**Что представляет собой веб-страница?**

В данном уроке мы подробнее разберёмся с самой важной для нас возможностью Selenium WebDriver — умению взаимодействовать с элементами веб-страницы. Любая страница в интернете представляет собой **html**-файл, в котором с помощью языка разметки HTML описана её структура.

Еще практически все сайты используют язык **JavaScript**, который позволяет сделать веб-страницу интерактивной, т.е. реагировать на действия пользователя, запрашивать у пользователя данные и возвращать их. WebDriver тоже позволяет выполнять javascript-команды в браузере, но это требуется не очень часто, поэтому мы рассмотрим эту возможность в дальнейших уроках.

Для полноты картины нужно упомянуть еще **CSS (**Cascading Style Sheets — *каскадные таблицы стилей*), который используется для вёрстки (красивого оформления страниц). Наверное, вы уже сталкивались с тем, что поломанная вёрстка страниц может приводить к очень неприятным багам на сайте. Благодаря WebDriver мы также можем поймать какую-то неожиданную проблему в оформлении — например, когда нужная пользователю кнопка оказывается перекрыта второстепенным элементом.

Сейчас нас будет интересовать в первую очередь структура страницы, т.е. ее описание на языке HTML. Умея описывать путь к элементу на странице, мы можем найти такой элемент и выполнить с ним необходимые действия, например, отправить текст в текстовое поле или нажать на правильную кнопку.

Далее мы рассмотрим несколько способов поиска элементов внутри страницы:

1. Поиск с помощью CSS-селекторов, когда путь к элементу описывается через синтаксис CSS. Селектор — это описание пути к элементу на странице.
2. Поиск с помощью указания значений тегов или атрибутов элементов: ID, class, и т.д.
3. Поиск с помощью языка запросов XPath.

Поиск с помощью CSS-селекторов, с нашей точки зрения, является наиболее удобным способом, т.к. он покрывает практически все возможные ситуации, и CSS-селекторы выглядят более читабельными. В реальности в разных случаях может понадобиться использовать и другие методы поиска.

## Структура веб-страницы

Как мы говорили ранее, язык CSS используется для описания внешнего вида страницы, которая написана с помощью языка разметки HTML. Если мы хотим сказать браузеру, что какую-то кнопку мы хотим сделать зелёной, мы сначала должны задать правило, которое позволяет выделить эту кнопку среди других элементов. Например, это можно сделать с помощью CSS-селектора. Селектор однозначно выделит кнопку среди других элементов html-страницы, а стиль color задаст кнопке цвет green. Синтаксис такого выражения в CSS-файле выглядит так:

button {

color: green;

}

button - это селектор, выражение color: green задает цвет элементу button.

Знание правил написания CSS-селекторов понадобится и нам, чтобы искать элементы на HTML-странице и взаимодействовать с ними.

Также нам понадобится знание трёх особенностей языка HTML:

1. Страница на языке HTML состоит из элементов, начало и конец элемента задаются с помощью специальных пометок – **тегов** (**tag**);
2. У тегов есть **атрибуты**, которые определяют свойства элементов;
3. Страница на языке HTML имеет **иерархическую** структуру.

### 1. **Tag**–элемент или тег

Начало и конец элементов обрамляются специальными словами – **тегами (tag)**. Задача тегов – обозначить, какой именно тип информации на странице они представляют (картинка, текст, блок, ссылка и так далее). Например:

<p>Здесь написан текст.</p>

В первом примере элемент обозначен открывающим (<p>) и закрывающим (</p>) тегами, между тегами помещён текст, который увидит пользователь. Некоторым тегам закрывающий тег не нужен, достаточно только открывающего. Например, тег:

<br>

Это тег для переноса строки, внутри него никакого текста или вложенных элементов быть не может.

### 2. **Attribute**–**атрибуты или свойства элемента**

Элементам можно задавать различные стандартные свойства, а также создавать собственные свойства или атрибуты. Атрибуты обычно имеют вид: имя="значение". Иногда знак "=" и значение могут быть опущены или значение может быть пустой строкой: имя="".

В примере ниже помимо тегов и текста добавлен еще атрибут**href**, который задаёт ссылку для тега <a>, на которую будет вести текст внутри тега.

<a href="/about">Посмотреть подробнее</a>

**Атрибут class**

В третьем примере появляется еще один важный атрибут – **класс (class)**. Элементу задано два класса: **simple-text** и **main**. Классы чаще всего используют для задания правил вёрстки с помощью CSS:

<div class="simple-text main">Здесь написан текст.</div>

**Атрибут name**

Свойство **name**, например, используется для задания якоря (закладки) в html-странице. Этот атрибут тоже можно использовать в качестве селектора для поиска элемента:

<a name="top"></a>

<p>Длинный текст</p>

...

<p><a href="#top">Наверх</a></p>

**Атрибут id**

В этом примере появился еще атрибут **id**– уникальный указатель на элемент. Значение id в общем случае не должно повторяться в пределах страницы:

<button id="save\_button">Сохранить</button>

!Важно. В современных JavaScript-фреймворках id-атрибуты чаще всего генерируются динамически самим фреймворком, поэтому они изменяются каждый раз при перезагрузке страницы и совершенно нечитабельны, например: вы увидите что-то вроде id="u\_ps\_0\_0\_n" или id="avadspffd". В таких случаях вам придется пользоваться другими селекторами или использовать собственные data-атрибуты, о которых мы расскажем далее. Названия классов также могут генерироваться автоматически. Поэтому предлагаем вам простое правило: если увидите нечеловекочитаемое значение атрибута или если значение атрибута меняется при перезагрузке страницы, то не используйте его. В наших примерах мы используем статические страницы, поэтому вы смело можете использовать описанные здесь CSS-селекторы.

### 3. **Иерархия** в HTML-документах

HTML-документ часто сравнивают с моделью семейного древа, в котором есть родители, дети, братья, предки и потомки. Такое сравнение допустимо благодаря возможности вкладывать одни элементы в другие. Вложенные (дочерние) элементы могут, в свою очередь, содержать свои дочерние элементы, следовательно, глубина вложения элементов может быть любой.

Давайте разберёмся с терминологией данного древа:

* потомок элемента X – элемент любой степени вложенности внутри элемента X;
* ребёнок или дочерний элемент — прямой потомок (т.е. элемент на первом уровне вложенности);
* предок элемента Y – любой элемент X, который включает в себя элемент Y;
* родитель — это прямой предок (т.е. элемент, который расположен выше строго на 1 уровень);
* братский или соседний элемент – элемент X, который расположен на одном уровне иерархии с элементом Y. Элементы X и Y имеют одного родителя.

Теперь попробуем закрепить терминологию на примере. Возьмём такой пример HTML кода:

<div class="news">

<div class="title">

<p class="title\_text">Selenium 4alpha is out now</p>

</div>

<div class="text\_block">

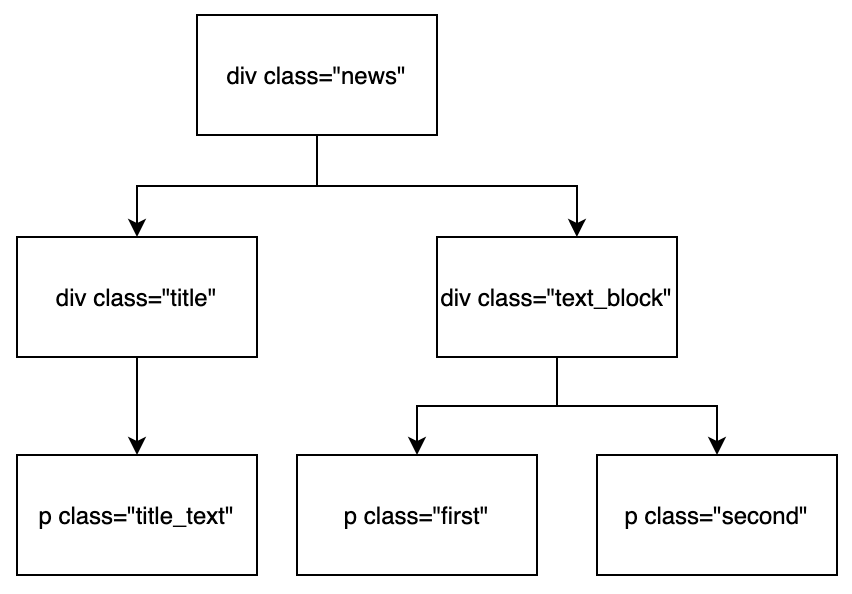
<p class="first">After 3 years from it’s a major release, now Selenium has put out its first alpha version of Selenium 4 on Apr 24, 2019.</p>

<p class="second">If you want you can make your hands dirty with Selenium 4 and it’s feature by downloading the dependencies from Maven Repository.</p>

</div>

</div>

Теперь нарисуем блок-схему этого HTML кода, где каждый блок будет соответствовать одному HTML элементу.



* Какие элементы являются потомками элемента, имеющего class="news"? Это все остальные элементы в этом html-коде: элемент с class="title", элемент с class="title\_text" и другие.
* Какие элементы можно назвать дочерними (child) элементами для элемента с class="news"? Только элементы с class= "title" и class="text\_block".
* Какие элементы являются предками элемента с class="first"? Это элементы с class="text\_block" и class="news".
* Какой элемент является родителем элемента с class="first"? Только элемент с class="text\_block".
* Какой элемент мы можем назвать братским или соседним элементом для элемента, имеющего class="first"? Это элемент с class="second".

**Подробнее об атрибутах (data-атрибуты)**

Как мы уже говорили, некоторые атрибуты напрямую не влияют на отображение элемента на страницах. О некоторых таких важных атрибутах мы уже поговорили в шаге 3 (например, **id**).

А еще список атрибутов можно расширять: это значит, что разработчик может создать свой собственный атрибут и присвоить ему любые значения. Что это значит для тестировщика? Это значит, что можно договориться с разработчиками о специальном атрибуте, который вы будете использовать в своих тестах для поиска нужных элементов и который не будет изменяться при исправлении верстки сайта. Это добавит стабильности вашим тестам. Правда, есть несколько ограничений:

* веб-сайт должен использовать стандарт HTML5 (большинство современных сайтов соответствует этому требованию)
* использовать можно только латинские буквы, и символы дефис (-), двоеточие (:) и подчёркивание (\_)

Также принято названия таких атрибутов начинать со слова: "**data"**, например, "data-button".

Что еще важно знать про атрибуты элементов?

Некоторые атрибуты являются универсальными, они могут относиться к любому тегу и любому типу элементов. Например, **hidden** (т.е. скрытым) можно сделать любой элемент. Некоторые же атрибуты ассоциированы строго с определенным тегом, например, для картинки, которая задается тегом **img**, обязательно нужно указывать атрибут **src**.

Если вы собираетесь в дальнейшем работать с автоматизацией тестирования веб-продуктов, то вам будет очень полезно изучить HTML более детально. Вы сможете быстро подбирать нужные селекторы, с первого взгляда на HTML-разметку будете видеть что <a> - это ссылка, <p> - текст, а <ul> - ненумерованный (маркированный) список. Но это большая и широкая тема, которая заслуживает отдельного курса (можно проходить такие курсы самостоятельно, например, <https://stepik.org/course/38218/>).

Образовательная рекомендация: если вы не совсем понимаете, что означает тег или атрибут элемента, попробуйте погуглить. Каждый кусочек этих знаний так или иначе поможет вам в будущем.