Все селекторы регистрозависимы

## Селекторы CSS

**Универсальный селектор \***

Синтаксис: \* ns|\* \*|\*

Пример: \* будет соответствовать всем элементам на странице.

**Селектор по атрибуту** нужно указывать в [ ].

Пример 1: [autoplay] выберет все элементы, у которых есть атрибут autoplay (независимо от его значения).

Пример 2: a[href$=".jpg"] выберет все ссылки, у которых адрес заканчивается на ".jpg".

Пример 3: a[href^="https"] выберет все ссылки, у которых адрес начинается на "https".

**Селектор по id:** #idname

**Селектор по атрибуту class:** .classname

.class1.class3.class2 - поиск элемента, относящегося к нескольким классам.

**Пробел = комбинатор потомков**

Селектор "**#post2 .title**": здесь "#" означает, что надо искать элемент с id "post2", а "." - что искать надо класс со значением "title". Элемент ".title" называется потомком (англ. **descendant**) элемента "#post2". Потомок может находиться на любом уровне вложенности.

Символ пробела " " выбирает элементы, которые находятся внутри указанного элемента, вне зависимости от уровня вложенности.

**Комбинатор дочерних элементов >**

Селектор **"#post2 > div.title"**: нужно взять элемент с тегом и классом "div.title", который находится строго на один уровень иерархии ниже чем элемент "#post2". Элемент "#post2" является родителем (англ. **parent**), "div.title" называется дочерним элементом (англ. **child**) для элемента "#post2". Если символа **">"** нет, то будут выполнен поиск всех элементов "div.title" на любом уровне ниже первого элемента. Пробелы вокруг ">" не могут быть опущены.

**Использование порядкового номера дочернего элемента**

**"#posts > .item:nth-child(2) > .title"**

Псевдо-класс **:nth-child(2)** - позволяет найти второй по порядку элемент среди дочерних элементов для "#posts**"**. Затем с помощью "> .title" мы указываем, что нам нужен элемент ".title", родителем которого является найденный ранее элемент ".item".

**Псевдокласс :** позволяет выбрать элементы, основываясь на информации, которой нет в дереве элементов.

Пример 1: a:visited соответствует всем элементам <a> которые имеют статус "посещённые".

Пример 2: div:hover соответствует элементу, над которым проходит указатель мыши.

Пример 3: input:focus соответствует полю ввода, которое получило фокус.

**Псевдоэлемент ::**

Знак :: позволяет выбрать вещи, которых нет в HTML.

Пример: p::first-line соответствует первой линии абзаца <p>.

**Комбинатор всех соседних элементов ~**

Выбирает элементы, которые находятся на этом же уровне вложенности, после указанного элемента, с тем же родителем.

Синтаксис: A ~ B

Пример: p ~ span выберет все элементы <span>, которые находятся после элемента <p> внутри одного родителя.

**Комбинатор следующего соседнего элемента +**

Выбирает элемент, который находится непосредственно после указанного элемента, если у них общий родитель.

Синтаксис: A + B

Пример: селектор ul + li выберет любой <li> элемент, который находится непосредственно после элемента <ul>.

**!** Не существует селекторов, которые бы позволили выбрать родителя (содержащий контейнер), или соседа родителя, или потомков соседа родителя.

## XPath селекторы

(XML Path Language) это язык запросов, который использует древовидную структуру документа.

1. XPath запрос всегда **начинается с символа / или //**

Символ / аналогичен символу > в CSS-селекторе, а символ // - пробелу. Их смысл:

el1/el2 - выбирает элементы el2, являющиеся прямыми потомками el1;

el1//el2 - выбирает элементы el2, являющиеся потомками el1 любой степени вложенности.

Разница состоит в том, что в XPath, когда мы начинаем запрос с символа /, мы должны указать элемент, являющийся корнем нашего документа. Корнем всегда будет элемент с тегом <html>. Пример: /html/body/header

Мы можем начинать запрос и с символа //. Это будет означать, что мы хотим найти всех потомков корневого элемента без указания корневого элемента. В этом случае, для поиска того же хедера, мы можем выполнить запрос //header, так как других заголовков у нас нет.

2. Символ **[ ]** - это команда фильтрации

Если по запросу найдено несколько элементов, то будет произведена фильтрация по правилу, указанному в скобках.

Правил фильтрации очень много:

* по любому **атрибуту**, будь то id, class, title (или любой другой). Например, мы хотим найти картинку с летящим котом, для этого можно выполнить запрос //img[@id='bullet']
* по порядковому **номеру**. Допустим, мы хотим выбрать вторую по порядку карточку с котом. Для этого найдем элемент с классом "row" и возьмем его второго потомка: //div[@class="row"]/div[2]
* **по полному совпадению текста**. Да, XPath - это единственный способ найти элемент по внутреннему тексту. Если мы хотим найти блок текста с котом-Лениным, можно воспользоваться XPath селектором //p[text()="Lenin cat"]. Такой селектор вернет элемент, только если текст полностью совпадёт. Здесь важно сказать, что не всегда поиск по тексту - это хорошая практика, особенно в случае мультиязычных сайтов.
* **по частичному совпадению** текста или атрибута. Для этого нужна функция contains. Запрос //p[contains(text(), "cat")] вернет нам все абзацы текста, которые содержат слово cat. Точно так же можно искать по частичному совпадению других атрибутов, это удобно, если у элемента несколько классов. Посмотрите на код навбара сайта с котами. Его можно найти селектором //div[contains(@class, "navbar")]
* в фильтрации еще можно использовать **булевы операции (and, or, not)** и некоторые простые **арифметические выражения** (но вообще не стоит, наверное). Допустим, мы хотим найти картинку обязательно с data-type "animal" и именем "bullet-cat", для этого подойдет запрос: //img[@name='bullet-cat' and @data-type='animal']

3. Символ **\*** - команда выбора всех элементов

Например можем найти текст в заголовке запросом //div/\*[@class="jumbotron-heading"]. Это может быть удобно, когда мы не знаем точно тег элемента, который ищем.

Что важно знать про XPath, чтобы пользоваться им безболезненно:

* Не используйте селекторы вида //div[1]/div[2]/div[3] без крайней нужды: по такому селектору невозможно с первого раза понять, что за элемент вы ищете. А когда структура страницы хоть немного изменится, то ваш селектор с большой вероятностью перестанет работать;
* Если есть возможность использовать CSS-селекторы: сlass, id или name - лучше использовать их вместо поиска по XPath;
* Можно искать по полному или частичному совпадению текста или любого атрибута;
* Можно использовать булевы операции и простую арифметику;
* Можно удобно перемещаться по структуре документа (переходить к потомкам и к родителям);
* Подойдет, когда у сайта всё плохо с атрибутами и нет возможности достучаться до разработчиков;
* Есть мнение, что поиск по XPath в среднем медленнее, чем по css. Но достоверно это неизвестно;
* Не стоит использовать разные расширения для браузеров по поиску XPath: они подбирают нечитабельные и переусложненные селекторы. Лучше потратить немного времени и разобраться в синтаксисе самостоятельно, тем более, что он не очень сложный.