БИБЛИОТЕКА ПРОГРАММИСТА

Грег Сидельников

НАГЛЯДНЫЙ CSS

ББК 32.988-02-018
УДК 004.738.5
С34

Сидельников Грег

С34 Наглядный CSS. — СПб.: Питер, 2021. — 224 с.: ил. — (Серия «Библиотека программиста»).

ISBN 978-5-4461-1618-8

На 1 июня 2018 года CSS содержал 415 уникальных свойств, относящихся к объекту style в любом элементе браузера Chrome. Сколько свойств доступно в вашем браузере на сегодняшний день? Наверняка уже почти шесть сотен. Наиболее важные из них мы и рассмотрим.

Грег Сидельников упорядочил свойства по основной категории (положение, размерность, макеты, CSS-анимация и т. д.) и визуализировал их работу.

Вместо бесконечных томов документации — две с половиной сотни иллюстраций помогут вам разобраться во всех тонкостях работы CSS. Эта книга станет вашим настольным справочником, позволяя мгновенно перевести пожелания заказчика и собственное видение в компьютерный код!

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.988-02-018
УДК 004.738.5

ISBN 978-1983065637 англ.
ISBN 978-5-4461-1618-8

Оглавление

1 Свойства и значения CSS

На 1 июня 2018 года CSS содержал **415** уникальных свойств, относящихся к объекту `style` в любом элементе браузера Chrome. По состоянию на 21 декабря 2018 года насчитывалось **522** уникальных свойства. Всего за семь месяцев в Chrome была добавлена поддержка более 100 новых свойств. Это будет происходить постоянно, так как спецификация CSS продолжает развиваться.

Сколько свойств доступно в вашем браузере на сегодняшний день? Вы можете проверить это самостоятельно с помощью простого фрагмента кода JavaScript.

```
001 // Создание нового HTML-элемента
002 let element = document.createElement("div");
003
004 let p = 0; // Создание счетчика
005 for (index in element.style)
006     p++;
007
008 // Выводит 522 в Chrome по состоянию на 21 декабря 2018 года
009 console.log( p );
```

Для просмотра всех доступных в вашем браузере свойств CSS запустите данный код JavaScript (быстрее всего проверить CSS и JavaScript можно с помощью codepen.io). Результаты могут различаться для разных браузеров и версий.

В процессе создания книги все свойства были распечатаны и упорядочены по основным категориям (*положение*, *размерность*, *разметка*, *CSS-анимация* и т. д.). Затем для каждого свойства, которое каким-то важным образом отображает или изменяет визуальный вывод, была создана схема с кратким описанием названия и значения.

Здесь не описаны редко используемые свойства CSS (или те, которые на момент написания книги не имели полной поддержки основных браузеров). Они бы только все запутали.

Мы сосредоточимся только на свойствах, которые в настоящее время широко применяются веб-дизайнерами и разработчиками. Много усилий было уделено созданию схем **grid-** и **flex-верстки**. Помимо этого, я включил краткое руководство по **SASS/SCSS**, но выбрал только наиболее важные функции, о которых вы должны знать.

1.1. Внешнее размещение

Код CSS можно сохранить в отдельном внешнем файле (например, `style.css`) и включить с помощью HTML-элемента `link`.

Исходный код файла `style.css`:

```
001 body p
002 {
003     background: white;
004     color: black;
005     font-family: Arial, sans-serif;
006     font-size: 16px;
007     line-height: 1.58;
008     text-rendering: optimizeLegibility;
009     -webkit-font-smoothing: antialiased;
010 }
```

Пример ссылки на внешний CSS-файл:

```
001 <html>
002     <head>
003         <title>Добро пожаловать на сайт.</title>
004         <link rel = "stylesheet"
005               type = "text/css"
006               href = "style.css" />
007      </head>
008      <body>
009          <p>CSS-стили записаны в файле style.css
010             и применяются к содержимому этой страницы.</p>
011      </body>
012 </html>
```

1.2. Внутреннее размещение

Вы можете ввести CSS-код непосредственно в HTML-документ между двумя тегами элемента `style`:

```
<html>
  <head>
    <style type = "text/css">
        body p
        {
            background: white;
            color: black;
            font-family: Arial, sans-serif;
            font-size: 16px;
            line-height: 1.58;
            text-rendering: optimizeLegibility;
            -webkit-font-smoothing: antialiased;
        }
    </style>
  </head>

  <body>
      <p>CSS-стили внутри элемента style
         и применяются к этому абзацу
         в HTML-коде веб-страницы</p>
  </body>
</html>
```

1.3. Строковое размещение

Строковое размещение CSS-кода с использованием атрибута `style` в элементе HTML:

```
<html>
  <head></head>
  <body style = "font-family: Arial;">
      <p>При выводе в браузере этот абзац
         наследует форматирование шрифтом Arial
         из строкового стиля
         в родительском элементе.</p>
  </body>
</html>
```

1.4. Селекторы

Теперь мы знаем, где CSS-код находится в HTML-документе. Но прежде, чем мы начнем рассматривать каждое свойство по отдельности, полезно ознакомиться с грамматикой языка CSS — правилами синтаксиса для определения свойств и значений.

Наиболее распространенный селектор — само *имя HTML-элемента* (напомню, что за редким исключением HTML-элемент состоит из двух тегов, открывающего и закрывающего).

Использование имени тега приведет к выделению всех элементов данного типа. Выберем элемент `body` по его имени:

```
001 body { /* Сюда помещаются свойства CSS */ }
```

На первый взгляд, поскольку в HTML-документе есть только один элемент `body`, это единственное, что будет выбрано.

Однако из-за каскадной специфики CSS любое свойство, которое мы берем в скобки, также будет применяться ко всем его потомкам (дочерним элементам, содержащимся в элементе `body`, даже если мы не станем явно указывать их стиль).

Это пустой селектор. Он выбирает элемент `body`, но пока не назначает ему никаких свойств.

Ниже приведены несколько других примеров выбора объектов по имени их HTML-элемента. Это самые распространенные приемы.

```
001 /* Выбираем все элементы абзаца, p */
002 p { }
003
004 /* Выбираем все элементы div */
005 div { }
006
007 /* Выбираем все элементы p, только если они находятся
008    в элементах div */
009 div p { }
```

Ваши стили CSS будут заключены в `{ ...`*`эти`*`... }` скобки.

Инструкция CSS состоит из *селектора* и пары `свойство: значение;`. Несколько свойств должны быть разделены *точкой с запятой*. Начнем с одного свойства, просто чтобы посмотреть, как выглядит синтаксис CSS-свойства:

```
001 <div id = "box">контент<div>
```

В CSS идентификатор указывается в виде символа *хештега* `#`:

```
001 #box { свойство: значение; }
```

Используйте идентификаторы (`id`) для маркировки элементов всякий раз при наличии *уникального* контейнера.

Не форматируйте каждый отдельный HTML-элемент с помощью идентификаторов, задействуйте их для именования глобальных родительских элементов или для более значимых элементов (например, тех, которые необходимо часто обновлять из-за изменения содержимого).

Что, если мы хотим выбрать несколько элементов одновременно?

```
001 <ul>
002     <li class = "элемент">1<li>
003     <li class = "элемент">2<li>
004     <li class = "элемент">3<li>
005 </li>
```

Аналогично атрибут класса (`class`) обозначается селектором точки (`.`):

```
001 .item { line-height: 1.80; }
```

В данном примере точка используется для выбора нескольких элементов, имеющих одно и то же имя класса. Так свойству `line-height` (высота строки) присваивается значение `1.50` (что примерно соответствует 150 % высоты шрифта).

Специальные правила CSS: селектор `:root` применяет их ко *всем* HTML-элементам. Вы можете использовать `:root`, чтобы установить значения CSS *по умолчанию* для всего документа.

Установим Arial в качестве шрифта по умолчанию для всего документа или sans-serif, если шрифт Arial недоступен; вы можете указать столько шрифтов, сколько пожелаете:

```
001 :root { font-family: Arial, sans-serif; }
```

Селектор `:root` также часто используется для *глобального* хранения CSS-переменных.

Создайте CSS-переменную с именем `--red-color` и присвойте ей значение цвета `red`:

```
001 :root { --red-color: red; }
```

Учтите, что все имена переменных CSS должны начинаться с двойного дефиса `--`.

Теперь вы можете использовать CSS-переменную `--red-color` в качестве значения в стандартных селекторах CSS:

```
001 div { color: var(--red-color) ; }
```

Мы изучили, как селектор `:root` позволяет сохранить CSS-переменные, и узнали, что он также может сбросить до значений по умолчанию *все форматирование документа*.

Селектор звездочка (`*`) выполняет те же функции.

Можно использовать селектор * для достижения эффекта, вызываемого применением `:root`. Единственное отличие состоит в том, что селектор * нацелен абсолютно на все элементы в документе, а `:root` — только на контейнер документа без его дочерних элементов:

```
001 * { font-family: Arial, sans-serif; }
```

Несмотря на то что добавление селектора * дает тот же эффект, менее целесообразно использовать его для применения стилей ко всему документу (вместо этого задействуйте `:root`).

Лучше всего селектор * подходит для пакетного выбора «всех элементов» в пределах определенного родительского элемента.

Селектор `#parent *` может использоваться для выбора всех потомков родительского элемента независимо от их типа:

```
001 <div id = "parent">
002     <div>A</div>
003     <div>B</div>
004     <ul>
005         <li>l</li>
006         <li>2</li>
007     </ul>
008     <p>Текст.</p>
009 </div>
```

Продолжая экспериментировать с селекторами, вы заметите, что можно выбирать одни и те же HTML-элементы, используя разные *комбинации* селекторов.

Например, все следующие комбинации выбирают один и тот же набор элементов (все потомки родительского элемента, без учета самого предка). Селектор `#parent *` может использоваться для выбора всех потомков родительского элемента независимо от их типа:

```
001 /* Выбираем все дочерние элементы #parent */
002 #parent * { color: blue; }
003
004 /* Объединяем несколько селекторов, используя запятую */
005 #parent div,
006 #parent ul,
007 #parent p { color: blue; }
008
009 /* Применяем псевдоселекторы :nth-child */
010 #parent nth-child(1),
011 #parent nth-child(2),
012 #parent nth-child(3),
013 #parent nth-child(4) { color: blue; }
```

Конечно, наличие возможности не означает, что так нужно делать. Это лишь пример.

Наиболее целесообразным решением в данном случае является селектор `#parent *`. Но каждый проект, сайт или приложение требуют разметки, уникальной по своей структуре и назначению.

Поначалу создание селекторов может показаться простой задачей. Но это до тех пор, пока вы не углубитесь в более сложные примеры пользовательского интерфейса. С каждым разом ваш CSS-код будет становиться все сложнее и сложнее.

Сложность CSS-кода тесно связана со структурой самого HTML-документа. Поэтому даже некоторые из самых «умных» селекторов часто могут «пересекаться» с селекторами, созданными впоследствии, вызывая конфликты. Изучение CSS — настоящее искусство. Ваши навыки создания CSS-селекторов будут улучшаться только в процессе регулярной практики!

При работе над реальным проектом и ввиду постоянного усложнения макета приложений вы не сможете даже предположить, как часто конкретное CSS-свойство не будет работать нужным образом.

Когда пропущен определенный сценарий и допущена ошибка, разработчики часто используют ключевое слово `!Important`, чтобы быстро исправить проблему.

Вы можете переопределить любой стиль CSS, добавив ключевое слово `!Important` в конец кода CSS.

```
001 /* Выбираем все дочерние элементы #parent */
002 #parent * { color: blue; }
003
004 /* Выбираем только div в #parent и меняем цвет на красный */
005 #parent div { color: red; }
006
007 /* Проверяем, что все div во всем документе зеленого цвета */
008 div { color: green !important; }
```

Весьма заманчиво использовать ключевое слово `!Important` для принудительной установки стиля CSS. Но обычно такая практика считается порочной, поскольку игнорируется каскадная логика таблиц стилей!

ПРЕДОСТЕРЕЖЕНИЕ

Директиву !Important лучше вообще не использовать. Даже если вам покажется, что вы решаете проблему, применение директивы может значительно усложнить обслуживание вашего кода CSS.

Идеальный вариант, когда CSS-селекторы максимально простые и эффективные. Однако такой баланс не всегда легко сохранить. Я обычно начинаю с набросков своего CSS-кода на бумаге. Потратив немного больше времени на обдумывание структуры приложения и написание заметок, вы быстрее создадите наиболее оптимальные селекторы.

1.5. Взаимосвязь между свойствами и значениями

Не все CSS-свойства одинаковы. В зависимости от типа свойства значение может быть **мерой пространства**, заданного в *пикселах*, единицах *pt*, *em* или *fr*, **цветом**, указанным в виде имени (*red*, *blue*, *black* и т. д.), шестнадцатеричного значения (*#0F0* или *#00FF00...*) или rgb (*r*, *g*, *b*).

В других случаях значение уникально для конкретного свойства, и его нельзя использовать с любым другим свойством. Например, CSS-свойство `transform` может принимать значение, указанное с помощью ключевого слова `rotate`.

В данном случае принимается угол в градусах — CSS требует добавления букв `deg` к числовому значению градуса.

```
001 /* поворот этого элемента на 45 градусов по часовой стрелке */
002
003 #box {
004     transform: rotate(45deg);
005 }
```

Однако это не единственный способ указать угол. CSS предлагает еще три *типа* единиц, специально предназначенных для указания угла поворота: `grad`, `rad` и `turn`.

```
001 /* 200 градиан (град)*/
002 transform: rotate(200grad);
003
004 /* 1,4 радиан */
005 transform: rotate(1.4rad);
006
007 /* 0,5 оборота или 180 градусов (1 оборот = 360 градусов) */
008 transform: rotate(0.5turn);
```

В данном случае мы используем `grad` (градианы), `rad` (радианы) и `turn` (повороты) в качестве альтернативного способа задания угла поворота HTML-элемента.

Альтернативные способы указания *значений* не редкость для многих других CSS-свойств. Например, `#F00`, `#FF0000`, `red`, `rgb(255, 255, 255)` и `rgba(255, 255, 255, 1.0)` задают один и тот же цвет.

1.6. Комментарии в CSS-коде

Для создания комментариев в коде CSS поддерживается только синтаксис блочных комментариев.

Это делается путем вставки блока текста с использованием символов `/* комментарий */`.

```
/* Установить белый цвет шрифта, используя
   шестнадцатеричное значение */
p { color: #FFFFFF; }

/* Установить белый цвет шрифта, используя сокращенное
   шестнадцатеричное значение */
p { color: #FFF; }

/* Установить белый цвет шрифта, используя название цвета */
p { color: white; }

/* Установить белый цвет шрифта, используя значение RGB */
p { color: rgb(255,255,255); }

/* Создать переменную CSS --white-color
   (обратите внимание на двойной дефис) */
:root { --white-color: rgba(255, 255, 255, 1.0); }

/* Установить белый цвет шрифта,
   используя CSS-переменную */
p { color: var(--white-color); }
```

Обратите внимание, как один и тот же цвет свойства может принимать различные типы значений. При использовании переменных CSS имя переменной предваряет двойной дефис (`--`).

Вы также можете закомментировать раздел CSS-кода целиком. Далее показано временное отключение блока кода CSS для тестирования нового кода или будущей ссылки и т. д.:

```
001 /* Временно отключить данный блок CSS
002     content:"hello";
003     border: 1px solid gray;
004     color: tfFFFFFF;
005     line-height: 48px;
006     padding: 32px;
007 */
```

CSS не поддерживает *// встроенные комментарии*, или, скорее, они не оказывают влияния на CSS-интерпретатор браузера.

1.7. Оформление стилей

В CSS имеется множество свойств, связанных с габаритами и размерами (`left`, `top`, `width`, `height` и др.). Было бы излишним перечислять их все. Поэтому далее в примерах я буду использовать слово *свойство*.

Для указания значения служит шаблон *свойство*: *значение*. Такая комбинация позволит установить фоновые изображения, цвет и другие основные свойства HTML-элементов.

В качестве альтернативы можете использовать шаблон *свойство*: *значение значение значение* для установки нескольких значений одному свойству. Это и есть *сокращенная форма записи*. Значения разделяются пробелом.

Без сокращенной формы записи вы бы указали каждую часть свойства в отдельной строке.

```
001 /* Фон */
002 background-color:    black;
003 background-image:    url("image.jpg");
004 background-position: center;
005 background-repeat:   no-repeat;
006
007 /* Сокращенная форма записи, только одна строка кода! */
008 background: black url("image.jpg") center no-repeat;
```