200+ Selenium Interview Questions

Selenium Interview Questions [Core Selenium Concepts]

What Is Selenium? Why Is It Used?

Selenium is a cross-platform and portable automation testing framework created by Jason Huggins in 2004.

It has seen several upgrades since then, WebDriver 2.0 in 2011, and 3.0 in 2016. Check the below Selenium timeline.

We can use Selenium to write automation scripts for testing web applications. It helps in reducing manual testing efforts, improves productivity and quality.

Cross-platform means a Selenium script written on one OS say Windows can run on other supported systems such as Linux or Mac OS without making any changes.

Portable means the same script created for a desktop-based device can run on different supported devices such as tablets, mobiles, etc.

What Is Selenium RC, And How Does It Talk To A Browser?

Selenium RC a.k.a. Remote Control came out in early 2004.

It exposed a set of client APIs which use to send commands to a server to perform actions on the browser.

What Is Selenium 2.0, And Why Is It Superior To RC?

Selenium 2.0 a.k.a. WebDriver is a set of native APIs which directly send commands to the browser instead of delegating to a server.

Its release came out in 2011. It provided the native WebDriver API interface replacing the old client-server API model.

What Is Selenium 3.0, And Why Is It Superior To 2.0?

It was a major WebDriver upgrade which landed up in 2016.

It is superior to 2.0 because it implemented the W3C specifications for Webdriver APIs to make them a global standard.

Until 3.0, it was the Selenium community that was supplying Web drivers for specific browsers. But now, they provide their respective drivers that support the standard Web driver API interface.

What Is The Latest Selenium Webdriver Architecture?

Below block diagram explains the latest architecture Selenium Webdriver supports.

<img class="aligncenter size-full wp-image-7170" src="//cdn.techbeamers.com/wp-content/uploads/2015/08/Selenium-Webdriver-Architecture.png" alt="Selenium Webdriver Architecture" width="537" height="251" srcset="//cdn.techbeamers.com/wp-content/uploads/2015/08/Selenium-Webdriver-Architecture.png 537w,//cdn.techbeamers.com/wp-content/uploads/2015/08/Selenium-Webdriver-Architecture-300x140.png 300w" sizes="(max-width: 537px) 100vw, 537px">

What Are The Different Components Of The Selenium Framework?

Selenium is an automation development kit that comprises the following components.

- **Selenium IDE:**

A Firefox/chrome extension to record and play the user actions performed on a web page.

- **Selenium RC:**

A Selenium Remote Control which exposes APIs for scripting tests in different languages and also runs them in browsers.

- **Selenium Webdriver:**

These are native APIs that directly interact with the browser. They give more control and faster than the RC APIs.

- **Selenium Grid:**

It provides concurrency. With its help, we can split testing and run a set of cases on one machine and some on another.

What Are The Benefits Does WebDriver Has Over Selenium RC?

Selenium RC had a complex architecture, whereas WebDriver removed those complications. The below points discuss why WebDriver is better than the RC.

- Selenium RC is slow because it has an additional JavaScript layer known as the core. On the contrary, WebDriver is fast as it natively interacts with the browser by utilizing its built-in engine.
- Selenium core can't ignore the disabled elements, whereas WebDriver handles the page elements more realistically.
- Selenium RC has a mature set of APIs but suffers from redundancies and complicated commands. On the other hand, WebDriver APIs have a cleaner interface and do not have any such problems.
- Selenium RC doesn't provide support for the HtmlUnit browser, whereas the WebDriver has a headless HtmlUnit driver.
- Selenium RC includes a test result generator to produce HTML reports. Web Driver doesn't have any built-in reporting ability.

What Are The Benefits Of Selenium Automation?

There are many benefits of using Selenium for automated testing.

- **Open source:** Since it is an OSS, so we don't have to bear any licensing cost for using it.
- **Cross-browser:** It works on all standard browsers such as Chrome, FF, IE, and Safari. We can run same the test script in all browsers.
- **Multi-language:** We can choose a familiar programming language from Java, Python, C#, Ruby to use with Selenium.
- **Cross-platform:** It provides test compatibility across Windows, Linux, and Mac OSX. We can run same the test script on all platforms.
- **Concurrency:** With Selenium Grid, we can execute thousands of tests in parallel.
- **CLI support:** We can create a test suite with hundreds of tests and launch it with a single command.
- **CI support:** Jenkins is the best CI tool. It provides a Selenium plugin to configure tests for nightly execution.
- **Third party integration**
- **Free help:** We can get quick technical support from its large community.
- **Tester friendly:** A non-programmer can also automate using Selenium. It is not too complicated to be used only by a programmer.
- **Ongoing project:** Active development and bug fixes on the latest project.

What Are Some Downsides To Selenium Automation?

Selenium is a perfect clinical tool to impersonate user actions in a browser. However, it also has a few limitations given below.

- Doesn't support automation of Windows applications
- Can't perform mobile automation on its own
- Lacks a good built-in reporting
- Not 100% perfect for handling dynamic web elements
- Poses challenges while processing popups or frames

- Not that efficient in coping with the page load
- Can’t automate a captcha

What Programming Languages Does Selenium Allow To Use?

Selenium Webdriver supports the following programming languages.

- Python
- Java
- C-Sharp
- JavaScript
- Ruby
- PHP
- Perl

What Are The OS/Platforms Does Selenium Support?

Following is the list of OS/platforms which Selenium supports.

- Windows Desktop
- Windows Mobile
- Linux
- Mac OS X
- IOS
- Android

What Types Of Cases Can You Automate With Selenium?

We can use Selenium for automating the following types of cases.

- Functional cases
- Regression test cases
- Acceptance tests
- Sanity test cases
- Smoke testing
- End-to-end test cases
- Cross-browser tests
- Integration tests
- Responsiveness cases
-

Can You Use Selenium For Testing Rest API Or Web Services?

Selenium provides Native APIs for interacting with the browser using actions and events. The Rest API and the web services don’t have any UI and hence can’t be automated using Selenium.

What Are The Different Types Of Web Driver APIs Supported In Selenium?

Following are the web drivers supported in Selenium.

WebDriver Name	WebDriver API	Supported Browser
1 Gecko Driver (a.k.a. Marinetto)	FirefoxDriver()	Firefox
2 Microsoft WebDriver (a.k.a. Edge)	InternetExplorerDriver()	IE
3 Google Chrome Driver	ChromeDriver()	Chrome
4 HTML Unit Driver	WebClient()	{Chrome, FF, IE}
5 OperaChromium Driver	ChromeDriver()	Opera
6 Safari Driver	SafariDriver()	Safari
7 Android Driver, AndroidDriver()	Android browser	
8 ios Driver	IOSDriver()	ios browser
9 EventFiringWebDriver	EventFiringWebDriver()	ALL

Which Of The WebDriver APIs Is The Fastest, And Why?

It is none other than the HTMLUnitDriver, which is faster than all of its counterparts.

The technical reason is that the **HTMLUnitDriver** doesn’t execute in the

browser. It employs a simple HTTP request-response mechanism for test case execution.
This method is far quicker than starting a browser to carry out test execution.

What Do You Know About Selenium IDE?
Selenium IDE is a trendy Firefox /CHROME browser extension. It presents a graphical interface to record, play, and save user actions. Test automation beginners can find it convenient for web automation testing. After capturing a workflow, we can also export the steps in various formats.
Shinya Kasatani was the developer who created the Selenium IDE. The idea crossed his mind while he was investigating the JavaScript code inside the Selenium core. He then did a proof of concept and developed it into a full-fledged IDE.
What Do You Know About Selenese?
Selenium IDE has a built-in linguistic system known as Selenese. It is a collection of Selenium commands to perform actions on a web page.
For example, it can detect broken links on a page, check the presence of a web element, Ajax, JavaScript alerts, and a lot more. There are mainly three types of command in Selenese.

- **Actions:** It can alter the state of an application. For example, you are clicking on a link, selecting a value from the drop-down, etc.
- **Accessors:** These commands monitor the state of an application and cache it into some variable. For example, storeTextPresent, storeElementPresent etc.
- **Assertions:** These commands allow adding a checkpoint or a verification point. They confirm the current state of a UI element.

What Do You Know About Selenium Grid?
Selenium Grid is a tool which can distribute tests across multiple browsers or different machines. It enables parallel execution of the test cases. Using this, we can configure to run thousands of test cases concurrently on separate devices or browsers.

What Do You Use Selenium Grid?
Selenium Grid allows the test cases to run on multiple platforms and browsers simultaneously, and hence, supports distributed testing.
This ability to parallelize the testing is what makes us use the Selenium Grid.

What Are The Pros/Benefits Of Using Selenium Grid?
There are a no. of benefits of using the Selenium Grid.

- Support concurrent test execution and hence saves us a lot of our time.
- It presents us with the ability to execute test cases in different browsers.
- After creating multi-machine nodes, we can use it to distribute tests

and execute them.
-

What Does A Hub Do In Selenium Grid?
A hub is analogous to a server that drives the parallel test execution on different machines.

What Does A Node Do In Selenium Grid?
The machine we register to a hub represents a node. The registration allows the grid to fetch node configuration and execute tests. There could be multiple nodes that we can bind to a Selenium Grid.

What Is The Command To Bind A Node To Selenium Grid?
Please make sure to download the Selenium-server jar before running the below command.
java -jar <selenium-server-standalone-x.xx.x.jar> -role node -hub http://<node IP>:4444/grid/register

Which Of Java, C-Sharp, Or Ruby Can We Use With Selenium Grid?
- **Java:** Yes, we can. Moreover, we can do parallel testing using TestNG to utilize the Selenium grid features.
- **C-Sharp:** Yes, we can. We need to use "Gallio" to run our tests concurrently in the grid.
- **Ruby:** Yes, we can. We'll use the "DeepTest" to distribute our tests across the grid.

What Are Selenium Grid Extras? What Additional Features Does It Add To Selenium Grid?

Selenium Grid Extras is a set of management scripts to manage nodes more efficiently.
Below is a summary of the features provided by extras.
- More control over the individual nodes
- Kill a browser running tests by name
- Stop a process or program by its PID
- Shift the mouse to a specific coordinate
- Retrieve physical memory (RAM) and persistent storage (Disk) usage statistics
- Auto upgrade to newer versions of WebDriver
- Reset/re-start test nodes after a specified no. of iterations
- Centrally manage the configuration of all nodes
- Capture screenshots during error conditions

What Is The Difference Between MaxSessions Vs. MaxInstances Properties Of Selenium Grid?

Sometimes we get confused while differentiating between the MaxSessions vs. MaxInstances. Let's understand the difference with clarity.

1. MaxInstances: It is the no. of browser instances (of the same versions) that can run on the remote machine. Check the below commands.
-browser browserName=firefox,version=59,maxInstances=3,platform=WINDOWS

-browser
browserName=InternetExplorer,version=11,maxInstances=3,platform=WINDOWS
It means we can run three instances of both Firefox and IE at the same time. So, a total of six different browsers (FF & IE) could run in parallel.

2. MaxSession: It dictates how many browsers (independent of the type & version) can run concurrently on the remote machine. It supersedes the "MaxInstances" setting.
In the previous example: If the value of "maxSession" is one, then no more than a single browser would run. While its value is two, then any of these combinations (2FF, 2IE, 1FF+1IE) can run at a time.

Selenium Interview Questions [Locators Related]

What Are The Locators Selenium Supports?
Following is the list of supported locators by Selenium.
- **ID:** Unique for every web element
- **Name:** Same as ID although it is not unique
- **CSS Selector:** Works on element tags and attributes
- **XPath:** Searches elements in the DOM, Reliable but slow
- **Class name:** Uses the class name attribute
- **TagName:** Uses HTML tags to locate web elements
- **LinkText:** Uses anchor text to locate web elements
- **Partial Link Text:** Uses partial link text to find web elements

Which Of The Id, Name, XPath, Or CSS Selector Should You Use?

If the page has unique names or identifiers available, then we should use them.

If they are not available, then go for a CSS selector as it is faster than the XPath.

When none of the preferred locators is present, then you may try the XPath.

What Is XPath? How Does It Work?
XPath is the most-used locator strategies Selenium uses to find web elements.

- It works by navigating through the DOM elements and attributes to locate the target object. For example – a text box or a button or checkboxes.
-
- Although it guarantees to give you the element you are looking after. But it is slower than as compared to other locators like ID, name, or CSS selectors.
-

What Does A Single Slash "/" Mean In XPath?

A single (forward) slash “/” represents the absolute path.
In this case, the XPath engine navigates the DOM right from the first node.
/html/body/div/div[2]/input

What Does A Double Slash “//” Mean In XPath?
A double (forward) Slash “//” represents the relative path.
In this case, the XPath engine searches for the matching element anywhere in the DOM.
//div//input[@id=’test’]

What Is An Absolute XPath, Explain With Example?

An absolute XPath will always search from the root node until it reaches the target. Such an XPath expression includes the single forward-slash (/) as the prefix.
/html/body/div[1]/div[5]/form/table/tbody/tr[3]/td/input

What Is A Relative XPath, Explain With Example?

A relative XPath doesn’t have a specific point to start. It can begin navigation from any node inside the DOM and continues. Such an XPath expression includes the double forward-slash (//), as given below.
//div[@id=app]
How Do You Locate An Element By Partially Comparing Its Attributes In XPath?

XPath supports the contains() method. It allows partially matching of attribute’s value.
It helps when the attributes use dynamic values while having some fixed part.
See the below example-
xPath usage => //*[contains(@category, 'tablet')]
id=test_123
test_234
test_345

//input[contains(@id, ‘test_’)]

The above expression would match all values of the category attribute having the word ‘tablet’ in them.

How Do You Locate Elements Based On The Text In XPath?
We can call the text() method. The below expression will get elements that have text nodes that equal ‘Python.’
xPath usage = //a[text()='Python']

How Do You Access The Parent Of A Node With XPath?
We can use the double dot (“..”) to point to the parent of any node using the XPath.
For example – The locator **//span[@id=”current”]/..** will return the parent of the span element matching id value as ‘current’.

How Do You Get To The Nth Sub-Element Using The XPath?
We can modify the XPath expression to get to the nth element in the following ways:
1. Use XPath as an array by appending the square brackets with an index.
Example
tr[2]
The above XPath expression will return the second row of a table.
2. By calling position() in the XPath expression
Example
tr[position()=4]

The above XPath will give the fourth row.

How Do You Use “Class” As A CSS Selector?
We can use the below syntax to access elements using the class CSS selector.
.<class>
e.g. .color
It can help to select all elements related to the specified class.

How Do You Use “ID” As A CSS Selector?
We can use the below syntax to access elements using ID as the CSS selector.
#<ID>
e.g. #name

How To Specify Attribute Value While Using The CSS Selector?
Here is the syntax to provide the attribute value with the CSS selector.
[attribute=value]
e.g. [type=submit]

How To Access The Nth Element Using The CSS Selector?
Here is the syntax to access the nth attribute using the CSS selector.
<type>:nth-child(n)
e.g. tr:nth-child(4)

What Is The Primary Difference Between The XPath And CSS Selectors?
With the XPath, we can traverse both forward and backward, whereas CSS selector only moves forward.

Part –2:

Selenium Interview Questions [API Or Commands Related]

What Are The Steps To Create A Simple Web Driver Script?
Below are the minimal steps to create a WebDriver script.

- Launch the Firefox or Chrome browser by creating the WebDriver ref pointing to ChromDriver /FirefoxDriver class.
-
- Open website www.google.co.in using the get() method.
- Wait for the web page to load.
- Display a message on the console that the webpage gets loaded successfully.
- Close the browser.

Source Code:

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

        WebDriver driver = new FirefoxDriver();


        String URL = "https://www.google.co.in";
        driver.get(URL);

        driver.manage().timeouts().implicitlyWait(25, TimeUnit.SECONDS);

        System.out.println("Webpage gets loaded successfully!!");

        driver.close();
  }
}
```

What Does FirefoxDriver Mean, A Class Or An Interface?

FirefoxDriver is a Java class, and it implements the WebDriver interface.

Which Is The Super Interface Of Selenium Web Driver?
Th SearchContext acts as the super interface for the Web Driver.

It is the external interface which has only two methods: findElement() and findElements()

What Does The Webdriver Driver = New FirefoxDriver(); Mean?

```
Webdriver driver = new FirefoxDriver();
```

The above line of code represents the following:
- The driver is a variable of type 'Webdriver' interface.
- We are instantiating an object of the FirefoxDriver class and storing it into the driver variable.

Why Do We Create A Reference Variable Of Type Webdriver, Not The Actual Browser Type?
It is because we could use the same Webdriver variable to hold the object of any browser, such as the ChromeDriver, IEDriver, or SafariDriver, etc.

```
# We follow this approach as it can work with any browser instance.
WebDriver driver = new FirefoxDriver();
# This approach is right too but will work only the Firefox.
FirefoxDriver driver = new FirefoxDriver();
```

How Do You Pass Credentials To An Authentication Popup In Selenium?
We combine the username and password strings using the colon separator and stuff them between the "http://" and the site URL. See the below example.
http://userid:passcode@somesite.com
e.g. http://userid:passcode@somesite.com

What Are The Different Exceptions Available In Selenium?
Like many programming languages, Selenium also provides exception handling. The standard exceptions in Selenium are as follows.
- **TimeoutException:** This occurs if a command doesn't finish within the specified duration.
- **NoSuchElementException:** This occurs if the web element with the specified attributes is not present on the page.
- **ElementNotVisibleException:** This occurs if the element is not visible but still there inside the DOM.
- **StaleElementException:** This occurs in the absence of an element that either got deleted or detached from the DOM.

What Do You Know About An Exception Test In Selenium?
An exception test is a special exception that occurs in a test class.
Suppose, we have created a test case that can throw an exception.
In this case, the @Test annotation can help us specify the exception that could occur.
Check out from the below example.

```
@Test(actualException = ElementNotVisibleException.class)
```

How Is An Assert Different From Verify?
- **Assert:** It allows us to verify the result of an expression or an operation. If the "assert" fails, then it will abort the test execution and continues with the next case.

-
- **Verify:** It also operates the same as the assert does. However, if the "verify" fails, then it won't abort the test instead continues with the next step.

What Difference Do You Make Between Soft Vs. Hard Assert In Selenium?

- **Soft Assert:** It aggregates the errors that occurred during the test execution. If such an assert fails, the control jumps to the next step.
- **Hard Assert:** It immediately responds with an AssertException and breaks the current test. After that, the next case in the sequence gets executed.
-

What Are The Different Waits Available In WebDriver?

In Selenium Webdriver, the following three types of wait mechanisms are available.

Static Wait – Thread.sleep(15000);

- **Implicit Wait** –
- **Explicit Wait** –
- **Fluent Wait** –

What Is Web Driver Implicit Wait?

Implicit Wait: It is a wait timeout which applies to a Webdriver instance. It implies that all actions of this instance will timeout only after waiting for a duration specified by the implicit wait.

```
WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
e1
e2
e3

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
e4 e5 e6

driver.manage().timeouts().implicitlyWait(0, TimeUnit.SECONDS);
e7 e8 e9
```

What Is Web Driver Explicit Wait?

Explicit Wait: It is an exclusive timeout method that works by adding code to delay the execution until a specific condition arises. It is more customizable in terms that we can set it up to wait for any suitable situation. Usually, we use a few of the pre-built Expected Conditions to wait for elements to become clickable, visible, invisible, etc.

```
WebDriver driver = new ChromeDriver();
driver.get("http://target_page_url");
WebElement dynamicElement = (new WebDriverWait(driver, 15))
 .until(ExpectedConditions.presenceOfElementLocated(By.id("dynamicElement"
)));
```

What Is The Command To Enter Text In The HTML Text Box Using Selenium?

We can do so by using the sendKeys() method.

```
WebDriver webdriver = new FirefoxDriver();
webdriver.get("https://www.google.com");
webdriver.findElement(By.xpath("<<xpath expr>>")).sendKeys("Selenium
Interview Questions");
```

How To Enter Text In The HTML Text Box Without Invoking The SendKeys()?

There is a Selenium JavascriptExecutor class that provides methods to perform actions on the HTML elements.

```
// Set up the JS object
JavascriptExecutor js = (JavascriptExecutor)driver;
// Issue command to enter the text
js.executeScript("document.getElementById('textbox').value = 'Some
Text';");
```

What Is The Method To Read The JavaScript Variable Using Selenium WebDriver?

Again, we can utilize the JavascriptExecutor class to read the value of a JS variable. See the below code.

```
// Set up the JavaScript object
JavascriptExecutor jscript = (JavascriptExecutor) webdriver;
// Read the site title
String strTitle = (String)jscript.executeScript("return document.title");
System.out.println("Webpage Title: " + strTitle);
```

What Is The Command To Reset/Clear The HTML Text Box In Selenium Webdriver?

Selenium provides the clear() function to reset the value inside the text element.

```
WebDriver webdriver = new FirefoxDriver();
webdriver.get("https://www.google.com");
webdriver.findElement(By.xpath("<<xpath expr>>")).sendKeys("Selenium
Interview Questions");

webdriver.findElement(By.xpath("<<xpath expr>>")).clear();
```

What Is The Command To Get The Value Of A Text Box In Selenium Webdriver?

Selenium provides the getText() function to read the value inside the text element.

```
 @Test
 public void readText() {
    System.setProperty("webdriver.chrome.driver", "<<chromedriver.exe>>");
    WebDriver webdriver = new ChromeDriver();
    webdriver.get("https://www.google.com");
    String strText = webdriver.findElement(By.xpath("<<xpath
expr>>")).getText();
    System.out.println("Text element contains: " + strText);
 }
```

```
};
```

What Is The Command To Get The Attribute Value In Selenium Webdriver?
Selenium provides the getAttribute(value) function to read the value inside the text element.

```
String strValue =
webdriver.findElement(By.name("<<attrname>>")).getAttribute("value");
System.out.println("Attribute value is: " + strValue);
```

What Is The Command To Click On A Hyperlink In Selenium Webdriver?
Selenium provides the click() function to click on a HTML link.

```
webdriver.findElement(By.linkText("Selenium Interview
Questions")).click();
```

What Is The Command To Submit The HTML Form In Selenium Webdriver?
Selenium provides the submit() function to finalize a HTML form.

```
webdriver.findElement(By.id("<<htmlform>>")).submit();
```

However, we can also achieve the same effect by calling the click() method.

What Is The Command To Press Enter Inside The HTML Text Box Using Selenium Webdriver?
Selenium provides Enum Key macros to simulate the enter action.

```
webdriver.findElement(By.xpath("<<xpath>>")).sendKeys(Keys.ENTER);
```

What Is The Selenium Command To Delay Test Execution For 10 Seconds?
In Java, we can use the following method to halt the execution for a specified no. of milliseconds.

```
java.lang.Thread.sleep(long milliseconds)
```

To halt for 10 seconds, we can issue the following command:

```
Thread.sleep(10000)
```

Is It Mandatory To Prefix The URL With HTTP Or HTTPS While Calling The Web Driver's Get() Method?
Yes, if the URL doesn't contain the HTTP prefix, then the program will throw an exception.

Hence, it is mandatory to pass the HTTP or HTTPS protocol while calling the web driver's get() method.

What Is The Other Method Which Gives The Same Effect As We Get From The Web Driver's Get()?

Selenium provides the "navigate.to(link)" method. It does the same thing as we achieve from the get() call.

What Is The Principal Difference Between "GET" And "NAVIGATE" Methods?
Get method makes a page to load or extracts its source or parse the full text. On the contrary, the navigate method tracks the history and can perform operations like refresh, back, and forward.

For example – We like to move forward, execute some functionality and then move back to the home page.
We can achieve this by calling the Selenium's **navigate()** API.
- The **driver.get()** method waits until the page finish loading.
- The **driver.navigate()** will only redirect and return immediately.

How Can I Move Back And Forth In A Browser Using Selenium?
Selenium provides the following methods for moving back and forth in a browser.
1) navigate().forward() – to move to the next web page as per the browser's history
2) navigate().back() – to move back to the previous page as per the browser's history
3) navigate().refresh() – to reload the current page
4) navigate().to("URL") – to start a new browser window and opening up the specified link

What Is The Selenium Command To Fetch The Current Page URL?
To retrieve the current page URL, we can call the getCurrentURL() function.
```
webdriver.getCurrentUrl();
```

What Is The Selenium Command To Set The Browser Maximized?
We can maximize the browser window by calling Selenium's maximize() method.
It sets the current window in the maximized state.
```
driver.manage().window().maximize();
```

What Is The Selenium Command To Delete Session Cookies?
To delete session cookies, we can invoke the deleteAllCookies() method.
```
webdriver.manage().deleteAllCookies();
```
State The Difference Between Web Driver's GetWindowHandle()

And GetWindowHandles() Methods?

webdriver.getWindowHandle() – It gets the handle of the active web page.
webdriver.getWindowHandles() – It gets the list of handles for all the pages opened at a time.

PART – 3:

State The Difference Between Web Driver's Close() And Quit() Methods?
These two methods perform the same task, i.e., the closing of the browser. However, there is a slight difference.
webdriver.close(): It closes the active WebDriver instance.
webdriver.quit(): It closes all the WebDriver instances opened at a time.
State the difference between Web driver's findElement() and findElements() methods?
Both these methods traverse the DOM looking for the target web element. However, there are some basic differences between them.
1. The findElement() method returns the first WebElement matching the locator, whereas the findElements() fetches all passing the locator criteria.

```
// Syntax of findElement()-
WebElement item = webdriver.findElement(By.id("<<ID value>>"));
// Syntax of findElements()-
List <WebElement> items = webdriver.findElements(By.id("<<ID value>>"));
```

2. Another difference is if no element is matched, then the findElement() raises NoSuchElementException whereas findElements() returns an empty list.
How Do You Check If An Object Is Present On Multiple Pages?
We can use the isElementPresent() command to verify the object on all pages.

```
assertTrue(selenium.isElementPresent(locator));
```

How Do You Check For The Presence Of A Web Element After The Successful Page Load?
We can verify the presence of a web element with the following code. While using the below function, do supply some timeout value (in seconds) to check the element in a regular interval.

```
public void checkIfElementPresent(String element, int timeout) throws
Exception {
    for (int sec = 0;; sec++) {
        if (sec >= timeout)
            fail("Timeout! Couldn't locate element." + element);
        try {
            if (selenium.isElementPresent(element))
```

```
                break;
            } catch (Exception ex) {
            }
            Thread.sleep(1000);
        }
}
```

How To Handle Web-Based Alerts/Pop-Ups In Selenium?

WebDriver exposes the following APIs to handle such popups.

- **Dismiss():** It handles the alert by simulating the Cancel button.
- **Accept():** It handles the alert window by simulating the Okay button.
- **GetText():** You may call it to find out the text shown by the alert.
- **SendKeys():** This method simulates keystrokes in the alert window.

How To Handle Windows-Based Alerts/Pop-Ups In Selenium?

Handling a window-based pop-up is not straight-forward. Selenium only supports web applications and doesn't provide a way to automate Windows-based applications. However, the following approaches can help.

1 Use the Robot class (Java-based) utility to simulate the keyboard and mouse actions. That's how you can handle the window based pop.
2 The KeyPress and KeyRelease methods simulate the user pressing and releasing a specific key on the keyboard.

How To Handle Multiple Popup Windows In Selenium?

Selenium provides the **getWindowHandles()** method, which returns the handles for all open popups.

We can store them into a **<String>** variable and convert it into an array. After that, we can traverse the array and navigate to a specific window by using the below code.

```
driver.switchTo().window(ArrayIndex);
```

Alternatively, we can use the Java Iterator class to iterate through the list of handles. Find below is the code to handle multiple windows using Selenium.

```
String hMyWindow = driver.getWindowHandle();
WebDriver popup = null;
Iterator<String> hWindows = driver.getWindowHandles();
while(hWindows.hasNext()) {
 String hWindow = hWindows.next();
 popup = driver.switchTo().window(hWindow);
 if (popup.getTitle().equals("HandlingMultipleWindows") {
 break;
 }
}
```

How Can You Access A Database From Selenium?

Selenium doesn't have a direct API to access a database. Hence, we can look for its support in the programming language we choose to work with Selenium.

For illustration purposes, we used Java with Selenium.

Java provides the Connection class to initiate a connection with the database. It has a getConnection() method that we need to call. For this, we'll make a Connection Object. It'll manage the connection to the database.

Please note: An application could have one or more connections opened to a database or different databases.

Here is a short overview of the steps to be performed.

- Firstly, we establish a connection with the database.
- After that, we call the **DriverManager.getConnection()** method.

- This method accepts a string pointing to the database URL.
- The DriverManager class then attempts to find a driver to access the database URL.
- After it finds a suitable driver, the call to the getConnection() method succeeds.

Syntax:

```
String url = "jdbc: odbc: makeConnection";

Connection con = DriverManager.getConnection(url, "userID", "password");
```

How To Handle AJAX Controls Using Selenium?

Let's understand the handling of AJAX with an example.

Consider the Google search text box, which is an Ajax control. Whenever we write some text into the box, it shows up a list of auto-suggested values. To automate this type of element, we need to grab the above list in a string as soon as the box receives input. After that, we can split and take the values one by one.

How To Work With AJAX Controls In WebDriver?

AJAX is an acronym for Asynchronous JavaScript and XML. It is independent of the opening and closing tags required for creating valid XML.

Sometimes, the WebDriver itself manages to work with the Ajax controls and actions. However, if it doesn't succeed, then try out the below code.

```
//Waiting for Ajax Control

WebElement AjaxCtrl = (new WebDriverWait(driver,
10)).until(ExpectedConditions.presenceOfElementLocated(By.
&lt;locatorType&gt;("&lt;locator Value&gt;")));
```

What Is A Page Object In Selenium WebDriver?

First of all, both these terms belong to the Page Object Model (POM), a design pattern in Selenium. Let's now see how are they different from each other.

Page Object is a class in POM corresponding to a web page. It captures the functionality as functions and objects as members.

```
public class LogInPage
{
    private WebElement user;
    private WebElement pass;

    public LogInPage() {
    }

    public void findObjects() {
        user = browser.findElement(By.id("userName"));
        pass = browser.findElement(By.id("password"));
    }

    public void processLogIn() {
        user.sendKeys("john");
        pass.sendKeys("password");
    }
}
```

What Is A Page Factory In Selenium WebDriver?

Page Factory is a method to set up the web elements within the page object.

```
public class LogInPage
```

```
{
    @FindBy(id="userName")
    private WebElement user;

    @FindBy(id="password")
    private WebElement pass;

    public LogInPage() {
        PageFactory.initElements(browser, this); // Setup the members as
browser.findElement()
    }

    public void processLogIn() {
        user.sendKeys("john");
        pass.sendKeys("password");
    }
}
```

What Are User Extensions, And How Do You Create Them?

User extensions are a set of functions written in JavaScript. They are present in a separate known as the extension file. Selenium IDE or Selenium RC access it to activate the extensions.

Selenium's core has a JavaScript codebase. So, we can also use it to create the extension.

The extension has a specific format, as given below.

```
// sample
Selenium.prototype.doFunctionName = function(){

}
```

The function name begins with a "do" prefix. It signals Selenium to interpret this function as a command.

It means we can call the above function inside any of our steps.

What Is The Right Way To Capture A Screenshot In Selenium?

Sometimes, an image than a trace log can help us identify the right reason for an error. The code in the below example will capture the image and store it in a file.

```
Import org.apache.commons.io.FileUtils;
WebDriver ins = new ChromeDriver();
ins.get("http://www.google.com/");

File screen = ((TakesScreenshot)ins).getScreenshotAs(OutputType.FILE);
// Now you can do whatever you need to do with it, for example copy
somewhere

FileUtils.copyFile(screen, new File("c:\tmp\myscreen.png"));
```

How To Simulate Mouse Over Action On A Submenu Option Under A Header Menu?

Using the Actions object, you can first move the menu title, and then proceed to the popup menu item and click it. Don't at all miss to invoke the "actions.Perform()" at the end. Check out the below Java code:

```
Actions acts = new Actions(driver);
WebElement menuHoverLink = driver.findElement(By.linkText("Menu
heading"));
acts.moveToElement(menuHoverLink);
WebElement subLink = driver.findElement(By.cssSelector("#headerMenu
.subLink"));
```

```
acts.moveToElement(subLink);
acts.click();
acts.perform();
```

How To Resolve The SSL Certificate Issue (Secured Connection Error) In Firefox With WebDriver?

There can be many reasons for the secured connection error. It could be because of the following:

- While a site is getting developed, it may not have the right SSL certificate.
- The site may be using a self-signed certificate.
- The SSL may not have configured appropriately at the server end.

However, you still want to test the site's standard functionality using Selenium. Then, the idea is to switch off the SSL setting and ignore the SSL error.

Check out the below code to disable the SSL in Selenium.

```
FirefoxProfile ssl = new FirefoxProfile();

ssl.setAcceptUntrustedCertificates(true);

ssl.setAssumeUntrustedCertificateIssuer(false);

WebDriver ins = new FirefoxDriver(ssl);
```

How To Resolve The SSL Certificate Issue In IE?

In the internet explorer, it is somewhat trivial to ignore the SSL error. Please add the below line of code and safely skip the SSL issue.

```
// Copy the below code after opening the browser.

driver.navigate().to("javascript:document.getElementById('overridelink').click()");
```

How To Handle A Proxy Using Selenium In Java?

Selenium implements a PROXY class to configure the proxy. See below example:

```
String setPROXY = "10.0.0.10:8080";

org.openqa.selenium.Proxy allowProxy = new.org.openqa.selenium.Proxy();
allowProxy.setHTTPProxy(setPROXY)
          .setFtpProxy(setPROXY)
          .setSslProxy(setPROXY);
DesiredCapabilities allowCap = new DesiredCapabilities();
allowCap.setCapability(CapabilityType.PROXY, allowProxy);
WebDriver driver = new FirefoxDriver(allowCap);
```

How To Handle A Proxy Using Selenium In Python?

```
from selenium import webdriver

PROXY = "10.0.0.10:8080" # IP:PORT or HOST:PORT

chrome_opt = webdriver.ChromeOptions()
chrome_opt.add_argument('--proxy-server=%s' % PROXY)

chrome = webdriver.Chrome(options=chrome_opt)
chrome.get("<< target URL >>")
```

What Are TestNG Annotations Frequently Used With Selenium?

TestNG annotations prioritize the calling of a test method over others. Here are the ones to use with Selenium:

- @BeforeSuite – to run before all tests.
- @AfterSuite – to run only once after all tests.
- @BeforeClass – to run only once before the first test method.
- @AfterClass – to run only once after all the test methods of the current class finish execution.
- @BeforeTest – to run before any test method inside the “Test” tag.
- @AfterTest – to run after any test method inside the “Test” tag.

What Do You Know About TestNG @Parameters?

In TestNG, the “@Parameters” is a keyword that allows the arguments to pass to “@Test” methods.

Please refer to this **TestNG tutorial** to learn more about parameters.

What Is Data Provider In TestNG?

The data provider is a TestNG annotation. It allows you to pass parameters like a property file or a database to a test method.

What Is Meant By Grouping In TestNG?

It is an innovative TestNG feature that didn’t exist in the JUnit. You can assign methods with proper context and refine groupings of test methods. You can not only link methods to groups but also tell groups to include other groups.

How To Associate A Single Test To Multiple Groups In TestNG?

TestNG framework allows multiple tests to run by using the test group feature.

We can associate a single test to multiple groups, as shown in the below example.

```
@Test(groups = {"regression-testing", "smoke-testing"})
```

Is TestNG Capable Of Running Multiple Suites?

Yes, we can run multiple testNG suites in the following manner:

```
<suite name="SuperSuite">
  <suite-files>
    <suite-file path="subSuite1.xml" />
    <suite-file path="subSuite2.xml" />
    ...
  </suite-files>
</suite>
```

We can also run various suites in parallel by using an Ant task.

Selenium Interview Questions [General Questions]

What Is A Framework? What Are The Different Types Of Frameworks Available?

A framework is a charter of rules and best practices to solve a problem systematically. There are various types of automation frameworks.

Some of the most common ones are as follows:

- Data-Driven Testing Framework
- Keyword Driven Testing Framework
- Hybrid Testing Framework
- Behavioural Driven Framework

What Are The Open-Source Frameworks (OSF) Does Selenium Support?

Selenium can integrate with the following Open Source frameworks.

- JUnit
- TestNG
- Maven
- FitNesse
- Xebium

What Is The Principal Difference Between A Data-Driven Framework & A Keyword Driven Framework?

Data-driven framework (DDF)

- In a Data-driven framework, the test case logic is the part of test scripts.
- It advises keeping the input data separate.
- Usually, the test cases receive the data from the external files (XLS or CSV) and store them into variables. Later during execution, the variables serve as input as well as verification points.

Keyword-driven framework (KDF)

- In KDF, there happens to be a table of keywords. Every keyword either maps to an action or an object.
- The actions reflect the product functionality, and the objects represent its resources such as web elements, command or the DB URL, etc.
- They remain detached from the test automation suite. Hence, the creation of the tests can continue with or without the AUT (application under test).
- The keyword-driven tests cover the full functionality of the application under test.

What Is The Difference Between Hybrid And Data-Driven Framework?

A hybrid framework is a combination of both data-driven and keyword-driven Selenium frameworks.

A data-driven framework works on the concept of separating data from the tests. It solely depends on the test input data.

What Are The Two Most Common Practices For Automation Testing?

The two most frequently used practices for automation are as follows:

- Test-Driven Development (TDD) introduced in 2003
- Behavior Driven Development (BDD) got first heard in 2009

What Is Test-Driven Development (TDD) Framework?

TDD a.k.a. test-driven design is a software development paradigm that proposes to conduct unit testing frequently on the source code. It recommends to write tests, monitor, and refactor the code on failure.

The concept of TDD came to light while a few XP (Extreme Programming) developers conceived it.

It intends to prepare tests to check if the code breaks or not. After each test case failure, the developer should fix and re-run the test to verify the same. This process should repeat until all units function as per the design.

What Is Behavior Driven Development (BDD) Framework?

BDD follows most of the principles of TDD but replaces its unit-centric approach with a domain-centric design.

It intends to bring in inputs not only from the Dev or QA but an array of stakeholders such as Product Owners, Technical Support, Managers, and Customers.

The goal is to identify and automate appropriate tests that reflect the behavior sought by the principal stakeholders.

What Are The Main Traits Of A Software Test Automation Framework?

A test automation framework should have the following features.

- Flexible in the programming language selection
- Support of keywords and actions
- Provision of data sources for input
- Allow test case creation and modification
- Define test case priority
- Manual or automated execution
- Maintain test results history
- Generate test metrics such as test vs. code coverage
- Report creation

- CI tool integration such as Jenkins
- Cross-browser and cross-platform support

What Type Of Test Framework Did You Create Using Selenium?

While replying to such questions, stay focused, and keep your answer short and crisp. You can start by telling about the different components in your framework and then explain them one by one.

Here is an illustration for your help.

- I worked on a framework built on top of the Page Factory framework.
- I've created a page class for every web page in my application. It keeps the objects and the handler functions.
- Every page class has a followup test class where I create tests for related use cases.
- I used separate packages to host the pages and their test classes. It's a best practice to do that way.
- The framework also had a lib package for utility and some standard wrapper functions over Selenium APIs.
- Java is ca ore programming language used for this project. It was primarily because the team had previous Java experience. Also, we could utilize the TestNG annotations and report features.
- Most test cases are data-driven. They require input from the external data source. So, I used Java property/POI class to read from the CSV/XLS files.
- We used the TestNG group feature for labeling test cases as P1, P2, and P3.
- The Log4J library provided the necessary support for tracing in our project.
- Instead of using the TestNG reporting, we preferred the Extent report. It has more graphical options and gives an in-depth analysis of the results.
- We built the framework with the help of Maven. Also, Jenkins provided support for automated build and execution.
- Bitbucket allowed us to manage our source code using git repositories.

What Are Challenges Have You Faced With Selenium? And How Did You Overcome Them?

Here are some of the problems that testers usually face while doing automation with Selenium.

- **Wrong implementation:** I used the **page object model** but had it implemented incorrectly. My classes were focussing on the web elements rather than they should have resembled the user actions.
- **Duplicate code:** The project had many category pages. Each category had a different search function instead of handling them at a central place.
- **Ineffective use of wait:** I used implicit wait with a fixed timeout. But some pages were timing out due to higher load time. I had to adopt the Fluent wait (with a variable timeout) to overcome this problem.
- **Improper error handling:** It was getting hard to debug the cause of a failed test. At some places, the {try-catch} blocks were missing, and hence, cases were skipping w/o giving a proper reason. Therefore, I had to refactor the code by adding asserts and exception handling.
- **Inconsistent XPath:** Most of the locators were using the XPath method. And the developers kept them changed while fixing new defects. I called up a discussion with them and agreed to have a fixed XPath or an ID for the web elements.

- **Performance & Localization:** We were using the flat files (CSV) initially to feed data to test cases. However, it had us failed in testing localization as well as beaten us on the performance. Ee migrated all of our test data to MySQL and fixed both issues.
- **Monolithic tests:** Earlier tests weren't using the labeling. Honestly, there wasn't a way to do it. Hence, we integrated our test suite with TestNG and got away with this limitation. Now, we have many test groups like features-based (F1, F2, F3…), priority-based (P1, P2, P3).

What Benefits Does TestNG Have Over The JUnit Framework?

TestNG vs. JUnit – The Benefits

1 In JUnit, we start by declaring the @BeforeClass and @AfterClass. However, there isn't such a constraint in TestNG.
2 TestNG allows adding setUp/tearDown routines, which JUnit doesn't.
3 -@BeforeSuite & AfterSuite, @BeforeTest & AfterTest, @BeforeGroup & AfterGroup
4 TestNG doesn't enforce a class to extend.
5 Method names aren't mandatory in TestNG, which are in JUnit.
6 We can make a case depend on another in TestNG, whereas in JUnit, it's not possible.
7 TestNG allows grouping of tests, whereas JUnit doesn't. Executing a group will run all tests under it. For example, if we have a no. of cases distributed in two groups, namely the Sanity and Regression. If the requirement is to run the "Sanity" tests, then we can configure TestNG to execute the tests under the "Sanity" group. TestNG will execute all cases associated with the "Sanity" group.

The parallelization of test cases is another crucial feature of TestNG.