

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 оформления 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

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

```
1 <body>
2     <h1>Основной заголовок</h1>
3     <p>Текст параграфа 1</p>
4     <h2>Подзаголовок 1</h2>
5     <p>Текст параграфа 2</p>
6     <h2>Подзаголовок 2</h2>
7     <p>Текст параграфа 3</p>
8 </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>
```

```
1 div.heading {  
2   font-size: 50px;  
3 }  
4 div.red {  
5   color: red;  
6 }
```

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>
```

```
1 div.heading {  
2     font-size: 50px;  
3 }  
4 div.red {  
5     color: red;  
6 }
```

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

```
1 div.heading {  
2     font-size: 50px;  
3 }  
4 div.red {  
5     color: red;  
6 }
```

Селекторы идентификаторов

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

Так, например, следующий код не будет работать:

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

```
1 div#heading {
2     font-size: 50px;
3 }
4 div#red {
5     color: red;
6 }
```

Селекторы идентификаторов

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

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

```
1 div#heading {  
2     font-size: 50px;  
3 }  
4 div#red {  
5     color: red;  
6 }
```

Универсальный селектор

```
1 <h2>Красный заголовок</h2>  
2 <div>Красный заголовок</div>  
3 <p>Красный заголовок</p>
```

```
1 * {  
2   color: red;  
3 }
```

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

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

- Комбинирование селектора тега и селектора идентификатора:

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

- Комбинирование селектора тега, селектора идентификатора и селектора класса:

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

- Можно даже селектора идентификатора, но это не имеет смысла:

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

- Также можно комбинировать универсальный селектор с любым селектором, но такая запись универсальной: эти два примера дадут одинаковый результат, и

```
* можно с  
1  *.class { color: red; }  
2  .class { color: red; }
```


Селекторы атрибутов

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

```
1  /* Элементы содержащие атрибут */
2  [href] { color: red; }
3
4  /* Значение атрибута в точности равно заданному */
5  [href="http://ya.ru"] { color: red; }
6  /* Значение атрибута содержит заданное значение */
7  [href*="http://ya.ru"] { color: red; }
8
9  /* Значение атрибута начинается с заданного значения */
10 [href^="https://"] { color: red; }
11 /* Значение атрибута заканчивается заданным значением */
12 [href$="ya.ru"] { color: red; }
```

Динамические псевдоклассы

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

Псевдоклассы состояния ссылок:

```
1 a:link { color: blue }
2 a:active { color: red }
3 a:hover { color: green }
4 a:visited { color: purple }
5 a:focus { color: yellow }
```

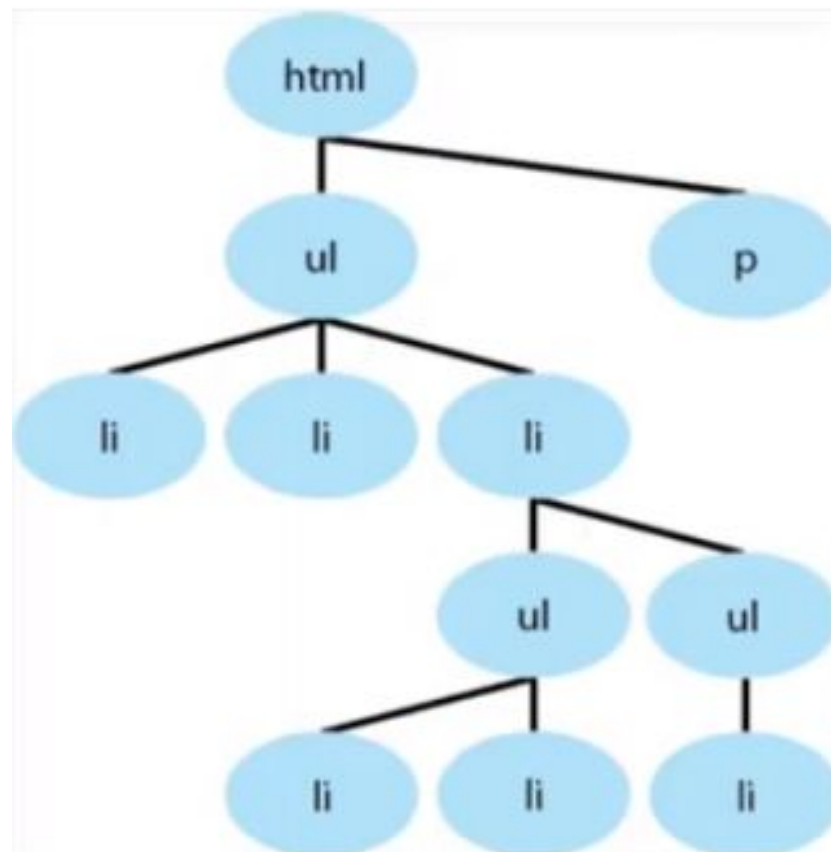
Динамические псевдоклассы

- `: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.

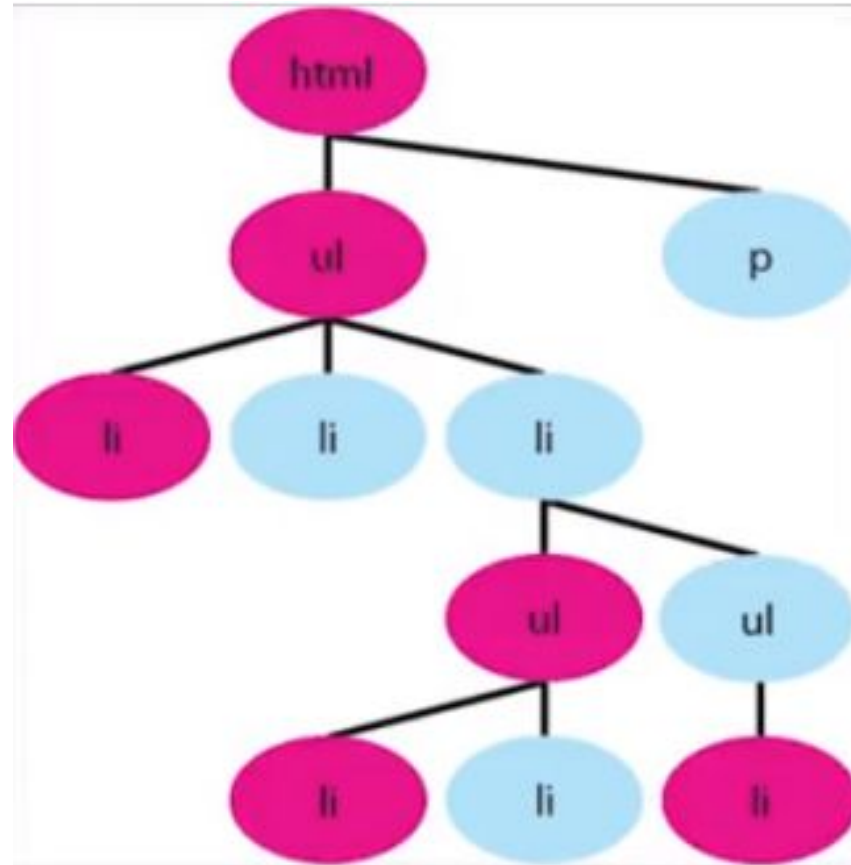
Структурные псевдоклассы

- `: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 n-ый 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.

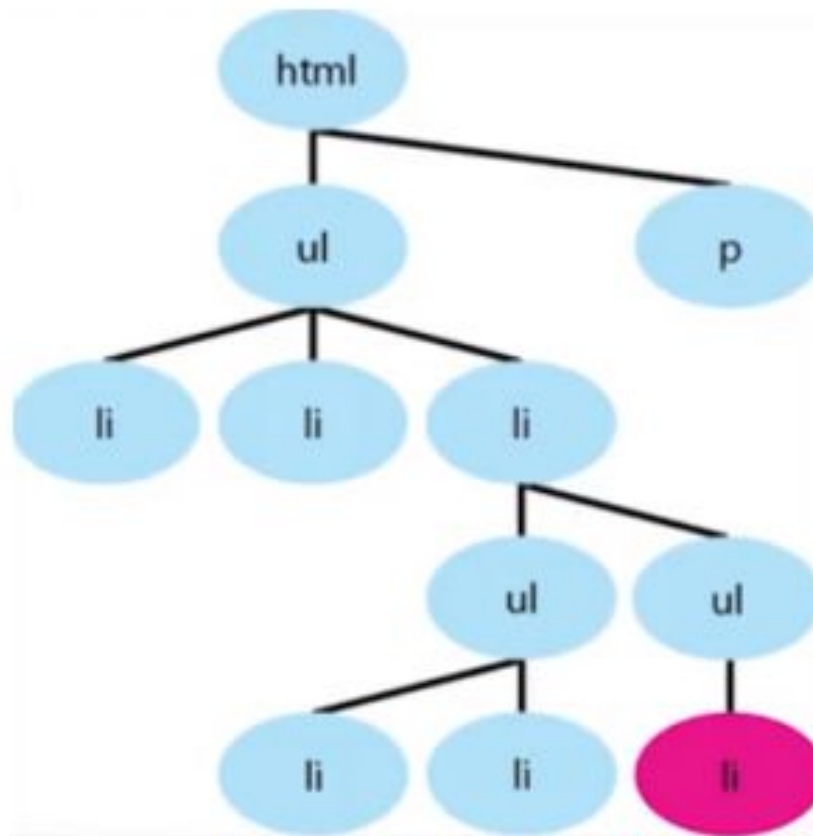
Рассмотрим следующую структуру HTML документа:



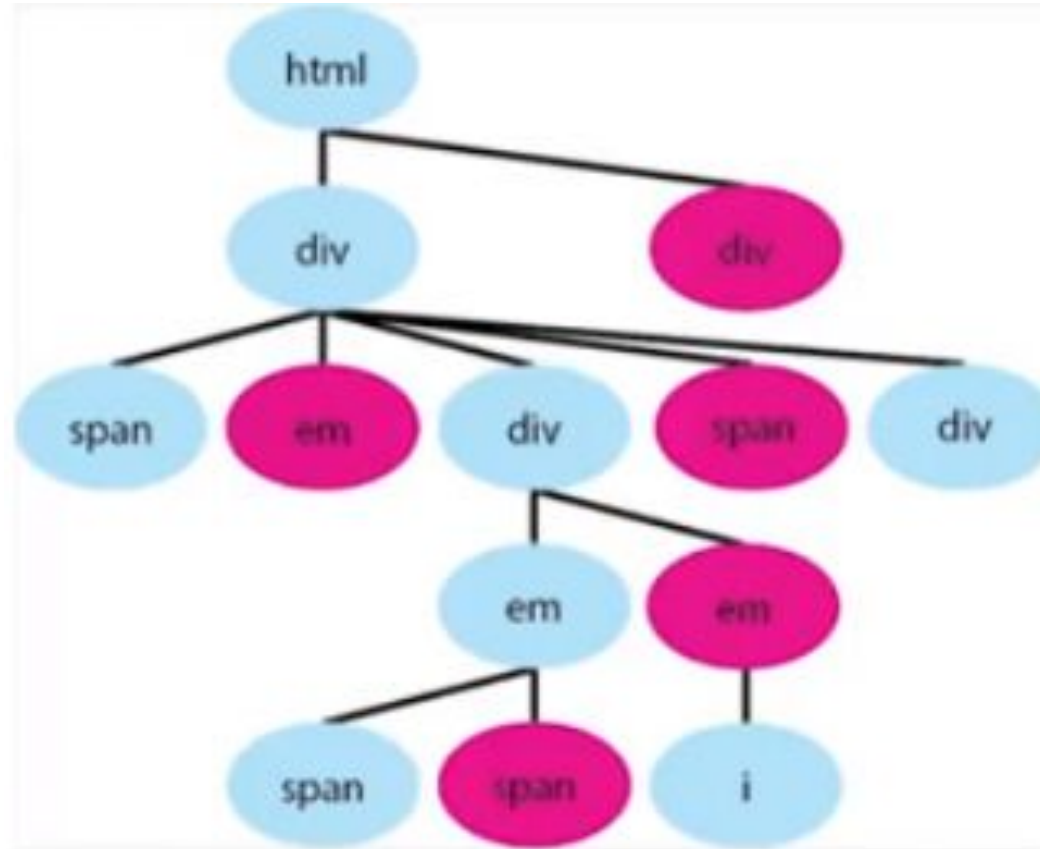
В результате применения селектора `:first-child` будут выделены следующие элементы:



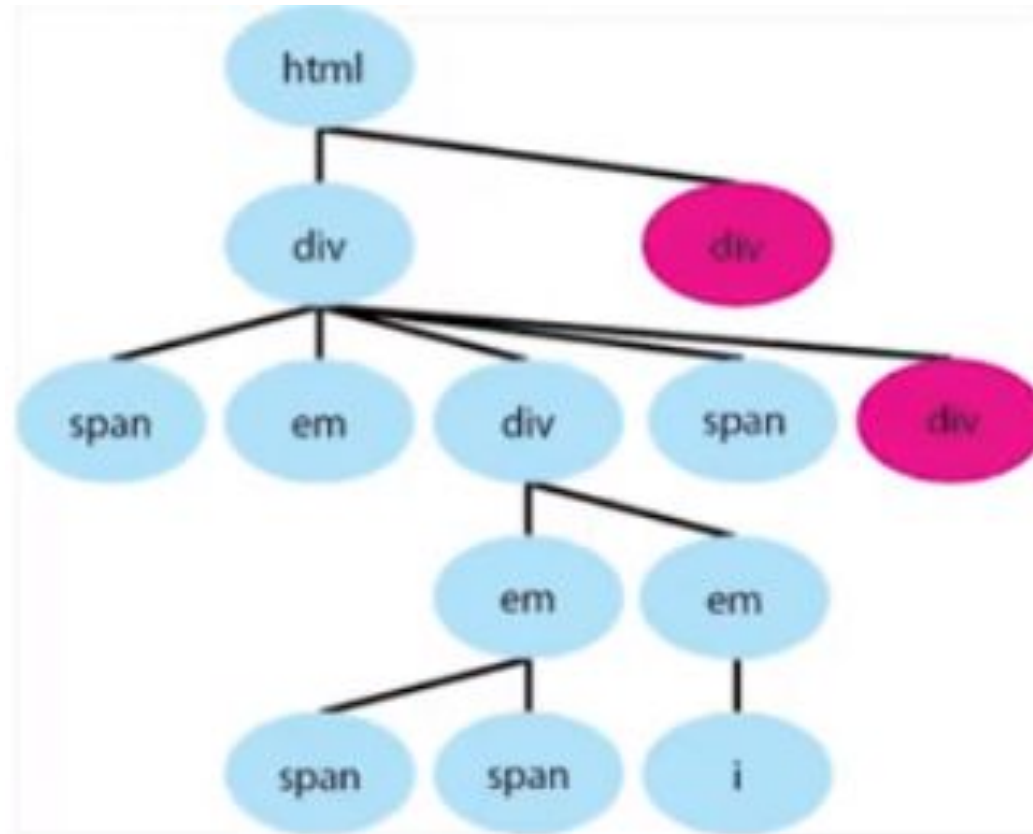
А в результате применения :only-child



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:

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

- `::first-line` который вылепляет первую строку текста:

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

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

```
1  ::before,  
2  ::after {  
3      color: red;  
4      font-weight: bold;  
5      content: '>';  
6  }  
7  ::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)

```
1 <ul>
2   <li>пункт</li>
3   <li>пункт</li>
4   <li>пункт</li>
5   <li>пункт</li>
6   <li>пункт</li>
7 </ul>
8 <ol>
9   <li>пункт</li>
10  <li>пункт</li>
11  <li>пункт</li>
12  <li>пункт</li>
13  <li>пункт</li>
14 </ol>
```

```
1 ul li {
2   color: red;
3 }
4 ol li {
5   color: green;
6 }
```

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:

```
1 <p>
2   текст <b><i>жирный курсив</i></b>
3 </p>
4 <p>
5   текст <i>просто курсив</i>
6 </p>
```

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

```
1 p i {
2   color: red;
3 }
```

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 `>`.

```
1 p > i {  
2   color: red;  
3 }
```

текст *жирный курсив*

текст *просто курсив*

```
1 <p>  
2   текст <b><i>жирный курсив</i></b>  
3 </p>  
4 <p>  
5   текст <i>просто курсив</i>  
6 </p>
```

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 работает, т.е. элемент с классом `b` находится после элемента с классом `a` в разметке.

Селектор `.c .a+.b` работает 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:

```
1 h1 ~ p {  
2   font-style: italic;  
3 }
```

```
1 <div>  
2   <p>Текст</p>  
3   <h1>Заголовок 1</h1>  
4   <p>Текст</p>  
5   <p>Текст</p>  
6   <h2>Заголовок 2</h2>  
7   <p>Текст</p>  
8 </div>  
9  
10 <div>  
11   <p>Текст</p>  
12 </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

```
1  div {
2    color: blue;
3    font-weight: bold;
4    font-size: 12px;
5  }
6
7  html div {
8    color: red;
9  }
10
11 div {
12   font-size: 15px;
13 }
```

```
1  div {
2    color: blue; /* 0 0 1 */
3    font-weight: bold; /* 0 0 1 */
4    font-size: 12px; /* 0 0 1 */
5  }
6
7  div {
8    font-size: 15px; /* 0 0 1 */
9  }
10
11 html div {
12   color: red; /* 0 0 2 */
13 }
```

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

Приоритет правил

Инлайн — это способ указать стили непосредственно внутри HTML элемента:

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

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

```
1 div {  
2   color: green !important;  
3 }
```

Также можно указать `!important` внутри инлайн стилей:

```
1 <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:

```
1 <div>
2   Привет,
3   <abbr title="Крымский федеральный университет">
4     КФУ
5   </abbr>
6 </div>

1 div {
2   color: green; /* 0 0 1 */
3   /* abbr { color : green } - нет специфичности */
4 }
```

Наследование стилей

```
1 * {
2   color: red; /* 0 0 0 */
3 }
4 div {
5   color: green; /* 0 0 1 */
6   /* abbr { color : green } - нет специфичности */
7 }
```

Наследование стилей

Наследуемые стили:

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

|

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.