Практическое занятие №4

Блочная модель и макетирование

Задание №1

Проверьте запущен ли у Вас Open Server. Папка с материалами урока (css4.loc) должна быть размещена в директории /domains сервера. Заходим на http://css4.loc и начинаем урок.

Задание №2

Свойство ***display*** определяет, как элемент будет выглядеть в документе. Это свойство имеет множество различных значений. Для нас сейчас главные три значения:

- ***block*** - элемент показывается как блочный.
- ***inline*** - элемент отображается как строчный.
- ***inline-block*** – это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно строчному элементу

Открываем файл *index-2.html* и *style.css* в папке задания. Создаём три универсальный блочных элемента , как показано ниже.

```
<div class="elem div1">Элемент 1</div>
<div class="elem div2">Элемент 2</div>
<div class="elem div3">Элемент 3</div>
```

Зададим этим элементам цвет фона, для того что видеть их размер и расположение на странице.

```
.div1 {
    background-color: red;
}

.div2 {
    background-color: navy;
}

.div3 {
    background-color: green;
}
```

Элемент изначально блочный. Как Вы сами видите блочные элементы по умолчанию имеют высоту, зависящую от содержимого этого элемента, в нашем случае это текст внутри блоков. По умолчанию блочный элемент занимает 100% ширины родительского элемента.

Для того что бы изменять размеры этих элементов существуют свойства ***width***(определяет ширину элемента) и ***height*** (определяет высоту элемента).

Определим высоту и ширину для наших элементов, как показано ниже.

```
.elem {
    width: 200px;
    height: 100px;
}
```

Как Вы можете видеть блочные элементы идут один под другим. Мы можем сделать вывод, что несмотря на то что блочный элемент мы ограничили по ширине, все равно для него была отведена вся ширина родительского элемента.

С помощью свойства display превратим наши блоки в строчные элементы.

```
.elem {
    width: 200px;
    height: 100px;
    display: inline;
}
```

Что мы наблюдаем? Как изменилось поведение элементов?

Строчные элементы имею ряд особенностей:

- они следуют друг за другом
- свойства width и height игнорируются, размер элементов определяется их содержимым.
- примером строчного элемента является тег <a>, который мы многократно использовали.
- - универсальный строчный тег, используется для разметки.

Далее превратим наши элементы в блочно-строчные, как показано ниже.

```
.elem {
    width: 200px;
    height: 100px;
    display: inline-block;

}
```

Как Вы можете видеть блочно-строчные элементы — это гибридные элементы, которые объединили в себе свойства как блочных, так и строчных элементов. На них действуют свойства width и height как на блочные элементы, так же они располагаются друг за другом как строчные элементы.

Внимание ЗАСАДА!

Как Вы у себя видите, что между блоками есть зазоры. Это связано с тем, что браузеры видят эти элементы как СТРОКИ и разделяют их. Что бы в этом убедиться увеличим размер шрифта для элемента <body>.

```
body {
    font-size: 70px;
}
```

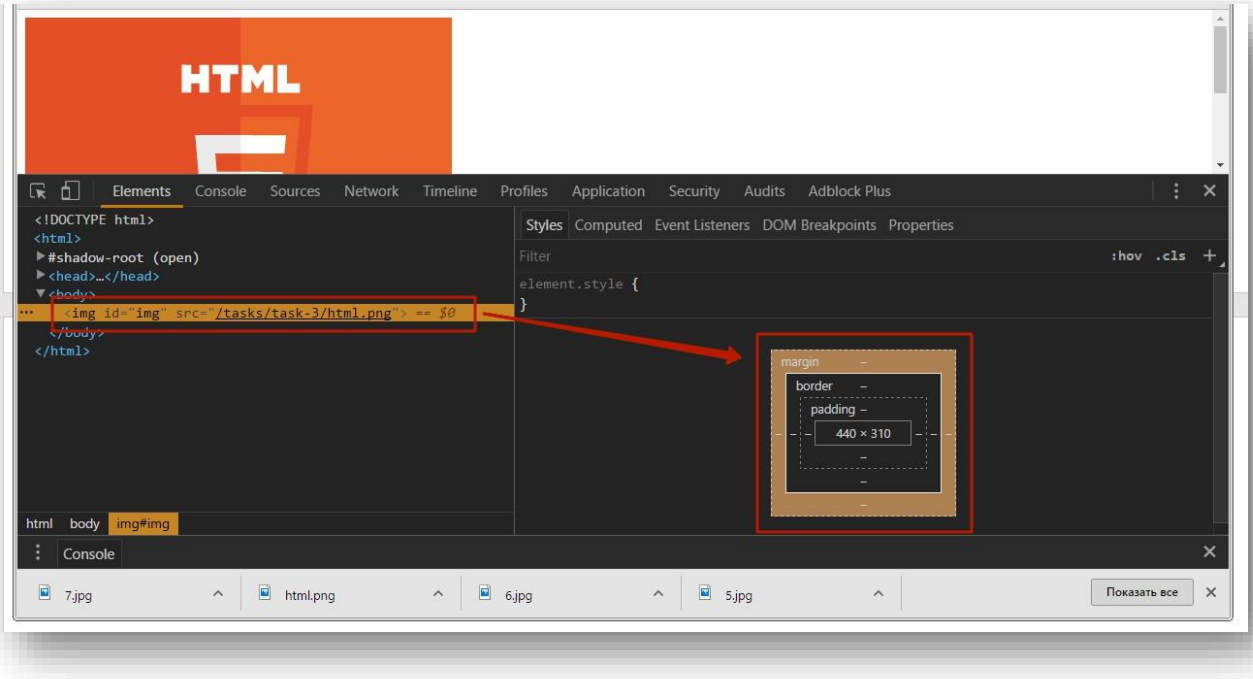
Как Вы можете видеть вместе с увеличением текста зазоры увеличились. Что бы убрать зазоры между блочно-строчными элементами нужно сбросить размер шрифта в ноль у родителя этих элементов, в данном случае это тот же элемент <body>.

Свойство ***display*** со значением ***none*** полностью скрывает элемент из верстки, как будто его там и не было. Скройте все элементы со страницы с помощью этого свойства.

Задание №3

В этом задании мы разберём анатомию HTML элементов. Открываем файлы index-3.html и style.css в папке данного урока. Открываем страницу урока в браузере Google Chrome http://css4.loc/task-3/ .

Щелкаем правой кнопкой мыши по картинке на странице и в появившемся контекстном меню выбираем «Просмотреть код». У Вас должна появиться панель разработчика. В панели разработчика кликните по элементу <img> на странице, в правой части Вы должны увидеть блочную модель этого элемента.



Браузер показываем Вам что размеры этой картинке 440 на 310 пикселей.

Padding *(pad – eng. прокладка)* – это отступ между границей элемента и его внутренним содержанием. Зададим padding для нашего изображения. Что бы его увидеть зададим так же цвет фона.

```
img {
    padding: 30px;
    background-color: green;
}
```

Что мы увидели в результате? Сам рисунок (внутренний контент) стал находиться на расстоянии 30px от всех границ элемента при этом размеры самого элемента возросли. Даже если задать свойство width, например, в 500px то мы увидим, что размер изменился у самого рисунка. Общая ширина рисунка вместе с padding будет больше, указанного значения width.

Следующий слой — это рамка вокруг элемента (border). Добавим рамку как показано ниже:

```
img {
    padding: 30px;
    background-color: green;
    width: 500px;
    border: 3px solid black;
}
```

Следующий слой это **margin** (отступ) позволяет нам установить пространство, которое окружает элемент. Поля margin находятся за пределами любых границ и полностью прозрачны в цвете. Они могут использоваться для позиционирования элементов в конкретном месте на странице или добавлять пустое пространство, сохраняя все другие элементы на безопасном расстоянии.

Используя визуальный интерфейс разработчика Chrome найдите элемент <body> и найдите его margin, чему ровно его значение?

Свойство margin у элемента <body> в нашем случае являются примером предустановленных стилей браузера. Для различных браузеров они различны. Для того что предустановленные стили нам не мешали мы можем их сбросить как показано ниже:

```
* {
    margin: 0;
    padding: 0;
    border: 0;
}
```

Теперь наш элемент вплотную прижат к границам окна браузера. Зададим ему свой margin, как показано ниже:

```
img {
    padding: 30px;
    background-color: green;
    width: 500px;
    border: 3px solid black;
    margin: 30px;
}
```

Теперь наш элемент окружает поле margin в 30px и он отодвинулся от всех других элементов на это расстояние, в нашем случае это граница окна браузера.

У строчных элементов нет **margin**!

Задание №4

Для создания рамок вокруг элементов в CSS определены следующие свойства:

border-width – задаёт толщину рамки.

border-color – устанавливает цвет рамки.

border-style – задаёт стиль прорисовки рамки. Возможные стили показаны на рисунке ниже.

Открываем файл style.css в папке задания и устанавливаем стили для двух первых блоков как показано на рисунке ниже.

```
.div1 {
    border-width: 1px;
    border-color: red;
    border-style: solid;
}

.div2 {
    border-width: 5px;
    border-color: navy;
    border-style: dotted;
}
```

Теперь добавим рамки для следующих двух блоков, но при этом будем использовать уже комбинированное свойство ***border***. Универсальное свойство border позволяет одновременно установить толщину, стиль и цвет границы вокруг элемента. Значения могут идти в любом порядке, разделяясь пробелом.

Добавим рамки для следующих блоков, как показано ниже.

```
.div3 {
    border: 10px double aqua;
}

.div4 {
    border: 20px inset crimson;
}
```

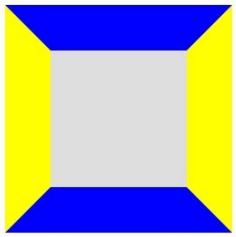
В CSS так же существует возможность установить рамку только с одной из сторон. Для этого существуют отдельные свойства для каждой стороны – ***border-left, border-right, border-bottom, border-top***. Воспользуемся ими и зададим верхнюю и нижнюю границы для следующего элемента как показано ниже.

```
.div5 {

    /* нижняя рамка - универсальное свойство */
    border-bottom: 3px solid olive;

    /* верхняя рамка - расписываем все свойство по отдельности */
    border-top-width: 5px;
    border-top-color: orange;
    border-top-style: dashed;
}
```

Далее самостоятельно определите стили рамок для последнего блока что бы он имел вид как на картинке ниже (подсказка – ширина рамки 50px, цвета blue, yellow).



Задание №5

Очень часто приходится создавать различные геометрические фигуры посредством CSS. Например, если оставить только одну рамку(border), а остальные рамки сделать прозрачными, мы получим треугольник нужного цвета. Создадим треугольник из первого блока, как показано на рисунке ниже.

```
#triangle {
    width: 0;
    height: 0;
    border-left: 50px solid transparent; /* transparent - "прозрачный цвет" */
    border-right: 50px solid transparent;
    border-bottom: 100px solid red;
}
```

Создадим еще и прямоугольный треугольник.

```
#triangle2 {
    width: 0;
    height: 0;
    border-bottom: 100px solid red;
    border-left: 100px solid transparent;
}
```

Задание №6

Свойство border-radius устанавливает радиус округления уголков рамки. Если рамка не задана, то округление также происходит и с фоном. Для того что бы понять, как работает это свойство мы наверстаем небольшой блок с аватаркой. Открываем файлы index-6.html и style.css в папке задания.

В файле index-6.html помещаем следующую верстку:

```
<div class="avatar_block" >
    <div class="avatar" ></div>
</div>
```

Набрасываем следующие стили на первый внешний блок, как показано ниже.

```
.avatar_block {
    width: 320px;
    background-color: #dedede;
    height: 320px;
}
```

На данном этапе у нас серый блок фиксированного размера. Разместим его посередине страницы. Для этого добавим с обоих сторон свойство ***margin*** со значением ***auto.*** При этом браузер сам будет рассчитывать значения свойства margin.

С помощью свойства **background-image** устанавливаем фоновое изображение и размеры для элемента «**.avatar**», как показано ниже:

```
.avatar {
    background-image: url('person.png');
    width: 120px;
    height: 120px;
}
```

Теперь нам нужно выставить блок «.avatar» ровно посередине своего родителя. Для этого делаем этот блок строчно-блочным и родителю выставляем выравнивание текста по середине (text-alingn: center;) как показано ниже.

```
.avatar_block {
    width: 320px;
    background-color: #dedede;
    height: 320px;
    margin-left: auto;
    margin-right: auto;
    text-align: center;
}

.avatar {
    display: inline-block;
    background-image: url('person.png');
    width: 120px;
    height: 120px;
}
```

Для того что бы выровнять наш элемент по вертикали определяем ему свойство **vertical-align:middle;** и его родителю определяем высоту линии (line-height) равную высоте самого элемента (320px).

```
.avatar_block {
    width: 320px;
    background-color: #dedede;
    height: 320px;
    margin-left: auto;
    margin-right: auto;
    text-align: center;
    line-height: 320px;
}

.avatar {
    display: inline-block;
    background-image: url('person.png');
    width: 120px;
    height: 120px;
    vertical-align: middle;
}
```

Округляем внутренний элемент «.avatar» свойством ***border-radius: 50%;*** и добавляем рамку для красоты.

```
.avatar {
    display: inline-block;
    background-image: url('person.png');
    width: 120px;
    height: 120px;
    vertical-align: middle;
    border-radius: 50%;
    border: 3px solid yellow;
}
```

Добавим элементу «.avatar_block» небольшие скругления по бокам, добавив свойство ***border-radius: 20px;***

В этом занятии мы познакомились как центрируются блочные элементы (margin: auto) и как центрируются строчные (и строчно-блочные) элементы (text-align: center;). Узнали, что свойство ***border-radius*** скругляет блочные и строчно-блочные элементы.

Задание №7

Свойство **box-shadow** позволяет задавать элементам тени. Ознакомиться с полным описанием этого свойства можно по ссылке https://webref.ru/css/box-shadow .

Аналогично можно добавить тень и к тесту элемента, с помощью свойства **text-shadow** (https://webref.ru/css/text-shadow).

Откройте файлы index-7.html и style.ccs в папке данного урока. Создайте блок с текстом. Добавьте блоку и тексту тени. Для удобного и наглядного создания теней можете воспользоваться следующими сервисами: http://www.cssmatic.com/box-shadow

https://css3gen.com/text-shadow/ - для теней к тексту.

Задание №8

В этом задании мы создадим простой макет страницы используя свойство inline-block. Открываем файлы index-8.html и style.css в папке текущего задания. Создаем базовую верстку страницы как показано ниже.

```
<div id="container" >
    <header></header>   <!-- тег заголовка сайта -->
    <main>
        <aside></aside> <!-- боковое меню -->
        <article></article><!-- основное содержимое документа -->
    </main>
    <footer></footer>   <!-- Футер или подвал сайта -->
</div>
```

В данном случаем мы используем вместо специализированные семантические теги, введенные в HTML5. Устанавливаем элементу #container ширину и центрируем его.

```
#container {
    width: 1024px;
    margin: 0 auto;
}
```

Задаём высоту и цвет элементам footer и header.

```
header {
    height: 160px;
    background-color: purple;
}


footer {
    height: 200px;
    background-color: black;
}
```

На данный момент у нас footer и header соприкасаются потому что высота контента страницы равна нулю. Исправим это, определим элементы aside и article как строчно-блочные как показано ниже.

```
aside {
    background-color: red;
    width: 300px;
    height: 200px;
    display: inline-block;
    vertical-align: top;
}

article {
    background-color: green;
    width: 700px;
    height: 600px;
    display: inline-block;
    vertical-align: top;
}
```

Колонки у Вас не сойдутся. Это происходит из-за того, что между строчно-блочными элементами образуется зазор и элемент article не помещается и переноситься ниже. Что бы убрать зазор нужно сбросить размер шрифта в ноль у родительского элемента main.

Задание №9

На прошлом задании мы рассмотрели, как можно с помощью inline-block можно создать простой двух колоночный макет. У такого макета есть один недостаток – колонки чаще всего будут разной высоты. Если необходимо что бы колонки всегда были одинаковой высоты (подстраивались друг под друга) и это важно, можно воспользоваться современной табличной версткой.

Перед Вами ниже таблица «симуляции» основных табличных элементов через свойство display.

Тег таблицы	**Свойство display**
<table>	display:table;
<tr>	display: table-row;
<td>, <th>	display: table-cell;

Теперь открываем файлы index-9.html и style.css в папке урока. Создаем базовый каркас трехколоночного макета, как показано ниже:

```
<div id="container">
    <header></header>
    <main class="table">
        <div class="tablerow">
            <aside class="tablecell" ></aside>
            <article class="tablecell" ></article>
            <aside class="tablecell" ></aside>
        </div>
    </main>
    <footer></footer>
</div>
```

Добавляем необходимые стили, как на рисунке ниже:

```css
.table {
   display: table;
   width: 100%;
}

.tablerow {
    display: table-row;
}

.tablecell {
    display: table-cell;
}

aside {
    background-color: red;
    width: 20px;
    height: 200px;
}

article {
    background-color: green;
    width: 60%;
    height: 1000px;
}
```

Вы можете видеть, как бы не менялась высота колонок, все равно они остаются одинаковой высоты. В этом «фишка» этого макета. Если нет требования делать колонки одинаковой высоты, то предпочтительнее верстать макет через inline-block (Задание №8).

Задание №10

Свойство **float** делает элемент «плавающим» и другие элементы начинают его «обтекать».

float: left - выравнивает элемент по левому краю, а все остальные элементы, вроде текста, обтекают его по правой стороне.

float: right - выравнивает элемент по правому краю, а все остальные элементы обтекают его по левой стороне

float: none – нет обтекания.

Это свойство изначально задумывалось для того что бы можно было размещать изображения в тексте, так что бы текст их «обтекал».

Откройте в браузере страницу задания http://css4.loc/task-10/ . Как Вы видите изображения в тексте расположены небрежно, сейчас с помощью свойства float мы это исправим. Откройте файлы index-9.html и style.css в папке урока. Зададим обтекание изображениям:

```css
#img1 {
    float: left;
    padding: 10px;
}

#img2 {
    float: right;
    padding: 10px;
}
```

Задание №11

Обтекание так же используют для создания макетов. В этом задании мы наверстаем простой макет используя обтекания. Открываем файлы index-11.html и style.css в папке задания.

Создаём каркас макета в традиционном стиле через элементы.

```
<div class="header">
    Верхний колонтитул
</div>
<div class="wrapper">
    <div class="menu">Блок для меню</div>
    <div class="main">Основной блок</div>
    <div class="clear"></div>
</div>
<div class="footer">
    Нижний колонтитул
</div>
```

Добавляем стили на страницу:

```
.wrapper, .header, .footer {
  width: 960px;
  margin-left: auto;
  margin-right: auto;
}
.header {
    background: yellow;
    height: 100px;
}
.menu {
    float: left;
    width: 33%;
    background: pink;
    height: 250px;
}
.main {
    float: right;
    width: 67%;
    background: #fffddf;
    height: 300px;
}

.footer {
    background: #dedede;
    height: 100px;
}
```

В текущем варианте макета нас ждет небольшая засада. Наш «футер» провалился внутрь плавающих элементов, это происходит потому что футер попал в поток обтекания. Для того что бы прервать поток обтекание используется свойство clear, которое запрещает обтекание слева(*left*), справа(*right*) или с двух сторон(*both*). Подробнее https://webref.ru/css/clear

Добавим запрет обтекания, чтобы вернуть footer на место.

```
.clear {
    clear: both;
}
```

Задание №12

В этом задании мы создадим блок карточек, которые следуют друг за другом.

Открываем файлы index-12.html и style.css в папке задания. Создаём каркас, как показано ниже

```
<div class="container">
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
    <div class="card"></div>
</div>
```

Центрируем блок «.container» и зальём его фоном.

```
.container {
    margin: 30px auto;
    background-color: gray;
    width: 1000px;
    font-size: 0px;
}
```

Стилизуем наши карточки (блоки «.card»):

```
.card {
    width: 220px;
    height: 280px;
    margin: 15px;
    background-color: green;
    display: inline-block;
    vertical-align: top;
}
```

Как Вы видите для того что бы разместить карточки друг за другом мы воспользовались свойством display: inline-block. Обратите внимание, что если у элемента «.container» убрать свойство ***font-size:0px;,*** то появиться дополнительный неконтролируемый отступ между карточками и уже не будет хватать места для 4 карточек в ряду.

Попробуем сделать то же самое используя свойство float. **Переписываем** свойства элементов «.card» как показано ниже.

```
.card {
    width: 220px;
    height: 280px;
    margin: 15px;
    background-color: green;
    float: left;
}
```

Как Вы можете сами видеть у нас возникли некоторые проблемы. В консоли разработчика видно, что высота блока «.container» равна нулю. Он просто не учитывает плавающие элементы. Что бы осознать глубину проблемы давайте разместим за элементом «.container» вот такой блок:

```
<div style="height: 400px; background-color: red;" ></div>
```

Как Вы можете сами видеть этот блок «проваливается» внутрь наших плавающих элементов. Для того что бы это исправить и заставить блок «.container» учитывать плавающие элементы внутри себя, нужно добавить свойство ***overflow: hidden;***

Подробнее об этом свойстве, Вы можете прочитать тут https://webref.ru/css/overflow

Задание №13

В этом задании мы создадим небольшой блок пагинации. Используя знания, полученные в предыдущем уроке, создадите через inline-block следующую вёрстку:

Размеры шрифтов и отступов произвольные.

Задание №14

В этом задании мы подробнее познакомимся со свойствами фонового цвета и изображения.

background-color – свойство, которое определяет цвет фона элемента.

background-image – свойство, которое устанавливает фоновое изображение для элемента.

Откройте файлы index-14.html и style.ccs в папке задания. Зададим элементу <body> фоновое изображение, как показано ниже:

```
body {
    background-image: url(bg.png);
}
```

Как Вы можете видеть по умолчанию эта небольшая картинка заполняет своими копиями все пространство блока, создавая иллюзию фактуры.

Свойство ***background-repeat*** определяет, как будет повторяться фоновое изображение. Попробуйте последовательно применить все значения свойства *background-repeat* в нашем примере ниже.

```
body {
    background-image: url(bg.png);

    background-repeat: no-repeat; /* изображение не повторяется */
    background-repeat: repeat-x; /* изображение повторяется только по оси x */
    background-repeat: repeat-y; /* изображение повторяется только по оси y */

}
```

Так же можно использовать одновременно свойства *background-color* и *background-image*. Например так:

```
body {
    background-image: url(bg.png);
    background-repeat: no-repeat; /* изображение не повторяется */
    background-color: #dedede; /* цвет зальёт весь фон, кроме изображения */

}
```

Задание №15

Откройте файл style.css в папке урока и зададим элементу «.bg» фоновое изображение и размер:

```
.bg {
    background-image: url(persons.jpg);
    background-repeat: no-repeat;
    width: 1000px;
    height: 10000px;
    margin: 0 auto;
}
```

Свойство ***background-size*** масштабирует фоновое изображение согласно заданным параметрам. У этого свойства может быть множество значений мы рассмотрим самые интересные. Попробуйте последовательно применить все значения свойства *background-size* в нашем примере ниже.

```
.bg {
    background-image: url(persons.jpg);
    background-repeat: no-repeat;
    width: 1000px;
    height: 10000px;
    margin: 0 auto;

    /* Масштабирует изображение с сохранением пропорций таким образом,
    чтобы картинка целиком поместилась внутрь блока. */
    background-size: contain;

    /* фон займет по половину ширины блока */
    background-size: 50%;

    /* указываем ширину и высоту фона напрямую */
    background-size: 300px 400px;
}
```

Свойства ***background-position*** задаёт начальное положение фонового рисунка, у него два значения, положение по горизонтали (может быть — left, center, right) и вертикали (может быть — top, center, bottom). Кроме использования ключевых слов положение также можно задавать в процентах, пикселях или других единицах. Зададим положение фонового рисунка как показано ниже:

```
background-position: top center;
```

Свойство ***background-attachment*** устанавливает, будет ли прокручиваться фоновое изображение вместе с содержимым элемента. Изображение может быть зафиксировано и оставаться неподвижным, либо перемещаться совместно с документом. Зафиксируем фон при прокрутке как показано ниже:

```
background-attachment: fixed;
```

Задание №16

Особенностью таблиц является уникальное свойство ***border-collapse***, которое позволяет нам «склеивать» границы соседних ячеек. Рассмотрим действия этого свойства на примере. Откройте файл style.css в папке задания и добавьте свойство border-collapse как показано ниже.

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

Понаблюдайте как это свойство воздействует на таблицу.

Задание №17

Свойство ***list-style-type*** изменяет вид маркера для каждого элемента списка. Открываем файл style.css и устанавливаем свойство *list-style-type* как показано ниже:

```
li {
    list-style-type: circle;
}
```

Понаблюдайте за изменением внешнего вида маркеров списка. С другими возможностями этого свойства можно познакомиться по ссылке https://webref.ru/css/list-style-type . Значение этого свойства «***none***» полностью уберет маркер списка.

Свойство ***list-style-position*** определяет, как будет размещаться маркер относительно текста. Имеется два значения: *outside* — маркер вынесен за границу элемента списка (по умолчанию) и *inside* — маркер обтекается текстом.

Добавим свойство list-style-position для нашего списка:

```
list-style-position: inside;
```

Так же существует возможность поставить вместо маркера списка изображение, за это отвечает свойство ***list-style-image.*** Установим изображение как маркер списка:

```
list-style-image: url(1.jpg);
```

Задание №18

Создать простое вертикальное меню из списка очень просто. Для этого нужно открыть файлы index-18.html и style.css в папке задания и вставить базовую верстку, как показано на рисунке ниже:

```
<ul>
    <li><a href="#">Ссылка 1</a></li>
    <li><a href="#">Ссылка 2</a></li>
    <li><a href="#">Ссылка 3</a></li>
    <li><a href="#">Ссылка 4</a></li>
    <li><a href="#">Ссылка 5</a></li>
</ul>
```

Далее необходимо настроить стили:

```css
li {
    /* убираем маркер у списка */
    list-style-type: none;
}

a {
    /* убираем подчеркивание у ссылок */
    text-decoration: none;
}

a:hover {
    /* убираем подчеркивание у ссылок при наведении */
    text-decoration: underline
}
```

Задание №19

Так же очень легко создать горизонтальное меню с помощью списка. Открываем файлы index-19.html и style.css и помещаем базовую верстку нашего будущего меню:

```html
<nav>
    <ul class="menu">
        <li class="menu-element">
            <a class="menu-link" href="#">Home</a>
        </li>
        <li class="menu-element">
            <a class="menu-link" href="#">About</a>
        </li>
        <li class="menu-element">
            <a class="menu-link" href="#">Clients</a>
        </li>
        <li class="menu-element">
            <a class="menu-link" href="#">Contact Us</a>
        </li>
    </ul>
</nav>
```

А так же добавим стили :

```css
.menu-element {
    display: inline-block;
    vertical-align: top;
    margin-left: 10px;
    margin-right: 10px;
}

.menu-link {
    color: orangered;
    font-size: 24px;
}
```

Как Вы видите мы воспользовались свойством inline-block для того что бы расположить элементы <li> горизонтально.

Домашнее задание

Прочитать статью о создании геометрических фигур на CSS https://habrahabr.ru/post/126207/