

# CSS: Introduction, box-model

# НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

# Повторение;)

■ Как создать якорную ссылку?

■ Какая разница между `div` и `span`?

■ Какие основные семантические теги вы знаете?

# ЦЕЛЬ

Познакомиться с основами CSS. Изучить концепцию блочной модели в css.  
Научиться отличать и использовать margin, padding, border

# ПЛАН ЗАНЯТИЯ

- CSS. Основы
- Стили: inline styles, internal CSS, external CSS
- CSS селекторы
- Class и id атрибуты
- Box-model



# CSS. Введение

# HTML



**HTML** the Skeleton



# CSS



**CSS** the Skin



# JS

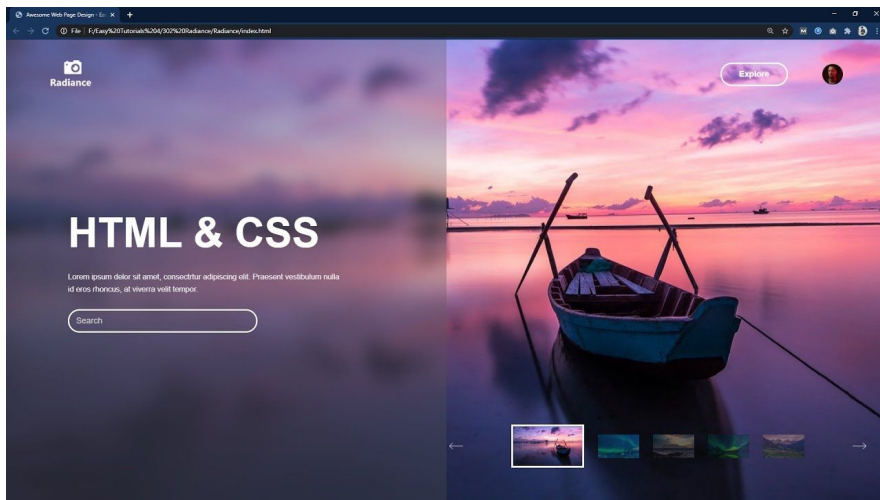


**Javascript** the Brain



**CSS (Cascading Style Sheets) - язык, который отвечает за описание внешнего вида HTML-документа.**

**Если HTML структурирует контент на странице, то CSS позволяет отформатировать его, сделать более привлекательным для читателя.**





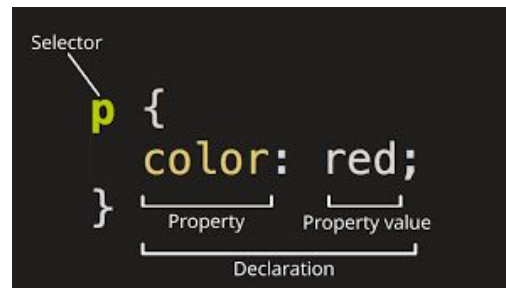
# Синтаксис CSS

У языка CSS относительно простой синтаксис. Сначала прописывается селектор — он выбирает конкретный элемент на странице. Потом, после фигурных скобок, указываются свойства со значениями — между ними ставится двоеточие. Сами свойства отделяются друг от друга точкой с запятой.

## Схема



## Пример

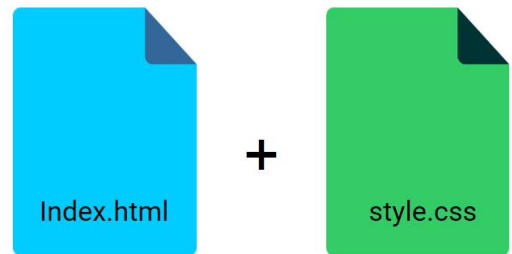


# Варианты подключения стилей



Существует 3 основных способа подключения или добавления CSS стилей к HTML документу, каждый из которых подходит для определенного круга задач.

- CSS стили для конкретного тега атрибутом `style`
- Подключение с помощью тега `link` через внешний файл стилей CSS
- Добавление CSS с помощью тега `style` в HTML файл



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

# С помощью атрибута style для тега

`<p style="color: red">Текст сообщения</p>`



# Подключение с помощью тега link через внешний файл стилей CSS

Подключение стилей CSS является использование элемента `link`, который позволяет подключать к HTML странице внешние файлы. Ссылка на внешний CSS файл помещается в контейнер `<head>` страницы:

```
<link rel="stylesheet" href="styles.css" type="text/css"/>
```

В атрибуте `href` необходимо указать URL адрес файла, содержащего набор стилей CSS. Атрибуты `rel="stylesheet"` и `type="text/css"` указывают, что указанный файл является таблицей стиля в формате CSS.

# Подключение с помощью тега link через внешний файл стилей CSS

## Преимущества внешних CSS:

- Меньший размер страницы HTML и более чистая структура файла.
- Быстрая скорость загрузки.
- Для разных страниц может быть использован один и тот же .css файл.

## Недостатки внешних CSS:

- Страница может некорректно отображаться до полной загрузки внешнего CSS.

# Добавление CSS с помощью тега style в HTML файл

Этот метод подойдет когда нужно вставить (определить) стили для группы уникальных элементов страницы, то есть, набор стилей, которые используются только в пределах одной страницы и не нужны для корректной работы остальных страниц сайта.

В любом месте областей `<head>` и `<body>` HTML документа используйте тег `<style>`, внутри которого поместите необходимые CSS правила.

```
<style>

    h2 {

        color: red;

    }

</style>
```

# CSS Селектор





# Что такое селекторы?

Селекторы - это выражения, которые говорят браузеру, к какому элементу HTML нужно применить те или иные свойства CSS, определённые внутри блока объявления стиля.

```
h1 {  
  color: blue;  
  background-color: yellow;  
}  
  
p {  
  color: red;  
}
```

# Типы базовых CSS селекторов

- Универсальный селектор
- Селекторы по типу элемента
- Селекторы по идентификатору
- Селекторы по классу
- Селекторы по атрибуту



## Универсальный селектор

Он применяет стили ко всем элементам страницы и обозначается символом \* (звёздочка).

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

## Селектор по элементу (тегу)

Этот селектор CSS применяет стили ко всем элементам с одинаковым тегом. Например, для всех <div>, <h2>, <p> и так далее.

```
p {  
    color: red;  
}
```

# Селекторы по идентификатору

Селектор по идентификатору обозначается символом **#** (решётка) и применяет стили к элементу, для которого задан атрибут **id** с соответствующим значением. При этом у элемента может быть только один **id**, и этот **id** должен быть уникальным в пределах веб-страницы.

## html code

```
<p id="article_content">В этой  
статье:</p>
```

## css code

```
#article_content {  
  
    font-family: sans-serif;  
  
    font-weight: bold;  
  
}
```

## Селекторы по классу

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

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

*Примечание:* Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчеркивания (\_). Использование русских букв в именах классов недопустимо.

# Добавление класса

Чтобы добавить класс, необходимо для элементов (тегов), к которым будут применены свойства класса, прописать атрибут **class** с придуманным названием класса в качестве значения атрибута:

```
<h1 class="class_name">Текст сообщения</h1>
```

# Возможности классов

- К одному тегу можно добавлять несколько классов, для это нужно перечислить названия в значении атрибута **class** через пробел

```
<h1 class="name1 name2">Текст сообщения</h1>
```

- В селекторе можно указать стили для определённого тега, входящего в класс

```
p.class_name {  
    свойство1: значение;  
    свойство2: значение;  
}
```



# Селекторы по атрибуту

Селекторы по атрибуту позволяют выбрать элемент по имени атрибута, его значению или части значения. Селектор атрибута обозначается в квадратных скобках **[ ]**

1. Селектор по названию атрибута **[attr]**

```
[title] {  
  
    font-weight: bold;  
  
}
```

# Селекторы по атрибуту

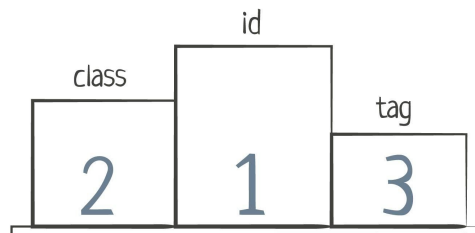
## 2. Селектор по имени и значению атрибута **[attr=value]**

```
[title="text"] {  
  
    font-weight: normal;  
  
}
```

Больше вариантов селекторов по атрибуту см.

[https://developer.mozilla.org/ru/docs/Web/CSS/Attribute\\_selectors](https://developer.mozilla.org/ru/docs/Web/CSS/Attribute_selectors)

# Приоритеты выполнения стиля



Важной чертой CSS является то, что разные типы селекторов имеют разный приоритет. Если стили противоречат друг другу в разных селекторах, то вступает в дело принцип приоритета.

Разберём на примере. Создадим параграф с классом `red` и идентификатором `blue`

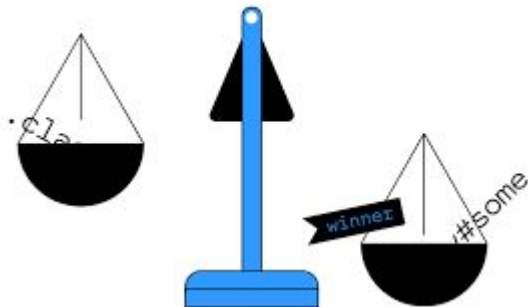
```
<p id="blue" class="red">Текст с классом и id</p>
```

Добавим противоречащие друг другу стили для класса, идентификатора и тега. Какого цвета будет параграф?

```
p {  
    color: black;  
}  
  
.red {  
    color: red;  
}  
  
#blue {  
    color: blue;  
}
```

Для определения приоритета можно использовать следующие правила, где каждому селектору задаётся его «вес»:

- Селектор по тегу: 1
- Селектор по классу и атрибуту: 10
- Селектор по ID: 100
- Стиль в атрибуте тега: 1000



# Список CSS свойств

<https://developer.mozilla.org/en-US/docs/Web/CSS/color>

# Комбинированные стили





# Группировка селекторов

Если нескольким селекторам нужно задать одно и то же правило, то можно написать длинно:

```
h1 {  
    color: red;  
}  
  
h2 {  
    color: red;  
}  
  
h3 {  
    color: red;  
}
```

А можно перечислить все селекторы через запятую и написать всего одно CSS-правило:

```
h1, h2, h3 {  
    color: red;  
}
```

## Группировка селекторов

При группировке любые виды селекторов записываются в произвольном порядке через запятую и в таком случае правила будут применяться к каждой из групп:

```
h1, .block, #id, h3 {  
    color: red;  
}
```

# Сложение стилей



Если разные правила для одного элемента содержат свойства, которые не конфликтуют, то они объединяются в один стиль, т.е. каждое новое правило добавляет новую информацию о стиле к тому правилу, которое находится выше:

```
h1 {  
  
    color: gray;  
  
    font-family: sans-serif;  
  
}
```

```
h1 {  
  
    border-bottom: 1px solid black;  
  
}
```

```
h1 {  
    color: gray;  
    font-family: sans-serif;  
    border-bottom: 1px solid black;  
}
```

Обычно дополнительные правила для элемента указываются в тех случаях, когда был задан один стиль сразу для нескольких элементов, но помимо этого необходимо добавить что-то ещё для определённого элемента:

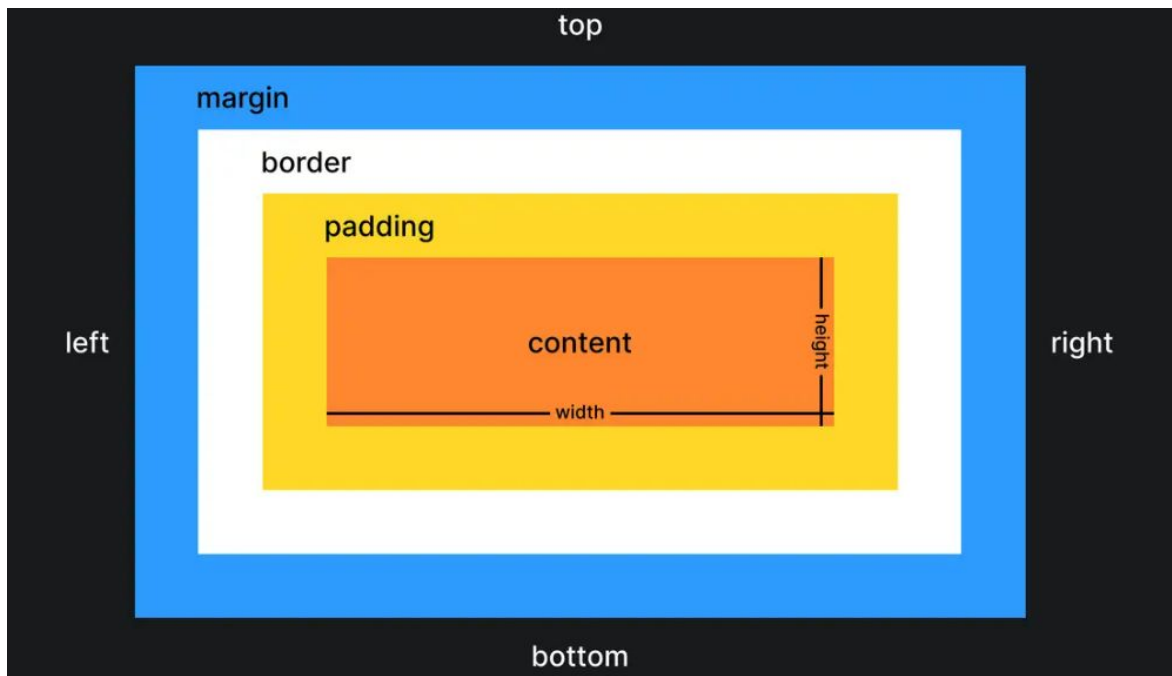
```
h1, h2, h3 {  
  
    color: gray;  
  
    font-family: sans-serif;  
  
}  
  
h2 {  
  
    border-bottom: 1px solid black;  
  
}
```

# Box model

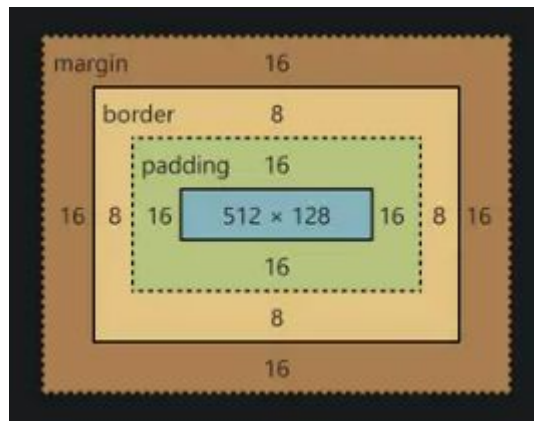


## Блочная модель

Блочная модель, она же box model — это алгоритм расчёта размеров каждого отдельного элемента на странице, которым браузеры пользуются при отрисовке.



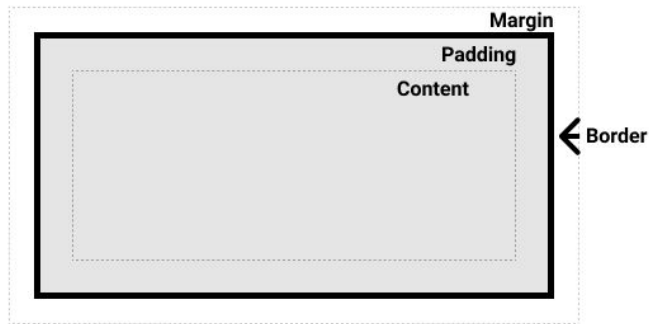
Ровно такую же схему, но в других цветах можно увидеть в инструментах разработчика любого из браузеров. Например, так выглядит блочная модель элемента в Chrome:



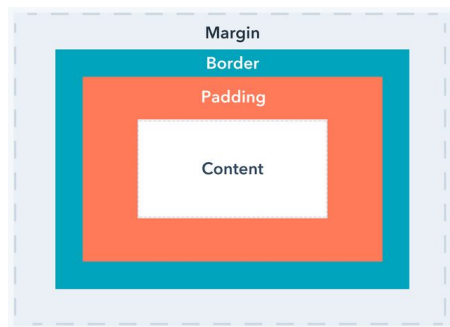


Блочная модель состоит из нескольких CSS-свойств, влияющих на размеры элемента:

- **width** — ширина элемента;
- **height** — высота элемента;
- **padding** — внутренние отступы от контента до краёв элемента;
- **border** — рамка, идущая по краю элемента;
- **margin** — внешние отступы вокруг элемента



# Внешние и внутренние отступы



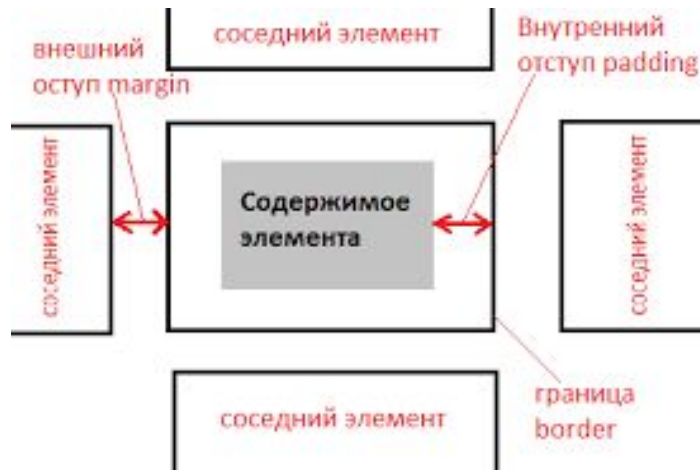
# Внутренние и внешние отступы

Внутренние отступы могут регулироваться отдельными свойствами:

- padding-top
- padding-right
- padding-bottom
- padding-left

Внешние отступы могут регулироваться отдельными свойствами:

- margin-top
- margin-right
- margin-bottom
- margin-left.



Общие свойства:

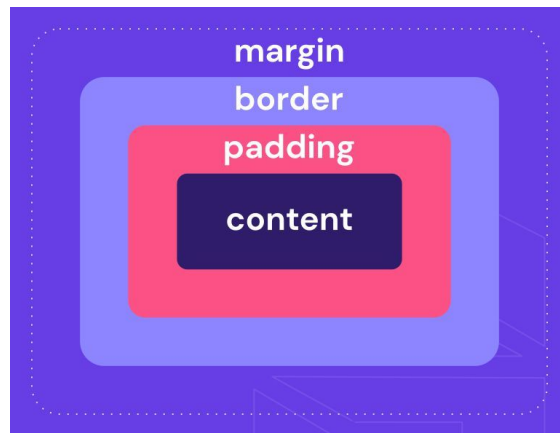
- padding
- margin

## Внутренний отступ

Внутренний отступ - пустое пространство между содержимым элемента и его рамкой (если она установлена). Для добавления и управления шириной внутренних отступов со всех четырех сторон элемента используется свойство `padding`.

padding: 10px 15px 22px 18px;

top right bottom left

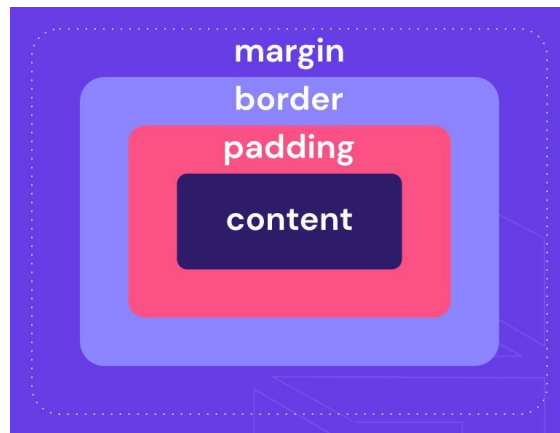


## Внешний отступ

Внешний отступ - пустое пространство, отделяющее элемент от других элементов или краев окна браузера. Для добавления и управления шириной внешних отступов со всех четырех сторон элемента используется свойство `margin`.

`margin: 10px 15px 22px 18px;`

top right bottom left

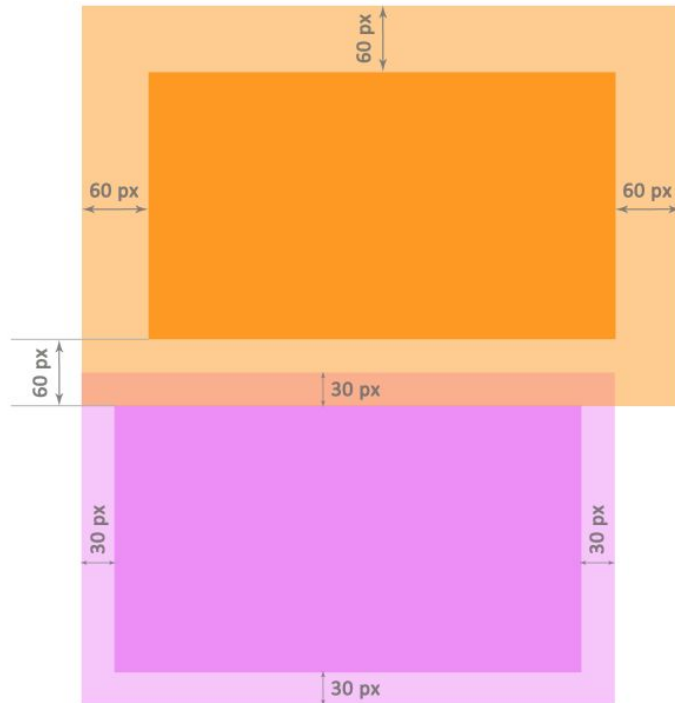


## Схлопывание margin (Margin Collapse)

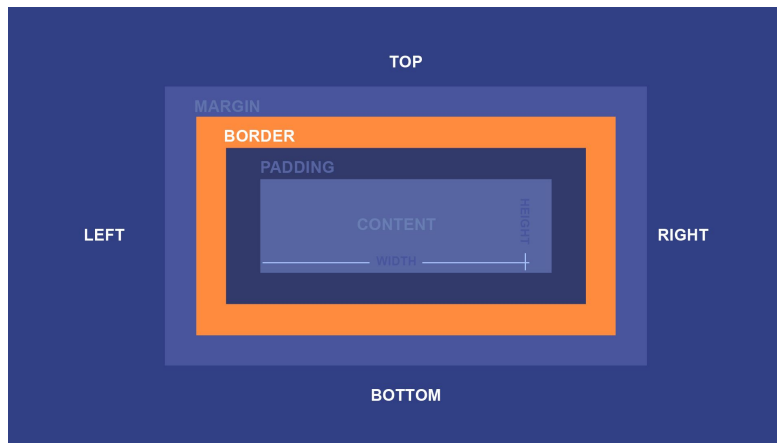
Представьте ситуацию: два блочных элемента находятся друг над другом и им заданы поля `margin`. Для верхнего блока установлено значение `margin: 60px`, а для нижнего – `margin: 30px`. Логично было бы предположить, что два граничащих поля двух элементов просто соприкоснутся и в итоге промежуток между блоками будет равен 90 пикселям.

Однако дела обстоят по-другому. На самом деле в такой ситуации проявляется эффект, который называют схлопыванием, когда из двух примыкающих полей элементов выбирается наибольший по размеру. В нашем примере итоговый промежуток между элементами будет равен 60 пикселям.

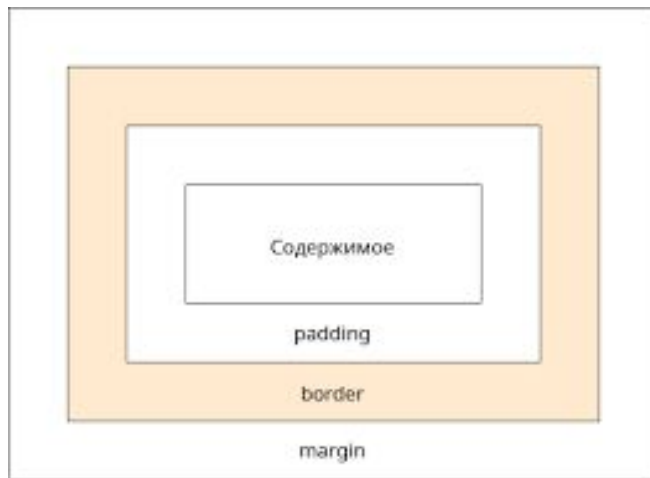
Схлопывание `margin` работает только для верхних и нижних полей



# Параметры рамки. Цвет и стиль рамки



**CSS-рамка элемента представляет собой одну или несколько линий, окружающих содержимое элемента и его поля.**





## Стиль рамки - border-style

По умолчанию рамки всегда отрисовываются поверх фона элемента, фон распространяется до внешнего края элемента. Стиль рамки определяет ее отображение, без этого свойства рамки не будут видны вообще. Для элемента можно задавать рамку для всех сторон одновременно с помощью свойства `border-style` или для каждой стороны отдельно с помощью уточняющих свойств `border-top-style` и т.д. Данное свойство не наследуется.



# Стиль рамки - border-style

## border-style

(border-top-style, border-right-style, border-bottom-style, border-left-style)

### Значения:

`none`

Значение по умолчанию, означает отсутствие рамки. Также убирает рамку элемента из группы элементов с установленным значением данного свойства.

`hidden`

Эквивалентно `none`.

`dotted`

`dotted`

`dashed`

`dashed`

`solid`

`solid`

`double`

`double`

## Стиль рамки - border-style

groove

groove

ridge

ridge

inset

inset

outset

outset

{1, 4}

Одновременное перечисление четырех разных стилей для рамок элемента, только для свойства `border-style` :

```
{border-style: solid dotted none dotted;}
```

initial

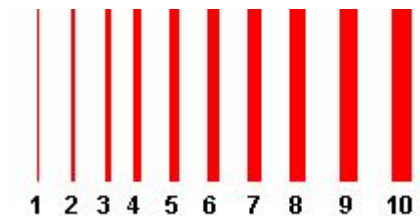
Устанавливает значение свойства в значение по умолчанию.

inherit

Наследует значение свойства от родительского элемента.

## Ширина рамки - border-width

Ширина рамки задается с помощью единиц измерения длины или ключевых слов. Если для свойства `border-style` задано значение `none`, и для рамки элемента установлена какая-то ширина, то в данном случае ширина рамки приравнивается к нулю. Свойство не наследуется.



# Ширина рамки - border-width

## border-width

(border-top-width, border-right-width, border-bottom-width, border-left-width)

### Значения:

`thin / medium`  
`/ thick`

Ключевые слова, устанавливают ширину рамки относительно друг друга. Первое значение уже, чем второе, второе — тоньше третьего. Значение по умолчанию — `medium`

`width (px, em)`

```
{border-width: 5px;}
```

`{1, 4}`

Возможность одновременного задания четырех разных ширин для рамок элемента, только для свойства `border-width` :

```
{border-width: 5px 10px 15px 3px;}
```

`initial`

Устанавливает значение свойства в значение по умолчанию.

`inherit`

Наследует значение свойства от родительского элемента.

## Цвет рамки - border-color

Свойство задаёт цвет рамок всех сторон одновременно. С помощью уточняющих свойств можно установить свой цвет для рамки каждой стороны элемента. Если для рамки цвет не задан, то он будет таким же, как и цвет текста элемента. Если в элементе нет текста, то цвет рамки будет таким же, как и цвет текста родительского элемента. Свойство не наследуется.



# Цвет рамки - border-color

## border-color

(border-top-color, border-right-color, border-bottom-color, border-left-color)

### Значения:

transparent

Устанавливает прозрачный цвет для рамки. При этом ширина рамки остается. Можно использовать для смены цвета рамки при наведении курсора мыши на элемент, чтобы избежать смещение элемента.

цвет

Цвет рамок задается при помощи значений свойства `color`.

```
{border-color: #cacd58;}
```

{1, 4}

Одновременное перечисление четырех разных цветов для рамок элемента, только для свойства `border-color`:

```
{border-color: #cacd58 #5faf8a #b9cea5 #aab238;}
```

initial

Устанавливает значение свойства в значение по умолчанию.

inherit

Наследует значение свойства от родительского элемента.

## Задание рамки одним свойством

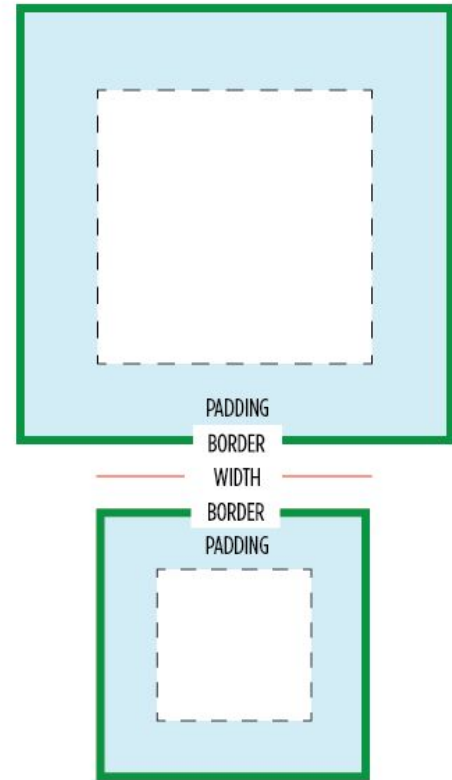
Свойство `border` позволяет объединить в себе следующие свойства: `border-width`, `border-style`, `border-color`, например:

```
div {  
  
    width: 200px;  
  
    height: 300px;  
  
    border: 1px solid grey;  
  
}
```

*Примечание:* Если какое-то из значений не указано, его место займет значение по умолчанию.

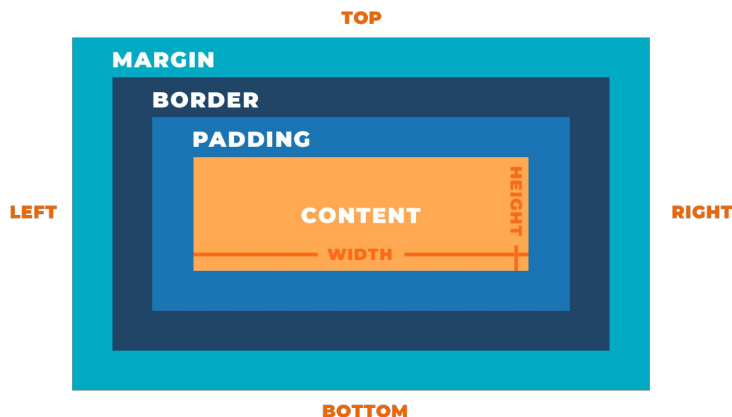


# box-sizing

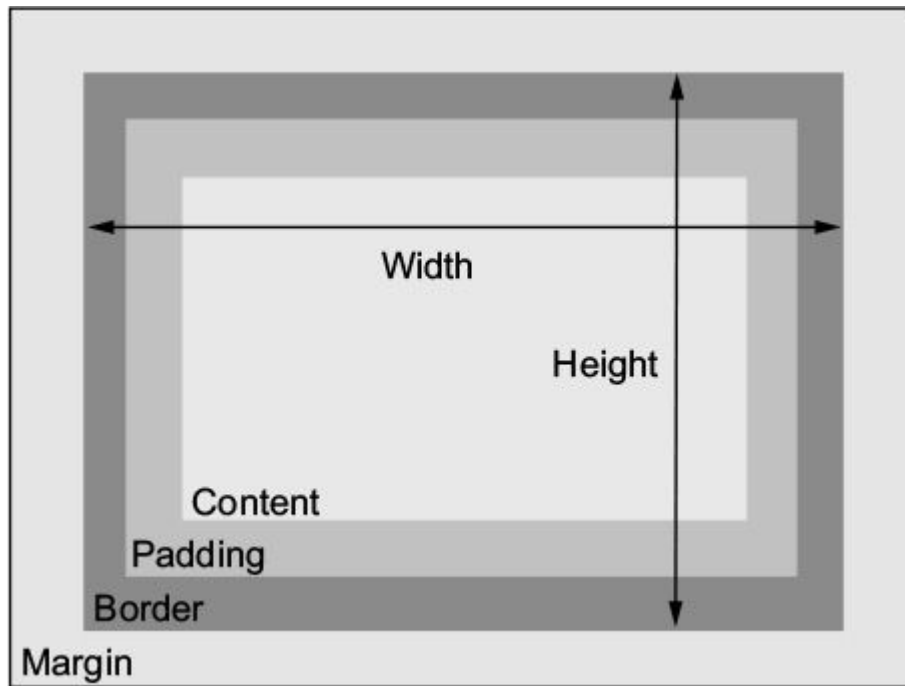


Свойство `box-sizing` в CSS определяет, как браузер рассчитывает общую ширину и высоту элемента, учитывая его содержимое, отступы (`padding`), границы (`border`) и внешние отступы (`margin`).

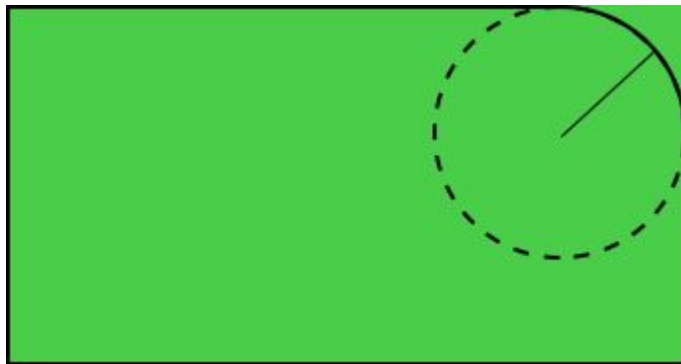
По умолчанию значение равно: `box-content` - ширина и высота, которые будут заданы для элемент, будут включать только контент



Когда свойство `box-sizing` установлено в значение `border-box`, размеры элемента включают в себя содержимое, отступы и границы, но не включают внешние отступы. Это позволяет более прозрачно управлять размерами элементов



# Скругление границ



## border-radius

в CSS это свойство, которое используется для задания скругления углов для границы элемента. Оно позволяет создавать элементы с закругленными углами, делая интерфейс более эстетичным и приятным для восприятия.

```
border-radius: <размер>;
```

Значение <размер> указывает радиус скругления угла в пикселях (px), процентах (%) или других единицах измерения.

# border-radius

- Указание одного значения для всех углов

```
.element {  
    border-radius: 10px;  
}
```

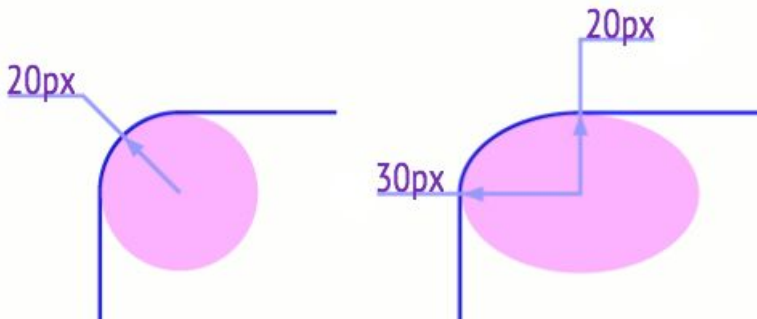
- Указание значений для каждого угла по отдельности (по часовой стрелке, начиная с верхнего левого угла):

```
.element {  
    border-radius: 10px 20px 30px 40px;  
}
```

# border-radius

- Указание значений для горизонтальных и вертикальных углов:

```
.element {  
  border-radius: 10px 20px 10px 20px / 20px 10px 20px 10px;  
}
```





# **Ваша новая IT-профессия – Ваш новый уровень жизни**

Программирование с нуля в  
немецкой школе AIT TR GmbH