ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Информационные технологии»

Методические указания к лабораторным работам

по дисциплине

«Разработка пользовательского интерфейса»

ЛАБОРАТОРНАЯ РАБОТА №1

Создание структуры проекта. Выбор текстового редактора. Базовая разметка страницы

Создание структуры проекта

Создание любого проекта начинается с создания базовой структуры папок.

Эта структура является однотипной:

В удобном для вас месте создайте папку с вашим будущем проектом. В корне этой папке создайте папку “static” (здесь будут хранится статические файлы: изображения, стили) и документ “index.html” (это базовый файл, с которым нам доведётся работать). В папке “static” создайте папку “css” и “images”. Последним штрихом добавляем файл “style.css” в папку “css”.

Примечание: Кроме того, более динамические сайты используют для интерактивных элементов, которые создаются с помощью JavaScript. И файлы со скриптами хранятся еще в папке “js”, которая находится в папке “static”.

Выбор текстового редактора

HTML - страницу можно сверстать, используя Блокнот или Notepad++, но на данный момент существуют редакторы кода, имеющие огромный функционал в виде подсветки синтаксиса, открытие целых директорий, множества файлов, разбитие на несколько окон и т.д. Одни из самых популярных это SublimeText, Brackets, Atom. Весь код, представленный в дальнейшем в описании лабораторных работ будет оформлен в SublimeText3.

Базовая разметка страницы

Откройте файл “index.html” и перепишите вручную текст с картинки. Всё, кроме того что заключено в теги вида <!-- какой-то комментарий -->(текст в данных тегах не будет отображаться в веб браузере). И посмотрите, что в браузере получилось.

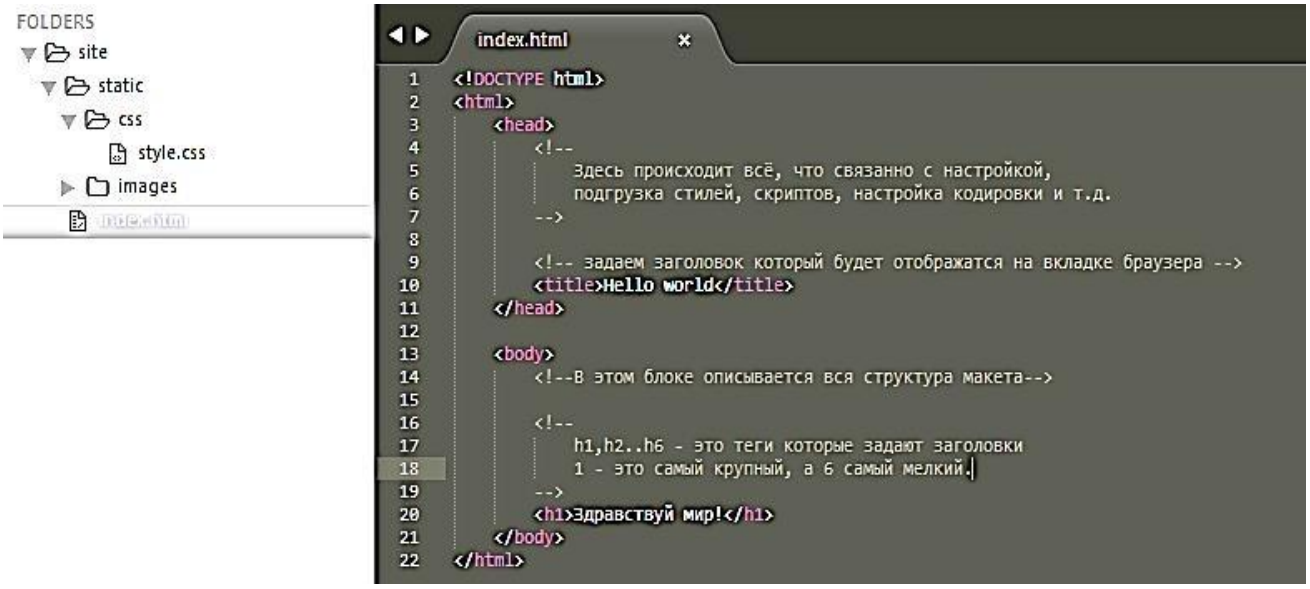


Рисунок 1 – Пример кода

В результате вы написали базовое приложение типа “Hello world”. Самым главным, что стоит запомнить из этого примера, что теги - это элементы языка вида <имя тега> и они обязательно должны закрываться: </имя тега>. В дальнейшем содержимое тегов может форматироваться. Указанная структура документа является базовой и обязательной.

ЛАБОРАТОРНАЯ РАБОТА №2

Дальнейшую работу продолжим в документе, созданном в лабораторной работе №1.

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello world</title>
        <!--Для того, чтобы браузеры понимали кирилические символы введите:-->
        <meta charset="utf-8">
    </head>
    <!--
                    Примечание
    теги можно писать как полностью большими так и полностью маленькими буквами,
    разницы никакой не будет. Тоесть <H1> тоже самаое, что и <h1>
    -->
    <body>
        <!--div - это обычный контейнер который группирует элементы в единое целое -->
        <!-- к div-ам мы еще еще более подробнее в дальнейшем -->
        <div align="center">
            <h1>Здравствуй мир!</h1>

            <!-- -->
            <br>

            <h2>Сегодня мы вспомним химию</h2>
            <!--разбивает текст на параграфы -->
            <p>
                <b>Сахар</b> – <u>бытовое название сахарозы</u> (C<sub>12</sub>H<sub>22</sub>O<sub>11</sub>).
                <i>Тростниковый</i> и <i>свекловичный сахар</i> <sup>(сахарный песок, рафинад)</sup>

                <!--
                    B - задает полужирный шрифт
                    U - задает нижнее подчеркивание
                    I - курсивный шрифт
                    SUB - подстрочный индекс
                    SUP - подстрочный индекс
                -->
            </p>

            <p>
                является важным пищевым продуктом. <font color="red" size="4">Обычный</font> сахар относится к
                <strike>углеводам</strike>,
```

Рисунок 2 – Работа с различными тэгами

```
40                  <!--
41                      strike - зачеркивание
42                      font - Тег для задание свойств шрифта
43
44                      На практике задание свойств текста через теги задается с помощью CSS,
45                      но для базового ознакомления будет полезным ознакомится с этим. В следующей
46                      лабораторной работе мы проделаем те же самые вещи с помощью CSS
47                  -->
48                  <!-- BR - перевод строки -->
49                  <!--
50                      Вы уже обратили, что водном из тегов была такая вещь, как align="center",
51                      это мы назначали тегу напрямую стиль, который задает центральное позиционирование,
52                      всему что находится в этом теге, кроме того, можно задать позиционирование: left(слева), right(
                        справа), top(сверху), bottom(снизу)
53                  -->
54
55                  <!--
56                      Font- это тег, позволяющий форматировать нам текст, на этом примере мы задали цвет через
57                      свойство "color" и размер шрифта "size", кроме того цвет можно задавать не только
                        зарезервированным словом, а его кодом, к примеру #000000  задаст черный цвет, подобрать код
                        цвета через палитру вы можете на сайте "getcolor.ru"
58                  -->
59              </p>
60
61              <p>
62                  которые считаются ценными питательными веществами, обеспечивающими организм необходимой энергией.
63              </p>
64          </div>
65      </body>
66  </html>
```

Рисунок 3 - Работа с различными тэгами

В комментариях вы можете увидеть описание каждого нового тега. Открывая страницув браузере, вы получите такой результат:

Здравствуй мир!

Сегодня мы вспомним химию

Сахар — бытовое название сахарозы ($C_{12}H_{22}O_{11}$). *Тростниковый* и *свекловичный сахар* (сахарный песок, рафинад)

является важным пищевым продуктом. Обычный сахар относится к ~~углеводам~~,

которые считаются ценными питательными веществами, обеспечивающими организм необходимой энергией.

Рисунок 4 – Результат работы кода

ЛАБОРАТОРНАЯ РАБОТА №3

Объявление стилей. Базовые свойства стилей стилей.

Рассмотрим форматирование текста в HTML-документе с помощью таблицы каскадных стилей (далее CSS - CascalSheetStyles).
Существуют различные способы объявление стилей: в самих тегах, в теге <style></style>, и в отдельном файле. Мы же рассмотрим последние два метода.

Для начала создайте новый документ «index03.html» и задайте ему базовую структуру,как в лабораторной работе № 1 и новое содержимое блока <style> как в примере:

index.html × index03.html ×

```
<!DOCTYPE html>
<html>
<head>
    <title>add style</title>
    <style type="text/css">
        h2{
            color: red;
        }
    </style>
    <!--
        Что мы сделали? Сы сказали нашей странице, что всем h2 присвоить крассный цвет.
        Больше об оформлении CSS мы поговорим далее.
        А пока стоит усвоить такой принцип задание свойств:

        то-чему-зажается-свойство{
            свойство: занчение;
            ...
            свойство: значение;
        }
    -->
</head>
<body>
    <h2>Давайте что-то новенькое напишем</h2>
</body>
</html>
```

Рисунок 5 - Объявление стилей

Чтобы ещё меньше нагружать наш документ стилями, мы подключим внешние стили.

Для этого вместо блока <style> вставьте эту строчку:

```
4       <link rel="stylesheet" type="text/css" href="static/css/style.css">
5       <!--На этом этапе мы указываем путь к нашему будующему файлу стилей,
6       который находится в static -> css -> style.css-->
```

Рисунок 6 – Подключение внешнего стиля

Затем открываем файл “style.css” из “static/css” и добавляем следующий код:

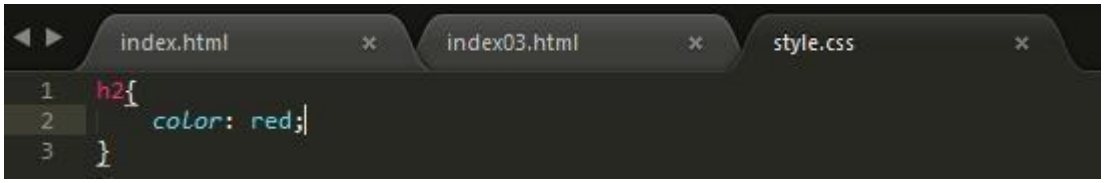


Рисунок 7 - Изменение цвета заголовка

Обновив страницу, вы обнаружите, что заголовок так и останется красным. Так идолжно быть, так как вы логику описания стилей перенесли в отдельный файл. С объявлением мы разобрались, но что если мы не хотим делать так, чтобы все элементы h2 принимали красный цвет, или даже чтобы только один элемент имел собственный стиль? К нам на помощь приходят классы и идентификаторы.

Как работают классы?

Для того чтобы тегу присвоить класс в самом теге вам надо написать class=”именна_классов_через_пробел” (имена должны писаться латинскими символами и использовать из сторонних символов только дефис и нижнее подчеркивание).

В самом же файле стилей, стилю задаются свойства таким образом

.имя_класса {свойства как обычно}. С идентификатором(id) дела обстоят почти также. В теге объявляется через id=”имя_id” а в стилях #имя_id{свойства}.

А в чем же собственно разница? Разница в том, что id это уникальное имя, которое следует объявлять только лишь раз один раз на странице, а класс сколько угодно раз.

Для большего понимания повторите следующий код и посмотрите результат:

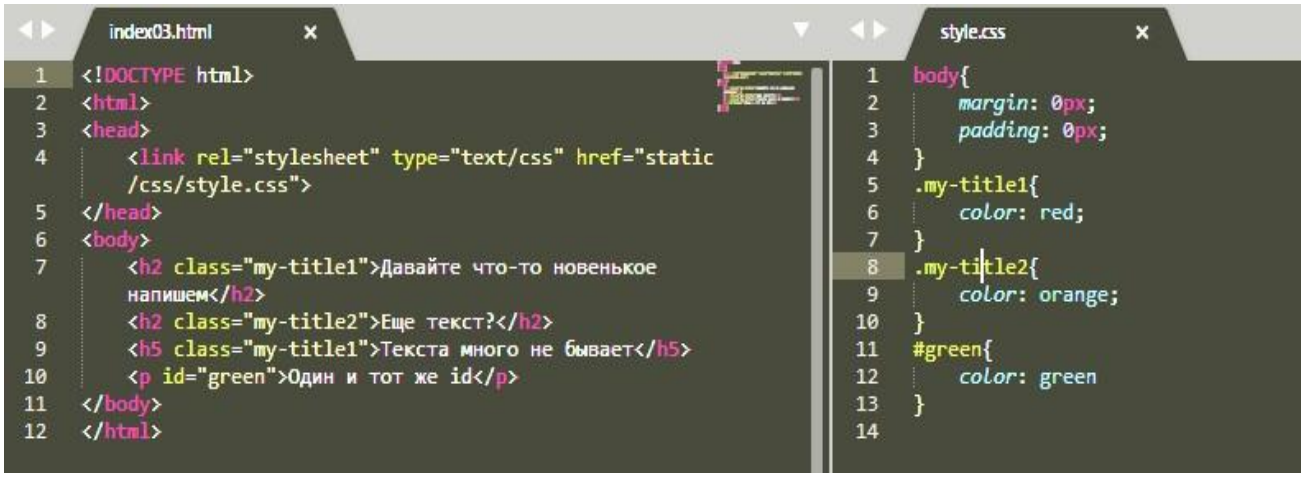


Рисунок 8 – Пример

Полученный результат для вас должен стать очевидным и понятным.

P.S. Если в дальнейшем css вызывает трудности, то можно воспользоваться сайтом http://htmlbook.ru/css. Здесь можно найти как описание тегов с подробным описанием поддержки тегов в различных браузерах, так и различные миниуроки.

ЛАБОРАТОРНАЯ РАБОТА №4

Списки. Типы списков. Ссылки. Hover-эффекты.

Для отображения различных элементов в связанной структуре используют такой элемент, как списки. На следующем рисунке представлен код по созданию простой шапки сайтас использованием немаркированных списков.

Прежде чем повторять код, примите к сведению, что у элементов <li> (элемент самого списка), есть html-атрибут «type». Этот атрибут может принимать различные значения, например «disc» (элементы списка будут иметь круглый черный маркер), «circle» (маркеры будут в виде кольца) и «square»(квадратные маркеры).

index04.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>lab-04</title>
    <link rel="stylesheet" type="text/css" href="static/css/
    style.css">
</head>
<body>
    <!--nav - наш тег для шапки сайта-->
    <nav>
        <ul class="list">
            <!--ul - контейнер элементов списка -->
            <!--li - эллемент списка -->
            <li>номер один</li>
            <li>номер два</li>
            <li>номер три</li>
        </ul>
    </nav>

    <!--Блок с контентом-->
    <content>
        <!-- ol - контейнер нумерованого списка-->
        <ol>
            <li>Кот</li>
            <li>Собака</li>
            <li>Попугай</li>
        </ol>
    </content>
</body>
</html>
```

style.css

```css
body{
    margin: 0px;
    padding: 0px;
}
nav{
    height: 60px;
    width: 100%;
    background-color: #FEA610;/*оранжеаый фон*/
}
/*задаем стили контейнеру с классом list*/
ul.list{
    position: absolute;
    margin-left: calc((100% - 800px)/2);
    width: 800px;
    height: 30px;
    list-style-type: none;/*убираем маркировку li-элементов*/

}

ul.list li{
    background-color: grey;/*фон элементов серый*/
    display: inline;/*li - элементы располагаются в ряд*/
}
```

Рисунок 9 – Списки

В результате вы будете иметь такой результат:

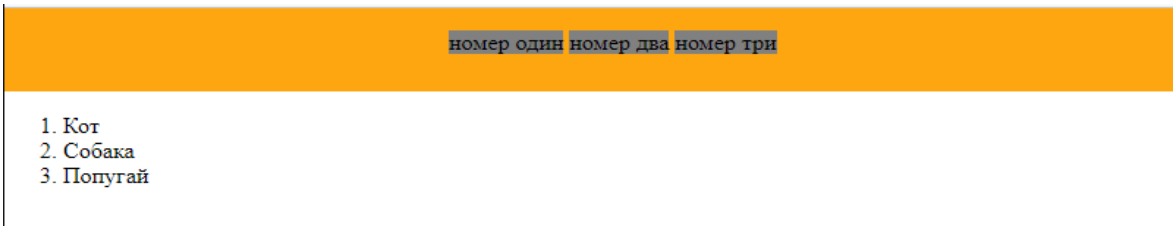


Рисунок 10 - Пример работы кода

Давайте теперь создадим на элементах списка ссылки и добавим к ней простую анимацию.

Ссылки описывает тег <a>.

Основной атрибут этого тега является «href», href – это

сылка на страницу. Ссылки бывают такого типа:

```html
<a href="имя файла.html">Текст</a>
<a href="папки/где/лежит/страница.html">ссылки</a>
<a href="http://портал.com"></a>
```

Рисунок 11 – Ссылки

Кроме того, в ссылку можно оборачивать любой объект (тег), что-либо: текст, изображение, блок и т.д.

Стандартно HTML задает тексту в ссылке подчеркивание и голубой цвет шрифта и изменение цвета шрифта ссылки на фиолетовый, после перехода по этой ссылке. Но с помощью css или тегов html атрибутов можно изменить эти стандартные свойства.

Кроме того, в CSS есть такое понятие, как псевдокласс. Псевдокласс в CSS - это ключевое слово, добавленное к селектору, которое определяет его особое состояние.

Например, :hover применит стиль, когда пользователь наводит курсор на элемент, указанныйселектором. Как раз этот псевдокласс и применим к ссылке и посмотрим, что получится.

```html
<!DOCTYPE html>
<html>
<head>
    <title>lab-04</title>
    <link rel="stylesheet" type="text/css" href="static/css/
    style.css">
</head>
<body>

    <nav>
        <ul class="list">
            <a href="index.html">
                <li>Первая лабораторная работа</li>
            </a>
            <a href="https://vk.com">
                <li>Переход на вконтакте</li>
            </a>
            <li>без ссылки</li>
        </ul>
    </nav>

    <content>
        <ol>
            <li>Кот</li>
            <li>Собака</li>
            <li>Попугай</li>
        </ol>
    </content>

</body>
</html>
```

```css
body{
    margin: 0px;
    padding: 0px;
    font-size: 14px;/*задаем размер шрифта для всего документа*/
}
nav{
    height: 60px;
    width: 100%;
    background-color: #FEA610;
}
ul.list{
    position: absolute;
    margin-left: calc((100% - 800px)/2);
    width: 800px;
    height: 30px;
    list-style-type: none;

}
ul.list li{
    margin-left: 15px;
    display: inline;
}
a{
    color: black;
    text-decoration: none;/*отменяем нижнее подчеркивание*/
}
a:hover{
    color: blue;
    font-size: 15px
}
```

Рисунок 12 - Псевдокласс

Ниже представлен результат написанного вами кода:

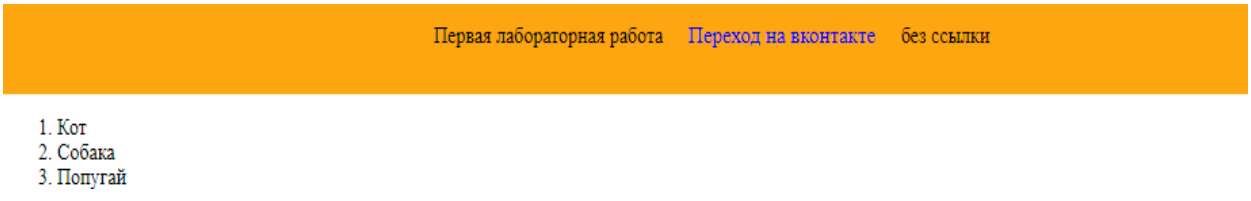


Рисунок 13 – Результат

Пример совсем не совершенен, но позволяет опробовать новые изученные свойства.

ЛАБОРАТОРНАЯ РАБОТА №5

Блоки. Изображения. Позиционирование. Отступы.

Блок (атрибут), это сущность, которая напоминает контейнер (коробку), которая хранит в себе различные элементы с другими свойствами (размер, цвет и т.д.). Вложенные элементы в этот контейнер наследуют многие из свойств (такие, как шрифт, цвет, текстовое форматирование). В CSS есть свойства, которые применяются в основном к блоку и отвечают за способы расположения блоков.

Следующий пример покажет, как создать плитку цветных блоков, которой мы воспользуемся в дальнейшем.

index05.html

```
<!DOCTYPE html>
<html>
<head>
    <title>lab-5</title>
    <link rel="stylesheet" type="text/css" href="static/css/style.css">
</head>
<body>
    <div class="centr-cont">
        <div class="block first"></div>
        <div class="block second"></div>
        <div class="block third"></div>
        <div class="block fourth"></div>
    </div>
</body>
</html>
```

style.css

```
body{
    margin: 0px;
    padding: 0px;
    font-size: 14px;/*задаем размер шрифта для всего документа*/
}
.centr-cont{
    height: 650px;
    width: 750px;
    background-color: #E7E4E4;
    margin-left: calc((100% - 750px)/2);
    /*операция калс выполняет мат операции,
    очень важно между - и + с двух сторон ставить пробелы*/
}
.block{
    position: relative;
    height: 300px;
    width: 350px;
    display: inline-block;
    /*элементыэтого класса будут следовать в линии друг за другом,
    пока им позволяеет ширина окна браузера*/
}
.first{
    background-color: #FAB933;
}
.second{
    background-color: #FC625F;
    margin-left: 46px;
}
.third{
    background-color: #5FF6FC;
    margin-top: 46px;
}
.fourth{
    margin-left: 46px;
    margin-top: 46px;
    background-color: #69FC5F;
}
```

Рисунок 14 – Блоки

Результат:

Рисунок 15 - Результат

Теория

```
/*
    Свойство padding позволяет задать величину поля
сразу для всех сторон элемента или определить ее только для указанных сторон.
Применяется для текста. И эти отступи применятся относительно того блока,
в котором он находится.

        Пример:
    padding-left: 25px;
    padding: 10px 30px;
    padding: 20px;

    Свойство margin Устанавливает величину отступа от каждого края элемента.
Отступом является пространство от границы текущего элемента до внутренней
границы его родительского элемента.

        Пример:
    padding-left: 25px;
    padding: 10px 30px;
    padding: 20px;

    Кроме того отступы можно задавать не только в пикселях(px),
но и процентах(%).
*/
```

Позиционирование (css свойство- position)

Position - устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице.

Способ обьявления: position: absolute | fixed | relative | static | inherit

absolute

Указывает, что элемент абсолютно позиционирован, при этом другие элементы отображаются на веб-странице словно абсолютно позиционированного элемента и нет. Положение элемента задается свойствами left, top, right и bottom, также на положение влияет значение свойства position родительского элемента. Так, если у родителя значение position установлено как static или родителя нет, то отсчет координат ведется от края окна браузера.

Если у родителя значение position задано как fixed, relative или absolute, то отсчет координат ведется от края родительского элемента.

fixed

По своему действию это значение близко к absolute, но в отличие от него привязывается к указанной свойствами left, top, right и bottom точке на экране и не меняет своего положения при прокрутке веб-страницы. Браузер Firefox вообще не отображает полосы прокрутки, если положение элемента задано фиксированным, и оно не помещается целиком в окно брау- зера. В браузере Opera хотя и показываются полосы прокрутки, но они никак не влияют на позицию элемента.

relative

Положение элемента устанавливается относительно его исходного места. Добавление свойств left, top, right и bottom изменяет позицию элемента и сдвигает его в ту или иную сторону от первоначального расположения.

static

Элементы отображаются как обычно. Использование свойств left, top, right и bottom неприводит к каким-либо результатам.

inherit

Наследует значение родителя.

Чтобы сделать еще красивее, добавим несколько изображений. Изображения очень похожи на ссылки, но имеют некоторые различия. Тег - <img> (не

закрывающийся), атрибут определяющий путь к файлу – src=”путь”. Путь указывается также, как и у ссылки, только является важным разрешение файла (например, .jpg)

Скачаем 4 изображения из интернета. Можно воспользоваться гугл, яндекс картинками. Или же сервисом со стоковыми изображениями, к примеру https://unsplash.com. Стоит иметь ввиду, что изображения с таких сервисов имеют огромный размер (до 15мб за изображение) и для быстрого отображения таких изображений на странице следует сжать изображение до некоторого минимума.

Скачанные изображения поместим в раннее созданную папку «images», а для большей простоты изображения переименуем в по образцу «01.расширение_изображения».

index05.html

```
<!DOCTYPE html>
<html>
<head>
    <title>lab-5</title>
    <link rel="stylesheet" type="text/css" href="static
    /css/style.css">
</head>
<body>
    <div class="centr-cont">
        <div class="block first">
            <img src="static/images/01.jpg">
        </div>
        <div class="block second">
            <img src="static/images/02.jpg">
        </div>
        <div class="block third">
            <img src="static/images/03.jpg">
        </div>
        <div class="block fourth">
            <img src="static/images/04.jpg">
        </div>
    </div>
</body>
</html>
```

style.css

```
body{
    margin: 0px;
    padding: 0px;
}
img{
    height: 300px;
    width: 350px;
}
.centr-cont{
    height: 650px;
    width: 750px;
    background-color: #E7E4E4;
    margin-left: calc((100% - 750px)/2);
}
.block{
    position: relative;
    height: 300px;
    width: 350px;
    display: inline-block;
}
.second{
    margin-left: 46px;
}
.third{
    margin-top: 46px;
}
.fourth{
    margin-left: 46px;
    margin-top: 46px;
}
```

Рисунок 16 - Работа с изображениями

Результат:

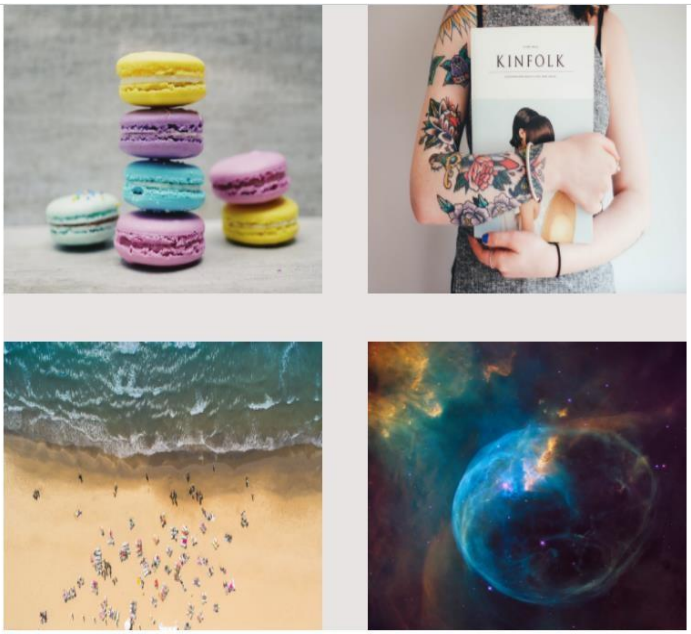


Рисунок 17 - Результат кода

П.С: Существует способ задания фона блоку через css-свойство «background-img», дляознакомления рекомендуем изучить его самостоятельно и опробовать на одном из блоков.

ЛАБОРАТОРНАЯ РАБОТА №6

Таблицы.

Прежде чем приступить к выполнению лабораторной создайте файл «index0?.html» и файл стилей для него «style0?.css»

1. Создание простейшей таблицы

Создание таблицы начинается с тега **<table></table>**, далее, внутри него располагается все содержимое, а именно строки **<tr>** и уже внутри строк ячейки **<td>.** Чтобы понять, как всеэто работает обратите внимание на пример ниже:

Примечание: заголовочную ячейку можно создать тегом <th>, текст в такой ячейке располагается посередине и выделяется жирным шрифтом.

1. Для начала зададим базовые стили

```
/* Задаем рамку таблице */
table{
    border:1px solid black;
}

/* Задаем рамку ячейке */
td{
    border:1px solid black;
}
```

Рисунок 18 – Базовый стиль

не забудьте подключить файл стилей к вашему HTML файлу.

2. Создадим простейшую таблицу и посмотрим на результат в браузере

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style07.css">
</head>
<body>
    <table>
        <tr>
            <td>Университет</td>
            <td>Кол-во студентов</td>
        </tr>
        <tr>
            <td>ДГТУ</td>
            <td>46 000</td>
        </tr>
        <tr>
            <td>ЮФУ</td>
            <td>33125</td>
        </tr>
        <tr>
            <td>РИНХ</td>
            <td>21 636</td>
        </tr>
    </table>
</body>
</html>
```

Рисунок 19 - Создание таблицы

Университет	Кол-во студентов
ДГТУ	46 000
ЮФУ	33125
РИНХ	21 636

Рисунок 20 - Результат

Задание 1

Дополнить таблицу как показано ниже:

Университет	Кол-во студентов	Год основания	Адрес
ДГТУ	46 000	1930	пл.Гагарина,1
ЮФУ	33125	2006	ул.Б.Садовая,105
РИНХ	21 636	1931	ул.Б.Садовая,69

Самостоятельно освойте свойство **border-collapse**, в выполнении задания оно обязательно понадобится.

2. Работа с рамками, отступами внутри и между ячейками, ячейки-заголовки

Иногда бывают ситуации, когда нужно чтобы отображалась только нижняя рамка ячейки и т.д, такие эффекты достигаются за счет свойств, представленных ниже:

border-right, border-left, border-top, border-bottom

Примечание: для того чтобы убрать рамку в значении свойства напишите **none**

Задание 2

В созданной таблице из Задания 1 оставьте рамки только справа и слева как на примере
ниже:

Университет	Кол-во студентов	Год основания	Адрес
ДГТУ	46 000	1930	пл.Гагарина,1
ЮФУ	33125	2006	ул.Б.Садовая,105
РИНХ	21 636	1931	ул.Б.Садовая,69

Отступы внутри ячеек задаются с помощью свойства **padding:**

padding: 5px - одно значение задает отступ всем сторонам,
padding: 5px 10px - первое значение задает отступ сверху-снизу, второесправа-слева.
padding: 5px 10px 5px - первое значение задает отступ сверху, второеодновременно справа-слева, третье снизу.

padding: 5px 10px 5px 10px - первое сверху, второе справа, третье снизу,четвертое слева.

Задание 3

Задайте ячейкам отступы сверху-снизу (5px), справа-слева (10px) как на примере ниже:

Университет	Кол-во студентов	Год основания	Адрес
ДГТУ	46 000	1930	пл.Гагарина,1
ЮФУ	33125	2006	ул.Б.Садовая,105
РИНХ	21 636	1931	ул.Б.Садовая,69

Задание 4

1. Замените тег **<td>** в категориях на **<th>.**
2. В стилях задайте тегу <th> рамку и внутренний отступ (10px)

Университет	**Кол-во студентов**	**Год основания**	**Адрес**
ДГТУ	46 000	1930	пл.Гагарина,1
ЮФУ	33125	2006	ул.Б.Садовая,105
РИНХ	21 636	1931	ул.Б.Садовая,69

3. Объединение ячеек в строках-столбцах

Для объединения двух и более ячеек используют атрибуты colspan и rowspan для тега

<td>

colspan — устанавливает число объединяемых ячеек по горизонталиrowspan — аналогично, только по вертикали

Следуя инструкциям, создайте таблицу:

Браузер	Посещения	
	Количество	В процентах
Mozila Firefox	163	59%
Google Chrome	78	28%
Safari	35	13%

1. Задайте базовую структуру таблицы и стили

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style07.css">
</head>
<body>
    <table>
        <tr>
            <th>Браузер</th>
            <th>Посещения</th>
        </tr>
        <tr>
            <th>Количество</th>
            <th>В процентах</th>
        </tr>
        <tr>
            <td>Mozila Firefox</td>
            <td>163</td>
            <td>59%</td>
        </tr>
        <tr>
            <td>Google Chrome</td>
            <td>78</td>
            <td>28%</td>
        </tr>
        <tr>
            <td>Safari</td>
            <td>35</td>
            <td>13%</td>
        </tr>
    </table>
</body>
</html>
```

```
table{
    border:1px solid black;
    border-collapse: collapse;
}

td{
    border:1px solid black;
    padding: 10px;
}

th{
    border:1px solid black;
    padding:10px;
}
```

2. Начинаем работать с объединением ячеек

Первый тег <th> занимает две ячейки по горизонтали и две ячейки по вертикали, поэтому задаем:

<th colspan = “2“ rowspan = “2“></th>

Второй занимает две ячейки по горизонтали:

<th colspan = “2“></th>

Тег <th> с названиями браузеров точно так же как и первый занимает две ячейки по горизонтали:

<th colspan = “2“></th>

Проделав все действия выше, вы должны получить такой код и результат изображенный выше:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style07.css">
</head>
<body>
    <table>
        <tr>
            <th colspan="2" rowspan="2">Браузер</th>
            <th colspan="2">Посещения</th>
        </tr>
        <tr>
            <th>Количество</th>
            <th>В процентах</th>
        </tr>
        <tr>
            <td colspan="2">Mozila Firefox</td>
            <td>163</td>
            <td>59%</td>
        </tr>
        <tr>
            <td colspan="2">Google Chrome</td>
            <td>78</td>
            <td>28%</td>
        </tr>
        <tr>
            <td colspan="2">Safari</td>
            <td>35</td>
            <td>13%</td>
        </tr>
    </table>
</body>
</html>
```

Задание 5

Создайте таблицу, изображенную ниже (размеры отступов кратны 5):

<table>
<tr><th>Город</th><th>Посещения</th><th>Страниц</th><th>Время</th></tr>
<tr><td>СПб</td><td>199</td><td>18,02</td><td>00:13:45</td></tr>
<tr><td>Москва</td><td>69</td><td rowspan="2">нет данных</td><td>00:00:44</td></tr>
<tr><td>Киев</td><td>5</td><td>00:18:07</td></tr>
<tr><td colspan="3">Всего посещений</td><td>273</td></tr>
</table>

ЛАБОРАТОРНАЯ РАБОТА №7

Формы

Тег **<form>** устанавливает форму на веб-странице. Форма предназначена для обмена данными между пользователем и сервером. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.

Для отправки формы на сервер используется кнопка Submit, того же можно добиться, если нажать клавишу Enter в пределах формы. Если кнопка Submit отсутствует в форме, клавиша Enter имитирует ее использование.

Атрибут action в форме принимает ссылку(URL) который ведет либо на локальный скрипт, либо на сервер которые примут на себя обработку данных формы.

Поля в форме именуются тегом <input> имеют один важный атрибут – type, который определяет тип поля, который будет использован. Type может принимать значения:

Тип	Описание	Вид
button	Кнопка.	Кнопка
checkbox	Флажки. Позволяют выбрать более одного варианта из предложенных.	☐ Пиво ☐ Чай ☐ Кофе
file	Поле для ввода имени файла, который пересылается на сервер.	Выберите файл Файл не выбран
hidden	Скрытое поле. Оно никак не отображается на веб-странице.	
image	Поле с изображением. При нажатии на рисунок данные формы отправляются на сервер.	Отправить
password	Обычное текстовое поле, но отличается от него тем, что все символы показываются звездочками. Предназначено для того, чтобы никто не подглядел вводимый пароль.	
radio	Переключатели. Используются, когда следует выбрать один вариант из нескольких предложенных.	○ Пиво ○ Чай ○ Кофе
reset	Кнопка для возвращения данных формы в первоначальное значение.	Сбросить
submit	Кнопка для отправки данных формы на сервер.	Отправить
text	Текстовое поле. Предназначено для ввода символов с помощью клавиатуры.	

Рисунок 21 - Работа с формами

Чтобы понять данный материал выполните следующий код (так как у нас нет сервера,то никаких действий не выполнится после подтверждения формы):

```
<!DOCTYPE html>
<html>
<head>
    <title>form</title>
</head>
<body>
  <form action="something_url">
    <p>
      <b>Сколько пальчев у человека?</b>
    </p>

    <p>
      <input type="radio" name="answer" value="v1">Колечно же 5<Br>
      <input type="radio" name="answer" value="v2">Всего 4<Br>
      <input type="radio" name="answer" value="v3">Нет - 3</p>
    <p>
    <input type="text" name="answer" placeholder="мой вариант">
    <input type="submit"></p>
  </form>

</body>
</html>
```

Рисунок 22 - Работа с формами

В результате должно получится:

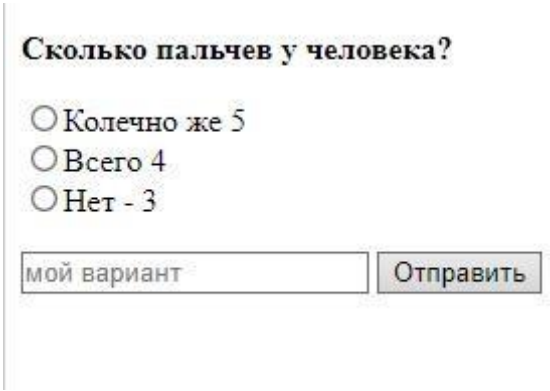


Рисунок 23 - Результат

Задание:

1) Изучите тег, который также относится к форме “SELECT”

2) Напишите форму, которая спрашивает у пользователя запрашивает персональные данные (такие как Имя, Фамилия, телефон, возраст (с помощью <select>), адрес и секретный вопрос (например: «Любимы цвет»)). Форма должна содержаться в блоке шириной 400px автоматической высотой, цветом на усмотрение, с отступом сверху в 75px и располагаться по центру экрана.

ЛАБОРАТОРНАЯ РАБОТА №8

Библиотека Bootstrap (Bootstrap framework)

Что такое Bootstrap?

- Bootstrap - это свободно распространяемая front-end библиотека, используемая для более быстрой и легкой разработки веб-страниц;
- Bootstrap включает основанные на HTML и CSS шаблоны для типографики, форм, кнопок, таблиц, навигации, модальных окон, слайдеров, и многие другие, такие как опциональные плагины JavaScript;

- Также Bootstrap дает возможность создавать адаптивный веб-дизайн.

Адаптивный веб-дизайн — дизайн веб-страниц, обеспечивающий правильное отображение сайта на различных устройствах, подключённых к интернету и динамически подстраивающийся под заданные размеры окна браузера.
Зачем использовать Bootstrap?

- Преимущества библиотеки Bootstrap: Простота использования:

Любой, у кого есть базовые знания HTML и CSS может использовать Bootstrap;

- Адаптивные возможности: Адаптивный CSS библиотеки Bootstrap
- Подстраивается под телефоны, планшеты и большие мониторы;
- Используется метод Mobile-first: в Bootstrap 3.0, стили для мобильнойразработки являются частью ядра библиотеки (т.е. они встроены в нее);
- Совместимость с различными браузерами: Bootstrap совместим со всемисовременными браузерами (Chrome, Firefox, Internet Explorer, Safari, and Opera).

В лабораторных работах будет использоваться Bootstrap 3.3.7, есть новая версияBootstrap 5, но ее использование требует понимания разметки с использованием Flexbox.

Где взять Bootstrap?

Есть два варианта, для того чтобы начать использовать Bootstrap на вашем собственномвеб-сайте:
Вы можете:

- Скачать Bootstrap с сайта getbootstrap.com;
- Или добавить ссылку на Bootstrap из CDN (Content Delivery Network - Сетьдоставки контента).

Скачать Bootstrap, если вы хотите скачать библиотеку Bootstrap и поместить ее в

каталог вашего веб-сайта собственноручно, то Вам необходимо зайти на getbootstrap.com, и далее следовать предложенным там инструкциям.

Bootstrap CDN Если Вы не хотите скачивать Bootstrap сами, то Вы можете поместить ссылку на него из CDN (Content Delivery Network). (<link rel = „stylesheet“ href = „ссылка CDN“).

Bootstrap 3.0 ориентирован на разработку для мобильных устройств

Bootstrap 3 создавался таким образом, чтобы быть была возможность создавать адаптивные страницы для мобильных устройств. Стили для разработки страниц длямобильных устройств являются частью ядра библиотеки.

Чтобы обеспечить правильное отображение и масштабирование на тач-скринах, добавляем следующий тег <meta> внутри элемента <head> :

<meta name="viewport" content="width=device-width, initial-scale=1">

Значение атрибута width=device-width задает ширину страницы в соответствие с шириной экрана устройства (которая будет меняться в зависимости от устройства).

Значение атрибута initial-scale=1 задает начальный масштаб страницы, когда страница загружается браузером впервые.

Контейнеры (Containers)

Bootstrap требует элемент "container" для всего содержимого сайта.Существует два вида контейнеров, из которых моно выбирать:

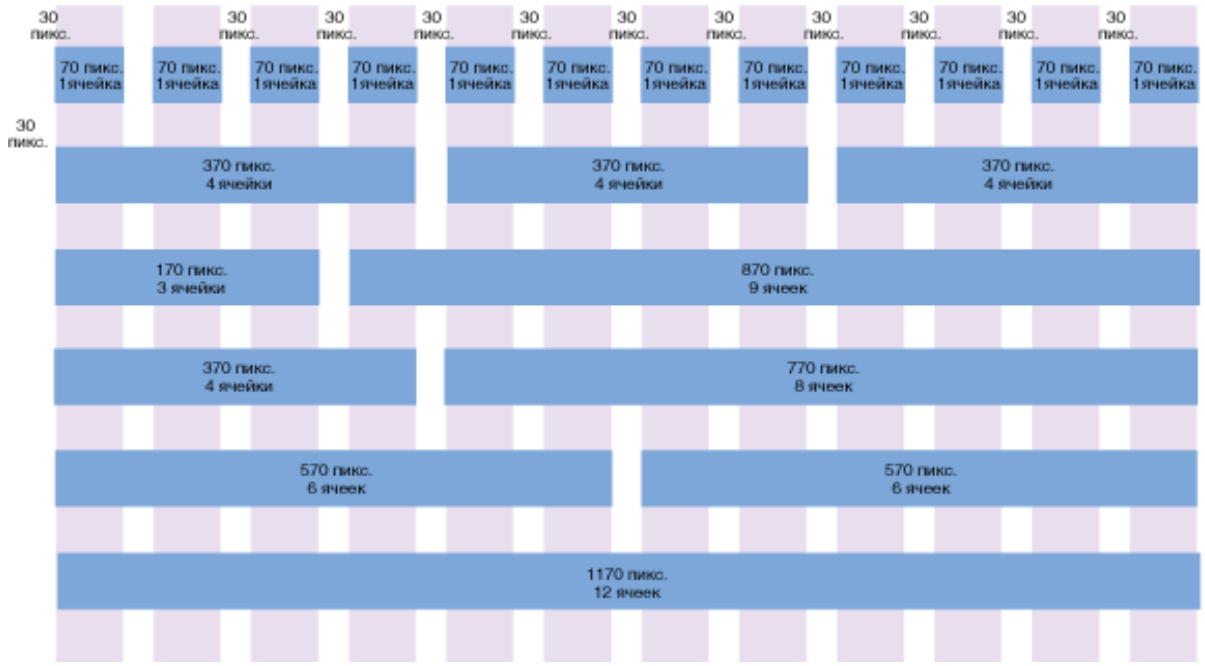
1. Класс .container представляет собой адаптивный контейнер с фиксированной шириной (справа и слева от контейнера есть некоторые отступы от границокна).
2. Класс .container-fluid обеспечивает контейнер на всю ширину экрана (охватываетвесь экран устройства без полей). Важно: Контейнеры не могут быть вложены в друг друга.

Важно: Контейнеры не могут быть вложены в друг друга.

Сетка Bootstrap

Сетка Bootstrap позволяет разместить контент в 12 колонок по ширине страницы.

ТакжеВы можете объединять эти колонки, делая их более широкими. Пример сетки показан ниже:



Сетка Bootstrap является адаптивной, и колонки меняют свою ширину относительно размера экрана устройства: на большом экране содержимое страницы может выглядеть лучшепри организации в три колонки, а на маленьком экране может быть лучше организовать блокисодержимого друг под другом.

Классы сетки Стека Bootstrap содержит 4 класса:

- xs (для телефонов)
- sm (для планшетов)
- md (для рабочих столов среднего размера)
- lg (для широких мониторов)

Классы, представленные выше, могут быть скомбинированы для создания болеединамической и гибкой разметки.

Важно: Каждый класс имеет автомасштабирование, т.е. если Вы хотитеустановить одинаковую ширину блоков с "xs" и "sm" классами, Вам необходимоустановить ширину только для класса xs.

Правила использования сетки Bootstrap

- Строки должны быть помещены в класс. Container (fixed-width)или

.containerfluid (full-width) для надлежащего выравнивания и установки отступов;

- Используйте строки для создания горизонтальных групп элементов;
- Контент должен быть размещен в колонках, и только колонки могут быть непосредственными дочерними элементами строк;
- Предопределенные классы, такие как .row и .col-sm-4 могут быть использованы для быстрой верстки страниц;
- Колонки имеют отступы относительно друг друга, которые создаются посредством свойства "padding". Эти отступы обнуляются в строках для первой и последней колонок посредством отрицательного значения margin для строк .rows;
- Колонки сетки создаются через определение количества колонок (из общего количества в 12 колонок), которые Вам необходимо объединить. Например, три одинаковые колонки на странице будут представлены тремя классами
.col-sm-4 (см. пример ниже).

```
<div class="row">
    <div class="col-sm-4">.col-sm-4</div>
    <div class="col-sm-4">.col-sm-4</div>
    <div class="col-sm-4">.col-sm-4</div>
</div>
```

Рисунок 24 - .col-sm-4

Две колонки разной ширины

Следующий пример показывает, как получить две колонки разной ширины для планшетов(которые останутся такими же при масштабировании до больших

рабочих столов):

Базовая структура сетки Bootstrap

```
<div class="row">
    <div class="col-sm-4">.col-sm-4</div>
    <div class="col-sm-8">.col-sm-8</div>
</div>
```

Рисунок 25 - Базовая структура сетки

Следующий код показывает базовую разметку сетки Bootstrap:

```
<div class="container">
    <div class="row">
        <div class="col-*-*">
    </div>
</div>

<div class="row">
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
</div>

<div class="row"> ... </div>

</div>
```

Рисунок 26 - Базовая разметка сетки Bootstrap

Таким образом, для создания необходимой разметки, создается контейнер ( ). Далее, создается строка ( ). Затем, добавляется желаемое количество колонок (теги с соответствующими классами .col-*-*). Важно, чтобы количество колонок в классах .col-*-* всегда было равно 12 для каждой строки

Свойства сетки

В скриншоте ниже приведены свойства сетки Bootstrap для различных размеров устройств:

	Экстра малые устройства, телефоны (<768px)	Малые устройства, Планшеты (>=768px)	Средние устройства, мониторы (>=992px)	Широкоформатные устройства, мониторы (>=1200px)
Поведение сетки	Все время горизонтальное	Свернуто на старте, горизонтальное при превышении указанного размера экрана	Свернуто на старте, горизонтальное при превышении указанного размера экрана	Свернуто на старте, горизонтальное при превышении указанного размера экрана
Ширина контейнера	Auto	750px	970px	1170px
Префикс класса	.col-xs-	.col-sm-	.col-md-	.col-lg-
Количество колонок	12	12	12	12
Ширина колонок	Auto	~62px	~81px	~97px
Размер отступов	30px (15px на каждую сторону колонки)	30px (15px на каждую сторону колонки)	30px (15px на каждую сторону колонки)	30px (15px на каждую сторону колонки)
Вложенность	Да	Да	Да	Да
Смещение	Да	Да	Да	Да
Порядок столбцов	Да	Да	Да	Да

Рисунок 27 - Свойства сетки Bootstrap

Задание:

- На основе полученных выше теоретических знаний сверстать макет, представленный ниже в виде скриншота с использованием Bootstrap (используйте официальный сайт для получения информации об элементах, которые вы не знаете, как сверстать)

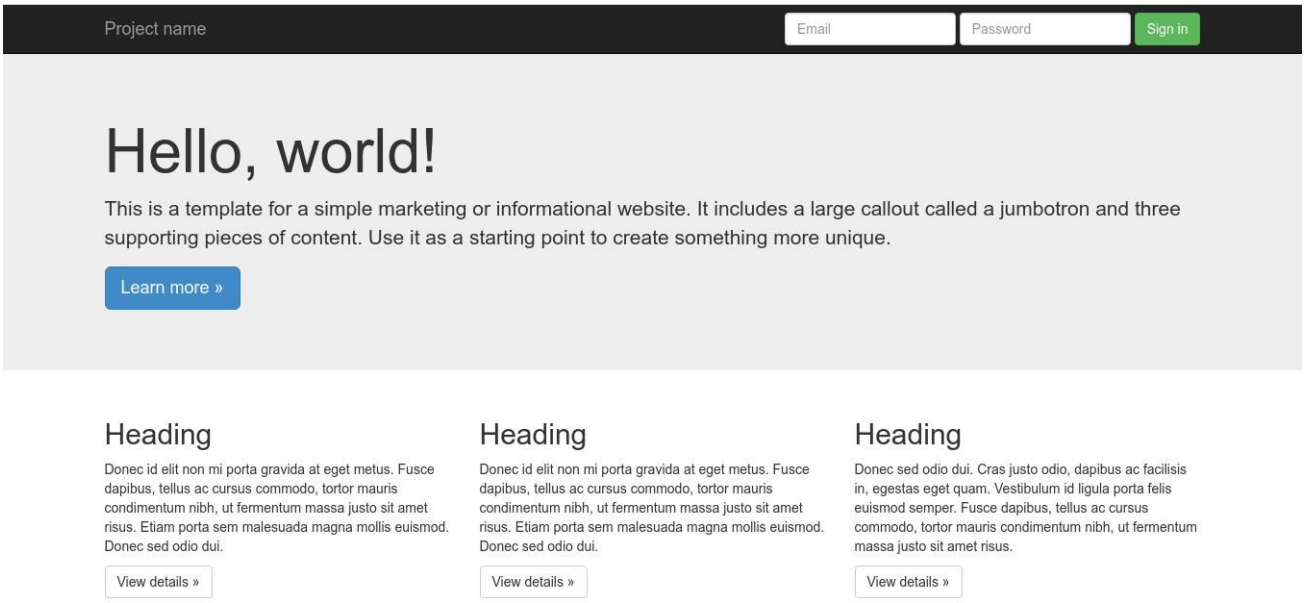


Рисунок 28 - Задание

ЛАБОРАТОРНАЯ РАБОТА №9

Создание выпадающего меню с помощью CSS

Выпадающее меню служит в качестве обзора иерархии разделов, которые содержатся в пункте меню, объединяющем их. Обычно в меню перечисляются все подразделы определенной секции, если навести указатель мыши на нее.

Выпадающее меню очень удобно, когда показывает все содержание всех секции, содержащихся на сайте, и дает возможность перейти на любую страницу из любого места сайта.

Ход работы:

1. Создайте HTML (index.htm) документ, в котором составьте разметку нашей страницы, то есть – меню с подключением стилей

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" type="text/css" href="styles.css">
    </head>
    <body>
       <ul id="nav">
            <li>
                <a href="#" title="Вернуться на главную страницу">Главная</a>
            </li>
            <li>
                <a href="#" title="Информация о компании">О нас</a>
                <ul>
                    <li><a href="#">Продукты</a></li>
                    <li><a href="#">Команда</a></li>
                </ul>
            </li>
            <li>
                <a href="#" title="Что мы можем для вас сделать">Услуги</a>
                <ul>
                    <li><a href="#">Первая услуга</a></li>
                    <li><a href="#">Вторая услуга</a></li>
                    <li><a href="#">Третья услуга</a></li>
                    <li><a href="#">Четвёртая услуга</a></li>
                </ul>
            </li>
            <li>
                <a href="#" title="Наша продуктовая линейка">Продукты</a>
                <ul>
                    <li><a href="#">Первый продукт</a></li>
                    <li><a href="#">Второй продукт</a></li>
                    <li><a href="#">Третий продукт </a></li>
                    <li><a href="#">Четвёртый продукт</a></li>
                    <li><a href="#">Пятый продукт</a></li>
                    <li><a href="#">Шестой продукт</a></li>
                    <li><a href="#">Седьмой продукт</a></li>
                    <li><a href="#">Восьмой продукт</a></li>
                </ul>
            </li>
            <li>
                <a href="#" title="Как с нами связаться">Контакт</a>
                <ul>
                    <li><a href="#">Часы работы</a></li>
                    <li><a href="#">Адрес</a></li>
```

```
                </ul>
            </li>
        </ul>
    </body>
    </html>
```

Элемент #nav <ul> содержит серию элементов <li>. Все пункты, которые нуждаются в выпадающих подпунктах, содержат другой элемент <ul>. Обратите внимание, что элемент <ul> выпадающих подпунктов не имеет класса.

2. Будем использовать CSS для трансформирования серии вложенных

списков <ul> в выпадающее меню. Создайте файл styles.css в корневой папке index.htm, и добавьте в него данный код:

```css
#nav{
    float:left;
    width:100%;
    list-style:none;
    font-weight:bold;
    margin-bottom:10px;
}
#nav li{
    float:left;
    margin-right:10px;
    position:relative;
    display:block;
}
#nav li a{
    display:block;
    padding:5px;
    color:#fff;
    background:#333;
    text-decoration:none;
    transition: .4s background; /* Делаем переход для плавности hover'а */
    text-shadow:1px 1px 1px rgba(0,0,0,0.75); /* Тень текста, чтобы приподнять его на немного */
    -moz-border-radius:2px;
    -webkit-border-radius:2px;
    border-radius:2px;
}
#nav li a:hover{
    color:#fff;
    background:#ffcc00;
    background:rgba(255,204,0,0.75); /* Выглядит полупрозрачным */
    text-decoration:none;  /* Убираем "декорацию" текста для избежание подчёркивания */
    transition: .4s background; /* Делаем переход для плавности hover'а */
}

/*--- ВЫПАДАЮЩИЕ ПУНКТЫ ---*/
#nav ul{
    list-style:none;
    position:absolute;
    left:-9999px; /* Скрываем за экраном, когда не нужно */
    opacity:0; /* Устанавливаем начальное состояние прозрачности */
    -webkit-transition:0.25s linear opacity; /* В Webkit выпадающие пункты будут проявляться */
}
#nav ul li{
    padding-top:1px; /* Вводим отступ между li чтобы создать иллюзию разделенных пунктов меню */
    float:none;
}
#nav ul a{
    white-space:nowrap; /* Останавливаем перенос текста и создаем многострочный выпадающий пункт */
    display:block;
```

```css
#nav li:hover ul{ /* Выводим выпадающий пункт при наведении курсора */
    left:0; /* Приносим его обратно на экран, когда нужно */
    opacity:1; /* Делаем непрозрачным */
}
#nav li:hover a{ /* Устанавливаем стили для верхнего уровня, когда выводится выпадающий список */
    background:#ffcc00;
    background:rgba(255,204,0,0.5); text-decoration:underline;
}
```

```
#nav li:hover ul a{ /* Изменяем некоторые стили верхнего уровня при выводе выпадающего пункта */
    text-decoration:none;
    -webkit-transition:-webkit-transform 0.075s linear;
}
#nav li:hover ul li a:hover{ /* Устанавливаем стили для выпадающих пунктов, когда курсор наводится на конкретный пункт */
    background:#333;
    background:rgba(51,51,51,0.5); /* Будет полупрозрачным */
    text-decoration:none;
    -moz-transform:scale(1.05);
    -webkit-transform:scale(1.05);
}
```

В первом разделе кода устанавливаем обычное горизонтальное меню.

!!! Обратите внимание, что селекторы #nav lu и #nav li a выделяют все элементы списка и ссылки в выпадающих пунктах тоже.

Следует отметить использование position:relative; для элементов списка. Таким образом, используется position:absolute; для вложенных элементов <ul>.

```
#nav ul{
    list-style:none;
    position:absolute;
    left:-9999px; /* Скрываем за экраном, когда не нужно */
    opacity:0; /* Устанавливаем начальное состояние прозрачности */
    -webkit-transition:0.25s linear opacity; /* В Webkit выпадающие пункты будут проявляться */
}
```

В данном коде устанавливаются стили для вложенных <ul> в пункт верхнего уровня. Очевидно, что нужно удалить метки пунктов списка с помощью list-style:none; и установить position:absolute; для позиционирования выпадающих подпунктов под пунктом списка, который их содержит.

Следующая строка гораздо более интересна. Обычно используется свойство display:none; для того, чтобы скрыть выпадающий пункт, когда он не используется. Но так как большинство программ для чтения с экрана игнорируют все, что имеет свойство display:none;, то использование такого метода очень нежелательно. Вместо этого используем абсолютное позиционирование <ul> для помещения его в позицию -9999px за пределами экрана, когда он не используется.

Затем следует свойство opacity:0;, для скрытия <ul>, и декларация для браузеров Webkit, для плавного вывода элемента <ul> при наведении курсора мыши.

```
#nav ul li{
    padding-top:1px; /* Вводим отступ между li чтобы создать иллюзию разделенных пунктов меню */
    float:none;
}
#nav ul a{
    white-space:nowrap; /* Останавливаем перенос текста и создаем многострочный выпадающий пункт */
    display:block;
```

```
#nav li:hover ul{ /* Выводим выпадающий пункт при наведении курсора */
    left:0; /* Приносим его обратно на экран, когда нужно */
    opacity:1; /* Делаем непрозрачным */
}
```

Здесь устанавливаются стили по умолчанию для пунктов списка и ссылок. Обратите внимание на свойство padding-top:1px; для элемента <li>. Так как все цвета устанавливаются для элементов <a>, то установка отступа в 1px для элемента <li> сдвигает элемент <a>, и, следовательно, цветную область от границы пункта списка. Таким образом, создается иллюзия, что пункты списка разделены.

Для элемента #nav ul a устанавливается свойство white-space:nowrap; для предотвращения переноса строк на другую строку.

Последняя часть кода выводит выпадающие подпункты, когда курсор мыши оказывается над соответствующим пунктом меню.

```
#nav li:hover a{ /* Устанавливаем стили для верхнего уровня, когда выводится выпадающий список */
    background:#ffcc00;
    background:rgba(255,204,0,0.5); text-decoration:underline;
}
```

#nav li:hover a определяет, что произойдет со ссылкой верхнего уровня, когда наследник будет иметь состояние *hover*:

- Выпадающий список <ul> расположен внутри элемента <li>.
- Если навести курсор мыши на ссылку (<a>) в выпадающем списке (<ul>), то одновременно пункт списка верхнего уровня (<li>) тоже будет иметь состояние *hover*, так как выделен контент внутри него.
- Так как технически используется состояние *hover* для элемента списка верхнего уровня, то #nav li:hover a действует, задавая стиля для ссылки.

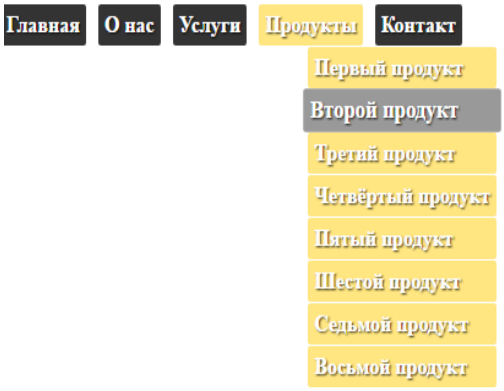
```
#nav li:hover ul a{ /* Изменяем некоторые стили верхнего уровня при выводе выпадающего пункта */
    text-decoration:none;
    -webkit-transition:-webkit-transform 0.075s linear;
}
```

Здесь изменяются некоторые аспекты для состояния *hover*,

чтобы выпадающие элементы отличались от ссылок верхнего уровня. В данном уровне просто отключается подчеркивание текста.

Также добавляется правило для браузеров Webkit -webkit-transition:-webkit-transform 0.075s linear;, которое анимирует -webkit-transform с помощью затухания/появления в течение 0.075 секунды.

```
#nav li:hover ul li a:hover{ /* Устанавливаем стили для выпадающих пунктов, когда курсор наводится на конкретный пункт */
    background:#333;
    background:rgba(51,51,51,0.5); /* Будет полупрозрачным */
    text-decoration:none;
    -moz-transform:scale(1.05);
    -webkit-transform:scale(1.05);
}
```



В последней части кода определяются стили для выделения определенного пункта в выпадающем списке при наведении курсора мыши.

Вначале определяются два свойства background:;.

Определение background:rgba(51,51,51,0.75); устанавливает умеренно серый фон для пункта с прозрачностью 0.75. Те браузеры, которые не распознают такое определение цвета будут использовать определение цвета в старом стиле в предыдущей строке.

ЛАБОРАТОРНАЯ РАБОТА №10

CSS: увеличение изображения при наведении курсора

В данной лабораторной работе рассмотрим, как создать эффект увеличения изображения при наведении на него курсором с помощью каскадных таблиц стилей.

1. Для выполнения лабораторной работы подготовьте 4 картинки (

*.jpg) одинакового размера. Затем создайте HTML (index.htm) документ, в котором составьте разметку страницы.

```html
<!DOCTYPE html>
    <html>
    <head>
        <title>Увеличение картинки при наведении</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <link rel="stylesheet" type="text/css" href="styles.css">
    </head>
    <body>
        <div class="blocks"><a href="#"><img src="1.jpg" alt="thumb" class="resize_thumb"><span>
            <img src="1.jpg" alt="large">
            Здесь можно написать текст.</span></a>
        </div>
        <div class="blocks"><a href="#"><img src="2.jpg" alt="thumb" class="resize_thumb"><span>
            <img src="2.jpg" alt="large">
            Здесь можно написать текст.</span></a>
        </div>
        <div class="blocks"><a href="#"><img src="3.jpg" alt="thumb" class="resize_thumb"><span>
            <img src="3.jpg" alt="large">
            Здесь можно написать текст.</span></a>
        </div>
        <div class="blocks"><a href="#"><img src="4.jpg" alt="thumb" class="resize_thumb"><span>
            <img src="4.jpg" alt="large">
            Здесь можно написать текст.</span></a>
        </div>
    </body>
</html>
```

В данной лабораторной работе картинки вставим непосредственно в HTML код. Добавим на всякий случай ссылки к каждой картинке, чтобы картинка была интерактивной. В будущем, для <a> ссылок мы будем применять стили в CSS.

2. Далее создаем файл styles.css в корневой папке index.htm, и добавляем в него данный код:

```css
.blocks {
    float: relative;
    clear: none; /* Можно установить left или right по необходимости */
    padding-bottom: 5px; /* Расстояние между миниатюрами */
    padding-right: 5px; /* Расстояние между миниатюрами и окружающим текстом */
    margin-left:700px; /* Отступ слева для центрирования */
}
.resize_thumb {
    /* Вводим нужный размер миниатюры здесь */
    width: 150px;
    height:100px;
}
```

3. Этот стиль будет использоваться для блока миниатюры.

Теперь добавим еще стиль, он будет для <a> ссылок. Уберем все “декорации” для ссылки и курсора.

```
.blocks a {
    display:block;
    text-decoration: none; /* Убираем подчеркивание ссылки */
    cursor:default; /* Меняем указатель мыши на обычный */
}
.blocks a:hover{
    position:relative;
}
```

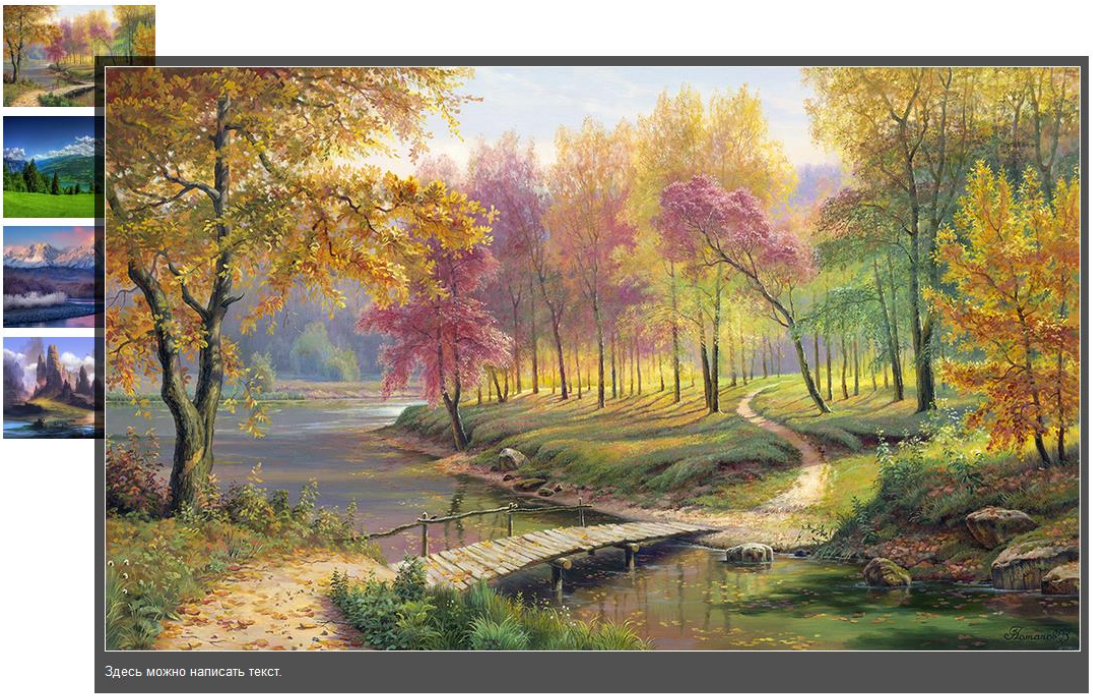
4. Следующим шагом добавим стиль для появляющейся увеличенной картинки. Описание всех строк имеется в коде.

```
.blocks span img {
    border: 1px solid #FFFFFF; /* Добавляем рамку вокруг изображения */
    margin-bottom: 8px; /* Сдвигаем текст вниз от изображения */
    width:100%; /* Размер картинки по всей площади блока */
}

.blocks a span {
    position: absolute;
    display:none;
    color: #FFFFFF; /* Цвет названия */
    text-decoration: none; /* Убираем подчеркивание ссылки */
    font-family: Arial, Helvetica, sans-serif; /* Устанавливаем шрифты */
    font-size: 13px; /* Размер шрифта названия */
    background-color: rgba(0,0,0,0.7); /* Делаем фон появляющегося блока прозрачным */
    padding-top: 10px;  /* Вводим отступы */
    padding-right: 10px;
    padding-bottom: 13px;
    padding-left: 10px;
}
```

5. И заключающий стиль будет hover. Для того, чтобы наша увеличенная картинка появилась, нужно добавить такой стиль:

```
.blocks a:hover span {
    display:block;
    top: 50px; /* Большое изображение появляется вверху на 50px от миниатюры */
    left: 90px; /* Большое изображение появляется справа на 90px от миниатюры */
    width:80%; /* Процентное соотношение размера блока со страницей */
    opacity:0.97; /* Небольшая прозрачность большого изображения */
    /* Если добавить правило cursor:default;, то отключится курсор в виде руки на большом изображении */
}
```

Итог:



ЛАБОРАТОРНАЯ РАБОТА №11

«CSS: эффект тени»

В данной лабораторной работе разберем, как создать эффект тени с помощью каскадных таблиц стилей.

Целью лабораторной работы является показать ещё один атрибут CSS, с помощью которого обычному элементу можно придать особенный вид. Этот атрибут – box- shadow: Данный атрибут добавляет тень к элементу и причём теней может быть несколько (они перечисляются через запятую).

Существует два вида записи:

Первый – сдвиг по **X**, сдвиг по **Y**, радиус тени, **rgb**-цвет, степень размытия (от 0.1 до 1);

RGB-цвет расшифровывается, как Red Green Blue, если вы воспользуетесь генератором html-цветов, то заметите, что помимо html-кода цвета есть ещё и три строчки с теми названиями, в каждой из них число. Код сгенерированного цвета записывают в виде трёх чисел через запятую.

Ход работы:

Разметка

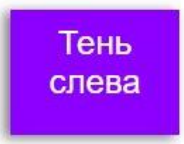
1. Для начала создадим HTML (index.htm) документ, в котором составим разметку нашей страницы, то есть – добавим блоки, для которых будем создавать тень.

```html
<!DOCTYPE html>
    <html>
    <head>
        <link rel="stylesheet" type="text/css" href="styles.css">
    </head>
     <body>
        <div id="left">
            Тень слева
        </div>
        <div id="right">
            Тень справа
        </div>
        <div id="bord">
            Тень вокруг блока и у текста
        </div>
        <div id="twoth">
            Тень внутри блока и под ним
        </div>
        <div id="theard">
            Тень внутри блока, снизу, сверху и вокруг него
        </div>
    </body>

</html>
```

Добавили 5 блоков в HTML код, теперь нам нужно придать им «форму».

-Создаём файл styles.css в корневой папке index.htm, и добавляем в него данный код:

```css
body {margin-left:700px; margin-top:50px;} /* Установим глобальные отступы */
#left {
  width: 100px; /* Размеры блока */
  height: 70px;
  background: #8c00ff; /* Цвет блока */
  color: white; /* Цвет текста в нём */
  padding: 10px;
  margin-bottom:50px; /* Доп. отступы */
  margin-left:100px;
  font: 24px Arial;
  text-align: center;
  box-shadow: -4px 2px 8px rgba(0,0,0,0.5); /* Добавляем тень блоку */
  user-select:none; /* "Запрещаем" выделять текст */
  cursor:default; /* Устанавливаем стандартный курсор при наведении на текст */
}
```



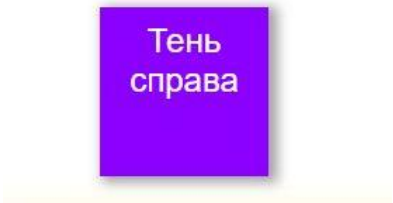
Теперь мы будем иметь квадратный блок с чёрной тенью (rgba(0,0,0,0.5)) и радиусом размытия в **8px**.

Как мы видим – тень смещена на 4 пикселя влево и 2 пикселя вниз и хоть

цвет тени чёрный, но из-за того, что прозрачность **0.5** цвет тени размытого серого цвета.

-Теперь добавим стиль к с тенью справа. Добавляем к каскадным стилям:

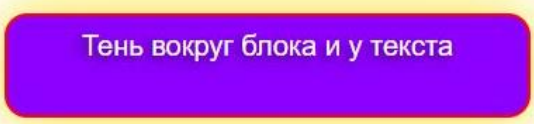
```
#right {
    width: 100px;
    height: 100px;
    background: #8c00ff;
    color: white;
    padding: 10px;
    margin-left:100px;
    font: 24px Arial;
    text-align: center;
    box-shadow: 4px 2px 8px 2px #A3A3A3;
    margin-bottom:50px;
    user-select:none;
    cursor:default;
  }
```



Обратите внимание на то, что тень можно задать и тексту. Ниже показан блок с текстом внутри него и тексту заданы некоторые параметры тени (1-ым способом). У блока закруглённые углы. Тень, заданная для него, автоматически "закругляется."

-Добавляем стили к 3-ему блоку.

```
#bord {
  width: 350px;
  height: 50px;
  background: #8c00ff;
  color: white;
  padding: 10px;
  border: 2px solid red; /* Обводка блока */
  border-radius: 16px; /* Закругление обводки */
  margin-bottom:50px;
  font: 20px Arial;
  text-align: center;
  text-shadow: -4px 3px 8px rgba(0,0,0,0.6); /* Тень текста */
  box-shadow: 0 0 40px rgba(255,217,0,0.9); /* Тень блока */
  user-select:none;
  cursor:default;
}
```



Теперь будет продемонстрировано то, что можно задать сразу несколько

теней. В этом примере их будет две и, причём, одна из теней задана внутри блока (**inset**) чёрного цвета.

```
#twoth {
  width: 350px;
  height: 50px;
  background: #8c00ff;
  color: white;
  padding: 10px;
  border-radius: 16px;
  margin-bottom:50px;
  font: 20px Arial;
  text-align: center;
  box-shadow:
      inset 5px 6px 3px rgba(0,0,0,0.7), /* Тень внутри блока */
      3px 3px 4px rgba(0,0,0,0.9); /* Тень снаружи блока */
  user-select:none;
  cursor:default;
}
```

Тень внутри блока и под ним

Зададим последнему блоку ещё пару теней.

```
#theard {
  width: 350px;
  height: 50px;
  background: #8c00ff;
  border-radius: 16px;
  font: 20px Arial;
  box-shadow:
      inset 5px 6px 3px rgba(0,0,0,0.6),
      3px 3px 4px rgba(0,0,0,0.6),
      -2px -2px 4px rgba(223,152,0,0.7),
      0 0 20px 3px red;
  color: white;
  padding: 10px;
}
```

Тень внутри блока, снизу, сверху и вокруг него

Итог:

ЛАБОРАТОРНАЯ РАБОТА №12

«CSS: свойство transition»

Например, если вы написали стиль, в котором у объекта при наведении курсора мышки должен меняться цвет заднего фона, то благодаря свойству **transition:** можно сделать плавную смену цвета. Плавность (длительность) смены того или иного параметра регулируется количеством секунд, записанных в свойстве.

Ход выполнения работы:

Разметка

Для начала создайте HTML (index.htm) документ, в котором будет

составлена разметка нашей страницы, то есть – добавим блоки, для которых будем применять стили с переходами.

```html
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="styles.css">
    </head>
    <body>
        <div id="tran_1">
            Фон полностью изменится через 2 секунды
        </div>
        <div id="tran_2">
            Фон полностью изменится через 2 секунды
        </div>
        <div id="tran_3">
            У блока меняется размер, рамка, фон и цвет текста.
        </div>
    </body>
</html>
```

Мы добавили 3 блока в HTML код, теперь мы применим к ним стили.

Пример записи transition:

transition: background-color 2s ease;

2s – это 2 секунды (можно задавать и нецелые значения);

ease - временная функция, используемая для анимации (помимо это существуют - linear, ease-in-out, ease-in, ease-out, cubic-bezier)

В коде примеров свойство transition: будет записываться три раза с разными префиксами – для трёх браузеров:

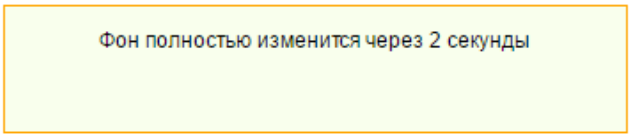
-moz- (Firefox 4);
-webkit- (Chrome и Safari);
-o- (Опера);

Internet Explorer «не понимает» это свойство.

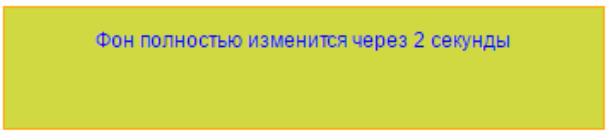
Создайте файл styles.css в корневой папке index.htm, и добавьте в него данный код:

```
#tran_1{
  width: 350px;
  height: 50px;
  margin: auto;
  border: 1px solid Orange;
  padding: 10px 0 10px;
  text-align: center;
  background-color: #F9FFED;
  color: Black;
  cursor: pointer;
    -moz-transition: background-color 2s ease; /* Устанавливаем переход для всех браузеров */
    -o-transition: background-color  2s ease;
    -webkit-transition: background-color 2s ease;
  }

  #tran_1:hover {
    background-color: #D0D941;
    color: Blue;
  }
```

Первоначальный вид блока.



Стиль блока изменился за 2 секунды плавным переходом.



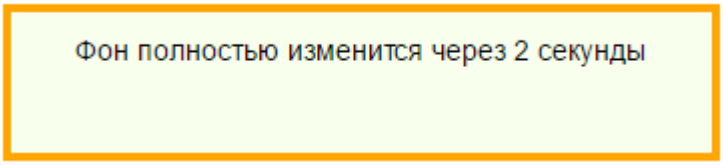
Если вы хотите, чтобы изменялось сразу несколько параметров, то нужно записать через запятую эти параметры и длительности их изменений.

В примере, продемонстрированном ниже, создан блок, у которого при наведении курсора мышки меняется сразу: цвет фона и рамки, а также размер, цвет и шрифт надписи. Каждому из параметров задана разная длительность перехода:

```css
#tran_2 {
  width: 350px;
  height: 50px;
  margin: auto;
  border: 4px solid Orange;
  padding: 10px 0 10px;
  text-align: center;
  font-size: 14px;
  background-color: #F9FFED;
  color: Black;
  cursor: pointer;
  -moz-transition: border 1s ease, color 1.5s ease,
  font 1.5s ease, background-color 2s ease;
  -o-transition: border 1s ease, color 1.5s ease,
  font 1.5s ease, background-color 2s ease;
  -webkit-transition: border 1s ease, color 1.5s ease,
  font 1.5s ease, background-color 2s ease;
}
```

```css
#tran_2:hover {
  background-color: #D0D941;
  color: Red;
  font: italic 16px COMIC SANS MS;
  border: 4px solid Blue;
}
```

Первоначальный вид блока.



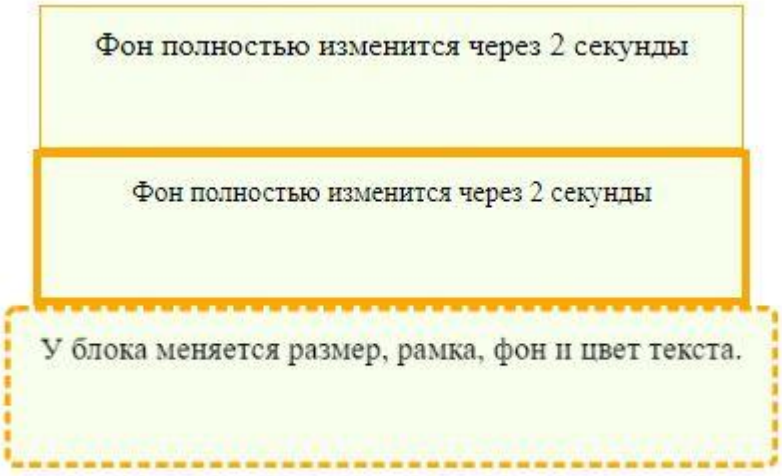
Стиль блока изменился за 2 секунды плавным переходом.

Если вы хотите применить свойство «перехода» сразу ко всем параметрам с одинаковой длительностью – то вместо их перечисления можно просто написать "all".

В последнем примере у блока будут изменяться - цвет заднего фона, цвет текста, а также радиус и тип рамки (с пунктирной на точечную). Помимо этого, с помощью свойства **transform:** блок будет увеличиваться на 20 процентов (1.2).

```css
#tran_3 {
  width: 380px;
  height: 55px;
  margin: auto;
  border: 3px solid Orange;
  border-style: dashed;
  border-radius: 6px;
  padding: 10px 0 10px;
  text-align: center;
  background-color: #F9FFED;
  color: Black;
  cursor: pointer;
  -moz-transition: all 1.5s ease;
    -o-transition: all 1.5s ease;
    -webkit-transition: all 1.5s ease;
}

#tran_3:hover {
  background-color: #D0D941;
  color: Red;
  border-style: dotted;
  border-radius: 28px;
  -webkit-transform: scale(1.2);
  -moz-transform: scale(1.2);
  -o-transform: scale(1.2);
}
```

Итог:



ЛАБОРАТОРНАЯ РАБОТА №13

«CSS: Всплывающее окно»

Для вывода важных сообщений или просто изменений, произведённых на сайте, можно использовать всплывающие окна. Всплывающие окна бывают

двух видов: обычные и модальные.

Примечание

Модальные окна отличаются от обычных тем, что пока модальное окно открыто пользователь не может взаимодействовать с другими элементами сайта до тех пор, пока не закроет модальное окно.

Ход выполнения работы:

Разметка

Первым шагом создания всплывающего окна является создание элемента (или любого другого элемента):

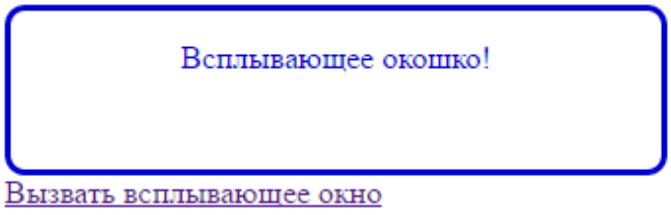
```
<body>
    <div id="okno">
        Всплывающее окно!
    </div>
    </a>
</body>
```

CSS

Создайте файл styles.css в корневой папке index.htm, и добавьте в него данный код:

```
#okno{
    display: none;
    width: 300px;
    height: 50px;
    text-align: center;
    padding: 15px;
    border: 3px solid #0000cc;
}
```

Следующий и будет использоваться в качестве всплывающего окна. Теперь его необходимо скрыть с помощью значения none свойства display и добавить ссылку, при нажатии на которую будет появляться всплывающее окно:



Используя псевдо-класс :target выбираем и применяем стили к элементу, к которому был осуществлён переход. Таким образом после перехода по ссылки значение свойства display элемента сменится с none на block.

Теперь надо расположить посередине страницы, чтобы он стал похож на всплывающее окно. Делаем его абсолютно позиционированным и центрируем его по вертикали и горизонтали:

```
#okno{
    display: none;
    width: 300px;
    height: 50px;
    text-align: center;
    padding: 15px;
    border: 3px solid #0000cc;
    position: absolute;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    margin: auto;
}
```

Следующим шагом будет реализация скрытия окна, при нажатии на любое место страницы или на само окно. Для этого нам нужно расположить элемент

 внутри элемента <a>:

```
<body>
    <a href="#okno" id="main">
        Показать всплывающее окно
    </a>
    <div id="okno">
        Всплывающее окно!
    </div>
</body>
```

Затем мы позиционируем элемент <a> и растягиваем его на всю ширину и высоту окна. Задаём ему display: none; и перенаправляем нашу ссылку на него:

```
#main{
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
}

#okno{
    display: none;
    width: 300px;
    height: 50px;
    text-align: center;
    padding: 15px;
    border: 3px solid #0000cc;
    position: absolute;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    margin: auto;
}

#okno:target{
    display: block;
}
```

Вызвать всплывающее окно

Всплывающее окошко!

У элемента убираем display: none; (он больше не нужен, так как скрываем мы теперь <a>). В итоге родительский <a> выполняет теперь скрытие всплывающего окна, перенаправляя выбор на страницу.

На этом создание простого всплывающего окна закончено.

Модальное окно

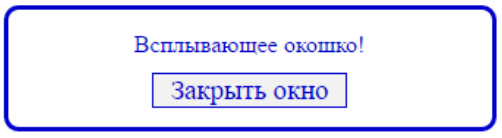
Для создания всплывающего модального окна, берём элемент , оформляем его и добавляем ссылку, при нажатии на которую он будет появляться:

```
<body>
    <a href="#okno" id="main">
        Показать всплывающее окно
    </a>
    <div id="okno">
        Всплывающее окно!<br>
        <a href="#" class="close">Закрыть</a>
    </div>
</body>
```

Следующим шагом в создании полноценного модального окна будет добавление кнопки, которая будет скрывать наше окно. Кнопку сделаем из обычной ссылки, добавив её к нашему и оформив:

```css
#main{
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
}

#okno{
    display: none;
    width: 300px;
    height: 50px;
    text-align: center;
    padding: 15px;
    border: 3px solid #0000cc;
    position: absolute;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    margin: auto;
}

#okno:target{
    display: block;
}

.close{
    display: inline-block;
    border: 1px solid #0000cc;
    color: #0000cc;
    padding: 0 12px;
    margin: 10px;
    text-decoration: none;
    background: #f2f2f2;
    font-size: 14pt;
    cursor: pointer;
}

.close:hover{
    background: #e6e6ff;
}
```

Вызвать всплывающее окно



Для эффекта затемнения изменим структуру html:

```
<body>
    <a href="#zatemnenie" id="main">
        Показать всплывающее окно
    </a>
    <div id="zatemnenie">
        <div id="okno">
            Всплывающее окно!<br>
            <a href="#" class="close">Закрыть</a>
        </div>
    </div>
</body>
```

Позиционируем и растягиваем его на всю ширину и высоту окна. Задаём ему display: none; и перенаправляем ссылку вызова окна на него.

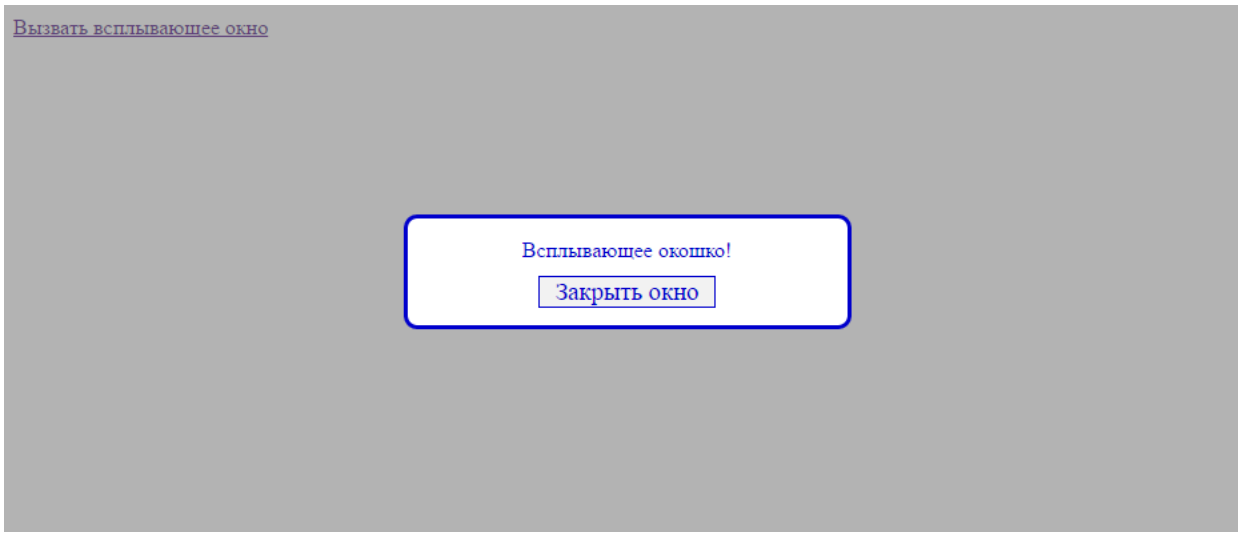
У дочернего убираем display: none; В итоге родительский теперь отвечает за отображение модального окна и за затемнение фона страницы, а дочерний только за оформление самого окна:

```css
#main{
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
}

#okno{
    width: 300px;
    height: 50px;
    text-align: center;
    padding: 15px;
    border: 3px solid #0000cc;
    position: absolute;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    margin: auto;
}

.close{
    display: inline-block;
    border: 1px solid #0000cc;
    color: #0000cc;
    padding: 0 12px;
    margin: 10px;
    text-decoration: none;
    background: #f2f2f2;
    font-size: 14pt;
    cursor: pointer;
}

.close:hover{
    background: #e6e6ff;
}
```

Добавляем стили для затемнения:

```css
#zatemnenie{
    display: none;
    width: 100%;
    height: 100%;
    position: absolute;
    background: rgba(102,102,102,.5);
    top: 0;
    left: 0;
}

#zatemnenie:target{
    display: block;
}
```

В итоге мы получаем такое всплывающее окно, активируемое по ссылке.



Примечание: если вам нужно, чтобы при заходе на страницу пользователь сразу видел всплывающее окно (а не вызывал его по ссылке), то адрес страницы надо будет прописывать вместе с id окна (например адрес может выглядеть так: site.ru/papka/documet.html#okno).

ЛАБОРАТОРНАЯ РАБОТА №14

Основные положения JavaScript

Сценарий JavaScript может быть помещен в любом месте web страницы внутри контейнера script. Если во время интерпретации HTMLдокумента браузер встретит тег script, он первым делом выполнит код

скрипта и лишь затем продолжит интерпретацию страницы дальше. Контейнеров script в одном документе может быть сколько угодно.

В общем виде встраивание скрипта в страницу с помощью тега script имеет вид:

```
<script type="text/javascript">
код сценария
</script>
```

Параметр type можно и не указывать, так как значение text/javascript является значением по умолчанию.

Пример web-страницы со встроенным сценарием, который выводит на экран окошко предупреждения с приветствием «Hello, World!»:

```
<html>
<head>
<title>Приветствие</title>
</head>
<body>

<script type="text/javascript">
alert('Hello, World!');
</script>
<p>Страница с приветствием</p>
</body>
</html>
```

Документ может содержать несколько тегов <script>. Все они последовательно обрабатываются интерпретатором JavaScript. В следующем примере в раздел <body> (в тело) HTML-документа вставлены операторы языка JavaScript.

Переменная – это «именованное хранилище» для данных. Мы можем использовать переменные для хранения товаров, посетителей и других данных.

Для создания переменной в JavaScript используйте ключевое слово let.

Приведённая ниже инструкция создаёт (другими словами, объявляет) переменную с именем «message»:

```
let message;
```

Теперь можно поместить в неё данные (другими словами, определить переменную), используя оператор присваивания =:

```
let message;

message = 'Hello'; // сохранить строку 'Hello' в переменной с именем message
```

Строка сохраняется в области памяти, связанной с переменной. Мы можем получить к ней доступ, используя имя переменной:

```
let message;
message = 'Hello!';

alert(message); // показывает содержимое переменной
```

Для краткости можно совместить объявление переменной и запись данных в одну строку:

```
let message = 'Hello!'; // определяем переменную и присваиваем ей значение

alert(message); // Hello!
```

Мы также можем объявить несколько переменных в одной строке:

```
let user = 'John', age = 25, message = 'Hello';
```

Такой способ может показаться короче, но мы не рекомендуем его. Для лучшей читаемости объявляйте каждую переменную на новой строке.

Многострочный вариант немного длиннее, но легче для чтения:

```
let user = 'John';
let age = 25;
let message = 'Hello';
```

Некоторые люди также определяют несколько переменных в таком вот многострочном стиле:

```
let user = 'John',
  age = 25,
  message = 'Hello';
```

В старых скриптах вы также можете найти другое ключевое слово: var вместо let:

```
var message = 'Hello';
```

Ключевое слово var – почти то же самое, что и let. Оно объявляет переменную, но немного по-другому, «устаревшим» способом.

Пример 1. Вычисление площади треугольника

Необходимо написать сценарий, определяющий площадь прямоугольного треугольника по заданным катетам. Сценарий разместим в разделе <body> HTML-документа (листинг 1).

Листинг 1. Первый сценарий в документе :

```
<HTML>
<HEAD>
<title>Первый сценарий в документе</title>
</HEAD>
<BODY>
<P>Страница, содержащая сценарий.</P>
<script>
<!--
let a=8; h=10 /*Инициализируются две переменные*/
document.write ("Площадь прямоугольного треугольника
равна ", a*h/2,".")
   /*Для формирования вывода используется метод write
   объекта document*/
//-->
</script>
<P>Конец формирования страницы, содержащей сценарий</P>
</BODY>
</HTML>
```

// document.write javascript - выводит данные в месте расположения.

Задания

1. Проверить пример из лабораторной работы.
2. Составить сценарий, в котором вычисляется площадь круга по заданному радиусу.
3. Составить сценарий, вычисляющий гипотенузу по заданным катетам.

ЛАБОРАТОРНАЯ РАБОТА №15
Функция и обработка события

Основным элементом языка JavaScript является функция. Описание функции имеет вид

```
function F (V) {S},
```

где F - идентификатор функции, задающий имя, по которому можно обращаться к функции; V - список параметров функции, разделяемых запятыми; S - тело функции, в нем задаются действия, которые нужно выполнить, чтобы получить результат. Необязательный оператор return определяет возвращаемое функцией значение. Обычно все определения и функции задаются в разделе <head> документа. Это обеспечивает интерпретацию и сохранение в памяти всех функций при загрузке документа в браузер.

В предыдущих примерах пользователю не предоставлялась возможность вводить значения, и в зависимости от них получать результат. Интерактивные документы можно создавать, используя формы. Предположим, что мы хотим создать форму, в которой поля Основание и Высота служат для ввода соответствующих значений. Кроме того, в форме создадим кнопку Вычислить. При щелчке мышью по этой кнопке мы хотим получить значение площади треугольника. Действие пользователя (например, щелчок кнопкой мыши) вызывает событие. События в основном связаны с действиями, производимыми пользователем с элементами форм HTML. Обычно перехват и обработка события задается в параметрах элементов форм. Имя параметра обработки события начинается с приставки on, за которой следует имя самого события. Например, параметр обработки события click будет выглядеть как onclick.

Листинг 1. Реакция на событие Click.

```
<HTML>
<HEAD>
<title>Обработка значений из формы</title>
<script language="JavaScript">
<!--//
function care (a, h)
{
var s=(a*h)/2;
document.write ("Площадь прямоугольного
треугольника равна ",s); return s
}
//-->
```

```
</script>
</HEAD>
<BODY>
<P>Пример сценария со значениями из формы</P>
<FORM name="form1">
Основание: <input type="text"
size=5 name="st1"><hr> Высота: <in-
put type="text" size=5
name="st2"><hr>
<input type="button" value=Вычислить
onClick="care(document.form1.st1.value,          docu-
ment.form1.st2.value)"> /*По клику мыши на кнопке в
функцию care передаются два параметра - содержимое
полей ввода*/
</FORM>
</BODY>
</HTML>
```

При интерпретации HTML-страницы браузером создаются объекты JavaScript. Взаимосвязь объектов между собой представляет иерархическую структуру. На самом верхнем уровне иерархии находится объект windows, представляющий окно браузера. Объект windows является предком или родителем" всех остальных объектов. Каждая страница кроме объекта windows имеет объект document. Свойства объекта document определяются содержимым самого документа: цвет фона, цвет шрифта и т. д. Для получения значения основания треугольника, введенного в первом поле формы, должна быть выполнена конструкция

```
document.forml.stl.value
```

т.е., говоря русским языком (при этом читаем с конца), используем данные value из поля ввода с именем st1 находящегося на форме form1 объекта document.

Пример 2. Вычисление площади квадрата.

Напишем сценарий, определяющий площадь квадрата по заданной стороне. Площадь должна вычисляться в тот момент, когда изменилось значение его стороны. Пусть форма содержит два текстовых поля: одно для длины стороны квадрата, другое для вычисленной площади. Кнопка Обновить очищает поля формы. Площадь квадрата вычисляется при возникновении события change, которое происходит в тот момент, когда значение элемента формы с именем num1 изменилось, и элемент потерял фокус. HTML-код представлен в примере 2.

Листинг 2. Реакция на событие Change

```
<HTML>
<HEAD>
<title>Обработка события Change - изменение значения
элемента</title>
<script>
function srec(obj)
{obj.res.value=obj.num1.value* obj.num1.value}
</script>
</HEAD>
<BODY>
<P>Вычисление площади квадрата</P>
<FORM name="form1">
Сторона: <input type="text" size=7 name="num1" on-
Change="srec(form1)">
<hr>
Площадь: <input type="text" size=7 name="res"><hr>
<input type="reset" value=Обновить>
</FORM>
</BODY>
</HTML>
```

Событие Focus возникает в момент, когда пользователь переходит к элементу формы с помощью клавиши <Tab> или щелчка мыши. Событие "потеря фокуса" (Blur) происходит в тот момент, когда элемент формы теряет фокус. Событие select вызывается выбором части или всего текста в текстовом поле. Например, щелкнув дважды мышью по полю, мы выделим поле, наступит событие select, обработка которого приведет к вычислению требуемого значения.

В языке JavaScript определены некоторые стандартные объекты и функции, пользоваться которыми можно без предварительного описания. Одним из стандартных объектов является объект Math. В свойствах упомянутого объекта хранятся основные математические константы, а его методы можно использовать для вызова основных математических функций.

Выражение y=log x запишется y=Math.log(x).

Задания

1. Проверить примеры из лабораторной работы.

ЛАБОРАТОРНАЯ РАБОТА №16

«Современные формы с помощью CSS3 и валидации HTML5»

Ход выполнения работы:

1. *Формирование представления о функциональности*

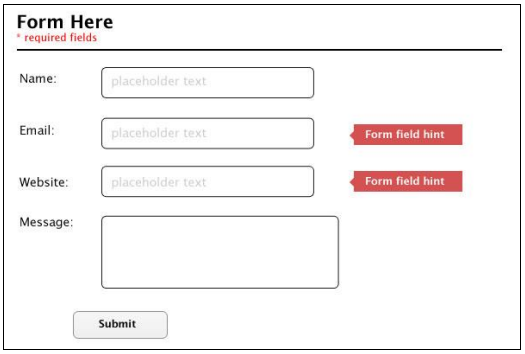
Составим концепцию того, как будет выглядеть наша форма, и как она будет функционировать. Для примера давайте создадим простую контактную форму, запрашивающую у пользователя следующую информацию:

- Имя
- Адрес электронной почты
- Вебсайт
- Сообщение

Нам нужно убедиться, что пользователь вводит информацию правильно. Чтобы этого добиться, применим новые техники валидации на стороне клиента HTML5. А как насчет пользователей, у которых нет возможности использовать HTML5? Можно просто применить валидацию со стороны сервера.

2. *Разработка формы*

Определим, как следует выглядеть нашей форме, создав грубую модель.



Нашу форму составляют следующие элементы:

- Заголовок

Обозначение обязательных для заполнения полей

- Названия полей ввода
- Поля ввода данных

Текст-заполнитель

- Подсказки к полям
- Кнопка «Отправить» (Submit)

Теперь, когда вы известно, какие элементы составляют форму, можно создадим разметку HTML.

3. Первичный код HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>HTML5 Contact Form</title>
        <link rel="stylesheet" media="screen" href="styles.css" >
    </head>
    <body>
    </body>
</html>
```

Вплоть до этого момента HTML-файл в браузере пока пуст. Это – просто начальный код для страницы HTML5.

4. Форма HTML

Давайте создадим форму HTML (оставим метод action пока пустым, так как проверка данных на стороне сервера в этой лабораторной работе не раскрывается):

```
<form class="contact_form" action="" method="post" name="contact_form">
</form>
```

5. Элементы формы HTML

Для получения организованного и структурированного контента своей формы, обернем ее элементы (label, input и т.д.) в список. Создадим заголовок формы и первого элемента input:

```
<ul>
<li>
    <h2>Contact Us</h2>
    <span class="required_notification">* Denotes Required Field</span>
</li>
<li>
    <label for="name">Name:</label>
    <input type="text" name="name" />
</li>
</ul>
```

Подсказки для полей формы

Как видно из макета, необходимо сделать форматированные подсказки для полей ввода электронного адреса “email” и вебсайта “website”. Поэтому добавим свои подсказки под поля ввода, где это нужно, и назначим им класс, чтобы можно было позже определить им стили.

```
<li>
<label for="email">Email:</label>
<input type="text" name="email" />
<span class="form_hint">Proper format "name@something.com"</span>
</li>
```

Остальные элементы input

Создадим остальные элементы формы, не забывая о том, что нужно обернуть каждый раздел в элемент списка.

```
<li>
<label for="website">Website:</label>
<input type="text" name="website" />
<span class="form_hint">Proper format "http://someaddress.com"</span>
</li>
<li>
<label for="message">Message:</label>
<textarea name="message" cols="40" rows="6" ></textarea>

</li>
<li>
<button class="submit" type="submit">Submit Form</button>
</li>
```

6. Добавляем атрибут placeholder

Одно из первых усовершенствований, которые предлагает HTML5 для веб-форм (с которым вы, возможно, уже знакомы) – это способность установить текст-подсказку. Он показывается, когда поле ввода либо пустое, либо находится не в фокусе.

Добавим атрибут placeholder для всех текстовых тегов input. Это поможет

пользователю понять, что нужно ввести в каждое поле.

```
<input type="text" name="name" placeholder="John Doe" />
<input type="text" name="email" placeholder="john_doe@example.com" />
<input type="text" name="website" placeholder="http://johndoe.com/" required/>
```

Подсказка: Назначьте placeholder'у стили

Вот вам подсказка: если нужно определить стили тексту-подсказке, к вашим услугам имеются префиксы браузеров:

```
-moz-placeholder {
    color: blue;
}
 ::-webkit-input-placeholder {
    color: blue;
}
```

В современных браузерах поддержка атрибута placeholder налажена довольно хорошо (кроме IE9, к сожалению). Если вам реально требуется поддерживать его во всех браузерах, можно посмотреть решение проблемы в javascript.

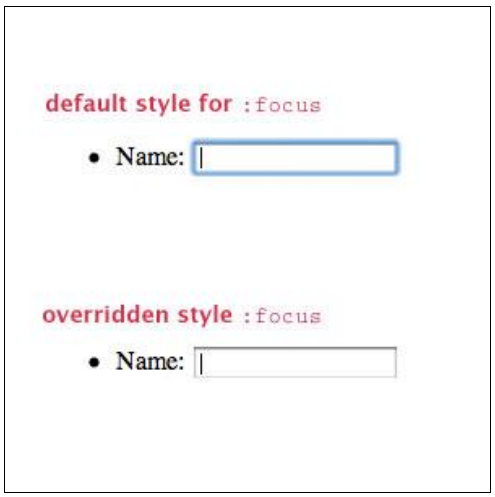
7. Основной CSS

Чтобы создать для формы некую структуру, добавим немного основного CSS.

Удалите стиль : focus

Webkit автоматически добавляет к input-ам стили, когда те находятся в фокусе. Так как мы будем добавлять собственные стили, то стили по умолчанию нужно отменить:

1.*:focus {outline: none;}



Типографские стили

Давайте определим элементам своей формы типографские стили:

```
body {font: 14px/21px "Lucida Sans", "Lucida Grande", "Lucida Sans Unicode", sans-serif;}
.contact_form h2, .contact_form label {font-family:Georgia, Times, "Times New Roman", serif;}
.form_hint, .required_notification {font-size: 11px;}
```

Стили списка

Давайте назначим стили элементам списка, чтобы придать форме структуру:

```
.contact_form ul {
    width:750px;
    list-style-type:none;
    list-style-position:outside;
    margin:0px;
    padding:0px;
}
.contact_form li{
    padding:12px;
    border-bottom:1px solid #eee;
    position:relative;
}
```

Кроме того, добавим тонкую рамку к верхнему и нижнему разделам формы. Выполнить это можно, применив селекторы :first-child и :last-child. Они выбирают, как подразумевается в их названиях, первый и последний элементы списка ul.

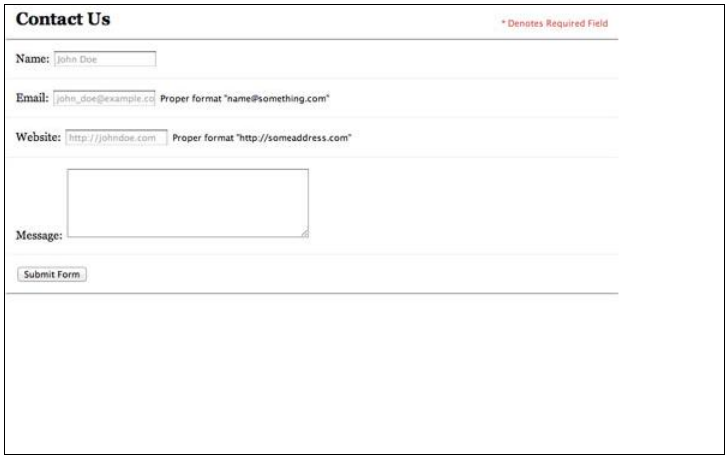
Так создается визуальное разделение формы. Помните, что эти селекторы CSS не поддерживаются старыми браузерами. Так как для основной функциональности это не жизненно важно, то в выгодном положении окажутся те, кто пользуется современными браузерами.

```
.contact_form li:first-child, .contact_form li:last-child {
border-bottom:1px solid #777;
}
```

Заголовок формы

Назначим стили разделу заголовка своей формы. Он включает тэг заголовка и уведомление, которое информирует пользователей о том, что звездочка (*) обозначает поля, обязательные для заполнения.

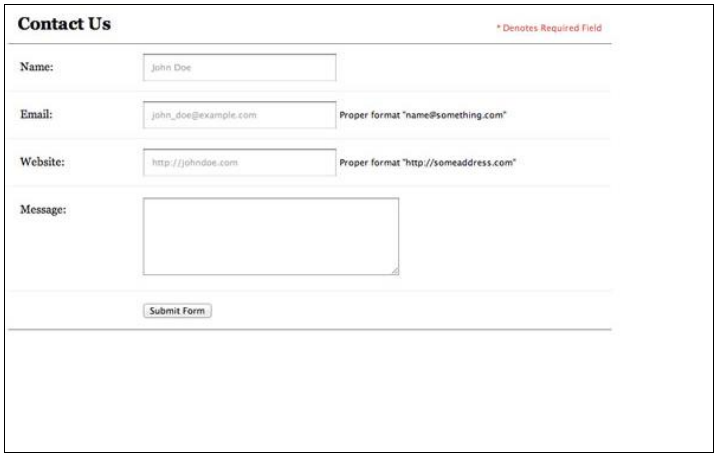
```
.contact_form h2 {
    margin:0;
    display: inline;
}
.required_notification {
    color: #d45252;
    margin:5px 0 0 0;
    display:inline;
    float:right;
}
```



Элементы input

Назначим стили основным элементам формы, тем, которые предназначены для сбора пользовательской информации.

```
.contact_form label {
width:150px;
margin-top: 3px;
display:inline-block;
float:left;
padding:3px;
}
.contact_form input {
height:20px;
width:220px;
padding:5px 8px;
}
.contact_form textarea {padding:8px; width:300px;}
.contact_form button {margin-left:156px;}
```



Теперь добавим несколько дополнительных визуальных стилей CSS. Некоторые из них видны только пользователям современных браузеров.

```
.contact_form input, .contact_form textarea {
    border:1px solid #aaa;
    box-shadow: 0px 0px 3px #ccc, 0 10px 15px #eee inset;
    border-radius:2px;
}
.contact_form input:focus, .contact_form textarea:focus {
    background: #fff;
    border:1px solid #555;
    box-shadow: 0 0 3px #aaa;
}
/* Button Style */
button.submit {
    background-color: #68b12f;
    background: -webkit-gradient(linear, left top, left bottom, from(#68b12f), to(#50911e));
    background: -webkit-linear-gradient(top, #68b12f, #50911e);
    background: -moz-linear-gradient(top, #68b12f, #50911e);
    background: -ms-linear-gradient(top, #68b12f, #50911e);
    background: -o-linear-gradient(top, #68b12f, #50911e);
    background: linear-gradient(top, #68b12f, #50911e);
    border: 1px solid #509111;
    border-bottom: 1px solid #5b992b;
    border-radius: 3px;
    -webkit-border-radius: 3px;
    -moz-border-radius: 3px;
    -ms-border-radius: 3px;
    -o-border-radius: 3px;
    box-shadow: inset 0 1px 0 0 #9fd574;
    -webkit-box-shadow: 0 1px 0 0 #9fd574 inset ;
    -moz-box-shadow: 0 1px 0 0 #9fd574 inset;
    -ms-box-shadow: 0 1px 0 0 #9fd574 inset;
    -o-box-shadow: 0 1px 0 0 #9fd574 inset;
```

```
    color: white;
    font-weight: bold;
    text-align: center;
    text-shadow: 0 -1px 0 #396715;
}
button.submit:hover {
    opacity:.85;
    cursor: pointer;
}
button.submit:active {
    border: 1px solid #20911e;
    box-shadow: 0 0 10px 5px #356b0b inset;
    -webkit-box-shadow:0 0 10px 5px #356b0b inset ;
    -moz-box-shadow: 0 0 10px 5px #356b0b inset;
    -ms-box-shadow: 0 0 10px 5px #356b0b inset;

    -o-box-shadow: 0 0 10px 5px #356b0b inset;
}
```

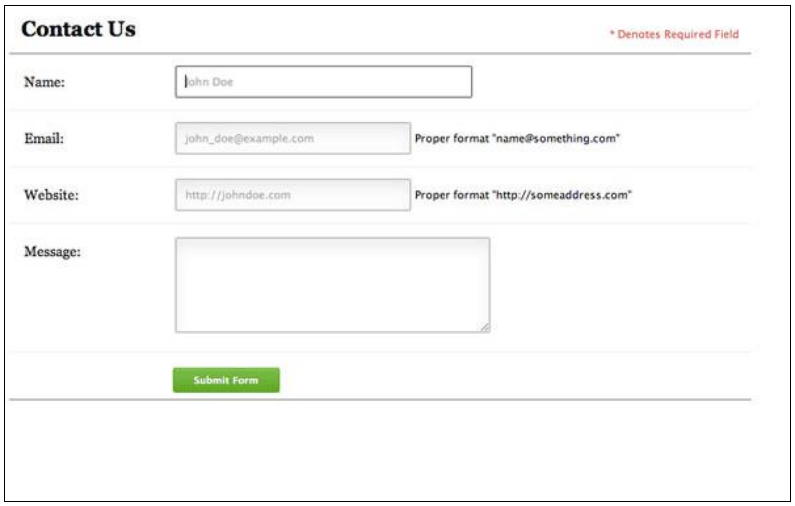
8. Добавляем интерактивность с помощью CSS3

Давайте добавим немного интерактивности. Заставим выбранное в данный момент поле увеличиваться путем добавления отступа.

```
.contact_form input:focus, .contact_form textarea:focus { /* add this to the already existing style */
padding-right:70px;
}
```

Теперь с помощью CSS3 для поддерживающих браузеров сделаем увеличение поля гладким переходом.

```
.contact_form input, .contact_form textarea { /* add this to the already existing style */
    -moz-transition: padding .25s;
    -webkit-transition: padding .25s;
    -o-transition: padding .25s;
    transition: padding .25s;
}
```



9. Атрибут required в HTML5

Теперь пора заняться инструментами управления формой HTML5.

Добавление атрибута required в любой элемент области ввода текста скажет браузеру, что перед отправкой формы требуется ввести значение. Таким образом,

форму нельзя отправить, если нужное поле осталось незаполненным.

Итак, продолжим и добавим атрибут required во все элементы своей формы (потому что нам нужно, чтобы они все были заполнены).

```
<input type="text" name="name" required />
<input type="text" name="email" required />
<input type="text" name="website" required />
<textarea name="message" cols="40" rows="6" required ></textarea>
```

10. Стили обязательных для заполнения полей

Вы, возможно, заметите, что в визуальном плане от добавления атрибута required ничего не изменилось. Мы собираемся назначить стили обязательных для заполнения полей с помощью CSS. В этом примере каждое поле в качестве фонового изображения получит красную звездочку. Чтобы это сделать, нам нужно сначала добавить padding справа input-a, где будет находиться фоновое изображение (так будет предотвращено наложение текста, если запись окажется длинной строкой):

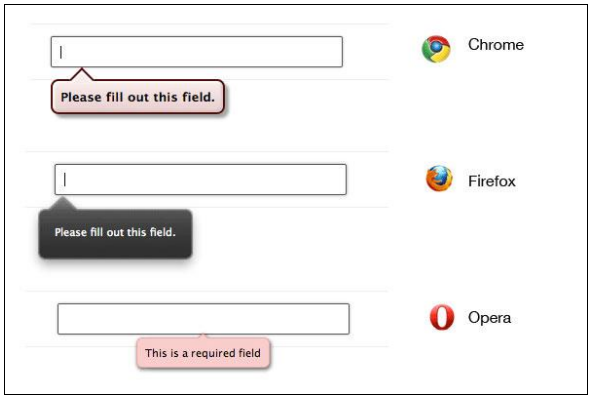
```
.contact_form input, .contact_form textarea {
padding-right:30px;
}
```

Теперь воспользуемся псевдоселектором CSS :required для того, чтобы выбрать все элементы формы с нужным атрибутом. В папке ***images*** есть иконка с красной звездочкой размером 16×16 px, которая послужит визуальным индикатором поля, обязательного к заполнению.

```
input:required, textarea:required {
    background: #fff url(images/red_asterisk.png) no-repeat 98% center;
}
```

Что происходит при отправке формы?

Прямо сейчас при отправке формы с использованием элементов HTML5 различные браузеры действуют по-разному. Большая часть браузеров воспрепятствует отправке и покажет пользователю подсказку, отмечающую первое обязательное к заполнению поле, в которое не введено значение. Визуальные стили и поддержка таких полей довольно широки. К счастью, в будущем такое поведение будет стандартизировано.



Подсказка:

На самом деле в webkit назначить стили сообщению в поле-«пузыре» можно с помощью следующего:

```
::-webkit-validation-bubble-message {
    padding: 1em;
}
```

11. Понимание новых атрибутов type и валидации на стороне клиента в HTML5

Валидация HTML5 работает в соответствии с атрибутом type, установленном в полях формы. Годами HTML поддерживал лишь несколько атрибутов type, таких, как type= «text», но у HTML5 имеется более дюжины новых типов ввода данных, включая электронную почту и url, которые мы и собираемся использовать в своей форме.

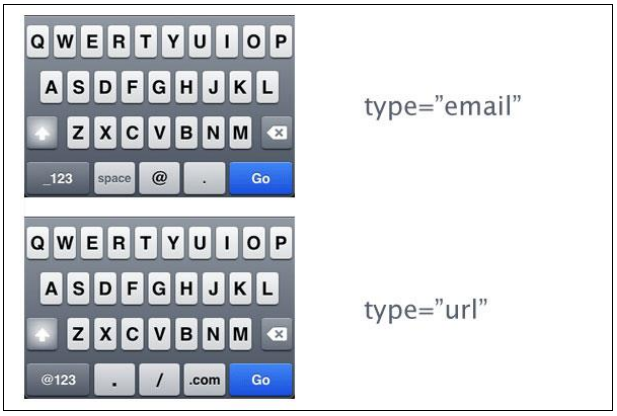
Сочетая атрибуты ввода type с новым атрибутом required, теперь браузер способен делать валидацию данных пользователя на клиентской стороне. Если браузер пользователя не поддерживаем новые атрибуты type, такие как type= «email», он просто по умолчанию вернется к type= «text».

Так что происходит, если браузер на самом деле поддерживает новые атрибуты type? Для браузеров десктопов визуально нет никакой разницы (кроме определенной пользовательскими правилами CSS). Поле type= «text» выглядит точно так, как поле type= «email». Однако для браузеров мобильных устройств в отношении пользовательского интерфейса разница имеется.

Пример: iPhone

iPhone от Apple’a распознает типы формы и динамично изменяет экранную клавиатуру, предоставляя требуемые по контексту символы. Например, все

электронные адреса требуют следующих символов: “@” и “.” И iPhone предоставляет эти символы, когда у input-а задан соответствующий тип.



12. Изменение атрибутов type

Поля нашей формы уже установлены по умолчанию на type= «text». Но теперь требуется изменить атрибут типа в полях, предназначенных для электронной почты и вебсайта на соответствующий тип HTML5.

```
<input type="email" name="email" placeholder="john_doe@example.com" required />
<input type="url" name="website" placeholder="http://johndoe.com" required/>
```

13. Валидация HTML5

Как уже говорилось, валидация HTML5 основана на атрибутах type и включена по умолчанию. Для активации валидации формы никакой особенной разметки не требуется. Если хотите ее выключить, можете употребить атрибут novalidate, как здесь:

```
<form novalidate>
<-- do not validate this form -->
<input type="text" />
</form>
```

Поле имени

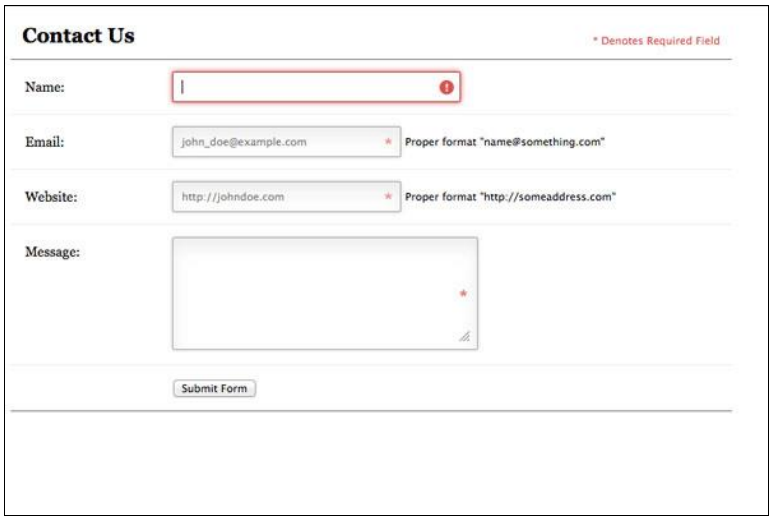
Взгляните на первое поле, которое запрашивает у пользователя его имя. Как уже описывалось ранее, мы добавили атрибут type= «text» и атрибут required. Они информируют веб-браузер о том, что это обязательное поле и тот должен делать его валидацию как простого текста. Так что, если пользователь введет в поле по меньшей мере один символ, оно будет достоверным.

Теперь мы создадим свой собственный CSS для назначения стилей полям ввода, считающимся браузером достоверными и недостоверными. Если помните, мы использовали :required в своем CSS для определения стилей всем элементам ввода с обязательным атрибутом. Теперь нам можно назначить стили

обязательным к заполнению полям, достоверным или нет, добавив в правила CSS :valid или :invalid.

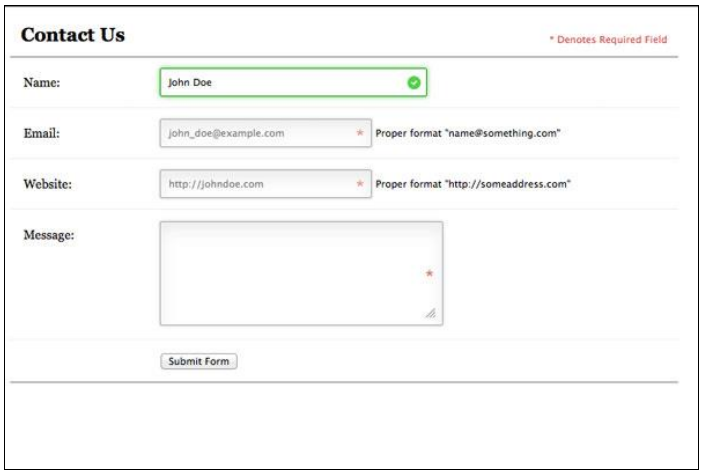
Во-первых, давайте назначим стили недостоверным полям. В этом примере нам нужно определить стили формы как невалидные, когда та находится в фокусе. Мы добавим красную окантовку, красную тень и созданную красную иконку (images/invalid.png) для обозначения невалидного поля.

```
.contact_form input:focus:invalid, .contact_form textarea:focus:invalid { /* when a field is considered invalid by the browser */
    background: #fff url(images/invalid.png) no-repeat 98% center;
    box-shadow: 0 0 5px #d45252;
    border-color: #b03535;
}
```



Теперь создадим правила, которые обозначат достоверное поле. Добавим зеленую окантовку, зеленую тень и зеленую иконку с «галочкой» (images/valid.png). Они будут применены ко всем валидным полям, независимо от того, находятся те в фокусе или нет.

```
.contact_form input:required:valid, .contact_form textarea:required:valid { /* when a field is considered valid by the browser */
    background: #fff url(images/valid.png) no-repeat 98% center;
    box-shadow: 0 0 5px #5cd053;
    border-color: #28921f;
}
```



Сейчас, когда мы активизируем поле формы, видны стили в красном цвете, означающие невалидность. Как только в поле вводится хотя бы один символ,

оно становится валидным, и отражают этот факт появляющиеся зеленые стили CSS.

Поля электронного адреса и URL'а

Стили CSS и правила валидации уже применены к полю электронного адреса, так как ранее мы уже установили атрибуты type и required.

14. Представляем атрибут HTML5 pattern

Если для примера использовать атрибут type= «email», то похоже, что большинство браузеров делают валидацию этого поля как *@* (любой символ + символ "@" + любой символ). Это, конечно, не очень ограничивает, однако предотвращает введение пользователем пробелов или совершенно неправильных значений.

В примере атрибута type= «url», похоже, что минимальные требования большинства браузеров – это символ, за которым следует двоеточие. Поэтому, если вы ввели "h:", то поле считается валидным. Это не очень-то помогает, но действительно предотвращает введение пользователями неподходящей информации, такой как их электронный или домашний адрес. Рассмотрим, как это сделать в HTML5.

Атрибут pattern

Атрибут pattern принимает регулярное выражение javascript'a. Это выражение, а не выражение браузера по умолчанию, применяется для валидации значения поля. Так что теперь наш HTML выглядит так:

```
<input type="url" name="website" placeholder="http://johndoe.com" required pattern="(http|https)://.+" />
```

Теперь наше поле будет принимать только значения, начинающиеся с "http://" или "https://" и один дополнительный символ.

15. Подсказки к полям формы (CSS)

Теперь давайте определим стили подсказок к своей форме, говорящих пользователю формат, который следует использовать при вводе информации.

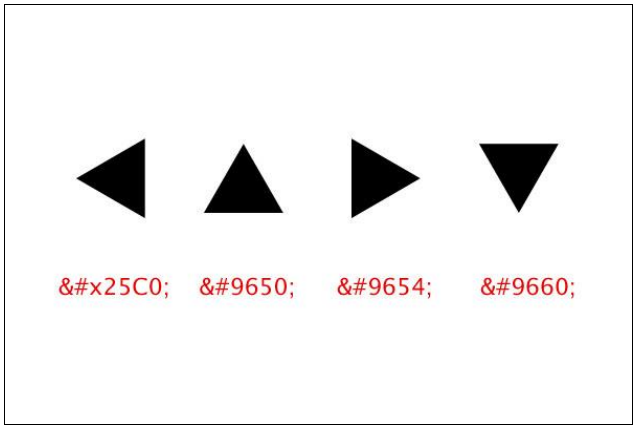
```
.form_hint {
    background: #d45252;
    border-radius: 3px 3px 3px 3px;
    color: white;
    margin-left:8px;
    padding: 1px 6px;
    z-index: 999; /* hints stay above all other elements */
    position: absolute; /* allows proper formatting if hint is two lines */
    display: none;
}
```

Мы устанавливаем display:none, потому что нам нужно показывать подсказки только тогда, когда пользователь фокусируется на поле ввода. Мы также по умолчанию устанавливаем всплывающие подсказки на красный цвет недостоверности, так как они всегда считаются невалидными, пока в них не будет введена правильная информация.

Применение селектора ::before

Теперь нам нужно добавить к блокам подсказок маленький треугольник, который поможет направлять взгляд и руководить им. Это можно делать с использованием изображений, но в данном случае мы сделаем все с помощью одного лишь CSS.

Так как это – чисто презентационный элемент, который не является жизненно важным для функциональности страницы, то добавим маленький треугольник, указывающий влево, при помощи псевдоселектора ::before. Это можно сделать с помощью одной из геометрических фигур unicode’a.



Обычно мы бы применили формат HTML Unicode, чтобы отобразить их в своем HTML (как на изображении вверху). Однако из-за использования селектора CSS ::before нам придется взять соответствующий треугольнику escaped unicode с применением правила content: «». Затем просто применим позиционирование, чтобы поместить его в нужное место.

```
.form_hint::before {
content: "\25C0"; /* треугольник, указующий влево, в  escaped unicode */
color: #d45252;
position: absolute;
top:1px;
left:-6px;
}
```

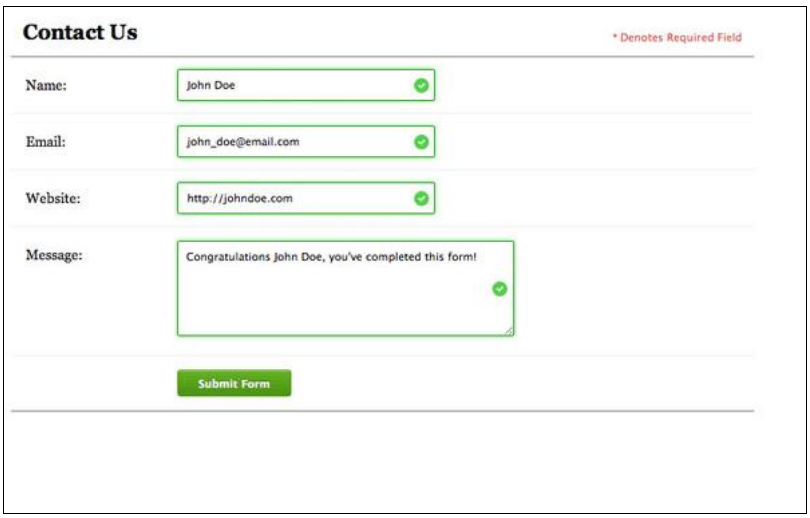
Применение смежного селектора +

Наконец, мы используем смежный селектор CSS для показа и скрытия подсказок к нашим полям формы. Смежный селектор (x + y) выбирает элемент, которому непосредственно предшествует предыдущий элемент. Так как подсказки для полей идут в нашем HTML сразу после полей ввода, можно применять этот селектор для показа/скрытия всплывающих подсказок.

```
.contact_form input:focus + .form_hint {display: inline;}
.contact_form input:required:valid + .form_hint {background: #28921f;} /* change form hint color when valid */
.contact_form input:required:valid + .form_hint::before {color: #28921f;} /* change form hint arrow color when valid */
```

Как видно из CSS, мы также установили подсказки к форме, чтобы те меняли цвет наряду с рамкой ввода, когда поле валидно или невалидно.

Итог:



ЛАБОРАТОРНАЯ РАБОТА №15

1. Проектирование структуры и информационного содержания сайта

Порядок выполнения:

- Выделить основные сайты и порталы, посвященные заданной теме.
- Подобрать информацию по теме сайта.
- Подобрать иллюстрации и инфографику. Для каждой иллюстрации подобрать название.
- Составить список использованных источников.
- Выделить ключевые слова. Проанализировать выборки поисковых систем на ключевые слова.
- Структурировать собранную информацию: составить список разделов сайта, выделить подразделы.
- Разбить информацию на отдельные статьи. Для каждой статьи выбрать название, ключевые слова, аннотацию, url.

- В отчете:

- Тема сайта
- Список разделов и подразделов сайта
- Структура информационного наполнения сайта (полностью) с разбивкой по статьям.

2. Методические указания по выполнению лабораторной работы

Выбор темы сайта

Указания. На первом этапе необходимо выбрать название сайта. Оно должно быть, как минимум:

• отражающим тему сайта,

• коротким _____(не более 7 слов),

• не занятым (отсутствует сайт с таким же названием).

Пример.

Тема сайта: Донорство в Ростове-на-Дону

3. Анализ информационной структуры в Интернете

Указания. Требуется проанализировать, какая информация уже имеется в Интернете. Для этого необходимо:

• найти аналогичные сайты (сайты-конкуренты, посвященные той же теме или сайты, близкие по теме) 5-10 шт;

• исследовать, как освещена тема в социальных сетях и сообществах (здесь можно привести также специализированные форумы или разделы форумов);

Пример.

Крупные порталы:

- Доноры.ру (http://www.donori.ru)
- Служба крови (http://www.yadonor.ru/)
- Доноры -детям (http://www.donors.ru/)
- Доноры крови (http://www.donor-krovi.ru/)

- Донор.ру (http://www.donor.ru)
- Седьмой лепесток (http://www.7lepestok.ru)

Аналогичные сайты:

- Река жизни (Нижний Новгород) (http://www.donori.ru)
- СЛУЖБА КРОВИ г. САНКТ-ПЕТЕРБУРГА

(http://www.transfusion.spb.ru)

- Саратовский центр крови (http://bloodsar.ru/)
- Красноярский центр крови (http://www.kkck.ru/)
- Новосибирский центр крови (http://nck.su/)
- Донорство крови в Мордовии (http://donor.3dn.ru /)
- Станция переливания крови в Москве
- (http://www.spkdzm.ru/)

В социальных сетях:

Живой журнал:

- ЖЖ «Я - донор» http://community.livejournal.com/yadonor_ru
- ЖЖ движения донорства в Нижнем Новгороде

http://community.livejournal.com/nnov_donors

Facebook :

- Группа «Донор»

http://www.facebook.com/group.php?gid=321902445820

ВКонтакте:

- Группа «Поиск доноров» http://vkontakte.ru/club21179173
- "Кронштадтский десант" - группа доноров из Кронштадта
- http://vkontakte.ru/club14761218

Форумы:

- Тема на форуме новосибирского академгородка. Люди предлагают

сдать кровь, указывая свои телефоны

http://forum.academ.org/index.php?showtopic=209763

- Тема донорства на официальном сайте правительства Москвы

http://forum.mos.ru/viewtopic.php?f=8&t=8541

4. Подбор ключевых слов

Указания. Ключевые слова – это поисковый образ ваших документов. Ключевые слова - это один из факторов, учитываемых поисковыми системами в процессе работы.

Составьте список всех слов и фраз, которыми вы бы воспользовались для нахождения информации, товаров и/или услуг, аналогичных вашим. Старайтесь не использовать одиночные ключевые слова. Это связано со следующими причинами:

• Во-первых, одиночное слово скорее всего попадает под сильную конкуренцию с другими сайтами. Набрав в строке поиска "туризм" или "путешествия" вы получите сотни тысяч страниц.

• Во-вторых, поскольку число страниц по одиночному ключевому слову может стремительно расти, большинство пользователей поисковых систем понимают, что они получат более точные результаты, используя два и более слова. Более того, статистические исследования показывают, что таких пользователей становится все больше и больше.

• В-третьих, одиночные ключевые слова не принесут вам целевой траффик. Например, когда кто-то ищет «донорство», он/она не обязательно хочет найти «донорство в Ростове-на-Дону». Даже, если вам удастся добиться высокой позиции в поисковой системе по одному ключевому слову, вы можете ничего не получить от прироста количества таких посетителей.

При подборе ключевых слов для реального сайта старайтесь учесть географический аспект. В лабораторной работе это проблематично, но так же приветствуется.

Пример.

Первичный набор ключевых слов для всего сайта:

• Донорство в Ростове-на-Дону

• Станция переливания крови

• Как сдать кровь

5. Анализ поисковой статистики и подбор синонимов

Указания. Анализ поисковой статистики преследует две цели:

• Оценить количество информации, имеющейся в киберпространстве по заданной теме;

• Подобрать ключевые слова (особенно важно для коммерческих сайтов).

Анализ поисковой статистики производится по двум показателям:

• Количество конкурирующих сайтов (для этого достаточно проанализировать количество документов, которые находит поисковая система по ключевым словам);

• Количество запросов в месяц по ключевому слову.

Существует несколько способов подобрать хорошие ключевые слова к сайту и отдельным его статьям. Рассмотрим самый простой. Он основывается на использовании инструментов анализа статистики поисковых машин:

• Google: https://adwords.google.com

• Яндекс: http://wordstat.yandex.ru

Далее приведен пример использования инструмента подбора ключевых слов на Яндексе.

Пример.

Количество сайтов по ключевым словам

ЯНДЕКС GOOGLE MAIL.RU

ДОНОРСТВО В РОСТОВЕ-НА-ДОНУ	106000	73 000	2080
СТАНЦИЯ ПЕРЕЛИВАНИЯ КРОВИ	1000000	231 000	68300
КАК СДАТЬ КРОВЬ	6000000	970 000	154000

(В примере чётко видно, что конкуренция по слову с географической конкретизацией (Ростов-на-Дону) значительно ниже)

Подбор близких ключевых слов и синонимов

Станция переливания крови

СЛОВА ПОКАЗОВ В МЕСЯЦ	
СТАНЦИЯ ПЕРЕЛИВАНИЯ КРОВИ	11693
ОБЛАСТНАЯ СТАНЦИЯ ПЕРЕЛИВАНИЯ КРОВИ	1041
ГОРОДСКАЯ СТАНЦИЯ ПЕРЕЛИВАНИЯ КРОВИ	431
ГУЗ СТАНЦИЯ ПЕРЕЛИВАНИЯ КРОВИ	168
АДРЕСА СТАНЦИЙ ПЕРЕЛИВАНИЯ КРОВИ	162
СТАНЦИЯ ПЕРЕЛИВАНИЯ КРОВИ САЙТ	158
СТАНЦИЯ ПЕРЕЛИВАНИЯ КРОВИ РЕЖИМ РАБОТЫ	42

СЛОВА ПОКАЗОВ В МЕСЯЦ	
СДАТЬ КРОВЬ +ЗА ДЕНЬГИ	2564

СДАТЬ КРОВЬ	34869
СДАЧА КРОВИ	17987

(Далее можно переходить, по ключевым словам, подбирая новые синонимы. В отчёте можно привести только список найденных синонимов и статистику просмотров за месяц по ним)

6. Анализ целевой аудитории сайта

Указания. Целевая аудитория – это посетители, которым будет интересна информация на вашем сайте (для информационных сайтов) или возможные покупатели ваших товаров и услуг (для коммерческого сайта). Чтобы найти эффективные способы привлечения целевой аудитории, нужно составить примерный портрет целевого посетителя. Он составляется из нескольких характеристик:

• возраст;

• пол;

• сфера деятельности, образование, отрасль работы;

• достаток;

• насколько часто и откуда пользователь выходит в сеть, какие сайты чаще всего посещает.

Для примера давайте сравним общение на форуме программистов и автомобилистов:

Не визуальные компоненты не являются потомками классов оконных компонент, поэтому никаких сообщений Windows в них вообще не поступает и что-либо перехватывать бессмысленно.

У меня 3 раз за полгода летит ступичный подшипник. "Автомобильные знатоки" говорят, что кулак 100% плохой, потому что имеет яйцевидную форму.

Чтобы наиболее качественно проанализировать целевую аудиторию необходимо составить демографический профиль её типичного представителя.

Демографический профиль учитывает род занятий, возраст, пол, частоту работы и интересы в Сети (какие сайты посещают пользователи и почему, как часто они совершают онлайновые покупки, насколько хорошо они ориентируются в сети). Большинство сайтов посещает несколько четко различающихся категорий пользователей, поэтому может потребоваться создание несколько общих профилей.

• Опишите типичного пользователя вашего сайта. Как часто он работает в режим онлайн и для чего вообще использует Сеть? Каков его возраст и чем он зарабатывает себе на жизнь?

• Какова основная цель посещения вашего сайта (сделать покупки, вступить в члены сообщества, найти информацию)?

• По каким главным причинам целевой пользователь выбирает продукцию и/или услуги вашей компании (цена, сервис, качество)?

Пример описания пользователя для книжного интернет-магазина:

"Типичный пользователь - это студент университета в возрасте от 18 до 22 лет, работающий в Сети ежедневно. Он очень хорошо ориентируется в Сети и регулярно (2 или 3 раза в месяц) делает онлайновые покупки книг, компакт-дисков, цифровых видеодисков и подарков. Он имеет высокоскоростной доступ в Интернет из общежития и из библиотеки, чаще всего используя библиотечные компьютеры для учебных задач, а компьютер в общежитии - для личной переписки. Его типичными задачами на сайте являются поиск авторов, названий

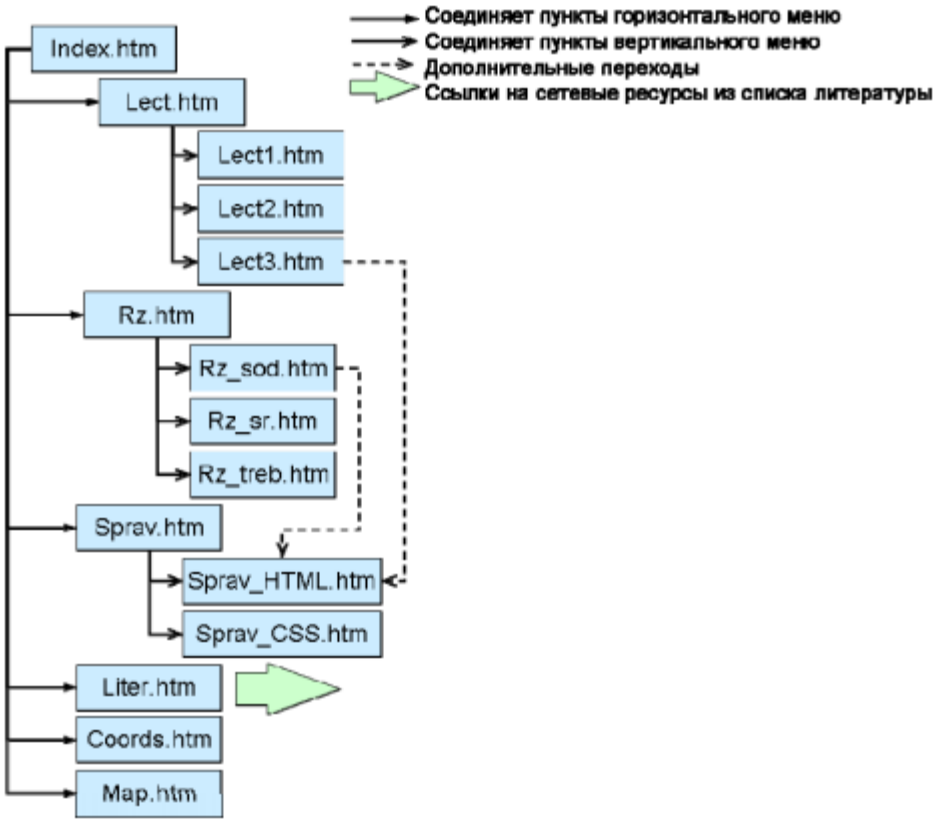
и товаров для совершения покупок. Он зарегистрирован на сайте, имеет имя пользователя и пароль и может делать покупки быстро и легко ".

7. Разработать карту сайта

Указания. Карта сайта – это карта переходов между страницами сайта. В простейшем случае это список разделов и подразделов сайта. Если вы решили не делать никаких других переходов на вашем сайте, то этот пункт считается невыполненным, и положительные бонусы за него не начисляются.

Пример.

Карта сайта, посвященного дисциплине «Киберпространство»:



8. Выбрать элементы навигации

Указания. Навигация – это перемещение по сайту. Система навигации сайта - одна из важнейших составляющих понятия " дизайн сайта". Список элементов навигации см. в лекционном материале.

Пример.

Элементы навигации для сайта, посвященного дисциплине «Киберпространство»

Основные элементы навигации:

• Горизонтальное меню для навигации по составляющим учебно-методического комплекса (лекции, расчётное задание, литература и проч.)

• Вертикальное меню для навигации по разделам лекционного материала и методическим указаниям .

Дополнительные элементы навигации:

• «Хлебные крошки»

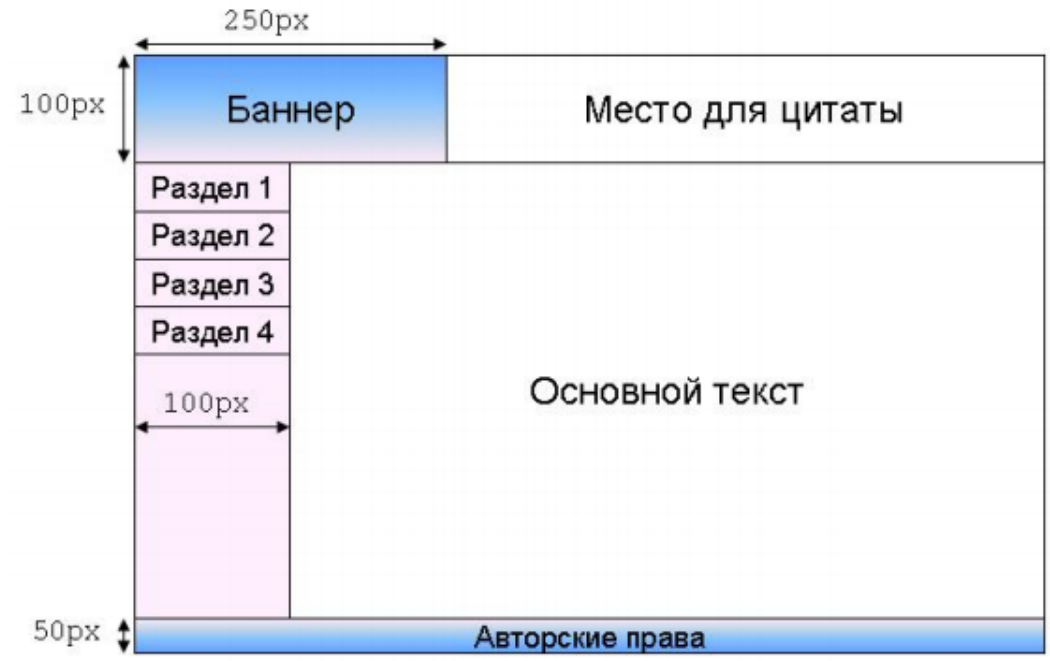
• Переход на главную страницу под баннером

• Карта сайта

9. Разработать макеты страниц сайта

Указания. Дизайн-макет (или просто макет) – это рисунок, представляющий предполагаемый будущий внешний вид страниц сайта. Макеты бывают с фиксированной шириной, или с плавающей шириной. Тип макета следует указывать в отчёте. Подробнее см. в лекционном материале.

Пример.
Макет с плавающей шириной.



10.Отчет

Указания. Необходимо привести всю подобранную по теме информацию в структурированном виде. Выделить разделы и подразделы. Для каждой структурной единицы от раздела до статьи выбрать название и ключевые слова, написать аннотацию и по возможности подобрать иллюстрации. Ко всем иллюстрациям подобрать подписи.

Далее приведен пример оформления одной статьи. Для лабораторной требуется количество материала, достаточное для раскрытия темы.