



BEETROOT  
ACADEMY

Каскадные  
таблицы стилей  
**CSS**

**CSS** – Cascading Style Sheets (каскадные таблицы стилей) – это средство, позволяющее задавать различные визуальные свойства HTML-элементам.

<http://www.w3.org/TR/CSS21/cover.html>

Блок определений

CSS - правило

CSS - правило

color:red; font-size:25px;

свойство

значение

### 3 способа

1. **Встраивание (inline)** - Атрибут элемента `style="стиль..."`
2. **Вложение (embedding)** - Информация о стиле в заголовке:  
элемент `<style> ... </ style >`
3. **Связывание (linking)** - Внешние таблицы стилей  
(подключается отдельный файл стилей)

Встроенный стиль определяется атрибутом

**style = "набор деклараций"**

```
<p style="font-size:18px; color:red;">
```

*содержимое параграфа*

```
</p>
```

При отображении этого элемента браузер заменит размер шрифта (который задан по умолчанию) на **18px** и отобразит текст в параграфе красным цветом

```
<p style = "font-size:20px;" >
```

Текст параграфа      текстом размером 20px

```
<i style = "font-size:30px; color:red;">
```

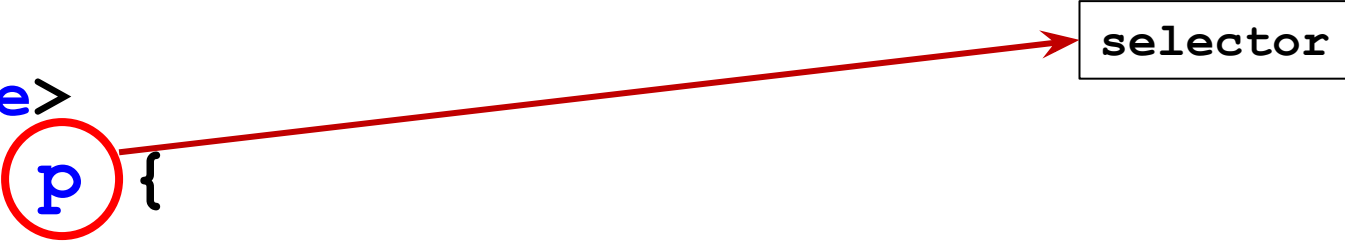
с курсивным текстом размером 30px красным цветом

```
</i>
```

```
</p>
```

Стили задаются в элементе `<style>...</style >` который располагается в элементе `<head> ...</head>`

```
<head>
  <style>
    p {
      color: red;
      font-family: arial;
      font-size: 25px;
    }
  </style>
</head>
```



The diagram illustrates the structure of an embedded style. The CSS rule `p { color: red; font-family: arial; font-size: 25px; }` is shown within a `<style>` block. The selector `p` is circled in red, and a red arrow points from it to a box labeled "selector".

## Подключение файлов CSS (связывание)

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

`<link>` - это служебная ссылка.

Она всегда указывается в элементе `<head> ... </head>`

<code>rel</code>	Атрибут указывает отношения между этим документом и тем документом адрес которого указан в атрибуте <code>href</code>
<code>href</code>	Адрес документа
<code>type</code>	Это <b>Content-Type</b> (то есть тип) подключаемого документа. Например <code>"text/html"</code> , <code>"text/css"</code> , <code>"image/png"</code> , <code>"image/gif"</code> , <code>"video/mpeg"</code> и т.д.
<code>title</code>	Заголовок (например для отображения в ленте новостей)
<code>media</code>	Для каких устройств документ указанный в атрибуте <code>href</code> предназначен. Это могут быть <b>screen</b> - дисплей, <b>print</b> - принтер, <b>all</b> - все устройства и т.д. Например можно записать несколько устройств <code>media="print, tv"</code>

## Преимущества подключение файлов CSS (связывание)

- осуществляется **кеширование** информации в браузере;
- файл HTML становится меньшим по размеру;
- возможность подключения нескольких файлов к HTML-странице;

Свойства CSS могут наследоваться дочерними элементами  
(см. сайт W3C)



Что может выступать в роли значений для свойств CSS ?

свойство : значение

	Пример
Ключевые слова	<code>color:red; text-align:center;</code>
Цифры	<code>z-index:20;</code>
Значения цвета	<code>color:#00cc00;</code>
Функционалы	<code>rgb(0, 255, 0); url(../img/myimg.png);</code>
Размеры	<code>font-size:16px; Line-height: .8em;</code>
Проценты	<code>rgb(20%, 50%, 12%);</code>

Не все значения CSS могут задаваться в %  
В спецификации эти элементы указаны.

## Размеры в CSS

В CSS размеры всегда пишутся с единицами размерности  
Между цифрой и единицей размерности нет пробелов !!!

### Абсолютные

in	Дюймы (1 дюйм = 2,54 см)
cm	Сантиметры
mm	Миллиметры
pt	Пункты (1pt = 1/72 дюйма)
pc	Пики (1pc = 12 pt)

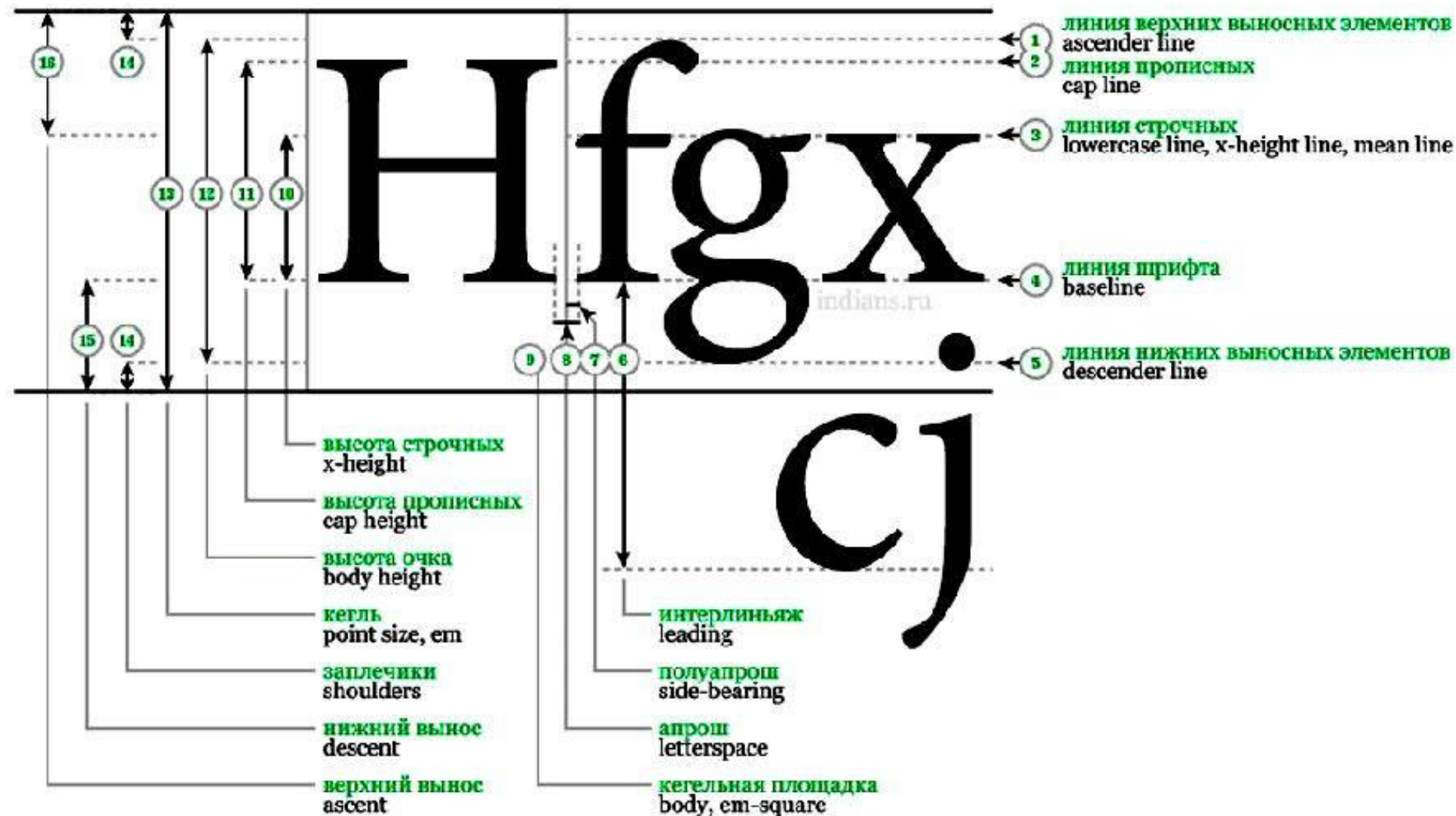
### Относительные

%	Проценты
px	Пиксели
em	Высоты используемого элементом шрифта
ex	Высота символа <b>x</b> элемента

**Абсолютные единицы** – это те размеры, которые связаны с реальным миром

**Относительные** – используются для мониторов, где в соответствии с разрешением монитора меняется размер пикселя.

# измерения шрифта



## Задание цвета в CSS

16-ричный код	Ключевое слово	RGB	
#000000	black	rgb(0, 0, 0)	Черный
#c0c0c0	silver	rgb(192, 192, 192)	светло-серый
#0000ff	blue	rgb(0, 0, 255)	Синий
#008000	green	rgb(0, 128, 0)	Зеленый
#000080	navy	rgb(0, 0, 128)	темно-синий

Допустимые сокращения (если в паре одинаковые символы)

#00cc<sup>red</sup>ee<sup>blue</sup> = #0<sup>red</sup>ce<sup>blue</sup>

#cc<sup>green</sup>00<sup>blue</sup>ff = #c<sup>green</sup>0<sup>blue</sup>f

Отобразить **все параграфы** на странице и отобразить их размером шрифта 20px, красным цветом

```
<head>
  <style>
    p { color:red; font-size:20px; }
    div { background: #e0e0e0;}
  </style>
</head>
```

Отобразить **все блоки** на странице и отобразить их фон цветом #e0e0e0

```
<h1> Заголовок H1 </h1>
<p> Параграф 1 </p>
<div> Элемент DIV </div>
<p> Параграф 3 </p>
```

Отобразить на странице элементы с **id=main-head** и отобразить их размером шрифта 20px, красным цветом

```
<head>
  <style>
    #main-head { color:red; font-size:20px; }
    #content { background: #e0e0e0;}
  </style>
</head>
```

Отобразить на странице элементы с **id=content** и отобразить их фон цветом #e0e0e0

```
<h1 id="main-head">Заголовок Н1</h1>
<p id="content">Параграф 1</p>
```

## Селектор класса

03\_selector\_class.html

Отобразить на странице элементы **h2** с классом **first** и отобразить их шрифтом **arial**

Отобразить на странице элементы с классом **first** и отобразить их красным цветом

```
<head>
  <style>
    .first { color:red; }
    h2.first { font-family: arial;}
    .second {font-size: 20px;}
  </style>
</head>
```

Отобразить на странице элементы с классом **second** и отобразить их размером шрифта 20px

```
<h2 class="first"> Заголовок красного цвета</h2>
<p class="first">text text text text text </p>
<p>text text text text text text text text </p>
<p class="sesond">text text text text text </p>
```

## Особенности классов

1. Элемент может иметь от одного и более классов

```
<p class="first second third"> ... </p>
```

2. Селектор класса (впрочем как и любой другой селектор) может дополняться типом элемента, например

`p.first {...}` -> отобразить только параграфы у которых есть класс `first`

`h2.first {...}` -> отобразить только заголовки 2-го уровня у которых есть класс `first`

3. Селектор класса может состоять из нескольких имен классов, например

`.first.second {...}` -> отобразить элементы у которых есть оба класса `first` и `second`



Отобразить на странице элементы **h1** и **p** и отобразить их красным цветом

```
<head>
  <style>
    h1,p { color:red;}
    #main-head, .content { font-size: 18px;}
  </style>
</head>
```

Отобразить на странице элементы с **id=content** и классом **content** и отобразить их размером шрифта **18px**

```
<h1 id="main-head">Заголовок H1</h1>
<p>Параграф 1</p>
<p class="content">Параграф 1</p>
```

## Селектор атрибута

<code>p[align] { color:red }</code>	У элемента <code>p</code> присутствует атрибут <code>align</code>
<code>[align="первый"] { ... }</code>	У элемента атрибут <code>title</code> равен <code>первый</code>
<code>.mycls[align~="right"] {...}</code>	У элементов с классом <code>mycls</code> в атрибуте <code>align</code> присутствует слово <code>right</code> целиком
<code>.cls[align*="right"] {...}</code>	У элементов с классом <code>cls</code> в атрибуте <code>align</code> присутствует сочетание <code>right</code>
<code>div[lang^="e"] {... }</code>	У элементов <code>div</code> атрибут <code>lang</code> начинается с буквы <code>e</code>
<code>div[lang\$="es"] {... }</code>	У элементов <code>div</code> атрибут <code>lang</code> заканчивается словосочетанием <code>es</code>

<code>:link {...}</code>	<code>a:link{...}</code>	Непосещенная ссылка
<code>:visited {...}</code>	<code>a:visited{...}</code>	Посещенная ссылка
<code>:active{...}</code>	<code>a:active{...}</code>	На ссылке нажали
<code>:hover{...}</code>	<code>a:hover{...}</code>	Указатель мыши над ссылкой

<code>::first-line</code>	<code>p::first-line{...}</code>	
<code>::first-letter</code>	<code>p::first-letter{...}</code>	
<code>::after</code>	<pre>p::after {     content:"test" }</pre>	Формирует область перед элементом
<code>::before</code>	<pre>P::before{     content:"Test" }</pre>	Формирует область после элементом

# Иерархия html страницы

Сестринские (sibling) элементы  
(элементы на одном уровне иерархии)

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
  <div id="f1">
```

```
    <div id="f2">
```

```
      <h2> Заголовок H2 </h2>
```

```
      <p> Текст параграфа </p>
```

```
      <p> Текст параграфа </p>
```

```
    </div>
```

```
  </div>
```

```
</body>
```

```
</html>
```

Корневой (root) элемент

Дочерние (child) элементы

## Контекстный селектор

Селекторы перечисляются через пробел по иерархии элементов

```
<body>
```

```
  <div id="f1">
```

```
    <div id="f2">
```

```
      <h2> Заголовок H2 </h2>
```

```
      <p> Текст параграфа </p>
```

```
      <p class="f3"> Текст параграфа </p>
```

```
    </div>
```

```
  </div>
```

```
</body>
```

#f1 #f2 h2 { ... }

#f1 #f2 p { ... }

#f1 #f2 .f3 { ... }

ИЛИ

#f1 #f2 p.f3 { ... }

<code>*{color:black}</code>	Все элементы на странице.
<code>div &gt; p { color:red }</code>	Прямой дочерний элемент (прямой наследник)
<code>h1 + p { color:red }</code>	Ближайший сестринский (sibling) элемент, следующий за <b>h1</b>
<code>div ~ p { color:red }</code>	Ближайшие сестринские (sibling) элементы, следующие за элементом <b>div</b>

elem1 elem2

Вернет все элементы с именем тега elem2 которые являются потомками элементов с именем elem1

body

div class="d1"

span

a

span

span

.d1 span

```
<div class=d1>  
  
  <span></span>  
  <a>  
    <span></span>  
  </a>  
  <span></span>  
  
</div>
```



```
parent > child
```

Вернет все элементы с именем тега `child` которые являются прямыми потомками элементов с именем тега `parent`

`body`

`div class="d1"`

`span`

`a`

`span`

`span`

```
.d1 > span
```

```
<div class=d1>  
  
  <span></span>  
  <a>  
    <span></span>  
  </a>  
  <span></span>  
  
</div>
```

element + next

Вернет все элементы с именем тега **next** которым непосредственно предшествует элемент **element** на том же уровне вложенности

body

a + span

div class="d1"

a

span

span

span

```
<div class=d1>
```

```
  <a></a>
```

```
  <span></span>
```

```
  <span></span>
```

```
  <span></span>
```

```
</div>
```

element ~ sibling

Вернет все элементы с именем тега sibling которым предшествует элемент element на том же уровне вложенности

body

a ~ span

div class="d1"

a

span

span

span

```
<div class=d1>
```

```
  <a></a>
```

```
  <span></span>
```

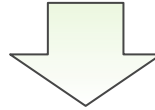
```
  <span></span>
```

```
  <span></span>
```

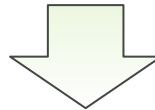
```
</div>
```

## Приоритеты стилей

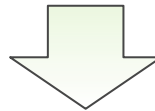
Пользовательские таблицы с значением !important



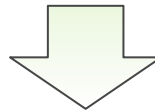
Стили, заданные автором сайта с значением !important



Стили, заданные автором сайта



Пользовательские таблицы  
(то есть те стили которые задал сам пользователь)



Стили браузера по умолчанию

<http://www.w3.org/TR/2011/REC-CSS2-20110607/sample.html>

При конфликтах идет подсчет количества **id**, **классов**, **элементов**, **псевдо-классов**, **псевдо-элементов**, **селекторов** **атрибутов** на пути к элементу в контекстном селекторе

наивысший

приоритет

наименьший

style=""  
!important

#id

.class  
:pseudo-class  
[] attributes

тег  
:pseudo-element

## Примеры

p.myClass { ... }

1тег + 1 класс

.simple p.myClass { ... }

1тег + 2 класса

#head h1 { ... }

1 id + 1тег

#head .simple p.myClass { ... }

1 id + 2 класса + 1тег

## Примеры каскадирования стилей

```
<div id="d1" class="outer">  
  <p id="p1" class="inner">text</p>  
</div>
```

```
p { color: red; }  
.inner { color: green; }
```

Оба селектора отбирают один и тот же элемент (параграф) но назначают ему разный цвет.

То есть получается неоднозначность (**ambiguity** [ æmbɪ'gjʊ:ti ])

Браузер должен принять решение, какой цвет применить к элементу

Алгоритм такой:

- первое правило содержит **селектор типа**
- второе правило содержит **селектор класса**
- **селектор класса** имеет более высокий приоритет над **селектором типа**, поэтому "выигрывает" второе правило, и к элементу параграф будет применен **зеленый** цвет

```
<div id="d1" class="outer">
  <p id="p1" class="inner">text</p>
</div>
```

```
#d1 p { color: red; }
div .inner { color: green; }
```

#### Алгоритм:

- первое правило содержит **селектор id + селектор типа**
- второе правило содержит **селектор типа + селектор класса**
- по селекторам типа приоритеты правил равны  
Но у первого правила приоритет по селектору id, поэтому "выигрывает" это правило, и к элементу параграф будет применен **красный** цвет

```
<div id="d1" class="outer">
  <p id="p1" class="inner">text</p>
</div>
```

```
div p.inner { color: red; }
div p { color: green; }
```

Алгоритм:

- первое правило содержит 2 селектора типа + селектор класса
- второе правило содержит 2 селектора типа
- по селекторам типа приоритеты правил равны  
Но у второго правила есть селектор класса, поэтому  
"выигрывает" это правило, и к элементу параграф будет  
применен **зеленый** цвет



```
<div id="d1" class="outer">
  <p id="p1" class="inner">text</p>
</div>
```

```
div p.inner { color: red; }
div.outer p { color: green; }
```

Алгоритм:

- первое правило содержит 2 селектора типа + селектор класса
- второе правило содержит 2 селектора типа + селектор класса
- по селекторам приоритеты правил равны, поэтому "выигрывает" правило которое расположено по коду ниже, то есть к элементу параграф будет применен **зеленый** цвет

<code>:focus</code>	<code>input:focus{...}</code>	Элемент получил фокус
<code>:enabled</code>	<code>.el:enabled {...}</code>	Элемент формы включен
<code>:disabled</code>	<code>#el:disabled {...}</code>	Элемент формы отключен
<code>:checked</code>	<code>input:checked {...}</code>	Элемент типа <code>radio</code> или <code>checkbox</code> включены

**Focus** получать могут только элементы форм и ссылки.

**Элемент получил focus** - означает, что элемент выбран с помощью мыши (или клавиатуры), курсор находится в пределах этого элемента, и если это элемент формы - то мы можем вводить в него текст.

**Элемент включен**, если его можно активировать (например, выбрать, нажать на него или ввести текст) или поставить фокус. У элемента также есть отключенное состояние, когда его нельзя активировать или сфокусировать.

<code>:first-child</code>	<code>p:first-child{...}</code>	Первые потомки всех элементов
<code>:last-child</code>	<code>p:first-child{...}</code>	Последние потомки всех элементов
<code>:only-child</code>	<code>p:only-child {...}</code>	элемент, являющийся единственным потомком родителя

<code>:target</code>	Применяется при наличии якоря на ссылке, то есть если будет переход по ссылке вида <code>href="test.html#d1"</code> то стили применяются к элементу с <code>id="d1"</code> после перехода на него
<code>:empty</code>	Выбирает элементы у которых нет дочерних, включая текстовые узлы
<code>:not(селектор)</code>	выбрать элементы согласно селектору за исключением указанного в <code>not()</code> Например выбрать все элементы <code>div</code> на странице за исключением элемента с <code>id = "d1"</code> <code>div:not(#d1) {...}</code>

## Структурные псевдоклассы CSS3

**:nth-child(n)**

Выбрать элемент, номер которого указан в скобках

Пояснения на след. слайдах

```
li:nth-child(2n) { ... }
```

```
li:nth-child(2n + 1) { ... }
```

**:nth-last-child(n)**

Выбрать элемент, номер которого указан в скобках, отсчет номера вести снизу

```
li:nth-last-child(2) { ... }
```

**X:nth-of-type(n)**

В списке потомков элемента выбрать элемент указанный в селекторе **X** номер которого указан в скобках  
Отсчет сверху

```
ul:nth-of-type(3) { ... }
```

выбираем 3-й дочерний элемент **ul**

**:nth-last-of-type(n)**

( список отсчитывая снизу)

```
ul:nth-last-of-type(3) { ... }
```

Как работает **`nth-child (expression)`**

**`expression`** – может задаваться например как **`3n+3`** или **`4n+2`**  
где **`n`** – это номер текущего элемента (начинается с 0)

`nth-child(5)` – будет выбран 5-й элемент из списка

**`3n + 3`**

**`n`** принимает значения  
от 0 и далее

$(3 \times 0) + 3 = 3$  – 3-ий элемент

$(3 \times 1) + 3 = 6$  – 6-ой элемент

$(3 \times 2) + 3 = 9$  – 9-ый элемент и т.д.

**`2n + 3`**

$(2 \times 0) + 3 = 3$  – 3-ий элемент

$(2 \times 1) + 3 = 5$  – 5-ый элемент

$(2 \times 2) + 3 = 7$  – 7-ой элемент и т.д.

**`4n - 1`**

$(4 \times 0) - 1 = -1$  – нет соответствия

$(4 \times 1) - 1 = 3$  – 3-ий элемент

$(4 \times 2) - 1 = 7$  – 7-ой элемент

$(4 \times 3) - 1 = 11$  – 11-ый элемент

и т.д.

$n$	$2n+1$	$4n+1$	$4n+4$	$4n$	$5n-2$	$-n+3$
0	1	1	4	-	-	3
1	3	5	8	4	3	2
2	5	9	12	8	8	1
3	7	13	16	12	13	-
4	9	17	20	16	18	-
5	11	21	24	20	23	-

<code>:nth-of-type (expr)</code>	Первые потомки всех элементов отобранных согласно (expr) одного типа
<code>:nth-last-of-type (expr)</code>	Последние потомки всех элементов отобранных согласно (expr) одного типа
<code>:only-of-type</code>	элемент, е которого нет siblings элементов данного типа



```
<section>
  <p>Первый параграф</p>
  <p>Второй параграф </p> <!-- Хотим этот элемент -->
</section>
```

```
p:nth-child(2) { color: red; }
```

Элемент отберется, если это:

1. Параграф
2. Второй потомок родителя

Работает правильно

```
p:nth-of-type(2){color: red; }
```

Элемент отберется, если это:

1. Второй дочерний параграф  
родителя

Работает правильно

```
<section>
  <h1>Заголовок H1</h1>
  <p>Первый параграф</p>
  <p>Второй параграф </p> <!-- Хотим этот элемент -->
</section>
```

```
p:nth-child(2) { color: red; }
```

Элемент отберется, если это:

1. Параграф
2. Второй потомок родителя

**Работает неправильно**

```
p:nth-of-type(2) {color: red; }
```

Элемент отберется, если это:

1. Второй дочерний параграф  
родителя

**Работает правильно**

```
<ul>
  <li>I'm all alone!</li>
</ul>
```

```
<ul>
  <p>I am a paragraph 1</p>
  <p>I am a paragraph 2</p>
  <li>We are together.</li>
  <li>We are together.</li>
  <li>We are together.</li>
</ul>
```

```
li:only-of-type {
  color: red;
}
```

Элемент отберется, если это:

1. li
2. Больше нет **sibling** элементов данного типа

**ВОПРОС** – будут ли отображены параграфы ?

```
p:only-of-type {
  color: orange;
}
```

## Наследование свойств элементами

наследуются всеми элементами	<code>visibility, cursor</code>
наследуются элементами типа <code>inline</code>	<code>letter-spacing, word-spacing, white-space, line-height, color, font, font-family, font-size, font-style, font-variant, font-weight, text-decoration, direction, text-transform</code>
наследуются элементами типа <code>block</code>	<code>text-indent      text-align</code>
наследуются элементами типа <code>list-item</code>	<code>list-style, list-style-type, list-style-position, and list-style-image</code>
не наследуются	<code>display, margin, border, padding, background, height, min-height, max-height, width, min-width, max-width, overflow, position, left, right, top, bottom, z-index, float, clear, table-layout, vertical-align, page-break-after, page-break-before, unicode-bidi</code>

