

[◀ Back To Blog](#)

- Selenium Tutorial (<https://www.lambdatest.com/blog/category/selenium-tutorial/>) • Selenium Locators (<https://www.lambdatest.com/blog/category/selenium-locators/>)


# Complete Guide For Using XPath In Selenium With Examples



(<https://www.lambdatest.com/blog/author/riadayal/>)

**Ria Dayal**

Posted On: November 18, 2021

 523494 Views

 18 Min Read

[Home \(https://www.lambdatest.com/\)](#) > [Blog \(https://www.lambdatest.com/blog/\)](#) >

**Complete Guide For Using XPath In Selenium With Examples**

This article is a part of our Content Hub. For more in-depth resources, check out our content hub on [Selenium Locators Tutorial](https://www.lambdatest.com/learning-hub/selenium-locators) (<https://www.lambdatest.com/learning-hub/selenium-locators>).

Selenium is a free, open-source [test automation](https://www.lambdatest.com/automation-testing) (<https://www.lambdatest.com/automation-testing>) framework that can help us test an application across different platforms and browsers. However, to implement a framework using Selenium, the first step is to locate any web element. Now, when I started with my first Selenium automation framework, XPath proved to be a boon when it came to capturing web elements. Learn more about [what Is Selenium?](https://www.lambdatest.com/selenium) (<https://www.lambdatest.com/selenium>)



([/#facebook](#))



([/#twitter](#))



([/#linkedin](#))



Although XPath is not the only technique that Selenium offers to locate any element, XPath provides an option to search a web element dynamically, and hence, gives the flexibility to tweak any locator to support your requirement.



In this [Selenium testing](https://www.lambdatest.com/selenium-automation) (https://www.lambdatest.com/selenium-automation) tutorial, we will learn about the types of XPath in Selenium and how to write basic and complicated XPath. We will also see how we can capture XPath of a few tricky web elements while performing Selenium automation testing.

Starting your journey with Selenium WebDriver? Check out this step-by-step guide to perform Automation testing using [Selenium WebDriver](https://www.lambdatest.com/blog/selenium-webdriver-tutorial-with-examples/) (https://www.lambdatest.com/blog/selenium-webdriver-tutorial-with-examples/).

Let's get started!

## TABLE OF CONTENTS

- What is XPath in Selenium?
- Types of XPath in Selenium
- How to write XPath in Selenium?
- How to write Xpath in Selenium using Axes methods?
- How to capture XPath of loader images?



**LAMBDATEST**  
**Run Your Selenium Scripts On**  
**Cloud Grid with 3000+ Browsers & OS**

**Get Started**

(<https://accounts.lambdatest.com/register>)

## What is XPath in Selenium?

XPath, also known as XML Path, is one of the most commonly used [locators in Selenium WebDriver](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/) (https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/) that can help you navigate through the HTML structure of a page. It can be used for HTML and XML documents to locate any element in a web page using HTML DOM structure.

The basic format of XPath in Selenium is explained below.

([/#facebook](#)) ([/#twitter](#)) ([/#linkedin](#))



```
1 XPath = //tagname[@Attribute='Value']
```

**LAMBDATEST**

(<https://www.lambdatest.com>)



1. **//**: denotes the current node
2. **tagname**: denotes the tagname of the current node
3. **@**: is the Select attribute
4. **Attribute**: denotes the attribute of the node
5. **Value**: denotes the value of the chosen attribute

If you are a [Selenium 4](https://www.lambdatest.com/blog/what-is-deprecated-in-selenium4/) (https://www.lambdatest.com/blog/what-is-deprecated-in-selenium4/) user, this [Selenium WebDriver Tutorial](https://www.lambdatest.com/blog/selenium-webdriver-tutorial-with-examples/) (https://www.lambdatest.com/blog/selenium-webdriver-tutorial-with-examples/) for beginners and professionals will help you learn what's new in Selenium 4 (Features and Improvements). However, [Selenium 3 vs Selenium 4](https://www.lambdatest.com/blog/what-is-deprecated-in-selenium4/) (https://www.lambdatest.com/blog/what-is-deprecated-in-selenium4/) comparison could be a good starting point to know more about the Selenium 4 framework. You can also go through the [Selenium 4 tutorial](https://www.lambdatest.com/learning-hub/selenium-4) (https://www.lambdatest.com/learning-hub/selenium-4) to further deep dive into Selenium 4.



## Types of XPath in Selenium

The XPath is the language used to select elements in an HTML page. XPath can be used to locate any element on a page based on its tag name, ID, CSS class, and so on. There are two types of XPath in Selenium.

1. Absolute XPath
2. Relative XPath



(/facebook)



(/twitter)



(/linkedin)



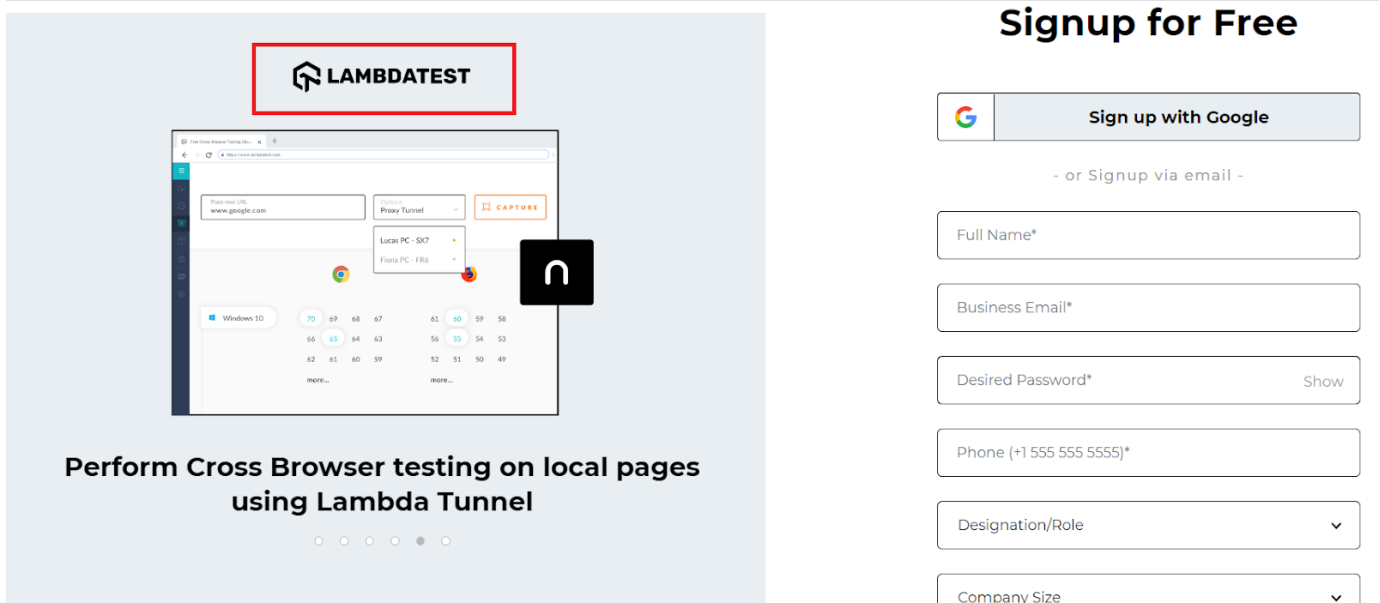
(<https://www.lambdatest.com>)

Absolute XPath is the simplest form of XPath in Selenium. It starts with a single slash '/' and provides the absolute path of an element in the entire DOM.


Let us understand writing an absolute XPath using the [LambdaTest SignUp Page](https://accounts.lambdatest.com/register/) (<https://accounts.lambdatest.com/register/>).

LambdaTest is a cloud-based [cross browser testing](https://www.lambdatest.com/feature) (<https://www.lambdatest.com/feature>) tool that supports Selenium Grid, providing a solution to every obstacle you face while performing automation testing using your local machine. Selenium testing tools like LambdaTest offer an [online Selenium Grid](https://www.lambdatest.com/selenium-grid-online) (<https://www.lambdatest.com/selenium-grid-online>) consisting of 2000+ browsers for you to perform automation testing effortlessly.

We will locate the LambdaTest page header highlighted in the below image using an absolute XPath in Selenium.



**Sign up for Free**

 **Sign up with Google**

- or Signup via email -

Full Name\*

Business Email\*

Desired Password\* Show

Phone (+1 555 555 5555)\*

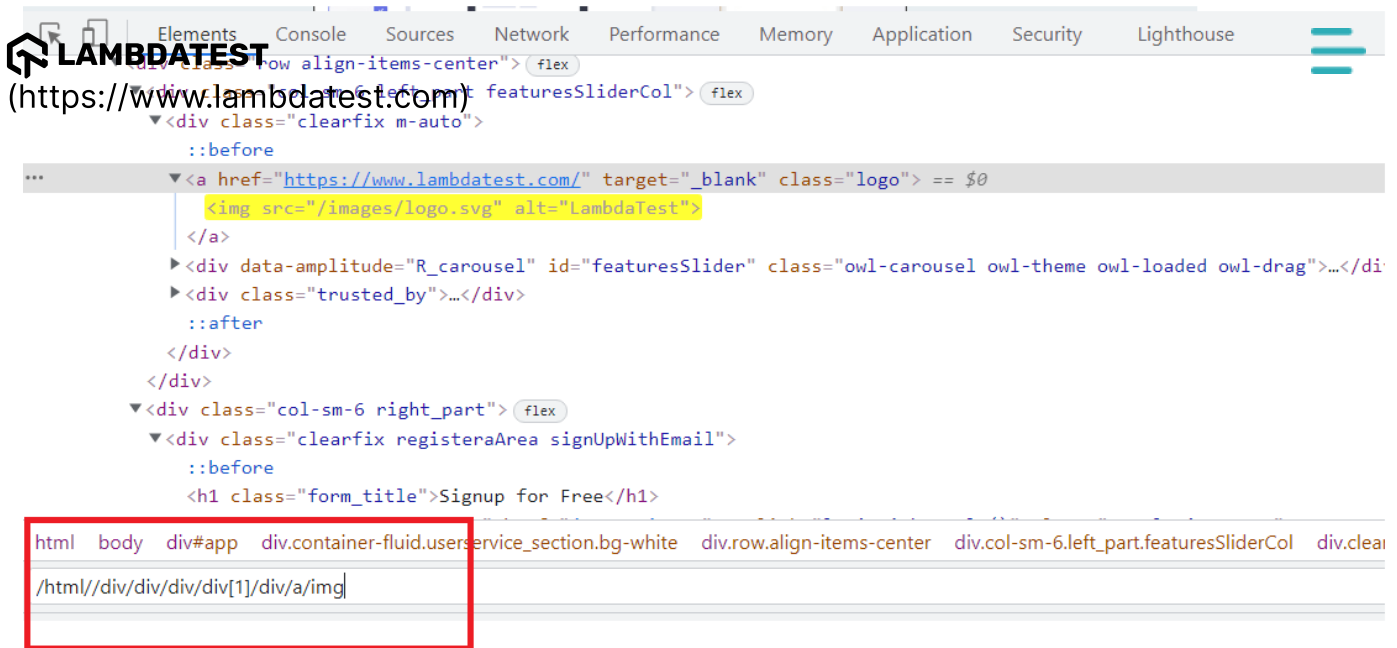
Designation/Role ▼

Company Size ▼

The below absolute XPath will help you locate the header as highlighted.

```
1 /html//div/div/div/div[1]/div/a/img
```

In order to locate the element, you can simply do a right-click on the web element and click on Inspect. Then, in the Elements tab, you can start writing the locator.



In this case, starting from the html tag, I have traversed one by one to the div, which contains the tag a and hence, the final img tag. Wasn't it pretty simple?

However, even though it is simple, the biggest disadvantage of using absolute XPath is that they are very vulnerable to any changes in the DOM structure and, as a result, can bring you a lot of automation failures.

Just imagine if a single div tag from our example is removed, and alas, this locator stops working!!

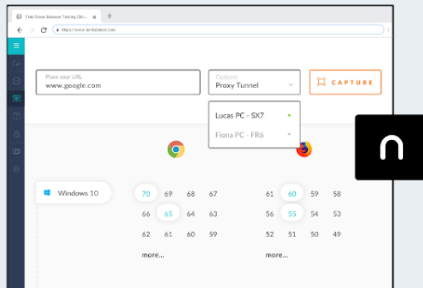
## Relative XPath

In the case of relative XPath in Selenium, the XPath expression starts from the middle of the DOM structure. It is represented by a double slash `///` denoting the current node.

It is always preferred over an absolute XPath as it is not a complete path from the root element.


We will locate the same element as in the case of absolute XPath, i.e., the [LambdaTest Sign Up page](https://accounts.lambdatest.com/register/) (`https://accounts.lambdatest.com/register/`) header.





**Perform Cross Browser testing on local pages  
using Lambda Tunnel**

## Signup for Free

 **Sign up with Google**

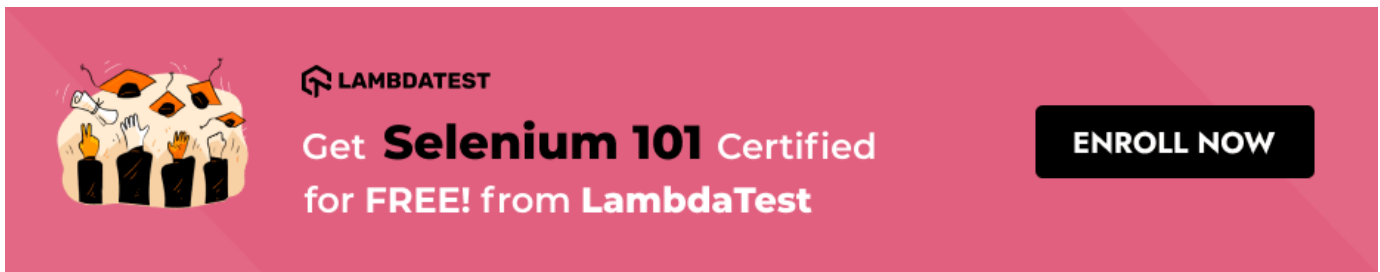
- or Signup via email -

Show

The Relative XPath of the highlighted web element will be:

```
1 //img[@alt='LambdaTest']
```

Here, I just used the corresponding img tag and its attribute alt to locate the title and wrote the corresponding relative XPath.



(<https://www.lambdatest.com/certifications/selenium-101>)

This certification is for anyone who wants to stay ahead among professionals who are growing their career in Selenium automation testing.

Here's a short glimpse of the Selenium 101 certification from LambdaTest:





# 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

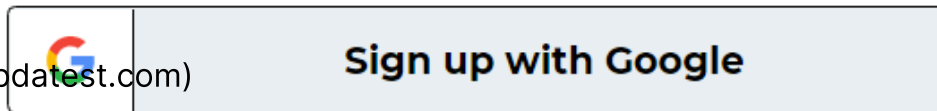
This is the common and syntactic approach to writing the XPath in Selenium, which combines a tagname and attribute value.

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

```
1 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/>) as highlighted in the below image. Here, I have used the class attribute and its 'googleSignInBtn' value of the corresponding tag.





- or Signup via email -




 [Show](#)  

☐ I agree to LambdaTest's [Privacy Policy](#) & [Terms of Service](#)

Already have an account? [Sign In](#)


2. **//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.

3. **//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.'

 (/#facebook)     (/#twitter)     (/#linkedin)





 4. **//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.  
(https://www.lambdatest.com)

5. **//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

```
1 //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.

```
1 //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:**



(/facebook)



(/twitter)



(/linkedin)



Business Email\*

Desired Password\*

Show

Phone (+1 555 555 5555)\*

Designation/Role



Company Size



☐ I agree to LambdaTest's [Privacy Policy](#) & [Terms of Service](#)

**FREE SIGN UP**




Already have an account? [Sign In](#)

```
1 //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 conditions should be true or both, whereas, in the case of AND, both the conditions should be true.

 `(/facebook)`
 `(/twitter)`
 `(/linkedin)`



(<https://www.lambdatest.com>)  
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:**

---

## Signup for Free

 Sign up with Google

- or Signup via email -

[Show](#)

**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?

```
1 //input[@name="email" or contains(@placeholder,'abc')]
```



(/facebook)



(/twitter)



(/linkedin)



```

Elements Console Sources Network Performance Memory Applic
<input type="hidden" name="_token" value="L1yms5B1cU7TdsMXCB0HSCFF
  <div class="form-group">
    <input type="text" placeholder="Full Name*" name="name" value re
  </div>
  <div class="form-group">
    <input type="email" placeholder="Business Email*" name="email" v
  </div>
  <div class="form-group password-group">...</div>
  <div class="form-group">...</div>

html body div#app div.container-fluid.userservice_section.bg-white div.row.align-items-center
//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.

```
1 //input[@name="email" and contains(@placeholder,'Email')]
```

```

Elements Console Sources Network Performance Memory Applic
<input type="hidden" name="_token" value="L1yms5B1cU7TdsMXCB0HSCFF
  <div class="form-group">
    <input type="text" placeholder="Full Name*" name="name" value re
  </div>
  <div class="form-group">
    <input type="email" placeholder="Business Email*" name="email"
  </div>
  <div class="form-group password-group">...</div>
  <div class="form-group">...</div>

html body div#app div.container-fluid.userservice_section.bg-white div.row.align-items-center
//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 substantial attribute value used via Starts with or Contains.



The Syntax for using text() in XPath is:



`1 //span[text()='Sign up with Google']`

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:

# Signup for Free



- or Signup via email -

Full Name\*

Business Email\*

Desired Password\*

Show

`1 //span[text()='Sign up with Google']`



(/facebook)

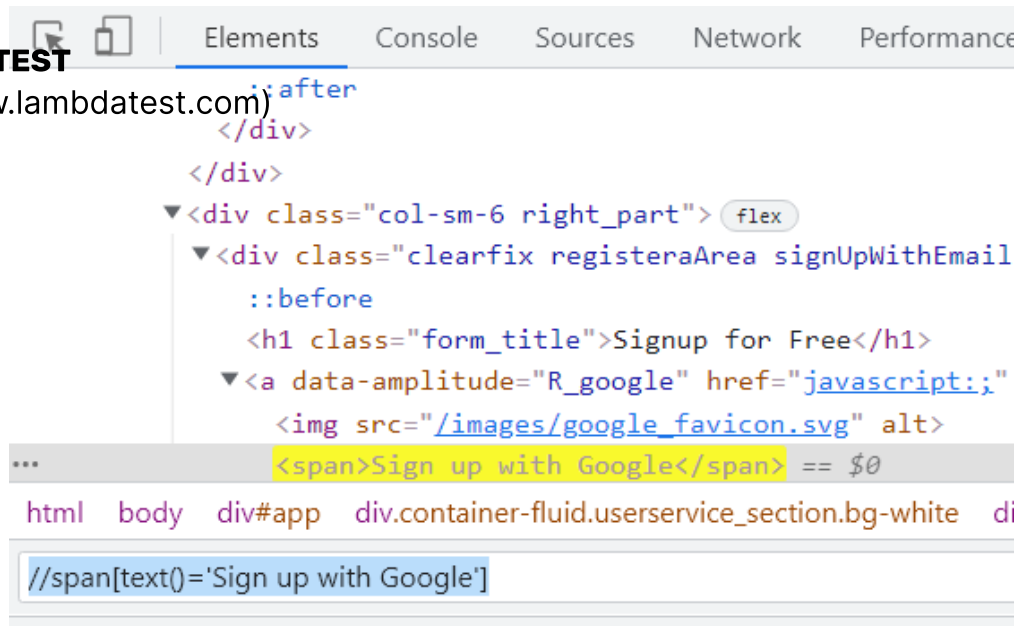


(/twitter)



(/linkedin)





Also Read - [SelectorsHub: The Next Gen XPath, CSS Selectors Tool](https://www.lambdatest.com/blog/selectorshub-the-next-gen-xpath-css-selectors-tool/)  
<https://www.lambdatest.com/blog/selectorshub-the-next-gen-xpath-css-selectors-tool/>

## 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:

```
1 //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:**





Sign up with Google

- or Signup via email -





Show

▼

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.

```
1 //input[starts-with(@placeholder, 'Phone')]
```

Elements

Console

Sources

Network

Performance

Memory

Application

Security

<div class="form-group">

<input type="email" placeholder="Business Email\*" name="email" value required="r

control ">

</div>

><div class="form-group password-group">...</div>

><div class="form-group">

<input type="phone" pattern="[0-9\+\- ]+" placeholder="Phone (+1 555 555 5555)\*"

required="required" class="form-control "> == \$0

... enter div.col-sm-6.right\_part div.clearfix.registeraArea.signUpWithEmail form#signup-form.form div.form-grou

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

f


(/#facebook)

t

(/#twitter)

in

(/#linkedin)



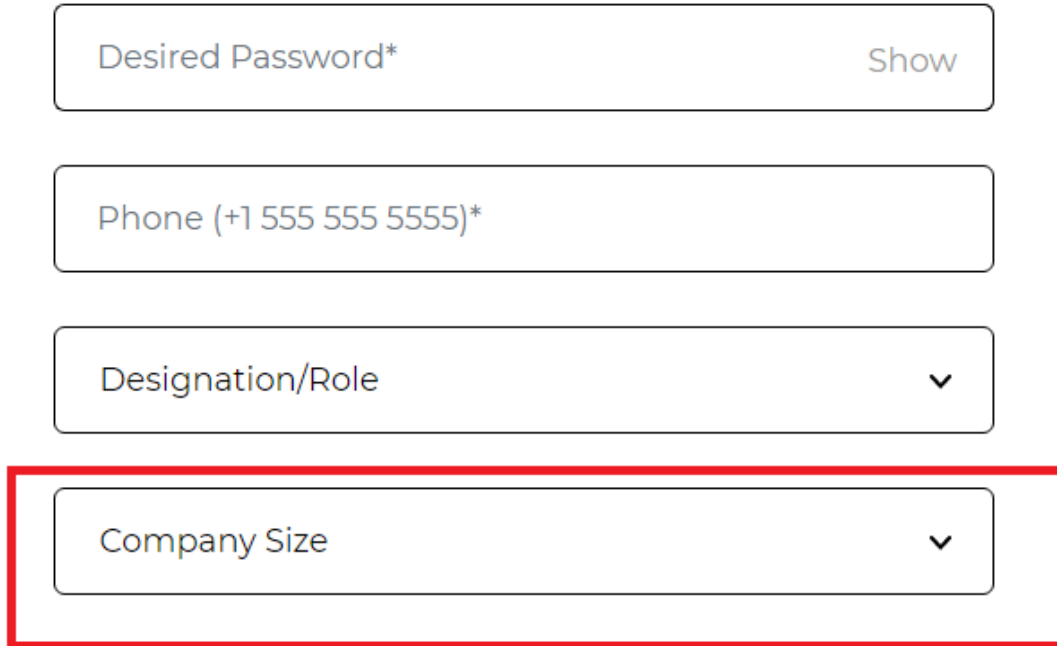
This approach comes in handy when you wish to specify a given tag name in terms of the index value you wish to locate too. For instance, consider a DOM with multiple input tags for each field value, and you wish to input text into the 4th field. In such cases, you can use the index to switch to the given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

The syntax for using Index in XPath is:

```
1 //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:**



Desired Password\* Show

Phone (+1 555 555 5555)\*

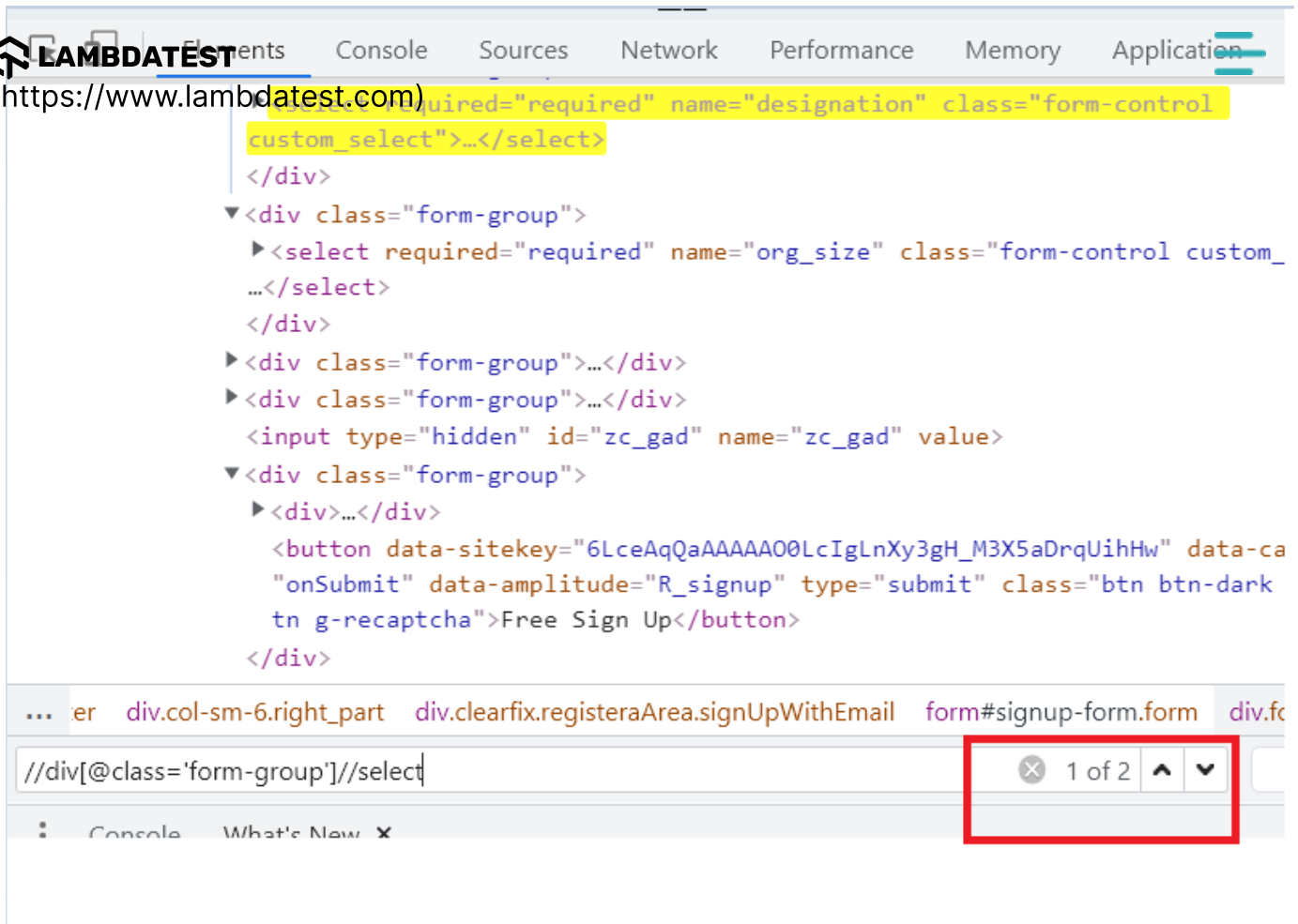
Designation/Role ▼

Company Size ▼

```
1 //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.

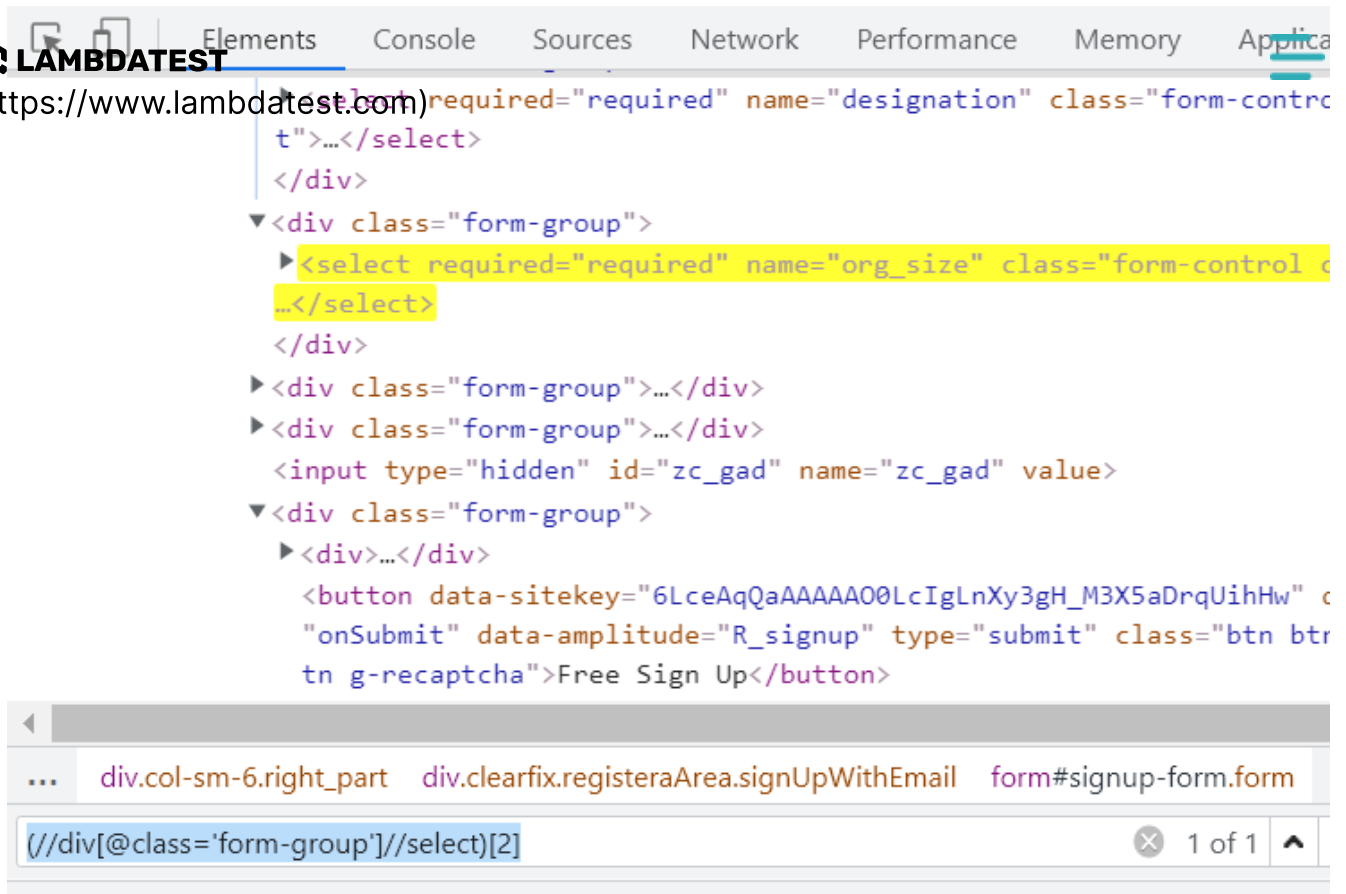




The screenshot shows the Chrome DevTools interface. The top bar includes tabs for Elements, Console, Sources, Network, Performance, Memory, and Application. The 'Elements' tab is active, displaying a tree view of the DOM. The selected element is a `<select>` tag with attributes `required="required"`, `name="org_size"`, and `class="form-control custom_select"`. The tree view shows a hierarchy of `div` elements with `class="form-group"`. Below the tree view, the 'XPath' pane shows the selected element's path: `//div[@class='form-group']//select`. A red box highlights the '1 of 2' indicator and the expand/collapse icons in the XPath pane.

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.

```
1 (//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.

```
1 //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.



Desired Password\*

Show

 Phone (+1 555 555 5555)\*

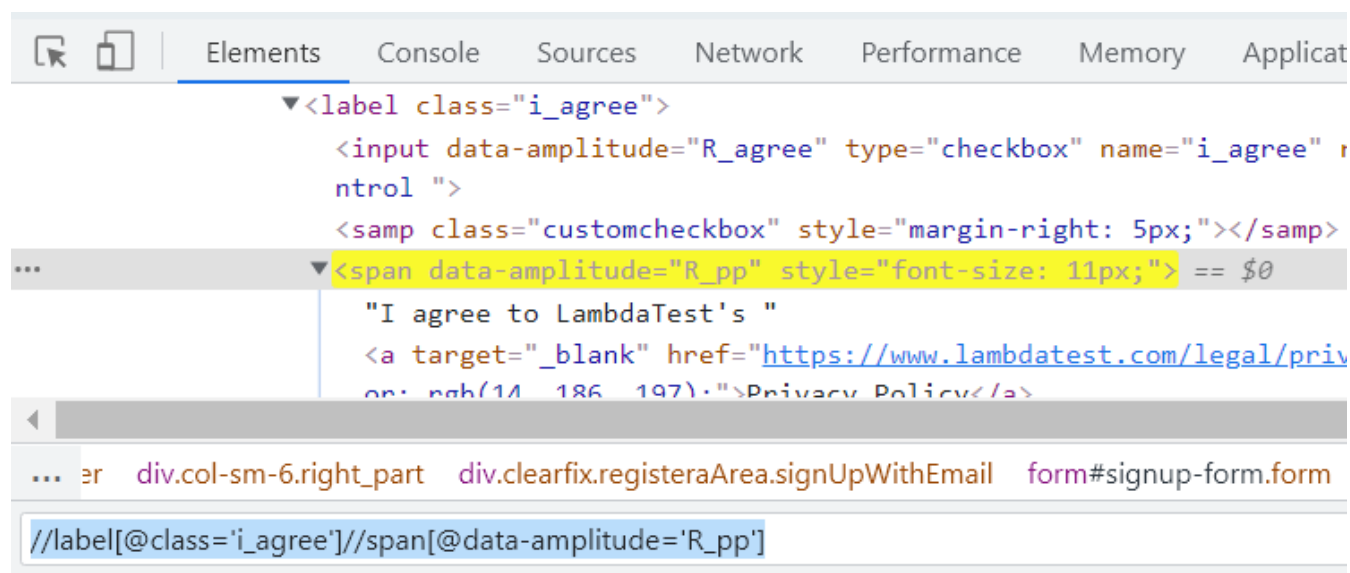
 Designation/Role

 Company Size

☐ I agree to LambdaTest's [Privacy Policy](#) & [Terms of Service](#)

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

```
1 //label[@class='i_agree']//span[@data-amplitude='R_pp']
```



Also read – [Selenium Locators in Protractor](https://www.lambdatest.com/blog/selenium-locators-in-protractor/) (<https://www.lambdatest.com/blog/selenium-locators-in-protractor/>)

# How to write Xpath in Selenium using Axes methods?

[\(/#facebook\)](#)
[\(/#twitter\)](#)
[in](#)  
[\(/#linkedin\)](#)


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:

```
1 //tagname[@attribute='value']//following::tagname
```

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

**Element:**

Full Name\*

Business Email\*

Desired Password\* Show

Phone (+1 555 555 5555)\*

Designation/Role ▼

Company Size ▼



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. (https://www.lambdatest.com)

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

...
<div class="form-group">
  <input type="phone" pattern="[0-9\+\- ]+" placeholder="Phone (+1 555 555 5555)*"
  required="required" class="form-control "> == $0
</div>
<div class="form-group">
  <select required="required" name="designation" class="form-control custom_select"
  </div>
<div class="form-group">
  <select required="required" name="org_size" class="form-control custom_select">...
  </div>
  <div class="form-group">...</div>
... ter div.col-sm-6.right_part div.clearfix.registArea.signUpWithEmail form#signup-form.form div.form-group

//div[contains(@class,'password-group')]/following::div//input[@type='phone'] 1 of 1
```

## 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

```
1 //tagname[@attribute='value']/following-sibling::tagname
```

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

**Element:**



Business Email\*

Desired Password\* Show

Phone (+1 555 555 5555)\*

Designation/Role ▼

- Student / Freelancer
- Developer / Tester
- Manager
- Director / VP**
- Founder / CXO

☐ I agree to LambdaTest's [Privacy Policy](#) & [Terms of Service](#)

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.

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

The screenshot shows the Chrome DevTools interface. The 'Elements' tab is active, displaying the DOM tree of the registration form. The following structure is visible:

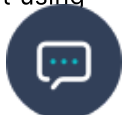
```

<div class="form-group">
  <input type="phone" pattern="[0-9\+\- ]+" placeholder="Phone (+1 555 555 5555)*" name="phone" required="required" class="form-control">
</div>
<div class="form-group">
  <select required="required" name="designation" class="form-control custom_select">
    <option selected="selected" hidden="hidden" value disabled="disabled">Designation/Role</option>
    <option value="Student/Freelancer">Student / Freelancer</option>
    <option value="Developer/Tester">Developer / Tester</option>
    <option value="Manager">Manager</option>
    <option value="Director/VP">Director / VP</option>
    <option value="Founder/CXO">Founder / CXO</option>
  </select>
</div>

```

The XPath expression `//option[@value='Manager']/following-sibling::option[1]` is entered in the search bar at the bottom of the DOM tree, and the 'Director / VP' option is highlighted as the first sibling following the 'Manager' option.

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.



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:

```
1 //tagname[@attribute='value']//preceding::tagname
```

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

**Element:**

---

Full Name\*

Business Email\*

Desired Password\*

Show

Phone (+1 555 555 5555)\*

Designation/Role

▼

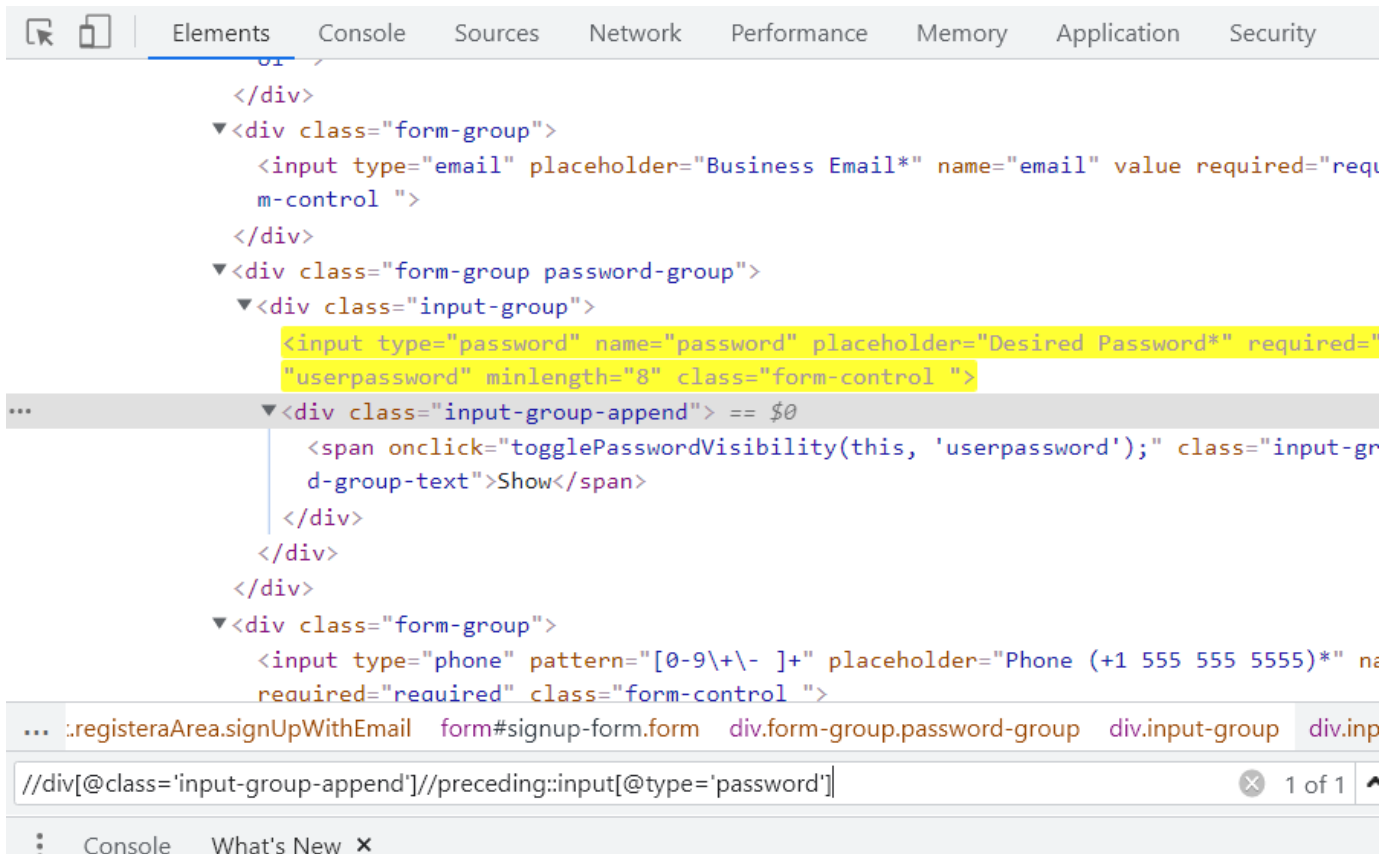
Company Size

▼

☐ I agree to LambdaTest's [Privacy Policy](#) & [Terms of Service](#)

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.

```
1 //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:

```
1 //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.





Business Email\*

Desired Password\*

Show

Phone (+1 555 555 5555)\*

Designation/Role



Student / Freelancer

Developer / Tester

Manager

Director / VP

Founder / CXO

I agree to LambdaTest's [Privacy Policy](#) & [Terms of Service](#)

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

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



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:

```
1 //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.

```
1 //a[@class='googleSignInBtn']//child::span
```



(/facebook)



(/twitter)



(/linkedin)



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:

```
1 //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:**



(/facebook)



(/twitter)



(/linkedin)



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.

```
1 //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:

```
1 //tagname[@attribute='value']//descendants::tagname
```

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

**Element:**



(/#facebook)



(/#twitter)



(/#linkedin)





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



This method is used for selecting the ancestors of the current node. Here, Ancestors refer to the parent nodes, grandparent nodes, etc.  
(https://www.lambdatest.com)



The syntax for using Ancestors is:

```
1 //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.

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



(/facebook)



(/twitter)



(/linkedin)



Also Read – [Locating Elements by TagName In Selenium](https://www.lambdatest.com/blog/locating-elements-by-tagname-in-selenium/)  
(<https://www.lambdatest.com/blog/locating-elements-by-tagname-in-selenium/>)

## 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.

```
1 //div[@aria-label='Loading']
```



(/facebook)



(/twitter)



(/linkedin)





**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 😊

Also Read – [Most Exhaustive XPath Locators Cheat Sheet](https://www.lambdatest.com/blog/most-exhaustive-xpath-locators-cheat-sheet/) (https://www.lambdatest.com/blog/most-exhaustive-xpath-locators-cheat-sheet/)

## Conclusion



(/facebook)




(/twitter)



(/linkedin)



In this [Selenium locators tutorial](https://www.lambdatest.com/learning-hub/selenium-locators) (<https://www.lambdatest.com/learning-hub/selenium-locators>) on  LAMBDATEST (<https://www.lambdatest.com>) XPath in Selenium, we learned all about XPath and what their types are. Then, we worked on writing some simple XPath using contains() or starts-with(), and we also wrote a few complicated ones using Following or Preceding.

Writing an XPath is the foundation stone for building or contributing to a Selenium automation framework, and when it comes to XPath, there could be a number of ways a simple XPath can be written. However, choosing the right one is the most important here as that determines the stability of your test cases and overall framework. Although there are a lot of plugins and extensions available today which can help you with XPath. You can go through the [10 Of The Best Chrome Extensions](https://www.lambdatest.com/blog/chrome-extensions-to-find-xpath-in-selenium/) (<https://www.lambdatest.com/blog/chrome-extensions-to-find-xpath-in-selenium/>) to find XPath in Selenium. I believe we must always opt for writing XPath on our own as we can decide which XPath would make our script the most stable, plus learning XPath can be fun 😊

I hope this article helped you in learning something more about XPath today.

Happy Testing !!

## Frequently Asked Questions

### What is XPath in Selenium?

XPath is a way to navigate the DOM (Document Object Model) of XML and HTML. XPath in Selenium allows you to run queries against localized documents even if they don't share the same base tag name.

### How do you find XPath?

Visit the name tab, right-click, and inspect. In this case, it will show you a text input tag and attributes such as "id" and "class." Use this information to construct XPath code which will locate the first name field.

 (/#facebook)

 (/#twitter)

 (/#linkedin)



## Which locator is faster in Selenium?

IDs are the most precise of all locators and should always be your first choice. IDs are supposed to be unique to each element, so they work even if you change the document's structure by adding, removing, or manipulating elements.

## What is local name () in XPath?

The XPath local-name function will return the name of the first node in a given set. It accepts string arguments.

## What is difference between CSS and XPath?

The main difference between XPath and CSS Selectors is that with the former we can navigate in both directions, while to use a CSS selector we must move forward.



### Author's Profile

[\(/#facebook\)](#)[\(/#twitter\)](#)[\(/#linkedin\)](#)



(<https://www.lambdatest.com/blog/author/riadayal/>)

## Ria Dayal

A Senior Quality Engineer By Profession, an automation enthusiast and loves to anchor. Her expertise revolves around Selenium, Java, Rest Assured, and Jenkins. Shell scripting too interests her a lot. Ria enjoys reading novels and writing is her comfort zone.

### Blogs: 11



(<https://www.linkedin.com/in/riadayal-18112485/>)

Got Questions? Drop them on LambdaTest Community.

### Test your websites, web-apps or mobile apps seamlessly with

- Selenium, Cypress, Playwright & Puppeteer Testing
- Real Devices Cloud
- Native App Testing
- Appium Testing
- Live Interactive Testing
- Smart Visual UI Testing

Book a Demo (<https://www.lambdatest.com/demo>)

## Related Articles



([/#facebook](#))



([/#twitter](#))



([/#linkedin](#))



(<https://www.lambdatest.com/blog/selenium-locators-cheat-sheet/>)

## **The Most Comprehensive Selenium Locators Cheat Sheet** (<https://www.lambdatest.com/blog/selenium-locators-cheat-sheet/>)



**Amrita Angappa**  
June 10, 2022

(<https://www.lambdatest.com/blog/author/amrita-a/>)

    
([/#facebook](#))  171637 Views ([/#twitter](#)) ([/#linkedin](#))





18 Min Read



**LAMBDATEST**



(<https://www.lambdatest.com/blog/category/cheat-sheet/>) | Selenium Tutorial  
(<https://www.lambdatest.com/blog/category/selenium-tutorial/>) |

(<https://www.lambdatest.com/blog/build-and-execute-selenium-projects/>)

## How To Build And Execute Selenium Projects

(<https://www.lambdatest.com/blog/build-and-execute-selenium-projects/>)



([/#facebook](#))





([/#twitter](#))

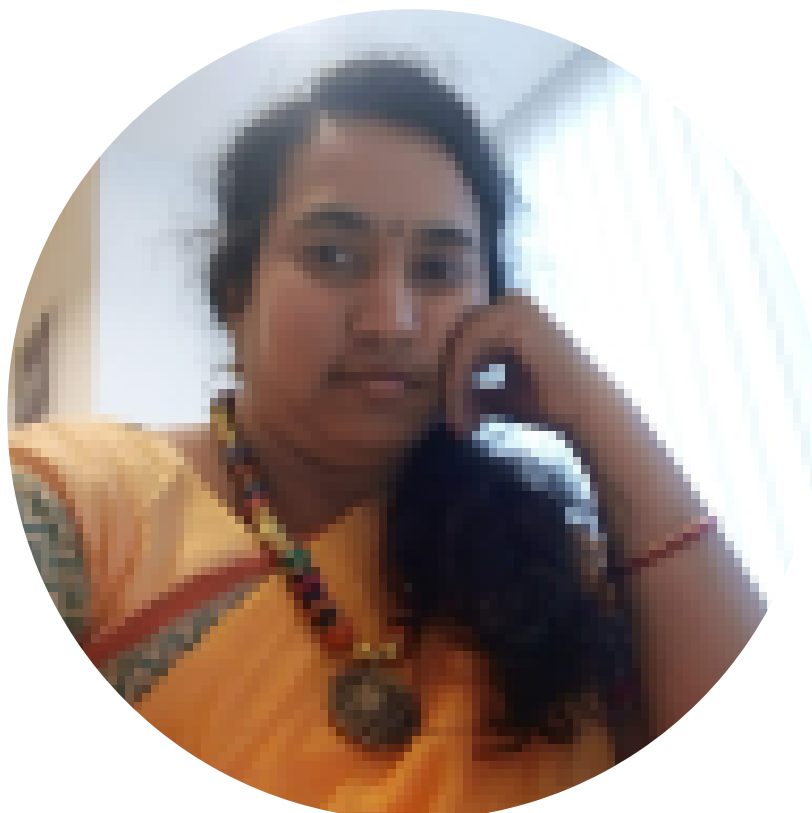


([/#linkedin](#))



<https://www.lambdatest.com/blog/author/veenadevi/> 537241 Views 22 Min Read[Automation \(https://www.lambdatest.com/blog/category/automation/\)](https://www.lambdatest.com/blog/category/automation/) | [Selenium Tutorial](#)<https://www.lambdatest.com/blog/category/selenium-tutorial/> |[\(https://www.facebook.com/lambdatest\)](#)[\(https://twitter.com/lambdatest\)](#)[\(https://www.linkedin.com/company/lambdatest\)](#)[\(https://www.lambdatest.com/blog/selenium-tutorial-one-server-and-selenium-server/\)](#)







**Veena Devi**

January 25, 2022

(<https://www.lambdatest.com/blog/author/veenadevi/>)

 412778 Views

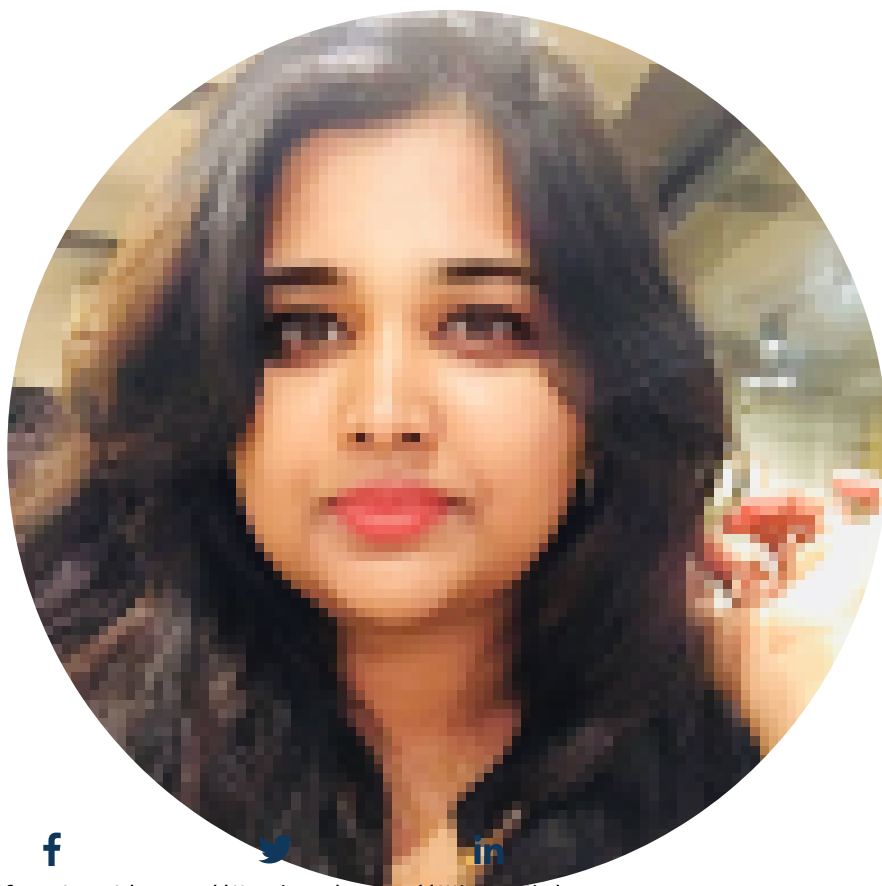
 12 Min Read

---

Automation (<https://www.lambdatest.com/blog/category/automation/>) | Selenium Tutorial  
(<https://www.lambdatest.com/blog/category/selenium-tutorial/>) |

(<https://www.lambdatest.com/blog/how-to-write-test-scripts-in-selenium/>)

## How to Write Test Scripts in Selenium (<https://www.lambdatest.com/blog/how-to-write-test-scripts-in-selenium/>)



**Shalini Baskaran**  
December 10, 2021



([/#facebook](#))



([/#twitter](#))



([/#linkedin](#))



(<https://www.lambdatest.com/blog/author/shalini-baskaran/>)



(<https://www.lambdatest.com>)



17 Min Read



Automation (<https://www.lambdatest.com/blog/category/automation/>) | Selenium Tutorial  
(<https://www.lambdatest.com/blog/category/selenium-tutorial/>) |

(<https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/>)

## Different Types Of Locators In Selenium WebDriver

(<https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/>)



([/#facebook](#))



([/#twitter](#))





([/#linkedin](#))



**Veena Devi**

November 15, 2021

<https://www.lambdatest.com/blog/author/veenadevi/> 421294 Views 28 Min Read

---

Selenium Tutorial (<https://www.lambdatest.com/blog/category/selenium-tutorial/>) | Selenium Locators  
(<https://www.lambdatest.com/blog/category/selenium-locators/>) |


<https://www.lambdatest.com/blog/shift-left-testing-approach/> (#Facebook)<https://www.lambdatest.com/blog/shift-left-testing-approach/> (#Twitter)<https://www.lambdatest.com/blog/shift-left-testing-approach/> (#LinkedIn)



Ria Dayal

November 9, 2021

(<https://www.lambdatest.com/blog/author/riadayal/>)

 97614 Views

 20 Min Read

Selenium Tutorial (<https://www.lambdatest.com/blog/category/selenium-tutorial/>) |

- Book a Demo  
(<https://www.lambdatest.com/demo>)
- Call Us (tel:+1-(866)-430-7087)
- Contact Us  
(<https://www.lambdatest.com/blog/>)

Help & Support



([/#facebook](#))



([/#twitter](#))



([/#linkedin](#))





## Products & Features

Automation Testing ( <a href="https://www.lambdatest.com/automation">https://www.lambdatest.com/automation</a> )	Cross Browser Testing ( <a href="https://www.lambdatest.com/features/cross-browser-testing">https://www.lambdatest.com/features/cross-browser-testing</a> )	Real Device Cloud ( <a href="https://www.lambdatest.com/real-device-cloud">https://www.lambdatest.com/real-device-cloud</a> )
Mobile App Testing ( <a href="https://www.lambdatest.com/mobile-app-testing">https://www.lambdatest.com/mobile-app-testing</a> )	HyperExecute ( <a href="https://www.lambdatest.com/hyperexecute">https://www.lambdatest.com/hyperexecute</a> )	LT Browser ( <a href="https://www.lambdatest.com/lt-browser">https://www.lambdatest.com/lt-browser</a> )
LT Debug ( <a href="https://www.lambdatest.com/lt-debug">https://www.lambdatest.com/lt-debug</a> )	Local Page Testing ( <a href="https://www.lambdatest.com/local-page-testing">https://www.lambdatest.com/local-page-testing</a> )	Automated Screenshots ( <a href="https://www.lambdatest.com/automated-screenshot">https://www.lambdatest.com/automated-screenshot</a> )
Geo-Location Testing ( <a href="https://www.lambdatest.com/geolocation-testing">https://www.lambdatest.com/geolocation-testing</a> )	Accessibility Testing ( <a href="https://www.lambdatest.com/accessibility-testing">https://www.lambdatest.com/accessibility-testing</a> )	Responsive Testing ( <a href="https://www.lambdatest.com/responsive-test-online">https://www.lambdatest.com/responsive-test-online</a> )
Localization Testing ( <a href="https://www.lambdatest.com/localization-testing">https://www.lambdatest.com/localization-testing</a> )	Visual Regression Testing ( <a href="https://www.lambdatest.com/visual-regression-testing">https://www.lambdatest.com/visual-regression-testing</a> )	Integrations ( <a href="https://www.lambdatest.com/integrations">https://www.lambdatest.com/integrations</a> )
Test Analytics ( <a href="https://www.lambdatest.com/test-analytics">https://www.lambdatest.com/test-analytics</a> )		

## Browser Automation

Selenium Testing ( <a href="https://www.lambdatest.com/selenium-automation">https://www.lambdatest.com/selenium-automation</a> )	Selenium Grid ( <a href="https://www.lambdatest.com/selenium-grid-online">https://www.lambdatest.com/selenium-grid-online</a> )	Cypress Testing ( <a href="https://www.lambdatest.com/cypress-testing">https://www.lambdatest.com/cypress-testing</a> )
Playwright Testing ( <a href="https://www.lambdatest.com/playwright-testing">https://www.lambdatest.com/playwright-testing</a> )	Puppeteer Testing ( <a href="https://www.lambdatest.com/puppeteer-testing">https://www.lambdatest.com/puppeteer-testing</a> )	

## Test on

List of Browsers ( <a href="https://www.lambdatest.com/list-of-browsers">https://www.lambdatest.com/list-of-browsers</a> )	Internet Explorer ( <a href="https://www.lambdatest.com/test-on-internet-explorer-browsers">https://www.lambdatest.com/test-on-internet-explorer-browsers</a> )	Firefox ( <a href="https://www.lambdatest.com/test-on-firefox-browsers">https://www.lambdatest.com/test-on-firefox-browsers</a> )
Chrome ( <a href="https://www.lambdatest.com/test-on-chrome-browsers">https://www.lambdatest.com/test-on-chrome-browsers</a> )	Safari ( <a href="https://www.lambdatest.com/test-on-safari-browsers">https://www.lambdatest.com/test-on-safari-browsers</a> )	Microsoft Edge ( <a href="https://www.lambdatest.com/test-on-edge-browsers">https://www.lambdatest.com/test-on-edge-browsers</a> )
Opera ( <a href="https://www.lambdatest.com/test-on-opera-browsers">https://www.lambdatest.com/test-on-opera-browsers</a> )	Yandex ( <a href="https://www.lambdatest.com/test-on-yandex-browsers">https://www.lambdatest.com/test-on-yandex-browsers</a> )	Mac OS ( <a href="https://www.lambdatest.com/test-on-macos-browsers">https://www.lambdatest.com/test-on-macos-browsers</a> )
Mobile Devices ( <a href="https://www.lambdatest.com/test-on-mobile-devices">https://www.lambdatest.com/test-on-mobile-devices</a> )	iOS Simulator ( <a href="https://www.lambdatest.com/test-on-ios-devices">https://www.lambdatest.com/test-on-ios-devices</a> )	Android Emulator ( <a href="https://www.lambdatest.com/android-emulator-online">https://www.lambdatest.com/android-emulator-online</a> )
Browser Emulator ( <a href="https://www.lambdatest.com/browser-emulator-online">https://www.lambdatest.com/browser-emulator-online</a> )		





## Resources

### Conferences

(<https://www.lambdatest.com/testconf-2022>)

Blogs  
(<https://www.lambdatest.com/blog/>)

Community  
(<https://community.lambdatest.com>)

### Product Updates

### Certifications

(<https://www.lambdatest.com/certifications/>)

Newsletter  
(<https://www.lambdatest.com/newsletter/>)

### FAQ

### Webinars

(<https://www.lambdatest.com/webinars/>)

### Videos

(<https://www.lambdatest.com/videos/>)

(<https://www.lambdatest.com/support-faq/>)

### Web Technologies

(<https://www.lambdatest.com/web-technologies>)

### Automation Testing Advisor

(<https://www.lambdatest.com/automation-testing-advisor>)

### Free Online Tools

(<https://www.lambdatest.com/free-online-tools>)

### Sitemap

(<https://www.lambdatest.com/sitemap.xml>)

### Status

(<https://status.lambdatest.io>)

## Learning Hub

### Cypress Tutorial

Selenium Tutorial  
(<https://www.lambdatest.com/selenium-tutorial>)

(<https://www.lambdatest.com/learn-playwright>)

Playwright Tutorial  
(<https://www.lambdatest.com/playwright>)

### Mocha Tutorial

Puppeteer Tutorial  
(<https://www.lambdatest.com/puppeteer>)

### Jest Tutorial

(<https://www.lambdatest.com/jest>)

### TestCafe Tutorial

(<https://www.lambdatest.com/testcafe>)

### Appium Tutorial

(<https://www.lambdatest.com/appium>)

### Espresso Tutorial

(<https://www.lambdatest.com/espresso>)

### More Learning Hubs

XCUIest Tutorial  
(<https://www.lambdatest.com/xcuitest>)

(<https://www.lambdatest.com/learning-hubs>)

## Company

### About Us

(<https://www.lambdatest.com/about>)

### Customers

(<https://www.lambdatest.com/customers/>)

### Press

(<https://www.lambdatest.com/press/>)

### Reviews

(<https://www.lambdatest.com/reviews/>)

### Community & Support

(<https://www.lambdatest.com/community>)

### Open Source

Partners  
(<https://www.lambdatest.com/partners>)

### Write for Us

(<https://www.lambdatest.com/lambdatest-write-for-us>)



([/#facebook](#))



([/#twitter](#))



([/#linkedin](#))





## What's New

Changelog  
[\(https://changelog.lambdatest.com\)](#)

October '22 Updates <a href="#">(https://www.lambdatest.com/blog/october-2022-updates/)</a>	Google Cloud - Issue 115 <a href="#">(https://www.lambdatest.com/newsletter/october-2022-updates/)</a>	Voices of Community: Move Forward with an Effective Test Automation Strategy [Webinar] <a href="#">(https://www.lambdatest.com/webinar/effective-test-automation-strategy)</a>
Whatfix [Case Study] <a href="#">(https://www.lambdatest.com/customers/whatfix)</a>	Software Testing [Glossary] <a href="#">(https://www.lambdatest.com/learning-hub/glossary)</a>	A/B Testing [Tutorial] <a href="#">(https://www.lambdatest.com/learning-hub/ab-testing)</a>
Test Design Specification [Tutorial] <a href="#">(https://www.lambdatest.com/learning-hub/test-design-specification)</a>	Agile in Distributed Development [Thought Leadership] <a href="#">(https://www.lambdatest.com/blog/agile-in-distributed-development/)</a>	How To Automate Toggle Buttons In Selenium Java [Blog] <a href="#">(https://www.lambdatest.com/blog/how-to-automate-toggle-buttons-in-selenium-java/)</a>
Playwright 102 [Certification] <a href="#">(https://www.lambdatest.com/certifications/playwright-102)</a>		

© 2022 LambdaTest. All rights reserved

Cross Browser Testing Cloud Built With ❤️ For Testers

