Wait commands

Why do we need waits?

When Your Script execution speed doesn't match with the browser speed of loading elements, then we get Synchronization.

Because of Synchronization issue.

Synchronization issue – When the Script execution speed doesn't match with the browser speed.

What are Wait commands in Selenium?

The wait commands are essential when it comes to executing Selenium tests. They help to observe and troubleshoot issues that may occur due to variation in time lag.(Synchronization issues)

While running Selenium tests, it is common for testers to get the message "*Element Not Visible Exception*", "**NoSuchElementException**" ,"NoAlertPresentException".

This appears when a particular web element with which WebDriver has to interact, is **delayed in its loading**. To prevent this Exception, Selenium Wait Commands must be used.

In automation testing, Selenium Webdriver wait commands direct test execution to **pause for a certain length of time** before moving onto the next step. This enables WebDriver to check if one or more web elements are present/visible/enriched/clickable, etc when identifying certain elements. If an element is not located, then the "**ElementNotVisibleException** " appears. Selenium Wait commands help resolve this issue.

Important point – When the element is actually there,enabled but there is a browser delay issue.

Implicit Wait in Selenium

Implicit Wait directs the Selenium WebDriver to wait for a certain measure of time before throwing an exception. Once this time is set, WebDriver will wait for the element before the exception occurs.

It sets a **global wait** for a certain amount of time for the entire script. Selenium will wait for a specified amount of time before throwing a `NoSuchElementException` if an element is not present

f

Once the command is run, Implicit Wait remains for the entire duration for which the browser is open. It's default setting is 0, and the specific wait time needs to be set by the following protocol.

Implicit Wait Syntax

```
driver.manage().timeouts().implicitlyWait(10,TimeUnit.Seconds); // Selenium 3
```

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10)); // Selenium 4
```

Explicit Wait in Selenium

In Explicit wait – we are giving certain conditions to be satisfied.

We are instructing the webdriver to wait until that condition is met.

There will be a certain time limit that we have to mention – 10 seconds(Example).

We will get **TimeOutException**.

By using the Explicit Wait command, the WebDriver is directed to wait until a **certain condition occurs** before proceeding with executing the code.

This is a conditional Wait.

Explicit wait acts as a Specific to Webelement Wait and not a Global wait

It focuses on Individual elements and we can specify different conditions for each element to be waited.

Setting Explicit Wait is important in cases where there are certain elements that naturally take more time to load. If one sets an implicit wait command, then the browser will wait for the same time frame before loading **every web element**. This causes an unnecessary delay in executing the test script.

Explicit wait is more intelligent, but can only be applied for specified elements. However, it is an improvement on implicit wait since it allows the program to pause for dynamically loaded Ajax elements.

In order to declare explicit wait, one has to use ExpectedConditions. The following

Expected Conditions can be used in Explicit Wait.

1. alertIsPresent()
2. elementSelectionStateToBe()
3. elementToBeClickable()
4. elementToBeSelected()

5. frameToBeAvailableAndSwitchToIt()
6. invisibilityOfTheElementLocated()
7. invisibilityOfElementWithText()
8. presenceOfAllElementsLocatedBy()
9. presenceOfElementLocated()
10. textToBePresentInElement()
11. textToBePresentInElementLocated()
12. textToBePresentInElementValue()
13. titleIs()
14. titleContains()
15. visibilityOf()
16. visibilityOfAllElements()
17. visibilityOfAllElementsLocatedBy()
18. visibilityOfElementLocated()

Explicit Wait Syntax

```
WebDriverWait wait = new WebDriverWait(driver,30); // Selenium 3

        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5)); // Selenium 4
```

JavaScriptExecutor

In simple words, JavascriptExecutor is an interface that is used to execute JavaScript with Selenium.

To simplify the usage of JavascriptExecutor in Selenium, think of it as a medium that enables the WebDriver to interact with HTML elements within the browser.

JavaScript is a programming language that interacts with HTML in a browser, and to use this function in Selenium, JavascriptExecutor is required.

```
//importing the package

Import org.openqa.selenium.JavascriptExecutor;

//creating a reference

JavascriptExecutor js = (JavascriptExecutor) driver;

//calling the method

js.executeScript(script, args);
```

How JavascriptExecutor works in Selenium

Let’s try to understand the working of JavascriptExecutor using a simple example and implementation of both the JavascriptExecutor methods.

1.JavascriptExecutor in Selenium to click a button

```
js.executeScript(“document.getElementByID(‘element id ’).click();”);
```

2.JavascriptExecutor in Selenium to send text

```
js.executeScript(“document.getElementByID(‘element id ’).value = ‘xyz’;”);
```

3.JavascriptExecutor in Selenium to scroll down.

```
js.executeScript("window.scrollBy(0,250)", "");
```

JavascriptExecutor in Selenium to scroll down until the element is found into view

```
// Scrolling down the page till the element is found
js.executeScript("arguments[0].scrollIntoView();", Element);
```

What are Broken Links?

- To start with, a link is an HTML object that enables users to migrate from one web page to another when they click on it. It is a means to navigate between different web pages on the internet.
- A broken link, also often called a **dead link**, is one that does not work i.e. does not redirect to the webpage it is meant to.
- This usually occurs because the website or particular web page is **down or does not exist**. When someone clicks on a broken link, an error message is displayed.

Common Reasons for Broken Links

- 404 Page Not Found – The destination web page has been removed by the owner
- 400 Bad Request – The server cannot process the HTTP request triggered by the link because the URL address requested is wrong
- Due to the user’s firewall settings, the browser cannot access the destination web page.
- The link is misspelt

How to identify broken links in Selenium WebDriver

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

  WebDriver driver = new ChromeDriver();

driver.get("https://www.india.gov.in/");



  List<WebElement> links = driver.findElements(By.tagName("a"));
```

```
for (WebElement link : links) {

String url = link.getAttribute("href");

if (url != null) {

         try {

                  driver = new ChromeDriver();

         driver.navigate().to(url);


        if(!driver.getTitle().isEmpty()) {

           System.out.println("Valid Link: " + url);




         }

         else

         System.out.println("Broken Link: " + driver.getCurrentUrl());




         } catch (Exception e) {

        System.out.println("Broken Link: " + url);

         }

}

}
```

```
        driver.quit();
```

Programs :

```
package Day4;

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

public class Ebay {

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


                WebDriver driver = new ChromeDriver();

                driver.get("https://www.ebay.com/sellercenter/selling/how-to-sell");

                driver.manage().window().maximize();


                JavascriptExecutor j = (JavascriptExecutor)driver;

                Thread.sleep(2000);


                WebElement threesteps = driver.findElement(By.linkText("3 Steps to sell"));


                String link = threesteps.getAttribute("href");

                System.out.println(link);


                j.executeScript("arguments[0].scrollIntoView();", threesteps);


        }

}

package Day4;

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

public class Ebay {

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



		WebDriver driver = new ChromeDriver();

		driver.get("https://www.ebay.com/sellercenter/selling/how-to-sell");

		driver.manage().window().maximize();


		JavascriptExecutor j = (JavascriptExecutor)driver;

		Thread.sleep(2000);


		WebElement threesteps = driver.findElement(By.linkText("3 Steps to sell"));


		String link = threesteps.getAttribute("href");

		System.out.println(link);


		j.executeScript("arguments[0].scrollIntoView();", threesteps);


	}

}

package Day4;

import java.time.Duration;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
```

```
public class ExplicitWaitDemo {

          public static void main(String[] args) {


                    WebDriver driver = new ChromeDriver();


                    driver.get("https://rahulshettyacademy.com/AutomationPractice/");

                    driver.manage().window().maximize();

                    // Step 1 - Create object of WebDriverWait class

                    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5));


                    //Step 2 - use until() and ExpectedConditions class to define the condition

                    wait.until(ExpectedConditions.titleContains("Practice"));

                    // It will check for the title , if the title is not visible,
                    // it will wait for 5 seconds, if title is not found
                    // then it will throw TimeOutException
                    //if title is found then it will continue the execution

                    //By using this condition, you are providing the title to be loaded
                    // and wait time to be 5 seconds
                    WebElement textbox = driver.findElement(By.id("autocomplete"));

                    wait.until(ExpectedConditions.visibilityOf(textbox));
// This will check for the visibility and makes sure that the texbox is visible


                    textbox.sendKeys("Its Monday again!!!!!!!");


                    WebElement radio = driver.findElement(By.name("radioButton"));

                    wait.until(ExpectedConditions.elementToBeClickable(radio));

                    radio.click();


          wait.until(ExpectedConditions.elementToBeClickable(By.id("checkBoxOption1")));

                    driver.findElement(By.id("checkBoxOption1")).click();
          }

}

package Day4;

import java.time.Duration;

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

public class ImplicitWaitDemo {

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


                    WebDriver driver = new ChromeDriver();

                    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

                    //Thread.sleep(5000);

                    driver.get("https://rahulshettyacademy.com/AutomationPractice/");

                    driver.manage().window().maximize();


                    driver.findElement(By.id("autocomplete")).sendKeys("Its Monday again!!!!!!!");


                    driver.findElement(By.name("radioButton")).click();

                    driver.findElement(By.id("checkBoxOption1")).click();



          }

}

package Day4;

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

public class JavasciptExecutordemo {

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

		WebDriver driver = new ChromeDriver();

		driver.get("https://rahulshettyacademy.com/AutomationPractice/");

		driver.manage().window().maximize();


		// Step 1 - To use JavaScriptExecutor

		JavascriptExecutor js = (JavascriptExecutor)driver; // We are casting driver
variable to JSE interface

		//js.executeScript("document.getElementById('openwindow').click();");

		js.executeScript("document.getElementById('checkBoxOption1').click();");

		js.executeScript("document.getElementById('autocomplete').value='Hi Good
morning';");

		Thread.sleep(2000);

		js.executeScript("window.scrollBy(0,1000)");

		WebElement mousehover_ele = driver.findElement(By.id("mousehover"));


		// The below method will take 2 parameters , one of the JS code , the other of
the Webelement
		//It will perform Scroll to a particular Webelement
		// Consider arguments[0].scrollIntoView(); as a fixed syntax and we have to
mention a webelement in other paramater.

		js.executeScript("arguments[0].scrollIntoView();",mousehover_ele);



	}

}

package Day4;
```

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

public class ScrollIntoViewDemo {

	public static void main(String[] args) {

		WebDriver driver = new ChromeDriver();

		driver.get("https://www.redbus.in/");

		driver.manage().window().maximize();


		JavascriptExecutor j = (JavascriptExecutor)driver;


		WebElement booktrain = driver.findElement(By.linkText("Book Train
Tickets"));


		j.executeScript("arguments[0].scrollIntoView();", booktrain);


	}

}
```