

# Selenium Locating Strategy

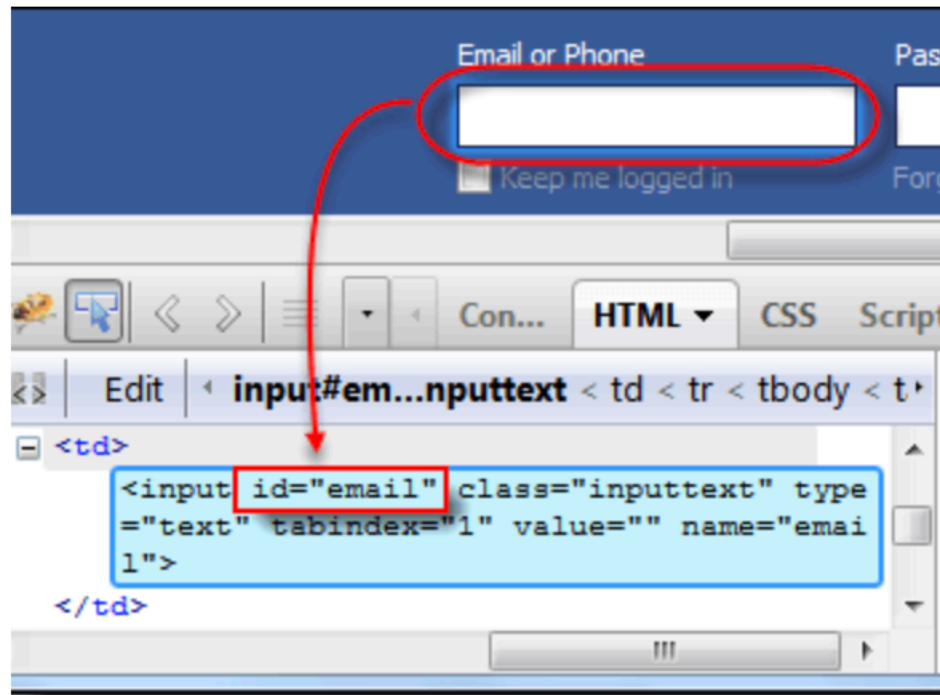
# Type of Locators

- ▶ ID Locator
- ▶ Name Locator
- ▶ Class name
- ▶ Link Text Locator
- ▶ Partial Link text
- ▶ Tag Name
- ▶ XPath Locator
- ▶ CSS

- ▶ **id** Select element with the specified **@id** attribute.
- ▶ **Name** Select first element with the specified **@name** attribute.
- ▶ **Linktext** Select link (anchor tag) element which contains text matching the specified link text
- ▶ **Partial Linktext** Select link (anchor tag) element which contains text matching the specified partial link text
- ▶ **Tag Name** Locate Element using a Tag Name .
- ▶ **Class name** Locate Element using a class Name ..
- ▶ **Css** Select the element using css selectors. You can check here for refer [W3C CSS Locatros](#)
- ▶ **Xpath** Locate an element using an XPath expression.

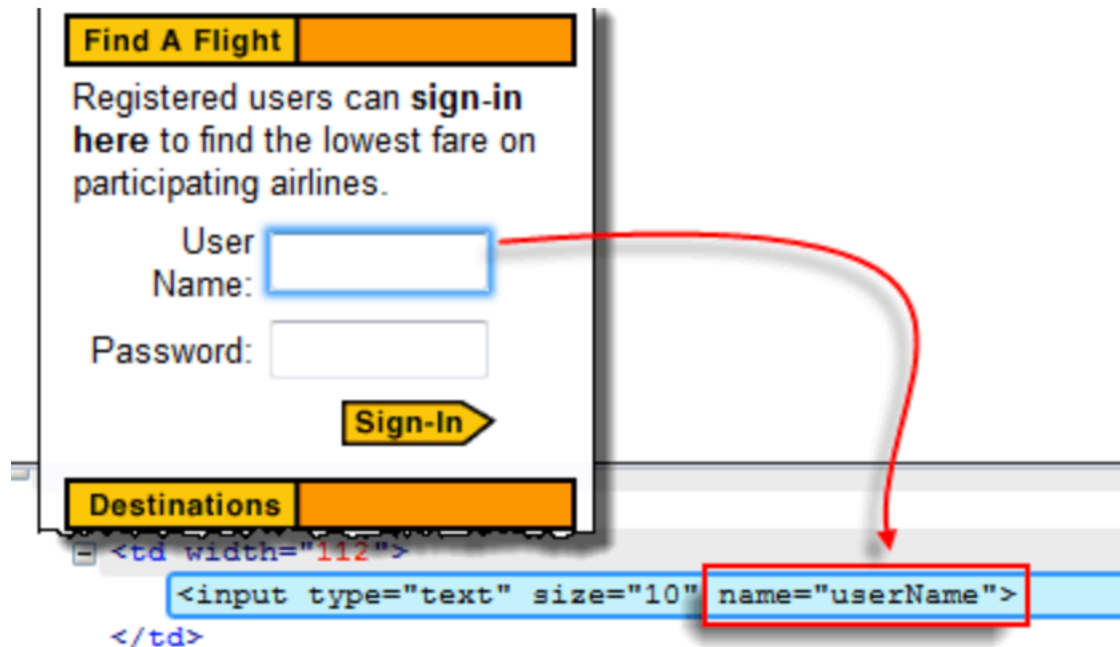
# Locating by ID

- ▶ Target Format: `id=id of the element`



# Locating by Name

- ▶ Target Format: `name=name of the element`



# Locating by Link Text

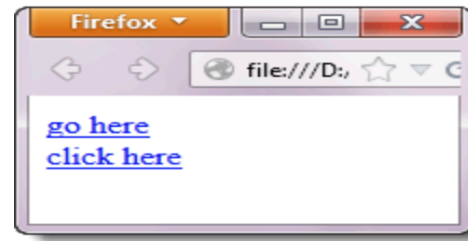
Target Format: link=*link\_text*



```
'mouseover') " onmouseout="changeStyle(th  
is, 'mouseout') ">  
  <a href="mercuryregister.php">REGISTER<  
  /a>  
</td>  
+ <td class="mouseout" width="73" height="
```

# Locating by Partial Link Text

```
<html>
  <head>
    <title>Partial Match</title>
  </head>
  <body>
    <a href="http://www.google.com">go here</a>
    <br>
    <a href="http://www.fb.com">click here</a>
  </body>
</html>
```



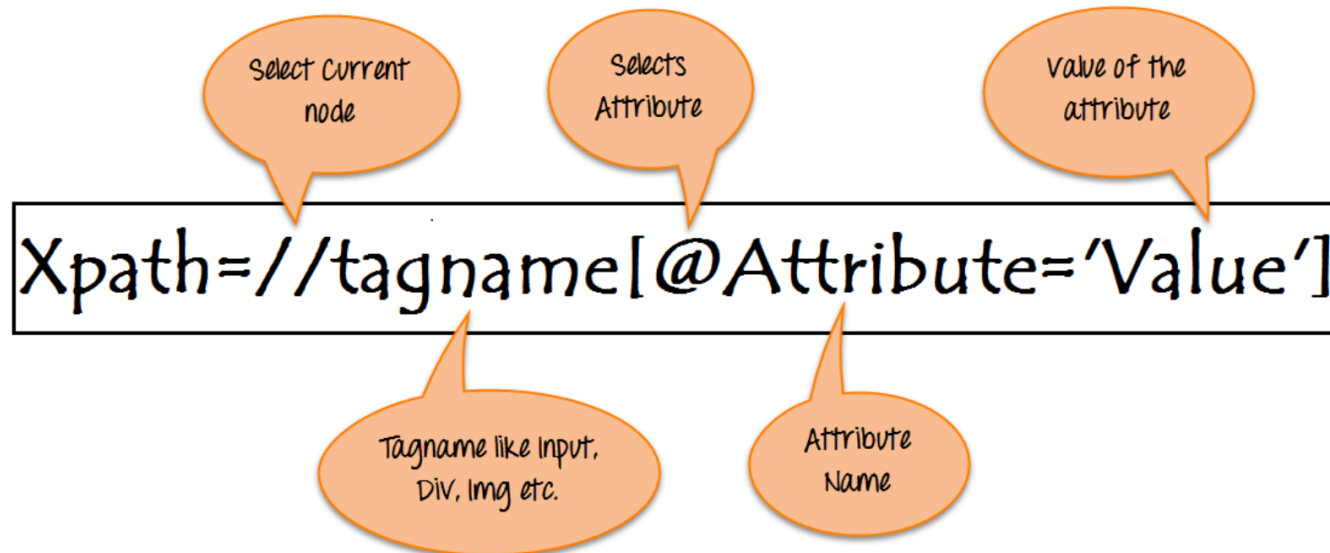
When you execute the WebDriver code below, you will still be taken to Google.

```
public static void main(String[] args) {
    String baseUrl = "file:///D:/partial_match.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    driver.findElement(By.partialLinkText("here")).click();
    System.out.println("Title of page is: " + driver.getTitle());
    driver.quit();
}
```

# Locating by XPath

- Syntax for Xpath : `Xpath=//tagname[@attribute='value']`





# 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.
- ▶ **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`

# Absolute xpath:

The screenshot displays the Selenium IDE interface. At the top, a table lists various learning topics under three columns: 'TESTING', 'SAP', and 'Live'. The 'TESTING' column includes 'Learn Software Testing', 'QTP (Quick Test Professional)', 'Learn Selenium', and 'Learn Mobile App Testing'. The 'SAP' column includes 'Learn SAP Beginner', 'Learn SAP ABAP', 'Learn SAP HR/HCM', and 'Learn SAP FICO'. The 'Live' column includes 'Live Test', 'Live Sele', 'Live Eco', and 'Live UFT'. Below the table, the 'FirePath' tab is active, showing the 'XPath' field with the absolute path: `html/body/div[1]/section/div[1]/div/div/div/div[1]/div/div/div/div/div[3]/div[1]/div/h4[1]/b`. The DOM tree on the right shows the corresponding HTML structure, with the `<b>Testing</b>` element highlighted. A red box at the bottom left indicates '1 matching node'. Hand-drawn orange callouts label 'Element' (pointing to the table) and 'Absolute Path' (pointing to the XPath field).

TESTING	SAP	Live
Learn Software Testing	Learn SAP Beginner	Live Test
QTP (Quick Test Professional)	Learn SAP ABAP	Live Sele
Learn Selenium	Learn SAP HR/HCM	Live Eco
Learn Mobile App Testing	Learn SAP FICO	Live UFT

FirePath

Top Window Highlight 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`

```
<br/>
<div class="row featured-boxes">
  <div class="col-md-3">
    <div class="featured-box" style="height: 700px;">
      <h4>
        <b>Testing</b>
      </h4>
      <ul id="java_technologies" class="menu">
        <p style="line-height: 15px;">
      <h4>
```

1 matching node

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

# Relative xpath:

Element

TESTING	SAP	
Learn Software Testing	Learn SAP Beginner	Live Tes
QTP (Quick Test Professional)	Learn SAP ABAP	Live Sel
Learn Selenium	Learn SAP HR/HCM	Live Ecc
Learn Mobile App Testing	Learn SAP FICO	Live UF

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

Relative Path

```
<div class="row featured-boxes">
  <div class="col-md-3">
    <div class="featured-box" style="height: 700px;">
      <h4>
        <b>Testing</b>
      </h4>
      <ul id="java_technologies" class="menu">
        <p style="line-height: 15px;">
      </p>
      <h4>
      <ul id="java_technologies" class="menu">
```

1 matching node

# Using XPath Handling complex & Dynamic elements in Selenium

## ► 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.

# 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.
- ▶ Syntax: **Xpath=//input[@name='uid']**

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 'name' is 'btnLogin' but using only partial value 'btn'.
  - ▶ `Xpath=//*[@contains(@name,'btn')]`
- ▶ Complete value of 'Type' is 'submit' but using only partial value 'sub'.
  - ▶ `Xpath=//*[@contains(@type,'sub')]`



- ▶ XPath=//\*[contains(@id,'message')]

UserID  User-ID must not be blank

Password  Password must not be blank

LOGIN RESET

FirePath

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

2 Nodes Matched

2 matching nodes

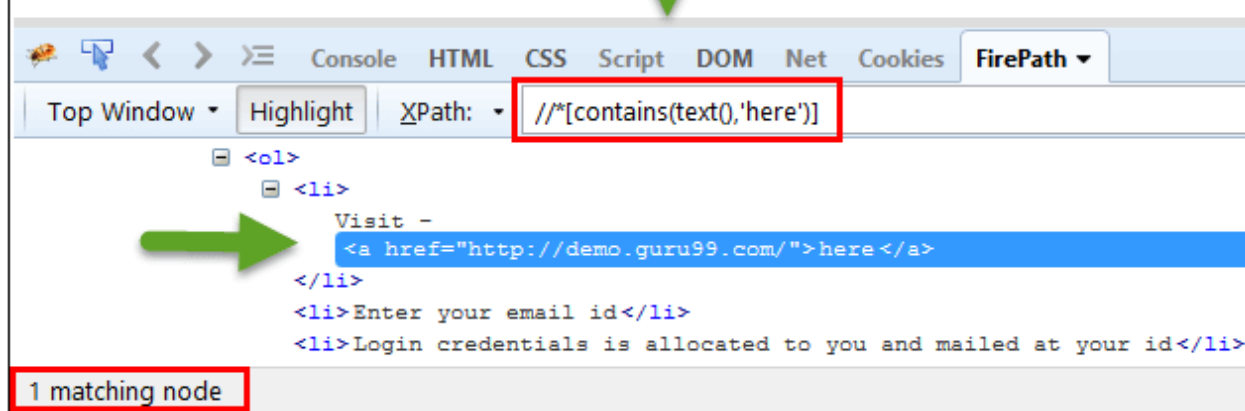

```
<td>
  <input type="text" onblur="validateuserid();" onkeyup="validateuserid();" maxlength="10" name="uid"/>
  <label id="message23" style="visibility: visible;">User-ID must not be blank</label>
</td>
<tr>
  <!-- Display Password and its text box-->
  <tr>
    <td align="right">Password</td>
  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

- ▶ 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**

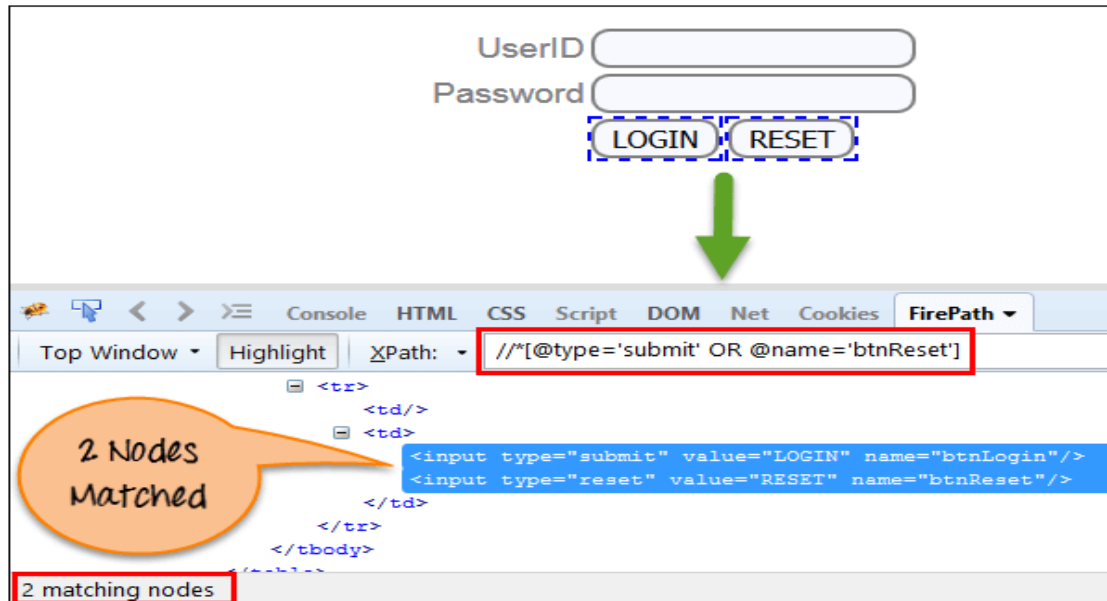
1. Visit - here
2. Enter your email id



1 matching node

### 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'.



# AND

- ▶ 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.
- ▶ Syntax **Xpath=//input[@type='submit' and @name='btnLogin']**

UserID

Password

FirePath

Top Window ▾ Highlight XPath: **//input[@type='submit' AND @name='btnLogin']**

```
<tr>
  <tr>
    <td/>
    <td>
      <input type="submit" value="LOGIN" name="btnLogin"/>
      <input type="reset" value="RESET" name="btnReset"/>
    </td>
  </tr>
</tbody>
```

**1 matching node**

## 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:
  1. Id=" message12"
  2. Id=" message345"
  3. 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'.
- ▶ Syntax: **Xpath=//label[starts-with(@id,'message')]**

The screenshot displays a web form with the following elements:

- UserID  User-ID must not be blank
- Password  Password must not be blank
- LOGIN RESET

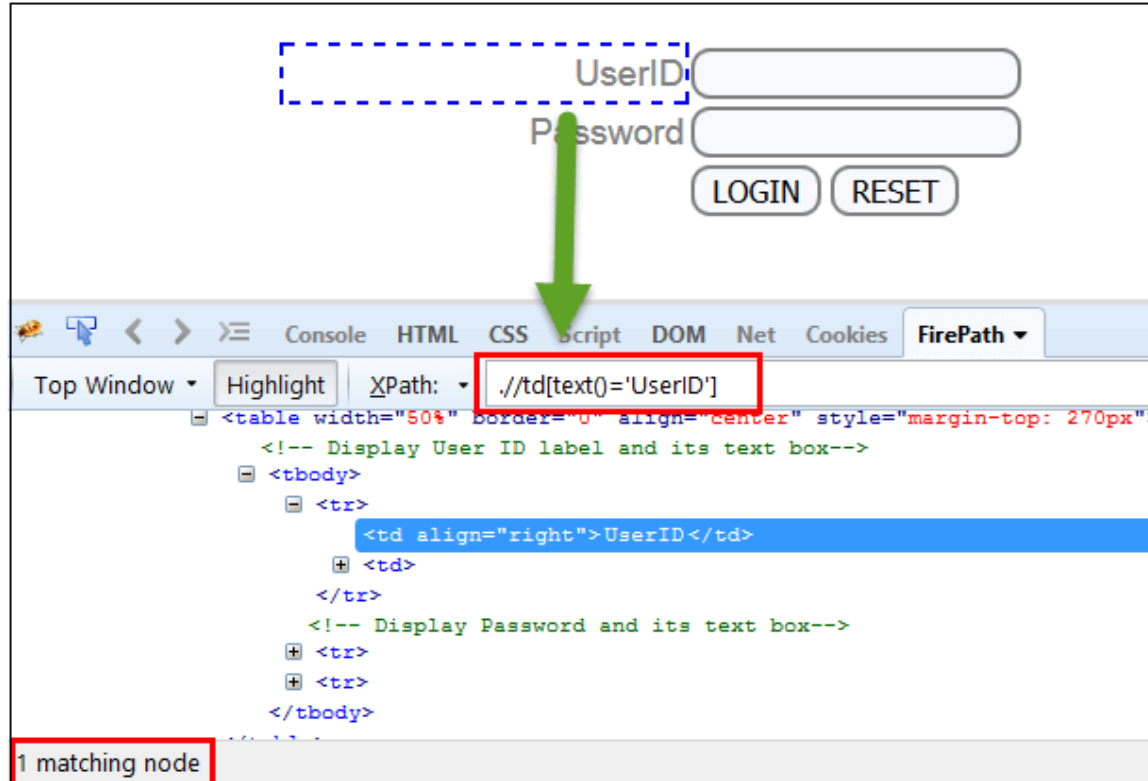
The FirePath tool interface shows the XPath expression `//label[starts-with(@id,'message')]` entered in the search bar. The DOM tree below shows the following structure:

```
<tr>
  <td align="right">UserID</td>
  <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>
```

The two matching nodes are highlighted in blue. A red box at the bottom left indicates "2 matching nodes". A green dashed arrow points from the XPath expression to the highlighted nodes. A speech bubble on the left says "Id starting with 'message'".

## 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".
- ▶ Syntax: `Xpath=//td[text()='UserID']`



# 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.
- ▶ **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.
- ▶ Syntax: **Xpath=//\*[@type='text']//following::input**





- ▶ 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:
- ▶ Syntax: **Xpath=//\*[@type='text']/following::input[1]**
- ▶ You can change the XPath according to the requirement by putting [1],[2].....and so on.



## 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).
- ▶ Syntax: **Xpath=//\*[text()='Enterprise Testing']/ancestor::div**

**Tutorials Library**

TESTING	SAP	LIVE PROJECTS	MUST LEARN!
▶ Learn Software Testing	▶ Learn SAP Beginner	▶ Live Testing Project	▶ Learn Excel Tutorials
▶ QTP (Quick Test Professional)	▶ Learn SAP ABAP	▶ Live Selenium Project	▶ Learn Accounting
▶ Learn Selenium	▶ Learn SAP CRM	▶ Live Ecommerce Project	▶ Learn Ethical Hacking
▶ Learn Mobile App Testing	▶ Learn SAP SD	▶ Live UFT Testing	▶ Cloud Computing for Beginners
▶ Learn Cucumber Testing	▶ Learn SAP E	▶ Live HP ALM Exercise	▶ Learn Photoshop CC
▶ Learn SoapUI	▶ Learn SAP SD	▶ Live Mobile Testing	▶ Learn BigData
▶ Learn Agile Testing	▶ Learn SAP CRM	▶ Live Security Testing	▶ Learn Digital Marketing

**13 Nodes matched**

**xpath using ancestor**

FirePath

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

<document>  
<html lang="en-gb" xml:lang="en-gb" slick-uniqueid="3">  
<head>

# c.Child

- ▶ Selects all children elements of the current node (Java) as shown in the below screen.
- ▶ Syntax: `Xpath=//*[@id='java_technologies']/child::li`

The screenshot displays a web browser with a menu structure. The menu is organized into four main categories, each with a list of links:

- 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

The FirePath extension is open, showing the XPath `//*[@id='java_technologies']/child::li` and 71 matching nodes. A speech bubble indicates "xpath using child".

## d. Preceding:

- ▶ Select all nodes that come before the current node as shown in the below screen.
- ▶ Syntax: **Xpath=//\*[@type='submit']//preceding::input**

Showing 2 Nodes

Xpath using preceding

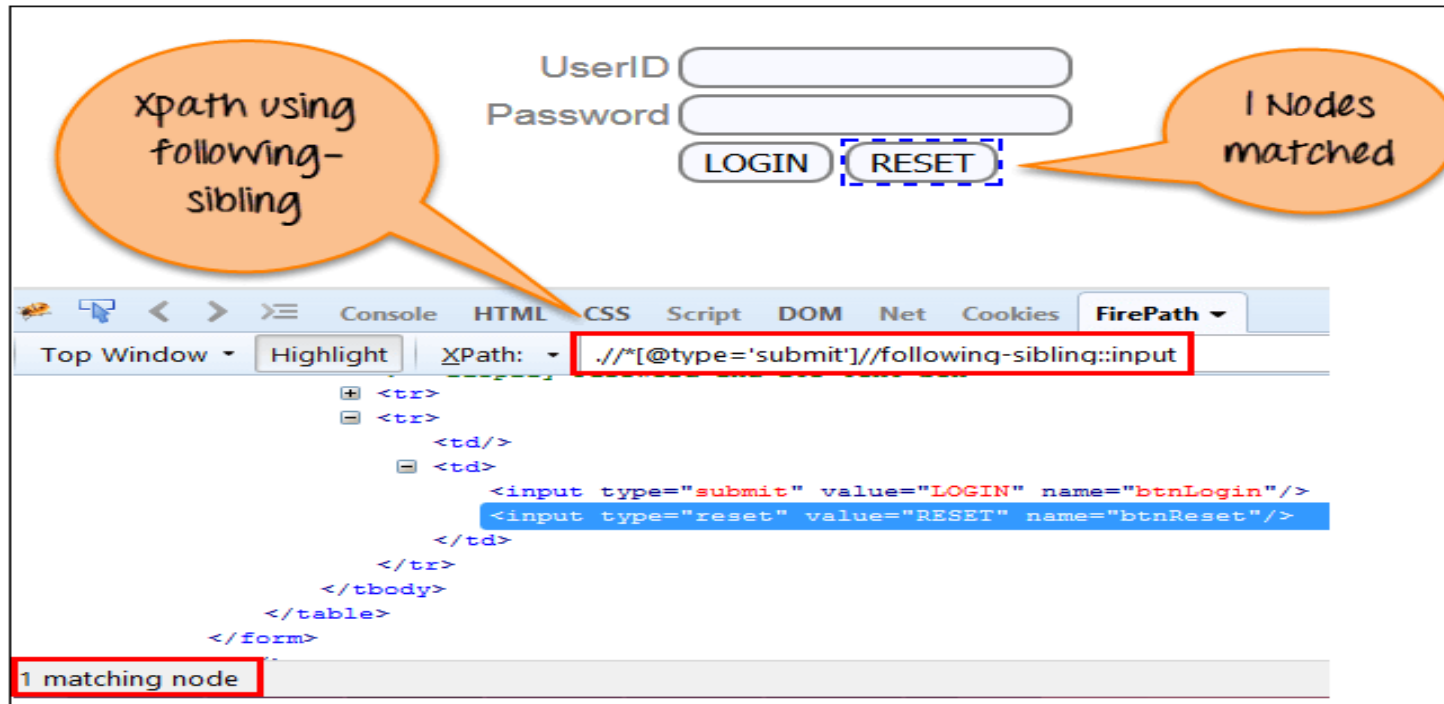
XPath: `//*[ @type='submit']//preceding::input`

```
<!-- Display User ID label and its text box-->
<tbody>
  <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="message16"/>
    </td>
  </tr>
</tbody>
```

2 matching nodes

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



## f.Parent:

- ▶ Selects the parent of the current node as shown in the below screen.
- ▶ Syntax: `Xpath=//*[@id='rt-feature']//parent::div`

The screenshot displays a web application interface with a section titled "A few of our most popular courses". Below the title, there are six course cards: SELENIUM, JAVA, QTP, SAP Beginners, Linux, and Test Management. Each card contains an icon and the course name. The Selenium IDE interface is overlaid on the bottom of the page. The XPath field is set to `//*[@id='rt-feature']//parent::div`, which is highlighted with a red box. A red box also highlights the text "65 matching nodes" at the bottom left. Two orange callout bubbles are present: one pointing to the XPath field with the text "xpath using parent", and another pointing to the "65 Nodes matched" text with the text "65 Nodes matched".

A few of our most popular courses

SELENIUM JAVA QTP SAP Beginners Linux Test Management

Top Window Highlight XPath: `//*[@id='rt-feature']//parent::div`

65 matching nodes

xpath using parent

65 Nodes matched

## g. Self:

- ▶ Selects the current node or 'self' means it indicates the node itself as shown in the below screen.
- ▶ Syntax: Xpath = `//*[@type='password']//self::input`

The screenshot displays a web browser window with the title "Guru99 Bank". The login form contains two input fields: "UserID" and "Password". The "Password" field is highlighted with a blue dashed border, and a speech bubble next to it says "Showing 1 Node". Below the input fields are two buttons: "LOGIN" and "RESET". The browser's developer tools are open, showing the "FirePath" tab. The "XPath" field contains the expression `//*[@type='password']//self::input`, which is highlighted with a red box. A speech bubble next to it says "xpath using self". The DOM tree below shows the HTML structure, with the password input field selected.

**Guru99 Bank**

UserID

Password

LOGIN RESET

Showing 1 Node

FirePath

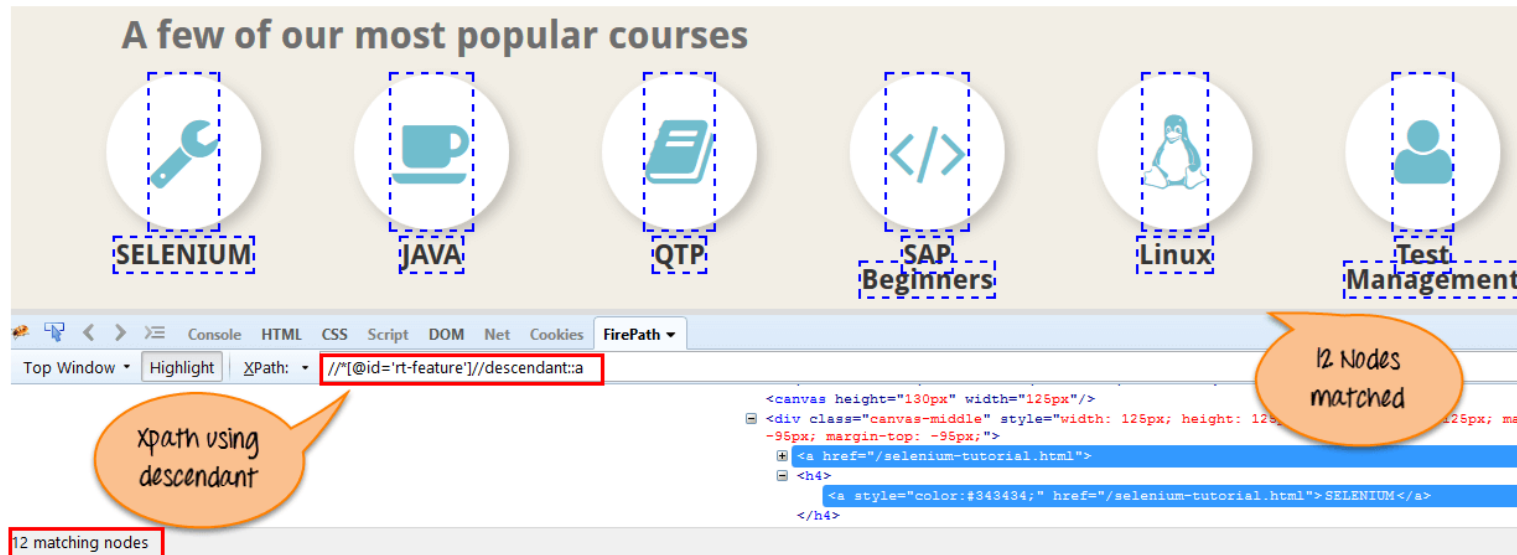
XPath: `//*[@type='password']//self::input`

xpath using self

```
<tr>
  <td align="right">Password</td>
  <td>
    <input type="password" onblur="validatepassword();" onkeyup="validatepassword();" name="password"/>
    <label id="message18"/>
  </td>
</tr>
```

## 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.).
- ▶ Syntax: **Xpath=//\*[@id='rt-feature']//descendant::a**





# Locating by CSS Selector

- ▶ CSS Selectors are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. Locating by CSS Selector is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.
- ▶ CSS Selectors have many formats, but we will only focus on the most common ones.
  - ✓ Tag and ID
  - ✓ Tag and class
  - ✓ Tag and attribute
  - ✓ Tag, class, and attribute
  - ✓ Inner text

# Locating by CSS Selector - Tag and ID

- ▶ Again, we will use Facebook's Email text box in this example. As you can remember, it has an ID of "email," and we have already accessed it in the "Locating by ID" section. This time, we will use a CSS Selector with ID in accessing that very same element.
- ▶ Syntax: `d.findElement(By.cssSelector("#twotabsearchtextbox"))`
- ▶ **Keep in mind that the ID is always preceded by a hash sign (#).**

Syntax	Description
<code>css=tag#id</code>	<ul style="list-style-type: none"><li>• tag = the HTML tag of the element being accessed</li><li>• # = the hash sign. This should always be present when using a CSS Selector with ID</li><li>• id = the ID of the element being accessed</li></ul>

# Locating by CSS Selector - tag, class, and attribute

Syntax	Description
<code>css=tag.class[attribute=value]</code>	<ul style="list-style-type: none"><li>• tag = the HTML tag of the element being accessed</li><li>• . = the dot sign. This should always be present when using a CSS Selector with class</li><li>• class = the class of the element being accessed</li><li>• [ and ] = square brackets within which a specific attribute and its corresponding value will be placed</li><li>• attribute = the attribute to be used. It is advisable to use an attribute that is unique to the element such as a name or ID.</li><li>• value = the corresponding value of the chosen attribute.</li></ul>

# Locating by CSS Selector - tag, class, and attribute

- ▶ `TAGNAME[attribute='value'][attribute='value']`
- ▶ `Input[id='email'][type='text']`

# Locating by CSS Selector - Class

- ▶ Syntax: `d.findElement(By.cssSelector(".twotabsearchtextbox"))`
- ▶ To find unique element by using more attributes
  - ▶ `Input.formBtn[name=go]`

# Locating by CSS Selector - inner text

- ▶ As you may have noticed, HTML labels are seldom given id, name, or class attributes. So, how do we access them? The answer is through the use of their inner texts. **Inner texts are the actual string patterns that the HTML label shows on the page.**

Syntax	Description
--------	-------------

`css=tag:contains("inner text")`

- tag = the HTML tag of the element being accessed
- inner text = the inner text of the element

# There are there important special characters:

1. '^' symbol, represents the starting text in a string.

Syntax: `css=input[id^='ema']`

2. '\$' symbol represents the ending text in a string.

Syntax: `css=input[id$='mail']`

3. '\*' symbol represents contains text in a string.

Syntax: `css=input[id*='mai']`

# CSS with Cascading classes part 1

## ► Syntax:

- Tagname.classname.classname.classname

```
<input id="username" type="email" tabindex="1" class="form-control private-form__control login-email" value> == $0
```

Example1:**input.form-control.private-form\_\_control.login-email**

Example2:**.form-control.private-form\_\_control.login-email**



# CSS with Cascading classes part 2

## ► Syntax:

- Tagname.classname.classname.classname (any one class can also be used)

```
<input id="username" type="email" tabindex="1" class="form-control private-form__control login-email" value> == $0
```

Example: **input.login-email**

# CSS with Cascading classes with id

- ▶ Syntax:

- ▶ Tagname#id.classname.classname.classname

```
<input id="username" type="email" tabindex="1" class="form-control private-form__control login-email" value> == $0
```

Example: **input#username.form-control.private-form\_\_control.login-email**

# CSS to find List of values

```
▼ <ul id="categories"> == $0
  ▶ <li class="201429665">...</li>
  ▶ <li class="360000211697">...</li>
  ▶ <li class="115000895445">...</li>
  ▶ <li class="360000217618">...</li>
  ▶ <li class="360000223318">...</li>
  ▶ <li class="201409709">...</li>
```

`ul#categories>li` → list of elements 1 of 14

`ul#categories>li:nth-of-type(n)` → list of elements 1 of 14

# CSS to find specific element in list of values using **nth-of-type**

```
▼ <ul id="categories"> == $0
  ▶ <li class="201429665">...</li>
  ▶ <li class="360000211697">...</li>
  ▶ <li class="115000895445">...</li>
  ▶ <li class="360000217618">...</li>
  ▶ <li class="360000223318">...</li>
  ▶ <li class="201409709">...</li>
```

`ul#categories>li:nth-of-type(2)` → list of elements 1 of 1

# CSS to find FIRST value in list of values using **first-of-type**

```
▼ <ul id="categories"> == $0
  ▶ <li class="201429665">...</li>
  ▶ <li class="360000211697">...</li>
  ▶ <li class="115000895445">...</li>
  ▶ <li class="360000217618">...</li>
  ▶ <li class="360000223318">...</li>
  ▶ <li class="201409709">...</li>
```

ul#categories>li:first-of-type → list of elements 1 of 1

# CSS to find LAST value in list of values using **last-of-type**

```
▼ <ul id="categories"> == $0
  ▶ <li class="201429665">...</li>
  ▶ <li class="360000211697">...</li>
  ▶ <li class="115000895445">...</li>
  ▶ <li class="360000217618">...</li>
  ▶ <li class="360000223318">...</li>
  ▶ <li class="201409709">...</li>
```

ul#categories>li:last-of-type → list of elements 1 of 1