As we know, for automating any Web Applications, ***locating web elements*** is one of the first steps. We need some unique attributes such as ***id, name, className***, etc. to identify the *HTML* elements. Sometimes, either due to dynamic web elements or poor development practices, there are no unique attributes associated with the *Web Elements,* and there comes the role of ***XPath***. As we know, when a web page loads in a browser, it generates a ***DOM*** (*Document Object Model*) structure. ***XPath*** is the ***query language*** that queries objects in the ***DOM***. Subsequently, in this article, we will understand the intricacies and detailed usage of ***XPath in Selenium*** to locate the *Web Elements* on a *Web Page* by covering the details under the following topics:

What is XPath?
  Syntax of XPath.
  What is HTML/XML DOM?
  How to locate web elements using XPath in Selenium?
What are the different types of XPaths in Selenium?
  What is Absolute XPath in Selenium?
  And, what is Relative XPath in Selenium?

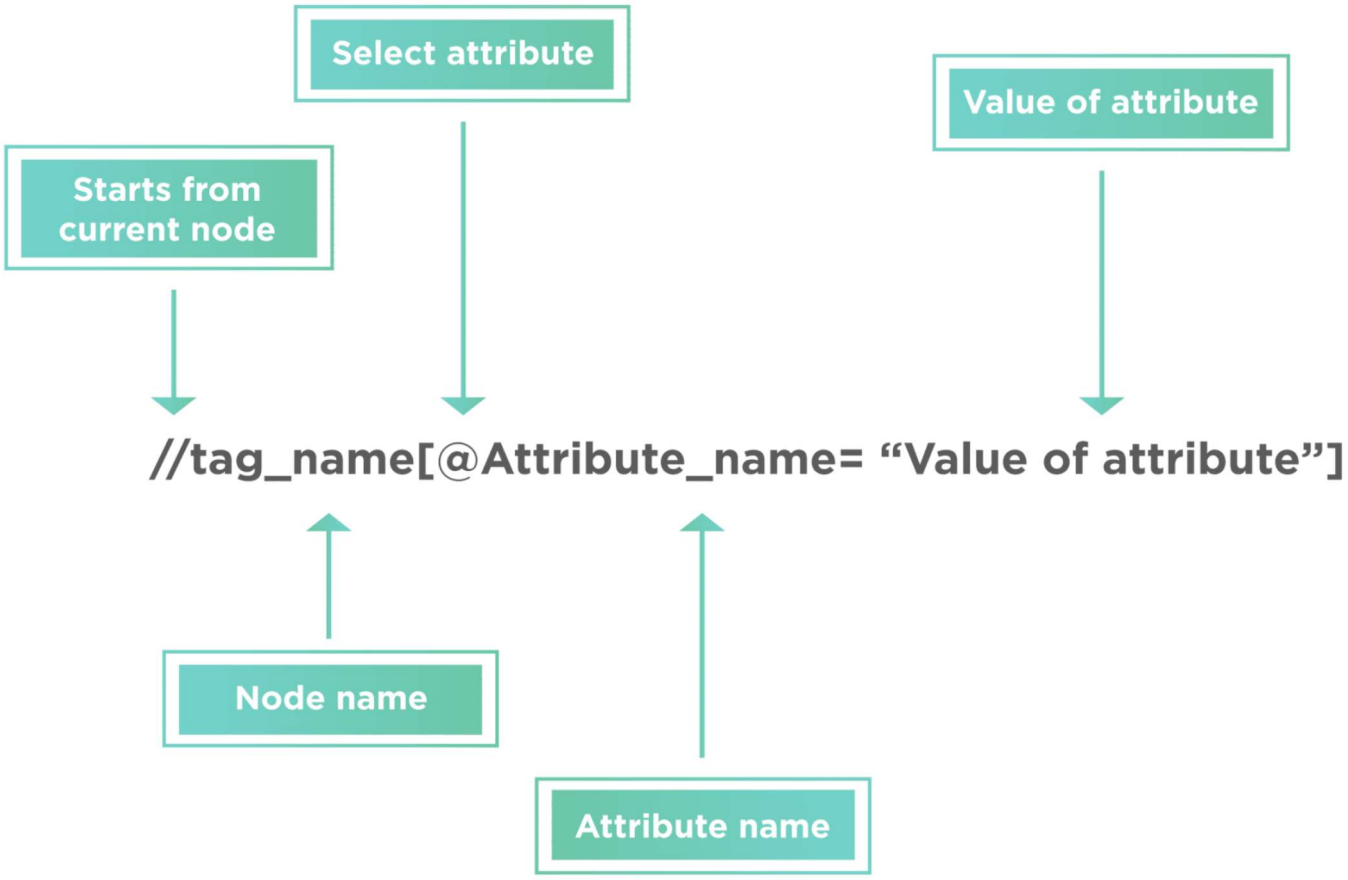
What is XPath?

XML Path or commonly known as ***XPath***, is a query language for ***XML*** documents. It is made of *path expression* governed by some pre-defined conditions. Moreover, *XPath* can be effectively used to identify and locate almost any element present on a web page. It is quite helpful while working on a dynamic web page where the web element's unique attributes are not constant. Moreover, it allows us to write ***XML document navigation flow*** for locating web elements. Consequently, let's first understand what syntax does *XPath* uses to locate the web elements:

Syntax of XPath

XPath finds any element within a *webpage* by using *DOM*. So, its syntax is also made up of ***DOM attributes and tags***, as shown below:

```
XPath = //tag_name[@Attribute_name = “Value of attribute”]
```



The XPath syntax generally starts with “//” double slash. That is to say, it will begin with the current node defined by the tag name.
The next part is ***tag_name****; it denotes the HTML tag name of the node.*
Subsequently, anything present inside the node encloses in the square brackets.
Additionally, the “@” sign selects the attribute.
*Moreover, the “****Attribute_name****” is the name of the attribute of the node.*
*And, the “****Value of attribute****” denotes the attribute value from the node.*

It is the basic structure of the *XPath*. We will learn more about *XPath* and write better *XPath* expression as we move ahead with the tutorial. Subsequently, let's first understand what XML DOM is and how a web page relates to an *XML* structure:

What is HTML/XML DOM?

All the ***HTML*** web pages are internally represented as an ***XML*** document. Additionally, the XML document has a *tree-like structure*, where we have different tags and attributes. For example, if we open the *Chrome Dev tools* section by *right-clicking* on the page "***https://demoqa.com/***" and selecting the "***Inspect***" option, the HTML structure will look as follows:

It starts with a tag `<html>` which has two child nodes **<**`<head>` and `<body>`. Here `<body>` again has a child node like `<div>`, which has its child nodes. Further, you can also see the first `<div>` tag has an "***id***" attribute, which has a value "***app***". Similarly, the child node of the `<div>` also has its own set of attributes and values.

One thing that we can notice here is every node that opens is again closed by using *forward slash* before the tag name, such as the `<body>` tag is closed using `</body>`. So, this way, we can see that all the *HTML* pages are internally represented as an *XML* document and constitute the *XML DOM (Document Object Model*).

How to locate web elements using XPath in Selenium?

In the previous section, we learned about the *basic syntax of XPath*. But did you notice some special characters like double slash ***"//"*** and ***"@"*** that help us to locate and select the desired node/element? Apart from the"//" and "@",Selenium provides various other syntax elements and attributes to *locate the web elements using XPath*. Few of these are:

Syntax Element	***Details***	***Example***	***Example Details***
Single slash "/"	It selects the node from the root. In other words, if you want to select the first available node, this expression is used.	/html	It will look for the *HTML* element at the start of the document.
Double slash "//"	It selects any element in the DOM that matches the selection. Additionally, it doesn't have to be the exact next node and can be present anywhere in the DOM.	//input	It will select the input node present anywhere in the *DOM*.
Address sign "@"	It selects a particular ***attribute*** of the node	//@text	It will select all the elements which have text attribute.
Dot "."	It selects the ***current*** node.	//h3/.	It will give the currently selected node, i.e., ***h3***.

Syntax Element	***Details***	***Example***	***Example Details***
Double dot ".."	It selects the ***parent*** of the current node.	//div/input/..	It will select the parent of the current node. The current node is input so that it will select the parent, i.e., ***"div"***.
Asterisk "*"	It selects ***any element*** present in the node	//div/*	This matches with any of the child nodes of the ***"div"***.
Address and Asterisk "@"*	It selects ***any attribute*** of the given node.	//div[@*]	It matches any of the ***div*** nodes that contain at least one attribute of any type.
Pipe "|"	This expression is used to select a ***different*** path.	//div/h5	//div/form

Apart from the syntax as mentioned above components, *XPath in Selenium* provides a few advanced concepts. Moreover, we can find a web element at a specified position if the locator's *XPath* has resulted in multiple elements. These are ***Predicates***. Subsequently, let's understand *how we can use Predicates to locate web elements?*

How to locate web elements using Predicates?

Predicates find a specific node/element by its index. For example: ***//div/input[1]***. Moreover, it selects the first input element, which is the child of the ***div*** element. Few of the most used ***Predicates*** are:

Predicate	***Details***	***Example***	***Example Details***
Get the last node	We can get the last node using the function "***last()***" inside the square bracket.	//div/input[last()]	It will give us the last input node, which is the child of the div node.
Get the node with specified Position	We can get the node from specific positions by using "***position()***" inside the square bracket.	//div/input[position()='2']	It will provide us with the child node of div. In other words, input present at the second position in the hierarchy.

Let's understand the usage of all these syntaxes of *XPath* with the help of the following code snippet:

```
package TestPackage;

import java.util.List;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class xPathDemo {
```

```
	public static void main(String args[]) {

		System.setProperty("webdriver.chrome.driver", "C:\\Selenium\\chromedriver
		WebDriver driver = new ChromeDriver();

		driver.get("https://demoqa.com/text-box");

		// Single slash "/" to validate image at start of page
		Boolean imgFlag = driver.findElement(By.xpath("/html/body/div/header/a/im
		System.out.println("The image is displayed : " + imgFlag);


		// Double slash "//" to validate image
		Boolean img_Flag = driver.findElement(By.xpath("//img")).isDisplayed();
		System.out.println("The image is displayed (located by //) : " + img_Flag

		// Address sign "@" full name textbox
		driver.findElement(By.xpath("//input[contains(@id, 'userN')]")).sendKeys(

		// Dot "." - Full name texbox
		driver.findElement(By.xpath("//input[contains(@id, 'userN')]/.")).sendKey

		// Double dot ".." - Full name label
		String label = driver.findElement(By.xpath("//input[contains(@id, 'userN'
		System.out.println("The label of full text is : " + label);

		// Asterisk "*" - Full Name textbox
		driver.findElement(By.xpath("//div[contains(@id, 'userName-wrapper')]/div

		// Address and Asterisk "@*" - full name text box
		driver.findElement(By.xpath("//input[@*= 'userName']")).sendKeys("Full Na

		// Pipe "|" - to locate both full name and Email label
		List<WebElement> lst = driver.findElements(By.xpath("//label[@*= 'userNam

		// Iterating and printing both labels
		for (WebElement e : lst) {
			System.out.println(" The label is : " + e.getText());
		}

		/*
		 * Opening web table page
		 */

		driver.get("https://www.demoqa.com/webtables");

		// Get the last node - Last val in table
		boolean lstCol = driver.findElement(By.xpath("//div[@class='rt-tr -odd']/
		System.out.println("The last table element is displayed : " + lstCol);

		// Get the 2 node - validate 2 position in table
		boolean positionCol = driver.findElement(By.xpath("//div[@class='rt-tr -o
		System.out.println("The 2nd table element is displayed : " + positionCol)

		driver.quit();
	}

}
```

Here, we have covered all the syntaxes of the *XPath* in Selenium, which can locate the web elements on a web page.

What are the different types of XPaths in Selenium?

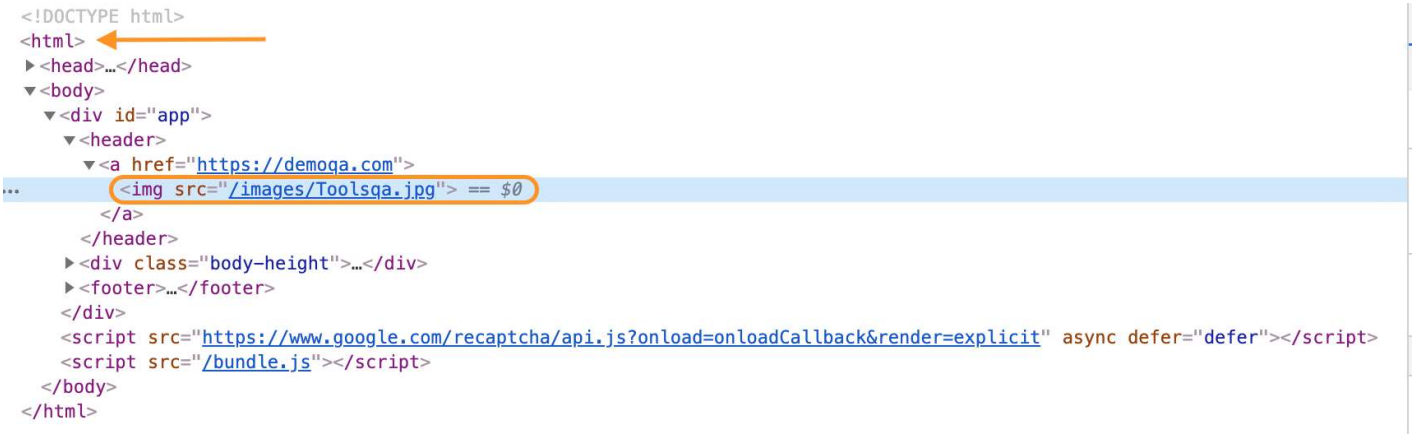
Selenium uses two strategies to locate elements using XPaths :

Locating a web element using an Absolute XPath.
Locating a web element using Relative XPath.

Let's understand both of these types in the following sections:

What is Absolute XPath in Selenium?

Absolute XPath is the direct way of finding the element. Moreover, it starts from the *first/root* node of the *XML/HTML* document and goes all the way to the required node following one node at a time. For better understanding, let's take an example, showing the DOM of page "***https://demoqa.com/***":



In the above document, if we want to find the ***absolute path of*** `<img>` ***node***, then we will have to transverse starting from the root node(`<html>` *node, as highlighted by the arrow in the above image*). So, the resultant absolute *XPath* of the element will look like this:

```
/html/body/div/header/a/img
```

Let's see the usage of the absolute *XPath in Selenium* to locate the *header image* of the web page "***https://demoqa.com/***".

```
import org.openqa.selenium.By;
```

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class XPathDemo {
    public static void main(String[] args) throws InterruptedException{

        System.out.println("Absolute XPath in Selenium");
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com");

        //Locate the web element using absolute xpath
        WebElement headerImage = driver.findElement(By.xpath("/html/body/div/header/a/img

        // Validate that the header image is displayed on the web page
        System.out.println("The image is displayed : " + headerImage.isDisplayed());

        driver.close();
        System.out.println("Execution complete!!");

    }
}
```

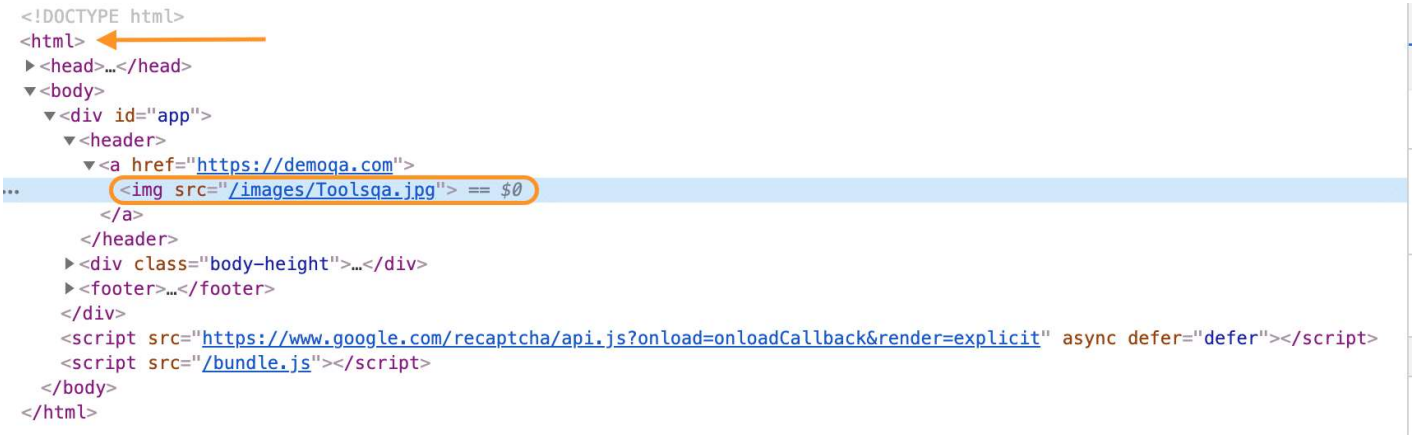
When you execute the above code snippet, it will show the output as follows:

```
Absolute XPath in Selenium
Starting ChromeDriver 85.0.4183.87 (cd6713ebf92fa1cacc0f1a598df280093af0c5d7-refs/branch-heads/4183@{#1689}) on port 29106
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Sep 18, 2020 11:47:02 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
The image is displayed : true
Execution complete!!
```

One major disadvantage or limitation of *absolute XPath* is that if there is any change in any of the elements on the *webpage*, then the *XPath* for any subsequent element will change. Hence, resulting in the failure of test scripts trying to locate the web element. So, generally, it is not recommended to use the absolute *XPath* for locating the Web Elements.

What is Relative XPath in Selenium?

Relative XPath starts from *any node* inside the *HTML DOM*; it need not start from the root node. It beings with a ***double forward slash***. For better understanding, let's take the same example as the absolute path above.



In the previous example, we followed the whole node hierarchy starting from "*<html>*" till the "*** <img? ***" tag. It allowed us to trace the whole path and trace the element's exact position on the page. But imagine after some time if one more node changes, this previous absolute *XPath* will become redundant, so this is where *relative XPath* comes handy.

The relative *XPath* uses the relative position of a particular node on the page. For example, if we want to write a relative *XPath* of the same image element that we defined in the absolute *XPath* section, we don't need to write all the nodes. Similarly, if we want to write the *XPath* for the *<img>* element, we don't need to start from the *<html>* tag. We can begin to directly from the *<img>* tag, as shown below:

```
//img[@src = "/images/Toolsqa.jpg"]
```

As we can see, relative *XPath* can start from any node. It also overcomes the fragile nature of *Absolute XPath*. Even if any preceding element changes or removes, it will not affect the ***Relative XPath***.

Let's modify the above code snippet to locate the image using the relative *XPath* in Selenium:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class XPathDemo {
    public static void main(String[] args) throws InterruptedException{

        System.out.println("Relative XPath in Selenium");
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com");

        //Locate the web element using relative XPath
        WebElement headerImage = driver.findElement(By.xpath("//img[@src = '/images/Tools

        // Validate that the header image is displayed on the web page
        System.out.println("The image is displayed : " + headerImage.isDisplayed());
```

```
            driver.close();
            System.out.println("Execution complete!!");

        }
    }
```

When you execute the above code snippet, it will show the output as follows:

```
Relative XPath in Selenium
Starting ChromeDriver 85.0.4183.87 (cd6713ebf92fa1cacc0f1a598df280093af0c5d7-refs/branch-heads/4183@{#1689}) on port 33660
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Sep 18, 2020 11:55:57 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
The image is displayed : true
Execution complete!!
```

So, this way, we can locate any of the web elements on a web page by using the *XPath in Selenium*, which always serves a savior locator strategy when we don't have the standard attributes of any web element are not available.

Key Takeaways

XPath is a powerful selector type for locating elements on a web page.
Additionally, XPath provides various syntax to locate the web elements on a web page.
Moreover, predicates in XPath locates specific nodes using the index of the web elements.
Finally, significant categories of the XPaths are Absolute and Relative XPath, and majorly relative Xpath is the recommended locator strategy.