страницы: например, показ ссылок на подстраницы (в нижней части окна); показ приложений (файлов, прикрепленных к странице); настройка возможности оставлять комментарии;
- изменить целевую страницу сайта: страницы, которая открывается первой (эта страница является главной страницей сайта).

Также в расширенной версии подробно рассмотрены базовые шаги создания сайта, описанные в краткой версии инструкции.

Требования к содержанию сайта

1. На главной странице сайта разместить ссылки на страницы с презентацией и видеороликом и краткую информацию об авторе и назначении сайта.
2. На сайте обязательно должны быть встроены **рисунки**, **видео** и **презентация**.
3. Встроить в одну из страниц форму. На главную страницу сайта добавить ссылку, ведущую к странице с формой.
4. На странице "Об авторе" разместить логотип.