Введение в CSS.

Другие селекторы.

```
<div class="container">
  <div class="main" id="example">
    <img src="img.jpg" class="preview">
    <p class="text">[DESCRIPTION]</p>
    <a href="#" class="link">Read More</a>
  </div>
</div>
```

По идентификатору.	`#example{}` `div#example{}`
По атрибуту.	`.link[href="#"]{}`
Для всех элементов.	`*{}`
Для всех дочерних элементов.	`.container *{}`

Для всех дочерних только на первом уровне вложенности.	`.container > .main{}` `.container > *{}`
Для всех на одном уровне вложенности.	`.preview:hover ~ .link{}`
Для ближайшего соседа на одном уровне вложенности.	`.preview:hover + .text{}`

Если одинаковый набор стилей используют сразу несколько элементов, можно перечислить их селекторы через запятую.

```
.main .text,
.main .link{}
```

Задание!

Давайте вспомним, у каких элементов есть стили по-умолчанию.

Подготовительные работы.

Для удобства разработки хорошо зарекомендовала себя такая структура проекта.

- Project name
  - assets (содержит файлы, использующиеся на фронте)
    - styles (содержит файлы стилей)
    - scripts (соответственно скрипты)
    - images (статичные картинки, такие как фон или иконки)
    - temp (содержит временные файлы, которые на сайте будут заменены реальными данными)
    - fonts
  - Project name

Также важно помнить о чистоте в папках. Представим следующую картину:

-Хм, пожалуй добавлю свой проект вот сюда. Что может пойти не так?

-Ага, стили привезли положим туда же. И сброс стилей сюда влезет.

-Картинка с фоном, картинка с чуваком, пару картинок, которые я скачал для рыбы, но потом просто не использовал, и забыл, psd макет туда же кину, куда же без него.

Приоритет стилей.

У CSS стилей есть одна особенность: они применяются к тегу на всем сайте, и используются как тегами, которые были ещё до загрузки страницы браузером, так и для тех, которые появились асинхронно (например с помощью js скрипта).

Кроме того, у них есть свой приоритет.

1. Для одинаковых по массе селекторов приоритетнее будет тот, который записан снизу.
2. У селекторов также есть свои приоритеты (указаны от меньшего)
   a. Тег - 1;
   b. Класс - 10;
   c. Идентификатор - 100;
3. У стилей, записанных через атрибут (инлайном) приоритет - 1000.
4. Еще более приоритетным является объявлением !important в конце свойства.

```
div.one      - 11
.one.two     - 20
#three       - 100
#three.two   - 110
div{
  display: none !important;
  это свойство будет выше всех приоритетов.
```

```
<div class="one two" id="three"></div>
```

Внимание!

Из перечисленных способов, !important считается самым не рекомендуемым.

Работа с размерами.

Для того, чтобы задать ширину, необходимо воспользоваться свойством **width**.

```
width: value | % | auto | calc();
```

Также можно ограничить ширину д

```
max-width: value | % | auto | c
min-width: value | % | auto | c
```

Однако, мы знаем, что блочный эле
разберемся что станет с блоком, ес

```
<div class="container" style="w
```

Внимание!

Мы приблизились к первому кардинальному отличию блочных и строчных элементов.
Строчному элементу невозможно задать высоту с помощью height. Тк строчный элемент - это просто строка текста, и высота у неё будет равна высоте строки

С высотой дела обстоят несколько с
разобраться с потоком.

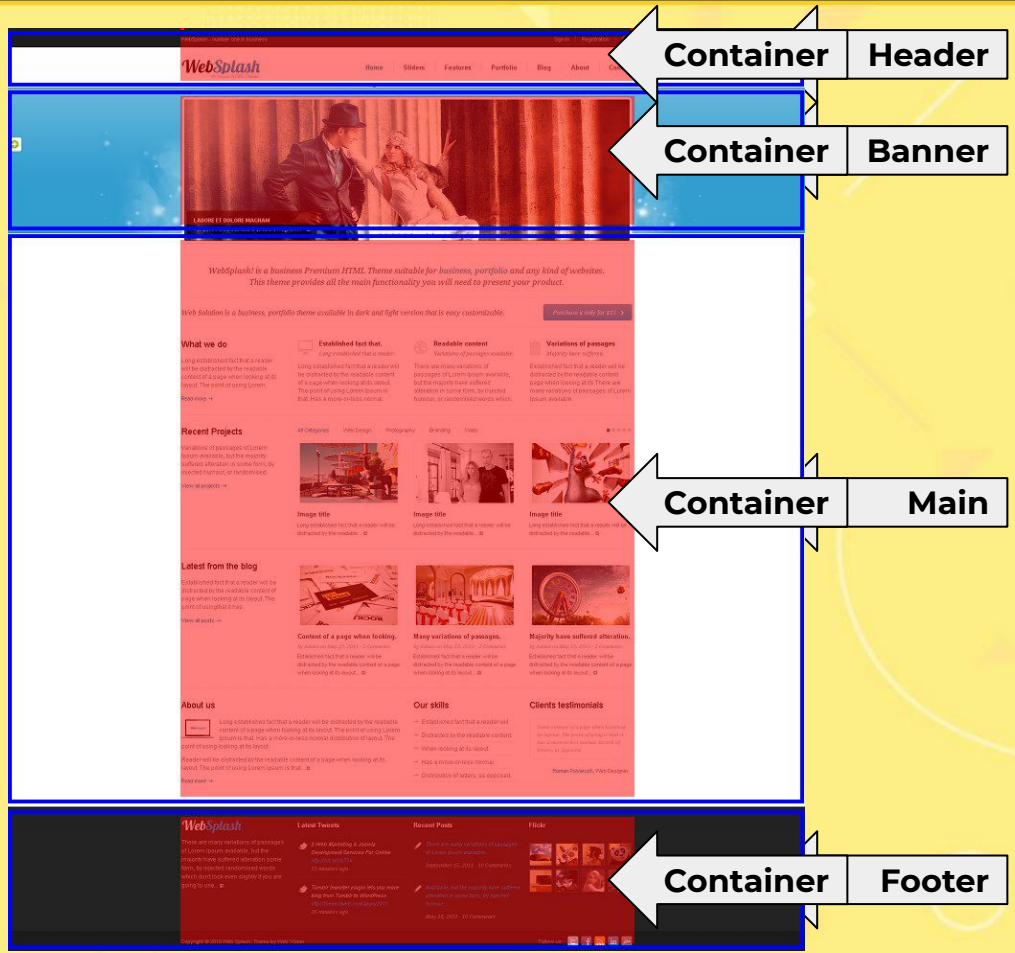
В остальном же, все ровно так же, как и с шириной. Свойство **height**

```
height: value | % | auto | calc();
```

Также можно ограничить ширину до максимальной и/или минимальной.

```
max-height: value | % | auto | calc();
min-height: value | % | auto | calc();
```

Рассмотрим контейнер сайта.



Ширина этого контейнера оказалась 1100px, теперь попробуем увидеть каркас нашего будущего сайта.

Шаг второй:

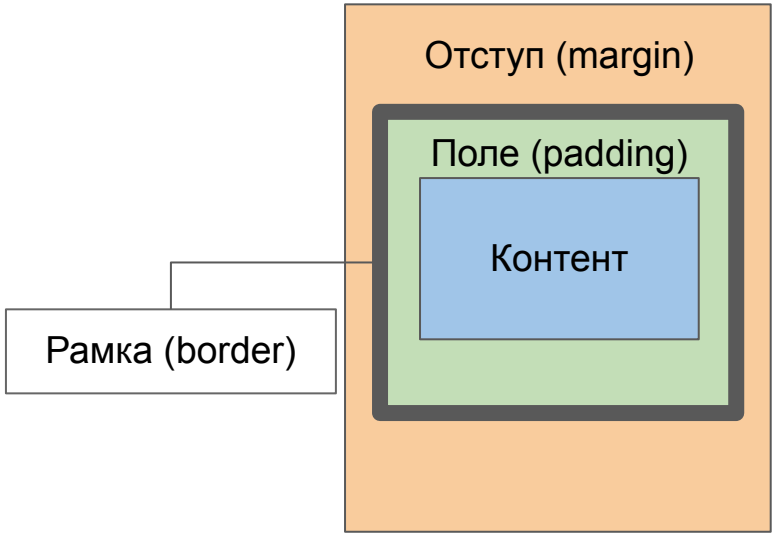
```
<body>
  <div class="header">
    <div class="container">...</div>
  </div>
  <div class="banner">
    <div class="container">...</div>
  </div>
  <div class="main">
    <div class="container">...</div>
  </div>
  <div class="footer">
    <div class="container">...</div>
  </div>
</body>
```

Отступы.

В разметке документа очень важными свойствами являются поля (**padding**), отступы (**margin**) и рамки (**border**).

Рассмотрим на примере отступа: margin.

- **margin-top**: 10px;
- **margin-right**: 20px;
- **margin-bottom**: 15px;
- **margin-left**: 20px;



Также можно использовать группировку значений.

margin: 10px 20px 15px 20px; (top) (right) (bottom) (left)
margin: 10px 20px 15px; (top) (right/left) (bottom)
margin: 10px 20px; (top/bottom) (right/left)
margin: 10px; (top/bottom/right/left)

Отступ может использовать отрицательные величины, и в таком случае он будет не отталкивать другой элемент, а наоборот перекрывать его.

Если у **блочного** элемента ширина меньше 100%, то с помощью значения **auto** он оттолкнется от каждого края по на одинаковое расстояние (работает только по горизонтали).

Поля.

Для полей (**padding**) применимы такие же правила группировки значений.

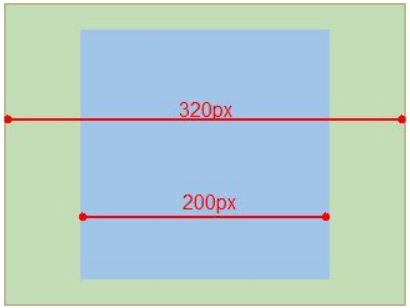
И разница между ними в том, что margin это отступ (дистанция), а поле элемента и его рамка является его частью.

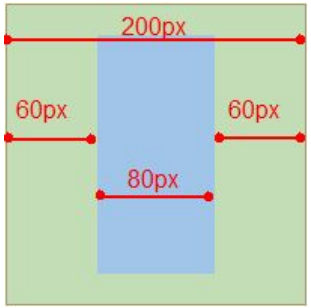
Для блочного элемента с шириной 100% padding будет вжимать контент внутрь (отталкивая от границ родительского блока,

Иначе padding будет увеличивать размер элемента. Что в некоторых ситуациях мешает верстке.

Свойство box-sizing позволяет изменить логику расчета ширины. Принимает три значения:

- **border-box** - ширина/высота суммирует элемент, его padding и border.
- **padding-box** - ширина складывается из элемента и его padding.
- **content-box** - к ширине контента ничего не добавляется. Используется по-умолчанию.





Рамки.

У рамок есть три основных значения:

- **border-width** - Размер границ;
- **border-style** - Начертание;
- **border-color** - Цвет.

border: размер начертание [цвет];

Также можно добавить рамку для любой из сторон элемента.

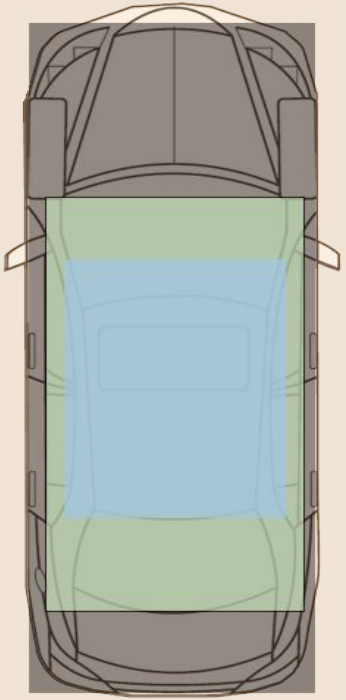
- **border-top**: 30px solid red;
- **border-right**: 30px solid red;
- **border-bottom**: 30px solid red;
- **border-left**: 30px solid red;

Или сгруппировать эти свойства в следующее:

- **border:** 30px solid red;

border-radius - CSS3 свойство для скругления углов.

- **border-radius**: value | %;



Стили для блоков.

overflow - управление контентом блока, позволяет обрезать видимую часть, или добавляет скролл, если контента больше, чем величина блока.

```
overflow: auto | hidden | scroll | visible;
```

visible - Отображается все содержимое элемента, даже за пределами установленной высоты и ширины.

hidden - Отображается только область внутри элемента, остальное будет скрыто.

scroll - Всегда добавляются полосы прокрутки.

auto - Полосы прокрутки добавляются только при необходимости.

Задает стиль маркеров для списков ul => li, для нумерованных списков читай документацию на http://htmlbook.ru

```
list-style-type: circle | disc | square;
```

Позволяет скрывать и отображать блок, оставляя его в потоке

```
visibility: visible | hidden | inherit;
```

Изменяет внешний вид курсора, необходимо для исполнительных элементов на странице, полный перечень возможных значений смотри http://htmlbook.ru

```
cursor: default | pointer;
```