1. **What is meant by a locator and name a few different types of locators present in Selenium**

A locator is an address for uniquely identifying web elements within a web page. There are different types of locators present in Selenium to identify web elements uniquely.

1. ID
2. ClassName
3. Name
4. Tagname
5. Linktex
6. Partial Linktext
7. Xpath
8. CSS selector
9. **What is the major difference between driver.close() and driver.quit()?**

driver.close() :

This command closes the browser’s current window. If multiple windows are open, the current window of focus will be closed.

driver.quit() :

When quit() is called on the driver instance and there are one or more browser windows open, it closes all the open browser windows.

1. **Different types of navigation commands.**

>> We have four Navigation commands

* **GoToUrl** driver.Navigate().GoToUrl("https://toolsqa.com");
* **Back** driver.Navigate().Back();
* **Forward** driver.Navigate().Forward();
* **Refresh** driver.Navigate().Forward();

1. **How to assert the title of a webpage?**

Assert.assertEquals(actualTitle, expectedTitle);

String actualTitle = driver.getTitle(); // Get the title of the webpage and store in a variable

String expectedTitle = “abcdefgh"; // Type in the expected title

if(actualTitle.equalsIgnoreCase(expectedTitle))

System.out.println("Title Matched");

else

System.out.println("Title didn't match");

**Assert.assertEquals(actualTitle, expectedTitle);**

1. **Screenshots in Webdriver**

TakeScreenshot interface can be used to take screenshots in WebDriver.

getScreenshotAs() method can be used to save the screenshot

File scrFile = ((TakeScreenshot)driver).getScreenshotAs(outputType.FILE);

1. **Explain the difference between FindElement and FindElements methods in Selenium WebDriver.**

**FindElement :** returns the first matching element on the page, while

**FindElements :** returns a collection of all matching elements. If no element is found,

FindElement throws an exception, while FindElements returns an empty list**.**

1. **How to wait until a web page has been loaded completely in Selenium?**

One approach is to use the "implicit wait" command in Selenium, which instructs the web driver to wait a certain amount of time before throwing an error if the element is not found or loaded. Another option is to use the "explicit wait" command to wait for a specific element to appear on the page before proceeding with the script.

1. **What are the types of waits supported by WebDriver?**

**Implicit Wait :**

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);

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

**Explicit Wait:**

In the Implicit Wait command, it waits for a specific period. However, in Explicit Wait, it 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. This is the reason it 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 methods, however, Explicit Wait has several possible conditions.

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

* WebDriverWait
* ExpectedConditions

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")));

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

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

**Fluent Wait Command in C#:**

The Fluent Wait command in Selenium is similar to Explicit Wait in so many aspects. It allows you to control the Web Driver to set the specified time for some condition to appear before it could throw the error “ElementNotVisibleException”.

The main advantage of implementing the Fluent Wait command is setting the polling frequency. Polling frequency is the frequency at which the driver checks for the element whether it has loaded or not. It has the attribute .pollingfrquency, and its default value is 500ms, which means the driver will check every 500 milliseconds before throwing the error.

**The syntax for the usage of Fluent Wait is as follows:**

**DefaultWait<IWebDriver> fluentWait = new DefaultWait<IWebDriver>(driver);**

**fluentWait.Timeout = TimeSpan.FromSeconds(5);**

**fluentWait.PollingInterval = TimeSpan.FromMilliseconds(polling\_interval\_in\_ms);**

DefaultWait<IWebDriver> fluentWait = new DefaultWait<IWebDriver>(driver);

fluentWait.Timeout = TimeSpan.FromSeconds(5);

fluentWait.PollingInterval = TimeSpan.FromMilliseconds(polling\_interval\_in\_ms);

1. **Basic Elements Operations**

* Click : driver.Click();
* SendKeys : driver.SendKeys(“input text”);
* Clear(); : driver.Clear();
* IWebElement element = driver.**FindElement**(By.**Id**(**"id"**));
* element.**Click**();
* element.**SendKeys**(**"someText"**);
* element.**Clear**();
* element.**Submit**();

1. **Select method in C# Selenium**

IWebElement element = driver.FindElement(By.Id("the web element id"));

* SelectElement select = new SelectElement(element);
* select.SelectByIndex(1);
* select.SelectByText("Ford");
* select.SelectByValue("ford");
* select.DeselectAll();
* select.DeselectByIndex(1);
* select.DeselectByText("Ford");
* select.DeselectByValue("ford");

1. **Dropdown in Selenium C#**

IWebDriver driver = new ChromeDriver(driverPath);

// Navigate to a webpage with the dropdown

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

// Find the dropdown element by its ID

IWebElement dropdown = driver.FindElement(By.Id("dropdown"));

// Option 1: Using SelectElement class

SelectElement select = new SelectElement(dropdown);

// Select by value

select.SelectByValue("option2");

// Option 2: Using FindElement and Click

// Click the dropdown to open it

dropdown.Click();

// Find and click the desired option

driver.FindElement(By.CssSelector("#dropdown option[value='option3']")).Click();

// Option 3: Using JavaScriptExecutor

IJavaScriptExecutor js = (IJavaScriptExecutor)driver;

// Execute JavaScript to select an option by value

js.ExecuteScript("arguments[0].value='option1'", dropdown);