**XPath is XML path** in Selenium is a way to navigate the structure of a webpage’s HTML, as well as a document's XML structure, to find elements that are not found by locators such as ID, class, or name.

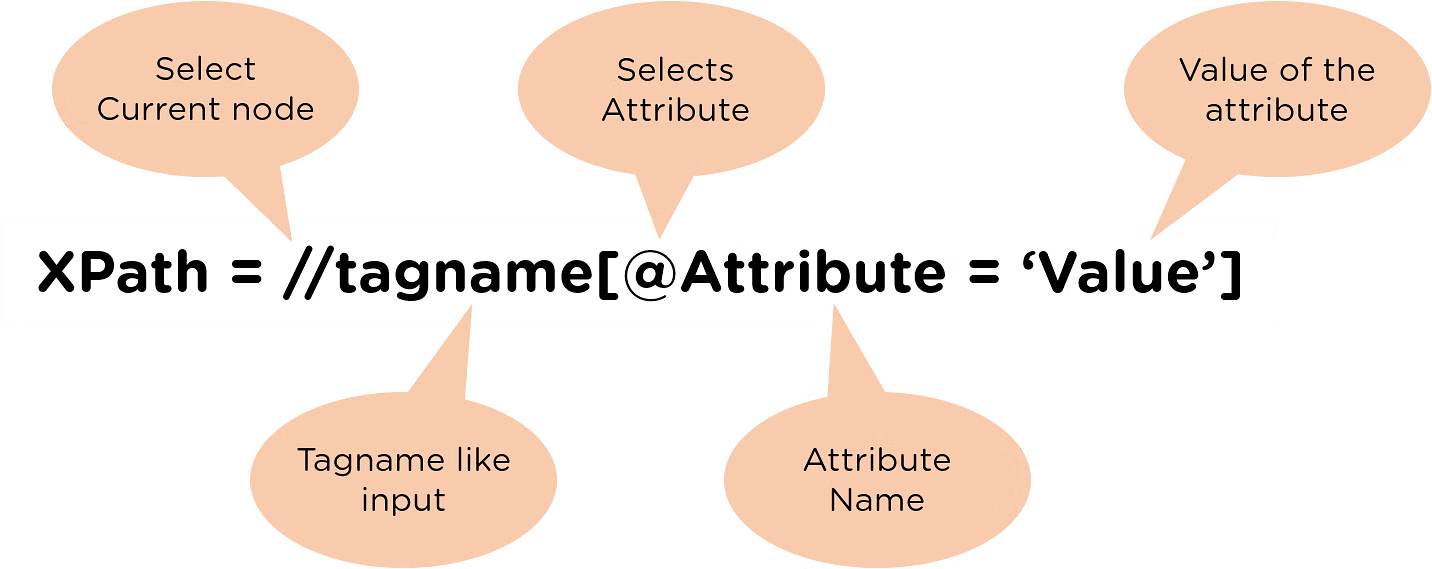
XPath in Selenium enables teams to search for a specific element within a web page in a dynamic fashion, which allows for increased flexibility.

Absolute XPath

This is the direct way of finding an element, and it starts with a single forward-slash (/), which states that the element can be selected from the root node. It provides the complete path from the root element to the desired element, and includes the names of all the parent elements leading to the target element. Absolute XPath expressions are generally longer and more prone to breaking if the structure of the document changes.

Relative XPath

This starts from a specific element and navigates through the DOM hierarchy to locate the desired element. It identifies elements dynamically, starting with '//' followed by the tag name. Relative XPath allows for navigating from a particular point in the DOM, making it adaptable to page updates without needing the full path.



XPath in Selenium provides XPath functions to write efficient XPaths to discover elements uniquely. Let's have a look at all the different functions in Selenium:

### XPath Contains() function

The XPath Contains() is a function used to create an XPath expression. The function can navigate to the web element with the partial text present. It is used when the value of any attribute changes dynamically, for example, login information.

The contain feature has an ability to find the element with partial text as well as attributes with partial values as shown in below XPath example.

| //tag\_name[contains(@attribute,’value\_of\_attribute’)]  //tag\_name[contains(text(),’partial\_value\_of\_text’)] |
| --- |

The Contains() method accepts two parameters:

* The attribute of the tag must validate to locate the web element.
* The value of an attribute is a partial value that the attribute must contain.

### XPath Text() function

The XPath Text() is a function used to locate the element on a web page using the web element's text. The function proves its worth if the element contains a text, like a label, etc.

| //tag\_name[text()= ’Text of the element’] |
| --- |

The text() method here returns the text of the web element when identified by the tag\_name, and compared with the value provided on the right side.

### XPath Starts-with() function

The XPath Starts-with() function is used to find the element in which the attribute value starts with some specific character or a sequence of characters. The function plays a major role while working with the dynamic web pages.

| /tag\_name[starts-with(@attribute,’Part\_of\_Attribute\_value’)] |
| --- |

The Starts-with() accepts two parameters:

1. The attribute of the tag must validate to locate the web element.
2. The attribute value is the partial value of the attribute with which the attribute is expected to start.

## What Are XPath Axes?

XPath axes are mainly used to find dynamic web elements that can't be identified by ID, name, class name, link text, or CSS selector. For example, in Selenium WebDriver, XPath axes methods or functions can be used to locate web elements that don't have any **attributes on their own**. An axis represents a relationship to the context (current) node, and is used to locate nodes relative to that node on the tree.

All the DOM elements are in a hierarchical structure and can be either located using Absolute paths or Relative paths. For this, XPath provides specific attributes called "XPath Axis."

Here, an axis shows a relationship to the current node and helps locate the relative nodes concerning the tree's current node. So, the XPath Axis uses the relation between several nodes to find those nodes in the DOM structure.

| **AxisName** | **Result** |
| --- | --- |
| ancestor | Selects all ancestors (parent, grandparent, etc.) of the current node |
| ancestor-or-self | Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself |
| attribute | Selects all attributes of the current node |
| child | Selects all children of the current node |
| descendant | Selects all descendants (children, grandchildren, etc.) of the current node |
| descendant-or-self | Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself |
| following | Selects everything in the document after the closing tag of the current node |
| following-sibling | Selects all siblings after the current node |
| parent | Selects the parent of the current node |
| preceding | Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes |
| preceding-sibling | Selects all siblings before the current node |
| self | Selects the current node |



**/ & // -> In XPath, a single slash (/) is used for an absolute path, while a double slash (//) is used for a relative path.**

**A single slash selects only the immediate child elements, while a double slash selects all descendants of the current node, regardless of their level.**

**//tagName[@attribute=’value’]/child**

**//tagName[@attribute=’value’]//descendants**

Absolute xpath is never recommended

/html/body/div[1]/div[1]/div[1]/div/div/div[2]/div/div[1]/form/div[2]/button

/html/body/div[1]/div[1]/div/div/div/div[2]/div/div[1]/form/div[2]/button

Values of the attributes should be in single quotes

//button[@data-testid='royal\_login\_button']

Check appropriately all the white spaces and use the cases

//button[text()='Log in ']

//div[starts-with(@class,'header')]

//div[contains(@class,'header')]

When there is no attributes at all - then you can use the Axes relationships to locate the element[Ancestor to descendant]

**//form/div/button**

From child to ancestor

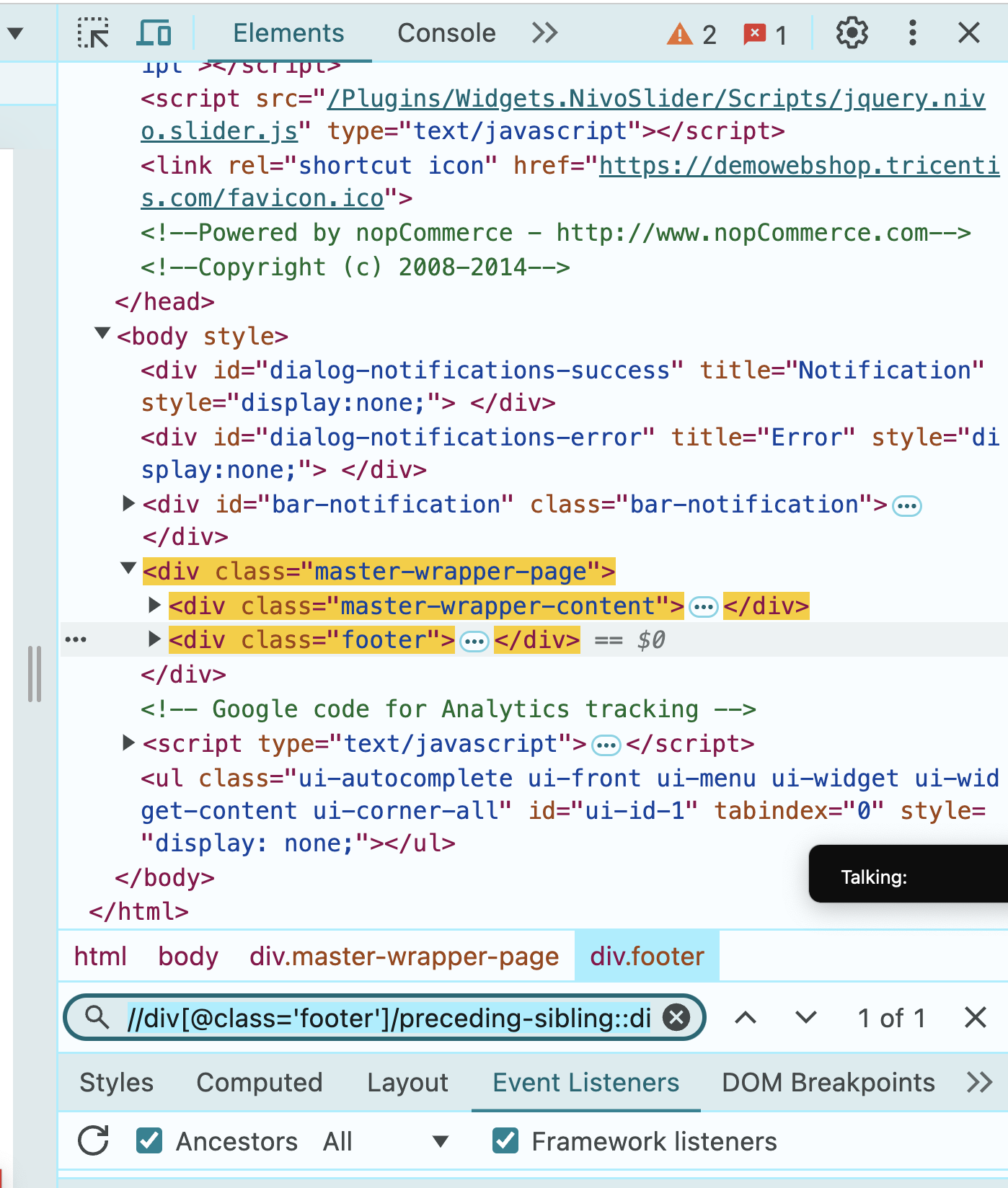
**//button/ancestor::form**

**To pick all the immediate children**

**//tag-Name/\***

**To pick all the descendants**

**//tag-Name//\***

****

**Class footer has a preceding sibling**

**//div[@class='footer']/preceding-sibling::div**

**Class master-wrapper-content has a following sibling**

**//div[@class='master-wrapper-content']/following-sibling::div**

**​​//button/parent::div/following-sibling::div[1]/a**

**//input[@id='email']/parent::div//following-sibling::div//input**