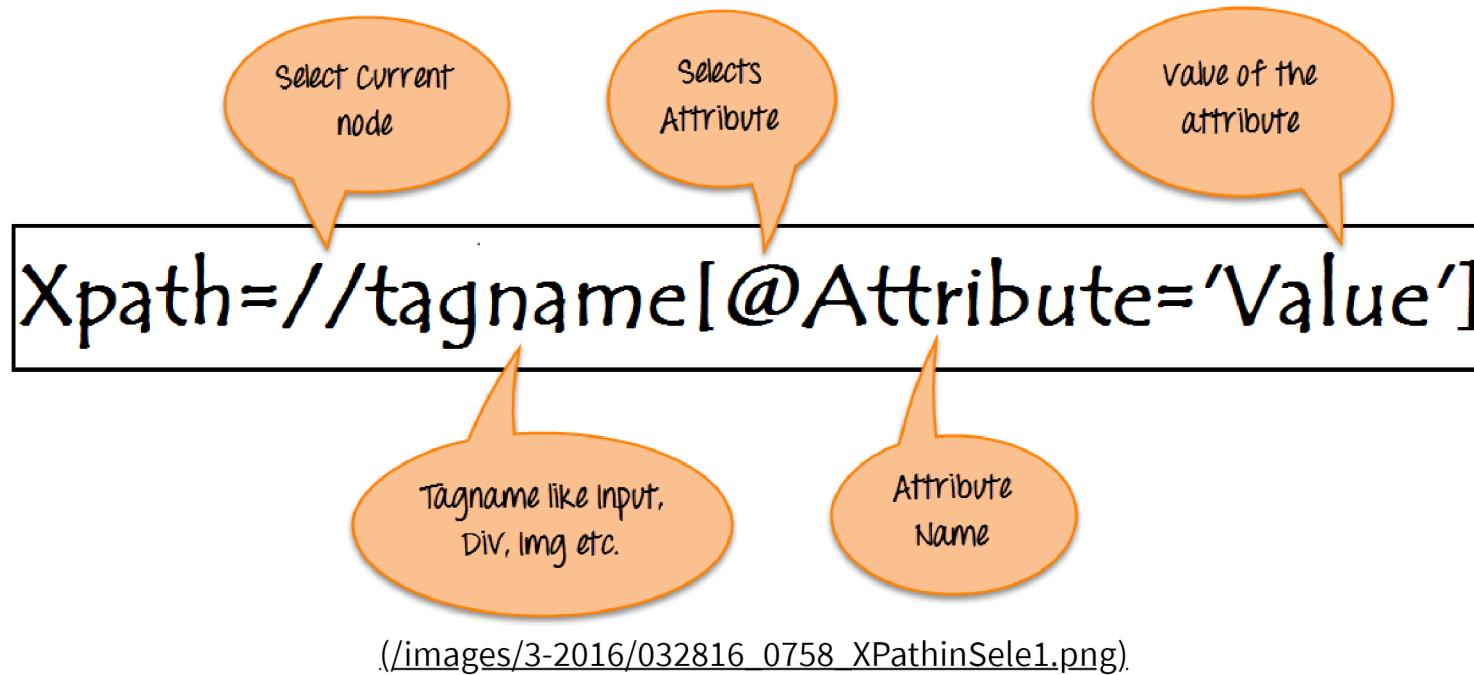


- [What is XPath?](#)
- [Types of X-path](#)
 - [Absolute XPath](#)
 - [Relative XPath](#)
- [Basic XPath](#)
- [Contains\(\)](#)
- [Using OR & AND](#)
- [Start-with function](#)
- [Text\(\)](#)
- [XPath axes methods](#)
 - [Following](#)
 - [Ancestor](#)
 - [Child](#)
 - [Preceding](#)
 - [Following-sibling](#)
 - [Parent](#)
 - [Self](#)
 - [Descendant](#)

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.



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[1]/div/div/div/div[3]/div[1]/div/h4[1]/b
```

The screenshot shows the FirePath extension for Firefox. At the top, there's a navigation bar with tabs like Console, HTML, CSS, Script, DOM, Net, Cookies, and FirePath. The FirePath tab is active. Below the tabs, there are three main sections: 'TESTING' (with 'Learn Software Testing', 'QTP (Quick Test Professional)', 'Learn Selenium', and 'Learn Mobile App Testing'), 'SAP' (with 'Learn SAP Beginner', 'Learn SAP ABAP', 'Learn SAP HR/HCM', and 'Learn SAP FICO'), and 'OTHER' (partially visible). A search bar at the bottom left says 'Top Window'. To its right is a 'Highlight' button, followed by an 'XPath:' dropdown containing the absolute XPath: 'html/body/div[1]/section/div[1]/div/div/div[1]/div/div/div/div[3]/div[1]/div/h4[1]/b'. This entire input field is highlighted with a red box. Below the search bar, a message '1 matching node' is displayed. On the left side of the main content area, there's a large orange speech bubble with the text 'Absolute Path'. The main content area shows the HTML DOM structure with the selected element highlighted in blue: '**Testing**'. The DOM tree is as follows:

```
<br/>


<b>Testing</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']`

The screenshot shows a web page with two main sections: 'TESTING' and 'SAP'. The 'TESTING' section contains links for 'Learn Software Testing', 'QTP (Quick Test Professional)', 'Learn Selenium', and 'Learn Mobile App Testing'. The 'SAP' section contains links for 'Learn SAP Beginner', 'Learn SAP ABAP', 'Learn SAP HR/HCM', and 'Learn SAP FICO'. Below the page, the FirePath toolbar is visible, with the 'XPath' tab selected. The XPath input field contains the expression `//*[@class='featured-box']//*[@text()='Testing']`. The FirePath interface displays the DOM tree, highlighting the node `Testing` under the heading 'Relative Path'. A callout bubble labeled 'Element' points to the 'Testing' link in the 'TESTING' section. Another callout bubble labeled 'Relative Path' points to the highlighted node in the DOM tree. A red box highlights the text '1 matching node' at the bottom left of the FirePath interface.

Element

TESTING

SAP

Learn Software Testing

QTP (Quick Test Professional)

Learn Selenium

Learn Mobile App Testing

Learn SAP Beginner

Learn SAP ABAP

Learn SAP HR/HCM

Learn SAP FICO

Live Tes

Live Sel

Live Ecc

Live UF

Console HTML CSS Script DOM Net Cookies FirePath

Top Window Highlight XPath: `//*[@class='featured-box']//*[@text()='Testing']`

Relative Path

1 matching node

(/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/>).

The screenshot shows a browser's developer tools with the FirePath tab selected. The URL bar displays 'http://www.guru99.com/demo/'. The 'Highlight' tab is active, and the 'XPath' input field contains the expression `./input[@name='uid']`. A large orange callout bubble labeled 'Basic xpath' points to this input field. The DOM tree below shows the HTML structure of a login form. A specific input element (`<input type="text" ... name="uid"/>`) is highlighted with a blue dashed border, and a red callout bubble labeled 'Element' points to it. A message label (`<label id="message23" ...>User-ID must not be blank</label>`) is also highlighted with a blue dashed border. A red callout bubble labeled 'Element' also points to this label. At the bottom left of the tool, a red box highlights the text '1 matching node'.

(/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.

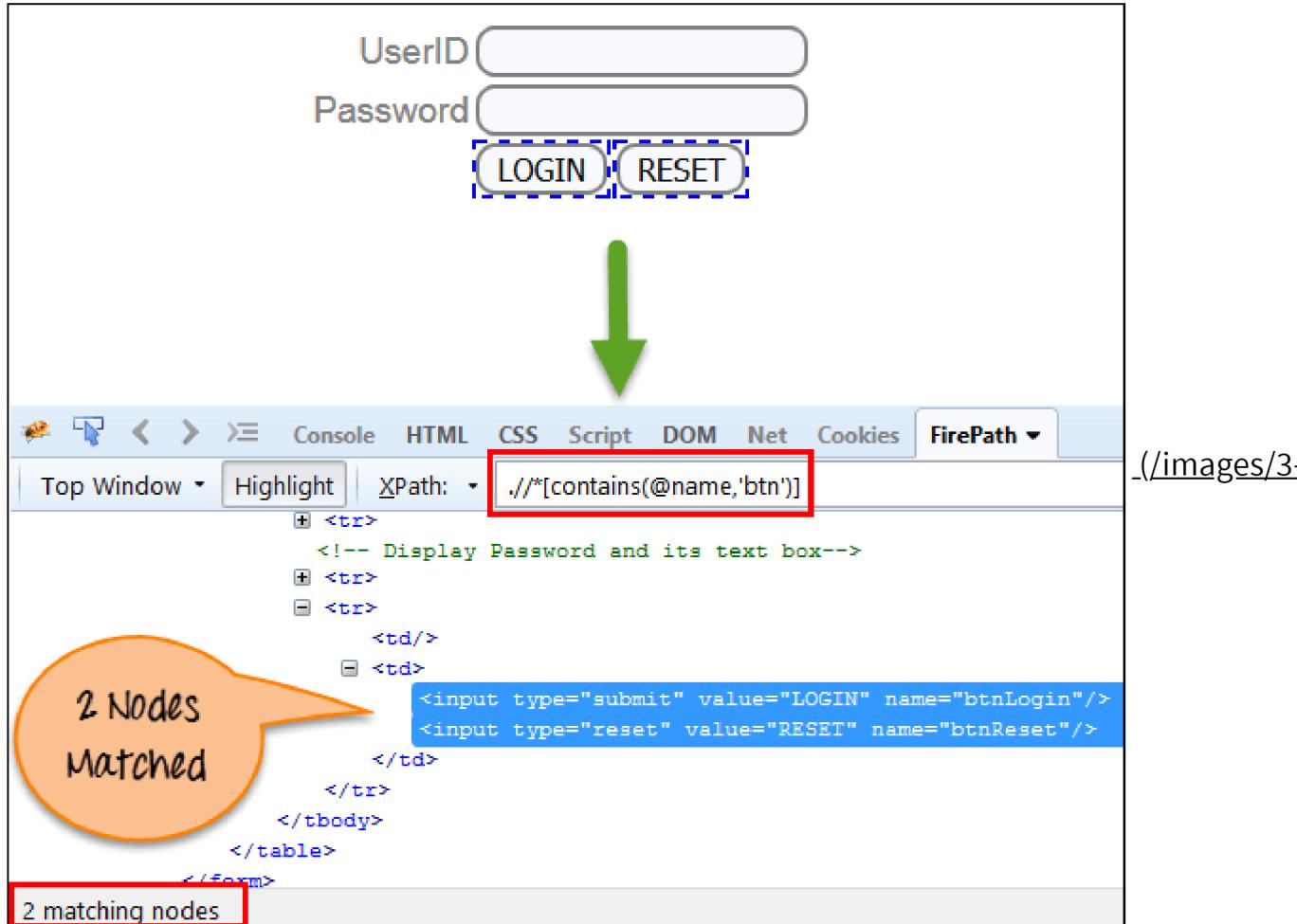
Complete value of 'Type' is 'submit' but using only partial value 'sub'!

```
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'.



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')]
```

User-ID User-ID must not be blank

Password Password must not be blank

LOGIN RESET

Console HTML CSS Script DOM Net Cookies FirePath ▾

Top Window ▾ Highlight XPath: **//*[contains(@id,'message')]**

2 Nodes Matched

2 matching nodes

```
<td>
  <input type="text" onblur="validateuserid();" onkeyup="validateuserid();" maxlength="10" name="uid"/>
  <label id="message23" style="visibility: visible;">User-ID must not be blank</label>
</td>
<tr>
  <!-- Display Password and its text box-->
  <tr>
    <td align="right">Password</td>
    <td>
      <input type="password" onblur="validatepassword();" onkeyup="validatepassword();" name="password"/>
      <label id="message18" style="visibility: visible;">Password must not be blank</label>
    </td>
  </tr>
</tr>
```

(/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')]
```

Steps To Generate Access

1. Visit - [here](#)
2. Enter your email id



Top Window ▾ Highlight XPath: //*[contains(text(),'here')]

Visit -
here

Enter your email id

Login credentials is allocated to you and mailed at your id

1 matching node

2016/032816_0758_XPathinSele7.png)

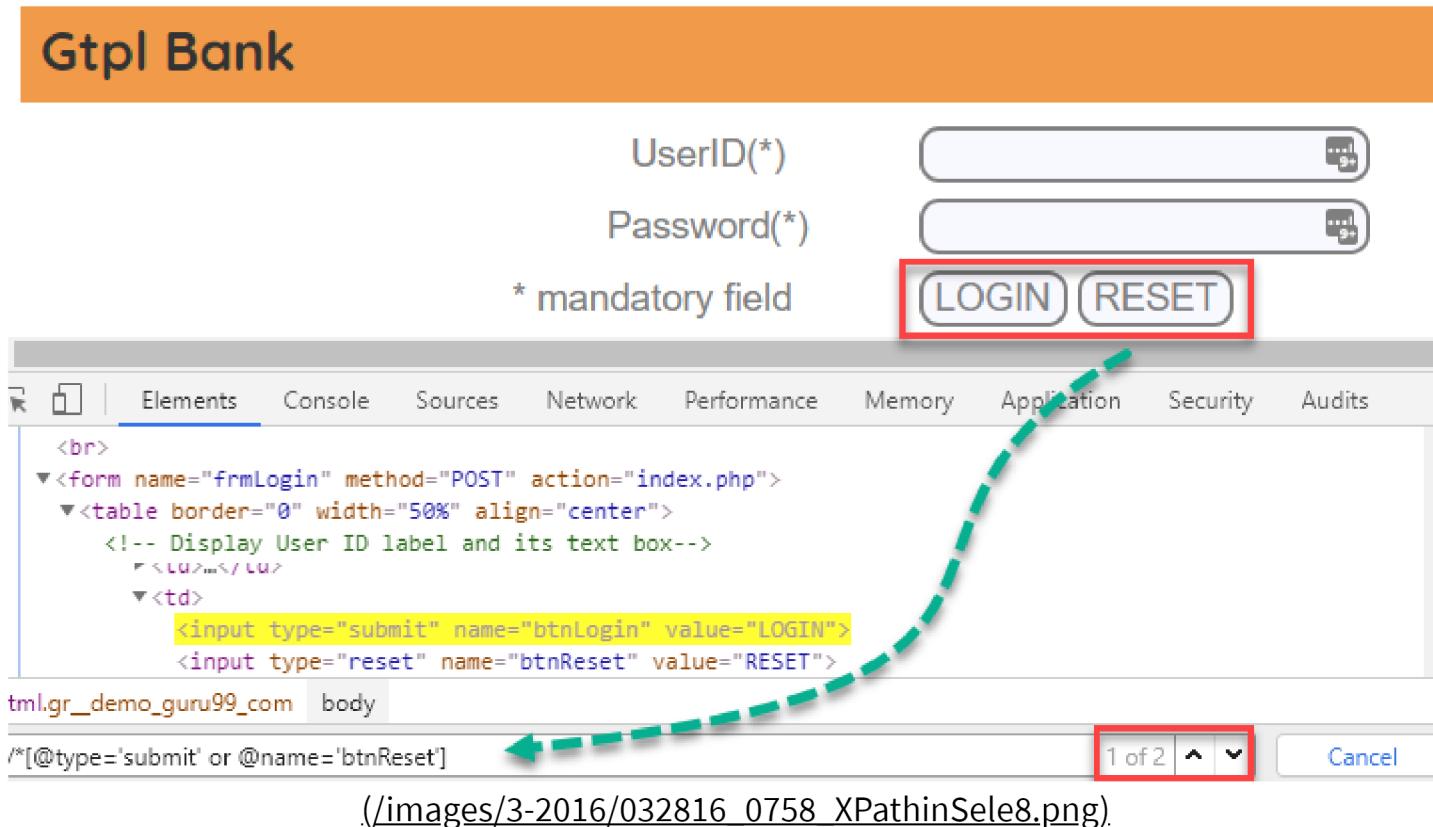
3) Using OR & AND:

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']
```

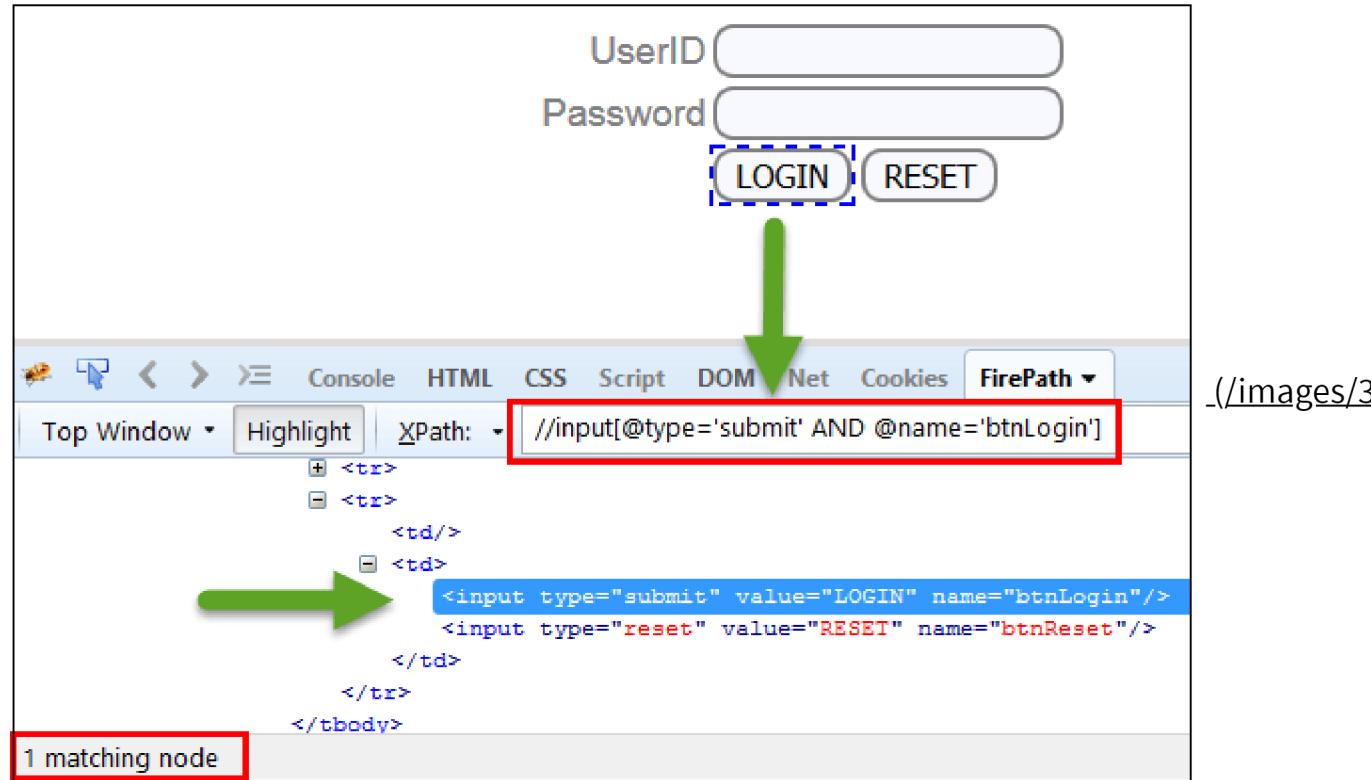
Highlighting both elements as "LOGIN" element having attribute 'type' and "RESET" element having attribute 'name'.



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'.



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')]

The screenshot shows a web browser interface with the FirePath extension active. The URL bar contains a placeholder 'http://www.example.com/'. The FirePath toolbar has tabs for Top Window, Highlight, and XPath. The XPath tab displays the expression `//label[starts-with(@id, 'message')]`. A green arrow points from this expression to the highlighted code in the DOM tree. The DOM tree shows two `<label>` elements with IDs `message23` and `message18` highlighted in blue. An orange callout bubble on the left says 'Id starting with 'message''. A red box highlights the XPath expression in the toolbar, and another red box highlights the text '2 matching nodes' at the bottom. The browser window shows a login form with fields for UserID and Password, each with an associated error message: 'User-ID must not be blank' and 'Password must not be blank' respectively.

(/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']
```

The screenshot shows a login interface with fields for 'UserID' and 'Password', and buttons for 'LOGIN' and 'RESET'. A green arrow points from the 'UserID' field to the XML structure below. The FirePath toolbar is visible at the top, with the 'XPath' tab selected and the query `//td[text()='UserID']`. The XML tree view highlights the first `<td>` node containing 'UserID' in blue. The status bar at the bottom indicates '1 matching node'.

(/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

The screenshot shows the Guru99 Bank login page. A callout bubble says "Showing 3 Nodes". The FirePath extension highlights three nodes: "Password" (with a dashed blue border), "LOGIN" (with a dashed blue border), and "RESET" (with a dashed blue border). The FirePath toolbar at the bottom shows the selected XPath: //*[@type='text']//following::input. The output pane shows the result: + <tr> <!-- Display Password and its text box--> </tr>

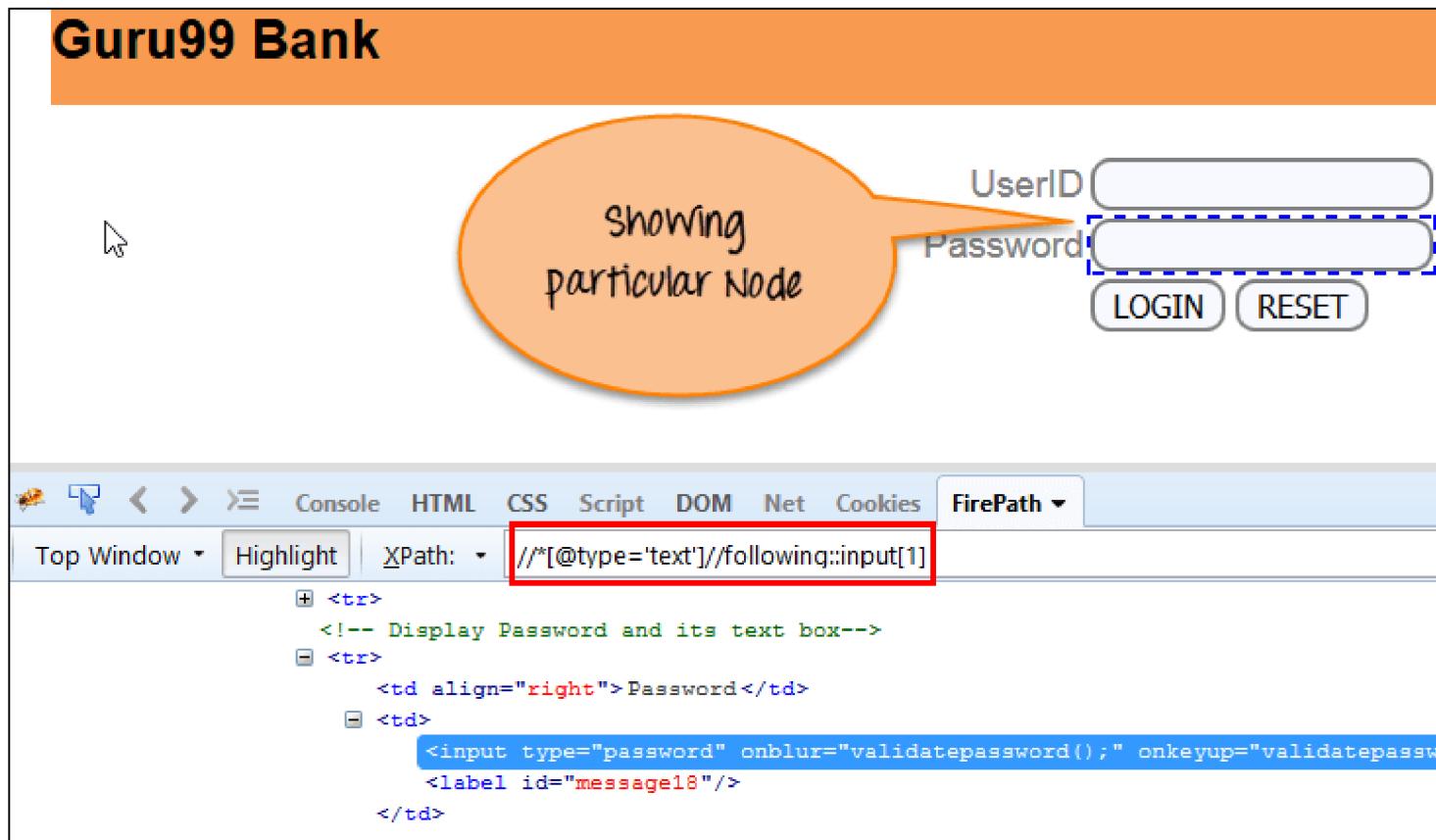
(./images/3-2016/032816_0758_XPathinSele12.png).

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
```

The screenshot shows a web page titled "Tutorials Library" with a sidebar containing four categories: "TESTING", "SAP", "LIVE PROJECTS", and "MUST LEARN!". The "SAP" section is highlighted with an orange oval, and the text "13 Nodes matched" is overlaid on it. The "TESTING" section also contains several items. At the bottom, a FirePath toolbar is visible, and the search bar displays the XPath expression: //*[text()='Enterprise Testing']//ancestor::div. An orange callout bubble points from this expression to the text "xpath using ancestor".

(./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
```

The screenshot shows a web page with a navigation menu. The menu items are organized into four main sections: TESTING, SAP, LIVE PROJECTS, and MUST LEARN!. Each section contains a list of learning topics. A speech bubble points to the TESTING section with the text "xpath using child". The browser's developer tools are open, showing the FirePath extension. The XPath expression //*[@id='java_technologies']/child::li is highlighted in the FirePath interface, and the results show 71 matching nodes. The FirePath interface also displays the HTML structure of the menu, specifically the element under the <div id='java_technologies'> element.

TESTING

- Learn Software Testing
- QTP (Quick Test Professional)
- Learn Selenium
- Learn Mobile App Testing
- Learn Cucumber Testing
- Learn SoapUI
- Learn Agile Testing

TEST MANAGEMENT

- Learn HP Quality Center/ALM
- Learn Test Management

SAP

- Learn SAP Beginner
- Learn SAP ABAP
- Learn SAP HR/HCM
- Learn SAP FICO
- Learn SAP Basis
- Learn SAP SD
- Learn SAP CRM
- Learn SAP MM
- Learn SAP CO
- Learn SAP Payroll

LIVE PROJECTS

- Live Testing Project
- Live Selenium Project
- Live Ecommerce Project
- Live UFT Testing
- Live IIP ALM Exercise
- Live Mobile Testing
- Live Security Testing
- Live PHP Project
- Live Scrum(Agile) Testing
- Live Insurance Testing

MUST LEARN!

- Learn Excel Tutorials
- Learn Accounting
- Learn Ethical Hacking
- Cloud Computing for Beginners
- Learn Photoshop CC
- Learn BigData
- Learn Digital Marketing
- Learn Business Analyst
- Learn Informatica
- Learn Project Management

xpath using child

Top Window Highlight Xpath: //*[@id='java_technologies']/child::li

71 matching nodes

(/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
```

The screenshot shows a browser developer tool's FirePath tab. The DOM tree is displayed with the following structure:

```
<!-- Display User ID label and its text box-->
<tr>
  <td align="right">UserID</td>
  <td>
    <input type="text" onblur="validateuserid();" onkeyup="validateuserid();" maxlength="10" name="uid"/>
    <label id="message23"/>
  </td>
</tr>
<!-- Display Password and its text box-->
<tr>
  <td align="right">Password</td>
  <td>
    <input type="password" onblur="validatepassword();" onkeyup="validatepassword();" name="password"/>
    <label id="message18"/>
  </td>
</tr>
```

The input fields for UserID and Password are selected by the XPath expression `//*[@type='submit']//preceding::input`, which is highlighted in red. A callout bubble labeled "Showing 2 Nodes" points to the selected inputs. Another callout bubble labeled "XPath using preceding" points to the highlighted XPath expression. The status bar at the bottom of the tool indicates "2 matching nodes".

(./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:

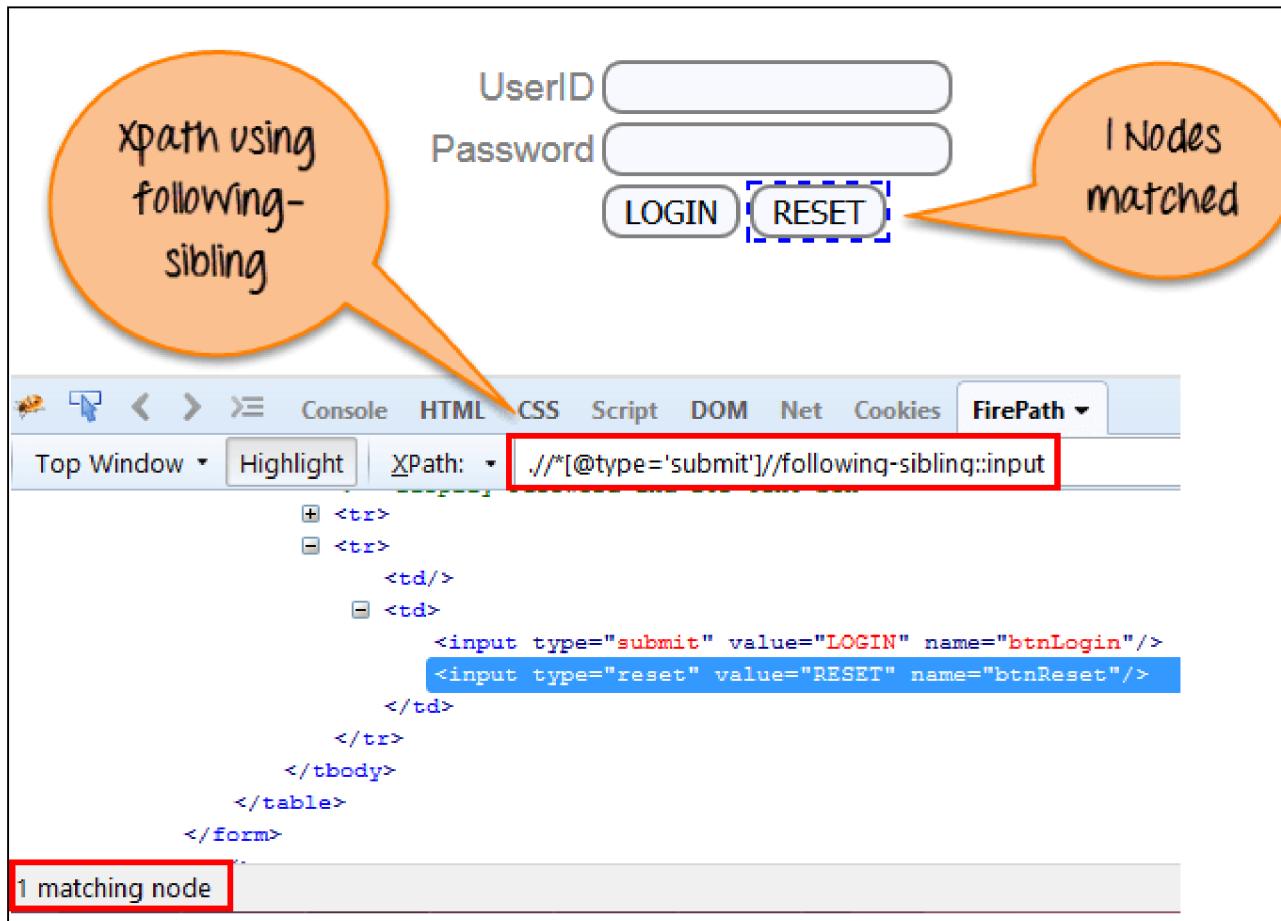
```
xpath=//*[@type='submit']//preceding::input[1]
```

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
```



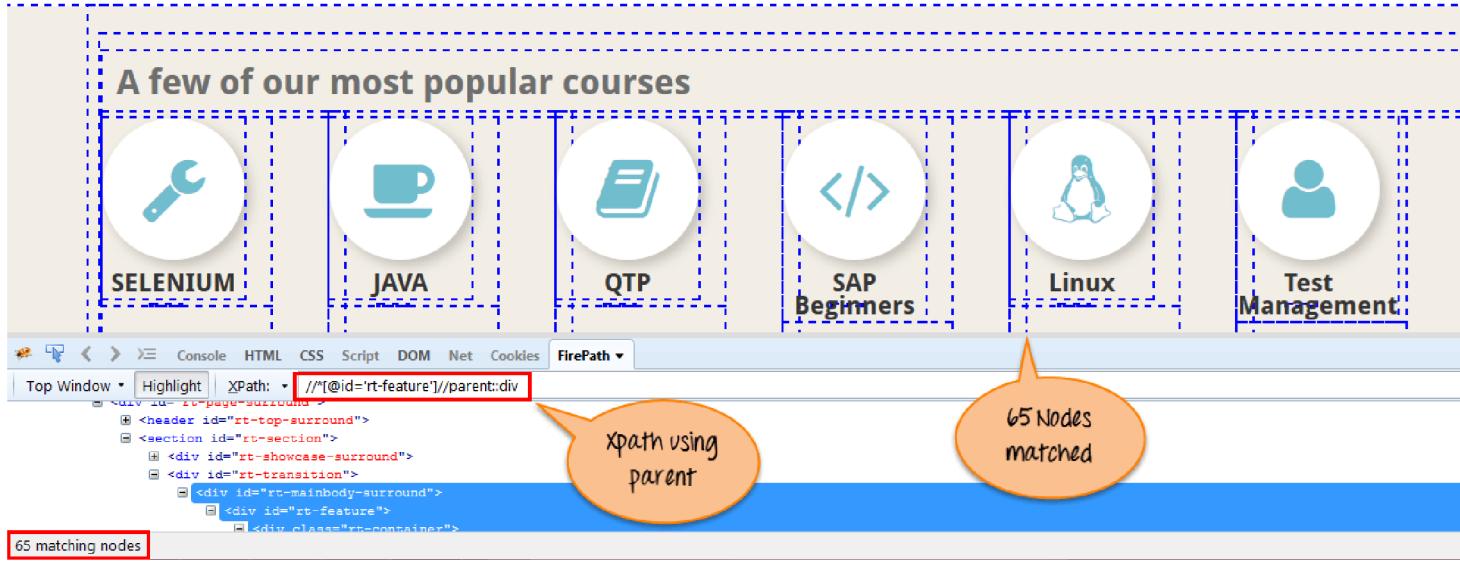
2016/032816_0758_XPathinSele17.png)

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](#)).

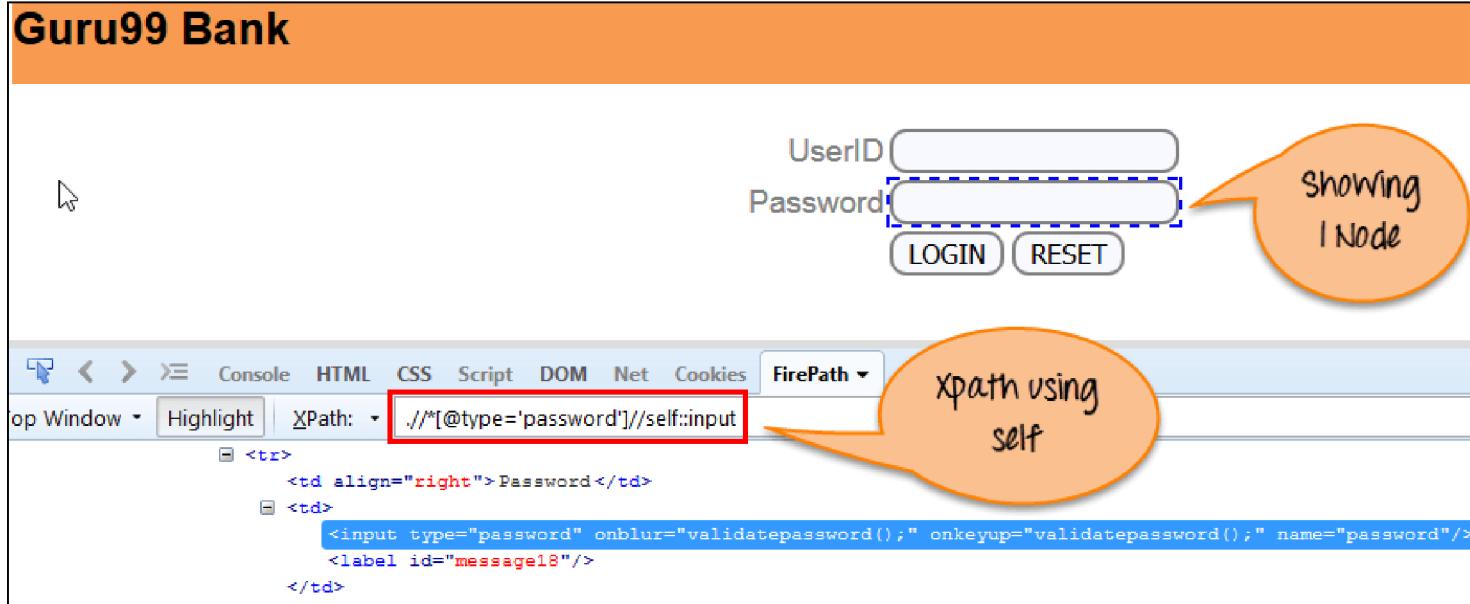
There are 65 "div" nodes matching by using "parent" axis. If you want to focus on any particular element then you can use the below XPath:

```
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
```

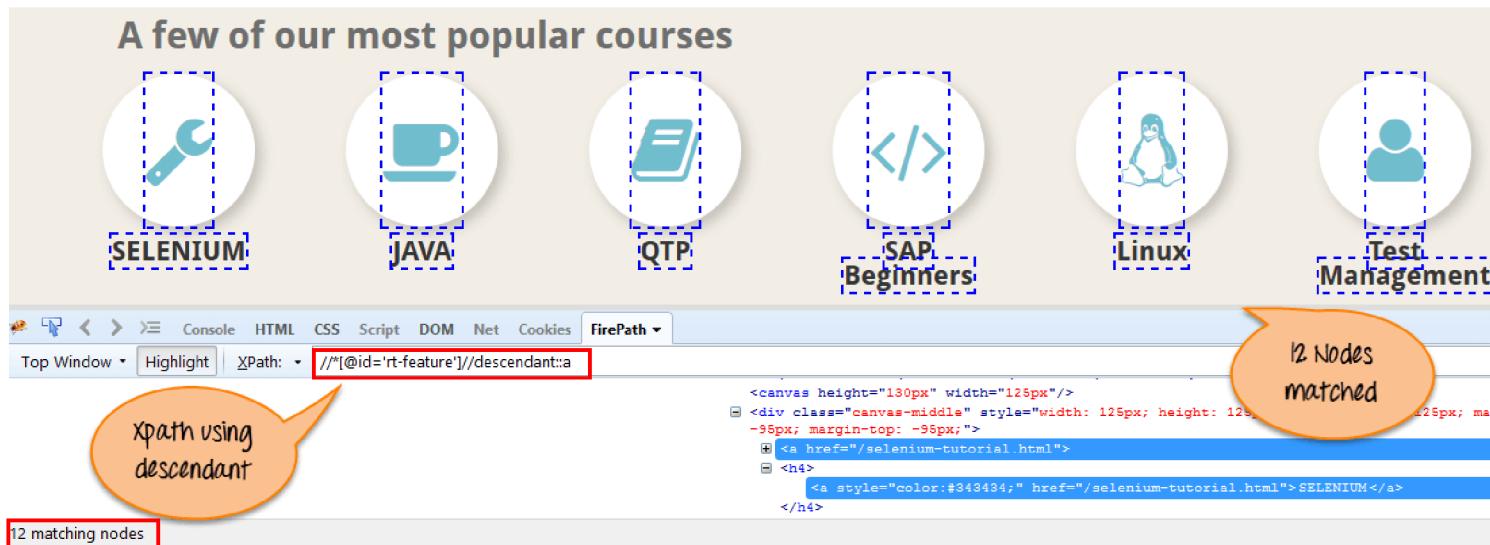
h) Descendant:

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
```

A few of our most popular courses



(./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 ➤ \(/alert-popup-handling-selenium.html\)](#)