

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
КАФЕДРА «АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ»

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ  
К ЛАБОРАТОРНЫМ РАБОТАМ (1 часть)**

по дисциплинам

**«WEB-ТЕХНОЛОГИИ»,**

**«WEB-ТЕХНОЛОГИИ И WEB-ПРОГРАММИРОВАНИЕ»**

для студентов направлений подготовки:

09.03.01 «Информатика и вычислительная техника»,

09.03.02 «Информационные системы и технологии»

всех форм обучения

Рассмотрено на заседании кафедры  
«Автоматизированные системы управления»  
Протокол № 6 от 17 января 2017 года

Утверждено на заседании  
учебно – издательского совета ДОННТУ  
Протокол № \_\_\_\_ от \_\_\_\_\_ 2017г.

Донецк  
2017

Методические указания к лабораторным работам (1 часть) по дисциплинам "Web-технологии", "Web-технологии и Web-программирование" для студентов направлений подготовки: 09.03.01 «Информатика и вычислительная техника», 09.03.02 «Информационные системы и технологии» всех форм обучения / сост.: Н. К. Андриевская, И. В. Матях – Донецк: ДОННТУ, 2017. – 107 с.

Методические указания содержат краткие теоретические сведения, методические рекомендации и задания для выполнения лабораторных работ по дисциплинам "Web-технологии", "Web-технологии и Web-программирование". Изложена методика выполнения каждой из лабораторных работ, требования к содержанию отчетов, список рекомендуемой литературы.

Методические указания к лабораторным работам предназначены для студентов направлений подготовки: 09.03.01 «Информатика и вычислительная техника», 09.03.02 «Информационные системы и технологии» всех форм обучения.

Утверждено методическими комиссиями обоих направлений.

Составители:

Н.К. Андриевская, ст. пр.  
И.В. Матях, асс.

Рецензенты:

В.В. Червинский, доцент каф. АТ  
М.В. Привалов, доцент каф. АСУ

Ответственный за выпуск:

М.В. Привалов, зав. каф. АСУ

## СОДЕРЖАНИЕ

Лабораторная работа №1 Представление текстовых документов в формате html.....	4
Лабораторная работа №2 Гиперссылки и мультимедиа.....	23
Лабораторная работа №3 Верстка таблиц в html .....	30
Лабораторная работа №4 Каскадные таблицы стилей css.....	42
Лабораторная работа №5 Создание прототипа интерфейса .....	54
Лабораторная работа №6 Форматирование сложных таблиц с данными .....	57
Лабораторная работа № 7 Верстка макета сайта.....	80
Лабораторная работа №8 Формы в html-документах .....	85
Лабораторная работа №9 Javascript .....	96
Лабораторная работа №10 Javascript2 .....	101
Лабораторная работа №11 Проверка форм с помощью javascript	110
Лабораторная работа №12 Тема: Использование языка SVG	107

## Лабораторная работа №1

Представление текстовых документов в формате HTML

**Тема лабораторной работы:** создание Web-страниц с использованием HTML.

**Цель работы:** изучение основ языка гипертекстовой разметки HTML.

### Структура Web-документа

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, должен начинаться со строки объявления версии HTML **<!DOCTYPE...>**, которая выглядит примерно так:

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

В данном случае мы говорим браузеру, что HTML соответствует международной спецификации версии 4.01. и он будет одинаково смотреться во всех браузерах (обычно это три основных Internet Explorer, Opera, Firefox).

Для спецификации версии HTML5 строка объявления имеет упрощенный синтаксис:

```
<!DOCTYPE html>
```

Далее обозначается начало и конец документа тегам **<html>** и **</html>** соответственно. Внутри этих тегов должны находиться теги головы (**<head></head>**) и тела документа (**<body></body>**).

Вслед за тегом DOCTYPE можно указать и язык страницы.

```
<html lang=ru>
```

Идентификатор языка указывается в привычном по старым стандартам теге **html**.

Обычно основой головы документа является элемент **TITLE** - заголовок документа. Также там содержится вся техническая информация, различные таблицы стилей и т.п.

Что касается конкретной кодировки страницы, то и этот элемент в HTML5 также, как и **DOCTYPE**, выглядит проще, чем это было в старых стандартах HTML. Теперь кодировка задаётся следующим образом:

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

Основное содержимое: текст, таблицы, картинки, - находится в теле документа.

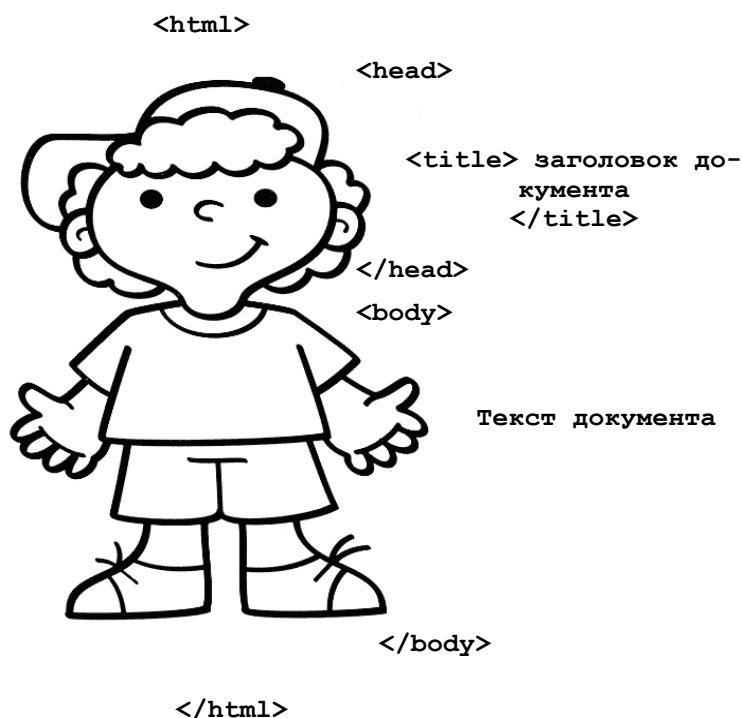
**Вставка комментария:** **<!-- -->** - закомментированный текст, при просмотре документа через браузер видно не будет.

В начале комментариев нужно открыть тегом **<!--** затем вписать текст, и закрыть тегом **-->**.

Тег **<!-- -->** внутри элемента **TITLE** не действует.

На этой картинке схематически изображена структура:

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



Как вы видите, голова находится над телом, поэтому никогда не размещайте голову документа в теле документа (или наоборот). Сначала закрываем голову документа **</HEAD>**, и лишь затем открываем тело **<BODY>**. И еще, у документа одна голова и одно тело, и не стоит пытаться создавать их большее количество.

**HTML5** позволяет пользователю проявлять некоторые вольности в написании кода. К примеру, закрывающий тег может быть указан как без слеша, так и со слешем. Кроме того, в общей структуре документа запросто могут отсутствовать целые разделы. Так, в документе можно убрать теги **HEAD** и **BODY**, и при этом оставить их внутреннее содержимое, **но не стоит нарушать стандарты...**

### Создание простейшей страницы

**ШАГ 1.** Создайте на своем компьютере папку с названием вашего будущего сайта, к примеру, в данном случае актуально будет назвать папку **obuchenie\_html**. Внутри папки создайте еще одну папку с названием **www**, а внутри нее еще одну для изображений с названием **img** (позже вставим туда картинки):



**ШАГ 2.** Зайдите в созданную на предыдущем шаге папку **www** и создайте внутри нее текстовый документ. Для этого зайдите в нее, кликните

правой кнопкой мыши в выпадающем меню выберите **Создать - Текстовый документ**.

**ШАГ 3.** Напишите в Блокноте текст простейшей странички.

**Пример:**

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title> Это моя первая страничка! </title>
  </head>
  <body>
    <!--сейчас я напишу свою первую строчку текста!-->
    Ура!!!! Я создал свою первую страницу на html!!!
  </body>
</html>
```

**ШАГ 4.** Сохраните ваш документ как "index.htm" (".html" даст тот же результат). После сохранения можете удалить текстовый документ.

**ШАГ 5.** Запустите получившийся файл index.htm.

Для того, чтобы внести изменения в страничку, необходимо открыть данный файл через обычный блокнот. Либо же, можно менять прямо в браузере, если браузер поддерживает такие возможности.

## Форматирование заголовков и абзацев

По умолчанию текст идет слитно, без разделений на абзацы, заголовки, т.е. форматирование отсутствует!

Основными элементами форматирования текста в html являются тэги, описанные в таблице 1.1.

Таблица 1.1 – Основные элементы форматирования текста

<b>P</b>	Используется для разбиения текста на параграфы
<b>H1,H2,...H6</b>	Применяются для создания заголовков 1,2...6 уровней
<b>BR</b>	Используется для переноса строки
<b>DIV , SPAN</b>	Используются для выделения части документа определенным способом.

Рассмотрим каждый тэг подробнее.

**P** - используется для разметки параграфов в html документах.

**Атрибуты:**

**ALIGN** - определяет способ горизонтального выравнивания параграфа. Возможные значения: **left**, **center**, **right**, **justify**. Это соответственно по левому краю, по центру, по правому краю, и по ширине. По умолчанию имеет значение **left**.

**Пример:**

```
<p align="center"> Этот параграф по центру </p>
```

```
<p align="left"> Этот по левому краю </p>
<p> Этот тоже по левому (т.к. по умолчанию) </p>
<p align="right"> Этот по правому краю </p>
<p align="justify"> В этом параграфе текст будет
выравниваться по ширине (одновременно по левому и пра-
вому краям документа). Не понимающие justify браузеры
будут выравнивать текст по левому краю</p>
```

**Примечание:** содержимое в кавычках должно быть написано без пробелов, т.е. `<p align="right">` а не `<p align=" right ">` иначе не будет работать!

**H1,H2,...H6** - Применяются для разбиения текста на смысловые уровни - разделы и подразделы.

Существует шесть уровней заголовков, различающихся по величине шрифта.

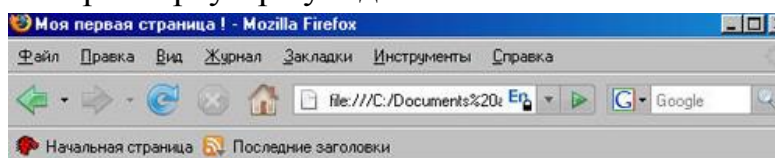
**Атрибуты:**

**ALIGN** - определяет способ горизонтального выравнивания заголовков. Возможные значения: **left, center, right**. По умолчанию - **left**.

**Пример:**

```
<!-- примеры заголовков от 1 до 6 уровня-->
<h1> Большой заголовок </h1>
<h2> Заголовок поменьше </h2>
<h3> Еще меньше </h3>
<h4> Совсем маленький </h4>
<h5> Малюсенький заголовочек </h5>
<h6> Ну просто лилипутский заголовочек </h6>
```

То при просмотре в браузере увидим:



**Большой заголовок**

**Заголовок поменьше**

**Еще меньше**

**Совсем маленький**

**Малюсенький заголовочек**

**Ну просто лилипутский заголовочек**

**BR** - данный элемент осуществляет перенос строки.

**Пример:**

Если внутри тела документа написать следующее:

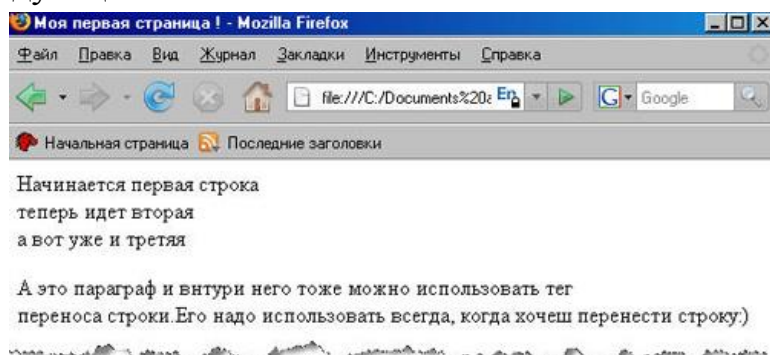
Начинается первая строка **<br>**

теперь идет вторая **<br>**

а вот уже и третья

**<p>** А это параграф и внутри него тоже можно использовать тег **<br>** переноса строки. Его надо использовать всегда, когда надо перенести строку:) **</p>**

Увидим следующее:



Можно заметить, что при переносе строки не начинается новый параграф! Перенос - он и есть перенос. Не имеет закрывающего тега.

**DIV** - используется как удобный контейнер для блоков html кода страницы, которым легко динамически манипулировать – перемещать, регулировать отступы, скрывать и т.п. Обязателен закрывающий тег!

**Атрибуты:**

**ALIGN** - определяет способ горизонтального выравнивания контейнера.

Возможные значения: **left**, **center**, **right**, **justify**. Это соответственно по левому краю, по центру, по правому краю, и по ширине. По умолчанию имеет значение **left**.

Допустим нам нужно выровнять первые две строчки текста из предыдущего примера по правому краю, не выделяя при этом их в абзац.

**<div align="right">**

Начинается первая строка **<br>** теперь идет вторая **<br>**

**</div>**

а вот уже и третья

**<p>**

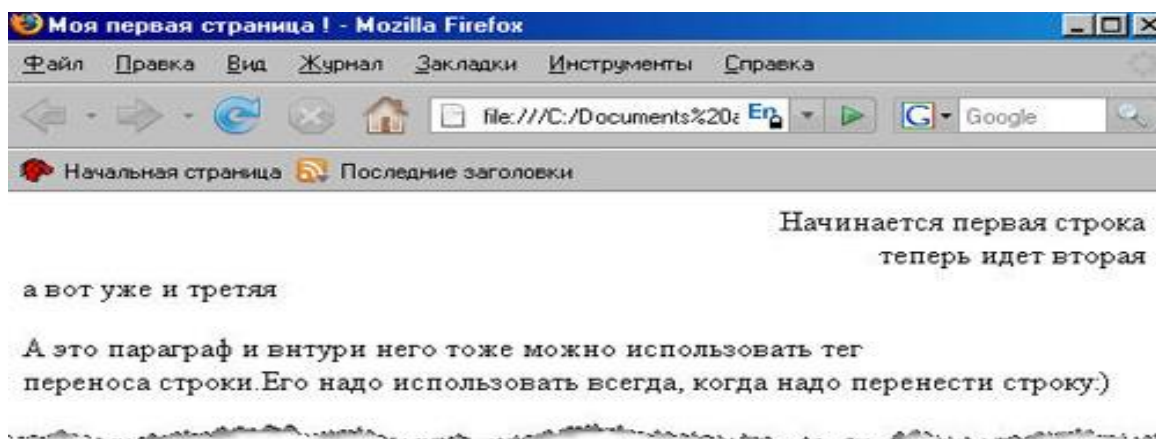
А это параграф и внутри него тоже можно использовать тег **<br>** переноса строки. Его надо использовать всегда, когда надо перенести строку:) **</p>**

Т.е. мы взяли две строчки кода, заключили их в контейнер **DIV** и выровняли его по правому краю.

**Примечание:** Находящиеся между начальным и конечным тегами текст или HTML-элементы выделяются как бы в отдельный параграф (но отступы гораздо меньше).

Текст содержащийся в элементе **DIV** выделяется в отдельную строку!





Чтобы придать тексту ту или иную гарнитуру в html используются элементы, описанные в таблице 1.2.

Таблица 1.2 – Элементы для форматирования символов

<b>STRONG</b>	Используется для выделения текста жирным
<b>EM</b>	Используется для выделения текста курсивом
<b>U</b>	Выделение текста подчеркиванием
<b>S</b>	Перечеркивание текста
<b>SUP</b>	Создание верхнего индекса
<b>SUB</b>	Создание нижнего индекса
<b>FONT</b>	Изменение цвета, типа, размера шрифта
<b>HR</b>	Вставляет в текст горизонтальную разделительную линию

Существуют теги-близнецы для выделения текста жирным шрифтом.

`<b></b>` `<strong></strong>`

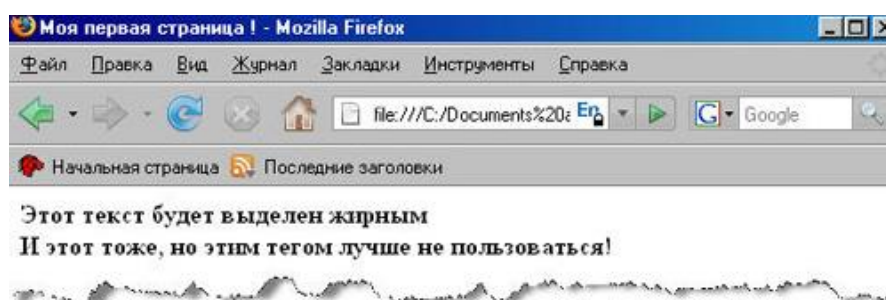
**STRONG** - Выделяет текст, заключенный между открывающим и закрывающим тегами, жирным шрифтом. Раньше везде использовался тег `<BOLD>` (или `<b>`), в принципе его можно использовать и сейчас, но это не приветствуется поисковыми системами.

Если в теле документа написать:

`<strong>` Этот текст будет выделен жирным `</strong>`  
`<br>`

`<b>` И этот тоже, но этим тегом лучше не пользоваться!  
`</b>`

В браузере увидим:



В HTML5 по-прежнему визуальное различие отсутствует, однако вышеупомянутые теги имеют совершенно различное семантическое значение. Фрагмент текста, выделенный тегом **strong**, принимает более важное значение, чем окружающий текст. Таким образом, это не просто выделенный жирным шрифтом текст, а важная смысловая единица страницы. Иными словами, тег **strong**, заключая внутрь себя часть текста, придаёт ему более высокий вес, нежели окружающий контент.

К слову сказать, поисковые системы также уделяют семантике немалое значение и умеют оценивать вес текстовых фрагментов, так что пренебрегать семантическим значением блоков не стоит. Тег **b** сохранил свою оформительскую функцию, и по-прежнему выделяет текст жирным шрифтом, однако сохраняет вес такого фрагмента по отношению к окружающему тексту. Например, вы можете использовать тег **b** для выделения терминов.

Аналогично в html существует и другая пара сходных между собой по смыслу тегов:

`<i></i> <em></em>`

**ЕМ ( Emphasis )** - Выделяет текст, заключенный между открывающим и закрывающим тегами, курсивом. Также аналогичен по действию тег `<I>`, однако его лучше не использовать т.к. это не приветствуется поисковыми системами.

`<em>`Никогда`</em>` не говори `<i>`никогда`</i>` !

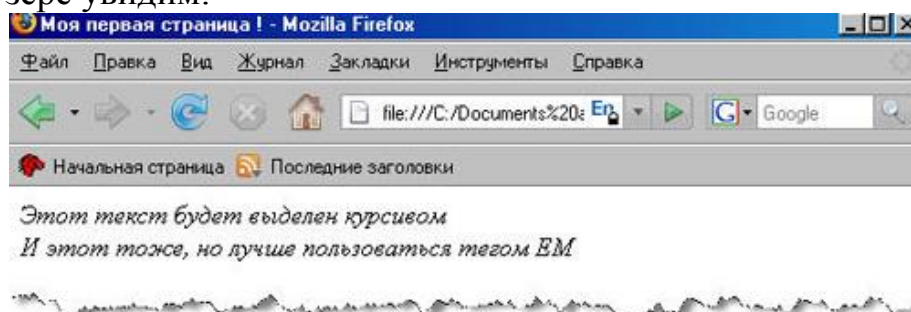
Первое слово обрaмлено в **em**, что однозначно акцентирует на нём ударение, второе слово «никогда» лишь выделяется визуально на странице, как слово, о котором и идёт речь в предложении. При этом оба варианта на странице отображаются в виде курсивного выделения.

*Никогда не говори никогда !*

### Пример:

`<em>` Этот текст будет выделен курсивом `</em>` `<br>`  
`<i>` И этот тоже, но лучше пользоваться тегом `ЕМ</i>`

В браузере увидим:

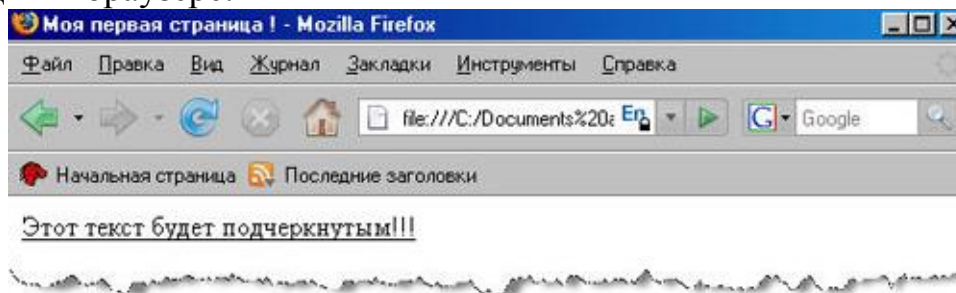


**U(Underline)** - Отображает помещенный между открывающим и закрывающим тегами текст как подчеркнутый:

Если в теле документа написать:

**<u>** Этот текст будет подчеркнутым!!! **</u>**

Увидим в браузере:



Наконец обратимся к последней паре:

**<s></s>** **<strike></strike>**

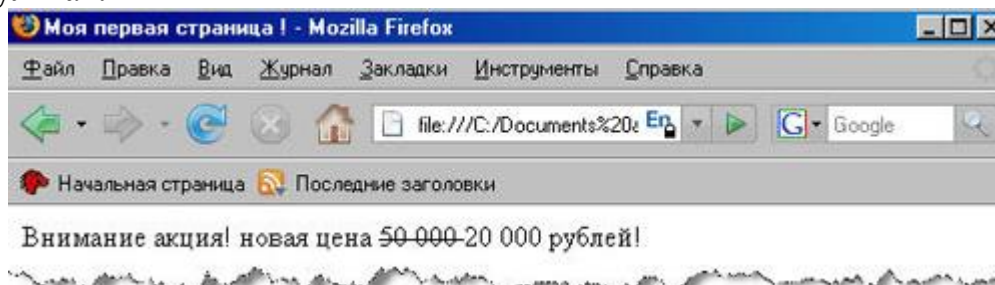
Визуально любая из этих пар тегов заставляет браузер выводить текст в зачёркнутом виде, но как вы уже догадались, в HTML5 каждой из пар соответствует своя семантика. Оба элемента отображаются браузерами в виде зачёркнутого текста, но тег **s** семантически означает фрагмент, который утратил своё значение или же стал в корне не верным.

**S(Strike)** - Текст, помещенный между открывающим **<s>** и закрывающим **</s>** тегами, будет перечеркиваться.

**Например:**

Внимание акция! новая цена **<s>** 50 000 **</s>** 20 000 рублей!

Результат:



Раз уж заговорили об элементе HTML5 **del**, то нельзя не упомянуть о парном ему теге **ins**. Элемент **ins** является для **del** комплементарным и служит для указания фрагмента текста, который был вставлен в документ позже. Пример:

Для зачёркнутого текста следует использовать

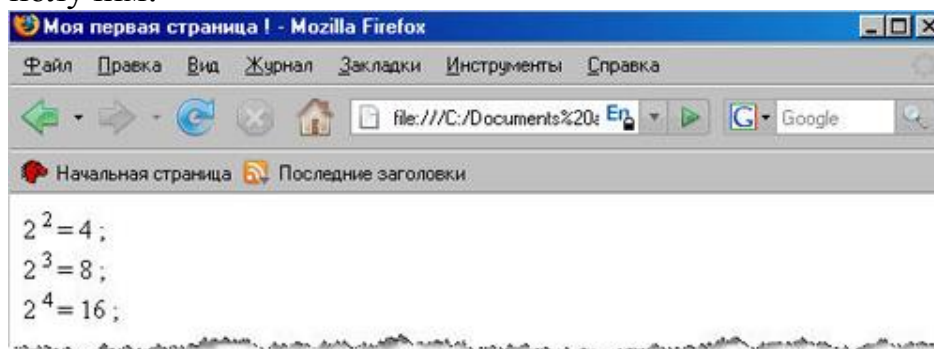
**<del>** тег **s** **</del>** **<ins>** тег **del** **</ins>**,

**SUP ( Superscript)** - Отображает текст, заключенный между открывающим **<SUP>** и закрывающим **</SUP>** тегами, как верхний индекс от основного текста.

Например:

2 **<sup>** 2 **</sup>** = 4 ;**<br>**  
2 **<sup>** 3 **</sup>** = 8 ; **<br>**  
2 **<sup>** 4 **</sup>** = 16 ;

И что получим:

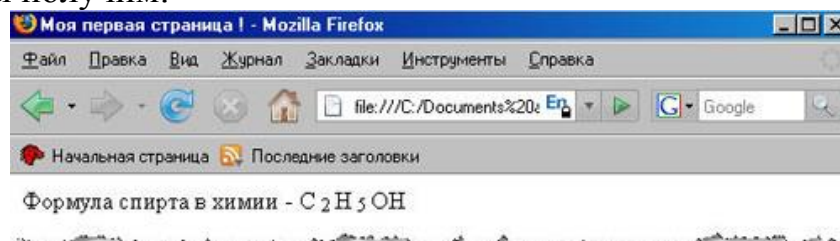


**SUB(Subscript)** - Отображает текст, заключенный между открывающим **<SUB>** и закрывающим **</SUB>** тегами, как нижний индекс от основного текста.

**Пример:**

Формула спирта в химии - С **<sub>2</sub>** Н **<sub>5</sub>** ОН

И что мы получим:



**FONT** - Позволяет изменять цвет, размер и тип шрифта текста, находящегося между открывающим **<font>** и закрывающим **</font>** тегами.

**Атрибуты:**

**SIZE** - Определяет размер шрифта. Возможные значения - 1, 2, 3, 4, 5, 6, 7.

**COLOR** - Определяет цвет текста. Задается либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов.

**FACE** - определяет используемый шрифт. Используйте Times New Roman, Arial, Tahoma, Courier, Courier New. Они установлены почти у всех. Иначе гарантий нет!

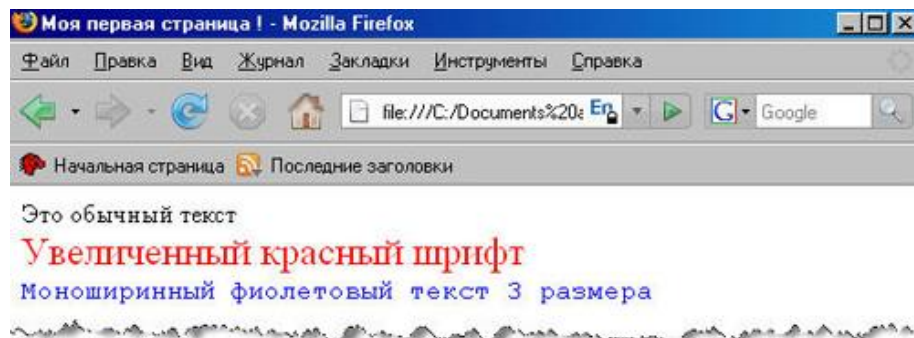
Если в теле документа написать:

Это обычный текст **<br>**

**<FONT SIZE="+2" COLOR="red">** Увеличенный красный шрифт**</FONT><br>**

**<FONT SIZE="3" FACE="Courier New" COLOR="Violet">** Моноширинный фиолетовый текст 3 размера**</FONT>**

При просмотре в браузере увидим:



Если написано **SIZE="+2"** то это означает увеличить на 2 единицы шрифт по сравнению со стандартным. Обычно стандартный размер шрифта равен 3.

**Примечание:** В дальнейшем (после изучения CSS) лучше вообще все что связано с типом, размером и цветом шрифта делать через таблицы стилей. Это гораздо удобнее, тем более многие элементы форматирования текста расположенные внутри элемента FONT, работают некорректно.

**HR**- служит для вставки в текст горизонтальной линии. Закрывающего тега нет.

**Атрибуты:**

**WIDTH** – определяет длину линии в пикселах или процентах от ширины окна браузера.

**SIZE** – толщина линии в пикселах.

**ALIGN** – определяет выравнивание горизонтальной линии. Атрибут может принимать следующие значения:

**left** – выравнивание по левому краю документа.

**right** – выравнивание по правому краю документа.

**center** – выравнивание по центру документа (используется по умолчанию).

**NOSHADE** – определяет способ закрашки линии как сплошной. Атрибут является флагом и не требует указания значения. Без данного атрибута линия отображается объемной.

**COLOR** – задает цвет линии. Можно использовать либо RGB-значение в шестнадцатеричной системе, либо один из 16-ти базовых цветов. Атрибут работает только в Internet Explorer.

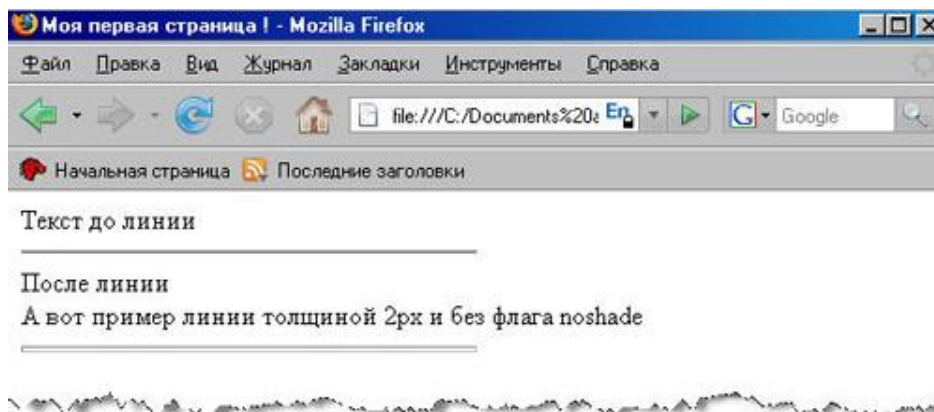
**Например:**

Текст до линии `<hr noshade width="50%" align="left">`

После линии `<br>` А вот пример линии толщиной 2px и без флага noshade `<hr width="50%" align="left" size="2">`

Что мы увидим:










## Создание списков

Чаще всего используют списки двух видов: нумерованные и нenumерованные. Единственным отличием является то, что перед пунктами неупорядоченных списков обычно ставятся символы-буллеты (bullets), например, точки, ромбики и т.п., в то время как пунктам упорядоченных списков предшествуют их номера.

Таблица 1.3 – Элементы для создания списков различного вида

<b>UL</b>	Создает нenumерованный список
<b>OL</b>	Создает нумерованный список
<b>LI</b>	Создает пункты списка внутри элементов <b>OL</b> или <b>UL</b>
<b>DL</b> 	Определяет список описаний
<b>DT</b> 	Определяет имя описания
<b>DD</b> 	Определяет значение описания
<b>MENU</b> 	Для контекстных меню, панелей инструментов
<b>MENITEM</b> 	Определяет элемент команды / меню

**UL (Unsorted List)** - Создает нenumерованный список. Обязательно наличие закрывающего тега, причем между начальным и конечным тегами должны присутствовать один или несколько элементов **LI**, обозначающих отдельные пункты списка.

**Например:**

**<ul>**

**<li>** Первый пункт списка **</li>**

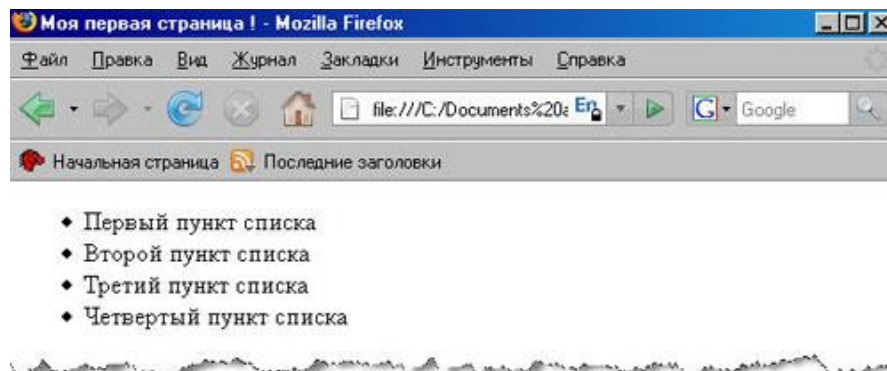
**<li>** Второй пункт списка **</li>**

**<li>** Третий пункт списка **</li>**

**<li>** Четвертый пункт списка **</li>**

**</ul>**

Что мы увидим в браузере:



По умолчанию элементы списка маркируются черным кружочком.

При помощи атрибута **TYPE** можно изменить стиль маркирования. В пределах одного списка можно использовать различную маркировку элементов списка.

HTML-код:	Отображение в браузере:
<pre>&lt;ul type="circle"&gt; &lt;li&gt;элемент 1&lt;/li&gt; &lt;li&gt;элемент 2&lt;/li&gt; &lt;li type="disc"&gt;элемент 3 &lt;/li&gt; &lt;li type="square"&gt;элемент 4 &lt;/li&gt; &lt;/ul&gt;</pre>	<ul style="list-style-type: none"> <li>○ элемент 1</li> <li>○ элемент 2</li> <li>• элемент 3</li> <li>▪ элемент 4</li> </ul>

**OL (Ordered List)** - создает нумерованный список. Между начальным и конечным тегами должны присутствовать один или несколько элементов **LI**, обозначающих отдельные пункты списка.

**Атрибуты:**

**START** – определяет первое число, с которого начинается нумерация пунктов. (только целые числа). Если не указывать, начинается с начала.

**TYPE** – определяет стиль нумерации пунктов списка.

**Возможные значения:**

"A" – заглавные буквы A, B, C ...

"a" – строчные буквы a, b, c ...

"I" – большие римские числа I, II, III ...

"i" – маленькие римские числа i, ii, iii ...

"1" – арабские числа 1, 2, 3 ...

Значение по умолчанию **<OL TYPE="1">**.

**LI (List Item)** - Создает пункт в списке. Располагается внутри элементов **OL** или **UL**. Закрывающий тег писать желательно, но не обязательно.

**Атрибуты:**

**VALUE** – изменяет порядок нумерации элементов списка. Используется только, если элемент **LI** находится внутри элемента **OL**. В качестве значения указывается порядковый номер элемента.

### Пример:

Чтобы создать сайт на домашнем компьютере необходимо:

```
<ol ><!--так как тип не указали будет использо-  
ваться по умолчанию-->
```

```
<li> Выучить html </li>
```

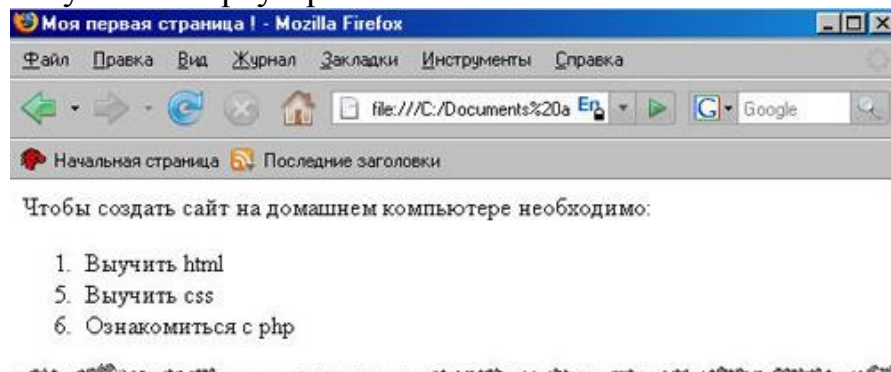
```
<!--сейчас нумерация пойдет с пятого номера-->
```

```
<li value="5"> Выучить css </li>
```

```
<li> Ознакомиться с php </li>
```

```
</ol>
```

Результат в браузере:



Списки описаний используются для формирования пар типа «имя/значение» / «вопрос/ответ» и т.п.

Список вида `<menu>` не нов для HTML, но в пятой версии языка изменилось его лексическое назначение. Теперь он представляет собой список команд, который в зависимости от установленного значения атрибута `type` может выполнять три различных функции — контекстное меню, панель инструментов (toolbar) и произвольное меню с командами:

list	— произвольный набор команд (значение по умолчанию).
toolbar	— панель инструментов.
context	— контекстное меню.

```
<menu type="list">  
  <li><input type="radio" name="num" value="1" /></li>  
  <li><input type="radio" name="num" value="2" checked="checked" /></li>  
  <li><hr /></li>  
  <li><input type="checkbox" name="on" value="true" /></li>  
</menu>
```

HTML5 позволяет создавать контекстные меню для любых видимых элементов. Для этого необходимо вставить в документ конструкцию `<menu>` со списком необходимых пунктов `<li>` и присвоить ей уникальный идентификатор. Если этот `id` указать в значении атрибута `contextmenu` необходимого нам элемента, то клик по нему правой кнопкой мыши вызовет соответствующее контекстное меню (при условии, что в браузере поддерживается и активирована эта функция).



```
<p contextmenu="myMenu">Клики правой кнопкой, чтобы открыть контекстное меню.</p>
<menu type="context" id="myMenu">
  <li><a href="link1.html">Пункт 1</a></li>
  <li><a href="link2.html">Пункт 2</a></li>
</menu>
```

## Изменение цветового оформления страницы

Можно задать цвет текста для всего документа. Также, можно задать и фоновое изображение. Все они прописываются для элемента **BODY**. Значения цветов задаются либо RGB-значением в шестнадцатеричной системе, либо одним из 16-ти базовых цветов.

### Атрибуты:

**BACKGROUND** – определяет изображение для "заливки" фона. Значение задается в виде полного URL или имени файла с картинкой в формате GIF или JPG.

**BGCOLOR** – определяет цвет фона документа.

**TEXT** – определяет цвет текста в документе.

### Пример 1:

```
<!-- задаем фоновый цвет и цвет текста -->
```

```
<body bgcolor="#FFF8D2" text="red">
```

```
<p> Этот текст будет красный, потому что мы изменили цвет текста в теге БОДИ и теперь весь текст на странице по умолчанию будет красный </p>
```

```
<font color ="green">
```

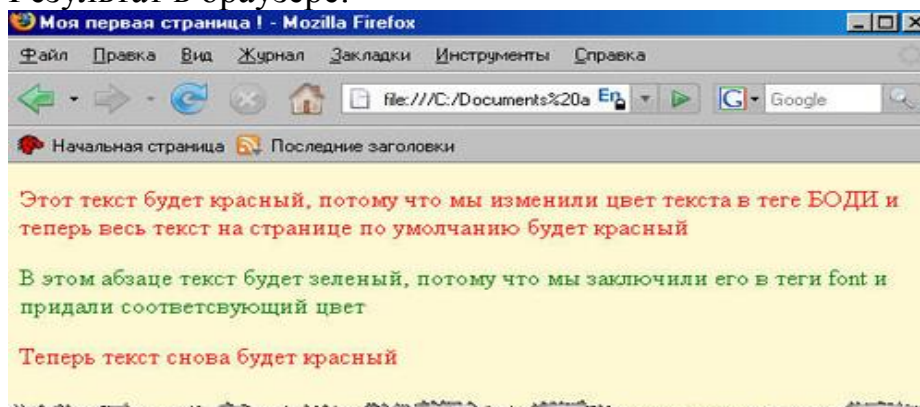
```
<p> В этом абзаце текст будет зеленый, потому что мы заключили его в теги font и придали соответствующий цвет </p>
```

```
</font>
```

```
<p> Теперь текст снова будет красный </p>
```

```
</body>
```

Результат в браузере:



## Пример 2:

`<!-- задаем фоновое изображение и цвет текста -->`

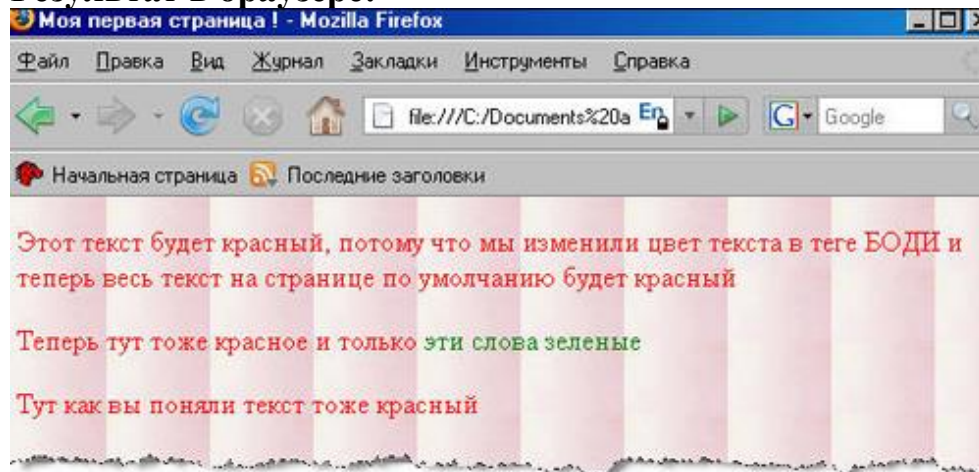
`<body background="fon.jpg" text="red">`

`<p>` Этот текст будет красный, потому что мы изменили цвет текста в теге БОДИ и теперь весь текст на странице по умолчанию будет красный `</p>`

`<p>`Теперь тут тоже красное и только `<font color="green">` эти слова зеленые `</font></p>`

`<p>` Тут как вы поняли текст тоже красный`</p>`  
`</body>`

## Результат в браузере:



## Новые атрибуты HTML5

Одним из самых неоднозначных является атрибут **hidden**. Он позволяет разместить на странице элемент, но при этом не заставляет браузер не отображать его. Иными словами, элемент на странице становится скрытым, что и объясняет происхождение наименования тега.

`<p hidden>`Этот текст браузер вообще не отобразит,  
хотя в коде страницы он будет присутствовать.`</p>`

Такой фрагмент не будет визуально отображён на странице. Сразу стоит сделать замечание, что поскольку в коде страницы элемент с атрибутом **hidden** всё равно присутствует, то не следует использовать его для размещения конфиденциальных данных. Стоит только нажать в браузере комбинацию **Ctrl+U**, и все ваши секреты тут же станут общеизвестными.

## Отмененные атрибуты

- **profile** для **head**
- **version** для **map**, **img**, **object**, **form**, **iframe**, **a**
- **summary** для **table**
- **headers**, **axis** и **abbr** для **td** и **th**
- **scope** для **td**

Кроме того, HTML 5 не имеет следующих атрибутов, поскольку они лучше обрабатываются CSS:

- **align** для **caption**, **iframe**, **img**, **input**, **object**, **legend**, **table**, **hr**, **div**, **h1-h6**, **p**, **col**, **colgroup**, **tbody**, **td**, **tfoot**, **th**, **thead**, **tr** и **body**
- **background** для **body**
- **bgcolor** для **table**, **tr**, **td**, **th** и **body**
- **cellpadding** и **cellspacing** для **table**
- **compact** для **menu**, **ol** и **ul**
- **height** для **iframe**, **td** и **th**
- **noshade** для **hr**
- **nowrap** для **td** и **th**
- **type** для **li**, **ol** и **ul**
- **Width** для **hr**, **table**, **td**, **th**, **col**, **colgroup**, **iframe** и др.

### Задание на лабораторную работу:

Создать Web-страницу на заданную тему (таблица 1.3). Схема страницы (таблица 1.4) и стиль оформления текста (таблица 1.5) выбираются по варианту. Параграф должен содержать не менее **трех** строк текста. Каждый уровень списка – не менее **трех** пунктов. Созданный файл должен быть назван по шаблону: **фамилия\_index.html**.

### Содержание отчета:

1. Задание.
2. Таблица с перечнем использованных тегов.
3. Текст HTML-документа.
4. HTML-страница.
5. Выводы по работе.

### Контрольные вопросы:

1. Что такое HTML? Что такое гипертекстовый документ?
2. Что такое тег? Структура тега HTML. Формат записи тега HTML.
3. Привести структуру HTML документа. Описать назначение тегов **<HEAD>**, **<BODY>**.
4. Что такое параметр тега? Формат записи параметра тега HTML.
5. Перечислить параметры тега **<BODY>**.
6. Перечислить теги для представления текстовой информации и дать их описание.
7. Как сделать перевод на новую строку?
8. Перечислить виды списков, существующих в HTML. Привести теги, представляющие списки в HTML.
9. Что такое вложенные списки в HTML? Привести пример вложенного списка HTML.
10. Как изменить цвет и размер символов части текста?
11. Перечислите теги позволяющие управлять начертанием шрифта?

Таблица 1.3 – Тема индивидуального задания

№	Темы	№	Темы	№	Темы
1	Цветы	23	Самолеты	45	Школа
2	Детские игрушки	24	Космос	46	Языки программирования
3	Актеры	25	Ремонт	47	Университет
4	Писатели	26	Кино	48	Хоккей
5	Программы	27	Футбол	49	Музыка
6	Операционные системы	28	Моря	50	Ноутбуки
7	Мебель	29	Стихи	51	Одежда
8	Недвижимость	30	Наука	52	Спортивный инвентарь
9	Бытовая техника	31	Погода	53	Журналы
10	Книги	32	Олимпиады	54	Столицы
11	Компьютерные игры	33	Горы	55	Оружие
12	Автомобили	34	Фокусники	56	Холодное оружие
13	Города	35	Компьютерные технологии	57	Бронетехника
14	Страны	36	Мода	58	Фреймворки
15	Фрукты	37	Сайт городского управления	59	Озера
16	Монеты	38	Гороскоп	60	Телефоны
17	Животные	39	Философия	61	Зимние виды спорта
18	История	40	Тесты	62	Легкая атлетика
19	Музыкальные группы	41	Аптека	63	Аквапарки
20	Спорт	42	Театр	64	Отдых
21	Путешествия	43	Поэты	65	Туризм
22	Картины	44	География	66	Корабли

Таблица 1.4 - Схема страницы

№	Схема
1	Заголовок, список нумерованный, 2 параграфа, заголовок, горизонтальная линия1, параграф, заголовок, неформатированный текст, горизонтальная линия2, список нумерованный
2	Заголовок, горизонтальная линия2, список нумерованный, 2 параграфа, заголовок, параграф, неформатированный текст, горизонтальная линия1, заголовок, список нумерованный
3	Заголовок, горизонтальная линия1, список нумерованный, параграф, заголовок, 2 параграфа, заголовок, неформатированный текст, список нумерованный, горизонтальная линия2
4	Список нумерованный, заголовок, неформатированный текст, список нумерованный, заголовок, параграф, заголовок, горизонтальная линия1, 2 параграфа, горизонтальная линия2
5	Список нумерованный, заголовок, неформатированный текст, 2 параграфа, горизонтальная линия1, заголовок, параграф, заголовок, горизонтальная линия2, список нумерованный
6	Горизонтальная линия2, заголовок, список нумерованный, 2 параграфа, заголовок, 2 параграфа, заголовок, неформатированный текст, горизонтальная линия1, список нумерованный
7	Горизонтальная линия, заголовок, 2 параграфа, заголовок, параграф, заголовок, неформатированный текст, список нумерованный, горизонтальная линия, список нумерованный
8	Заголовок, горизонтальная линия1, список нумерованный, 2 параграфа, заголовок, 2 параграфа, неформатированный текст, горизонтальная линия2, заголовок, список нумерованный
9	Неформатированный текст, заголовок, горизонтальная линия2, список нумерованный, параграф, заголовок, параграф, горизонтальная линия1, заголовок, список нумерованный, параграф
10	Неформатированный текст, заголовок, 2 параграфа, заголовок, горизонтальная линия1, список нумерованный, параграф, горизонтальная линия2, заголовок, список нумерованный

Таблица 1.5 - Стили оформления текста

Параметр	Вар 1	Вар 2	Вар 3	Вар 4	Вар 5	Вар 6	Вар 7	Вар 8	Вар 9	Вар 10
<b>Нумерованный Список:</b>										
Маркер	1	i	I	a	A	i	a	A	I	1
Номер перв. ел.	5	3	1	2	4	7	2	5	6	-5
Уровень вложен-ти	2	3	3	2	3	2	3	3	3	2
Цвет	Си-ний	Крас-ный	Голу-бой	жел-тый	Чер-ный	Бе-лый	Зеле-ный	Розо-вый	Оран-жевый	Сала-тый
<b>Ненумерованный список</b>										
Маркер	disk	circle	squar e	disk	circle	squar e	disk	circle	square	disk
Уровень вложен-ти	2	2	3	3	2	3	2	2	3	3
Цвет	Са-лат-ный	Си-ний	Крас-ный	Голу-бой	Жел-тый	Чер-ный	Бе-лый	Зеле-ный	Розо-вый	Оран-жев
<b>Заголовки</b>										
Цвет	Крас-ный	Голу-бой	Жел-тый	Чер-ный	Бе-лый	Зеле-ный	Розо-вый	Оран-жевый	Салат-ный	Си-ний
Размер	2	3	4	5	6	5	4	3	2	6
Вырав-нив.	left	cen-ter	right	left	center	right	left	center	right	center
<b>Параграф</b>										
Вырав-нив.	cen-ter	right	left	center	right	left	cen-ter	right	center	left
Размер текста	Big	smal	Medi um	big	small	big	Medi um	Small	Big	Medi um
Шрифт	Arial	Ta- homa	Cou- rier	Times NewR	Arial	Ta- homa	Cou- rier	Times New R	Arial	Ta- homa
	жирн	кур- сив	под- черк	жирн	кур- сив	под- черк	жирн	курсив	под- черк	жирн
<b>Страница</b>										
Фон	Бе- лый	Жел- тый	Чер- ный	Рису- нок	Крас- ный	Голу- лу- бой	Рису- нок	Черный	Темно синий	белый
Цвет тек- ста	Си- ний	Крас- ный	крас- ный	Оран- жев	Зеле- ный	си- ний	зеле- ный	Жел- тый	белый	са- латн
<b>Горизонтальная линия</b>										
Высота	2	5	10	20	10	30	2	20	30	5
Выравнив ширина	ceter	right	left	center	right	left	centr	right	left	Center
Цвет	Крас- ный	Голу- бой	жел- тый	Чер- ный	Бе- лый	Зеле- ный	Розо- вый	Оран- жевый	Са- латн	Си- ний
Объем	+	-	+	-	+	-	+	-	+	-

## Лабораторная работа №2

### Гиперссылки и мультимедиа

**Тема лабораторной работы:** связывание Web-страниц с помощью гиперссылок и размещение мультимедиа на Web-страницах.

**Цель работы:** изучить возможности языка HTML по связыванию страниц через гиперссылки и размещению различного мультимедиа контента на Web-странице.

### Создание HTML ссылки

Создать html - ссылку очень просто. Для этого используется элемент с одним атрибутом:

```
<a href="http://www.google.com.ua">Это ссылка на сайт www.google.com.ua </a>
```

Элемент **a (anchor)** является как бы якорем, т.е. текст, заключенный между открывающим **<a>** и закрывающим **</a>** тегом будет текстом ссылки.

Атрибут **href** - это сокращение от "hypertext reference/гипертекстовая ссылка", определяет место, на которое выполняется переход по данной ссылке - обычно это internet-адрес и/или имя файла.

Появление нового тега **figure** в HTML5 позволяет ещё больше упростить реализацию размещения изображений на странице. Тег **figure** используется не вместо **img**, а совместно с ним. Кроме того, для организации подписей к изображениям применяется тег **figcaption**.

Пример кода

```
<figure>
  
  <figcaption>
    <small>Обязательно изучите, что нового появилось в html5!</small>
  </figcaption>
</figure>
```

Данный фрагмент кода обеспечивает вывод на страницу, как самого изображения, так и подписи под ним. Тег **figure** как бы заключает в себя все детали, относящиеся к изображению. К таким деталям относятся вложенный **img**, определяющий непосредственно изображение, и **figcaption**, формирующий подпись.

### Ссылка между собственными страницами

Если у нас есть две страницы (к примеру, **page1.htm** и **page2.htm**), расположенные в одной папке, то код ссылки с **page1** на **page2** будет выглядеть так:

```
<a href="page2.htm"> Для перехода на page2 щелкни здесь! </a>
```

Если страница page2 находится в подпапке "subfolder", код ссылки выглядит так:

`<a href="subfolder/page2.htm">` Для перехода на page2 щелкни здесь! `</a>`

Так будет выглядеть код html ссылки со страницы page 2 на page1(в обратную сторону):

`<a href=" ../page1.htm">` Для перехода на page1 щелкни здесь! `</a>`

Символ "../" указывает на папку, расположенную на один уровень выше от данной позиции файла, с которого делается ссылка.

### Ссылка внутри страницы

Бывают случаи, когда необходимо сделать ссылку с начала страницы в конец или к примеру с оглавления на главы и т.д.

Для этого необходимо пометить место документа следующей конструкцией:

`<!-- этот способ используется при маркировании заголовков -->`

`<h2 id="razdel1">Раздел 1</h2>`

`<!-- а такую метку можно поставить везде где понадобится-->`

`<h2> <a name="razdel1"></a> Раздел1: </h2>`

Теперь вы можете ссылаться на помеченную область простым указанием ее имени после значка #:

`<a href="#razdel1"> Ссылка на Раздел 1 </a>`

### Изменение цвета ссылок

Чтобы изменять цвет ссылок во всем документе, существуют специальные атрибуты элемента **BODY**:

**LINK** - цвет просто ссылок

**ALINK(Active Link)** - цвет активных ссылок (активная означает в момент нажатия на нее).

**VLINK(Visited Link)** - цвет уже посещенных ссылок.

Все цвета задаются либо RGB-значением в шестнадцатеричной системе, либо одним из 16-ти базовых цветов.

К примеру, если при открытии тела документа, элементу **body** прописать:

`<body link="red" vlink="green" alink="white">`

то все ссылки в данном документе станут красными, уже посещенные ссылки станут зелеными, а ссылки в момент нажатия будут белыми.

Если нужно чтобы в каком-то месте ссылка имела другой цвет, тогда необходимо внутри html ссылки прописать уже знакомый элемент **font** с атрибутом цвета:



```
<a href="http://www.google.com.ua"><font color="black">Черная ссылка</font></a>
```

## Работа с изображениями

Существует три типа файлов изображений, которые можно вставить на страницы:

- GIF (Graphics Interchange Format);
- JPG / JPEG (Joint Photographic Experts Group);
- PNG (Portable Network Graphics)

### Форматы:

**GIF** - использует всего 256 цветов и соответственно лучше подходит для рисунков с малым количеством оттенков. Этот формат поддерживает прозрачность изображений.

**JPEG** - формат изображений, который использует до миллиона цветов. Обычно используется для фотографий.

**PNG** - сравнительно новый формат. По многим параметрам превосходит JPEG и GIF: миллионы цветов и эффективное сжатие. Также поддерживает прозрачность.

Для вставки изображения на страницу необходимо использовать тег **IMG**. Элемент **IMG** не имеет закрывающего тега.

Пример:

```
<!-- если бы изображение находилось в той же папке  
что и страница -->  
  
<!-- если бы изображение находилось в папке images  
-->  
  
<!-- если б находилось на сайте www.zv.com -->  
  
<!-- если б находилось на сайте www.zv.com в папке  
images -->  

```

### АТРИБУТЫ:

**SRC** (сокращение от source - положение)- указывает, где находится изображение.

**ALIGN** - определяет способ выравнивания картинки по горизонтали. Обычно используют **LEFT** (выравнивание по левому краю, текст будет обтекать справа) и **RIGHT** (выравнивание по правому краю, текст обтекает слева). Если на странице есть текст, то это обязательное свойство.

**HSPACE** и **VSPACE** - отступы в пикселях по горизонтали и по вертикали от картинки до других объектов документа. Легко запомнить название, если взять и просто перевести с английского. **HSPACE** - Horizontal Space - горизонтальный отступ и **VSPACE** - Vertical Space - вертикальный отступ.

**HEIGHT** и **WIDTH** - высота и ширина изображения в пикселях. Золотое правило – всегда явно задавать размеры картинки в атрибутах **HEIGHT** и **WIDTH**, резервируя тем самым место в окне браузера еще до загрузки изображения. В противном случае документ при загрузке каждой картинки будет заново перерисовываться. Но, в принципе, можно и не задавать размеры, просто будет дольше загружаться. Также рекомендуется не искажать реальные размеры картинки.

**ALT** - определяет текст, отображаемый браузером на месте изображения, если браузер не может найти файл с изображением или включен в текстовый режим. В качестве значения задается текст с описанием изображения.

```

```

**TITLE** - Определяет заголовок картинки, т.е. при наведении на картинку высветится надпись с текстом из атрибута **title**.

```

```

Пример:

```
<!-- пример с отступами и выравниванием по левому  
краю-->  
<p align="justify"> 
```

### Ссылка в виде изображения

Для этого вместо текста ссылки необходимо вставить изображение. Когда делаем изображение ссылкой, вокруг него появляется некрасивая рамка (**border**) и, чтобы ее убрать, используют дополнительный атрибут **border**.

Пример:

```
<a href="http://www.zv.com">   
</a>
```

### Ссылки не на http

Ссылка на анонимный ftp:

```
ftp://ftp.example.com/example.zip
```

Ссылка на не анонимный ftp:

```
ftp://username:password@ftp.example.com/example.zip
```

### Ссылки на почту

При нажатии на ссылку запустится почтовая программа-клиент с заполненным полем имени получателя.

`<a href="mailto:admin@mail8.dgtu.dn.ua">Написать письмо админу </a>`

### АТТРИБУТЫ:

**TARGET** - указывает где открывать страницу, на которую ведет **html** ссылка. По умолчанию она открывается в том же окне. Чтобы открывалась в новом, следует написать **target = "\_blank"**.

**TITLE** - указывает заголовок ссылки, который появляется при наведении на нее.

Ссылка на почту с указанием темы документа:

[mailto:stas@gmail.com?subject=Hi&body=text\\_body](mailto:stas@gmail.com?subject=Hi&body=text_body)

## Ссылки на Skype

Простейшая ссылка на Skype имеет вид:

`<a href="skype:SkypeUser">SkypeUser</a>`

Звонок пользователю Skype-to-Skype или на телефон:

`<a href="skype:SkypeUser?call">SkypeUser</a>`

`<a href="skype:+12345678?call">+12345678</a>`

## ICQ

`<a href="http://www.icq.com/whitepages/cmd.php?uin=UserNumber&action=message">UserNumber</a>`

Значок ICQ с номером:

``

В HTML5 отменены:

- **shape** и **coords** для **a**
- **target** для **link**
- **nohref** для **area**
- **align** для **caption**, **iframe**, **img**, **input**, **object**, **legend**, **table**, **hr**, **div**, **h1-h6**, **p**, **col**, **colgroup**, **tbody**, **td**, **tfoot**, **th**, **thead**, **tr** и **body**
- **alink**, **link**, **text** и **vlink** для **body**
- **hspace** и **vspace** для **img** и **object**

### Задание на лабораторную работу:

Создать три страницы на общую тему из лабораторной работы №1. Создать файл фамилия\_index.html, который содержит ссылки на три созданные страницы. Файлы должны называться фамилия\_index1.html, фамилия\_index2.html, фамилия\_index3.html. Созданные дополнительно страницы должны находиться в папке pages, изображения в папке images.

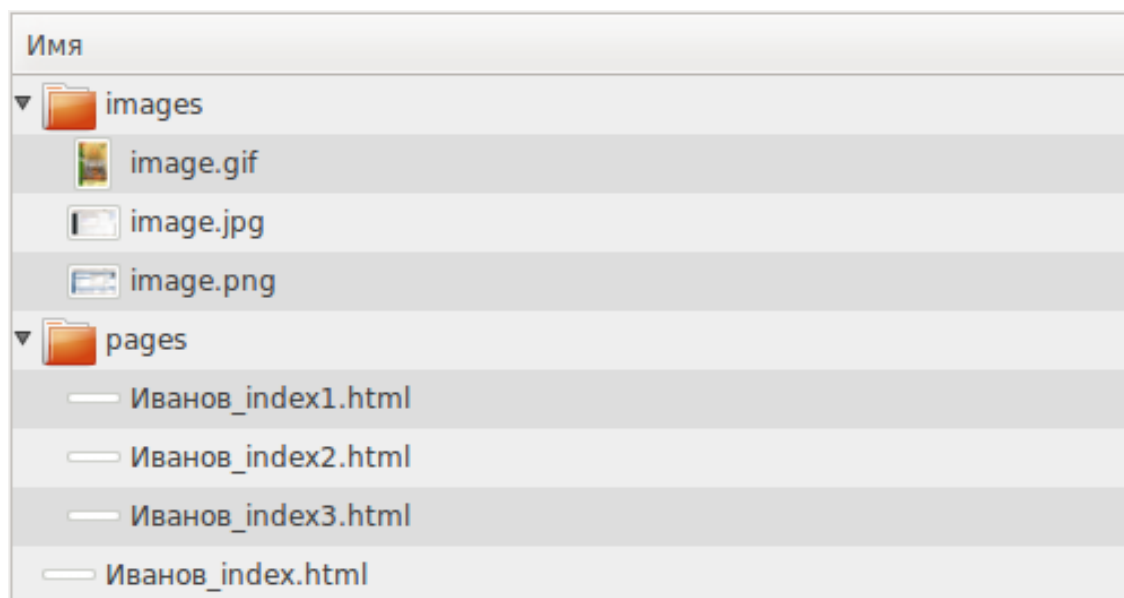


Рисунок 2.1 - Пример размещения файлов

В лабораторной работе необходимо использовать изображения разных типов – JPG, PNG, GIF (см. рис. 2.1). На одной из страниц использовать фоновое изображение. Одна из картинок должна быть расположена горизонтально по центру и на расстоянии 200px от вышерасположенного контента, иметь заголовок, подпись под рисунком(HTML5).

На главной странице(фамилия\_index.html) разместить уменьшенную копию одного из изображений с ссылкой на полнуюразмерную картинку (сохранять пропорции исходного изображения обязательно!). На каждой из дополнительных страниц следует разместить обратную ссылку на главную страницу. Параметры ссылок выбираются по варианту (см. табл. 2.1).

На главной странице добавить контакты разработчика.

### Содержание отчета:

1. Задание.
2. Таблица с перечнем использованных тегов.
3. Текст HTML-документа.
4. HTML-страницы.
5. Выводы по работе.

Таблица 2.1 - Параметры ссылок

№	Параметры ссылок (0 – открывать в этом же окне, 1 – открывать в новом окне)	№	Параметры ссылок (0 – открывать в этом же окне, 1 – открывать в новом окне)
1	0011	12	0110
2	0110	13	1100
3	1100	14	0101
4	0101	15	1001
5	1001	16	1010
6	1010	17	1110
7	1110	18	0111
8	0111	19	1101
9	1101	20	1011
10	1011	21	0011
11	0011	22	0110

### Контрольные вопросы:

1. Есть ли ошибки в следующей ссылке:  
`<a href="report.pdf" title="report as PDF, 2.3MB">  
get our latest report</a>?`
2. Для чего в ссылках используется атрибут target?
3. Существует ли атрибут для ссылок, который описывает отношения между документами?
4. Как можно создать ссылку, которая после щелчка на ней переходит к элементу дальше вниз по странице?

## Лабораторная работа №3

### Верстка таблиц в HTML

**Тема лабораторной работы:** создание таблиц средствами языка HTML.

**Цель работы:** изучение возможностей языка HTML по созданию и управлению внешним видом таблиц.

### Верстка таблиц

Таблицы являются очень удобным средством форматирования данных на Web странице. Основное удобство заключается в том, что браузер берет на себя заботу о прорисовке рамки таблицы. Размер рамки может быть автоматически согласован с размером окна просмотра в браузере и, разумеется, с размером находящихся в ячейках таблицы строк текста и рисунков. Кроме того, таблицы позволяют решать чисто дизайнерские задачи: выравнивать части страницы друг относительно друга, размещать рядом рисунки и текст, управлять цветовым оформлением. Необходимо помнить, что последовательность элементов описывает таблицу сверху вниз и справа налево.

Элемент таблица создается с помощью тега `<table> ... </table>`. Вся таблица должна находиться внутри него. Он автоматически переводит строку до и после таблицы.

Атрибуты тега `<table>` позволяют управлять отображением таблицы; изменять ее цвет, толщину рамки и многое другое. Атрибуты таблицы приведены в таблице 3.1.

Таблица 3.1 - Атрибуты тега `<table>`

Атрибут	Значение по умолчанию	Возможные значения	Описание
<b>align</b>	left	left, right, center	Выравнивание таблицы.
<b>width</b>	Рассчитывается на основе ширины ячеек	100px, 500px, 100% и т.д.	Ширина таблицы. Можно указать конкретный размер в пикселях или в процентах от свободного пространства.
<b>border</b>	0	Любое положительное целое число	Ширина границы таблицы в пикселях.
<b>cellspacing</b>	0 если <b>border</b> не задан, 2 в противном случае	Любое положительное целое число	Каждую ячейку браузер обводит своей собственной рамкой, и этот параметр задает ширину пространства между ними.
<b>cellpadding</b>	0	Любое положительное целое число	Ширина пространства между рамкой ячейки таблицы и ее содержанием внутри.
<b>bgcolor</b>	Обычно белый	Цвет в 16ом формате или название цвета	Цвет фона таблицы
<b>background</b>	-	Любое изображение, которое за-	Изображение, которое

Атрибут	Значение по умолчанию	Возможные значения	Описание
<b>d</b>		дается адресом	будет отображаться в виде фона таблицы.
<b>Cols</b>	-	Любое положительное целое число	Количество столбцов в таблице, помогая браузеру в подготовке к ее отображению.
<b>Frame</b>	border	Void - Не отрисовывать границы. Border - Граница вокруг таблицы. Above - Граница по верхнему краю таблицы. Below - Граница снизу таблицы. Hsides - Добавить только горизонтальные границы (сверху и снизу таблицы). Vsides - Рисовать только вертикальные границы (слева и справа от таблицы). Rhs - Граница только на правой стороне таблицы. Lhs - Граница только на левой стороне таблицы.	Сообщает браузеру, где отображать границы вокруг таблицы. Толщина границы указывается с помощью параметра border.
<b>rules</b>	В зависимости о значения атрибута <b>border</b> .	All - Линия рисуется вокруг каждой ячейки таблицы. Groups - Линия отображается между группами, которые образуются тегами <thead>, <tfoot>, <tbody>, <colgroup> или <col>. Cols - Линия отображается между колонками. None - Все границы скрываются. Rows - Граница рисуется между строками таблицы, созданных через тег.	Сообщает браузеру, где отображать границы между ячейками. Толщина границы и ее цвет указывается с помощью параметров border и bordercolor.

Таблица состоит из строк, строки состоят из ячеек. Количество ячеек в каждой строке таблицы должно быть одинаковым. Таким образом, структура таблицы имеет вид:

```
<TABLE>
```

```
<TR>
```

```
<TD>Если в таблице два тега TR, то в ней две строки</TD>
```

```
<TD>Второй столбец</TD>
```

```
<TD>Третий столбец</TD>
```

```
</TR>
```

```
<TR>
```

```
<TD>В этой таблице три</TD>
```

```
<TD>столбца</TD>
```

```
<TD>Это последняя ячейка </TD>
```

```
</TR>
```

</TABLE>

Строки таблицы создаются с помощью тегов <TR>... </TR>, а ячейки создаются с помощью тегов <TD> ... </TD>. Кроме того, можно задать специальный вид ячеек – заголовочные. Это можно сделать с помощью тега <TH>. Ячейки, созданные с помощью тега <TH>, имеют специальное форматирование: текст выделен жирным и выровнен по центру.

Между тегами создания строк и тегами создания ячеек не рекомендуется ставить пробелы либо другие символы. Рассмотрим пример простейшей таблицы:

```
<table border=1>
  <tr><td>Первая ячейка</td><td>Вторая ячейка</td>
</tr>
  <tr><td>новый</td><td>ряд</td>
</tr>
</table>
```

В этом случае получаем следующую таблицу (рис. 3.1):

Первая ячей- ка	Вторая ячей- ка
новый	ряд

Рисунок 3.1 – Результат примера

Тег <TR> имеет несколько атрибутов, которые описаны в таблице

3.2. Таблица 3.2 - Атрибуты элемента строки - <TR> .

Атрибут	Значение по умолчанию	Возможные значения	Описание
<b>align</b>	Left	Left, Center, Right, Justify	Выравнивание внутри всех ячеек таблицы
<b>valign</b>	Top	Top, Middle, Bottom, Baseline	Выравнивание по вертикали.
<b>bgcolor</b>	Обычно белый	Цвет в 16ом формате или название цвета	Цвет фона таблицы
<b>background</b>	-	Любое изображение, которое задается адресом	Изображение, которое будет отображаться в виде фона таблицы.

Тег <TD> предназначен для создания ячейки таблицы, атрибуты тега приведены в таблице 3.7. В ячейке можно сделать вложенную таблицу, а в ней еще одну.

Таблица 3.3 - Атрибуты элемента ячейка - <TD> .

Атрибут	Значение по умолчанию	Возможные значения	Описание
<b>align</b>	Left	Left, Center, Right, Justify	Выравнивание внутри



		fy	всех ячеек таблицы
<b>valign</b>	Top	Top, Middle, Bottom, Baseline	Выравнивание по вертикали.
<b>bgcolor</b>	Обычно белый	Цвет в 16ом формате или название цвета	Цвет фона таблицы
<b>background</b>	-	Любое изображение, которое задается адресом	Изображение, которое будет отображаться в виде фона таблицы.
<b>rowspan</b>	1	Любое положительное целое число	Объединяет указанное количество ячеек в одну по вертикали.
<b>colspan -</b>	1	Любое положительное целое число	Объединяет ячейки по горизонтали.
<b>width</b>	-	Любое положительное целое число	Рекомендуемые размеры ячейки по горизонтали.
<b>height</b>		Любое положительное целое число	Рекомендуемые размеры ячейки по вертикали.
<b>Nowrap</b>			Запрещает перенос текста на другую строку

Рассмотрим пример управления свойствами ячеек таблицы.

```
<table border=2>
  <tr><td>1</td><td>2</td><td>3</td>
  <td>4</td><td>5</td></tr>
  <tr><td colspan=2 rowspan=2>6</td>
  <td colspan=2>7</td><td rowspan=2>8</td></tr>
  <tr><td>9</td><td>10</td></tr>
  <tr><td>11</td><td>12</td><td>13</td><td>14</td>
  <td>15</td></tr>
</table>
```

В этом случае получаем следующую таблицу (рис. 3.2):

1	2	3	4	5
6		7	8	
		9		
1	1	1	1	1
1	2	3	4	5

Рисунок 3.2 – Результат примера

В первой строке получаем 5 элементов `<td>`. Во второй строке элементов `<td>` теперь уже всего 3, но заметьте, что если просуммировать все значения `colspan`, то снова получится 5. По умолчанию для каждой ячейки параметр `colspan` равен 1.

При создании таблиц можно использовать ряд дополнительных элементов. Описание этих элементов приведено в таблице 3.4.

Таблица 3.4 - Элементы создания таблиц.

Элемент	Назначение
<caption>	Заголовок таблицы.
<col>	Позволяет задать атрибуты колонки. Ширина и другие атрибуты одной или нескольких колонок таблицы.
<colgroup>	Ширина и стил одной или нескольких колонок таблицы.
<tbody>	Для хранения одной или нескольких строк таблицы. Это позволяет создавать структурные блоки, к которым можно применять единое оформление через стили.
<tfoot>	Для хранения одной или нескольких строк, которые представлены внизу таблицы.
<thead>	Для хранения одной или нескольких строк, которые представлены вверху таблицы.

### Задание на лабораторную работу:

Основной задачей лабораторной работы является изучение возможностей таблиц при создании структуры сайта.

Сверстайте с помощью таблиц шаблон сайта, согласно заданному варианту. Поместите в этот шаблон контент трех страниц из лабораторной работы №1. Структура шаблона сайта задается с помощью универсального шаблона 3.5.

Рассмотрим подробнее правила задания шаблона с помощью универсальной таблицы:

1. По умолчанию, каждая ячейка таблицы является резиновой, т.е. расширяется в лево/право или вверх/вниз если там нет другой ячейки.
2. Ширина или высота ячейки, заданная фиксированным значением не может быть изменена.
3. Ширина и высота ячейки может быть задана в процентах от размеров родительского элемента.
4. В создании шаблона принимают участие только те ячейки, номера которых перечислены в задании.

Таблица 3.5 - Универсальный шаблон

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

Возможные значения параметры ячеек таблицы описаны в таблице 3.6.

Таблица 3.6 - Описание параметров ячеек таблицы

Параметр	Значение	Описание
Ширина	Ф - 200	Ширина ячейки фиксированная, 200 пикселей
	ФП-50	Ширина ячейки фиксированная, 50% ширины контейнера
	Н	Ширина не задана, определяется содержимым
	М	Максимальная ширина ячейки
Содержимое	МВ-5	Вертикальное меню, в виде списка из 5 элементов
	МГ-3	Горизонтальное меню в виде набора ссылок из 3 элементов
	З	Заголовок
	П	Параграф
	ЗП	Заголовок и параграф
	А	Автор и дата последнего обновления
	И	Изображение
	И-4	4 упорядоченных изображения
	Е	e-mail
Фон	FF00FF	Цвет фона
	img	Фон задается изображением
	-	Нет фона
	A0F	Цвет в 16ом формате, сокращенная запись для AA00FF

Варианты заданий на лабораторную работу определяются номером студента по списку. Контент, который помещается в шаблон, студент берет из лабораторной работы №1. Варианты структуры таблицы и свойства ее ячеек задаются таблицей 3.8

Параметры таблицы задаются таблицей 3.7.

Таблица 3.7 - Расположение и размер таблицы

Вариант	Ширина	Выравнивание	cellpadding	Cellspacing	Граница
1	ФП-100	-	5	3	2
2	Ф-600	Центр	-	10	1
3	Ф-800	Left	-	5	5
4	М	Right	2	4	2
5	ФП-100	-	3	-	4
6	ФП-80	Центр	-	-	3
7	М	Left	5	2	2
8	Ф-800	Right	4	3	2
9	Н	-	3	4	3
10	ФП-50	Центр	-	5	4

Таблица 3.8 - Варианты заданий

№	Задание	№ ячеек							
1	Таблица	1	2	3	4	5	7	9	13

№	Задание	№ ячеек							
	Ширина	Ф200, ФП50, Н							
	Содержимое	2хИ, А, Е, МГ-5, 3П, П, МВ-6							
	Фон	#ABC	#00D	#D00	img	#CDE	#ABC	#CDE	img
2	Таблица	2	5	8	9	10	11	12	13
	Ширина	Ф300, ФП30, М							
	Содержимое	И, А, Е, МГ-3, 3П, 2хП, МВ-4							
	Фон	img	#CDE	#DEF	#AAA	img	#BCE	img	img
3	Таблица	1	4	7	9	10	13	14	15
	Ширина	Ф200, ФП50, Н							
	Содержимое	И, А, Е, 2хМГ-2, 3П, П, МВ-5							
	Фон	#DDE	img	img	#CDE	#BBC	img	#DED	#ACB
4	Таблица	1	3	4	7	8	13	14	15
	Ширина	Ф200, М, Н							
	Содержимое	И, А, Е, МГ-2, 3П, П, 2хМВ-3							
	Фон	#FCE	#DCE	#CCC	img	img	#ADC	img	img
5	Таблица	1	4	6	9	12	13	14	15
	Ширина	М, ФП50, Н							
	Содержимое	И-4, А, Е, МГ-2, 3П, П, МВ-5							
	Фон	img	#CCE	#DBF	img	#AAA	img	#BCE	img
6	Таблица	1	3	5	7	9	10	12	13
	Ширина	Ф200, ФП25, Н							
	Содержимое	И, А, Е, МГ-4, 3х3П, П, МВ-2							
	Фон	img	#FCE	#AAA	img	#BBB	#CDE	#AED	img
7	Таблица	3	4	7	8	9	10	12	15
	Ширина	Ф200, ФП50, Н							
	Содержимое	2хИ, А, Е, МГ-3, 3П, П, МВ-5							
	Фон	#DDC	#BCE	img	img	img	#EDA	img	#FAB
8	Таблица	2	3	5	6	9	12	14	15
	Ширина	Ф240, ФП40, М							
	Содержимое	И, А, Е, МГ-4, 3П, 3хП, МВ-5							
	Фон	#DCC	img	#CDE	#AAC	#ABC	#FEC	img	img
9	Таблица	1	3	4	5	6	10	12	13
	Ширина	Ф250, ФП50, Н							
	Содержимое	И-5, А, Е, МГ-2, 3П, П, МВ-4							
	Фон	img	img	#DEE	#AAC	#ADA	img	#EDB	#BBD
10	Таблица	1	2	3	5	7	10	11	13
	Ширина	ФП40, М, Н							
	Содержимое	И, А, Е, 2хМГ-2, 3П, П, МВ-5							
	Фон	#ABC	#AAF	img	img	#DDE	#BCE	#CCE	img
11	Таблица	1	2	4	5	7	9	12	15
	Ширина	Ф200, ФП50, Н							

№	Задание	№ ячеек							
	Содержимое	3xИ, А, Е, МГ-2, 3П, П, МВ-2							
	Фон	img	#BCE	img	#DDF	#ACA	img	#ABA	img
12	Таблица	1	3	7	9	10	11	13	15
	Ширина	М ФП50, Н							
	Содержимое	И, А, Е, МГ-4, 3П, П, 3xМВ-5							
	Фон	#ADA	img	#CCC	img	#DDD	#BCB	img	#ADE
13	Таблица	3	4	7	9	10	11	12	14
	Ширина	Ф200, ФП50, Н							
	Содержимое	И, А, Е, МГ-3, 2x3П, П, МВ-5							
	Фон	img	#CDE	#FFA	img	img	#AAF	#BCE	img
14	Таблица	1	2	3	4	5	6	10	12
	Ширина	Ф250, ФП40, Н							
	Содержимое	И, А, Е, 2xМГ-2, 3П, П, МВ-5							
	Фон	#CEA	#FED	img	#ACA	img	#DCE	#BBC	img
15	Таблица	2	3	5	6	7	8	13	14
	Ширина	Ф300, ФП50, М							
	Содержимое	И, А, Е, 2xМГ-2, 3П, П, 2xМВ-3							
	Фон	#BCA	img	#CDA	#ACE	img	#DDD	img	img

Пояснения к таблице:

1. Если в качестве фона указано **img**, то это означает использовать любую фоновую картинку.

2. Для каждого варианта есть обязательный набор содержимого (расшифровку см. в таблице 3.6). Если перед обозначением указан множитель (например, 2xИ), то это говорит о том, что данный элемент должен быть размещен указанное число раз. Место для размещения элементов выбирается студентом самостоятельно, исходя из получившейся после упрощения таблицы. К примеру, горизонтальные меню лучше размещать в длинных ячейках, вертикальные - в высоких и т.д.

3. Студент самостоятельно выбирает к каким ячейкам таблицы следует применить параметры ширины (расшифровку см. в таблице 3.6)

### Пример реализации

Разберем пример реализации одного из заданий. В таблице 3.9 приведено задание по реализации шаблона сайта. Таблица занимает 100% ширины родительского контейнера (ФП-100), Выравнивание ячеек по умолчанию, отступы от границ ячейки **cellpadding=2**, отступы между ячейками **Cellspacing=2**, ширина границы **border=2**.

Таблица 3.9 - Пример задания

Вариант	Ширина	Выравнивание	cellpadding	Cellspacing	Граница
Пример	ФП-100	-	2	2	2

№ варианта	Задание	№ ячеек							
Пример	Таблица	1	4	5	8	11	13	15	
	Ширина		Ф300					Ф600	
	Содержимое	МГ-5	МВ-4	31	2П	И	А	Е	
	Фон	Синий	Синий				Синий	Синий	

Посмотрим на универсальную таблицу 3.5 и вычеркнем из нее не использованные поля, таблица 3.10а. Ячейка 1 расширяется влево за счет ячеек 2 и 3. Ячейки 5, 6 и 11 также расширяются влево за счет ячеек 6, 9 и 12 соответственно. Ячейку 14 убираем, расширяя ячейку 13. В результате получаем таблицу 3.10б. 4я ячейка имеет фиксированную ширину в 300 пикселей (Ф300), ячейка 13, также имеет фиксированную ширину в 600 пикселей, ширина остальных ячеек зависит от их содержимого и самой таблицы.

Таблица 3.10а - Универсальная таблица

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

Таблица 3.10б - Шаблон согласно заданию

1	
4	5
	9
	12
13	15

```
<html>
  <head><title>Пример создания таблиц</title></head>
<body>
  <table width=100% border="2" cellpadding="2" cell-
spacing="2">
    <tr><td colspan="3" bgcolor="blue">1</td></tr>
    <tr><td rowspan="3" width=200px bgcolor=
or="blue">4</td><td colspan="2">5</td></tr>
    <tr><td colspan="2">9</td></tr>
    <tr><td colspan="2">12</td></tr>
    <tr><td colspan="2" bgcolor="blue">13</td><td
bgcolor="blue" width="600px">15</td></tr>
  </table>
</body>
```

</html>

Создадим шаблон страницы согласно заданию. Результат приведен на рис. 3.3.

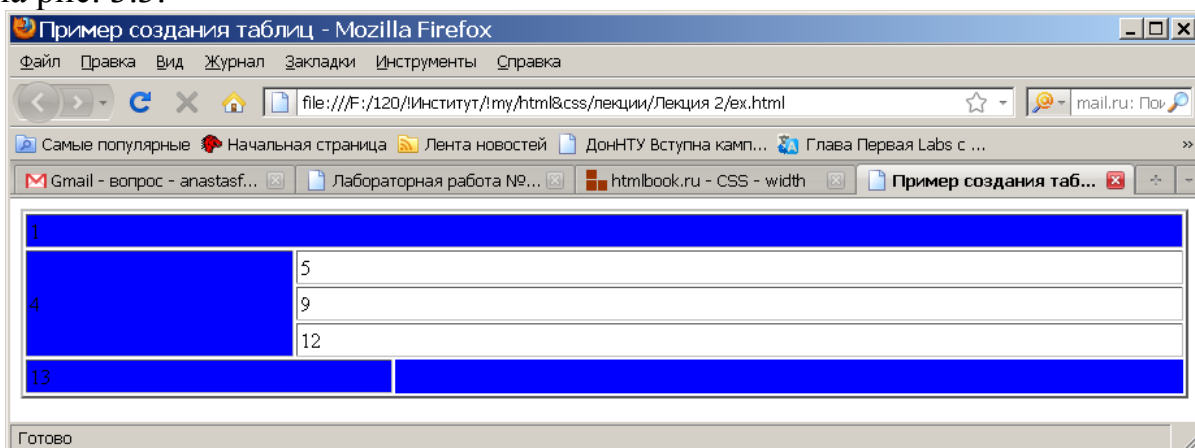


Рисунок 3.3 - Пример создания таблицы

Теперь заполним ячейки таблицы контентом. В ячейке 4 необходимо разместить горизонтальное меню из 5ти элементов (МГ-5). В ячейке 4, поместим вертикальное меню из 4х пунктов; 5я ячейка содержит заголовок, 9я – 2 параграфа текста. Изображение помещаем в ячейку 12. Ячейки 13 и 15 содержат сведения об авторе и его адрес электронной почты соответственно.

Так как мы не можем пока развернуть список (средствами HTML) в горизонтальную линию, реализуем горизонтальное меню с помощью таблицы.

#### Реализация горизонтального меню:

<html>

```
<head><title>Пример создания таблиц</title></head>
```

```
<body link="red" vlink="yellow" text="red">
```

```
<table width=100% border="2" cellpadding="2" cellspacing="2">
```

```
<tr><td colspan="3" bgcolor="lightblue">
```

```
<table border="0" rows="1" cols="5" width="100%">
```

```
<tr>
```

```
<td width="19%">Пункт1</td>
```

```
<td width="19%">Пункт2</td>
```

```
<td width="19%">Пункт3</td>
```

```
<td width="19%">Пункт4</td>
```

```
<td width="19%">Пункт5</td>
```

```
</tr>
```

```
</table>
```

```
</td></tr>
```

```
<tr><td rowspan="3" width=200px bgcolor="lightblue" align="left" valign="top">
```

```
<ol type="none">
```

```
<li>Пункт1</li>
```

```

        <li>Пункт2</li>
        <li>Пункт3</li>
        <li>Пункт4</li>
    </ol>
</td><td colspan="2">Заголовок 1</td></tr>
<tr><td colspan="2">
    <p>Это первый параграф текста ...</p>
    <p>Это второй параграф текста ...</p>
</td></tr>
<tr><td colspan="2">
    
</td></tr>
<tr><td colspan="2" bgcolor="lightblue"><author>Фонотов А.М.</author></td>
    <td bgcolor="lightblue" width="600px">Пишите:
    <a href="mailto:afm-mars@rambler.ru">afm-
mars@rambler.ru</a></td></tr>
</table>
</body>
</html>

```

Пункт1	Пункт2	Пункт3	Пункт4	Пункт5
Пункт1 Пункт2 Пункт3 Пункт4	<p>Синтаксис HTML5</p> <p>Это первый параграф текста. HTML5 возвращает нас к стилю десятилетней давности, когда практиковалось не закрывать некоторые теги, писать значения без кавычек и по желанию набирать теги в верхнем или нижнем регистре. Такая вольность не означает, что любые правила должны игнорироваться, по-прежнему следует соблюдать корректную вложенность тегов и вставлять обязательные элементы. Отход от жесткого синтаксиса XHTML позволяет сосредоточиться на содержании сайта, а не на соблюдении пустых формальностей, большинство из которых вызывает раздражение из-за своего несущественного значения и ненужности.</p> <p>Это второй параграф текста. Реализации и спецификации должны идти вместе в изящном танце. Вы не хотите, чтобы реализация происходила до окончания работ над спецификацией, потому что люди начнут зависеть от деталей реализации, и это будет сдерживать спецификацию. Однако вы также не желаете, чтобы спецификация была завершена раньше реализации и авторы начнут экспериментировать с реализацией, потому что вам нужна обратная связь. Здесь существует неизбежное напряжение, но мы просто должны колебаться до конца.</p> 			
Фонотов А.М.		Пишите: <a href="mailto:afm-mars@rambler.ru">afm-mars@rambler.ru</a>		

Рисунок 3.4 – Пример реализации индивидуального задания

### Содержимое отчета:

1. Задание.
2. Таблица с перечнем использованных тегов.
3. Подготовка к выполнению. Анализ таблицы и ее содержимого.
4. Текст HTML-документа.
5. Выводы по работе.



### **Контрольные вопросы:**

1. Какие теги создания структуры таблицы Вы знаете?
2. Перечислить параметры тега `<TABLE>`.
3. Перечислить параметры тега `<TR>`.
4. Перечислить параметры тега `<TD>`.
5. Приведите пример задания фона и изображения заднего плана для таблиц, строк и ячеек.
6. Назначение параметров `cellpadding` и `cellspacing`.
7. Варианты задания ширины таблицы и столбцов.
8. Объясните правила объединения ячеек в таблице.
9. Поясните назначение параметров `colspan` и `rowspan`.
10. Выравнивание контента в ячейке таблицы.

## Лабораторная работа №4

### Каскадные таблицы стилей CSS

**Тема лабораторной работы:** управление стилями оформления Web-страниц.

**Цель работы:** изучить возможности CSS по созданию различных стилей оформления Web-страниц.

### Каскадные таблицы стилей CSS

В CSS, в отличие от HTML нет ни элементов, ни атрибутов, ни тегов. Основной структурной единицей здесь является правило, которое определяет, как будет выглядеть тот или иной элемент в документе.

Рассмотрим структуру правила (рисунок 4.1):



Рисунок 4.1 – Структура правила в CSS

Как видно из рисунка 4.1, сначала записывается так называемый **селектор**, показывающий к какому html тегу (тегам) применяется то или иное свойство

Далее, непосредственно за селектором, пишется **блок объявления стилей**, который обязательно заключается в фигурные скобки.

Каждое объявление в свою очередь состоит из **свойства** и его **значения**. После свойства ставится двоеточие. Правило может содержать в себе несколько объявлений. В таком случае они должны быть отделены друг от друга точкой с запятой (см. рисунок 4.1), причем после последнего объявления точку с запятой можно не ставить.

Показанное выше правило указывает на то, что все заголовки первого уровня в документе будут голубого цвета с размером шрифта 14 пикселей.

В каждом документе который мы хотим подключить, в голове документа (между тегами <head> и </head>) необходимо прописать строчку:

```
<link rel="stylesheet" type="text/css"
href="style.css">
```

Эта строка указывает браузеру, что он должен использовать правила отображения HTML-файла из CSS-файла.

### В HTML5!!!!

```
<link rel="stylesheet" href="styles.css">
```

### Свойства текста в CSS

В состав свойств текста входят свойства выравнивания такие, как **text-align** и **word-spacing**, а также свойства изменения стиля **text-decoration** и новое свойство **text-shadow**.

**text-indent** - определяет длину отступа в первой строке блока. Значение:

- любое соответствующее стандарту (длина); возможны отрицательные значения; значение по умолчанию равно 0;
- любое соответствующее стандарту процентное значение;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {text-indent: 20px;}
```

**text-align** - определяет выравнивание текста в элементе. Значение:

- left - выравнивание по левому краю;
- center - выравнивание по центру;
- right - выравнивание по правому краю;
- justify - выравнивание по обоим краям (по ширине);
- любая строка, соответствующая стандарту - определяет последовательность символов;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {text-align: center;}
```

**text-decoration.** Определяет оформление текста. Значение:

- none - отсутствие элементов оформления (по умолчанию);
- underline - подчеркивание;
- overline - линия над текстом;
- line-through - перечеркивание;
- blink - мерцание;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {text-decoration: underline;}
```

**text-transform.** Автоматически переводит в тексте буквы в верхний (нижний) регистр. Значение:

- none - отсутствие изменения регистра (по умолчанию);
- capitalize - переводит первую букву каждого слова в верхний регистр;
- uppercase - переводит все буквы в верхний регистр;
- lowercase - переводит все буквы в нижний регистр;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {text-transform: uppercase;}
```

**letter-spacing** - определяет интервал между символами текста. Значение:

- none — интервал, обычный для используемого шрифта;
- любая длина, соответствующая стандартам, длина интервала между букв;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {letter-spacing: 5px;}
```

**word-spacing** - определяет интервал между словами. Значение:

- none - интервал, обычный для используемого шрифта;
- любая длина, соответствующая стандарту, длина интервала между словами;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {word-spacing: 2px;}
```

**white-space** - определяет, как обрабатывать пробелы. Значение:

- normal - пробелы сворачиваются, если это необходимо для размещения элемента; этот способ HTML использует по умолчанию;
- pre - пробелы обрабатываются так, как указано в коде (предварительно отформатированный текст);
- nowrap - сворачиваются все пробелы;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {white-space: pre;}
```

## Свойства шрифта

Возможность управлять шрифтом - меняете ли вы его семейство, кегель или толщину - позволяет увеличить «блеск» и неповторимость веб-страниц. Шрифты в CSS имеют следующие свойства:



Рисунок 4.2 – Свойства шрифтов в CSS

**font-family** - определяет семейства шрифта, используемого в этом тексте. Может быть несколько таких семейств, отделенные друг от друга запятыми. Приоритет определяется порядком в этом списке. Значение:

- имя семейства - семейства шрифта, применяемое для вывода текста. Имена, состоящие из нескольких слов, должны заключаться в кавычки;
- имя типового шрифта, применяемого для вывода текста; типовыми могут быть следующие шрифты: serif, sans-serif, fantasy и monospace;
- inherit - применяется значение родительского элемента.

**Пример:**

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

**font-style** - определяет начертания шрифта, такие как курсив или наклонное. Значение:

- normal - обычное начертание (по умолчанию);
- italic - курсив;
- oblique - наклонное начертание; это начертание исключительно экранного шрифта; он имеет несколько меньший наклон чем курсив;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {font-style: italic;}
```

**font-variant** - определяет, будет ли шрифт выведен в виде малых прописных букв. Значение:

- normal - обычное начертание (по умолчанию);
- small-caps - выводит шрифт в виде малых прописных букв;
- inherit - применяется значение родительского элемента.

**Пример:**

```
p {font-variant: small-caps;}
```

**font-weight** - определяет толщину выводимого шрифта. Значение:

- normal - обычное начертание (по умолчанию);
- bold - полужирное начертание; полужирное начертание выделяет текст; линии становятся толще, а сам текст немного шире, чем при обычном начертании;
- bolder - жирный шрифт;
- lighter - светлый шрифт; тонкие, светлые начертания шрифтов производят менее сильное впечатление, чем обычные или полужирные, но они незаменимы, когда требуется легкий и простой внешний вид;
- 100-900 - число, указывающее толщину шрифта. 100 соответствует самой светлой толщине (lighter), 400 - normal, 700 - bold, 900 - bolder;
- inherit - применяется значение родительского элемента.

#### Пример:

```
p {font-weight: bold;}
```

**font-size** - определяет кегль (высоту символов) шрифта. Значение:

- абсолютный размер - для выражения кегля шрифта используется ключевые слова: xx-small (крайне малый), small (малый), medium (средний) (по умолчанию), large (большой), x-large (очень большое), xx-large (крайне большое);
- относительный размер - для выражения кегля шрифта используется ключевые слова: larger (меньше), smaller (больше);
- любое соответствующая стандарту высота - абсолютный размер шрифта; отрицательное значение не допускается;
- любое соответствующее стандарту процентное значение;
- inherit - применяется значение родительского элемента.

#### Пример:

```
p {font-size: 20px;}
```

**font** - свойство для установки сразу всех параметров шрифта. Если какие-то значение пропущены, то берется их значение по умолчанию. Значение:

- font-style - начертание;
- font-variant - значение, определяющее вывод шрифта в виде малых прописных букв;
- font-weight - толщина;
- font-size - кегль;
- line-height - интерлиньяж;
- font-family - семейство шрифтов;
- inherit - применяется значение родительского элемента.

#### Пример:

```
p {font: oblique 12pt "Helvetica Nue", serif; font-stretch: condensed;}
```

Скруглённые рамки

- border-radius

Тени блоков и тени текста

- text-shadow
- box-shadow

Анимация

- transitions
- transforms

Градиенты

- linear-gradient
- radial-gradient
- repeating-linear-gradient
- repeating-radial-gradient

### **Задание на лабораторную работу:**

Основной задачей лабораторной работы является изучение возможностей CSS для оформления стилей html-страниц.

Лабораторная работа базируется на материалах, созданных в лабораторной работе №2. В ходе ее выполнения были сформированы главная страница и три вспомогательные страницы.

**Задание.** Оформить с помощью CSS главную страницу и три вспомогательные страницы по следующим требованиям:

1. Главная страница должна содержать название работы, данные об авторе и его группе, ссылку на его почтовый адрес и меню, организованное в виде списка ссылок на 3 следующие страницы.

2. Каждая дополнительная страница, кроме ссылки возврата на главную страницу, должна содержать:

- а. не менее 2 заголовков одного уровня;
- б. один список;
- с. три параграфа;
- д. два изображения;
- е. на одной из трех страниц должна быть стилизованная таблица.

3. Верстка и форматирование страниц осуществляется с помощью CSS.

4. Общие стили оформления для текста, фона, списков, ссылок и таблиц необходимо вынести в отдельный css-файл. Файл должен содержать стили оформления, общие для всех 3 страниц.

5. Необходимо выделить локальные стили, каждой страницы согласно варианту и описать их в заголовке страниц.

6. Необходимо выполнить индивидуальное задание в соответствии с таблицами вариантов (табл. 4.1-4.8).

Таблица 4.1 - Форматирование body

№ варианта	Фон	Замостить	Заливка	Размер шрифта
1	картинка	нет		10
2	зеленый		обычная	11
3	картинка	горизонтально		12
4	картинка	вертикально		13
5	синий		линейная	14
6	картинка	все		15
7	картинка	нет		10
8	красный		градиентная	11
9	картинка	горизонтально		12
10	картинка	вертикально		13
11	красный		обычная	14
12	картинка	все		15
13	картинка	горизонтально		10
14	картинка	вертикально		11
15	желтый		линейная	12
16	картинка	все		13
17	картинка	нет		14
18	голубой		градиентная	15
19	картинка	горизонтально		10
20	картинка	вертикально		11
21	розовый		обычная	12
22	картинка	все		13

Таблица 4.2 - Форматирование заголовков

№ варианта	Размер заголовка	Выравнивание	Цвет
1	1	left	красный
2	2	center	голубой
3	3	right	зеленый
4	4	left	синий
5	5	center	черный
6	6	right	белый
7	1	left	розовый
8	2	center	салатный
9	3	right	оранжевый
10	4	left	фиолетовый
11	5	center	красный
12	6	right	голубой
13	1	left	зеленый
14	2	center	синий
15	3	right	черный



№ варианта	Размер заголовка	Выравнивание	Цвет
16	4	left	белый
17	5	center	розовый
18	6	right	салатный
19	1	left	оранжевый
20	2	center	фиолетовый
21	3	right	красный
22	4	left	голубой

Таблица 4.3 - Форматирование параграфов

№	Выравнивание	Начертание текста	Цвет шрифта	Гарнитура	Размер
1	left	жирный	черный	Times New R	12
2	center	курсив	белый	Courier	14
3	right	подчеркнутый	розовый	Tahoma	16
4	left	зачеркнутый	салатный	Arial	18
5	center	наклонный	оранжевый	Times New R	12
6	right	надчеркнутый	фиолетовый	Courier	14
7	left	жирный	красный	Tahoma	16
8	center	курсив	голубой	Arial	18
9	right	подчеркнутый	зеленый	Times New R	12
10	left	зачеркнутый	синий	Courier	14
11	center	наклонный	черный	Tahoma	16
12	right	надчеркнутый	белый	Arial	18
13	left	жирный	розовый	Times New R	12
14	center	курсив	салатный	Courier	14
15	right	подчеркнутый	оранжевый	Tahoma	16
16	left	зачеркнутый	фиолетовый	Arial	18
17	center	наклонный	красный	Times New R	12
18	right	надчеркнутый	голубой	Courier	14
19	left	жирный	черный	Tahoma	16
20	center	курсив	белый	Arial	18
21	right	подчеркнутый	розовый	Times New R	12
22	left	зачеркнутый	салатный	Courier	14

Таблица 4.4 - Форматирование изображений

№	Граница	Отступы(гориз-верт)	Обтекание	Скругление	Тень
1	OUTSET	left	Есть	нет	есть
2	INSET	right	нет	есть	нет
3	RIDGE	center	Есть	нет	есть
4	GROOVE	top	нет	есть	нет
5	DOUBLE	bottom	Есть	нет	есть

№	Граница	Отступы(гориз- верт)	Обтекание	Скругление	Тень
6	DASHED	top right	нет	есть	нет
7	DOTTED	75% 25%	Есть	нет	есть
8	SOLID	300px 500px	нет	есть	нет
9	OUTSET	left bottom	Есть	нет	есть
10	INSET	center top	нет	есть	нет
11	RIDGE	25% 75%	Есть	нет	есть
12	GROOVE	500px 300px	нет	есть	нет
13	DOUBLE	left	Есть	нет	есть
14	DASHED	right	нет	есть	нет
15	DOTTED	center	Есть	нет	есть
16	SOLID	top	нет	есть	нет
17	OUTSET	bottom	Есть	нет	есть
18	INSET	top right	нет	есть	нет
19	RIDGE	75% 25%	Есть	нет	есть
20	GROOVE	300px 500px	нет	есть	нет
21	DOUBLE	left bottom	Есть	нет	есть
22	DASHED	right bottom	нет	есть	нет

Таблица 4.5 - Форматирование списков

№ вари- анта	Маркер	Вид списка (обыч- ный или в одну стро- ку - inline)	Оформление текста в списке
1	disk	обычный	оранжевый
2	square	inline	фиолетовый
3	circle	обычный	красный
4	none	inline	голубой
5	img	обычный	зеленый
6	lower-alpha	inline	синий
7	lower-roman	обычный	черный
8	upper-alpha	inline	белый
9	decimal	обычный	розовый
10	disk	inline	салатный
11	square	обычный	оранжевый
12	circle	inline	фиолетовый
13	none	обычный	красный
14	img	inline	голубой
15	lower-alpha	обычный	зеленый
16	lower-roman	inline	синий
17	upper-alpha	обычный	черный
18	decimal	inline	оранжевый
19	disk	обычный	фиолетовый
20	square	inline	красный

№ варианта	Маркер	Вид списка (обычный или в одну строку - inline)	Оформление текста в списке
21	circle	обычный	голубой
22	none	inline	зеленый

Таблица 4.6 - Форматирование ссылок

№ варианта	фон	Оформление текста	Цвет
1	красный	наклонный	черный
2	голубой	надчеркнутый	белый
3	зеленый	жирный	розовый
4	синий	курсив	салатный
5	черный	подчеркнутый	оранжевый
6	белый	зачеркнутый	фиолетовый
7	розовый	наклонный	красный
8	салатный	надчеркнутый	голубой
9	оранжевый	жирный	зеленый
10	фиолетовый	курсив	синий
11	красный	подчеркнутый	черный
12	голубой	наклонный	белый
13	зеленый	надчеркнутый	розовый
14	синий	жирный	салатный
15	черный	курсив	оранжевый
16	белый	подчеркнутый	фиолетовый
17	розовый	зачеркнутый	красный
18	салатный	наклонный	голубой
19	оранжевый	надчеркнутый	зеленый
20	фиолетовый	жирный	синий
21	красный	курсив	черный
22	зеленый	зачеркнутый	белый

Таблица 4.7 - Форматирование таблиц

№	Фон	Выравнивание	Цвет четных строк	Цвет нечетных строк	Стиль линий
1	голубой	left	черный	фиолетовый	DASHED
2	зеленый	center	белый	красный	DOTTED
3	синий	right	розовый	голубой	SOLID
4	черный	left	салатный	зеленый	OUTSET
5	белый	center	оранжевый	синий	INSET
6	розовый	right	фиолетовый	черный	RIDGE
7	салатный	left	красный	белый	GROOVE
8	оранжевый	center	голубой	розовый	DOUBLE

№	Фон	Выравнивание	Цвет четных строк	Цвет нечетных строк	Стиль линий
9	фиолетовый	right	зеленый	салатный	DASHED
10	красный	left	синий	оранжевый	DOTTED
11	голубой	center	черный	фиолетовый	SOLID
12	зеленый	right	белый	красный	OUTSET
13	синий	left	розовый	голубой	INSET
14	черный	center	салатный	зеленый	RIDGE
15	белый	right	оранжевый	синий	GROOVE
16	розовый	left	фиолетовый	черный	DOUBLE
17	салатный	center	красный	белый	DASHED
18	оранжевый	right	голубой	розовый	DASHED
19	фиолетовый	left	зеленый	салатный	DOTTED
20	красный	center	синий	оранжевый	SOLID
21	голубой	right	черный	фиолетовый	OUTSET
22	зеленый	left	белый	красный	INSET

Таблица 4.8 - Элементы с индивидуальным форматированием (оформление выбрать самостоятельно)

№ варианта	Элемент 1	Элемент 2
1	Параграф	Элементы списка
2	Ссылка внутри параграфа	Изображение
3	Изображение в списке	Ячейка таблицы
4	Ссылка внутри ячейки таблицы	Параграф
5	Изображение	Заголовок
6	Изображение в списке	Параграф
7	Ссылка внутри ячейки таблицы	Заголовок
8	Изображение	Изображение
9	Параграф	Ячейка таблицы
10	Ссылка внутри параграфа	Изображение
11	Параграф	Ячейка таблицы
12	Ссылка внутри параграфа	Параграф
13	Изображение в списке	Заголовок
14	Ссылка внутри ячейки таблицы	Параграф
15	Изображение	Изображение
16	Параграф	Ячейка таблицы
17	Ссылка внутри параграфа	Параграф
18	Изображение в списке	Заголовок
19	Ссылка внутри ячейки таблицы	Элементы списка
20	Изображение	Ячейка таблицы

<b>№ варианта</b>	<b>Элемент 1</b>	<b>Элемент 2</b>
21	Параграф	Изображение
22	Ссылка внутри параграфа	Элементы списка

### **Содержимое отчета:**

1. Задание.
2. Таблица с перечнем использованных тегов.
3. Текст HTML-документа и CSS-файла.
4. Выводы по работе.

### **Контрольные вопросы:**

1. Что такое каскадные таблицы стилей?
2. Что такое правило стиля? Опишите различные компоненты и синтаксис?
3. Предположим, что имеется набор правил стиля, что нужно сделать, чтобы превратить их во внешнюю таблицу стилей?
5. Какие вы знаете свойства текста? Приведите пример.
6. Какие свойства таблиц стилей (CSS) предназначены для применения жирного шрифта и курсива?
7. Какие свойства таблиц стилей (CSS) являются альтернативой использования тега <FONT>?

## Лабораторная работа №5

### Создание прототипа интерфейса

**Тема лабораторной работы:** Разработка структуры сайта.

**Цель лабораторной работы:** Приобретение умений по созданию электронного прототипа веб-интерфейса. Приобретение практических навыков по созданию прототипов интерфейса.

### Общие сведения

Карта сайта - это структура системы. На карте сайта в виде дерева показаны все разделы, подразделы и страницы системы. Такая карта позволяет выстроить удобную информационную архитектуру продукта.

Схема навигации - это другой взгляд на карту сайта. На ней также показаны разделы и страницы, но здесь они сгруппированы по различным меню.

Проектирование каркаса является очень важным шагом при разработке дизайна сайта. В первую очередь каркас позволяет определить иерархию информации, благодаря ему вам будет проще планировать расположение элементов так, чтобы пользователь сразу обратил внимание на нужную информацию.

Каркас - это как архитектурный проект. Сначала нужно представить всё в двумерной черно-белой гамме прежде, чем начать подбирать цветовое решение. Каркасы важны, поскольку они позволяют разработчику планировать расположение и взаимодействие элементов интерфейса, не отвлекаясь на цвет, выбор шрифта и т.д. Если пользователь не может понять назначение элементов на черно-белом каркасе, не имеет значения, какие цвета вы в конечном итоге будете использовать.

### Инструментальные средства. Прототипирование

**Axure** - один из первых профессиональных инструментов для создания каркасов сайтов. До недавнего времени он был доступен только для Windows.

**Flairbuilder** - программа для создания блочной разметки и динамических прототипов (flairbuilder.com).

Каркасы **Balsamiq** напоминают эскизы, благодаря чему сразу видно, что каркасы не являются законченным продуктом. Balsamiq имеет огромную библиотеку повторно используемых компонентов, которые можно перетаскивать и с их помощью легко создать свои каркасы.

**Pencil Project** - средство для создания диаграмм и прототипов GUI. Полученные прототипы можно выгрузить в такие форматы, как PNG, HTML, PDF, ODF. (<http://pencil.evolus.vn>).

**Moqups** - интересное приложение для создания каркасов.

**Sketchy Illustrator wireframes** - готовые эскизы для создания каркаса сайта.

**Evernote** - набор каркасов для сайта в формате .psd есть тёмный и светлый вариант

**Wix.com** – бесплатный конструктор сайтов.

### Создание сетки

Есть много шаблонов сеток, доступных для скачивания, но если вы заинтересованы в том, чтобы самостоятельно настроить параметры и создать адаптивный шаблон, используйте **responsify.it**.

### Создание карты сайта

**XMind**. **XMind** - это открытое программное обеспечение для проведения мозговых штурмов и составления интеллект-карт. Удобно использовать для создания карты сайта.

**www.gliffy.com** – онлайн инструмент.

**mindmeister.com** – еще один инструмент для создания карт сайта онлайн.

### Цветовые схемы

Выбор цветовой схемы: <http://colorschemedesigner.com/>

**Kuler** от компании Adobe давно признан одним из лучших помощников профессионального веб-дизайнера.

**Color Palette Generator**- генератор цветовой палитры на основе фотографий.

**ColorZilla** - популярный плагин для Firefox позволяет прямо в браузере определить значения отдельных цветов и измерить разницу между ними.

**Material Design Palette Generator**-простой ресурс для тестирования и быстрого копирования цветов для Material Design.

### Задание на лабораторную работу:

Согласуйте индивидуальный шаблон с преподавателем. В соответствии с вариантом шаблона сайта и его тематикой определитесь с наполнением каждой страницы. Изучите выданный шаблон сайта, выделите логические блоки, на основании шаблона создайте макет первой страницы. В качестве рассматриваемых логических блоков можно рассматривать: название сайта, заголовок статьи, сама статья, заголовок новости, новость, баннер новости, основное меню, второстепенное меню, галерея картинок, элемент галереи и т.п.

Разработайте структуру главной страницы и ее наполнение. Представьте состав элементов главной страницы веб-сайта в виде вложенных друг друга блоков, как в примере на рисунке 5.1.

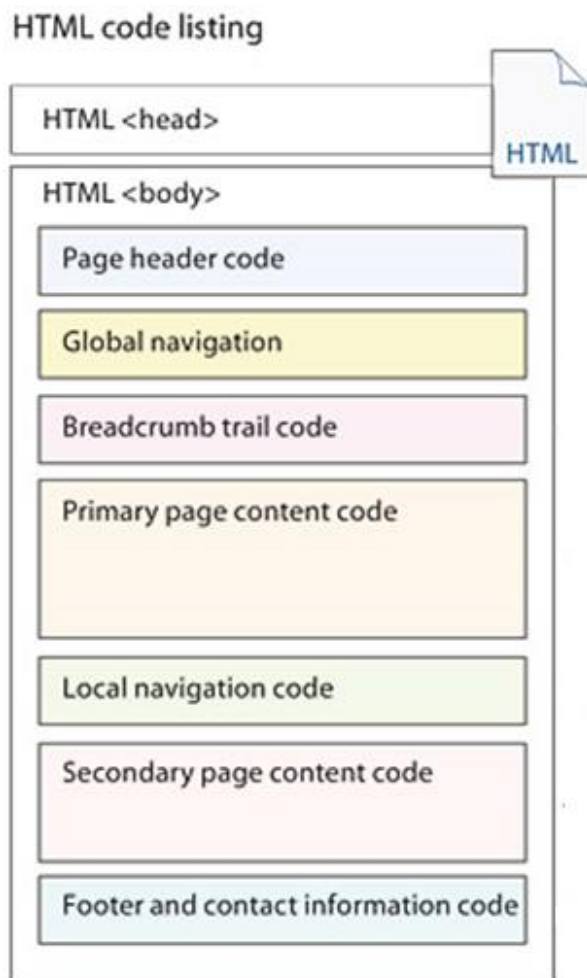


Рисунок 5.1 - Состав главной страницы

Самостоятельно продумайте и разработайте структуру второй страницы сайта, которая несет более подробную информацию по определенной новости, статье, заметке и т.п. Ориентируясь на цветовую схему выданного шаблона, доработайте цветовое оформление разрабатываемого сайта.

При принятии решения о выборе цветовой схемы рекомендуется использовать специальные инструменты.

### Содержимое отчета:

1. Задание.
2. Состав главной страницы
3. Структура страниц.

### Контрольные вопросы:

1. Что такое карта сайта?
2. Что такое каркас сайта?
3. Какие инструментальные средства для создания каркаса сайта вы знаете?
4. Как создать карту сайта?



## Лабораторная работа №6

### Форматирование сложных таблиц с данными

**Тема лабораторной работы:** Форматирование таблиц.

**Цель лабораторной работы:** Изучение способов сложных форматирования таблиц и отображения данных в таблицах.

### Группировка строк

В любой таблице можно выделить три логически цельных части: шапка, тело таблицы и «футер» (нижняя часть таблицы, footer или итоговая строка). Эти части состоят из строк, то есть строки таблицы можно сгруппировать в шапку, тело и футер.

Рассматриваемые части таблицы задаются следующим образом: шапка обозначается HTML элементом **THEAD** (**<THEAD>** и **</THEAD>**), тело задается элементом **TBODY** (**<TBODY>** и **</TBODY>**), а футер – элементом **TFOOT** (**<TFOOT>** и **</TFOOT>**). Все закрывающие теги являются необязательными. Внутри этих HTML элементов (между задающими их тегами) помещаются строки, относящиеся к соответствующим частям таблицы.

#### Пример 1 - Задание частей таблицы

```
<TABLE>
<THEAD>
<TR>Первая строка шапки...
<TR>Вторая строка шапки...
</THEAD>
<TBODY>
<TR>Строка данных...
...
</TBODY>
<TFOOT>
<TR>Строка футера...
</TFOOT>
</TABLE>
```

Данные таблицы задаются с использованием элементов **TH** и **TD**. Задавая различные значения атрибутам **align**, **valign**, **bgcolor** в открывающих тегах, рассматриваемых HTML элементов.

#### Пример 2 - Группировка строк таблицы

```
<TITLE>Группировка строк таблицы</TITLE>
<TABLE align = center border = 3 bordercolor =
black rules = groups>
  <CAPTION align = top><B>Доходы от продаж за второе
полугодие XXXX
года</B></CAPTION>
<THEAD>
<!--формирование первой строки шапки таблицы-->
```

```

<TR>
<TH rowspan = 2>Филиал\Период
<TH colspan = 3>3 квартал
<TH colspan = 3>4 квартал
<!--формирование второй строки шапки (названия месяцев)->
<TR>
<TH>Июль<TH>Август<TH>Сентябрь
<TH>Октябрь<TH>Ноябрь<TH>Декабрь
<TBODY align = right>
<!--далее следуют строки с данными (первая ячейка
каждой строки - название филиала->
<TR><TD align = left>Филиал 1<TD>123123<TD>323233
<TD>323453<TD>231423<TD>323212<TD>243673
<TR><TD align = left>Филиал 2<TD>223523<TD>225243
<TD>314423<TD>212445<TD>373812<TD>274673
<TR><TD align = left>Филиал 3<TD>183123<TD>186834
<TD>323453<TD>231423<TD>323212<TD>243673
<TR><TD align = left>Филиал 4<TD>125163<TD>334343
<TD>123553<TD>167423<TD>254412<TD>132367
<TBODY align = right>
<!--Строка с итоговыми данными->
<TR><TD align = left>Всего:<TD>654932<TD>1069653
<TD>1084882<TD>842714<TD>1274648<TD>894386
</TABLE>

```

Таблица, задаваемая в примере 2, выглядит так, как показано на рисунке 6.1.

Филиал\Период	3 квартал			4 квартал		
	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
Филиал 1	123123	323233	323453	231423	323212	243673
Филиал 2	223523	225243	314423	212445	373812	274673
Филиал 3	183123	186834	323453	231423	323212	243673
Филиал 4	125163	334343	123553	167423	254412	132367
Всего:	654932	1069653	1084882	842714	1274648	894386

Рисунок 6.1 - Таблица со сгруппированными строками

Чтобы подчеркнуть отделение частей таблицы друг от друга, задано отображение границ только между группами строк и столбцов (см. атрибут **rules** элемента **TABLE**). В приведенной таблице сгруппированы лишь строки, поэтому и отображаются только горизонтальные границы.

### Описание и группировка столбцов

Помимо группировки строк таблицы, в HTML присутствуют элементы, позволяющие определять столбцы и формировать группы столбцов: HTML элементы **COL** и **COLGROUP**. Если элементы **COL** или **COLGROUP** используются, то они должны следовать до элементов, группирующих строки таблицы (или до первой строки данных при отсутствии явной группировки строк). Сначала будет рассмотрено использование элемента **COL**.

Итак, HTML элемент **COL** задается одиночным тегом **<COL>**. Этот элемент позволяет установить общие параметры отображения всех ячеек, входящих в столбец или столбцы, заданием следующих атрибутов:

- **align** – задает горизонтальное выравнивание текста ячеек столбца (столбцов), может принимать значения *left*, *right*, *center* или *justify*;
- **valign** – задает вертикальное выравнивание текста ячеек столбца (столбцов), может принимать значения *top*, *bottom*, *middle* или *baseline*;
- **bgcolor** – задает цвет фона ячеек столбца (столбцов);
- **width** – позволяет указать ширину столбца (столбцов);
- **span** – задает количество столбцов, к которым применяются параметры, заданные в описанных выше атрибутах (по умолчанию имеет значение 1).

Использование элемента **COL** не позволяет создавать группы столбцов – для этого используется HTML элемент **COLGROUP**. Однако использование элемента **COL** значительно облегчает настройку внешнего вида таблицы, позволяя задавать одинаковые настройки для нескольких столбцов одновременно. Например, чтобы создать таблицу, показанную на рис. 6.2, пришлось бы задавать значения атрибутов **bgcolor** почти для всех ячеек таблицы.

Филиал\Период	3 квартал			4 квартал		
	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
Филиал 1	123123	323233	323453	231423	323212	243673
Филиал 2	223523	225243	314423	212445	373812	274673
Филиал 3	183123	186834	323453	231423	323212	243673
Филиал 4	125163	334343	123553	167423	254412	132367
Всего:	654932	1069653	1084882	842714	1274648	894386

Рисунок 6.2 - Раскрашенная таблица

При использовании элемента COL все гораздо проще (пример 3).

### Пример 3 - Задание параметров отображения столбцов

```

<TITLE>Использование элемента COL</TITLE>
<TABLE align = center border = 3 bordercolor =
black rules = groups>
  <CAPTION align = top><B>Доходы от продаж за второе
полугодие XXXX
года</B></CAPTION>
  <!--Определение столбцов таблицы-->
  <COL align = left bgcolor = green>
  <COL span = 3 bgcolor = blue>
  <COL span = 3 bgcolor = yellow>
  <THEAD>
    <!--формирование первой строки шапки таблицы-->
    <TR bgcolor = magenta>
      <TH rowspan = 2>Филиал\Период
      <TH colspan = 3>3 квартал
      <TH colspan = 3>4 квартал
    <!--формирование второй строки шапки (названия ме-
сяцев)-->
    <TR bgcolor = magenta>
      <TH>Июль<TH>Август<TH>Сентябрь
      <TH>Октябрь<TH>Ноябрь<TH>Декабрь
    <TBODY align = right>
      <!--далее следуют строки с данными (первая ячейка
каждой строки - название филиала-->
      <TR><TD>Филиал 1<TD>123123<TD>323233<TD>323453
      <TD>231423<TD>323212<TD>243673
      <TR><TD>Филиал 2<TD>223523<TD>225243<TD>314423
      <TD>212445<TD>373812<TD>274673
      <TR><TD>Филиал 3<TD>183123<TD>186834<TD>323453

```

```

<TD>231423<TD>323212<TD>243673
<TR><TD>Филиал 4<TD>125163<TD>334343<TD>123553
<TD>167423<TD>254412<TD>132367
<TBODY align = right>
<!--Строка с итоговыми данными-->
<TR bgcolor = red><TD>Всего:<TD>654932<TD>1069653
<TD>1084882<TD>842714<TD>1274648<TD>894386
</TABLE>

```

В приведенном примере, кроме задания цвета столбцов, указание значения `left` атрибута `align` для первого столбца избавило от необходимости задавать для первой ячейки каждой строки выравнивание текста по левому краю, как это было в предыдущих примерах.

Обратите внимание, что в примере 3 также использовалось задание цвета двух первых и последней строки при помощи атрибута `bgcolor` элемента `TR`. В этом и подобных случаях при отображении каждой ячейки браузеры выбирают параметры в следующем порядке.

1. Атрибуты элементов `TD` и `TH`.
2. Атрибуты элемента `TR`.
3. Атрибуты элемента `COL`.
4. Атрибуты элемента `COLGROUP`.
5. Атрибуты элементов `THEAD`, `TFOOT`, `TBODY`.
6. Атрибуты элемента `TABLE`.

Приведенный список отражает приоритет, с которым используются значения каждого атрибута HTML элементов, описывающих данные таблицы. Проще всего это пояснить на примере 4.

#### Пример 4 -Приоритеты элементов при отображении ячеек

```

<TABLE width = 300>
<THEAD align = center bgcolor = yellow>
<TR align = right>
<TD>Ячейка 1
<TD bgcolor = green>Ячейка 2
</THEAD>
</TABLE>

```

В данном случае ячейка с текстом **Ячейка 1** будет иметь желтый фон, выравнивание по правому краю, ширину 300 пикселей. Ячейка с текстом **Ячейка 2** будет отображаться аналогично первой, но с зеленым фоном.

Следует сказать несколько слов о возможностях задания значения атрибута **width**. Итак, значением атрибута может быть либо абсолютная ширина ячеек в пикселах, либо доля от ширины таблицы (в процентах), либо относительный или пропорциональный размер ячеек. Для задания пропорционального размера используется запись вида `width = "i*"`, где `i` является целым положительным числом ("`*`" интерпретируется как "`1*`"). Рассмотрим, каким образом по пропорциональному размеру определяется абсолютный размер. Пусть есть столбцы, заданные в следующем виде:

...

```
<COL width = "2*">
<COL width = "3*">
<COL width = "30%">
...
```

Кроме того, пусть таблица имеет ширину 100 пикселей. Сначала вычисляются процентные размеры, а потом пропорциональные, поэтому третий столбец таблицы будет иметь ширину 30. Оставшиеся 70 пикселей распределяются между первым и вторым столбцами в соотношении 2:3, то есть ширина этих столбцов получится равной  $70 : 5 \cdot 2 = 28$  и  $70 : 5 \cdot 3 = 42$  соответственно.

Теперь, после достаточно долгого изучения HTML элемента COL, рассмотрим, как можно создавать группы столбцов таблицы с использованием элемента COLGROUP. Элемент COLGROUP задается парными тегам **<COLGROUP>** и **</COLGROUP>** (закрывающий тег необязателен). Он поддерживает тот же набор атрибутов, что и элемент COL.

Элемент COLGROUP может одновременно с группировкой задавать одинаковые параметры отображения всех столбцов группы аналогично тому, как это делается с использованием элемента COL. Однако если требуется задать различные значения атрибутов для разных столбцов группы, то нужно включить внутрь элемента COLGROUP описания всех столбцов таблицы с помощью элементов COL.

Например, если нужно создать группу из десяти столбцов, имеющих одинаковую ширину 30, а также одинаковые остальные параметры, то это можно сделать так:

```
<COLGROUP span = 10 width = 30>
или
<COLGROUP>
<COL span = 10 width = 30>
</COLGROUP>
или
<COLGROUP>
<COL width = 30>
<COL width = 30>
...
</COLGROUP>
```

В первом случае использовалась возможность задания параметров всех столбцов группы через установление соответствующих значений атрибутов элемента COLGROUP. Во втором случае внутри группы были явно определены десять столбцов. Использование третьего варианта в данном примере является нерациональным, хотя и допустимым.

Без явного определения столбцов не обойтись в том случае, если нужно включить в группу столбцы с различными параметрами отображения (например, с различным выравниванием). Допустим, нужно создать группу из десяти столбцов: выравнивание первого столбца левое, со второго по

восьмой – правое, девятого и десятого – по центру. Группу столбцов наиболее кратко можно задать следующим образом:

```
<COLGROUP>
<COL align = left>
<COL span = 7 align = right>
<COL span = 2 align = center>
</COLGROUP>
```

Рассмотрим, как отразится наличие групп столбцов на отображении таблицы браузером. Можно дополнить таблицу из примера 1 группировкой столбцов следующим образом (оставлены только части текста HTML документа, отличные от приведенного в примере 1)

### **Пример 5 - Группировка столбцов**

```
<TITLE>Группировка строк и столбцов табли-
цы</TITLE>
<TABLE align = center border = 3 bordercolor =
black rules = groups>
  <CAPTION align = top><B>Доходы от продаж за второе
полугодие XXXX
года</B></CAPTION>
  <COLGROUP align = left>
  <COLGROUP span = 3>
  <COLGROUP span = 3>
  <THEAD>
  ...
  <TBODY align = right>
  ...
  <TR><TD>Филиал 1...
  <TR><TD>Филиал 2...
  <TR><TD>Филиал 3...
  <TR><TD>Филиал 4...
  <TBODY align = right>
  <TR><TD>Всего:...
</TABLE>
```

Теперь таблица примет окончательный вид, показанный на рисунке

6.3.

**Доходы от продаж за второе полугодие XXXX года**

Филиал/Период	3 квартал			4 квартал		
	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
Филиал 1	123123	323233	323453	231423	323212	243673
Филиал 2	223523	225243	314423	212445	373812	274673
Филиал 3	183123	186834	323453	231423	323212	243673
Филиал 4	125163	334343	123553	167423	254412	132367
Всего:	654932	1069653	1084882	842714	1274648	894386

Рисунок 6.3 - Таблица со сгруппированными столбцами и строками

### Особенности задания ширины столбцов

В начале было сказано, что структурирование может позволить браузеру не дожидаться загрузки всей таблицы, а отображать ее частями по мере получения данных. Теперь пришло время уточнить, какие параметры и как должны быть заданы для того, чтобы браузер мог начать рисовать таблицу по частям.

Дело в том, что самой трудной задачей при отображении таблицы является определение ширины как таблицы в целом, так и каждого столбца таблицы, а также определение количества столбцов в ней. Если ширина и количество столбцов не указаны явно, то браузер вынужден дожидаться загрузки всей таблицы, после чего определяется количество ячеек в самой длинной строке – количество столбцов. Ширина каждого столбца выбирается такой, чтобы поместить содержимое самой широкой ячейки этого столбца.

Чтобы браузер мог начать отображение таблицы по частям, он должен до получения первой строки с данными ячеек обладать сведениями о количестве и абсолютной (в пикселах) ширине каждого столбца, о созданных группах столбцов. Для этого нужно определить все столбцы с использованием COL или COLGROUP, а также указать ширину таблицы и ее столбцов так, чтобы абсолютные значения могли быть однозначно определены, например:

```
<TABLE width = 300>
<COLGROUP>
<COL width = 30>
<COL width = "*">
</COLGROUP>
<COLGROUP width = 50 span = 4>
```



```
...  
</TABLE>
```

В этом случае точно известно, что таблица, содержащая два столбца, будет шириной 300 пикселей. На первый столбец приходится 30 пикселей и, соответственно, на второй – оставшиеся 270.

Можно также задавать для таблицы и ее столбцов процентную ширину, но это нужно делать внимательно, чтобы, зная размер окна браузера, можно было всегда однозначно определить точную ширину каждого столбца.

Пример усовершенствованной таблицы:

В начале идет стандартная структура таблицы:

```
<table class="table1">  
  <thead>  
    <tr>  
      <th></th>  
      <th scope="col" abbr="Starter">Smart  
Starter</th>  
      <th scope="col" abbr="Medium">Smart Medi-  
um</th>  
      <th scope="col" abbr="Business">Smart  
Business</th>  
      <th scope="col" abbr="Deluxe">Smart  
Deluxe</th>  
    </tr>  
  </thead>  
  <tfoot>  
    <tr>  
      <th scope="row">Price per month</th>  
      <td>$ 2.90</td>  
      <td>$ 5.90</td>  
      <td>$ 9.90</td>  
      <td>$ 14.90</td>  
    </tr>  
  </tfoot>  
  <tbody>  
    <tr>  
      <th scope="row">Storage Space</th>  
      <td>512 MB</td>  
      <td>1 GB</td>  
      <td>2 GB</td>  
      <td>4 GB</td>  
    </tr>  
    <tr>  
      <th scope="row">Bandwidth</th>  
      <td>50 GB</td>
```

```

        <td>100 GB</td>
        <td>150 GB</td>
        <td>Unlimited</td>
    </tr>
    <tr>
        <th scope="row">MySQL Databases</th>
        <td>Unlimited</td>
        <td>Unlimited</td>
        <td>Unlimited</td>
        <td>Unlimited</td>
    </tr>
    <tr>
        <th scope="row">Setup</th>
        <td>19.90 $</td>
        <td>12.90 $</td>
        <td>free</td>
        <td>free</td>
    </tr>
    <tr>
        <th scope="row">PHP 5</th>
        <td><span class="check"></span></td>
        <td><span class="check"></span></td>
        <td><span class="check"></span></td>
        <td><span class="check"></span></td>
    </tr>
    <tr>
        <th scope="row">Ruby on Rails</th>
        <td><span class="check"></span></td>
        <td><span class="check"></span></td>
        <td><span class="check"></span></td>
        <td><span class="check"></span></td>
    </tr>
</tbody>
</table>

```

В примере есть все элементы, что используются в таблицах: заголовок (header), тело (body), и подвал (footer). В примере будем использовать данные тарифов одной хостинговой компании.

Таблица 6.1 - Данные тарифов одной хостинговой компании

	Smart Starter	Smart Medium	Smart Business	Smart Deluxe
Storage Space	512 MB	1 GB	2 GB	4 GB
Bandwidth	50 GB	100 GB	150 GB	Unlimited
MySQL Databases	Unlimited	Unlimited	Unlimited	Unlimited
Setup	19.90 \$	12.90 \$	free	free
PHP 5	✓	✓	✓	✓
Ruby on Rails	✓	✓	✓	✓
Price per month	<b>\$ 2.90</b>	<b>\$ 5.90</b>	<b>\$ 9.90</b>	<b>\$ 14.90</b>

Первая таблица будет в зеленых тонах с градиентом в заглавных ячейках (элементы “th”). Начнем с основного стиля таблицы:

```
table.table1{
  font-family: "Trebuchet MS", sans-serif;
  font-size: 16px;
  font-weight: bold;
  line-height: 1.4em;
  font-style: normal;
  border-collapse: separate;
}
```

Для того, чтобы было некоторое расстояние между ячейками, будем использовать свойство **border-collapse** со значением **separate**.

Можно использовать стили для шрифта с сайта [Typechart](http://Typechart). Это очень полезный ресурс с хорошими примерами шрифтов и готовым CSS кодом.

Элементы **th** в заголовке будут описаны следующим образом:

```
.table1 thead th{
  padding:15px;
  color:#fff;
  text-shadow:1px 1px 1px #568F23;
  border:1px solid #93CE37;
  border-bottom:3px solid #9ED929;
  background-color:#9DD929;
  background:-Webkit-gradient(
    linear,
    left bottom,
    left top,
    color-stop(0.02, rgb(123,192,67)),
    color-stop(0.51, rgb(139,198,66)),
```

```

        color-stop(0.87, rgb(158,217,41))
    );
background: -moz-linear-gradient(
    center bottom,
    rgb(123,192,67) 2%,
    rgb(139,198,66) 51%,
    rgb(158,217,41) 87%
);
-Webkit-border-top-left-radius:5px;
-Webkit-border-top-right-radius:5px;
-moz-border-radius:5px 5px 0px 0px;
border-top-left-radius:5px;
border-top-right-radius:5px;
}

```

В примере используем свойство градиента для ров Firefox и Webkit (Safari и Chrome), чтобы создать красивый градиент из трех цветов. Свойство `border-radius` скругляет верхний левый и верхний правый углы ячеек.

Теперь нам нужно позаботиться об элементе `th`, являющимся пустым. С помощью селекторов CSS3 можно делать невероятные вещи, и одна из них - это выбор элементов, являющихся пустыми. Делается следующим образом:

```

.table1 thead th:empty{
    background:transparent;
    border:none;
}

```

Подвал таблицы оформляется так:

```

.table1 tfoot td{
    color: #9CD009;
    font-size:32px;
    text-align:center;
    padding:10px 0px;
    text-shadow:1px 1px 1px #444;
}
.table1 tfoot th{
    color:#666;
}

```

Усилим отображение контента с помощью свойства **text shadow**. Внутренние ячейки таблицы оформлены следующим образом: светло-зеленый фон и белая тень текста для получения эффекта гравировки:

```

.table1 tbody td{
    padding:10px;
    text-align:center;
    background-color:#DEF3CA;
    border: 2px solid #E7EFE0;
}

```

```

-moz-border-radius:2px;
-Webkit-border-radius:2px;
border-radius:2px;
color:#676;
text-shadow:1px 1px 1px #fff;
}

```

Добавим очень тонкий бордюр и скругления для ячеек. Получим небольшой светящийся эффект. Можно использовать свойство **box shadow** для получения подобного эффекта.

Теперь, нужно добавить иконку для всех ячеек, у которых есть элемент `span` с классом `check`. С помощью следующего CSS кода получим:

```

.table1 tbody span.check::before{
    content : url(../images/check0.png)
}

```

Это свойство позволит добавить контент (в нашем случае это изображение) внутрь элемента. Также можно добавить текст. Можно использовать свойства **::before** или **::after**, которые и будут определять, куда вставлять контент.

### Задание на лабораторную работу:

Пошагово изучить форматирование таблиц на примерах. Закрепить материал, выполнив контрольные примеры.

Кроме рассмотренных ранее возможностей по манипулированию таблицами, HTML поддерживает разбиение таблицы на логически цельные части: группы строк и столбцов. Для отдельных частей таблицы можно устанавливать общие параметры отображения данных, которые автоматически применяются браузером при отображении таблицы.

## КОНТРОЛЬНЫЕ ПРИМЕРЫ

Таблица №1

	Smart Starter	Smart Medium	Smart Business	Smart Deluxe
Storage Space	512 MB	1 GB	2 GB	4 GB
Bandwidth	50 GB	100 GB	150 GB	UNLIMITED
MySQL Databases	UNLIMITED	UNLIMITED	UNLIMITED	UNLIMITED
Setup	19.90 \$	12.90 \$	FREE	FREE
PHP 5	✓	✓	✓	✓
Ruby on Rails	✓	✓	✓	✓
PRICE PER MONTH	\$ 2.90	\$ 5.90	\$ 9.90	\$ 14.90

Таблица №2

Driver	Sebastian Vettel	Jenson Button	Fernando Alonso	Mark Webber
Nationality	German	British	Spanish	Australian
Team	RBR-Renault	McLaren-Mercedes	Ferrari	RBR-Renault
Points	374	240	227	221

Таблица №3

	Q1	Q2	Q3	Q4
Microsoft	20.3	30.5	23.5	40.3
Google	50.2	40.63	45.23	39.3
Apple	25.4	30.2	33.3	36.7
IBM	20.4	15.6	22.3	29.3

Таблица №4

Имя	Фамилия	Email	Дата	Сумма
Александр	Петров	petrov@mail.ru	2006-11-07	400.00
Сергей	Петров	petrov.s@mail.ru	2007-07-25	500.00
Константин	Петров	petrov@yandex.ru	2006-04-23	600.00
Елена	Синцова	elena@mail.ru	2007-02-15	400.00
Ольга	Синцова	olga@yandex.ru	2007-08-02	600.00

Таблица №5

Table caption

Header	Header	Header	Header	Header	Header
sample data	sample data	sample data	sample data	sample data	sample data
sample data	sample data	sample data	sample data	sample data	sample data
sample data	sample data	sample data	sample data	sample data	sample data
sample data	sample data	sample data	sample data	sample data	sample data
sample data	sample data	sample data	sample data	sample data	sample data
footer	footer	footer	footer	footer	footer

Таблица №6

Travel Expense Report				
	Meals	Hotels	Transport	subtotals
<b>San Jose</b>				
25-Aug-97	37.74	112.00	45.00	
26-Aug-97	27.28	112.00	45.00	
subtotals	65.02	224.00	90.00	379.02
<b>Seattle</b>				
27-Aug-97	96.25	109.00	36.00	
28-Aug-97	35.00	109.00	36.00	
subtotals	131.25	218.00	72.00	421.25
<b>Totals</b>	196.27	442.00	162.00	800.27

Таблица №7

	SMART STARTER	SMART MEDIUM	SMART BUSINESS	SMART DELUXE
STORAGE SPACE	512 MB	1 GB	2 GB	4 GB
BANDWIDTH	50 GB	100 GB	150 GB	UNLIMITED
MYSQL DATABASES	UNLIMITED	UNLIMITED	UNLIMITED	UNLIMITED
SETUP	19.90 \$	12.90 \$	FREE	FREE
PHP 5				
RUBY ON RAILS				
PRICE PER MONTH	\$ 2.90	\$ 5.90	\$ 9.90	\$ 14.90

Таблица №8

	Starter Pack	Power pack	Ultimate pack
Slots	10 slots	20 slots	50 slots
Memory	512MB (RAM)	1024MB (RAM)	3072MB (RAM)
Bandwidth	50GB	50GB	100GB
Control Panel	Yes	Yes	Yes
FTP Access	Yes	Yes	Yes
Price	\$4.90 / monthly	\$12.25 / monthly	\$31.85 / monthly
Order now!	<a href="#">Order now!</a>	<a href="#">Order now!</a>	<a href="#">Order now!</a>

Таблица №9

Company	Q1	Q2	Q3	Q4
Microsoft	20.3	30.5	23.5	40.3
Google	50.2	40.63	45.23	39.3
Apple	25.4	30.2	33.3	36.7
IBM	20.4	15.6	22.3	29.3



Таблица №10

Table caption

Header	Header	Header	Header	Header	Header
sample data	sample data	sample data	sample data	sample data	sample data
sample data	sample data	sample data	sample data	sample data	sample data
sample data	sample data	sample data	sample data	sample data	sample data
sample data	sample data	sample data	sample data	sample data	sample data
sample data	sample data	sample data	sample data	sample data	sample data
footer	footer	footer	footer	footer	footer

Таблица №11

Plans&Pricing		
Plan	Price	
Basic	10\$	Select
Professional	20\$	Select
Premium	30\$	Select
Platinum	60\$	Select

Таблица №12

Free	Pro	Business <small>Best value!</small>	Dev
<b>\$Free</b> for free	<b>\$11</b> single	<b>\$35</b> webmaster	<b>\$75</b> devs
1 Domain	1 Domain	5 Domain	60 Domain
1 GB Storage	1 GB Storage	5 GB Storage	60 GB Storage
1 FTP Account	2 FTP Account	3 FTP Account	5 FTP Account
[n] Add-on domain	[y] Add-on domain	[y] Add-on domain	[y] Add-on domain
Sign Up	Buy	Buy	Buy

Таблица №13

	Product Name	Price
1	lorem	\$1.00
2	lorem	\$5.00
3	lorem	\$4.00
4	lorem	\$20.00
5	lorem	\$10.00
6	lorem	\$5.00
7	lorem	\$5.00
Summary		\$50.00

Таблица №14

	starter	econo	standard	professional
choose your plan	\$10 per month	\$30 per month	\$59 per month	\$99 per month
Amount of space	10GB	30GB	100GB	Unlimited
Bandwidth per month	100GB	200GB	500GB	1000GB
No. of e-mail accounts	1	10	50	Unlimited
No. of MySql databases	1	10	50	Unlimited
24h support	Yes	Yes	Yes	Yes
Support tickets per mo.	1	3	5	10
<div>sign up!</div> <div>sign up!</div> <div>sign up!</div> <div>sign up!</div>				

Таблица №15

Name		
Fore	Surname	Price
Neil	Crosby	\$1.96
Becca	Courtley	\$23.95
David	Freidman-Jones	\$14.00
Annabel	Tyler	\$104.00
Carl	Conway	\$17.00
		\$160.91

Таблица №16

Beautiful design tables in HTML in the style of a zebra.

Date	Start time	End time	Name
02.06.2010	10:00	12:00	Cleaning
02.06.2010	12:00	15:00	Training
02.06.2010	15:00	17:00	Rest
02.06.2010	17:00	21:00	Work
02.06.2010	21:00	07:00	Sleep

Таблица №17

## Responsive Table

First Name	Last Name	Score
John	Cena	121
The	Rock	112
Brave	Heart	86
Martin	Jerry	48

Таблица №18

Date	Start time	End time	Name
02.06.2010	10:00	12:00	Cleaning
02.06.2010	12:00	15:00	Training
02.06.2010	15:00	17:00	Rest
02.06.2010	17:00	21:00	Work
02.06.2010	21:00	07:00	Sleep

Таблица №19

	Name			
Date	Fore	Surname	Price	IP Address
21/01/2006	Neil	Crosby	\$1.96	192.168.1.1
01/02/2006	Becca	Courtley	\$23.95	192.167.2.1
17/11/2004	David	Freidman-Jones	\$14.00	192.168.2.1
17/10/2004	Annabel	Tyler	\$104.00	192.168.2.17
17/11/2005	Carl	Conway	\$17.00	192.168.02.13
			<b>\$160.91</b>	

Таблица №20

Номер	Описание эффекта	Номер	Описание эффекта
0	Прямоугольники внутрь	12	Растворение
1	Прямоугольники наружу	13	Вертикальная панорама внутрь
2	Круг внутрь	14	Вертикальная панорама наружу
3	Круг наружу	15	Горизонтальная панорама внутрь
4	Наплыв наверх	16	Горизонтальная панорама наружу
5	Наплыв вниз	17	Уголки влево - вниз
6	Наплыв вправо	18	Уголки влево - вверх
7	Наплыв влево	19	Уголки вправо – вниз
8	Вертикальные жалюзи	20	Уголки вправо – вверх
9	Горизонтальные жалюзи	21	Случайные горизонтальные полосы
10	Шажки горизонтальные	22	Случайные вертикальные полосы
11	Шажки вертикальные	23	Случайный выбор эффекта

Таблица №21

	Name			
Date	Fore	Surname	Price	IP Address
21/01/2006	Neil	Crosby	\$1.96	192.168.1.1
01/02/2006	Becca	Courtley	\$23.95	192.167.2.1
17/11/2004	David	Freidman-Jones	\$14.00	192.168.2.1
17/10/2004	Annabel	Tyler	\$104.00	192.168.2.17
17/11/2005	Carl	Conway	\$17.00	192.168.02.13
			<b>\$160.91</b>	

Таблица №22

Main mode	Area of workplace					
	Central London	Rest of Inner London	Outer London	All London	Rest of Great Britain	Great Britain
Car and van	48	32	25	29	20	20
Motorbike, moped, scooter	36	29	27	31	19	21
Bicycle	33	24	20	25	15	17
Bus and coach	47	39	36	40	33	34
National Rail	69	66	43	66	47	58
Underground, tram, light rail	49	45	37	47	42	46
Walk	21	16	13	15	12	13
All modes	55	39	27	39	20	23

Таблица №23

Recent Major Volcanic Eruptions in the Pacific Northwest			
Volcano Name	Location	Last Major Eruption	Type of Eruption
<b>Mt. Lassen</b>	California	1914-17	Explosive Eruption
<b>Mt. Hood</b>	Oregon	1790s	Pyroclastic flows and Mudflows
<b>Mt. St. Helens</b>	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Таблица №24

Recent Major Volcanic Eruptions in the Pacific Northwest			
Volcano Name	Location	Last Major Eruption	Type of Eruption
<b>Mt. Lassen</b>	California	1914-17	Explosive Eruption
<b>Mt. Hood</b>	Oregon	1790s	Pyroclastic flows and Mudflows
<b>Mt. St. Helens</b>	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Таблица №25

Position	Movie	Year of Release
1	Citizen Kane	1941
2	The Godfather	1972
3	Casablanca	1942
4	Raging Bull	1980
5	Singin' In The Rain	1952

Таблица №26

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
<b>Aerobics</b>	09.00 - 10.00	-	11.00 - 12.00	-	12.00 - 13.00	-
<b>Aqua</b>	-	10.00 - 11.00	-	12.00 - 13.00	-	13.00 - 14.00
<b>Boxing</b>	10.00 - 11.00	-	11.00 - 12.00	-	12.00 - 13.00	13.00 - 14.00
<b>Pilates</b>	09.00 - 10.00	-	-	12.00 - 13.00	-	13.00 - 14.00
<b>Running</b>	-	10.00 - 11.00	-	12.00 - 13.00	-	13.00 - 14.00
<b>Aerobics</b>	09.00 - 10.00		11.00 - 12.00		12.00 - 13.00	













Таблица №26

Font format	Opera 10	Mozilla Firefox 3.5	Internet Explorer 8	Safari 3.1	Google Chrome 4
<b>TTF, OTF</b>	✓	✓	✗	✓	✓
<b>EOT</b>	✗	✗	✓	✗	✗
<b>SVG</b>	✓	✗	✗	✓	✓
<b>WOFF</b>	✗	✓(v3.6)	✗	✗	✗

Таблица №27

Sales Order				
Your Name				
Your Address				
City, ST 55555				
No.	Description	Quantity	Price	Amount
11111	Microsoft Windows XP Professional	2	169.99	339.98
22222	Microsoft Office XP Professional	2	449.99	999.98
33333	Adobe Photoshop 7.0	1	579.95	579.95
44444	HP PhotoSmart 7550 Printer	1	299.99	299.99
55555	USB Device Cable	1	5.49	5.49
Shipping				50.00
Sales Tax				155.77
Total				\$ 2,431.16

Таблица №28

NO	NAME	EMAIL	OPTION	
1	Ken Arok	ken_arok@mail.com	 Edit	 Delete
2	Ken Dedes	ken_dedes@mail.com	 Edit	 Delete
3	Tunggul Ametung	tunggul_ametung@mail.com	 Edit	 Delete
4	Mpu Gandring	mpu_gandring@mail.com	 Edit	 Delete
5	Lohgawe	lohgawe@mail.com	 Edit	 Delete
6	Kertajaya	kertajaya@mail.com	 Edit	 Delete

### Содержимое отчета:

1. Задание.
2. Текст HTML-страницы.
3. Структура страниц.

### Контрольные вопросы:

1. Из каких частей состоит таблица? Какими тегами они обозначаются?
2. С помощью каких элементов задаются данные таблицы?
3. Как сгруппировать строки таблицы? Приведите пример.
4. Как сгруппировать столбцы таблицы? Приведите пример.
4. Как задать ширину столбцов?
5. Как задать расстояние между ячейками таблицы?



## Лабораторная работа № 7

### Верстка макета сайта

**Тема лабораторной работы:** верстка макета сайта по заданному шаблону с использованием HTML5 и CSS3.

**Цель работы:** изучить возможности CSS для верстки Web-страниц.

### Резиновый дизайн сайта

Резиновый дизайн сайта – это такой дизайн, при котором ширина колонок задаётся в процентах или сочетаются проценты и пикселы таким образом, что макет занимает всю свободную ширину окна браузера. При изменении размеров окна или другом разрешении монитора макет подстраивается под них.

### Резиновый двухколоночный макет

Двухколоночный резиновый макет позволяет эффективно использовать площадь браузера, и вместе с тем достаточно удобен для самого широкого круга задач. Кроме того, такой макет не требует сложной работы над собой и его можно использовать на многих сайтах, комбинируя ширину колонок в пикселах и процентах. Существует несколько подходов к формированию такого макета, но самый простой и удобный заключается в сочетании свойств **float** и **margin**.

Рассмотрим вариант, когда левая колонка имеет определенный размер, а ширина правой колонки устанавливается автоматически, исходя из ширины окна браузера. При этом ширина левой колонки может задаваться в пикселах или процентах. В таблице 7.1 приведены основные стилевые свойства для формирования двух колонок.

Табл. 7.1 - Левая колонка заданной ширины

Для левого слоя шириной 20%	
Левая колонка	Правая колонка
float: left width: 20%	margin-left: 21%
Для левого слоя шириной 200px	
float: left width: 200px	margin-left: 210px

Для левой колонки требуется всего два свойства: **float** — заставляет вторую колонку располагаться рядом по горизонтали с первой и **width**, которое устанавливает ширину колонки. Вторая колонка будет занимать всё оставшееся место, поэтому для нее указывать width не нужно.

Правая колонка характеризуется лишь одним свойством — **margin-left**, оно смещает левый край колонки на ширину левого слоя, плюс задает отступ между колонками. Поэтому величина этого свойства в таблице 7.1 указана 21%, где 20% сама ширина колонки, а на один процент приходится



расстояние между колонками. В случае задания ширины одной из колонок в пикселах, код останется прежним, но поменяются единицы измерения.

Из-за того, что мы имеем дело с резиновым макетом, который может занимать всю ширину окна браузера и уменьшаться до какого-то предела, имеет смысл сделать ограничения. Свойство **min-width** для слоя **layout** устанавливает минимальную ширину, если окно браузера будет меньше этого значения, появится горизонтальная полоса прокрутки. Свойство **max-width**, наоборот, задаёт максимальную ширину, которая не будет превышена.

### Пример 1 - Навигация слева

```
<head>
  <meta      http-equiv="Content-Type"      con-
tent="text/html; charset=utf-8" />
  <title>Две колонки</title>
  <style type="text/css">
    .header, .sidebar, .content, .footer {
      padding: 1 0px; /* Поля */
      border: solid 1px #000; /* Параметры рамки */
      background: #e3e8cc; /* Цвет фона */
    }
    .header { /* Верхняя часть с заголовком */
      background: #e3e8cc; /* Цвет фона */
      font-size: 24px; /* Размер шрифта */
    }
    .layout {
      margin: 15px 0; /* Отступы сверху и снизу */
      overflow: hidden; /* Отменяем действие float
*/
      min-width: 800px; /* Минимальная ширина */
      max-width: 1200px; /* Максимальная ширина */
    }
    .sidebar { /* Навигация по сайту */
      margin: 5px 0; /* Отступы сверху и снизу */

      width: 100 px; /* Ширина меню */
      float: left; /* Состыковка с другим слоем по
горизонтали */
    }
    .sidebar ul {
      list-style: none; /* Убираем маркеры */
      padding: 0; /* Убираем отступы */
    }
    .content { /* Основное содержание страницы */
      margin-left: 135px; /* Отступ слева */
    }
  }
```

```

</style>
</head>
<body>
  <div class="header">Заголовок сайта</div>
  <div class="layout">
    <div class="sidebar">
      <h2>Меню</h2>
      <ul>
        <li><a href="link1.html"> 1</a></li>
        <li><a href="link2.html"> 2</a></li>
        <li><a href="link3.html"> 3</a></li>
        <li><a href="link4.html"> 4</a></li>
      </ul>
    </div>
    <div class="content">
      <h1>Название страницы</h1>
      <p>Текст.</p>
    </div>
  </div>
  <div class="footer">Подвал</div>
</body>
</html>

```

## Адаптивный макет

Этот макет подстраивается под разрешение монитора и окна браузера, меняя при необходимости ширину макета, число колонок, размеры изображений и текста. Для этого заготавливается несколько стилевых правил или файлов под разный диапазон разрешений, выбор правил происходит через скрипты или CSS3, которые и определяют нужную для этого информацию о пользователе.

Особенность верстки эластичного шаблона в том, что все величины должны быть указаны не в пикселях (px) и не в процентах, а в em. По умолчанию 1 em равен величине шрифта, который мы укажем у элемента body. Если величина шрифта не задана, то большинство браузеров автоматически устанавливают размер 16px, тогда и 1em = 16px.

### Пример "эластичной" верстки:

Нам нужно прописать правила для элементов H1, P, IMG и блока wrap. Для последнего установим ширину в 600px, предварительно переведя в em:

$600\text{px}/10 = 60\text{em}$ .

```

#wrap{
width: 60em;
margin: 1.5em auto; /* 15px/10 =1.5em*/
border: 0.1em solid #ccc; /* 1px/10 =0.1em*/

```

```
background: #fff;
text-align: left;
}
```

Для заголовка назначим размер шрифта, эквивалентный 16px, а для абзацев 12px

```
h1{
font-size: 1.6em; /* 16px/10 =1.6em*/
margin: 0.833em; /* 10px/16 =0.833em*/
font-weight: 300;
}

p{
font-size: 1.2em;
line-height: 1.333em; /* 16px/12 =1.33em*/
margin-bottom: 1.25em; /* 15px/12 =1.25em*/
}
```

Не забудем также, что габариты картинок теперь тоже надо указывать в em. Но это не проблема, когда формула преобразований так проста. Присвоим элементу **img** следующие правила:

```
img {
width: 8.333em; /* 100px/12 =8.33em 12 - у блока
p*/
height: 8.333em;
margin:0 0.833em 0.833em 0; /* 10px/12 =0.833em*/
}
```

### Задание на лабораторную работу:

1. Сверстайте сайт по созданному макету. Для первой страницы приведите список ключевых слов, укажите автора, используемый стандарт, язык Web-сайта, используемую кодировку.

2. В структуре страницы обязательно использовать параграфы, списки, заголовки.

3. Для создания структуры сайта необходимо использовать новые элементы HTML5: header, footer, nav, article и т.п.

4. На основе структуры первой страницы определите перечень используемых стилей для позиционирования элементов Web-сайта в соответствии с выданным заданием.

5. При разработке стилей позиционирования рекомендуется использовать возможности обтекания элементов и относительного задания их размеров (т.н. «резиновый» дизайн). Использование абсолютного и фиксированного позиционирования разрешено в крайних случаях и должно быть обосновано.

6. Выделите блочные элементы, которые позволяют создать нужный интерфейс сайта.

Разделите стили на несколько файлов:

- создание структуры сайта
- форматирование текста
- форматирование изображений
- форматирование списков и меню.

7. Приведите текст всех CSS-файлов.

### **Выкладывание на хостинг**

1. Создайте структуру каталогов для разработанного сайта. Зарегистрируйтесь на сайте ucoz.ua (или любом другом бесплатном хостинге). Имя сайта должно соответствовать шаблону: Имя\_сайта-ius12.ucoz.ua. Например: anastas-ius12.ucoz.ua.

2. В этом шаблоне «имя\_сайта» – заменяем на свою фамилию, можно добавить инициалы. Для сдачи лабораторной работы №7 выложите сверстаный макет на зарегистрированный хостинг. Сообщите информацию о зарегистрированном сайте по почте в виде: ФИО, Группа, сайт.

3. Разработанный шаблон должен пройти валидацию на ресурсе [validator.w3.org](http://validator.w3.org).

5. Для проверки правильности HTML-кода Web-страницы, еще не помещенной на сервер, щелкните на гиперссылке **ValidatebyUploadFiles**. В окне Web-браузера появится страница с формой загрузки файла. Нажмите на кнопке **Check**. Программа проверки проверит полученный HTML-документ и передаст Web-браузеру страницу результатов проверки.

### **Содержимое отчета:**

1. Задание.
2. Информация о зарегистрированном сайте.
3. Текст HTML-страницы.

### **Контрольные вопросы:**

1. Что значит «резиновый» дизайн сайта?
2. Какие вы знаете типы HTML-макетов сайта?
3. Что значит адаптивный макет?
4. В чем особенность верстки адаптивного макета?

## Лабораторная работа №8

Формы в HTML документах

**Тема лабораторной работы:** создание HTML-форм.

**Цель работы:** изучение основ работы с HTML - формами.

### Теоретический материал

Некоторые WWW браузеры позволяют пользователю, заполнив специальную форму, возвращающую полученное значение, выполнять некоторые действия на вашем WWW – сервере. Обычно пользователи "заполняют" форму, модифицируя управляющие элементы перед тем, как предоставить форму пользователя для обработки (например, на Web-сервер, на почтовый сервер и т.д.).

Когда форма интерпретируется WEB - браузером, создается специальные экранные элементы GUI, такие, как поля ввода, поля типа переключатель, поля типа флажок, выпадающие меню, списки, кнопки и т.д.

Пользователь заполняет форму и нажимает кнопку "Подтверждение" (SUBMIT - специальный тип кнопки, который задается при описании документа) и информация, введенная пользователем в форму, посылается HTTP-серверу для обработки и передаче другим программам, работающим под сервером, в соответствии с CGI (Common Gateway Interface) интерфейсом.

Любые элементы формы находятся в **теле формы**. Т.е. у любой формы есть каркас, а уже внутри него вставляются различные элементы формы, и пишется html код.

#### Пример 1 - Каркас html формы

```
<form name="forma zakaza" method="post" action="obrabotchik.php">
```

```
<!-- сюда вставляют различные элементы -->
</form>
```

#### Атрибуты:

**NAME** – определяет имя формы, уникальное для данного документа. Используется только, если в документе присутствует несколько форм.

**ACTION** – обязательный атрибут. Указывает путь к скрипту (или программе) сервера, обслуживающих данную форму.

**METHOD** – определяет способ отправки содержимого html-формы. Возможные значения GET (по умолчанию) и POST.

Метод **GET** используется для передачи различных переменных или очень коротких сообщений. Информация передается в явном виде через строку браузера, т.е. ее можно перехватить. Например, если вы видите в строке набора браузера нечто вроде `http://adress.com/lessons.php?rub=28` это значит, что передается значение переменной `rub` равное 28. В html-формах обычно не используется.

Метод **POST** создан специально для передачи текстовых сообщений. Почти всегда применяется в формах. Передает информацию в скрытом виде.

Когда вы описываете форму, каждый элемент ввода данных имеет тэг **<INPUT>**. Когда пользователь помещает данные в элемент формы, информация размещается в разделе **VALUE** данного элемента.

### Текстовое поле

Элемент **INPUT** - создает поле html-формы. Элемент не имеет конечного тега. Например:

```
<form      name="primer1"      method="post"      ac-
tion="obrabotchik.php">
```

Введите ФИО: <br>

```
<input type="text" name="fio" size="30"><br>
```

#### Основные атрибуты:

**TYPE** - определяет тип поля для ввода данных. По умолчанию – это "text".

**NAME** - определяет имя, используемое при передаче содержания данной html-формы на сервер. Если Вы собираетесь куда-нибудь передавать информацию из формы, то обязательно вводите имя.

**SIZE** - определяет размер поля в символах. По умолчанию имеет значение равное 24.

**MAXLENGTH** - определяет максимальное количество символов, которые можно ввести в текстовом поле. Оно может быть больше, чем количество символов, указанных в атрибуте **SIZE**. По умолчанию количество символов не ограничено.

**VALUE** - определяет, что будет по умолчанию написано в поле для ввода.

### Пароль

Элемент **INPUT**. Тип "password" указывает на то, чтоб информация отображалась при вводе звездочками.

#### Пример 2:

```
<form      name="primer2"      method="post"      ac-
tion="obrabotchik.php">
```

Введите пароль (максимум шесть символов) :<br>

```
<input type="password" name ="pass" maxlength="6">
</form>
```

### Радиокнопка

Данный тип элементов html формы называется радиопереключатель (переключает: либо одно значение, либо другое - два одновременно быть не может). Поэтому атрибут type имеет значение **type="radio"**.

Если вы хотите сделать, чтобы по умолчанию радиопереключатель был установлен на каком-либо варианте, то просто допишите атрибут-флаг **checked** (включен). В радиопереключателе обязательно должен присутствовать атрибут **value**.

#### Пример 3:

```

<form name="primer3" method="post" action="obrabotchik.php">
<p> Какой диск вы хотите получить?</p>
<p>
<input name="disc" type="radio" value="cd"
checked>
CD<br>
<input name="disc" type="radio" value="dvd">
DVD </p></form>

```

### Флажок

Данный тип элементов html-формы называется **checkbox** и отличается от радиопереключателя тем, что здесь можно выбрать несколько вариантов.

В данном элементе атрибут **value** не является обязательным, в отличие от радиопереключателя.

#### Пример 4:

```

<form name="primer4" method="post"
action="obrabotchik.php">
<p>Какие обучающие курсы вы хотите видеть на дис-
ке?</p>
<input type="checkbox" name="fotoshop" value="yes"
checked>
Курсы по Фотошопу <br>
<input type="checkbox" name="dreamweaver"
value="yes">
Курсы по Adobe Dreamweaver <br>
<input type="checkbox" name="php"
value="yes">Курсы по PHP
</form>

```

### Выпадающий список

Базовым элементом здесь является **SELECT** (выбор). У него обязательно должен быть закрывающий тег. Внутри него содержатся элементы **OPTION** (опции выбора).

У элемента **SELECT** есть атрибут **name**, а у элементов **OPTION** - **value** (в элементе **select** это не обязательный атрибут). Они нужны для того, чтобы обработчик мог идентифицировать выбранный вариант.

Например, мы выбираем способ доставки - срочная. Тогда переменная **dostavka** примет значение **srochnaya**. Если б не было значения **value="srochnaya"**, то переменная **dostavka** приняла бы значение **срочная** (т.е. значение текста, заключенного в элемент **option**).

Атрибут **SIZE** задает количество одновременно видимых пунктов меню.

Атрибут флаг **SELECTED** означает, что данное значение будет выбрано по умолчанию.

### Пример 5:

```
<form name="primer5" method="post"
action="obrabotchik.php">
<p>Выберите способ доставки:</p>
<p>
<select name="dostavka" size="1">
<option value="srochnaya" > Срочная </option>
<option value="ne srochnaya" selected> Не срочная
</option>
<option value="kurierom"> Курьером </option>
</select>
</p>
</form>
```

### Список с множественным выбором

Можно также сделать, чтобы имелась возможность выбора нескольких пунктов меню при удержании клавиши Ctrl. Для этого используется атрибут-флаг **MULTIPLE**.

### Пример 6:

```
<form name="primer6" method="post"
action="obrabotchik.php">
<p>Выберите способ доставки:</p>
<p>
<select name="dostavka" size="3" multiple >
<option value="srochnaya" > Срочная </option>
<option value="ne srochnaya"> Не срочная </option>
<option value="kurierom"> Курьером </option>
</select>
</p>
</form>
```

### Текстовый блок, элемент TEXTAREA

Для создания больших текстовых полей используется элемент **TEXTAREA**, который даже так и переводится - текстовая площадь. У этого элемента обязательно наличие закрывающего тега.

Атрибут **name** поможет идентифицировать заполненное поле на сервере.

Атрибуты **COLS** и **ROWS** определяют ширину и высоту поля. К примеру, cols="45" rows="5" означает, что ширина будет 45 колонок, а высота 5 рядов. Колонки и ряды естественно формируются из символов.

### Пример 7:

```
<form name="primer7" method="post" action="obrabotchik.php">
Введите адрес для доставки:<br>
```



```
<textarea name="adress" cols="45"
rows="5"></textarea>
</form>
```

## Кнопка

Для создания кнопок используется уже знакомый нам элемент **INPUT**. Если необходимо создать кнопку, которая будет отправлять данные обработчику, то пишем **type="submit"** (тип-отправить), а если хотите, чтоб кнопка выполняла функцию сброса, то **type="reset"**.

Атрибут **value** указывает на то, что будет написано на кнопке.

### Пример 8:

[illegible]

## Загрузка файла, элемент INPUT с атрибутом FILE

Идея передачи больших объемов данных давно владела умами разработчиков программного обеспечения для World Wide Web. При этом возможностей полей **textarea** для решения этой задачи было явно недостаточно. Во-первых, можно передавать только текстовую информацию; во-вторых, текст нужно набирать вручную непосредственно в браузере; в-третьих, существует (точнее — существовало) ограничение на длину строки в поле **textarea**, которое перекочевало в формы из стандарта RFC822. Таким образом, в первой спецификации форм возможности передать любые данные с компьютера клиента на сервер не было.

Элемент **INPUT** с атрибутом **FILE** служит для реализации загрузки файлов на сервер.

Начинающий тэг **<form>** содержит необходимый атрибут **encrypt**. Атрибут **encrypt** принимает значение **multipart/form-data**, который извещает сервер о том, что вместе с обычной информацией посылается и файл.

При создании текстового поля также необходимо указать тип файла – “file”.

### Пример 9:

```
<form enctype="multipart/form-data"
action="upload.php" method="post">
Загрузить файл: <input name="my_file" type="file">
<input type="submit" value="Отправить">
</form>
```

### Задание на лабораторную работу:

1. Создать HTML- форму на заданную тему.
2. Организовать формы для заполнения тематических сайтов. В заданиях указаны, какие элементы управления необходимо обязательно использовать.

зовать, можете добавить другие элементы и информацию на свое усмотрение. Картинки добавляются в виде файла, перечисляемые типы из списка. **Обязательно использовать элементы форм из стандарта HTML 5**, такие как: Email, url, tel, Search, Range, Number, date, Time, Datetime, Month, Color, List.

### **Индивидуальные задания:**

1. Цветы. Организовать форму для добавления информации о цветке: наименование цветка, изображение цветка, частота цветения - список, способ размножения – список, температура - два числа, описание цветка.

3. Детские игрушки. Организовать форму для добавления информации о детских игрушках: наименование игрушки, Фирма производитель – из списка, две картинки, вид игрушки {для мальчиков, для девочек, для всех}, возраст – два числа.

4. Актеры. Организовать форму для добавления информации об актере: ФИО актера, год рождения, перечень фильмов с его участием, пол {мужской, женский}, жанры – список с множественным выбором, фото, официальный сайт.

5. Писатели. Организовать форму для добавления информации о писателях: ФИО писателя, год рождения, жанры – множественный список, книги – в виде архивных файлов, страна в которой родился – выпадающий список, e-mail адрес.

6. Программы. Организовать форму для добавления информации о программах: название программы, архив с программой, тип программы – из списка, условия распространения - радио кнопки, операционная система – множественный список.

7. Операционные системы. Организовать форму для добавления информации об операционных системах: название, фирма производитель, версия, логотип – файл, требования к оперативной памяти – минимум, объем требуемой оперативной памяти, ссылка на сайт.

8. Мебель. Организовать форму для добавления информации о мебели: название, вид мебели – выпадающий список, коллекция – выпадающий список, фотография, цветовые оттенки – множественный список, ссылка на сайт фирмы производителя.

9. Недвижимость. Организовать форму для добавления информации об недвижимости: вид недвижимости – радио кнопка, этажность, площадь, стоимость, местонахождение, файл с фотографией, ссылка на место на карте.

10. Бытовая техника. Организовать форму для добавления информации о бытовой технике, название, фотография, фирма производитель – список, характеристики техники, год производства.

11. Книги. Организовать форму для добавления информации о книге: название, автор, год издательства, число страниц, жанр – из списка, языки – множественный список, архив с текстом книги, содержание книги.

12. Компьютерные игры. Организовать форму для добавления информации о компьютерных играх: название, версия, жанр - список, фирма производитель – список, постер – файл, описание игры, условия распространения.

13. Автомобили. Организовать форму для добавления информации об автомобилях: производитель – список, марка, тип двигателя, год производства, фотография, тип кузова – список.

14. Города. Организовать форму для добавления информации о городах: название города, страна – список, число жителей, площадь, ссылка на место на картах google, краткое описание.

15. Страны. Организовать форму для добавления информации о странах: название страны, число жителей, площадь, материк – список, климат, государственный язык.

16. Фрукты. Организовать форму для добавления информации о фруктах: название, описание, название дерева, наличие косточки, единицы измерения, диапазон веса для тех, которые можно измерять в штуках, примерная стоимость, сезонность.

17. Монеты. Организовать форму для добавления информации о монетах: название монеты, два изображения монеты, вес, размер монеты, стоимость монеты, вид – {юбилейная, разменная}, страна выпуска – список.

18. Животные. Организовать форму для добавления информации о животных: название животного, класс, к которому относится животное – список, фотография, продолжительность жизни животного, место обитания, занесен в красную книгу – checkbox.

19. История. Организовать форму для добавления информации о истории: название статьи, описываемый период, список основных героев, автор статьи, изображения (от 1 до 10), e-mail автора.

20. Музыкальные группы. Организовать форму для добавления информации о музыкальных группах: название группы фотография группы, состав группы, фотография каждого члена группы, дата образования, список альбомов, жанр – множественный список, действующая группа – checkbox.

21. Спорт. Организовать форму для добавления информации о видах спорта: вид спорта, является олимпийским – checkbox, логотип спорта, описание, вид – {зимний, летний}, {одиночный, командный}.

22. Путешествия. Организовать форму для добавления информации о путешествиях: страна, фотографии, описание и дата для каждой фотографии, ссылка на карту google.

23. Картины. Организовать форму для добавления информации о картинах: название картины, фотография картины, художник, место нахождения, стиль - выпадающий список, размеры картины, дата написания, оценочная стоимость, статья о картине.

24. Туризм. Организовать форму для добавления информации об туристических походах: Название похода, маршрут (из списка), фотографии,

описание маршрута, инструктор, e-mail инструктора, стоимость участия, длительность похода, дата начала похода.

25. Корабли. Организовать форму для добавления информации о кораблях: название корабля, тип корабля (из списка), водоизмещение, дата спуска, число человек экипажа, порт приписки, капитан.

26. Самолеты. Организовать форму для добавления информации о самолетах: название, страна производитель – выпадающий список, летные характеристики, вместительность, число пилотов, назначение – {грузовой, пассажирский}.

27. Космос. Организовать форму для добавления информации о космосе: название статьи, автор статьи, фотографии, список звезд.

28. Ремонт. Организовать форму для добавления информации о ремонте: наименование статьи, картинка, видео, текст статьи, вид проводимых работ – множественный список.

29. Кино. Организовать форму для добавления информации о фильмах: название, длительность в минутах, постер, список актеров, режиссер, саундтреки к фильму, кассовый сбор, язык – список.

30. Футбол. Организовать форму для добавления информации о футбольных командах: название команды, название страны из списка, дата создания, текущий рейтинг, клубные цвета, ссылка на сайт, эмблема в виде файла, признак участия в лиге чемпионов.

31. Моря. Организовать форму для добавления информации о морях: название – русское и английское, тип – внутреннее или внешнее, максимальная глубина, площадь, океан – список, уровень солености.

32. Стихи. Организовать форму для добавления информации о стихах: название стиха, стиль – из списка, текст стиха, автор, год написания, сборник.

33. Наука. Организовать форму для добавления информации о науке: название статьи, автор, ссылка на сайт автора, раздел наук – из списка, файл публикации, сборник в котором была опубликована, год издания.

34. Погода. Организовать форму для добавления информации о погоде: название природного явления, место возникновения – список множественного выбора, шкала измерения, картинки, описание.

35. Олимпиада. Организовать форму для добавления информации о олимпиаде: страна, проводящая олимпиаду из списка, год проведения, вид олимпиады (радиокнопка), число стран участниц, город проведения, гимн олимпиады и символ олимпиады в виде файла, почта медиацентра.

36. Горы. Организовать форму для добавления информации о горных вершинах: название горной вершины, высота, степень опасности, минимальная температура, фотография в виде файла, число жертв, страны (страны) в которой находится из списка с возможностью множественного выбора.

37. Фокусники. Организовать форму для добавления информации о фокусниках: ФИО, год рождения, страна рождения (список), статья о фо-

куснике - текст, фотографии в виде файла, перечень фокусов – множественный список, e-mail, телефон.

38. Компьютерные технологии. Организовать форму для добавления информации о компьютерных новинках: название, дата выпуска, несколько фотографий, статья, ссылки на производителя, категория – выпадающий список, ориентировочная стоимость в долларах, признак подключения к интернету.

39. Мода. Организовать форму для добавления информации о мероприятиях в мире моды: название мероприятия, проводящая организация, страна и город проведения (из списка), дата проведения, время начала мероприятия, телефон, e-mail, признак периодическое мероприятие или нет.

40. Сайт городского управления. Организовать форму для добавления информации на сайт городского управления: название статьи, дата добавления, дата новости, краткое содержание, полное содержание, фотография, e-mail для связи, раздел (из списка).

41. Гороскоп. Организовать форму для добавления информации о гороскопах: вид гороскопа, знак, изображение знака, описание, дата с – дата по, основные черты знака, драгоценный камень гороскопа.

42. Философия. Организовать форму для добавления информации о философских направлениях: название направления, основоположник, url-ссылки на литературу, год появления, краткое описание.

43. Тесты. Организовать форму для добавления информации о тестах: название теста, направленность (из списка), описание теста, сложность теста (радиокнопки), число прошедших тестирование.

44. Аптека. Организовать форму для добавления информации об аптеках города: название аптеки, признак социальной аптеки (чекбокс), адрес аптеки, признак наличия филиалов (чекбокс), владелец аптеки, интернет-страница, e-mail – адрес.

45. Театр. Организовать форму для добавления информации об известных театрах: название театра, фотография, адрес, страничка в Интернете, вид театра (радиокнопки), вместимость, год создания.

46. Поэты. Организовать форму для добавления информации о поэтах: Имя, год рождения, страна рождения (из списка), перечень лучших произведений, ссылка на страничку, фотография.

47. География. Организовать форму для добавления информации о географических открытиях: название, дата, первооткрыватель, страна первооткрывателя (из списка), коротко информация о значимости открытия, иллюстрации.

48. Аквапарки. Организовать форму для добавления информации об аквапарках: название, температура воды, вид аквапарка (открытый-закрытый, радиокнопки), время открытия и закрытия, город (из списка), стоимость взрослого билет, стоимость детского билета, наличие столовой или кафе на территории (чекбокс), фотография, число горок.

49. Отдых. Организовать форму для добавления информации о базах отдыха и отелях: название, администратор, телефон администратора, e-mail

администратора, фотография, страна (из списка), стоимость проживания, максимальное число отдыхающих, наличие питания (чекбокс).

50. Школа. Организовать форму для добавления информации о школах города: номер школы, вид школы (из списка средняя, лицей, общеобразовательная), номер школы, направление подготовки в школе (из списка множественного выбора: математическая, физическая, ...), директор школы, число учащихся, район города, адрес.

51. Языки программирования. Организовать форму для добавления информации о языках программирования: название языка, год появления, признак является ли он объектно-ориентированным языком, логотип, официальный сайт, процент пользователей, операционная система (список множественного выбора).

52. Университет. Организовать форму для добавления информации об университетах мира: название университета, тип университета (из списка), страна (из списка), признак наличия филиалов, логотип, официальный сайт, год создания.

53. Хоккей. Организовать форму для добавления информации о хоккейных клубах: название клуба, страна (из списка), год создания, официальный сайт, владелец клуба, список титулов, признак (является на данный момент клубом высшего дивизиона), два основных клубных цвета.

54. Музыка. Организовать форму для добавления информации о музыкальных произведениях: название произведения, год создания, автор, файл с музыкальным произведением, исполнитель, название альбома (если есть), жанр, длительность.

55. Ноутбуки. Организовать форму для добавления информации о моделях ноутбуков: название модели, дата выпуска, фирма производитель (из списка), процессор (радиокнопки), объем жесткого диска, диагональ экрана, разрешение экрана, операционная система (из списка) емкость батареи, цвет ноутбука.

56. Одежда. Организовать форму для добавления информации об одежде: название, тип одежды из списка, размер, цвет, модельер, фотография, материал (список множественного выбора), чекбокс (индивидуальный или серийный заказ), страна производитель (из списка).

57. Спортивный инвентарь. Организовать форму для добавления информации о спортивном инвентаре: название, описание, фотография, список фирм производителей, вес, сайт производителя, год выпуска.

58. Журналы. Организовать форму для добавления информации о периодических изданиях: название, число страниц, признак глянца, стоимость, периодичность выпуска (из списка), реклама журнала в виде файла.

59. Столицы. Организовать форму для добавления информации о столицах стран: столица, площадь, карта столицы-графический файл, численность населения, название страны, описание столицы, чекбокс – прибрежный город, город основания города.

60. Оружие. Организовать форму для добавления информации об оружии: название, вид оружия {огнестрельное, холодное, ...} в виде радио-

кнопок, материал, технические характеристики (текстовое поле), фотография, признак коллекционного оружия.

61. Холодное оружие. Организовать форму для добавления информации о холодном оружии: название, вид оружия из выпадающего списка {меч, кинжал, копье ...}, фотография оружия, вес, примерный год появления.

62. Бронетехника. Организовать форму для добавления информации бронетехнике: название, фотография, год выпуска, кем разработана, страна производитель, ориентировочная стоимость, вес, максимальная скорость, возможность плавать.

63. Фреймворки. Организовать форму для добавления информации о фреймворках: название, дистрибутив, версия, сайт разработчика, список поддерживаемых языков (из списка множественного выбора), радиокнопка (свободный, платный, условно-бесплатный), число пользователей.

64. Озера. Организовать форму для добавления информации об озерах: название, страна (множественный выбор), глубина, площадь, признак соленого озера, признак судоходности, карта глубин озера, длина береговой линии.

65. Телефоны. Организовать форму для добавления информации о моделях телефонах: производитель из списка, модель, год выпуска, операционная система если есть (из списка), фотография телефона, наличие фотоаппарата, наличие вспышки, поддержка GSM.

66. Зимние виды спорта. Организовать форму для добавления информации о зимних видах спорта: вид спорта, является ли он олимпийским, логотип вида спорта, описание вида спорта, страны лидеры (выбор из множественного списка).

67. Легкая атлетика. Организовать форму для добавления информации о спортсменах легкоатлетах: имя, страна за которую выступал, годы выступлений, признак олимпийского чемпиона, признак чемпиона мира, пол, год рождения, фотография, в каких дисциплинах выступал (список с множественным выбором).

### **Содержание отчета:**

1. Задание.
2. Таблица с перечнем использованных тегов.
3. Текст HTML-документа.
4. HTML-страница.
5. Выводы по работе.

### **Контрольные вопросы**

1. Что такое HTML-форма?
2. Какой тэг образует скелет формы?
3. Привести структуру HTML-формы. Описать назначение атрибутов name, action, method.

4. Перечислить атрибуты текстового поля.
5. Перечислить атрибуты поля переключателя.
6. Перечислить атрибуты для представления текстовой области.
7. Как ввести пароль в форму?
8. Как создать выпадающий список?
9. Как организовать отправку данных из формы?

## Лабораторная работа №9 Основы JavaScript

**Тема:** Основы JavaScript.

**Цель:** Научиться подключать JavaScript к html-документу, изучить основные средства ввода\вывода, изучить основные операторы языка.

### Добавление JavaScript на веб-страницы

JavaScript добавляется на веб-страницы с помощью тэга **<script>**.

```
<html>
<body>
<script type="text/javascript">
//Пишите код здесь
</script>
</body>
</html>
```

Функция **write** языка JavaScript используется для динамического ввода текста и тегов HTML в окно обозревателя.

**Синтаксис функции:**

**document.write("текст");**

В JavaScript используются три встроенных диалоговых окна:

– **alert ('сообщение')** – открывает диалоговое окно с текстом сообщения и единственной кнопкой **ОК**. Это диалоговое окно применяется для показа предупреждений или информационных сообщений, не требующих от пользователя принятия каких-либо решений.

– **confirm ('сообщение')** – открывает диалоговое окно с текстом сообщения и двумя кнопками – **ОК** и **Cancel**. В зависимости от выбора кнопки функция **confirm** возвращает значение **true (ОК)** или **false (Cancel)**.

– **prompt ('сообщение', 'текст по умолчанию')** – открывает окно для ввода данных пользователем. Кроме кнопок **ОК** и **Cancel** данное окно содержит текстовое поле ввода. Чтобы оставить текстовое поле пустым, введите во втором аргументе пустую строку – " ".

В JavaScript поддерживает следующие виды циклов:

- **for**
- **while**
- **do..while**



В таблице 9.1 приведены свойства и методы классов Math и DATE.  
Таблица 9.1 - Свойства и методы классов Math и DATE

<b>static Math: Математические константы и функции</b>	
number E	E
number LN2	ln(2)
number LN10	ln(10)
number LOG2E	log <sub>2</sub> (e)
number LOG10E	log <sub>10</sub> (e)
number PI	π
number Sqrt1_2	1/sqrt(2)
number Sqrt2	sqrt(2)
number abs(number a)	абсолютное значение
number acos(number a)	арккосинус
number asin(number a)	арксинус
number atan(number a)	арктангенс
number ceil(number a)	наименьшее целое, не меньшее a
number cos(number a)	косинус
number exp(number a)	экспонента
number floor(number a)	наибольшее целое, не большее a
number log(number a)	натуральный логарифм
number max(number a, number b)	максимум
number min(number a, number b)	минимум
number pow(number a, number b)	a в степени b
number random()	случайное число от 0 до 1
number sin(number a)	синус
number sqrt(number a)	квадратный корень
number tan(number a)	тангенс
<b>Date: Дата и время</b>	
static number UTC(number year, number month, number day [[, number hrs], number min], number sec])	Возвращает число миллисекунд, прошедших от Epoque (01.01.1970 00:00:00 GMT) до даты, указанной в качестве параметра
static number parse(string time)	Возвращает число миллисекунд, прошедших от Epoque (01.01.1970 00:00:00 по местному времени) до даты, указанной в качестве параметра
number getDate()	число
number getDay()	день недели
number getHours()	часы
number getMinutes()	минуты
number getMonth()	месяц
number getSeconds()	секунды
number getTime()	число миллисекунд с Epoque
number getTimeZoneOffset()	смещение временной зоны

<b>static Math: Математические константы и функции</b>	
number getYear()	год, начиная с 1900
setDate(number Date)	установка даты
setHours(number Hours)	установка часов
setMinutes(number Minutes)	установка минут
setMonth(number Month)	установка месяца
setSeconds(number Seconds)	установка секунд
setTime(number Time)	установка времени в миллисекундах, прошедших с Epochе
setYear(number Year)	установка года
string toGMTString()	преобразование к строке с датой и временем по Гринвичу
string toLocaleString()	преобразование к строке с местными датой и временем

### Задание на лабораторную работу:

1. Создать html-файл, который будет выводит текст индивидуального задания в тело документа.

2. Создайте диалоговое окно, которое будет запрашивать все необходимые данные о студенте (фамилия, имя, отчество студента с начальными и конечными пробелами, пол, дата рождения, группа) и выводить их в документе. Сформировать фразу по индивидуальному заданию, присоединив ее к общей части фразы. Результаты также вывести в документ.

**Я, студент (или студентка) группы <группа> <фамилия и инициалы>, родился (родилась) <дата>, это был день <название дня недели>;-общая часть фразы.**

### Индивидуальные задания:

1. Я учусь на <число> курсе. До моего дня рождения осталось <число> часов;

2. Мне исполнилось <число> лет, <число> месяцев и <число> дней; до 1 сентября осталось <число> недель

3. В следующем <число> году мне исполнится <число> лет; с 1 сентября прошло <число> часов

4. Я закончу университет в <число> году в возрасте <число> лет; с начала текущего месяца прошло <число> дней <число> часов и <число> минут

5. В следующем <число> году мой день рождения придется на <название дня недели>; до начала лета осталось <число> часов.

6. На сегодня <дата> я прожил (прожила) <число> дней и с начала пары прошло <число> минут.

7. Я получу бакалавра через <число> дней <число> часов и <число> минут; До ближайшей пятницы, выпадающей на 13-е число осталось <число> дней.

8. 1 сентября прошлого года был <название дня недели>; До дней весеннего и осеннего равноденствий (22 марта и 22 сентября) осталось <число> полных суток

9. Я учусь в потоке <название>. До начала лета осталось <число> часов и это будет <название дня недели>

10. Я получу специалиста через <число> дней <число> часов и <число> минут; через полгода будет <дата> и это будет <название дня недели>

3. Написать скрипт, реализующий вычисление выражений по индивидуальному варианту (таблица 9.2). Обеспечить ввод исходных данных. Сверить вычисленное по формуле выражение, записанное в виде бесконечного ряда, с точным значением по формуле, находящейся в левой части индивидуального задания. Отобразить основные и промежуточные результаты. **Массивы при выполнении этого задания не используются!**

Таблица 9.2 – Индивидуальное задание

№ варианта	Индивидуальное задание
1	$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + R_n(x),$
2	$(1+x)^\alpha = 1 + \frac{\alpha}{1!}x + \frac{\alpha(\alpha-1)}{2!}x^2 + \dots + \frac{\alpha(\alpha-1)\dots(\alpha-n+1)}{n!}x^n + R_n(x), \quad x \in ]-1; 1[,$
3	$\operatorname{arctg} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \dots + (-1)^m \frac{x^{2m+1}}{2m+1} + R_{2m+2}(x),$
4	$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + (-1)^{m-1} \frac{x^{2m-1}}{(2m-1)!} + \dots, \quad x \in R,$
5	$\operatorname{arctg} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \dots + (-1)^{m-1} \frac{x^{2m-1}}{2m-1} + \dots, \quad x \in [-1; 1].$
6	$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots, \quad x \in R,$
7	$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^m \frac{x^{2m}}{(2m)!} + \dots, \quad x \in R,$
8	$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \dots + (-1)^{n-1} \frac{x^n}{n} + \dots, \quad x \in ]-1; 1[,$
9	$\operatorname{sh} x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^{2m-1}}{(2m-1)!} + \dots, \quad x \in R,$

№ варианта	Индивидуальное задание
10	$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \dots - \frac{x^n}{n} - \dots, \quad x \in [-1; 1[.$
11	$\frac{1}{1+x} = 1 - x + x^2 - \dots + (-1)^n x^n + \dots, \quad x \in ]-1; 1[,$
12	$\frac{1}{1-x} = 1 + x + x^2 + \dots + x^n + \dots, \quad x \in ]-1; 1[,$
13	$\operatorname{ch} x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + \frac{x^{2m}}{(2m)!} + \dots, \quad x \in \mathbb{R},$
14	$\ln \frac{1+x}{1-x} = 2 \left( x + \frac{x^3}{3} + \frac{x^5}{5} + \dots + \frac{x^{2m-1}}{2m-1} + \dots \right), \quad x \in ]-1; 1[,$
15	$\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + \dots + (n+1)x^n + \dots, \quad x \in ]-1; 1[,$

### Содержание отчета:

1. Задание.
2. Текст HTML-документа.
3. Текст JavaScript-документа
4. Выводы по работе.

### Контрольные вопросы

1. Как добавить JavaScript на страницу?
2. Для чего нужна функция **write**? Приведите пример использования.
3. Какие вы знаете встроенные диалоговые окна в JavaScript?
4. Какие виды циклов поддерживает JavaScript?
5. Какие вы знаете свойства и методы классов Math и DATE?

## Лабораторная работа №10

### JavaScript. Массивы

**Тема:** Основы JavaScript. Массивы, функции, регулярные выражения.

**Цель:** Изучить основы работы с массивами, создать пользовательскую функцию, изучить основы работы с регулярными выражениями

### Теория Массивы

Массивы представляют собой контейнеры, в которых может быть сохранено неограниченное количество переменных.

Каждому значению, которое заносится в массив, присваивается уникальный идентификатор, по которому можно обращаться к данному элементу внутри массива.

Создать массивы можно тремя разными способами:

#### Первый способ:

```
item = new Array();  
item[0] = "Автомобиль";  
item[1] = "Микроволновая печь";  
item[2] = "Стиральная машина";  
item[3] = "Пылесос";
```

#### Второй способ:

```
var item = new Array("Автомобиль", "Микроволновая  
печь", "Стиральная машина",  
"Пылесос");
```

#### Третий способ:

```
var item = ["Автомобиль", "Микроволновая печь",  
"Стиральная машина", "Пылесос"];
```

Для того, чтобы обратиться к элементу, сохраненному в массиве, необходимо указать имя массива и индекс желаемого элемента в квадратных скобках (массив[индекс]).

Нумерация индексов в массивах начинается с 0.

С помощью свойства **length** можно узнать количество элементов в массиве.

С помощью метода **concat()** можно объединить два и более массива в один.

С помощью метода **sort()** можно отсортировать значения массива.

С помощью метода **pop()** происходит удаление последнего элемента массива.

Метод **slice(начало, конец)** позволяет извлечь (вырезать) элементы из массива и создать из извлеченных элементов новый массив.

### JavaScript функции

Функции являются одним из наиболее важных строительных блоков кода в JavaScript.

Функции состоят из набора команд и обычно выполняют какую-то одну определенную задачу (например, суммирование чисел, вычисление корня и т.д.).

Код помещенный в функцию будет выполнен только после явного вызова этой функции.

**Пример:**

```
//Объявление функции
function имяФункции(пер1, пер2){
    Код функции
}
//Вызов функции
имяФункции(пер1, пер2);
```

**Синтаксис:**

```
//Объявление функции
var имяфункции=function(пер1, пер2){Код функции}
//Вызов функции
имяфункции(пер1, пер2);
```

**Имяфункции** задает имя функции. Каждая функция на странице должна иметь уникальное имя. Имя функции должно быть задано латинскими буквами и не должно начинаться с цифр.

**пер1** и **пер2** являются переменными или значениями, которые можно передавать внутрь функции. В каждую функцию может быть передано неограниченное количество переменных.

Имена функций в JavaScript чувствительны к регистру!

**Пример JavaScript функции:**

Функция `messageWrite()` в примере будет выполнена только после нажатия на кнопку.

```
<html>
<head>
<script type='text/javascript'>
// Функция выводит текст на страницу
function messageWrite() {
    document.write('Данный текст был выведен на
страницу с помощью JavaScript!');
}
</script>
</head>
<body>
<input type='button' value='Нажми на меня' on-
click='messageWrite()' />
</body>
</html>
```

**Передача функциям переменных.** Вы можете передавать функциям неограниченное количество переменных. Все манипуляции над переменными внутри функций на самом деле производятся не над самими переменными а над их копией, поэтому содержимое самих переменных в результате выполнения функций не изменяется.

**Пример:**

```
/* Зададим функцию, которая прибавляет к передан-
ной переменной 10 и выводит результат на страницу */
function plus(a){
    a=a+10;
    document.write('Вывод функции: ' + a+'<br />');
}
var a=25;
document.write('Значение переменной до вызова
функции: '+a+'<br />');
// Вызовем функцию передав ей в качестве аргумента
переменную a
plus(a);
document.write('Значение переменной после вызова
функции: '+a+'<br />');
```

Чтобы обращаться к глобальной переменной из функции, а не ее копии используйте **window.имя\_переменной**.

**Пример:**

```
function plus(a){
    window.a=a+10;
}
var a=25;
document.write('Значение переменной до вызова
функции: '+a+'<br />');
plus(a);
document.write('Значение переменной после вызова
функции: '+a+'<br />');
```

**Команда return.** С помощью команды **return** можно возвращать из функций значения.

**Пример:**

```
<html>
<head>
<script type='text/javascript'>
//Функция sum возвращает сумму переданных в нее
переменных
function sum(v1,v2){
    return v1+v2;
}
</script>
```

```

</head>
<body>
<script type='text/javascript'>
document.write('5+6=' + sum(5,6) + '<br />');
document.write('10+4=' + sum(10,4) + '<br />');
</script>
</body>
</html>

```

**Встроенные функции.** Помимо определяемых пользователем функций в JavaScript существуют еще и **встроенные функции**. К примеру встроенная функция **isFinite** позволяет проверить является ли переданное значение допустимым числом.

**Пример:**

```

document.write(isFinite(40)+'<br />');
document.write(isFinite(-590)+'<br />');
document.write(isFinite(90.33)+'<br />');
document.write(isFinite(NaN)+'<br />');
document.write(isFinite('Это строка')+'<br />');

```

**Локальные и глобальные переменные.** Переменные, создающиеся внутри функций, называются **локальными переменными**. Можно обращаться к таким переменным только внутри функций, в которых они были определены.

После завершения выполнения кода функции такие переменные уничтожаются. Это значит, что в разных функциях могут быть определены переменные с одинаковым именем.

Переменные, которые создаются вне кода функций называются **глобальными переменными** к таким переменным можно обращаться из любого места кода.

Если объявить переменную без **var** внутри функции она тоже становится глобальной.

Глобальные переменные уничтожаются только после закрытия страницы.

**Пример:**

```

<html>
<head>
<script type='text/javascript'>
//Объявим глобальные переменные var1 и var2
var var1="var1 существует";
var var2;

function func1() {
    //Присвоим var2 значение внутри функции func1
    var var2="var2 существует";

```



```

    }
    //Из другой функции выведем содержимое переменной
var1 и var2 на страницу
    function func2() {
        //Выводим содержимое переменной var1
        document.write(var1 + '<br />');
        //Выводим содержимое переменной var2
        document.write(var2);
    }
</script>
</head>
<!-- Вызовем функцию func2() после полной загрузки
документа -->
<body onload='func2()'>
</body>
</html>

```

## Регулярные выражения

В шаблонах регулярных выражений могут использоваться специальные символы (метасимволы). Специальные символы с описаниями приведены в таблице 10.1.

Таблица 10.1 – Специальные символы в регулярных выражениях

Специальный символ	Описание
.	Совпадает с любым символом, кроме символа конца строки.
\w	Совпадает с любым буквенным символом.
\W	Совпадает с любым не буквенным символом.
\d	Совпадает с символами, которые являются цифрами.
\D	Совпадает с символами, которые не являются цифрами.
\s	Совпадает с пробельными символами.
\S	Совпадает с не пробельными символами.
\b	Совпадения будут искаться только на границах слов (в начале или конце).
\B	Совпадения будут искаться только не на границах слов.
\n	Совпадает с символом перевода строки.

### Пример:

/\* Выражение reg1 найдет все слова, начинающиеся на две произвольные буквы и заканчивающиеся на 'вет'. Так как слова в предложении разделяются пробелом, то в начале и в конце добавим спецсимвол \s) \*/

```
reg1=/\s..вет\s/g;
```

```
txt=' привет завет вельвет клозет ';
```

```
document.write(txt.match(reg1) + '<br />');
```

**Квантификатор** - это конструкция, позволяющая задать сколько раз предшествующий ей символ или группа символов должна встречаться в совпадение.

**Синтаксис:**

**{x}**-Предшествующий символ должен встречаться x – раз.

**{x,y}**-Предшествующий символ должен встречаться от x до y раз включительно.

**{x,}**-Предшествующий символ должен встречаться не менее x раз.

**\***-Указывает, что предшествующий символ должен встречаться 0 или более раз.

**+**-Указывает что предшествующий символ должен встречаться 1 или более раз .

**?**-Указывает что предшествующий символ должен встречаться 0 или 1 раз .

**Задание на лабораторную работу:**

**Задание1.** Написать скрипт, реализующий работу с массивом по индивидуальному заданию. Для обработки массива и вывода результата работы использовать функцию. Вывод производить с задержкой `timeout(1000)`;

**Индивидуальные задания:**

**Вариант 1** При помощи языка программирования Java Script разработать алгоритм программы, которая находит сумму положительных чисел от x до  $x^4$ . Пользователь вводит значение x в текстовое поле формы. Для вычисления создать пользовательскую функцию `calculation_x`, переменная x является глобальной. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 2** При помощи языка программирования Java Script разработать алгоритм программы, которая находит сумму отрицательных чисел в диапазоне от x до y. Пользователь вводит значение x и y в поля формы. Для вычисления создать пользовательскую функцию `calculation_x`, значение x и y передаются в качестве параметра. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 3** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  подсчитывает количество отрицательных чисел. Пользователь вводит значение n в текстовое поле формы. Массив значений x генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, значения n и x передаются в качестве параметра. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 4** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$

подсчитывает количество положительных чисел. Пользователь вводит значение  $n$  в текстовое поле формы. Массив значений  $x$  генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные  $n$  и  $x$  являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 5** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  подсчитывает сумму положительных чисел. Пользователь вводит значение  $n$  в текстовое поле формы. Массив значений  $x$  генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, значения  $n$  и  $x$  передаются в качестве параметра. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 6** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  подсчитывает сумму отрицательных чисел. Пользователь вводит значение  $n$  в текстовое поле формы. Массив значений  $x$  генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные  $n$  и  $x$  являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 7** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  находит максимум и минимум. Пользователь вводит значение  $n$  в текстовое поле формы. Массив значений  $x$  генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные  $n$  и  $x$  являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 8** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  подсчитывает количество нулевых элементов. Пользователь вводит значение  $n$  в текстовое поле формы. Массив значений  $x$  генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные  $n$  и  $x$  являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 9** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  определяет числа с четными номерами и больше семи. Пользователь вводит значение  $n$  в текстовое поле формы. Массив значений  $x$  генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные  $n$  и  $x$  являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 10** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  определяет количество элементов, имеющих четные порядковые номера и являющимися нечетными числами. Пользователь вводит значение  $n$  в текстовое поле формы. Массив значений  $x$  генерируется случайным образом.

Для вычисления создать пользовательскую функцию `calculation_x`, переменные `n` и `x` являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 11** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  определяет квадратный корень суммы квадратов элементов. Пользователь вводит значение `n` в текстовое поле формы. Массив значений `x` генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные `n` и `x` являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 12** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  заменяет все числа 13 в массиве на 0. Пользователь вводит значение `n` в текстовое поле формы. Массив значений `x` генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные `n` и `x` являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 13** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  заменяет все отрицательные числа на 1. Пользователь вводит значение `n` в текстовое поле формы. Массив значений `x` генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные `n` и `x` являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 14** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  удаляет все отрицательные элементы. Пользователь вводит значение `n` в текстовое поле формы. Массив значений `x` генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные `n` и `x` являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Вариант 15** При помощи языка программирования Java Script разработать алгоритм программы, которая для заданного ряда чисел  $x_1, x_2, \dots, x_n$  удаляет все четные числа. Пользователь вводит значение `n` в текстовое поле формы. Массив значений `x` генерируется случайным образом. Для вычисления создать пользовательскую функцию `calculation_x`, переменные `n` и `x` являются глобальными. Вывести итоговый и промежуточные результаты в виде таблицы.

**Задание 2.** Выполнить индивидуальное задание по работе с регулярными выражениями.

### **Индивидуальные задания:**

1. Проверить, является ли переменная вещественным числом.
2. Запретить пользователю использовать в своем имени любые символы, кроме букв русского и латинского алфавита, знака "\_" (подчерк), пробела и цифр.
3. Проверка адреса e-mail.
4. Проверка имени файла на корректность.
5. Проверка расширения файла Архивы (zip, rar, ...).
6. Проверка расширения файла Аудио (mp3, wav, ...).
7. Проверка расширения файла Изображения (jpg, png, ...).
8. Проверка расширения файла Видео (mpeg, avi, ...).
9. Является ли строка ценой.
10. Является ли ссылкой.
11. Состоит ли строка только из букв, цифр и "\_", длиной от 8 до 20 символов.
12. Проверка повторяющихся символов. Есть ли в строке идущие подряд символы, не менее 3-х символов подряд (типа "абвгДДДеё", но не "ааб-баабб").
13. Есть ли в строке многократно повторяющихся знаков препинания.
14. Проверка URL на корректность.
15. Является ли локальным URL.
16. Является ли строка датой.
17. Проверить телефонный номер (номер должен быть длиной 10 знаков и начинаться с 8 или 9).

### **Содержание отчета:**

1. Задание.
2. Текст JavaScript-документа
3. Выводы по работе.

### **Контрольные вопросы**

1. Что такое массив?
2. Способы создания массивы? Приведите примеры.
3. Как обратиться к элементу массива?
4. Что такое функция? Приведите пример объявления и вызова функции.
5. Как передать переменную в функцию.
6. Как обратиться к глобальной переменной из функции?
7. Для чего используется команда return?
8. Что такое встроенные функции?
9. Что такое регулярные выражения?
10. Какие вы знаете специальные символы в регулярных выражениях?

## Лабораторная работа №11

### Проверка форм с помощью JavaScript

**Цель работы:** изучение возможностей языка JS, создание интерактивных форм при помощи клиентских скриптов.

**Тема работы:** создание формы на Web-странице с функциями проверки полей при помощи языка клиентских скриптов JS.

### Поля формы

Элементы управления форм (текстовое поле ввода, поле редактора, флажок, переключатель, список), с помощью которых можно вводить данные, называются полями формы.

Общим для всех полей является то, что они могут хранить данные, вводимые или изменяемые пользователями. С помощью методов формы можно автоматически извлечь данные из всех полей и передать их на сервер для программной обработки.

#### Пример 1:

```
<HTML>
<BODY>
<FORM ID="form1" NAME="form1">
<BR><INPUT TYPE=radio NAME=color>Белый
<BR><INPUT TYPE=radio NAME=color>Красный
<BR><INPUT TYPE=radio NAME=color>Желтый
<BR><INPUT TYPE=radio NAME=color>Зеленый<BR>
<BR><INPUT TYPE='button' VALUE='ВЫВОД результата'
ONCLICK='show_select()'>
</FORM>
<SCRIPT>
function show_select()
{if (form1.color[0].checked)
  alert ('Белый');
if (form1.color[1].checked)
  alert ('Красный');
if (form1.color[2].checked)
  alert ('Желтый');
if (form1.color[3].checked)
  alert ('Зеленый');};
</SCRIPT>
</BODY>
</HTML>
```

Примеры проверок, которые возможно реализовать с помощью JavaScript:

1. Было ли данное поле заполнено;
2. В правильном ли формате пользователь указал свой адрес или Email в соответствующем поле;

3. Совпадают ли значения, введенные в два различных поля (часто используется для полей с паролем);

4. Не превышает ли значение, введенное в поле, максимально допустимый размер и т.д.

Проверка форм в JavaScript возможна благодаря событию onSubmit.

Если у тега form атрибут onsubmit="return true" форма будет отправлена на сервер, если же onsubmit="return false", то форма на сервер отправлена не будет.

**Пример 2:**

```
<html>
<head>
<script type="text/javascript">
function validate() {
//Считаем значения из полей name и email в пер. x и y
    var x=document.forms["form"]["name"].value;
    var y=document.forms["form"]["email"].value;
    //Если поле name пустое выведем сообщение и
предотвратим отправку формы
    if (x.length==0) {

document.getElementById("namef").innerHTML="*данное
поле обязательно для
        заполнения";
        return false;
    }
    //Если поле email пустое выведем сообщение и
предотвратим отправку формы
    if (y.length==0) {

document.getElementById("emailf").innerHTML="*данное
поле обязательно для
        заполнения";
        return false;
    }
//Проверим содержит ли значение введенное в поле email
символы @ и .
    at=y.indexOf("@");
    dot=y.indexOf(".");
    //Если поле не содержит эти символы знач email
введен не верно
    if (at<1 || dot <1){
        docu-
ment.getElementById("emailf").innerHTML="*email введен
не верно";
        return false;
    }
}
```

```

    }
}
</script>
</head>
<body>
<form name="form" onsubmit="return validate()">
Имя: <input type="text" name="name"> <span
style="color:red" id="namef"></span><br />
e-mail: <input type="text" name="email"> <span
style="color:red" id="emailf"></span>
<br /><input type="submit" value="Отправить
форму">
</form>
</body>
</html>

```

### **Задание на лабораторную работу:**

1. Организовать проверки для формы, созданной в процессе выполнения лр №8.

### **Порядок проведения работы:**

1. Создать файл index.html. Используя теги <HTML>, <HEAD>, <TITLE> и <BODY> описать общую структуру HTML документа. Создать HTML форму на заданную тему (см задание к лр №8).

2. Организовать проверку корректности заполнения формы с помощью JavaScript. При помощи языка JS осуществить проверку содержимого основных полей формы при нажатии кнопки “Отправить”.

3. При отсутствии данных в обязательных полях вывести соответствующее сообщение и прекратить обработку формы. При правильном заполнении формы открыть страницу сайта, предназначенную для обработки данных формы.

4. Организовать проверку введенного пароля или проверку от роботов в виде ответа на проверочный вопрос или ввод числа с картинки.

5. Организовать проверку на соответствие поля шаблону.

### **Содержание отчета:**

1. Задание.
2. Текст HTML-документа и JavaScript-документа.
3. Выводы по работе.

### **Контрольные вопросы:**

1. Что такое поле формы?
2. Какие проверки можно реализовать с помощью JavaScript?





### Тема: Использование языка SVG

**Цель работы:** приобретение навыков создания векторных изображений с использованием языка SVG, а также навыков рисования на холсте Canvas.

#### Теоретические сведения

Базовые типы данных и базовые интерфейсы SVG

Атрибуты элементов SVG делятся на две категории: свойства (атрибуты, позаимствованные из CSS2, например, font-family) и собственно атрибуты SVG (например, fill).

В SVG определены следующие **базовые типы данных**:

- целое (integer) – целое число, возможно со знаком "+" или "-";
- число (number) – действительное число с целой и дробной частью (в атрибутах SVG можно использовать также экспоненциальную форму);
- длина (length) – данное типа число, в котором длина измеряется в тех же единицах, что и в CSS2: пикселях, точках, сантиметрах и т.д. (если единица измерения не задана, предполагается, что длина задана в пикселях);
- координата (coordinate) – данное типа длина, представляющая собой расстояние в пользовательской координатной системе от ее начала вдоль оси X или оси Y;
- список (list) – список данных одного типа, разделенных запятыми и/или пробелами (частным случаем списка является список данных типа число);
- угол (angle) – данное типа число, за которым следует единица измерения: deg – градусы, grad – грады или rad – радианы (в атрибутах SVG единица измерения deg может быть опущена);
- цвет (color) – цвет, задаваемый либо как наименование цвета (используются те же наименования цветов, что и в спецификации HTML 4.01), либо как шесть шестнадцатеричных значений с префиксом "#" (первая пара – интенсивность красного цвета, вторая – зеленого и третья – синего), либо как вызов функции rgb() с тремя десятичными параметрами для интенсивностей красного, зеленого и синего цветов;
- раскраска (paint) – тип раскраски, используемый для заполнения или контура изображения (эта величина может иметь значение либо "none"
- отсутствие раскраски, либо "currentColor" – ссылка на цвет, задаваемый атрибутом color, либо быть данным типа цвет);
- процент (percentage) – данное типа число, за которым следует символ "%";

- список-преобразований (transform-list) – содержит список совершаемых над объектом преобразований (например, перемещения или растягивания); URI – адрес ресурса в Web;
- время (time) - данное типа число, за которым следует единица измерения: s – секунда, ms – миллисекунда.

В SVG определены следующие **графические модули атрибутов**:

- модуль цвета и раскраски Paint.attrib (Paint Attribute Module);
- модуль атрибутов прозрачности Opacity.attrib (Opacity Attribute Module);
- модуль графических атрибутов Graphics.attrib (Graphics Attribute Module);
- модуль маркера Marker.attrib (Marker Module);
- модуль градиента Gradient.attrib (Gradient Module);
- модуль клипа Clip.attrib (Clip Module);
- модуль маски Mask.attrib (Mask Module);
- модуль фильтра Filter.attrib (Filter Module);
- модуль цветового фильтра FilterColor.attrib (Filter Color Module).

**Модуль цвета и раскраски Paint.attrib** включает атрибуты color, fill, fill-rule, stroke, stroke-width, stroke-linecap, stroke-linejoin, stroke-miterlimit, stroke-dasharray, stroke-dashoffset, color-interpolation и color-rendering.

Атрибут color определяет текущий цвет для атрибутов, значением которых являются данные типа раскраска (атрибут fill для заполнения фигуры и атрибут stroke для закраски контура фигуры).

Атрибут fill-rule определяет алгоритм закраски фигуры и может иметь либо значение "nonzero", либо значение "evenodd".

Атрибут stroke-width, значением которого является данное типа длина, определяет ширину контура.

Атрибут stroke-linecap определяет форму фигуры на концах открытого контура и может иметь значения "butt" (значение по умолчанию), "round" или "square".

Атрибут stroke-linejoin определяет форму фигуры при соединении контуров и может иметь значения "miter" (значение по умолчанию), "round" или "bevel".

Атрибут stroke-miterlimit используется, когда два сегмента соединяются под острым углом и атрибут stroke-linejoin имеет значение "miter". Значение этого атрибута задает ограничение на отношение длины пересечения к ширине контура (по умолчанию это значение равно "4"). Если это ограничения превышено, то соединение преобразуется в соединение типа "bevel".

Атрибут stroke-dasharray задает шаблон длин штрихов и промежутков между штрихами при заполнении контура штрихами, например, значение "5px 2px" задает длину штриха 5 пикселей и промежутка между штрихами 2

пикселя. Значение атрибута "none" задает отсутствие штриховки контура (это значение по умолчанию).

Атрибут stroke-dashoffset определяет расстояние (данное типа длина), на котором выполняется штриховка (по умолчанию это значение равно "0").

Атрибут color-interpolation задает пространство цветов для градиентной интерполяции и цветовой анимации и может иметь одно из следующих значений: "sRGB" – выполнение цветовой интерполяции в цветовом пространстве sRGB (значение по умолчанию), "linearRGB" – выполнение цветовой интерполяции в линейном цветовом пространстве sRGB, "auto" – пользовательский агент сам выбирает пространство цветов для интерполяции.

Атрибут color-rendering задает, как будет выполняться цветовая интерполяция и может иметь одно из следующих значений: "optimizeSpeed" – оптимизация по скорости, "optimizeQuality" – оптимизация по качеству, "auto" – способ оптимизации выбирает пользовательский агент.

Модуль атрибутов прозрачности `Opacity.attrib` содержит три атрибута: `opacity` и `fill-opacity`. Атрибут `opacity` задает прозрачность объекта в виде числа в диапазоне от 0 (прозрачный) до 1 (непрозрачный) (значение по умолчанию равно "1.0"). Атрибуты `stroke-opacity` и `fill-opacity` задают ту же характеристику отдельно для контура объекта и содержимого объекта.

**Модуль графических атрибутов `Graphics.attrib`** задает графические атрибуты объекта: `display`, `image-rendering`, `pointer-events`, `shape-rendering`, `text-rendering` и `visibility`.

Атрибут `image-rendering` задает, как будет выводиться изображение и может иметь одно из следующих значений: "optimizeSpeed" – оптимизация по скорости, "optimizeQuality" – оптимизация по качеству, "auto" – способ оптимизации выбирает пользовательский агент.

Атрибут `shape-rendering` задает, как будет выводиться контур изображения и может иметь одно из следующих значений: "optimizeSpeed" – оптимизация по скорости, "crispEdges" – оптимизация по различимости контура по отношению к фону, "geometricPrecision" – оптимизация по геометрической точности отображения контура, "auto" – способ оптимизации выбирает пользовательский агент.

Атрибут `shape-rendering` задает, как будет выводиться текст и может иметь одно из следующих значений: "optimizeSpeed" – оптимизация по скорости, "optimizeLegibility" – оптимизация по четкости текста, "geometricPrecision" – оптимизация по геометрической точности отображения текста, "auto" – способ оптимизации выбирает пользовательский агент.

Атрибуты `display` и `visibility` позаимствованы из CSS2.

Атрибут `display` задает тип выводимого элемента (по умолчанию "inline" – строковый тип), а атрибут `visibility` – видимость элемента (по умолчанию "visible" – видимый).

Атрибут `pointer-events` определяет, при каких условиях может быть мишенью для события указателя. Значения "visiblePainted" (значение по умолчанию), "visibleFill", "visibleStroke" и "visible" определяют, что элемент

может быть мишенью события, когда значение свойства `visibility` равно `"visible"`, и указатель находится соответственно над областью раскраски, областью заполнения, контуром или любой точкой элемента. Значения `"painted"`, `"fill"`, `"stroke"` и `"all"` определяют те же условия, но независимо от значения свойства `visibility`. Значение `"none"` задает, что элемент не может быть мишенью события.

**Модуль градиента `Gradient.attrib`** содержит атрибуты `stop-color` и `stop-opacity` определяющие цвет и прозрачность при завершении градиента. Для атрибута `stop-color` может быть задано либо значения `"currentColor"`, либо значение цвета (по умолчанию значение этого атрибута равно `"black"`), а для атрибута `stop-opacity` – значение прозрачности в диапазоне от 0 до 1 (по умолчанию `"1"`).

Модуль фильтра `Filter.attrib` содержит атрибут `filter`. Значением этого атрибута может быть либо URI элемента `filter`, используемого в качестве фильтра для данного элемента, либо `"none"` – фильтр не используется (это значение по умолчанию). Модуль цветового фильтра `FilterColor.attrib` содержит атрибут `color-interpolation-filters`, определяющий пространство цветов для операций над изображениями, выполняемыми с помощью фильтров. Этот атрибут те же значения, что и атрибут в модуле `Paint.attrib`.

В SVG определены следующие **базовые фигуры**:

- прямоугольник (элемент `rect`);
- круг (элемент `circle`);
- эллипс (элемент `ellipse`);
- линия (элемент `line`);
- ломаная линия (элемент `polyline`);
- многоугольник (элемент `polygon`).

Для всех элементов можно использовать следующие модули атрибутов: `Core.attrib`, `Conditional.attrib`, `Style.attrib`, `Paint.attrib`, `Color.attrib`, `Opacity.attrib`, `Graphics.attrib`, `Clip.attrib`, `Mask.attrib`, `Filter.attrib`, `GraphicalEvents.attrib`, `Cursor.attrib` и `External.attrib`. Для элементов базовых фигур определены интерфейсы `SVGRectElement`, `SVGCircleElement`, `SVGEllipseElement`, `SVGLineElement`, `SVGPolylineElement` и `SVGPolygonElement`, атрибутами которых являются атрибуты соответствующих элементов. Интерфейс `SVGAnimatedPoints` используется для создания анимации ломаных линий и многоугольников.

### **Атрибут `transform`**

Для многих элементов SVG можно задавать атрибут `transform`. Значением этого атрибута является данное типа список-преобразований, содержащее определения преобразований, выполняемых над элементом, в порядке, определяемом списком.

В SVG доступны следующие **определения преобразований**:

- `matrix (a b c d e f)` – определяет преобразование в форме матрицы преобразования из шести значений;

- `translate (tx [ty])` – определяет перемещение на величину `tx` по оси `x` и `ty` по оси `y` (если `ty` не задано, оно предполагается равным 0);
- `scale (sx [sy])` – определяет операцию масштабирования на величину `sx` по оси `x` и величину `sy` по оси `y` (если `sy` не задано, оно предполагается равным `sx`);
- `rotate (угол-вращения [sx sy])` – определяет вращение на угол-вращения вокруг заданной точки (если не заданы `sx` и `sy`, вращение выполняется относительно начала пользовательских координат);
- `skewX(угол-наклона)` – определяет преобразование наклона вдоль оси `x`;
- `skewY(угол-наклона)` – определяет преобразование наклона вдоль оси `y`.

Все значения в преобразованиях являются данными типа число.

### Элемент **g**

Элемент **g** используется для помещения связанных между собой элементов SVG в контейнер (обычно для повторного использования или анимации). Атрибутами этого элемента могут быть атрибуты модуля `Presentation.attrib` и описанный ранее атрибут преобразования **`transform`**.

Здесь оба прямоугольника, заданные в контейнере будут закрашены синим цветом.

Элементу **g** в DOM SVG соответствует интерфейс **`SVGGElement`**.

### Контур в SVG

Контур используется для фигур, которые можно заполнить, выделить границы или использовать для вырезания (`clipping`) графической фигуры.

Контур представляет геометрию границы объекта, определенную в терминах перемещения (установки новой текущей точки), линии (рисования прямой линии), кривой (рисования кривой Безье), дуги (эллиптической дуги или дуги окружности) и закрытия контура (закрытие текущей фигуры с помощью рисования линии к первому перемещению). Допустимы также составные контуры, т.е. пути с несколькими подконтурами.

Контур определяется в SVG с использованием элемента **`path`**. Для этого элемента определены атрибуты наборов **`Core.attrib`**, **`Conditional.attrib`**, **`Style.attrib`**, **`Paint.attrib`**, **`Color.attrib`**, **`Opacity.attrib`**, **`Graphics.attrib`**, **`Marker.attrib`**, **`Clip.attrib`**, **`Mask.attrib`**, **`Filter.attrib`**, **`GraphicalEvents.attrib`**, **`Cursor.attrib`**, **`External.attrib`** и атрибут **`transform`**.

Обязательный атрибут **`d`** задает данные для контура, а атрибут **`pathLength`** задает рассчитанную автором длину контура.

Элементу `path` в SVG DOM соответствует интерфейс `SVGPathElement`.

`Path` позволяет задать любую фигуру компактной строкой, описывающей путь от начальной точки до конечной через любые промежуточные координаты. Строка с данными задаётся атрибутом `d` тега `path` и содержит команды, закодированные набором букв и чисел. Буква определяет тип команды, а числа - её параметры (обычно координаты).

Команды позволяют описывать фигуры, состоящие из отрезков прямых (L, H, V), кривых Безье (C, S, Q, T) и дуг (A).

Пример, описывающий звезду из 5 линий, содержит строку данных с командами M (moveto - переместить) и L (lineto - нарисовать линию), содержащими в качестве аргументов координаты точек по X и Y.

```
<path fill="none"
      stroke="black"
      d="M 227 239 L 328 90 L 346 250 L 201 124 L 410 150 L
228 238» />
```

## ГРАДИЕНТЫ И ШАБЛОНЫ В SVG

### Линейный градиент

Градиент – постепенный плавный переход вдоль заданного вектора от одного цвета к другому. Если градиент задан, то него можно ссылаться с использованием свойств **fill** или **stroke** в графическом элементе. Язык SVG обеспечивает два типа градиентов: линейный градиент или радиальный градиент.

Линейный градиент задается с помощью элемента **linearGradient**. Для этого элемента можно задавать атрибуты модулей **Core.attrib**, **Style.attrib**, **Color.attrib**, **Gradient.attrib**, **XLink.attrib** и **External.attrib**. Кроме этого, для элемента **linearGradient** могут быть заданы атрибуты **x1**, **y1**, **x2**, **y2**, **gradientUnits**, **gradientTransform** и **spreadMethod**.

Атрибуты **x1**, **y1**, **x2** и **y2** задают вектор градиента для линейного градиента (числа или проценты). Вектор градиента обеспечивает начальные и конечные точки, которым соответствует окончание градиента. По умолчанию значения координат равны "0%".

Атрибут **gradientUnits** определяет систему координат для атрибутов **x1**, **y1**, **x2** и **y2**. Если значение этого атрибута равно "**userSpaceOnUse**", то координаты представляют значения в системе координат текущего пользователя во время ссылки на элемент **linearGradient**. Если же значение атрибута **gradientUnits** равно "**objectBoundingBox**" (это значение по умолчанию), то пользовательская система координат устанавливается исходя из рамки элемента, к которому применяется градиент.

Атрибут **gradientTransform** задает список дополнительных преобразований из координатной системы градиента в целевую координатную систему, определяемую атрибутом **gradientUnits**.

Атрибут **spreadMethod** указывает, что произойдет, если градиент начинается или заканчивается в пределах целевого прямоугольника. Атрибут может иметь одно из следующих значений: "**pad**" – концевые цвета градиента должны заполнить весь прямоугольник, "**reflect**" – повторять градиент шаблона от начала до конца, затем от конца до начала, затем снова от начала до конца до тех пор, пока не заполнится весь прямоугольник и «**repeat**» – повторять градиент шаблона от начала до конца до тех пор, пока не заполнится весь прямоугольник.

В элементе может быть также задан атрибут **xlink:href** для ссылки на другой элемент градиента.

Окончание вывода цветов, используемых в градиенте, определяется элементами **stop**, являющихся дочерними элементами элементов линейного или радиального градиентов. Для этого элемента можно задавать атрибуты модулей **Core.attrib**, **Style.attrib**, **Color.attrib** и **Gradient.attrib**. Кроме этого, для элемента **stop** должен быть задан атрибут **offset**, который указывает, где будет заканчиваться градиент. Значение этого атрибута может быть либо числом (обычно в диапазоне от 0 до 1) или в процентах (от 0% до 100%). Для линейного градиента значение этого атрибута указывает на положение вдоль вектора градиента.

Для элементов градиента определен базовый интерфейс **SVGGradientElement**. Дочерними интерфейсами этого интерфейса является интерфейс **SVGLinearGradientElement** для элемента **linearGradient** и интерфейс **SVGStopElement** для элемента **stop**.

### Радиальный градиент

Радиальный градиент определяется с помощью элемента **radialGradient**. Для этого элемента можно задавать атрибуты модулей **Core.attrib**, **Style.attrib**, **Color.attrib**, **Gradient.attrib**, **XLink.attrib** и **External.attrib**, а также определенные для элемента **linearGradient** атрибуты **gradientUnits**, **gradientTransform** и **spreadMethod**. Кроме этого, для элемента могут быть заданы атрибуты **cx**, **cy**, **r**, **fx** и **fy**.

Атрибуты **cx**, **cy** и **r** определяют координаты x и y центра и радиус наибольшего круга для радиального градиента.

Атрибуты **fx** и **fy** определяют координаты x и y фокальной точки радиального градиента. Градиент выводится таким образом, что 0% окончания градиента соответствует фокальной точке. Если атрибуты не заданы, их значения совпадают соответственно со значениями **cx** и **cy**.

Для элемента **radialGradient** определен интерфейс **SVGRadialGradientElement**.

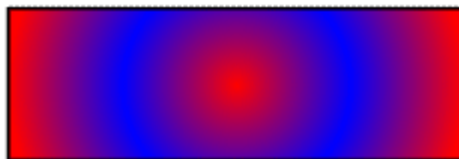
Элемент **stop** для радиального градиента определяет в атрибуте **offset** расстояние (в процентах) от фокальной точки до края самого большого круга.

### Пример использования радиального градиента:

```
<defs>
<radialGradient id="MyGradient"
gradientUnits="userSpaceOnUse"
cx="400" cy="200" r="300" fx="400" fy="200">
<stop offset="0%" stop-color="red" />
<stop offset="50%" stop-color="blue" />
<stop offset="100%" stop-color="red" />
</radialGradient>
</defs>
<rect fill="url(#MyGradient)" stroke="black"
stroke-width="5" x="100" y="100"
width="600" height="200"/>.
```



Выводимый прямоугольник будет иметь следующий вид:










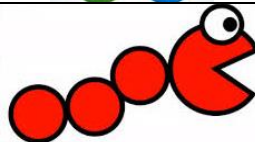
### Индивидуальное задание:





















#### Задание 1.






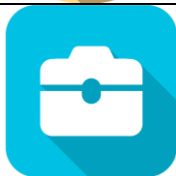











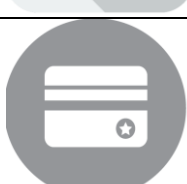

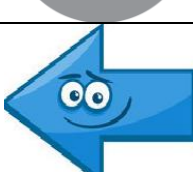
Создайте файл SVG с цветным изображением с помощью текстового редактора, **не пользуясь программами для построения изображений** (см. вариант по таблице).

#### Задание 2.

Загрузите редактор Inkscape, пакет Sketsa SVG Editor (<http://www.kiyut.com/products/sketsa/download.php>), или любой другой редактор векторного типа и установите его на своем компьютере (можно воспользоваться онлайн-редактором) и создайте SVG-изображение, используя возможности данной программы. Просмотрите получившийся код (меню «Окно»-> Source Editor). Встройте изображение в HTML-страницу различными способами.

№	Базовая фигура	Заливка базовой фигуры	Фигура для редактора
	<b>Задание 1</b>		<b>Задание 2</b>
1		цвет	
2		Градиент радиальный	
3		Градиент линейный	
4		цвет	

5		цвет	
6		Градиент ради- альный	
7		Градиент линей- ный	
8		Градиент линей- ный	
9		текстура	
10		текстура	
11		цвет	
12		Градиент ради- альный	
13		цвет	
14		текстура	

15		Градиент линей- ный	
16		цвет	
17		Градиент ради- альный	
18		цвет	
19		текстура	
20		Градиент линей- ный	
21		Градиент ради- альный	
22		цвет	
23		текстура	
24		текстура	

## СПИСОК ЛИТЕРАТУРЫ

1. Хеслоп, Б. HTML с самого начала / Б. Хеслоп, Л. Бадник ; пер. с англ. А. Малышева. - Санкт-Петербург : Питер, 1997. - 416 с. : ил. - (Для пользователей Windows). - Перевод изд.: HTML/Heslop B., Budnick L.
2. Дронов, В. А. JavaScript в Web-дизайне : практическое руководство / В. А. Дронов. - Санкт-Петербург : БХВ-Петербург, 2001. - 880 с. : ил. - (Мастер). - ISBN 5-94157-059-7.
3. Дронов, В. А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов [Электронный ресурс] / В. А. Дронов. - Электрон. дан. (1 файл : 11 Мб). - Санкт-Петербург : БХВ-Петербург, 2011. - (Профессиональное программирование). - Систем. требования: Acrobat Reader.
4. Хольцнер, С. Dynamic HTML : Руководство разработчика / С. Хольцнер ; пер. с англ. С. А. Бойко. - Киев : BHV, 1999. - 400 с. : ил. - Перевод изд.: Web Developer.Com Guide to Dynamic HTML/S.Holzner.
5. Климов, А. П. JavaScript на примерах / А. П. Климов. - 2-е изд., перераб. и доп. - Санкт-Петербург : БХВ-Петербург, 2009. - 336 с. : ил. - ISBN 978-5-9775-0361
6. HTML, XHTML и CSS: библия пользователя / Б. Пфаффенберг [и др.] ; пер. с англ. Л. М. Ильичевой, А. П. Сергеева ; под ред. А. П. Сергеева. - 3-е изд. - Москва : Диалектика, 2007. - 752 с. : ил. - Перевод изд.: HTML, XHTML and CSS Bible/B. Pfaffenberger et al. - ISBN 978-5-8459-1154-4.
7. Муссиано, Ч. HTML и XHTML [Электронный ресурс] : подробное руководство / Ч. Муссиано, Б. Кеннеди ; пер. с англ. С. Иноземцева ; гл. ред. А. Галунов. - 6-е изд. - Электрон. дан. ( 1 файл : 7 Мб). - Санкт-Петербург : Символ-Плюс, 2008. - (O'Reilly). - Систем. требования: Acrobat Reader.
8. Шмитт, К. HTML5. Рецепты программирования [Электронный ресурс] : готовые решения для веб-разработчиков / К. Шмитт, К. Симпсон ; пер. с англ. А. Лузган, Е. Шикарева. - Электрон. дан. (1 файл : 6 Мб). - Санкт-Петербург : Питер, 2012. - (O'Reilly). - Систем. требования: Acrobat Reader.
9. Петюшкин, А. В. HTML в Web-дизайне / А. В. Петюшкин. - Санкт-Петербург : БХВ-Петербург, 2005. - 400 с. : ил. - (PRO. Профессиональное программирование). - ISBN 5-94157-513-0.

