

ESS



CSS – Cascading Style Sheets

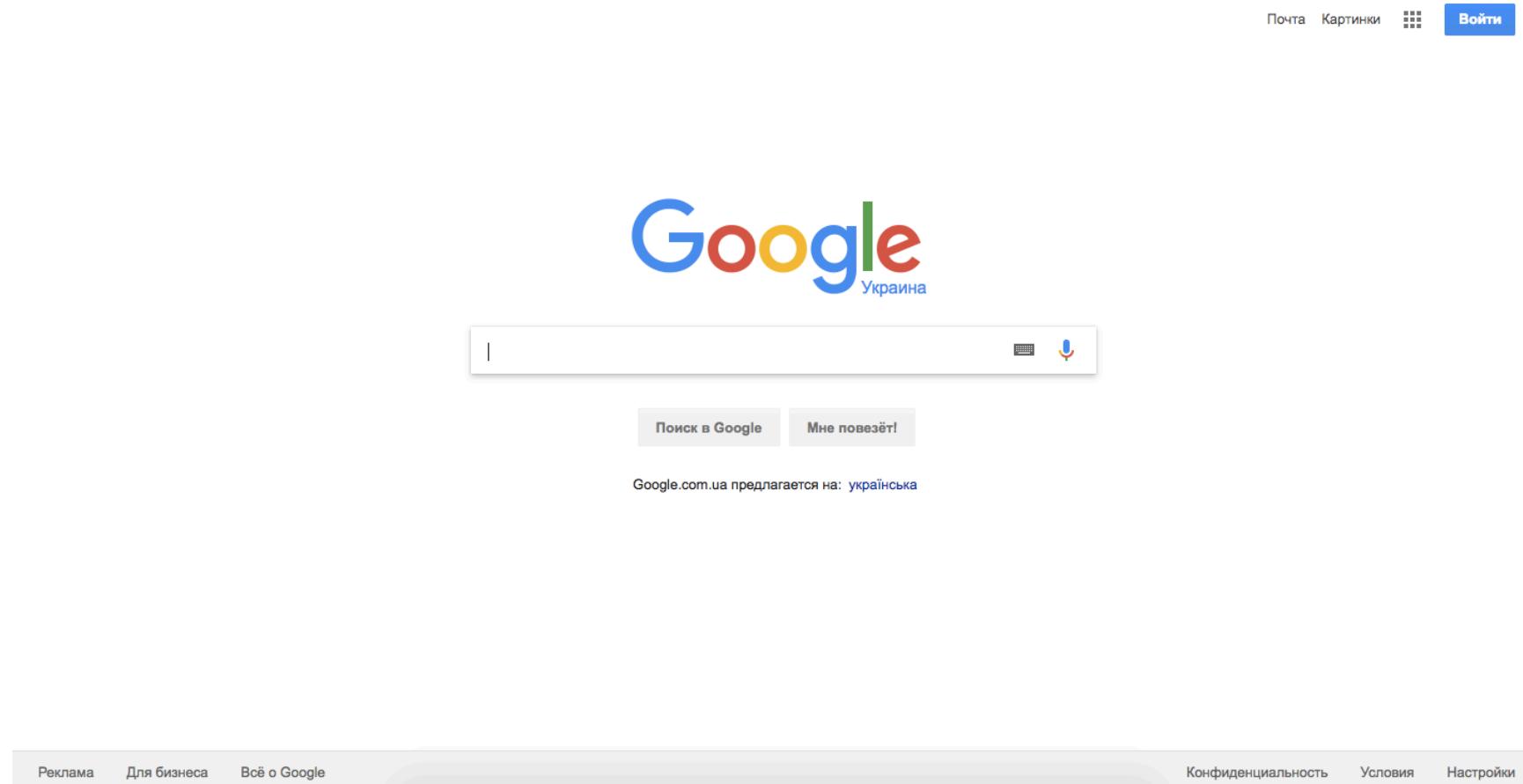
CSS – это язык описания внешнего вида документа.

CSS описывает, как будут выглядеть HTML элементы на странице.

HTML определяет контент, CSS определяет внешний вид контента.

Внешний вид подразумевает настройку цвета, шрифтов, размеров, полей, положения элементов веб-страницы.

HTML + CSS



HTML

- Контакты
- [Hangouts](#)
- [Google Keep](#)

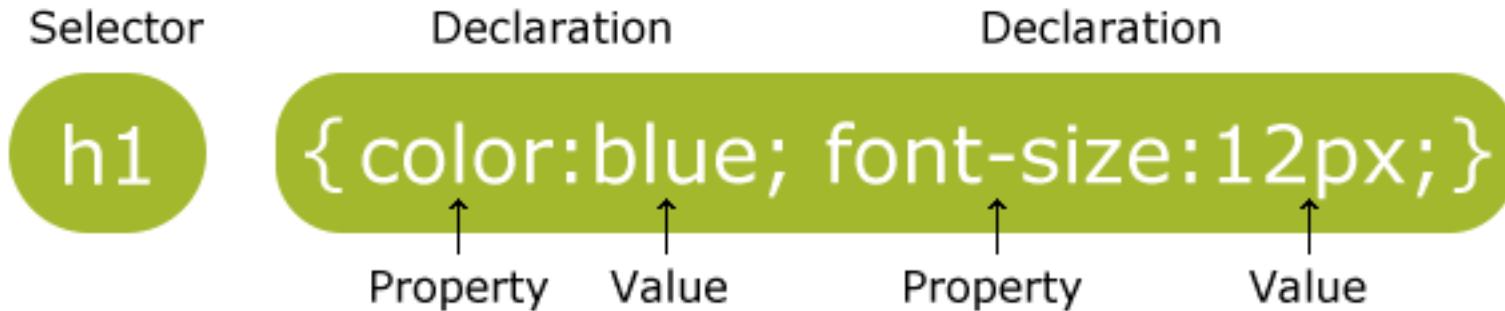
Другие сервисы Google
Более



[Поиск в Google](#) [Мне повезёт!](#)

Украина
Google

Синтаксис



Selector – HTML элемент, к которому будут применены стили
Property – свойство для изменения (цвет, размер, отступ и т.д.)
Value – определяет, как изменить свойство.

Весь блок (селектор, свойство, значение) является **правилом CSS**.

CSS комментарии

```
/* Это комментарий CSS */  
div {  
    background: blue;  
    color: red;  
    height: 100px;  
}  
/* Комментарии предназначены только для чтения  
людьми и браузер их разбирать не будет */
```

Способы подключения CSS

- Стили задаются через атрибут **style** у элемента

```
<h1 style="color:blue">This is a heading</h1>
```

- Стили пишутся внутри тега **<style>**

```
<style>h1{ color: orange; }</style>
```

- Стили находятся в отдельном файле с расширением .css, на который ссылается элемент **<link>**.

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

Базовые селекторы тегов

```
a { /* Ссылки */ }
```

```
p { /* Абзацы */ }
```

```
ul { /* Неупорядоченные списки */ }
```

```
li { /* Пункты списка */ }
```

Классы

```
<p class="date">  
    21 февраля, суббота  
</p>
```

```
.date {  
    color: red;  
}
```

Идентификаторы

```
<h1 id="orange">
```

Этот заголовок будет оранжевым.

```
</h1>
```

```
#orange {  
    color: red;  
}
```

Объединение селекторов

HTML	Возможные селекторы	Что это значит
<p></p>	p	Каждый <p>
<div class="global"></div>	div .global div.global	Каждый <div> Каждый элемент с class="global" Каждый <div> с class="global"
<ul id="menu">	#menu ul#menu	Только элемент с id="menu" Только элемент с id="menu"
<ol class="dico"> Un cobaye Des cobaux 	li ol li .dico li	Каждый Каждый внутри предка Каждый внутри предка с class="dico"

Селектор потомка

```
article h2 {  
    /* ... */  
}
```

```
<h2>...</h2>  
<article>  
    <h2>Этот заголовок будет выбран</h2>  
    <div>  
        <h2>Этот заголовок будет выбран</h2>  
    </div>  
</article>
```

Прямой дочерний селектор

```
article > p {  
    /* ... */  
}
```

```
<p>...</p>  
<article>  
    <p>Этот абзац будет выбран</p>  
    <div>  
        <p>...</p>  
    </div>  
</article>
```

Обзор дочерних селекторов

Пример	Название	Описание
article h2	Селектор потомка	Выбирает элемент, который находится в любом месте внутри определённого предка.
article > p	Прямой дочерний селектор	Выбирает элемент, который находится непосредственно внутри определённого родителя.

Селекторы атрибутов

a[target]	Селектор наличия атрибута	Выбирает элемент если данный атрибут присутствует.
a[href="http://google.com/"]	Селектор атрибута =	Выбирает элемент, если значение данного атрибута в точности соответствует указанному.
a[href*="login"]	Селектор атрибута *=	Выбирает элемент, если значение данного атрибута содержит по крайней мере один экземпляр указанного текста.

Список селекторов по атрибутам

https://www.w3schools.com/css/css_attribute_selectors.asp

Селекторы псевдоклассов

HTML элементы могут иметь разные состояния.

```
a {  
    color: blue;  
}
```

```
a:hover {  
    color: red;  
}
```

Приоритет селекторов

```
<p class="message" id="question">
```

Какого цвета будет текст ?

```
</p>
```

```
p { color: blue; }
```

```
.message { color: green; }
```

```
# question{ color: red; }
```

Браузер может выбрать только один цвет.

Быстрый способ выяснить приоритет **селектора**:

- **идентификаторы** стоят 100;
- **классы** стоят 10;
- селекторы **тега** стоят 1.

Селектор с наивысшим «счётом» будет преобладать, независимо от порядка, в котором появляются правила CSS.

Если в CSS есть одинаковые селекторы, то последний из них будет иметь приоритет.

Повторяем селекторы

- `*` – любые элементы.
- `div` – элементы с указанным тегом.
- `#id` – элемент с данным id.
- `.class` – элементы с указанным классом.
- `[name="value"]` – селекторы по атрибутам.
- `:visited` – «псевдоклассы»

Повторяем селекторы

- `.c1.c2` – элементы одновременно с двумя классами `c1` и `c2`
- `a#id.c1.c2:visited` – элемент `a` с данным `id`, классами `c1` и `c2`, и псевдоклассом `visited`
- `div p` – элементы `p`, являющиеся потомками `div`.
- `div > p` – только непосредственные потомки

Единицы размеров в CSS

В CSS много свойств, которые требуют **размер** в качестве значения:

Наиболее часто используемые единицы:

- **px** для пикселей;
- **%** для процентов;
- **em** для определения размера относительно родительского значения **font-size**.
- **rem** похож на **em**, но вместо зависимости от родительского значения, опирается на значение **корневого элемента**

Единицы цвета в CSS

Цвета широко используются в CSS для изменения цвета текста, фона, градиентов, теней, границ и др.

Цвета обычно задают в таких форматах

- Шестнадцатеричное значение - `#db4e44`
- В формате RGB - `rgb(219, 78, 68)`
- В формате RGBA - `rgba(0, 0, 0, 0.8)`
- Название цвета – `red, green, blue ...`

CSS Текст

- **color** (цвет)
- **text-align** (left, right, center, justify)
- **text-decoration** (overline, line-through, underline, none)
- **text-transform** (uppercase, lowercase, capitalize, none)
- **text-indent** (размер)
- **line-height** (размер)
- **letter-spacing** (размер)
- **word-spacing** (размер)

CSS Шрифты

- **font-family** (шрифт)
- **font-size** (размер)
- **font-style** (normal, italic, oblique)
- **font-variant** (normal, small-caps)
- **font-weight** (bold, bolder, lighter, normal, 100 – 900)

F

Sans-serif

F

Serif

F

Serif
(red serifs)

Наследование в CSS

Есть ряд свойств, которые могут быть унаследованы от предков.

Это в основном **текстовые** свойства:

- цвет текста;
- шрифт (семейство, размер, стиль, насыщенность);
- межстрочное расстояние.

Например значение `color` может быть унаследовано от предка.

`body { color: grey; }` сделает серый шрифт у всех элементов.

Блочная модель

Каждый элемент на странице представляет собой прямоугольный блок и может иметь **ширину, высоту, границы и отступы**.

CSS свойства связанные с блочной моделью:

- Изменяют поведение элементов на странице (изменяют тип)
- Изменяют размер элементов и занимаемую ими площадь

Типы элементов

Основные

- блочные
- строчные

Дополнительные

- блочно-строчные
- табличные
- другие ...

Блочные элементы

Блочными элементами являются:

`<div> <h1>-<h6> <p> <form>`
`<article> <section> <aside> <header> <footer>`

И другие теги, которые обычно служат контейнерами для других блоков или для большого количества текста.

Особенности блочных элементов

- начинаются с новой строки (перенос строки до и после)
- занимают все доступное по ширине пространство
- к ним можно применять ширину, высоту, внешние отступы
- высота подстраивается под содержимое элемента

Строчные элементы

Строчными элементами являются:

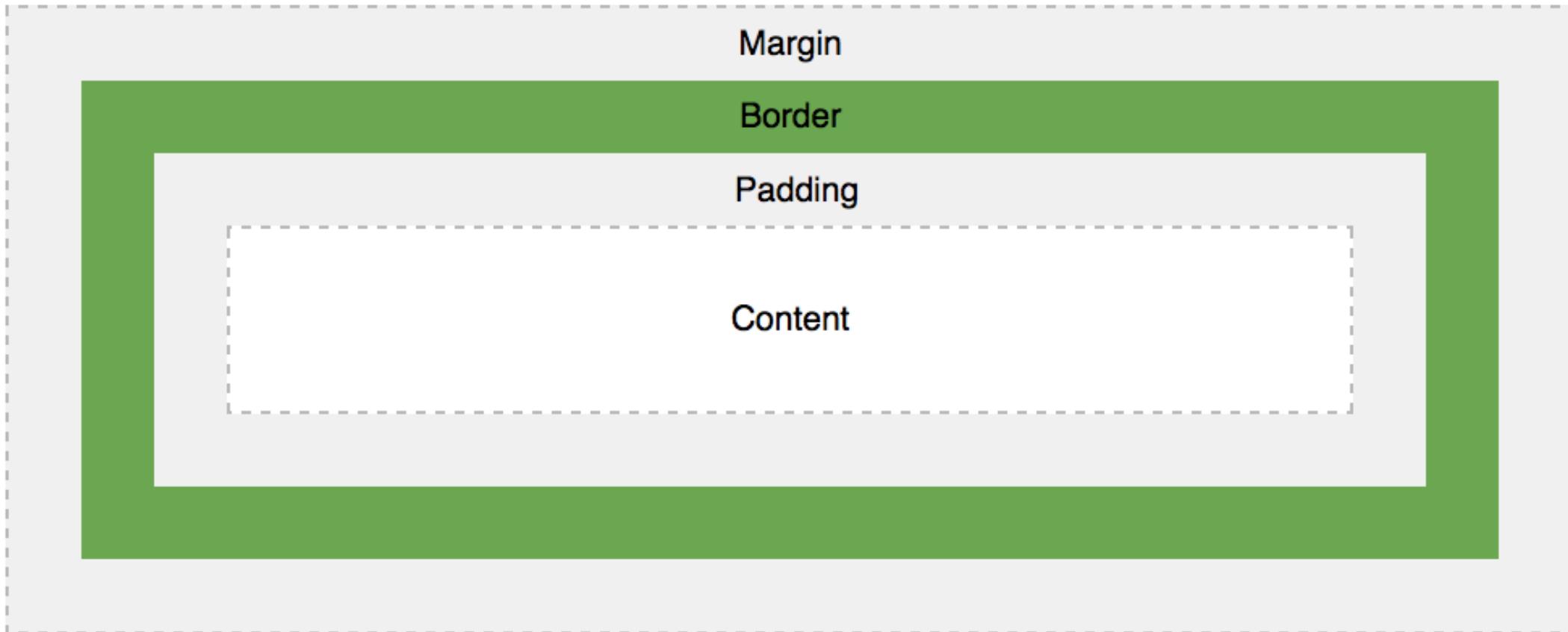
` <a> <i> `
`<q> <code> <mark>`

И другие теги, которые обычно содержат в себе небольшие фрагменты текста.

Особенности строчных элементов

- нет переноса строки
- ширина и высота зависят только от контента
- нельзя задать размеры с помощью CSS
- нельзя задать верхние и нижние **внешние** отступы

Блочная модель



Размеры элемента

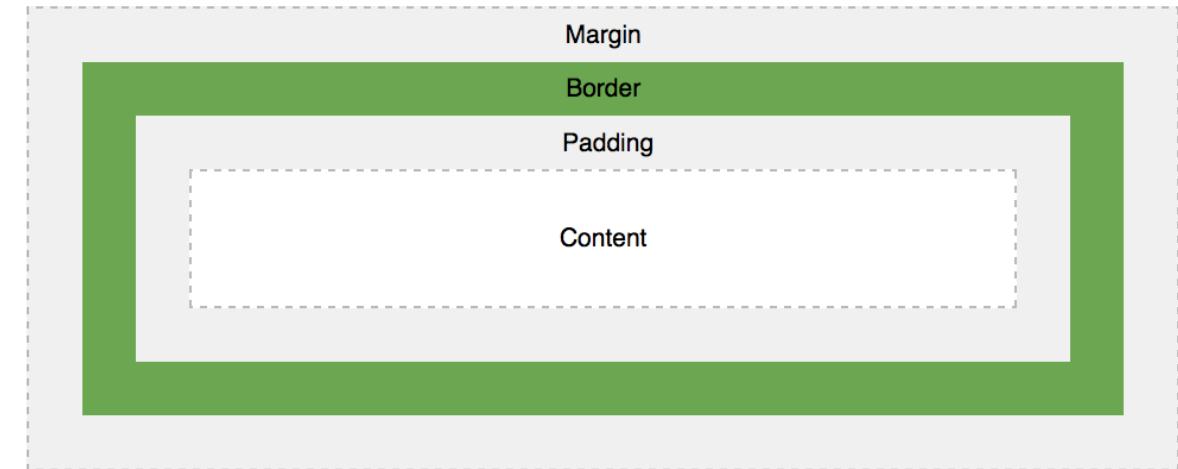
- height
- width
- max-height
- min-height
- max-width
- min-width

```
div {  
    height: 200px;  
    width: 200px;  
    background: red;  
}
```



Внутренние и внешние отступы

- padding
- margin
- margin-top / padding-top
- margin-right / padding-right
- margin-bottom / padding-bottom
- margin-left / padding-left



Рамки

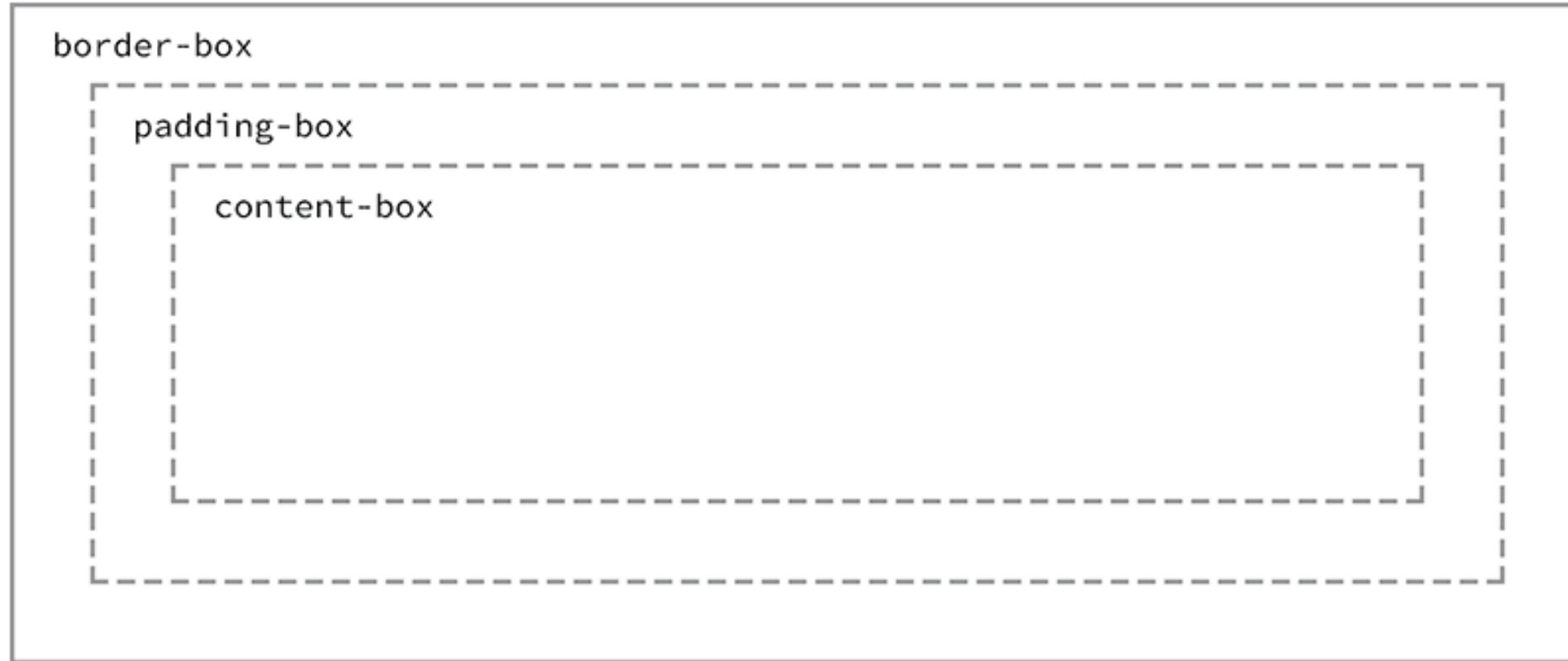
- border
- border-width
- border-style (solid, dotted, dashed ...)
- border-color



```
div {  
    border: 6px solid #949599;  
}
```

Свойство box-sizing

Для изменения алгоритма расчёта ширины и высоты элемента.



Свойство `display`

`display` позволяет изменить тип элемента.

Наиболее используемые значения:

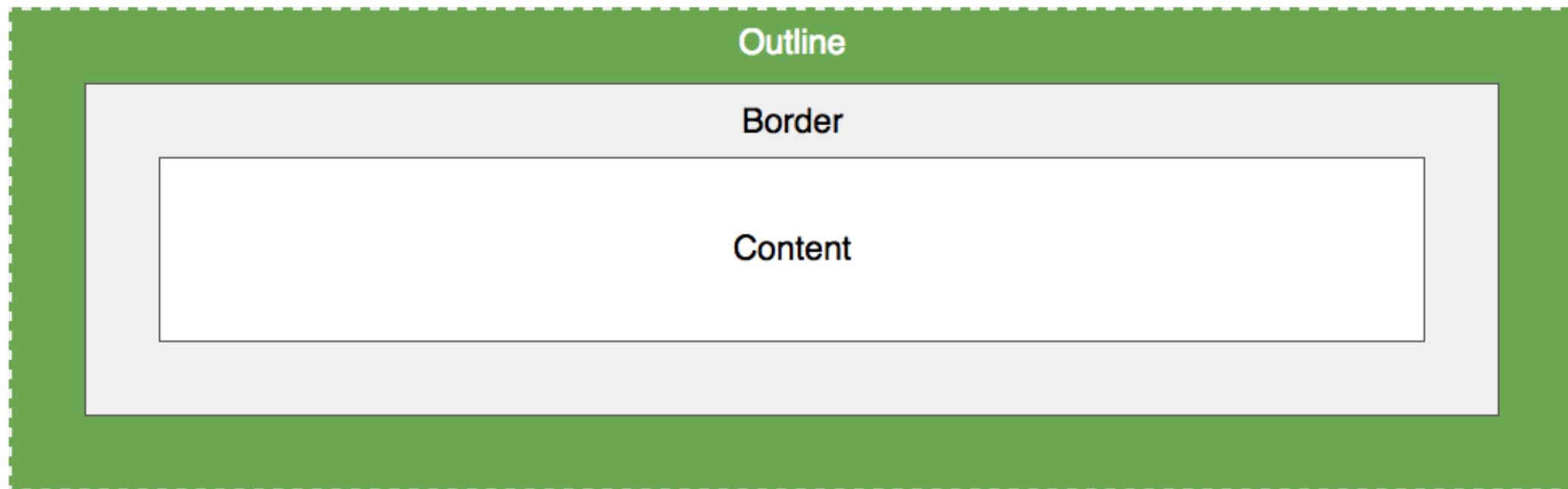
- `block` – элемент отображается как блочный
- `inline` – элемент отображается как строчный
- `inline-block` – генерирует блочный элемент, который обтекается другими элементами страницы подобно строчному элементу
- `none` - удаляет элемент со страницы

Фон

- **background-color** (цвет)
- **background-image** (путь к файлу)
- **background-repeat** (repeat, no-repeat, repeat-x, repeat-y, round)
- **background-attachment** (fixed, scroll, local)
- **background-position** (left, center, right | top, center, bottom или размер)
- **background-size** (auto, cover, contain, размер)
- **background-origin** (padding-box, border-box, content-box)
- **background-clip** (padding-box, border-box, content-box, text)

Свойство outline

Свойство **outline** не влияет на положение блока и его ширину.



Разметка страницы

☎ (044) 537-02-22, (044) 503-80-80, 0 800 503-808
 Вопросы и ответы Кредит Доставка и оплата Гарантия Контакты Отследить заказ Продавать на Розетке

Здравствуйте,  войдите в личный кабинет

 ROZETKA
город Запорожье По всем... Найти Сравнение Желания Корзина

Каталог товаров

- Ноутбуки и компьютеры >
- Смартфоны, ТВ и электроника >
- Бытовая техника >
- Товары для дома >
- Инструменты и автотовары >
- Сантехника и ремонт >
- Дача, сад и огород >
- Спорт и увлечения >
- Одежда, обувь и украшения >
- Красота и здоровье >
- Детские товары >
- Канцтовары и книги >
- Алкогольные напитки и продукты >
- Товары для бизнеса >
- Билеты, отели, платежи и переводы >
- Скидки ко Дню св. Патрика! ☘
- Все категории

Топ акций С подарками (208) Со скидками (255) Розыгрыш призов (15)

Акция

19 199 грн

- 4K
- HDR
- Smart Hub



14 марта - 2 апреля

Кредит 0%* на 24 месяца на телевизор Samsung UE43KU6000UXUA + бесплатная доставка!

* Подробнее на странице акции

Купить

 Ноутбуки	 Смартфоны	 Бытовая химия	 Продукты	 Одежда и обувь
 Алкогольные напитки	 Косметика и парфюмерия	 Нижнее белье	 Ювелирные украшения	 Детский мир

(044) 537-02-22, (044) 503-80-80, 0 800 503-808

Здравствуйте,  войдите в личный кабинет

Вопросы и ответы Кредит Доставка и оплата Гарантия Контакты Отследить заказ Продавать на Розетке



РОЗЕТКА
интернет-супермаркет

Город
Запорожье

По все...



Сравнение



Желания



Корзина

Каталог товаров

- [Ноутбуки и компьютеры](#)
- [Смартфоны, ТВ и электроника](#)
- [Бытовая техника](#)
- [Товары для дома](#)
- [Инструменты и автотовары](#)
- [Сантехника и ремонт](#)
- [Дача, сад и огород](#)
- [Спорт и увлечения](#)
- [Одежда, обувь и украшения](#)
- [Красота и здоровье](#)
- [Детские товары](#)
- [Канцтовары и книги](#)
- [Алкогольные напитки и продукты](#)
- [Товары для бизнеса](#)
- [Билеты, отели, платежи и переводы](#)

[Скидки ко Дню св. Патрика! !\[\]\(a04c5602c87b51151a5d2267e8426a6d_img.jpg\)](#)

[Все категории](#)

Топ акции

С подарками (208)

Со скидками (255)

Розыгрыш призов (15)



Акция

19 199 грн

- 4K
- HDR
- Smart Hub

14 марта - 2 апреля

Кредит 0%* на 24 месяца
на телевизор
Samsung UE43KU6000UXUA
+ бесплатная доставка!

* подробнее на странице акции



Ноутбуки



Смартфоны



Бытовая химия



Продукты



Одежда и обувь



Алкогольные напитки



Косметика и парфюмерия



Нижнее белье



Ювелирные украшения



Детский мир

**Смартфон уже есть?
Подари портативную
батарею!**



ЦИТРУС
гаджети та аксесуари

Поиск Например: iPhone 6s 🔍

Оформить заказ:
0 800 20 70 20
Служба поддержки клиентов:
0 800 20 70 30

Пусто :)

Apple Samsung Meizu Xiaomi | Смартфоны Планшеты и ультрабуки Смарт-часы Наушники TV Акустика Аксессуары

Персональный транспорт Фитнес и здоровье Фото и видео Умный дом Развлечения Селфи-стикеры Батареи Подарки **SALE**

Только 1 - 31 марта



**ГРАНДИОЗНАЯ
РАСПРОДАЖА СМАРТ-ЧАСОВ**
СКИДКИ до **3000** ГРИВЕН

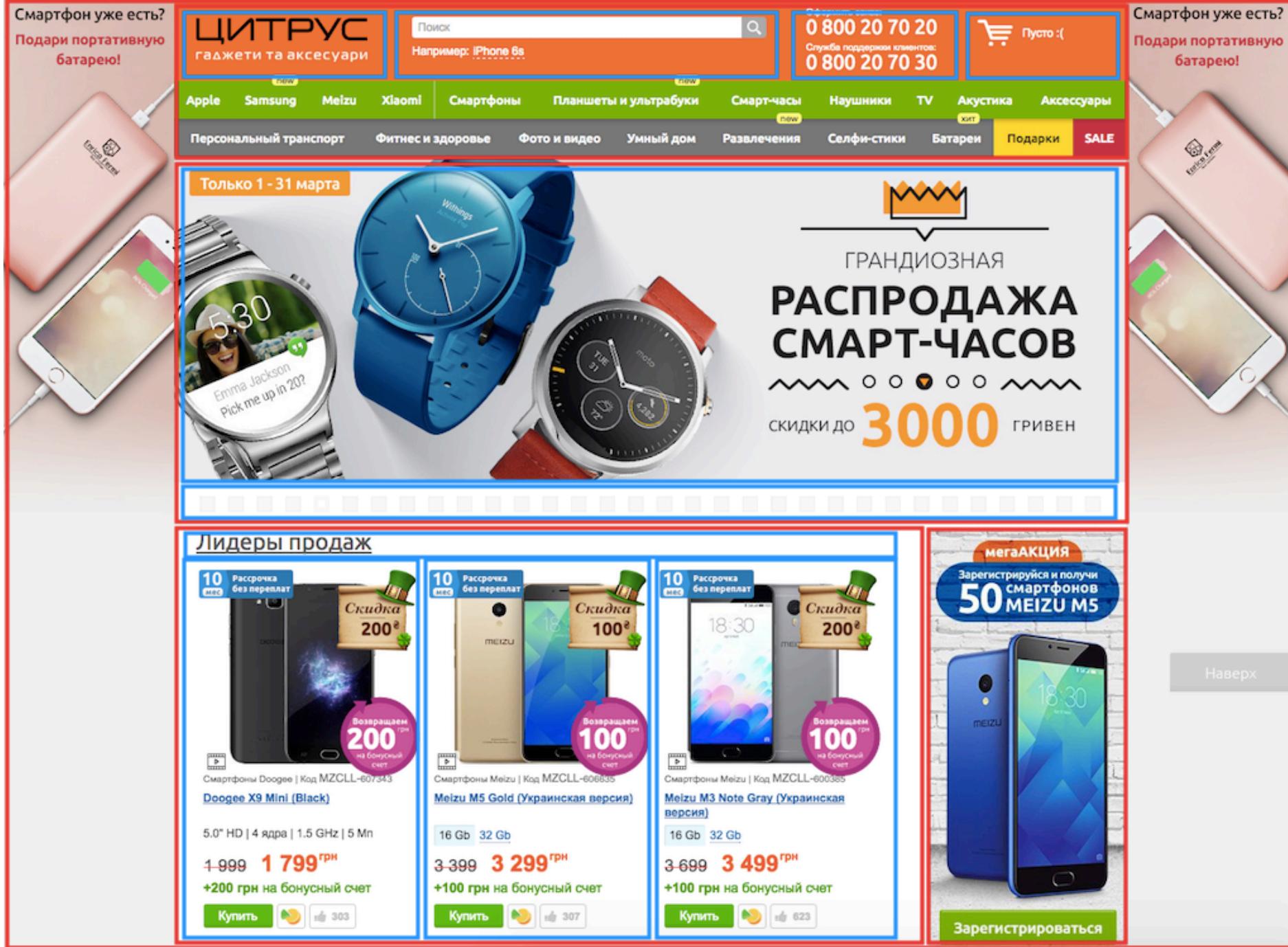
Лидеры продаж

Смартфон	Цена	Скидка	Описание
Doogee X9 Mini (Black)	1 799 грн	+200 грн на бонусный счет	10 мес. рассрочка без переплат
Meizu M5 Gold (Украинская версия)	3 299 грн	+100 грн на бонусный счет	10 мес. рассрочка без переплат
Meizu M3 Note Gray (Украинская версия)	3 499 грн	+100 грн на бонусный счет	10 мес. рассрочка без переплат

мегАКЦІЯ
Зарегистрируйся и получи **50 смартфонов MEIZU M5**

Наверх

Смартфон уже есть?
Подари портативную
батарею!



ПОТОК ДОКУМЕНТА

```
<body>
  <div class="header"></div>
  <div class="column1"></div>
  <div class="column2"></div>
  <div class="column3"></div>
  <div class="footer"></div>
</body>
```

.header

.column1

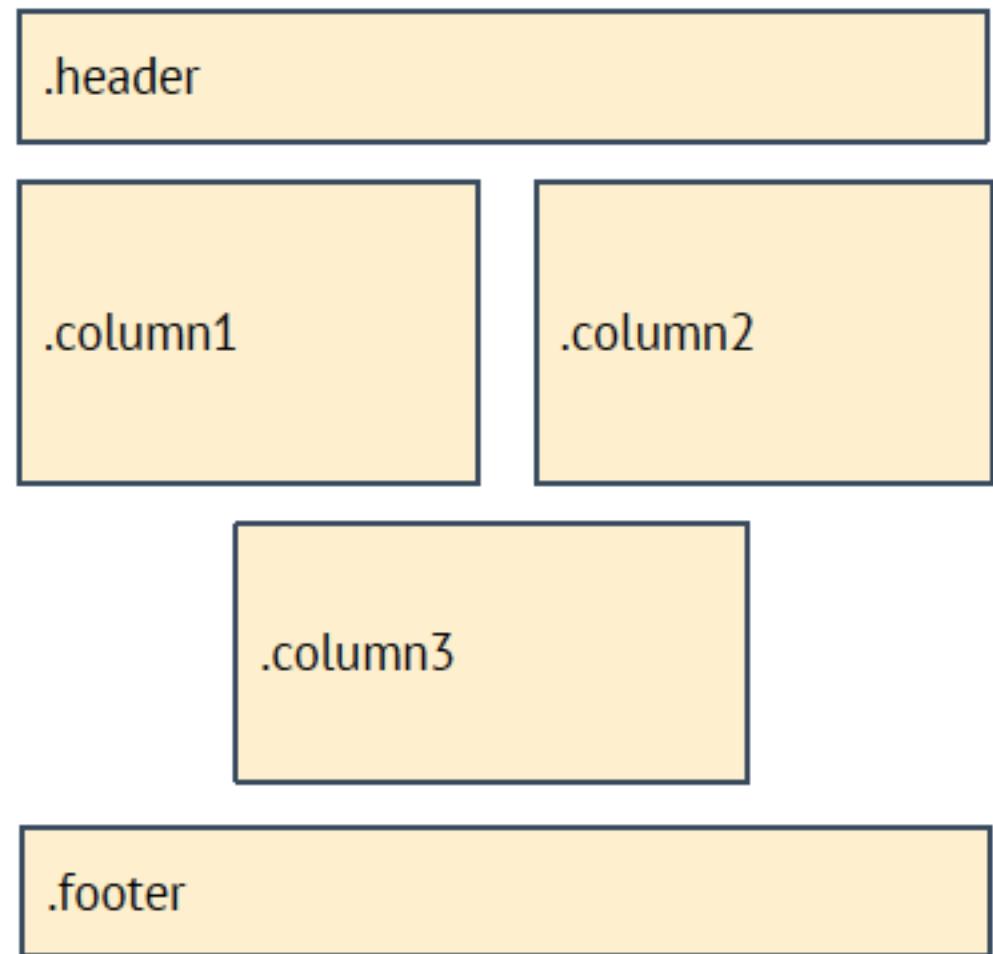
.column2

.column3

.footer

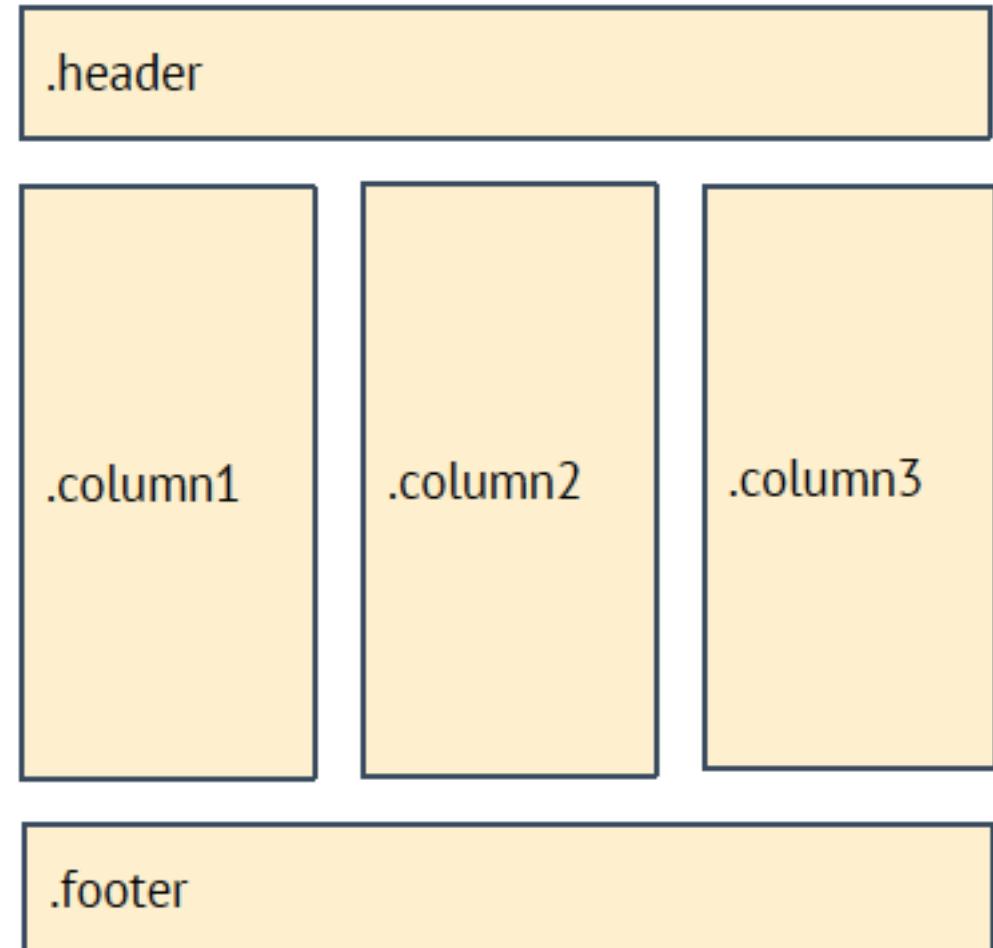
ПОТОК ДОКУМЕНТА

```
<body>
  <div class="header"></div>
  <div class="column1"></div>
  <div class="column2"></div>
  <div class="column3"></div>
  <div class="footer"></div>
</body>
```



ПОТОК ДОКУМЕНТА

```
<body>
  <div class="header"></div>
  <div class="column1"></div>
  <div class="column2"></div>
  <div class="column3"></div>
  <div class="footer"></div>
</body>
```



Поток документа

Управляя потоком можно создавать необходимую разметку.

Для управления потоком, нужно знать:

- блочную модель документа (✓)
- свойства, которые управляют поведением элементов в потоке
 - `display` (✓)
 - `float`
 - `clear`
 - `position`

Свойство float

Определяет, по какой стороне будет выравниваться элемент, при этом остальные элементы будут обтекать его с других сторон.

Последующие строчные элементы обтекают плавающий элемент

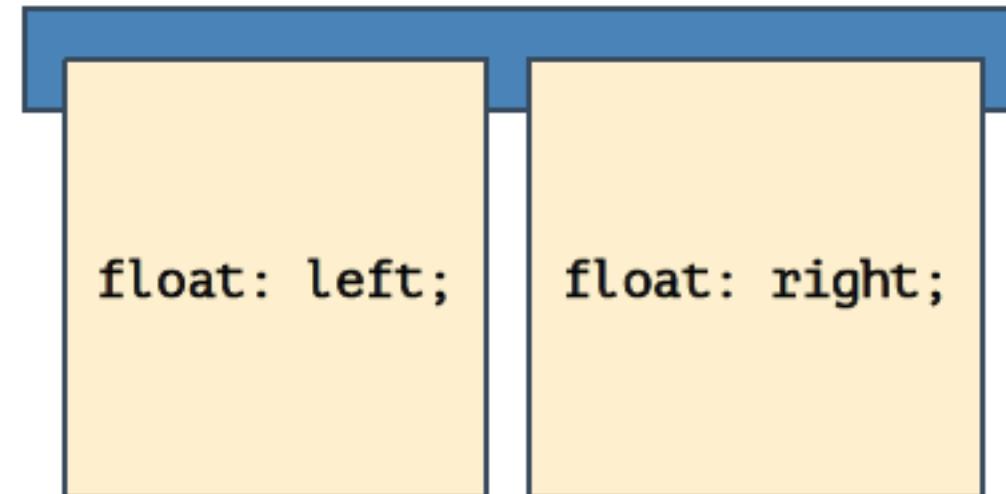
Значения:

- left
- right
- none



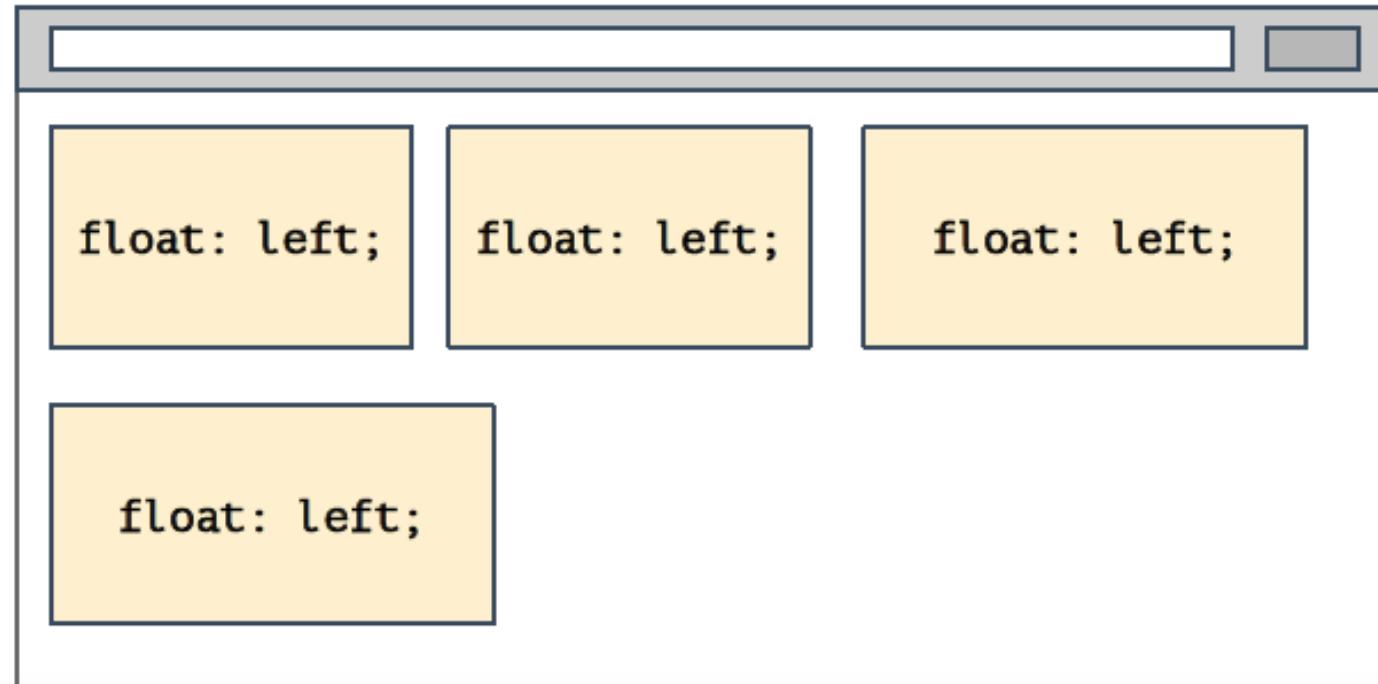
Свойство float

- Плавающий элемент частично выпадает из потока
- Последующие блочные элементы не видят плавающий элемент
- Плавающие элементы выпадают из родительских блоков



Свойство float

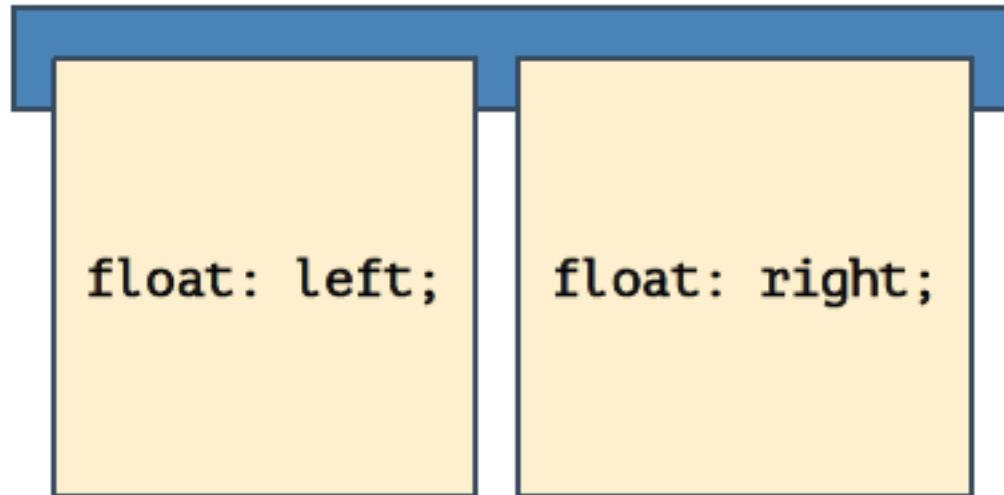
- Плавающие элементы располагаются в ряд друг за другом
- Если места не достаточно, то становятся ниже



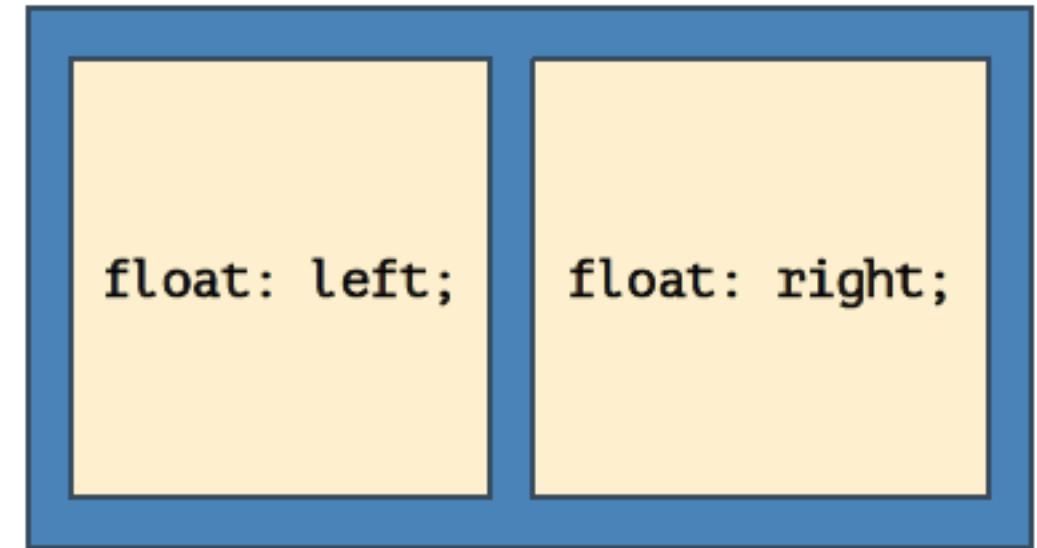
Свойство clear

Плавающие элементы позволяют создавать колонки, но проблема в том, что они выпадают из родителя.

Без clear



С clear

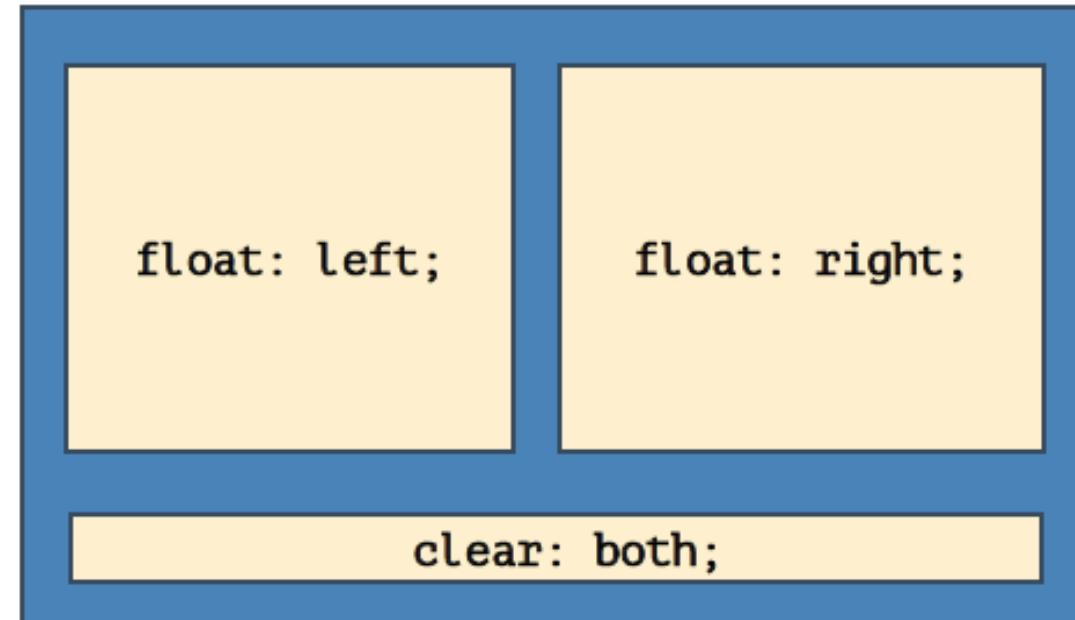


Свойство clear

clear применяется к элементу расположенному после плавающих.

Доступные значения

- left
- right
- both
- none



clearfix

```
.container:after {  
    content: '';  
    display: block;  
    clear: both;  
}
```

Свойство position

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

Возможные значения

- relative
- absolute
- fixed
- static

Координаты элемента

position часто используют с четырьмя свойствами координат:

- **left**
- **right**
- **top**
- **bottom**

При значении **static** у свойства **position**, вышеперечисленные свойства игнорируются.

relative

При **относительном** позиционировании, элемент может перемещаться относительно его **текущей позиции**.

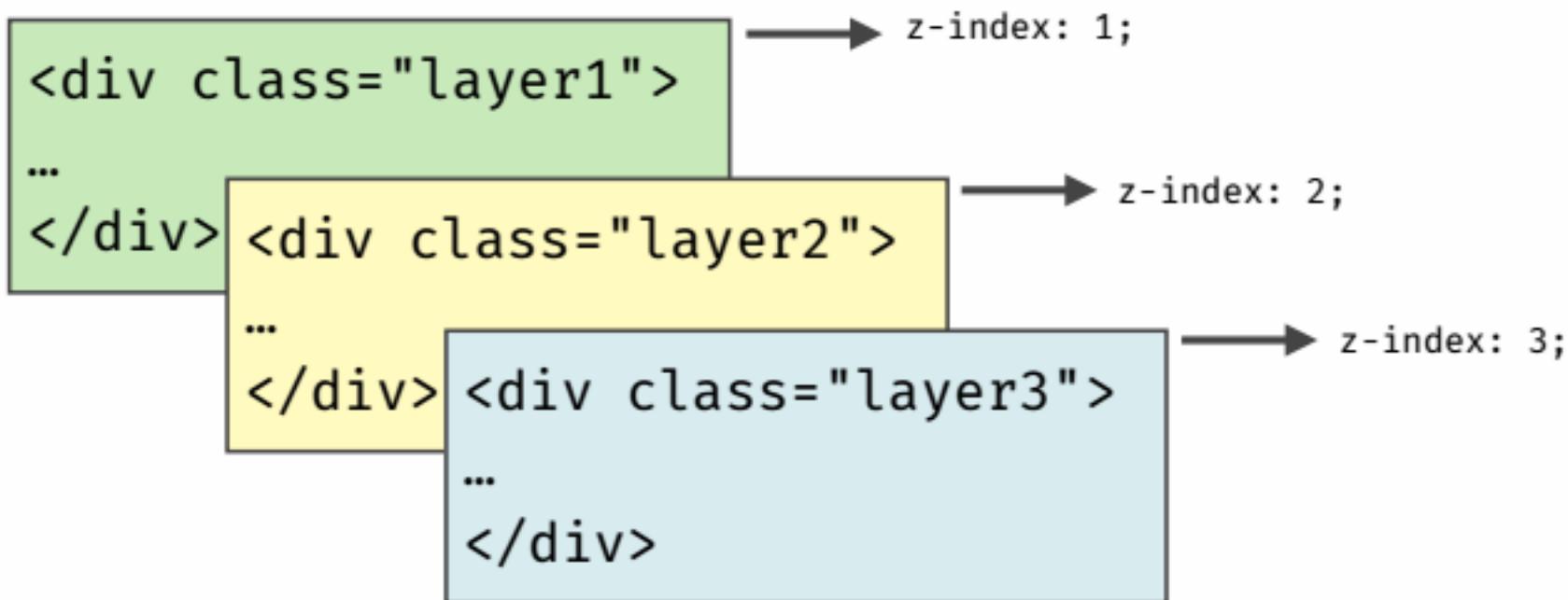
absolute

При **абсолютном** позиционировании, элемент перемещаться относительно **первого позиционированного предка**([relative](#), [absolute](#) или [fixed](#)).

fixed

Поведение элемента, как и у [absolute](#), только фиксированный элемент не перемещается со страницей при прокрутке.

Свойство z-index



Вывод по позиционированным элементам

- Абсолютные и фиксированные элементы полностью выпадают из потока документа. Остальные элементы ведут себя так, как будто их нет.
- Позиционированный элемент выступает точкой отсчета для своих дочерних элементов.
- Свойства `left`, `right`, `top`, `bottom` задают координаты расположения позиционированного элемента
- `z-index` позволяет менять порядок наложения элементов.

СПИСКИ

- `list-style-type` (none, circle, disc, square ...)
- `list-style-image` (путь к изображению)
- `list-style-position` (outside, inside)

Убрать значения по умолчанию:

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}
```

CSS3

CSS3

- CSS3 это последняя версия стандарта CSS
- CSS3 обратно совместим с предыдущими версиями CSS
- Современные браузеры поддерживают практически все нововведения

Проверить поддержку браузером свойства можно на

<http://caniuse.com/>

Закругленные углы

```
#block1 {  
    border-radius: 25px;  
}
```

Rounded corners!

```
#block2 {  
    border-radius: 15px 50px;  
}
```



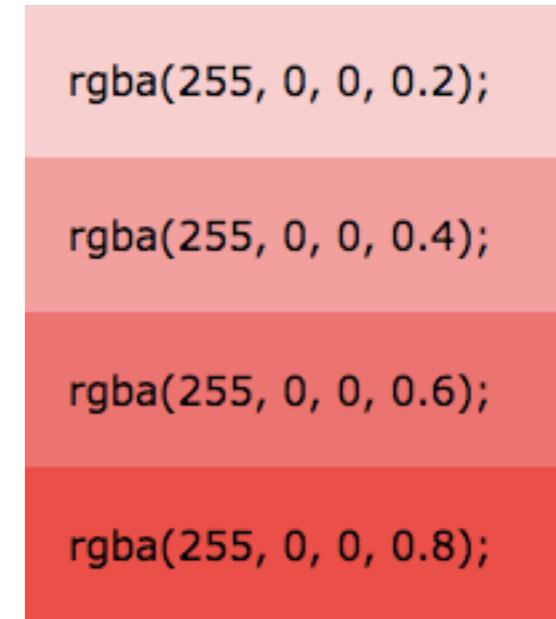
Прозрачность

opacity



прозрачный блок включая контент

rgba



прозрачный только фон

Градиенты

Gradient Background

```
#grad {  
  background: linear-gradient(red, yellow);  
}
```



```
#grad {  
  background: linear-gradient(to right, red, yellow);  
}
```



Позиция	Угол	Описание	Вид
to top	0deg	Снизу вверх.	
to left	270deg -90deg	Справа налево.	
to bottom	180deg	Сверху вниз.	
to right	90deg -270deg	Слева направо.	
to top left		От правого нижнего угла к левому верхнему.	
to top right		От левого нижнего угла к правому верхнему.	
to bottom left		От правого верхнего угла к левому нижнему.	
to bottom right		От левого верхнего угла к правому нижнему.	

Радиальный градиент

```
#grad {  
  background: radial-gradient(circle, red,  
yellow, green);  
}
```



Тени

- box-shadow
- text-shadow



With CSS3 you can create
shadow effects!

```
.shadow {  
    box-shadow: 0 0 5px 3px rgba(0,0,0,0.5);  
    /* сдвиг по x, сдвиг по y, размытие, растяжение, цвет */  
}
```

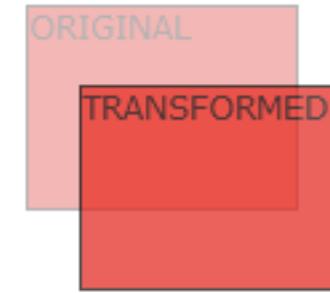
Шрифты @font-face

```
@font-face {  
    font-family: awesomeFont;  
    src: url(awesome_font.ttf);  
}  
  
div {  
    font-family: awesomeFont;  
}
```

With CSS3, web designers are no longer forced to use only "web-safe" fonts.

2D Трансформации

- translate(x, y)
- rotate(угол)
- scale(x, y)
- skewX(угол)
- skewY(угол)



```
div {  
    transform: translate(50px, 100px);  
}
```

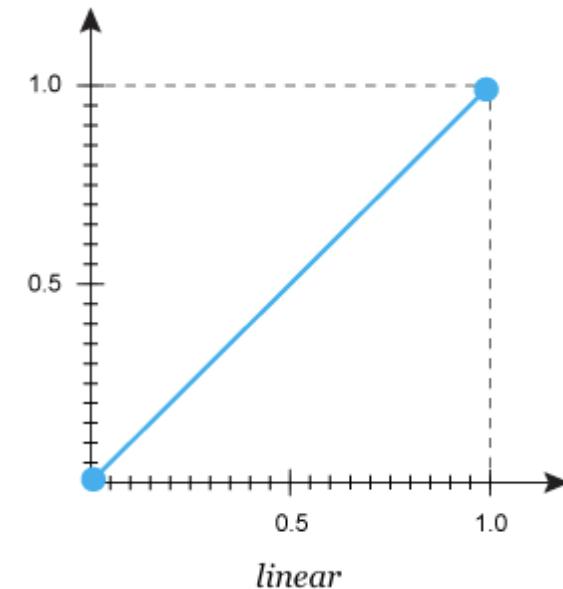
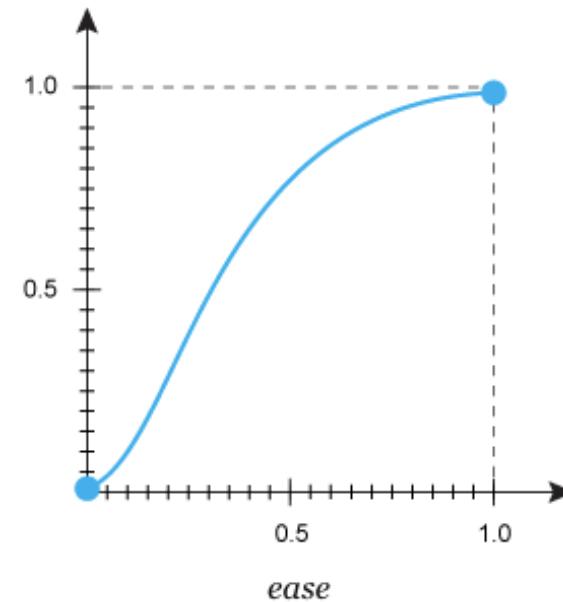
Плавные переходы

```
div {  
    transition-property: width;  
    transition-duration: 2s;  
    transition-timing-function: linear;  
    transition-delay: 1s;  
}
```

```
div {  
    transition: width 2s linear 1s;  
}
```

transition-timing-function

Функция, показывающая, как меняется указанное свойство.



<http://cubic-bezier.com/>

CSS анимации `@keyframes`

Правило `@keyframes` устанавливает ключевые кадры при анимации элемента.

Ключевой кадр - это свойства элемента (прозрачность, цвет, положение и др.), которые должны применяться к элементу в заданный момент времени.

Анимация - это плавный переход стилевых свойств от одного ключевого кадра к другому.

Пример описания двух кадров

```
@keyframes example {  
    from {  
        left: 0;  
    }  
    to {  
        left: 300px;  
    }  
}
```



Тот же пример

```
@keyframes example {  
  0% {  
    left: 0;  
  }  
  100% {  
    left: 300px;  
  }  
}
```



Можно задавать больше промежуточных состояний и делать анимации более сложными.

Свойство animation-name

Устанавливает анимации, которые применяются к элементу.

Представляет собой имя связанное с правилом [@keyframes](#).

```
div {  
    animation-name: example;  
}
```

Свойство animation-duration

Задаёт время в секундах или миллисекундах, сколько должен длиться один цикл анимации. По умолчанию значение равно 0s, это означает, что никакой анимации нет.

```
div {  
    animation-name: example;  
    animation-duration: 1s;  
}
```

Свойство animation-timing-function

Устанавливает, согласно какой функции времени должна происходить анимация каждого цикла между ключевыми кадрами.

```
div {  
    animation-name: example;  
    animation-duration: 1s;  
    animation-timing-function: linear;  
}
```

Свойство animation-delay

Устанавливает время ожидания перед запуском цикла анимации.

```
div {  
    animation-name: example;  
    animation-duration: 1s;  
    animation-timing-function: linear;  
    animation-delay: 0s;  
}
```

Свойство animation-iteration-count

Указывает, сколько раз проигрывать анимацию до её остановки.

```
div {  
    animation-name: example;  
    animation-duration: 1s;  
    animation-timing-function: linear;  
    animation-delay: 0s;  
    animation-iteration-count: infinite;  
}
```

Свойство animation-direction

Устанавливает направление движения анимации.

normal - анимация идёт с самого начала, после завершения цикла возвращается к исходному состоянию.

alternate - анимация идёт с начала до конца, затем плавно возвращается в исходное положение.

reverse - анимация идёт с конца цикла, после его завершения возвращается к исходному состоянию.

alternate-reverse - анимация идёт с конца до начала, затем плавно возвращается в исходное положение.

Все вместе

```
div {  
    animation-name: example;  
    animation-duration: 1s;  
    animation-timing-function: linear;  
    animation-delay: 0s;  
    animation-iteration-count: infinite;  
    animation-direction: alternate;  
}
```

Свойство animation

Сокращенный вариант предыдущей записи

```
div {  
    animation: example 1s linear 0s infinite alternate;  
}
```

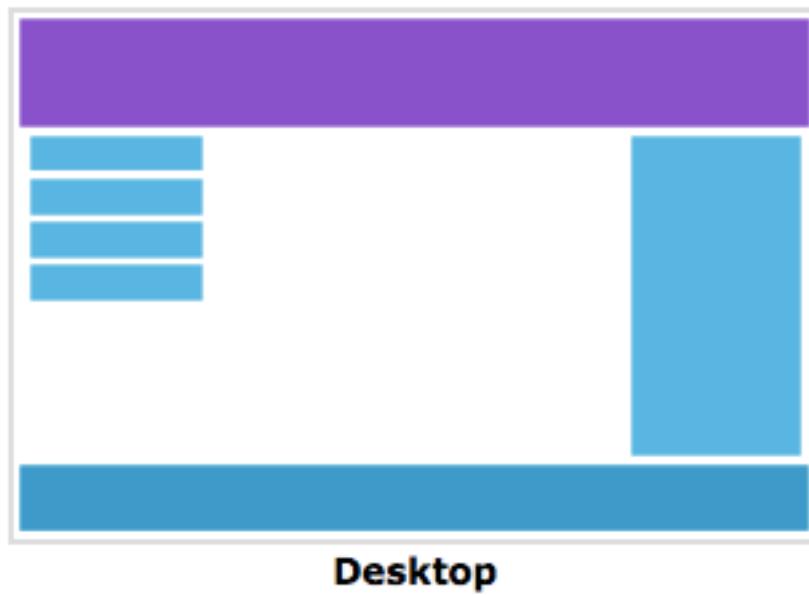
@keyframes + animation

```
@keyframes example {  
  from {  
    left: 0;  
  }  
  to {  
    left: 300px;  
  }  
}
```

```
div {  
  animation: example 1s linear 0s infinite alternate;  
}
```



Отзывчивый веб дизайн



Отзывчивый, адаптивный или резиновый?

Адаптивный дизайн – фиксированная разметка для разных типов устройств. Фиксированные размеры, которые меняются в медиазапросах.

Резиновый дизайн – резиновая разметка, которая тянеться или сужается в зависимости от ширины окна. Все размеры в % и относительных величинах.

Отзывчивый дизайн – адаптивность с резиновой разметкой.

Viewport

Viewport это та часть страницы, которую видит пользователь.

Видимая часть(viewport) варьируется от устройства к устройству.

```
<meta name="viewport"  
content="width=device-width, initial-scale=1.0">
```

Мета тег viewport позволяет нам контролировать размеры и масштабирование страницы.

Пример с мета тегом viewport и без него.



Grid (сетка)

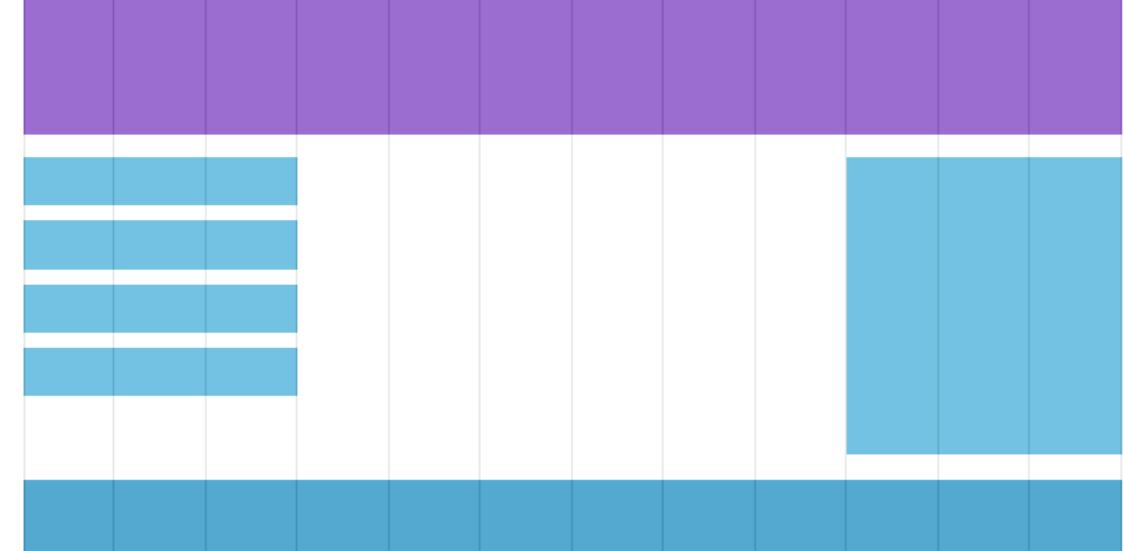
Систему grid используют практически все CSS фреймфорки.
Среди них [Bootstrap](#), [Foundation](#), [Materialize](#), [SemanticUI](#).

Во многих проектах bootstrap используется только ради **grid**.

Grid (сетка)

Страница разбита на колонки.

Ширина блоков определяется, колонками, которые они занимает.



Делаем свой grid

```
* {  
    box-sizing: border-box;  
}
```

Все внутренние отступы и рамки элементов будут входить в размер элементов.

Это позволит сделать отступы между колонками, при этом, не меняя их ширину.

Делаем свой grid

В нашей разметке будет **12** колонок.

Ширина колонки составит **$100\% / 12 = 8.33\%$**

Класс **col-n** будет определять, сколько колонок занимает элемент

.col-1 {width: 8.33%;}	.col-7 {width: 58.33%;}
.col-2 {width: 16.66%;}	.col-8 {width: 66.66%;}
.col-3 {width: 25%;}	.col-9 {width: 75%;}
.col-4 {width: 33.33%;}	.col-10 {width: 83.33%;}
.col-5 {width: 41.66%;}	.col-11 {width: 91.66%;}
.col-6 {width: 50%;}	.col-12 {width: 100%;}

Делаем свой grid

Каждая колонка

- выравнивается на лево
- имеет небольшой отступ

```
[class*="col-"] {  
    float: left;  
    padding: 15px;  
}
```

Делаем свой grid

Каждая колонка должна находиться в строке, как в таблице.

Максимальное количество колонок в строке [12](#).

Строка это элемент, который содержит в себе элементы колонок.

```
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-9">...</div> <!-- 75% -->
</div>
```

Делаем свой grid

- Каждая колонка выравнивается на лево (`float: left`)
- Строки содержат в себе колонки
- У строк должен быть `clearfix`.

```
.row:after {  
    content: "";  
    clear: both;  
    display: block;  
}
```

Grid готов, но

- grid полностью резиновый
- колонки имеет одинаковую ширину при любом разрешении
- контент будет сильно сплющиваться на мелких экранах

У нас полностью резиновая разметка.

Нужно добавить немного адаптивности.

Медиа-запросы решат нашу проблему.

Синтаксис медиа-запросов

Задается тип устройства и в круглых скобках условие, при выполнении которого, активируется CSS код.

```
@media screen and (max-width: 600px) {  
    body {  
        background: white;  
    }  
}
```

При уменьшении окна до 600 пикселей и меньше меняется фон.

Типы устройств

all	Все типы. Это значение используется по умолчанию.
print	Принтеры и другие печатающие устройства.
screen	Экран монитора.
speech	Речевые синтезаторы, а также программы для воспроизведения текста вслух.

Несколько условий в одном @media

Можно комбинировать условия с помощью:

- “and” – логическое “и”
- “or” - логическое “или”

```
@media (min-width: 700px) and (orientation: landscape) {}
```

```
@media (min-width: 700px) or (orientation: portrait) {}
```

Несколько условий в одном @media

Чаще всего комбинируются условия ширины viewport

`@media (min-width: 720px) and (max-width: 1024px)`

Если ширина от 720px до 1024px

Условия, которые можно проверить

- min-width
- max-width
- orientation
- resolution
- monochrome
- height

Отзывчивый grid

На мобильных телефонах колонка занимает всю ширину окна.

```
@media screen and (max-width: 768px) {  
  [class*="col-"] {  
    width: 100%;  
  }  
}
```

Отзывчивый grid

Добавим классы для планшетов и устройств небольшого размера.

```
@media screen and (max-width: 1024px) {  
  
.col-m-1 {width: 8.33%;}  
.col-m-2 {width: 16.66%;}  
.col-m-3 {width: 25%;}  
.col-m-4 {width: 33.33%;}  
.col-m-5 {width: 41.66%;}  
.col-m-6 {width: 50%;}  
  
.col-m-7 {width: 58.33%;}  
.col-m-8 {width: 66.66%;}  
.col-m-9 {width: 75%;}  
.col-m-10 {width: 83.33%;}  
.col-m-11 {width: 91.66%;}  
.col-m-12 {width: 100%;}  
}
```

Отзывчивый grid готов

Теперь его можно использовать

```
<div class="row">
    <div class="col-3 col-m-3">...</div>
    <div class="col-6 col-m-9">...</div>
    <div class="col-3 col-m-12">...</div>
</div>
```

Отзывчивые изображения

```
img {  
    width: 100%;  
    height: auto;  
}
```

Изображение растягивается на всю ширину родителя сохраняя пропорции.

Изображение может быть больше изначального размера, что б этого избежать, можно использовать **max-width** вместо **width**.

Отзывчивые изображения background-size

```
div {  
    width: 100%;  
    height: 400px;  
    background-image: url('img_flowers.jpg');  
    background-size: 100% 100%;  
}
```

Изображение растягивается на всю ширину и высоту блока, но если размеры блока и изображения не совпадают, то пропорции будут потеряны.

Отзывчивые изображения `background-size`

```
div {  
    width: 100%;  
    height: 400px;  
    background-image: url('img_flowers.jpg');  
    background-size: cover;  
}
```

Изображение растягивается на всю ширину и высоту блока, пропорции будут сохранены, но часть изображения может быть скрыта, если размеры блока и изображения не совпадают.

Разные изображения для разных устройств

```
/* Для ширины меньше чем 400px: */  
body {  
    background-image: url('small.jpg');  
}  
  
/* Для ширины 400px и более: */  
@media screen and (min-width: 400px) {  
    body {  
        background-image: url('big.jpg');  
    }  
}
```

Flexbox

Flexbox layout

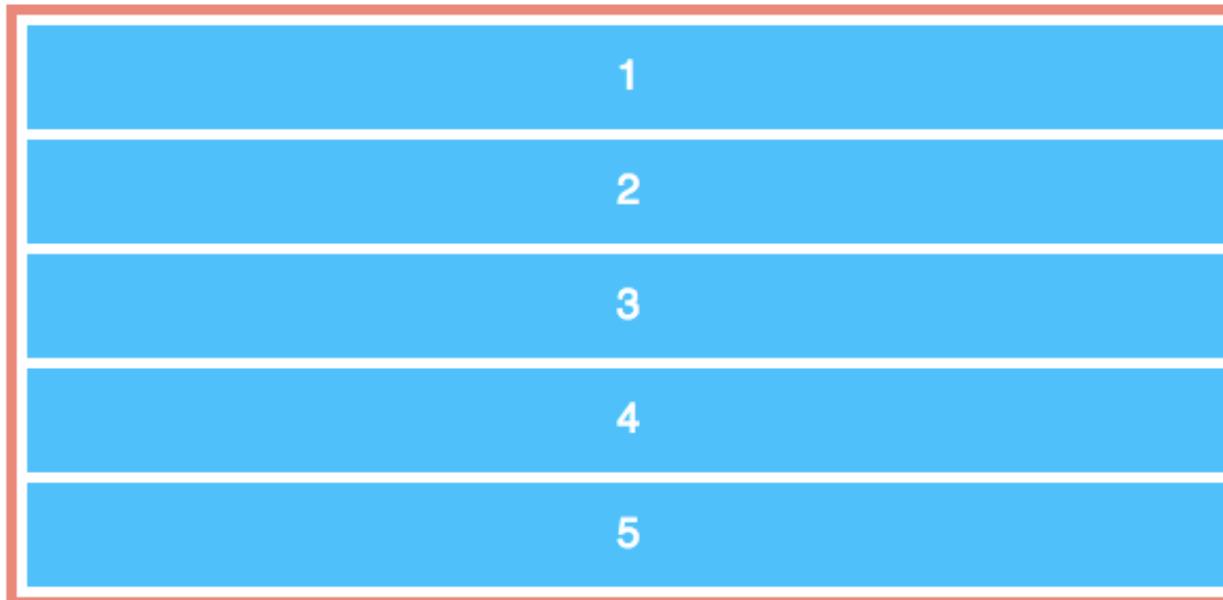
Новый вид разметки.

Создан для создания более предсказуемого поведения элементов на разных видах устройств и разрешениях.

Служит альтернативой блочной разметки, которая использует выравнивания (float).

Код песочницы и результат

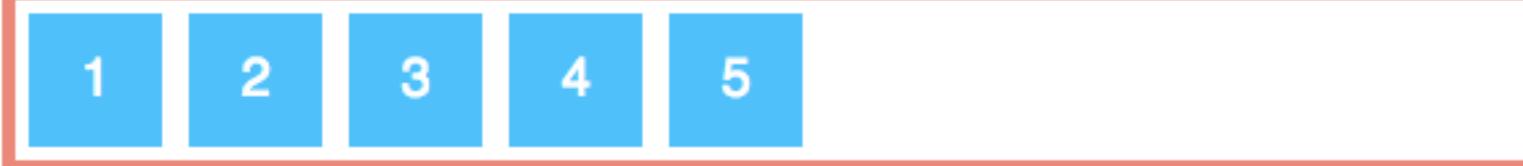
```
<div class="parent">  
  <div class="item">1</div>  
  <div class="item">2</div>  
  <div class="item">3</div>  
  <div class="item">4</div>  
  <div class="item">5</div>  
</div>
```



```
.parent {  
  border: 5px solid salmon;  
}  
.item {  
  height: 50px;  
  margin: 5px;  
  color: white;  
  background: deepskyblue;  
  text-align: center;  
  line-height: 50px;  
  font-size: 20px;  
  font-family: sans-serif;  
}
```

display: flex

```
.parent {  
  display: flex;  
}  
  
.item {  
  width: 50px;  
}
```

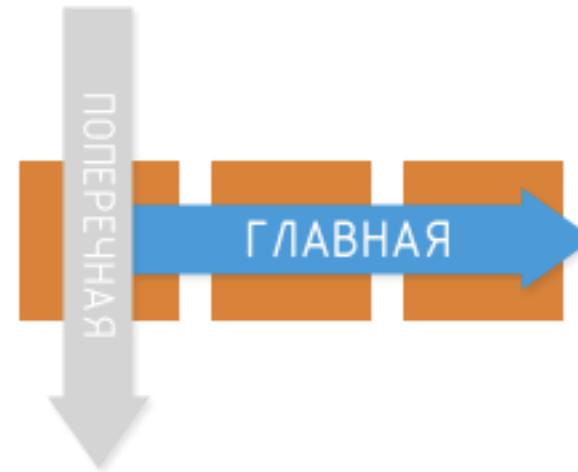


Свойство flex-direction

Задаёт направление основных осей в контейнере и тем самым определяет положение флекс-элементов в контейнере.



flex-direction: `column`



flex-direction: `row`

Значения flex-direction

- **row** - главная ось направлена так же, как и ориентация текста, по умолчанию слева направо.
- **row-reverse** - похоже на значение row, но меняются местами начальная и конечная точки и главная ось направлена справа налево.
- **column** - главная ось располагается вертикально и направлена сверху вниз.
- **column-reverse** - главная ось располагается вертикально, но меняется положение начальной и конечной точек и ось направлена снизу вверх.

```
.example-direction .parent {  
    display: flex;  
    flex-direction: row-reverse;  
}  
  
.example-direction .item {  
    width: 50px;  
}
```



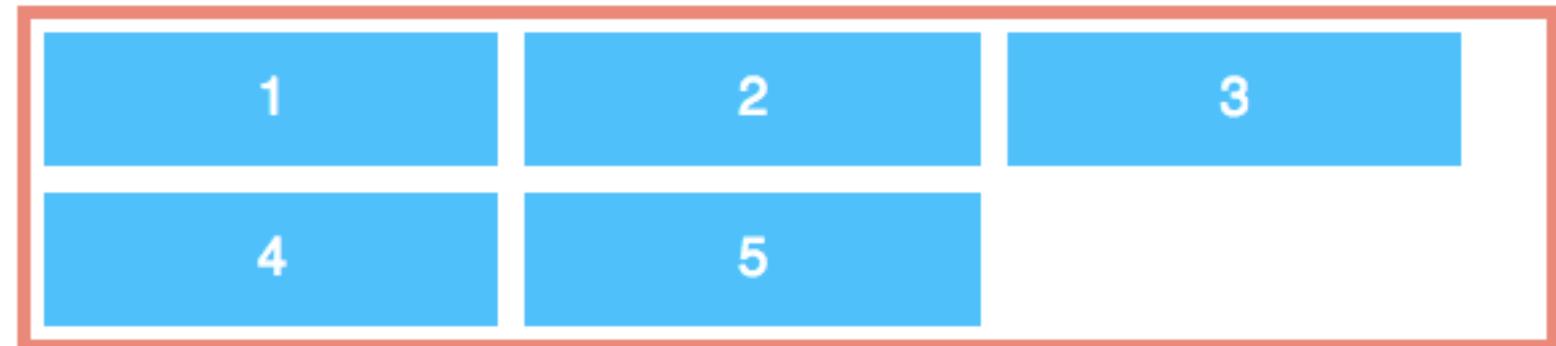
```
.example-direction .parent {  
    display: flex;  
    flex-direction: column;  
}  
  
.example-direction .item {  
    width: 50px;  
}
```



Свойство flex-wrap

Возможные значения: nowrap | wrap | wrap-reverse

```
.parent {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.item {  
  width: 30%;  
}
```



Свойство justify-content

Определяет, как браузер распределяет пространство вокруг флекс-элементов вдоль **главной оси** контейнера.

```
.parent {  
  display: flex;  
  justify-content: center;  
}
```

```
.item {  
  width: 50px;  
}
```



Примеры justify-content



flex-start



flex-end



space-around



space-between

Свойство align-items

Свойство выравнивает флекс-элементы внутри контейнера в перпендикулярном направлении.

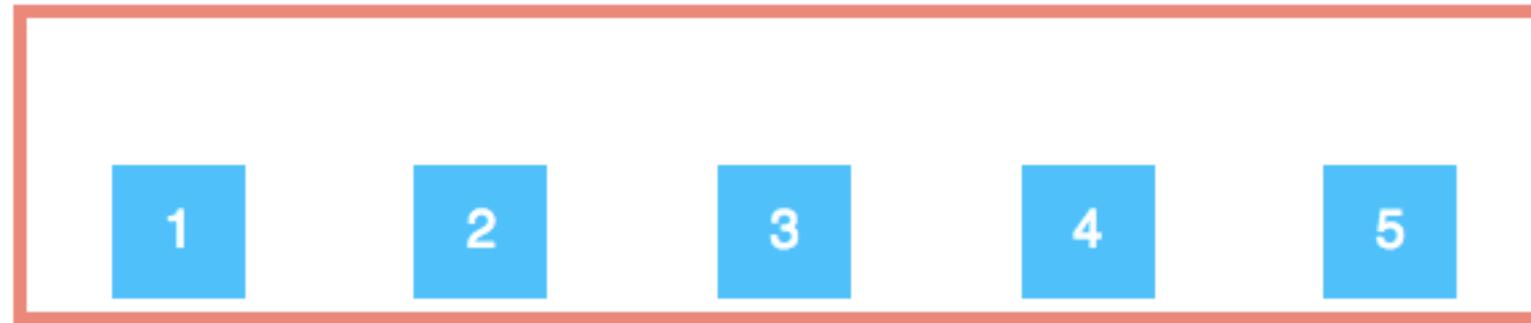
```
.example-align .parent {  
    height: 110px;  
    display: flex;  
    justify-content: space-around;  
    align-items: flex-start;  
}  
  
.example-align .item {  
    width: 50px;  
}
```



Примеры align-items



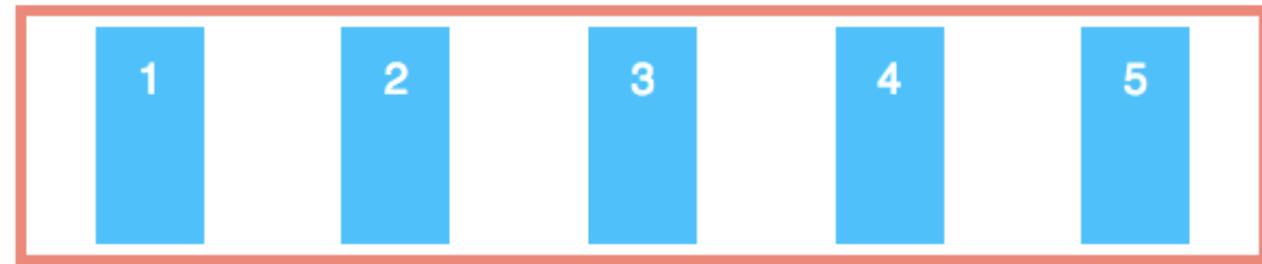
center



flex-end

Пример align-items

```
.parent {  
    height: 110px;  
    display: flex;  
    align-items: stretch;  
    justify-content: space-around;  
}  
  
.item {  
    height: auto;  
    width: 50px;  
}
```



Пример align-items

```
.item:nth-child(3) {  
  font-size: 4rem;  
}
```

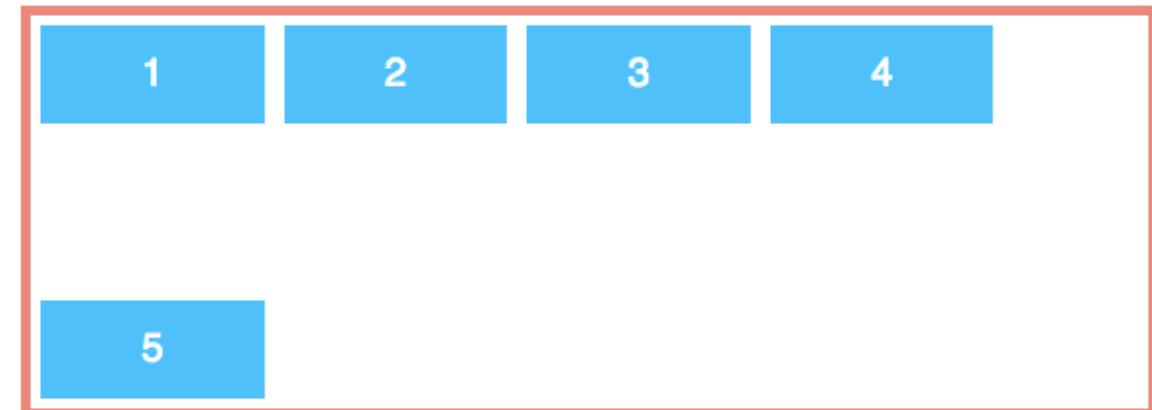


baseline

Свойство align-content

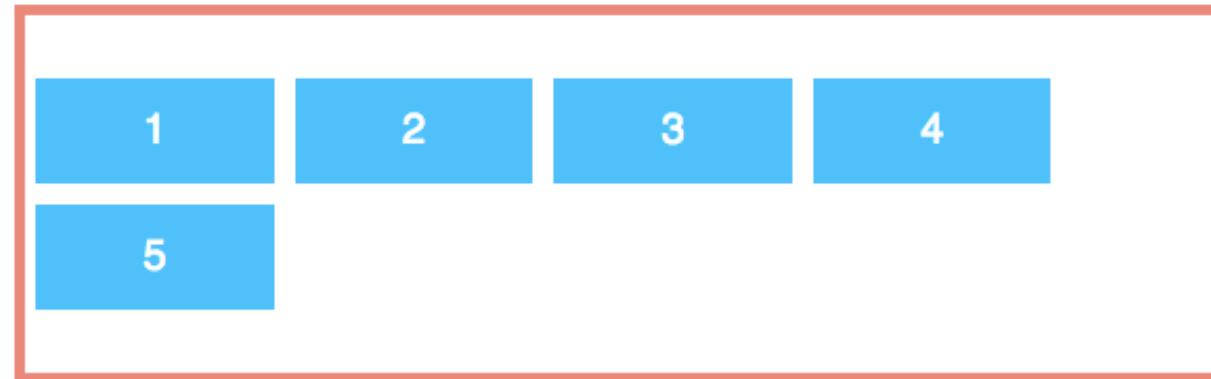
Задаёт тип выравнивания строк внутри флекс-контейнера по **поперечной оси** при наличии свободного пространства. Свойства те же, что и у [justify-content](#). Принцип работы тот же. Разница в оси.

```
.parent {  
    height: 200px;  
    display: flex;  
    flex-wrap: wrap;  
    align-content: space-between;  
}  
  
.item {  
    width: 20%;  
}
```

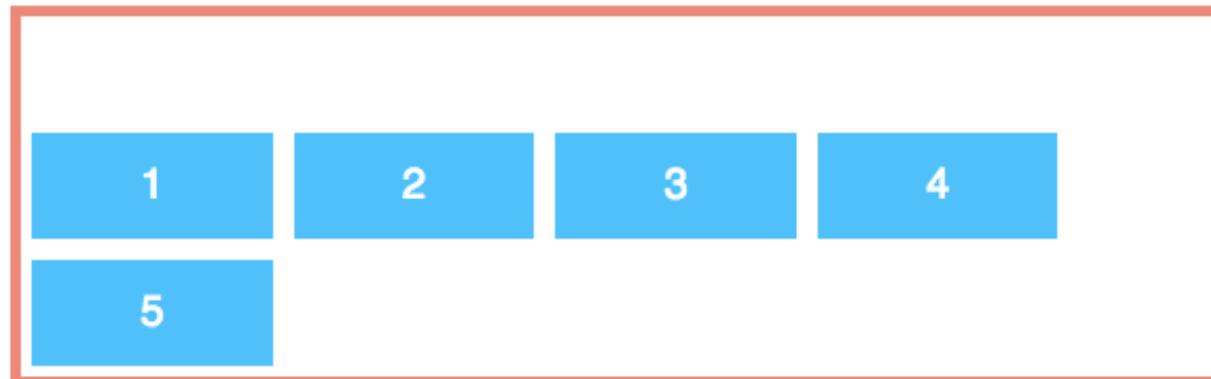




space-around



center

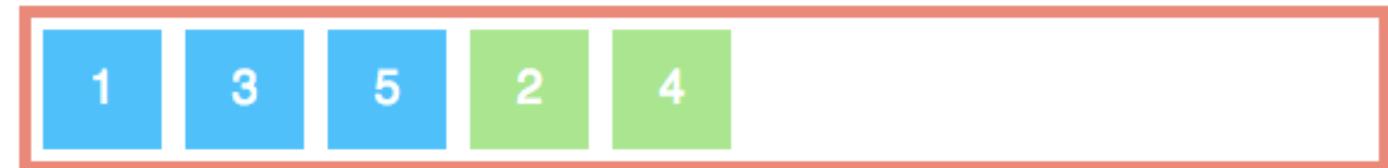


flex-end

Свойство order

Определяет порядок вывода флекс-элементов внутри флекс-контейнера. Элементы располагаются согласно значениям свойства `order` от меньшего к большему.

```
.item:nth-child(2) {  
    order: 1;  
    background: lightgreen;  
}  
  
.item:nth-child(4) {  
    order: 2;  
    background: lightgreen;  
}
```



Свойство flex-grow

Задает пространство, которое может занимать флекс-элемент.
Числа задают пропорции каждого флекс-элемента.

```
.item {  
    width: 10%;  
}  
  
.item:nth-child(2) {  
    flex-grow: 2;  
    background: lightgreen;  
}  
  
.item:nth-child(4) {  
    flex-grow: 3;  
    background: lightcoral;  
}
```



Свойство flex-basis

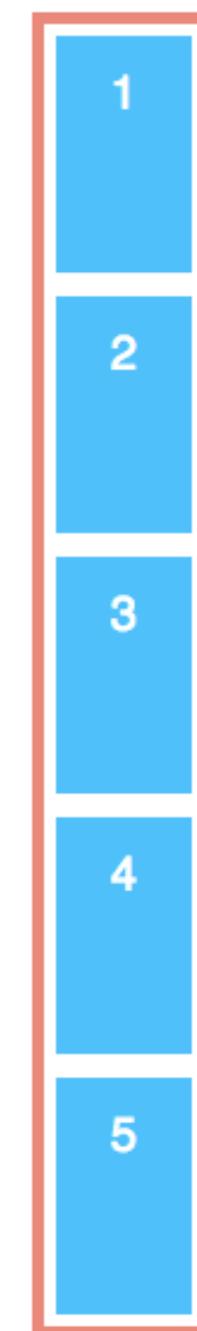
```
.parent {  
  display: flex;  
  flex-direction: row;  
}
```

```
.item {  
  flex-basis: 100px;  
}
```



```
.parent {  
  display: flex;  
  flex-direction: column;  
}
```

```
.item {  
  flex-basis: 100px;  
}
```



Свойство align-self

```
.parent {  
    height: 160px;  
    display: flex;  
    justify-content: space-between;  
}  
  
.item {  
    width: 50px;  
    height: 50px;  
}  
  
.item:nth-child(2) {  
    align-self: flex-end;  
    background: lightgreen;  
}
```

Свойство выравнивает флекс-элементы текущей строки, переписывая значение **align-items**.

