CSS Selectors Cheat Sheet

Ahmed Rafik – Web Scraping with Python using Requests, LXML and Splash (Udemy)
1st edition

About the author

Hi! I'm Ahmed nice to meet you, my students prefer to call me **web scraping Ninja** and by the time preparing this Cheat Sheet I have taught more than 2000 students around the world how to do web scraping. I personally do web scraping on daily basis whether for fun, for personal projects or as a freelancer and guess what ? I even have a master degree in computer science.

You can find me at :

Twitter : https://twitter.com/ahmedrafik__

Youtube : https://www.youtube.com/channel/UCKHo1-b7ZH_CKvc7rtqWEvA

About this Cheat Sheet

* This Cheat Sheet covers all the examples I included in my **Web Scraping course with Python using Requests, LXML and Splash**.

* This Cheat Sheet covers only the basics of how to use CSS Selector to locate elements from the HTML markup, if you want to master Web Scraping then I recommend my to check my course on Udemy, you can follow this link in case you're interested in learning web scraping : (LINK)

* This Cheat sheet doesn't cover CSS as a language, instead it covers what we call **CSS Selectors** which we use to select elements from HTML web pages.

* All the CSS Selectors I'm gonna cover on this Cheat Sheet will be applied on the HTML markup I've added after this page.

HTML web page

```
<!DOCTYPE html>
  <html lang="en">
    <head>
      <title>XPath and CSS Selectors</title>
    </head>
    <body>
      <h1>CSS Selectors simplified</h1>
      <div class="intro">
        <p>
           I'm paragraph within a div with a class set to
intro
        <span id="location">I'm a span with ID set to
location and i'm within a paragraph</span>
        </p>
        <p id="outside">I'm a paragraph with ID set to
outside and i'm within a div with a class set to intro</p>
      </div>
       <p>Hi i'm placed immediately after a div with a class
set to intro
       </p>
       <span class='intro'>Div with a class attribute set to
intro
      </span>
      <ul id="items">
        <li data-identifier="7">Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
      </ul>

      <a href="https://www.google.com">Google</a>
      <a href="http://www.google.fr">Google France</a>
      <p class='bold italic'>Hi, I have two classes</p>
      <p class='bold'>Hi i'm bold</p>
    </body>
</html>
```

BASICS

An element is a **tag** in the HTML markup.

Example:

The **'p'** tag aka paragragh is called an element.

To select any element from HTML web pages we simply call it by its tag name.

Example:

To select all **p** elements we can use the following CSS Selector

```
p
```

Although this approach works perfectly fine, it's not recommended to use it, because if for example we want only to select the "**p**" elements that are inside the first div with a class attribute equals to "**intro**" this approach won't be the best solution, this is why we always prefer to target elements either by their **class** attribute, **id** or by **position** so we can limit the scope of the CSS selector.

CLASS & ID

So to select any element by its **class** attribute value we use the following syntax:

```
.className
```

If we want to target an element by its **id** attribute value we use the following syntax:

```
#id
```

Example:

Let's say we want to select the "**p**" elements that inside the "**div**" with a class attribute equals to "**intro**" in this case we use the following CSS Selector:

```
.intro p
```

If we want to select the "**p**" element with "**id**" equals to "**outside**" we can use the following CSS selector:

```
#outside
```

REMEBER:

Please note, the same exact class attribute value can be assigned to more than one element however, and id can be assigned to only and only one element.

Sometimes we want also to select elements based on a foreign attribute which doesn't belong to HTML markup standard. For example to select the "**li**" element with the attribute "**data-identifier**" equals to 7 in this case we use the following CSS Selector:

```
li[data-identifier="7"]
```

Sometimes the element we want to select does have two classes, for example, to select the "**p**" element with a class attribute equals to "**bold**" and "**italic**" in this case we use the following CSS Selector:

```
.bold.italic
```

OR:

```
p[class='bold tialic']
```

OR:

```
p.bold.italic
```

Speaking which one is better than the other one, it depends on you, which syntax you like more and which one you can remember as fast as possible.

Value lookup

Let's say you want to select all the "**a**" elements in which the "**href**" attribute value starts with "**https**" and not "**http**", in this case we can use the following CSS Selector:

```
a[href ^='https']
```

OR:

```
[href ^='https']
```

So search for the text at the beginning we use the caret sign "**^**"

Now if you want to search for a value at the end we use the "**$**" sign, for example to select the "**a**" elements where the "**href**" attribute value ends with "**fr**" and not "**com**" we use the following CSS Selector:

```
a[href $='fr']
```

OR:

```
[href $='fr']
```

Finally if we want to search for a particular value in between we use the tilde "**~**":

```
elementName[attribute ~='fr']
```

CSS Combinators

In CSS a combintor is a selector which describes the relationship between two elements, at the time making this Cheat Sheet we have 4 CSS combinators:

The descendant

The descendant combinator can be used to get all the desendants of an element, in other words it will return the elements that are placed **inside** a particular element for example, to get all the "**p**" elements that are descendants of the "**div**" element with a class attribute equals to "**intro**" we use the following selector:

```
.intro  p
```

The child (>)

The child combinator can be used to all the elements that are **immediate** children of a specific element, for example, to get all the children of the "**ul**" element **items** with id equals to "**items**" we use the following CSS selector:

```
#items > li
```

Adjacent sibling (+)

The adjacent sibling combinator can be used to select the element that is **immediately** placed after the closing the tag of the a particular element, for example, to get the "**p**" element that is placed **immediately** after the "div" element with a class attribute equals to intro we use the following selector:

```
.intro + p
```

General sibling (~)

The general sibling combinator can be used to select all the elements that are placed after the closing tag of an element but unlike the adjacent sibling combinator the elements we are looking do not necessarily need to be placed immediately after it, for example, to select all the "**p**" elements that are placed after the "**div**" with a class attribute equals to "**intro**" we use the following selector:

```
.intro ~ p
```

REMEBER:

The adjacent sibling combinator mentioned above will return only the "p" element that is placed immediately after the closing tag of the div.intro, however the general sibling combinator will return all the "p" elements (3).