CSS basic

Vira Huskova

CSS (Cascading Style Sheets), или каскадные таблицы стилей, используются для описания внешнего вида документа, написанного языком разметки.

Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.

Объявление стиля состоит из двух частей: элемента веб-страницы — **селектора**, и команды форматирования — **блока объявления**. Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.

Каскадные таблицы и их виды

1. Внешняя

Внешняя таблица стилей представляет собой текстовый файл с расширением **.css**, в котором находится набор CSS-стилей элементов.

Внешняя таблица стилей подключается к веб-странице с помощью тега <link>, расположенного внутри раздела <head></head>. Такие стили работают для всех страниц сайта.

К каждой веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько тегов <link>, указав в атрибуте тега media назначение данной таблицы стилей. rel="stylesheet" указывает тип ссылки (ссылка на таблицу стилей).

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

Атрибут type="text/css" не является обязательным по стандарту HTML5, поэтому его можно не указывать. Если атрибут отсутствует, по умолчанию используется значение type="text/css".

2. Внутренние стили

Внутренние стили встраиваются в раздел <head></head> HTML-документа и определяются внутри тега <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>
```

3. Встроенные стили

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

```
<p style="font-weight: bold; color:
red;">Some text.</p>
```

Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям (заданным через атрибут style).

@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>
```

Селекторы

- **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

Соответствует любому HTML-элементу. Например, * {margin-top: 20px;} добавит верхний отступ для всех элементов сайта.

Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом: *:after {CSS-стили}, *:checked {CSS-стили}.

Селектор элемента

Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта. Например, h1 {font-family: Lobster, cursive;} задаст общий стиль форматирования всех заголовков h1.

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. Например, для создания заголовка с классом headline необходимо добавить атрибут class со значением headline в открывающий тег <h1> и задать стиль для указанного класса. Стили, созданные с помощью класса, можно применять к другим элементам, не обязательно данного типа.

```
.headline {
text transform: uppercase;
color: lightblue;
}
```

Селектор идентификатора

Селектор идентификатора позволяет форматировать **один** конкретный элемент. Идентификатор id должен быть уникальным и на одной странице может встречаться только один раз.

```
<div id="sidebar"></div>

#sidebar {
        width: 300px; float: left;
}
```

Селекторы потомка

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера.

```
ul li {
        text-transform: uppercase;
}
```

выберет все элементы li, являющиеся потомками всех элементов ul.

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

```
p.first a {
        color: green;
}
```

данный стиль применится ко всем ссылкам, потомкам абзаца с классом first;

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

```
.first a {
        color: green;
}
```

данный стиль применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом .first.

```
.list li a .upper img {color: red;}

ul li span {color:green;}

div nav .wrapper li a {color:blue;}
```

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 тегом <h1>, не затрагивая остальные абзацы;
https://codepen.io/anon/pen/MPmvEp

h1 ~ p — выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку h1 и идущие сразу после него.
https://codepen.io/anon/pen/ZqKJaK

Селекторы 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[атрибут|="значение"] — элементы, список значений атрибута которых начинается с указанного слова, p[class|="feature"] — абзацы, имя класса которых feature или начинается на feature;
селектор[атрибут^="значение"] — элементы, значение атрибута которых начинается с указанного значения, a[href^="http://"] — все ссылки, начинающиеся на http://;
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.

Селекторы псевдокласса

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

: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;

: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;

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

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

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

:checked — выделенные (выбранные пользователем) элементы формы.

Селекторы 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;

https://codepen.io/anon/pen/XxRaoZ

: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 тега;
:last-child — позволяет форматировать последний дочерний элемент тега;
: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 тега:
: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.

Селектор псевдоэлемента

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

:first-letter — выбирает первую букву каждого абзаца, применяется только к блочным элементам;

:first-line — выбирает первую строку текста элемента, применяется только к блочным элементам;

:before — вставляет генерируемое содержимое перед элементом;

:after — добавляет генерируемое содержимое после элемента.

Комбинация селекторов

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

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.

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: tomato; background: white; }
```

Наследование и каскад

Наследование заключается в том, что **элементы наследуют свойства от своего родителя** (элемента, их содержащего).

Каскад проявляется в том, как разные виды таблиц стилей применяются к документу, и как **конфликтующие правила переопределяют друг друга**.

Наследование

CSS предусмотрено наследование свойств, относящихся к **текстовому содержимому** страницы, таких как `color`, `font`, `letter-spacing`, `line-height`, `list-style`, `text-align`, `text-indent`, `text-transform`, `visibility`, `white-space` и `word-spacing`. 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`, `display`, `float` и `clear`, `height` и `width`, `margin`, `min-max-height` и `width`, `outline`, `overflow`, `padding`, `position`, `text-decoration`, `vertical-align` и `z-index`.

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.

https://codepen.io/lostsou41216364/pen/BqRbvG

Каскад

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.

!important

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

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

Правило необходимо размещать в конец объявления перед закрывающей скобкой. Такое объявление будет иметь приоритет над всеми остальными правилами **(Если не перекрывается другим !important).**

Это правило позволяет отменить значение свойства и установить новое для элемента из группы элементов в случае, когда нет прямого доступа к файлу со стилями.

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 `id` добавляется `0, 1, 0, 0`;
для `class` добавляется `0, 0, 1, 0`;
для каждого элемента и псевдоэлемента добавляется `0, 0, 0, 1`;
для встроенного стиля, добавленного непосредственно к элементу — `1, 0, 0, 0`;
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.

```
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*/
```

https://htmlacademy.ru/courses/66/run/15

Порядок подключения таблиц

Можно 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.

Шрифты

CSS-шрифты — набор свойств для управления внешним видом текста веб-страниц. Используя различные шрифты для заголовков, абзацев и других элементов, можно задавать определенный стиль письменных сообщений.

Текст основного содержимого веб-страницы должен быть в первую очередь читабельным. Не рекомендуется использовать более двух шрифтов на странице.

font-family

```
@font-face {
    font-family: Pompadur; /* Имя шрифта */
    src: url(fonts/pompadur.ttf); /* Путь к файлу со шрифтом */
   }
```

Свойство используется для выбора начертания шрифта.

Поскольку невозможно предсказать, установлен тот или иной шрифт на компьютере посетителя вашего сайта, рекомендуется прописывать все возможные варианты однотипных шрифтов.

В таком случае браузер будет проверять их наличие, последовательно перебирая предложенные варианты. **Наследуется**.

Важно! Если в названии шрифта имеются пробелы или символы (например, #, $, %), то оно заключается в кавычки. Это делается для того, чтобы браузер мог понять, где начинается и заканчивается название шрифта.

```
p {font-family: "Times New Roman", Georgia, Serif;}
```

Font-style

Свойство позволяет выбрать стиль начертания для текста. При этом разница между курсивом и **наклонным** текстом заключается в том, что **курсивное** начертание вносит небольшие изменения в структуру каждой буквы, а наклонный текст представляет собой наклонную версию прямого текста. **Наследуется**.

`normal`	Значение по умолчанию, устанавливает для текста обычное начертание шрифта.
`italic`	Выделяет текст курсивом.
`oblique`	Устанавливает наклонное начертание шрифта.
`initial`	Устанавливает значение свойства в значение по умолчанию.
`inherit`	Наследует значение свойства от родительского элемента.

```
h1 {font-style: normal;}
h1 {font-style: italic;}
h1 {font-style: oblique;}
```

font-weight

```
h1 {font-weight: normal;}
span {font-weight: bold;}
span {font-weight: bolder;}
span {font-weight: lighter;}
h1 {font-weight: 100;}
```