CSS-селекторы

(тут должен быть красивый подзаголовок, но его съели бобры)

Основные виды селекторов

- * – любые элементы.
- div – элементы с таким тегом.
- #id – элемент с данным id.
- .class – элементы с таким классом.
- [name="value"] – селекторы на атрибут (см. далее).
- :visited – «псевдоклассы», остальные разные условия на элемент (см. далее).

Селекторы можно комбинировать, записывая последовательно, без пробела:

- .c1.c2 – элементы одновременно с двумя классами c1 и c2
- a#id.c1.c2:visited – элемент a с данным id, классами c1 и c2,

и псевдоклассом visited

Основные виды селекторов

- * – любые элементы.
- div – элементы с таким тегом.
- #id – элемент с данным id.
- .class – элементы с таким классом.
- [name="value"] – селекторы на атрибут (см. далее).
- :visited – «псевдоклассы», остальные разные условия на элемент (см. далее).

1. *

```
* { margin: 0; padding: 0; }
```

- Этот CSS-селектор выделяет каждый элемент на странице.
- Многие разработчики используют его для того, чтобы скинуть у всех элементов значения margin и padding.
- На первый взгляд это удобно, но все-таки в рабочем коде так лучше не делать. Этот CSS-селектор слишком сильно грузит браузер.
- Также можно использовать для выделения дочерних элементов

```
#container * { border: 1px solid black; }
```

2. #X

```
#container {
        width: 960px;
        margin: auto;
}
```

- Знак решетки перед CSS-селектором X выделит нам элемент с id = X. Это очень просто, но будьте аккуратны при использовании id.
- id жестко привязывают стиль к элементу и не дает возможности повторного использования. Более предпочтительным будет использование классов, названий тэгов или даже псевдо-классов.

.X

```
.error { color: red; }
```

- Это CSS-селектор класса X. Разница между id и классом заключается в том, что одному классу может принадлежать несколько элементов на странице.
- Используйте классы, когда вы хотите применить стиль к нескольким однотипным элементам.
- При использовании id вам придется писать стиль для каждого отдельного элемента.

4. X Y

```
li a { text-decoration: none; }
```

- CSS-селектор дочерних элементов встречается чаще всего.
- Если вам надо выделить элементы определенного типа из множества дочерних элементов, используете этот селектор.
- Например, вам надо выделить все ссылки, которые находятся в элементе li. В этом случае используйте этот селектор.

Не следует делать CSS-селекторы вида X Y Z A B.error. Всегда спрашивайте себя, а обязательно ли для выделения данного элемента писать такой громоздкий CSS-селектор.

5. X

```
a { color: red; }
ul { margin-left: 0; }
```

- Что делать, если вы хотите, чтобы охватить все элементы данного типа на странице?
- Будьте проще, используйте CSS-селектор типа.
- Если вы должны выделить все неупорядоченные списки, используйте ul {} .

6. X:visited and X:link

```
a:link { color: red; }
a:visted { color: purple; }
```

- Мы используем псевдо-класс :link, для выделения всех ссылок, на которые еще не кликнули.
- Если же нам надо применить определенный стиль у уже посещенным ссылкам, то используем псевдо-класс :visited.

Отношения

В CSS3 предусмотрено четыре вида отношений между элементами.

- Самые известные вы наверняка знаете:
- div p – элементы p, являющиеся потомками div.
- div > p – только непосредственные потомки

Есть и два более редких:

- div ~ p – правые соседи: все p на том же уровне вложенности, которые идут после div.
- div + p – первый правый сосед: p на том же уровне вложенности, который идёт сразу после div(если есть).

7. X+Y

```
ul + p { color: red; }
```

- Выделяет последующий элемент.
- Он будет выбирать *только* элемент типа Y, который идет сразу после элемента X.
- В примере текст первого абзаца после каждого ul будет красного цвета.

8. X>Y

```
div#container > ul { border: 1px solid black; }
```

- Разница между стандартными X Y и X > Y состоит в том, что рассматриваемый CSS-селектор будет выбирать только непосредственные дочерние элементы.
- Например, рассмотрим следующий код.

```
<div id="container">
  <ul>
    <li> Элемент списка
      <ul>
        <li> Дочерний элемент</li>
      </ul>
    </li>
    <li> Элемент списка </li>
    <li> Элемент списка </li>
    <li> Элемент списка </li>
  </ul>
</div>
```

- CSS-селектор #container > ul выберет только ul-ы, которые являются непосредственными дочерними элементами div с id =container .
- Он не выберет, например, ul-ы, являющиеся дочерними элементами первых li .
- Поэтому можно получить выигрыш в производительности используя данный CSS-селектор.
- Это особенно рекомендуется при работе с jQuery или другими библиотеками, выбирающими элементы на основе правил CSS -селекторов.

9. X ~ Y

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

- Этот CSS-селектор очень похож на X + Y, однако, является менее строгим.
- При использовании ul + p будет выбрать только первый элемент, идущий за X.
- В данном случае будут выбраны все элементы p, идущие за ul.

Селекторы атрибутов

На атрибут целиком:

- [attr] – атрибут установлен,
- [attr="val"] – атрибут равен val.
- На начало атрибута:
- [attr^="val"] – атрибут начинается с val, например "value".
- [attr|="val"] – атрибут равен val *или* начинается с val-, например равен "val-1".

На содержание:

- [attr*="val"] – атрибут содержит подстроку val, например равен "myvalue".
- [attr~="val"] – атрибут содержит val как одно из значений через пробел.
- Например: [attr~="delete"] верно для "edit delete" и неверно для "undelete" или "no-delete".

На конец атрибута:

- [attr$="val"] – атрибут заканчивается на val, например равен "myval".a

10. X[title]

```
a[title] { color: green; }
```

- В CSS-селекторах также можно использовать атрибуты.
- Например в данном примере мы выделили все ссылки, имеющие атрибут титле.
- Остальные ссылки останутся не затронутыми.

11. X [href="Foo"]

```
a[href="http://donnu.ru"] { color: #ffde00; }
```

- Все ссылки, которые ссылаются на everstudent.ru будут золотыми. Все остальные ссылки останутся неизменными неизменным.

Обратите внимание, что на кавычки. Не забудьте так же делать в jQuery и других JavaScript библиотеках, в которых элементы выбираются по CSS-селекторам. По возможности, всегда используйте CSS3 CSS-селекторы.

- Хорошее правило, но слишком строгое. Что же делать, если ссылка ведет не непосредственно на everstudent.ru, а например на http://donnu.ru/portfolio ? В этих случаях мы можем использовать регулярные выражения.

12. X [HREF *= "donnu.ru"]

```
a[href*=“donnu"] { color: # 1f6053; }
```

- Вот то, что нам нужно. Звезда обозначает, что искомое значение должно появляться *где-нибудь* в атрибуте. Таким образом, CSS-селектор охватывает *donnu.ru, http://donnu.ru/portfolio* и т.д.
- Но что делать, если ссылка ведет на какой-то сторонних и не связанный ресурс, в адресе которого присутствует donnu? Тогда нужно использовать "^" или "&", для ссылки на начало и конец строки соответственно.

13. X[href^="http"]

```
a[href^="http"] {
  background: url(path/to/external/icon.png) no-repeat;
  padding-left: 10px;
}
```

- Вы никогда не задумывались, как некоторые веб-сайты могут отображать маленький значок рядом с внешними ссылками? Я уверен, что вы видели их прежде, они хорошо запоминаются.
- "^" - наиболее часто используемый в регулярных выражениях символ. Он используется для обозначения начала строки. Если мы хотим охватить все тэги, у которых href начинается с http, нам надо использовать CSS-селектор, приведенных выше.
- Если мы хотим задать стиль только для ссылок, ведущих на фотографию? Нужно искать *конец* строки

Обратите внимание, что мы не ищем "http://". Это не правильно, поскольку не учитываются адреса, начинающиеся с https://

14. X [href$=". JPG"]

```
a[href$=".jpg"] { color: red; }
```

- Опять же, мы используем символ регулярного выражения "$" для обозначения конца строки.
- В данном мы ищем ссылки, которые ссылаются на jpg-файлы, или url-ы, в конце у которых стоит ".jpg".

15. X[data-*="foo"]

- Как же нам теперь написать CSS-селектор, который бы выделял ссылки на все виды изображений? Например, мы могли бы написать так:

```
a[href$=".JPG"], a[href$=". JPEG"],
a[href$=".PNG"], a[href$=".GIF"] {
color: red;
}
```

- Другим возможным решением является применение пользовательских атрибутов. Почему бы нам не добавить наш собственный атрибут data-filetype в каждую ссылку?

```
<a href="path/to/image.jpg" data-filetype="image"> Ссылка</a>
```

- Теперь мы можем выделить такие ссылки при помощи данного CSS-селектора:

```
a[data-filetype="image"] { color: red; }
```

16. X[foo~="bar"]

```
a[data-info~="external"] { color: red; }
a[data-info~="image"] { border: 1px solid black; }
```

- А вот кое-что особенное. Не все знают об этом CSS-селекторе. Тильда (~) позволяет выделить определенный атрибут из списка атрибутов, разделенных запятой.
- Например, мы можем задать наш собственный атрибут data-info, в котором указывать несколько ключевых слов через пробел. Так, мы можем указать, что ссылка является внешней и что она ссылается на изображение

HTML:

```
"<a href="path/to/image.jpg" data-info="external image"> Click</a>
```

CSS:

```
/ * Выбираем ссылки с атрибутом data-info, содержащий значение "external" * /
a[data-info~="external"] { color: red; }
/ * И которые содержат значения "image" * /
a[data-info~="image"] { border: 1px solid black; }
```

16. X[foo~="bar"]

```
a[data-info~="external"] { color: red; }
a[data-info~="image"] { border: 1px solid black; }
```

- А вот кое-что особенное. Не все знают об этом CSS-селекторе. Тильда (~) позволяет выделить определенный атрибут из списка атрибутов, разделенных запятой.
- Например, мы можем задать наш собственный атрибут data-info, в котором указывать несколько ключевых слов через пробел. Так, мы можем указать, что ссылка является внешней и что она ссылается на изображение

HTML:

```
"<a href="path/to/image.jpg" data-info="external image"> Click</a>
```

CSS:

```
/ * Выбираем ссылки с атрибутом data-info, содержащий значение "external" * /
a[data-info~="external"] { color: red; }
/ * И которые содержат значения "image" * /
a[data-info~="image"] { border: 1px solid black; }
```

Другие псевдоклассы

- :not(селектор) – все, кроме подходящих под селектор.
- :focus – в фокусе.
- :hover – под мышью.
- :empty – без детей (даже без текстовых).
- :checked, :disabled, :enabled – состояния INPUT.
- :target – этот фильтр сработает для элемента, ID которого совпадает с анкором #... текущего URL.

Например, если на странице есть элемент с id="intro", то правило :target { color: red }подсветит его в том случае, если текущий URL имеет вид http://...#intro.

17. X:checked

```
input[type=radio]:checked { border:1px solid black; }
```

- Этот псевдокласс выделяет только элементы пользовательского интерфейса, такие как переключатель или флажок. Причем те, которые отмечены/выбраны. Очень просто.

Псевдоэлементы ::before, ::after

- «Псевдоэлементы» – различные вспомогательные элементы, которые браузер записывает или может записать в документ.
- При помощи *псевдоэлементов* ::before и ::after можно добавлять содержимое в начало и конец элемента:

```
<style>
  li::before {
    content: " [[ ";
  }

  li::after {
    content: " ]] ";
  }
</style>

Обратите внимание: содержимое добавляется <b>внутрь</b> LI.

<ul>
  <li>Первый элемент</li>
  <li>Второй элемент</li>
</ul>
```

Обратите внимание: содержимое добавляется **внутрь** LI.

- [[Первый элемент]]
- [[Второй элемент]]

18. X:after

- Псевдоклассы :before и :after очень крутые. Создается впечатление, что каждый день появляются новые способы их эффективного использования. Они просто генерируют контент вокруг выбранного элемента.

```
.clearfix:after {
    content: "";
    display: block;
    clear: both;
    visibility: hidden;
    font-size: 0;
    height: 0;
    }
```

Согласно спецификации CSS3, вы должны использовать два двоеточия (::). Однако, можно использовать и одно двоеточие для совместимости.

19. X:hover

```
div:hover { background: #e3e3e3; }
```

- Хотите применить стиль к элементу, когда наводите на него мышкой? Тогда этот CSS-селектор для вас.

Имейте в виду, что старые версии Internet Explorer применяют :hover только к ссылкам.

- Этот CSS-селектор часто используют для того, чтобы поставить border-bottom на ссылки, когда на них наводят мышкой.

```
a:hover {  border-bottom: 1px solid black; }
```

border-bottom: 1px solid black; выглядит лучше, чем text-decoration: underline;

20. X:not(selector)

```
div:not(#container) { color: blue; }
```

- Псевдокласс отрицания бывает очень полезным. Скажем, я хочу выбрать все div, за исключением того, который имеет id = container . Приведенный выше код справиться с этим!
- Или, если бы я хотел выбрать все элементы, за исключением p.

```
*:not(p) { color: green; }
```

21. X::псевдо элемент

```
p::first-line {
   font-weight: bold;
   font-size: 1.2em;
}
```

- Мы можем использовать псевдо элементы, чтобы оформлять фрагменты элементов, такие как первая строка, или первая буква. Имейте в виду, что они должны быть применены к элементам уровня блока для того, чтобы вступили в силу.

Псевдо-элемент задается двумя двоеточиями: ::

```
p::first-letter {
    float: left;
    font-size: 2em;
    font-weight: bold;
    font-family: cursive;
    padding-right:2px;
}
```

Фильтр по месту среди соседей

При выборе элемента можно указать его место среди соседей.

- Список псевдоклассов для этого:
- :first-child – первый потомок своего родителя.
- :last-child – последний потомок своего родителя.
- :only-child – единственный потомок своего родителя, соседних элементов нет.
- :nth-child(a) – потомок номер a своего родителя, например :nth-child(2) – второй потомок. Нумерация начинается с 1.
- :nth-child(an+b) – расширение предыдущего селектора через указание номера потомка формулой, где a,b – константы, а под n подразумевается любое целое число.

Этот псевдокласс будет фильтровать все элементы, которые попадают под формулу при каком-либо n.

Например:

- :nth-child(2n) даст элементы номер 2, 4, 6…, то есть чётные.
- :nth-child(2n+1) даст элементы номер 1, 3…, то есть нечётные.
- :nth-child(3n+2) даст элементы номер 2, 5, 8 и так далее.

Фильтр по месту среди соседей с тем же тегом

Есть аналогичные псевдоклассы, которые учитывают не всех соседей, а только с тем же тегом:

- :first-of-type
- :last-of-type
- :only-of-type
- :nth-of-type
- :nth-last-of-type

Они имеют в точности тот же смысл, что и обычные :first-child, :last-child и так далее, но во время подсчёта игнорируют элементы с другими тегами, чем тот, к которому применяется фильтр.

22. X:nth-child(n)

```
li:nth-child(3) { color: red; }
```

- Раньше мы не могли выделить, например, третий дочерний элемент? nth-child решает это!
- Обратите внимание, что nth-child принимает целое число в качестве параметра, однако отсчет ведется не с 0. Если вы хотите выбрать второй пункт списка, используйте li:nth-child(2) .
- Мы даже можем выбрать каждый четвертый элемент списка, просто написав {0}li:nth-child(4n){/0}.

23. X:nth-last-child(n)

```
li:nth-last-child(2) { color: red; }
```

- Что делать, если у вас огромный список элементов в ul , а нем нужно выбрать третий элемент с конца? Вместо того, чтобы писать li:nth-child(397), можно использовать nth-last-child.
- Этот метод почти идентичен приведенному выше, однако отсчет ведется с конца.

24. X:nth-of-type(n)

```
ul:nth-of-type(3) { border: 1px solid black; }
```

- Бывает, что надо выбрать не дочерний элемент, а элемент определенного типа.
- Представьте себе, что на странице пять неупорядоченных списков. . Если вы хотите применить стиль только к третьему ul, не имеющему уникального id, нужно использовать nth-of-type.

25. X:nth-last-of-type(n)

```
ul:nth-last-of-type(3) { border: 1px solid black; }
```

- Мы можем также использовать nth-last-of-type, отсчитывая элементы с конца.

26. X:first-child

```
ul li:first-child { border-top: none; }
```

- Этот псевдокласс выбирает первый дочерний элемент. Часто используется чтобы убрать border в первом и последнем элементе списка.

28. X:only-child

```
div p:only-child { color: red; }
```

- Вы не часто встретите этот псевдокласс, тем не менее он существует.
- Он позволяет вам выбрать элементы, которые являются единственными дочерними. Например, применим приведенный выше стиль к этому html-коду:

```
<div>
    <p> Один параграф.</p>
</div>
<div>
    <p> Два параграфа </p>
    <p> Два параграфа </p>
</div>
```

- Будет выбран p только первого div`a, потому что он единственный дочерний элемент.

29. X:only-of-type

```
li:only-of-type { font-weight: bold; }
```

- Очень интересный псевдокласс. Он затрагивает элементы, не имеющие соседей в пределах родительского элемента. В качестве примера выберем ul только с одним элементом в списке.
- Единственное решение заключается в использовании only-of-type .

```
ul > li:only-of-type { font-weight: bold; }
```

30. X:first-of-type

- first-of-type выбирает первый элемент заданного типа. Чтобы лучше понять, приведем

```
<div>
    <p> Параграф </p>
    <ul>
        <li> Пункт 1 </li>
        <li> Пункт 2 </li>
    </ul>
    <ul>
        <li> Пункт 3 </li>
        <li> Пункт 4 </li>
    </ul>
<div>
```

А теперь попытайтесь понять как выделить пункт 2.

```
ul:first-of-type > li:nth-child(2) { font-weight: bold; }
```

```
p + ul li:last-child { font-weight: bold; }
```

```
ul:first-of-type li:nth-last-child(1) { font-weight: bold; }
```