

XPath

in Selenium WebDriver

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.

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.
- 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.
- Ex:
- `/html[1]/body[1]/div[1]/table[1]/tbody[1]/tr[1]/td[2]/table[1]/tbody[1]/tr[4]/td[1]/table[1]/tbody[1]/tr[1]/td[2]/table[1]/tbody[1]/tr[2]/td[3]/form[1]/table[1]/tbody[1]/tr[4]/td[1]/table[1]/tbody[1]/tr[2]/td[2]/input[1]`

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.

Ex:

`//input[@name='userName']`

Syntax for Relative 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.

Absolute & Relative XPath

elcome.php?osCsid=459b53e8a965de9b80e488932268ea9f

Summer ARUBA

STER SUPPORT CONTACT

Jan 6, 2019

Find A Flight

Registered users can sign-in here to find the lowest fare on participating airlines.

User Name:

Password:

Sign-In

Elements Console Sources Network Performance Memory Application Security Audits

Styles Computed Event Listeners DOM Breakpoints Properties

selectors type selector and press enter

1 matching node found. Find the matching node below:

rel XPath //input[@name='userName']

abs XPath /html[1]/body[1]/div[1]/table[1]/tbody[1]/tr[1]/td[2]/table[1]/tbody[1]/tr[4]/td[1]/table[1]/tbody[1]/tr[2]/td[3]/form[1]/table[1]/tbody[1]/tr[4]/td[1]/table[1]/tbody[1]/tr[2]/td[2]

CSS sel... table:nth-child(1) tbody:nth-child(1) tr:nth-child(2) td:nth-child(2) > input:nth-child(1)

<input type="text" name="userName" size="10" style="" xpath="1">

Relative XPath

Absolute XPath

Contains()

`//input[contains(@name,'first')]`

- 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

The screenshot displays a web browser interface on the left and the Chrome DevTools DOM panel on the right. The browser shows a 'Create an account' form with fields for 'Email or Phone', 'Password', 'First name', 'Surname', and 'Mobile number or email address'. The 'First name' field is highlighted with a dashed blue border. The DevTools DOM panel shows the corresponding HTML structure. The `<input type="text" class="inputtext _58mg _5dba _2ph-" data-type="text" name="firstname" aria-required="true" placeholder="First name" id="u_0_j" style="xpath='1'> == $0` element is highlighted. The 'Selectors' tab in the right-hand pane shows the XPath expression `//input[contains(@name,'first')]` selected, with a message indicating '1 matching node found'.

Contains() with text()

`//div[contains(text(),'Facebook helps')]`



The screenshot displays the Facebook homepage on the left, with the text "Facebook helps you connect and share with the people in your life." highlighted by a dashed blue box. On the right, the browser's developer tools are open, showing the DOM tree and the 'selectors' panel. The 'selectors' panel contains the XPath expression `//div[contains(text(),'Facebook helps')]`, which is highlighted with a red box. Below the XPath expression, it indicates "1 matching node found. Find the matching node". The DOM tree shows the structure of the page, with the selected node highlighted in the 'Elements' panel. The selected node is `<div class="_5iyx" style xpath="1">Facebook helps you connect and share with the people in your life.</div>`, which is also highlighted with a red box.

Using OR

```
//button[@type='submit' or name='web']
```

Create an acc

It's free and always will be.

First name

Su

Mobile number or email address

New password

Birthday

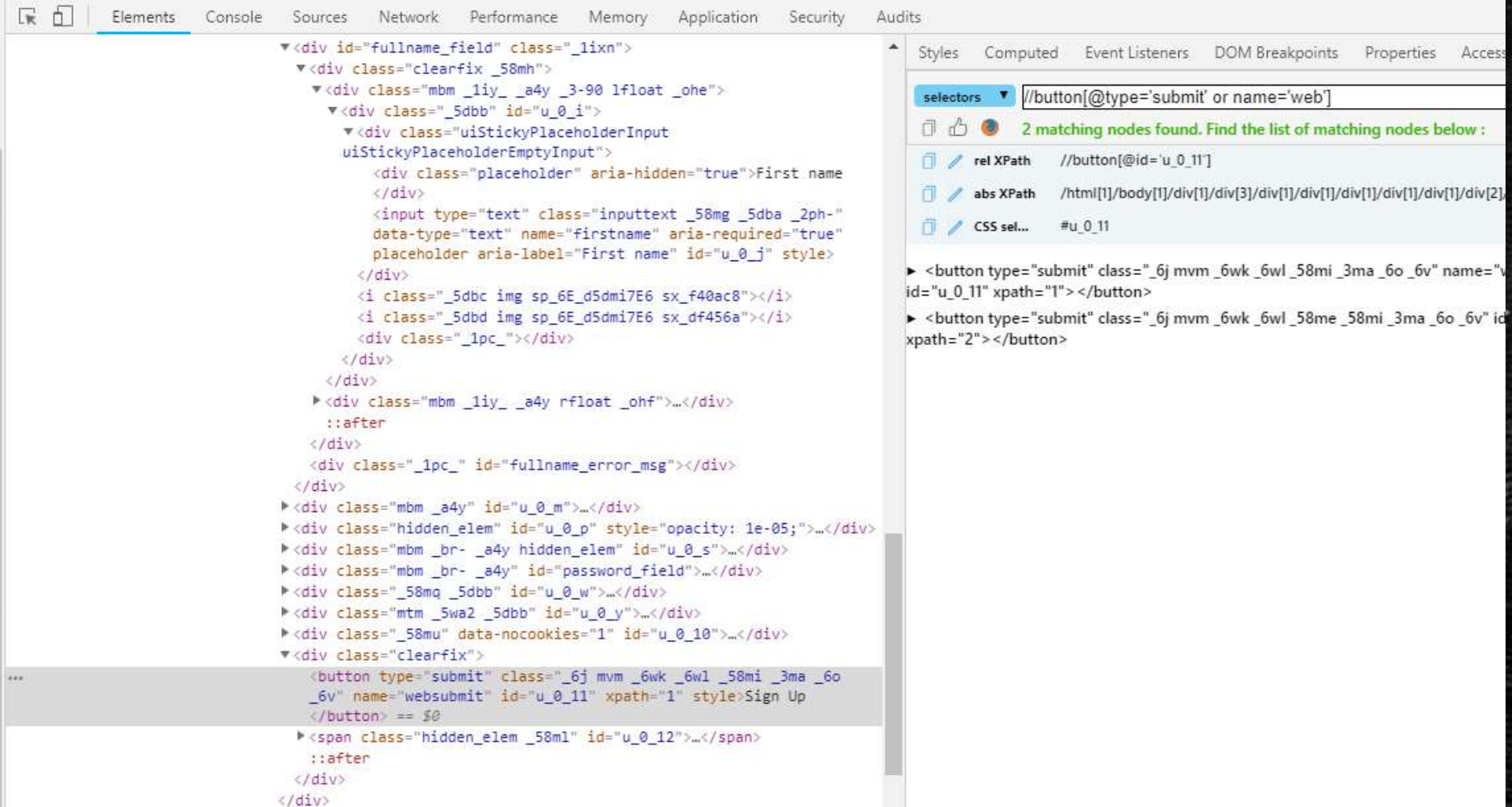
6 ▼ Jan ▼ 1994 ▼

Why do
date of

☐ Female ☐ Male

By clicking Sign Up, you agree to our [Terms](#), [Data Policy](#), and [Cookie Policy](#). You may receive SMS notification from us and you can opt out at any time.

Sign Up



Using AND

```
//button[@type='submit' and @name='websubmit']
```

Create an account

It's free and always will be.

First name

Surname

Mobile number or email address

New password

Birthday

6

▼

Jan

▼

1994

▼

Why do I need to provide my date of birth?

☐ Female ☐ Male

By clicking Sign Up, you agree to our [Terms](#), [Data Policy](#) and [Cookie Policy](#). You may receive SMS notifications from us and can opt out at any time.

Sign Up

Start-with()

//span[starts-with(text(),'Message')]

Text Labels

Message_12
Message-123
Message \$ 1234
Message **** 12345
Message &&&123456
Message#### 1234567

```
<div class="widget-content">  
  <span style="font-family:Georgia, serif;" xpath=  
    "1">Message_12</span>  
  <div>  
    <span style="font-family: Georgia, serif;"  
      xpath="2">Message-123</span>  
  </div>  
  <div>  
    <span style="font-family: Georgia, serif;"  
      xpath="3">Message $ 1234</span>  
  </div>  
  <div>  
    <span style="font-family: Georgia, serif;"  
      xpath="4">Message **** 12345</span>  
  </div>  
  <div>  
    <span style="font-family: Georgia, serif;"  
      xpath="5">Message &&&123456</span>  
  </div>  
  <div>  
    <span style="font-family: Georgia, serif;"  
      xpath="6">Message#### 1234567</span>  
  </div>  
</div>
```

Styles Computed Event Listeners DOM Breakpoints Pr

selectors //span[starts-with(text(),'Message')]

6 matching nodes found. Find the list of matching

rel XPath //div[@class='column-left-inner']/aside

abs XPath /html[1]/body[1]/div[4]/div[2]/div[2]/div[2]

CSS sel... body.variant-wide:nth-child(2) div.content:nth-ch

- >
- >
- >
- >
- >
- >

text()

//span[text()='Message-123']

Text Labels

Message_12

Message-123

Message \$ 1234

Message ***** 12345

Message &&&123456

Message#### 1234567

```
</div>
▶<div class="widget HTML" data-version="1" id="HTML9">...</div>
▶<div class="widget HTML" data-version="1" id="HTML5">...</div>
▶<div class="widget HTML" data-version="1" id="HTML6">...</div>
▼<div class="widget Text" data-version="1" id="Text1">
  <h2 class="title">Text Labels</h2>
  ▼<div class="widget-content">
    <span style="font-family: Georgia, serif;">Message_12</span>
    ▼<div>
      <span style="font-family: Georgia, serif;" xpath="1">
        Message-123</span>
      </div>
    ▼<div>
      <span style="font-family: Georgia, serif;">Message $ 1234
    </span>
```

selectors ▼ **//span[text()='Message-123']**

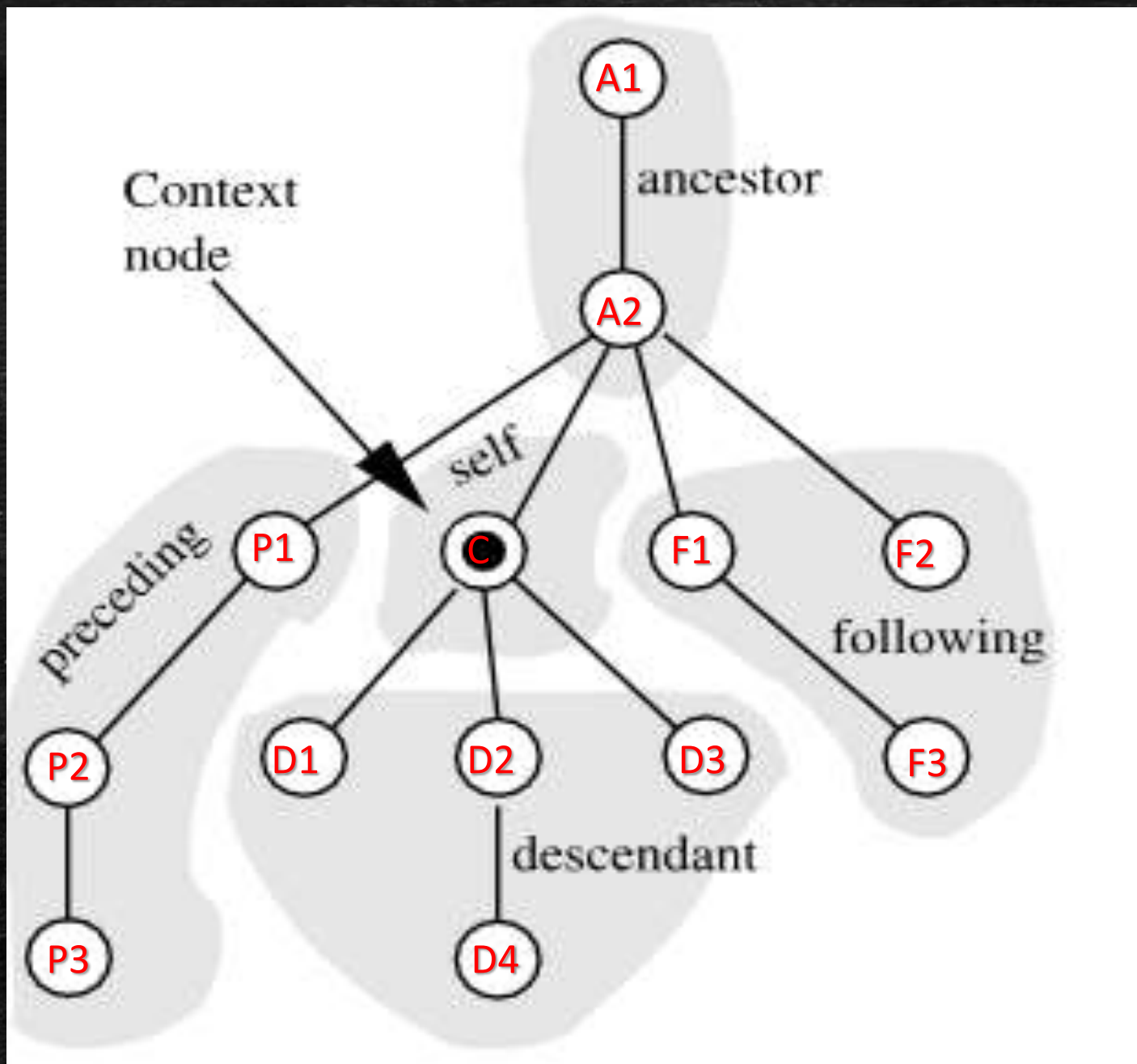
📄 👍 🔍 1 matching node found. Find the matching node

| | | |
|---|------------|---|
| 📄 | rel XPath | //div[@class='column-left-inner']//asp |
| 📄 | abs XPath | /html[1]/body[1]/div[4]/div[2]/div[2]/d |
| 📄 | CSS sel... | body.variant-wide:nth-child(2) div.cont |

▶

XPath axes methods

- XPath methods are used to find the complex or dynamic elements.
 - Ancestor
 - Child
 - Parent
 - Preceding
 - Following
 - Self
 - Descendant



Ancestor

- selects all ancestors element (grandparent, parent, etc.) of the current node

| | |
|-------------------------------------|---|
| <code>//employee/ancestor::*</code> | Select all ancestor node of the employee node. |
| <code>//ancestor::name</code> | Select all ancestor of the name node in context node. |

Child

- select child of the context node.

| | |
|-----------------|---|
| child::* | Select All child nodes of the context node. |
| child::employee | Select all child elements of employee node. |

Parent

- select the parent node of the context node.

| | |
|---------------------------------------|--|
| <code>//name/parent::*</code> | Select parent node of the 'name' context node. |
| <code>//email/parent::employee</code> | Return result node if employee node is parent node of the context node, otherwise no node found. |

Preceding

- select all nodes before the context node, excluding attributes node or namespace

`//employee[@id=3]/preceding::employee`

Select all nodes (with child nodes) before the context node.

The screenshot displays the 'XPath Axes' tool interface. On the left, a list of three employees is shown, each in a dashed blue box:

- 101 David Senior Engineer david@myemail.com
- 102 John DBA Engineer john@email.com
- 103 Marry Application Developer marry@email.com

On the right, the 'XPath' tab is active, showing the query `//employee[@id=3]/preceding::employee`. A message states 'matching nodes found. Find the list of matching nodes'. Below the query, the results are listed as XML elements:

```
<employee id="1" style="" xpath="1"></employee>
<employee id="2" style="" xpath="2"></employee>
```

Red arrows indicate the mapping: one arrow points from the '101' entry to the first result element, and another points from the '102' entry to the second result element. A large red arrow also points from the 'preceding::employee' part of the query to the list of employees.

Preceding...

```
//input[@id='u_0_v']//preceding::input[3]
```

The image shows a web form titled "Create an account" with fields for "Email or Phone", "Password", "First name", "Surname", "Mobile number or email address", and "New password". A red box highlights the "New password" field, which contains an exclamation mark icon. To the right, the browser's developer tools are open, showing the "Elements" panel. The "rel XPath" dropdown is set to "//input[@id='u_0_v']//preceding::input[3]", which is also highlighted with a red box. Below the XPath, it says "1 matching node found. Find the matching node". A red arrow points from the XPath box to the "New password" field in the form.

rel XPath `//input[@id='u_0_v']//preceding::input[3]`

1 matching node found. Find the matching node

`<input type="text" class="inputtext _58mg _5dba _2ph-" data-number="or email address" id="u_0_o" style="" xpath="1">`

Following

- select all nodes after the context node, excluding attributes node or namespaces node.

`//employee[@id=1]/following::employee`

Select all nodes (with child nodes) after the context node.

The screenshot displays the 'XPath Axes' tool interface. On the left, a list of nodes is shown, with the first node (ID 101) selected. An orange arrow points from the 'following' axis label in the tool's header to the first node in the list. The list contains three nodes, each with an ID, name, title, and email address. The second and third nodes are enclosed in dashed blue boxes. On the right, the 'Elements' panel shows the XML structure with the selected node highlighted.

| ID | Name | Title | Email |
|-----|-------|-----------------------|-------------------|
| 101 | David | Senior Engineer | david@myemail.com |
| 102 | John | DBA Engineer | john@email.com |
| 103 | Marry | Application Developer | marry@email.com |

Following..

`//input[@id='u_0_j']//following::input[2]`

The screenshot displays a web browser interface with a 'Create an account' form. The form includes fields for 'Email or Phone', 'Password', 'First name', 'Surname', 'Mobile number or email address', and 'New password'. The 'Mobile number or email address' field is highlighted with a red dashed box. The browser's developer tools are open, showing the 'rel XPath' tab with the expression `//input[@id='u_0_j']//following::input[2]` selected. A red box highlights the XPath expression, and a red arrow points from it to the highlighted field in the form.

Self

- Selects the current node 'name'

//name/self::*

Descendant

- Selects all descendants (children, grandchildren, etc.) of the current node

`//descendant::employee`

Select all descendant of the employee node in context node.