**How to write XPath in Selenium?**

Now that you have seen Absolute and Relative XPaths in Selenium, let us see a few of the basic XPath examples.

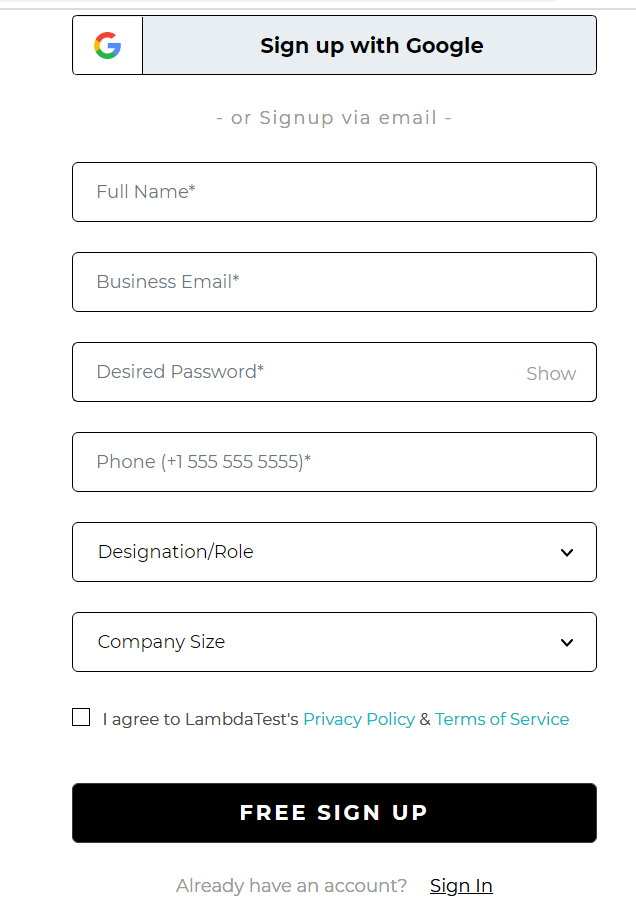
### **Basic XPath in Selenium**

This is the common and syntactic approach to writing the XPath in Selenium, which combines a tagname and attribute value.

Here are a few basic XPath examples in Selenium using the syntax:

**XPath = //tagname[@Attribute=’Value’]**

1. **//a[@class=’googleSignInBtn’]** – This XPath is for locating the Google Sign In button on the [LambdaTest SignUp Page](https://accounts.lambdatest.com/register/" \t "_blank) as highlighted in the below image. Here, I have used the class attribute and its ‘googleSignInBtn’ value of the corresponding tag.



1. **//input[@placeholder=’Full Name\*’]**– This XPath is for locating the Full Name text box in the above-shown image. Here, I have used the placeholder attribute and its corresponding value ‘Full Name\*’ for the input tag.
2. **//input[@name=’phone’]** – Similar to the first two options, this XPath is for locating the Phone text box where the name attribute is being used with its value being ‘phone.’
3. **//select[@name=designation]**– This XPath is for locating the Designation/Role dropdown and hence, has a select tag with its name attribute having value as designation.
4. **//a[@href=’/login’]**– This XPath is for the Sign In Option of the web page, where I am using its href attribute, which has a value of ‘/login.’

### **XPath using Contains**

Contains() is a very useful method in XPath. It can be used for all such web elements whose value can change dynamically. The syntax for using Contains() method in XPath is

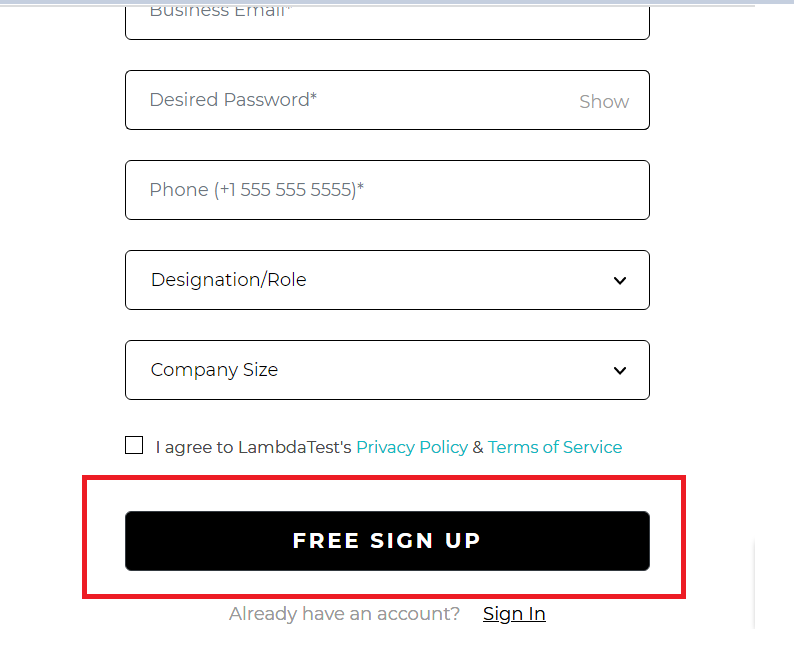
**//tagname[contains(@attribute,constantvalue)]**

For example, let’s say the ID for the login field, for instance, signin\_01 has the ending number that keeps changing every time the page is loaded. In this case, using contains helps us locate the element in the below way.

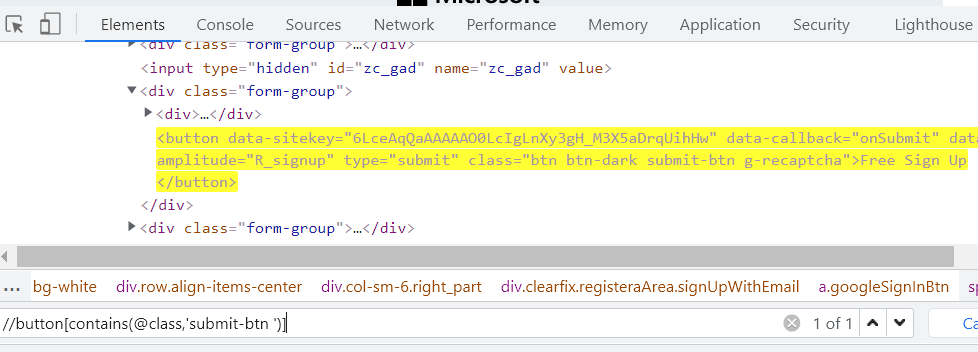
**//tagname[contains(@attribute,”signin”)]**

Let us check a similar example on the LambdaTest Page for the Free SignUp Button. On inspecting the element, you will see that the class attribute has a lot of values. However, you can use the contains() method and easily locate the web element.

**Element:**



**//button[contains(@class,'submit-btn ')]**



### **XPath using Logical Operators: OR & AND**

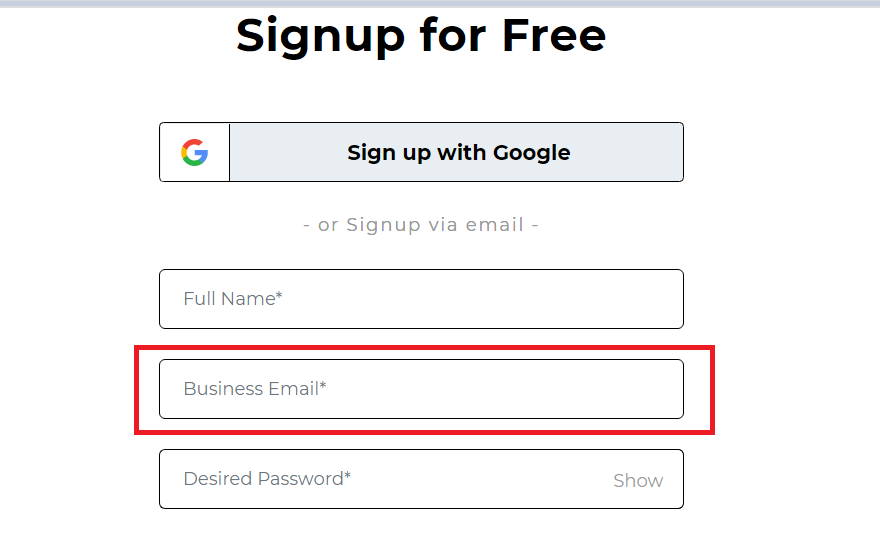
We can use logical operators such as OR & AND on the attributes condition. In the case of OR, any one of the conditions should be true or both, whereas, in the case of AND, both the conditions should be true.

The syntax for using these operators are:

|  |  |
| --- | --- |
| OR | XPath=//tagname[@attribute1=value1 OR @attribute2=value1] |
| AND | XPath=//tagname[@attribute1=value1 AND @attribute2=value1] |

Now, let us see an example for each of these operators. First, we will locate the input text box of Business Email on the Signup page as highlighted below.

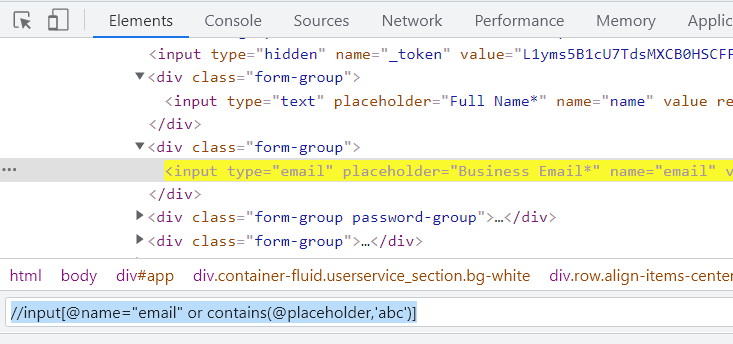
**Element:**



**OR**

Here, I have made use of the name attribute and the placeholder attribute. In addition, I have also used the Contains method on the placeholder attribute. Now, if you observe closely, I have used an incorrect value for the placeholder attribute. However, I am still able to locate the element as the first expression satisfies the condition. Interesting right?

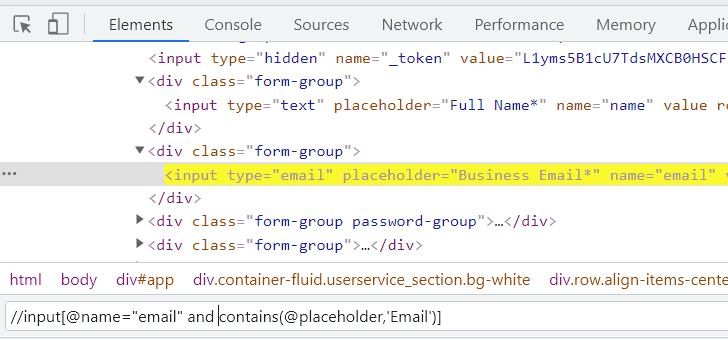
|  |  |
| --- | --- |
|  | **//input[@name="email" or contains(@placeholder,’abc’)]** |



**AND**

Here, I have made use of the same attributes and have used the AND operator. However, both the expressions need to satisfy the condition; hence, the below XPath locates the element.

|  |  |
| --- | --- |
|  | **//input[@name="email" and contains(@placeholder,'Email')]** |



**Note**: Both ‘and’ and ‘or’ should be case-sensitive. If you tend to use ‘OR’ or ‘AND,’ you will get an error in the console stating an invalid XPath expression.

### **XPath using Text()**

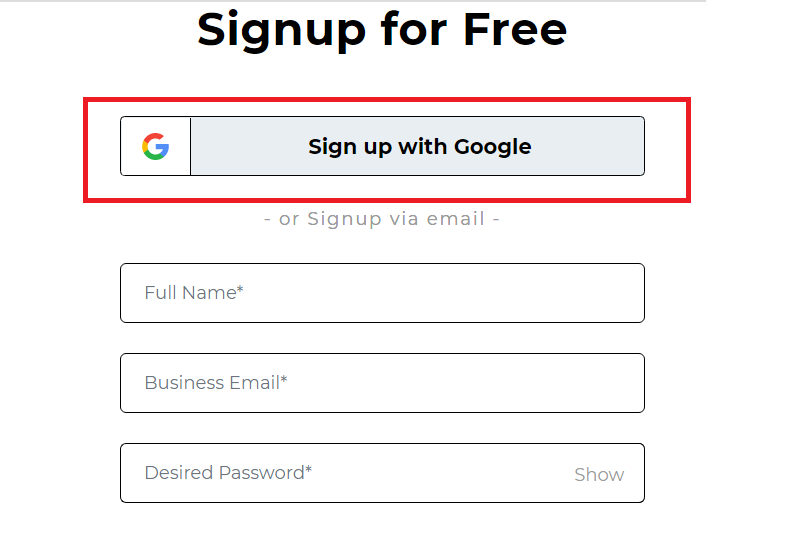
The text() method is used in XPath whenever we have a text defined in an HTML tag, and we wish to identify that element via text. This comes in handy when the other attribute values change dynamically with no substantial attribute value used via Starts-with or Contains.

The Syntax for using text() in XPath is:

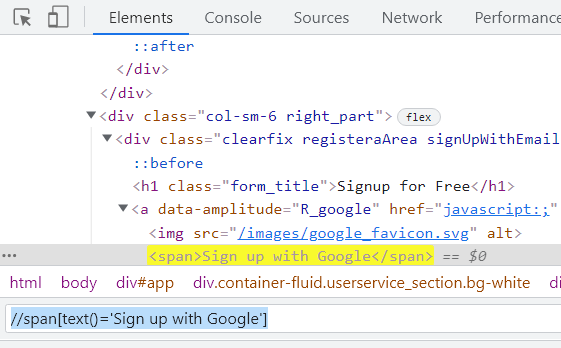
|  |  |
| --- | --- |
|  | **//tagname[text()=’Text of the Web Element’]** |

Let’s write the XPath for the Sign up with Google button on the LambdaTest Sign Up Page, as highlighted in the below image using the Text() method.

**Element**:



|  |  |
| --- | --- |
|  | **//span[text()='Sign up with Google']** |



### **XPath using Starts-With()**

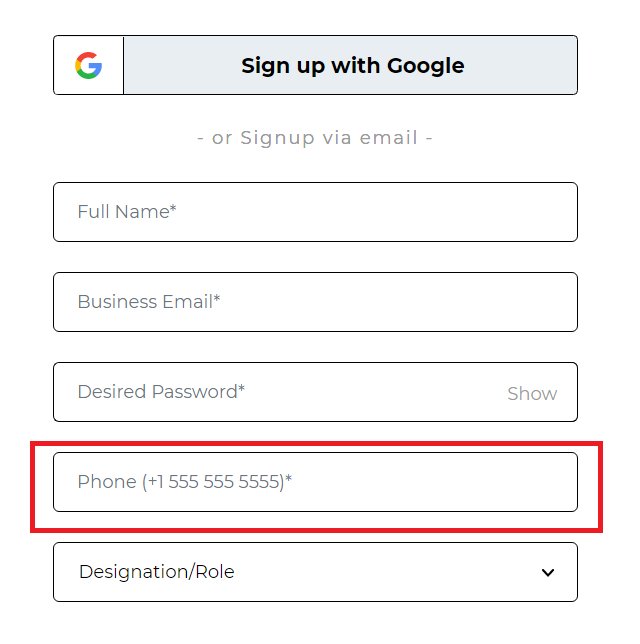
The Starts-With() method is similar to the Contains() method. It is helpful in the case of web elements whose attribute value can change dynamically. In the Starts-With method, the starting value of the attribute’s text is used for locating the element.

Below is the syntax for using Starts-With() method:

|  |  |
| --- | --- |
|  | **//tagname[starts-with(@attribute,value)]** |

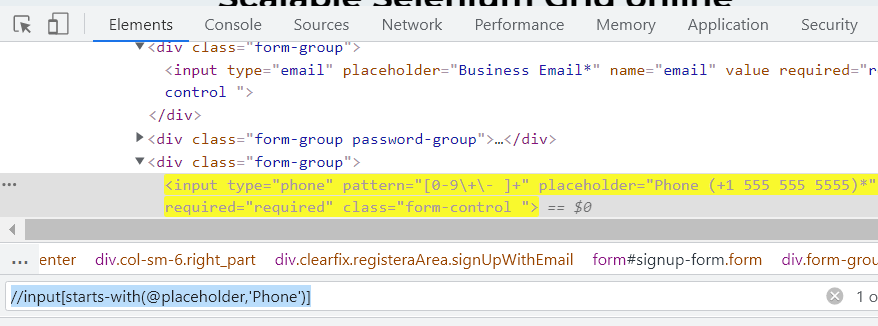
Let us locate the text box for Phone on the LambdaTest SignUp page using the starts-with method, as highlighted in the below image.

**Element**:



Here, the placeholder attribute of the Phone textbox contains a lot of characters. However, it starts with the word Phone. Hence, by using the starts-with method, you can simply locate the element in this case.

|  |  |
| --- | --- |
|  | **//input[starts-with(@placeholder,'Phone')]** |



### **XPath using Index**

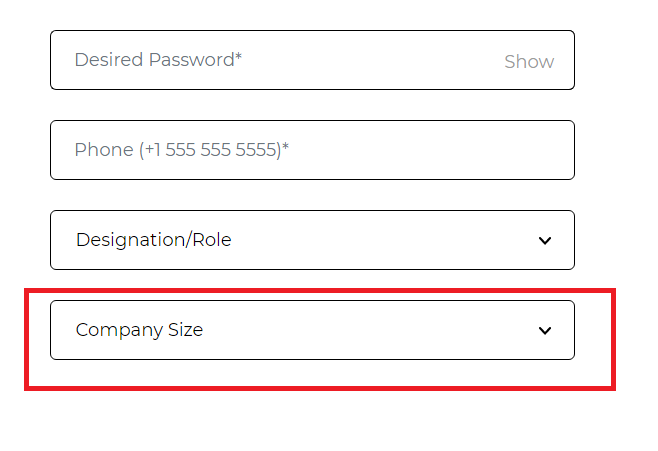
This approach comes in handy when you wish to specify a given tag name in terms of the index value you wish to locate too. For instance, consider a DOM with multiple input tags for each field value, and you wish to input text into the 4th field. In such cases, you can use the index to switch to the given tag name.

The syntax for using Index is XPath is:

|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’][Index Number]** |

Indexes can also be helpful in such cases where the same XPath is returning you multiple web elements. Let us understand how to use Indexes in such cases by using the Company Size dropdown button on the Signup page. First, we will use a generic XPath for the web element and later use an index for locating the exact element.

**Element:**



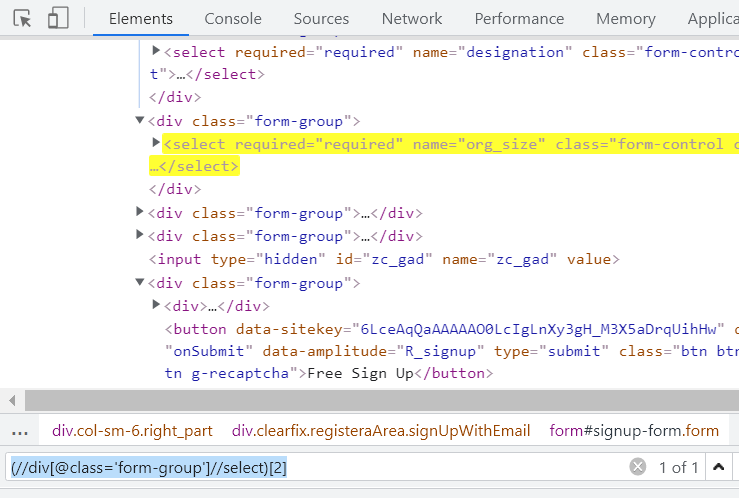
|  |  |
| --- | --- |
|  | **//div[@class='form-group']//select** |

Now, observe the Elements tab; you will see that the above XPath returns you two elements since there are two dropdowns on the web page.



However, in such cases, you can simply specify the index for the entire XPath, and bang on, you get the right element. In our case, the Company Size web element is the second dropdown option on the web page. Hence, by specifying the 2nd index for the entire XPath, I can get the Company Size web element.

|  |  |
| --- | --- |
|  | **(//div[@class='form-group']//select)[2]** |

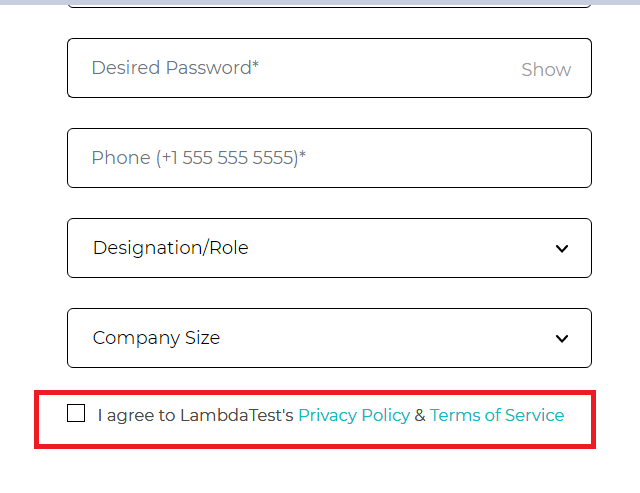


### **Chained XPath in Selenium**

As the name signifies, we can use multiple XPath expressions and chain them. The syntax for using Chained XPath is as mentioned below.

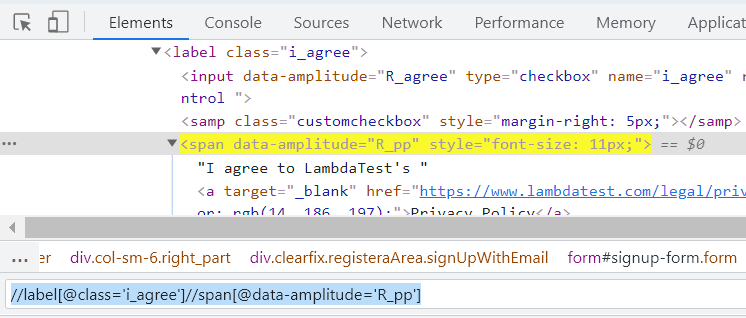
|  |  |
| --- | --- |
|  | **//tagname1[@attribute1=value1]//tagname2[@attribute2=value2]** |

Let us write a chained XPath for the I agree to LambdaTest’s Privacy Policy text after the checkbox on the LambdaTest Signup page as highlighted below.



Here, we first locate the checkbox using the label tag and then navigate to the text that follows the checkbox.

|  |  |
| --- | --- |
|  | **//label[@class='i\_agree']//span[@data-amplitude='R\_pp']** |



## How to write Xpath in Selenium using Axes methods?

XPath axes come in handy when the exact element tagname or its attribute value is dynamic and cannot be used to locate an element. In such cases locating elements after traversing through child/sibling or parent becomes an easy approach.

Some of the widely used XPath axes are:

### **XPath using Following**

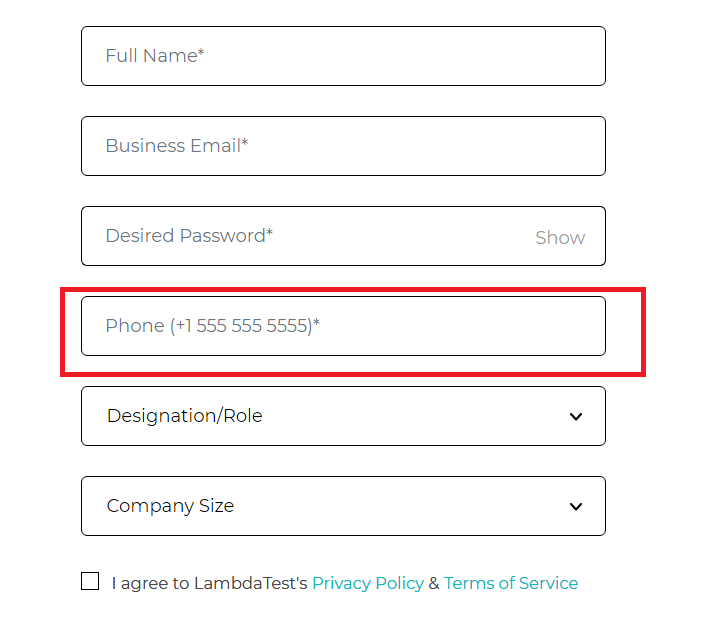
This can be used when you have a unique attribute of the tag before your actual web element. For example, on using Following, you can have all the elements that follow the current node, and you can simply use Index or another chained XPath to locate your actual web element.

The Syntax for using the Following is:

|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’]//following::tagname** |

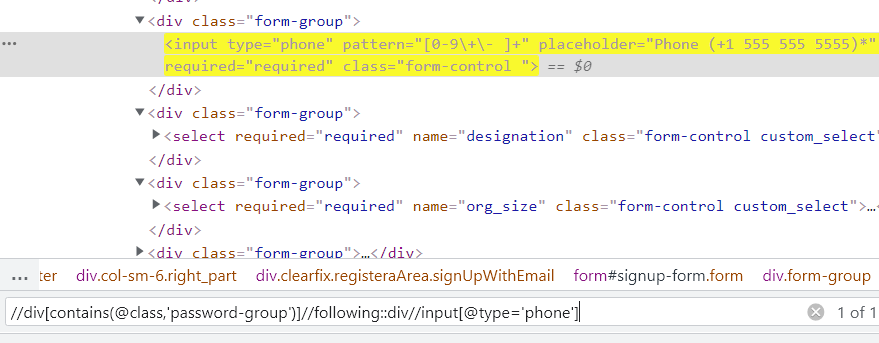
Now, let us locate the input text box for Phone on the LambdaTest signup page using the Following.

**Element**:



In this case, we have first located the div tag for the Password, and then, by using the Following, we get the list of all the div tags after Password. From there, we use the input tag and then locate the input box for the Phone.

|  |  |
| --- | --- |
|  | **//div[contains(@class,'password-group')]//following::div//input[@type='phone']** |



### **XPath using Following-Sibling**

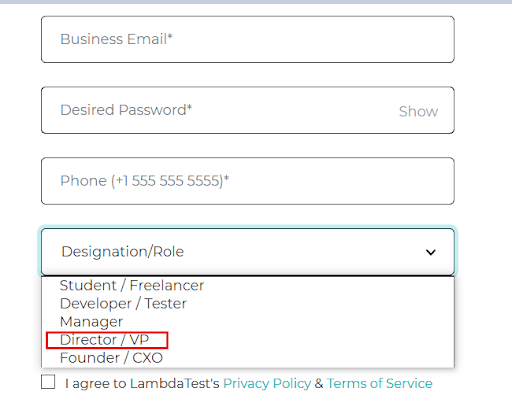
As the term signifies, siblings are those nodes that share the same parent or are at the same level. Hence, Following-Sibling will return you the node at the same level and after the current node.

The syntax for using Following-Sibling is

|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’]//following-sibiling::tagname** |

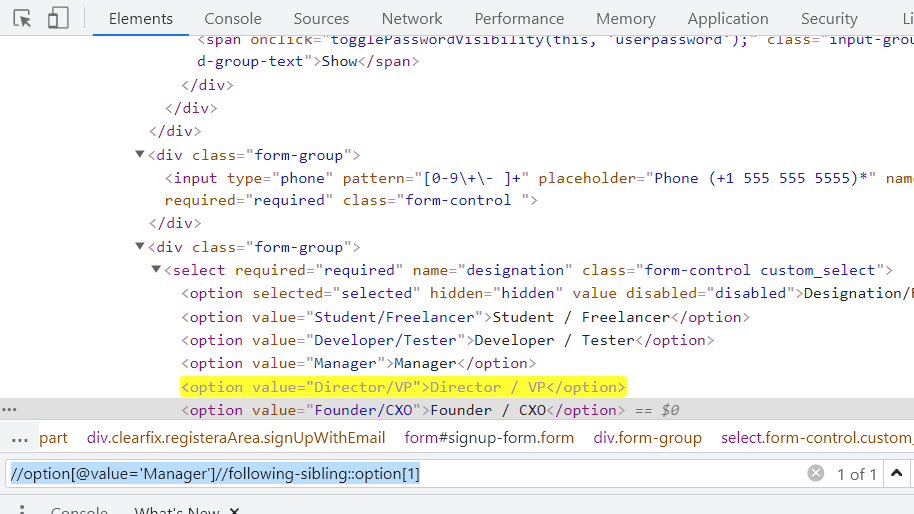
Let us understand Following-Sibling using the options present in the Designation/Role dropdown in LambdaTest Sign Up page.

**Element**:



In this case, we first locate the Manager option and later, by using the following-sibling, locate the Director/VP option, since all the option tags share the same parent, or we can say they are at the same level.

|  |  |
| --- | --- |
|  | **//option[@value='Manager']//following-sibling::option[1]]** |



However, it is important for you to understand that in the above option, the simplest way of writing this XPath would be an option[@value=’Director/VP’]. The above is one of the implementations for the same element using the following-sibling.

### **XPath using Preceding**

In contrast to the Following, this method helps locate all the elements before the current node, as in the preceding element from the current node with XPath in Selenium.

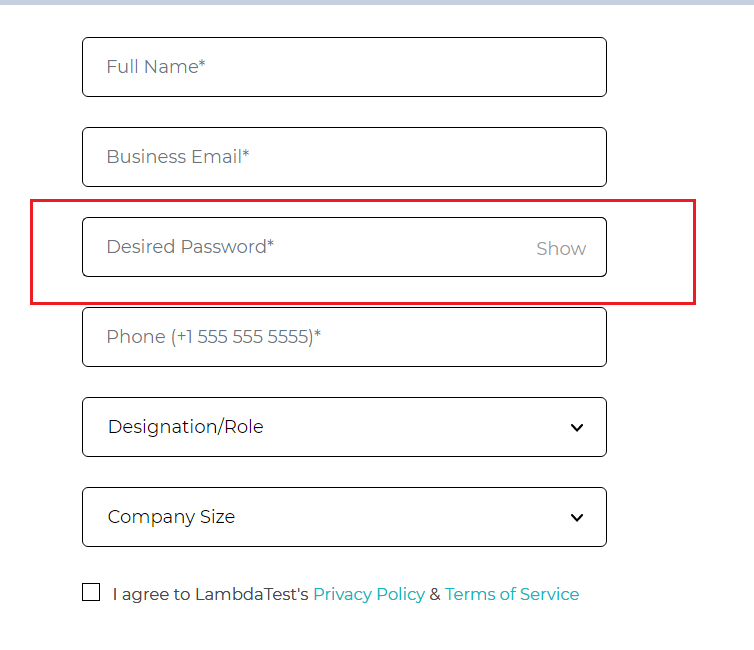
Using Preceding, you can have all the elements before your current node, and by using Index or another chained XPath, you can locate the actual web element.

The syntax for using Preceding is:

|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’]//preceding::tagname** |

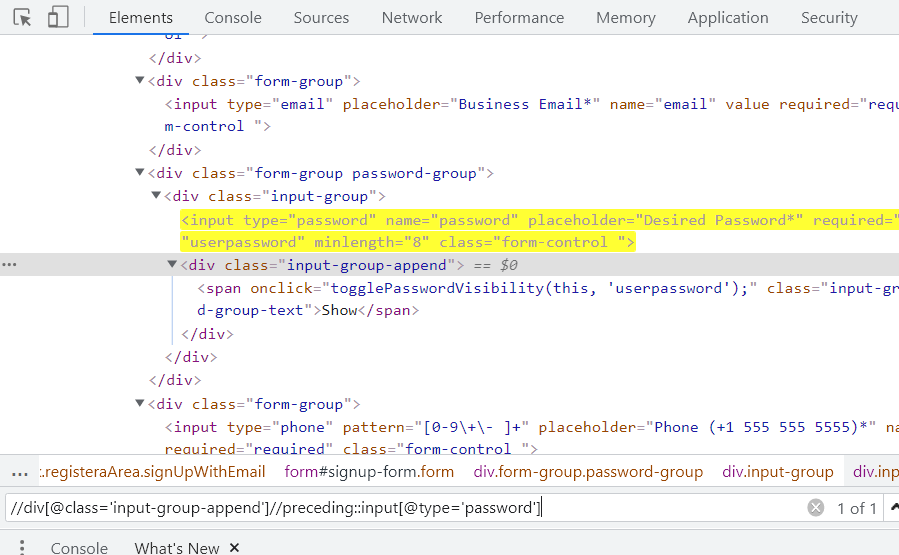
Let us write the XPath for the input text box of Password on LambdaTest Signup page using preceding.

**Element**:



In this case, I have first located the Show button next to the Password text box. From there, by using the preceding, I get all the input tags before that node, and then by using the type attribute as Password, the actual web element is located.

|  |  |
| --- | --- |
|  | **//div[@class='input-group-append']//preceding::input[@type='password']** |



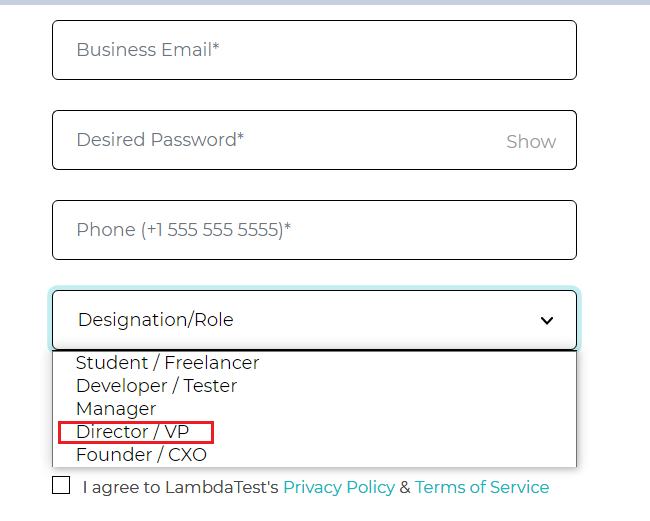
### **XPath using Preceding-Sibling**

This is a concept very similar to Following-Siblings. The only difference in functionality is that of preceding. So, here, in contrast to Following-Sibling, you get all the nodes that are siblings or at the same level but are before your current node.

The syntax for using Preceding-Sibling is:

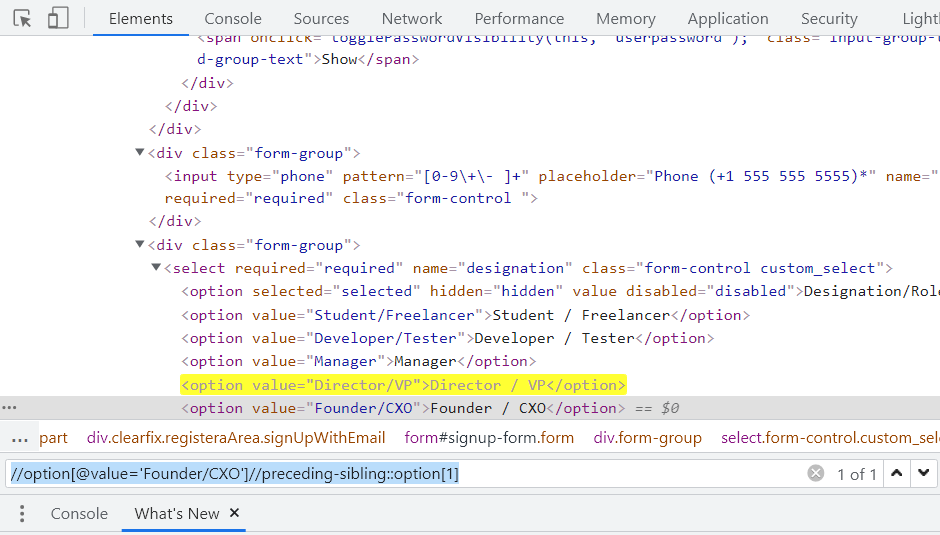
|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’]//preceding-sibling::tagname** |

Let us see the same example as in Following-Sibling for locating the below-highlighted web element using preceding-sibling.



In this case, we have first located the Founder/CXO option and later, by using preceding-sibling, navigating to the Director/VP option.

|  |  |
| --- | --- |
|  | **//option[@value='Founder/CXO']//preceding-sibling::option[1]** |



### **XPath using Child**

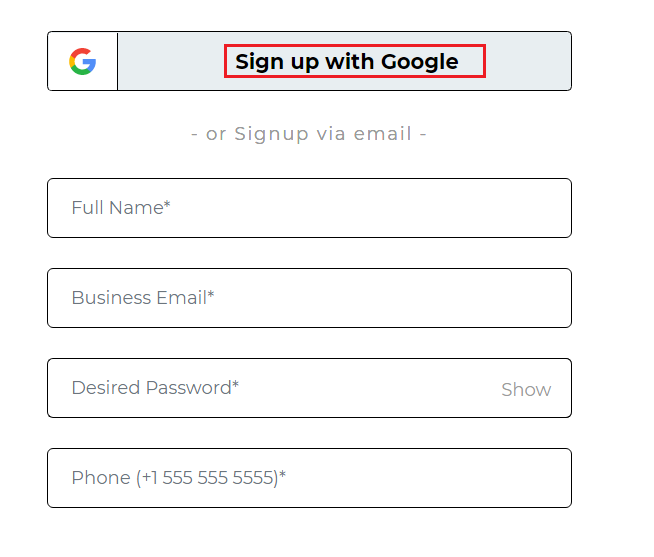
As the name specifies, this approach is used to locate all the child elements of a particular node. For example, a basic use case of this approach could be to circulate all the data in a table through the rows.

The Syntax for using Child is:

|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’]//child::tagname** |

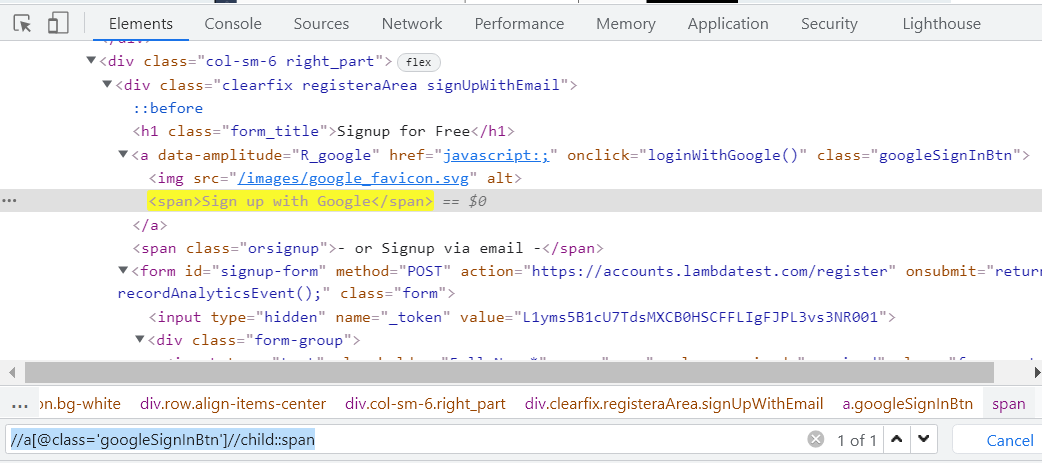
Let us write the XPath for Sign Up With Google Option using Child.

**Element**:



In this case, I first locate the Google SignIn Button node, which has two child nodes, and later by using Child and specifying the span tag.

|  |  |
| --- | --- |
|  | **//a[@class='googleSignInBtn']//child::span** |



For example, using the below referenced DOM structure, we can create an XPath in Selenium as follows:

### **XPath using Parent**

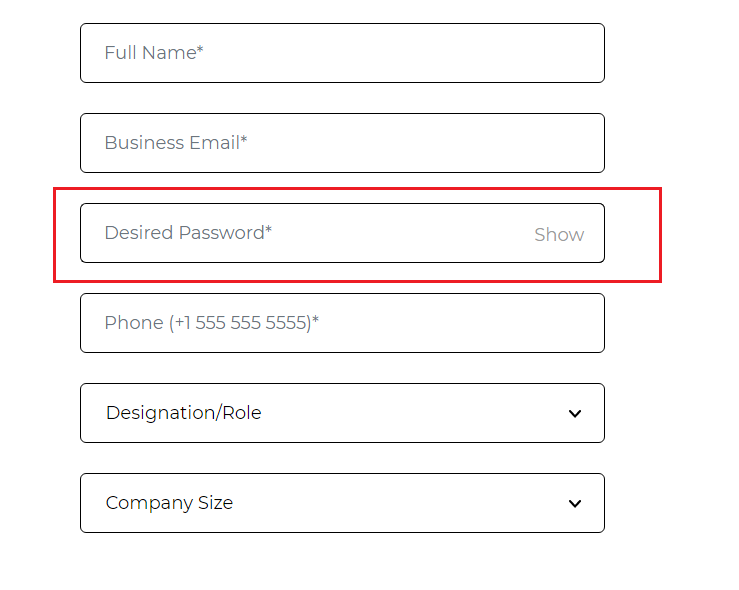
This method is used to select the parent node of the current node.

The syntax for using Parent is:

|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’]/parent::tagname** |

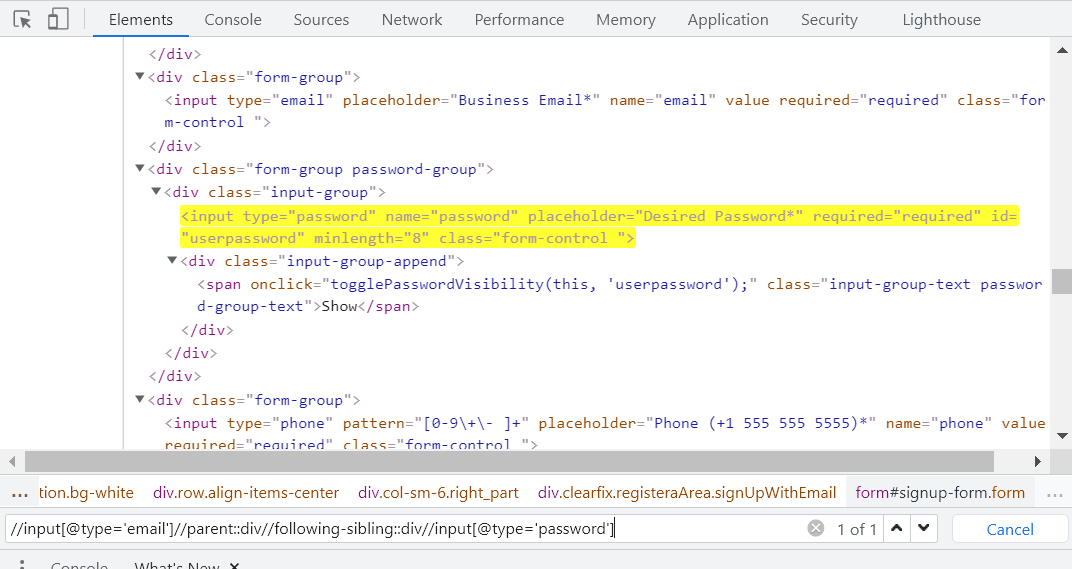
Let us see the XPath of the Password Input text box of the LambdaTest Signup page, shown in the below image.

**Element**:



In this case, I have first located the node for the Email Text box and then navigated to the parent div of it. From there, by using the following-sibling, we get all the divs at the same level, from where the desired input tag is located with the help of the type attribute.

|  |  |
| --- | --- |
|  | **//input[@type='email']//parent::div//following-sibling::div//input[@type='password']** |



### **XPath using Descendants**

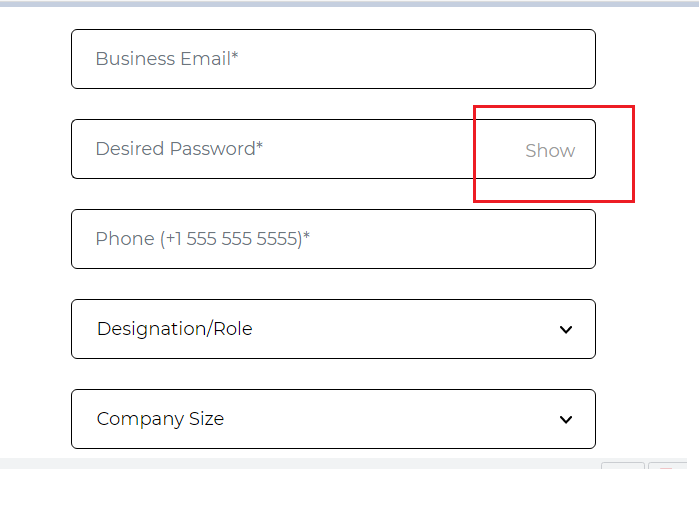
This method is used for selecting the descendants of the current node. Here, Descendants refer to the child nodes, grandchild nodes, etc.

The syntax for using Descendants is:

|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’]//descendants::tagname** |

Let us locate the Show button next to the input box for Password by making use of Descendants.

**Element**:



|  |  |
| --- | --- |
|  | **//div[contains(@class,'password-group')]//descendant::span** |



### **XPath using Ancestors**

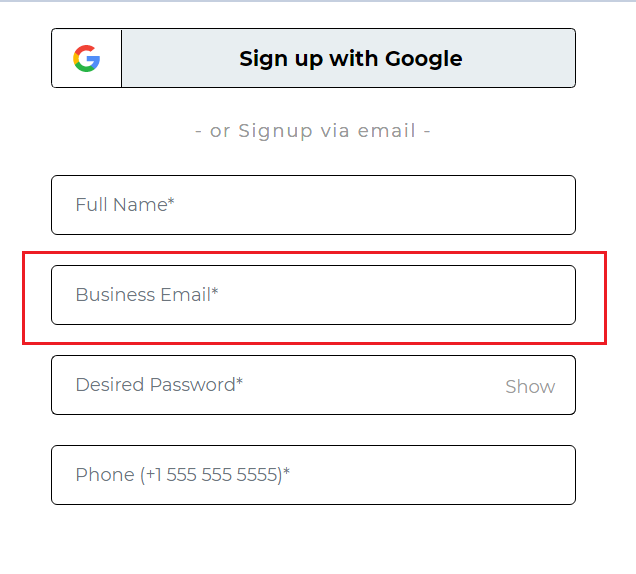
This method is used for selecting the ancestors of the current node. Here, Ancestors refer to the parent nodes, grandparent nodes, etc.

The syntax for using Ancestors is:

|  |  |
| --- | --- |
|  | **//tagname[@attribute=’value’]//ancestors::tagname** |

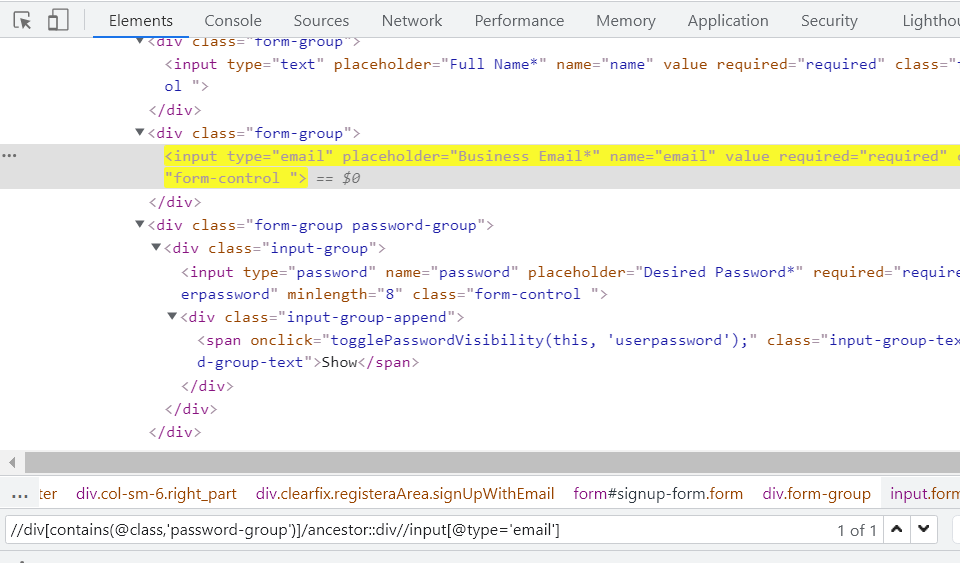
Let us write the XPath of Email Input box on the LambdaTest Signup page.

**Element**:



In this case, I first locate the div tag for the Password input box, and by using its ancestor div, locate the email input element.

|  |  |
| --- | --- |
|  | **//div[contains(@class,'password-group')]/ancestor::div//input[@type='email']** |



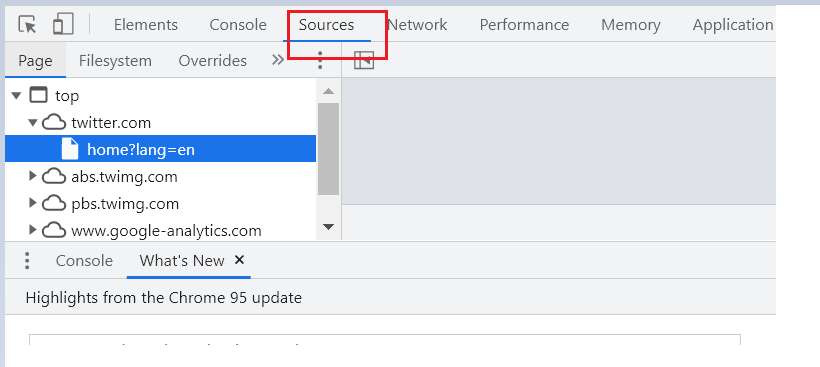
## How to capture XPath of loader images?

While automating Web pages using Selenium, I often came across a few web elements that appeared for a very short duration on the screen. By the time I would start writing the XPath, the elements would disappear.

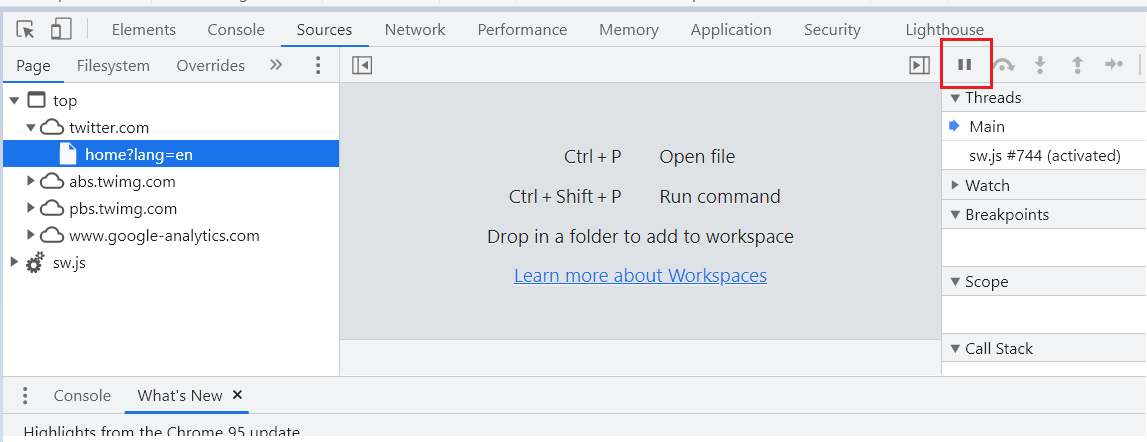
For example, think of Loading images; they do not appear on your screen for a longer period of time, and hence, identifying their locators can be tricky sometimes.

Below is a screen capture from Twitter which shows its loader image. Now how do you capture the XPath for it? Let me show you that!

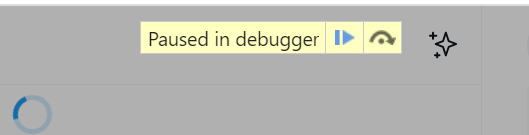
**Step 1**: As soon as the page starts to open, press F12 to be ready to inspect the element. Switch to the Sources tab as highlighted in the below image.



**Step 2**: When you see that the loading symbol appears on your screen, simply press F8 or click on the image highlighted below.

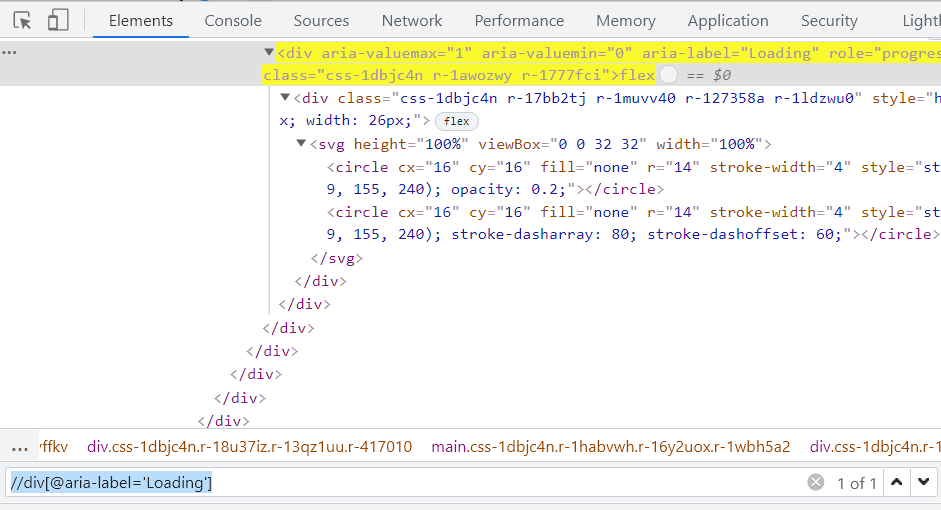


This will pause the execution, and you will see a notification badge like below on top of your screen.

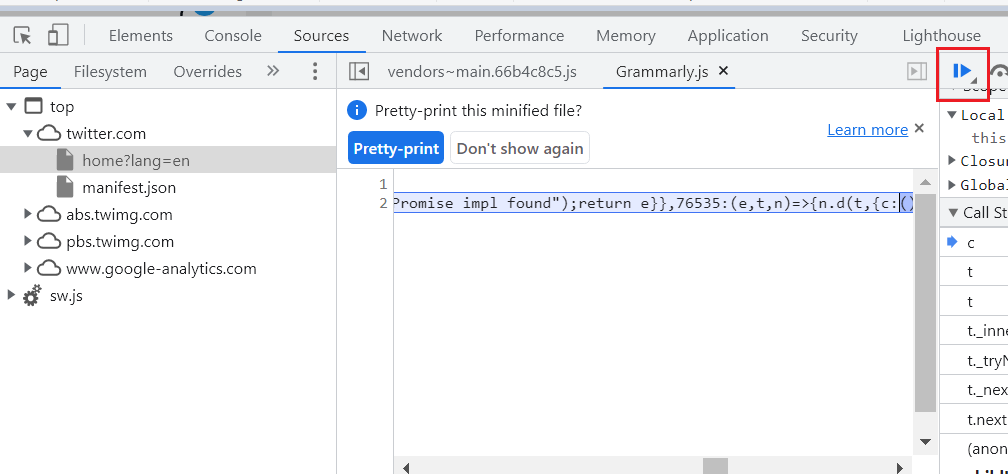


**Step 3**: Now, you can go back to the Elements tab and start writing the locator.

|  |  |
| --- | --- |
|  | **//div[@aria-label='Loading']** |



**Step 4**: Go back to the Sources tab and click on the below option to resume.



Wasn’t this easy? This way, you can simply pause the execution and let the element be on your screen till you identify the locator of it 🙂