**Web Element Locators**

We Have 6 types of web element locators

1. ID
2. Name
3. Class
4. Name
5. Css-Selector
6. XPaths

**ID Locator:**

ID are the simplest, faster and best locators on HTML DOM.

**Element on Web Page**

*<input id="search\_form\_input\_homepage" class="js-search-input search\_\_input--adv" type="text" autocomplete="off" name="q" tabindex="1" value="" autocapitalize="off" autocorrect="off" placeholder="Search the web without being tracked">*

**ID Locator:**

#search\_form\_input\_homepage

**Name Locator:**

Name are also simplest and fastest locators on web pages.

**Element on Web Page**

*<input id="search\_form\_input\_homepage" class="js-search-input search\_\_input--adv" type="text" autocomplete="off" name="q" tabindex="1" value="" autocapitalize="off" autocorrect="off" placeholder="Search the web without being tracked">*

**Name Locator:**

#q -> *which is Name attribute*

**Class Locator:**

Class is also simplest and fastest way to locate an element on web page.

**Element on Web page**

*<input id="search\_form\_input\_homepage" class="js-search-input search\_\_input--adv" type="text" autocomplete="off" name="q" tabindex="1" value="" autocapitalize="off" autocorrect="off" placeholder="Search the web without being tracked">*

**Class Locator:**

#js-search-input search\_input--adv *-> class name are not unique we use this to get list of elements*

**CSS Selector:**

Selecting elements using CSS attributes or properties

**Element on Web page**

*<input id="search\_form\_input\_homepage" class="js-search-input search\_\_input--adv" type="text" autocomplete="off" name="q" tabindex="1" value="" autocapitalize="off" autocorrect="off" placeholder="Search the web without being tracked">*

**CSS Selector Query (Simple):**

1. ***Div.js-search-input*** ->*js-search-input is a class name which will select all elements on web page which contain js-search-input class.*
2. **.js-search-input** -> *by searching through class name directly.*
3. **#search\_form\_input\_homepage** -> *When searching through ID in CSS Selector we search by starting # and the ID then.*

**Parent Child relationship or Dynamic CSS Selector (Advanced):**

**Decanted relationship:**

* div.cw dev.result **-> ………..**

**Parent relationship:**

.Parent-Name > Child-Name

.class-name > element inside that class’s element

.result > a

**OR Expression of CSS Selector:**

* div, a -> *this comma represent OR meaning that select element that contain dev or a tag.*

**OR Expression in Parent relationship:**

* a, li > ul, li -> *first expression represent parent element and after comma it represent child element.*

**Attribute Based Selection:**

* [class] -> typing attribute name inside brackets

To limit it more further

* div[attribute-name] -> This will pick the block which contain that attribute
* a[attribute-name = “value”] -> This will pick the block which contain that attribute which has same value as mentioned in the css selector query.
* div[class=”result”] -> This will pick block which has same value

**Contain value in CSS Selector:**

* div[class\*=”resu”]

**Sudo Class in CSS Selector:**

* div.class:not(.class-which-you-don’t-want) -> This will pick all elements but the one you mentioned in sudo class will be exclude.

**Index sudo class**

* div.result:nth-child(3) -> what ever index you want
* div.class-name a(data-id=”tips”) img

**Xpath Selector**

**Parent to Child**

//ul/a/li

//li[@class=”-----”]

With Logical Operators

//img[@weight < 20][@height <20] -- **and**

//img[@weight <20] and [@height<20] ---- **and**

//input[@name=”---’ or @id = “---”] ---- **OR**

**Functions in Xpaths**

//div[contains(@contains(@class, ’-------’] ----- **Contains functions**

//div[starts-with(@class, ‘-------’)] ------ **Starts-with**

//div[not(contains(@class, ‘-------’)] --- **Not**

//div[not(starts-with(@class, ‘-----’))] ----- **Not-starts-with**

//button[normalize-space=’sign in’] --- **Normalize-space**