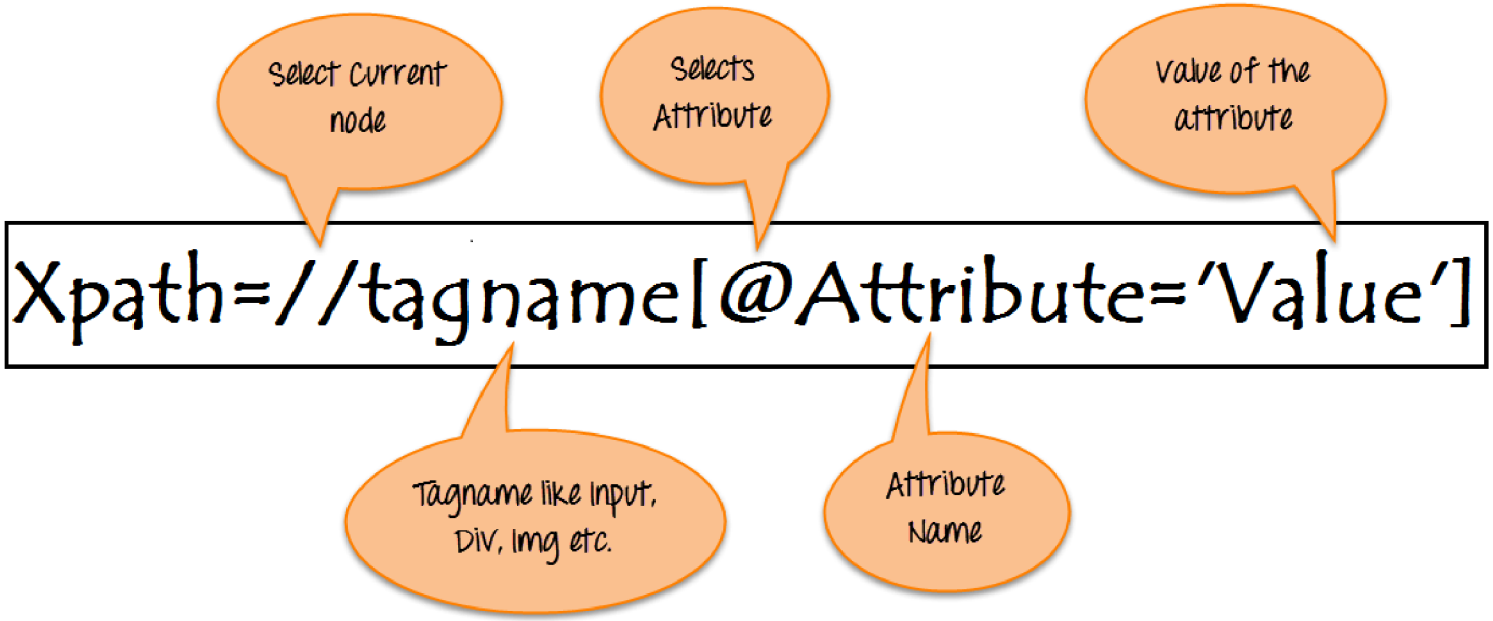
What is XPath?

XPath is defined as **XML path**. **It is a syntax or language for finding any element on the web page using XML path expression**. XPath is used to find the location of any element on a webpage using HTML DOM structure. The basic format of XPath is explained below with screen shot.



(/images/3-2016/032816_0758_XPathinSele1.png)

Syntax for XPath:

XPath contains the path of the element situated at the web page. Standard syntax for creating XPath is.

```
Xpath=//tagname[@attribute='value']
```

- **// :** Select current node.
- **Tagname:** Tagname of the particular node.
- **@:** Select attribute.
- **Attribute:** Attribute name of the node.
- **Value:** Value of the attribute.

To find the element on web pages accurately there are different types of locators:

XPath Locators	Find different elements on web page
ID	To find the element by ID of the element
Classname	To find the element by Classname of the element
Name	To find the element by name of the element
Link text	To find the element by text of the link
XPath	XPath required for finding the dynamic element and traverse between various elements of the web page
CSS path	CSS path also locates elements having no name, class or ID.

Types of X-path

There are two types of XPath:

1) Absolute XPath

2) Relative XPath

Absolute XPath:

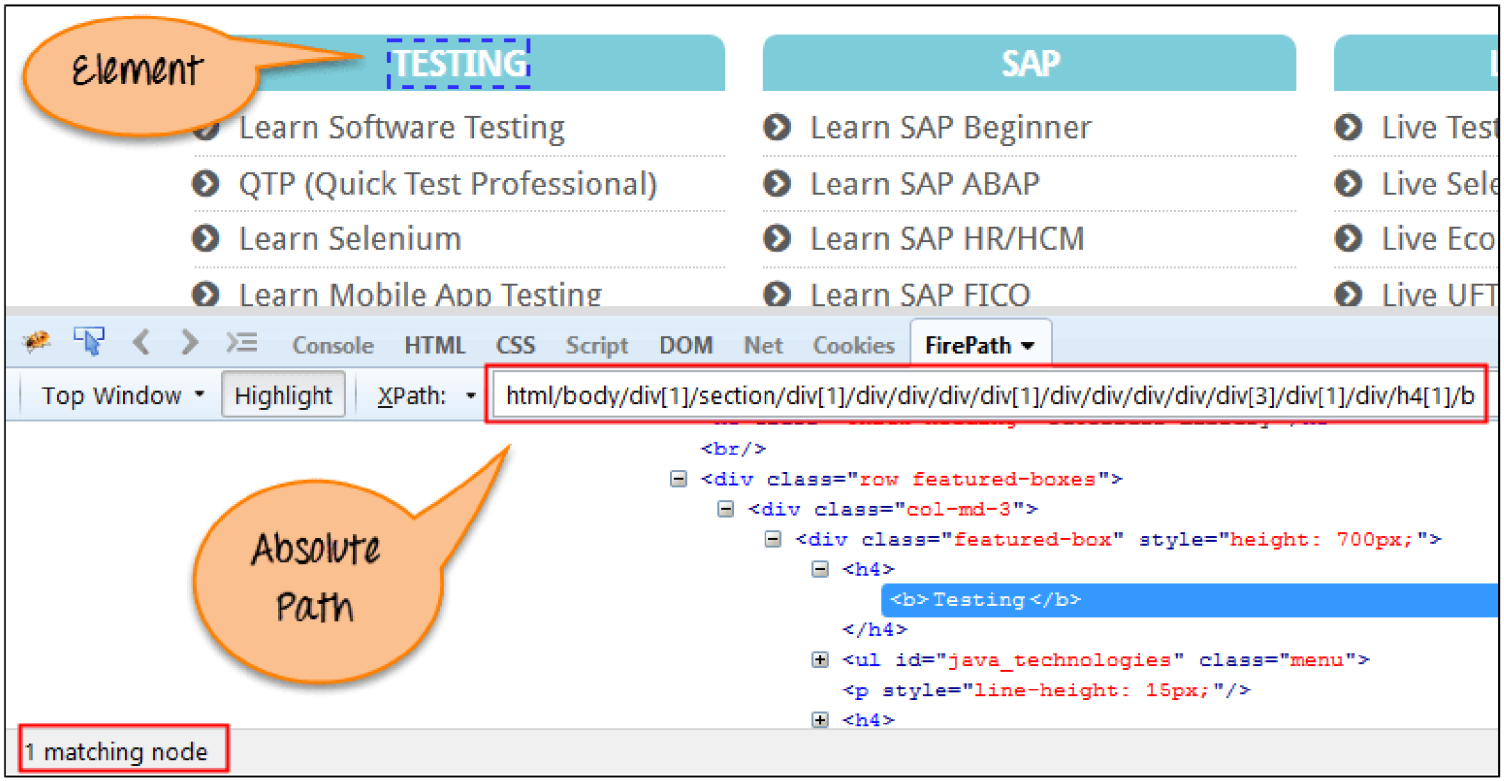
It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.

The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node.

Below is the example of an absolute xpath expression of the element shown in the below screen.

Absolute xpath:

```
html/body/div[1]/section/div[1]/div/div/div/div[1]/div/div/div/div/div[3]/div[1]/div/h4[1]/b
```



(/images/3-2016/032816_0758_XPathinSele2.png)

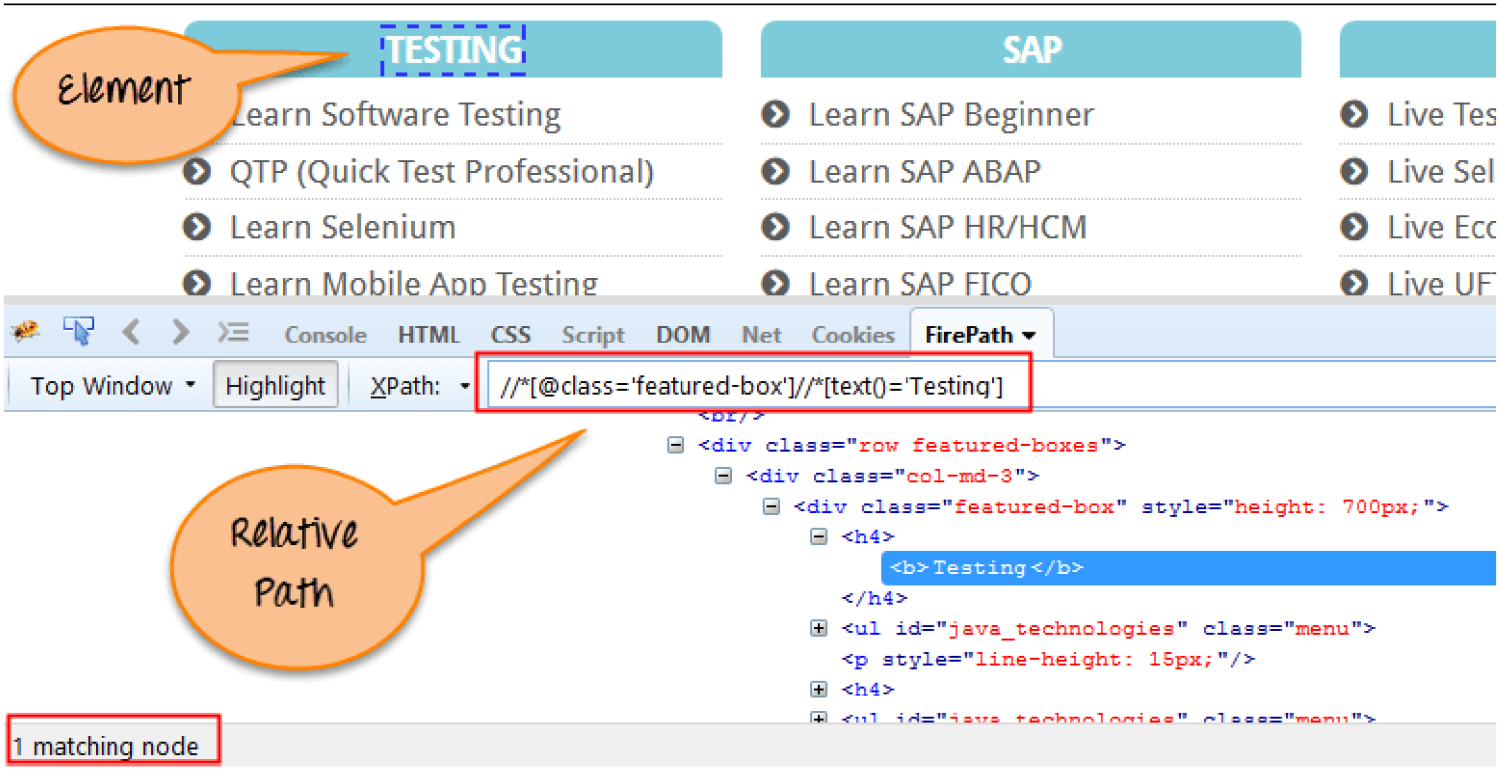
Relative xpath:

For Relative Xpath the path starts from the middle of the HTML DOM structure. It starts with the double forward slash (//), which means it can search the element anywhere at the webpage.

You can start from the middle of the HTML DOM structure and no need to write long xpath.

Below is the example of a relative XPath expression of the same element shown in the below screen. This is the common format used to find element through a relative XPath.

```
Relative xpath: //*[@class='featured-box']//*[text()='Testing']
```



(/images/3-2016/032816_0758_XPathinSele3.png)

What are XPath axes.

XPath axes search different nodes in XML document from current context node. XPath Axes are the methods used to find dynamic elements, which otherwise not possible by normal XPath method having no ID , Classname, Name, etc.

Axes methods are used to find those elements, which dynamically change on refresh or any other operations. There are few axes methods commonly used in Selenium Webdriver like child, parent, ancestor, sibling, preceding, self, etc.

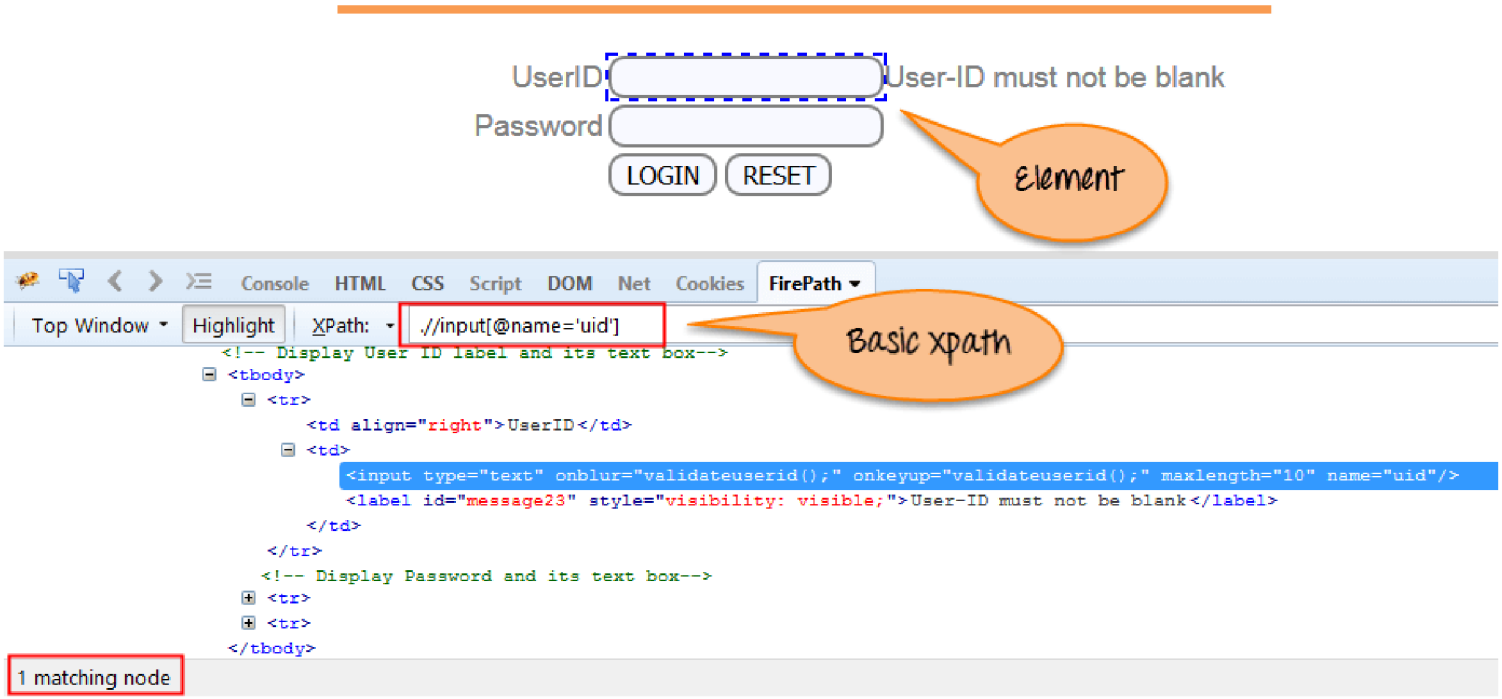
Using XPath Handling complex & Dynamic elements in Selenium

1) Basic XPath:

XPath expression select nodes or list of nodes on the basis of attributes like **ID , Name, Classname**, etc. from the XML document as illustrated below.

```
Xpath=//input[@name='uid']
```

Here is a link to access the page http://demo.guru99.com/v1/ (http://demo.guru99.com/v1/)



(/images/3-2016/032816_0758_XPathinSele4.png)

Some more basic xpath expressions:

```
Xpath=//input[@type='text']
Xpath=  //label[@id='message23']
Xpath=  //input[@value='RESET']
Xpath=//*[@class='barone']
Xpath=//a[@href='http://demo.guru99.com/']
Xpath= //img[@src='//cdn.guru99.com/images/home/java.png']
```

2) Contains():

Contains() is a method used in XPath expression. It is used when the value of any attribute changes dynamically, for example, login information.

The contain feature has an ability to find the element with partial text as shown in below example.

In this example, we tried to identify the element by just using partial text value of the attribute. In the below XPath expression partial value 'sub' is used in place of submit button. It can be observed that the element is found successfully.

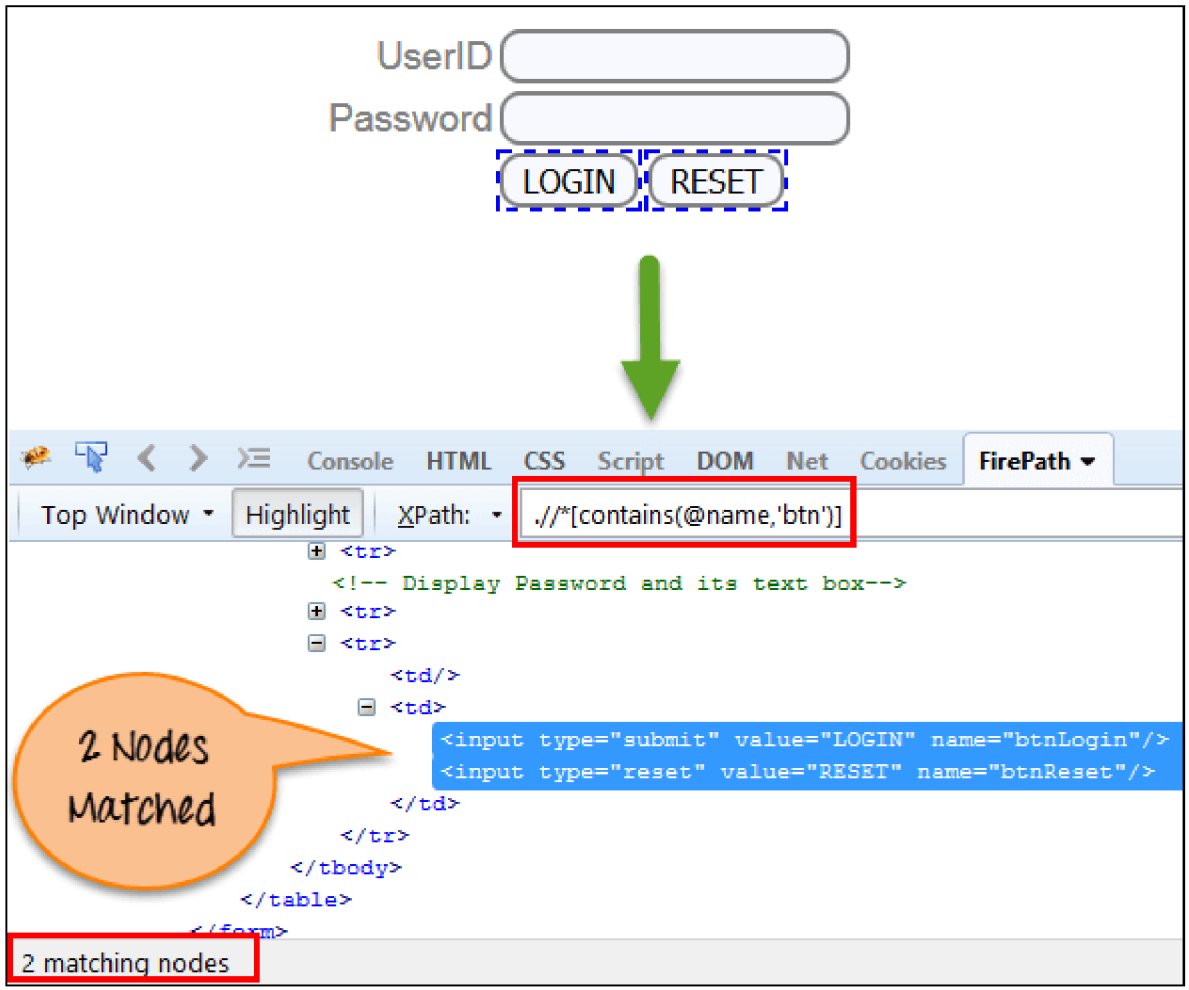
Complete value of 'Type' is 'submit' but using only partial value 'sub'.

```
Xpath=//*[contains(@type,'sub')]
```

Complete value of 'name' is 'btnLogin' but using only partial value 'btn'.

```
Xpath=//*[contains(@name,'btn')]
```

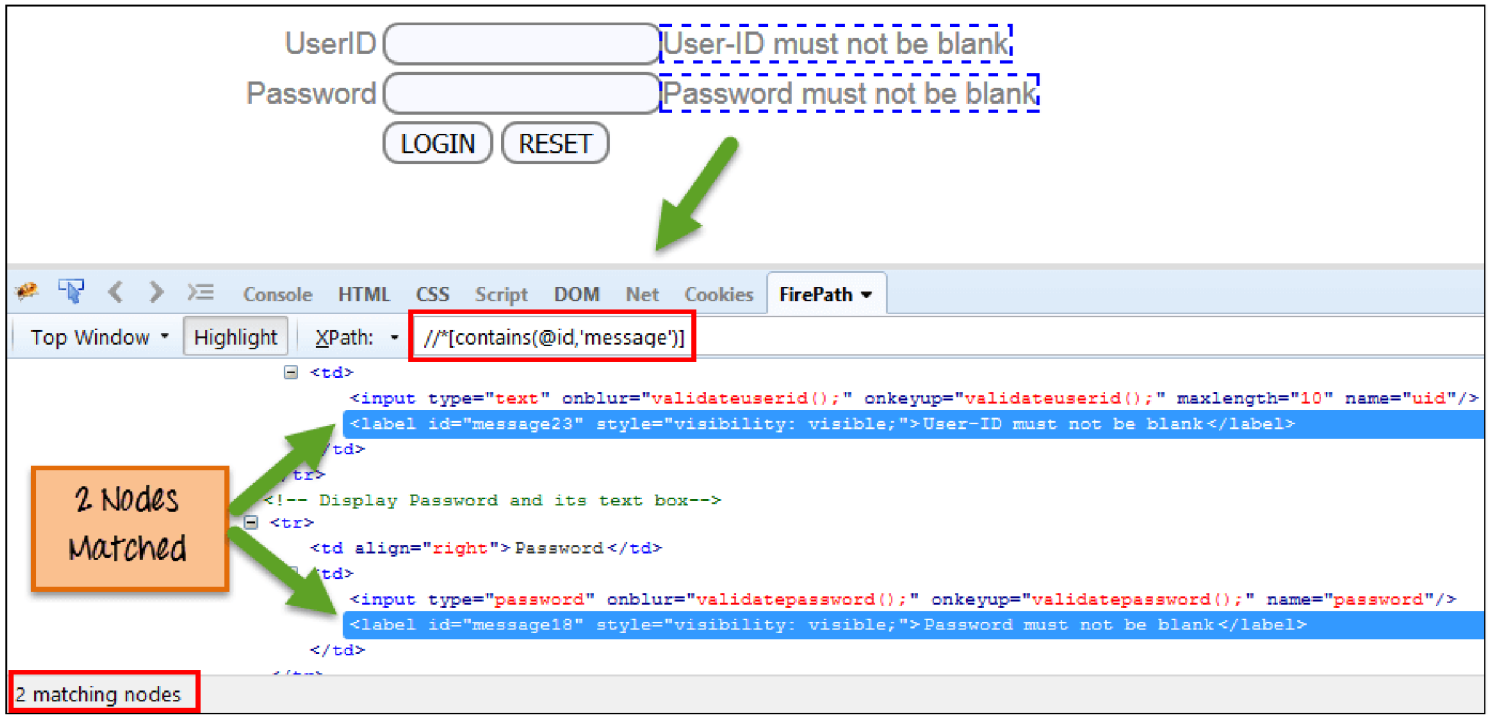
In the above expression, we have taken the 'name' as an attribute and 'btn' as an partial value as shown in the below screenshot. This will find 2 elements (LOGIN & RESET) as their 'name' attribute begins with 'btn'.



(/images/3-2016/032816_0758_XPathinSele5.png)

Similarly, in the below expression, we have taken the 'id' as an attribute and 'message' as a partial value. This will find 2 elements ('User-ID must not be blank' & 'Password must not be blank') as its 'name' attribute begins with 'message'.

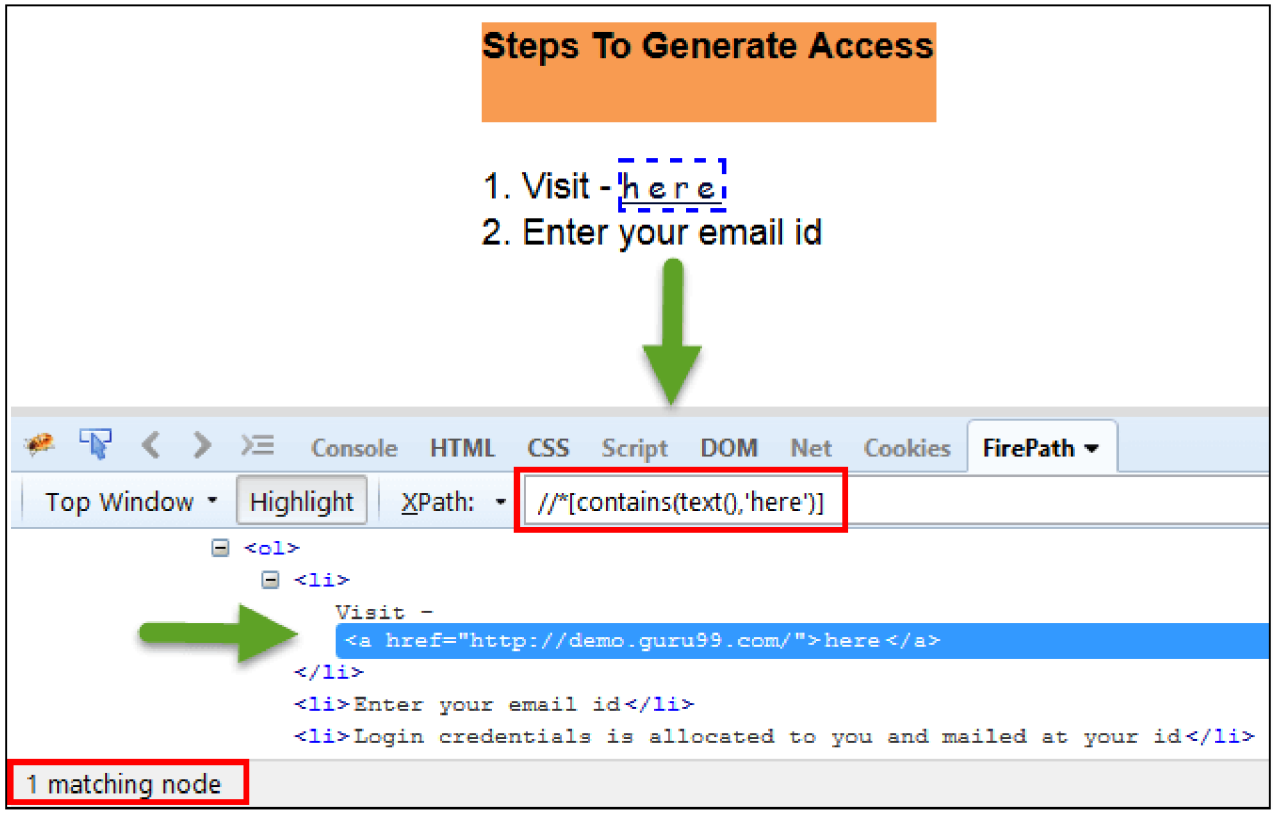
```
Xpath=//*[contains(@id,'message')]
```



(/images/3-2016/032816_0758_XPathinSele6.png)

In the below expression, we have taken the "text" of the link as an attribute and 'here' as a partial value as shown in the below screenshot. This will find the link ('here') as it displays the text 'here'.

```
Xpath=//*[contains(text(),'here')]
Xpath=//*[contains(@href,'guru99.com')]
```



(/images/3-2016/032816_0758_XPathinSele7.png)

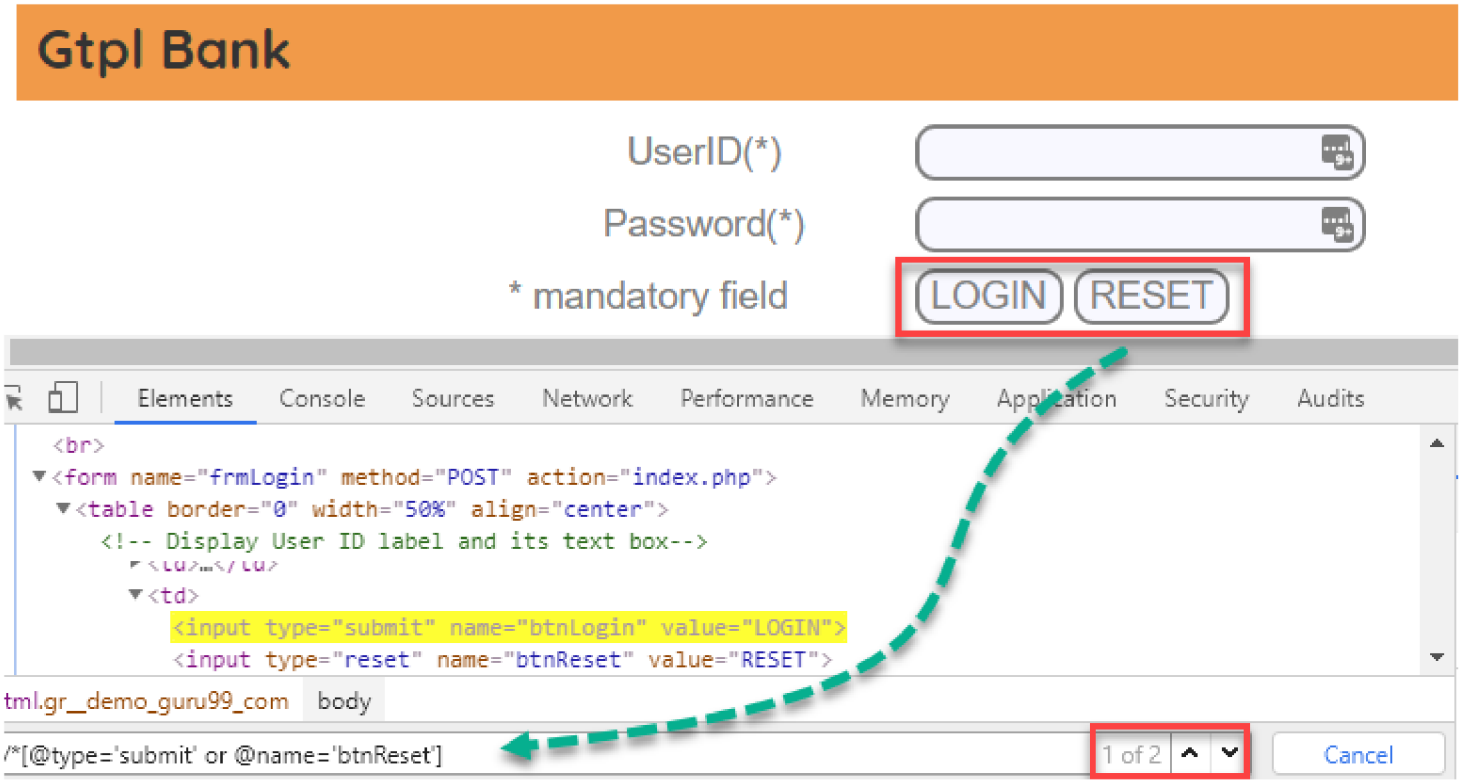
3) Using OR & AND:

In OR expression, two conditions are used, whether 1st condition OR 2nd condition should be true. It is also applicable if any one condition is true or maybe both. Means any one condition should be true to find the element.

In the below XPath expression, it identifies the elements whose single or both conditions are true.

```
Xpath=//*[@type='submit' or @name='btnReset']
```

Highlighting both elements as "LOGIN " element having attribute 'type' and "RESET" element having attribute 'name'.

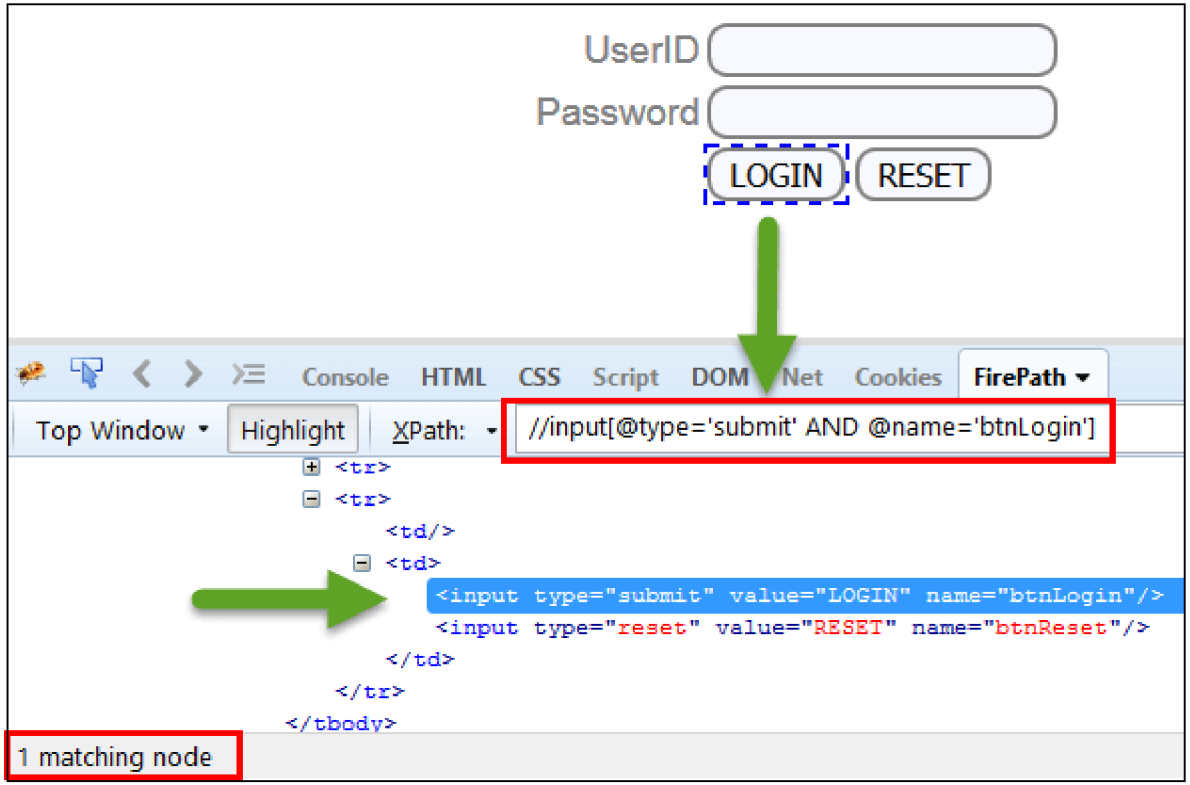


(/images/3-2016/032816_0758_XPathinSele8.png)

In AND expression, two conditions are used, both conditions should be true to find the element. It fails to find element if any one condition is false.

```
Xpath=//input[@type='submit' and @name='btnLogin']
```

In below expression, highlighting 'LOGIN' element as it having both attribute 'type' and 'name'.



(/images/3-2016/032816_0758_XPathinSele9.png)

4) Start-with function:

Start-with function finds the element whose attribute value

changes on refresh or any operation on the webpage. In this expression, match the starting text of the attribute is used to find the element whose attribute changes dynamically. You can also find the element whose attribute value is static (not changes).

For example -: Suppose the ID of particular element changes dynamically like:

Id=" message12"

Id=" message345"

Id=" message8769"

and so on.. but the initial text is same. In this case, we use Start-with expression.

In the below expression, there are two elements with an id starting "message"(i.e., 'User-ID must not be blank' & 'Password must not be blank'). In below example, XPath finds those element whose 'ID' starting with 'message'.

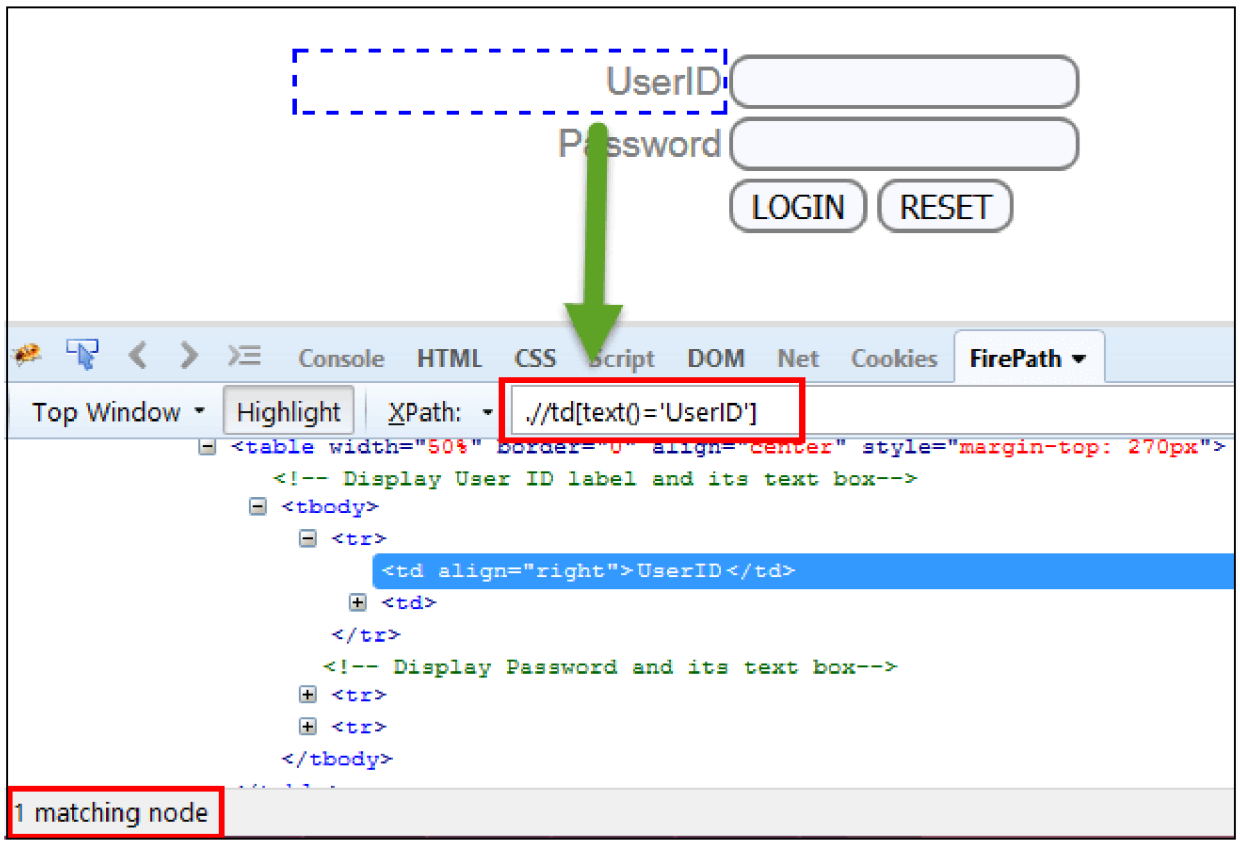
```
Xpath=//label[starts-with(@id,'message')]
```

(/images/3-2016/032816_0758_XPathinSele10.png)

5) Text():

In this expression, with text function, we find the element with exact text match as shown below. In our case, we find the element with text "UserID".

```
Xpath=//td[text()='UserID']
```



(/images/3-2016/032816_0758_XPathinSele11.png)

6) XPath axes methods:

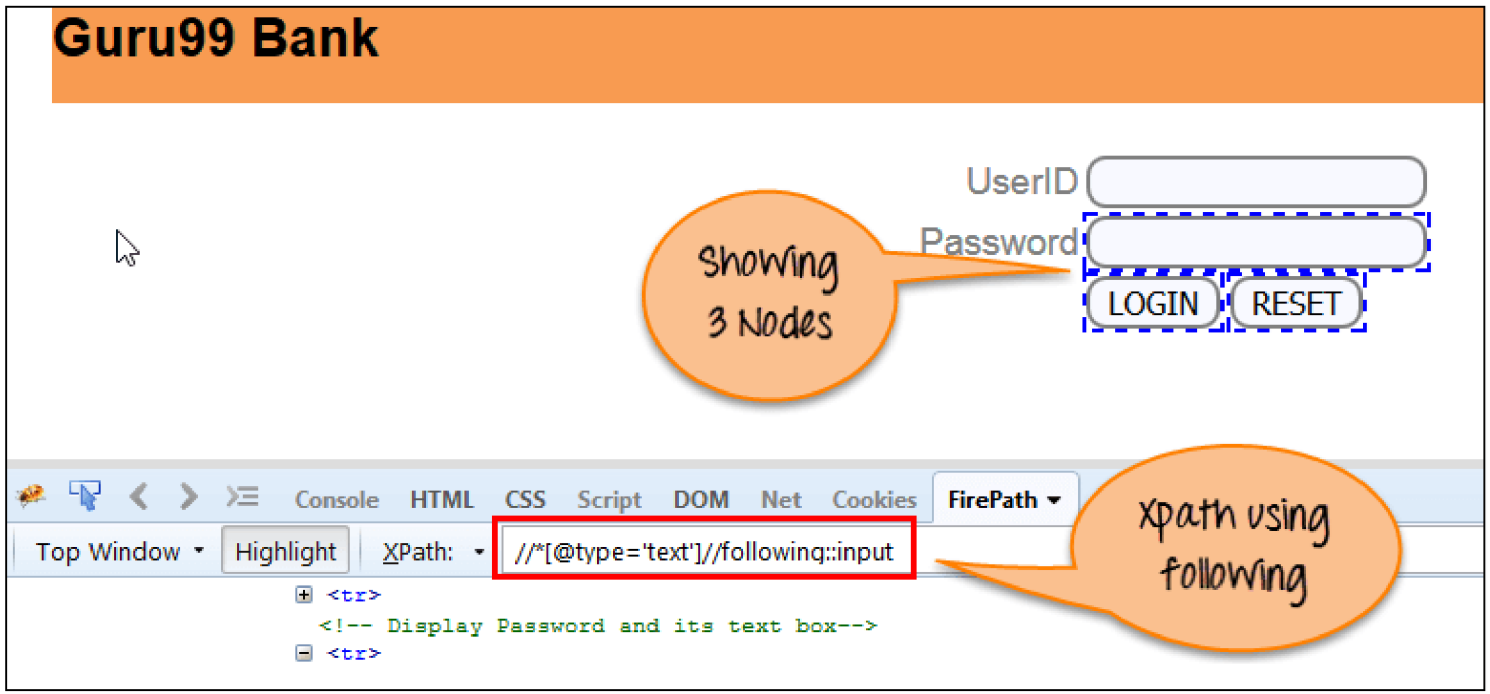
These XPath axes methods are used to find the complex or dynamic elements. Below we will see some of these methods.

For illustrating these XPath axes method, we will use the Guru99 bank demo site.

a) Following:

Selects all elements in the document of the current node( ) [UserID input box is the current node] as shown in the below screen.

```
Xpath=//*[@type='text']//following::input
```



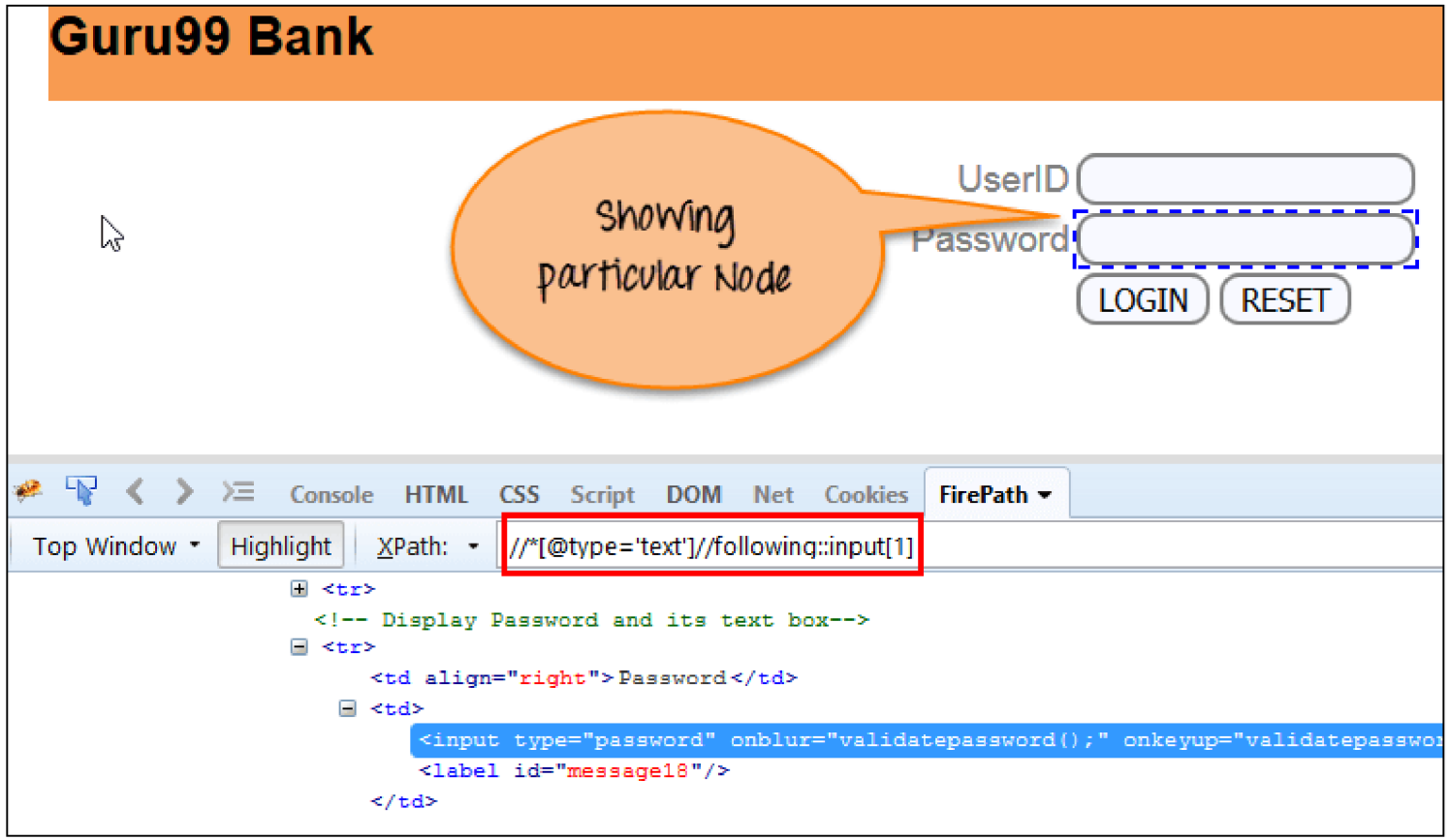
(/images/3-2016/032816_0758_XPathinSele12.png)

There are 3 "input" nodes matching by using "following" axis- password, login and reset button. If you want to focus on any particular element then you can use the below XPath method:

```
Xpath=//*[@type='text']//following::input[1]
```

You can change the XPath according to the requirement by putting [1],[2]……and so on.

With the input as '1', the below screen shot finds the particular node that is 'Password' input box element.



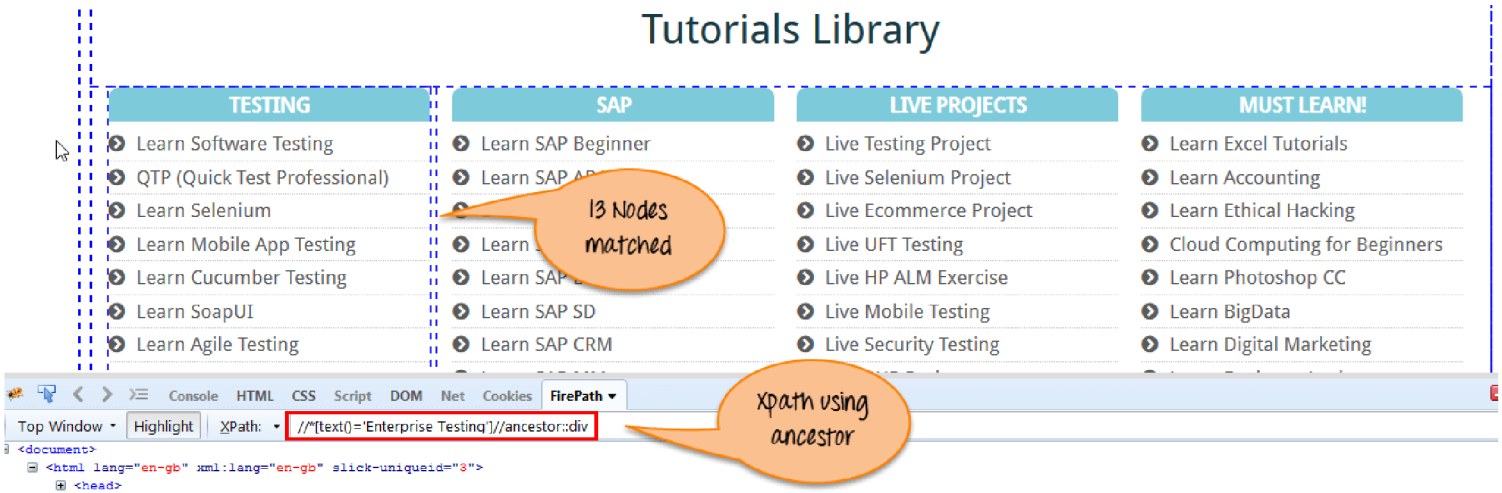
(/images/3-2016/032816_0758_XPathinSele13.png)

b) Ancestor:

The ancestor axis selects all ancestors element (grandparent, parent, etc.) of the current node as shown in the below screen.

In the below expression, we are finding ancestors element of the current node("ENTERPRISE TESTING" node).

```
Xpath=//*[text()='Enterprise Testing']//ancestor::div
```



(/images/3-2016/032816_0758_XPathinSele14.png)

There are 13 "div" nodes matching by using "ancestor" axis. If you want to focus on any particular element then you can use the below XPath, where you change the number 1, 2 as per your requirement:

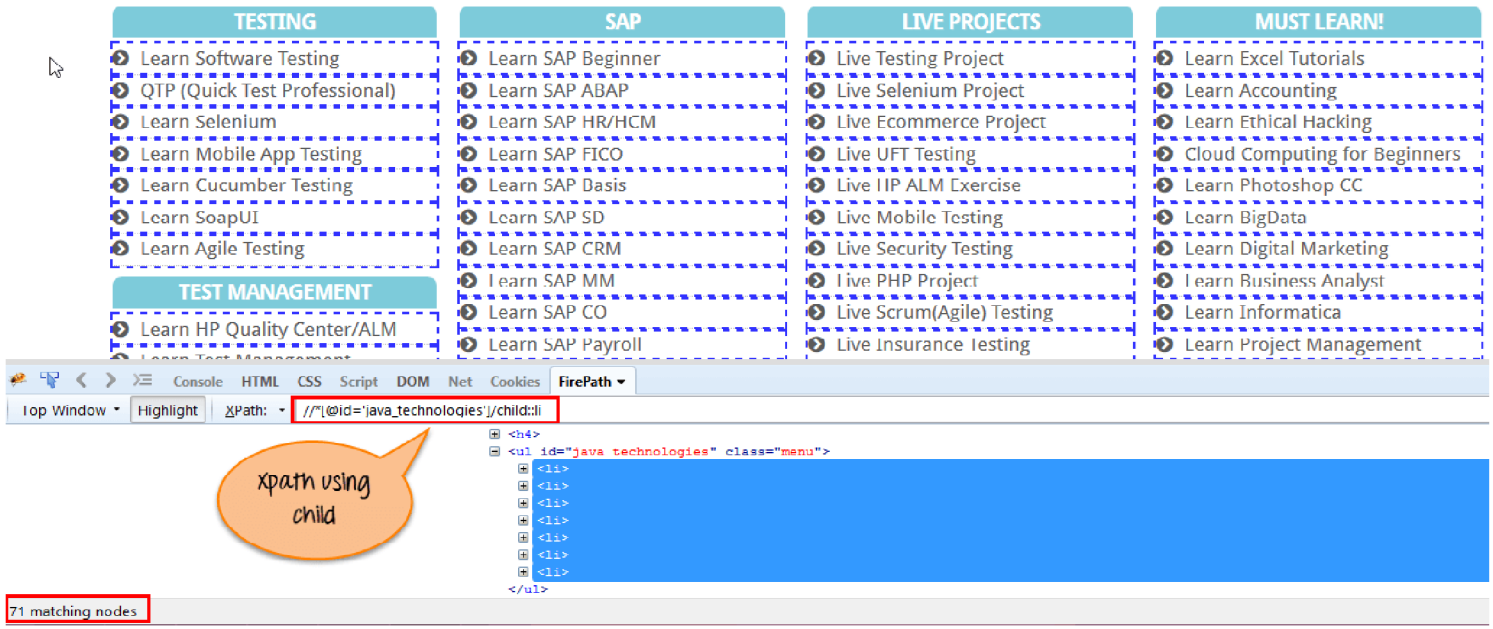
```
Xpath=//*[text()='Enterprise Testing']//ancestor::div[1]
```

You can change the XPath according to the requirement by putting [1], [2]……and so on.

c) Child:

Selects all children elements of the current node (Java) as shown in the below screen.

```
Xpath=//*[@id='java_technologies']/child::li
```



(/images/3-2016/032816_0758_XPathinSele15.png)

There are 71 "li" nodes matching by using "child" axis. If you want to focus on any particular element then you can use the below xpath:

```
Xpath=//*[@id='java_technologies']/child::li[1]
```

You can change the xpath according to the requirement by putting [1],[2]……and so on.

d) Preceding:

Select all nodes that come before the current node as shown in the below screen.

In the below expression, it identifies all the input elements before "LOGIN" button that is **Userid** and **password** input element.

```
Xpath=//*[@type='submit']//preceding::input
```

(/images/3-2016/032816_0758_XPathinSele16.png)

There are 2 "input" nodes matching by using "preceding" axis. If you want to focus on any particular element then you can use the below XPath:

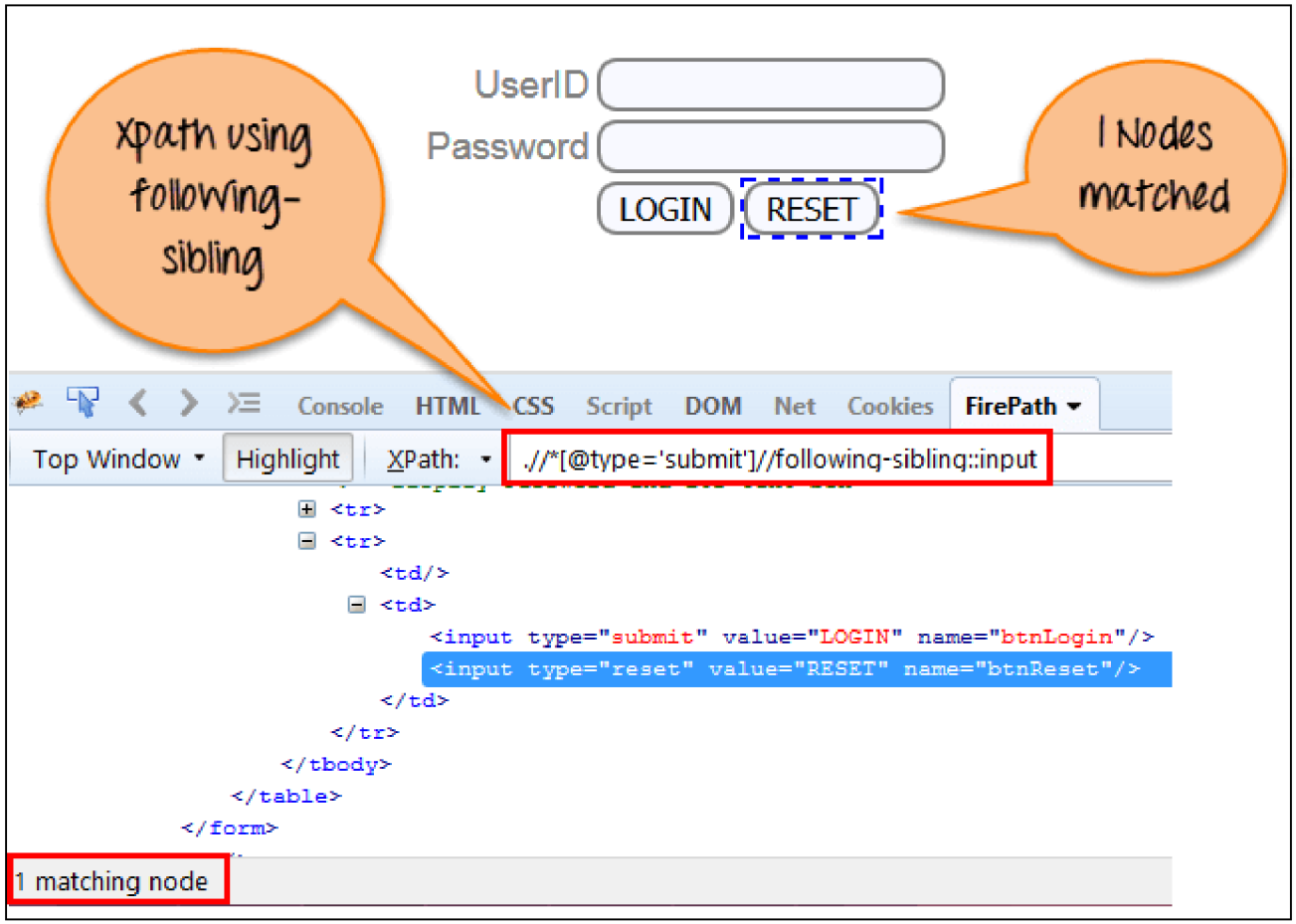
```
Xpath=//*[@type='submit']//preceding::input[1]
```

You can change the xpath according to the requirement by putting [1],[2]……and so on.

e) Following-sibling:

Select the following siblings of the context node. Siblings are at the same level of the current node as shown in the below screen. It will find the element after the current node.

```
 xpath=//*[@type='submit']//following-sibling::input
```



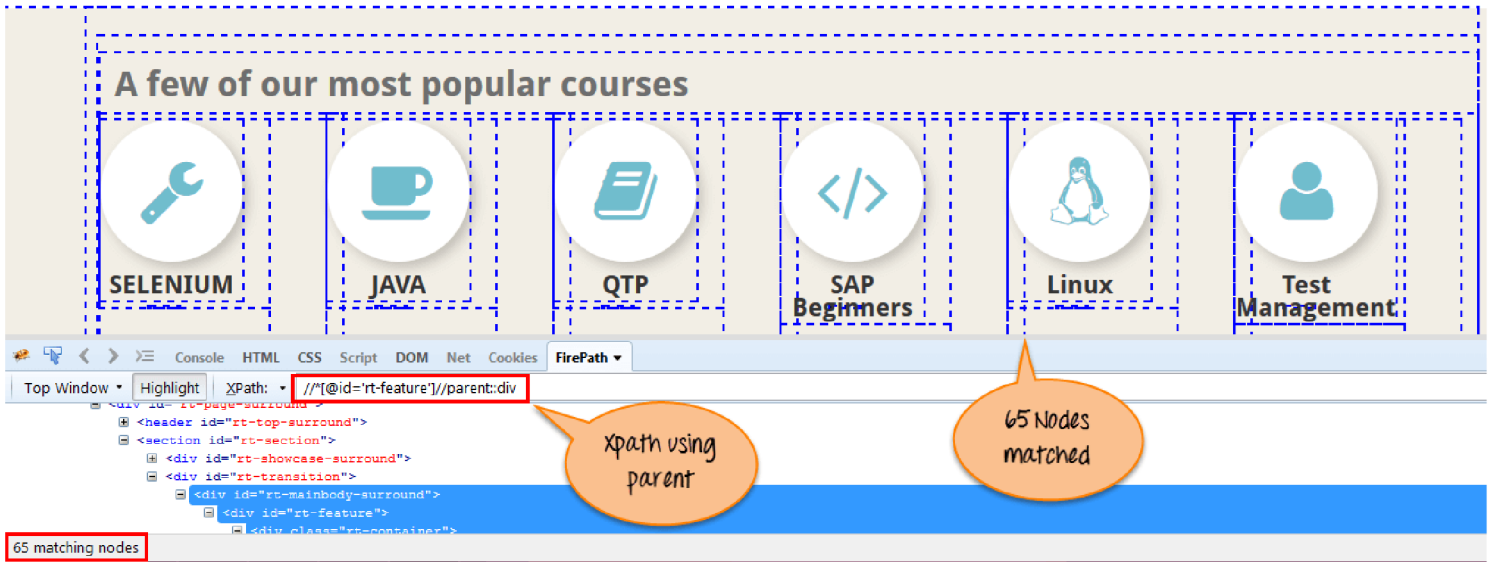
(/images/3-2016/032816_0758_XPathinSele17.png)

One input nodes matching by using "following-sibling" axis.

f) Parent:

Selects the parent of the current node as shown in the below screen.

```
Xpath=//*[@id='rt-feature']//parent::div
```



(/images/3-2016/032816_0758_XPathinSele18.png)

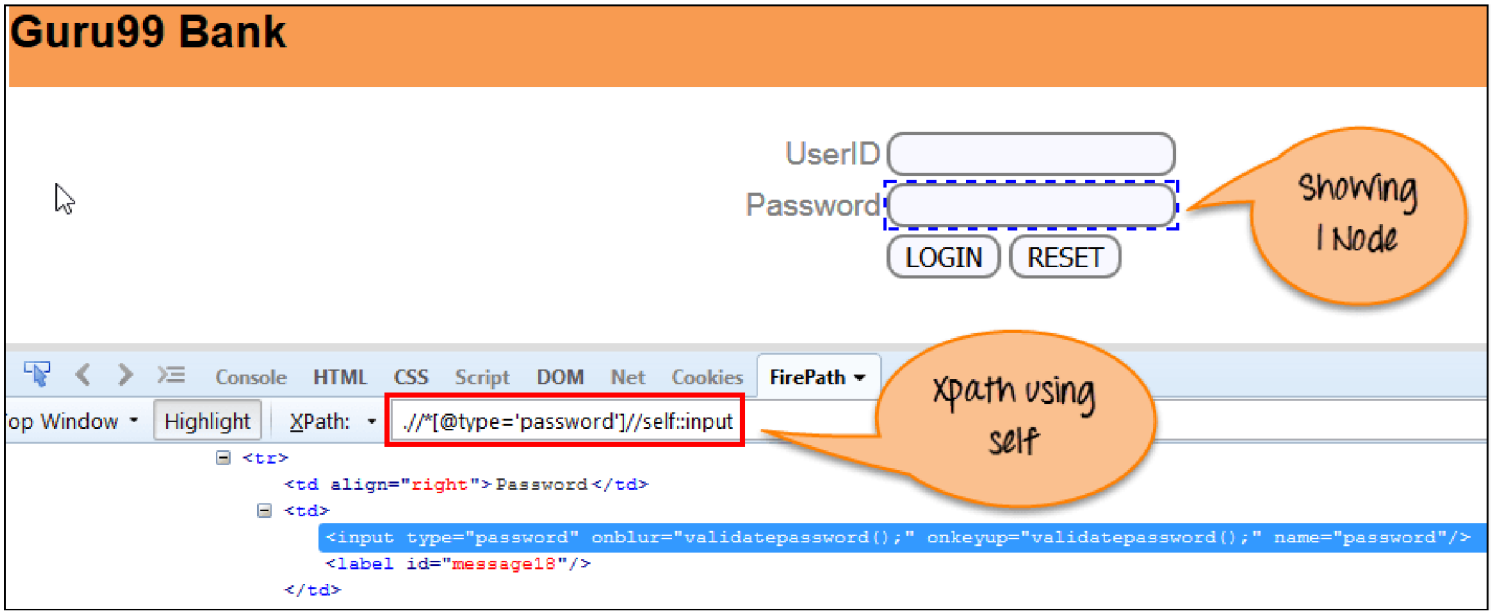
There are 65 "div" nodes matching by using "parent" axis. If you want to focus on any particular element then you can use the below XPath:

```
Xpath=//*[@id='rt-feature']//parent::div[1]
```

You can change the XPath according to the requirement by putting [1],[2]……and so on.

g) Self:

Selects the current node or 'self' means it indicates the node itself as shown in the below screen.



(/images/3-2016/032816_0758_XPathinSele19.png)

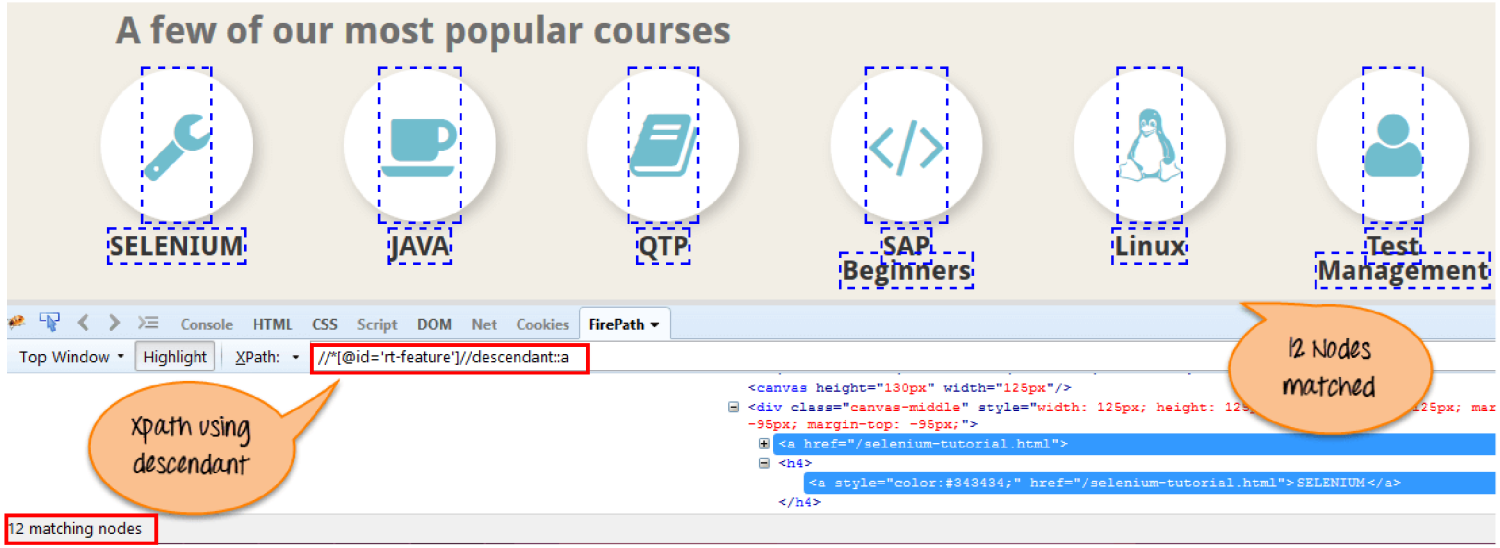
One node matching by using "self " axis. It always finds only one node as it represents self-element.

```
Xpath =//*[@type='password']//self::input
```

h) Descendant:

Selects the descendants of the current node as shown in the below screen.
In the below expression, it identifies all the element descendants to current element ('Main body surround' frame element) which means down under the node (child node , grandchild node, etc.).

```
Xpath=//*[@id='rt-feature']//descendant::a
```



(/images/3-2016/032816_0758_XPathinSele20.png)

There are 12 "link" nodes matching by using "descendant" axis. If you want to focus on any particular element then you can use the below XPath:

```
Xpath=//*[@id='rt-feature']//descendant::a[1]
```

You can change the XPath according to the requirement by putting [1],[2]……and so on.

Summary:

XPath is required to find an element on the web page as to do an operation on that particular element.

- There are two types of XPath:
  - **Absolute XPath**
  - **Relative XPath**

- XPath Axes are the methods used to find dynamic elements, which otherwise not possible to find by normal XPath method
- XPath expression select nodes or list of nodes on the basis of attributes like ID , Name, Classname, etc. from the XML document .

Prev (/upload-download-file-selenium-webdriver.html)

Report a Bug

Next (/alert-popup-handling-selenium.html)