Most Exhaustive XPath Locators Cheat Sheet

Posted by Vinayak Sharma (https://www.lambdatest.com/blog/author/vinayaksharma/) | July 19, 2021

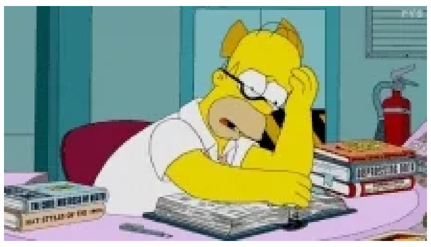
(https://www.lambdatest.com/blog/author/vinayaksharma/)

CheatSheets • Selenium Testing •

46386 Views | 7 Min Read

The Selenium framework lets you interact with the WebElements in the DOM. For realizing the interaction(s), it is important to choose the appropriate locator from the available Selenium web locators. As per my opinion, Selenium web locators can be considered as the backbone of any web automation script.

There are many options for web locators in Selenium – ID, Name, XPath, LinkText, Partial LinkText, Tag Name, and more. XPath is one of the widely preferred web locators, as it easily traverses through the DOM elements and attributes to identify an object.



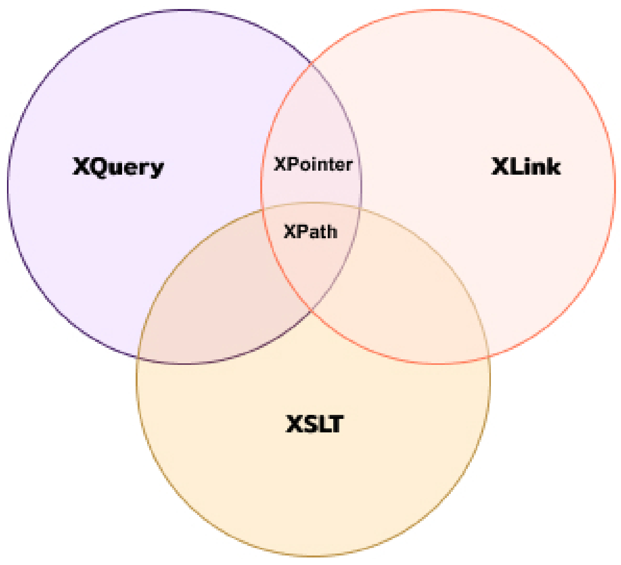
Source (https://giphy.com/gifs/lPbS5R4fSUI5S)

If you plan to use an XPath web locator to shape your Selenium automation testing (https://www.lambdatest.com/selenium-automation) scripts, you might not have to look further than this comprehensive XPath cheat sheet. In addition, you can keep this XPath cheat sheet handy in scenarios where you want to have a quick look at the XPath syntax or other aspects related to this web locator.

In this Selenium locators (https://www.lambdatest.com/learning-hub/selenium-locators) cheat sheet, we have demonstrated the XPath locator in the Java language. So, let's get started with this exhaustive XPath cheat sheet that will come in super handy in Selenium automation testing.

What is XPath Web Locator in Selenium

XPath, which stands for XML Path Language, uses 'path-like' syntax to identify and navigate nodes in an XML or HTML document. It follows the XSLT(eXtensible Stylesheet Language Transformations) standard, which is predominantly used to navigate WebElements and attributes.



In order to navigate through the HTML document, the DOM (HTML tags, JavaScript, etc.) is used as a map where all the WebElements exist along with the WebElement that we intend to locate using the appropriate web locator.

You can check out our blog on locators in Selenium (https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples) WebDriver to check other web locators that we have not covered in this XPath cheat sheet.

Different types of XPath in Selenium

You can locate the desired WebElement in the DOM, either using the absolute path or using a relative path with respect to another element. In this part of the XPath cheat sheet, we look at the different ways in which you can use an XPath locator for locating a WebElement.

Here are the two main types of XPath in Selenium:

Absolute XPath

As the name indicates, an absolute XPath contains the complete path from the root element to the desired element. The downside of using absolute XPath is that any changes made in the path of the element, HTML path, etc., results in the failure of the XPath.

Absolute Xpath begins with the single forward-slash(/), which means selecting the element from the document's root node.

Syntax – Absolute Xpath

```
/html/body/div[x]/div[y]/
```

Relative XPath

In contrast to absolute XPath, a relative XPath starts by referencing the element we want to locate relative to a particular location. This means that the element is positioned relative to its normal position.

Since the path of XPath is relative, it starts with a double forward-slash (//). As mentioned earlier in this Selenium locators cheat sheet, relative XPath is usually preferred for Selenium automation testing since it is not a complete path from the root element.

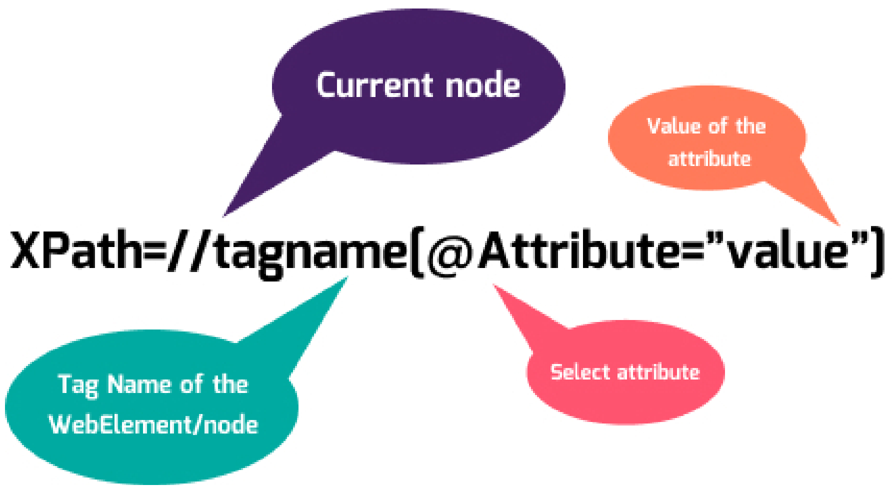
Hence, any changes in the page design or the DOM hierarchy will have a minimal (to no) impact on the existing XPath selector.

Syntax – Relative Xpath

```
//tagname[@attribute='value']
```

The XPath syntax

XPath uses path expressions to select nodes (or node-sets) in an HTML document. As shown below, the element or tag is selected by following a path or series of steps:


Current node
Value of the attribute
XPath=//tagname[@Attribute="value"]
Tag Name of the WebElement/node
Select attribute

Here are some of the useful path expressions that you can keep in handy as a part of the XPath cheat sheet:

Expression / XPath	**Description**
node	Selects all elements with the name
/	Selects from the root node
//	Selects elements in the document from the current element that matches the selection no matter where they are.
	Selects the current node
	Selects the parent of the current node element.
@	Selects Attributes

XPath Expressions in Selenium

XPath expression is a pattern that is used for selecting a set of nodes. These patterns are used by XSLT for performing transformations. XPointer also uses these patterns for addressing purposes.

XPath uses a path expression to select the node or a list of nodes from an XML or HTML document.

Syntax – XPath Expression

```
//tagname[@attribute='value']
```

Here is a brief description of the Steps and Axis:

//	ul	/	a[@id='link']
Axis	Step	Axis	Step

- ### Prefixes

Prefixes in XPath are used at the beginning of the XPath expression.

Expression	**Example**	**Description**
//	//p[@class='footer']	Select paragraph tag from anywhere in the DOM

/	/a	Find all anchor (a) tags from the given node
/	/html/body/div	Find all div, starting from root element

Here are the examples of different kinds of Steps in XPath:

XPath	Description
//div	Select all div tags
//[@id='btn']	Select all elements with ID 'btn'
//div[@name='post']	Selecting all div elements with the name 'post'
//a/text()	Gives the text of all the anchor tag(s)
//a/@href	Gives href value of anchor tag

- ## Selecting Nodes

In this part of the XPath cheat sheet, we look at the different ways to select a node (or WebElement) in the document using XPath.

XPath	Description
div	Selects all div elements
/div	Selects div element from the root element
div/tr	Select all tr elements which are children of div
//tr	Select all tr elements anywhere in the document
div//tr	Selecting all tr elements which are children of div anywhere in the document
//@class	Select all class attributes

Read – Using XPath in Selenium for Selenium automation testing (https://www.lambdatest.com/blog/complete-guide-for-using-xpath-in-selenium-with-examples/)

- ## Predicates

Predicates in XPath are used to find a specific node that contains a designated value. It uses syntax enclosed in square brackets.

The table shown below in this Selenium locators cheat sheet shows some examples of predicates, along with different ways of selecting WebElements using those predicates.

XPath	Description
/div/a[1]	Select first anchor element within the div element.
/div/a[last()]	Select last anchor element within the div element.

/div/a[last()-1]	Select second-last anchor element within a div element.
/div/a[position()<3]	Select the first two anchor elements within a div element.
//a[@class]	Select all anchor elements with class attribute.
//a[@class='btn']	Select all anchor elements with class attribute and value 'btn'.
/div/h1[1]/a	Select all anchor elements within h1, which are children of the div element.
/div/h1/a[1]	Select all anchor elements which are children of h1 within a div element.

- ## Chaining order

Chaining order in XPath indicates the importance of order in an Xpath expression. The meaning of XPath changes with the change in order. For example a[1][@href='/'] and a[@href='/'][1] are different.

- ## Indexing

Indexing in XPath can be done using [ ] with a number that specifies the node we intend to select. It can also be done using functions like last() or position() that specify the index of the elements.

//a[1]	Select the first anchor tag
//a[last()]	Select the last anchor tag
//ul/li[2]	Selecting second li tag which is a child of ul tag
//ul/li[position()=2]	Select second li tag which is a child of ul tag
//ul/li[position()>1]	Select li tag which is not a first child of ul tag

- ## Nesting predicates

XPath can also be nested to find the targeted node (or WebElement).

Example – Nested Predicates

```
//section[.//a[@id='btn']]
```

This example will return

if it has an anchor tag descendant with an ID value of 'btn.'

- ## Selecting Unknown Nodes

Wildcards can also be used with an XPath locator to find unknown HTML document elements.

*	It matches any HTML element
@*	It matches any attribute of an element
node()	It matches any kind of element

Here are some of the examples that should be a part of the XPath cheat sheet:

/div/*	Select all children of a div element
//*	Select all elements in the HTML document
//a[@*]	Select all anchor elements with any attribute

- ### Selecting Several Paths

Using the '|' operator in the XPath expression, you can select several paths. Shown below in this Selenium locators cheat sheet are some examples that demonstrate the usage of the '|' operator.

//div | //a	Select all div and anchor elements in the HTML document.
//div/h1 | //div/a	Select all h1 and anchor elements within a div element

XPath Selectors in Selenium

Selectors are used to 'select' certain parts of the HTML document specified by the element's XPath. It's the same as Axes in XPath.

- ### Descendant selectors

The descendant selectors indicate all the children of the current node, all children of children, etc. Attribute and namespace nodes are not included in descendant selectors. The parent of an attribute node is its element node, and on the other hand, attribute nodes are not the offspring of their parents.

div	//div	Select all div elements
div h1	//div//h1	Select all h1 within a div element
ul > li	//ul/li	Select all li elements which are children of ul
div > p > a	/div/p/a	Select all anchor tags within paragraph tag of div element
div > *	//div/*	Select all elements in div tag
:root	/	Select root element of the DOM
:root > body	/body	Select body tag

- ### Attribute selectors

The XPath attribute selector matches elements based on the presence or value of a given attribute.

Element	**Xpath**	**Description**
#id	//*[@id="id"]	Select all elements with matching ID attribute .
.class	//*[@class="class"]	Select all elements with matching class attribute.
a[rel]	//a[@rel]	Select all anchor tag(s) with rel attribute.

a[href^='/']	//a[starts-with(@href, '/')]	Select all anchor tag(s) with href starting with '/'.
a[href$='txt']	//a[ends-with(@href, '.txt')]	Select all anchor tag(s) with href ending with '.txt'.
a[rel~='details']	//a[contains(@rel, 'details')]	Select all anchor tag(s) with rel value 'details'.
input[type="password"]	//input[@type="password"]	Select all input tag(s) of type password.
a#btn[for="lambdatest"]	//a[@id="btn"][@for="Lambdatest"]	Select all anchor tag(s) with 'btn' ID linked with Lambdatest.

- ## Order Selectors

Order selector in XPath is used for getting the elements from a list.

Element	XPath	Description
ul > li:first-of-type	//ul/li[1]	Select first li tag which is a child of ul
ul > li:nth-of-type(2)	//ul/li[2]	Select second li tag which is a child of ul
li#id:first-of-type	//li[1][@id="id"]	Select first li with id value "id"
ul > li:last-of-type	//ul/li[last()]	Select last li which is child of ul
a:first-child	//*[1][name()="a"]	Select first child of anchor tag
a:last-child	//*[last()][name()="a"]	Select last child of anchor tag

- ## Siblings

Sibling in Selenium WebDriver is used for fetching a WebElement that is a sibling to a parent element. In this part of the XPath cheat sheet, we look at how to fetch elements using siblings in Selenium WebDriver.

Element	XPath	Description
h1 ~ ul	//h1/following-sibling::ul	Select all ul tags which are following sibling of h1 tag
h1 ~ #id	//h1/following-sibling::[@id="id"]	Select all elements with ID value "id" that are siblings of h1
h1 + ul	//h1/following-sibling::ul[1]	Select first ul tags which are following sibling of h1

- ## Using different Operators

You can use different operators like NOT, text matches (i.e., string, substring, etc.), union, and more to locate elements using operators and XPath in Selenium.

Operators	XPath	Description
Not Operators	//p[not(@id)]	Select all paragraph tag(s) with attributes not matching id

Text match	//button[text()="Submit"]	Select button with text input "Submit"
Text match (substring)	//button[contains(text(),"pass")]	Select button that has string 'pass' present in it
Arithmetic	//product[@price > 3]	Select price with value > 3
Has children	//ul[*]	Select ul with any number (or type) if it has children
Has children (specific)	//ul[li]	Select ul with any number and li tag present as a child
logic	//a[@name or @href]	Select all anchor tag(s) with the name and href attribute
Union (joins results)	//a | //div	Union of a and div tags

Axes in XPath

Axes in XPath represent a relationship from the current element to its relative element on the tree using either the absolute XPath or relative XPath. Its relationship to elements is identified as a parent, child, sibling, etc.

They are named Axes because they refer to the axis on which elements lie relative to a particular WebElement. Shown below is the syntax of Axes in XPath:

```
axisname::element[attribute]
```

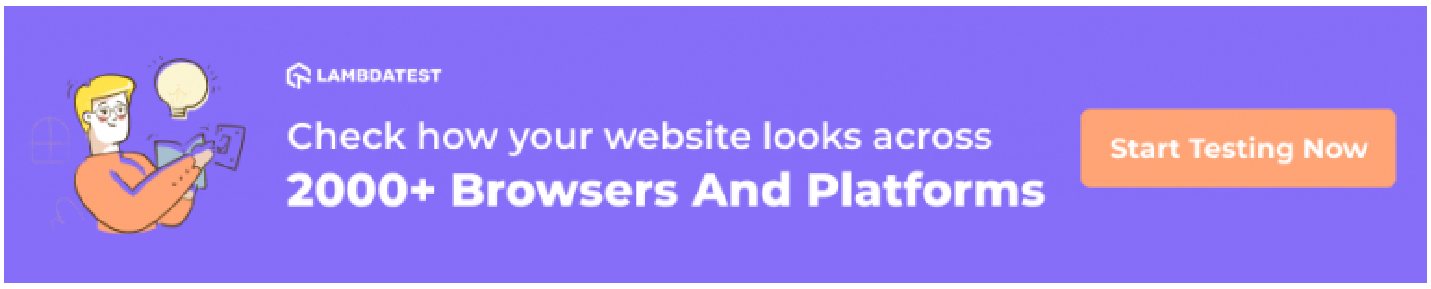
This part of the Selenium locators cheat sheet shows the axis names and their relationship with the elements.

Element	**XPath**
self	For referring current elements
parent	For referring parent of the current node
preceding	Selects all elements that appear before the current element in the document
preceding-sibling	Selects all siblings before the current elements
namespace	Selects all namespace elements of the current element
ancestor	Selects all ancestors with a relationship like a parent, grandparent, etc. relative to the current element
ancestor-or-self	Selects the current element as well as all the ancestors of that element
attribute	Selects all attributes of the current node
child	Selects all children of the current element
descendant	Selects all children, grandchildren, etc. of the current element
descendant-or-self	Selects all descendants of the current elements and the current element itself

following	Selecting everything in the document after the closing tag
following-sibling	Selecting all siblings after the current element

Here are some of the ways in which siblings in XPath can be used to find WebElements for Selenium automation testing.

child::div	Selects all div elements that are children of the current element
attribute::lang	Selects lang attribute of the current
	element
child::text()	Finds all text elements that are children of the current element
child::node()	Selects all children of the current tag
child::*	Selects all elements that are children of the current element
descendant::div	Selects all divs that are descendants of the current element


LAMBDATEST
Check how your website looks across
2000+ Browsers And Platforms
Start Testing Now

(https://accounts.lambdatest.com/register/?utm_source=BlogCTA&utm_medium=blog)

Operators in XPath

An XPath expression can return either a node-set(div,a,li), a string, a Boolean (true or false), or a number. As a result, XPath provides many operators to manipulate their return values so that you get the desired value.

In this part of XPath cheat sheet, we look at the list of operators that can be used with XPath expressions:

Operator	**Description**	**Example**
+	Addition	5+6
–	Subtraction	5 – 1
*	Multiplication	2*6
div	Division	14 / 7
=	Equal	5 = 6
!=	Not equal	//div != //a
<	Less than	//div < //a

<=	Less than or equal to	//div <= //a
>	Greater than	//div > //a
|	Computes two elements set	//div | //a
or	or	5 or 6
and	and	/div and /a
mod	mod	5 mod 2

Functions in XPath

XPath provides a number of core libraries that bring in functionalities to deal with node sets, strings, booleans, and numbers.

Syntax Demonstration

```
button[text()="Submit"]
//button/text()
//ol/li[position()=2]
```

Node functions

There are seven kinds of nodes in XPath – element, attributes, namespaces, comments, document nodes, text, and processing instructions. First, let's look at the permitted operations on nodes in further detail in this section of the XPath cheat sheet.

- name()

This function provides the name of the element

Example:

```
//[starts-with(name(), 'h')]
```

- text()

This function Selects a text node.

Example:

```
//button[text()="Submit"]
//button/text()
```

- position()

This function provides the position of the element.

```
/div/a[position()<3]
```

- last()

This function selects the last node relative to the current element.

Example:

```
/div/a[last()]
```

- comment()
- This function selects the elements, which are comments.

Example

```
/div/comment()
```

String functions

XPath string functions let you access elements using string type functions (e.g., contains, starts-with, etc.)

- contains()

  This function is used to see if a certain element contains a particular string

  Example:

```
font[contains(@class,"nav")]
```

- starts-with()

  This function is used to check if our element starts with a certain text.

  Example:

```
font[starts-with(@class,"nav")]
```

- ends-with()

  Same as starts-with(), look (or check) at the end of the string.

  Example:

```
font[ends-with(@class,"nav")]
```

- substring-before()

  This function returns a string that is part of the input string before a given substring.

  Example:

```
substring-before("01/02", "/")
```

  Output: 01

- substring-after()

  Same as substring-before(), returns a string that is part of the input string after a given substring.

  Example:

```
substring-after("01/02", "/")
```

  Output: 02

Math functions

You can use mathematics-type functions like floor, ceil, sum, etc., to find elements using the XPath selector.

Here are the math functions that we have covered in our XPath cheat sheet:

- ceiling(number)

  This function returns the smallest integer value greater than or equal to the decimal number.

- floor(number)

  This function returns the largest integer value less than or equal to the decimal number.

- round(decimal)

  This function returns a number that is the nearest integer to the given number.

- sum(elements)

  This function returns a number that is the sum of the numeric values of each element in a given element-set.

Boolean functions

Boolean operators can also be used to find elements using the XPath selector.

- not()

  This function is used to check whether certain conditions are met or not.

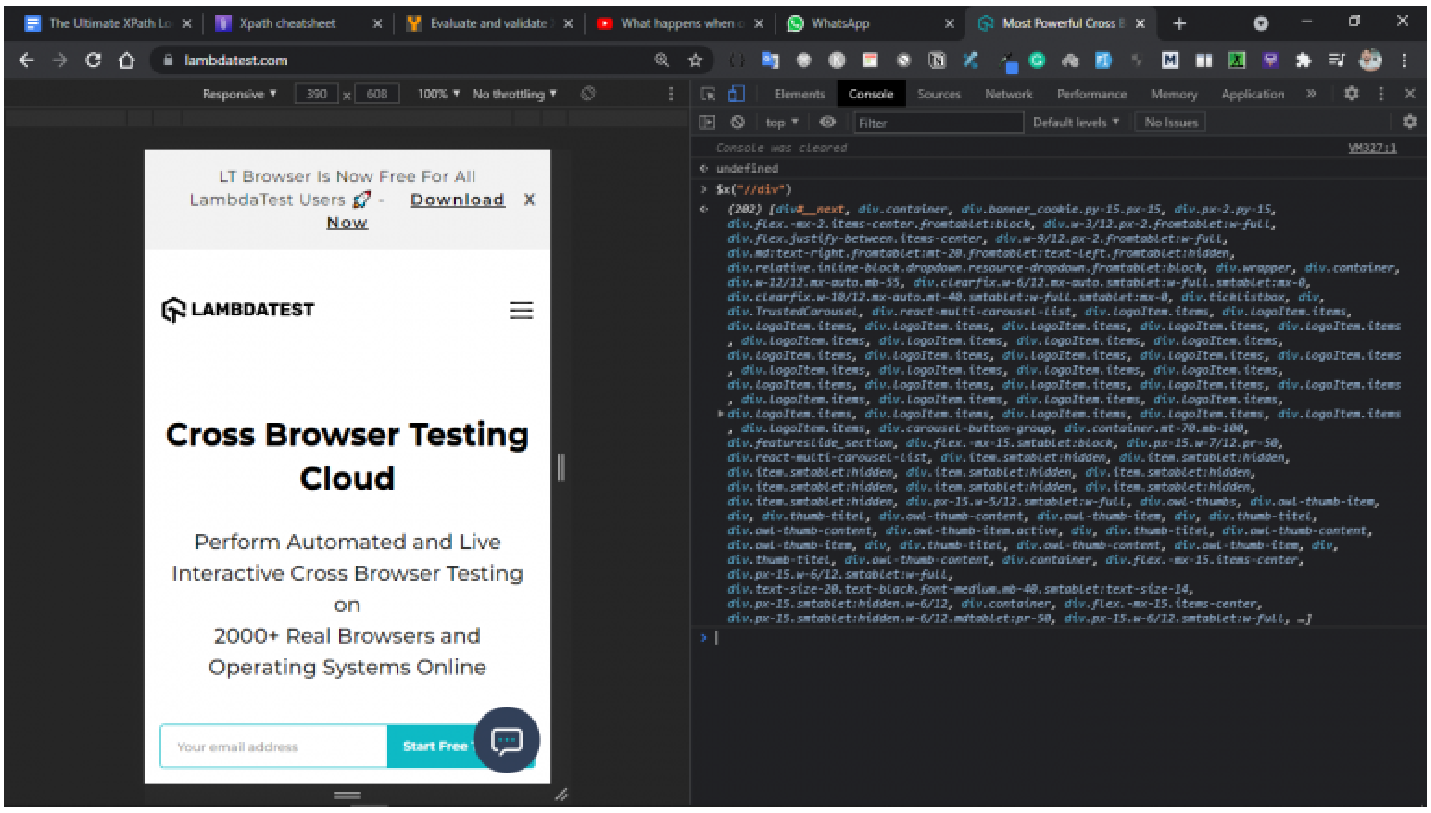
  Example:

```
button[not(starts-with(text(),"Submit"))]
```

Browser console to locate XPath

Most modern web browsers (e.g., Chrome, Firefox, etc.) have a built-in debugging tool, which includes a feature that allows us to evaluate or validate XPath/CSS selectors. These tokens execute in the console panel, thereby providing you an option to validate the XPath selector.

Here is a screenshot of the Debug Console in Chrome:



```
Console was cleared                                                    VM327:1
← undefined
> $x("//div")
← (282) [div#__next, div.container, div.banner_cookie.py-15.px-15, div.px-2.py-15,
   div.flex.-mx-2.items-center.fromtablet:block, div.w-3/12.px-2.fromtablet:w-full,
   div.flex.justify-between.items-center, div.w-9/12.px-2.fromtablet:w-full,
   div.md:text-right.fromtablet:mt-20.fromtablet:text-left.fromtablet:hidden,
   div.relative.inline-block.dropdown.resource-dropdown.fromtablet:block, div.wrapper, div.container,
   div.w-12/12.mx-auto.mb-55, div.clearfix.w-6/12.mx-auto.smtablet:w-full.smtablet:mx-0,
   div.clearfix.w-10/12.mx-auto.mt-40.smtablet:w-full.smtablet:mx-0, div.ticklistbox, div,
   div.TrustedCarousel, div.react-multi-carousel-list, div.logoItem.items, div.logoItem.items,
   div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items
   , div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items,
   div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items
   , div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items,
   div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items
   , div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items,
  ▸div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items, div.logoItem.items
   , div.logoItem.items, div.carousel-button-group, div.container.mt-70.mb-100,
   div.featureslide_section, div.flex.-mx-15.smtablet:block, div.px-15.w-7/12.pr-50,
   div.react-multi-carousel-list, div.item.smtablet:hidden, div.item.smtablet:hidden,
   div.item.smtablet:hidden, div.item.smtablet:hidden, div.item.smtablet:hidden,
   div.item.smtablet:hidden, div.item.smtablet:hidden, div.item.smtablet:hidden,
   div.item.smtablet:hidden, div.px-15.w-5/12.smtablet:w-full, div.owl-thumbs, div.owl-thumb-item,
   div, div.thumb-titel, div.owl-thumb-content, div.owl-thumb-item, div, div.thumb-titel,
   div.owl-thumb-content, div.owl-thumb-item.active, div, div.thumb-titel, div.owl-thumb-content,
   div.owl-thumb-item, div, div.thumb-titel, div.owl-thumb-content, div.owl-thumb-item, div,
   div.thumb-titel, div.owl-thumb-content, div.container, div.flex.-mx-15.items-center,
   div.px-15.w-6/12.smtablet:w-full,
   div.text-size-20.text-black.font-medium.mb-40.smtablet:text-size-14,
   div.px-15.smtablet:hidden.w-6/12, div.container, div.flex.-mx-15.items-center,
   div.px-15.smtablet:hidden.w-6/12.mdtablet:pr-50, div.px-15.w-6/12.smtablet:w-full, …]
> |
```

In this example, we have used $x("//div"). With this, it selects all div elements from the DOM. An exception is thrown in case there is no matching node or elements in the DOM.

Obtain XPath using JQuery

JQuery supports all basic kinds of XPath expressions; major ones are listed below in this part of the XPath cheat sheet:

JQuery XPath	XPath	Description
$('ul > li').parent()	//ul/li/..	Selects all ul elements which are parent of li tag
$('li').closest('section')	//li/ancestor-or-self::section	Selects all anchor tags with href attribute
$('p').text()	//p/text()	Selects text within paragraph tags

Bonus Examples

XPath	Description
//*	Select all elements on the page

count(//*)	Count all the elements present on the page
(//a)[1]/text()	Returns text of the first anchor heading
//li[span]	Find li with span tag inside it
//section[a[@id='btn']]	Finds a section that directly contains a#btn
//section[//a[@id='btn']]	Finds a section that contains a#btn.
//item[@price > 2*@discount]	Finds item and check its attributes
//*[@id="btn"]/ul/li[6]/a	Select all anchor elements within 6th li of ul element in elements with id="btn"

Conclusion

XPath is one of the widely used web locators in Selenium. However, coming with an efficient XPath is an art that largely depends on the document's structure and the current position of the WebElement.

One of the most important things is finding the nearest unique WebElement relative to the target element and using the best technique to locate the WebElement. We covered the major aspects of the XPath locator in this exhaustive XPath cheat sheet.

Source (https://giphy.com/gifs/runner-files-gomi-1bHdnX1QMeQTe)

This Selenium locators cheat sheet can be used as a handy resource when you are performing Selenium automation testing.

When it comes to Selenium automation testing, you can leverage the capabilities offered by cloud-based Selenium Grid like LambdaTest that lets you run tests on different browsers, platforms, and device combinations.

Do share your views on how you would use this XPath cheat sheet for Selenium automation testing. Leave your feedback in the comments section…

Happy Testing!

(https://www.lambdatest.com/blog/author/vinayaksharma/)

Vinayak Sharma (https://www.lambdatest.com/blog/author/vinayaksharma/)

Full stack python developer and a tech enthusiast with strong communication and interpersonal skills. Highly adaptable to new environments, challenges, and increasing levels of responsibilities.

See author's profile (https://www.lambdatest.com/blog/author/vinayaksharma/)

Related Articles

(https://www.lambdatest.com/blog/selenium-python-cheat-sheet/)
The Ultimate Selenium Python Cheat Sheet for Test Automation (https://www.lambdatest.com/blog/selenium-python-cheat-sheet/)

2289 Views | 16 Min Min Read

(https://www.lambdatest.com/blog/building-an-automated-testing-pipeline-with-gocd/)
Building An Automated Testing Pipeline with GoCD [Tutorial] (https://www.lambdatest.com/blog/building-an-automated-testing-pipeline-with-gocd/)

144645 Views | 13 Min Min Read

(https://www.lambdatest.com/blog/css-selectors-cheat-sheet/)
The Ultimate CSS Selectors Cheat Sheet You Must Know (https://www.lambdatest.com/blog/css-selectors-cheat-sheet/)

42031 Views | 25 Min Min Read

(https://www.lambdatest.com/blog/speed-up-selenium-test-cases-execution/)

How To Speed Up Selenium Test Cases Execution? (https://www.lambdatest.com/blog/speed-up-selenium-test-cases-execution/)

94556 Views | 19 Min Min Read

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

Help & Support

- +1-(866)-430-7087 (tel:+1-(866)-430-7087)
- support@LambdaTest.com (mailto:support@LambdaTest.com)

Products & Features

Selenium (https://www.lambdatest.com/selenium-automation)	Cypress (https://www.lambdatest.com/cypress-testing)	Browser Testing (https://www.lambdatest.com/feature)
LT Browser (https://www.lambdatest.com/lt-browser)	Local Page Testing (https://www.lambdatest.com/local-page-testing)	Automated Screenshots (https://www.lambdatest.com/automated-screenshot)
Geo-Location Testing (https://www.lambdatest.com/geolocation-testing)	Responsive Testing (https://www.lambdatest.com/responsive-test-online)	Smart Testing (https://www.lambdatest.com/smart-visual-ui-testing)
Mobile App Testing (https://www.lambdatest.com/mobile-app-testing)	Integrations (https://www.lambdatest.com/integrations)	Status (https://status.lambdatest.io)

Browsers

List of Browsers (https://www.lambdatest.com/list-of-browsers)
Test on IE (https://www.lambdatest.com/test-on-internet-explorer-browsers)
Test on Firefox (https://www.lambdatest.com/test-on-firefox-browsers)
Test on Chrome (https://www.lambdatest.com/test-on-chrome-browsers)
Test on Safari (https://www.lambdatest.com/test-on-safari-browsers)
Test on Microsoft Edge (https://www.lambdatest.com/test-on-edge-browsers)
Test on Opera (https://www.lambdatest.com/test-on-opera-browsers)
Test on Yandex (https://www.lambdatest.com/test-on-yandex-browsers)
Test on Mac (https://www.lambdatest.com/test-on-macos-browsers)
Test on Mobile (https://www.lambdatest.com/test-on-mobile-devices)
Test on iOS Simulator (https://www.lambdatest.com/test-on-ios-devices)
Test on Android Emulator (https://www.lambdatest.com/android-emulator-online)
Test On Browser Emulator (https://www.lambdatest.com/browser-emulator-online)

Resources

Blogs (https://www.lambdatest.com/blog/)
Press (https://www.lambdatest.com/press/)
Community (https://community.lambdatest.com)
Certifications (https://www.lambdatest.com/certifications/)
Learning Hub (https://www.lambdatest.com/learning-hub/)
Product Updates (https://www.lambdatest.com/blog/category/lambdatest-updates/)
Newsletter (https://www.lambdatest.com/newsletter/)
Webinars (https://www.lambdatest.com/webinar/)
Videos (https://www.lambdatest.com/video/)
FAQ (https://www.lambdatest.com/support-faq)
Sitemap (https://www.lambdatest.com/sitemap.xml)

Company

About Us (https://www.lambdatest.com/about)
Customers (https://www.lambdatest.com/customers/)
Reviews (https://www.lambdatest.com/reviews)
Partners (https://www.lambdatest.com/partners/)
Reseller (https://www.lambdatest.com/reseller)
Become an Affiliate (https://www.lambdatest.com/affiliate-program-partnership)
Terms of service (https://www.lambdatest.com/legal/terms-of-service)
Privacy Policy (https://www.lambdatest.com/legal/privacy/)
Security (https://www.lambdatest.com/security)
Careers (https://www.lambdatest.com/career)
Team (https://www.lambdatest.com/team)
Contact Us (https://www.lambdatest.com/contact-us)

What's New

Changelog (https://changelog.lambdatest.com)

Introduction to Selenium Basics (https://www.lambdatest.com/selenium)
Accelerate Delivery With Mobile App Testing Cloud (https://www.lambdatest.com/mobile-app-testing)
August '21 Updates (https://www.lambdatest.com/blog/august-2021-product-updates/)
Building A CI/CD Pipeline With Travis CI, Docker, And LambdaTest (https://www.lambdatest.com/blog/ci-cd-pipeline-with-travis-ci-docker-and-lambdatest/)
How To Write Tests With Cypress UI Commands
Selenium Automation Testing On The Cloud