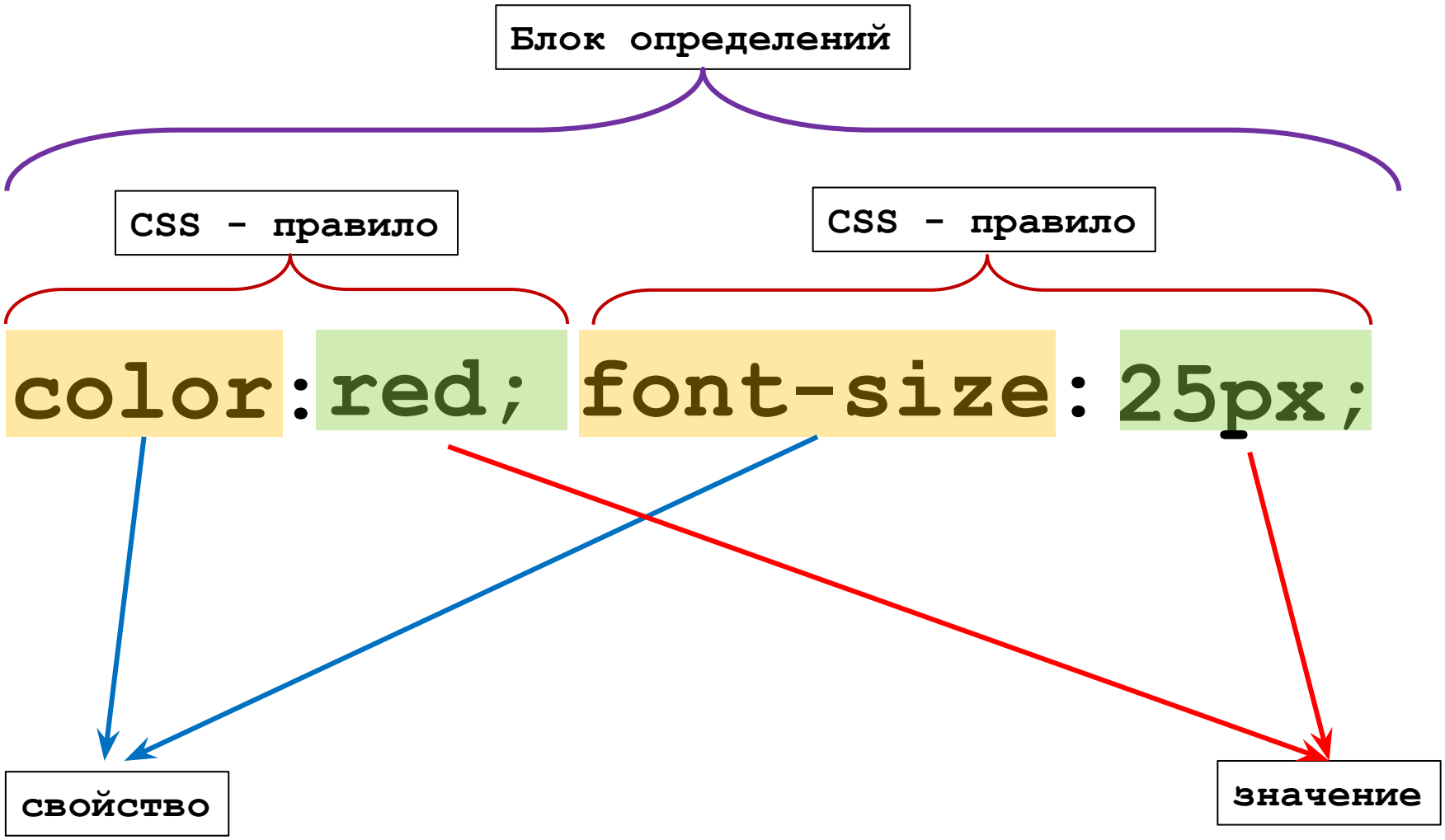
BEETROOT ACADEMY

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

Способы подключения CSS стилей к HTML файлу

3 способа

1. **Встраивание(inline)-** Атрибут элемента **style="стиль…"**

2. **Вложение(embeding) -** Информация о стиле в заголовке:
   элемент **<style> … </ style >**

3. **Связывание(linking)-** Внешние таблицы стилей
   (подключается отдельный файл стилей)

1. Встроенный стиль

index.html

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

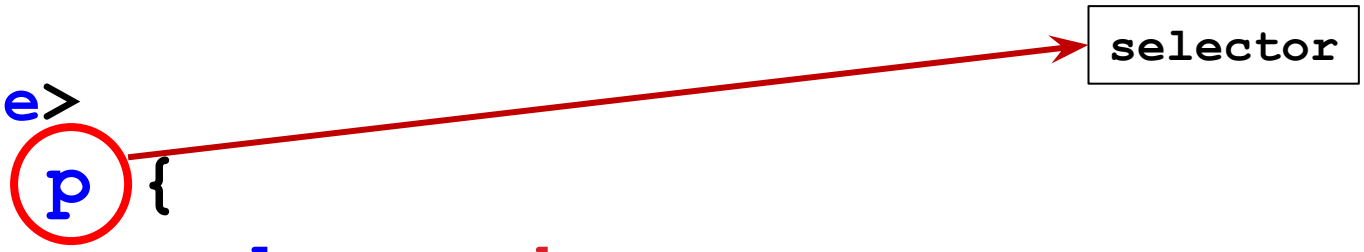
```
<p style = "font-size:20px;" >
        Текст параграфа  текстом размером 20px
      <i style = "font-size:30px;  color:red;">
            с курсивным текстом размером 30px красным цветом
      </i>
</p>
```

2. Вложенный стиль

index.html

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>
```



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>

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

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

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 !!!

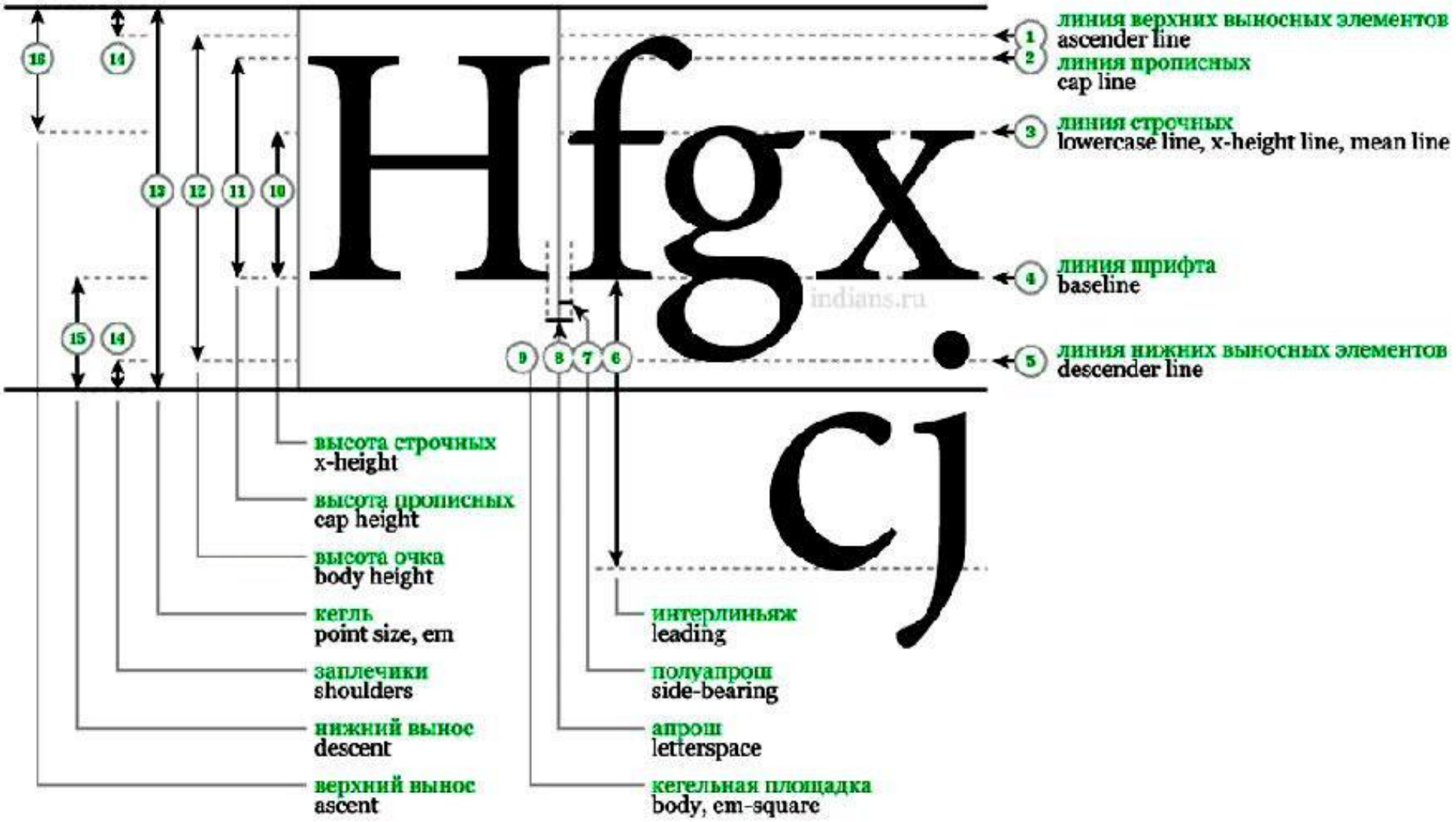
Абсолютные	
in	Дюймы (1 дюйм = 2,54 см)
см	Сантиметры
мм	Миллиметры
pt	Пункты (1pt = 1/72 дюйма)
pc	Пики (1pc = 12 pt)

Относительные	
%	Проценты
px	Пикселы
em	Высоты используемого элементом шрифта
ex	Высота символа x 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)

#00ccee = #0ce

#cc00ff = #c0f

Виды селекторов - селекторы типа

01_selector_type.html

Отобрать 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>
```

Отобрать 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>
```

Селектор id

02_selector_id.html

Отобрать 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>
```

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

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

Селектор класса

03_selector_class.html

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

Отобрать 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>
```

Отобрать 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>
```

Особености 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 {...}   -> отобрать только параграфы у которых есть
                   класс first

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

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

```
.first.second {...}  -> отобрать элементы у которых есть оба
                        классы first и second
```

Группировка селекторов

04_selector_group.html

Отобрать 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>
```

Отобрать 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>
```

05_seletor_attributes.html

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

Псевдоклассы

selectors/index.html

:link {…}	a:link{…}	Непосещенная ссылка
:visited {…}	a:visited{…}	Посещенная ссылка
:active{…}	a:active{…}	На ссылку нажали
:hover{…}	a:hover{…}	Указатель мыши над ссылкой

Псевдоэлементы

07_pseudo_elements.html

::first-line	p::first-line{…}	
::first-letter	p::first-letter{…}	

::after	p::after {
	content:”test”
	}
		Формирует область перед элементом

::before	P::before{
	content:”Test”
	}
		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  { … }
```

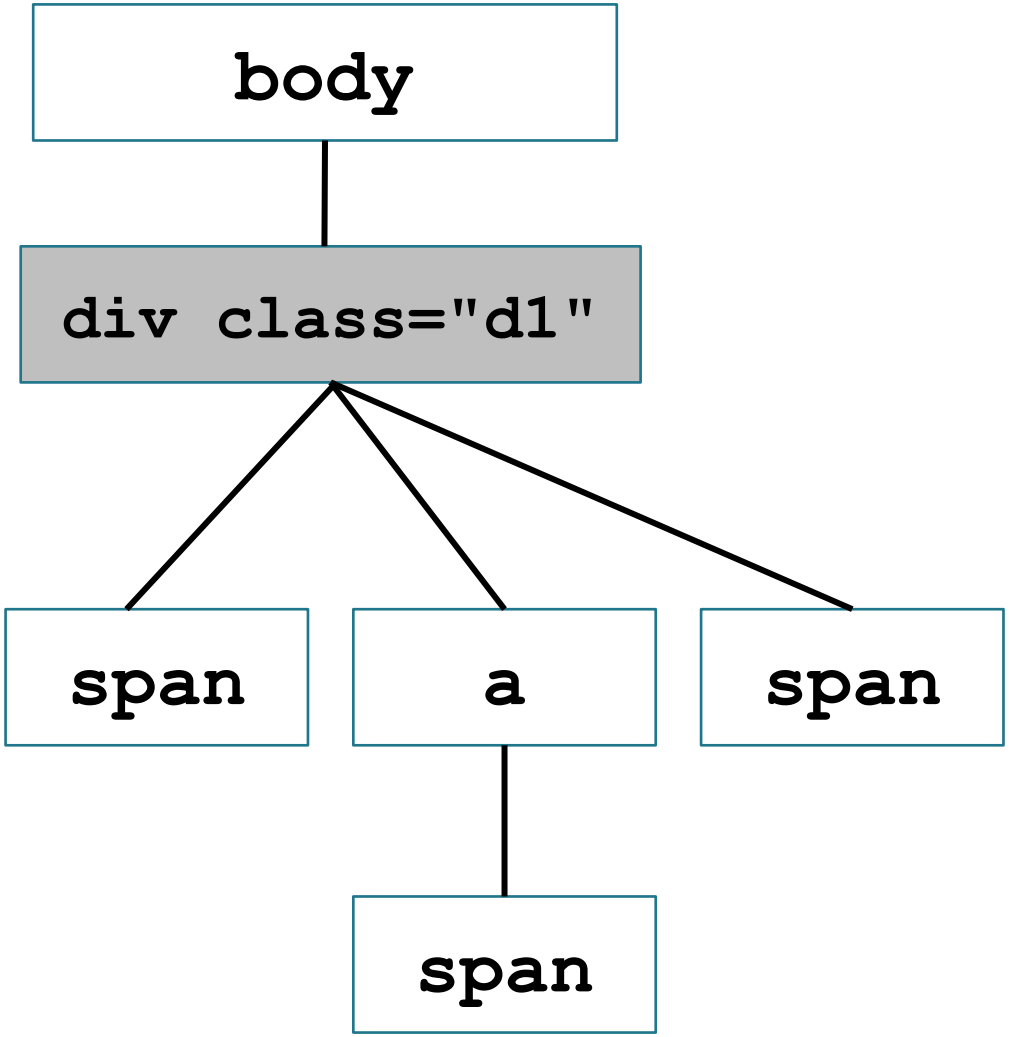
Семейные селекторы

08_selector_context.html

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

elem1 elem2

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

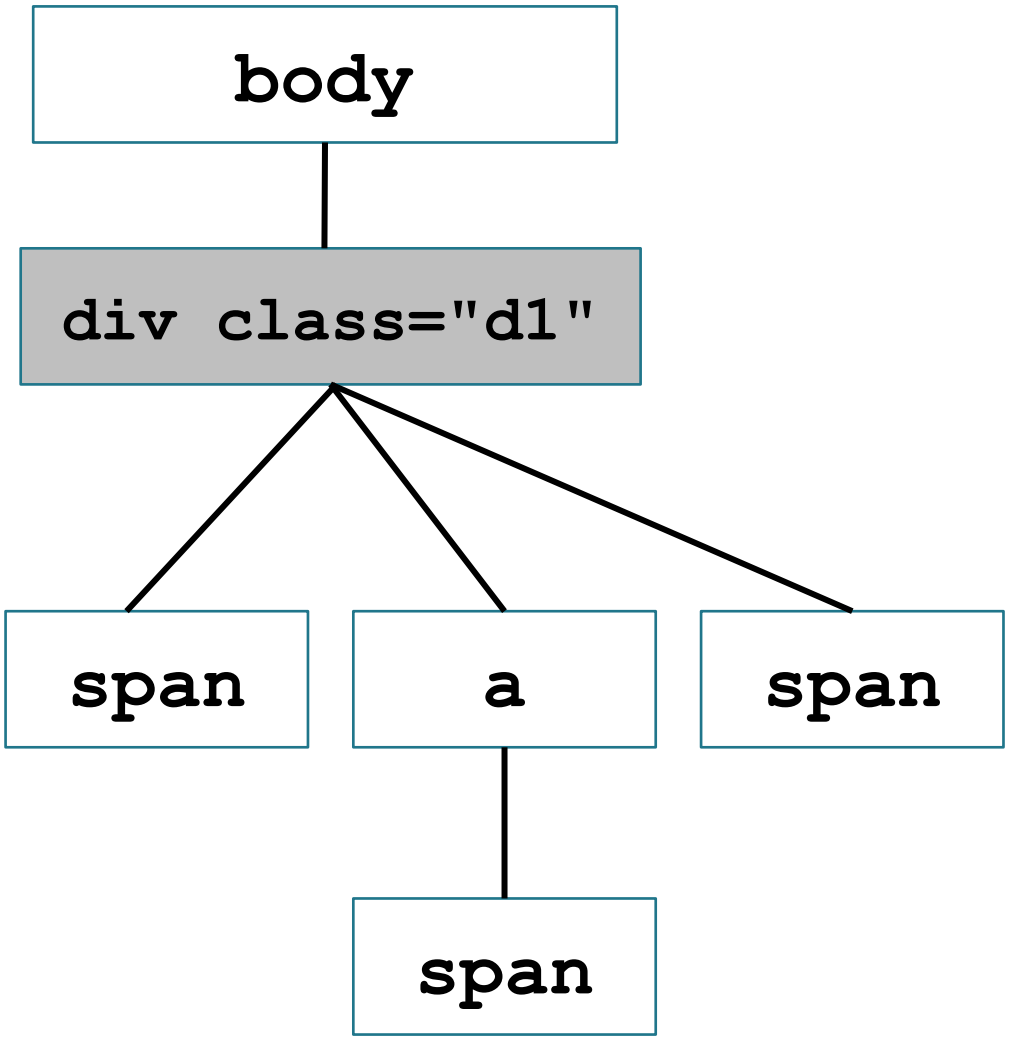


.d1 span

```
<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

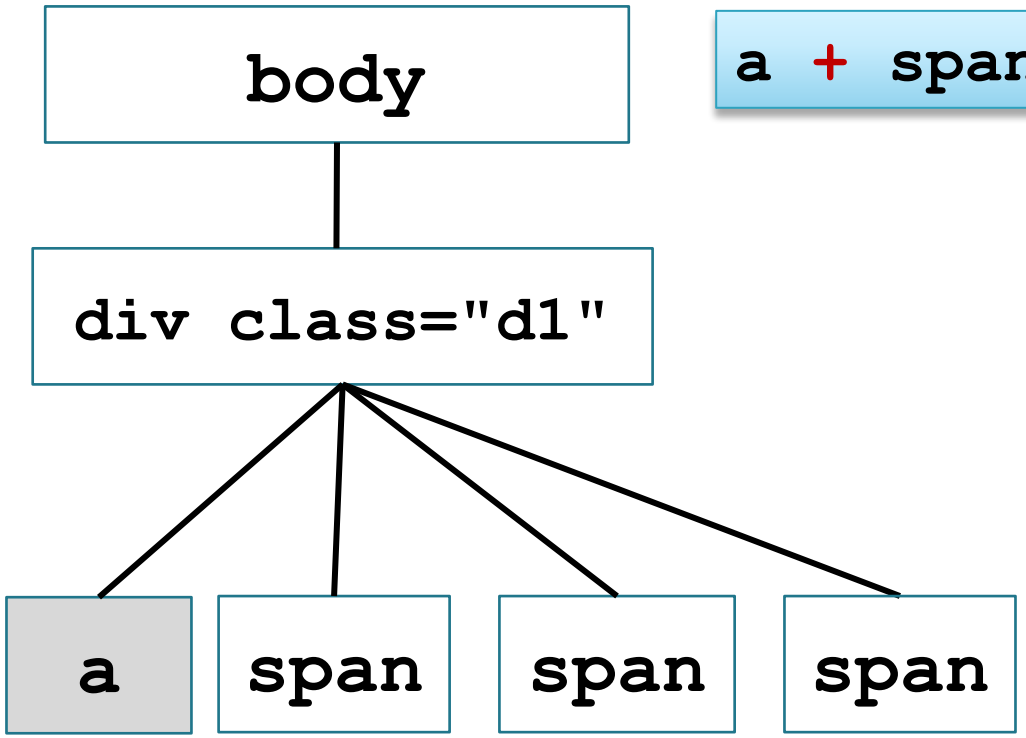


.d1 > span

```
<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

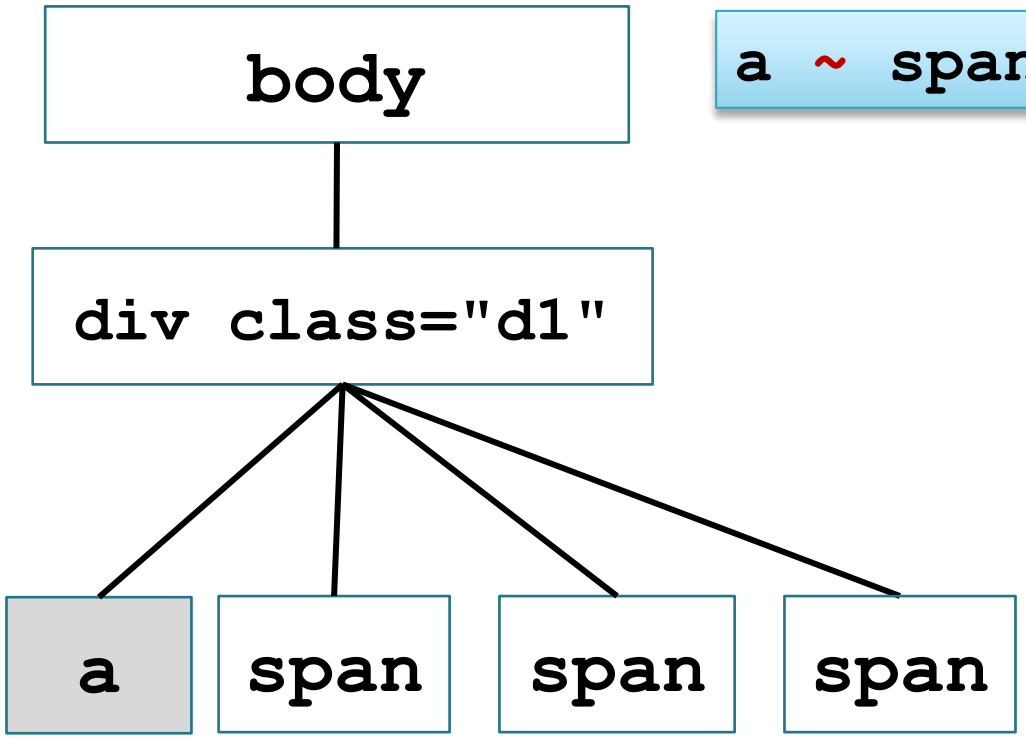


a + span

```
<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

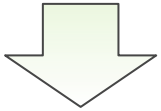


a ~ span

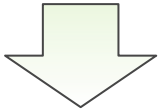
```
<div class=d1>

    <a></a>
    <span></span>
    <span></span>
    <span></span>

</div>
```

Приоритеты стилей

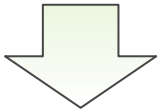
Пользовательские таблицы с значением !important



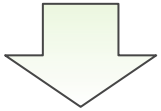
Стили, заданные автором сайта с значением !important



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

Каскадирование при конфликтах

09_cascading.html

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

style="" !important	#id	.class :pseudo-class [] attrubutes	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 [æmbɪˈgjuːɪtɪ])
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

Псевдоклассы (прордолжение)

selectors/index.html

:focus	input:focus{…}	Элемент получил фокус
:enabled	.el:enabled {...}	Элемент формы включен
:disabled	#el:disabled {...}	Элемент формы отключен
:checked	input:checked {...}	Элемент типа radio или checkbox 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.

Псевдоклассы (прордолжение)

selectors/index.html

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

Псевдоклассы (прордолжение)

selectors/index.html

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

Псевдоклассы (прордолжение)

selectors/index.html

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 x 0) + 3 = 3-ий элемент
(3 x 1) + 3 = 6-ой элемент
(3 x 2) + 3 = 9-ый элемент и т.д.

2n + 3

(2 x 0) + 3 = 3-ий элемент
(2 x 1) + 3 = 5-ый элемент
(2 x 2) + 3 = 7-ой элемент и т.д.

4n - 1

(4 x 0) - 1 = -1 = нет соответствия
(4 x 1) - 1 = 3-ий элемент
(4 x 2) - 1 = 7-ой элемент
(4 x 3) - 1 = 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	-

selectors/type.html

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

selectors/type.html

```
<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 отобраны параграфы ?

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

Наследование свойств элементами

наследуются всеми элементами	visibility, cursor
наследуются элементами типа inline	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
наследуются элементами типа block	text-indent text-align
наследуются элементами типа list-item	list-style, list-style-type, list-style-position, and list-style-image
не наследуются	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,