

CSS Selector

XPath Statements or CSS Selectors? Which is preferred ?

XPath statements slow down the tests where as CSS selectors run faster than the equivalent XPath. So CSS Selectors are preferred out of all the available locators to identify the elements. In order to understand the CSS selectors you should have knowledge on HTML

What is CSS ?

CSS stands for Cascading Style Sheets. CSS defines how HTML elements are to be displayed (i.e. The styles of the HTML elements are defined in the CSS document).

Suppose if you have specified that all the headings i.e. elements between the <h1> to <h6> tags should be displayed in red color in the CSS document. When the HTML document uses this CSS document, all the heading tags in the HTML document will import the style defined for the headings in the external CSS document and display all the headings in Red color.

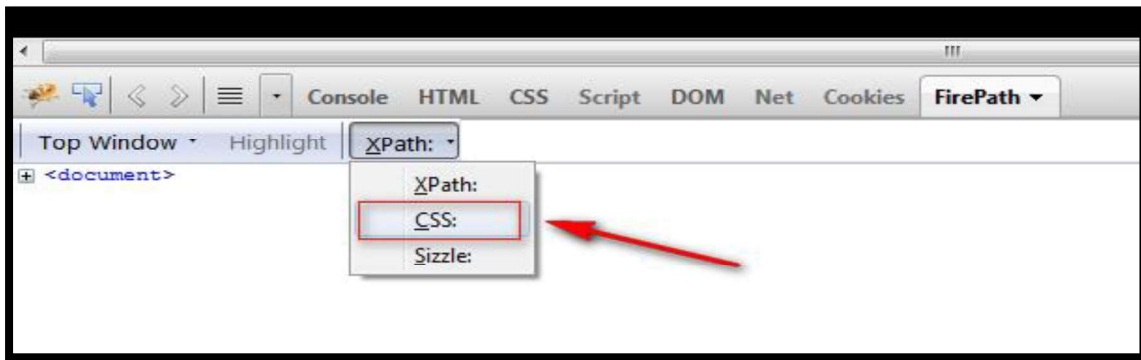
CSS documents save a lot of time for the developers. By changing the style properties in CSS document, the style of the elements in the HTML code changes. That is the developer won't have to go through each and every step in HTML code to change the style of the elements.

Suppose the developer has to change the color of the headings from red to blue. By changing the color property of headings in CSS will save the time of the developers. If not he has to find all the heading tags in the HTML code and change the style properties for every tag.

What are CSS selectors ?

In CSS, selectors are the patterns used to select the elements. In order to start with CSS syntaxes follow the below mention process:

1. Open <http://www.google.com/> in Firefox Browser
2. Click on 'FireBug' option
3. Select 'Firepath' tab from the displayed 'FireBug' options
4. In 'Firepath' tab, select 'CSS' option from the below shown drop down field:



5. Now type **body** CSS selector into the 'CSS' text box as shown below and click on 'Eval' button:



6. Observe that the body of the web page is highlighted with blue dots

This is how the **body** CSS selector locates the body of the web page.

Basic Syntax of CSS selector:

Absolute CSS Selector path

Absolute path means full path. In CSS path, it starts from html.

1. Suppose if we want to locate all paragraph text elements on the http://compendiumdev.co.uk/selenium/basic_web_page.html page by using Absolute CSS Selectors path then follow the below steps.
2. View the page source of this page as shown below:



3. By looking at the hierarchy of the above HTML Source, let's create absolute CSS path for locating the 'Another paragraph on the page' text.

4. Since we are creating an absolute path, the path should start with **html** selector (i.e. Root tag)

5. There are two child tags for **<html>** tag, i.e. **<head>** and **<body>**. But we need to locate 'Another paragraph on the page' text which is inside the **<p>** tag. There are two **<p>** tags and both of them are under the **<body>** tag. So **html > body > p**

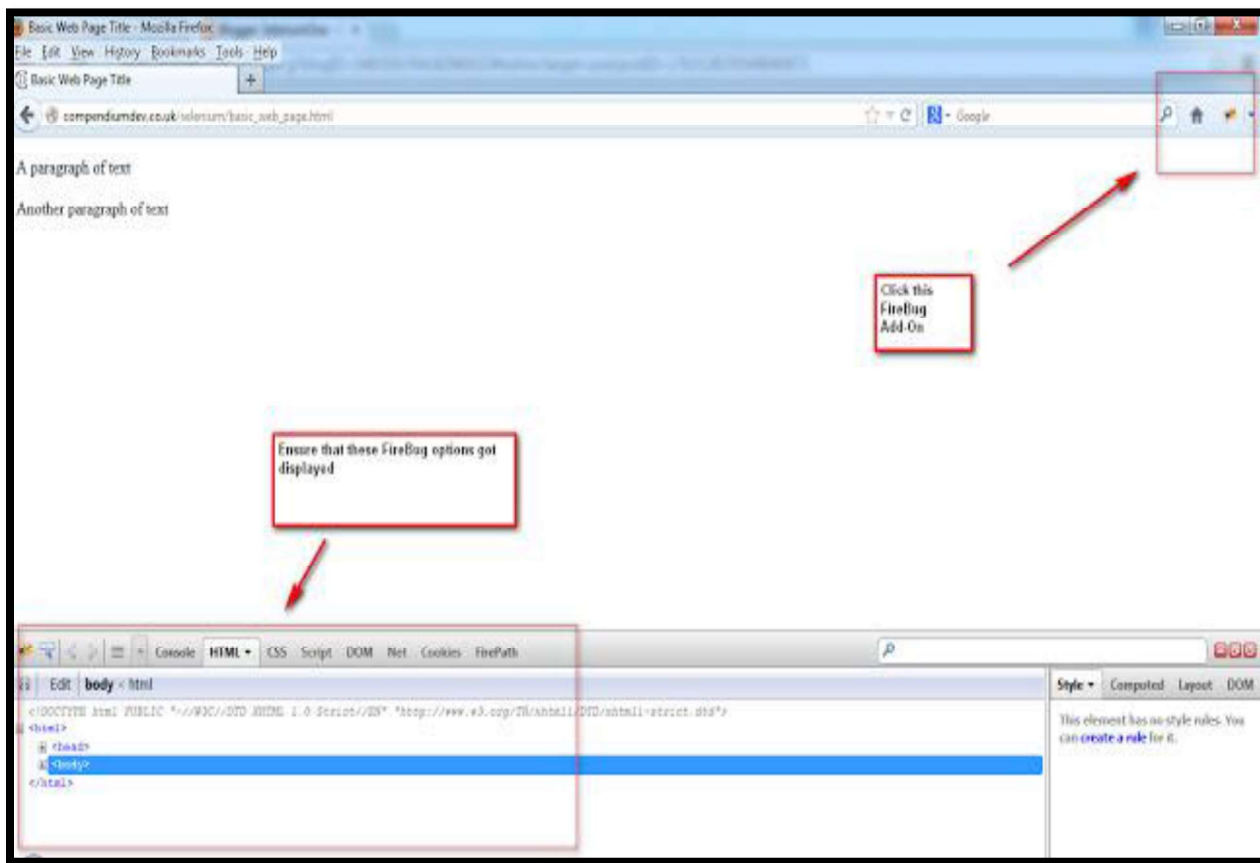
7. As **<p>** tags are children to the **<body>** tag, we can write the absolute CSS path for identifying both the paragraphs as **html > body > p**

So from this Process we've found that **html > body > p** is the absolute CSS path for locating all the paragraph text elements on the page. But let's verify whether we can locate them using this Absolute path by following the below steps:

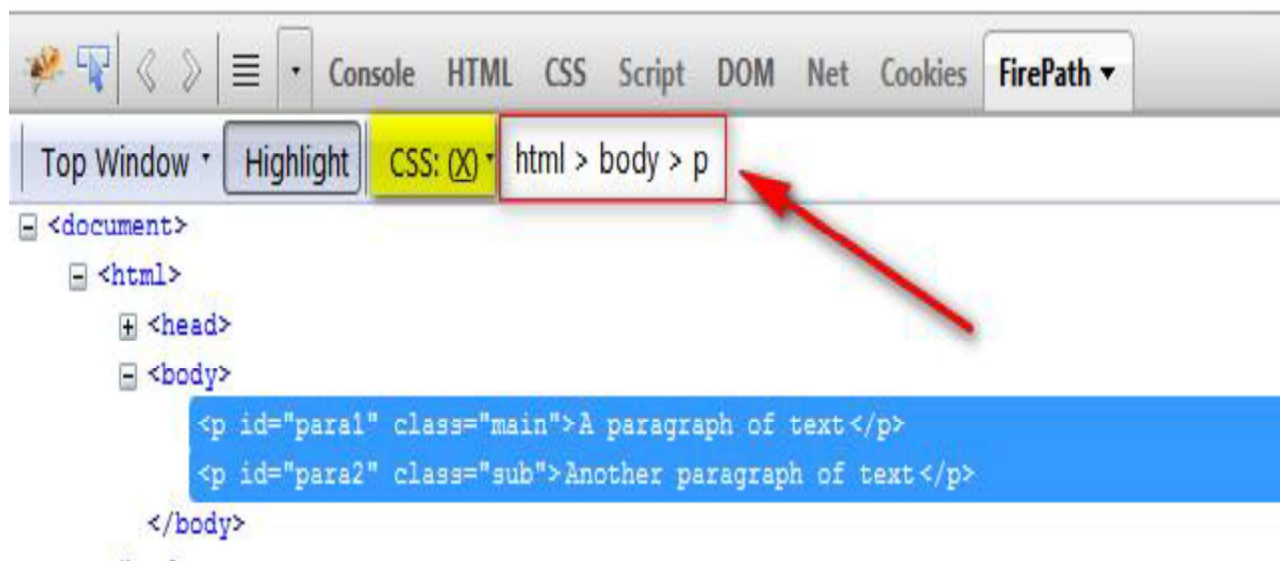
1. Launch Firefox Browser

2. Open http://compendiumdev.co.uk/selenium/basic_web_page.html in the Firefox Browser

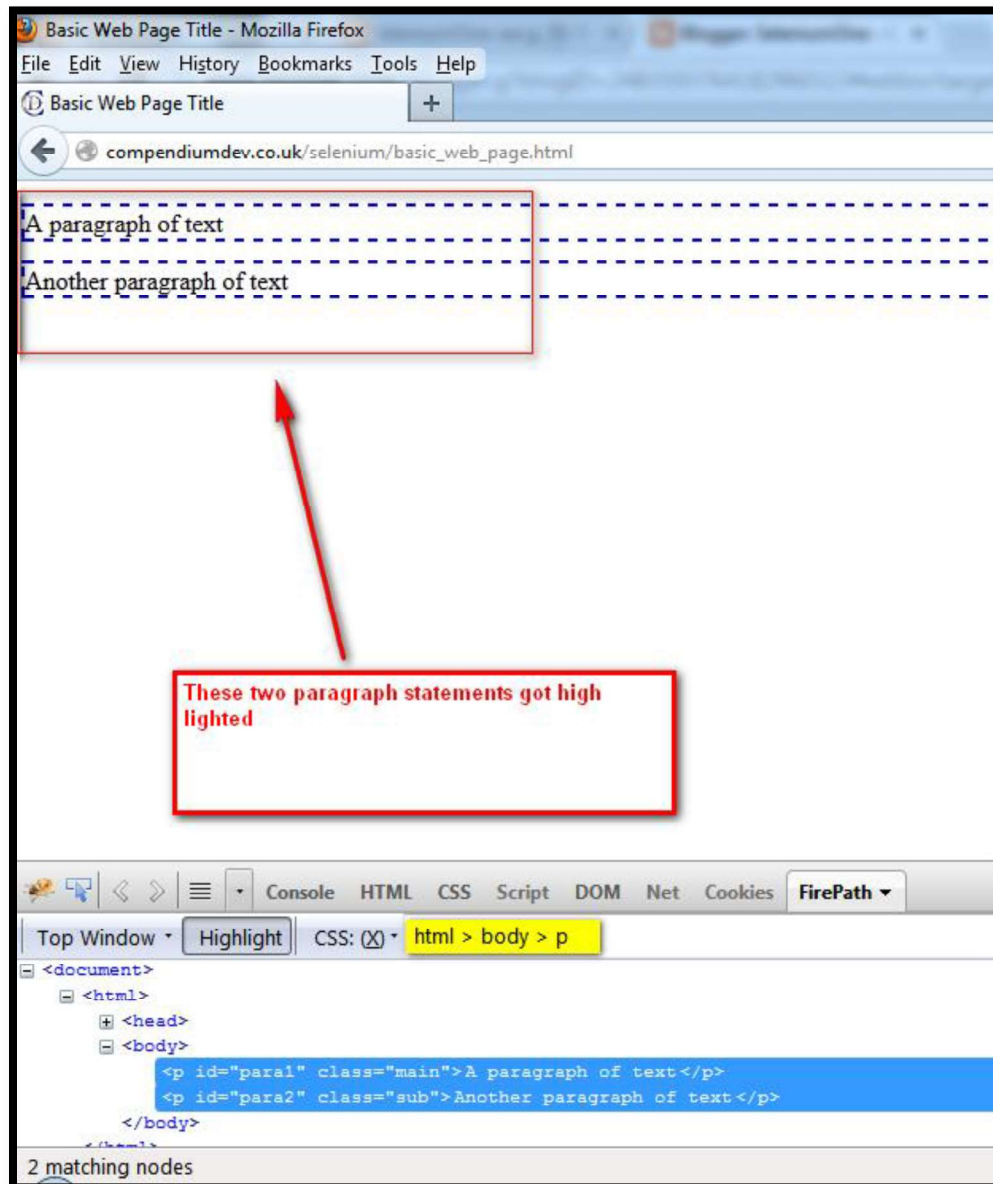
3. Click on 'FireBug' option and ensure that the 'FireBug' options got displayed as shown below:



4. Click on the 'Firepath' tab, select the CSS option, paste the created absolute CSS path i.e. **html > body > p** into the text box as shown below and click on 'Eval' button:



5. Observe that all the paragraph texts on the page are highlighted as shown below:



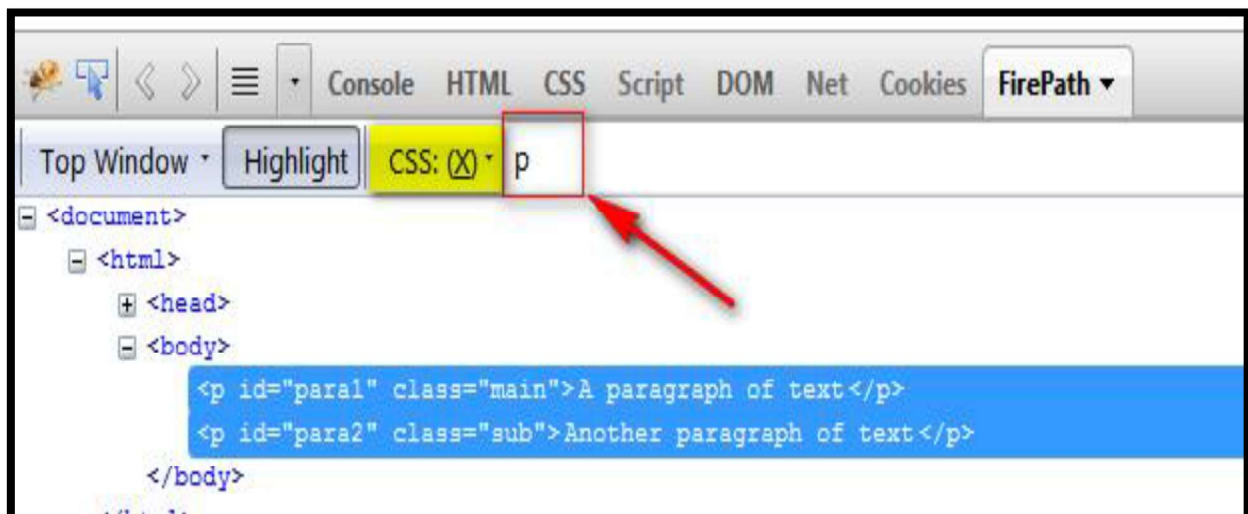
So we've used the Absolute CSS path **html > body > p** for locating all the paragraph text elements on the page.

Relative CSS Selector path

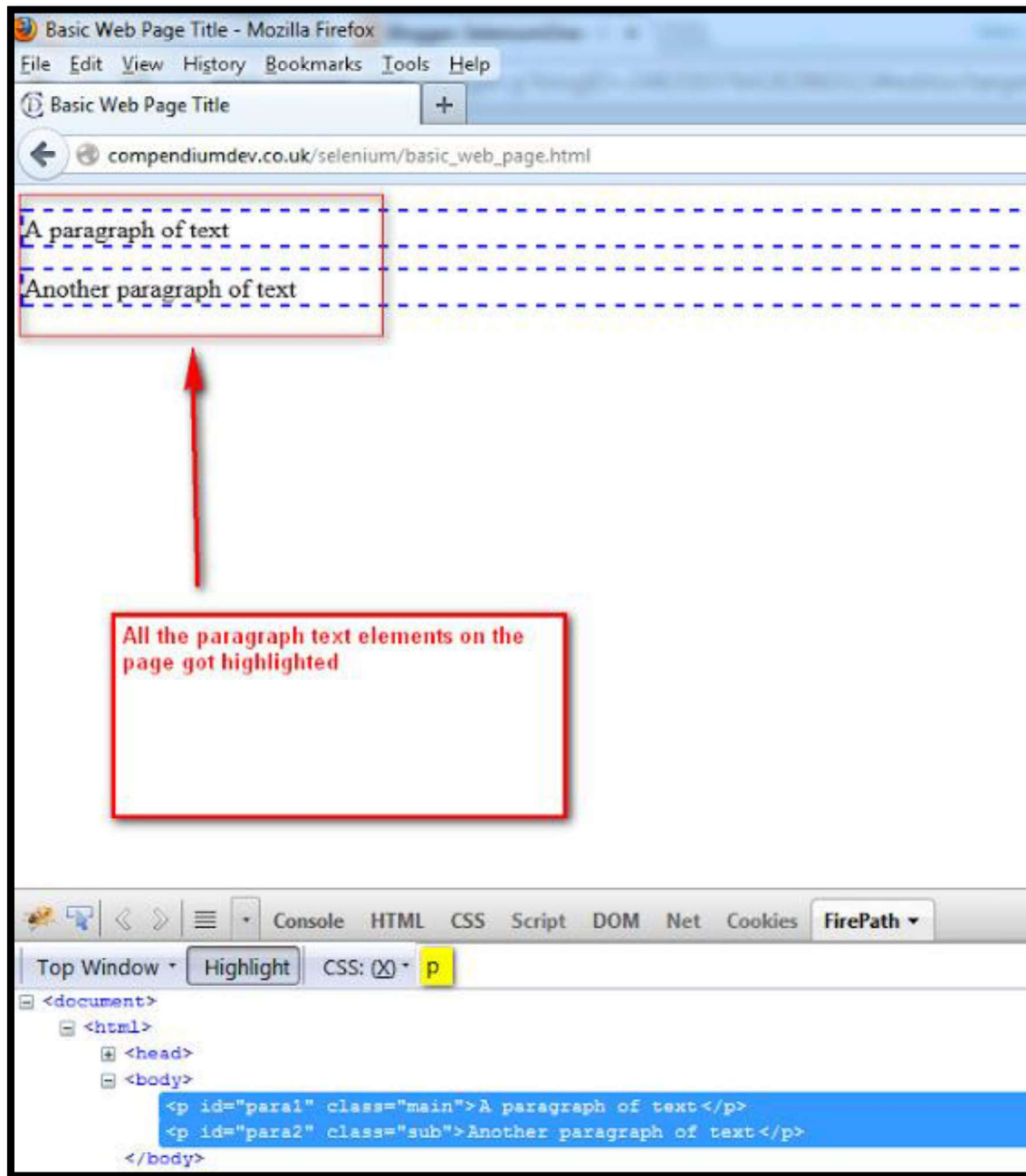
Relative path means shortcut path. It goes directly to the element instead of parsing or searching from the root element **html**.

the **absolute path** for locating all the paragraph text elements on the page is **html > body > p** . So the **relative path** for locating all the paragraph text elements on the page is just **p** .

1. Suppose if we want to locate all paragraph text elements on the http://compendiumdev.co.uk/selenium/basic_web_page.html page by using Relative CSS path then follow the below steps.
2. Open http://compendiumdev.co.uk/selenium/basic_web_page.html in the Firefox Browser
3. Click on the 'Firepath' tab, select the CSS option, paste the created Relative CSS path i.e. **p** into the text box as shown below and click on 'Eval' button:



4. Observe that all the paragraph texts on the page are highlighted as shown below:



So we've used the Relative CSS path **p** for locating all the paragraph text elements on the page.

CSS Selector by ID :

In CSS Selector, we can use **#** to locate the elements by id.

Examples:

if the HTML code of an web element is –

```
<input type="text" id="para1" class="divoUtil" name="teamDivo" placeholder="username">
```

Possible css selector syntax using ID will be :

p#para1 -> Locates the paragraph element which has the id value as 'para1'

#para1 -> Locates all the elements which has the id value as 'para1'

CSS Selector by class:

CSS Selector Rule - .class

Example -

For the Sample HTML below-

```
<button id="submitButton1" type="button" class="btn">Submit</button>
```

Possible css selector syntax using classname will be :

CSS Locator - **.btn**

Description - **'btn'** will select all the elements with class 'btn'.

```
<button id="submitButton1" type="button" class="show-tooltip personalDataName personData">Submit</button>
```

Possible css selector syntax using classname will be :

CSS Locator - **.show-tooltip.personalDataName.personData**

CSS Locator - **.personData**

CSS Locator - **.personalDataName**

CSS Locator - **.personalDataName.personData**

CSS Locator - **.show-tooltip.personalDataName**

Description – Every space in class name should be replace with “.” On this syntax

CSS Selector by tag

CSS Selector Rule - tagName

Example -

For the Sample HTML below-

```
<input id="fname" type="text" name="firstName" class="textbox">
```

Possible css selector syntax using tag name will be :

CSS Locator - input

Description - 'input' will select all the input type elements.

CSS Selector by attributes and their value

CSS Selector Rule - [attributeName='attributeValue']

Example -

For the Sample HTML below-

```
<input id="fname" type="text" name="firstName" class="textbox">
```

Possible css selector syntax using attribute will be :

CSS Locator - [name='firstName']

Description - [name='firstName'] will select the elements with name attribute having value 'firstName'.

Now, using these basic rule of locating web elements, we can use them in conjunction to create more robust locators, selecting unique elements.

CSS Selector by tags and id

CSS Selector Rule - tag#id

Example -

For the Sample HTML below-

```
<input id="fname" type="text" name="firstName" class="textbox">
```

Possible css selector syntax using tag name along with id will be :

CSS Locator - input#fname

Description - input#fname will select the 'input' element with id 'fname'.

CSS Selector by tags and class

CSS Selector Rule - tag.class

Example -

For the Sample HTML below-

```
<input id="fname" type="text" name="firstName" class="textbox">
```

Possible css selector syntax using tag name along with class name will be :

CSS Locator - input.textbox

Description - input.textbox will select the 'input' element with id 'textbox'.

CSS Selector by tags and attributes

CSS Selector Rule - tag[attributeName='attributeValue']

Example -

For the Sample HTML below-

```
<input id="fname" type="text" name="firstName" class="textbox">
```

Possible css selector syntax using tag name along with any attribute will be :

CSS Locator - input[name='firstName']

Description - input[name='firstName'] will select the 'input' element with 'name' attribute having value 'firstName'.

CSS Selector by first child, last child and nth child

We can locate the first, last and nth child elements by using ***:first-child**, ***:last-child** and ***:nth-child(Number)** notations in CSS path.

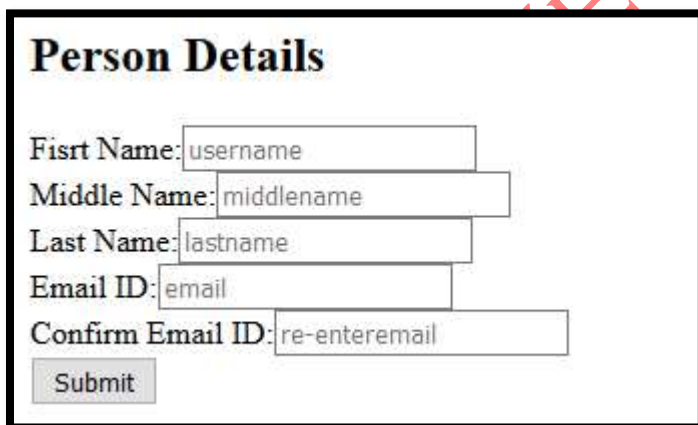
Examples:

body > *:first-child -> Locates the first child element under the body tag

body > *:last-child -> Locates the last child element under the body tag

body > *:nth-child(2) -> Locates the 2nd child element under the body tag

Find below the web page UI :



Person Details

First Name: username

Middle Name: middlename

Last Name: lastname

Email ID: email

Confirm Email ID: re-enteremail

Submit

HTML code of above Webpage :

```
<html>
<head>
<body>
  <h2> Person Details</h2>
  Firrt Name:
  <input id="username" class="detailsPage" type="text" placeholder="username">
  <br>
  Middle Name:
  <input id="m_name" class="detailsPage" type="text" placeholder="middlename">
  <br>
  Last Name:
  <input id="l_name" class="detailsPage" type="text" placeholder="lastname">
  <br>
  Email ID:
  <input id="e_mail" class="detailsPage" type="text" placeholder="email">
  <br>
  Confirm Email ID:
  <input id="c_email" class="detailsPage" type="text" placeholder="re-enteremail">
  <br>
  <input type="button" value="Submit">
</body>
</html>
```

Use first child in CSS selector :

Person Details

Firrt Name:

Middle Name:

Last Name:

Email ID:

Confirm Email ID:

Top Window | Highlight | CSS: (X) | body>*:first-child

```
<document>
<html>
  <head>
  <body>
    <h2> Person Details</h2>
    Firrt Name:
    <input id="username" class="detailsPage" type="text" placeholder="username"/>
    <br/>
    Middle Name:
    <input id="m_name" class="detailsPage" type="text" placeholder="middlename"/>
    <br/>
    Last Name:
    <input id="l_name" class="detailsPage" type="text" placeholder="lastname"/>
    <br/>
    Email ID:
    <input id="e_mail" class="detailsPage" type="text" placeholder="email"/>
    <br/>
    Confirm Email ID:
    <input id="c_email" class="detailsPage" type="text" placeholder="re-enteremail"/>
    <br/>
    <input type="button" value="Submit"/>
```

Use of last child in CSS selector :

Person Details

First Name:

Middle Name:

Last Name:

Email ID:

Confirm Email ID:

FirePath CSS: (X) - body>*:last-child

```

<body>
  <h2> Person Details</h2>
  First Name:
  <input id="username" class="detailsPage" type="text" placeholder="username"/>
  <br/>
  Middle Name:
  <input id="m_name" class="detailsPage" type="text" placeholder="middlename"/>
  <br/>
  Last Name:
  <input id="l_name" class="detailsPage" type="text" placeholder="lastname"/>
  <br/>
  Email ID:
  <input id="e_mail" class="detailsPage" type="text" placeholder="email"/>
  <br/>
  Confirm Email ID:
  <input id="c_email" class="detailsPage" type="text" placeholder="re-enteremail"/>
  <br/>
  <input type="button" value="Submit"/>
</body>
</html>
</document>
  
```

Use of nth-child in CSS selector :

Person Details

First Name:

Middle Name:

Last Name:

Email ID:

Confirm Email ID:

FirePath CSS: (X) - body>div>*:nth-child(1)

```

<document>
  <html>
    <head>
    <body>
      <h2> Person Details</h2>
      <div>
        First Name:
        <input id="username" class="detailsPage" type="text" placeholder="username"/>
        <br/>
        Middle Name:
        <input id="m_name" class="detailsPage" type="text" placeholder="middlename"/>
        <br/>
        Last Name:
        <input id="l_name" class="detailsPage" type="text" placeholder="lastname"/>
        <br/>
        Email ID:
        <input id="e_mail" class="detailsPage" type="text" placeholder="email"/>
        <br/>
        Confirm Email ID:
        <input id="c_email" class="detailsPage" type="text" placeholder="re-enteremail"/>
        <br/>
      </div>
    </body>
  </html>
</document>
  
```

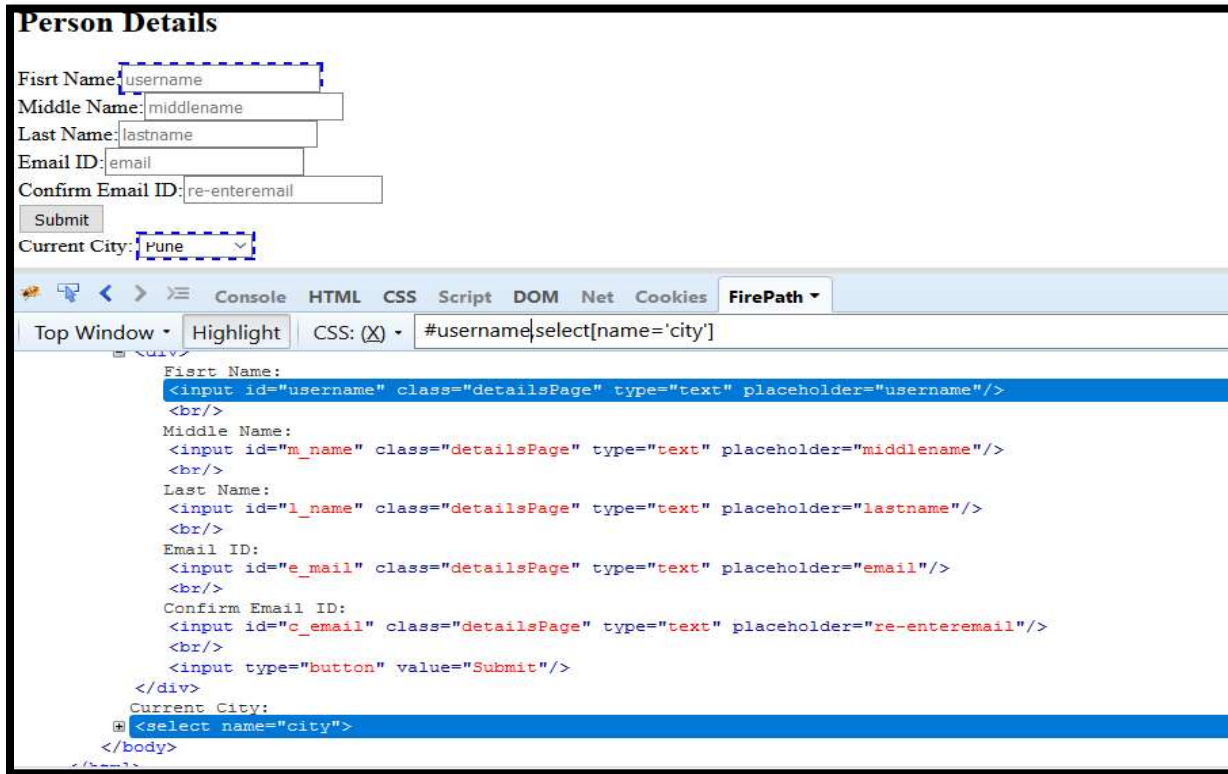
The screenshot shows a web form titled "Person Details" with the following fields: First Name (placeholder: username), Middle Name (placeholder: middlename), Last Name (placeholder: lastname), Email ID (placeholder: email), and Confirm Email ID (placeholder: re-enteremail). A "Submit" button is at the bottom. Below the form is the FirePath DOM tree. The CSS path "body>div>*:nth-child(5)" is entered in the search bar. The DOM tree shows the following structure:

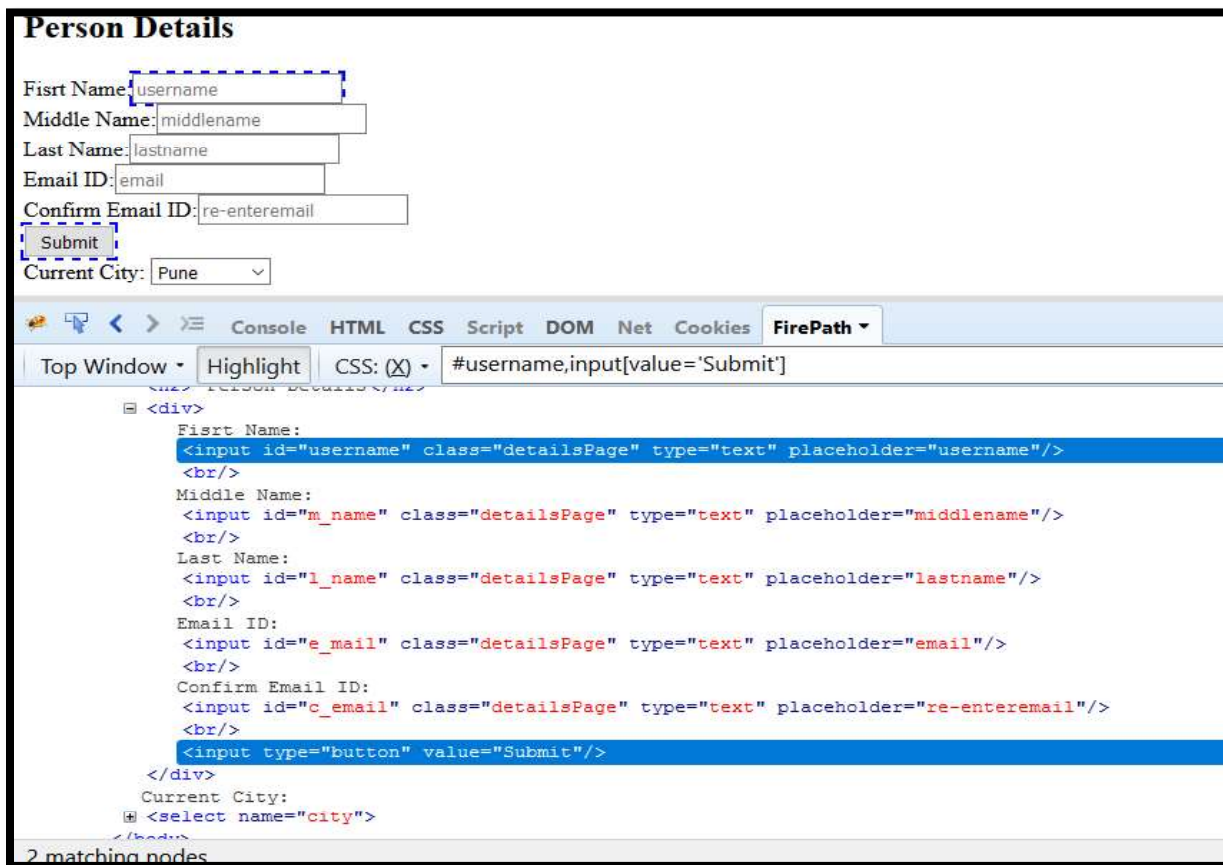
```
<document>
  <html>
    <head>
    <body>
      <h2> Person Details</h2>
      <div>
        First Name:
        <input id="username" class="detailsPage" type="text" placeholder="username"/>
        <br/>
        Middle Name:
        <input id="m_name" class="detailsPage" type="text" placeholder="middlename"/>
        <br/>
        Last Name:
        <input id="l_name" class="detailsPage" type="text" placeholder="lastname"/>
        <br/>
        Email ID:
        <input id="e_mail" class="detailsPage" type="text" placeholder="email"/>
        <br/>
        Confirm Email ID:
        <input id="c_email" class="detailsPage" type="text" placeholder="re-enteremail"/>
        <br/>
      </div>
    </body>
  </html>
</document>
```

So using ***:first-child**, ***:last-child** and ***:nth-child(number)** notations in CSS path, we can locate the child elements of any element.

CSS Selector by comma(,) to select different elements

We can locate different elements using a single CSS path. All we have to do is separate the CSS selectors using comma (i.e.,)





So using **,** notation in CSS path, we can combine two CSS paths into single CSS path. As a result we can locate different types of elements using a single/combined CSS path.

^ - Starts with

CSS Selector Rule - [attribute^=attributeValue]

Example -

For the Sample HTML below-

```
<button id="user1_btn_263" type="button" class="btn">Submit</button>
```

CSS Locator - id^="user1"

Description - 'id^="user1"' will select the element whose id starts with "user1" value

\$ - Ends with

CSS Selector Rule - [attribute\$=attributeValue]

Example -

For the Sample HTML below-

```
<button id="user1_btn_263" type="button" class="btn">Submit</button>
```

CSS Locator - id\$="btn_263"

Description - 'id\$="btn_263"' will select the element whose id ends with "btn_263" value

* - Contains

*

is used in CSS path to locate all the elements.

Example:

* -> All the elements of the web page (i.e. from root element **html** to the grand child elements) will be highlighted/located.

head > * -> All the elements of the web page under the **head** node will get highlighted/located.

body > * -> All the elements of the web page under the **body** node will get highlighted/located.

CSS Selector Rule - [attribute*=attributeValue]

Example -

For the Sample HTML below-

```
<button id="user1_btn_263" type="button" class="btn">Submit</button>
```

CSS Locator - id*="btn"

Description - 'id*="btn"' will select the element whose id contains with "btn" value

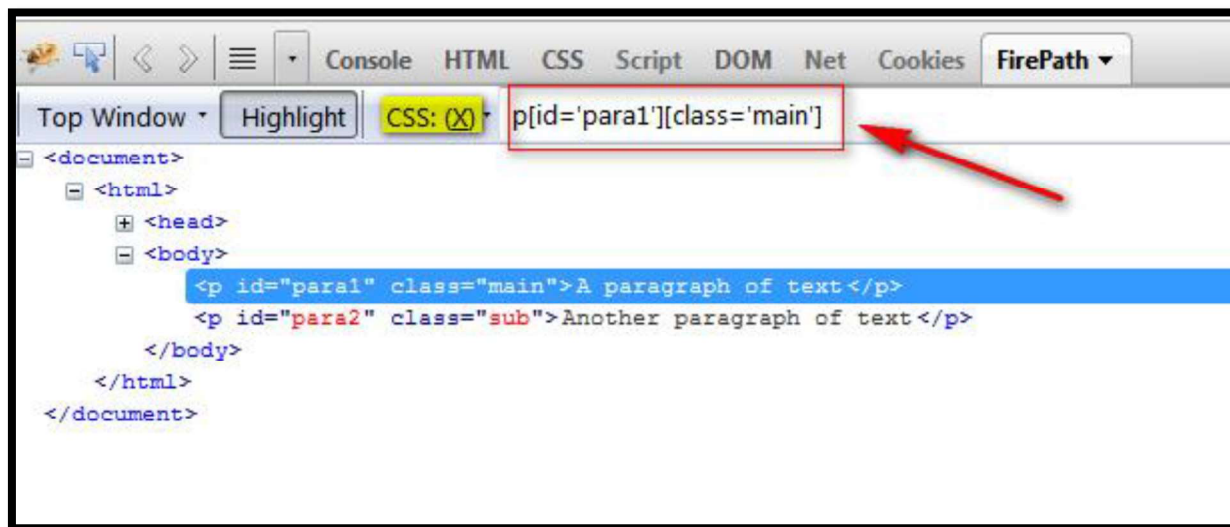
Using Boolean Operators in CSS selector

We can use the boolean operators in CSS selectors by:

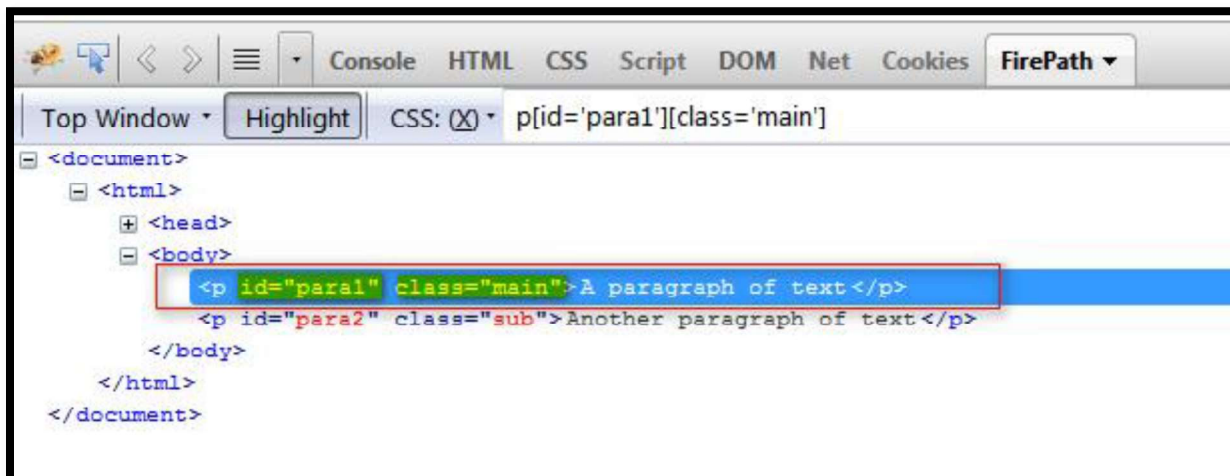
- adding **more conditions** (Example: **p[id='para1'][class='main']** - Locates the paragraph elements which have id attribute value as 'para1' and class attribute value as 'main').
- adding **:not** to the conditions (Example: **p:not(p[id='para1'])** - Locates all the paragraph elements which don't have the id attribute value specified as 'para1')

First lets add **more conditions** to the CSS path by following the below steps:

1. Open http://compendiumdev.co.uk/selenium/basic_web_page.html in Firefox Browser
2. Click on the 'Firepath' tab, select the CSS option, enter the CSS path **p[id='para1'][class='main']** into the text box as shown below and click on 'Eval' button:

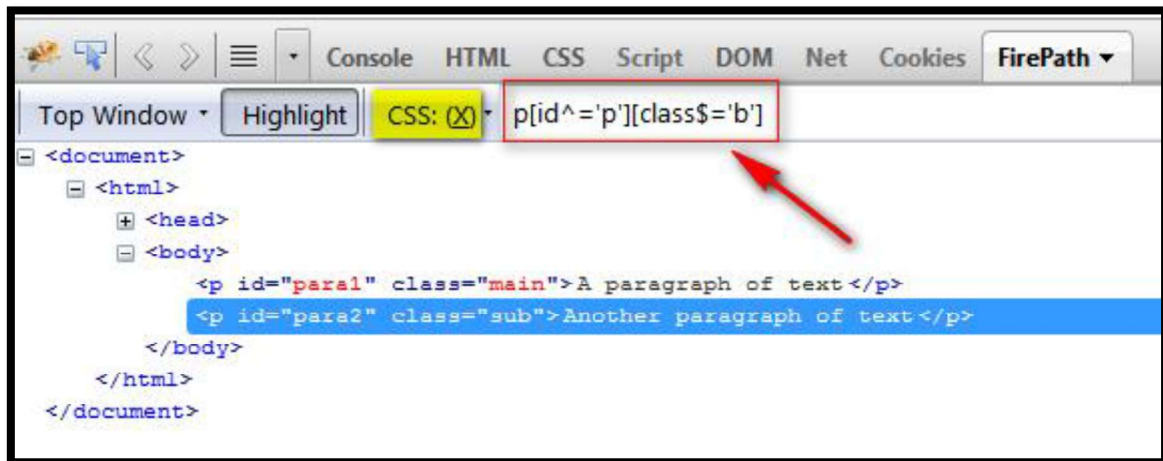


3. Observe that all the paragraph elements having the id attribute value as 'para1' and the class attribute value as 'main' got highlighted as shown below:

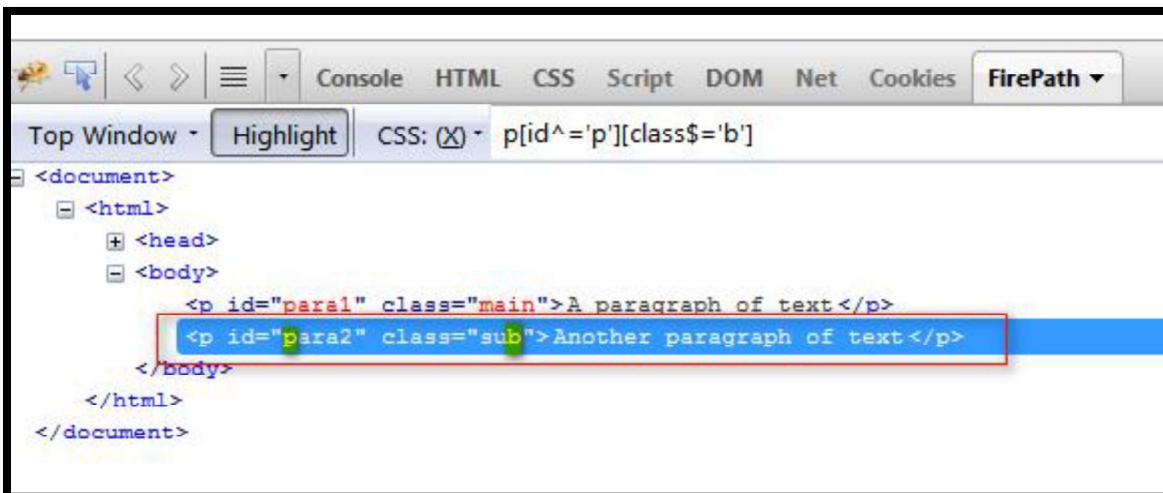


4. Now lets enter the CSS path `p[id^='p'][class$='b']` into the text box as shown below and click on 'Eval' button:





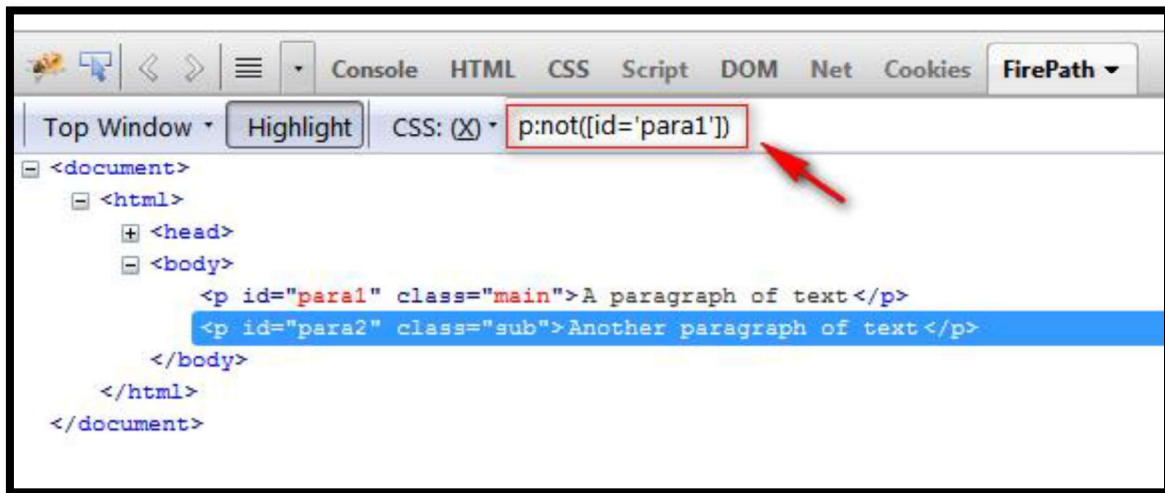
5. Observe that all the paragraph elements having the id attribute value starting with 'p' text and the class attribute value ending with 'b' text got highlighted as shown below:



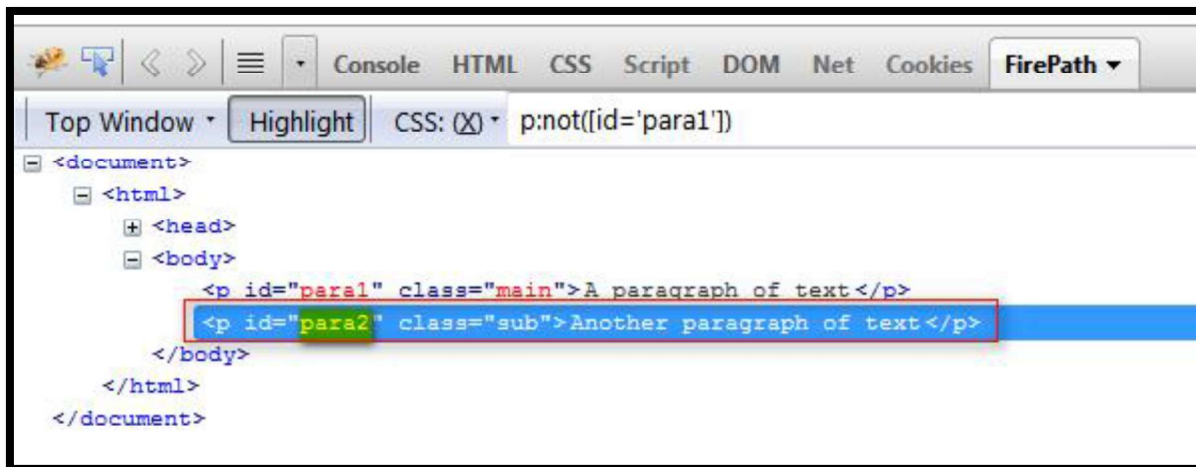
Now lets add :not to the conditions in the CSS path by following the below steps:

1. Open http://compendiumdev.co.uk/selenium/basic_web_page.html in Firefox Browser
2. Click on the 'Firepath' tab, select the CSS option, enter the CSS path `p:not([id='para1'])` into the text box as shown below and click on 'Eval' button:

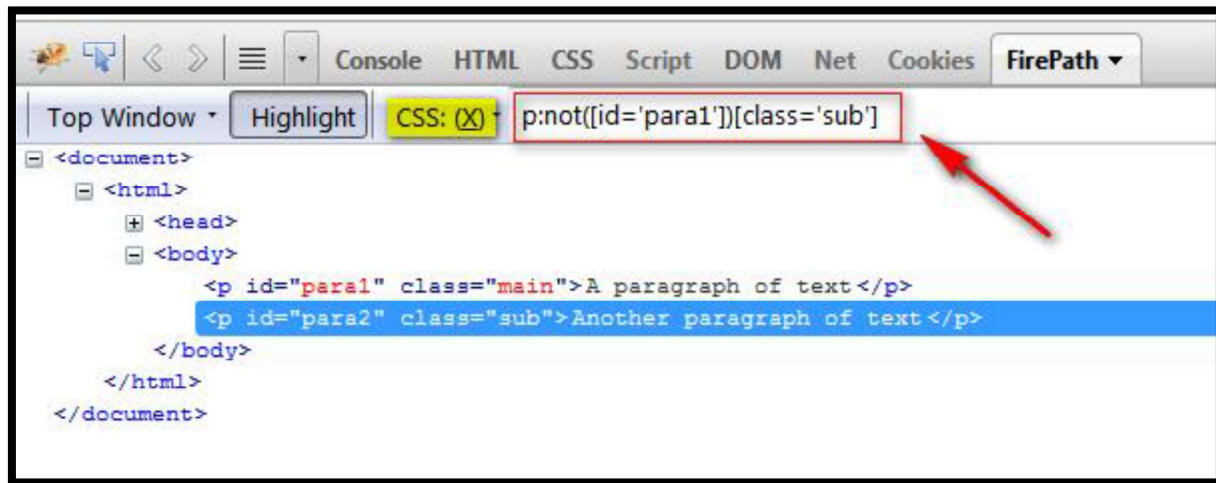
SV



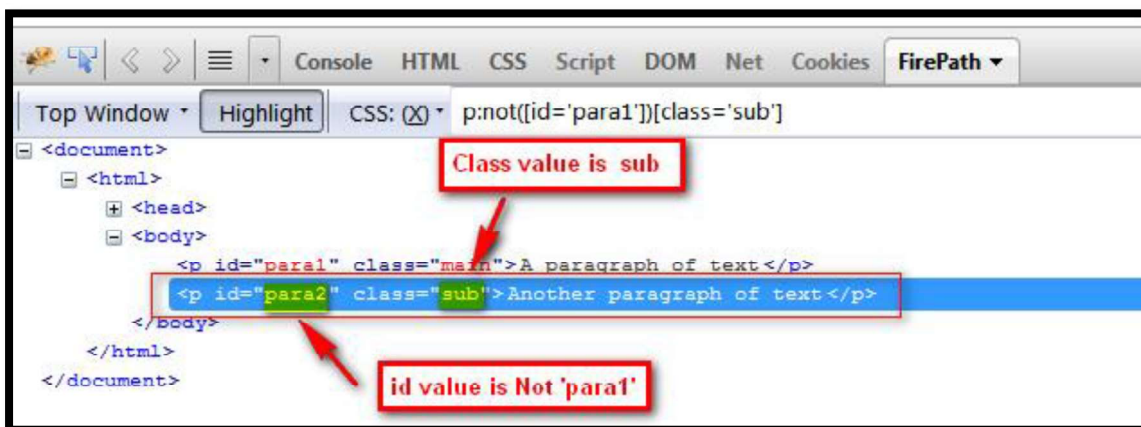
3. Observe that all the paragraph elements **not** having the id attribute value as 'para1' text got highlighted as shown below:



4. Now let's identify the paragraph elements **not having** the id attribute value as 'para1' and **having** the class attribute value as 'sub'. So enter the CSS path `p:not([id='para1'])[class='sub']` into the text box as shown below and click on 'Eval' button:

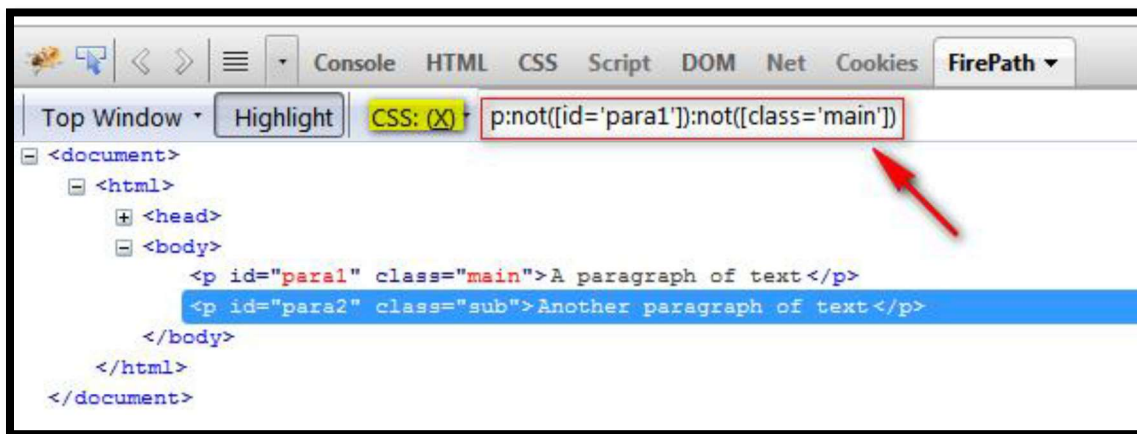


5. Observe that all the paragraph elements not having the id attribute value as 'para1' and having the class attribute value as 'sub' got highlighted as shown below:

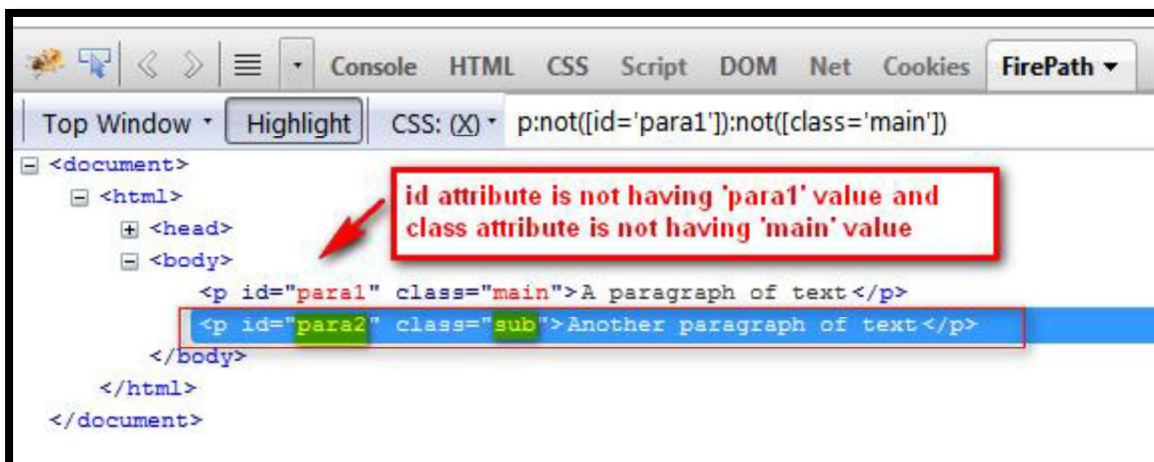


6. Now let's identify the paragraph elements **not having** the id attribute value as 'para1' and also **not having** the class attribute value as 'main'. So enter the CSS path `p:not([id='para1']):not([class='main'])` into the text box as shown below and click on 'Eval' button:

SE



7. Observe that all the paragraph elements not having the id attribute value as 'para1' and not having the class attribute value as 'main' got highlighted as shown below:



That's it with the boolean operators.

CSS selector Optimization

There may be cases where simple changes are made to the application and our previously identified CSS statements won't work after those simple changes are made to the application. In order to avoid these kind of problems, we've to optimize our CSS selector Statements before using them in our Selenium Automation.

So that Advantages of Optimizing the CSS Statements are to get the **shortest and least breakable** CSS selector Statements.

We've to follow the below three strategies in order to make the CSS statements Optimized:

1. Use the **id** attribute if available but not used

2. Use the combination of attributes to make the CSS more specific
3. Use the Relative CSS instead of Absolute CSS Statements

Locating Siblings

CSS Selector Rule - locator1+locator2

Example -

For the Sample HTML below-

```
<ul id="testingTypes">
  <li id="automation">Automation Testing</li>
  <li>Performance Testing</li>
  <li>Manual Testing</li>
</ul>
```

CSS Locator - li#automation + li

Description - 'li#automation + li' will first go to li element with id 'automation' and then select its adjacent li i.e. 'Performance Testing' list item.

For handling dynamic elements having ids and other locators dynamically generated(not known beforehand). We can make use of the above locators by using different parent-sibling relationships of the dynamic elements. Apart from this, we can also use some special CSS locators using which we can match partial values of the attributes.

CSS Selector using Parent (indirect child)

Consider the following HTML

```
<div id="ParentDiv">
  <input class="txtclass" value="Text" type="text"/>
  <button class="btnclass">Button</button>
  <a class="anchorclass" id="anchor" href="#"> My Link </a>
</div>
```

The above HTML contains our beloved text box, button and anchor again. But this time inside a parent element.

The syntax to select element using its parent is

syntax = AnyParentSelector space

Anychildselector example.

div#parentdiv input.txtclass

or

div#parentdiv

a[id=anchor]

Code in

WebDriver:

```
var ElementA = driver.FindElement(By.CssSelector("div#parentdiv input.txtclass"));
```

```
var ElementB = driver.FindElement(By.CssSelector("div#parentdiv id=anchor"));
```

CSS Selector using Parent (direct child)

Consider the following HTML

```
<div id="ParentDiv">
  <input class="txtclass" value="Text" type="text"/>
  <button class="btnclass">Button</button>
  <a class="anchorclass" id="anchor" href="#"> My Link </a>
  <div id="ChildDiv">
    <input class="txtclass" value="Text" type="text"/>
    <button class="btnclass">Button</button>
    <a class="anchorclass" id="anchor" href="#"> My Link </a>
  </div>
</div>
```

The above HTML now contains two divs. If we want to select a child element using indirect child method, it will return us more than one occurrence of child.

example

`div#parentdiv input.txtclass` (This will return two text boxes with class "txtclass". As both these text boxes are

the children of "parentDiv". One is the direct child and one is indirect child.

Now if we want to access the direct child, we should use ">" symbol instead of space.

example

`div#parentdiv > input.txtclass` (this will return the first textbox which is the direct child of parentdiv

Code in WebDriver:

```
var ElementA = driver.FindElement(By.CssSelector("div#parentdiv > input.txtclass"));
var ElementB = driver.FindElement(By.CssSelector("div#parentdiv > id=anchor"])
```

Choosing a specific match

CSS selectors in Selenium allow us to navigate lists with more finesse than the above methods. If we have a ul and we want to select its fourth li element without regard to any other elements, we should use nth-child or nth-of-type.

```
<p>Heading</p>
```

```
Cat
```

```
Dog
```

```
Car
```

```
Goat
```

If we want to select the fourth li element (Goat) in this list, we can use the nth-of-type, which will find the fourth li in the list.

```
css=ul#recordlist li:nth-of-type(4)
```

On the other hand, if we want to get the fourth element only if it is a li element, we can use a filtered nth-child which will select (Car) in this case.

```
css=ul#recordlist li:nth-child(4)
```

Note, if you don't specify a child type for nth-child it will allow you to select the fourth child without regard to type. This may be useful in testing CSS layout in Selenium.

```
css=ul#recordlist *:nth-child(4)
```

Getting the last child from html hierarchy

```
css="ul#fruit li:last-child"
```

Attribute NOT contain a specific value

In WebDriver, how do you find elements whose attribute contain values which you don't want to select? This CSS selector example shows how NOT to select by specific attribute value

Suppose you have many elements which share the same attribute and attribute value, but some of those elements have other variables appended to the value. e.g:

```
<div class="day past calendar-day-2017-02-13 calendar-dow-1 unavailable">
<div class="day today calendar-day-2017-02-14 calendar-dow-2 unavailable">
<div class="day calendar-day-2017-02-15 calendar-dow-3">
<div class="day calendar-day-2017-02-16 calendar-dow-4">
```

In the above snippet, we want to select an available day (i.e. the two last divs)

As can be seen, all four divs contain "calendar-day-" but the first two also contain "unavailable" which we don't want.

The CSS selector for Not selecting the first two divs is

```
css = "div[class*=calendar-day-]:not([class*='unavailable'])"
```

DIRECT CHILD

A direct child in XPATH is defined by the use of a "/", while on CSS, it's defined using ">"

Examples:

```
//div/a
```

```
css=div > a
```

CHILD OR SUBCHILD

If an element could be inside another or one of its children, it's defined in XPATH using "/" and in