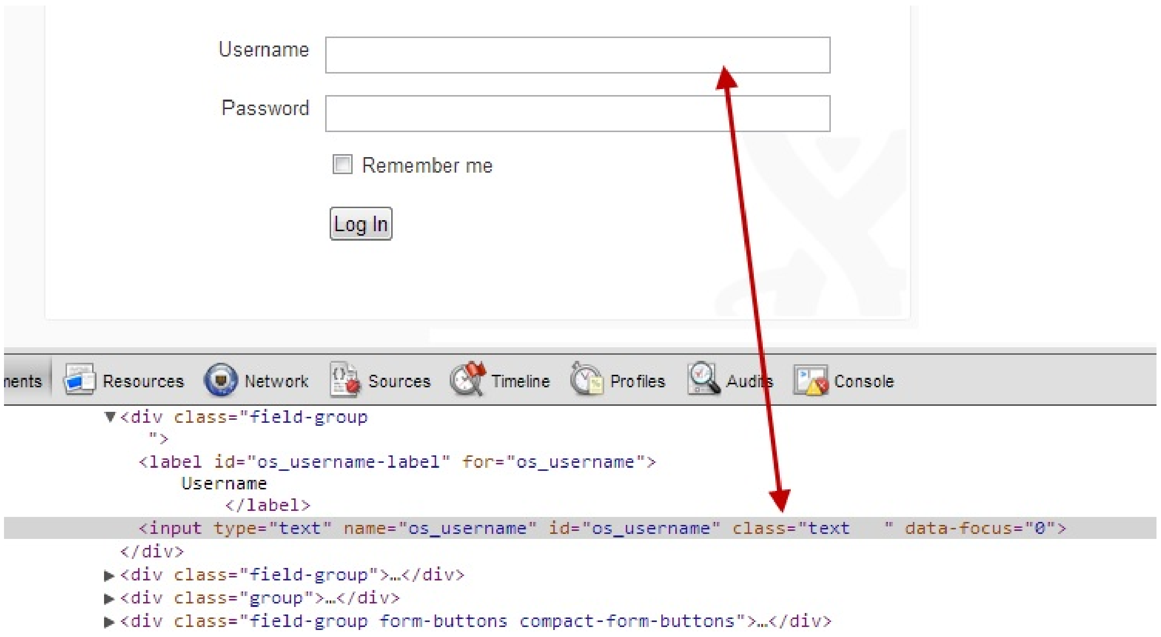
3.Locating the elements

Locating the elements

The next step after opening the browser and navigating to the webpage, is to locate the elements with which we want to interact.



```
▼<div class="field-group
   ">
  <label id="os_username-label" for="os_username">
      Username
          </label>
  <input type="text" name="os_username" id="os_username" class="text   " data-focus="0">
</div>
▶<div class="field-group">…</div>
▶<div class="group">…</div>
▶<div class="field-group form-buttons compact-form-buttons">…</div>
```

In the above figure the complete highlighted row, is the HTML element.
<input> is the tag.
type, name, id, class, data-focus are the various attributes and "text", "os_username", text, and "0" are their respective values.

To identify an element on webpage we have to see the various attribute-value pairs and judge which attribute will uniquely identify the element.Locating a web element can be done in context of driver or in context of other web element. When searched in context of driver, the element is searched on the whole page but when searched using a web-element it's located in reference to it Selenium webdriver API offers two methods to find an element i.e. findElement and findElements.

findElement returns the uniquely identified web element, and in case of multiple elements having the same locator it returns the first element. If no element is found it throws an exception.
findElements returns the list of elements having the same locator. In case no element is found than an empty list is returned

By Id

Identifying an element by its Id is the easiest, safest and hence most preferred way to uniquely locate it. For example the in above figure the element is having an id = "os_username". So the corresponding command in selenium would be :

WebElement e = driver.findElement (By.id("os_username"));

By name

In cases when id of an element is not present, than the next choice for element identification is via 'name ' attribute.
But name element not guarantees always of identifying the element uniquely, so this should be used with other techniques when trying to uniquely identify the element.
WebElement e = driver.findElement (By.name("os_username"));
OR
WebElement e = driver.findElements (By.name("os_username")).get(0);

By classname

This locating technique is least preferred when locating element uniquely. However most preferred when we need to find multiple elements.

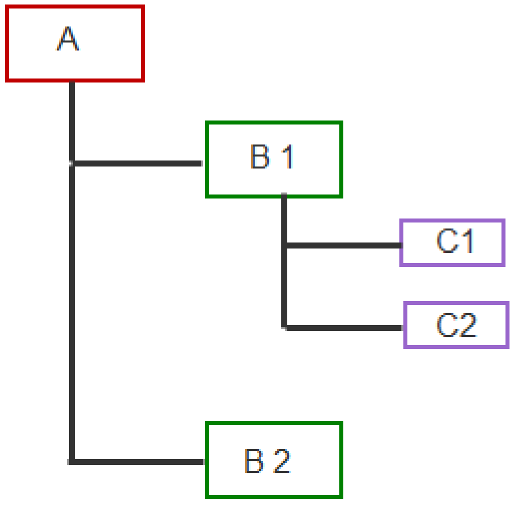
List<WebElement> elements = driver.findElemets(By.className("text"));

By XPath

Xpath is the most reliable and robust locating technique. But scripts executes a little slower with xpaths. Xpaths can be divided into two categories :

- Absolute Xpaths : These xpaths starts with root element (html) and end with the desired descendant element. Absolute xpaths are mostly long, more vulnerable to break and hence not robust.
- Relative Xpaths : Xpaths can be created using relative location paths. Instead a starting from the root element we start referencing from an element. Relative xpaths are short and more robust.

To use xpath efficiently, we should use xpath axis : An xpath axis is a path through the node tree, making use of particular relationship between nodes.



Nodes can be classified based on:

- parent
- children
- siblings
- ancestor
- descendant

Axis	**Description**	**Example from above figure**
descendant	Children of the current node and their children too.	All the nodes are descendant of node A
ancestor	Parent of current node and their parent.	B1 and A are ancestor of C1.
following-sibling	following siblings	C2 is following sibling of C1 and B2 is following sibling of B1
preceding-sibling	preceding siblings	C1 is preceding sibling of C2 and B1 is preceding sibling of B2
following	following siblings and their descendants.	B2 is following for B1.
parent	parent of the current node.	B1 is parent of C1.
preceding	preceding siblings and their descendants.	B1, C1 and C2 are preceding to B2.

Selecting nodes:

Expression	**description**
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Conditions in an xpath can be defined using predicate, meaning including the element identification inside "[ ]". For example "//div / span [1]" will point to the first span inside a div.
We can use attributes-value pairs also in xpath by prefixing the attribute with '@' symbol. For example to use id as an attribute we'll use "//div / input [@id = os_username]".

To make complex xpaths look simple and more robust, we can use xpath functions as follows :

Function	**Description**
last()	The last function returns a number equal to the context size from the expression evaluation context.
not()	The not function returns true if its argument is false, and false otherwise.
count()	The count function returns the number of nodes in the argument node-set.
string-length(str)	The string-length returns the number of characters in the string str. If the argument is omitted, it defaults to the context node converted to a string, in other words the string-value of the context node.
position()	The position function returns a number equal to the context position from the expression evaluation context.
contains(str1,str2)	The contains function returns true if the first argument string contains the second argument string, and otherwise returns false.

Example :

The below table contains some examples of Xpaths and CSS element identification for Login form from Yahoo.

```
<div id="inputs" class="yui-skin-sam">
 <input id="username" type="text" value="" autocomplete="off" autocorrect="off"
placeholder="ID Yahoo" aria-required="true" tabindex="1" maxlength="96" name="login">
 <input id="passwd" type="password" value="" autocomplete="off" autocorrect="off"
placeholder="Parol" aria-required="true" tabindex="2" maxlength="64" name="passwd">
<div id="captchaDiv"></div>

<div id="persistency">
 <input id="persistent" type="hidden" value="" name=".persistent"
style="background-color: rgb(255, 255, 255);">
 <span id="pLabel" tabindex="3" for="persistent">
  <span id="pLabelC" class="checkbox lg-sprite "> </span>
  <span class="labeltxt">Pstreaz-m autentificat</span>
 </span>
</div>

<div id="submit">
 <button id=".save" class="lgbx-btn purple-bg" tabindex="4" name=".save"
type="submit">Autentificare</button>
</div>
```

Element	**XPATH**	**CSS**

password field	//div[@id='inputs']/input[@id='passwd']	css=div[id='inputs'] input[id='passwd']
		
		css=div#inputs input#passwd

	//div[@class='yui-skin-sam']/input[@id='passwd']	css=.yui-skin-sam input#passwd
	//div/input[contains(@placeholder,'Parol')]	css=div input[placeholder=Parol]
	//div/input[starts-with(@placeholder,'Par')]	css=div input[placeholder^=Par]
	//div/input[@name="passwd"]	css=div input[name="passwd"]
	//div/input[@name="passwd"][@id="passwd"]	css=div input[name="passwd"][id="passwd"]
	//div/input[@name="login"]/following::input	css=div input[name="login"] ~ input
button	//div/button[contains(text(),'Autentificare')]	css=div button:contains('Autentificare')
	//div[@id="persistency"]/span/span[contains(text(),"Pstreaz-m autentificat")]/following::button	
"pastreaza-ma autentificat" checkbox	//div/descendant::span[@class='checkbox lg-sprite ']	

Xpaths can be used in selenium to find an element as shown below :

String xpathString = "//div[@id='inputs']/input[@id='passwd']";
WebElement e = driver.findElement(By.xpath (xpathString));

or

String passwordField = By.xpath(//div[@id='inputs']/input[@id='passwd']);

WebElement e = driver.findElement(passwordField);