XPath Axes

1. Ancestor
2. Siblings
   a. Following-sibling Axis
   b. Preceding-sibling Axis

Ancestor Axis:

The ancestor axis selects the common parent (parent, grandparent, great-grandparents, etc.) of the current node.

Syntax: /ancestor::tagname

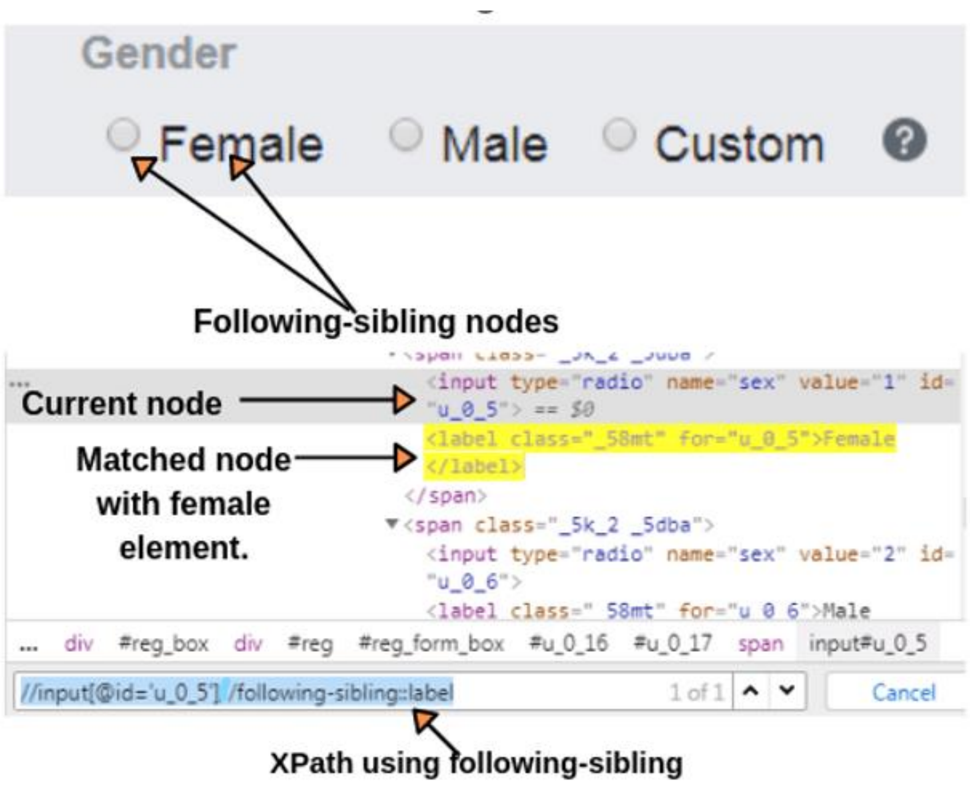
Siblings:

following-sibling:

The following-sibling selects all sibling nodes after the current node at the same level. i.e. It will find the element after the current node.

Syntax: /following-sibling::tagname

For example, the radio button of female and female text both are siblings on the Facebook home page as shown in the below screenshot.


Gender
Female
Male
Custom
Following-sibling nodes
Current node
Matched node with female element.
XPath using following-sibling

```
<input type="radio" name="sex" value="1" id="u_0_5"> == $0
<label class="_58mt" for="u_0_5">Female</label>
</span>
▼<span class="_5k_2 _5dba">
  <input type="radio" name="sex" value="2" id="u_0_6">
  <label class="_58mt" for="u_0_6">Male
```

... div #reg_box div #reg #reg_form_box #u_0_16 #u_0_17 span input#u_0_5

`//input[@id='u_0_5']//following-sibling::label` 1 of 1 Cancel

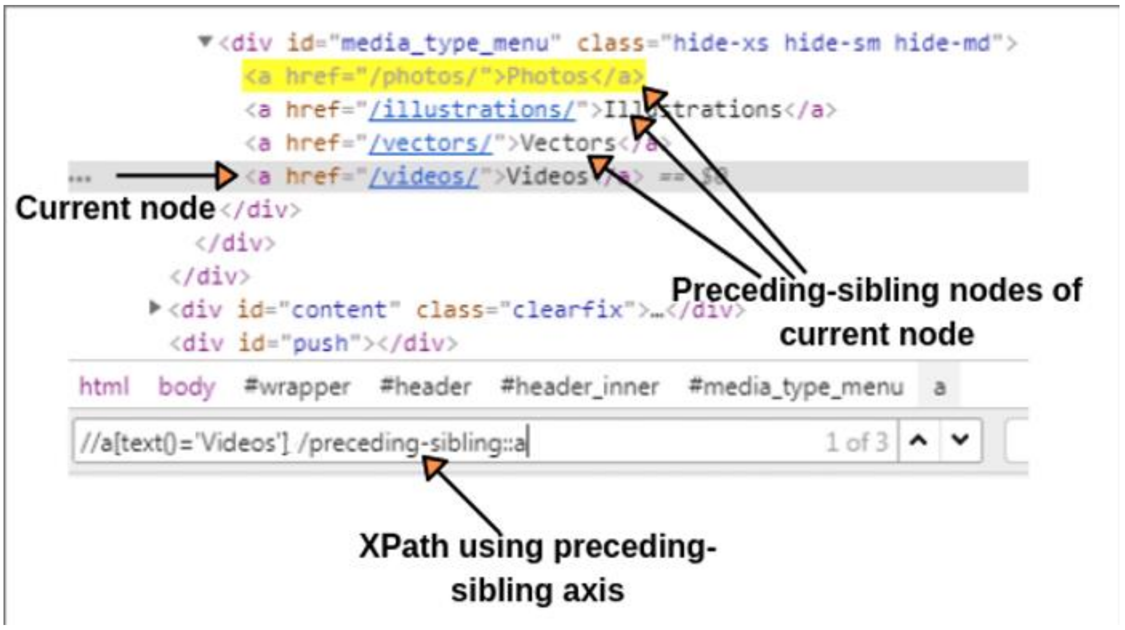
Preceding-sibling Axis:

The preceding-sibling axis selects all siblings before the current node

Syntax: /preceding-sibling::tagname

. Let’s take an example to understand the concept of the preceding-sibling axis.

Let’s consider videos link as current node as shown in below screenshot and find the XPath of current node by using text() method.



```
▼<div id="media_type_menu" class="hide-xs hide-sm hide-md">
   <a href="/photos/">Photos</a>
   <a href="/illustrations/">Illustrations</a>
   <a href="/vectors/">Vectors</a>
...<a href="/videos/">Videos</a> == $0
  </div>
 </div>
</div>
▶<div id="content" class="clearfix">…</div>
<div id="push"></div>
```

```
html  body  #wrapper  #header  #header_inner  #media_type_menu  a
```

```
//a[text()='Videos']/preceding-sibling::a          1 of 3
```

Current node
Preceding-sibling nodes of current node
XPath using preceding-sibling axis

Synchronization:

Synchronization: It is a process of matching the selenium speed with Application speed.

Types of Synchronization:

1.Implicitly wait

2. Explicitly wait

3.Thread.sleep

1.Implicitly wait :

It is a selenium wait which is used to match the selenium speed with application speed.

Syntax:

```
driver.manage().timeouts().implicitlyWait(Duration,
TimeUnit.SECONDS);
```

```
driver.manage().timeouts().implicitlyWait(20,
TimeUnit.SECONDS);
```

driver ---->Reference variable

manage()---->WebDriver method

timeouts()---->method inside the options nested interface.

implicitlyWait()---->method is used to set implicit wait time and available in TimeOuts nested interface.

TimeUnit-----> it is an inbuilt class in java available in util package

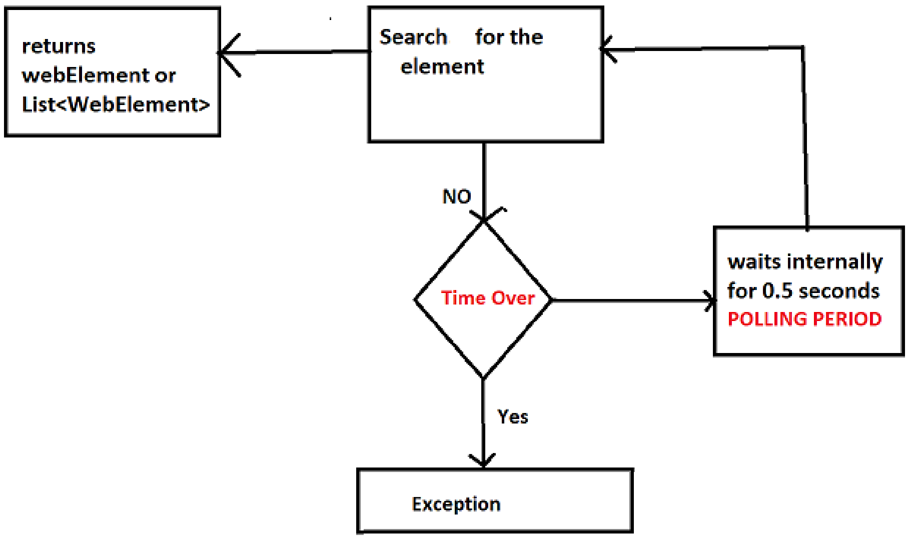
TimeUnit.SECONDS------>indicates that waiting period is in seconds.

- The **Implicit Wait in Selenium** is used to tell the web driver to wait for a certain amount of time before it throws a “No Such Element Exception”.
- Once we set the time, the web driver will wait for the element for that time before throwing an exception.

- Implicit wait is applicable for only findElement() and findElements();

How to Write implicit wait in a code:

```
driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
```



returns webElement or List<WebElement>
Search for the element
NO
Time Over
waits internally for 0.5 seconds
POLLING PERIOD
Yes
Exception

Explanation:

- It will search for the element in HTML tree Structure.
- If Element is found it will returns the address
- If Element is not found it will check for the given time.

- If the given time is not over it will internally wait for 0.5 seconds which is called Polling Period
- If the given time is over, it will throw No such element Exception
- This process repeats

1. Until the element is found
2. Until the given time is over

Drawbacks:

It is applicable only for findElement() and findElements().

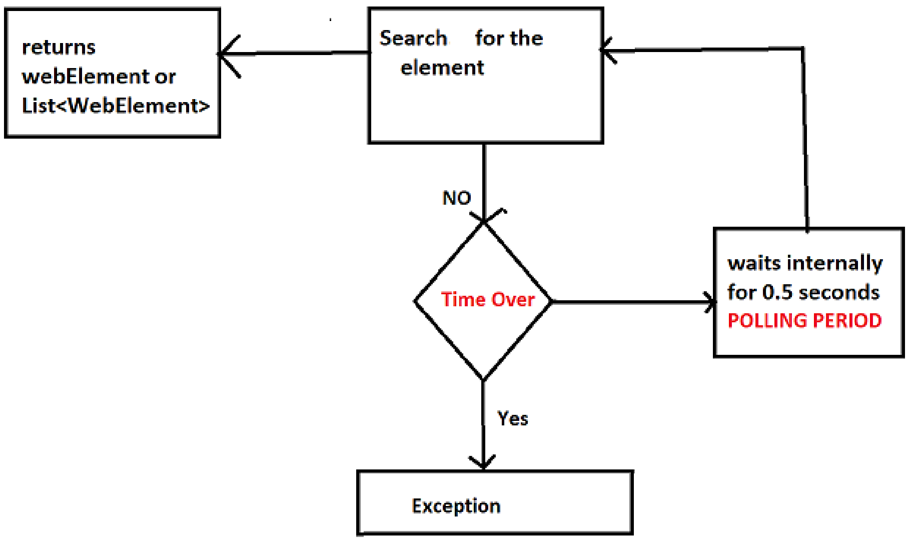
2.Explicitly Wait: It is a selenium wait which is used to match the selenium speed with application speed.

Syntax:

WebDriverWait wait=new WebDriverWait(driver,timeoutinseconds);

Wait.until(Expected conditions. visibilityOf(element)
elementToBeClickable(element)
titleContains(String));

```
WebDriverWait wait=new WebDriverWait(driver,10);
wait.until(ExpectedConditions.visibilityOf(ele));
wait.until(ExpectedConditions.elementToBeClickable(el
e).
```


returns webElement or List<WebElement>
Search for the element
NO
Time Over
waits internally for 0.5 seconds
POLLING PERIOD
Yes
Exception

<u>**Explanation:**</u>

It will search for the element in HTML tree Structure.

If Element is found it will returns the address

If Element is not found it willcheck for the given time.

If the given time is not over it will internally wait for 0.5 seconds which is called Polling Period

If the given time is over, it will throw TimeOut Exception

This process repeats

1. **Until the element is found**
2. **Until the given time is over**

3.Thread.sleep

- It is java wait
- It is applicable for only particular line
- **Syntax**: Thread.Sleep()

WebElements:

- It is an interface in selenium which contains 3 types abstract methods.

action()

getter()

Verification()

action():

sendKeys(): which is used send the data.

clear(): which is used to clear data

click():which is used to click the webelement.

Submit(): which is used to submit (nothing but a click). We will use this only whenever we have a type =submit in html tree structure.

Script:

```
package WebElements;

import java.util.concurrent.TimeUnit;

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

import io.github.bonigarcia.wdm.WebDriverManager;

// open amazon and type computer and clear the data
public class Actions_Method {
	public static void main(String[] args) throws
InterruptedException {
		WebDriverManager.chromedriver().setup();
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
		driver.get("https://www.amazon.in/");
		WebElement textField =
driver.findElement(By.xpath("//input[@id='twotabsea
rchtextbox']"));
		textField.sendKeys("computers");
		Thread.sleep(5000);


		textField.clear();


		Thread.sleep(5000);


		driver.close();
```

```
    }

}
```

Open the browser enter the Url of www.amazon.in

Type computers and wait for 5 seconds and clear the data.

getter():

getText():used to get the value.

getAttribute(): It is used to get the attribute value when we pass the attribute name as an argument.

getLocation():used to get X and Y coordinates of an element. Return type is point

getSize(): used to get the height and width of an web element. Return type is dimension

```
package WebElements;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.Point;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

public class Getter_Methods {

	public static void main(String[] args) throws Throwable {
		WebDriverManager.chromedriver().setup();
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
		driver.get("https://www.facebook.com/");
		WebElement password =
driver.findElement(By.xpath("//input[@id='pass']"));
		Point loc = password.getLocation();
		System.out.println("The x and y co ordinates of
password textfield are as below");
		System.out.println(loc.getX());
		System.out.println(loc.getY());
		Thread.sleep(5000);

		System.out.println("The size and width of password
textfield are as below");
		Dimension size = password.getSize();
		System.out.println(size.getHeight());
		System.out.println(size.getWidth());
	}

}
```

Verification():

isDisplayed()--->to check the element is displayed or not. Return type is Boolean

```
public class IsDisplayed {

	public static void main(String[] args)
	{
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://www.amazon.in/");
	driver.manage().timeouts().implicitlyWait(10,TimeUnit.SEC
ONDS);

		WebElement ele = 
driver.findElement(By.id("twotabsearchtextbox"));
		if (ele.isDisplayed())
		{
			System.out.println("Pass");
			ele.sendKeys("phone");
		}
		else
		{
			System.out.println("Fail");
		}
		driver.close();
	}

}
```

isEnabled()----->Used to check the element is enabled or not.Return type is Boolean

```
public class Isenabled {

	public static void main(String[] args)
	{
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://www.facebook.com/");

	driver.manage().timeouts().implicitlyWait(10,TimeUnit.SEC
ONDS);

		WebElement button = 
driver.findElement(By.name("login"));
		if (button.isEnabled())
		{
			System.out.println("Pass: element is enabled");
			button.click();
		}
		else
		{
			System.out.println("Fail: element is not 
enabled");
```

```
		}

		driver.close();
	}

}
```

isSelected(): Used to check the element is selected or not .Return type is Boolean

```
public class IsSelected {

	public static void main(String[] args)
	{
	// chrome driver
		WebDriverManager.firefoxdriver().setup();
		// opening chrome browser
		WebDriver driver = new FirefoxDriver();
		// maximizing the browser
		driver.manage().window().maximize();
		// implicitwait
		driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

		// entering the url
		driver.get("https://www.facebook.com/");
		driver.findElement(By.xpath("//a[text()='Create new
account']")).click();
		// address of female radio button
		WebElement radio =
driver.findElement(By.xpath("//input[@value='1']"));
		radio.click();
		if (radio.isSelected()) {
			System.out.println("pass");
		} else {
			System.out.println("fail");
		}
```

Navigation API:

Traversing from one page to another page is called as Navigation API.

API --->Application Programming interface

back--->diver.navigate().back();

farward()---> diver.navigate().forward();

refresh()----> diver.navigate().refresh();

navigate().to() ---->used to navigate from one website to another website.

driver.navigate().to(“url”);