**CSS selectors for Selenium with example**

When we don't have an option to choose Id or Name, we should prefer using CSS locators as the best alternative.  
CSS is "Cascading Style Sheets" and it is defined to display HTML in structured and colorful styles are applied to webpage.

Selectors are patterns that match against elements in a tree, and as such form one of several technologies that can be used to select nodes in an XML document. Visit to know more [W3.Org Css selectors](http://www.w3.org/TR/css3-selectors/)

* CSS has more Advantage than Xpath
* CSS is much more faster and simpler than the Xpath.
* In IE Xpath works very slow, where as Css works faster when compared to Xpath.

[Click here to view examples compared with xpath and css](http://seleniumeasy.com/selenium-tutorials/examples-for-xpath-and-css-selectors)

**Syntax:**

tagName[attributename=attributeValue]

Example 1: input[id=email]

Example 2: input[name=email][type=text]

**In CSS there are two special characters which has important role to play.**

**1. dot(.) refers to class.**

Syntax:

css=input.submitbtn

For example if the below is the html for a sign button

<button class="submit btn primary-btn flex-table-btn js-submit" **type**="submit" style="background-color: rgb(85, 172, 238);">

Log **in**

</button>

In the above html there are multiple classes used for the single button. How to work in such a situation????  
Below are the examples to work with classes. If you observe, we have combined multiple classes to work. As the class is not unique like ID, we may require to join two classes and find the accurate element.

Example 1: css=.primary-btn

Example 2: css=.btn.primary-btn

Example 3: css=.submit.primary-btn

The above can be written like below **in** selenium

WebElement ele1 = driver.findElement(By.cssSelector(".primary-btn"));

WebElement ele2 = driver.findElement(By.cssSelector(".btn.primary-btn"));

WebElement ele3 = driver.findElement(By.cssSelector(".submit.primary-btn"));

**Hash(#) refers to Id**

Example:

css=input[id=email]

The above one can be re-written as

css=input*#destination*

**CSS locator Examples using ID and Class attributes**

/\* below syntax will find "input" tag node which contains "id=email" attribute \*/

css=input[id=email]

In selenium we can write it as

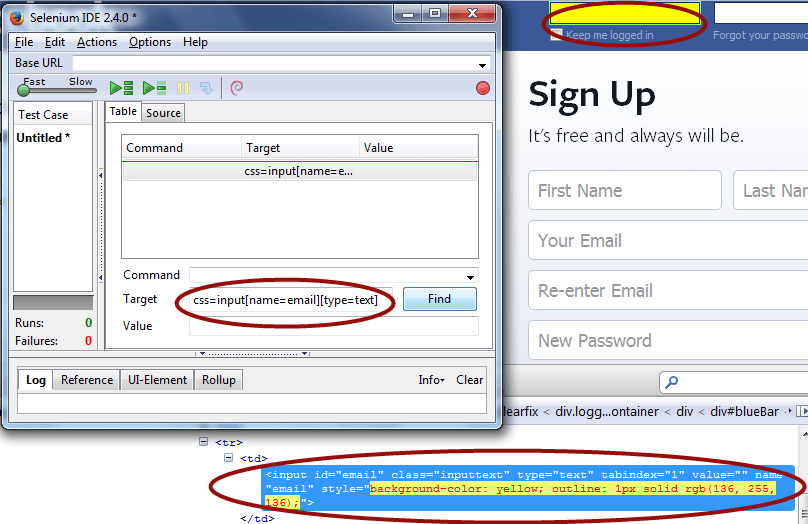
WebElement Email = driver.findElement(By.cssSelector("input[id=email]"));

Email.SendKeys("[hello@sampleemail.com](mailto:hello@sampleemail.com)");

You can make use of Selenium IDE to verify if the identifier is working fine or not. If the element has identified, it will highlight the field and html code in Yellow color.

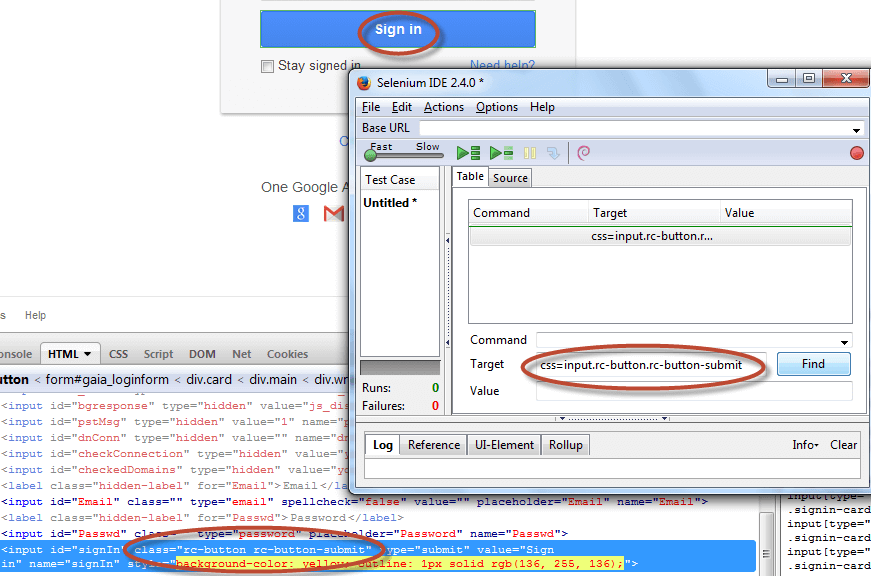
Below syntax will find "input" tag which contains "id=email" and "type=text" attribute. Again here we have added multiple attributes which the input tag has. For username, we will have the text type as 'text' and for password the text type will be 'password'.

css=input[name=email][type=text]

Please find the below screen shot with example:  


Below is the syntax for using input Tag and class attribute: It will find input tag which contains "submitButton" class attribute.

css=input.rc-button.rc-button-submit

Please find the below screen shot with example:  


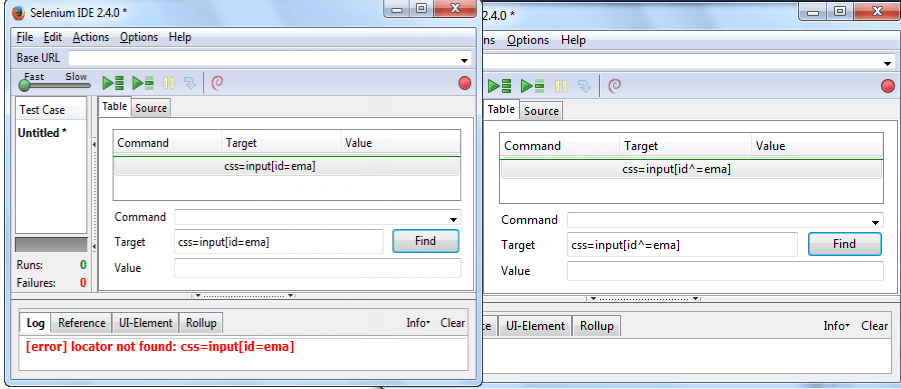
Using CSS locators, we can also locate elements with **sub-strings**. Which are really help full when there are dynamically generated ids in webpage

There are there important special characters:  
**1. '^' symbol, represents the starting text in a string.**  
**2. '$' symbol represents the ending text in a string.**  
**3. '\*' symbol represents contains text in a string.**

**CSS Locators for Sub-string matches(Start, end and containing text) in selenium**  
/\*It will find input tag which contains 'id' attribute starting with 'ema' text. Email starts with 'ema' \*/

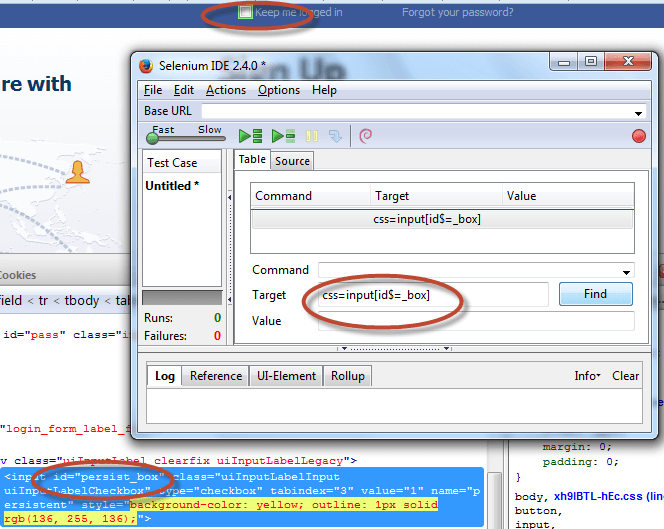
css=input[id^='ema']

If you remove the symbol an try to find the element with same sub-string, it will display error as "locator not found". We can observe the error in the below screen shot. one with error and the other with success



/\*It will find input tag which contains 'id' attribute ending with 'ail' text. Email ends with 'mail' \*/

css=input[id$='mail']

Please find the below screen shot with example:  


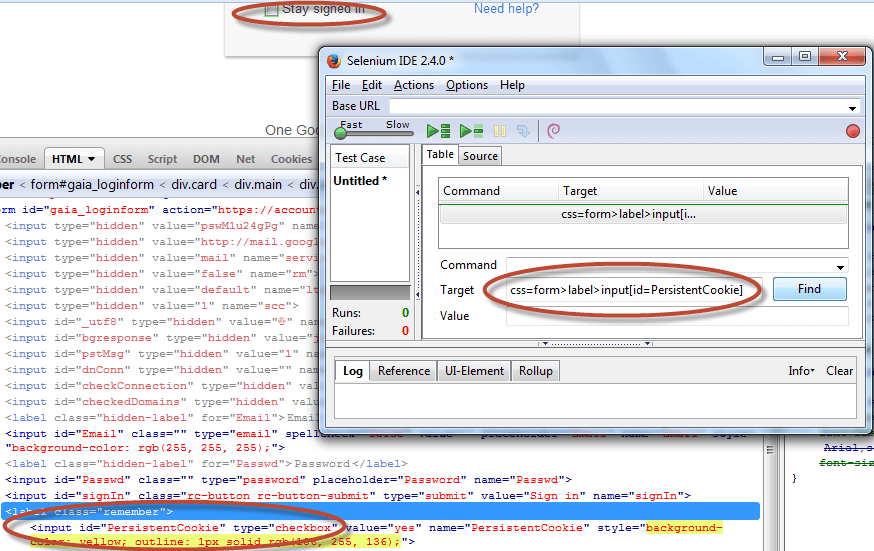
/\* It will find input tag which contains 'id' attribute containing 'mai' text. Email contains 'mai' \*/

css=input[id\*='mai']

**CSS Element locator using child Selectors**

/\* First it will find Form tag following remaining path to locate child node.\*/

css=form>label>input[id=PersistentCookie]

Please find the below screen shot with example:  


**CSS Element locator using adjacent selectors**  
/\* It will try to locate "input" tag where another "input" tag is present on page. the below example will select third input adjacent tag.

css=input + input + input