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, 2024

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.

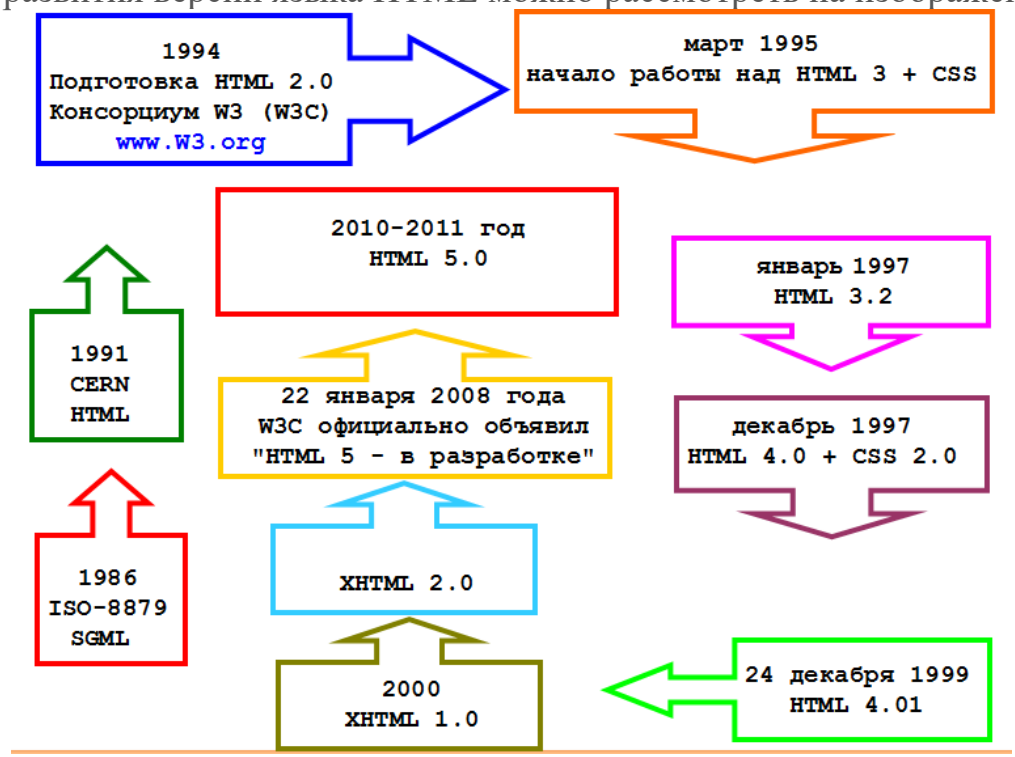
Цель: изучить способы создания простейших веб страниц.

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**:

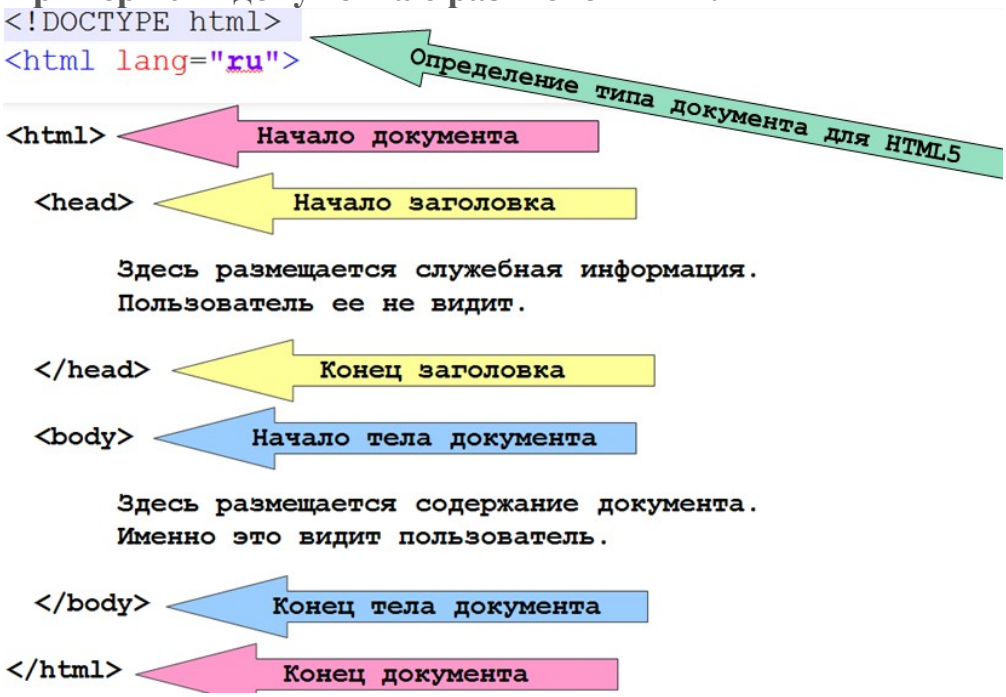
- Целиком пара открывающий-закрывающий тег называется **элементом** или контейнером.

- 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.

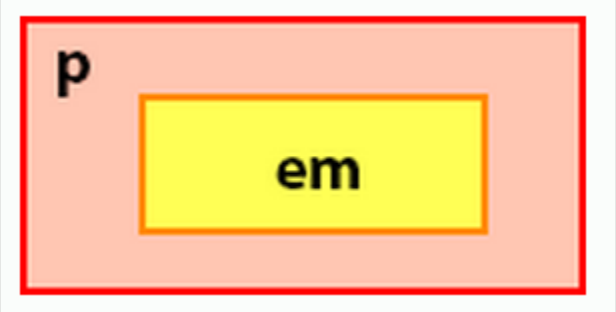
Важно: Для того, чтобы не было проблем с кодировкой символов, необходимо указать тип кодировки в области head (т.е. после открывающего тега head - головная часть документа)

```
<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>
```

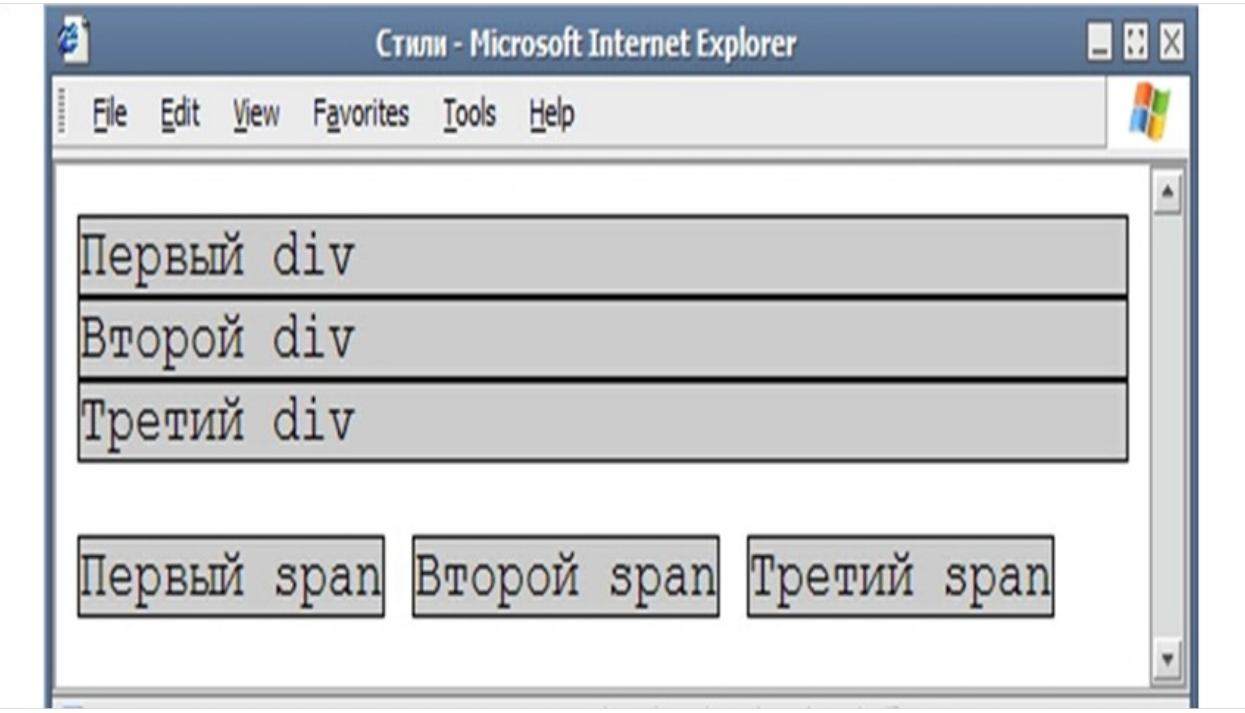
Схематично родительские теги и вложенные (дочерние) выглядят так:



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:

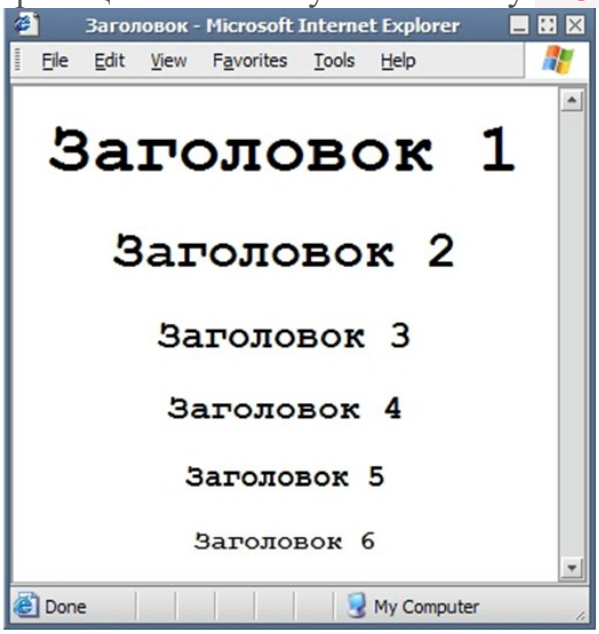


Элементы форматирования текста

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>:

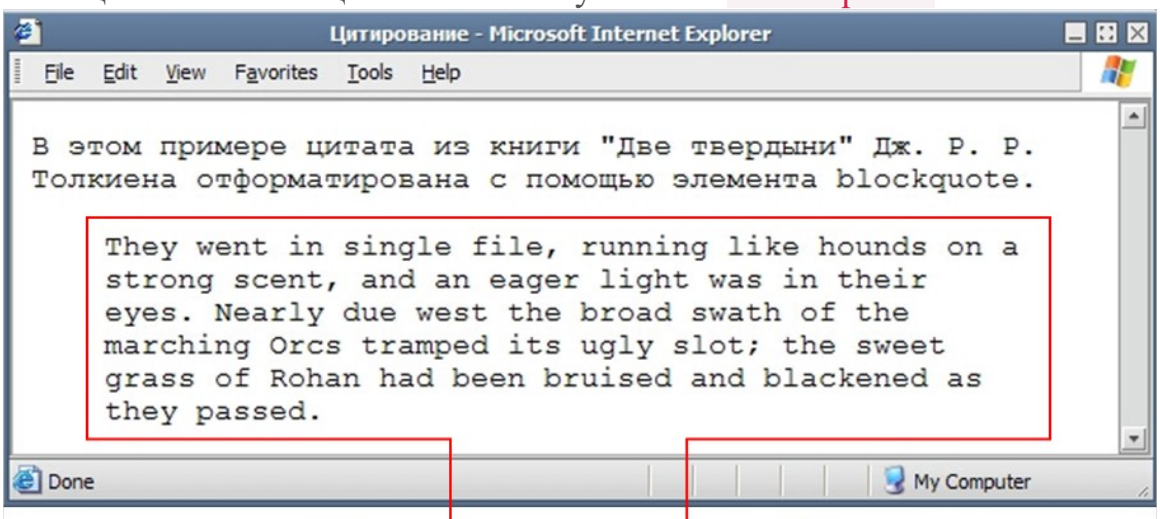
```
<h1>Заголовок 1</h1>
<h2>Заголовок 2</h2>
<h3>Заголовок 3</h3>
<h4>Заголовок 4</h4>
<h5>Заголовок 5</h5>
<h6>Заголовок 6</h6>
```



БЛОЧНАЯ ЦИТАТА

<blockquote></blockquote>

Для размещения в тексте цитаты используется тег <blockquote>:



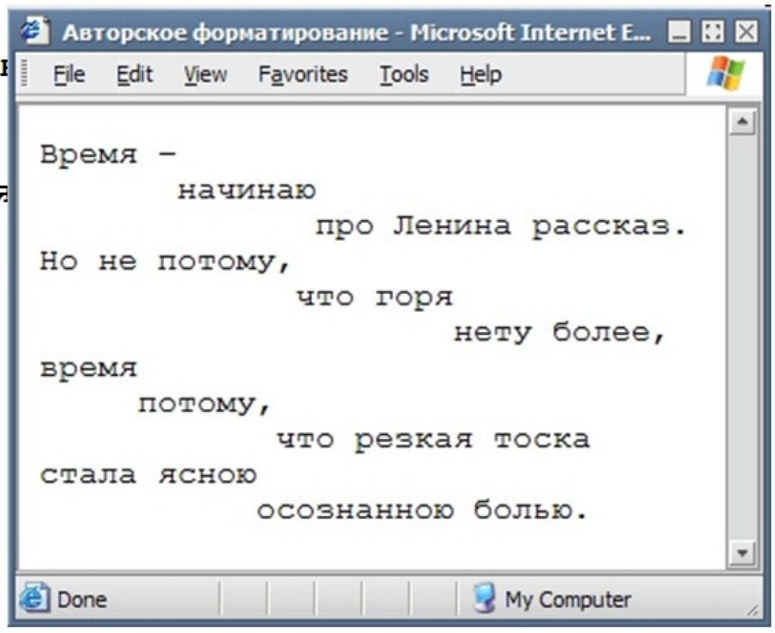

```
<blockquote>
They went in single file, running like hounds on a
strong scent, and an eager light was in their eyes.
Nearly due west the broad swath of the marching Orcs
tramped its ugly slot; the sweet grass of Rohan had
been bruised and blackened as they passed.</blockquote>
```

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

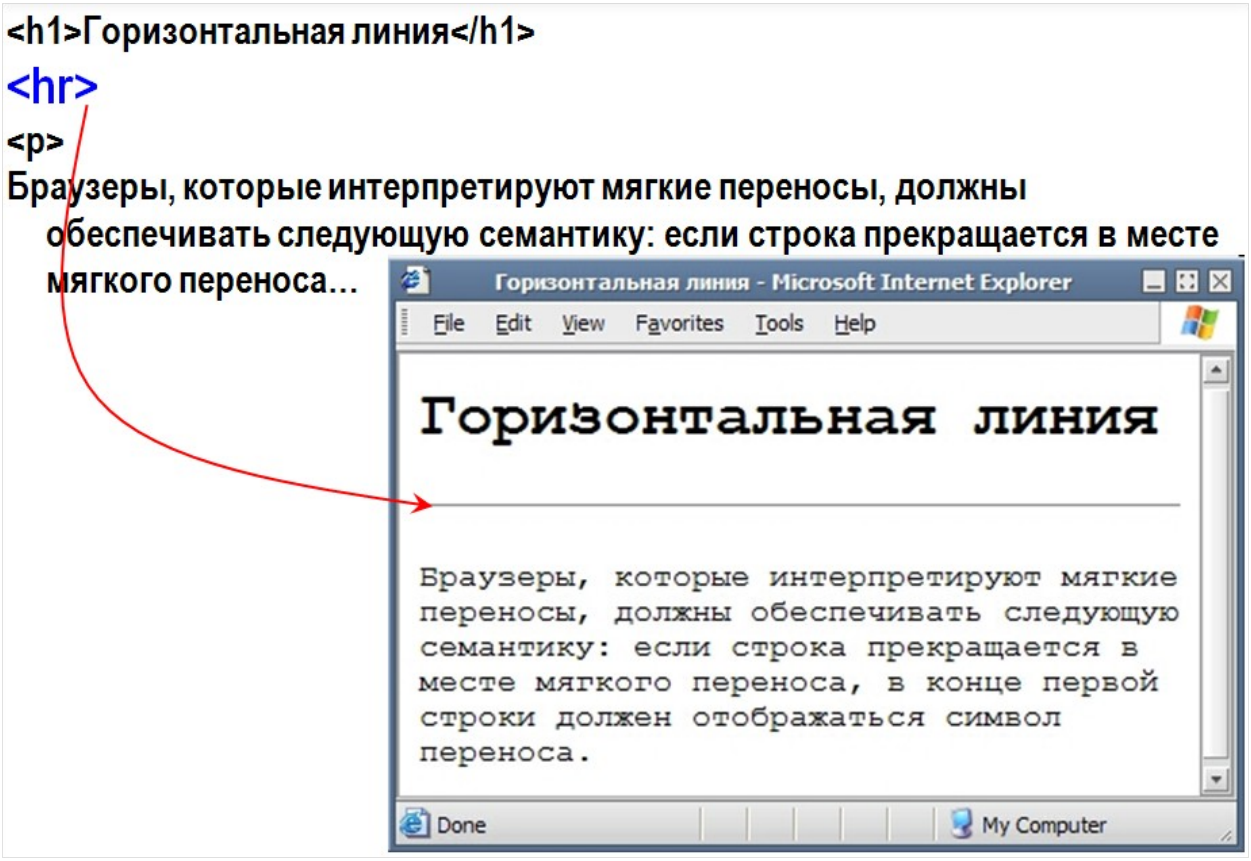
```
<i> - курсив
<b> - полужирный
<u> - подчеркнутый
<strike> - перечеркнутый
<tt> - моноширинный
<big> - увеличить шрифт
<small> - уменьшить шрифт
<sup> - надиндекс
<sub> - подиндекс
```

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>

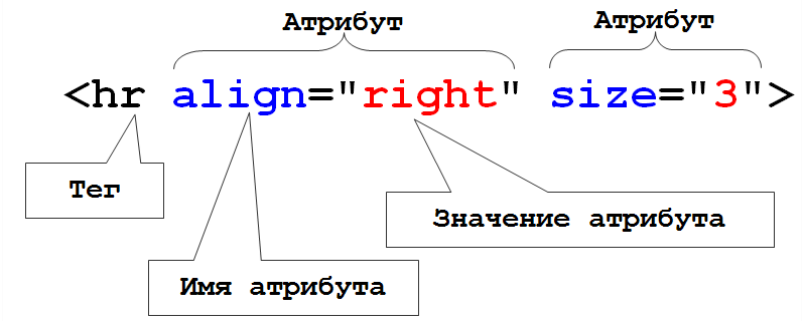
Данный элемент служит для разделения некоторых структурных элементов текста друг от друга. Либо может быть использован просто как эстетический элемент оформления документа:



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:

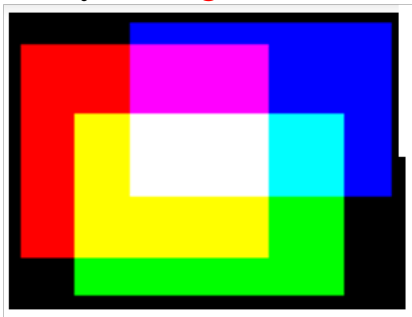
- ***bgcolor*** - задний фон страницы и
- ***text*** - цвет текста на всей странице.

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**:

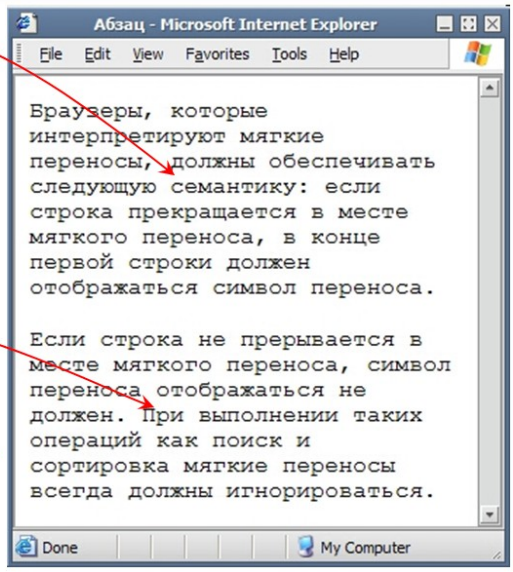
Например, тег ***del*** делает содержимое перечеркнутым, при этом указывая прежде всего на смысл удаления текста.

```
<em> - выделение важных фрагментов курсивом
<strong> - 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

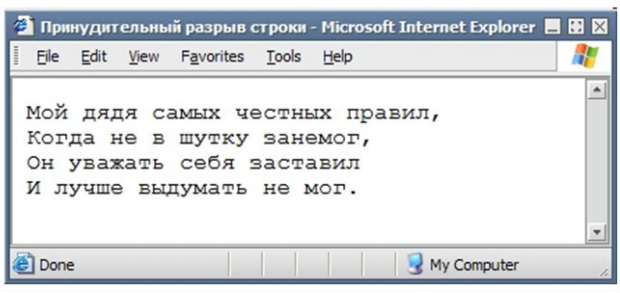
- Для перехода на другую строку текста служит пустой элемент ***
***.
- Тогда как для выделения в тексте абзаца служит элемент ***<p>***, содержимое которого и является сам абзац. Перед абзацем и после него добавляются отступы, но красная строка при этом не предусмотрена.

```
<P>
  Браузеры, которые
  интерпретируют мягкие
  переносы, должны
  обеспечивать следующую
  семантику: если строка
  прекращается в месте
  мягкого переноса, в конце
  первой строки должен
  отображаться символ
  переноса.
</P>
<P>
  Если строка не прерывается
  в месте мягкого переноса,
  символ переноса
  отображаться не должен. При
  выполнении таких операций
  как поиск и сортировка
  мягкие переносы всегда
  должны игнорироваться.
```



**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. Гиперссылки.

Цель: изучить способы создания гиперссылок, переходов.

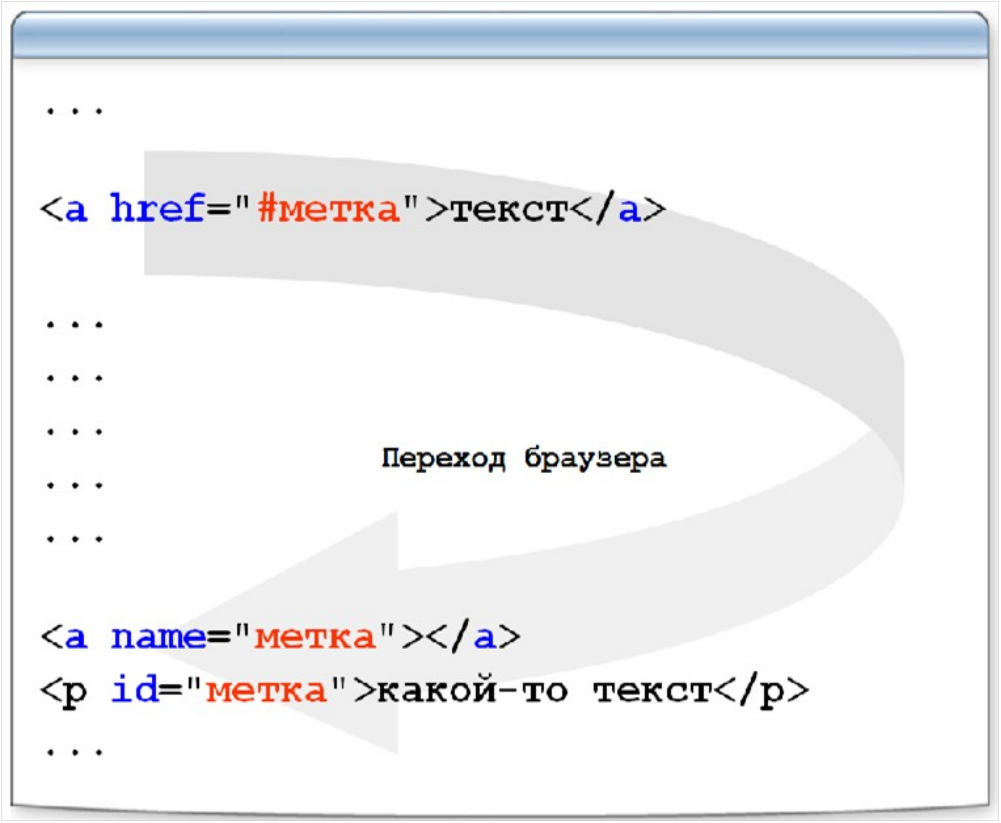
Оборудование: ПК, ОС Windows, MS Office, Notepad++, Visual Studio Code, методические указания по выполнению лабораторного занятия.

Ход работы:

Внутренние ссылки в HTML (якорная ссылка)

Представим себе реферат, состоящий из нескольких глав, изложенный в электронном виде на одной веб-странице. Как бы красиво ни был оформлен текст, для того чтобы искать главы придется использовать полосу прокрутки и спускаться «вниз» по странице в поисках необходимой главы.

В таком случае обычно в самом начале страницы делается оглавление из гиперссылок. В HTML такие ссылки, которые организовывают переходы внутри одной страницы, называются **внутренними или якорными ссылками**.



Рассмотрим **механизм создания внутренних ссылок в HTML**. Он состоит из двух шагов:

1. **Создание закладок или якорей (на которые необходимо переходить по ссылкам):**

1 способ:

```
<a name="название_закладки"></a>
<p>Текст для закладки</p>
```

В качестве якоря служит тег a с атрибутом name - название якоря (закладки)

2 способ:

```
<p id="название_закладки">Текст для закладки</p>
```

Для обозначения якоря используется атрибут id, добавляемый к тегу (теги могут быть практически любые: div, span, p, h…)

2. Создание ссылок на якоря (на закладки):

```
<a href="#название_закладки">Текст ссылки</a>
```

Знак *шарп* или *решетка* (#) ставится обязательно перед названием якоря

Пример: на веб-странице, состоящей из трех глав реферата создать оглавление на 3 главы

Решение:

```
<ol>
<!-- создание ссылок -->
<li><a href="#glava1">Глава 1</a></li>
<li><a href="#glava2">Глава 2</a></li>
</ol>
<!-- создание якоря -->
<h1 id="glava1">Глава 1. "Язык HTML - история"</h1>
<p>Текст главы</p>
...
<!-- создание якоря -->
<h1 id="glava2">Глава 2. "Структура HTML-страницы"</h1>
<p>Текст главы</p>
...
```

Скопируйте текст кода, расположенный ниже. Создайте новый документ в блокноте (notepad++), вставьте код на созданную страницу и сохраните в формате html. Выполните задание.

Задание:

1. Измените внешнюю ссылку на внутреннюю: измените значение атрибута href на "#footer", а текст самой ссылки - с *«Фото кота»* на *«Перейти вниз»*.
2. Удалите атрибут target="_blank" из ссылки, так как он служит для того, чтобы открывать ссылку в новой вкладке или в новом окне.
3. Добавьте атрибут id="footer" для элемента <footer> внизу страницы.

Код:

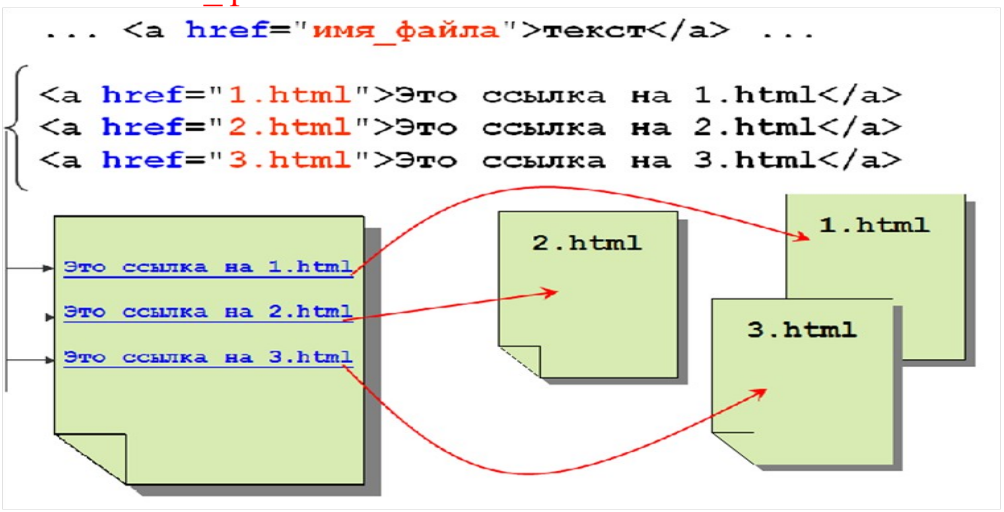
```
<h2>CatPhotoApp</h2>
<main>
  <a href="http://cats.ru/cat1.jpg" target="_blank" rel="noopener noreferrer">Фото кота</a>
 <img src="https://bit.ly/fcc-relaxing-cat" alt="Симпатичный оранжевый кот.">
  <p>Ко́шка, или дома́шняя ко́шка (лат. Félis silvéstris cátus), - домашнее животное, одно из наиболее популярных[1] (наряду с собакой) «животных-компаньонов»[2][3][4].</p>
 <p>С точки зрения научной систематики, домашняя кошка - млекопитающее семейства кошачьих отряда хищных. Ранее домашнюю кошку нередко
```

рассматривали как отдельный биологический вид. С точки зрения современной биологической систематики домашняя кошка (Felis silvestris catus) является подвидом лесной кошки (Felis silvestris).

Являясь одиночным охотником на грызунов и других мелких животных, кошка - социальное животное, использующее для общения широкий диапазон звуковых сигналов, а также феромоны и движения тела.
В настоящее время в мире насчитывается около 600 млн домашних кошек[8], выведено около 200 пород, от длинношёрстных (персидская кошка) до лишённых шерсти (сфинксы), признанных и зарегистрированных различными фелинологическими организациями.
</main>

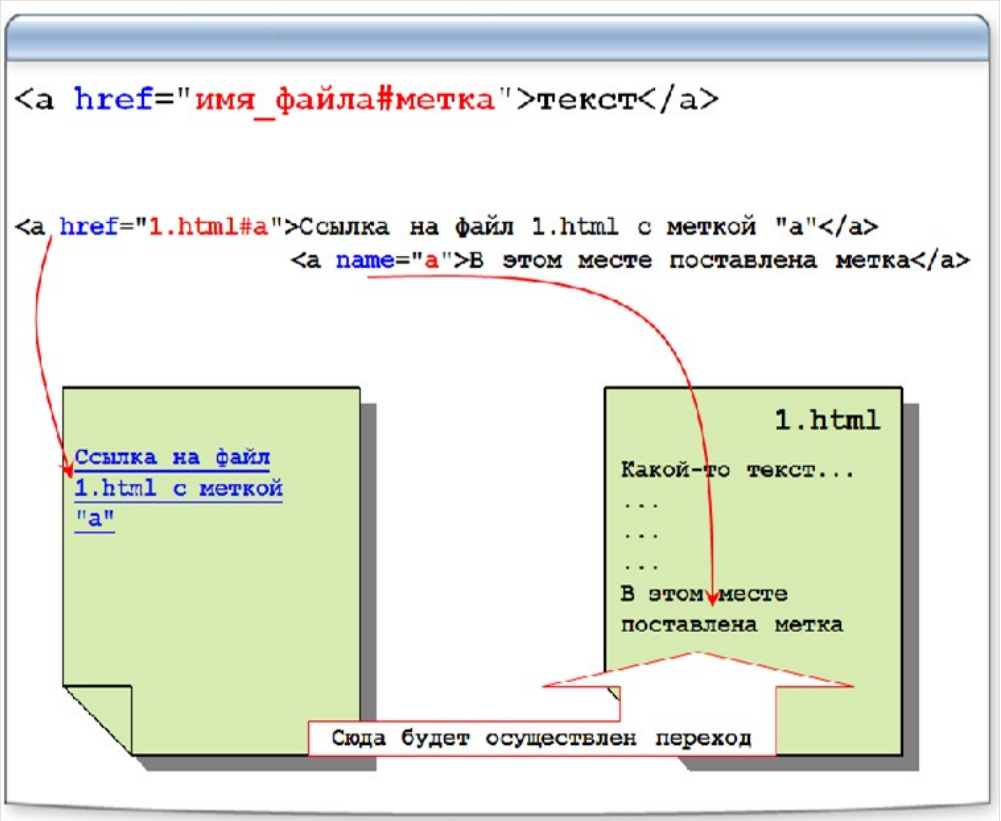
<footer>Copyright Сайт про котов и кошек</footer>

Оформление ссылок HTML для переходов к другим документам
Синтаксис:
текст



ПЕРЕХОД К ДРУГОМУ ДОКУМЕНТУ С ЯКОРЕМ

Иногда необходимо организовать ссылку не просто на другой документ, а на конкретное место - якорь - другого документа.



Синтаксис:

текст

Пример: организовать ссылку на файл *1.html*, а, конкретнее, на якорь, расположенный в данном файле

Выполнение:

Файл с ссылкой:

...

Ссылка

...

Файл 1.html:

...

Якорь

...

Абсолютные ссылки HTML

Синтаксис:

текст

Рассмотрим примеры:

Ссылка на html-файл по протоколу HTTP:

текст

Ссылка на zip-файл по протоколу HTTP:

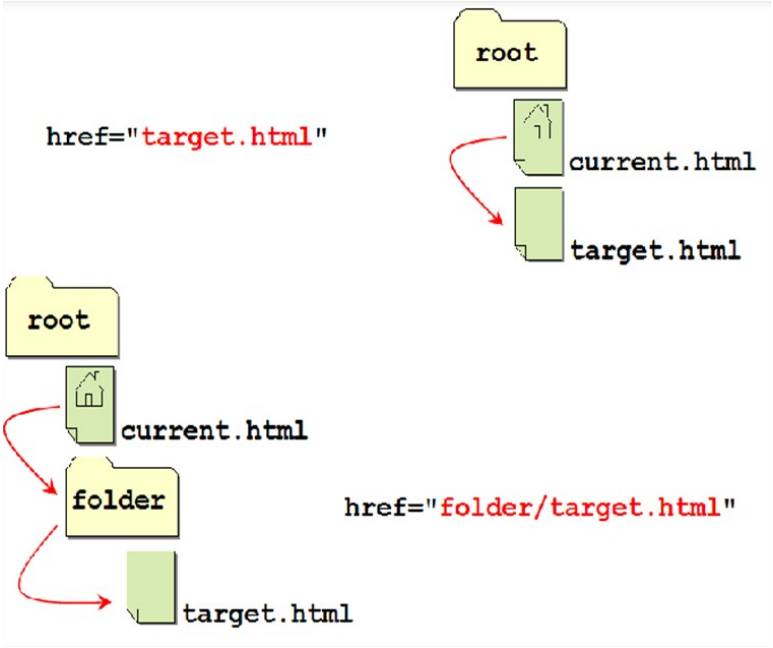
текст

Ссылка на e-mail по протоколу mailto:

текст

Относительный путь ссылок HTML

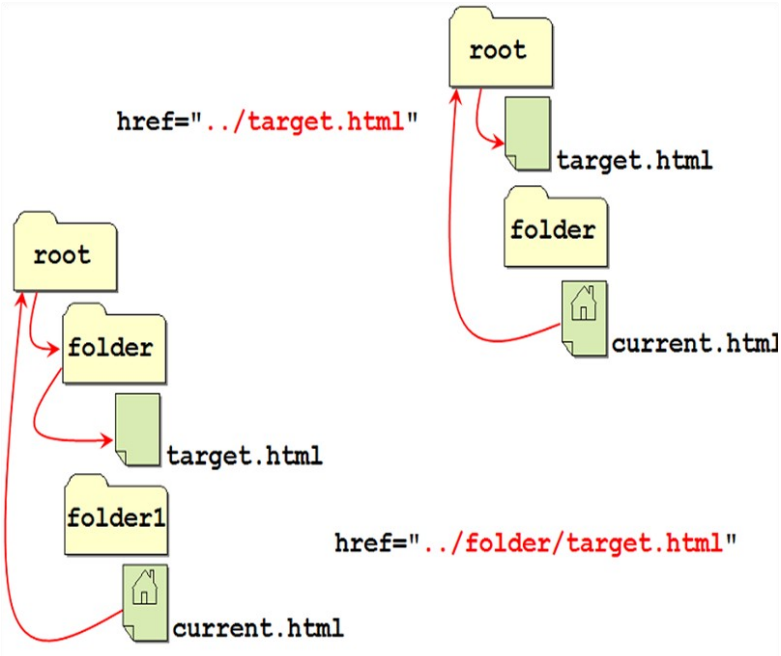
Рассмотрим примеры того, как правильно должны быть оформлены относительные ссылки.



На изображении файлом для ссылки является *target.html*. Сама же ссылка оформляется в файле *current.html*.

Атрибут href гиперссылки должен иметь таковое значение (как на картинке) при указанных расположениях файлов.

Рассмотрим более сложный вариант расположения файлов:



В каком окне открывать ссылку?

За это отвечает атрибут тега гиперссылки - target.

Рассмотрим возможные значения атрибута:

_blank - открывает документ в новом окне

_self - открывает документ в том же окне

_parent - открывает документ в родительском окне
_top - открывает документ на весь экран

Изменение цвета гиперссылки

За цвет гиперссылки отвечают ее атрибуты: link, alink, vlink

```
<body bgcolor="white" text="black" link="blue"
   alink="red" vlink="purple">

<body bgcolor="#ffffff" text="#000000"
   link="#0000ff" alink="#ff0000"
   vlink="#800080">

      Атрибуты <body>:

      link – цвет неотработанной ссылки
      alink – цвет активной ссылки
      vlink – цвет отработанной ссылки
```

Вопросы для самоконтроля

1. Адаптивность.
2. Адаптивная верстка.

Лабораторная работа №3. 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

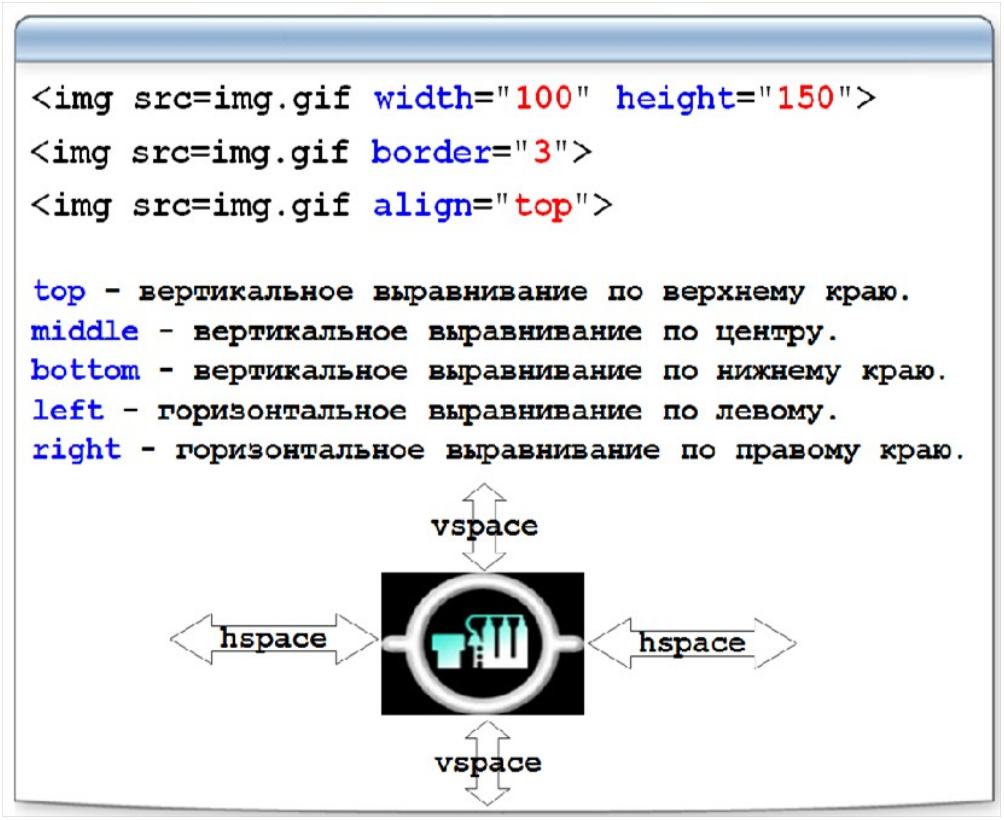
- alt - подпись
- title - всплывающая подпись

Выравнивание по вертикали:

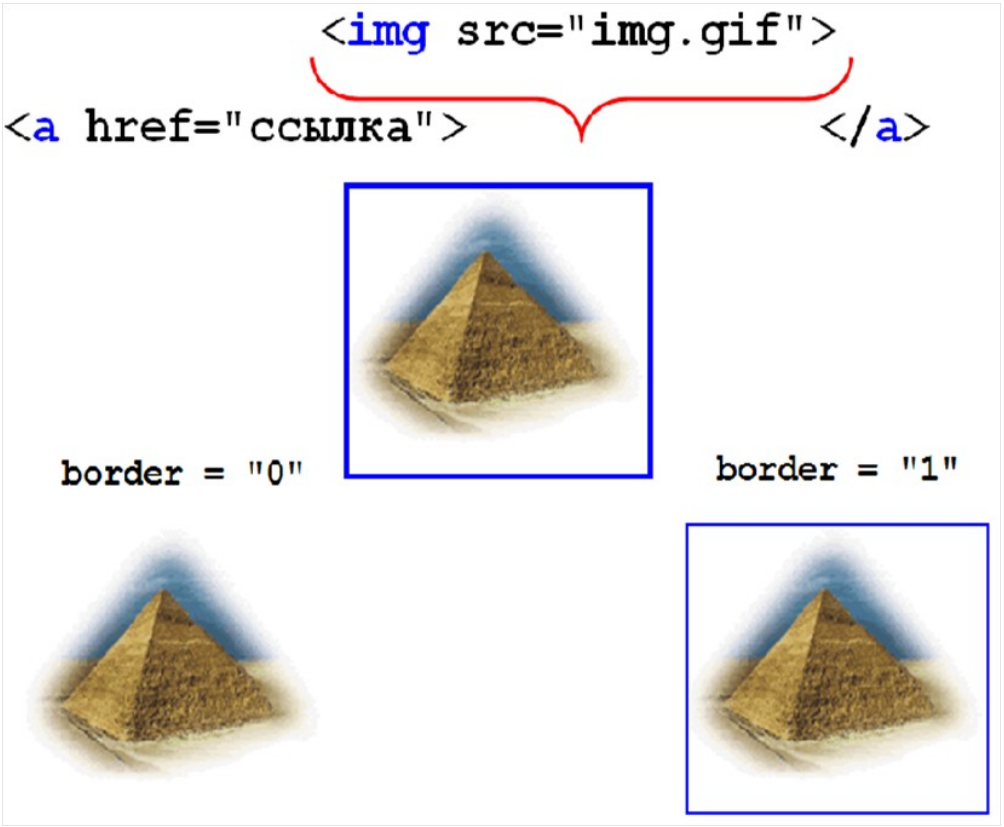
- align="top"
- align="middle"
- align="bottom"

Выравнивание по горизонтали:

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



ИЗОБРАЖЕНИЕ КАК ССЫЛКА



ФОНОВОЕ ИЗОБРАЖЕНИЕ СТРАНИЦЫ

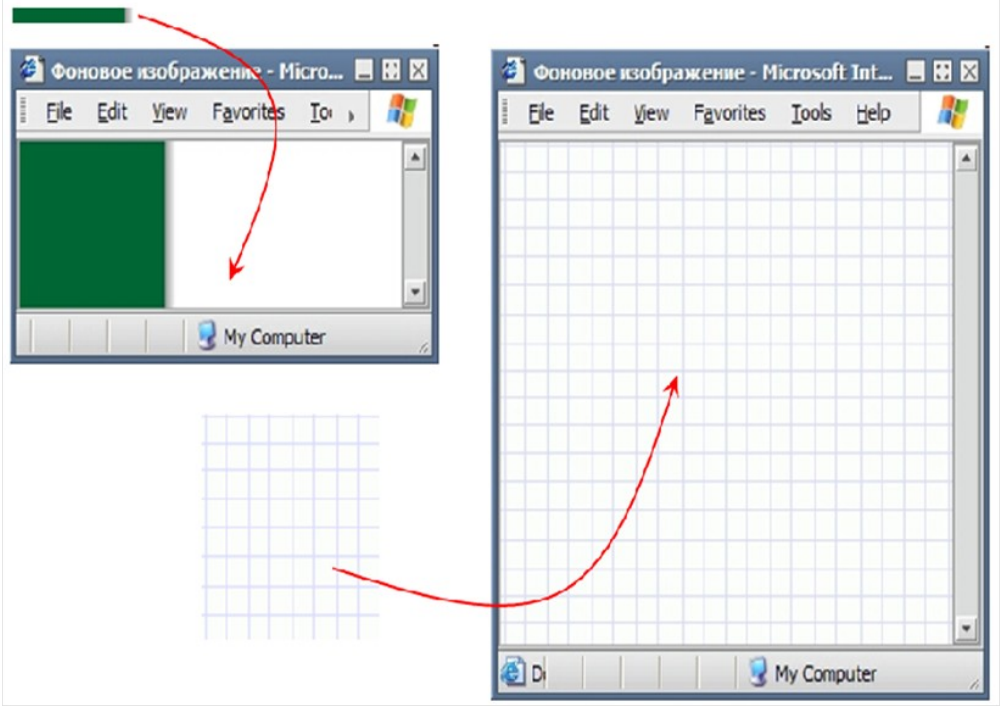
Синтаксис:

<**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 №4. Создание списков.

Цель: изучить способы создание маркированных и нумерованных списков.

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

Ход работы:

Создание списков в HTML

В HTML поддерживается работа со всеми вилами списков, рассмотрим их

НЕУПОРЯДОЧЕННЫЙ СПИСОК HTML

Неупорядоченный или ненумерованный список HTML содержит следующие теги:

Синтаксис:

```
<ul>
  <li>элемент 1</li>
  <li>элемент 2</li>
  <li>элемент 3</li>
  …
  <li>элемент n</li>
</ul>
```

Т.е. каждый пункт списка должен начинаться тегом li и завершаться его закрывающей парой, вместо которого будет отображаться маркер (диск, окружность или квадрат - зависит от предустановок браузера)

Чтобы задать вид маркера, необходимо использовать атрибут type:

```
<ul type="disk"> заполненный диск
<ul type="circle"> окружность
<ul type="square"> квадрат
```

УПОРЯДОЧЕННЫЙ СПИСОК HTML

Упорядоченный список в HTML или нумерованный список практически такой же, как и ненумерованный, за исключением первого главного тега:

Синтаксис:

```
<ol>
  <li>элемент 1</li>
  <li>элемент 2</li>
  <li>элемент 3</li>
  …
  <li>элемент n</li>
</ol>
```

Для нумерованного списка тоже есть различные варианты нумерации:

```
<ol type = "a">
<ol type = "A">
<ol type = "I">
<ol type = "1">
```

Можно также применять к отдельному пункту:

```
<li type = "a">
```

Для того, чтобы продолжить список с определенного номера:

```
<ol start = "5">
```

МНОГОУРОВНЕВЫЙ СПИСОК HTML

Списки с определениями или многоуровневые списки в HTML создаются следующим образом:

```
<dl>
<dt>Пункт 1
  <dd>Элемент пункта 1
  <dd>Элемент пункта 1
<dt>Пункт 2
  <dd>Элемент пункта 2
<dt>Пункт 3
  <dd>Элемент пункта 3
</dl>
```

Результат:

Пункт 1
Элемент пункта 1
Элемент пункта 1
Пункт 2
Элемент пункта 2
Пункт 3
Элемент пункта 3

КРАСИВЫЙ СПИСОК HTML ИЛИ СМЕШАННЫЙ СПИСОК

Чтобы сделать список максимально красивым, можно использовать в одном списке разные виды списков. Рассмотрим пример:

```
<dl><strong>Смешанный список</strong>
        <dt><i>Новость дня</i>
                <dd>
                        <li>Сегодня идет дождь</li>
                        <li>Дождь будет идти весь день</li>
        <dt><i>Новость ночи</i>
                <dd>
                        <li>Ночью будет идти дождь</li>
                        <li>Завтра начнется новый день</li>
</dl>
```

Результат:

Смешанный список

Новость дня

- Сегодня идет дождь
- Дождь будет идти весь день

Новость ночи

- Ночью будет идти дождь
- Завтра начнется новый день

- Создайте смешанный список

- Используйте нумерованный и маркированный список html и список определений так, чтобы:
    - вместо знака - отображался маркер
    - вместо порядковых чисел и букв (1, 2, 3, a, b, c) отображались соответствующие им числа и буквы автоматически
- знак -, порядковые числа и буквы из текста удалить

Вопросы для самоконтроля

1. Использование тегов.
2. Разбиение страницы на блоки.
3. Тег div.
4. Тег span.
5. Тег header.
6. Тег nav.
7. Тег aside.
8. Тег article.
9. Тег section.
10. Тег footer.
11. Тег main.

Лабораторная работа №5. 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

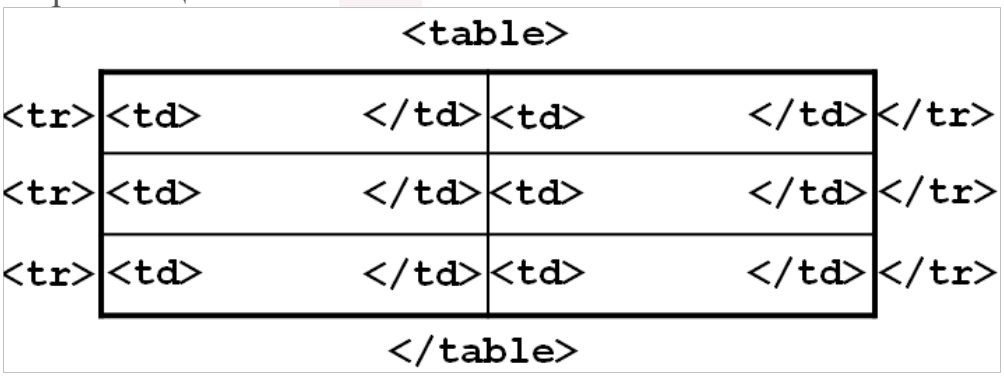
Добавим границу для таблицы - атрибут border:

```
<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



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

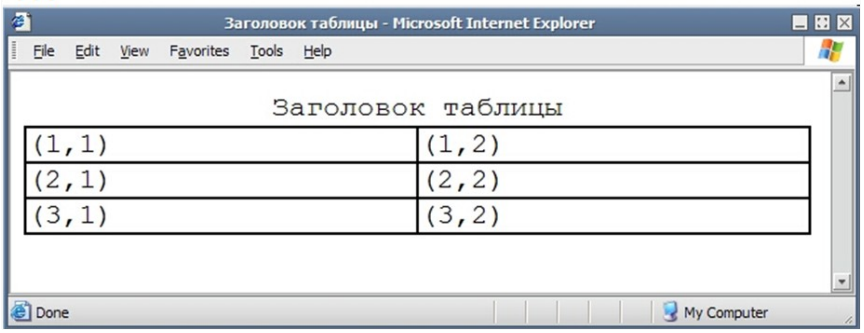
		отображаться без переносов, одной сплошной строкой

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:

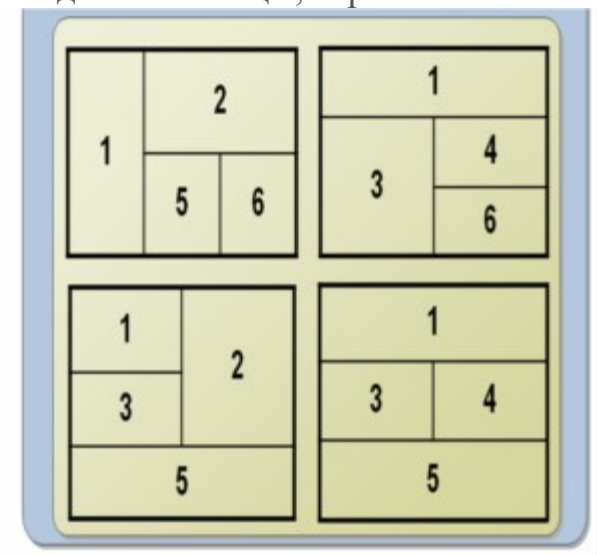
```
<table>
<tr>
  <td rowspan="2"> </td>
  <td> </td>
</tr>
<tr>
  <td> </td>
</tr>
</table>
```

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

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:

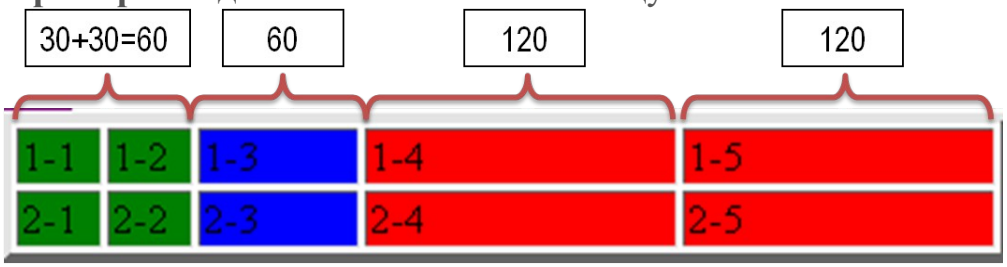


Группировка ячеек: COLGROUP

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.

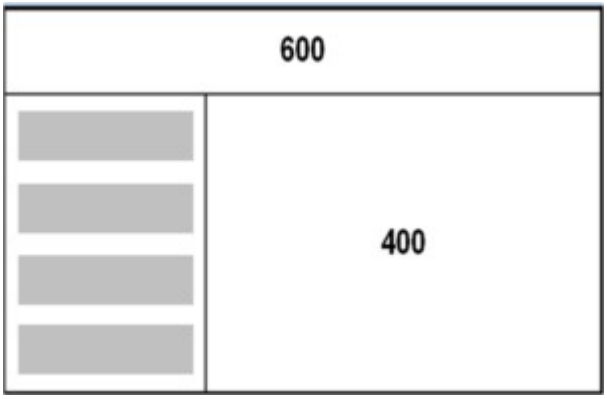
Пример: создайте вложенные таблицы по образцу:

Выполнение:

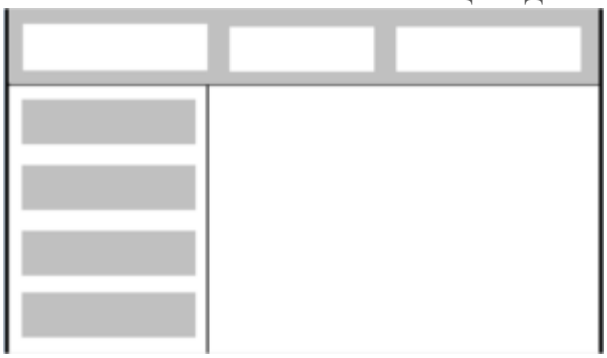
```
<TABLE border="4" bgcolor="yellow">
<tr>
  <td>Таблица 1</td>
  <td>
    <TABLE>
    <tr>  <td>Таблица 2</td><td>Ячейка 2-2</td>  </tr>
    <tr>  <td>Ячейка 3-2</td><td>Ячейка 4-2</td> </tr>
    </TABLE>
  </td>
<tr>
  <td>Ячейка 3-1</td>
  <td>Ячейка 4-1</td>
</tr>
</TABLE>
```

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 №6. Фреймовая структура.

Цель: изучить способы создания фреймовой структуры на сайте.

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

Ход работы:

Фреймовая структура в HTML

С самого начала следует сказать, что прием использования фреймовой структуры хоть и очень удобен в некоторых случаях, тем не менее, использование структуры не желательно для коммерческих проектов.

ЭЛЕМЕНТ FRAMESET

Синтаксис деления по вертикали (на колонки):

<**FRAMESET** cols="n%,n%"> … </**FRAMESET**>

где n - ширина фреймов в процентах слева направо

Синтаксис деления по горизонтали:

<**FRAMESET** rows="n%,n%"> … </**FRAMESET**>

где n - высота фреймов в процентах сверху вниз

Фреймов в структуре может быть не два, а более.

```
<frameset cols="10%, 60%, 30%">

<frameset rows="80, 200, *">

<frameset cols="1*, 250, 3*">
```

Файл с фреймовой структурой называется **файлом-раскладкой** и обычно называется *index.html*

Как происходит загрузка файлов во фрейм рассмотрим на примере.

Синтаксис:

<**FRAME** src="Имя файла.html" name="имя фрейма">

Пример: Создать файл с фреймовой структурой с двумя колонками: в левую (ширина 25%) загружать файл *menu.html*, в правую (ширина 70%) загружать файл *content.html*

Выполнение:

```
<html>
<head>
        <title>Пример</title>
</head>
<frameset cols="25%, 75%">
        <frame src="menu.html">
        <frame src="content.html">
</frameset>
</html>
```

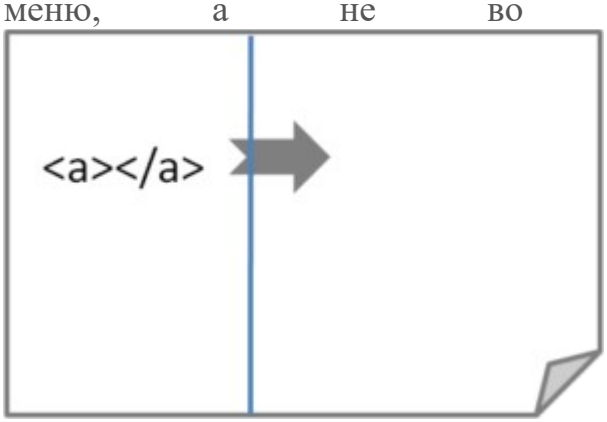
Создать файл с фреймовой структурой (имя файла *index.html*) с тремя колонками (ширина: **25%**, **50%**, *). Создать три файла для загрузки в фреймы, расположить их в той же папке, в которой находится *index.html*: 1. *menu.html* (в нем расположить заголовок h1 *«Меню»*), 2. *content.html* (в нем расположить заголовок h1 *«Контент»*), 3. *news.html* (в нем расположить заголовок h1 *«Новости»*).
Поменяйте расположение с колонок на ряды (горизонтальное деление), посмотрите на результат.

Атрибуты тега frameset:
frameborder - значение 1 или 0 (есть или нету)
border - значение размера границы
bordercolor - цвет границы
framespacing - ширина граней фреймов в пикселях (только в IE)

```
<frameset cols="25%, 75%" frameborder="1" bordercolor="red" border="1">
```

Атрибуты элемента frame:
name - имя фрейма
noresize - запрещает изменение размеров для определенного фрейма.
scrolling - управляет прокруткой внутри одной области (yes, no, auto).
marginheight - задает величину отступа фрейма от верхнего и нижнего краев страницы.
marginwidth - создает поля слева и справа
frameborder - указывает, нужна или нет рамка вокруг фрейма (0 и 1)
bordercolor – цвет рамки

Правила создания ссылки во фреймах в html
Если представить структуру, в которой слева находится фрейм с меню сайта, а справа фрейм, предназначенный для вывода содержимого выбранных пунктов меню, то становится очевидна следующая проблема:
при щелчке на пункте меню содержимое будет загружаться в тот же фрейм с меню, а не во фрейм, расположенный справа



Важно: для загрузки пункта меню в определенный фрейм необходимо:

1. добавить название необходимого фрейма при помощи атрибута name (в файле-раскладке)
   Пример:
   ```
   <frame src="content.html" name="mainFrame">
   ```

2. в файле с меню в гиперссылке добавить атрибут target

Пример:

<a href="glava1.html" target="mainFrame" rel="noopener noreferrer">

Посмотрим полный код обоих файлов: Файл index.html

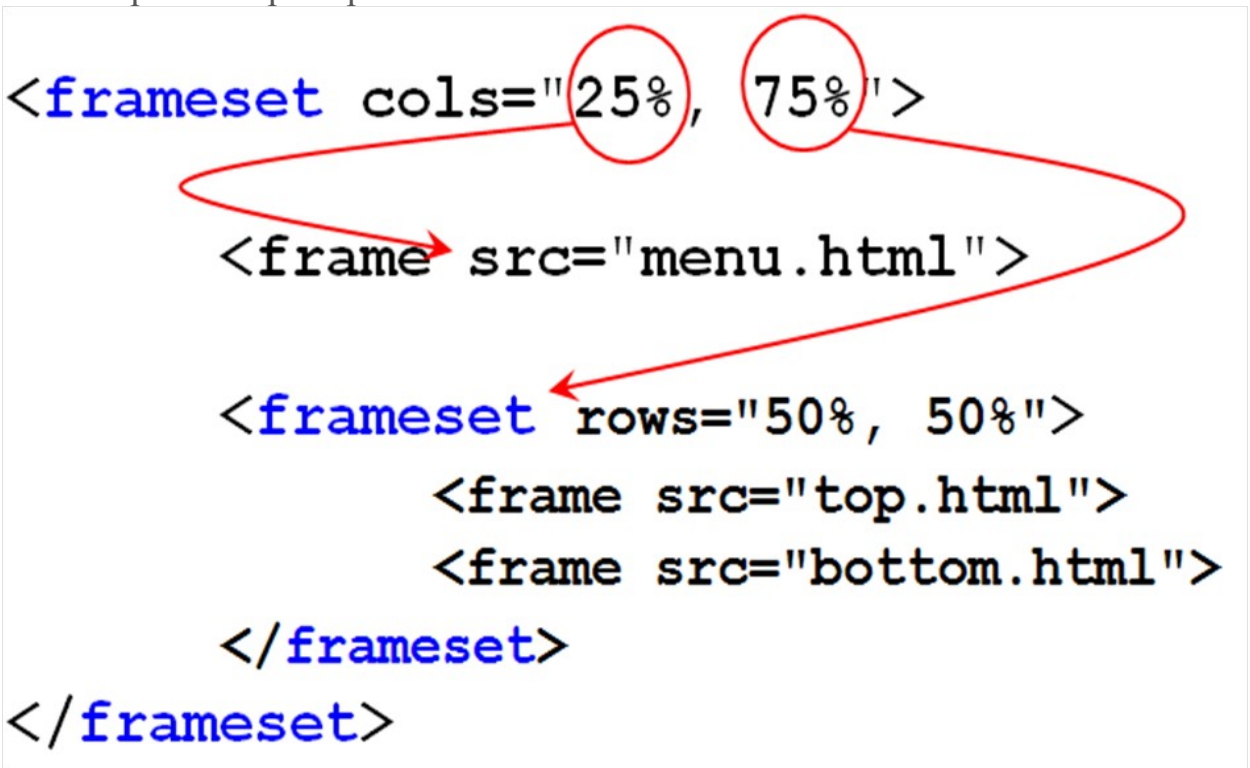
```
<html>
...
<frameset cols="25%, 75%">
        <frame src="menu.html">
        <frame src="content.html" name="mainFrame">
</frameset>
</html>
```

Файл menu.html:

```
<html>
...
<body>
        <h2>Меню:</h2>
<ul>
<li><a href="glava1.html" target="mainFrame" rel="noopener noreferrer">Глава 1</a></li>
<li><a href="glava2.html" target="mainFrame" rel="noopener noreferrer">Глава 2</a></li>
</ul>
</html>
```

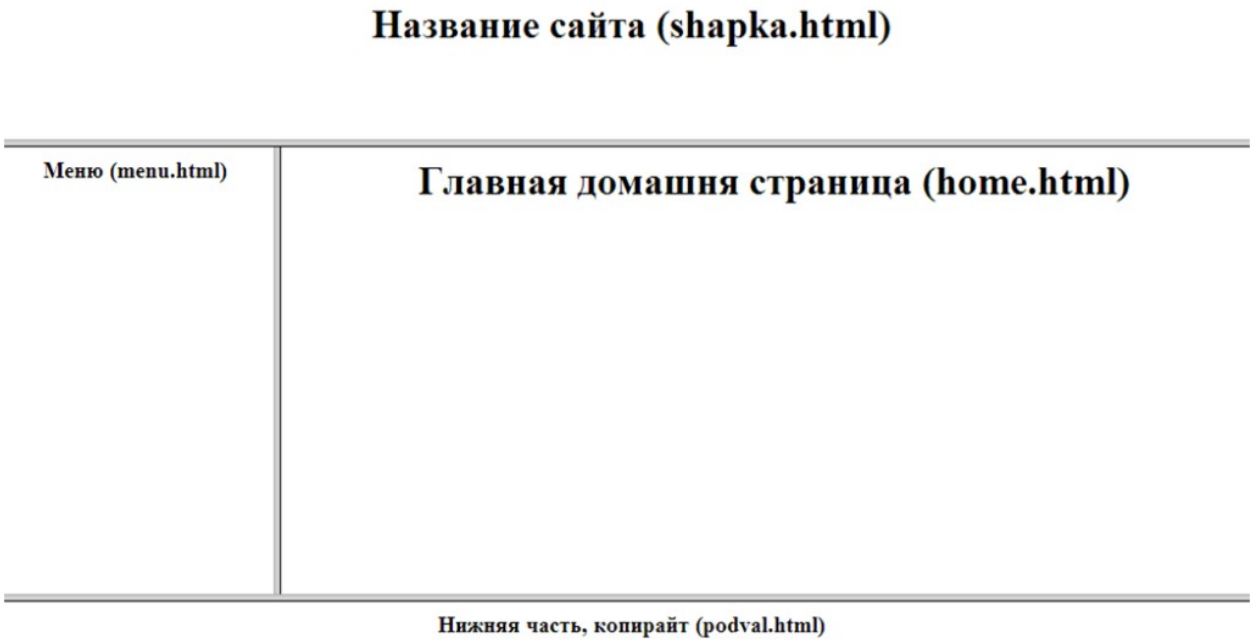
Вложенные фреймы

Работа с фреймами в html подразумевает и более сложную структуру. Рассмотрим на примере:



Каким будет результат?

Создать фреймовую структуру и загружаемые в нее файлы согласно изображению:

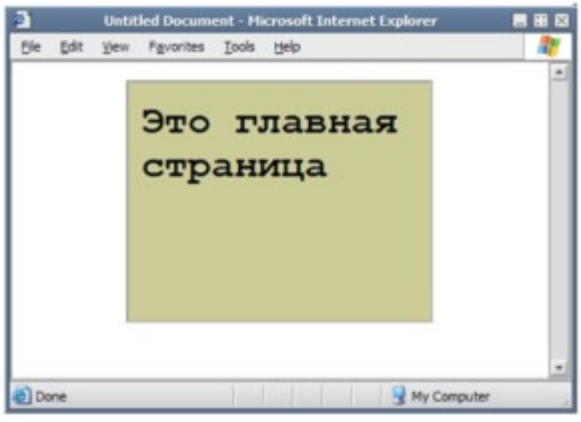


Встроенные (плавающие) фреймы

Такого вида фреймы нежелательно использовать в коммерческих проектах, они могут блокироваться браузером и неправильно восприниматься поисковыми системами Пример плавающего фрейма:

```
<html>
...
<body>
...
<iframe src="main.html" width="150" height="100"></iframe>
...
</body>
</html>
```

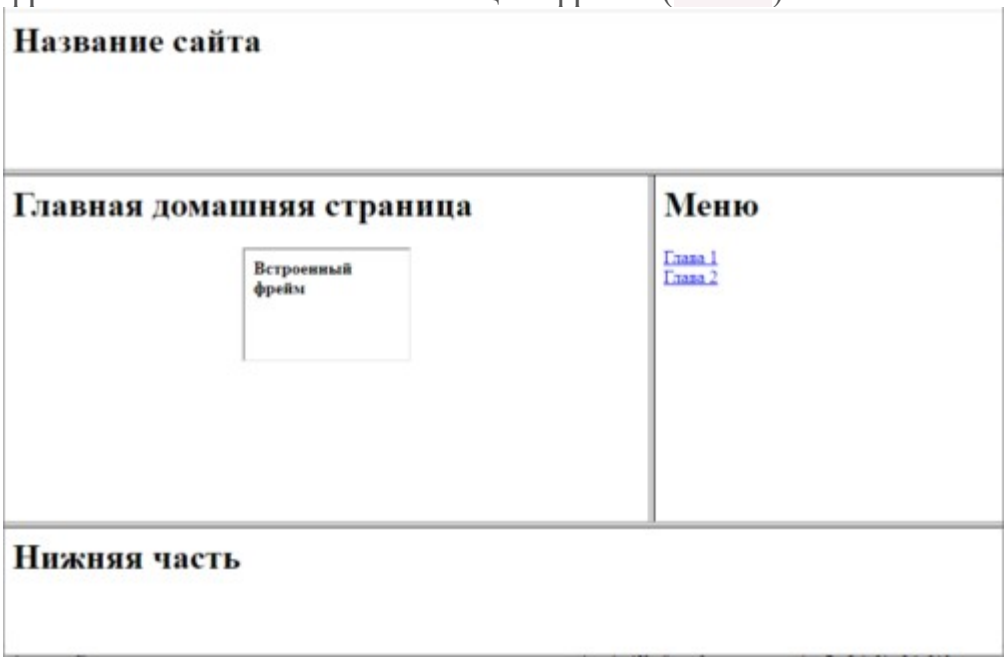
Фрейм будет вставлен в виде окна указанных размеров (атрибуты width и height).



Дополнительные атрибуты:

- name - имя фрейма
- frameborder - граница фрейма
- scrolling - полоса прокрутки
- hspace - отступы по горизонтали
- vspace - отступы по вертикали
- marginwidth - отступ внутренней страницы от границы фрейма по горизонтали
- marginheight - отступ внутренней страницы от границы фрейма по вертикали

Создать фреймовую структуру и загружаемые в нее файлы согласно изображению и списку файлов. В файле с меню необходимо организовать две гиперссылки: по щелчку на первую из них - файл *glava1.html* загружается в левый фрейм (там, где *Главная домашняя страница*), по щелчку на вторую из них - файл *glava2.html* загружается тоже в левый фрейм. В качестве внутреннего фрейма использовать плавающий фрейм (iframe).



Список файлов:

index.html - Главная раскладка с фреймовой структурой
shapka.html - Название сайта
menu.html - Меню
home.html - Главная домашняя страница
podval.html - Нижняя часть
inner.html - Встроенный фрейм
glava1.html - Глава 1
glava2.html - Глава 2

Вопросы для самоконтроля

1. HTML изображения.
2. Элемент img.
3. Параметры изображений.
4. Элемент map.
5. Элемент area.

Лабораторная работа №7. Создание форм.

Цель: изучить способы добавления форм на сайт.

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

Ход работы:

Создание и работа с формами в html

Все элементы управления, коими являются текстовые поля, кнопки, флажки, выпадающие списки и т.п., как правило, **размещаются внутри формы**:

```
<form action="file.php" method="post" id="form">
...
содержимое формы
...
</form>
```

- Как раз внутри элемента form должны располагаться элементы управления, которых может быть сколь угодно много.

Атрибуты формы:

action (англ. «действие») Файл на сервере с кодом для отработки отосланных данных	action="http://www.название.домен/имя программы"
enctype (англ. «тип кодировки»)	**text/plain** (обычный текст) **application/x-www-dorm-urlencoded** (для метода Post отправки формы) **multipart/form-data** (для метода Post, если прикрепляются файлы)
method (метод отправки данных)	post get

- В атрибуте action указывается серверный файл со скриптом, ответственным за основную обработку данных, пересылаемых из формы. Обычно код этого файла пишется на серверном языке программирования, например, на языке *php* или *perl*.
- Атрибут enctype указывает на тип передаваемой на сервер информации, если это просто текстовые данные - text/plain, если с формой отсылаются файлы, то следует указать multipart/form-data.
- Атрибут method указывает и определяет форму передачи данных. Подробно мы на этом останавливаться не будем, однако следует сказать, что для более надежной передачи следует указать метод post.

Элементы формы html

Текстовое поле html:

```
<input type="text" name="login" size="20" value="Логин" maxlength="25">
```

```
<input type="text" name="login" size="20"
maxlength="25" value="Логин">
```

Результат:

Атрибуты:

- Значение атрибута type - text - указывает на то, что это именно текстовое поле
- size - размер текстового поля в символах
- maxlength - максимальное кол-во вмещающихся в поле символов
- value - первоначальный текст в текстовом поле
- name - имя элемента, необходимо для обработки данных в файле-обработчике

Поле ввода пароля html:

```
<input type="password" name="pass" size="20" value="Пароль" maxlength="25">
```

```
<input type="password" name="pass" size="20"
maxlength="25" value="Пароль">
```

Результат:

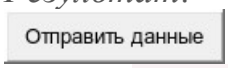
Вместо текста в поле отображается маска - звездочки или кружочки

Кнопка submit html:

```
<input type="submit" value="Отправить данные">
```

```
<input type="submit" value=" Послать форму ">
```

Результат:



Кнопка submit собирает все данные с формы, введенные пользователем и отправляет их по адресу, указанному в атрибуте action формы.

Кнопка очистки формы html:

```
<input type="reset" value="Очистить форму">
```

```
<input type="reset" value=" Очистить форму ">
```

Результат:

Кнопка возвращает состояние всех элементов управления к первоначальному (очищает форму)

Html флажок:

```
<input type="checkbox" name="asp" value="yes">ASP<br>
<input type="checkbox" name="js" value="yes" checked="checked">javascript<br>
<input type="checkbox" name="php" value="yes">PHP<br>
<input type="checkbox" name="html" value="yes" checked="checked">HTML<br>
```

```
<input type="checkbox" name="asp" value="yes">ASP<br>
<input type="checkbox" name="js" value="yes"
checked>JavaScript<br>
<input type="checkbox" name="php" value="yes">PHP<br>
<input type="checkbox" name="html" value="yes"
checked>HTML<br>
```

Результат:

- [ ] ASP
- [x] javascript
- [ ] PHP
- [x] HTML

В html флажок служит для организации множественного выбора, т.е. когда необходимо и возможно выбрать несколько вариантов ответа.

Атрибуты:

- Атрибут checked устанавливает сразу элемент отмеченным.
- Атрибуты name и value необходимы для обработки элемента программистом. Поэтому их можно опустить.

Radio кнопка html:

```
<input type="radio" name="book" value="asp">ASP<br>
<input type="radio" name="book" value="js">Javascript<br>
<input type="radio" name="book" value="php">PHP<br>
<input type="radio" name="book" value="html" checked="checked">HTML<br>
```

```
<input type="radio" name="book" value="asp">ASP<br>
<input type="radio" name="book" value="js">JavaScript<br>
<input type="radio" name="book" value="php">PHP<br>
<input type="radio" name="book" value="html"
checked>HTML<br>
```

Результат:

- ( ) ASP
- ( ) Javascript
- ( ) PHP
- (•) HTML

radio кнопка html служит для единственного выбора из нескольких вариантов.

Атрибуты:

- Атрибут checked устанавливает сразу элемент отмеченным.
- Атрибут value необходим для обработки элемента программистом. Поэтому его можно опустить.

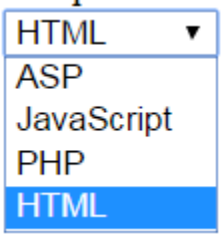
Важно: Для элементов **radio** необходимо, чтобы значение атрибута **name** у всех элементов в группе было одинаковым: в таком случае элементы будут работать взаимосвязано, при включении одного элемента, другие буду отключаться

ВЫПАДАЮЩИЙ СПИСОК HTML

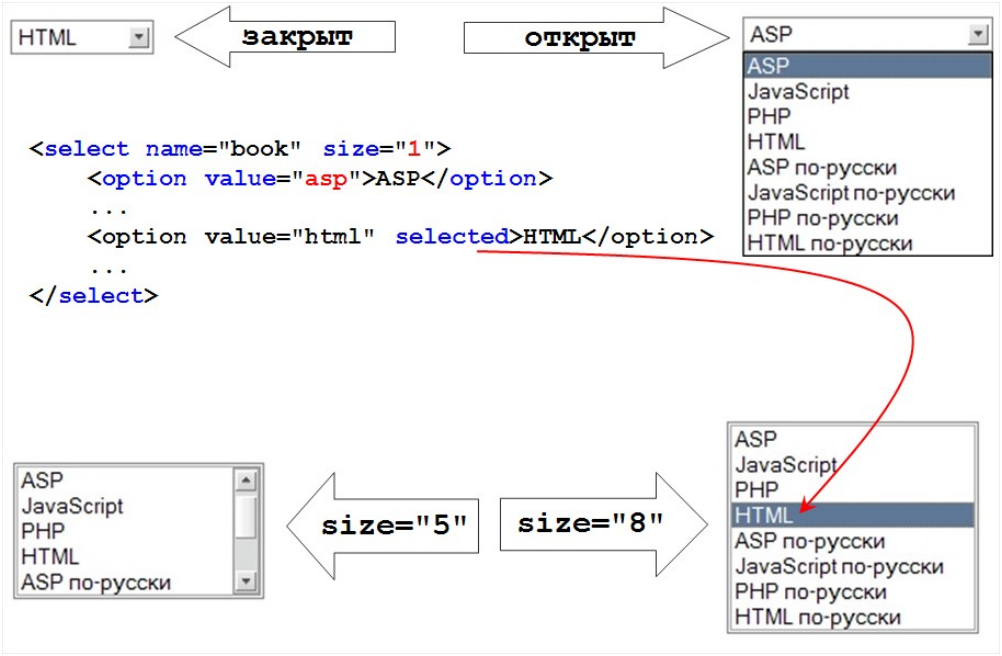
Рассмотрим пример добавления выпадающего списка:

```
<select name="book" size="1">
        <option value="asp">ASP</option>
        <option value="js">JavaScript</option>
        <option value="php">PHP</option>
        <option value="html" selected="selected">HTML</option>
</select>
```

Результат:

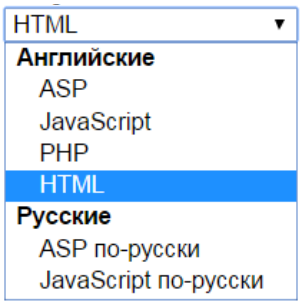


- Выпадающий список состоит из главного тега - select - который имеет закрывающую пару, а каждый пункт списка - это тег option, внутри которого отображается текст пункта
  Атрибуты:
  - Атрибут size со значением *«1»* указывает на то, что список в свернутом виде отображает один пункт, остальные открываются при щелчке на стрелочке меню
  - Атрибут selected у пункта (option) указывает на то, что именно этот пункт будет изначально виден, а остальные пункты «свернуты»
  - Атрибут value необходим для обработки элемента программистом. Поэтому его можно опустить.



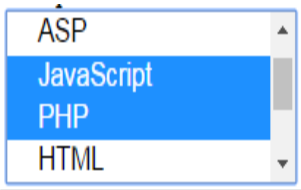
Для больших и сложных списков есть возможность **добавить подзаголовки** - тег optgroup с атрибутом label (надпись):

```
<select name="book" size="1">
        <optgroup label="Английские">
                <option value="asp">ASP</option>
                <option value="js">JavaScript</option>
                <option value="php">PHP</option>
                <option value="html" selected="selected">HTML</option>
        </optgroup>
        <optgroup label="Русские">
                <option value="asp_rus">ASP по-русски</option>
                <option value="js_rus">JavaScript по-русски</option>
        </optgroup>
</select>
```



Для предоставления возможности **выбора нескольких пунктов одновременно** необходимо добавить атрибут multiple. Но в таком случае и атрибут size следует установить в значение, большее, чем 1:

```
<select name="book" size="4" multiple="multiple">
...
```



ТЕКСТОВАЯ ОБЛАСТЬ В HTML

Для ввода большого фрагмента текста служит элемент текстовая область:

```
<textarea name="description" cols="30" rows="10">Текст</textarea>
```

```
<textarea name="description" cols="30" rows="10"
wrap="off|virtual|physical">Какой-то текст</textarea>
```

Результат:

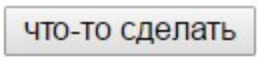
Атрибуты:

- Ширина элемента зависит от атрибута cols, который указывает сколько символов помещается по горизонтали.
- Атрибут rows определяет количество строк в элементе.
- Атрибут name необходим для обработки элемента программистом. Поэтому его можно опустить.

Другие элементы

Обычная кнопка:

```
<input type="button" value="что-то сделать">
```



Элемент кнопка-изображение:

```
<input type="image" width="32" height="32" src="1.png">
```

Результат:

Элемент загрузка файла:

Для прикрепления файла к форме существует специальный элемент управления:

```
<input type="file" name="userfile" size="20">
```

Результат:

При его использовании значение кодировки формы (атрибут enctype у тега form) должен иметь значение multipart/form-data

```
<input type="file" name="userfile" size="20">
<form enctype="multipart/form-data">
```

Скрытое поле:

Важным элементом при программировании является скрытое поле. Поле не отображается в окне браузера, а призвано передавать дополнительные данные через форму в файл-обработчик.

```
<input type="hidden" name="uid" value="15362">
```

Дополнительные элементы и атрибуты

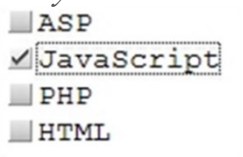
Label for:

- Для элементов управления **radio** и **checkbox** удобно использовать дополнительные элементы, которые, во-первых, делают привязку текста к самому элементу radio или checkbox, во-вторых, добавляют обводку при клике:

```
<input type="checkbox" id="book1">
<label for="book1">ASP</label>
```

В примере создана надпись (тег label) для элемента checkbox. Привязка осуществляется через атрибут id, значение которого указано в атрибуте for надписи.

Результат:



Атрибут disabled:

- Атрибут disabled позволяет блокировать элемент, делая его недоступным для изменения пользователем:

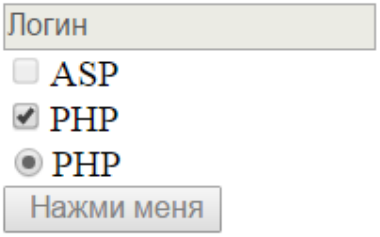
```
<input type="text" name="login" size="20" value="Логин" maxlength="25"
disabled="disabled"><br>
<input type="checkbox" name="asp" value="yes">ASP<br>
<input type="checkbox" name="js" value="yes" checked="checked"
disabled="disabled">javascript<br>
```

```
<input type="text" name="login" value="Логин"
disabled><br>
<input type="checkbox" name="asp" value="yes"
checked disabled>ASP<br>

...

<input type="radio" name="book" value="php"
checked disabled>PHP<br>
<input type="button" name="push" value="
Нажми меня " disabled><br>
```

Результат:



Атрибут readonly:

- Атрибут readonly делает текстовые документы доступными только для чтения (вносить и изменять текст нельзя):

```
<input type="text" name="login" size="20" value="Логин"
maxlength="25" readonly="readonly">
<input type="text" name="login"
value="Логин" readonly><br>
```

Результат:

Элемент fieldset:

- Для визуального оформления группы объектов можно использовать элемент fieldset:

```
<fieldset>
<legend>Книги</legend>
<input type="checkbox" value="html">HTML<br>
<input type="checkbox" value="asp">ASP<br>
<input type="checkbox" value="js">javaScript<br>
</fieldset>
```

```
<fieldset>
<legend>Книги</legend>
<input type="checkbox" id="asp">ASP<br>
<input type="checkbox" id="js">JavaScript<br>
<input type="checkbox" id="php">PHP<br>
<input type="checkbox" id="html">HTML<br>
</fieldset>
```

Результат:

Книги

- HTML
- ASP
- javaScript

Атрибут tabindex:

- Можно задать очередность передвижения по элементам клавишей TAB:

  <элемент tabindex="1">

  Элемент будет первым в очереди переходов.

Задание: Необходимо создать анкету web-разработчика согласно изображению. Элементы располагать в табличном виде

Анкета Web-разработчика

Регистрационное имя	
Пароль	: подтвердите пароль
Ваша специализация	Web-мастер
Пол	М Ж
Ваши навыки	знание HTML и CSS знание Perl знание ASP знание Adobe Photoshop знание JAVA знание JavaScript знание Flash
Дополнительные сведения о себе	

зарегистрировать | очистить форму

Вопросы для самоконтроля

1. HTML таблицы.
2. Группировка строк и столбцов таблиц.
3. Атрибуты элементов таблиц.

Лабораторная работа №8. 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:

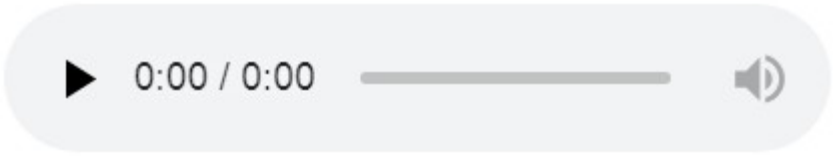
- mp3
- wav
- ogg

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 №9. 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.

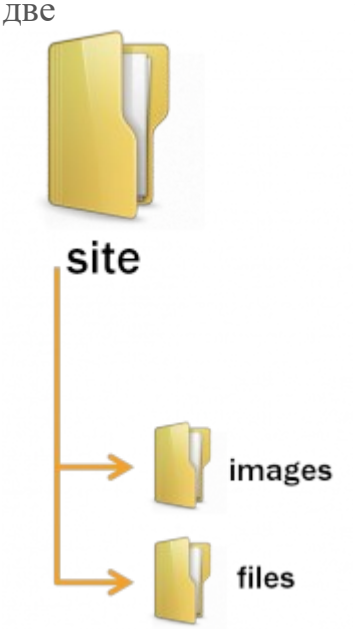
Ход работы:

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 №10. 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

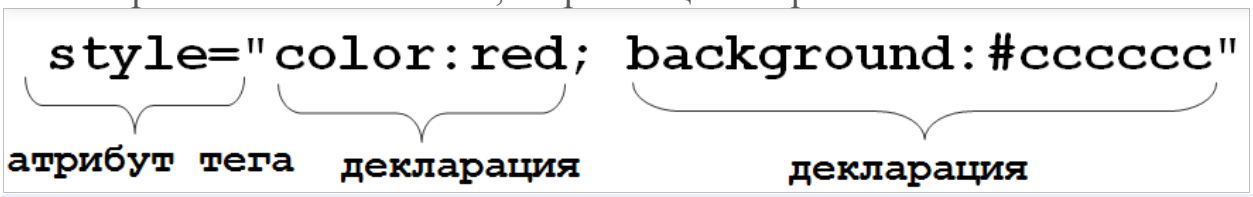
Метод встраивания (inline) в CSS

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:

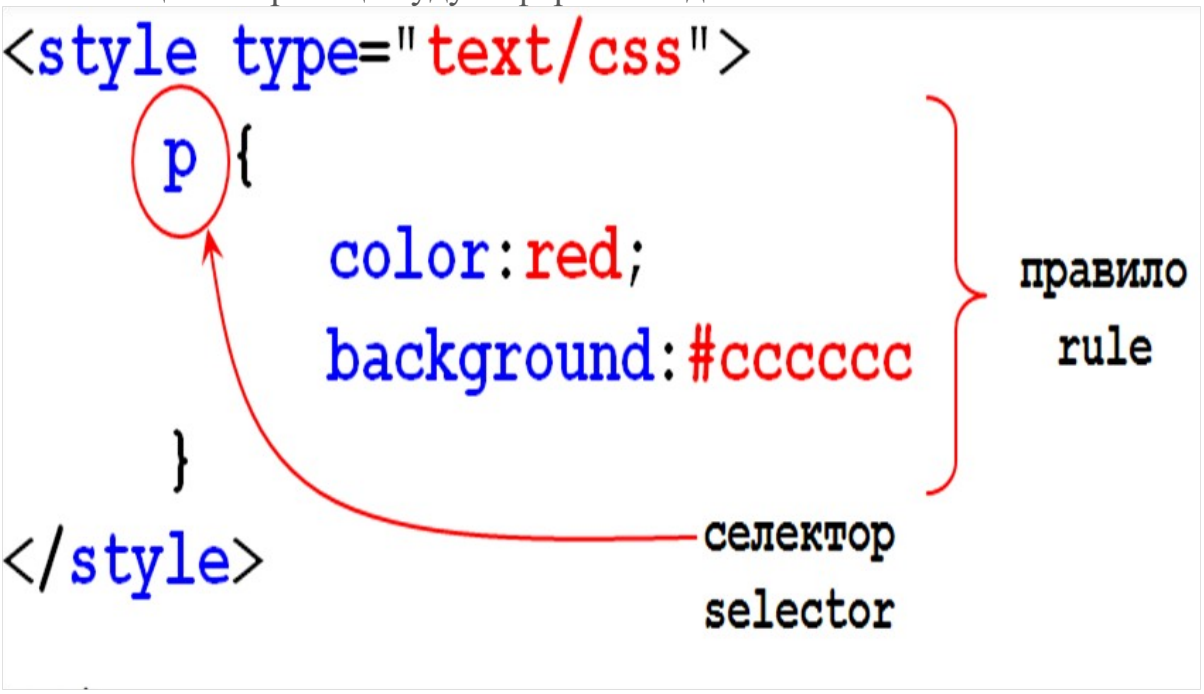
Обычный текст

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)

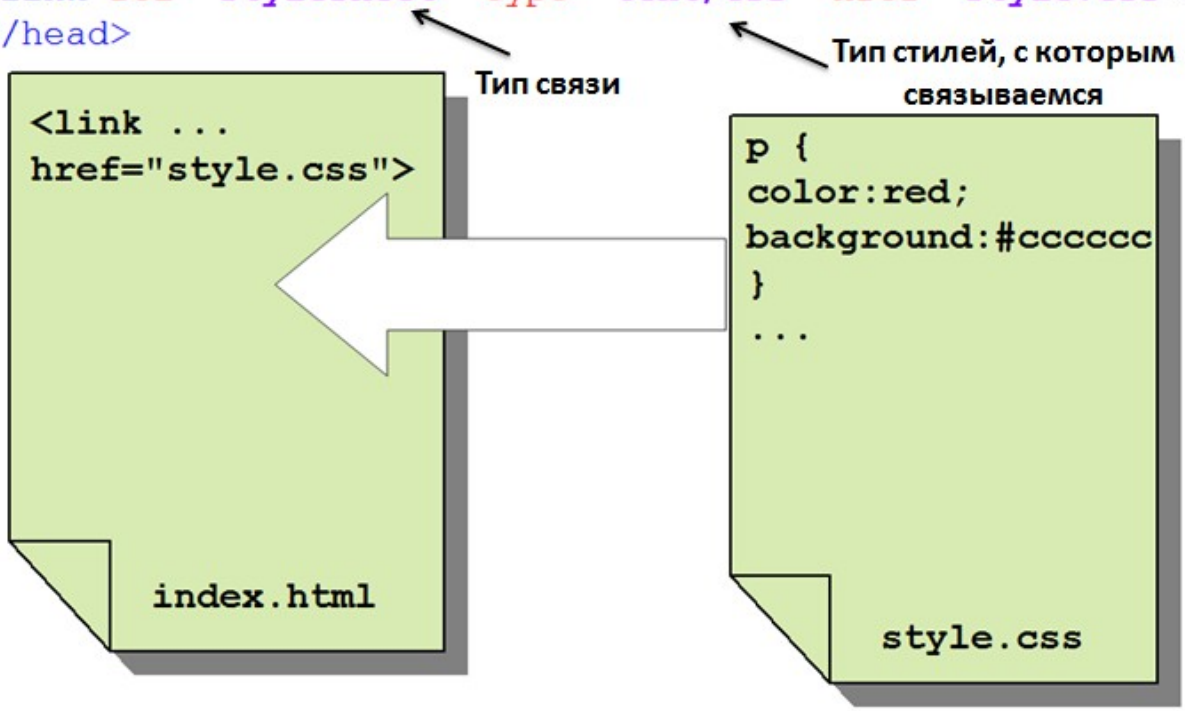
```
<head>
<style type="text/css">
h1 {
        border-width: 1;
        border: groove;
        text-align: center;
        color: green
        }
h2 {
        color: maroon;
        font-style: italic
        }
body {
        background-color: #FF0000;
        }
...
</style>
```

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 №11. 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:

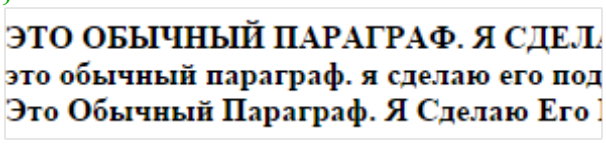
```
{
text-align:center|right|left|justify;
}
```

Оформление текста:

{
text-decoration:overline|line-through|underline|blink;
}

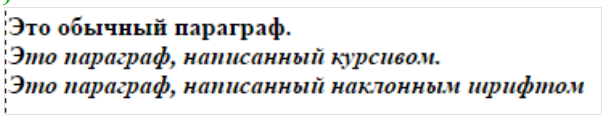
Преобразование текста:

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



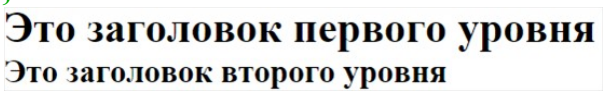
Стиль шрифта:

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



Размер шрифта:

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

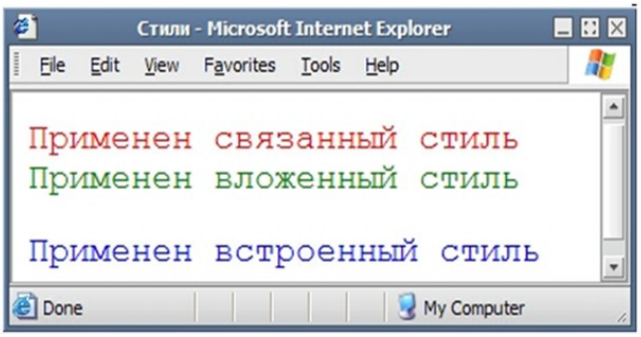


Каскадирование css стилей

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;
}
```

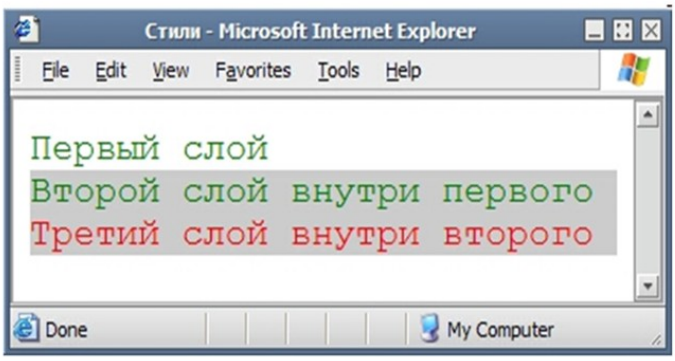
Тег с классом .children будет иметь цвет #666, а тег с классом .parent - #999.

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 №12. 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:

```
<body>
<div>
  <h1>Заголовок</h1>
  <p><strong>Первый</strong> параграф в div</p>
  <p><strong>Второй</strong> параграф в div</p>
  <strong>Просто полужирный текст в div</strong>
</div>
<p><strong>Первый</strong> параграф в body</p>
</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 <body>
2   <p><b>Жирное начертание текста</b></p>
3   <p><i><b>Одновременно жирное начертание текста
4     и выделенное цветом</b></i></p>
5 </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:
```
<div><i>Кто на горе</i>, тот раньше солнце встретит.
</div>
<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:
```
<html>
<head>
  <style type="text/css">
  ...
  </style>
</head>
<body>
<h1 id="header1"><i>Стихотворение</i> <u>2013 год</u></h1>
<h2>I</h2>
<p>В земные страсти вовлеченный,
я знаю, что из тьмы на свет
однажды выйдет ангел черный
и крикнет, что спасенья нет. </p>
<h2>II</h2>
<p>Но простодушный и несмелый,
прекрасный, как благая весть,
идущий следом ангел белый
прошепчет, что надежда есть.</p>
<p class="end">Конец</p>
<h1>Продолжение следует</h1>
</body>
</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 №13. Селекторы атрибутов и универсальный селектор.

Цель: изучить способы соединения селекторов.

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:

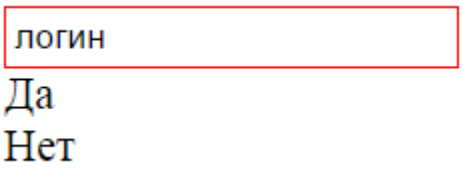
```
<body>
<input type="text" value="логин"><br>
<input type="radio" value="yes">Да<br>
<input type="radio" value="no">Нет<br>
</body>
</html>
```

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;
}
```

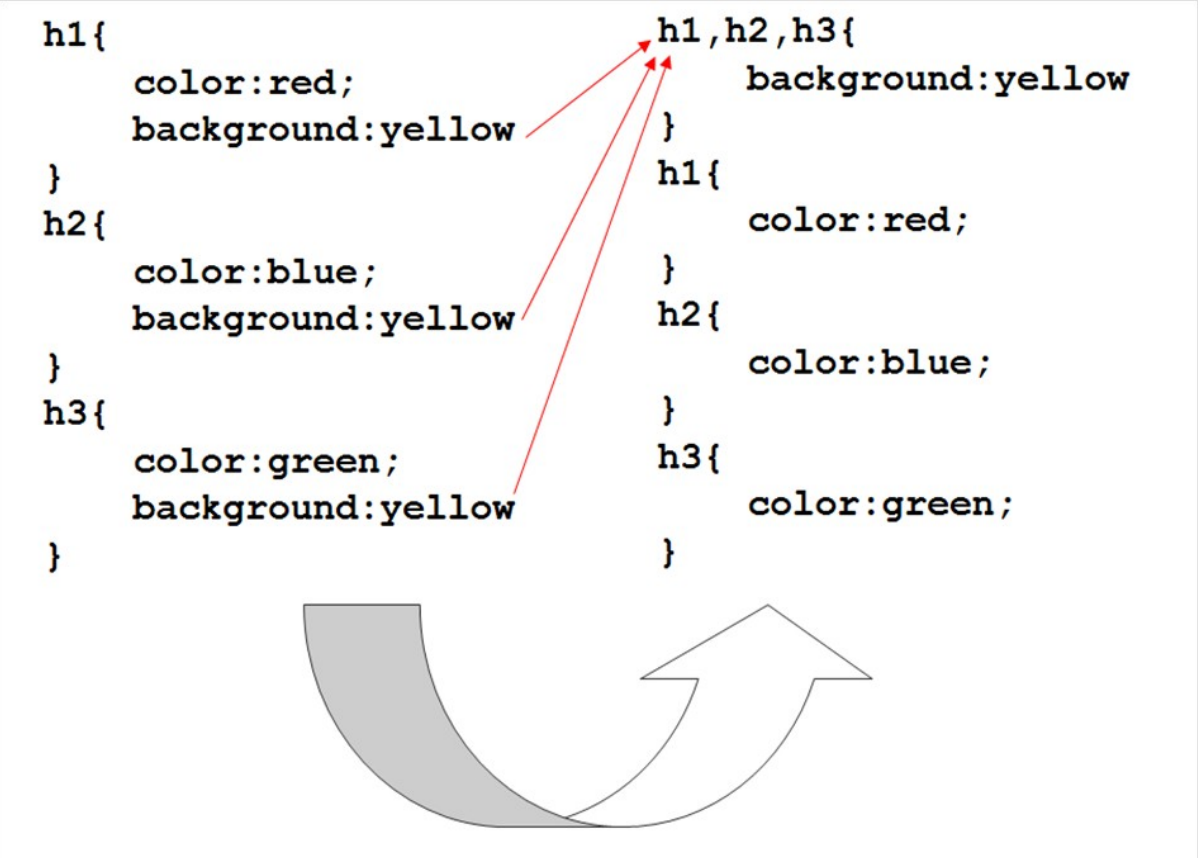
Пример: для всех элементов страницы определить единое семейство шрифта и цвет шрифта

Выполнение:

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

Группировка CSS селекторов

Для сокращения записи правил в CSS принята группировка:



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>
...
```

Вопросы для самоконтроля

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

Лабораторная работа №14. Псевдоклассы и псевдоэлементы CSS.

Цель: изучить способы добавления псевдоклассов и псевдоэлементов на сайт.

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

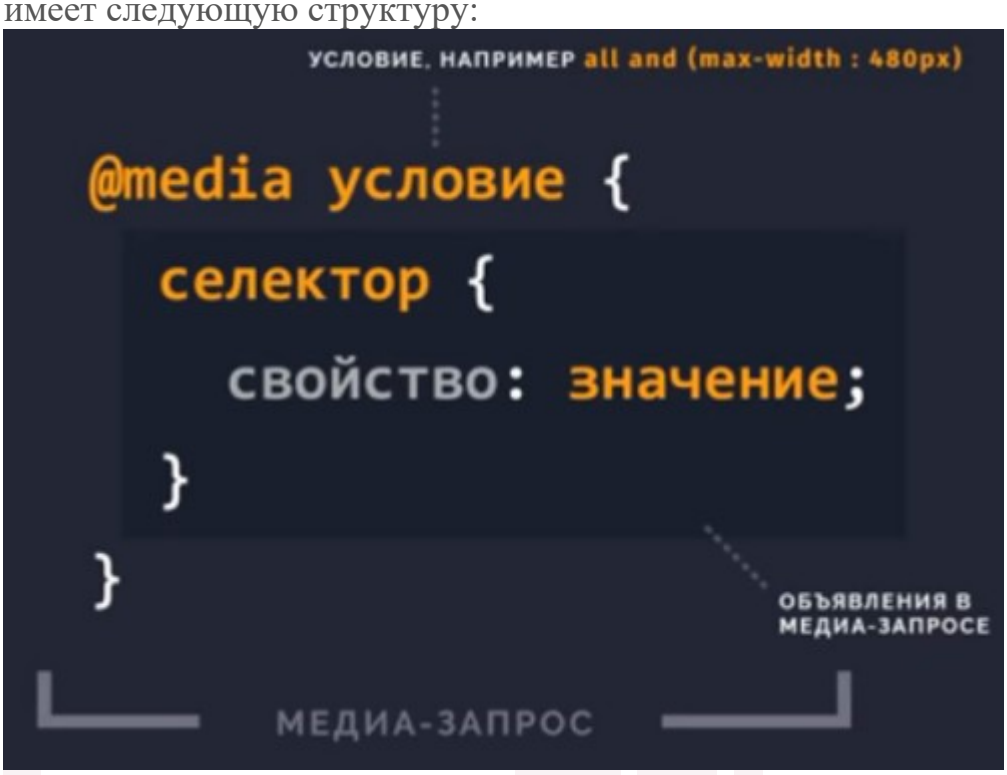
Ход работы:

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**:

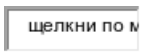
```
input:focus{...} /* в тот момент, когда элемент получает фокус */
input:active{...}/* в момент активации элемента */
```

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>
```

Чебурашка
Крокодил Гена
Шапокляк
Крыса Лариса

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 №15. Основные стили CSS: оформление текста.

Цель: изучить способы оформления текста.

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 среднеазиатско
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>
Это текст
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

Вопросы для самоконтроля

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

Лабораторная работа №16. Основные стили CSS: свойства шрифта.

Цель: изучить способы оформления шрифта.

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

Ход работы:

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 №17. Свойства фона.

Цель: изучить способы добавления фона.

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

Ход работы:

Свойства фона CSS

Перечислим основные свойства для стилизации фона и их значения

BACKGROUND-COLOR (ЦВЕТ ФОНА)

Возможные значения:

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

Пример:

```
h5{background-color:#ff0000}
p{background-color:rgb(0,100,0)}
```

Результат:

Это пример параграфа

Это пример заголовка

BACKGROUND-IMAGE (ФОНОВОЕ ИЗОБРАЖЕНИЕ)

Возможные значения:

- url('...') (url-адрес)
- none (без фонового изображения)

Изображение тиражируется по всему экрану.

Пример:

```
p{background-image:url('https://labs-org.ru/wp-content/uploads/2016/05/logo.png')}
```

Результат:

Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа.

BACKGROUND-REPEAT (ПОВТОРЕНИЕ ФОНА)

Возможные значения:

- repeat-x (Фоновый рисунок повторяется только по горизонтали)
- repeat-y (Фоновый рисунок повторяется только по вертикали)
- no-repeat (изображение без повторений)

Пример:

```
p{background-image:url('https://labs-org.ru/wp-content/uploads/2016/05/logo.png');
background-repeat:repeat-y}
```

Результат:

Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример

параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа. Это пример параграфа.

BACKGROUND-POSITION (ПОЗИЦИОНИРОВАНИЕ ФОНА)

Возможные значения:

- xpos ypos (координаты)
- left top (по левому и верхнему краю)
- left center (слева и по центру)
- left bottom (слева и по нижнему краю)
- right top (справа и по верхнему краю)
- right center (справа и по центру)
- right bottom (справа и по нижнему краю)
- center top (центр по горизонтали и верхний край)
- center center (центр по горизонтали и по вертикали)
- center bottom (центр по горизонтали и нижний край)

Пример:

```
p{
  background-image:url('https://labs-org.ru/wp-content/uploads/2016/05/logo.png');
  background-repeat:no-repeat;
  background-position:right top;
}
```

Результат:

Это пример параграфа. Это пример параграфа.

BACKGROUND-ATTACHMENT (ПРОКРУТКА ФОНА)

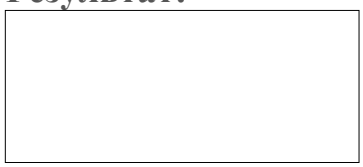
Возможные значения:

- scroll (прокрутка вместе с содержимым)
- fixed (делает изображение неподвижным)

Пример:

```
textarea{
background-image:url('https://labs-org.ru/wp-content/uploads/2016/05/logo.png');
background-attachment:fixed
}
```

Результат:



Краткая запись background

Свойства фона задаются в следующем порядке:

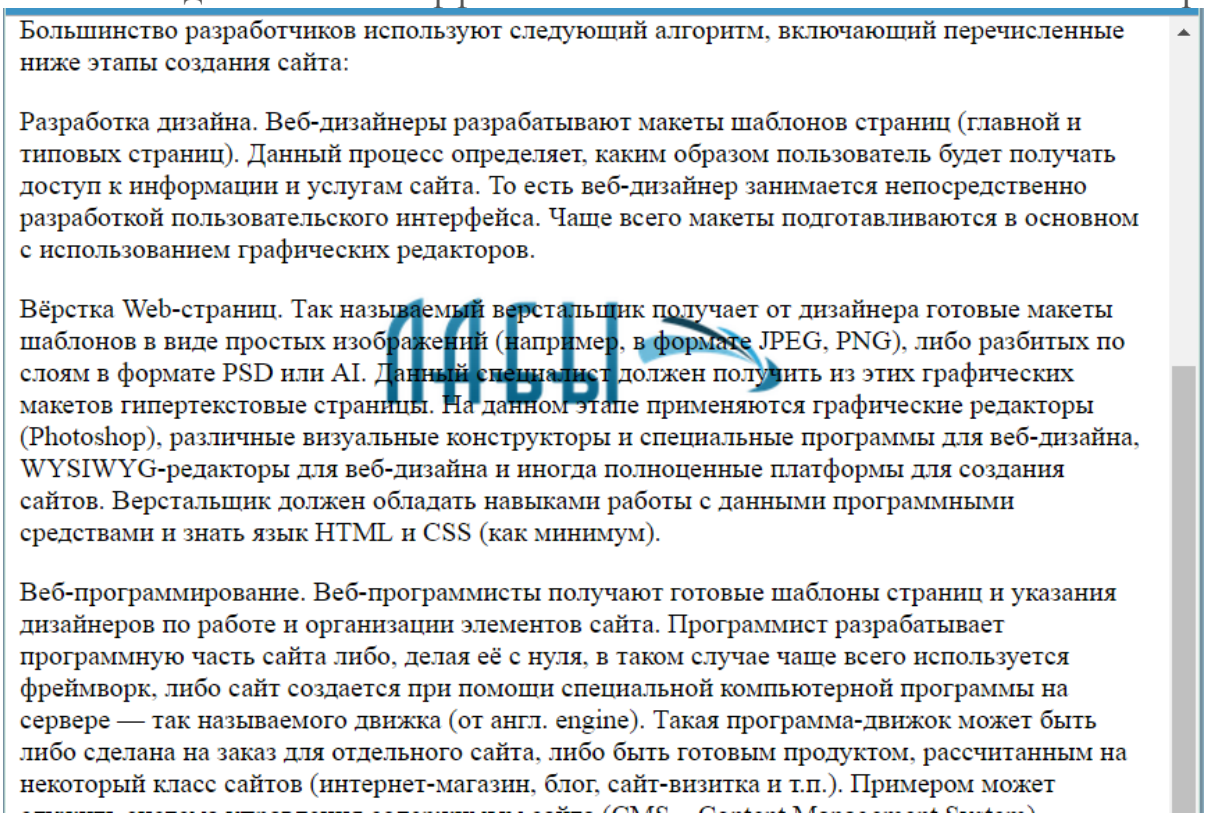
элемент {background-color background-image background-repeat background-

attachment background-position}

Пример:

body{**background**:#fff url('1,gif') no-repeat right top;}

Задание: Скачайте файл. Расположите файл-картинку на задний фон страницы, чтобы добиться эффекта как на итоговом изображении:


Большинство разработчиков используют следующий алгоритм, включающий перечисленные ниже этапы создания сайта:

Разработка дизайна. Веб-дизайнеры разрабатывают макеты шаблонов страниц (главной и типовых страниц). Данный процесс определяет, каким образом пользователь будет получать доступ к информации и услугам сайта. То есть веб-дизайнер занимается непосредственно разработкой пользовательского интерфейса. Чаще всего макеты подготавливаются в основном с использованием графических редакторов.

Вёрстка Web-страниц. Так называемый верстальщик получает от дизайнера готовые макеты шаблонов в виде простых изображений (например, в формате JPEG, PNG), либо разбитых по слоям в формате PSD или AI. Данный специалист должен получить из этих графических макетов гипертекстовые страницы. На данном этапе применяются графические редакторы (Photoshop), различные визуальные конструкторы и специальные программы для веб-дизайна, WYSIWYG-редакторы для веб-дизайна и иногда полноценные платформы для создания сайтов. Верстальщик должен обладать навыками работы с данными программными средствами и знать язык HTML и CSS (как минимум).

Веб-программирование. Веб-программисты получают готовые шаблоны страниц и указания дизайнеров по работе и организации элементов сайта. Программист разрабатывает программную часть сайта либо, делая её с нуля, в таком случае чаще всего используется фреймворк, либо сайт создается при помощи специальной компьютерной программы на сервере — так называемого движка (от англ. engine). Такая программа-движок может быть либо сделана на заказ для отдельного сайта, либо быть готовым продуктом, рассчитанным на некоторый класс сайтов (интернет-магазин, блог, сайт-визитка и т.п.). Примером может

Вопросы для самоконтроля

1. CSS переходы.
2. CSS трансформация.
3. CSS анимация.

Лабораторная работа №18. Видимость элемента, оформление ссылок и списков, курсор.

Цель: изучить способы оформления ссылок, списков, курсора.

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

Ход работы:

Видимость элемента в CSS

DISPLAY

Возможные значения:

- block (отображается как блочный)
- none (временно удаляет элемент из документа)
- inline (отображается как встроенный (строчный))
- inline-block (внутренняя часть форматируется как блочный элемент, а сам элемент - как встроенный, обтекается другими элементами (подобно img))
- list-item (отображается как блочный и добавляется маркер списка)
- inline-table (элемент в качестве таблицы, но при этом таблица является встроенным элементом и происходит ее обтекание другими элементами)
- table (элемент в качестве блочной таблицы)

Пример:

```
div:hover{
        display:none;
}
```

Результат:

Наведите курсор

А я без стиля

VISIBILITY

Возможные значения:

- hidden (становится невидимым, но участвует в нормальном потоке элементов)
- visible (отображается как видимый)
- collapse (для содержимого ячеек таблиц, отображаются как будто применен стиль display: none (заданные строки и колонки убираются))

Пример:

```
div:hover{
        visibility:hidden;
}
```

Результат:

Наведите курсор

А я без стиля

OVERFLOW

Возможные значения:

- visible (отображается все содержание элемента, даже за пределами установленной высоты и ширины)

- hidden (отображается только область внутри элемента, остальное будет обрезано)
- scroll (всегда добавляются полосы прокрутки)
- auto (полосы прокрутки добавляются только при необходимости)

Пример:

```
div{
        width:150px;
        height:150px;
        overflow:scroll;
}
```

Результат:

Это пример параграфа1. Это пример параграфа2. Это пример параграфа3. Это пример параграфа4. Это пример параграфа5. Это пример параграфа6. Это пример параграфа7. Это пример параграфа8. Это пример параграфа9. Это пример параграфа10. Это пример параграфа11. Это пример параграфа12. Это пример параграфа13. Это пример параграфа14. Это пример параграфа15. Это пример параграфа16. Это пример параграфа17. Это пример параграфа18.

Тот же самый пример с другим свойством:

```
div{
        width:150px;
        height:150px;
        overflow:hidden;
}
```

Результат:

Это пример параграфа1. Это пример параграфа2. Это пример параграфа3. Это пример параграфа4. Это пример параграфа5. Это пример параграфа6. Это пример параграфа7. Это пример параграфа8. Это пример параграфа9. Это пример параграфа10. Это пример параграфа11. Это пример параграфа12. Это пример параграфа13. Это пример параграфа14. Это пример параграфа15. Это пример параграфа16. Это пример параграфа17. Это пример параграфа18.

Изменение курсора в CSS

CURSOR

Возможные значения:

- url('путь к файлу') (свой собственный курсор из файла изображения)
- crosshair
- help
- move
- pointer
- progress
- text
- wait
- n-resize
- ne-resize
- e-resize
- se-resize

- s-resize
- sw-resize
- w-resize
- nw-resize

Пример:

```
.crosshair{
        cursor: crosshair;
}
.help{
        cursor: help;
}
...
```

Результат:

crosshair
help
move
pointer
progress
text
wait
n-resize
ne-resize
e-resize
se-resize
s-resize
sw-resize
w-resize
nw-resize

Оформление ссылок в CSS

Для оформления ссылок в css, прежде всего, следует использовать псевдоклассы, рассмотренные ранее.

Для стилизации ссылок также используются следующие свойства CSS:

TEXT-DECORATION

со значением none: смотри пример здесь.

BACKGROUND-COLOR

например, при наведении курсора менять цвет фона ссылки; смотри пример здесь.

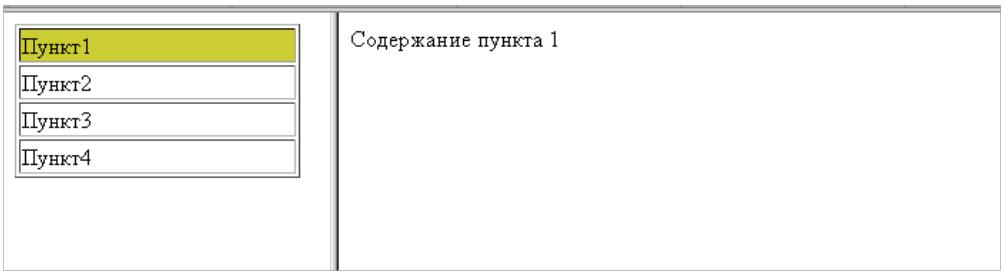
Задание: Создать страницу с меню. Пункты меню поместить в таблицу. По наведению курсора мыши на пункт, подсвечивать ячейку и менять вид курсора мыши. Оформить пункты как гиперссылки (для гиперссылок убрать оформление).

Использовать свойства:

псевдокласс hover

свойство text-decoration

Желаемый **результат:**



Оформление списков в CSS

Для оформления списков используются следующие свойства:

LIST-STYLE-TYPE

Возможные значения:

- none (отмена маркера или нумерации)
- disc
- circle
- square
- decimal
- decimal-leading-zero
- armenian
- georgian
- lower-alpha
- upper-alpha
- lower-greek
- lower-latin
- upper-latin
- lower-roman
- upper-roman

Пример:

```
ul.none{
        list-style-type: none;
}
ul.disc{
        list-style-type: disc;
}
...
ol.decimal{
        list-style-type: decimal;
}
ol.decimal-leading-zero{
        list-style-type: decimal-leading-zero;
}
```

Результат:

- none
- none
- disc

- disc
- circle
- circle
- square
- square

1. decimal
2. decimal

1. decimal-leading-zero
2. decimal-leading-zero

1. armenian
2. armenian

1. georgian
2. georgian

1. lower-alpha
2. lower-alpha

1. upper-alpha
2. upper-alpha

1. lower-greek
2. lower-greek

1. lower-latin
2. lower-latin

1. upper-latin
2. upper-latin

1. lower-roman
2. lower-roman

1. upper-roman
2. upper-roman

LIST-STYLE-IMAGE

Возможные значения:

- url('путь к файлу') (установка своего изображения)

Пример:

```
ul{
  list-style-image:url('1.png');
}
```

Результат:

1. пункт 1
2. пункт 1

LIST-STYLE-POSITION (ПОЛОЖЕНИЕ МАРКЕРА ИЛИ НОМЕРА)

Возможные значения:

- inside (Маркер является частью текстового блока и отображается в элементе списка)
- outside (Текст выравнивается по левому краю, а маркеры размещаются за пределами текстового блока)

Пример:

```
ul.inside{
```

```
  list-style-position:inside;
}
ul.outside{
  list-style-position:outside;
}
```

Результат:

inside:

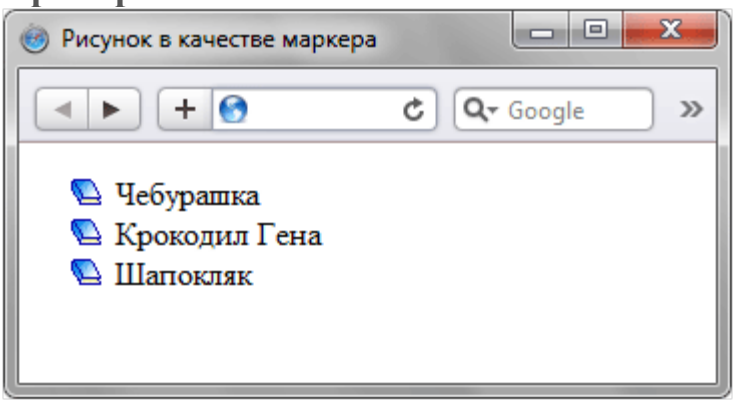
- пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1
- пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2

outside:

- пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1 пункт 1
- пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2 пункт 2

Задание: Создать маркированный список, вместо маркеров установить изображение.

Примерный результат:



Вопросы для самоконтроля

1. CSS переходы.
2. CSS трансформация.
3. CSS анимация.

Лабораторная работа №19. Отступ и граница элемента CSS.

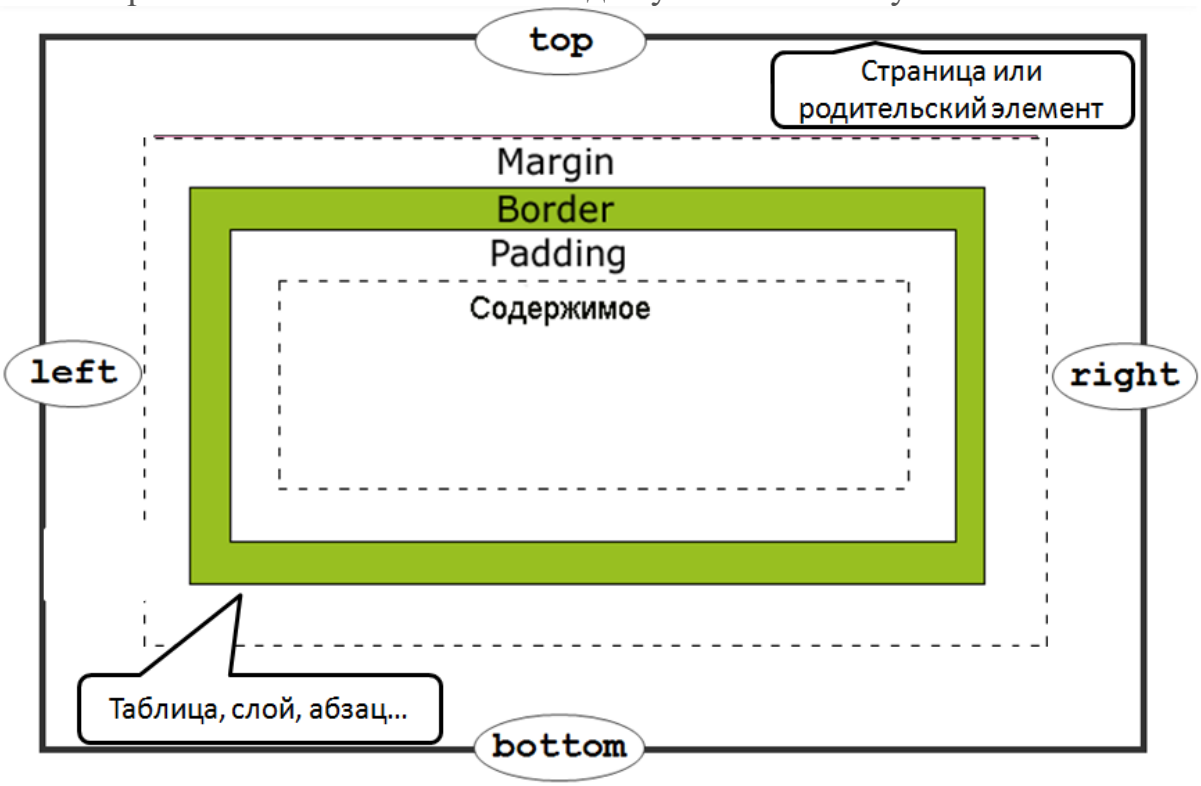
Цель: изучить способы оформления отступов и границ.

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

Ход работы:

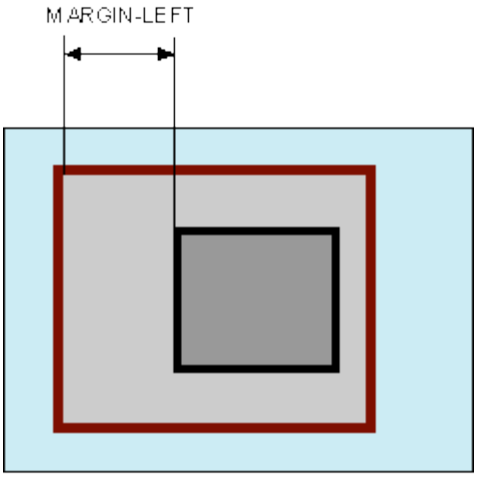
Отступы в CSS

Элемент в html - это прямоугольник, для которого можно указать величины внутренних и внешних отступов, а также границу, которая разделяет их. Отступы в CSS устанавливаются, например, для блочных элементов и таблиц. Рассмотрим основные свойства CSS для установки отступов:



ВНЕШНИЕ ОТСТУПЫ

Внешние отступы в CSS задаются при помощи свойства margin, которое устанавливает величину отступа от границ текущего элемента до внутренней границы его родительского элемента.



Свойство:

- margin-bottom (нижний отступ)
- margin-left (отступ слева)
- margin-right (отступ справа)
- margin-top (верхний отступ)

Значения:

- auto
- величина: px или em
- %

Краткая запись:

margin:margin-top margin-right margin-bottom margin-left;
margin:10px 20px 20px 30px;

Пример:

```
...
<style type="text/css">
p{
        border:5px solid red; /* красная рамка */
        margin:20px;
}
div{
        border:5px solid green; /* зеленая рамка */
}
</style></head>
<body>
<div>
Агния Барто - стихи
<p>
Я на уроке в первый раз.<br/>
Теперь я ученица.<br/>
Вошла учительница в класс,-<br/>
Вставать или садиться?<br/>
</p>
...
```

Результат:

Агния Барто - стихи

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

ВНУТРЕННИЕ ОТСТУПЫ

Внутренние отступы в css создаются при помощи свойства padding, которое устанавливает значение полей от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое.

Свойства:

- padding-bottom (нижний отступ)
- padding-left (отступ слева)
- padding-right (отступ справа)
- padding-top (верхний отступ)

Значения:

- auto
- величина: px или em
- %

Краткая запись:

padding:padding-top padding-right padding-bottom padding-left;
padding:10px 20px 20px 30px;

Пример:

```
...
<style type="text/css">
p{
        border:5px solid red; /* красная рамка */
        padding:20px; /* внутренний отступ */
}
</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>
...
```

Результат:

Я на уроке в первый раз.
Теперь я ученица.

Вошла учительница в класс,-
Вставать или садиться?

Граница элемента (рамка)

Граница элемента в CSS устанавливается при помощи свойства border.

BORDER-STYLE (СТИЛЬ ГРАНИЦЫ)

Значения:

- none (без границы)
- dotted (из точек)
- dashed (пунктирная)
- solid (сплошная)
- double (двойная)
- groove (трехмерная)
- ridge (трехмерная)
- inset (трехмерная с тенью)
- outset (трехмерная с тенью)

Пример:

none

dotted

dashed

solid

BORDER-WIDTH (ШИРИНА ГРАНИЦЫ)

Значения:

- thin (тонкая)
- medium (средняя)
- thick (толстая)
- значение

Пример:

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

p.one{
        border-style:solid;
        border-width:5px;
}
p.two{
        border-style:solid;
        border-width:medium;
}
</style></head>
<body>
<p class="one">
Я на уроке в первый раз.<br/>
Теперь я ученица.<br/>
```

```
Вошла учительница в класс,-<br/>
Вставать или садиться?<br/>
</p>
<p class="two">
Как надо парту открывать,<br/>
Не знала я сначала,<br/>
И я не знала, как вставать,<br/>
Чтоб парта не стучала.<br/>
</p>
```

Результат:

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

Как надо парту открывать,
Не знала я сначала,
И я не знала, как вставать,
Чтоб парта не стучала.

BORDER-COLOR (ЦВЕТ ГРАНИЦЫ)

Значения:

- red (цвет)
- rgb(255,0,0) (в системе rgb)
- #ff0000 (в шестнадцатиричной системе)
- transparent (прозрачная)

Для каждой стороны границы можно задать свой стиль:

```
p{
        border-top-style:solid;
        border-right-style:dotted;
        border-bottom-style:dashed;
        border-left-style:none;
        border-top-color:#0f0;
        border-right-color:#f00;
}
```

Результат:

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

Краткая запись:

border:border-width border-style border-color;
border: 1px solid #000;

Внешние границы (outline)

Внешние границы в CSS создаются при помощи свойства outline, одновременно устанавливающее цвет, стиль и толщину внешней границы на всех четырех сторонах элемента. В отличие от линии, задаваемой через border, свойство outline не влияет на положение блока и его ширину.

OUTLINE-COLOR (ЦВЕТ)

Значения:

- red (цвет)
- rgb(255,0,0) (в системе rgb)
- #ff0000 (в шестнадцатеричной системе)
- invert (инвертированный, противоположный)

OUTLINE-WIDTH (ШИРИНА)

Значения:

- thin (тонкая)
- medium (средняя)
- thick (толстая)
- значение

OUTLINE-STYLE (СТИЛЬ ГРАНИЦЫ)

Значения:

- none (без границы)
- dotted (из точек)
- dashed (пунктирная)
- solid (сплошная)
- double (двойная)
- groove (трехмерная)
- ridge (трехмерная)
- inset (трехмерная с тенью)
- outset (трехмерная с тенью)

Краткая запись:

outline:outline-color outline-style outline-width;

outline: #0f0 solid thick;

Некоторые приемы с границей

Рамка вокруг изображения

Пример:

```
img{
        padding:20px; /* Поля вокруг изображения */
        margin-right:10px; /* отступ справа */
        margin-bottom:10px; /* отступ снизу */
        outline:1px solid #666; /* параметры границы */
        background:#f0f0f0; /* цвет фона */
}
```

Результат:

Двойная рамка с использованием CSS

Пример:

```
p{
        border:5px solid red;
        outline:6px solid orange;
}
```

Результат:

Путь осилит идущий,

И поэтому я иду.
Через горы и пущи,
Через радость мою и беду.

Эффектные рамки для изображений

Пример:

```
...
.photo {
        width   : 150px;
        padding         : 10px 10px 20px 10px;
        border : 1px solid #BFBFBF;
        background-color        : white;
        -webkit-box-shadow      : 2px 2px 3px rgba(135, 139, 144, 0.4);
        -moz-box-shadow         : 2px 2px 3px rgba(135, 139, 144, 0.4);
        box-shadow              : 2px 2px 3px rgba(135, 139, 144, 0.4);
}
</style></head>
<body>
```

```
<div class="photo">
        <img  src="2.JPG" width="150" alt="">
        <p>Не ешь меня!!!</p>
</div>
...
```

Результат:

Вопросы для самоконтроля

1. Таблицы стилей.
2. Разделение контента от дизайна.
3. Библиотеки шаблонов.
4. Отказ от излишнего кода.
5. Проверка кода на ошибки.

Лабораторная работа №20. Свойства таблицы и табличная верстка.

Цель: изучить способы оформления таблиц, верстку таблиц.

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

Ход работы:

Свойства таблицы

Рассмотрим основные CSS свойства таблицы

BORDER

Свойство рассматривается в одном из предыдущих уроков и включает одновременно несколько свойств:

- BORDER-STYLE (СТИЛЬ ГРАНИЦЫ)
- BORDER-WIDTH (ШИРИНА ГРАНИЦЫ)
- BORDER-COLOR (ЦВЕТ ГРАНИЦЫ)

Существует также сборное правило:

border:border-width border-style border-color;

border: 1px solid #000;

BORDER-COLLAPSE (СЛИЯНИЕ ГРАНИЦЫ)

Значения:

- collapse (слитая граница)
- separate (вокруг каждой ячейки - своя собственная рамка)

Пример:

```
table.collapse{
    border-collapse:collapse;
}
table.separate{
    border-collapse:separate;
}
```

Результат:

Таблица с collapse	Таблица с collapse
Таблица с collapse	Таблица с collapse

Таблица с separate	Таблица с separate
Таблица с separate	Таблица с separate

WIDTH И HEIGHT (ВЫСОТА И ШИРИНА ТАБЛИЦЫ)

Значения:

- px
- %

Пример:

```
table{
        width:100%;
```

```
        height:100px;
}
```

Результат:

Таблица	Таблица
Таблица	Таблица

TEXT-ALIGN (ВЫРАВНИВАНИЕ ПО ГОРИЗОНТАЛИ)

Значения:

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

VERTICAL-ALIGN (ВЫРАВНИВАНИЕ ПО ВЕРТИКАЛИ)

Значения:

- baseline (по базовой линии)
- sub (как подиндекс)
- super (как надиндекс)
- top (по верхнему краю)
- middle (посередине)
- bottom(по нижнему краю)
- % (от высоты межстрочного интервала)

Пример:

```
table{
        text-align:right;
        height:100px;
        vertical-align:middle;
}
```

Результат:

Таблица	Таблица
Таблица	Таблица

PADDING (ВНУТРЕННИЕ ОТСТУПЫ В ТАБЛИЦЕ)

Данное свойство полностью соответствует правилам данного свойства для всех элементов. Поэтому рассмотреть его можно в одном из предыдущих уроков.

BACKGROUND-COLOR (ЗАДНИЙ ФОН)

COLOR (ЦВЕТ ТЕКСТА)

Данные свойства соответствуют правилом их определения для всех остальных элементов. Поэтому темы можно рассмотреть из предыдущих уроков: задний фон и цвет.

Задание:

Открыть/создть файл *style.css*:

1. Добавить свойства для следующих тегов (если еще не добавлены):
    - body основная страница
    - p абзац
    - a гиперссылка
    - h1, h2, h3, … заголовки
    - ul, ol, li списки, пункты списков
    - table, tr, td таблица, строка, ячейка строки
    - hr линия
    - span, div строчный тег, блочный тег
2. Добавить комментарий с пояснением к каждому свойству:

dama.html | style.css | menu.html | shapka.html | index.html | glava4.html

```
p { /* стиль для параграфа  */
text-indent:20px; /* красная строка  */
text-align:justify; /* выравнивание по ширине */
color:brown;   /* цвет текста */
}
h1{ /* стиль для заголовка  */
font-family: "Bookman Old Style";  /* гарнитура шрифта*/
font-size: 30px; /* размер шрифта */
border-style: double; /* стиль рамки */
border-width:3px;    /* ширина рамки */
border-color: brown; /* цвет рамки */
}
```

3. Прикрепите стилевой файл к какой-либо готовой веб-странице

Табличная верстка CSS

Благодаря большому числу свойств таблиц и вариациям их оформления, таблицы долгое время были некоторым станартом верстки веб-страниц. Если сделать границы таблицы невидимыми, то можно использовать ее отдельные ячейки в качестве отдельных блоков сраницы: шапка, меню, подвал и т.п.

Но это не совсем правильно, ведь каждому тегу есть свое назначение, и таблицы не должны были служить для верстки страниц. Однако отсутствие альтернативы сподвигало дизайнеров именно на такой метод верстки.

Сейчас есть другой способ - использование слоев, которые постепенно заменили таблицы в этом виде работы с веб-страницей. Однако и в наше время некоторые дизайнеры успешно используют табличную верстку.

ТАБЛИЧНАЯ ВЕРСТКА ИЗ ДВУХ КОЛОНОК

Один из самых распространённых способов верстки - две колонки, т.е. страница делится на две части.

- Обычно левая часть - меню и дополнительные элементы, а правая часть - основная, для главного контента.
- В таком случае ширина левой части задается определенным значением, т.е. жестко, а правая часть занимает оставшуюся область страницы.

- В таком случае необходимо задать общую ширину всей таблицы (тега table) в процентах через свойство width (100%), а для первой ячейки (тега td) установить ширину (также свойство width) в пикселах или процентах.

Пример: задать основной каркас страницы из двух колонок: первая - с фиксированным размером, вторая - на оставшуюся область браузера. Выполнить задание используя CSS стили (метод вложения)

Выполнение:

```
<style type="text/css">
/* для таблицы */
table#maket{
        width:100%;
        padding:5px; /* внутренние отступы */
        border-collapse:collapse; /* убираем двойную границу */
}
/* для левой ячейки */
td#left{
        width:200px;
}
/* для всех ячеек */
#maket td{
        vertical-align:top;
      border:1px solid black; /* временно обозначим границы */
}
</style>
</head>
<body>
<table id="maket" cellspacing="0">
        <tr>
                <td id="left">1</td>
                <td id="right">2</td>
        </tr>
</table>
...
```

Результат:

1	2

- Теперь попробуем визуально отделить две колонки таблицы друг от друга.

Пример: задать разный фон ячеек (чтобы разделить две колонки друг от друга) и установить расстояние между колонками (разделитель)

Выполнение:

Добавим новые свойства стилей:

```
/* для левой ячейки */
td#left{
        width:200px;
        background: #ccc; /* Цвет фона левой колонки */
```

```
        border:1px solid black; /* временно обозначим границы */
}
/* для правой ячейки */
td#right{
            background: #fc3; /* Цвет фона правой колонки */
        border:1px solid black; /* временно обозначим границы */
}
/* для разделителя */
#razdel{
    width: 10px; /* Расстояние между колонками */
   }
```

Все вместе:

```
<style type="text/css">
/* для таблицы */
table#maket{
            width:100%;
            padding:5px; /* внутренние отступы */
            border-collapse:collapse; /* убираем двойную границу */
}
/* для левой ячейки */
td#left{
            width:200px;
            background: #ccc; /* Цвет фона левой колонки */
        border:1px solid black; /* временно обозначим границы */
}
/* для правой ячейки */
td#right{
            background: #fc3; /* Цвет фона правой колонки */
        border:1px solid black; /* временно обозначим границы */
}
/* для всех ячеек */
#maket td{
            vertical-align:top;
}
/* для разделителя */
#razdel{
    width: 10px; /* Расстояние между колонками */
   }
</style>
</head>
<body>
<table id="maket" cellspacing="0">
            <tr>
                        <td id="left">1</td>
                        <td id="razdel"></td>
```

```
                <td id="right">2</td>
            </tr>
</table>
```

Для разделителя была добавлена новая ячейка.

Результат:

1	2

- Разделитель между колонками можно также сделать менее выделяющимся. Для этого воспользуемся стилями границ.

Пример: сделать разделитель между колонками таблицы, используя пунктирную линию границы смежных ячеек

Выполнение:

Добавим новые свойства границ для ячеек:

```
/* для левой ячейки */
td#left{
        width:200px;
        background: #ccc; /* Цвет фона левой колонки */
/* новое */
    border-right: 1px dashed #000; /* Параметры правой пунктирной границы */
}
```

Все вместе:

```
<style type="text/css">
/* для таблицы */
table#maket{
        width:100%;
        padding:5px; /* внутренние отступы */
        border-collapse:collapse; /* убираем двойную границу */
}
/* для левой ячейки */
td#left{
        width:200px;
        background: #ccc; /* Цвет фона левой колонки */
    border-right: 1px dashed #000; /* Параметры правой пунктирной границы */
}
/* для правой ячейки */
td#right{
        background: #fc3; /* Цвет фона правой колонки */
}
/* для всех ячеек */
#maket td{
        vertical-align:top;
}
</style>
</head>
<body>
```

```
<table id="maket" cellspacing="0">
        <tr>
                <td id="left">1</td>
                <td id="right">2</td>
        </tr>
</table>
```

Результат:

ТАБЛИЧНАЯ ВЕРСТКА ИЗ ТРЕХ КОЛОНОК

Существует понятие фиксированного или «резинового» макета верстки.

Фиксированный макет CSS

- При использовании **фиксированного макета ширина всей таблицы задается в пикселях**, и тогда, независимо от разрешения монитора и окна браузера, таблица будет всегда иметь одинаковую ширину.
- В таком случае **ширину остальных колонок стоит также сделать фиксированной**.
- Можно не указать ширину одной ячейки, тогда она будет вычислена автоматически, исходя из размеров остальных ячеек и всей таблицы.

Пример: создать шаблон страницы из трех колонок. Использовать фиксированный макет табличной верстки:

- левая колонка - 150 пикселей;
- средняя колонка - 400 пикселей;
- правая колонка - 200 пикселей;

Задать фон для колонок и визуально разделить колонки границей.

Выполнение:

```
<style type="text/css">
/* для таблицы */
table#maket{
        width:750px;
        padding:5px; /* внутренние отступы */
        border-collapse:collapse; /* убираем двойную границу */
}
/* для левой ячейки */
td#left{
        width:150px;
        background: #ccc; /* Цвет фона левой колонки */
        border-right: 1px dashed #000; /* граница между колонками */
}
/* для центральной ячейки */
td#central{
        width:400px;
        background: #fc3; /* Цвет фона колонки */
        border-right: 1px dashed #000; /* граница между колонками */
}
```

```
/* для правой ячейки */
td#right{
        width:200px;
        background: #ccc; /* Цвет фона правой колонки */
        }
/* для всех ячеек */
#maket td{
        vertical-align:top;
}
</style>
</head>
<body>
<table id="maket" cellspacing="0">
        <tr>
                <td id="left">1</td>
                <td id="central">2</td>
                <td id="right">3</td>
        </tr>
</table>
```

Результат:

Резиновый макет

- **Ширина таблицы** при использовании «резинового» дизайна **устанавливается в % от ширины окна браузера**. Т.о. при изменении окна браузера, изменяются и размеры таблицы.
- **Ширина всех ячеек** может устанавливаться **в процентах**.
- Второй вариант, когда **ширина некоторых ячеек** устанавливается **в процентах**, а **некоторых - в пикселях**.

Важно: Сумма ширины всех колонок должна получиться 100%, независимо от ширины таблицы.

Пример: создать шаблон страницы из трех колонок. Использовать резиновый макет табличной верстки:

- левая колонка - 20%;
- средняя колонка - 40%;
- правая колонка - 40%;

Задать фон для колонок и визуально разделить колонки границей.

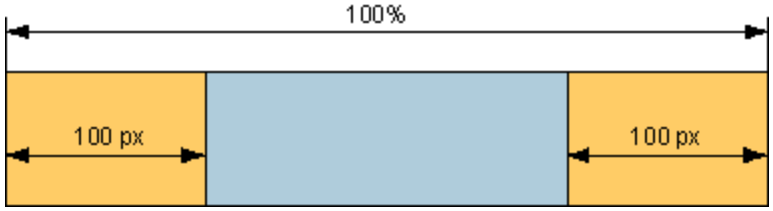
Выполнение:

```
<style type="text/css">
/* для таблицы */
table#maket{
        width:90%;
        padding:5px; /* внутренние отступы */
        border-collapse:collapse; /* убираем двойную границу */
}
```

```
/* для левой ячейки */
td#left{
        width:20%;
        background: #ccc; /* Цвет фона левой колонки */
        border-right: 1px dashed #000; /* граница между колонками */
}
/* для центральной ячейки */
td#central{
        width:40%;
        background: #fc3; /* Цвет фона колонки */
        border-right: 1px dashed #000; /* граница между колонками */
}
/* для правой ячейки */
td#right{
        width:40%;
        background: #ccc; /* Цвет фона правой колонки */
        }
/* для всех ячеек */
#maket td{
        vertical-align:top;
}
</style>
</head>
<body>
<table id="maket" cellspacing="0">
        <tr>
                <td id="left">1</td>
                <td id="central">2</td>
                <td id="right">3</td>
        </tr>
</table>
```

Результат:

Рассмотрим второй вариант, когда ширина центральной колонки автоматически подбирается браузером; примером может служить рисунок:



Пример: создать шаблон страницы из трех колонок. Использовать резиновый макет табличной верстки:

- левая колонка - 150 пикселей;
- средняя колонка - 40%;

- правая колонка - 200 пикселей;

Задать фон для колонок и визуально разделить колонки границей.

Выполнение:

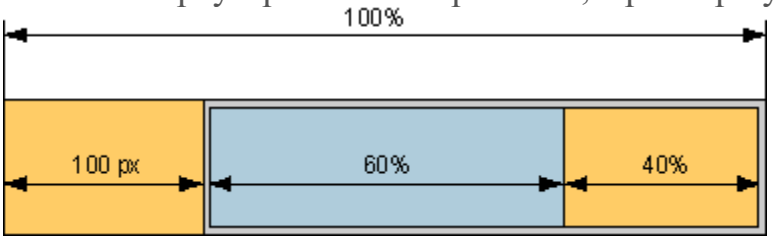
```
<style type="text/css">
/* для таблицы */
table#maket{
        width:90%;
        padding:5px; /* внутренние отступы */
        border-collapse:collapse; /* убираем двойную границу */
}
/* для левой ячейки */
td#left{
        width:150px;
        background: #ccc; /* Цвет фона левой колонки */
        border-right: 1px dashed #000; /* граница между колонками */
}
/* для центральной ячейки */
td#central{
        background: #fc3; /* Цвет фона колонки */
        border-right: 1px dashed #000; /* граница между колонками */
}
/* для правой ячейки */
td#right{
        width:200px;
        background: #ccc; /* Цвет фона правой колонки */
        }
/* для всех ячеек */
#maket td{
        vertical-align:top;
}
</style>
</head>
<body>
<table id="maket" cellspacing="0">
        <tr>
                <td id="left">1</td>
                <td id="central">2</td>
                <td id="right">3</td>
        </tr>
</table>
```

Результат:

Результат будет примерно такой же, только «растягивание» будет происходит за счет центральной колонки.

ИСПОЛЬЗОВАНИЕ ВЛОЖЕННОЙ ТАБЛИЦЫ В РЕЗИНОВОМ МАКЕТЕ

Если ширина двух колонок устанавливается в процентах, а третьей - в пикселях, обойтись одной таблицей не получится. Так, если ширина всей таблицы равна 100 процентов, первой колонки - 200 пикселей, а оставшихся колонок по 20 процентов, то простое вычисление показывает, что размер первой колонки получается равным 60 процентов. В таком случае заданное значение в пикселях браузером не воспримется, а размер будет установлен в процентах.



- Исходная таблица создается с двумя ячейками. **Ширина таблицы задается в процентах**.
- **Для левой ячейки** (первой колонки) устанавливается ширина **в пикселях**.
- **Ширина правой ячейки** (основа для других колонок) **не указывается**. Внутрь этой ячейки вставляется вторая таблица, тоже состоящая из двух ячеек.
- **У ячеек вложенной таблицы ширина устанавливается в процентах**.
- **Ширина внутренней таблицы должна быть установлена в 100 процентов**, чтобы эта таблица занимала все свободное пространство во внешней таблице.
- Ширина центральной и правой колонки вычисляется относительно ширины ячейки, а не внешней таблицы в целом.

Пример: создать шаблон страницы из трех колонок. Использовать резиновый макет с вложенной таблицей:

- левая колонка - 150 пикселей;
- средняя колонка - 60%;
- правая колонка - 40%;

Задать фон для колонок.

Выполнение:

```
<style type="text/css">
/* для таблицы */
table{
        width:100%;
        border-collapse:collapse; /* убираем двойную границу */
}
/* для левой ячейки */
td#left{
        width:150px;
        background: #ccc; /* Цвет фона левой колонки */
}
/* для центральной ячейки */
```

```
td#central{
        width:60%;
        background: #fc3; /* Цвет фона колонки */
}
/* для правой ячейки */
td#right{
        width:40%;
        background: #ccc; /* Цвет фона правой колонки */
        }
/* для всех ячеек */
td{
        vertical-align:top;
}
#left,#central,#right{
        padding:5px; /* внутренние отступы */
}
</style>
</head>
<body>
<table cellpadding="0" cellspacing="0">
        <tr>
                <td id="left">1</td>
                <td>
                <table cellpadding="0" cellspacing="0">
                        <td id="central">2</td>
                        <td id="right">3</td>
                </table>
                </td>
        </tr>
</table>
```

Атрибуты тегов cellpadding и cellspacing здесь необходимы, для того, чтобы не было «зазора» между таблицами.

Результат:

Вопросы для самоконтроля

1. Таблицы стилей.
2. Разделение контента от дизайна.
3. Библиотеки шаблонов.
4. Отказ от излишнего кода.
5. Проверка кода на ошибки.

Лабораторная работа №21. Позиционирование блоков.

Цель: изучить способы добавления блочной структуры.

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

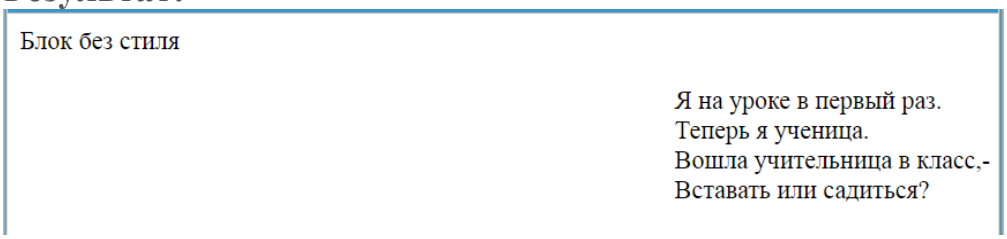
POSITION: FIXED

- 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

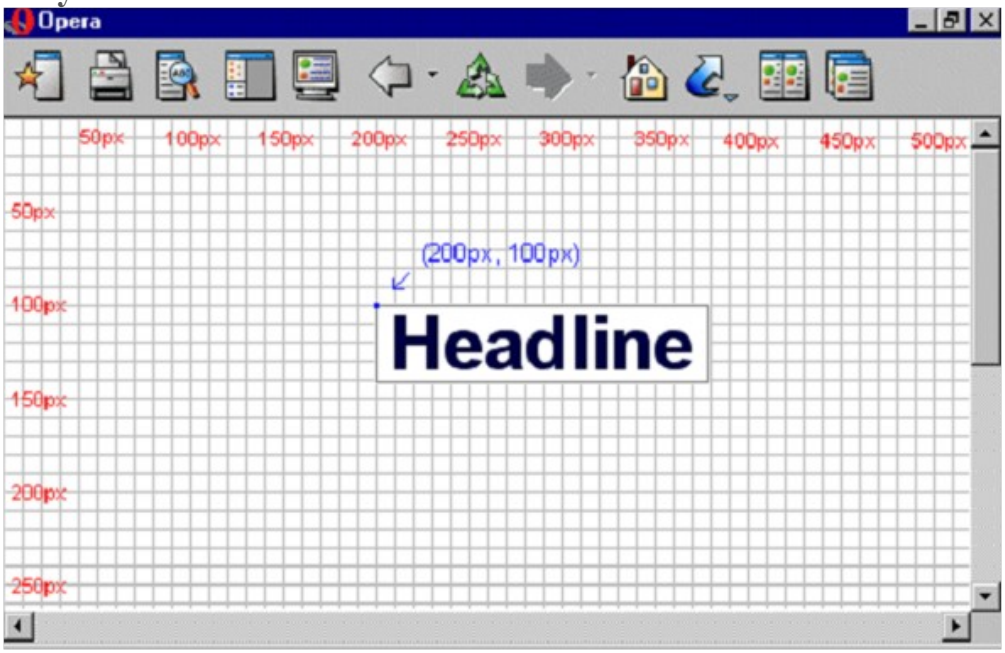
Значения:

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:

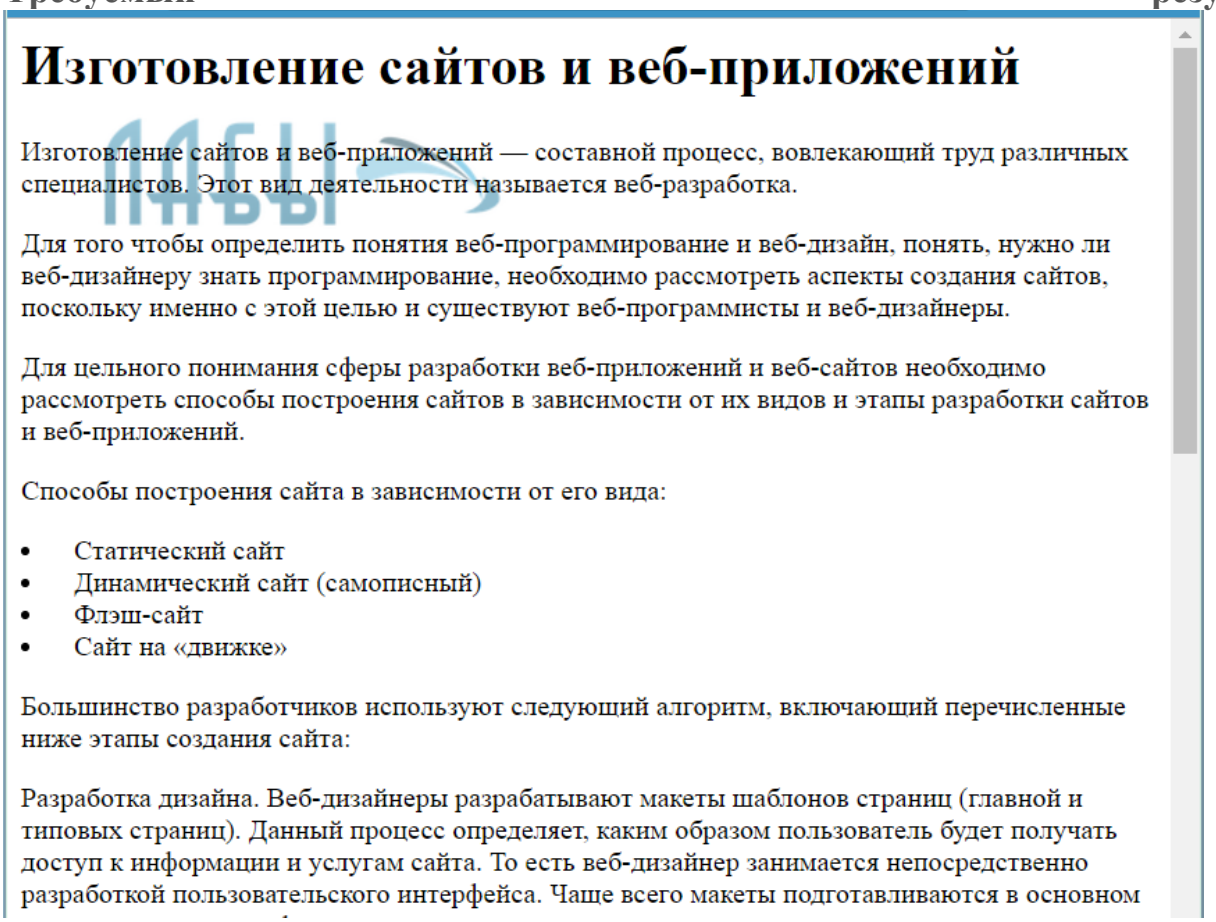
Задание: Скачайте файл.

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:



Изготовление сайтов и веб-приложений

Изготовление сайтов и веб-приложений — составной процесс, вовлекающий труд различных специалистов. Этот вид деятельности называется веб-разработка.

Для того чтобы определить понятия веб-программирование и веб-дизайн, понять, нужно ли веб-дизайнеру знать программирование, необходимо рассмотреть аспекты создания сайтов, поскольку именно с этой целью и существуют веб-программисты и веб-дизайнеры.

Для цельного понимания сферы разработки веб-приложений и веб-сайтов необходимо рассмотреть способы построения сайтов в зависимости от их видов и этапы разработки сайтов и веб-приложений.

Способы построения сайта в зависимости от его вида:

- Статический сайт
- Динамический сайт (самописный)
- Флэш-сайт
- Сайт на «движке»

Большинство разработчиков используют следующий алгоритм, включающий перечисленные ниже этапы создания сайта:

Разработка дизайна. Веб-дизайнеры разрабатывают макеты шаблонов страниц (главной и типовых страниц). Данный процесс определяет, каким образом пользователь будет получать доступ к информации и услугам сайта. То есть веб-дизайнер занимается непосредственно разработкой пользовательского интерфейса. Чаще всего макеты подготавливаются в основном

Вопросы для самоконтроля

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

Лабораторная работа №22. Другие свойства блоков CSS.

Цель: изучить способы оформления блоков.

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

Ход работы:

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:

float: left

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:

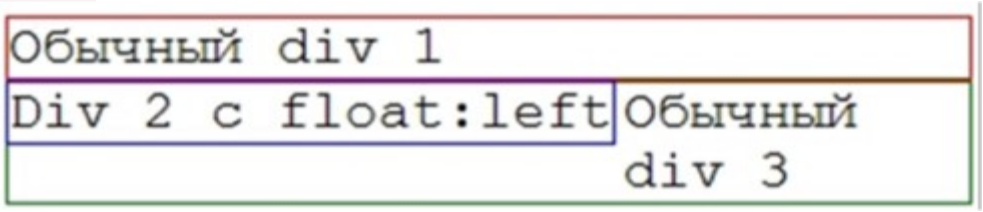


Рис. Расположение блоков при использовании свойства float

Возможные значения float:

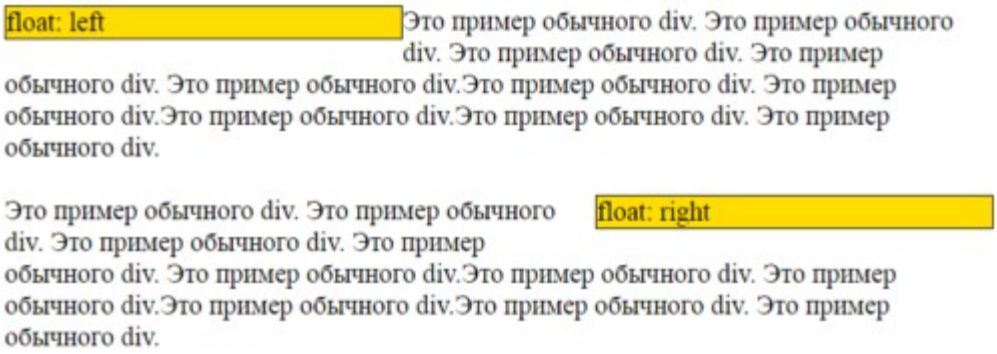
- left
  элемент выравнивается по левому краю, давая остальным элементам обтекать его справа
- right
  элемент выравнивается по правому краю, давая остальным элементам обтекать его слева
- none
  обтекание не задается (значение по умолчанию)
- inherit
  значение наследуется от родителя

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
*/
```

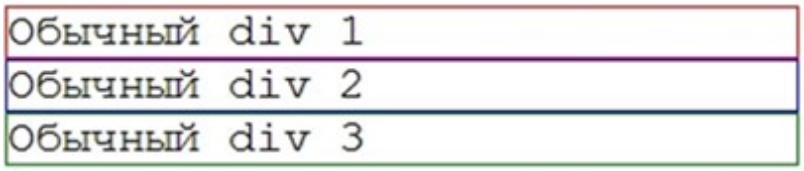
```
    background: #fd0; /* Цвет фона */
    border: 1px solid black; /* Параметры
рамки */
    width: 40%; /* Ширина блока */
}
</style></head>
<body>
<div class="left">float: left</div>
<div>Это пример обычного div. Это пример обычного div. Это пример обычного
div. Это пример обычного div. Это пример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

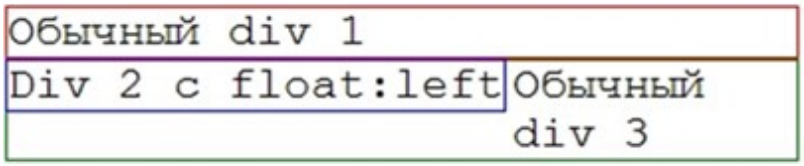
Результат:

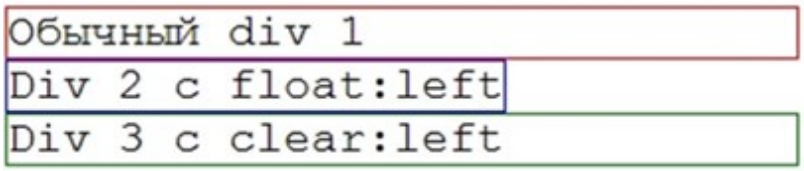


CLEAR

отмена обтекания







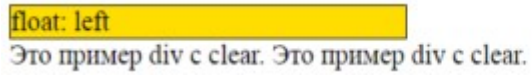
Возможные значения:

- 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.

Лабораторная работа №23. Блочная верстка сайта.

Цель: изучить способы верстки сайта инструментами css.

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

Ход работы:

Блочная верстка сайта

- При работе со слоями или, иначе говоря, блоками и элементами div, основная нагрузка ложится на CSS, т.к. без свойств слои из себя практически ничего не представляют.
- К сожалению до сих пор существует проблема с кроссбраузерностью при работе с блоками. Т.е. одни и те же свойства дают разный результат в разных браузерах. Для борьбы с такими проблемами существуют так называемые хаки. *Хак* - это набор приемов, когда отдельным браузерам подается код, который понимается только этим браузером, а остальными игнорируется.

ОТЛИЧИТЕЛЬНЫЕ ЧЕРТЫ ОТ ТАБЛИЧНОЙ ВЕРСТКИ

Допустим, необходимо создать шаблон страницы с «шапкой», «подвалом» и двумя колонками.

Отличия:

1. Высота слоев div ограничена высотой контента:

Табличная верстка	Блочная верстка

2. В случае, когда содержимое слоя превышает его установленную высоту, то браузеры по-разному ведут себя - одни увеличивают высоту слоя под новый контент, а другие, оставляя высоту первоначальной, накладывают контент поверх слоя.

Фиксированный дизайн или
жесткая блочная верстка (две колонки)

- Фиксированный макет подразумевает использование слоев заданной ширины, которая определяется разрешением монитора пользователя.
- Так как наиболее популярным среди пользователей сети является разрешение монитора 1024×768, то желательно ориентироваться именно

на него. Общая ширина блоков в таком случае составляет 900–1000 пикселей (небольшая часть пикселей требуется на полосы прокрутки и границы окна браузера).

- **Основной блок с контентом размещается по центру**, тогда «свободные» поля по краям неплохо смотрятся даже при большом разрешении монитора.

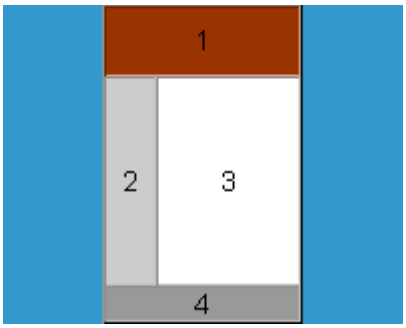


Рис.1. Пример фиксированного дизайна

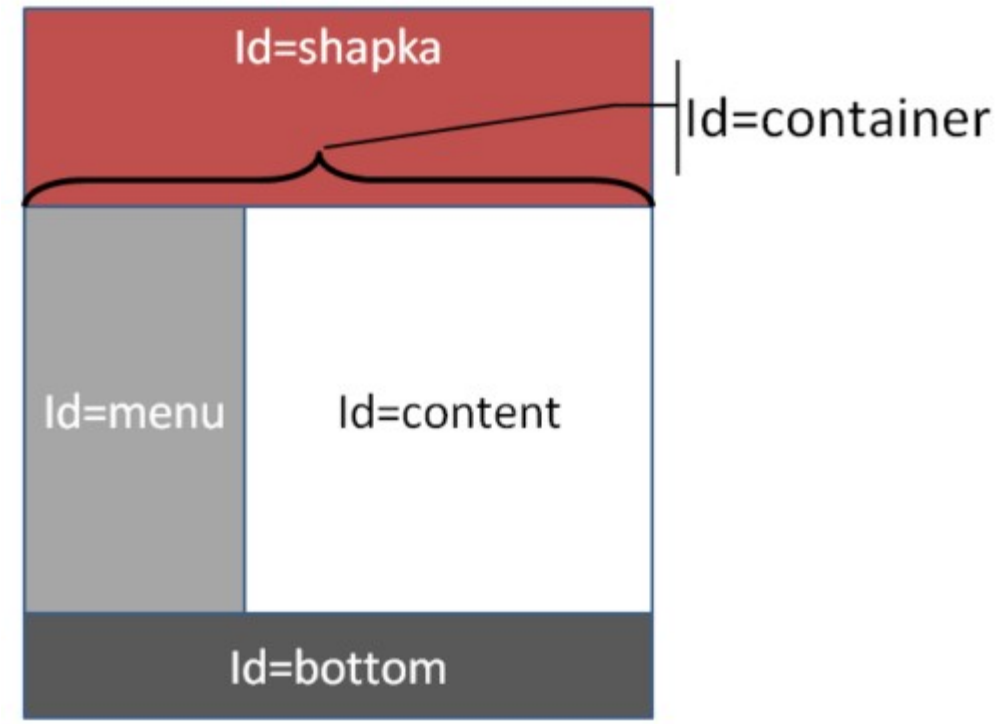
- Общая ширина может выбираться дизайнером, что называется «на глаз», или после сбора каких-либо дополнительных сведений.

Пример: создать фиксированный дизайн сайта на основе представленного выше изображения

Выполнение:

- «Разбиваем» все основные элементы страницы на блоки следующим образом:
  - **блок 1** - слой первый (id="shapka"),
  - **блок 2 и 3** заключаются в единый блок (id="container"),
  - **блок 2** - слой с меню (id="menu"),
  - **блок 3** - слой с контентом (id="content"),
  - **блок 4** - слой с id="bottom".

Схематично изобразим расположение блоков:




```
<body>
<div id="shapka">1</div>
<div id="container">
        <div id="menu">2</div>
        <div id="content">3</div>
</div>
<div id="bottom">4</div>
</body>
```

1. Задаем свойства «шапки» (блок 1)

- Выбираем общую ширину слоев «на глаз» - 750 пикселей.
  width: 750px;
- Центрируем блок (значение auto для свойств margin-left и margin-right)
  margin-right: auto;
  margin-left: auto;
- Задаем внутренние поля, чтобы текст не «приклеился» к краю слоя (свойство padding)
  padding: 10px;
- Высоту любого слоя, а в нашем случае высоту первого блока можно задать двумя способами:

1. либо **напрямую задав значение свойству height** в пикселях, процентах или др. единицах;
   например, для верхнего слоя «шапки»:
   height: 50px;
2. либо **задать высоту при помощи отступа**
   например:

```
    padding-top: 15px;
    padding-bottom: 15px;
```

Весь код для шапки:

```
#shapka{
 text-align: left; /* Выравнивание внутреннего контента по левому краю */
 width: 750px; /* Ширина блока и общая ширина*/
 background: #900000; /* Цвет фона */
 height: 50px; /* Высота блока */
 margin-right: auto; /* Авто-отступ справа */
 margin-left: auto; /* Авто-отступ слева */
 padding: 10px; /* Внутренние поля вокруг содержимого */
}
```

2. Создаем свойства контейнера

- Задаем ширину контейнера: она должна быть 750px + 20px (общая ширина макета + внутренние отступы в общей ширине макета). Т.е. в контейнере нет внутренних отступов, поэтому увеличим его ширину на 10 пикселей слева и 10 пикселей справа
- Если макет должен прилегать к левой части экрана, то браузер выполнит это автоматически. В нашем случае отцентрируем блок контейнера

```
#container {
 width: 770px; /* Ширина слоя или (ширина макета+20) */
 margin-right: auto; /* Авто-отступ справа */
 margin-left: auto; /* Авто-отступ слева */
}
```

3. Создаем свойства для блока 2 - меню

- Слой номер 2 шириной в 200 пикселей (width)
- Для этого же слоя (меню) устанавливаем обтекание соседним блоком, т.е. свойство float: left
- Задаем внутренние поля, чтобы текст не «приклеился» к краю слоя (свойство padding)
- Задаем цвет текста и заднего фона (background, color)

```
#menu {
 width: 200px; /* Ширина слоя */
 float: left; /* Обтекание с соседним слоем */
 color: white; /* Цвет текста */
 background: #008080; /* Цвет фона */
 padding: 10px; /* Внутренние поля вокруг содержимого */
}
```

4. Создаем свойства для блока 3 - контент

- Задаем ширину слоя из расчета 770px - 200px = 570px, **Но!** так как мы установили внутренние отступы в обоих блоках 2 и 3, то мы должны вычесть еще 40 пикселей: 20 - внутренние отступы блока с меню и 20 - внутренние отступы блока с контентом. Получим ширину слоя 770px - 200px - 40px = 530px

- Задаем обтекание float: left, исключая баг браузера Internet Explorer: если не установить свойство, то между слоями будет зазор. Кроме того, если не установить это свойство, то блок окажется за блоком меню, и только лишь его контент (текст) будет обтекать блок меню справа.
- Задаем цвет заднего фона (background) и внутренние поля (padding)

```
#content { /* Правая колонка*/
    width: 550px; /* Ширина слоя */
    float: left; /* Обтекание с соседним слоем */
    padding: 10px; /* Внутренние поля вокруг содержимого */
    background: #e0e0e0; /* Цвет заднего фона */
   }
```

5. Создаем свойства для блока 4 - «подвал»

- Ширину слоя устанавливаем в 750 пикселей
- Для этого блока надо убрать обтекание, т.е. установить свойство clear
- Устанавливаем внутренние поля padding
- Задаем цвет для фона (background) и текста (color)
- Центрируем блок (margin-right и margin-left)

```
#bottom{
        width:750px; /* Ширина слоя */
        clear:left; /* возвращаем блочность и располагаем слой слева */
        padding: 10px; /* Внутренние поля вокруг содержимого */
    background:#444; /* фон */
        color:#fff; /* цвет текста */
    margin-right: auto; /* Авто-отступ справа */
    margin-left: auto; /* Авто-отступ слева */
}
```

Итоговый код: всё вместе

```
<style type="text/css">
/* для блока 1 - шапка */
#shapka{
  text-align: left; /* Выравнивание внутреннего контента по левому краю */
  width: 750px; /* Ширина блока и общая ширина*/
  background: #900000; /* Цвет фона */
  height: 50px; /* Высота блока */
  margin-right: auto; /* Авто-отступ справа */
  margin-left: auto; /* Авто-отступ слева */
  padding: 10px; /* Внутренние поля вокруг содержимого */
}
/* для контейнера */
#container {
  width: 770px; /* Ширина слоя или ширина макета+20px */
  margin-right: auto; /* Авто-отступ справа */
  margin-left: auto; /* Авто-отступ слева */
 }
```

```
/* для блока 2 - меню */
#menu {
  width: 200px; /* Ширина слоя */
  float: left; /* Обтекание с соседним слоем */
  color: white; /* Цвет текста */
  background: #008080; /* Цвет фона */
  padding: 10px; /* Внутренние поля вокруг содержимого */
 }
/* для блока 3 - контент */
#content { /* Правая колонка*/
    width: 530px; /* Ширина слоя */
    float: left; /* Обтекание с соседним слоем */
    padding: 10px; /* Внутренние поля вокруг содержимого */
    background: #e0e0e0; /* Цвет заднего фона */
   }
/* для блока 4 - подвал */
#bottom{
            width:750px; /* Ширина слоя */
            clear:left; /* возвращаем блочность и располагаем слой слева */
            padding: 10px; /* Внутренние поля вокруг содержимого */
            background:#444;
            color:#fff;
            margin-right: auto; /* Авто-отступ справа */
    margin-left: auto; /* Авто-отступ слева */
}
</style>
</head>
<body>
<div id="shapka">1</div>
<div id="container">
            <div id="menu">2</div>
            <div id="content">3</div>
</div>
<div id="bottom">4</div>
</body>
```

Результат:

Рис.2. Жесткая блочная верстка из двух колонок

Фиксированный дизайн для трех колонок

При фиксированном дизайне для макета из трех колонок можно выделить два основных подхода построения модульной сетки:

1. Использование свойства float для расположения колонок рядом.
2. Использование набора тех CSS свойств, которые предназначены для позиционирования слоев.

Рассмотрим первый случай.

ИСПОЛЬЗОВАНИЕ СВОЙСТВА FLOAT ДЛЯ МАКЕТА В ТРИ КОЛОНКИ

На *рис. 3* - результат использования свойства float для трехколонного макета. На самом деле используется 6 слоев - отдельно для заголовков колонок и отдельно для самих колонок.

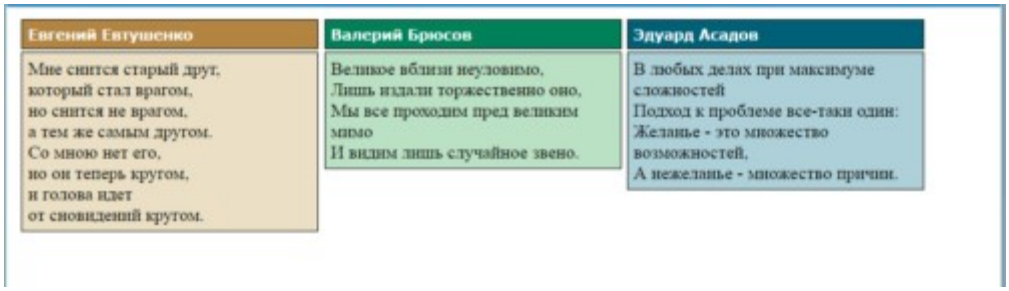


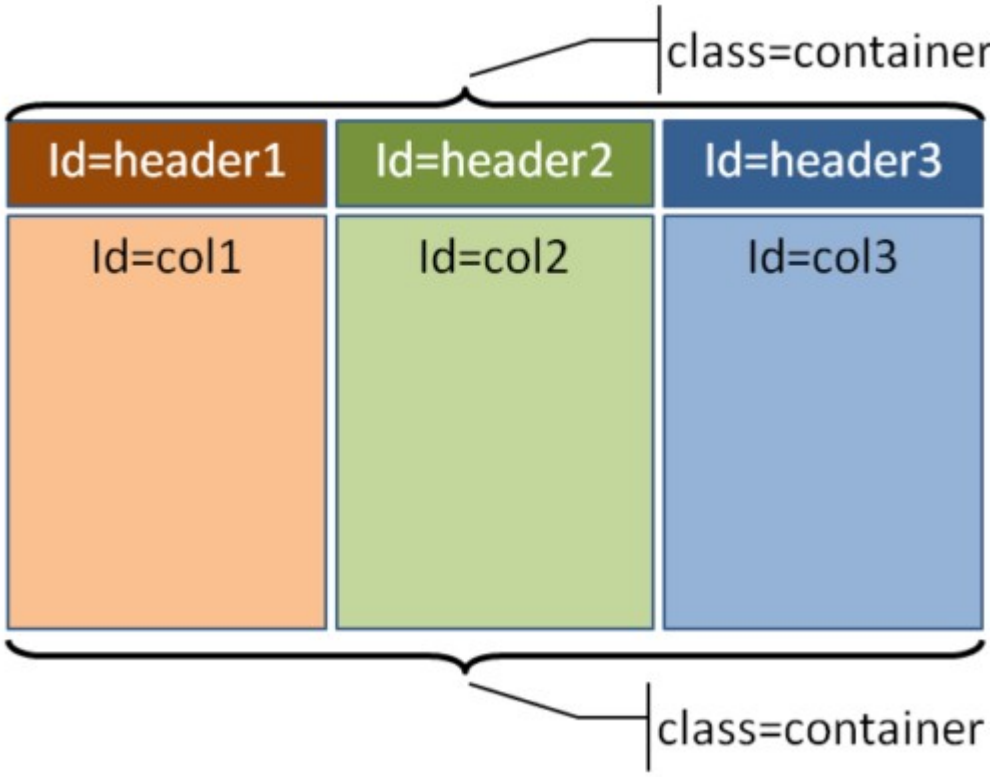
Рис. 3. Фиксированный дизайн в три колонки

Пример: создать веб-страницу с макетом из трех колонок, изображенную на *рис. 3*. Использовать приемы фиксированного блочного макетирования

Выполнение:

1. Создание структуры html-кода

- Определим для заголовков три слоя (#header...) и для колонок три слоя (#col...).
- Так как колонки и их заголовки расположены на двух разных строках, то объеденим их в контейнеры (class="container").
- Получим схематичное изображение расположения блоков:



Получим следующую html-структуру:

1 <**body**>

```
<div class="container">
  <div id="header1">Евгений Евтушенко</div>
  <div id="header2">Валерий Брюсов</div>
  <div id="header3">Эдуард Асадов</div>
 </div>
 <div class="container">
  <div id="col1">
                Мне снится старый друг,<br/>который стал врагом,<br/>
                но снится не врагом,<br/>а тем же самым другом.<br/>
                Со мною нет его,<br/>но он теперь кругом,<br/>
                и голова идет<br/>от сновидений кругом.
        </div>
  <div id="col2">
        Великое вблизи неуловимо,<br/>Лишь издали торжественно оно,<br/>
        Мы все проходим пред великим мимо<br/>И видим лишь случайное звено.
        </div>
  <div id="col3">
        В любых делах при максимуме сложностей<br/>Подход к проблеме все-таки
один:<br/>
        Желанье - это множество возможностей,<br/>А нежеланье - множество причи
        </div>
 </div>
</body>
```

2. Добавление стилей для заголовков (селектор header...) и колонок (селектор col...)

- Ширину колонок и заголовков сделаем у всех одинаковой. Поскольку колонок 3, а средняя ширина страницы должна быть примерно 700-900 пикселей, то установим ширину колонок в 250 пикселей.

```
#header1, #header2, #header3, #col1, #col2, #col3 {
  width: 250px; /* Ширина колонок */
}
```

- Добавим внутренние поля (отступы от содержимого текста) - padding и внешние отступы, чтобы обеспечить зазор между колонками. Поскольку свойства задаются сразу для все колонок одновременно, а сам весь макет не центрируется, а выравнивается по левому краю, то установим отступ только с одной стороны - левой - у всех колонок одновременно (margin-left).

```
#header1, #header2, #header3, #col1, #col2, #col3 {
  width: 250px; /* Ширина колонок */
  padding: 5px; /* Поля вокруг текста */
  margin-left: 5px; /* Отступ слева */
 }
```

- Добавим также внешний отступ сверху, обеспечив зазор по вертикали между заголовками и колонками, а также отступ заголовков от верха страницы (margin-top).

```
#header1, #header2, #header3, #col1, #col2, #col3 {
  ...
  margin-top: 2px; /* Отступ сверху */
 }
```

- Для того, чтобы блочные теги div, расположились рядом друг с другом, необходимо задать им обтекание - float.

```
#header1, #header2, #header3, #col1, #col2, #col3 {
  ...
  float: left; /* Состыковка колонок по горизонтали */
 }
```

- Добавляем границу для блоков (border) и и задаем параметры шрифта (font-family, font-weight, font-size, color).

```
#header1, #header2, #header3, #col1, #col2, #col3 {
  ...
  border: 1px solid black; /* Рамка вокруг слоя */
  font-family: Verdana, Arial, sans-serif; /* Не серифный или рубленый шрифт */
  font-weight: bold; /* Жирное начертание текста заголовка */
  font-size: 80%; /* Размер шрифта */
  color: white; /* Цвет текста заголовка */
 }
```

Получим код:

```
#header1, #header2, #header3, #col1, #col2, #col3 {
  width: 250px; /* Ширина колонок */
  padding: 5px; /* Поля вокруг текста */
  margin-left: 5px; /* Отступ слева */
  margin-top: 2px; /* Отступ сверху */
  float: left; /* Состыковка колонок по горизонтали */
  border: 1px solid black; /* Рамка вокруг слоя */
  font-family: Verdana, Arial, sans-serif; /* Не серифный или рубленый шрифт */
  font-weight: bold; /* Жирное начертание текста заголовка */
  font-size: 80%; /* Размер шрифта */
  color: white; /* Цвет текста заголовка */
}
```

- Установим задний фон отдельно для каждого селектора.

```
#header1 { background: #B38541; }
#header2 { background: #008159; }
#header3 { background: #006077; }
#col1 { background: #EBE0C5; }
#col2 { background: #BBE1C4; }
#col3 { background: #ADD0D9; }
```

Смотрим промежуточный результат:

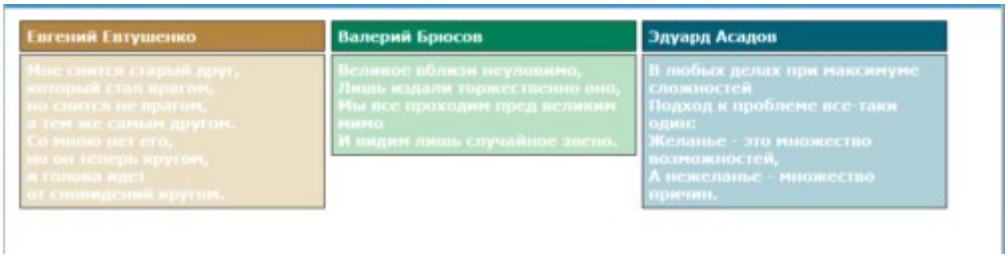


Рис. 4. Промежуточный результат

3. Задание стиля для контейнеров

- Теперь необходимо объединить в отдельные слои-контейнеры заголовки и колонки и задать им соответствующий стиль - отмену обтекания (clear)

```
.container {
    clear: both;  /* Отменяет действие обтекания float */
  }
```

- Теперь осталось доработать шрифт текста в колонках, изменив его размер, начертание и цвет.

```
#col1, #col2, #col3 {
  font-family: "Times New Roman", Times, serif; /* Шрифт серифный или с засечками */
  font-size: 100%; /* Размер шрифта */
  font-weight: normal; /* Нормальное начертание */
  color: black; /* Цвет текста */
 }
```

Итоговый код: всё вместе

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

/* для колонок и их заголовков */
 #header1, #header2, #header3, #col1, #col2, #col3 {
    width: 250px; /* Ширина колонок */
  padding: 5px; /* Поля вокруг текста */
  margin-left: 5px; /* Отступ слева */
  margin-top: 2px; /* Отступ сверху */
  float: left; /* Состыковка колонок по горизонтали */
  border: 1px solid black; /* Рамка вокруг слоя */
  font-family: Verdana, Arial, sans-serif; /* Рубленый шрифт */
  font-weight: bold; /* Жирное начертание текста заголовка */
  font-size: 80%; /* Размер шрифта */
  color: white; /* Цвет текста заголовка */
 }
 /* для колонок */
 #col1, #col2, #col3 {
  font-family: "Times New Roman", Times, serif; /* Шрифт с засечками */
  font-size: 100%; /* Размер шрифта */
  font-weight: normal; /* Нормальное начертание */
  color: black; /* Цвет текста */
 }

 /* Цвет фона каждого слоя */
```

```
  #header1 { background: #B38541; }
  #header2 { background: #008159; }
  #header3 { background: #006077; }
  #col1 { background: #EBE0C5; }
  #col2 { background: #BBE1C4; }
  #col3 { background: #ADD0D9; }

  .container {
    clear: both;  /* Отменяет действие float */
  }
</style>
</head>
<body>
<div class="container">
   <div id="header1">Евгений Евтушенко</div>
   <div id="header2">Валерий Брюсов</div>
   <div id="header3">Эдуард Асадов</div>
 </div>
 <div class="container">
   <div id="col1">
                  Мне снится старый друг,<br/>
                  который стал врагом,<br/>
                  но снится не врагом,<br/>
                  а тем же самым другом.<br/>
                  Со мною нет его,<br/>
                  но он теперь кругом,<br/>
                  и голова идет<br/>
                  от сновидений кругом.
          </div>
   <div id="col2">
          Великое вблизи неуловимо,<br/>
          Лишь издали торжественно оно,<br/>
          Мы все проходим пред великим мимо<br/>
          И видим лишь случайное звено.
          </div>
   <div id="col3">
          В любых делах при максимуме сложностей<br/>
          Подход к проблеме все-таки один:<br/>
          Желанье - это множество возможностей,<br/>
          А нежеланье - множество причин.
          </div>
 </div>
</body>
```

Результат:

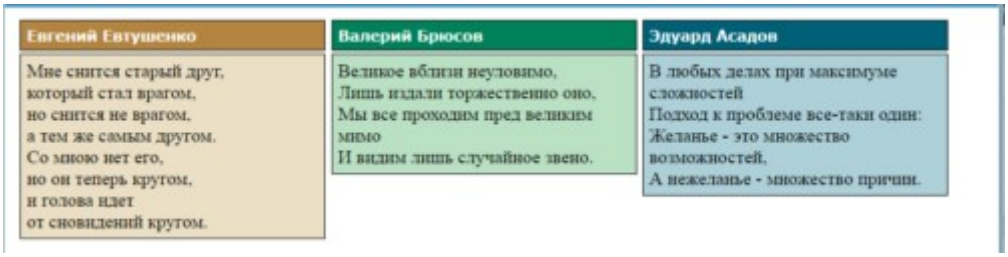


Рис. 5. Результат

ИСПОЛЬЗОВАНИЕ ПОЗИЦИОНИРОВАНИЯ СЛОЕВ ДЛЯ МАКЕТА В ТРИ КОЛОНКИ

Возьмем в качестве примера макет, состоящий из трех колонок разграниченных разделительной линией (рис. 6).

Иван Бунин На окне, серебряном от инея, За ночь хризантемы расцвели. В верхних стёклах - небо ярко-синее И застреха в снеговой пыли.	Всходит солнце, бодрое от холода, Золотится отблеском окно. Утро тихо, радостно и молодо. Белым снегом всё запушено.	И всё утро яркие и чистые Буду видеть краски в вышине, И до полдня будут серебристые Хризантемы на моём окне. 1903

Рис. 6. Позиционирование слоев для макета в три колонки с разделителем

Пример: создать веб-страницу с макетом из трех колонок с разделительной линией, изображенную на *рис. 6*. Использовать приемы фиксированного макетирования с позиционированием слоев

Выполнение:

1. Создание структуры html-кода

- Для данного макета достаточно создать три слоя div для каждой колонки.

Поэтому структура будет простой:

```
<body>
<div id="col1">Иван Бунин</br></br>
        На окне, серебряном от инея,</br>
        За ночь хризантемы расцвели.</br>
        В верхних стёклах - небо ярко-синее</br>
        И застреха в снеговой пыли.
</div>
<div id="col2">
        Всходит солнце, бодрое от холода, </br>
        Золотится отблеском окно. </br>
        Утро тихо, радостно и молодо. </br>
        Белым снегом всё запушено.
</div>
<div id="col3">
        И всё утро яркие и чистые</br>
        Буду видеть краски в вышине, </br>
        И до полдня будут серебристые </br>
        Хризантемы на моём окне.</br></br>

        1903
</div>
```

</body>

2. Добавление стилей для колонок

- Установим одинаковую ширину колонок (width) и внутренние поля по вертикали и горизонтали (padding).

```
#col1, #col2, #col3 {
    width: 250px; /* Ширина колонок */
    padding: 0 6px; /* Поля по вертикали и горизонтали */
   }
```

- Для удаления блочности слоев, т.е. для того, чтобы расположить их рядом по горизонтали, необходимо задать свойство css float.

```
#col1, #col2, #col3 {
    width: 250px; /* Ширина колонок */
    padding: 0 6px; /* Поля по вертикали и горизонтали */
    float: left; /* Обтекание слоев */
   }
```

- Так как разделительная граница должна присутствовать только с внутренних сторон слоев, то рамку необходимо добавить только у двух слоев с одной стороны (border).

```
#col1, #col2 {
    border-right: 1px solid #000; /* Параметры линии справа от текста */
   }
```

Итоговый код: всё вместе

```
<style type="text/css">
#col1, #col2, #col3 {
    width: 250px; /* Ширина колонок */
    padding: 0 6px; /* Поля по вертикали и горизонтали */
    float: left; /* Обтекание слоев */
   }
#col1, #col2 {
    border-right: 1px solid #000; /* Параметры линии справа от текста */
   }
</style>
</head>
<body>
<div id="col1">Иван Бунин</br></br>
        На окне, серебряном от инея,</br>
        За ночь хризантемы расцвели.</br>
        В верхних стёклах - небо ярко-синее</br>
        И застреха в снеговой пыли.
</div>
<div id="col2">
        Всходит солнце, бодрое от холода, </br>
        Золотится отблеском окно. </br>
```

```
        Утро тихо, радостно и молодо. </br>
        Белым снегом всё запушено.
</div>
<div id="col3">
        И всё утро яркие и чистые</br>
        Буду видеть краски в вышине, </br>
        И до полдня будут серебристые </br>
        Хризантемы на моём окне.</br></br>

        1903
</div>
</body>
```

Дизайн в три колонки готов!

Вопросы для самоконтроля

1. Выбор хостинга.
2. Подбор доменного имени.
3. Загрузка сайта на сервер.

Лабораторная работа №24. Блочная верстка сайта: абсолютное и относительное позиционирование; резиновый дизайн.

Цель: изучить способы настройки позиционирования и растяжения элементов сайта.

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

Ход работы:

Абсолютное позиционирование CSS в фиксированном дизайне

Данный вариант верстки, который еще носит название жесткая блочная верстка, предполагает размещение слоев с определением их точных координат относительно края окна браузера. Соответственно, при изменении размера окна размещение слоев не меняется, но при этом может произойти так, что слои не поместятся на странице и появится горизонтальная полоса прокрутки.

Таким образом, **недостатки верстки с фиксированным абсолютным позиционированием:**

- сложность последующих модификаций дизайна;
- жесткая привязка к координатной сетке.

НЕОБХОДИМЫЕ СВОЙСТВА

POSITION

1. Для всех основных слоев устанавливается абсолютное позиционирование: position: absolute

   WIDTH (HEIGHT)
2. Для всех слоев необходимо задать ширину (иногда и высоту)

   TOP И LEFT (RIGHT)
3. Необходимо задать координаты слоев по горизонтали и вертикали. Достаточно двух координат, например, расстояние от левого края браузера до левого края слоя - left (иногда легче задать расстояние от правого края - right) и расстояние от верхнего края браузера до слоя - top

Пример: Создать дизайн страницы, изображенной на *рис. 1*. Выполнить блочную верстку с абсолютным позиционированием слоев

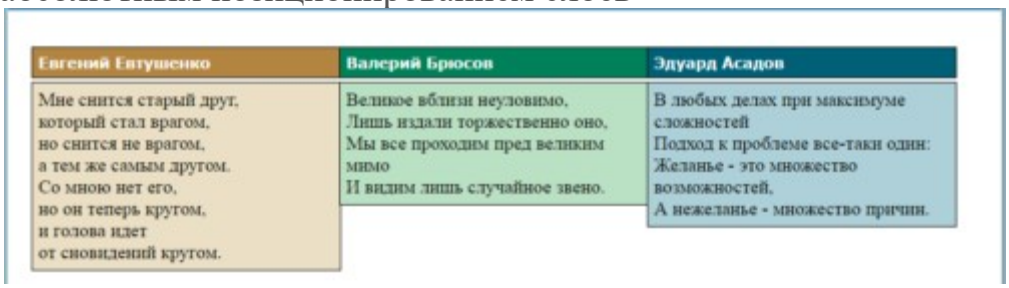


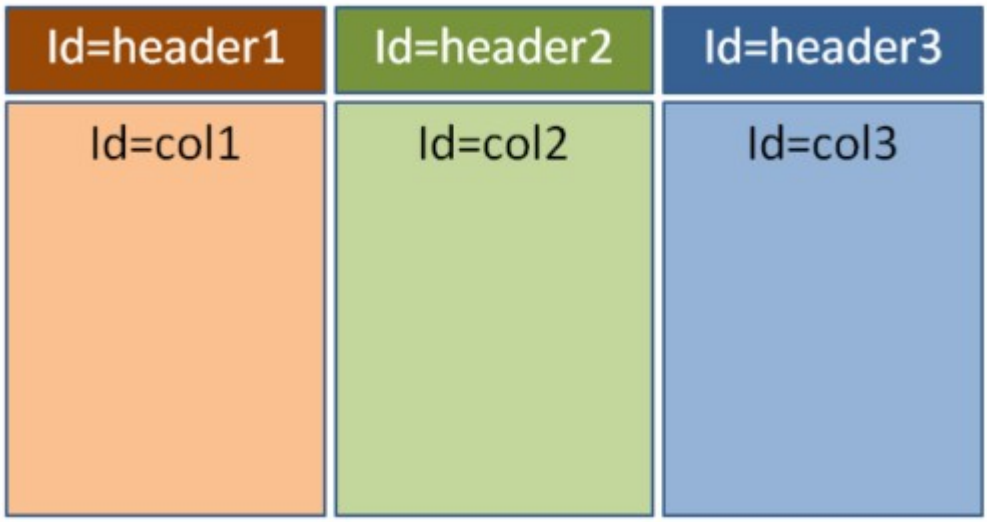
Рис. 1. Пример абсолютного позиционирования слоев в блочной верстке

Выполнение:

1. «Разбиваем» все основные элементы страницы на блоки следующим образом:

- Отдельные слои для заголовков с id: header1, header2, header3
- Отдельные слои для основных колонок с контентом: col1, col2, col3

- Схематично получаем следующее размещение блоков:




```
<body>
   <div id="header1">Евгений Евтушенко</div>
   <div id="header2">Валерий Брюсов</div>
   <div id="header3">Эдуард Асадов</div>
   <div id="col1">
                  Мне снится старый друг,<br/>
                  который стал врагом,<br/>
                  но снится не врагом,<br/>
                  а тем же самым другом.<br/>
                  Со мною нет его,<br/>
                  но он теперь кругом,<br/>
                  и голова идет<br/>
                  от сновидений кругом.
   </div>
   <div id="col2">
         Великое вблизи неуловимо,<br/>
         Лишь издали торжественно оно,<br/>
         Мы все проходим пред великим мимо<br/>
         И видим лишь случайное звено.
         </div>
   <div id="col3">
         В любых делах при максимуме сложностей<br/>
         Подход к проблеме все-таки один:<br/>
         Желанье - это множество возможностей,<br/>
         А нежеланье - множество причин.
   </div>
</body>
```

2. Задаем общие свойства для колонок и заголовков, касающиеся оформления шрифта, ширины, границ и внутренних отступов:

```
#header1, #header2, #header3, #col1, #col2, #col3 {
   width: 250px; /* Ширина колонок */
```

```
    border: 1px solid black; /* Рамка вокруг слоя */
    padding: 5px; /* Поля вокруг текста */
    font-family: Verdana, Arial, sans-serif; /* Не серифный или рубленый шрифт */
    font-weight: bold; /* Жирное начертание текста заголовка */
    font-size: 80%; /* Размер шрифта */
    color: white; /* Цвет текста заголовка */
}
```

3. Добавляем для тех же селекторов свойства абсолютного позиционирования:

```
...
    position: absolute; /* Абсолютное позиционируем */
    top: 30px; /* Расстояние от верхнего края */
...
```

4. Отдельно создаем свойства для колонок. Для позиционирования устанавливаем расстояние от верха (top):

```
#col1, #col2, #col3 {
    font-family: "Times New Roman", Times, serif; /* Шрифт с засечками */
    font-size: 100%; /* Размер шрифта */
    font-weight: normal; /* Нормальное начертание */
    color: black; /* Цвет текста */
    top: 60px; /* Расстояние от верхнего края */
}
```

5. Для каждого слоя необходимо отдельно задать координату расстояния от левого края (left):

```
  #header1 { left: 20px; }
  #header2 { left: 280px; } /* 20 + 250 + 5 + 5 */
  #header3 { left: 540px; } /* 20 + 250 + 250 + 5 + 5 + 5 + 5 */
  #col1 { left: 20px; }
  #col2 { left: 280px; }
  #col3 { left: 540px; }
```

6. Для каждого слоя отдельно задаем фон:

```
  #header1 { background: #B38541; }
  #header2 { background: #008159; }
  #header3 { background: #006077; }
  #col1 { background: #EBE0C5; }
  #col2 { background: #BBE1C4; }
  #col3 { background: #ADD0D9; }
```

РЕЗЮМЕ ПО АБСОЛЮТНОМУ ПОЗИЦИОНИРОВАНИЮ

Самыми главными свойствами, необходимыми для абсолютного позиционирования слоев являются:

1. Ширина всех слоев - свойство width:

- для всех слоев width: 250px;

2. Установка абсолютного позиционирования для всех слоев:

- для всех слоев position: absolute;

3. Установка двух координат (одной - по вертикали, другой - по горизонтали) для каждого слоя:

- Координаты по вертикали (от верхнего края) для всех слоев заголовков header: top: 30px;
- Координаты по вертикали (от верхнего края) для всех слоев колонок col: top: 60px;
- Координаты по горизонтали (от левого края) для всех слоев заголовков:
  - #header1{ left: 20px; }
  - #header2{ left: 280px; } (20 + 250 + 5 + 5)
  - #header3{ left: 540px; } (20 + 250 + 250 + 5 + 5 + 5 + 5)

Весь код стиля будет выглядеть следующим образом:

```
/* для колонок и их заголовков */
#header1, #header2, #header3, #col1, #col2, #col3 {
    font-family: Verdana, Arial, sans-serif; /* Рубленый шрифт */
    font-weight: bold; /* Жирное начертание текста заголовка */
    font-size: 80%; /* Размер шрифта */
    color: white; /* Цвет текста заголовка */
    width: 250px; /* Ширина колонок */
    border: 1px solid black; /* Рамка вокруг слоя */
    padding: 5px; /* Поля вокруг текста */
          /* позиционируем */
    position: absolute; /* Абсолютное позиционируем */
    top: 30px; /* Расстояние от верхнего края */
}
   /* для колонок */
#col1, #col2, #col3 {
    font-family: "Times New Roman", Times, serif; /* Шрифт с засечками */
    font-size: 100%; /* Размер шрифта */
    font-weight: normal; /* Нормальное начертание */
    color: black; /* Цвет текста */
    top: 60px; /* Расстояние от верхнего края */
}

   /* Расстояние от левого края отдельно каждого слоя */
   #header1 { left: 20px; }
   #header2 { left: 280px; } /* 20 + 250 + 5 + 5 */
   #header3 { left: 540px; } /* 20 + 250 + 250 + 5 + 5 + 5 + 5 */
   #col1 { left: 20px; }
   #col2 { left: 280px; }
   #col3 { left: 540px; }

   /* Цвет фона каждого слоя */
   #header1 { background: #B38541; }
   #header2 { background: #008159; }
   #header3 { background: #006077; }
```

```
#col1 { background: #EBE0C5; }
#col2 { background: #BBE1C4; }
#col3 { background: #ADD0D9; }
```

В результате получаем веб-страницу, как на *рис. 1* задания.

Относительное позиционирование CSS в фиксированном дизайне

Один из наиболее распространенных вариантов верстки. Блочная верстка css с относительным позиционированием подразумевает, что слои привязываются друг к другу, а их точные координаты относительно окна браузера не задаются. Поэтому сам макет можно центрировать на странице или выравнивать как по левому краю, так и по правому.

НЕОБХОДИМЫЕ СВОЙСТВА

Для двух колонок

При наличии двух колонок, для того чтобы располагать их в любом местоположении горизонтальной плоскости (по центру, слева, справа), необходимо **помещение их в общий контейнер**:

```
<div class="container">
   <div id="col1">
         колонка 1
   </div>
   <div id="col2">
         колонка 2
   </div>
</div>
```

Тогда необходимы следующие свойства:

WIDTH

Ширина задается как контейнеру, так и обеим колонкам. Для контейнера она рассчитывается как сумма ширины двух колонок (при отсутствии иных отступов).

```
.container{
         width:700px;
}
```

Ширина колонок может быть любой, с учетом что сумма обоих показателей вместе с другими отступами не превысит ширину контейнера. Для того чтобы расположить колонки плотно друг другу сумма их ширины должна соответствовать ширине контейнера (конечно, нужно учесть и другие отступы, включая padding)

```
#col1, #col2{
      border: 1px solid black; /* Рамка вокруг слоя */
         padding: 5px; /* Поля вокруг текста */
         width: 338px; /* Ширина колонок */
}
```

В данном примере ширина колонок рассчитана, как **ширина контейнера 700px - внутренние отступы padding 5+5+5+5px - ширина рамки 1+1+1+1 px = 676px на обе колонки**. Для одной колонки получаем ширину 338px. (На рамку на самом деле при заданном 1px отводится 0.8px, поэтому чтобы расположить колонки стык в стык, лучше задать ширину 338.5px)

FLOAT

Для расположения колонок рядом друг с другом, необходимо удалить у них блочность. Для этого левую колонку располагаем слева, и правую, если колонки идут стык в стык, тоже слева. если между колонками предусмотрен зазор, то правую колонку следует расположить справа.

```
#col1 {float:left}
#col2 {float:left}
```

Результат:

Рис. 2. Две колонки рядом при относительном позиционировании

Рис. 3. Две колонки с зазором при относительном позиционировании

MARGIN-LEFT И MARGIN-RIGHT

Для того, чтобы расположить колонки по центру, необходимо отцентрировать сам контейнер:

```
.container{
        margin-left:auto;
            margin-right:auto;
            width:700px;
}
```

Весь код стилей:

```
.container{
            margin-left:auto;
            margin-right:auto;
            width:700px;
}
/* Общее для колонок */
#col1, #col2{
   width: 335px; /* Ширина колонок */
   border: 1px solid black; /* Рамка вокруг слоя */
   padding: 5px; /* Поля вокруг текста */
}
/* Отдельно для колонок */
  #col1 {float:left }
  #col2 {float:right }
/* Цвет фона каждого слоя */
  #col1 { background: #EBE0C5; }
  #col2 { background: #BBE1C4; }
```

Для трех колонок

Расположить три колонки рядом не получится, произойдет смещение. Поэтому необходимо два слоя также оставить в контейнере:

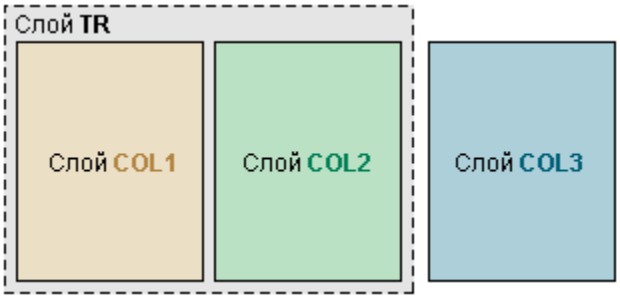


Рис. 4. Три колонки при относительном позиционировании

```
<div class="container">
   <div id="col1">
                Колонка 1
        </div>
   <div id="col2">
                Колонка 2
   </div>
</div>
   <div id="col3">
                Колонка 3
   </div>
```

POSITION

Для контейнера можно установить абсолютное позиционирование, а для третьей колонки - относительное позиционирование CSS. Для двух других колонок - в зависимости от макета и взаиморасположения слоев.

```
  .container{
        position:absolute;
}
  #col1 {position: absolute}
  #col2 {position: relative}
  #col3 {position: relative}
```

WIDTH

Необходимо задать ширину для всех трех колонок. Для контейнера ширина не нужна.

```
#col1, #col2, #col3 {
   width: 200px;
}
```

LEFT

Для второй третьей колонок необходимо задать расстояние от левого края браузера или контейнера:

```
  #col2 {
                position: relative;
                left:205px;
  }
  #col3 {
                position: relative;
                left: 410px;
  }
```

Весь код стилей:

```
 .container{
        position:absolute;
}
/* Общее для колонок */
#col1, #col2, #col3 {
   width: 200px; /* Ширина колонок */
   border: 1px solid black; /* Рамка вокруг слоя */
   padding: 5px; /* Поля вокруг текста */
}
/* Отдельно для колонок */
  #col1 {position: absolute}
  #col2 {
                position: relative;
                left:205px;
  }
  #col3 {
                position: relative;
                left: 410px;
  }
/* Цвет фона каждого слоя */
  #col1 { background: #EBE0C5; }
  #col2 { background: #BBE1C4; }
  #col3 { background: #ADD0D9; }
```

Результат:

Рис. 5. Три колонки при относительном позиционировании

Резиновый дизайн и необходимые свойства CSS

Резиновый дизайн сайта означает, что **при изменении размеров окна браузера все слои масштабируются согласно текущей ширине**. Это и есть основное преимущество резинового дизайна: независимо от размеров экрана, вся его область будет занята, при этом не возникают лишние полосы прокрутки при использовании «маленького» монитора.

«Резиновость» достигается за счет того, что определенные слои макета не имеют фиксированную ширину, приспосабливая свои размеры под размеры окна браузера.

Минусом данного типа дизайна являются «разногласия» между браузерами, **необходимость применять «хитрости» css-стилей**, для того чтобы достичь необходимой кроссбраузерности.

Две колонки в резиновой дизайне

Дизайн CSS две колонки, пожалуй, самый популярный. Создать страницу с таким расположением колонок можно двумя способами, рассмотрим каждый отдельно.

СВОЙСТВО FLOAT ДЛЯ СОЗДАНИЯ ЭФФЕКТА ПЛАВАЮЩЕГО ЭЛЕМЕНТА

Рассмотрим пример, в котором присутствуют следующие блоки, изображенные на *рис. 1*: шапка (бордовый), меню (светло-серый), контент (белый), подвал (темно-серый).

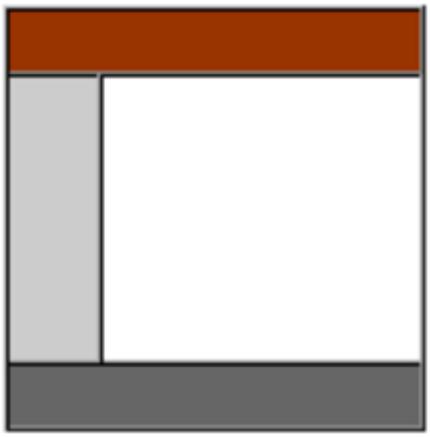


Рис. 1. Резиновый дизайн сайта в две колонки

Левую колонку выполним с фиксированным размером, а правая колонка будет занимать остальную ширину окна, что и обеспечит «резиновость» дизайна.

Основные свойства

1. «Разбиваем» все основные элементы страницы на блоки следующим образом:

```
<div id="top">Заголовок сайта</div>
 <div id="menu">
  Ссылка 1<br>
  Ссылка 2<br>
  Ссылка 3<br>
  Ссылка 4<br>
 </div>
 <div id="content">
  <h1>Заголовок</h1>
  <p>Контент. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis te feugifacilisi.
 </p>
 </div>
 <div id="bottom">
  Подвал
 </div>
```

2. Для левой колонки задаем свойства, обязательные при использовании резинового дизайна: float и width

1 вариант:

```
#menu{
   float: left;
   width: 20%;
}
```

2 вариант:

```
#menu{
   float: left;
   width: 200px;
}
```

3. Для правой колонки обязательным является свойство, определяющее внешний отступ от левого края окна браузера (margin-left). Этот отступ должен соответствовать ширине левой колонки + 1px или 1%:

1 вариант:

```
#content{
   margin-left: 21%;
}
```

2 вариант:

```
#content{
   margin-left: 210px;
}
```

Все остальные свойства зависят от оформления сайта.

Весь код CSS будет выглядеть так:

```
#top { /* Шапка с заголовком страницы */
   background: #e3e8cc; /* Цвет фона */
   border: solid 1px black; /* Параметры рамки  */
   padding: 10px 0 10px 10px; /* Поля вокруг текста */
   margin-bottom: 5px; /* Расстояние между шапкой и колонками */
   font-size: 24px; /* Размер шрифта */
   font-weight: bold; /* Жирное начертание */
}
#menu { /* Меню сайта */
   width: 200px; /* Ширина меню */
   background: #e3e8cc; /* Цвет фона */
   border: solid 1px black; /* Параметры рамки */
   float: left; /* Состыковка со слоем контента по горизонтали */
   padding: 3px; /* Внутренние поля */
}
#content { /* Основное содержание */
   background: #e3e8cc; /* Цвет фона */
   border: solid 1px black; /* Параметры рамки */
   margin-left: 210px; /* Отступ слева с учетом ширины левой колонки */
   margin-bottom: 5px /* Отступ снизу */
}
#content p{
         padding: 3px; /* Внутренние поля контента*/
}
#bottom { /* Подвал */
   background: #e3e8cc; /* Цвет фона */
```

```
    border: solid 1px black; /* Параметры рамки */
    padding: 3px; /* Поля вокруг текста */
}
```

Результат:

Рис. 2. Левая колонка с фиксированной шириной в резиновом дизайне

Рассмотрим пример, когда фиксированной шириной обладает не левая, а правая колонка.

Правая колонка с фиксированной шириной: основные свойства

Код html-структуры остается прежним, меняются лишь CSS-свойства.

1. Для левой колонки задаем свойства, обязательные при использовании резинового дизайна: float и width

1 вариант:

```
#menu{
    float: right;
    width: 20%;
}
```

2 вариант:

```
#menu{
    float: right;
    width: 200px;
}
```

2. Для правой колонки задаем свойство, определяющее внешний отступ от правого края окна браузера (margin-right). Этот отступ должен соответствовать ширине левой колонки + 1px или 1%:

1 вариант:

```
#content{
    margin-right: 21%;
}
```

2 вариант:

```
#content{
    margin-right: 210px;
}
```

Результат:

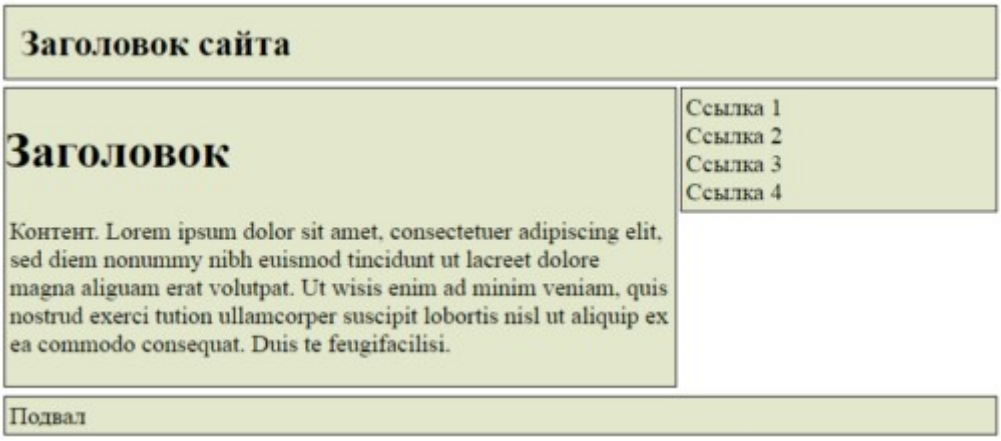


Рис. 3. Правая колонка с фиксированной шириной в резиновом дизайне

ПРИМЕНЕНИЕ ПОЗИЦИОНИРОВАНИЯ В РЕЗИНОВОМ ДИЗАЙНЕ

При использовании данного варианта дизайна для левой или правой колонки устанавливается абсолютное позиционирование с заданием координат.

Например:

Левая колонка:

position:absolute;
width: ...px;
left: ...px;
top: ...px;

Правая колонка:

margin-left: ...px; */* отступ слева, равный ширине первой колонки */*

«Резиновость» достигается в данном случае за счет правой колонки, у которой нет определенной ширины.

Пример: Создать веб-страницу в формате резинового дизайна. Для левой колонки задать абсолютное позиционирование с координатами от левого и верхнего края окна браузера

Выполнение:

1. Расположим основные html-элементы страницы следующим образом:

```
<div id="col1">
 <p>
  Ссылка 1<br>
  Ссылка 2<br>
  Ссылка 3<br>
  Ссылка 4<br>
  </p>
 </div>
 <div id="col2">
  <p>Контент. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy
  nibh euismod tincidunt ut lacreet dolore magna aliguam erat
  volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper
  suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis te feugifacilisi.
 </p>
```

```
  </div>
```
2. Для левой колонки задаем свойства, обязательные при использовании позиционирования в резиновом дизайне - position, width, left и top:
```
#col1 { /* Левая колонка */
    position: absolute; /* Абсолютное позиционирование */
    width: 190px; /* Ширина блока*/
    left: 0px; /* Координата от левого края окна */
    top: 0px; /* Координата от верхнего края окна */
    background: #800000; /* Цвет фона левой колонки */
    padding: 5px; /* Поля вокруг текста */
}
```
3. Задаем свойства для правой колонки, обязательные при использовании позиционирования в резиновом дизайне - margin-left:
```
#col2 { /* Правая колонка */
    margin-left: 200px; /* Отступ слева=ширине левой колонки + padding */
    background: #e0e0e0; /* Цвет фона правой колонки */
    padding: 5px; /* Поля вокруг текста */
}
```
4. Удаляем лишние отступы браузера по умолчанию:
```
body {
    padding: 0; /* Отступы для браузера Opera */
    margin: 0; /* Обнуляем значения отступов на веб-странице */
   }
```
Результат:

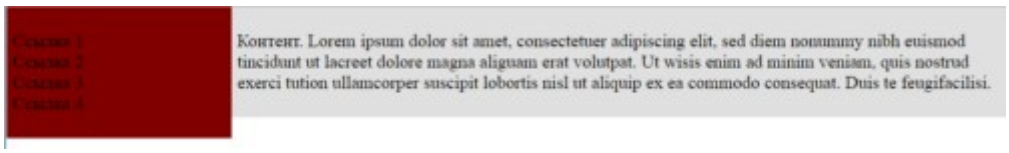


Рис. 4. Резиновый дизайн сайта с позиционированием

Одинаковая высота колонок

На *рис. 4* результат показывает, что колонки имеют неодинаковую высоту. Для того, чтобы колонки приобрели одинаковую высоту есть несколько вариантов использования дополнительных свойств, одним из которых является добавление границы (border).

1. Обе колонки заключаются в слой-контейнер:
```
<div id="container">
  <div id="col1">
  <p>
   Ссылка 1<br>
   Ссылка 2<br>
   Ссылка 3<br>
   </p>
  </div>
  <div id="col2">
   <p>Контент. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem
```

```
nonummy
   nibh euismod tincidunt ut lacreet dolore magna aliguam erat
   volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper
   suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis te feugifacilisi.
  </p>
  </div>
</div>
```

2. Для левой колонки устанавливаем ширину, абсолютное позиционирование, координаты от верхнего левого угла браузера, внешний отступ слева отрицательного значения (!), равный ширине блока, и внутреннее поле слева, равное также ширине блока:

```
#col1 { /* Левая колонка */
   position: absolute; /* Абсолютное позиционирование */
   width: 200px; /* Ширина слоя */
   left: 0px; /* Положение от левого края окна */
   top: 0px; /* Положение от верхнего края окна */
   margin-left: -200px; /* внешний левый отступ отрицательного значения */
   padding-left: 200px; /* внутренне левое поле */
   background: #800000; /* Цвет фона левой колонки */
}
```

3. Для правой колонки никакие свойства позиционирования не устанавливаются, но можно задать свойства для оформления колонки:

```
#col2 { /* Правая колонка */

   background: #e0e0e0; /* Цвет фона правой колонки */
   padding: 5px; /* Поля вокруг текста */
}
```

4. Для слоя контейнера необходимо задать границу, ширина которой должна совпадать с шириной левой колонки, а цвет - с цветом левой колонки:

```
#container {
   border-left: 200px solid #800000;
}
```

Результат:

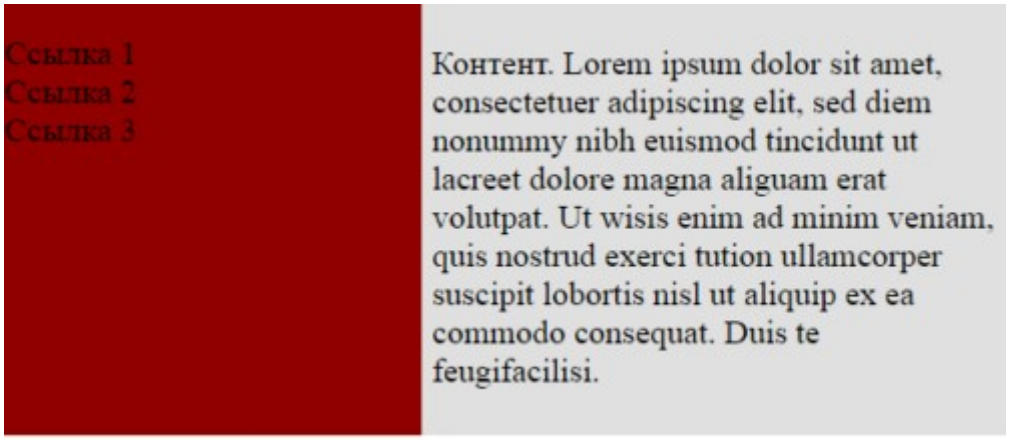


Рис. 6. Колонки одинаковой высоты

Вопросы для самоконтроля

1. Выбор хостинга.
2. Подбор доменного имени.
3. Загрузка сайта на сервер.

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