

CSS (Cascading Style Sheets, каскадные таблицы стилей)

Стили представляют собой набор параметров, управляющих видом и положением элементов веб-страницы.

Поскольку на файл со стилем можно ссылаться из любого веб-документа, это приводит в итоге к сокращению объема повторяющихся данных. А благодаря разделению кода и оформления повышается гибкость управления видом документа и скорость работы над сайтом.

CSS представляет собой свой собственный язык, который совпадает с HTML только некоторыми значениями, например способом определения цвета.

Спецификацию можно почитать по адресу <http://www.w3.org/TR/CSS21/>

Типы стилей

Различают несколько типов стилей, которые могут совместно применяться к одному документу. Это стиль браузера, стиль автора и стиль пользователя.

Стиль браузера

Оформление, которое по умолчанию применяется к элементам веб-страницы браузером. Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей. Например, заголовок страницы, формируемый тегом <H1>, в большинстве браузеров выводится шрифтом с засечками размером 24 пункта.

Стиль автора

Стиль, который добавляет к документу его разработчик.

Стиль пользователя

Это стиль, который может включить пользователь сайта через настройки браузера. Такой стиль имеет более высокий приоритет и переопределяет исходное оформление документа. В браузере Internet Explorer подключение стиля пользователя делается через меню Сервис > Свойство обозревателя > Кнопка «Оформление».

В браузере Opera аналогичное действие происходит через команду Инструменты > Общие настройки > Вкладка «Расширенные» > Содержимое > Кнопка «Параметры стиля».

Указанные типы стилей могут спокойно существовать друг с другом, если они не пытаются изменить вид одного элемента. В случае возникновения противоречия вначале имеет приоритет стиль пользователя, затем стиль автора и последним идет стиль браузера.

Преимущества стилей

Стили являются удобным, практичным и эффективным инструментом при верстке веб-страниц и оформления текста, ссылок, изображений и других элементов. Несмотря на явные плюсы применения стилей, рассмотрим все преимущества CSS, в том числе и незаметные на первый взгляд.

Разграничение кода и оформления

Идея о том, чтобы код HTML был свободен от элементов оформления вроде установки цвета, размера шрифта и других параметров, стара как мир. В идеале, веб-страница должна содержать только теги логического форматирования, а вид элементов задается через стили. При подобном разделении работа над дизайном и версткой сайта может вестись параллельно.

Разное оформление для разных устройств

С помощью стилей можно определить вид веб-страницы для разных устройств вывода: монитора, принтера, смартфона, КПК и др. Например, на экране монитора отображать страницу в одном оформлении, а при ее печати — в другом. Эта возможность также позволяет скрывать или показывать некоторые элементы документа при отображении на разных устройствах.

Расширенные по сравнению с HTML способы оформления элементов

В отличие от HTML стили имеют гораздо больше возможностей по оформлению элементов веб-страниц. Простыми средствами можно изменить цвет фона элемента, добавить рамку, установить шрифт, определить размеры, положение и многое другое.

Ускорение загрузки сайта

При хранении стилей в отдельном файле, он кэшируется и при повторном обращении к нему извлекается из кэша браузера. За счет кэширования и того, что стили хранятся в отдельном файле, уменьшается код веб-страниц и снижается время загрузки документов.

Кэшем называется специальное место на локальном компьютере пользователя, куда браузер сохраняет файлы при первом обращении к сайту. При следующем обращении к сайту эти файлы уже не скачиваются по сети, а берутся с локального диска. Такой подход позволяет существенно повысить скорость загрузки веб-страниц.

Единое стилевое оформление множества документов

Сайт это не просто набор связанных между собой документов, но и одинаковое расположение основных блоков, и их вид. Применение единообразного оформления заголовков, основного текста и других элементов создает преемственность между страницами и облегчает пользователям работу с сайтом и его восприятие в целом. Разработчикам же использование стилей существенно упрощает проектирование дизайна.

Централизованное хранение

Стили, как правило, хранятся в одном или нескольких специальных файлах, ссылка на которые указывается во всех документах сайта. Благодаря этому удобно править стиль в одном месте, при этом оформление элементов автоматически меняется на всех страницах, которые связаны с указанным файлом. Вместо того чтобы модифицировать десятки HTML-файлов, достаточно отредактировать один файл со стилем и оформление нужных документов сразу же поменяется.

Способы добавления стилей на страницу

Для добавления стилей на веб-страницу существует несколько способов, которые различаются своими возможностями и назначением. Далее рассмотрим их подробнее.

Связанные стили

При использовании связанных стилей описание селекторов и их значений располагается в отдельном файле, как правило, с расширением `css`, а для связывания документа с этим файлом применяется тег `<link>`. Данный тег помещается в контейнер `<head>`.

```
<link rel="stylesheet" type="text/css" href="mysite.css">
```

```
<link rel="stylesheet" type="text/css" href="http://www.htmlbook.ru/main.css">
```

Значения атрибутов тега `<link>` — `rel` и `type` остаются неизменными независимо от кода, как приведено в данном примере. Значение `href` задает путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Заметьте, что таким образом можно подключать таблицу стилей, которая находится на другом сайте.

Файл со стилем не хранит никаких данных, кроме синтаксиса CSS. В свою очередь и HTML-документ содержит только ссылку на файл со стилем, т.е. таким способом в полной мере реализуется принцип разделения кода и оформления сайта. Поэтому использование связанных стилей является наиболее универсальным и удобным методом добавления стиля на сайт. Ведь стили хранятся в одном файле, а в HTML-документах указывается только ссылка на него.

Глобальные стили

При использовании глобальных стилей свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы. По своей гибкости и возможностям этот способ добавления стиля уступает предыдущему, но также позволяет хранить стили в одном месте, в данном случае прямо на той же странице с помощью контейнера `<style>`, который входит в контейнер `<head>`.

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```
<title>Глобальные стили</title>
```

```
<style type="text/css">
```

```
h1 {
```

```
font-size: 120%;
```

```
font-family: Verdana, Arial, Helvetica, sans-serif;
```

```
color: #333366;
```

```
}
```

```
</style>
```

```
</head>
```

Внутренние стили

Внутренний или встроенный стиль является по существу расширением для одиночного тега используемого на текущей веб-странице. Для определения стиля используется атрибут `style`, а его значением выступает набор стилевых правил.

```
<p style="font-size: 120%; font-family: monospace; color: #cd66cc">Пример текста</p>
```

В данном примере стиль тега `<p>` задается с помощью атрибута `style`, в котором через точку с запятой перечисляются стилевые свойства.

Внутренние стили рекомендуется применять на сайте ограниченно или вообще отказаться от их использования. Дело в том, что добавление таких стилей увеличивает общий объем файлов, что ведет к повышению времени их загрузки в браузере, и усложняет редактирование документов для разработчиков.

Все описанные методы использования CSS могут применяться как самостоятельно, так и в сочетании друг с другом. В этом случае необходимо помнить об их иерархии. Первым всегда применяется внутренний стиль, затем глобальный стиль и в последнюю очередь связанный стиль.

Импорт CSS

В текущую стилевую таблицу можно импортировать содержимое CSS-файла с помощью команды `@import`. Этот метод допускается использовать совместно со связанными или глобальными стилями, но никак не с внутренними стилями. Общий синтаксис следующий.

```
@import url("имя файла") типы носителей;
```

```
@import "имя файла" типы носителей;
```

После ключевого слова `@import` указывается путь к стилевому файлу одним из двух приведенных способов — с помощью `url` или без него.

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```
<title>Импорт</title>
```

```
<style type="text/css">
```

```
@import url("style/header.css");
```

```
h1 {
```

```
font-size: 120%;
```

```
font-family: Arial, Helvetica, sans-serif;
```

```
color: green;
```

```
}
```

```
</style>
```

```
</head>
```

В данном примере показано подключение файла `header.css`, который расположен в папке `style`.

Аналогично происходит импорт и в файле со стилем, который затем подключается к документу.

```
@import "/style/print.css";
```

```
@import "/style/palm.css";
```

```
BODY {
```

```
font-family: Arial, Verdana, Helvetica, sans-serif;
```

```
font-size: 90%;
```

```
background: white;
```

```
color: black;
```

```
}
```

Каскадирование

Аббревиатура CSS расшифровывается как Cascading Style Sheets (каскадные таблицы стилей), где одним из ключевых слов выступает «каскад». Под каскадом в данном случае понимается одновременное применение разных стилевых правил к элементам документа — с помощью подключения нескольких стилевых файлов, наследования свойств и других методов. Чтобы в подобной ситуации браузер понимал, какое в итоге правило применять к элементу, и не возникало конфликтов в поведении разных браузеров, введены определенные приоритеты.

Ниже приведены приоритеты браузеров, которыми они руководствуются при обработке стилевых правил. Чем выше в списке находится пункт, тем ниже его приоритет, и наоборот.

1. Стиль браузера.
2. Стиль пользователя.
3. Стиль автора.
4. Стиль автора с добавлением `!important`.
5. Стиль пользователя с добавлением `!important`.

Самым низким приоритетом обладает стиль браузера — оформление, которое по умолчанию применяется к элементам веб-страницы браузером. Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей.

!important

Ключевое слово `!important` играет роль в том случае, когда пользователи подключают свою собственную таблицу стилей.

При использовании пользовательской таблицы стилей или одновременном применении разного стиля автора и пользователя к одному и тому же селектору, браузер руководствуется следующим алгоритмом.

- `!important` добавлен в авторский стиль — будет применяться стиль автора.
- `!important` добавлен в пользовательский стиль — будет применяться стиль пользователя.
- `!important` нет как в авторском стиле, так и стиле пользователя — будет применяться стиль автора.
- `!important` содержится в авторском стиле и стиле пользователя — будет применяться стиль

пользователя.

Синтаксис применения **!important** следующий.

Свойство: значение `!important`

Вначале пишется желаемое стилевое свойство, затем через двоеточие его значение и в конце после пробела указывается ключевое слово `!important`.

Повышение важности требуется не только для регулирования приоритета между авторской и пользовательской таблицей стилей, но и для повышения специфичности определенного селектора.

Специфичность

Если к одному элементу одновременно применяются противоречивые стилевые правила, то более высокий приоритет имеет правило, у которого значение специфичности селектора больше. Специфичность это некоторая условная величина, вычисляемая следующим образом. За каждый идентификатор (в дальнейшем будем обозначать их количество через *a*) начисляется 100, за каждый класс и псевдокласс (*b*) начисляется 10, за каждый селектор тега и псевдоэлемент (*c*) начисляется 1. Складывая указанные значения в определенном порядке, получим значение специфичности для данного селектора.

```
*          {} /* a=0 b=0 c=0 -> специфичность = 0 */
li         {} /* a=0 b=0 c=1 -> специфичность = 1 */
li:first-line {} /* a=0 b=0 c=2 -> специфичность = 2 */
ul li      {} /* a=0 b=0 c=2 -> специфичность = 2 */
ul ol+li   {} /* a=0 b=0 c=3 -> специфичность = 3 */
ul li.red   {} /* a=0 b=1 c=2 -> специфичность = 12 */
li.red.level {} /* a=0 b=2 c=1 -> специфичность = 21 */
#t34        {} /* a=1 b=0 c=0 -> специфичность = 100 */
#content #wrap {} /* a=2 b=0 c=0 -> специфичность = 200 */
```

Встроенный стиль, добавляемый к тегу через атрибут `style`, имеет специфичность 1000, поэтому всегда перекрывает связанные и глобальные стили. Однако добавление `!important` перекрывает в том числе и встроенные стили.

Если два селектора имеют одинаковую специфичность, то применяться будет тот стиль, что определен в коде ниже. Чтобы исправить ситуацию, необходимо либо понизить специфичность первого селектора, либо повысить специфичность второго.

Добавление идентификатора используется не только для изменения специфичности селектора, но и для применения стиля только к указанному списку. Поэтому понижение специфичности за счет убирания идентификатора применяется редко, в основном, повышается специфичность нужного селектора.

Селекторы тегов

В качестве селектора может выступать любой тег HTML для которого определяются правила форматирования, такие как: цвет, фон, размер и т.д. Правила задаются в следующем виде.

Тег { свойство1: значение; свойство2: значение; ... }

Вначале указывается имя тега, оформление которого будет переопределено, заглавными или строчными символами не имеет значения. Внутри фигурных скобок пишется стилевое свойство, а после двоеточия — его значение. Набор свойств разделяется между собой точкой с запятой и может располагаться как в одну строку, так и в несколько.

Классы

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега. При использовании совместно с тегами синтаксис для классов будет следующий.

тег.имя класса { свойство1: значение; свойство2: значение; ... }

Внутри стиля вначале пишется желаемый тег, а затем, через точку пользовательское имя класса. Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах классов недопустимо. Чтобы указать в коде HTML, что тег используется с определенным классом, к тегу добавляется атрибут class="Имя класса".

Можно, также, использовать классы и без указания тега. Синтаксис в этом случае будет следующий.

.имя класса { свойство1: значение; свойство2: значение; ... }

При такой записи, класс можно применять к любому тегу.

Классы удобно использовать, когда нужно применить стиль к разным элементам веб-страницы: ячейкам таблицы, ссылкам, абзацам и др.

Одновременное использование разных классов

К любому тегу одновременно можно добавить несколько классов, перечисляя их в параметре class через пробел. В этом случае к элементу применяется стиль, описанный в правилах для каждого класса. Поскольку при добавлении нескольких классов они могут содержать одинаковые стилевые свойства, но с разными значениями, то берется значение у класса, который описан в коде ниже.

В стилях также допускается использовать запись вида .layer1 .layer2, где layer1 и layer2 представляют собой имена классов. Стиль применяется только для элементов, у которых одновременно заданы классы layer1 и layer2.

Идентификаторы

Идентификатор (называемый также «ID селектор») определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты.

Синтаксис применения идентификатора следующий.

#Имя идентификатора { свойство1: значение; свойство2: значение; ... }

При описании идентификатора вначале указывается символ решетки (#), затем идет имя идентификатора. Оно должно начинаться с латинского символа и может содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах идентификатора недопустимо. В отличие от классов идентификаторы должны быть уникальны, иными словами, встречаться в коде документа только один раз.

Обращение к идентификатору происходит аналогично классам, но в качестве ключевого слова у тега используется параметр id, значением которого выступает имя идентификатора. Символ решетки при этом уже не указывается.

Как и при использовании классов, идентификаторы можно применять к конкретному тегу. Синтаксис при этом будет следующий.

тег#имя идентификатора { свойство1: значение; свойство2: значение; ... }

Вначале указывается имя тега, затем без пробелов символ решетки и название идентификатора.

Идентификаторы

- В коде документа каждый идентификатор уникален и должен быть включён лишь один раз.
- Имя идентификатора чувствительно к регистру.
- Стил для идентификатора имеет приоритет выше, чем у классов.

Классы

- Классы могут использоваться в коде неоднократно.
- Имена классов чувствительны к регистру.
- Классы можно комбинировать между собой, добавляя несколько классов к одному тегу.

Контекстные селекторы

При создании веб-страницы часто приходится вкладывать одни теги внутрь других. Чтобы стили для этих тегов использовались корректно, помогут селекторы, которые работают только в определенном контексте. Например, задать стиль для тега `` только когда он располагается внутри контейнера `<p>`. Таким образом можно одновременно установить стиль для отдельного тега, а также для тега, который находится внутри другого.

Контекстный селектор состоит из простых селекторов разделенных пробелом. Так, для селектора тега синтаксис будет следующий.

Тег1 Тег2 { ... }

В этом случае стиль будет применяться к Тегу2 когда он размещается внутри Тег1, как показано ниже.

Не обязательно контекстные селекторы содержат только один вложенный тег. В зависимости от ситуации допустимо применять два и более последовательно вложенных друг в друга тегов.

Более широкие возможности контекстные селекторы дают при использовании идентификаторов и классов. Это позволяет устанавливать стиль только для того элемента, который располагается внутри определенного класса.

При таком подходе легко управлять стилем одинаковых элементов, вроде изображений и ссылок, оформление которых должно различаться в разных областях веб-страницы.

Соседние селекторы

Соседними называются элементы веб-страницы, когда они следуют непосредственно друг за другом в коде документа. Рассмотрим несколько примеров отношения элементов.

`<p>Lorem ipsum dolor sit amet.</p>`

В этом примере тег `` является дочерним по отношению к тегу `<p>`, поскольку он находится внутри этого контейнера. Соответственно `<p>` выступает в качестве родителя ``.

`<p>Lorem ipsum dolor <var>sit</var> amet.</p>`

Здесь теги `<var>` и `` никак не перекрываются и представляют собой соседние элементы. То, что они расположены внутри контейнера `<p>`, никак не влияет на их отношение.

`<p>Lorem ipsum dolor sit amet, <i>consectetur</i> adipiscing <tt>elit</tt>.</p>`

Соседними здесь являются теги `` и `<i>`, а также `<i>` и `<tt>`. При этом `` и `<tt>` к соседним элементам не относятся из-за того, что между ними расположен контейнер `<i>`.

Для управления стилем соседних элементов используется символ плюса (+), который устанавливается между двумя селекторами. Общий синтаксис следующий.

Селектор 1 + Селектор 2 { Описание правил стиля }

Пробелы вокруг плюса не обязательны, стиль при такой записи применяется к Селектору 2, но только в том случае, если он является соседним для Селектора 1 и следует сразу после него.

Соседние селекторы удобно использовать для тех тегов, к которым автоматически добавляются отступы, чтобы самостоятельно регулировать величину отбивки. Например, если подряд идут теги `<h1>` и `<h2>`, то расстояние между ними легко регулировать как раз с помощью соседних селекторов. Аналогично дело обстоит и для идущих подряд тегов `<h2>` и `<p>`, а также в других подобных случаях.

Поскольку при использовании соседних селекторов стиль применяется только ко второму элементу, то размер отступов уменьшается за счет включения отрицательного значения у свойства margin-top. При этом текст поднимается вверх, ближе к предыдущему элементу.

Родственные селекторы

Родственные селекторы по своему поведению похожи на соседние селекторы (запись вида E + F), но в отличие от них стилевые правила применяются ко всем близлежащим элементам. К примеру, для селектора h1~p стиль будет применяться ко всем элементам <p>, располагающихся после заголовка <h1>. При этом <h1> и <p> должны иметь общего родителя, так что если <p> вставить внутрь <div>, то стили применяться уже не будут.

Здесь красный цвет текста будет установлен для всех абзацев.

```
h1 ~ p { color: red; }
```

```
<h1>Заголовок</h1>
```

```
<p>Абзац 1</p>
```

```
<p>Абзац 2</p>
```

Здесь красный цвет текста будет установлен для первого и третьего абзацев. Ко второму абзацу стиль не применяется, поскольку <h1> и <p> не имеют общего родителя.

```
h1 ~ p { color: red; }
```

```
<h1>Заголовок</h1>
```

```
<p>Абзац 1</p>
```

```
<div><p>Абзац 2</p></div>
```

```
<p>Абзац 3</p>
```

Синтаксис

E ~ F { Описание правил стиля }

Для управления стилем родственных элементов используется символ тильды (~), он добавляется между двумя селекторами E и F. Пробелы вокруг тильды не обязательны. Стиль при такой записи применяется к элементу F в том случае, если он имеет того же родителя, что и элемент E и следует сразу после него.

Дочерние селекторы

Дочерним называется элемент, который непосредственно располагается внутри родительского элемента. Синтаксис применения таких селекторов следующий.

Селектор 1 > Селектор 2 { Описание правил стиля }

Стиль применяется к Селектору 2, но только в том случае, если он является дочерним для Селектора 1.

По своей логике дочерние селекторы похожи на селекторы контекстные. Разница между ними следующая. Стиль к дочернему селектору применяется только в том случае, когда он является прямым потомком, иными словами, непосредственно располагается внутри родительского элемента. Для контекстного селектора же допустим любой уровень вложенности.

Иногда правила являются равносильными, поскольку все условия для них выполняются и не противоречат друг другу. В подобных случаях применяется стиль, который расположен в коде ниже.

Заметим, что в большинстве случаев от добавления дочерних селекторов можно отказаться, заменив их контекстными селекторами. Однако использование дочерних селекторов расширяет возможности по управлению стилями элементов, что в итоге позволяет получить нужный результат, а также простой и наглядный код.

Удобнее всего применять указанные селекторы для элементов, которые обладают иерархической структурой — сюда относятся, например, таблицы и разные списки.

Селекторы атрибутов

Многие теги различаются по своему действию в зависимости от того, какие в них используются атрибуты. Например, тег <input> может создавать кнопку, текстовое поле и другие элементы формы всего лишь за счет изменения значения атрибута type. При этом добавление правил стиля к селектору INPUT

применит стиль одновременно ко всем созданным с помощью этого тега элементам. Чтобы гибко управлять стилем подобных элементов, в CSS введены селекторы атрибутов. Они позволяют установить стиль по присутствию определенного атрибута тега или его значения.

Простой селектор атрибута

Устанавливает стиль для элемента, если задан специфичный атрибут тега. Его значение в данном случае не важно. Синтаксис применения такого селектора следующий.

[атрибут] { Описание правил стиля }

Селектор[атрибут] { Описание правил стиля }

Стиль применяется к тем тега, внутри которых добавлен указанный атрибут. Пробел между именем селектора и квадратными скобками не допускается.

Атрибут со значением

Устанавливает стиль для элемента в том случае, если задано определенное значение специфичного атрибута. Синтаксис применения следующий.

[атрибут="значение"] { Описание правил стиля }

Селектор[атрибут="значение"] { Описание правил стиля }

В первом случае стиль применяется ко всем тега, которые содержат указанное значение. А во втором — только к определенным селекторам.

Значение атрибута начинается с определенного текста

Устанавливает стиль для элемента в том случае, если значение атрибута тега начинается с указанного текста. Синтаксис применения следующий.

[атрибут^="значение"] { Описание правил стиля }

Селектор[атрибут^="значение"] { Описание правил стиля }

В первом случае стиль применяется ко всем элементам, у которых значение атрибута начинается с указанного текста. А во втором — только к определенным селекторам. Использование кавычек не обязательно, но только если значение содержит латинские буквы.

Предположим, что на сайте требуется разделить стиль обычных и внешних ссылок — ссылки, которые ведут на другие сайты. Чтобы не добавлять к тегу <a> новый класс, воспользуемся селекторами атрибутов. Внешние ссылки характеризуются добавлением к адресу протокола, например, для доступа к гипертекстовым документам используется протокол HTTP. Поэтому внешние ссылки начинаются с ключевого слова http://, его и добавляем к селектору A.

Значение атрибута оканчивается определенным текстом

Устанавливает стиль для элемента в том случае, если значение атрибута оканчивается указанным текстом. Синтаксис применения следующий.

[атрибут\$="значение"] { Описание правил стиля }

Селектор[атрибут\$="значение"] { Описание правил стиля }

В первом случае стиль применяется ко всем элементам у которых значение атрибута завершается заданным текстом. А во втором — только к определенным селекторам.

Таким способом можно автоматически разделять стиль для ссылок на сайты домена ru и для ссылок на сайты других доменов вроде com.

Значение атрибута содержит указанный текст

Возможны варианты, когда стиль следует применить к тегу с определенным атрибутом, при этом частью его значения является некоторый текст. При этом точно не известно, в каком месте значения включен данный текст — в начале, середине или конце. В подобном случае следует использовать такой синтаксис.

[атрибут*="значение"] { Описание правил стиля }

Селектор[атрибут*="значение"] { Описание правил стиля }

Одно из нескольких значений атрибута

Некоторые значения атрибутов могут перечисляться через пробел, например, имена классов. Чтобы задать стиль при наличии в списке требуемого значения применяется следующий синтаксис.

[атрибут~="значение"] { Описание правил стиля }

Селектор[атрибут~="значение"] { Описание правил стиля }

Стиль применяется в том случае, если у атрибута имеется указанное значение или оно входит в список значений, разделяемых пробелом.

Аналогичный результат можно получить, если использовать конструкцию *= вместо ~=.

Дефис в значении атрибута

В именах идентификаторов и классов разрешено использовать символ дефиса (-), что позволяет создавать значащие значения атрибутов id и class. Для изменения стиля элементов, в значении которых применяется дефис, следует воспользоваться следующим синтаксисом.

[атрибут|= "значение"] { Описание правил стиля }

Селектор[атрибут|= "значение"] { Описание правил стиля }

Стиль применяется к элементам, у которых атрибут начинается с указанного значения или с фрагмента значения, после которого идет дефис.

Например имя класса задано как block-menu-therm, поэтому в стилях будет использоваться конструкция |= "block", поскольку значение начинается именно с этого слова и в значении встречаются дефисы.

Все перечисленные методы можно комбинировать между собой, определяя стиль для элементов, которые содержат два и более атрибута. В подобных случаях квадратные скобки идут подряд.

[атрибут1="значение1"][атрибут2="значение2"] { Описание правил стиля }

Селектор[атрибут1="значение1"][атрибут2="значение2"] { Описание правил стиля }

Универсальный селектор

Иногда требуется установить одновременно один стиль для всех элементов веб-страницы, например, задать шрифт или начертание текста. В этом случае поможет универсальный селектор, который соответствует любому элементу веб-страницы.

Для обозначения универсального селектора применяется символ звездочки (*) и в общем случае синтаксис будет следующий.

**** { Описание правил стиля }***

Псевдоклассы

Псевдоклассы определяют динамическое состояние элементов, которое изменяется со временем или с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на нее курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице. Синтаксис применения псевдоклассов следующий.

селектор:псевдокласс { Описание правил стиля }

Вначале указывается селектор, к которому добавляется псевдокласс, затем следует двоеточие, после которого идет имя псевдокласса. Допускается применять псевдоклассы к именам идентификаторов или классов (a.menu:hover {color: green}), а также к контекстным селекторам (.menu a:hover {background: #fc0}). Если псевдокласс указывается без селектора впереди (:hover), то он будет применяться ко всем элементам документа.

Условно все псевдоклассы делятся на три группы:

- определяющие состояние элементов;
- имеющие отношение к дереву элементов;
- указывающие язык текста.

Псевдоклассы, определяющие состояние элементов

К этой группе относятся псевдоклассы, которые распознают текущее состояние элемента и применяют стиль только для этого состояния.

:active

Происходит при активации пользователем элемента. Например, ссылка становится активной, если навести на нее курсор и щелкнуть мышкой. Несмотря на то, что активным может стать практически любой элемент веб-страницы, псевдокласс :active используется преимущественно для ссылок.

:link

Применяется к непосещенным ссылкам, т.е. таким ссылкам, на которые пользователь еще не нажимал. Браузер некоторое время сохраняет историю посещений, поэтому ссылка может быть помечена как посещенная хотя бы потому, что по ней был зафиксирован переход раньше.

:focus

Применяется к элементу при получении им фокуса. Например, для текстового поля формы получение фокуса означает, что курсор установлен в поле, и с помощью клавиатуры можно вводить в него текст.

Результат будет виден только для тех элементов, которые могут получить фокус. В частности, это теги <a>, <input>, <select> и <textarea>.

:hover

Псевдокласс :hover активизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит.

:visited

Данный псевдокласс применяется к посещенным ссылкам. Обычно такая ссылка меняет свой цвет по умолчанию на фиолетовый, но с помощью стилей цвет и другие параметры можно задать самостоятельно.

Имеет значение порядок следования псевдоклассов. Вначале указывается :visited, а затем идет :hover, в противном случае посещенные ссылки не будут изменять свой цвет при наведении на них курсора.

Псевдокласс :hover не обязательно должен применяться к ссылкам, его можно добавлять и к другим элементам документа. Например ячейкам таблицы.

Псевдоклассы, имеющие отношение к дереву документа

К этой группе относятся псевдоклассы, которые определяют положение элемента в дереве документа и применяют к нему стиль в зависимости от его статуса.

:first-child

Применяется к первому дочернему элементу селектора, который расположен в дереве элементов документа.

Псевдокласс :first-child удобнее всего использовать в тех случаях, когда требуется задать разный стиль для первого и остальных однотипных элементов

Псевдоклассы, задающие язык текста

Для документов, одновременно содержащих тексты на нескольких языках имеет значение соблюдение правил синтаксиса, характерные для того или иного языка. С помощью псевдоклассов можно изменять стиль оформления иностранных текстов, а также некоторые настройки.

:lang

Определяет язык, который используется в документе или его фрагменте. В коде HTML язык устанавливается через атрибут charset тега <meta>. С помощью псевдокласса :lang можно задавать определенные настройки, характерные для разных языков, например, вид кавычек в цитатах.

















Элемент:lang(язык) { ... }

В качестве языка могут выступать следующие значения: ru — русский; en — английский ; de — немецкий ; fr — французский; it — итальянский.

cursor

Устанавливает форму курсора, когда он находится в пределах элемента. Вид курсора зависит от операционной системы и установленных параметров. Значение по умолчанию auto.

Прежде чем воспользоваться возможностью переделать вид курсора, решите, а будет ли он использоваться к месту. Многих пользователей подобные изменения могут ввести в заблуждение, когда, например, вместо традиционной руки, появляющейся при наведении на ссылку, возникает нечто другое. В большинстве случаев, лучше оставить все как есть.

Вид	Значение	Пример
	url	Позволяет установить свой собственный курсор, для этого нужно указать путь к файлу с курсором.
	auto	Вид курсора по умолчанию для текущего элемента.
	inherit	Наследует значение родителя.
	default	P {cursor: default}
	crosshair	P {cursor: crosshair}
	help	P {cursor: help}
	move	P {cursor: move}
	pointer	P {cursor: pointer}
	progress	P {cursor: progress}
	text	P {cursor: text}
	wait	P {cursor: wait}
	n-resize	P {cursor: n-resize}
	ne-resize	P {cursor: ne-resize}
	e-resize	P {cursor :e-resize}
	se-resize	P {cursor: se-resize}
	s-resize	P {cursor: s-resize}
	sw-resize	P {cursor: sw-resize}
	w-resize	P {cursor: w-resize}
	nw-resize	P {cursor :nw-resize}

В зависимости от операционной системы и ее настроек вид курсора может отличаться от приведенных в таблице.

При добавлении курсора из файла синтаксис несколько видоизменится.

cursor: url('путь к курсору1'), url('путь к курсору2'), ..., <ключевое слово>

Через запятую допускается указывать несколько значений url, в этом случае браузер попытается открыть первый файл с курсором и если это по каким-либо причинам не получится, перейдет к следующему файлу. Список обязательно заканчивается ключевым словом, например, auto или pointer, допустимые значения перечислены выше.

Псевдоэлементы

Псевдоэлементы позволяют задать стиль элементов не определенных в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста.

Селектор:Псевдоэлемент { Описание правил стиля }

Вначале следует имя селектора, затем пишется двоеточие, после которого идет имя псевдоэлемента. Каждый псевдоэлемент может применяться только к одному селектору, если требуется установить сразу несколько псевдоэлементов для одного селектора, правила стиля должны добавляться к ним по отдельности, как показано ниже.

.foo:first-letter { color: red }

.foo:first-line {font-style: italic}

Псевдоэлементы не могут применяться к внутренним стилям, только к таблице связанных или глобальных стилей.

Далее перечислены все псевдоэлементы, их описание и свойства.

:after

Применяется для вставки назначенного контента после элемента. Этот псевдоэлемент работает совместно со стилевым свойством content, которое определяет содержимое для вставки.

:before

По своему действию :before аналогичен псевдоэлементу :after, но вставляет контент до элемента.

:first-letter

Определяет стиль первого символа в тексте элемента, к которому добавляется. Это позволяет создавать в тексте буквицу и выступающий инициал.

Буквица представляет собой увеличенную первую букву, базовая линия которой ниже на одну или несколько строк базовой линии основного текста. Выступающий инициал — увеличенная прописная буква, базовая линия которой совпадает с базовой линией основного текста.

:first-line

Определяет стиль первой строки блочного текста. Длина этой строки зависит от многих факторов, таких как используемый шрифт, размер окна браузера, ширина блока, языка и т.д.

К псевдоэлементу :first-line могут применяться не все стилевые свойства. Допустимо использовать свойства, относящиеся к шрифту, изменению цвет текста и фона, а также: clear, line-height, letter-spacing, text-decoration, text-transform, vertical-align и word-spacing.

Наследование

Наследованием называется перенос правил форматирования для элементов, находящихся внутри других. Такие элементы являются дочерними, и они наследуют некоторые стилевые свойства своих родителей, внутри которых располагаются.

При этом следует понимать, что не все стилевые свойства наследуются. Так, border задает рамку вокруг таблицы в целом, но никак не вокруг ячеек. Аналогично не наследуется значение свойства background. Тогда почему цвет фона у ячеек будет темный, раз он не наследуется? Дело в том, что у свойства background в качестве значения по умолчанию выступает transparent, т.е. прозрачность. Таким образом цвет фона родительского элемента «проглядывает» сквозь дочерний элемент.

Чтобы определить, наследуется значение стилевого свойства или нет, требуется заглянуть в справочник по свойствам CSS и посмотреть там. Подключать свою интуицию в подобном случае бесполезно, может и подвести.

Наследование позволяет задавать значения некоторых свойств единожды, определяя их для родителей верхнего уровня. Допустим, требуется установить цвет и шрифт для основного текста. Достаточно воспользоваться селектором BODY, добавить для него желаемые свойства, и цвет текста внутри абзацев и других текстовых элементов поменяется автоматически.

Однако бывают варианты, когда требуется все-таки изменить цвет для отдельного контейнера. В этом случае придется переопределять нужные параметры явно непосредственно для нужного тега.