**What is Wait Commands in Selenium?**

[Selenium wait commands](https://www.browserstack.com/guide/wait-commands-in-selenium-webdriver) are used in [test automation](https://www.browserstack.com/guide/test-automation-standards-and-checklist) to handle the dynamic loading of [web elements](https://www.browserstack.com/guide/selenium-webelement-commands). Since some elements might have been created with the help of ajax or front-end frameworks, it takes some time for them to load, but since they haven’t been loaded yet, running tests on them fails.

Selenium wait commands pause the test execution for a certain amount of time until those elements are ready in the DOM.

**Types of Wait Commands in Selenium**

**Implicit Wait Command in C#**

In Selenium C#, **Implicit Wait** is used to instruct the WebDriver to wait for a certain amount of time before throwing an exception if it cannot find an element. This wait is applied globally for all elements, meaning the WebDriver will repeatedly check for the presence of an element in the DOM during the specified time period before proceeding with the next step.

Implicit wait pauses the execution of the web driver for a specified period before throwing any error. The specified time is based upon the time required by the web elements to get ready for the test, and hence get loaded on the page. However, the execution time of the overall test increases.

If the particular element takes more time than what is specified, the Selenium web driver throws an error “**NoSuchElementException**“.

**The syntax for using the Implicit wait command in Selenium C# is as follows.**

driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(Value);

Let’s understand how to use implicit wait with the help of an example. In this example, we will navigate to Google’s homepage and will wait there for 20 seconds before performing the next step of our test. Moreover, this will also increase the execution time by 20 seconds.

using System;

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

namespace Test

{

    class Program

    {

     Public static void Main(string[] args)

     {

         //Initialising ChromeDriver

         IWebDriver driver = new ChromeDriver();

         //Navigating to Google's homepage

         driver.Navigate().GoToUrl(" https://www.google.com ");

         //Applying Implicit Wait command for 20 seconds

            driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(20);

         //Clicking on an element after waiting for 20 seconds

         driver.FindElement(By.LinkText("I'm Feeling Lucky")).Click();

     }

    }

}

**When to Use Implicit Wait**

* For general waiting when you are unsure about the load time of elements.
* For static pages where elements are mostly available without too much delay.
* When you want a simple, global wait configuration for all element searches.

**When Not to Use Implicit Wait**

* When dealing with dynamic web pages or AJAX calls where elements might load at different intervals.
* When precise control over waiting times is needed, such as waiting for specific conditions like, element visibility, Explicit Wait is preferred.

**Explicit Wait Command in C#**

In Selenium C#, an **Explicit Wait** is used to wait for a certain condition to occur before proceeding with further actions. Unlike **Implicit Wait**, which is applied globally, **Explicit Wait** is more targeted and allows you to wait for specific conditions to be met, such as an element becoming visible, clickable, or present.

Implicit Wait command, in Selenium C# waits for a specific period. However, Explicit Wait in Selenium C# will wait till certain conditions occur. The wait here for a web element is not for a specific period but till the web element is ready in the DOM for testing. Explicit Wait C# is also known as “**smart wait**“.

The Explicit Wait command checks the condition (element to become clickable, displayed, etc) every 250ms. Moreover, Implicit wait is only applicable with [FindElement](https://www.browserstack.com/guide/findelement-in-selenium" \o "findElement vs findElements in Selenium" \t "_blank) methods, however, Explicit Wait has several possible conditions.

**The Selenium Webdriver provides two classes for the implementation of Explicit Wait.**

* WebDriverWait
* [ExpectedConditions](https://www.browserstack.com/guide/expectedconditions-in-selenium" \o "Understanding ExpectedConditions in Selenium" \t "_blank)

The **WebDriverWait** calls the **ExpectedConditions** class method until it returns successfully or the specified time is over. It is a major improvement for Implicit Wait, as there is no extra delay in the test.

**The syntax for the usage of Explicit Wait is as follows**

WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

wait.Until(ExpectedConditions.ElementExists(By.Id("id")));

In the above syntax, **ElementExists()** is used as a method, for example, however, **ExpectedConditions** class can contain several other methods also. Some of the common methods used are.

* AlertIsPresent()
* ElementIsVisible()
* ElementExists()
* ElementToBeClickable(By)
* ElementToBeClickable(IWebElement)
* ElementToBeSelected(By)
* TitleContains()
* UrlMatches()
* TextToBePresentInElementValue(IWebElement, String)
* TextToBePresentInElement()

Now, let’s understand how to use the Explicit wait command in Selenium C# with the help of an example.

**To use Explicit wait in your code, first install the following packages into your script.**

* **OpenQA.Selenium.Support.UI** – It will help in executing the WebDriverWait class
* **SeleniumExtras.WaitHelpers** – It will help execute ExpectedConditions class and its methods into our test script.

In this test, the test will navigate to the Gmail website, will fill up login details, and after signing in, we shall wait for the compose button to get loaded, and once it is loaded, the button will be clicked. After clicking the button, we shall wait for 20 seconds with the help of Implicit Wait, and then will close the browser.