## Task-4 - Selenium Wait - types with descriptions

## What is Selenium Waits?

Waits in Selenium is one of the important pieces of code that executes a ****test case.**** It runs on certain commands called scripts that make a page load through it. Selenium Waits makes the pages ****less vigorous**** and ****reliable.**** It provides various types of wait options adequate and suitable under favorable conditions. This ensures you don't mess up and get ended into failed scripts while performing automation testing with it.

Elaborately, [Selenium](https://www.javatpoint.com/selenium-tutorial) Waits helps the user to troubleshoot various issues while page redirection across different web pages. It is achieved by refreshing the entire web page and reloading it with new elements. At times, there's a call from ****Ajax**** as well. Thus, some time lag might exist while reloading pages and reflecting elements present on the web pages after refreshing.

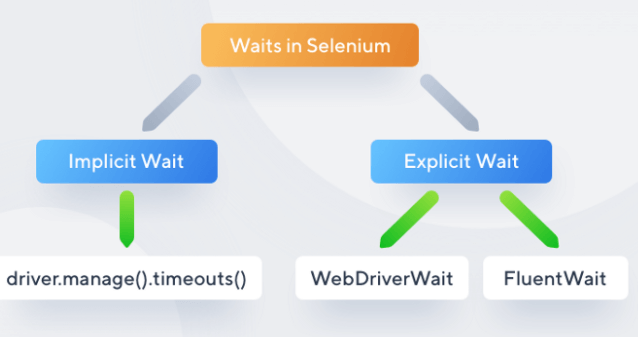
Another instance to understand Selenium Waits is navigating web pages back and forth with the ****navigate()**** command. This navigate() method comes from **[WebDriver,](https://www.javatpoint.com/selenium-webdriver)** whose main task is to simulate and manifest real-time scenarios like navigating between web pages concerning browsing history.

## Why Do You Need Waits In Selenium?

Today, most of the modern application's front-end is built on either [Ajax](https://www.javatpoint.com/ajax-tutorial) or [JavaScript](https://www.javatpoint.com/javascript-tutorial), followed by popular frameworks like Angular, React, or any other, which takes some time for loading elements on the web page. Hence, in such a case, Selenium throws an ****'ElementNotVisibleException'**** message when you tend to locate an element present in your script which is still not loaded on the web page.

To clarify, you can look at the below code snippet where automation testing with Selenium is executed.

The given code will help you showcase the same problem as you execute automation testing with Selenium. An example of easemytrip.com is used, where the posting user selects the 'From' and 'To' destination with a date of journey. The web application takes a certain time to load the required flight details. In this case, without applying Wait, the user tends to book the first flight from the list. Since the page hasn't loaded yet, the script failed to find the 'book now button. It results in throwing a ****'NoSuchElementException'.****



### **Implicit Waits**

The main function of implicit Wait is to tell the web driver to wait for some time before throwing a ****"No Such Element Exception".**** Its default setting is knocked at zero. Once the time is set, the driver automatically will wait for the amount of time defined by you before throwing the above-given exception.

****Syntax:****

driver.manage().timeouts().implicitlyWait(TimeOut,

TimeUnit.SECONDS);

### Explicit Waits

Explicit Waits also known as ****Dynamic Waits**** because it is highly specific conditioned. It is implemented by WebDriverWait class. To understand why you need Explicit Wait in Selenium, you must go through the basic knowledge of the wait statements in a program. In simple terms, you must know some conditions. Such conditions have been created to give you a gist of the Explicit Waits and why they are important.

**Syntax:**

WebDriverWait wait=new

webDriverWait(WebDriveReference,TimeOut);

**Fluent Wait**

Fluent Wait is quite similar to explicit Wait. It is similar in terms of management and functioning. In Fluent Wait, you can perform wait for action for an element only when you are unaware of the time it might take to be clickable or visible. Few differential factors that Fluent offers are as follows:

**The pooling frequency**

The pooling frequency in the case of Explicit is 500 milliseconds. But, using Fluent Wait, this pooling frequency can be changed to any value based upon your need. This usually means telling the script to keep an eye on the element after every 'x' seconds.

**Ignore Exception**

While pooling, if an element is not found, you can ignore some expectations like 'NoSuchElement'. Apart from this factor, similar to Explicit and Implicit Wait, you can define the amount of time for the element to be actionable or visible.

**Syntax:**

Wait<WebDriver> fluentWait = new FluentWait<WebDriver>(driver)

.withTimeout(60, SECONDS) // this defines the total amount of

time to wait for

.pollingEvery(2, SECONDS) // this defines the polling frequency

.ignoring(NoSuchElementException.class); // this defines the

exception to ignore

WebElement foo = fluentWait.until(new Function<WebDriver,

WebElement>()

{

public WebElement apply(WebDriver driver) //in this method

defined your own subjected conditions for which we need to wait for

{ return driver.findElement(By.id("foo"));

}});

## **Difference Between Implicit Wait Vs Explicit Wait**

Following is the main difference between implicit wait and explicit wait in Selenium:

|  |  |
| --- | --- |
| **Implicit Wait** | **Explicit Wait** |
| * Implicit Wait time is applied to all the elements in the script | * Explicit Wait time is applied only to those elements which are intended by us |
| * In Implicit Wait, we need ****not**** specify “ExpectedConditions” on the element to be located | * In Explicit Wait, we need to specify “ExpectedConditions” on the element to be located |
| * It is recommended to use when the elements are located with the time frame specified in Selenium implicit wait | * It is recommended to use when the elements are taking long time to load and also for verifying the property of the element like(visibilityOfElementLocated, elementToBeClickable,elementToBeSelected) |