Selenium locators

What's a locator?

A web page can have a wide variety of elements. When a page loads, the browser generates a Document Object Model ("DOM") for it, which is a tree of nested elements that are present in that page. We should be able to identify these elements whenever they're involved in a test (e.g., to verify a specific element is visible, or to interact with it). This is done through a variety of strategies and might require just a little bit of HTML/CSS/XPath knowledge (just enough to understand what's going on).

Identifying these elements will be necessary in order to interact with them using Selenium (e.g.: click on a button, verify if some text has the expected formatting, enter text in an input, etc.), and Selenium provides a few ways of doing so:

- Identifier
- Id
- Name
- Link
- DOM
- XPath
- CSS
- UI-element

HTML/CSS

This calls for some basic **HTML** and **CSS** knowledge. As a brief overview, each element in an HTML document is represented by a tag, like <button>, <p> (paragraph), (division or section), <img> (image), etc. These elements can also have a variety of attributes, some of them affecting the way they look and some others that don't, but that are important in other ways. For instance, an element can have an "id" that will identify this element in a unique way, or can also have a "class" applied to a group of elements that share some meaning: <p class="footer">. Also, CSS allows to alter the way elements look or behave, and for that it uses its own selection strategies, using IDs, classes, inheritance and so on. So, let's say a DOM has a element containing a <p class="footer">, then we can express that with a CSS selector like: **div p.footer** (reference: https://www.w3schools.com/cssref/css_selectors.asp).

Strategies

Whatever the strategy used, we should make sure the element is correctly identified and that our locator matches that element only. That means the locators must be reliable. For example, if we use the class of an element, since classes can be applied to more than one element in the DOM, this could (and probably will) lead to identifying more than one element with one locator. On the other hand, since ids are unique (or should be) and independent of the element type and location in the DOM, an id might be a good choice to pinpoint an element.

It's not always easy to find a good balance between flexibility (to allow valid small changes in the DOM) and bug detection. If a locator is too strict, any change in the DOM will cause a failed test when there might be no bug at all; but if it's too flexible, then it might end matching more than one element. Usually, ID, CSS and XPath (in that order) are the preferred strategies to locate elements. However, XPath is the slowest and less flexible way to do it, so it should be used with caution (XPath can also get very complicated and it would take some time to master it).

So, from the strategies named above, only a few of them are used most of the time:

- **ID** is a global attribute (meaning an ID can show up only once per DOM) that allows us to select an element based on the #id that it has in the DOM. As an example, let's suppose that there is some button with id #submit: <button type="button" id="submit">Submit data</button>. In this case, the id #submit will only select this element. However, although the HTML specification calls for unique IDs, there are exceptional cases where they are not. This should be taken into account when assessing which strategy to use to locate elements.
- **CSS** allows to combine IDs and other attributes using CSS selector strategies. This is where we can make locators more rigid or flexible as needed.
- **XPath** is a language in itself (reference: https://www.w3schools.com/xml/xpath_syntax.asp), allowing to select nested elements in a very specific way. Experience dictates that XPath is not the best strategy to be used, as it's too strict and leaves no place for small changes, as a nice CSS selector would.

How to create a locator?

When trying to identify an element, a good way is to open the web page and use some browser tool like "Inspect Element" (by right-clicking on the element we want to locate). The HTML code will be displayed and we'll be able to see what kind of element it is (a paragraph, a header, an image, etc.), if there are any available IDs that we can use, or if we must use some other strategy.

If there are no unique IDs or names that can be used, then we should try to build a CSS selector. Usually, the best way is to build the locator from that element up, looking at the element type first, then at the element one level above that contains it, and so on. Also, browser dev tools normally provide a way to just right-click on an element and copy the CSS selector (or the XPath). But preferrably we should know what this selector means in order to be able to maintain or improve it (doing something without understanding what it implies is never a good thing).

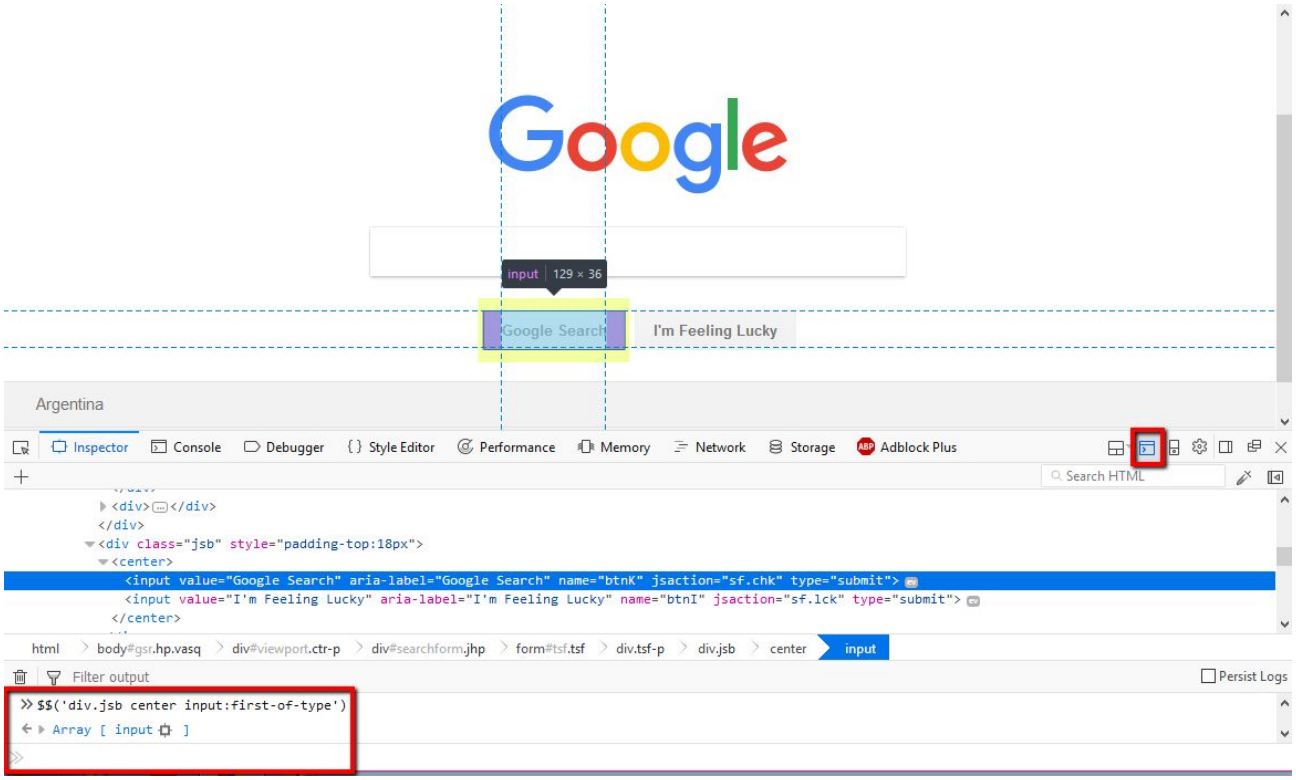
As an example, let's say we want to find the "Google Search" button in Google's home page. So, upon element inspection (right click and "inspect element" or Ctrl+Shift+C in Firefox or Chrome), we can see that this is an "input" element type that's contained in a "center" element inside a "div" within the DOM, and this "div" element has a class called "jsb". Also, this is not the only input element within this div (since there's another search button), so we must select only the one we care about. So the following CSS selector will match the desired button: div.jsb center input:first-child

This is where some CSS knowledge will be helpful, as we must indicate that something is a class applied to some element (with the '.' dot), or that something is a nested object (with the ' ' space

character), or that we want to select only the first of the same kind elements that the div contains (with the :first-child selector). For this purpose it's always good to keep in mind CSS selectors.

To test that the selector we've come up with is right, we can use the browser's web console (pressing ESC in Firefox or Chrome, or the web console button in the web inspector). In this console, type the CSS locator wrapped by $$('') like this:
$$('div.jsb center input:first-child')

(As a reference, web console helpers in Firefox are listed here: https://developer.mozilla.org/docs/Tools/Web_Console/Helpers).



An array with only one element will be returned. Clicking on that element we can verify it's the button we were trying to locate. If the array size is bigger than 1, then the locator is matching more than one element and we should try to find a way to identify just the one we want.

In the end, all comes down to experience...

Writing reliable, readable, flexible locators is an art, and there are many good articles about it. Mostly, this knowledge will be acquired with experience, so keep practising!