**Поиск элементов с помощью CSS-селекторов**

Ниже приведены части элементов HTML-страницы, по которым можно найти элемент:

* id
* tag
* значение атрибута
* name
* class

Давайте откроем страницу <http://suninjuly.github.io/cats.html> и попробуем найти элемент, который содержит картинку с Котом-пулей (Bullet cat). Ниже приведён упрощенный кусок html-кода страницы:

<div class="col-sm-4">

<div class="card mb-4 box-shadow">

  <img id="bullet" name="bullet-cat" data-type="animal" class="card-img-top" src="images/bullet\_cat.jpg">

</div>

</div>

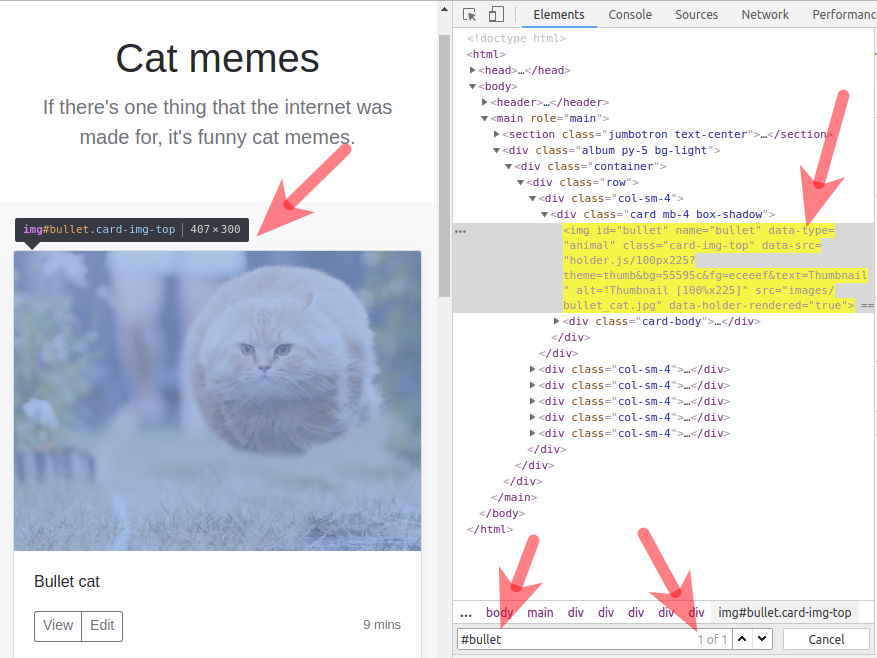
Для начала мы попробуем искать элементы вручную с помощью консоли браузера, а в следующем уроке научимся писать код, который выполняет ту же задачу поиска.

**Поиск по id**

Какое везение! У элемента с нашей картинкой есть атрибут id="bullet", а значит, мы однозначно можем найти её с помощью селектора **#bullet**(знак **#** означает, что мы ищем по **id** со значением **bullet**).

*Можно проверить правильность подобранного селектора непосредственно в браузере в консоли разработчика. Откройте консоль разработчика и перейдите в ней на вкладку Elements. Затем нажмите ctrl+F и в открывшейся внизу поисковой строке введите селектор. Если селектор написан правильно, то вы увидите подсвеченный элемент на веб-странице, а также элемент будет подсвечен жёлтым цветом в html-коде. Еще в поисковой строке вы увидите количество найденных элементов. Желательно писать точные селекторы, которые позволяют найти ровно один элемент. В написании таких селекторов мы потренируемся в одной из следующих задач.*

*Еще один способ открыть консоль разработчика в браузере: нажать правой кнопкой мыши на любой элемент страницы и выбрать пункт меню "Посмотреть код" (англ. "Inspect") в контекстном меню. При этом на вкладке Elements сразу будет подсвечен кусок HTML-кода, описывающий данный элемент.*



**Поиск по tag**

Чтобы найти элемент по тегу, просто напишите название тега в поисковой строке, как мы делали это при поиске по id (только без знака **#**), например, **h1**. Поиск по **h1** найдёт для нас элемент с названием страницы. Поиск по тегам не очень удобен, т.к. разработчики используют небольшое количество тегов для разметки страниц, и скорее всего, одному тегу будет соответствовать множество элементов.

**Поиск по значению атрибута**

Можно найти элемент, указав название атрибута и его значение. Например, можно переписать поиск по id в следующем виде **[id="bullet"]**вместо **#bullet**.

Лучше использовать вариант с квадратными скобками при поиске значения атрибута для тех атрибутов, у которых нет собственных коротких команд поиска. Например, давайте найдем элемент h1 по значению его атрибута value: **[value="Cat memes"]**.

**Поиск по name**

Этот вариант поиска является разновидностью поиска по значению атрибута и записывается так же: **[name="bullet-cat"]**. Мы выделяем этот вариант потому что он довольно часто используется, а также выделяется как отдельный вид поиска элементов в Selenium WebDriver.

**Поиск по class**

Поиск по классу можно записать в виде **[class="jumbotron-heading"]**, так как class тоже является атрибутом элемента. Но раз уж классы используются практически в каждой странице при задании стилей страниц, то для них также имеется свой короткий вариант поиска: **.jumbotron-heading**. То есть мы пишем значение класса и предваряем его точкой.

Давайте рассмотрим важную разницу между двумя способами поиска по классу. Допустим, у элемента **article** задано больше одного класса, как на странице <http://suninjuly.github.io/cats.html>:

<article id="moto" class="lead text-muted" title="one-thing" name="moto">If there's one thing that the internet was made for, it's funny cat memes.</article>

Вариант **[class="lead"]** не найдет нам этот элемент, так как он ищет по точному совпадению. Чтобы найти элемент, нам нужно будет написать **[class="lead text-muted"]**, порядок классов при этом важен. **[class="text-muted lead"]** — уже не найдет искомый элемент.

Вариант **.lead** при этом позволит найти данный элемент, так как он ищет простое вхождение класса в элемент. Для уточнения селектора можно задать также оба класса, для этого нужно добавить второй класс к строке поиска без пробела и предварить его точкой: **.lead.text-muted**. Порядок классов в отличие от первого способа здесь не важен — **.text-muted.lead** так же найдет нужный элемент. Рекомендуем пользоваться вторым способом поиска классов, так как он является более гибким.

Еще одно важное замечание. Поиск по классу чувствителен к регистру, то есть **.Lead** уже не найдет нужный элемент.

В консоли браузера вы также можете искать по простому совпадению текста в HTML, например, запрос **lead** подсветит текст **lead**. Однако, не стоит пользоваться таким поиском для выбора элементов, так как он слишком общий и не может использоваться в качестве селектора.

Мы рассмотрели разные варианты написания пути к элементу на странице, используя синтаксис CSS, т.е. научились писать CSS-селекторы. Слово "селектор" является буквальным переводом от английского слова selector. Selector в свою очередь происходит от глагола select, что переводится как "выбирать".

Далее в этом уроке мы научимся искать элементы, комбинируя способы составления селекторов, рассмотренные в данном шаге.

## Поиск элементов с помощью составных CSS-селекторов

Теперь предположим, что не можем найти элемент на странице, используя простой селектор, так как такой селектор находит сразу несколько элементов. Ниже мы привели часть кода простой HTML-страницы, описывающей блог. Саму страницу вы можете посмотреть по https://suninjuly.github.io/blog\_example.html.

Вопрос: как нам найти селектор для подписи у второй картинки? Вот здесь нам поможет иерархическая структура страницы и возможность комбинировать CSS-селекторы. CSS-селекторы позволяют использовать одновременно любые селекторы, рассмотренные ранее, а также имеют некоторые дополнительные возможности для уточнения поиска.

<div id="posts" class="post-list">

  <div id="post1" class="item">

    <div class="title">Как я провел лето</div>

    <img src="./images/summer.png">

  </div>

  <div id="post2" class="item">

    <div class="title second">Ходили купаться</div>

    <img src="./images/bad\_dog.jpg">

  </div>

  <div id="post3" class="item">

    <div class="title">С друзьями</div>

    <img src="./images/friends.jpg">

  </div>

</div>

**Использование потомков**

Попробуем найти элемент с текстом "Ходили купаться". Для решения этой задачи мы можем взять элемент, стоящий выше в иерархии нужного нам элемента, и написать следующий селектор:

#post2 .title

Здесь символ **#** означает, что надо искать элемент с id post2, пробел - что также нужно найти элемент-потомок, а **.**, что элемент-потомок должен иметь класс со значением title.

Элемент .title называется **потомком** (англ. **descendant**) элемента #post2. Потомок может находиться на любом уровне вложенности, все элементы с селектором .title также являются и потомками элемента #posts, хотя и расположены от него на два уровня ниже. #posts .title найдет все 3 элемента с классом title.

!Внимание. Символ пробела " " является значащим в CSS-селекторах. Это важный символ, который разделяет описание предка и потомка. Если бы мы записали селектор #post2.title без пробела, то в данном примере не было найдено ни одного элемента. Такая запись означала бы, что мы хотим найти элемент, который одновременно содержит id "post2" и класс "title". Таким образом #post2 .title и #post2.title — это разные селекторы**.**

**Использование дочерних элементов**

Другой способ найти этот элемент:

#post2 > div.title

Здесь мы указали еще тег элемента divи уточнили, что нужно взять элемент с тегом и классом: div.title, который находится строго на один уровень иерархии ниже чем элемент #post2. Для этого используется символ >.

Элемент #post2 в этом случае называется **родителем** (англ. **parent**) для элементаdiv.title, а элемент div.title называется **дочерним элементом** (англ. **child**) для элемента #post2. Если символа > нет, то будет выполнен поиск всех элементов div.title на любом уровне ниже первого элемента.

!Внимание. В данном случае символы пробела вокруг символа ">" не несут важного значения в отличие от предыдущего примера, и могут быть опущены. Запись #post2>div.titleаналогична записи #post2 > div.title.

**Использование порядкового номера дочернего элемента**

Еще один способ найти этот элемент:

#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 конструкции > .title мы указываем, что нам нужен элемент .title, родителем которого является найденный ранее элемент .item.

**Использование нескольких классов**

Также мы можем использовать сразу несколько классов элемента, чтобы его найти. Для этого классы записываются подряд через точку: .title.second

Мы рассмотрели базовые селекторы, которых будет достаточно для написания простых UI-тестов. Если вы захотите разобраться подробнее в css-селекторах, то мы рекомендуем вам посмотреть следующие статьи:

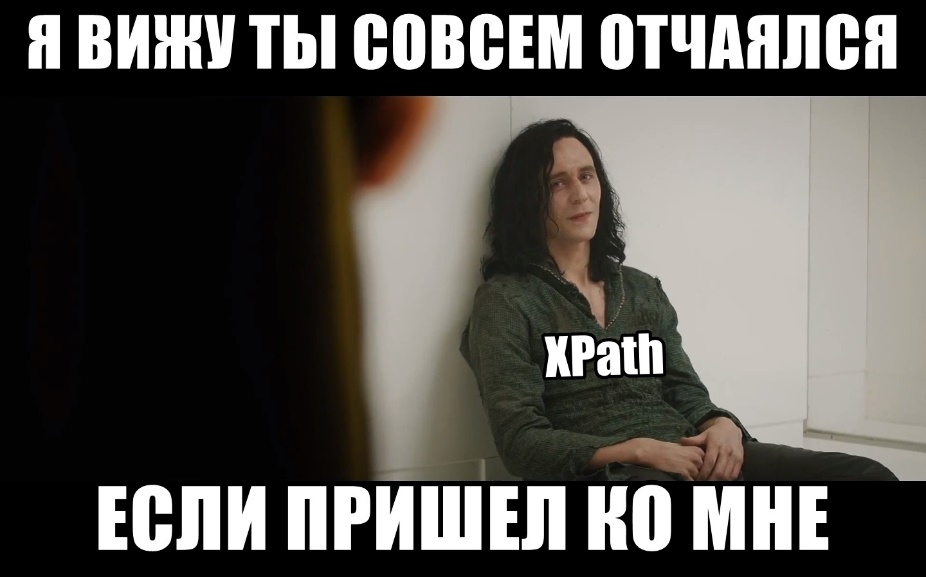
<https://learn.javascript.ru/css-selectors>

<https://www.w3schools.com/cssref/css_selectors.asp>

<https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Selectors>

## Поиск элементов с помощью XPath

В работе с веб-страницами не всегда получается найти селектор, однозначно описывающий путь к нужному элементу. В такой ситуации для тестировщика лучшим решением проблемы будет пойти к фронтенд-разработчику проекта и договориться о специальном атрибуте, который будет использоваться в автотестах. Таким образом можно повысить тестируемость приложения. Увы, проекты бывают разные, и не всегда это возможно. И когда другого выхода больше нет, а автоматизировать как-то надо, можно обратиться к помощи языка запросов **XPath**.



На тему XPath мнения расходятся, но, как бы то ни было, это мощный и гибкий инструмент, который позволяет писать сложные запросы для поиска элементов.

Прежде всего, XPath (XML Path Language) это язык запросов, который использует древовидную структуру документа. Проверять XPath-запросы можно точно так же как и CSS-селекторы — в консоли разработчика. Откройте консоль на странице с котиками <http://suninjuly.github.io/cats.html>, и давайте на её примере разберемся в основах синтаксиса. Попробуйте вбить каждый из запросов-примеров в строку поиска, чтобы увидеть, что именно находит поисковый запрос.

### 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.

Важно! Такой поиск может быть неоднозначным. Например, запрос //div вернет вам все элементы с тегом <div>. Избегайте неоднозначных ситуаций, они плохо влияют на здоровье ваших автотестов.

### 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 совпадет. 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.

### 4. Поиск по классу в XPath регистрозависим

Также как и в случае поиска по CSS-селектором будьте внимательными к регистру при поиске по классам:

**//div/\*[@class="Jumbotron-heading"]** не найдет элемент на нашей странице.

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.

В курсе мы не будем работать с XPath-селекторами, и в основном будем использовать CSS. В случае необходимости можно познакомиться с XPath подробнее по следующим ссылкам:

<https://www.w3schools.com/xml/xpath_syntax.asp>

<https://msdn.microsoft.com/ru-ru/library/ms256086(v=vs.120).aspx>

<https://msiter.ru/tutorials/xpath/syntax>

<https://habr.com/post/114772/>

<https://testerslittlehelper.wordpress.com/2016/07/10/real-xpath/>