МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ
Колледж Пятигорского института (филиал) СКФУ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

ПМ.05 ВЕБ ТЕХНОЛОГИИ И ЗАЩИТА ИНФОРМАЦИИ

МДК.05.04 ВЕБ ПРОГРАММИРОВАНИЕ

Специальность СПО
09.02.01Компьютерные системы и комплексы
Квалификация специалист по компьютерным системам

Пятигорск, 2026

Методические указания для лабораторных работ по дисциплине МДК.05.04 Веб программирование составлены в соответствии с требованиями ФГОС СПО. Предназначены для студентов, обучающихся по специальности 09.02.01 Компьютерные системы и комплексы.

Пояснительная записка

Данные методические указания предназначены для закрепления теоретических знаний и приобретения необходимых практических навыков и умений по программе дисциплины «Веб программирование» для специальности СПО 09.02.01 Компьютерные системы и комплексы.

В результате освоения учебной дисциплины обучающийся должен

знать:

- технологии создания веб-сайта;
- теорию использования графики на веб-страницах;
- методы обработки и редактирования цифровых изображений;
- состав, структуру, принципы реализации и функционирования технологии клиент - сервер;
- программные средства, используемые для размещения и сопровождения веб-страниц.

уметь:

- проектировать структуру веб-ресурса;
- разрабатывать систему навигации по веб-ресурсу;
- разрабатывать статичные веб страницы используя языки разметки веб-страниц;
- разрабатывать стилевое оформление веб ресурса на основе CSS;
- использовать графические программы для создания веб-сайта;
- использовать графические редакторы для обработки изображений, размещаемых на веб-сайте;
- использовать язык гипертекстовой разметки HTML и каскадные таблицы стилей CSS для создания веб-страниц;

иметь практический опыт:

- создания веб-сайтов с использованием различных технологий и программ;
- поддержки и сопровождения веб-сайтов при загрузке их на сервер и подключении домена;
- модернизации и устранения ошибок в результате переноса веб-сайта с одного домена на другой.

Лабораторная работа №1. Структура и история языка гипертекстовой разметки.

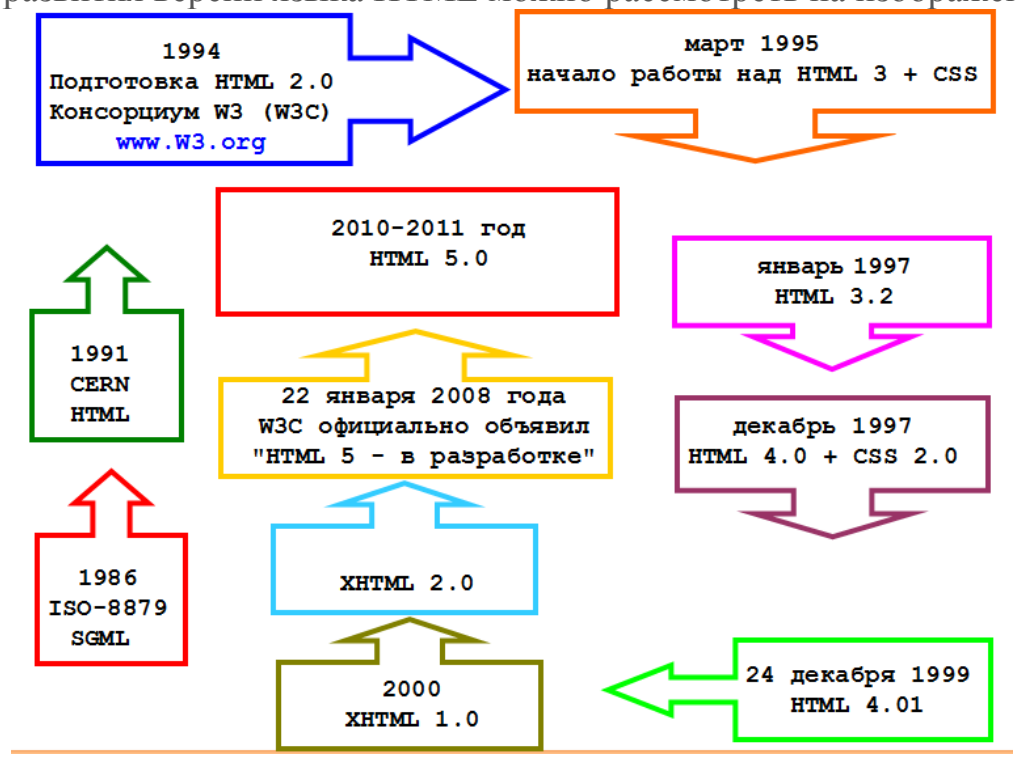
Цель: изучить способы создания простейших веб страниц.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

История версий html

Историю развития версий языка HTML можно рассмотреть на изображении:



Структурные элементы языка

- Единицей языка HTML является **тег** - команда, заключенная в угловые скобки.
- Теги бывают **открывающие** (запускающие действие команды) и **закрывающие** (останавливающие действие команды, соответственно).
- Закрывающий тег отличается наличием прямого слэша /.
- Команды тегов распространяются на их **содержимое**:

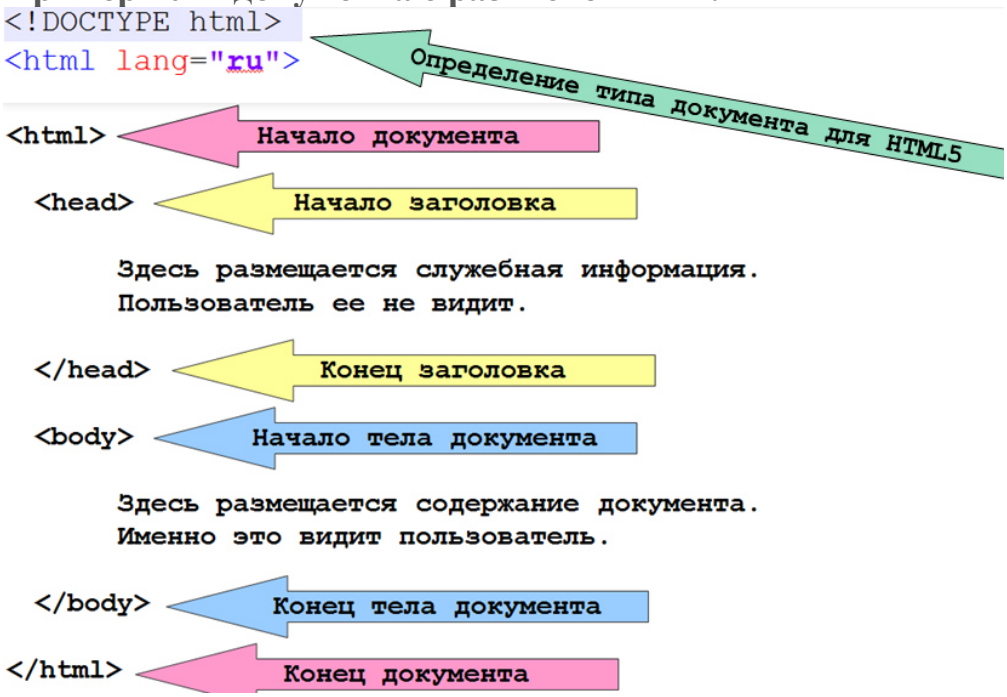
- Целиком пара открывающий-закрывающий тег называется **элементом** или контейнером.

- Помимо элементов, которые содержат и открывающий и закрывающий тег, есть **пустые элементы**, в которых содержание как бы отсутствует, и, соответственно у них нет закрывающего тега. Хотя на самом деле их содержанием является сам открывающий тег.

Пример пустых элементов:

```
<img/>
<br/>
```

Пример html-документа с разъяснениями:



Еще пример HTML-страницы:

```
   <html>
1  <head>
2
3  ...Служебная информация...
4
5  </head>
6  <body>
7  <h1>Мой первый HTML документ</h1>
8  <hr>     <!-- горизонтальная линия -->
9  <p>Некоторый текст. Основное содержание текущей страницы. Первый абзац
10 <p>Второй абзац. Для форматирования текста используют разные
11 элементы языка HTML.</p>    <!–- абзац -->
12 </body>
   </html>
```

На языке html **комментарии** ставятся при помощи символов

```
<!-- содержание комментария строчного-->
<!-- содержание
        комментария
            блочного-->
```

Такой комментарий может быть как строчным, т.е. занимать одну строку документа, так и блочным, занимающим несколько строк.

Важно: Для того, чтобы не было проблем с кодировкой символов, необходимо указать тип кодировки в области head (т.е. после открывающего тега head - головная часть документа)

```
<head>
    <meta charset="utf-8">
    <title>Заголовок окна</title>
</head>
```

Элементы могут быть **вложены друг в друга**, тогда это выглядит примерно вот так:

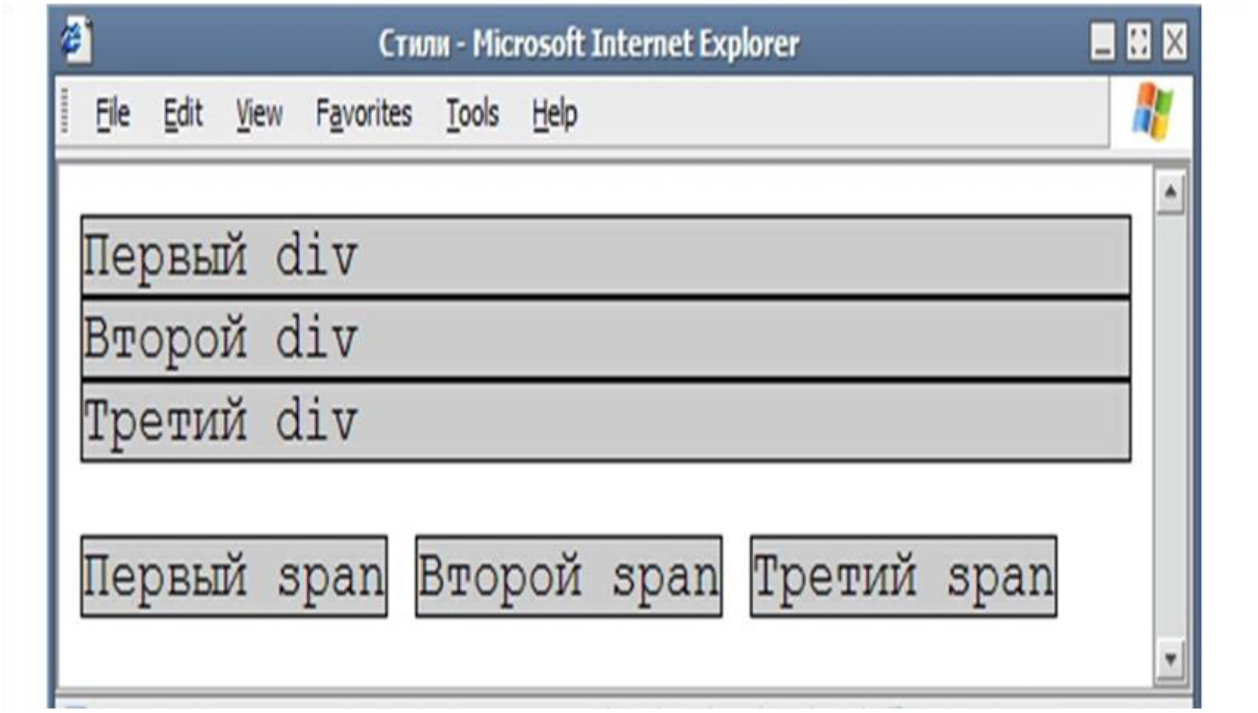
<**p**>Это <**em**>очень</**em**> интересно</**p**>

Схематично родительские теги и вложенные (дочерние) выглядят так:

Блочные и строчные элементы

Все элементы языка html делятся на **строчные (inline)** и **блочные (block)**. Строчные элементы отличаются тем, что они позволяют разместиться на «своей» строке следующим за ними элементам (позволяют «встать» рядом).

Тогда как блочные элементы займут всю строку, не «пуская» на нее следующие за ними элементы. Схематично это выглядит так:

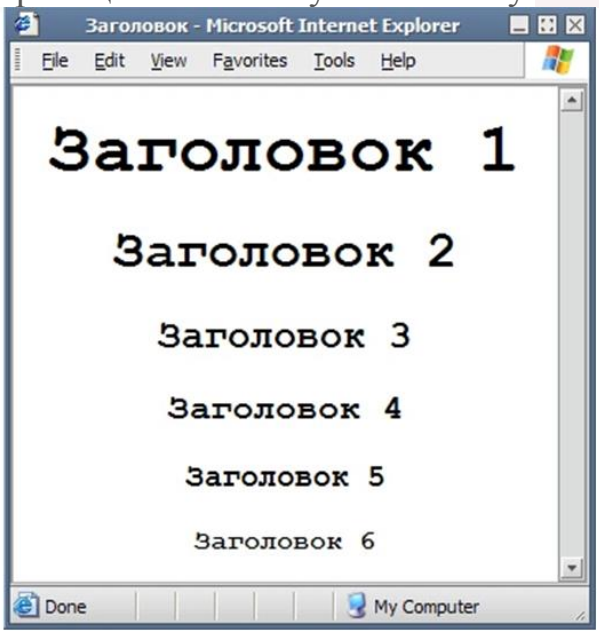


Элементы форматирования текста

ЗАГОЛОВКИ

- Для размещения заголовков существует тег <h> с номером уровня заголовка.
  <h1></h1>
- Самый крупный заголовок соответствует тегу <h1>, соответственно заголовок самого низкого уровня (самый мелкий размер шрифта) - <h6>.
- Базовый размер шрифта на странице соответствует заголовку <h3>:

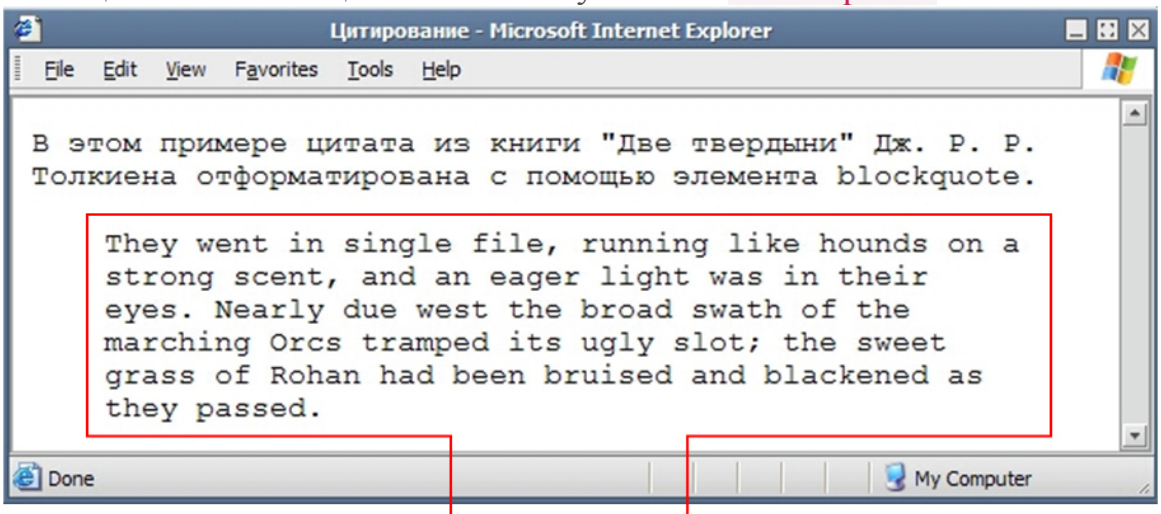
```
<h1>Заголовок 1</h1>
<h2>Заголовок 2</h2>
<h3>Заголовок 3</h3>
<h4>Заголовок 4</h4>
<h5>Заголовок 5</h5>
<h6>Заголовок 6</h6>
```



БЛОЧНАЯ ЦИТАТА

<blockquote></blockquote>

Для размещения в тексте цитаты используется тег <blockquote>:



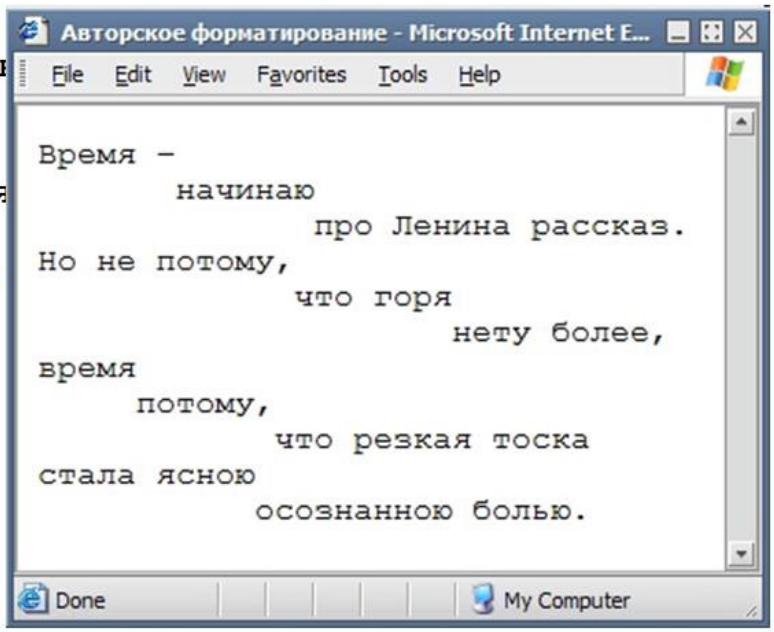

```
<blockquote>
They went in single file, running like hounds on a
strong scent, and an eager light was in their eyes.
Nearly due west the broad swath of the marching Orcs
tramped its ugly slot; the sweet grass of Rohan had
been bruised and blackened as they passed.</blockquote>
```

ПРЕФОРМАТИРОВАННЫЙ ТЕКСТ

<pre></pre>

Для того, чтобы сохранить в тексте все пробельные символы, необходимо использовать тег <pre>. Но при этом следует учесть, что для содержимого данного тега невозможно задать стиль шрифта:

```
<pre>
Время -
        начинаю
                про Ленина рассказ.
Но не потому,
              что горя
                              нету более,
время
     потому,
              что резкая тоска
стала ясною
            осознанною болью.
</pre>
```



КУРСИВ, ЖИРНОСТЬ, ПОДЧЕРКИВАНИЕ И ДРУГИЕ ТЕГИ

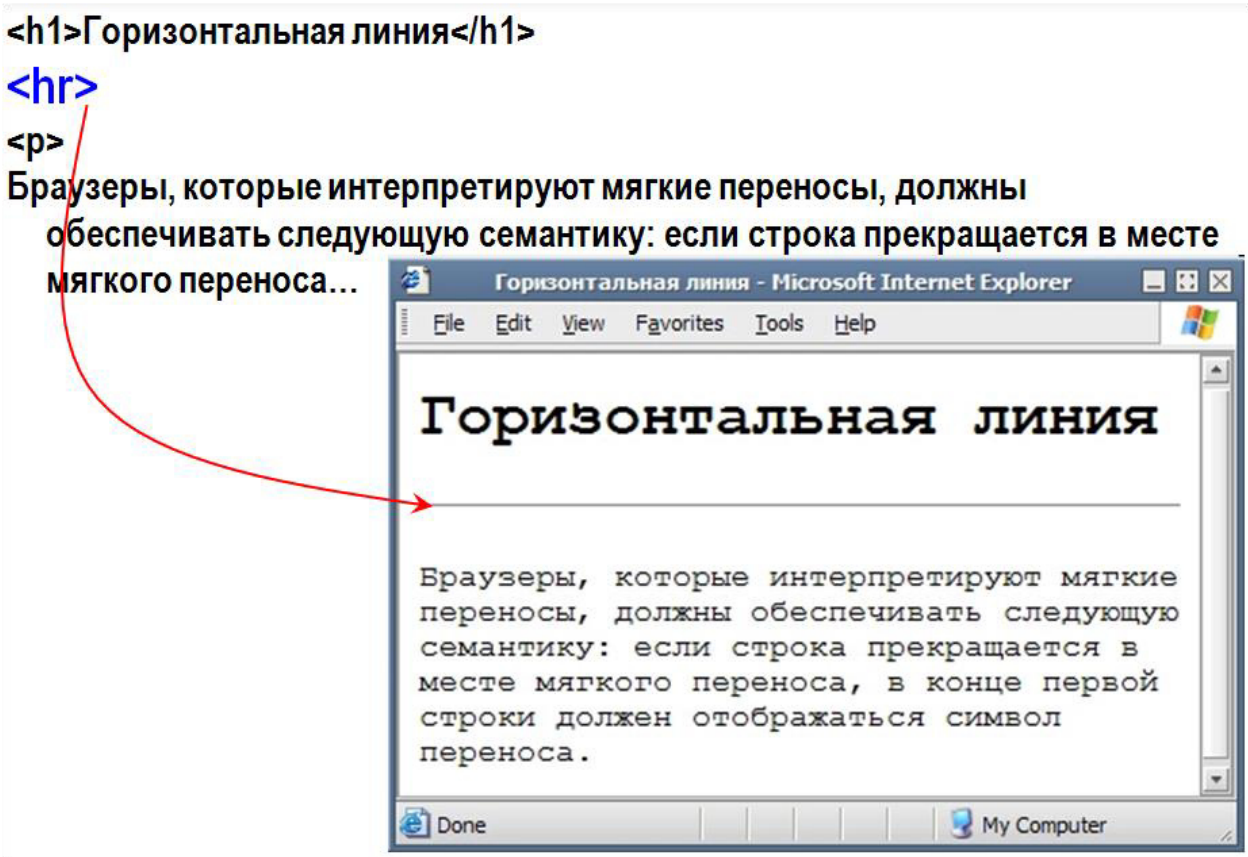
```
<i> - курсив
<b> - полужирный
<u> - подчеркнутый
<strike> - перечеркнутый
<tt> - моноширинный
<big> - увеличить шрифт
<small> - уменьшить шрифт
<sup> - надиндекс
<sub> - подиндекс
```

Пример текста тэга <i>
Пример текста тэга <b>
Пример текста тэга <u>
~~Пример текста тэга <strike>~~
Пример текста тэга <tt>
Пример текста тэга <big>
Пример текста тэга <small>
Сколько будет 2^3?
Формула воды: H_2O

ГОРИЗОНТАЛЬНАЯ ЛИНИЯ

<hr></hr>

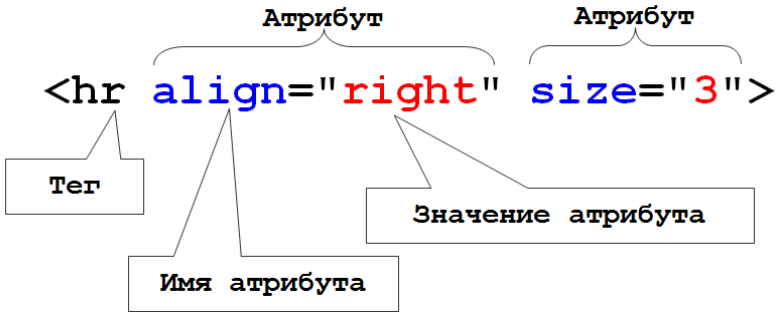
Данный элемент служит для разделения некоторых структурных элементов текста друг от друга. Либо может быть использован просто как эстетический элемент оформления документа:



Атрибуты тегов

- Для уточнения действия некоторых тегов они дополняются **атрибутами**.
- Так, у рассмотренного тега горизонтальной линии есть дополнительные свойства, выраженные в атрибутах
  - ***size*** - ширина линии,
  - ***width*** - длина линии,
  - ***align*** - выравнивание линии

  и другие.



- Атрибуты указываются в открывающем теге в виде ***атрибут=значение***.

- Атрибутов может быть несколько, тогда они указываются через пробелы, и их порядок следования практически не важен.

```
<hr size="3">
<hr color="red">
<hr noshade>
<hr align="right">
<hr width="450">
<hr size="3" width="50%" align="center">
```

АТРИБУТЫ ТЕГА BODY

Для начала рассмотрим два основных атрибута тега body:

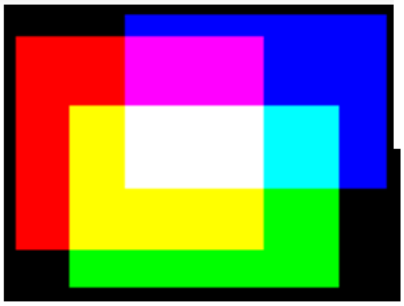
- ***bgcolor*** - задний фон страницы и
- ***text*** - цвет текста на всей странице.

Для задания цвета можно использовать названия цветов на английском языке, либо код цвета в шестнадцатеричной системе счисления.

```
<body text="#00ff00">
```

или

```
<body text="green">
```



```
Шестнадцатеричная система
счисления:
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Диапазон: 00 - FF (0 - 255)
#00FF00 - зеленый.
#FF0000 - красный.
#0000FF - синий.
#FFFFFF - белый.
#000000 - черный.
```

Перед указанием цвета в 16-й системе обязательно ставится символ *«шарп»* - #
Для подбора подходящего цвета перейдите на страницу палитры цветов онлайн.

Теги логического форматирования текста

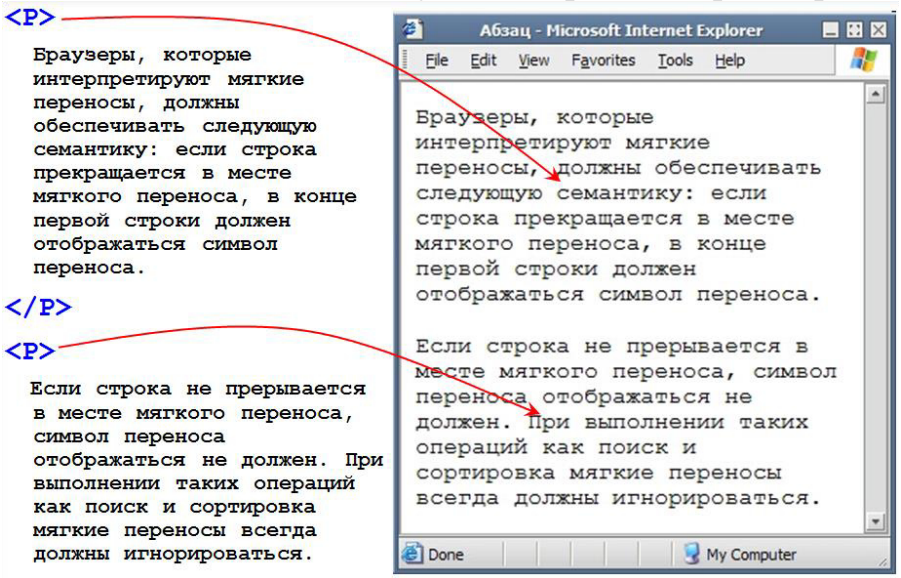
В html есть теги, которые несут больше не эстетическую нагрузку, а логическую или смысловую нагрузку. Т.е. большинство из них внешне влияет на содержимое, делая его подчеркнутым или выделяя каким-либо другим образом. Но созданы эти теги логического форматирования **с целью выделить смысловую характеристику содержимого**:

Например, тег ***del*** делает содержимое перечеркнутым, при этом указывая прежде всего на смысл удаления текста.

 - выделение важных фрагментов курсивом
 - выделение особо важных фрагментов полужирным
<ins> - выделение фрагмента подчеркиванием, когда требуется показать явно, что текст был вставлен после опубликования документа.
<del> - выделение фрагмента перечеркиванием, когда требуется показать явно, что текст был удален после опубликования документа.
<cite> - выделение цитат курсивом
<code> - отображение фрагментов программного кода моноширинным шрифтом
<kbd> - текст, вводимый с клавиатуры: отображается моноширинным шрифтом
<var> - название переменных: отображается курсивом
<samp> - выделение нескольких символов моноширинным шрифтом
<dfn> - определение вложенного термина курсивом
<abbr title="Какое-то слово"> - аббревиатура
<acronym title="Какое-то слово"> - акроним
<q lang="ru"> - определение кавычек

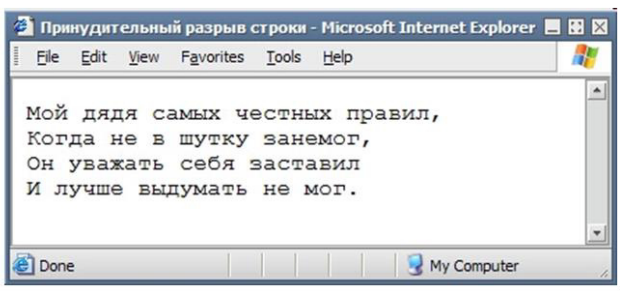
Элементы форматирования абзацев

- Для перехода на другую строку текста служит пустой элемент ***
***.
- Тогда как для выделения в тексте абзаца служит элемент ***<p>***, содержимое которого и является сам абзац. Перед абзацем и после него добавляются отступы, но красная строка при этом не предусмотрена.



**Пример совместного использования тегов
 и <p>:**

```
<p>Мой дядя самых честных правил,
<br>Когда не в шутку занемог,
<br>Он уважать себя заставил
<br>И лучше выдумать не мог.
```



ЦВЕТ И ГАРНИТУРА ШРИФТА

Для форматирования шрифта существует тег . Однако, тег уже практически не используется.

Тег используется только со своими атрибутами:

- size - размер шрифта, от 1 до 7 (3 - базовый размер, 6 - размер заголовка H1)
- face - семейство шрифта (serif - с засечками, sans-serif - рубленый или без засечек, monospace - моноширинный)
- color - цвет

Пример:

```
<font size="4" color="ff0000" face="Arial, Verdana, sans-serif">
Текст красного цвета, шрифт без засечек
</font>
```

Результат а браузере:

Текст красного цвета, шрифт без засечек

СПЕЦИАЛЬНЫЕ СИМВОЛЫ

	код html	
©	©	Копирайт
®	®	Знак зарегистрированной торговой марки
™	™	Знак торговой марки
		Мягкий перенос
×	×	Умножить
÷	÷	Разделить
±	±	Плюс/минус
≤	≤	Меньше или равно
≥	≥	Больше или равно

Вопросы для самоконтроля

1. Развитие HTML.
2. Развитие Веб-серверов.
3. Развитие языков Веб программирования.
4. Развитие мультимедийных платформ.

Лабораторная работа №2. Вставка изображения.

Цель: изучить способы добавление картинок.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Размещение изображения в HTML

Форматы изображений для размещения на сайте: *.gif*, *.png-8*, *.png-24*, *.png-32* и *.jpg* (в случае необходимости размещения качественного фото)

Синтаксис:

<**img** src=«имя_файла»>

img - строчный элемент с замещаемым контентом

Пример: разместить на странице:

- изображение *prob.gif*, файл которого располагается в папке со странницей,
- изображение *banner.gif*, файл которого располагается в папке на уровень выше текущей страницы (необходимо выйти из папки),
- изображение с сайта *http://www.rambler.ru/ban.jpg*

Выполнение:

```
<html>
...
<body>
  <p><img src="prob.gif">
  <p><img src="../banner.gif">
  <p><img src="http://www.rambler.ru/ban.jpg">
</body></html>
```

АТРИБУТЫ ТЕГА IMG

- alt - подпись
- title - всплывающая подпись

Выравнивание по вертикали:

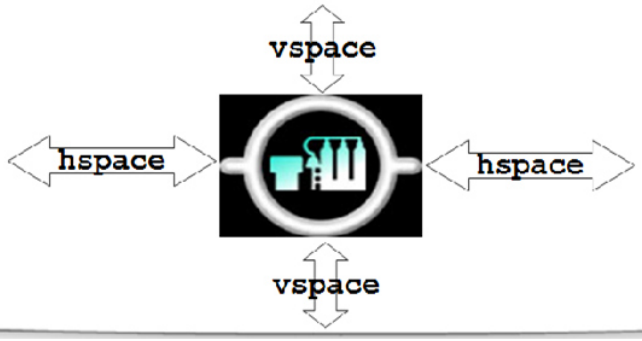
- align="top"
- align="middle"
- align="bottom"

Выравнивание по горизонтали:

- align="left"
- align="right"
- width - ширина (значение в пикселях)
- height - высота (значение в пикселях)
- border - рамка (значение 0 или 1)

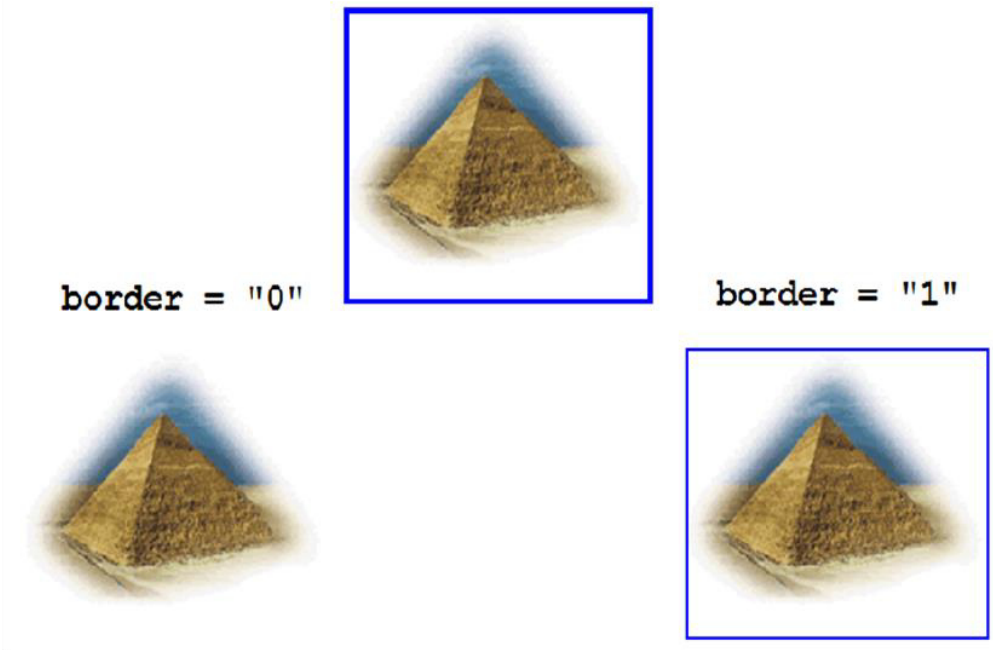
```
<img src=img.gif width="100" height="150">
<img src=img.gif border="3">
<img src=img.gif align="top">

top - вертикальное выравнивание по верхнему краю.
middle - вертикальное выравнивание по центру.
bottom - вертикальное выравнивание по нижнему краю.
left - горизонтальное выравнивание по левому.
right - горизонтальное выравнивание по правому краю.
```



ИЗОБРАЖЕНИЕ КАК ССЫЛКА

```
<a href="ссылка"> <img src="img.gif"> </a>
```



ФОНОВОЕ ИЗОБРАЖЕНИЕ СТРАНИЦЫ

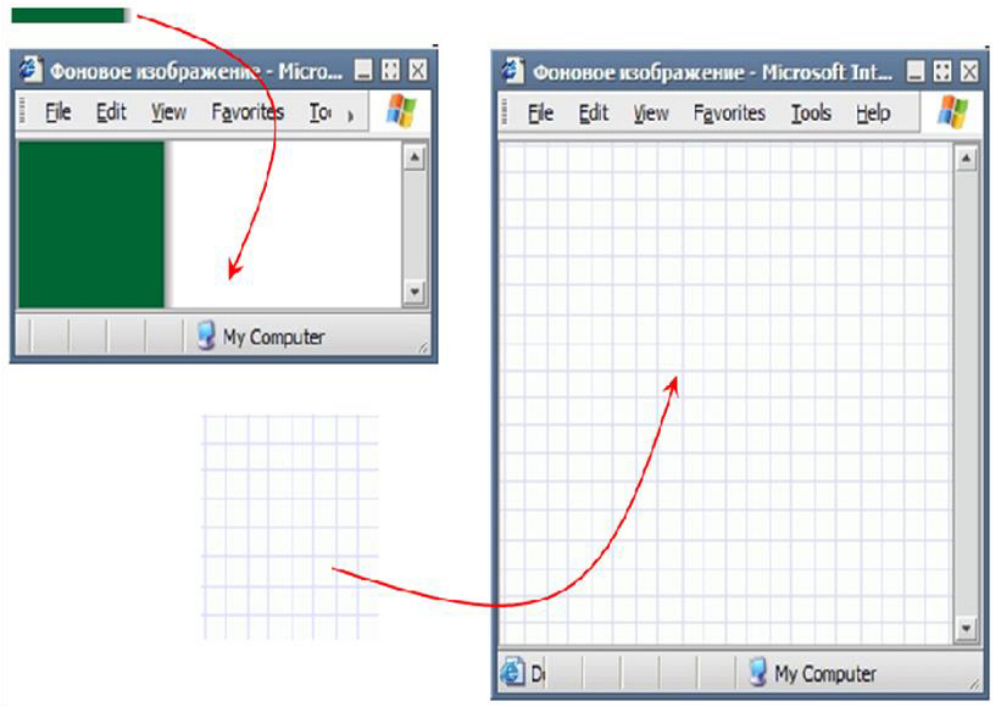
Синтаксис:

```
<body background="fon.gif">
```

Изображение будет растиражировано по всей странице.

Атрибут bgproperties со значением fixed позволит сделать задний фон статичным во время использования прокрутки страницы.

```
<body background="fon.gif">
<body background="fon.gif" bgproperties="fixed">
```



Вопросы для самоконтроля

1. Основные теги.
2. Структура страницы.
3. Картинки.
4. Форматирование текста.

Лабораторная работа №3. Создание таблиц.

Цель: изучить способы добавления таблиц.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Создание таблицы в HTML

Рассмотрим теги для создания таблицы:

```
<table>
        <tr>
                <td> содержание </td>
        </tr>
</table>
```

Результат:

содержание

Добавим границу для таблицы - атрибут border:

```
<table border="1">
        <tr>
                <td> содержание </td>
        </tr>
</table>
```

Результат:

содержание

Создания таблицы начинается с тега table (от англ. «таблица»). Тег tr служит для создания строки. В строке располагаются ячейки - тег td. Завершается таблица закрывающим тегом /table

<table>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
</table>

Или пример таблицы посложнее:

```
<table>
  <tr> <!-- Первая строка -->
    <td>(1,1)</td> <!-- Первая ячейка -->
    <td>(1,2)</td> <!-- Вторая ячейка -->
</tr>
  <tr> <!-- Вторая строка -->
    <td>(2,1)</td> <!-- Первая ячейка -->
    <td>(2,2)</td> <!-- Вторая ячейка -->
</tr>
  <tr> <!-- Третья строка -->
    <td>(3,1)</td> <!-- Первая ячейка -->
    <td>(3,2)</td> <!-- Вторая ячейка -->
</tr>
</table>
```

(1,1)	(1,2)
(2,1)	(2,2)
(3,1)	(3,2)

АТРИБУТЫ ТЕГА TABLE

align	left - таблица влево; center – табл. по центру; right - табл. вправо.
width	400 50%
border	ширина
bordercolor	цвета рамки
cellspacing	ширина грани рамки
cellpadding	внутреннее расстояние до рамки
bgcolor	Цвет #rrggbb
background	файл (фон таблицы)

Важно: Для ячеек-заголовка таблицы используется тег th вместо td. Браузер размещает содержимое таких ячеек по центру и делает шрифт полужирным

АТРИБУТЫ ТЕГА TR - СТРОКИ

align	left, center, right	выравнивание по горизонтали
valign	top, middle, bottom, baseline	выравнивание по вертикали
bgcolor	цвет	задний фон
bordercolor	цвет	цвет границы

АТРИБУТЫ ТЕГА TD ИЛИ TH - ЯЧЕЙКИ

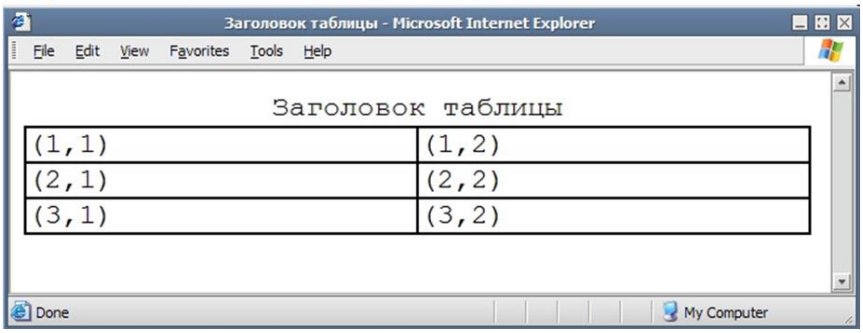
align	left, center, right	выравнивание по горизонтали
valign	top, middle, bottom, baseline	выравнивание по вертикали
width	число или процент	ширина ячейки
bgcolor	цвет	цвет фона
background	файл	файл фона
bordercolor	цвет	цвет границы
nowrap		заставляет текст внутри ячейки отображаться без переносов, одной сплошной строкой

Важно:

- Тег caption служит для создания заголовка таблицы
- Для группировки заголовочных ячеек используется тег thead
- Для группировки основного содержимого таблицы используется тег tbody
- Тег tfoot определяет нижнюю часть таблицы

Тег caption заголовка таблицы может иметь атрибут, определяющий расположение заголовка - align - со следующими значениями: top - заголовок над таблицей, bottom - заголовок под таблицей, left - заголовок вверху и выровнен влево, right - заголовок вверху и выровнен вправо. К сожалению не все значения «работают» во всех браузерах.

```
<table>
<caption align="top">Заголовок таблицы</caption>
...
```



Пример: Создать «каркас» таблицы со всеми тегами группировки

Выполнение:

```
<table  border="1">
<caption>таблица</caption>
<thead>
        <tr>
        <th>Заголовок 1</th><th>Заголовок 2</th>
        </tr>
</thead>
<tbody>
        <tr>
        <td>содержимое</td><td>содержимое</td>
        </tr>
</tbody>
<tfoot>
...
</tfoot>
</table>
```

Создайте таблицу по образцу. У таблицы должен быть заголовок и области для группировки (*thead* - 1-я строка таблицы, *tbody* - 2-я и 3-я строки таблицы, *tfoot* - 4-я строка таблицы).

Таблица с областями группировки			
Столбец 1	**Столбец 2**	**Столбец 3**	**Столбец 4**
Строка2 Ячейка1	Строка2 Ячейка2	Строка2 Ячейка3	Строка2 Ячейка4
Строка3 Ячейка1	Строка3 Ячейка2	Строка3 Ячейка3	Строка3 Ячейка4
Строка4 Ячейка1	Строка4 Ячейка2	Строка4 Ячейка3	Строка4 Ячейка4

Объединение ячеек в таблице

В происходит при помощи двух атрибутов тега td: COLSPAN - объединение ячеек по горизонтали, ROWSPAN - объединение ячеек по вертикали.

Синтаксис COLSPAN:

```
<table>
<tr>
  <td colspan="2"> </td>
</tr>
<tr>
  <td> </td>
  <td> </td>
</tr>
</table>
```

Синтаксис ROWSPAN:

```
<table>
<tr>
  <td rowspan="2"> </td>
  <td> </td>
</tr>
<tr>
  <td> </td>
</tr>
</table>
```

Пример: создать таблицу по образцу на картинке. Использовать слияние ячеек

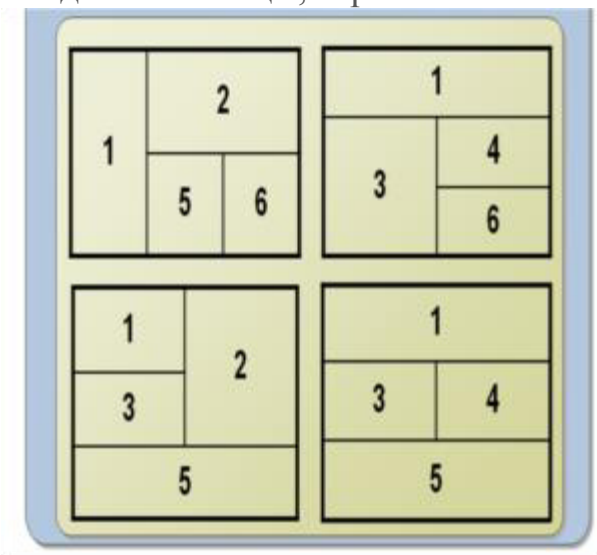
Таблица с объединенными ячейками

	Заголовок 1	
	Заголовок 1.1	Заголовок 1.2
Заголовок 2	Ячейка 1	Ячейка 2
Заголовок 3	Ячейка 3	Ячейка 4

Выполнение:

```
<table  border="1">
<caption>Таблица с объединенными ячейками </caption>
<tr>
  <th rowspan="2"> </th>
  <th colspan="2">Заголовок 1</th>
</tr>
<tr>
  <th> Заголовок 1.1</th>
  <th> Заголовок 1.2</th>
</tr>
<tr>
  <th> Заголовок 2</th>
  <td> Ячейка 1</td>
  <td> Ячейка 2</td>
</tr>
<tr>
  <th> Заголовок 3</th>
  <td> Ячейка 3</td>
  <td> Ячейка 4</td>
</tr>
</table>
```

Создайте таблицы, с расположенными в их ячейках цифрами, точно по образцу:

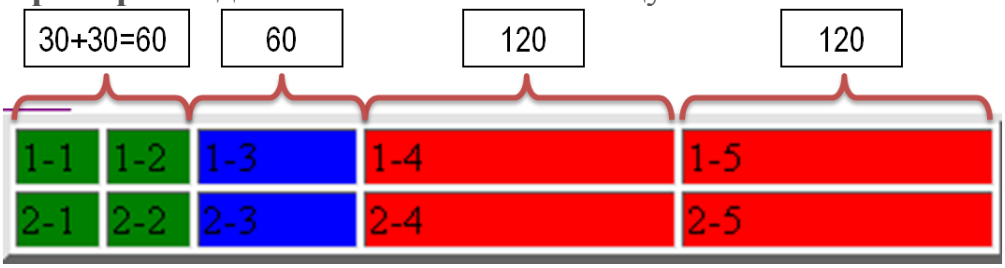


Группировка ячеек: COLGROUP

Элемент colgroup позволяет создавать группы колонок с одинаковыми свойствами.

Рассмотрим на примере:

Пример: Создать таблицу по образцу



Выполнение:

```
<TABLE border="4">
<colgroup >
        <col span="2" width="30" style="background-color: green" />
        <col width="60" style="background-color: blue" />
</colgroup>
<colgroup style="background-color:red">
        <col span=2 width="120"/>
</colgroup>
<TR>
<TD> 1-1 </TD><TD> 1-2 </TD><TD> 1-3 </TD><TD> 1-4 </TD><TD> 1-5</TD>
</TR>
<TR>
<TD> 2-1 </TD><TD> 2-2 </TD><TD> 2-3 </TD><TD> 2-4 </TD><TD> 2-5 </TD>
</TR>
</table>
```

АТРИБУТЫ ТЕГА COLGROUP

align	выравнивание содержимого столбца 1. left - по левому краю (по умолчанию) 2. center - по центру 3. right - по правому краю не работает в Firefox
dir	определяет направление символов: 1. ltr - слева направо 2. rtl - справа налево не работает в Firefox
span	количество столбцов, к которым будет применяться оформление (по умолчанию 1)
style	задает встроенную таблицу стилей
valign	вертикальное выравнивание в ячейке таблицы 1. bottom - по нижнему краю ячейки 2. middle - по центру ячейки (по умолчанию) 3. top - по верхнему краю ячейки не работает в Firefox
width	ширина столбца

HTML вложенные таблицы

Таблицы со сложной структурой проще заменять на вложенные таблицы.

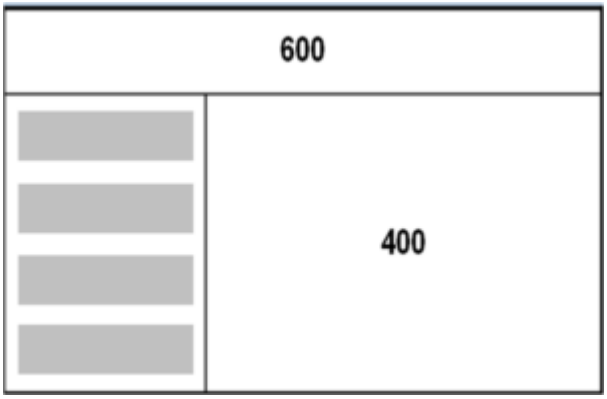
Пример: создайте вложенные таблицы по образцу:

Выполнение:

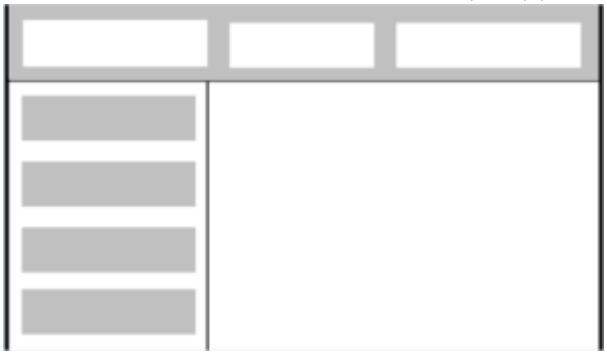
```
<TABLE border="4" bgcolor="yellow">
<tr>
  <td>Таблица 1</td>
  <td>
    <TABLE>
    <tr>  <td>Таблица 2</td><td>Ячейка 2-2</td>  </tr>
    <tr>  <td>Ячейка 3-2</td><td>Ячейка 4-2</td> </tr>
    </TABLE>
  </td>
<tr>
  <td>Ячейка 3-1</td>
  <td>Ячейка 4-1</td>
</tr>
</TABLE>
```

Создайте таблицу с фиксированными размерами, содержащую ячейки указанной на рисунке ширины

Вставьте в левую нижнюю ячейку вложенную таблицу
Фон ячеек вложенной таблицы сделайте серым



Откройте задание, выполненное на прошлой лабораторной работе
Добавьте в верхнюю ячейку еще одну вложенную таблицу
Фон ячеек вложенной таблицы сделайте белым



Вопросы для самоконтроля

1. HTML ссылки.
2. Структура ссылки.
3. Абсолютный и относительный путь.
4. Якоря.
5. Изображение ссылка.
6. Атрибуты ссылок.

Лабораторная работа №4. Вставка видео и аудио на сайт.

Цель: изучить способы добавления аудио и видео файлов на сайт.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Вставка аудио

Форматы аудио-файлов:

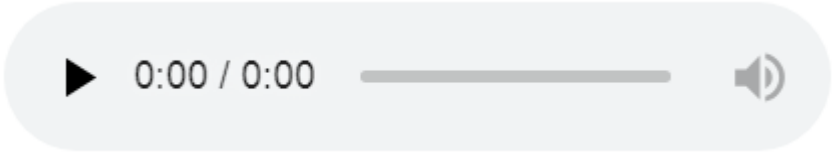
- mp3
- wav
- ogg

Для вставки аудио-плеера используется следующий код:

```
<audio controls="controls">
    <source src="song.ogg" type="audio/ogg">
    <source src="song.mp3" type="audio/mpeg">
</audio>
```

```
<audio controls="controls">
        <source src="song.ogg" type="audio/ogg">
        <source src="song.mp3" type="audio/mpeg">
</audio>
```

В браузере Google Chrome плеер будет выглядеть:



Атрибуты тега <audio> для плеера:

Атрибут	Значение	Описание
autoplay	autoplay	Указывает, что аудио должен начать играть, как только будет готов
controls	controls	Указывает, что элементы управления воспроизведением должны отображаться
loop	loop	Указывает, что аудио должно начаться снова, когда оно будет закончено
preload	auto metadata none	Определяет, должно ли аудио быть загруженным при загрузке страницы
src	url	Указывает адрес аудио для проигрывания

Другие возможности вставки аудио на сайт

1. `<a href="имя_файла.mp3">Щелкни </a>`
2.

```
<bgsound src="04.wav" loop="5">
```
*только для форматов: wav, mp3, aiff, wma

3.
```
<embed src="имя_файла.wav" height="150" width="180">
```

Вставка видео

Формат видео-файлов:

- MP4
- WebM
- Ogg

```
<video width="320" height="240" controls="controls" poster="logo.png">
    <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
Ваш баузер не поддерживает video.
</video>
```

```
<video width="320" height="240" controls="controls" poster="logo.png">
        <source src="movie.mp4" type="video/mp4">
        <source src="movie.ogg" type="video/ogg">
Ваш браузер не поддерживает video.
</video>
```

Результат в браузере:

Результат выполнения кода выше:

Атрибуты тега <video> для плеера:

Атрибут	Значение	Описание
audio	muted	Определяет по умолчанию состояние звука. В настоящий момент только "muted" разрешено
autoplay	autoplay	Если указан, видео начнет играть сразу как только оно будет готово
controls	controls	Если указан, кнопки управления будут показаны, такие как

		кнопка воспроизведения
height	пиксели	Указывает высоту видео плеера
loop	loop	Если указан, видео начнет проигрываться снова, как только закончится
poster	url	Указывает URL изображения, представляющего видео
preload	auto metadata none	Если указан, видео будет загружено при загрузке страницы, и готово к запуску. Игнорируется, если "autoplay" указан
src	url	Адрес URL видео для проигрывания
width	пиксели	Указывает ширину видео плеера

Пример:

```
<video src="04.avi" loop="loop" audio="muted">
```

Другой вариант вставки видео (без плеера):

```
<a href="имя_файла.avi">Щелкни и смотри</a>
<!-- Пример: -->
<a href="ocean.qt"> Видеоклип 1 Мб</a>
```

* для форматов mpeg, avi

Фавикон Favicon

Фавиконка отображается в адресной строке браузера перед URL страницы, также Фавикон можно заметить во вкладке браузера страницы. Поисковые системы передают Favicon вместе с результатами поиска.

- файлы с расширением *.ico*
- размер 16×16 пикселей

Сервис для преобразования в ico-формат: http://image.online-convert.com/

```
<html>
<head>
<link rel="icon" href="favicon.ico" type="image/x-icon">
</head>
```

HTML 5: семантические теги

- Семантические теги обычно несут смысловую нагрузку и не имеют никакого внешнего оформления в html. Но их можно и нужно оформлять стилями CSS. Такие теги важны для удобства читаемости кода и влияют на выдачу поисковиков.
- Рассмотрим примеры семантических тегов и их использования:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Заголовок</title>
```

```
</head>
<body>
  <header>
    header элемент - здесь следует какая-то информация в заголовке сайта. Обычно это лого компании и иногда дополнительная навигация по сайту.
    <nav>nav (сокращенное от navigation) элемент - обычно представляет горизонтальное меню для навигации по отдельным частям сайта.</nav>
  </header>
  <h1>Главный заголовок страницы</h1>
  <section>
    Секция 1
    <article>Статья 1</article>
    <article>Статья 2</article>
    <article>Статья 3</article>
  </section>
  <section>
    Секция 2
    <article>Статья 4</article>
    <article>Статья 5</article>
    <article>Статья 6</article>
    <div>Обычный div, блочный элемент</div>
  </section>
  <aside>
    ASIDE - какая-то информация, имеющая отношение к теме страницы.
  </aside>

  <footer>
    labs-org.ru, Copyright 2020
  </footer>
</body>
</html>
```

Вопросы для самоконтроля

1. HTML формы.
2. Правила создания макета.

Лабораторная работа №5. Разработка сайта с нуля.

Цель: изучить способы разработки сайта.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

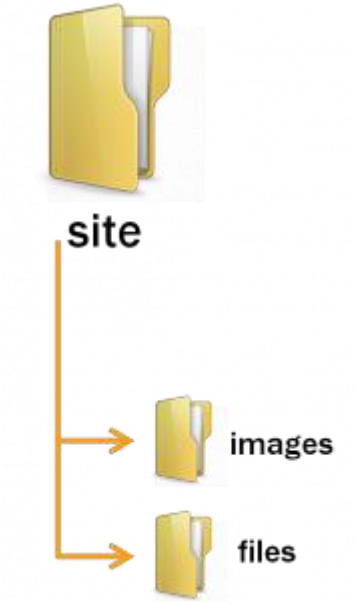
Ход работы:

Разработка структуры

Сегодня появилось много средств для разработки сайтов, это и специализированный порталы, предназначенные для конструирования сайта на основе предложенных шаблонов, и системы управления сайтами и другие подобные средства. Данные системы, естественно, облегчают процесс создания ресурса. Но в учебных целях необходимо научиться создавать сайт с нуля самостоятельно. Для этого выберем простую схему, при которой для организации интерфейса страниц сайта используется *фреймовая структура*.

Для начала необходимо подготовить каталог для работы и основные необходимые файлы.

Для работы с файлами рекомендуется использовать ПО *Notepad++*. Создайте каталог для работы (с названием *site*, например) и расположите в нем две папки согласно рисунку:

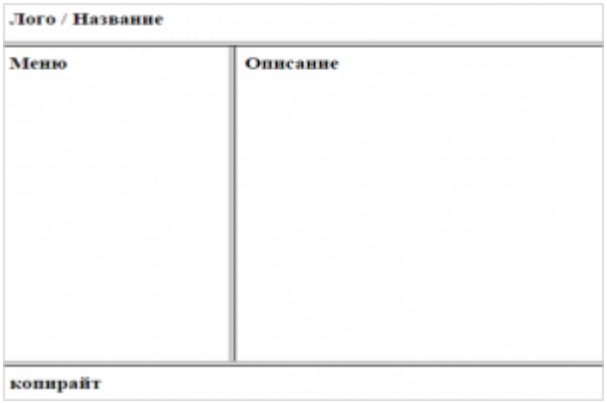


В основном каталоге создайте и сохраните html-файлы со следующими названиями:

- index.html
- header.html
- menu.html
- titul.html
- footer.html
- glava1.html

Откройте файл главный запускаемый файл - *index.html* (файл-раскладка) и добавьте код для организации фреймовой структуры:

```
<html>
<head>
        <title> Фреймы </title>
</head>
<frameset rows="10%,80%,*">
        <frame src="header.html" noresize>
        <frameset cols="20%,80%">
                <frame src="menu.html">
                <frame src="titul.html" name="main">
        </frameset>
        <frame src="footer.html" >
</frameset>
</html>
```



Затем необходимо добавить код в предназначенные для этого веб-страницы:

- header.html

```
<html>
<head>
</head>
<body>
<h1>Лого/Заголовок</h1>
</body>
</html>
```

- menu.html

```
<html>
<head>
</head>
<body>
<h1>Меню</h1>
</body>
</html>
```

- titul.html

```
<html>
<head>
</head>
```

```
<body>
<h1>Описание</h1>
</body>
</html>
```

- footer.html

```
<html>
<head>
</head>
<body>
<h1>Копирайт</h1>
</body>
</html>
```

- glava1.html

```
<html>
<head>
</head>
<body>
<h1>Глава 1</h1>
</body>
</html>
```

Организация работы гиперссылок в меню

Для правильной работы гиперссылок в файле *menu.html*, а именно, чтобы запускаемые файлы открывались именно в правом от меню фрейме, необходимо выполнить два пункта:

1. В файле фреймовой структуры добавить имя для фрейма, в котором мы собираемся открывать веб-страницы в меню:

   ```
   <frame src="titul.html" name="main">
   ```

   Это мы уже сделали при создании фреймовой структуры. Нужный нам фрейм называется у нас *main*.

2. В файле *menu.html* добавить код гиперссылок:

   ```
   <html>
   <head>
   </head>
   <body>
   <h1> Меню</h1>
   <table>
   <tr><td>
           <a href="glava1.html" target="main" rel="noopener noreferrer">Глава1 </a>
   </td></tr>
   <tr><td>
   ```

```
        <a href="glava2.html" target="main" rel="noopener noreferrer">Глава2 </a>
</td></tr>
<tr><td>
        <a href="glava3.html" target="main" rel="noopener noreferrer">Глава3 </a>
</td></tr>
</table>
</body>
```

В ссылках мы указали цель вывода страниц - фрейм с названием *main* - target="main".

Добавление CSS-стилей

Для оформления сайта, а в некоторых случаях и придания динамичности его элементов необходимо использовать каскадные таблицы стилей.

Сначала создайте документ для хранения стилей - файл *style.css*, и расположите его в папке - *files*.

Для подключения созданного стилевого файла ко всем страницам сайта, необходимо расположить ссылку на этот файл в код всех страниц, кроме файла с фреймовой структурой (*index.html*):

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

Вопросы для самоконтроля

1. Контентные модели.
2. Метаданные.
3. Потоковый и секционный контент.

Лабораторная работа №6. Добавление стилей.

Цель: изучить способы добавления стилей.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Повторение: блочные и строчные теги

- Для оформления и использования стилей css применяются блочные теги div и строчные теги span.
- Вспомним, как они позиционируются на странице и используются:
  Пример:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Элементы span и div</title>
</head>
<body>
  <div>*** DIV 1: Какой-то контент ***</div>
  <div>*** DIV 2: Следует прямо за div 1 ***</div>
  <span>*** SPAN 1: Следует прямо за div 2 ***</span>
  <div> *** DIV 3: Следует прямо за span 1
    <span>*** SPAN 2: ВНУТРИ div 3 ***</span>
    Продолжение содержимого div 3 ***
  </div>
</body>
</html>
```

Результат:

*** DIV 1: Какой-то контент ***
*** DIV 2: Следует прямо за div 1 ***
*** SPAN 1: Следует прямо за div 2 ***
*** DIV 3: Следует прямо за span 1 *** SPAN 2: ВНУТРИ div 3 *** Продолжение содержимого div 3 ***

- Обратим внимание, что **span 1** оказался на отдельной строке, только из-за того, что перед ним и после него находятся блочные теги **div**, которые не «пустили» его на свои строки.
- Тег span может размещаться внутри тега div, как в нашем примере span 2. Тогда как div никогда не может располагаться внутри тега span. **Блочный тег не может быть внутри строчного!**

CSS стили. Методы добавления

***CSS* - Cascading Style Sheets** (каскадные таблицы стилей) - это средство, позволяющее задавать различные визуальные свойства html-тегам

Рассмотрим основные методы добавления стилей:

- Встраивание (inline)
- Вложение (embeding)

- Связывание (linking)
- Импорт

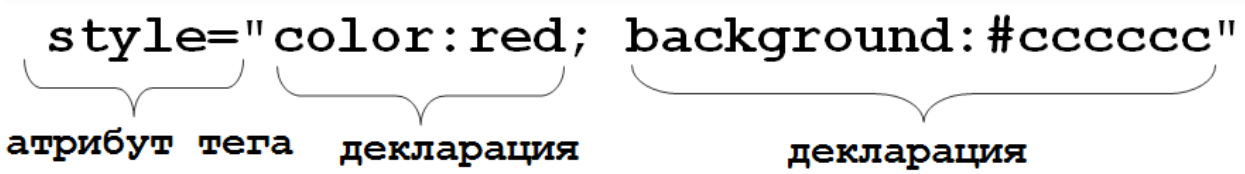
Метод встраивания (inline) в CSS

Метод встраивания (или строковый стиль) - это самый простой метод добавления стиля. Применяется в том случае, когда необходимо применить стиль только к одному единственному конкретному тегу и только один раз

Синтаксис:

<элемент style = "свойство1: значение; свойство2: значение; . . .">;

Рассмотрим основные понятия, встречающиеся при использовании стилей.



Пример: Первый абзац без стилей, ко второму абзацу применен стиль

```
<p>Обычный текст</p>
<p style="color:red; background:#cccccc">Абзац с методом встраивания</p>
```

Результат:

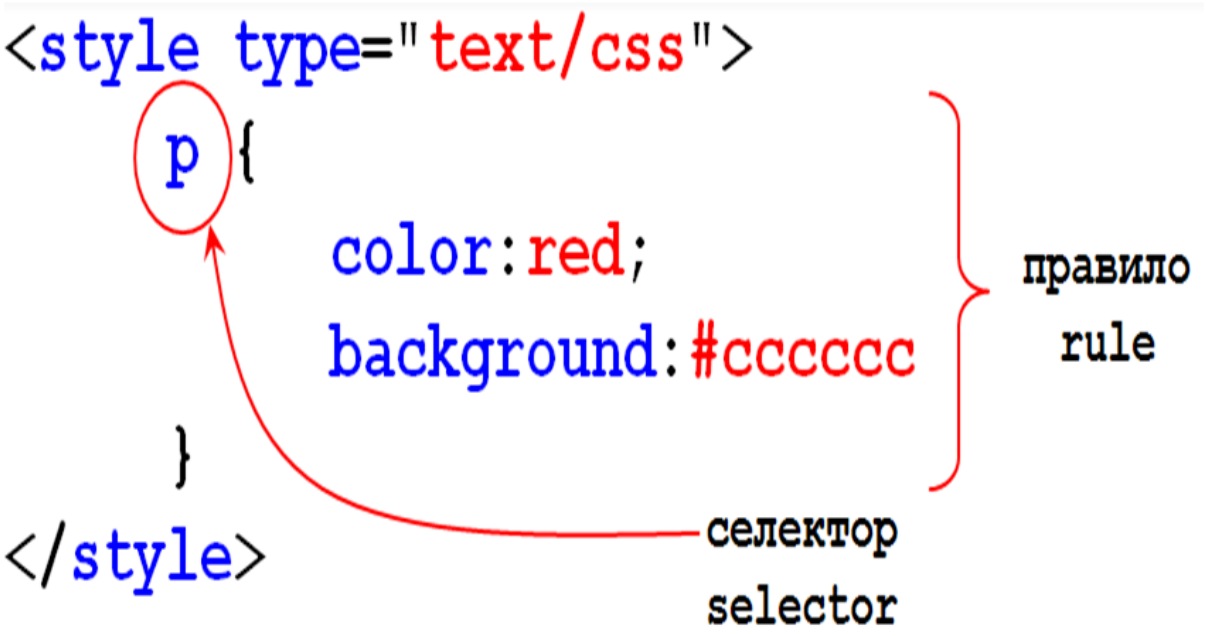
Обычный текст

Абзац с методом встраивания

Метод вложения (embeding) CSS

Метод вложения или внутренний стиль описывается в области head веб-страницы.

Метод вложения (внутренний стиль) нужен при необходимости использования одинакового стиля для тега (селектора) во всем документе. Например, для тега абзаца p можно добавить стиль данным методом, при этом имея в виду, что все теги абзаца на странице будут оформлены данным стилем



 Итак, селектор - это формальное описание элемента (тега), или их группы, к которому должны быть применены созданные правила стиля.
 В описании селекторов и имен стилей не должно быть пробелов.

Пример: в примере использован метод вложения, поэтому оба абзаца в документе будут стилизованы

```
<head>
<style type="text/css">
 p{
        color:red;
        background:#cccccc;
 }
</style>
</head>
<body>
<p>Абзац1</p>
<p>Абзац2</p>
...
```

Результат:

Абзац1
Абзац2

В качестве селектора может выступать любой тег HTML для которого определяются правила форматирования, такие как: цвет, фон, размер и т.д.

Пример использования вложенного стиля для разных селекторов (тегов)

```
<head>
<style type="text/css">
h1 {
        border-width: 1;
        border: groove;
```

```
        text-align: center;
        color: green
        }
h2 {
        color: maroon;
        font-style: italic
        }
body {
        background-color: #FF0000;
        }
...
</style>
```

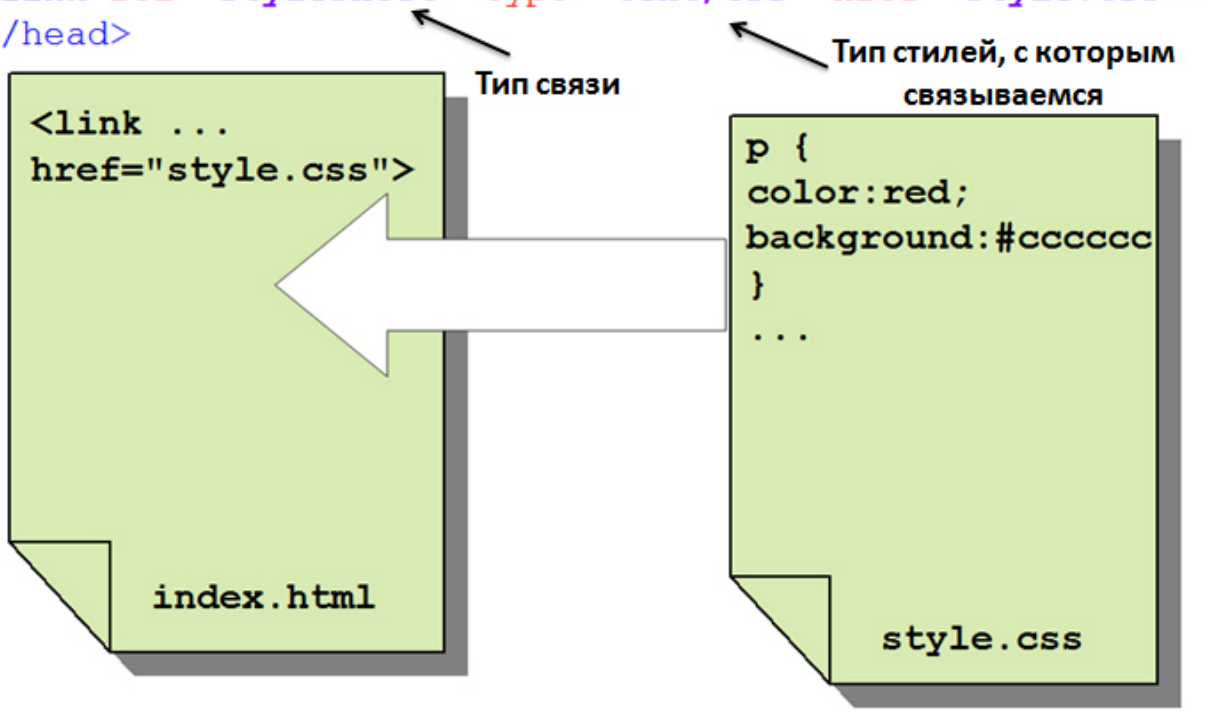
Метод связывания (Linking) в CSS

Это самый надежный и самый эффективный способ использования каскадных таблиц стилей.

Используется **метод связывания** (или внешний стиль) для того, чтобы обеспечить единый стиль для всех страниц сайта. Подключив файл со стилями ко всем страницам, обеспечится влияние правил, описанных в файле, на все теги

Чтобы подключить к странице файл с таблицами стилей, надо использовать элемент LINK в секции HEAD:

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



Задание css1_1.

- Используя метод встраивания определите цвет текста элемента h1.
- Используя метод вложения определите задний фон страницы.

- Используя метод связывания определите написание параграфа курсивным стилем.

Работайте над текстом:

<**body**>
<**h1**>В моей душе</**h1**>
<**p**>
Я хочу быть ребенком, наивным и смелым,<**br**/>
Ничего не бояться и верить в добро.<**br**/>
Я бы снова писала по черному белым:<**br**/>
Два плюс два - ну, четыре, конечно равно!
</**p**>

Вопросы для самоконтроля

1. Виды таблиц стилей.
2. Виды селекторов.
3. Наследование.

Лабораторная работа №7. Использование классов.

Цель: изучить способы подключения классов.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Создание и использование классов CSS

Для выделения какой-то группы объектов (элементов), которые необходимо наделить одними и теми же свойствами css, необходимо создать класс.

Синтаксис:

```
.имя_класса{
  свойство1: значение;
  свойство2: значение;
}
```

Подробнее:

Чтобы добавить класс, необходимо:

1.

Для элементов (тегов) к которым будут применены свойства класса, нужно **прописать атрибут class** с придуманным названием класса в качестве значения атрибута:

```
<h1 class="my_class">В моей душе</h1>
```

2.

В отельном стилевом файле (*style.css*) или в области head текущего документа **прописать свойства созданного класса**

Идентификатор класса всегда начинается с точки (.)

```
h1.my_class {
        color: RGB(215,40,40);
        text-align: center
        }
```

В данном примере класс my_class будет «привязан» именно к тегу h1, т.е. для других тегов с аналогичным классом свойства работать не будут.

Можно написать **без привязки к конкретному тегу**:

```
.my_class {
        color: RGB(215,40,40);
        text-align: center
        }
```

В данном случае класс будет применен к любым тегом данного класса

Пример: Создать два класса с названиями red1 и class1. Один класс применить для заголовков h1, другой класс применить для тегов p

Выполнение:

```
<html>
<head>
<style type="text/css">
```

```
h1.red1 {
        color: RGB(215,40,40);
        text-align: center
        }
p.class1{color:#3366FF; font-family:Arial}
</style>
</head>
<body>
<h1 class="red1">В моей душе</h1>
<p class="class1">
Я хочу быть ребенком: наивным и смелым,<br>
Ничего не бояться и верить в добро.<br>
Я бы снова писала по черному белым:<br>
Два плюс два - ну, четыре, конечно равно!
</p>
<p> Конец </p>
```

Результат:

В моей душе

Я хочу быть ребенком: наивным и смелым,
Ничего не бояться и верить в добро.
Я бы снова писала по черному белым:
Два плюс два - ну, четыре, конечно равно!
Конец

Задание: скопируйте код страницы. Измените страницу меню сайта так, чтобы одни пункты меню были темного цвета (класс .dark), а другие – светлого (класс .light).

Для гиперссылки добавить свойство: text-decoration:none;

```
<html>
<head>
        <title> Классы </title>
<style type="text/css">
...
</style>
</head>
<body>
<center><h3> Главное меню </h3></center>
<ul>
        <a href="#" class="dark"><li>Введение</li></a>
        <a href="#" class="dark"><li>Глава1</li></a>
        <a href="#" class="dark"><li>Глава2</li></a>
        <a href="#" class="dark"><li>Заключение</li></a>
</ul>
<center><h3> Дополнительное меню </h3></center>
<ul>
```

```
        <a href="#" class="light"><li>Тест</li></a>
        <a href="#" class="light"><li>Глоссарий</li></a>
        <a href="#" class="light"><li>Литература</li></a>
</ul>
</body>
</html>
```

Примеры использования классов с тегами и просто классов:

```
     тег ──────────────────→ div{color:red}
тег + класс ──────────────→ div.green{color:green}
   класс ──────────────────→ .blue{color:blue}
```

```
<div>Обычный div</div>
<div class="green">Div с классом green</div>
<p class="green">Абзац с классом green
<p class="blue">Абзац с классом blue
<div class="blue">Div с классом blue</div>
```

Обычный div
Div с классом green

Абзац с классом green

Абзац с классом blue
Div с классом blue

Универсальные классы или CSS селектор ID

Универсальные классы необходимы для того, чтобы оформить определенным стилем один единственный элемент на странице (на разных страницах сайта).

Синтаксис:

```
#имя_класса{
  свойство1: значение;
  свойство2: значение;
}
```

Подробнее:

1.

Необходимо **задать атрибут** `id` **с уникальным значением** для того элемента, к которому будет применен универсальный класс:

```
<h2 id="steel">Заголовок</h2>
```

Значение атрибута `id` придумывается самостоятельно и должно быть единственным таким значением `id` на всей веб-странице.

2.

В стилевом файле (**style.css**) либо в области head текущей веб-страницы **задается стиль для селектора id**:

```
<style type="text/css">
#steel {
  color: RGB(155,180,190);
  font-weight:bold
}
</style>
```

В стилях опознавательным знаком для универсального селектора является символ #

Пример: для первого заголовка h2 создать универсальный стиль, оформив его каким-либо цветом и увеличив жирность шрифта

Выполнение:

```
<html>
<head>
<style type="text/css">
#steel {
  color: RGB(155,180,190); /* изменили цвет в системе RGB */
  font-weight:bold /* установили жирный шрифт */
}
</style>
</head>
<body>
<h2 id="steel">Заголовок со стилем</h2>
<h2>Заголовок без стиля</h2>
</body>
</html>
```

Результат:

Заголовок со стилем

Заголовок без стиля

Стиль универсального селектора также может быть определен **для конкретного тега**:

```
h2#steel {
  color: RGB(155,180,190); /* изменили цвет в системе RGB */
  font-weight:bold /* установили жирный шрифт */
}
```

В таком случае стиль будет влиять только на селекторы h2, другие теги с атрибутом id="steel" оформлены стилем не будут.

Задание: Задайте уникальный стиль для главного заголовка сайта. Опишите стиль в отдельном файле **style.css** и подключите стилевой файл к странице. В качестве свойств созданного стиля желательно использовать следующие:

Выравнивание текста:

```
{
```

text-align:center|right|left|justify;
}
Оформление текста:
{
text-decoration:overline|line-through|underline|blink;
}

Это заголовок первого уровня
~~Это заголовок второго уровня~~
Это заголовок третьего уровня
Это заголовок четвертого уровня

Преобразование текста:
{
text-transform:uppercase|lowercase|capitalize;
}

ЭТО ОБЫЧНЫЙ ПАРАГРАФ. Я СДЕЛ
это обычный параграф. я сделаю его под
Это Обычный Параграф. Я Сделаю Его

Стиль шрифта:
{
font-style:normal|italic|oblique;
}

Это обычный параграф.
Это параграф, написанный курсивом.
Это параграф, написанный наклонным шрифтом

Размер шрифта:
{
font-size:30px|2.5em;
}

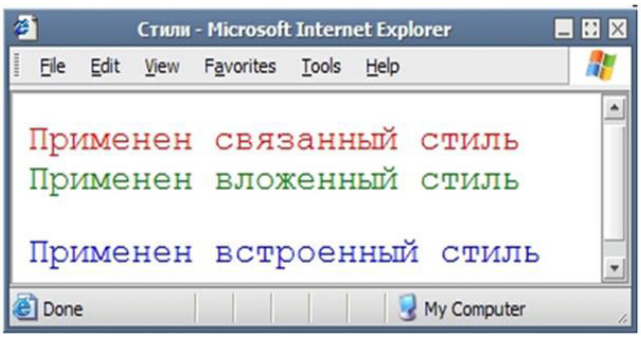
Это заголовок первого уровня
Это заголовок второго уровня

Каскадирование css стилей

Каскадирование css стилей означает **преемственность применения того или иного стиля в зависимости от используемого метода подключения css**. Рассмотрим на примере:

```
<link rel="stylesheet" type="text/css"
href="style.css">
<style>
        div{color:green}
</style>
...
<p>Применен связанный стиль
<div>Применен вложенный стиль</div>
<p style="color:blue">Применен встроенный стиль
```

```
p{color:red}
div{color:red}
```



На примере видно, что приоритетным является метод встраивания использования стилей. Следующим по приоритету следует метод вложения и только потом - метод связывания (стиль в отдельном файле)

Важно: В случае, если пользовательская таблица стилей содержит !important, то это правило имеет приоритет над любым правилом, описанным в таблице стилей:

P { **font-size**: 24pt!important; }

Пример: создать правило для дочернего класса .children, исходя из того, что тег данного класса вложен в родительский тек с классом .parent

```
//HTML:
<div class="parent">
  Далеко-далеко за словесными горами в стране.
  <div class="children">
    Далеко-далеко за словесными горами.
  </div>
</div>

//CSS
.parent .children {
  color: #666;
}
.parent {
  padding: 10px;
  color: #999;
}
```

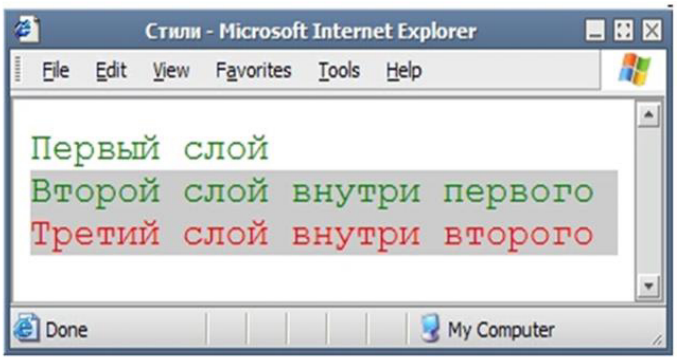
Тег с классом .children будет иметь цвет #666, а тег с классом .parent - #999.

Однако, если мы уберем свойство color: #666 у селектора .parent .children, то цвет дочернего элемента наследуется от родителя и станет равным #999. В этом заключается особенность **наследования**.

CSS наследование стилей

Вложенность тегов или их иерархия распространяется и на наследование стилей. Рассмотрим на примере:

```
<div style="color:green">Первый слой
       <div style="background:#cccccc">Второй слой
внутри первого
              <div style="color:red">Третий слой
внутри второго</div>
       </div>
</div>
```



Вопросы для самоконтроля

1. Преобразование текста.
2. Обработка пробелов и переносы строк.
3. Настройка табуляции.
4. Разрыв строки и границы слов.
5. Выравнивание и выключка строк.

Лабораторная работа №8. Контекстные, соседние и дочерние селекторы.

Цель: изучить способы соединения селекторов.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Три вида взаимоотношений в дереве элементов

1. Предки-потомки;
2. Родитель-ребенок;
3. Братья (соседи)

Рассмотри дерево элементов на примере:

```
1 <body>
2 <div>
3   <h1>Заголовок</h1>
4   <p><strong>Первый</strong> параграф в div</p>
5   <p><strong>Второй</strong> параграф в div</p>
6   <strong>Просто полужирный текст в div</strong>
7 </div>
8 <p><strong>Первый</strong> параграф в body</p>
9 </body>
```

Родительский элемент – тег, который содержит в себе рассматриваемый элемент.

В примере отношения родитель-ребенок:

- Элемент div (2 строка) - родительский по отношению к тегу h1 (3 строка), тогда как h1 - напротив, дочерний элемент (ребенок)
- Элемент p (4 строка) является родительским по отношению к strong (4 строка), тогда как strong - напротив, дочерний элемент (ребенок)
- Элемент div (2 строка) является родительским по отношению к strong (6 строка), тогда как strong - напротив, дочерний элемент (ребенок)
- Элемент p (8 строка) является родительским по отношению к strong (8 строка), тогда как strong - напротив, дочерний элемент (ребенок)
- Элемент body (1 строка) является родительским по отношению к div (2 строка) и p (8 строка)

Предок – элемент, который располагается на несколько уровней выше и содержит в себе рассматриваемый элемент.

В примере отношение предок-потомок:

- Элемент body является предком для элементов h1(3 строка), p и strong (4 строка), p и strong (5 строка), strong (6 строка) и strong (8 строка); в то время как все, перечисленные элементы являются потомками body
- Элемент div (2 строка) является предком для элементов strong (4 и 5 строка); в то время как strong является потомком для div

Братский элемент (соседний) – элемент, который имеет общий родительский элемент с рассматриваемым.
В примере отношение соседи (братья):

- Элементы h1 (3 строка), p(4 строка), p(5 строка) и strong(6 строка) являются братьями или соседними элементами.
- Элементы div(2 строка) и p(8 строка) являются братьями или соседними элементами.

Контекстные селекторы CSS (предки-потомки)
Правила контекстных селекторов также распространяются и на отношение родитель-ребенок.
Синтаксис:

```
x y{
  свойство1: значение;
  свойство2: значение;
}
```

x - селектор-предок, y - селектор-потомок
Правило будет действовать на селектор y
Пример: В hrml-код страницы необходимо добавить правило для контекстных селекторов:

```
1
2  <body>
3    <p><b>Жирное начертание текста</b></p>
4    <p><i><b>Одновременно жирное начертание текста
5      и выделенное цветом</b></i></p>
   </body>
```

Необходимо: Создать правило контекстных секторов для элемента b
Выполнение:

```
<style type="text/css">
p b{
        font-family: Times, serif; /* Семейство шрифта */
        font-weight: bold; /* Жирное начертание */
        color: navy; /* Синий цвет текста */
}
</style>
```

Результат:
Жирное начертание текста

Одновременно жирное начертание текста и выделенное цветом

В примере отношение предок-потомок характерно для элементов p и b в третьей строке. Но контекстные селекторы распространяются и на отношения родитель-ребенок, т.е. в примере - элементы p и b во второй строке.

Рассмотрим еще примеры, иллюстрирующие использование контекстных селекторов:

```
      id  ──────────────────→ #back{color:red}
 тег + id ──────────────────→ div#back{color:black}
контекстные ────────────────→ div b{color:green}
  селекторы ────────────────→ td td td{color:blue}
```

```
<div id="back">Div c id = back</div>
<div>Div c <b>контекстным</b> селектором</div>
<table><tr><td><table><tr><td><table><tr><td>
Третий уровень вложенности
</td></tr></table></td></tr></table></td></tr>
</table>
```

Div c id = back
Div c **контекстным** селектором

Первый уровень вложенности
Второй уровень вложенности
Третий уровень вложенности

Соседние селекторы CSS (братья)
Синтаксис:

```
x + y{
  свойство1: значение;
  свойство2: значение;
}
```

Стиль применяется для соседнего селектора y
Пример: В hrml-код страницы необходимо добавить правило для соседних селекторов:

```
<p><b>Студенты</b> - это <i>всезнающий</i> народ.</p>
```

Необходимо: Создать правило для соседнего селектора i
Выполнение:

```
b + i{
  color: red;
}
```

Результат:

Студенты - это *всезнающий* народ.

В примере теги i и b не перекрываются между собой другими элементами и поэтому представляют собой соседние селекторы. То, что они расположены внутри элемента p, никак не влияет на их отношение.

Дочерние селекторы в CSS (селектор родитель-ребенок)
Синтаксис:

```
x > y{
  свойство1: значение;
  свойство2: значение;
}
```

Здесь x - родитель, y - ребенок

Правило будет действовать на дочерний селектор y

Пример: В hrml-код страницы необходимо добавить правило для дочернего селектора:

```
1 <div><i>Кто на горе</i>, тот раньше солнце встретит.
2 </div>
3 <div><p><i>Кто средь друзей</i>, тот силой не шути.</p> </div>
```

Необходимо: создать правило для дочернего селектора i (1 строка)

Выполнение:

```
div > i{
  color: red;
}
```

Результат:

Кто на горе, тот раньше солнце встретит.

Кто средь друзей, тот силой не шути.

В третьей строке примера тег i не является дочерним для тега div, так как они перекрываются тегом p.

Задание: Создайте всевозможные правила для стилизации каждого из селекторов в расположенном ниже фрагменте кода:

```
1  <html>
2  <head>
3    <style type="text/css">
4    ...
5    </style>
6  </head>
7  <body>
8  <h1 id="header1"><i>Стихотворение</i> <u>2013 год</u></h1>
9  <h2>I</h2>
10 <p>В земные страсти вовлеченный,
11 я знаю, что из тьмы на свет
12 однажды выйдет ангел черный
13 и крикнет, что спасенья нет. </p>
14 <h2>II</h2>
15 <p>Но простодушный и несмелый,
16 прекрасный, как благая весть,
17 идущий следом ангел белый
18 прошепчет, что надежда есть.</p>
19 <p class="end">Конец</p>
```

```
20 <h1>Продолжение следует</h1>
21 </body>
22 </html>
```

Используйте CSS-свойства для оформления текста и для форматирования шрифта.

Вопросы для самоконтроля

1. Приоритетные цвета.
2. Цвета модели RGB.
3. Фон CSS.
4. Блоки CSS.

Лабораторная работа №9. Селекторы атрибутов и универсальный селектор. Псевдоклассы и псевдоэлементы CSS.

Цель: изучить способы соединения селекторов. изучить способы добавления псевдоклассов и псевдоэлементов на сайт.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

CSS селекторы атрибутов

1. **Селектор атрибута, значение которого не важно**

   Синтаксис:

```
[атрибут]{
  свойство1: значение;
  свойство2: значение;
}
/* или */
x[атрибут]{
  свойство1: значение;
  свойство2: значение;
}
```

   где x - селектор (тег), для которого и создается правило

   Пример: В hrml-код страницы необходимо добавить правило для селектора атрибута:

```
<html>
<head>
<style type="text/css">
...
</style>
</head>
<body>
<img src="1.jpg" alt="надпись">
</body>
</html>
```

   Необходимо: Создать правило для селектора img с атрибутом alt

   Выполнение:

```
<style type="text/css">
img[alt]{
  border:1px solid red; /* граница красного цвета */
}
</style>
```

   Результат:

2. **Селектор атрибута со значением**

Синтаксис:

```
[атрибут="значение"]{
  свойство1: значение;
  свойство2: значение;
}
/* или */
x[атрибут="значение"]{
  свойство1: значение;
  свойство2: значение;
}
```

где x - селектор (тег), для которого и создается правило

Пример: В hrml-код страницы необходимо добавить правило для селектора атрибута:

```
<body>
<input type="text" value="логин"><br>
<input type="radio" value="yes">Да<br>
<input type="radio" value="no">Нет<br>
</body>
</html>
```

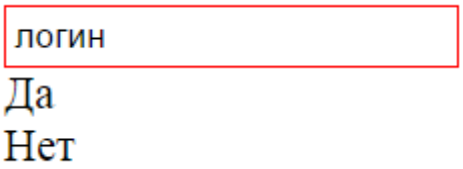
Необходимо: Создать правило для текстового поля, добавив рамку и внутренние отступы, и для radio кнокпи, скрыв ее (свойство display)

Выполнение:

```
input[type="text"]{
  padding:3px; /* внутренние отступы */
  border:1px solid red;
}
input[type="radio"]{
```

```
  display:none;/* скрываем элемент */
}
```

Результат:



Рассмотрим еще примеры:

Пример: Для цитатного текста (тег q), у которого установлен атрибут title, необходимо добавить свойство для цвета элемента. Цитатный текст без атрибута необходимо сделать курсивом

Выполнение:

```
<html>
<head>
<style type="text/css">
q{
  font-style:italic;
}
q[title]{
  color:maroon;
}
</style>
</head>
<body>
<p>Закон Мерфи гласит:
<q>Если неприятность может случиться, то она обязательно случится</q>, можем ввести
<q title="Из законов Фергюсона-Мержевича">После того, как веб-страница будет коррек
показывается в другом</q>.
</p>
</body>
</html>
```

Результат:

Закон Мерфи гласит:
Если неприятность может случиться, то она обязательно случится, можем ввести свое наблюдение:
После того, как веб-страница будет корректно отображаться в одном браузере, выяснится, что она неправильно показывается в другом.

Пример: Для гиперссылки, которая выводится в новом окне (атрибут target="_blank") установить задний фон и отступ слева

Выполнение:

```
<html>
<head>
<style type="text/css">
```

```
a[target="_blank"]{
        background-color:#00cc00;
        padding-left:15px
}
</style>
</head>
<body>
<a href="2.html">Обычная ссылка</a>
<a href="2.html" target="_blank">Ссылка в новом окне</a>
</body>
</html>
```

Результат:

Обычная ссылка

Ссылка в новом окне

ДОПОЛНИТЕЛЬНЫЕ ПАРАМЕТРЫ В ФИЛЬТРАЦИИ АТРИБУТОВ

[атрибут*=значение]{...}

Атрибут содержит данное значение (но не строго равняется ему)

Пример:

```
...
<style type="text/css">
div[id*=post]{color:red;}
</style>
</head>
<body>
<div id="post_1">one</div>
<div id="post_two">two</div>
<div id="third_post">three</div>
 ...
```

Результат:

one

two

three

[атрибут^=значение]{...}

Атрибут начинается с данного значения

Пример:

```
div[id^=post]{color: red;}
...
<div id="post_1">one</div>
<div id="post_two">two</div>
<div id="third_post">three</div>
```

Результат:

one
two
three

[атрибут$=значение]{...}
Атрибут заканчивается на данное значение

```
div[id$=post]{color: red;}
...
<div id="post_1">one</div>
<div id="post_two">two</div>
<div id="third_post">three</div>
```

Результат:

one
two
three

Важно: необходимо обратить внимание на то, что данные примеры работают во всех современных браузерах: Safari, Chrome, Firefox, Opera и IE7 и выше.

Универсальный селектор CSS

Порой возникает необходимость использования единого стиля одновременно для всех элементов веб-страницы. Универсальный селектор соответствует любому элементу веб-страницы, т.е. стиль, определенный для универсального селектора, будет выполнен над всеми элементами.

Синтаксис:

```
*{
        свойство1: значение;
        свойство2: значение;
        свойство3: значение;
}
```

Пример: для всех элементов страницы определить единое семейство шрифта и цвет шрифта

Выполнение:

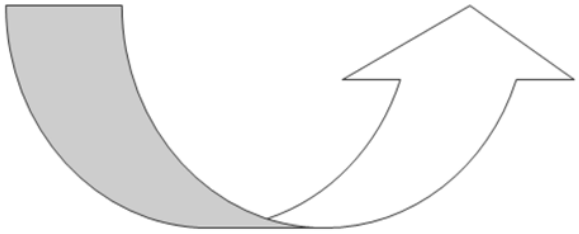
```
*{
        font-family: Arial, Verdana, sans-serif;
        color: navy;
}
```

Группировка CSS селекторов

Для сокращения записи правил в CSS принята группировка:

```
h1{
    color:red;
    background:yellow
}
h2{
    color:blue;
    background:yellow
}
h3{
    color:green;
    background:yellow
}
```

```
h1,h2,h3{
    background:yellow
}
h1{
    color:red;
}
h2{
    color:blue;
}
h3{
    color:green;
}
```



CSS приоритет селекторов

Как мы убедились, существует несколько способов создания связи между CSS и html-документом. Более того, к одному и тому же элементу веб-страницы могут быть назначены несколько стилей одновременно. В таком случае отображение такого элемента регулируется правилами приоритета селекторов CSS:

Общие правила:

- Внешние стили (метод связывания): 3 (наименее приоритетны)
- Внутренние стили (метод вложения): 2
- Строковые стили (метод встраивания): 1 (наивысший приоритет)

Подробные правила:

1. Внешняя таблица стилей (подключаемая методом связывания), ссылка на которую встречается в html-документе позже, имеет приоритет по отношению к внешней таблице стилей, ссылка на которую встречается раньше.
   В примере стили файла *style2.css* будут приоритетней стилей файла *style1.css*:

```
<head>
<link rel="stylesheet" type="text/css" href="style1.css" />
<link rel="stylesheet" type="text/css" href="style2.css" />
</head>
```

2. Более конкретные стили имеют приоритет перед менее конкретными.
   - Стилевой **класс приоритетнее** стиля для **конкретного тега**:
     p.classname {**color**:red} /* *более высокий приоритет* */

```
p {color:blue}
```

- **Универсальный id** имеет более высокий приоритет, чем у класса:

```
 #in{color:red} /* более высокий приоритет */
 .news{color:blue}
</style></head>
<body>
<p id="in" class="news">Параграф будет красного цвета</p>
```

- Свойства стиля, объявленные как !important, имеют приоритет перед всеми другими значениями.
  В примере стиль весь текст в рамках тегов p сделается красным вне зависимости от любых других переопределений стиля для тега p:

```
...
p {color: red !important}
p.blue{color:blue}
...
<body>
<p class="blue">Текст красный</p>
...
```

Итак, приоритетность:

I	II	III	IV
!important;	#id	.class :pseudo-class []atributes	стиль тегов :pseudo-elements

3. В случае привязки к тегу нескольких стилевых классов, приоритетными считаются те, что указаны правее.
   В примере приоритетным будет class2, то есть текст станет синего цвета:

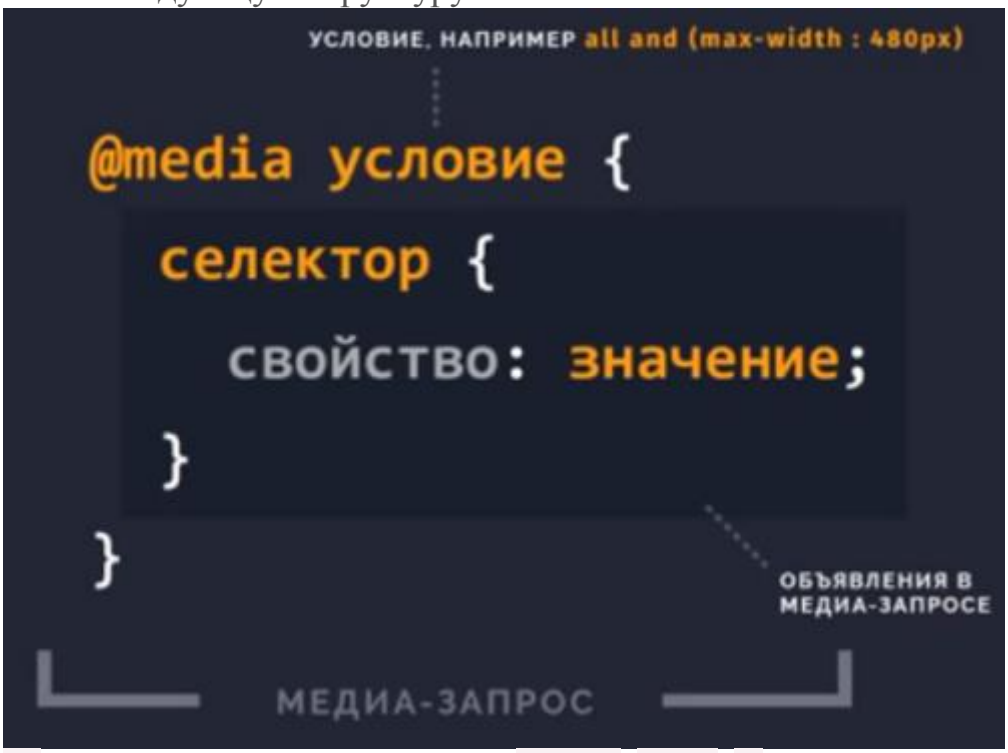
```
...
p.class1 {color:red} /* более высокий приоритет */
p.class2 {color:blue}
...
<body>
<p class="class1,class2">Текст синий</p>
...
```

Медиа-запросы в CSS

Медиа-запросы - логическое выражение, которое может быть равно истине (true) или лжи (false)

Условием является либо параметры устройства, на котором отображается веб-страница, либо размеры экрана пользователя.

Медиа-запрос записывается либо в стилевом файле, либо во вложенном стиле и имеет следующую структуру:



all - все устройста. Может быть screen | print | tv
max-width - **медиа функция**, которая может задавать параметры указанного устройства или разрешение экрана
В примере устройство с максимальным разрешением экрана - 480px и с минимальным - 320px:

```
@media all and (max-width : 480px), all and (min-width: 320px) {
  .my-class {
    color: #999;
  }
}
```

Из примера видно, что функции могут содержать логические условия: AND - И, NOT - НЕ и ONLY - только
Медиа-запросы логично размещать после всех описанных стилей

Псевдоклассы в CSS
Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя.
Важно: На псевдокласс указывает наличие двоеточия (:)
 Три псевдокласса определены именно **для гиперссылки** (для тега a):

```
a:link{...} /* для ссылки непосещенной */
a:visited{...} /* для посещенной ссылки */
a:active{...} /* для активной ссылки, в момент щелчка */
```

* active - псевдокласс не только для гиперссылки
 Псевдоклассы **для всех элементов**:

элемент:hover{...} /* *по наведению курсора на элемент* */

 Псевдоклассы **для всех элементов управления**:

input:focus{...} /* *в тот момент, когда элемент получает фокус* */
input:active{...}/* *в момент активации элемента* */

Пример: На странице расположено текстовое поле. Необходимо взять элемент в толстую черную рамку, когда он в фокусе, и в толстую красную, когда он активен

Выполнение:

```
<style type="text/css">
.el1:focus { outline: thick solid black }
.el1:active { outline: thick solid red }
</style>
</head>
<body>
<input type="text" class="el1" value="щелкни по мне">
```

Результат:

Псевдоэлементы CSS

Псевдоэлементы позволяют, во-первых, задать стиль некоторых частей элементов, которые не определены в дереве элементов документа, а, во-вторых, генерировать содержимое, которого нет в исходном коде текста.

Псевоэлементы, определяющие новые элементы:

элемент:first-letter {...}/* *первая буква или символ элемента* */
элемент:first-line {...}/* *первая строка элемента* */

Пример: Для первой строки параграфа применить курсивный стиль шрифта, первую букву абзаца сделать красным цветом

Выполнение:

```
<style type="text/css">
.f1:first-letter {color: red}
.f1:first-line {font-style: italic}
</style>
</head>
<body>
<p class="f1">К этому тексту применен стиль. К этому тексту применен стиль. К этому т
применен стиль.
К этому тексту применен стиль. К этому тексту применен стиль.
</p>
```

Результат:

К этому тексту применен стиль. К этому тексту
применен стиль. К этому тексту применен стиль. К
этому тексту применен стиль. К этому тексту применен
стиль.

Псевоэлементы, генерирующие содержимое:

элемент:before {**content**:""}*/* генерирует текст перед элементом */*
элемент:after {**content**:""} */* генерирует текст после элемента */*

Пример: К содержимому абзаца с классом new добавить дополнительное слово - *Ого!*.

Выполнение:

```
<style type="text/css">
p.new:after{
        content: "- Ого!"
        }
</style>
</head>
<body>
<p class="new">Ловля льва в пустыне с помощью метода золотого сечения.</p>
<p>Метод ловли льва простым перебором.</p>
```

Результат:

Ловля льва в пустыне с помощью метода золотого сечения.- Ого!

Метод ловли льва простым перебором.

Пример: Для маркированного списка убрать маркер и установить вместо него какой-либо символ

Выполнение:

```
<style type="text/css">
ul {
   list-style-type: none; /* Прячем маркеры списка */
}
li:before {
content: "\20aa  "; /* Добавляем перед элементом списка символ в юникоде */
}
</style>
</head>
<body>
<ul>
<li>Чебурашка</li>
<li>Крокодил Гена</li>
<li>Шапокляк</li>
<li>Крыса Лариса</li>
</ul>
```

Результат:

- Чебурашка
- Крокодил Гена
- Шапокляк
- Крыса Лариса

Задание: Скопируйте текст, расположенный ниже, и вставьте в веб-страницу. Используя вложенный стиль css либо отдельный файл css, создайте следующие правила, чтобы достигнуть эффекта как на итоговом изображении.

Текст:

```
<body>
<h1>Стих 1</h1>
<p>
Вопрос такой: зачем все это мне?<br/>
Но, не найдя ответа на подходе,<br/>
Я понял, что пришедшее извне<br/>
Стремительно вовне же и уходит.
</p><p>
Другой вопрос - как быть? Но тут<br/>
Ответ просился сам: бездействуй!<br/>
(Угли недолго проживут<br/>
В костре, где все без происшествий.)
</p><p>
Вот так и жду, не шевелясь,<br/>
Когда уйдут из сердца лица,<br/>
Ушедшие однажды с глаз,<br/>
Чтоб никогда не возвратиться.
</p>
<hr>
<h1>Стих 2</h1>
<p>
Обещай, что вернешься Домой.<br/>
Эти зимы меня одолеют.<br/>
Я смотрю на тебя и не смею<br/>
Прикоснуться холодной рукой.
</p><p>
Обещай, что не будешь скучать.<br/>
Нам обоим запомнятся годы<br/>
Нашей странной и глупой свободы,<br/>
Научившей любить и прощать.
</p><p>
Обещай, что в далеком краю,<br/>
Если станет тебе одиноко,<br/>
Ты прочтешь эти добрые строки<br/>
Про бескрайнюю Нежность мою.
</p><p>
```

И поселится в сердце покой.<**br**/>
Нет тебя мне на свете дороже.<**br**/>
Обещай,<**br**/>
что когда-нибудь все же<**br**/>
ты конечно вернешься Домой.
</**p**>
<**hr**>
</**body**>

Необходимые правила:

```
p:first-line{…}
p:first-letter{…}
p {…}
h1{
        text-transform:…;
        border-bottom-style:...;
        border-bottom-color:...;
   }
hr {…}
```

Свойства:

text-transform
border-bottom-style
border-bottom-color
Оформление шрифта

Итоговое **изображение:**

СТИХ 1

Вопрос такой: зачем все это мне?
Но, не найдя ответа на подходе,
Я понял, что пришедшее извне
Стремительно вовне же и уходит.

Другой вопрос — как быть? Но тут
Ответ просился сам: бездействуй!
(Угли недолго проживут
В костре, где все без происшествий.)

Вот так и жду, не шевелясь,
Когда уйдут из сердца лица,
Ушедшие однажды с глаз,
Чтоб никогда не возвратиться.

СТИХ 2

Обещай, что вернешься Домой.
Эти зимы меня одолеют.
Я смотрю на тебя и не смею

Вопросы для самоконтроля

1. CSS позиционирование.
2. Выбор схемы позиционирования.
3. Смещение блока.
4. Обтекание.
5. Наложение.

Лабораторная работа №10. Основные стили CSS: оформление текста. свойства шрифта.

Цель: изучить способы оформления текста и способы оформления шрифта.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Оформление текста css

Перечислим основные свойства для оформления текста и их значения

COLOR (ЦВЕТ)

Цвет текста

Возможные значения:

- red (название)
- rgb(255,0,0) (код RGB)
- #ff0000 (шестнадцатиричный код)

Пример:

```
.ex {color:blue;}
 h1 {color:#00ff00;}
 p {color:rgb(255,0,0);}
```

Результат:

Это заголовок первого уровня

Это обычный параграф

Это параграф с классом ex

TEXT-ALIGN (ВЫРАВНИВАНИЕ)

Выравнивание текста по горизонтали

Возможные значения:

- center (по центру)
- left (по левому краю)
- right (по правому краю)
- justify (по ширине)

Пример:

```
<style type="text/css">
 h1 {text-align:center;}
 p.date {text-align:right;}
 p.main {text-align:justify;}
</style></head>
<body>
```

```
<h1>Алгоритм</h1>
<p class="main">
Название "алгоритм" произошло от латинской формы имени величайшего среднеазиатск
своей книге "Об индийском счете" он изложил правила записи натуральных чисел с помо
каждому школьнику. В XII веке эта книга была переведена на латынь и получила широко
</p>
<p class="date">Мухаммед ибн Муса ал-Хорезми, 783-850 гг</p>
```

Результат:

Алгритм

Название "алгоритм" произошло от латинской формы имени величайшего среднеазиатского математика Мухаммеда ибн Муса ал-Хорезми (Alhorithmi), жившего в 783—850 гг. В своей книге "Об индийском счете" он изложил правила записи натуральных чисел с помощью арабских цифр и правила действий над ними "столбиком", знакомые теперь каждому школьнику. В XII веке эта книга была переведена на латынь и получила широкое распространение в Европе.

Мухаммед ибн Муса ал-Хорезми, 783—850 гг

TEXT-DECORATION (ОФОРМЛЕНИЕ)

Возможные значения:

- overline (подчеркивание сверху)
- line-through (перечеркивание)
- underline (подчеркивание)
- blink (мигание текста)
- none (без оформления)

Пример:

```
...
<style type="text/css">
 a{text-decoration:none}
 p.class1{text-decoration:overline}
 p.class2{text-decoration:line-through}
 p.class3{text-decoration:underline}
 p.class4{text-decoration:blink}
</style></head>
<body>
<a href="#">Гиперссылка</a>
<p class="class1">класс 1</p>
<p class="class2">класс 2</p>
<p class="class3">класс 3</p>
<p class="class4">класс 4</p>
...
```

Результат:

Гиперссылка
класс 1
класс 2

~~класс 3~~
класс 4

TEXT-TRANSFORM (ПРЕОБРАЗОВАНИЕ ТЕКСТА)

Возможные значения:

- uppercase (верхний регистр)
- lowercase (нижний регистр)
- capitalize (каждое слово с заглавной буквы)
- none (без преобразования)

Пример:

```
...
<style type="text/css">
 p.uppercase{text-transform:uppercase}
 p.lowercase{text-transform:lowercase}
 p.capitalize{text-transform:capitalize}
</style></head>
<body>
<p class="uppercase">Это параграф для примера</p>
<p class="lowercase">Это параграф для примера</p>
<p class="capitalize">Это параграф для примера</p>
```

Результат:

ЭТО ПАРАГРАФ ДЛЯ ПРИМЕРА
это параграф для примера
Это Параграф Для Примера

TEXT-INDENT (КРАСНАЯ СТРОКА)

Возможные значения:

- 20px (значение в пикселях)

Пример:

```
...
<style type="text/css">
p{text-indent:50px}
</style></head>
<body>
<p>Оформление текста в значительной степени влияет на то, как будет воспринята передаваемая им информация. Зачастую плохо, неграмотно оформленный текст вообще не читают, так как он вызывает дискомфорт у читателя на подсознательном уровне.</p>
```

Результат:

Оформление текста в значительной степени влияет на то, как будет воспринята передаваемая им информация. Зачастую плохо, неграмотно оформленный текст вообще не читают, так как он вызывает дискомфорт у читателя на подсознательном уровне.

DIRECTION (НАПРАВЛЕНИЕ ТЕКСТА)

Возможные значения:

- ltr (слева направо)
- rtl (справа налево)

Пример:

```
...
<style type="text/css">
p{direction:rtl}
</style></head>
<body>
<p>Это текст</p>
```

Результат:

Это текст

LINE-HEIGHT (МЕЖСТРОЧНЫЙ ИНТЕРВАЛ)

Возможные значения:

- normal
  Межстрочное расстояние вычисляется автоматически (значение по умолчанию)
- число
  Множитель, на который будет умножен размер текущего шрифта для вычисления межстрочного расстояния
- размер
  Фиксированное межстрочное расстояние в px (пикселях), pt (пунктах), cm (сантиметрах) и т.д.
- %
  Межстрочное расстояние в % от размера текущего шрифта

Пример:

```
...
<style type="text/css">
 p.small{line-height:90%}
 p.big{line-height:200%}
</style></head>
<body>
<p class="small">Это параграф1 для примера. Это параграф1 для примера. Это параграф1 для примера. Это параграф1 для примера. Это параграф1 для примера. Это параграф1 для примера. </p>
<p class="big">Это параграф2 для примера. Это параграф2 для примера. Это параграф2 для примера. Это параграф2 для примера. Это параграф2 для примера. Это параграф2 для примера. </p>
```

Результат:

Это параграф1 для примера. Это параграф1 для примера. Это параграф1 для примера. Это параграф1 для примера. Это параграф1 для примера. Это параграф1 для примера.

Это параграф2 для примера. Это параграф2 для примера. Это параграф2 для примера. Это параграф2 для примера. Это параграф2 для примера. Это параграф2 для примера.

LETTER-SPACING (МЕЖСИМВОЛЬНОЕ РАССТОЯНИЕ)

Возможные значения:

- значение

Пример:

```
...
<style type="text/css">
h5{letter-spacing:2px}
h3{letter-spacing:-1px}
</style></head>
<body>
<h5>Первый заголовок</h5>
<h6>Второй заголовок</h6>
```

Результат:

Первый заголовок

Второй заголовок

WORD-SPACING (РАССТОЯНИЕ МЕЖДУ СЛОВАМИ)

Возможные значения:

- значение

Пример:

```
...
<style type="text/css">
p{word-spacing:30px}
</style></head>
<body>
<p>Пример параграфа для демонстрации стиля.</p>
...
```

Результат:

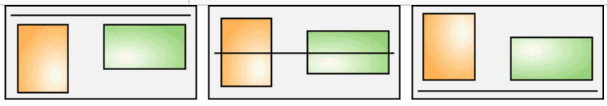
Пример параграфа для демонстрации стиля.

VERTICAL-ALIGN (ВЕРТИКАЛЬНОЕ ВЫРАВНИВАНИЕ)

Возможные значения:

Значение	Описание
число	Поднимает или опускает элемент относительно базовой строки. Положительные числа поднимают элемент, отрицательные - опускают (можно использовать пиксели, сантиметры и т.д.)
%	Поднимает или опускает элемент относительно базовой строки на величину % от высоты строки ("line-height"). Положительные %

	поднимает элемент, отрицательный - опускает.
baseline	Выравнивание по базовой линии родительского элемента. Значение по умолчанию
sub	Выравнивание, как подстрочный индекс (H2O)
super	Выравнивание как надстрочный индекс (x2)
top	Верхний край элемента выравнивается по верхнему краю самого высокого элемента строки
text-top	Верх элемента выравнивается по верху самого высокого текстового элемента на строке
middle	Элемент выравниваетя по середине родительского элемента
bottom	Низ элемента выравнивается по низу самого низкого элемента строки
text-bottom	Низ элемента выравнивается по низу самого низкого элемента строки
inherit	Значение наследуется от родительского элемента



а. Выравнивание по верхнему краю *б. Выравнивание по центру* *в. Выравнивание по нижнему краю*

Пример:

```
...
<style type="text/css">
 h5 {text-align:center}
 p.main {text-align:justify}
 p.date {text-align:right}
</style></head>
<body>
<h5>Стих 1</h5>
<p class="main">
```

Вопрос такой: зачем все это мне? Но, не найдя ответа на подходе, Я понял, что пришедш
просился сам: бездействуй! (Угли недолго проживут В костре, где все без происшествий

```
</p>
<p class="date">31/01/2017</p>
...
```

Результат:

Стих 1

Вопрос такой: зачем все это мне? Но, не найдя ответа на подходе, Я понял, что пришедшее извне Стремительно вовне же и уходит. Другой вопрос - как быть? Но тут Ответ просился сам: бездействуй! (Угли недолго проживут В костре, где все без происшествий.)

31/01/2017

Свойства шрифта css

Перечислим основные свойства шрифта и их значения

FONT-FAMILY (СЕМЕЙСТВО ШРИФТА)

Возможные значения:

- Serif (серифный шрифт с засечками)
- Sans-serif (без засечек)
- Monospace (моноширинный)
- Название шрифта

Пример:

```
p{font-family:"Times New Roman", Times, serif;}
h5{font-family:Arial}
```

Результат:

Это пример параграфа со свойством font-family. Это пример параграфа со свойством font-family. Это пример параграфа со свойством font-family со свойством font-family. Это пример параграфа со свойством font-family.

Это пример заголовка со свойством font-family

FONT-STYLE (СТИЛЬ ШРИФТА)

Возможные значения:

- normal (обычный текст)
- italic (курсив)
- oblique (наклонный текст)

Пример:

```
p.normal{font-style:normal}
p.italic{font-style:italic}
p.oblique{font-style:oblique}
```

Результат:
Параграф со стилем normal
Параграф со стилем italic
Параграф со стилем oblique

FONT-SIZE (РАЗМЕР ШРИФТА)

Возможные значения:

- px (обычный текст)
- em (16px=1em)
- дополнительные:
    - xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger

Пример:

```
h5{font-size:40px}
h6{font-size:1.875em} /* 30px/16=1.875em */
```

Результат:
Заголовок пятого уровня
Заголовок шестого уровня
Важно: 1em соответствует размеру шрифта по умолчанию. Размер шрифта по умолчанию в браузерах устанавливается равным 16 пикселям.
Получаем: 1em = 16px
Размер шрифта в em можно перевести из пикселей по формуле: пиксели/16=em

FONT-WEIGHT (ШИРИНА ЛИНИЙ ШРИФТА)

Возможные значения:

- normal (обычный шрифт)
- bold («жирный»)
- bolder («жирнее»)
- lighter (менее «жирный»)

Первые два значения - **абсолютные значения**. Вторые два - **относительные** (зависят от соседних или родительских элементов)

Пример:

```
p.normal{font-weight:normal}
p.bold{font-weight:bold}
```

Результат:
Параграф с классом normal
Параграф с классом bold

Краткая запись font

Свойства шрифта задаются в следующем порядке:

```
элемент {font-style  font-variant  font-weight  font-size line-height  font-family}
```

Пример:

```
p{font:15px Arial,sans-serif}
```

Обязательны свойства: font-size и font-family

Вопросы для самоконтроля

1. Центрирование дизайна.
2. Шаблоны на основе обтекания.

Лабораторная работа №11. Позиционирование блоков. Другие свойства блоков CSS.

Цель: изучить способы добавления блочной структуры и способы оформления блоков.

Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Позиционирование блоков

Элемент в html - это прямоугольник; **схемы позиционирования определяют где должен располагаться этот прямоугольник**, а также как он должен влиять на элементы вокруг себя.

Рассмотрим **два элемента, служащих для стилизации**:

1. Тег div - это **блочный элемент**, который может содержащать другие блоки, такие как, например, изображения и таблицы.
   Пример:
   <**div**>блок текста<**img** src="…"/></**div**>
2. Тег span - **строчный элемент**, содержащий любой текст; элемент можно вставить прямо в строку (встраиваемый элемент). Может заменить собой элементы: FONT, I, B, U и т.п.

Пример:

Текст <**span**>продолжение текста</**span**>

СТАТИЧЕСКОЕ ПОЗИЦИОНИРОВАНИЕ

POSITION: STATIC

Статическое позиционирование - это позиционирование элемента по умолчанию. Такой элемент занимает на веб-странице место соответственно своему положению в html-коде, т.е. определенное в нормальном потоке.

Нормальный поток элементов - это отображение элементов соответственно их месту в коде

На статически позиционированный элемент не действуют свойства top, bottom, left и right.

Пример:

```
div{
  position: static;
}
```

ФИКСИРОВАННОЕ ПОЗИЦИОНИРОВАНИЕ CSS

POSITION: FIXED

- Фиксированно позиционированный элемент **удаляется из потока HTML** и позиционируется относительно окна браузера. Элемент как будто «плавает» над другими тегами страницы.
- Он остается на месте при прокрутке страницы.

- При фиксированном позиционировании **необходимо задать координаты расположения элемента**.

Установка координат:

- top | bottom
- left | right

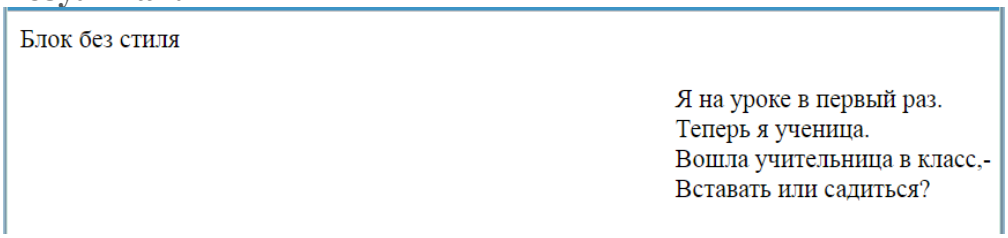
Значения:

auto | величина | %

Пример:

```
...
<style type="text/css">
p{
        position: fixed;
        top:30px;
        right:5px;
}
</style></head>
<body>
<p>
Я на уроке в первый раз.<br/>
Теперь я ученица.<br/>
Вошла учительница в класс,-<br/>
Вставать или садиться?<br/>
</p>
<div>Блок без стиля</div>
```

Результат:



ОТНОСИТЕЛЬНОЕ ПОЗИЦИОНИРОВАНИЕ CSS

POSITION: RELATIVE

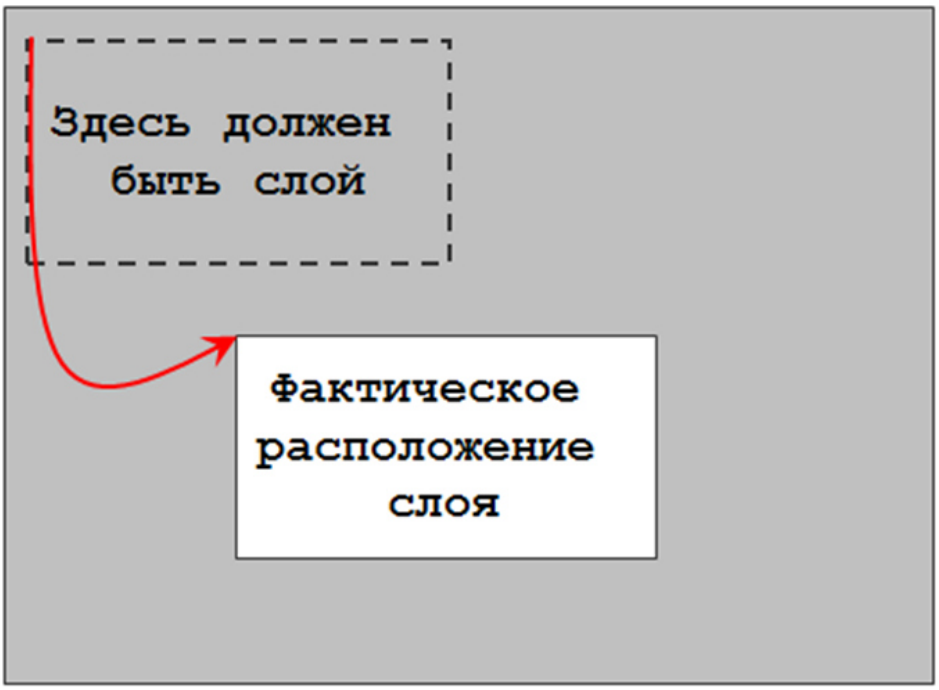
- Относительное позиционирование **располагает элемент относительно его нормального положения**.
- При относительном позиционировании обязательна **установка координат** расположения элемента относительно его нормального положения.

Установка координат:

- top | bottom
- left | right

Значения:

auto | величина | %



Пример:

```
<p>
Я на уроке в первый раз.<br/>
Теперь я ученица.<br/>
Вошла учительница в класс,-<br/>
Вставать или садиться?<br/>
</p>
<div style="position:relative; top:-100px; left:150px; color:red">
Относительно позиционированный блок
</div>
```

Результат:

Я на уроке в первый раз.
Теперь я ученица.
Вошла учительница в класс,-
Вставать или садиться?
Относительно позиционированный блок

АБСОЛЮТНОЕ ПОЗИЦИОНИРОВАНИЕ CSS

POSITION: ABSOLUTE

- Абсолютная позиция отсчитывается от ближайшего к элементу родительского контейнера, который имеет позиционирование absolute, relative или fixed (кроме static). Если такой родительский элемент отсутствует, то им считается окно браузера.
- При абсолютном позиционировании обязательна **установка координат** расположения элемента относительно его родителя (см. пред. пункт).

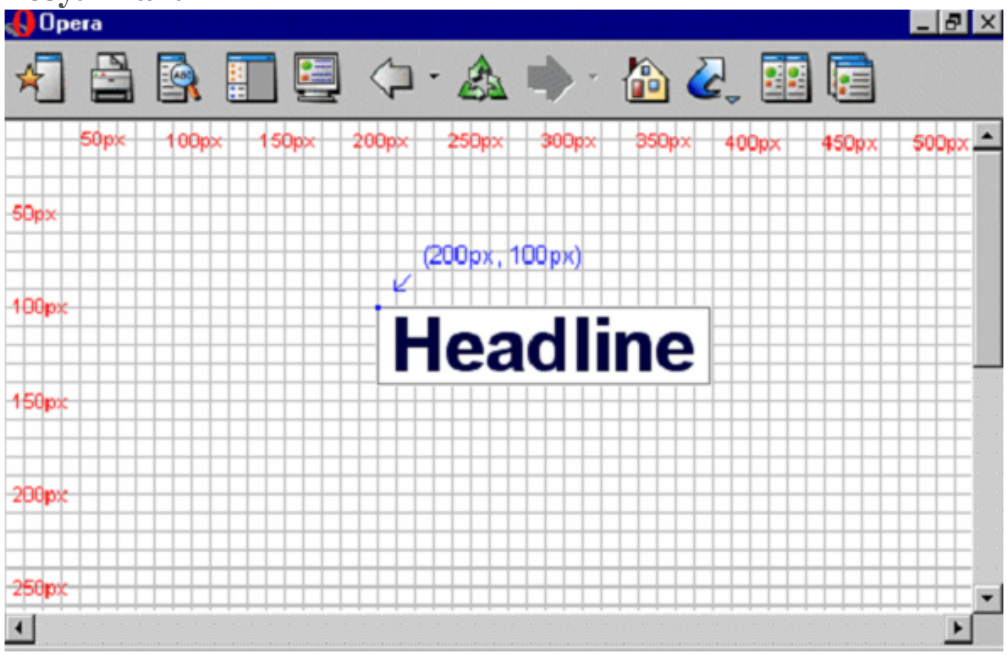
Установка координат:

- top | bottom
- left | right

Значения:
auto | величина | %
Пример:

```
h1 {
  position:absolute;
  top: 100px;
  left: 200px
}
```

Результат:



Пример взаимодействия с родительским блоком:

1.

```
<div style="position:absolute; top: 50px; background-color:#F00">
        блок 1
</div>
<div style="position:absolute; top:20px; background-color:#0F0">
        блок 2
</div>
```

Результат:

блок 1
блок 2

2.

```
<div style="position:absolute; top: 50px; background-color:#F00">
        блок 1
          <div style="position:absolute; top:20px; background-color:#0F0">
                блок 2
        </div>
</div>
```

Результат:

блок 1

блок 2

Перекрытие слоев в CSS

Z-INDEX

Блоки с абсолютным позиционированием могут быть расположены на разных уровнях, которые определяются свойством z-index.

Значения:

- auto «нулевой уровень», задаваемый по умолчанию;
- **отрицательное число** - объект располагается «ниже» нулевого уровня (удаляется вглубь, под элементы);
- **положительное число** - объект располагается «выше» нулевого уровня.

Пример:

```
...
<style type="text/css">
img{
        position:absolute;
        left:0px;
        z-index:-1;
}
</style></head>
<body>
<div style="position:relative">
<img src="..."/>
<p>Параграф нормального потока</p>
</div>
```

Результат:

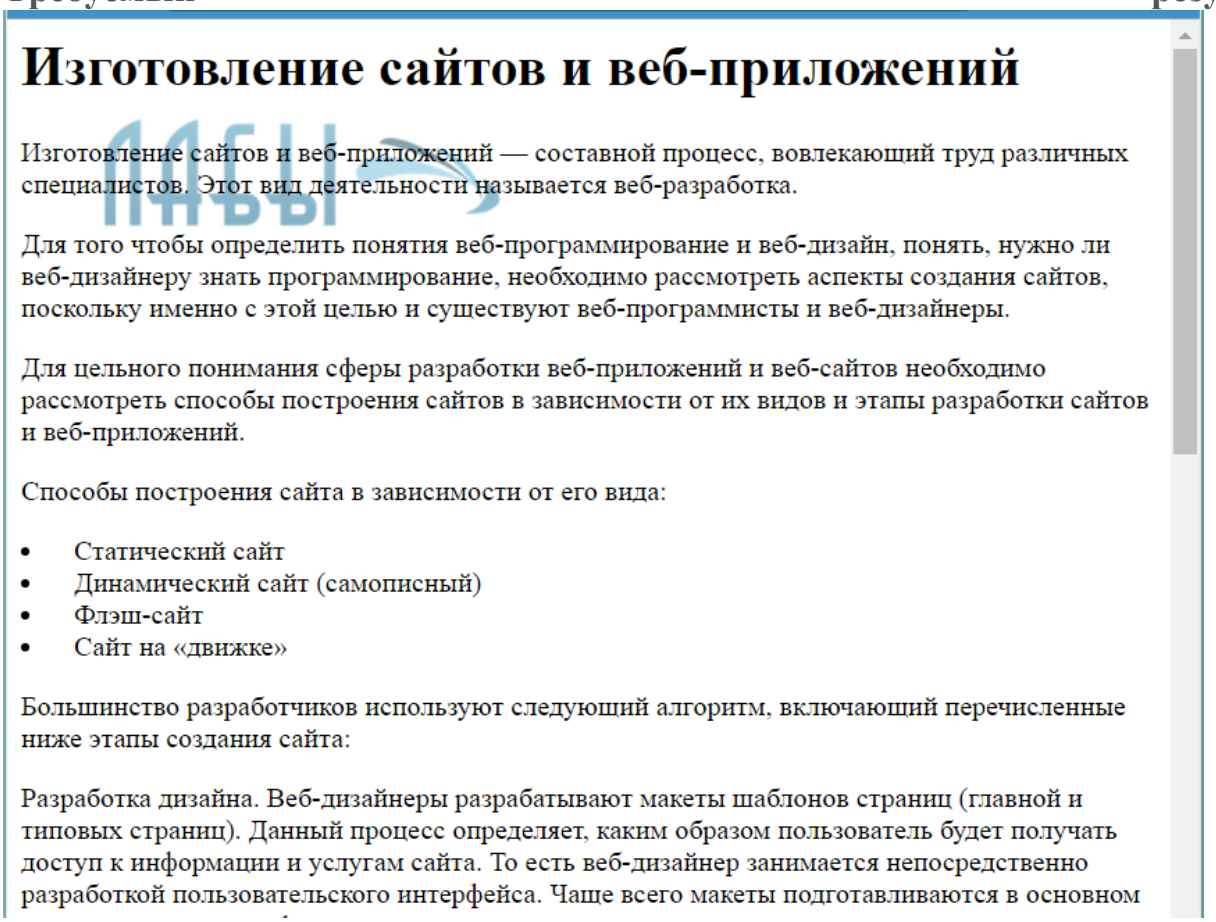
Задание: Скачайте файл.

Добейтесь эффекта, как на итоговом изображении, используя файл logo.png в качестве тега img.

Используйте следующие свойства CSS:

- position:
- z-index:
- top:
- left:
- opacity:0.5; /* прозрачность */
- filter: Alpha(Opacity=50); /* прозрачность для IE */

Требуемый результат:



Свойства блоков width, max-width, height, max-height

WIDTH

ширина блока

MAX-WIDTH

HEIGHT

высота блока

MAX-HEIGHT

Возможные значения для всех свойств:

- auto

- величина в пикселях
- %

Пример:

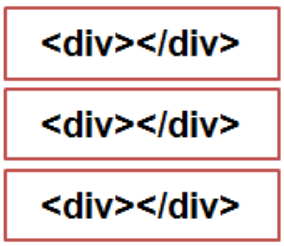
```
.div1{
        width:100%;
        background-color:#ccc;
}
.div2{
        width:50%;
        height:100px;
        background-color:#fc3;
}
</style></head>
<body>
<div class="div1">div1
        <div class="div2">div2</div>
</div>
```

Результат:

FLOAT

блочность или обтекание

В обычном состоянии элемент div - блочный:



При использовании float элемент теряет блочность и выравнивается согласно значению:

При этом следует иметь в виду, что если блок, следующий дальше, не имеет установленного свойства float, то сам блок поместится за блок со значением свойства float:left. А содержимое блока (текст) будет обтекать блок со значением свойства float:left:

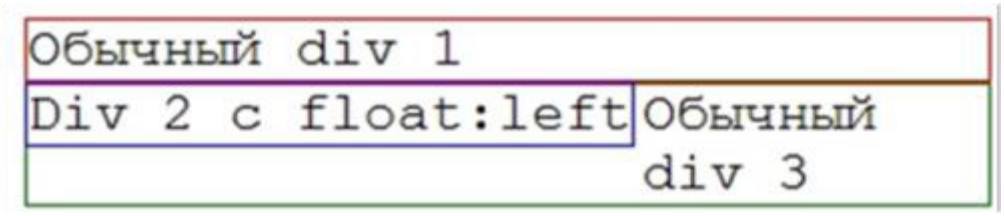


Рис. Расположение блоков при использовании свойства float

Возможные значения float:

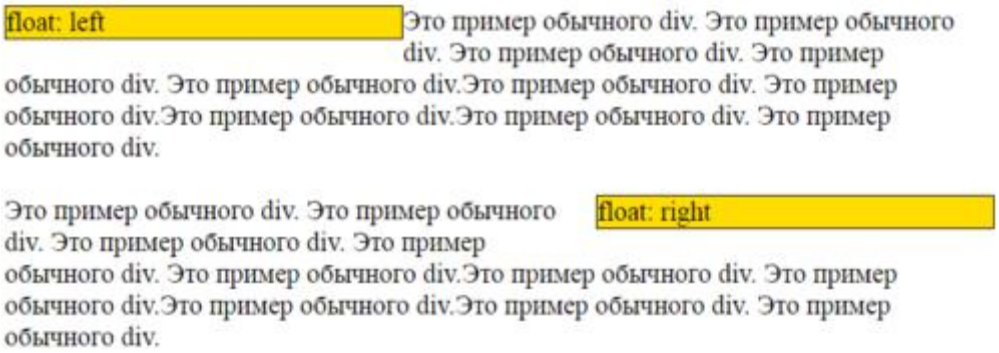
- left
  элемент выравнивается по левому краю, давая остальным элементам обтекать его справа
- right
  элемент выравнивается по правому краю, давая остальным элементам обтекать его слева
- none
  обтекание не задается (значение по умолчанию)
- inherit
  значение наследуется от родителя

Пример:

```
.left{
   float: left; /* Обтекание по правому краю
*/
   background: #fd0; /* Цвет фона */
   border: 1px solid black; /* Параметры
рамки */
   width: 40%; /* Ширина блока */
}
.right{
   float: right; /* Обтекание по правому краю
*/
   background: #fd0; /* Цвет фона */
   border: 1px solid black; /* Параметры
рамки */
   width: 40%; /* Ширина блока */
}
</style></head>
<body>
<div class="left">float: left</div>
<div>Это пример обычного div. Это пример обычного div. Это пример обычного div. Это пример обычного div. Это пример обычного div.Это пример обычного div. Это пример обычного div.Это пример обычного div.Это пример обычного div. Это пример обычного div.</div>
<br/>
<div class="right">float: right</div>
```

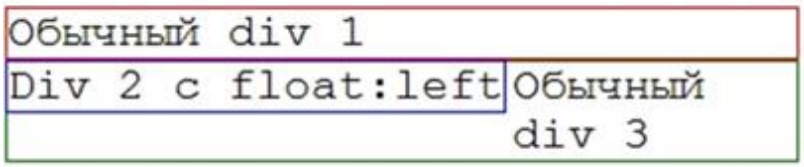
<**div**>Это пример обычного div. Это пример обычного div. Это пример обычного div. Это пример обычного div. Это пример обычного div.Это пример обычного div. Это пример обычного div.Это пример обычного div.Это пример обычного div. Это пример обычного div.</**div**>
</**div**>

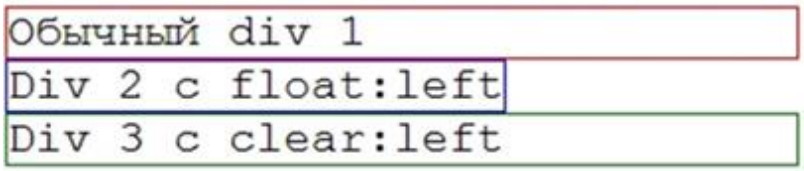
Результат:



CLEAR

отмена обтекания





Возможные значения:

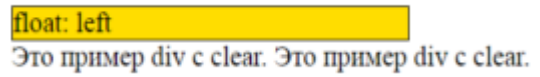
- left
  отмена обтекания элемента слева
- right
  отмена обтекания элемента справа
- both
  отмена обтекания элемента и справа и слева
- none
  нет отмены обтекания (значение по умолчанию)
- inherit
  значение наследуется от родителя

Пример:

```
.left{
   float: left; /* Обтекание по правому краю */
```

```
    background: #fd0; /* Цвет фона */
    border: 1px solid black; /* Параметры рамки */
    width: 40%; /* Ширина блока */
}
.clear{
    clear: left; /* Обтекание по правому краю */
}
</style></head>
<body>
<div class="left">float: left</div>
<div class="clear">Это пример div с clear. Это пример div с clear.</div>
```

Результат:



MARGIN-LEFT И MARGIN-RIGHT

или выравнивание блока по центру

 Значение auto для свойств margin-left и margin-right **выравнивает блок по центру** в своем контейнере (или браузере). При этом у элемента **должна быть задана ширина**.

Пример:

```
.center{
    background: #fd0; /* Цвет фона */
    border: 1px solid black; /* Параметры рамки */
    width: 300px; /* Ширина блока */
    margin-left: auto;
    margin-right: auto;
}
</style></head>
<body>
<div class="center">Это пример div</div>
```

Результат:

Вопросы для самоконтроля

1. Домен.
2. Виды доменов.
3. Доменные зоны.
4. Хостинг.
5. Технологии серверов.
6. Характеристики серверов.

Рекомендуемая литература

Основная литература:

1. Богун, В. В. Web-программирование. Интерактивность статических Интернет-сайтов с применением форм: учебное пособие для СПО / В. В. Богун. - Саратов: Профобразование, Ай Пи Ар Медиа, 2020. - 65 с. - ISBN 978-5-4488-0815-9, 978-5-4497-0481-8. - Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. - URL: https://www.iprbookshop.ru/92633.html

2. Гладкий, А. А. Веб-самоделкин. Как самому создать сайт быстро и профессионально: практическое пособие: [16+] / А. А. Гладкий. – Москва; Берлин: Директ-Медиа, 2020. – 266 с.: ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=577164

3. Хромушин, В. А. Сборник примеров HTML страниц: учебное пособие / В. А. Хромушин, Р. В. Грачев, Н. Д. Юдакова. - Тула: ТулГУ, 2022. - 192 с. - ISBN 978-5-7679-5040-9. - Текст: электронный // Лань: электронно-библиотечная система. - URL: https://e.lanbook.com/book/264062

Дополнительная литература:

1. Адамс, Д. Р. Основы работы с XHTML и CSS: учебник / Д. Р. Адамс, К. С. Флойд. - 3-е изд. - Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. - 567 с. - ISBN 978-5-4497-0907-3. - Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. - URL: https://www.iprbookshop.ru/102037.html

2. Беликова, С. А. Основы HTML и CSS: проектирование и дизайн веб-сайтов: учебное пособие по курсу «Web-разработка» / С. А. Беликова, А. Н. Беликов. - Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2020. - 174 с. - ISBN 978-5-9275-3435-7. - Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. - URL: https://www.iprbookshop.ru/100186.html

3. Маркин, А. В. Web-программирование: учебное пособие для СПО / А. В. Маркин. - Саратов, Москва: Профобразование, Ай Пи Ар Медиа, 2021. - 267 с. - ISBN 978-5-4488-1198-2, 978-5-4497-1031-4. - Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. - URL: https://www.iprbookshop.ru/107576.html