CSS (Cascading Style Sheets)

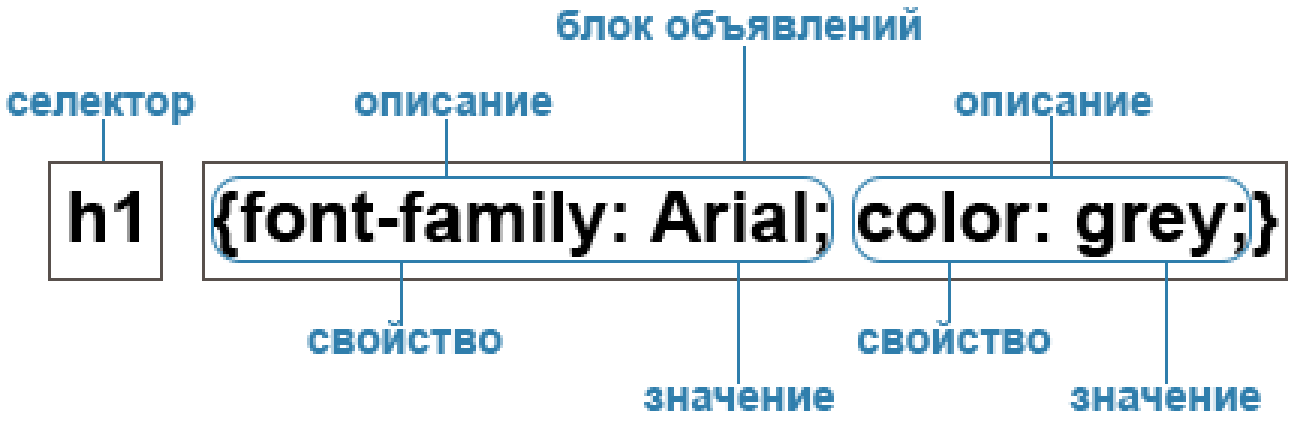
CSS (Cascading Style Sheets) — язык таблиц стилей, который позволяет прикреплять стиль к структурированным документам (например, документам HTML).

Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

CSS (Cascading Style Sheets)

Объявление стиля состоит из двух частей: селектора и объявления.



Виды таблиц стилей

- Внешняя таблица стилей
- Внутренние стили
- Встроенные стили
- Правило @import

Встроенные стили

Когда мы пишем встроенные стили, мы пишем CSS-код в HTML-файл, непосредственно внутри элемента с помощью атрибута style

```
<p style="color: red;">
    Обратите внимание на этот текст.
</p>
```

Задание цвета

Существуют несколько основных способов представления цветов:

- В виде #123ABC. Представление в виде трёх пар шестнадцатеричных цифр, где каждая пара отвечает за свой цвет:
  - Две первые цифры — красный
  - Две в середине — зелёный
  - Две последние цифры — синий

  Возможно также краткое представление цвета в виде #ABC, что будет интерпретировано как #AABBCC.
- Представление ключевыми словами, например green, black. Во избежание случаев, когда указанное ключевое слово «не понимается» браузером, следует использовать лишь небольшой набор основных цветов, используемых во всех браузерах.
- В виде RGB(*,*,*), где «*» — числа от 0 до 255, обозначающих количество соответствующего цвета (красный, зелёный, синий) в получаемом цвете.
- Возможен и RGBA(*,*,*,*), где первые 3 «*» — компоненты цвета, задающиеся в диапазоне 0 до 255, а последняя «*» — уровень непрозрачности (альфа-канал), задающийся рациональными числами от 0 до 1.

Примеры цветов: https://www.w3schools.com/colors/colors_picker.asp

Внешняя таблица стилей

Внешняя таблица стилей представляет собой текстовый файл с расширением .css, в котором находится набор CSS-стилей элементов. Файл создаётся в редакторе кода, так же как и HTML-страница. Внутри файла могут содержатся только стили, без HTML-разметки. Внешняя таблица стилей подключается к веб-странице с помощью элемента `<link>`, расположенного внутри раздела `<head></head>`. Такие стили работают для всех страниц сайта.

```
<head>
    <link rel="stylesheet" href="css/style.css">
    <link rel="stylesheet" href="css/assets.css" media="all">
</head>
```

Внутренние стили

Внутренние стили встраиваются в раздел `<head></head>` HTML-документа и определяются внутри элемента `<style></style>`. Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям (заданным через атрибут `style`).

```
<head>
    <style>
    h1,  h2 {
       color: red;
       font-family: "Times New Roman", Georgia, Serif;
       line-height: 1.3em;
    }
    </style>
</head>
<body>
    ...
</body>
```

Правило @import

Правило @import позволяет загружать внешние таблицы стилей. Чтобы директива @import работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами:

```
<style>
    @import url(mobile.css);
    p {
     font-size: 0.9em;
     color: grey;
    }
</style>
```

Виды селекторов

Селекторы представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы. Селекторами могут быть элементы, их классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

- Универсальный селектор
- Селектор элемента
- Селектор класса
- Селектор идентификатора
- Селектор потомка
- Дочерний селектор
- Сестринский селектор
- Селектор атрибута
- Селектор псевдокласса
- Селектор структурных псевдоклассов
- Селектор структурных псевдоклассов типа
- Селектор псевдоэлемента

Селектор элемента

Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта.

```
h1 {
    font-family: Lobster, cursive;
    color: red;
  }
p {
   color: grey;
   border: solid;
}
```

Селектор класса

Селекторы класса позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта.

```
<h1 class="headline">Заголовок</h1>
```

```
h1.headline {
    text-transform: uppercase;
    color: blue;
}
```

Универсальный селектор

Универсальный селектор обозначается звёздочкой (*). Соответствует любому тегу.

Убирая звёздочки с простых селекторов достигается тот же эффект. Например, * .warning и .warning считаются равными.

```
* {margin: 0; }

*.warning {color:red;}

*#maincontent {border: 1px solid blue;}

*[lang^=en]{color:green;}
```

Селектор класса и универсальный селектор

Если элемент имеет несколько атрибутов класса, их значения объединяются с пробелами.

```
<h1 class="headline rad">Заголовок</h1>
```

```
.headline {
    text-transform: uppercase;
    color: blue;
}
.rad {
    border: 1px solid blue;
}
```

Селектор идентификатора

Селектор идентификатора позволяет форматировать один конкретный элемент. Значение id должно быть уникальным, на одной странице может встречаться только один раз и должно содержать хотя бы один символ. Значение не должно содержать пробелов.

Нет никаких других ограничений на то, какую форму может принимать id, в частности, идентификаторы могут состоять только из цифр, начинаться с цифры, начинаться с подчеркивания, состоять только из знаков препинания и т. д.

Уникальный идентификатор элемента может использоваться для различных целей, в частности, как способ ссылки на конкретные части документа с использованием идентификаторов фрагментов, как способ нацеливания на элемент при создании сценариев и как способ стилизации конкретного элемента из CSS.

Селектор идентификатора

```
<div id="main">
    Основной контент.
</div>
```

```
#main {
    border: 1px solid red;
}
```

Селектор потомка

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера. Например, `ul li {color: green;}` — выберет все элементы `li`, являющиеся потомками всех элементов `ul`.

Если нужно отформатировать потомки определенного элемента, этому элементу нужно задать стилевой класс:

`p.grass a {color: green;}` — данный стиль применится ко всем ссылкам потомкам абзаца с классом grass;

`p .grass a {color: green;}` — если добавить пробел, то будут стилизованы ссылки, расположенные внутри любого элемента класса .grass, который является потомком элемента `<p>`;

`.grass a {color: green;}` — данный стиль применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом .grass.

Дочерний селектор

Дочерний элемент является прямым потомком содержащего его элемента. У одного элемента может быть несколько дочерних элементов, а родительский элемент у каждого элемента может быть только один. Дочерний селектор позволяет применить стили только если дочерний элемент идёт сразу за родительским элементом и между ними нет других элементов, то есть дочерний элемент больше ни во что не вложен.

Например, `p > strong` — выберет все элементы `strong`, являющиеся дочерними по отношению к элементу `p`.

Сестринский селектор

Сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать элементы из группы элементов одного уровня:

`h1 + p` — выберет все первые абзацы, идущие непосредственно за любым элементом `<h1>`, не затрагивая остальные абзацы;

`h1 ~ p` — выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку `<h1>` и идущие сразу после него.

Селектор атрибута

Селекторы атрибутов выбирают элементы на основе имени атрибута или значения атрибута:

[атрибут] — все элементы, содержащие указанный атрибут, `[alt]` — все элементы, для которых задан атрибут alt;

селектор[атрибут] — элементы данного типа, содержащие указанный атрибут, `img[alt]` — только картинки, для которых задан атрибут alt;

селектор[атрибут="значение"] — элементы данного типа, содержащие указанный атрибут с конкретным значением, `img[title="flower"]` — все картинки, название которых содержит слово flower;

селектор[атрибут~="значение"] — элементы частично содержащие данное значение, например, если для элемента задано несколько классов через пробел, `p[class~="feature"]` — абзацы, имя класса которых содержит feature;

Селектор атрибута

селектор[атрибут^="значение"] — элементы, значение атрибута которых начинается с указанного значения, `a[href^="https://"]` — все ссылки, начинающиеся на https://;

селектор[атрибут$="значение"] — элементы, значение атрибута которых заканчивается указанным значением, `img[src$=".png"]` — все картинки в формате png;

селектор[атрибут*="значение"] — элементы, значение атрибута которых содержит в любом месте указанное слово, `a[href*="book"]` — все ссылки, название которых содержит book.

Селектор псевдокласса

Псевдоклассы — это классы, фактически не прикрепленные к HTML-элементам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу.

Псевдоклассы характеризуют элементы со следующими свойствами:

:link — не посещенная ссылка;

:visited — посещенная ссылка;

:hover — любой элемент, по которому проводят курсором мыши;

:focus — интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;

:active — элемент, который был активизирован пользователем;

:lang() — элементы с текстом на указанном языке;

:target — элемент с символом #, на который ссылаются в документе;

Селектор псевдокласса (для форм)

:valid — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;

:invalid — поля формы, содержимое которых не соответствует указанному типу данных;

:enabled — все активные поля форм;

:disabled — заблокированные поля форм, т.е., находящиеся в неактивном состоянии;

:in-range — поля формы, значения которых находятся в заданном диапазоне;

:out-of-range — поля формы, значения которых не входят в установленный диапазон;

:not(селектор) — элементы, которые не содержат указанный селектор — класс, идентификатор, название или тип поля формы — :not([type="submit"]);

:checked — выделенные (выбранные пользователем) элементы формы.

Селектор структурных псевдоклассов

Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

:nth-child(odd) — нечётные дочерние элементы;

:nth-child(even) — чётные дочерние элементы;

:nth-child(3n) — каждый третий элемент среди дочерних;

:nth-child(3n+2) — выбирает каждый третий элемент, начиная со второго дочернего элемента (+2);

:nth-child(n+2) — выбирает все элементы, начиная со второго;

:nth-child(3) — выбирает третий дочерний элемент;

Селектор структурных псевдоклассов

:nth-last-child() — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с :nth-child(), но начиная с последнего, в обратную сторону;

:first-child — позволяет оформить только самый первый дочерний элемент;

:last-child — позволяет форматировать последний дочерний элемент;

:only-child — выбирает элемент, являющийся единственным дочерним элементом;

:empty — выбирает элементы, у которых нет дочерних элементов;

:root — выбирает элемент, являющийся корневым в документе — элемент html.

Селектор структурных псевдоклассов типа

Указывают на конкретный тип дочернего элемента:

:nth-of-type() — выбирает элементы по аналогии с :nth-child(), при этом берёт во внимание только тип элемента;

:first-of-type — выбирает первый дочерний элемент данного типа;

:last-of-type — выбирает последний элемент данного типа;

:nth-last-of-type() — выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением, начиная с конца;

:only-of-type — выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.

```
p:first-of-type {
  font-size: 1.25em;
}
div:nth-of-type(even) {
  background: red;
}
```

Селектор псевдоэлемента

Псевдоэлементы отличаются от псевдоклассов тем, что они не реагируют на состояние платформы, а действуют так, как если бы они вставляли новый элемент с помощью CSS. Кроме того, синтаксис псевдоэлементов отличается от синтаксиса псевдоклассов, потому что вместо одинарного двоеточия (:) в них используется двойное двоеточие (::).

Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства content:

::before — вставляет генерируемое содержимое перед элементом;

::after — добавляет генерируемое содержимое после элемента.

```
.my-element::before {
   content: 'Prefix - ';
}
```

Селектор псевдоэлемента

Перечень функций псевдоэлементов не ограничен вставкой содержимого. Их также можно использовать для нацеливания на определенные части элемента. Предположим, у вас есть список. С помощью псевдоэлемента `::marker` можно применить стиль к каждому маркеру (или номеру) в списке.

```
/* Теперь в списке будут либо красные точки, либо красные номера */
li::marker {
  color: red;
}
```

Кроме того, с помощью псевдоэлемента ::selection можно применять стили к содержимому, выделенному пользователем.

```
::selection {
  background: black;
  color: red;
}
```

Группировка селекторов

Один и тот же стиль можно одновременно применить к нескольким элементам. Для этого необходимо в левой части объявления перечислить через запятую нужные селекторы:

```
h1, h2, p, span {
    color: blue;
    background-color: white;
}
```

Комбинация селекторов

Для более точного отбора элементов для форматирования можно использовать комбинации селекторов:

`#main-article p` — выберет все абзацы-потомки элемента с идентификатором p

`a[href][title]` — выберет все ссылки, для которых заданы атрибуты href и title;

`img[alt*="css"]:nth-of-type(even)` — выберет все четные картинки, альтернативный текст которых содержит слово css.

Наследование и каскадирование

- Наследование – это механизм, когда определенные в контейнере свойства, автоматически назначаются вложенным в этот контейнер элементам.
- Каскадность – это механизм CSS, который определяет какие стили в итоге будут применены к элементу и как конфликтующие правила переопределяют друг друга.

Наследование

Наследование является механизмом, с помощью которого определенные свойства передаются от предка к его потомкам. Спецификацией CSS предусмотрено наследование свойств, относящихся к **текстовому содержимому страницы**, таких как `color, font, text-align и др.` Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

Свойства, относящиеся к **форматированию блоков**, не наследуются. Это `background, border, height и width, margin и др.`

Наследование

Принудительное наследование

- С помощью ключевого слова `inherit` можно принудить элемент наследовать любое значение свойства родительского элемента. Это работает даже для тех свойств, которые не наследуются по умолчанию.

Как задаются и работают CSS-стили

- Стили могут наследоваться от родительского элемента (наследуемые свойства или с помощью значения `inherit`).
- **Стили, расположенные в таблице стилей ниже, отменяют стили, расположенные в таблице выше.**

Каскадирование

Каскадирование — это механизм, который управляет конечным результатом в ситуации, когда к одному элементу применяются разные CSS-правила. Существует три критерия, которые определяют порядок применения свойств — **правило !important, специфичность и порядок**, в котором подключены таблицы стилей.

Правило `!important`

Вес правила можно задать с помощью ключевого слова `!important`, которое добавляется сразу после значения свойства, например,

```
span {font-weight: bold!important;}.
```

Правило необходимо размещать в конец объявления перед закрывающей скобкой, без пробела. Такое объявление будет иметь приоритет над всеми остальными правилами. Это правило позволяет отменить значение свойства и установить новое для элемента из группы элементов в случае, когда нет прямого доступа к файлу со стилями.

Порядок подключённых таблиц

Вы можете создать несколько внешних таблиц стилей и подключить их к одной веб-странице. Если в разных таблицах будут встречаться разные значения свойств одного элемента, то в результате к элементу применится правило, находящееся в таблице стилей, идущей в списке ниже.

Специфичность

Для каждого правила браузер вычисляет специфичность селектора, и если у элемента имеются конфликтующие объявления свойств, во внимание принимается правило, имеющее наибольшую специфичность. Значение специфичности состоит из четырех частей: 0, 0, 0, 0.

Специфичность селектора определяется следующим образом:

- для встроенного стиля, добавленного непосредственно к элементу — 1, 0, 0, 0;
- для id добавляется 0, 1, 0, 0;
- для class добавляется 0, 0, 1, 0;
- для каждого элемента и псевдоэлемента добавляется 0, 0, 0, 1;
- универсальный селектор не имеет специфичности.

В результате к элементу применятся те правила, специфичность которых больше.

Например, если на элемент действуют две специфичности со значениями 0, 0, 0, 2 и 0, 1, 0, 1, то выиграет второе правило.

Специфичность

```
h1 {color: lightblue;} /*специфичность 0, 0, 0, 1*/
em {color: silver;} /*специфичность 0, 0, 0, 1*/
h1 em {color: gold;} /*специфичность: 0, 0, 0, 1 + 0, 0, 0, 1 = 0, 0, 0, 2*/
div#main p.about {color: blue;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 + 0, 0, 0, 1 + 0, 0, 1, 0 = 0, 1, 1, 2*/
.sidebar {color: grey;} /*специфичность 0, 0, 1, 0*/
#sidebar {color: orange;} /*специфичность 0, 1, 0, 0*/
li#sidebar  {color: aqua;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 = 0, 1, 0, 1*/
```