# Язык CSS

ВВЕДЕНИЕ

```
1 <body bgcolor="#f1f1f1">
2     <h1 align="center"><font size="16px" color="red"
      ↪  face="Tahoma">Основной заголовок</font></h1>
3     <p><font size="8px" color="gray" face="Arial">Текст параграфа
      ↪  1</font></p>
4     <h2><font size="10px" face="Tahoma">Подзаголовок 1</font></h2>
5     <p><font size="8px" color="gray" face="Arial">Текст параграфа
      ↪  2</font></p>
6     <h2><font size="10px" face="Tahoma">Подзаголовок 2</font></h2>
7     <p><font size="8px" color="gray" face="Arial">Текст параграфа
      ↪  3</font></p>
8 </body>
```

## Причины появления CSS:

- Избыточность HTML. HTML стал нести слишком много информации, кроме как про разметку документа.
- Желание большего числа возможностей для оформления. Это еще больше раздувало HTML разметку.
- Упрощение оформление страницы.
- Желание избежать копирования одних и тех же тегов и атрибутов для создания одинакового оформления.
- Отделение контента страницы от его представления.

С появлением CSS упомянутый выше код можно переписать в следующем виде

```css
body {
    background: #f1f1f1;
}
h1 {
    font-size: 16px;
    color: red;
    font-family: Tahoma;
    text-align: center;
}
h2 {
    font-size: 10px;
    font-family: Tahoma;
}
p {
    font-size: 8px;
    color: gray;
    font-family: Arial;
}
```

```html
<body>
    <h1>Основной заголовок</h1>
    <p>Текст параграфа 1</p>
    <h2>Подзаголовок 1</h2>
    <p>Текст параграфа 2</p>
    <h2>Подзаголовок 2</h2>
    <p>Текст параграфа 3</p>
</body>
```

CSS — язык стилей, определяющий отображение HTML (и других) документов.

CSS работает со шрифтами, цветом, полями, строками, высотой, шириной, фоновыми изображениями, позиционированием элементов и многими другими вещами.

- CSS уровень 1 (1996, 1999) – параметры шрифтов, цвета, ...
- CSS уровень 2 (12 мая 1998) – блочная вёрстка, селекторы, …
- CSS уровень 2.1 (07 июня 2011)
- CSS уровней 3 все еще находится в стадии разработки – трансформации, анимация, ...

# Подключение CSS

Файл со стилями должен иметь расширение **.css**. Чтобы подключить такой файл к HTML странице, в теге <head> следует написать такую конструкцию:

```
<link rel="stylesheet" href="имяФайла.css">
```

# Синтаксис CSS

Каждое правило CSS состоит из двух частей: селектора и блока объявлений:

```
селектор {
   правило
}
```

# Синтаксис CSS

Каждое правило CSS состоит из двух частей: селектора и блока объявлений:

```
селектор {
   правило
}
```

```
селектор, селектор, селектор {
      /* блок объявления стилей */
      свойство: значение;
      свойство: значение;
      свойство: значение;
      свойство: значение;
      свойство: значение;
}
```

# Основные ошибки при написании CSS

```
1  body {
2      background: #f1f1f1;
3  }
4  h1 {
5      font-size: 34px;
6      color: red
7      text-align: center;
8  }
9  h2 {
10     font-size: 24px;
11 }
12 h1,
13 h2 {
14     font-family: Tahoma;
15 }
16 p {
17     font-size: 18px;
18     color: gray;
19     font-family: Arial;
20 }
```

# Селекторы тегов, классов и идентификаторов

Селекторы тегов записываются как имена тегов. Поэтому селектор, который выделяет все заголовки первого уровня выглядит просто как «h1».

# Селекторы классов

Этот элемент принадлежит двум классам: heading и red. Чтобы стилизовать этот элемент, в CSS следует записать:

```
<div class="heading red">Красный заголовок</div>
```

```
div.heading {
    font-size: 50px;
}
div.red {
    color: red;
}
```

# Селекторы классов

Этот элемент принадлежит двум классам: heading и red. Чтобы стилизовать этот элемент, в CSS следует записать:

```
<div class="heading red">Красный заголовок</div>
```

```
div.heading {
    font-size: 50px;
}
div.red {
    color: red;
}
```

```
<div class="heading red">Красный заголовок</div>
<div class="red">Красный заголовок</div>
```

```
div.heading {
    font-size: 50px;
}
div.red {
    color: red;
}
```

# Селекторы идентификаторов

Селекторы идентификаторов задаются с помощью символа решетки, после которого идет значение атрибута id.

Так, например, следующий код не будет работать:

```
<div id="heading red">Красный заголовок</div>
```

```
div#heading {
    font-size: 50px;
}
div#red {
    color: red;
}
```

# Селекторы идентификаторов

Селекторы идентификаторов задаются с помощью символа решетки, после которого идет значение атрибута id.
Рабочий пример:

```
<div id="red">Красный заголовок</div>
```

```
div#heading {
    font-size: 50px;
}
div#red {
    color: red;
}
```

# Универсальный селектор

```
<h2>Красный заголовок</h2>
<div>Красный заголовок</div>
<p>Красный заголовок</p>
```

```
* {
    color: red;
}
```

- Комбинирование селектора тега и селектора класса (или нескольких селекторов класса):

```
div.class { color: red; }
p.class.class2 { color: red; }
```

- Комбинирование селектора тега и селектора идентификатора:

```
div#id { color: red; }
```

- Комбинирование селектора тега, селектора идентификатора и селектора класса:

```
a#id.class { color: red; }
```

- Можно да

```
div#id#id2 { color: red; }
```

селектора идентификатора, но это не имеет смысла:

- Также можно комбинировать универсальный селектор с любым селектором, но такая зап

```
*.class { color: red; }
.class { color: red; }
```

чной: эти два примера дадут одинаковый результат, и * можно с

# Селекторы атрибутов

Например, можно выделить все элементы, которые содержат определенный атрибут:

```
/* Элементы содержащие атрибут */
[href] { color: red; }

/* Значение атрибута в точности равно заданному */
[href="http://ya.ru"] { color: red; }
/* Значение атрибута содержит заданное значение */
[href*="http://ya.ru"] { color: red; }

/* Значение атрибута начинается с заданного значения */
[href^="https://"] { color: red; }
/* Значение атрибута заканчивается заданным значением */
[href$="ya.ru"] { color: red; }
```

# Динамические псевдоклассы

Псевдоклассы — такие селекторы, которые позволяют выбрать элементы в зависимости от их состояния.

Псевдоклассы состояния ссылок:

```
a:link { color: blue }
a:active { color: red }
a:hover { color: green }
a:visited { color: purple }
a:focus { color: yellow }
```

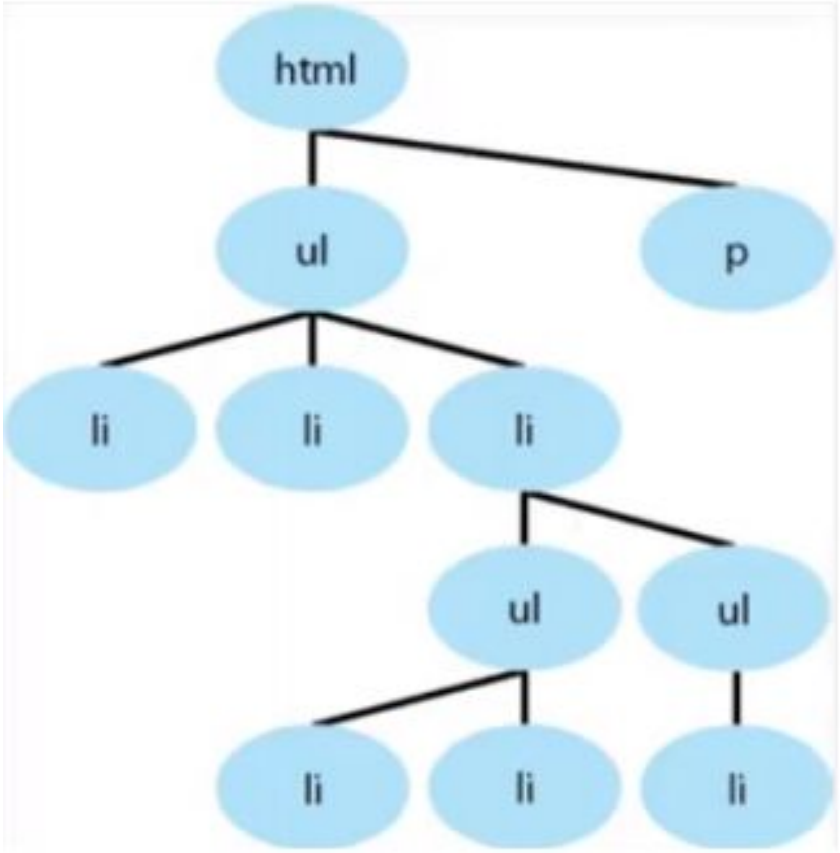
# Динамические псевдоклассы

- :enabled и :disabled позволяют выбирать элементы в зависимости от их доступности для взаимодействия с пользователем.
- :checked позволяет выбрать все элементы, в которых есть атрибут checked.
- :indeterminate соответствует неопределенному состоянию checkbox’ов.
- :read-only позволяет выбрать элементы с атрибутом ”только для чтения”.
- :valid позволяет выбрать валидные элементы форм.

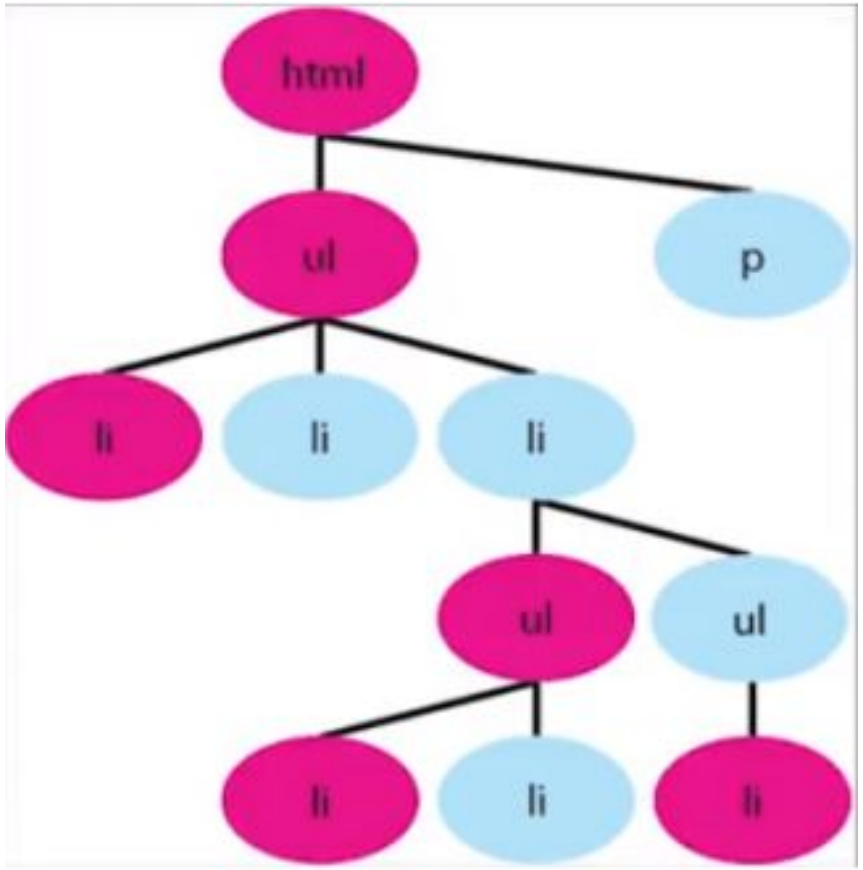
# Структурные псевдоклассы

- :root выбирает корневой элемент HTML-документа
- :first-child/:last-child выбирают первый или последний дочерний элемент.
- :nth-child/:nth-last-child(2n+1) позволяют выбрать n-ый элемент или nый элемент с конца.
- :nth-of-type/:nth-last-of-type(-n+4)
- :only-child/:only-of-type выбирает элемент, который является единственным дочерним для своего родительского элемента, или единственным данного типа.
- :empty выбирает пустые элементы.

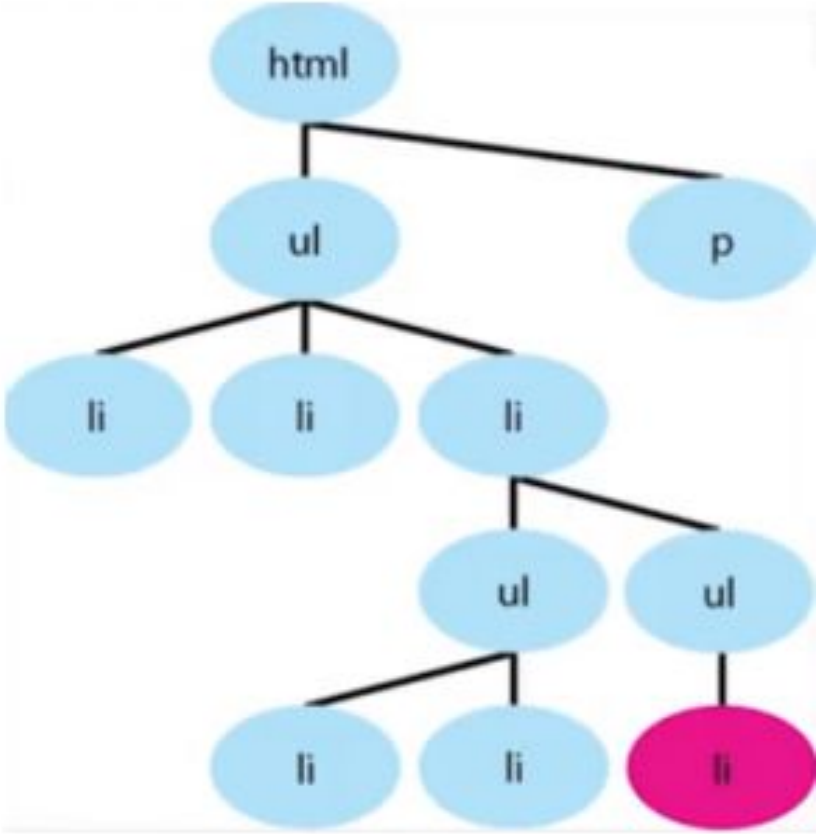
Рассмотрим следующую структуру HTML документа:



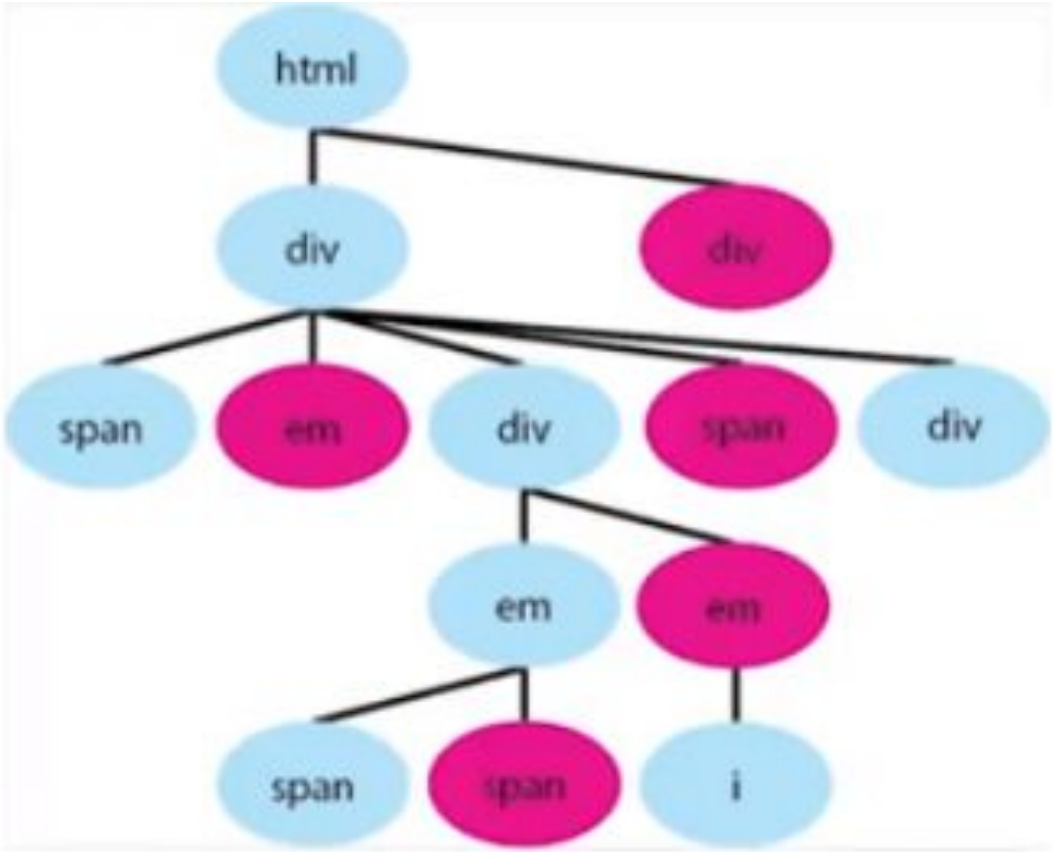
В результате применения селектора :first-child будут выделены следующие элементы:



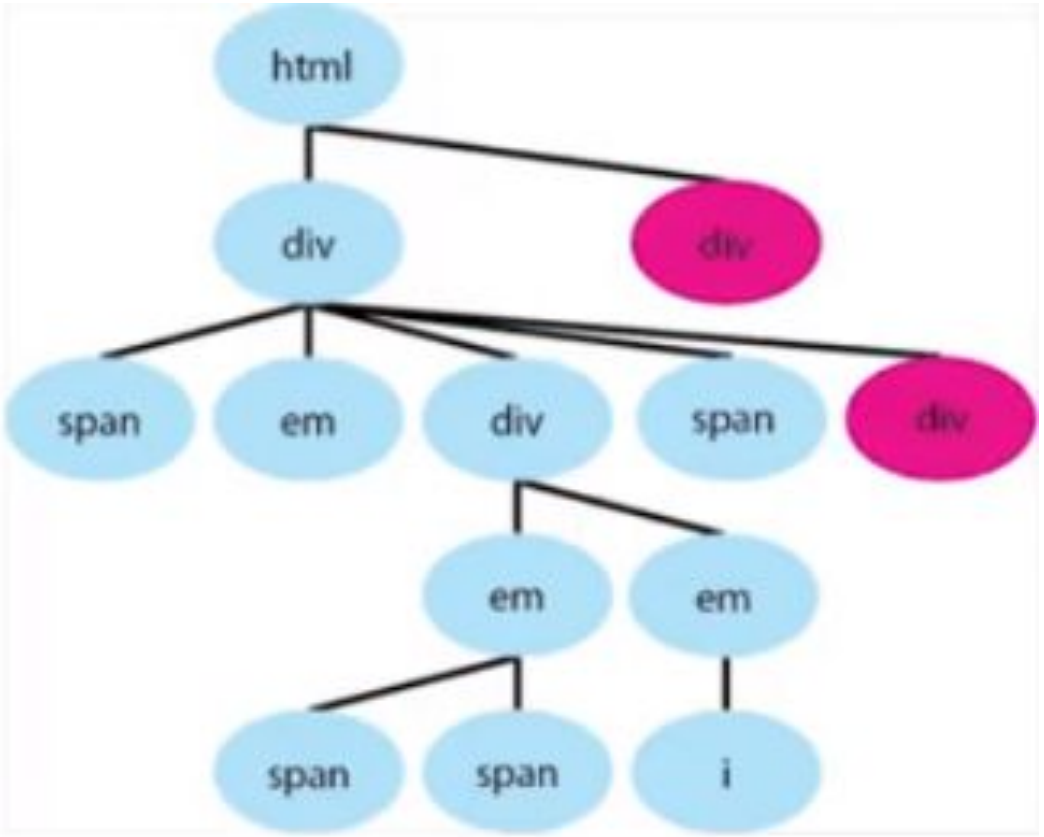
А в результате применения :only-child



Результат применения селектора :nth-child(2n), который выбирает четные дочерние элементы, имеет вид:



При применении селектора :nth-of-type типом считается часть селектора до двоеточия. Результат применения div:nth-child(2n):



# Псевдоэлементы

- ::first-letter который выделяет первую букву текста:

```
::first-letter { color: red }
```

- ::first-line который выделяет первую строку текста:

```
::first-line { color: blue }
```

- ::before, ::after выделяют создаваемые в структуре документа элементы сразу после открывающего тега и сразу до закрывающего соответственно.

```
::before,
::after {
    color: red;
    font-weight: bold;
    content: '»';
}
::before { content: '«';}
```

# Отношения элементов в CSS

Рассмотрим для примера следующий код:

```
<ul>
    <li>пункт <i>курсив</i></li>
    <li>пункт <i>курсив</i></li>
    <li>пункт <i>курсив</i></li>
    <li>пункт <i>курсив</i></li>
</ul>
```

# Селектор потомков (контекстный селектор)

```html
<ul>
    <li>пункт</li>
    <li>пункт</li>
    <li>пункт</li>
    <li>пункт</li>
    <li>пункт</li>
</ul>
<ol>
    <li>пункт</li>
    <li>пункт</li>
    <li>пункт</li>
    <li>пункт</li>
    <li>пункт</li>
</ol>
```

```css
ul li {
    color: red;
}
ol li {
    color: green;
}
```

# Селектор потомков

Селектор потомков не обязательно должен состоять из двух селекторов тегов - их может быть любое количество, записанное через пробел.

```
1  ul li i {
2      color: red;
3  }
```

# Дочерний селектор

Пусть у нас есть следующий код:

```
<p>
    текст <b><i>жирный курсив</i></b>
</p>
<p>
    текст <i>просто курсив</i>
</p>
```

Пусть мы хотим выбрать все теги i, являющиеся потомками абзацев.

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

текст ***жирный курсив***

текст *просто курсив*

# Дочерний селектор

Давайте теперь выберем те теги i, которые являются непосредственными потомками наших абзацев. В этом нам поможет дочерний селектор >.

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

```
<p>
    текст <b><i>жирный курсив</i></b>
</p>
<p>
    текст <i>просто курсив</i>
</p>
```

текст ***жирный курсив***

текст *просто курсив*

# Соседний селектор

Контекстные селекторы используются для вложенных друг в друга элементов, а соседние селекторы для расположенных рядом.

Соседние селекторы записываются с помощью знака +.

```
1  <ul>
2      <li class="a"></li>
3      <li class="b"></li>
4  </ul>
```

```
1  .a + .b {
2      color:red;
3  }
```

# Соседний селектор

Селектор .a+li , а также li+.b , или даже li+li тоже применит стили к элементу с классом b, т.е. ко второму элементу списка.

А вот селектор .b+.a не сработает, т.е. элемент с классом b находится после элемента с классом a в разметке.

Селектор .c .a+.b сработает для тега с классом b, если сразу перед ним расположен тег с классом a и оба тега расположены внутри тега с классом c.

```
1  <ul>
2      <li class="a"></li>
3      <li class="b"></li>
4  </ul>
```

# Сестринский селектор

Этот селектор похож на соседний селектор, но он менее строгий. Соседний селектор выберет только первый элемент, следующий сразу же за указанным элементом.

Селектор селектор1 ~ селектор2 выберет все элементы, подходящие под селектор2, расположенные после элемента селектор1.

Представим, что вам нужно сделать наклонный шрифт для всех абзацев <p>, которые идут после заголовка <h1>. Код CSS будет выглядеть так:

```css
h1 ~ p {
  font-style: italic;
}
```

```html
<div>
    <p>Текст</p>
    <h1>Заголовок 1</h1>
    <p>Текст</p>
    <p>Текст</p>
    <h2>Заголовок 2</h2>
    <p>Текст</p>
</div>

<div>
    <p>Текст</p>
</div>
```

# Специфичность

Пусть дан следующий CSS файл:

```
1  div {
2      color: red;
3  }
4
5  div {
6      color: blue;
7  }
```

Для одного и того же элемента сначала задается красный цвет, а потом синий. В результате цвет будет выбран синий.

# Специфичность

Пусть теперь дан другой CSS код:

```
1  html div {
2      color: red;
3  }
4
5  div {
6      color: blue;
7  }
```

# Специфичность

Считается специфичность следующим образом:

• Первое число — количество селекторов по идентификатору.

• Второе число — количество селекторов классов, псевдоклассов (кроме :not) или по атрибуту.

• Третье число — количество селекторов элементов (тегов) или псевдоэлементов

# Специфичность

```
div {
    color: blue;
    font-weight: bold;
    font-size: 12px;
}

html div {
    color: red;
}

div {
    font-size: 15px;
}
```

```
div {
    color: blue; /* 0 0 1 */
    font-weight: bold; /* 0 0 1 */
    font-size: 12px; /* 0 0 1 */
}

div {
    font-size: 15px; /* 0 0 1 */
}

html div {
    color: red; /* 0 0 2 */
}
```

```
color: blue; /* 0 0 1 */
font-weight: bold; /* 0 0 1 */
font-size: 12px; /* 0 0 1 */
font-size: 15px; /* 0 0 1 */
color: red; /* 0 0 2 */
```

# Приоритет правил

Инлайн — это способ указать стили непосредственно внутри HTML элемента:

```
<div style="color: blue"></div>
```

Ключевое свойство !important используется внутри значения объявления, которое повышает приоритет данного объявления:

```
div {
    color: green !important;
}
```

Также можно указать !important внутри инлайн стилей:

```
<div style="color: white !important"></div>
```

# Специфичность

Инлайн стили по умолчанию приоритетнее стилей в CSS.

Стили в CSS с !important приоритетнее инлайн стилей.

Инлайн стили с !important приоритетнее всего.

# Наследование стилей

Пусть даны следующий CSS код и HTML разметка:

```html
<div>
    Привет,
    <abbr title="Крымский федеральный университет">
        КФУ
    </abbr>
</div>
```

```css
div {
    color: green; /* 0 0 1 */
    /* abbr { color : green } - нет специфичности */
}
```

# Наследование стилей

```
* {
    color: red; /* 0 0 0 */
}
div {
    color: green; /* 0 0 1 */
    /* abbr { color : green } - нет специфичности */
}
```

# Наследование стилей

Наследуемые стили:

```
color                   list-style-position
cursor                  list-style-image
direction               list-style
empty-cells             line-height
font-family             quotes
font-size               text-align
font-weight             text-indent
font-style              text-transform
font-variant            visibility
font                    white-space
letter-spacing          word-spacing
list-style-type
```

# Наследование стилей

Порядок применения стилей CSS следующий:

- Стили браузера
- Стили пользователя и/или плагинов браузера
- Стили страницы
- Стили страницы с !important
- Стили пользователя и/или плагинов браузера с !important

Именно такая последовательность наложения стилей и называется каскад.