Selenium has been the leading skill for automation testing from last many years. And yet there is even more demand for this technology. Not only browser automation but the companies are also using it for web scraping and data mining. Hence, we bring you the real Selenium interview questions to fill the gap between your knowledge and the interviewer's expectations.

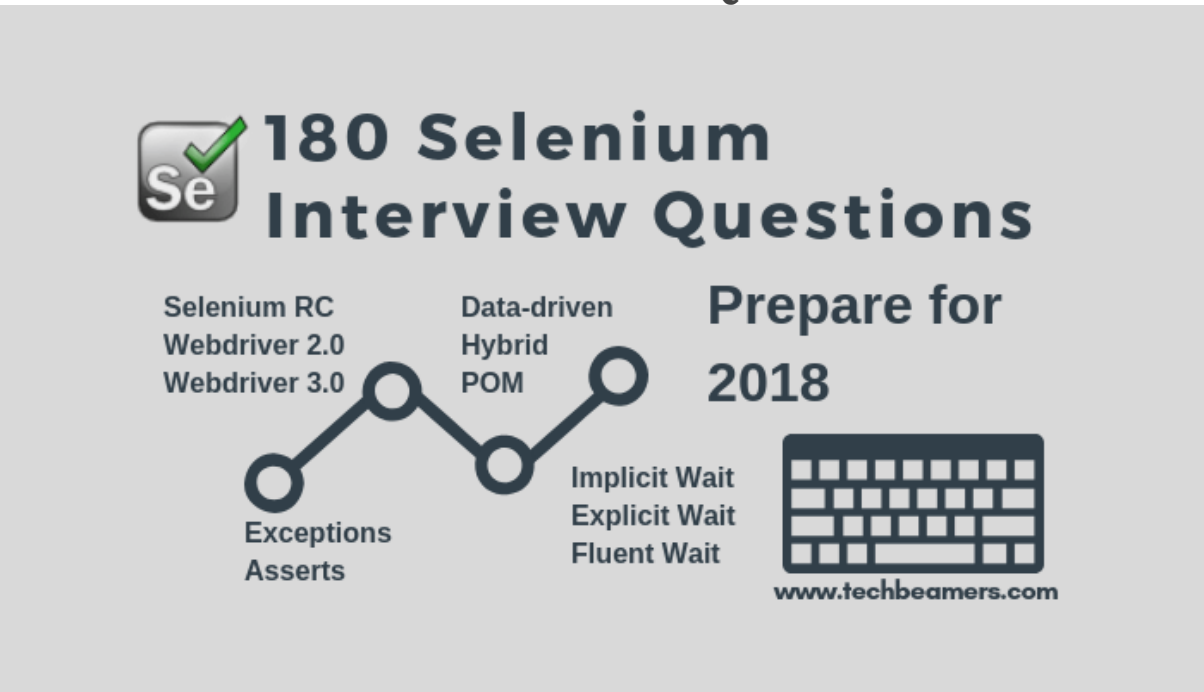
The primary reason for Selenium popularity is its ability to integrate with mainstream languages like Java, Python, and C-Sharp. If you are comfortable with any of these technologies, then it would be a lot easier to learn Selenium.

Many online classes are providing Selenium training to meet its increasing demand in the job market. It has become the must-have skill for automation test engineers. So, we recommend you carefully read all these Selenium interview questions as these have everything you should learn.

If you have a thorough command of Selenium, then proceed directly to the question/answer section. However, if you like to build Selenium skill, then we highly recommend the following two posts.

- **Selenium Webdriver Tutorial**
- **A Quick Selenium Quiz**

## 180 Real Selenium Interview Questions For 2018




## Q-1. What Is A Software Test Automation Framework?

A test automation framework is a system of rules to guide a Software tester in automating the functional flows of a product or service. It intends to bring more efficiency to the Software testing process.

The testing framework lays down the following rules or best practices.

- Flexibility in the selection of a programming language
- Generation of keywords and actions
- Definition of input sources (internal or external)
- Test case creation and design
- Labeling of test case for prioritization
- Manual or scripted execution
- History of test results
- Metrics such as test coverage/ code coverage
- Report generation
- CI tool integration such as Jenkins
- Multiplatform support

A tester may not necessarily follow all these rules while creating test cases. But a modern-day test framework like Selenium can still let him/her write automated scripts with a little effort.

## Q-2. What Do You Understand About Selenium?

Selenium is a modern and continuously evolving test automation framework. It is the most frequently used tool for automated web testing. All leading browsers (Chrome, Firefox, IE) provide its support. It can run on all major platforms (Windows, Linux & Mac OS X). Moreover, the testers can choose from Java, Python or C# to use with Selenium. All of these traits bring high business value and make it a leader amongst the UI automation tools. Additionally, the factors like reusability and user-friendliness are also behind the popularity of this tool.

Since inception in 2004, Selenium had seen two big releases including a few minor ones. Let's step through them one by one.

### *Selenium Remote Control (RC) – 1.0*

Selenium RC came out in early 2004. It included a client API set and a server to automate the browser.

### *Selenium WebDriver – 2.0*

It was a significant release of Selenium during the middle of 2011. It introduced an entirely new API set and replaced the old server with native driver calls. The new API also posed a challenge for automation testers to migrate from old API.

### *Selenium WebDriver – 3.0*

Another major version, Selenium 3.0 saw the light towards the end of Yr. 2016. It enforced the W3C specifications for Webdriver APIs to make them a global standard. From this release, all browser vendors started delivering their custom version of web drivers.

## Q-3. What Are The Pros/Advantages Of Using Selenium For Automation?

Selenium is a time-proven test framework which can dramatically reduce the testing efforts. It equips with many useful features to make a tester's life easier. Moreover, it can align with other tools to bring in more power.

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 Window, Linux, and Mac OSX. We can run same the test script on all platforms.
- **Concurrency:** With Selenium Grid, we can execute thousands of test 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.
- **Free help:** We can get quick technical help 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.
- **Active project:** Active development and bug fixes on the latest project.

## Q-4. What Are The Cons/Limitations Of Using Selenium For 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

## Q-5. What Are The Core Components Selenium Provides?

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

- **Selenium IDE:**

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

- **Selenium RC:**

A Selenium server 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.

**Must Read – Selenium Performance Testing**

- **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.

## Q-6. What Is Selenium Server And How Is It Different From Selenium Hub?

Selenium server is a standalone program which acts as an interface between the browser and the client. However, the Selenium hub works as a proxy for one or more Selenium test nodes.

The combination of a hub and the node(s) represent a Selenium grid. Executing a Selenium server is equivalent to creating a Selenium grid with one Hub and a node on the same machine.

## Q-7. What Are The Type Of Tests You Can Automate With Selenium?

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

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

## Q-8. What Is Your Velocity Of Automating Tests Per Sprint?

There is no simple rule to compute the velocity of test case automation per sprint. The following factors influence how many tests we can automate.

- Our product or feature knowledge
- The complexity of the feature
- The probability of unknowns

However, my average velocity comes around ~(30-50) test cases in a 2-week sprint, i.e., 3-5 cases per day.

## Q-9. What Types Of Automated Testing Frameworks Do You Know?

Mainly, there are six standard types of testing frameworks. Each of these has some benefits as well as a few limitations. So, you must pick the right one based on your test requirements.

1. **Linear Automation Testing Framework:** It is a simple record-play framework.
2. **Modular-Based Testing Framework:** It works by splitting the AUT (Application Under Test) into small units. Each of them gets tested in isolation before running all of them inside a batch script.
3. **Library-Based Testing Framework:** It slices up the application testing into distinct areas having the same objectives/functions/features. After that, a library gets formed including these functions. The test scripts can call the library methods as needed.
4. **Data-driven Testing Framework:** It segregates the test data from the script logic, expecting the testers to store it externally. The test scripts can request the external data source, read and fetch the required data depending on the use case.
5. **Keyword-driven Testing Framework:** It allows keywords to perform actions on the test input. The tester can define them in an object repo where the framework can find and take action.
6. **Hybrid Test Automation Framework:** It is a mixture of the previously mentioned frameworks intended to utilize the pros of some and overcome the limitations of others.

## Q-10. What Type Of Test Framework Either You Used Or Created? Please Explain.

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 core 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 feature.
- 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 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.

## Q-11. What Challenges Have You Faced While Using Selenium And How You Overcame?

Here is an illustration for you to explain the challenges faced 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.
- **Less Abstraction:** The framework missed on abstracting several steps into one. For example – Searching for an item took three actions such as clicking the box, typing the word, and clicking the search.
- **Ineffective use of wait:** I used implicit wait with a fixed timeout. But some pages were timing out due to higher load time. I'd 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 message. Therefore, I'd to refactor the code by adding asserts and exception handling.
- **Inconsistent XPath:** The 90% of the objects were using XPath. And they were inconsistent as the developers kept them changed while fixing new defects. The unfortunate practice was prevailing at both ends, the Dev/QA. I'd to call a discussion with them. And finally, we came to an agreement that either 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. Finally, we migrated all of our test data to MySQL DB and could get fixed both these 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 the TestNG framework and got away with this limitation. Now, we have many test groups like features-based (F1, F2, F3…), priority-based (P1, P2, P3).

## Q-12. How Is A Page Object Different From The Page Factory?

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");
    }
}
```

**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");
    }
}
```

## Q-13. What Do You Know About Selenium IDE?

Selenium IDE is a very popular Firefox 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 recording a workflow, we can also export the steps in various formats.

Shinya Kasatani was the developer who created 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.

## Q-14. 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, 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.

## Q-15. What Are Different Exceptions Available In Selenium?

Like many programming languages, Selenium also provides exception handling. The standard exceptions in Selenium are as follows.

- **TimeoutException:** Occurs if a command doesn't finish within the specified duration.
- **NoSuchElementException:** Occurs if the web element with the specified attributes is not present on the page.
- **ElementNotVisibleException:** Occurs if the element is not visible but still there inside the DOM.
- **StaleElementException:** Occurs in the absence of an element which either got deleted or detached from the DOM.

## Q-16. What Do You Know About An Exception Test In Selenium?

An exception test is a special exception which occurs in a test class.

Suppose, we have a 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)
```

## Q-17. What Do 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.

## Q-18. What Is The Reason To 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.

## Q-19. What Are 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 on them and execute.

## Q-20. What Is The Purpose Of A Hub In Selenium Grid?

A hub is analogous to a server that drives the parallel test execution on different machines.

## Q-21. What Do You Call A Node 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.

## Q-22. 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
```

**Must Read – Selenium Grid Tutorial**

## Q-23. 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 features provided by extras.

- 
- More control over the individual nodes
  - Kill a browser running tests simply 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

## Q-24. 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 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,platfor
m=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.

## Q-25. What Are Different Types Of Web Driver APIs Supported In Selenium 3.0?

Following are the web drivers supported in Selenium 3.0.

**| 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

## Q-26. What Programing Languages Does Selenium Allow To Use?

Following is the list of supported languages by Selenium.

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

## Q-27. What Are The OS/Platforms Does Selenium Support?

Following is the list of supported OS/platforms by Selenium.

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

## Q-28. What Are The Open-Source Frameworks (OSF) Does Selenium Support?

Following is the list of supported Open-source frameworks (OSF) by Selenium.

- JUnit
- TestNG
- Maven
- FitNesse
- Xebium

## Q-29. What Are The Locators Does Selenium Support?

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

## Q-30. What Do You Know About XPath?

XPath is one of the locator strategies Selenium uses to find the 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.

## Q-31. What Difference Do You Make Between The “/” And “//” Used In XPath?

- **Single (forward) Slash “/”:** It represents the absolute path. In this case, the XPath engine navigates the DOM right from the first node.
- **Double (forward) Slash “//”:** It represents the relative path. In this case, the XPath engine searches for the matching element anywhere in the DOM.

## Q-32. What Difference Do You Make Between The Absolute And Relative XPath?

- 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
```

- 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.

```
//input[@id='username']
```

## Q-33. 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.

## Q-34. What Difference Do You Make Between Soft Vs. Hard Assert In Selenium?

- **Soft Assert:** It aggregates the errors 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.

## Q-35. 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 the illustration purpose, 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 steps to be performed.

- Firstly, we'll establish the connection with the database.
- We'll call the **DriverManager.getConnection()** method.
- It accepts a string pointing to the database URL.
- The DriverManager class will attempt to find a driver to access the DB URL.
- After the DriverManager class finds an appropriate driver, it calls the getConnection() method.

**Syntax:**

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

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

## Q-36. 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));
```

## Q-37. 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 code base. 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.

## Q-38. 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);
```

```
    }
}
```

## Q-39. How Will You Start The Selenium Server From Java Code?

```
try {
 int svr_port = RemoteControlConfiguration.DEFAULT_PORT;
 RemoteControlConfiguration cfg = new
RemoteControlConfiguration();
 cfg.setPort(svr_port);
 sel_svr = new SeleniumServer(false, cfg);
 sel_svr.start();
} catch (Exception ex) {
 ex.printStackTrace();
}
```

## Q-40. What Do You Know About The CSS Selector/Locator Strategy In Selenium?

CSS locator/selector strategy allows us to locate elements using the HTML tags, attributes, Id, and Class. In CSS, specific symbols represent the parent/child nodes. A space sign ” ” denotes the direct child while the “>” serves as the relative child.

Below are some CSS selector expressions for the illustration purpose.

```
css=input[name='q']

css=input[id='lst-ib'] or input#lst-ib

css=input[class='lst'] or input.lst
```

CSS supports searching for patterns of id/name/class by using the “contains()” method.

```
css=input[id*='lst-ib')]
```

We can also use RegEx to find elements.

```
css = div:contains("^Python")
```

## Q-41. 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.

## Q-42. How To Handle Multiple Popup Windows In Selenium?

Selenium provides **getWindowHandles()** method which returns the handles for all open popups.
We can store them into a **<String>** variable and convert 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 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;
 }
}
```

## Q-43. 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.

## Q-44. What Are The Benefits Does WebDriver Have 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 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.

## Q-45. 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.

**Learn more about -> Selenium Commands**.

## Q-46. What Difference Do You Make Between The Implicit And Explicit Wait?

There are some inherent differences between the two wait constructs of Selenium Webdriver.

- **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);
```

- **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("dyn
amicElement")));
```

**Must Read – Selenium Explicit Wait** and **Fluent Wait Examples**.

## Q-47. How To Handle JS Alerts/Pop-Ups In Selenium WebDriver?

Usually, we come across following two type of alerts while using the web.

- Windows-based alerts
- Web-based alerts

### *Web-Based Alert Pop-Ups*

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.

### *Windows-Based Alert Pop-Ups*

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 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.

## Q-48. 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 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"));
```

## Q-49. 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);
```

## Q-50. 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()");
```

## Q-51. 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;")));
```

## Q-52. 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();
```

## Q-53. What Is The Difference Between The TDD, BDD And DDD Frameworks?

There are some fundamental differences between the TDD, BDD and DDD frameworks. Let's cut through them one by one.

### *Test Driven Development (TDD)*

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.

### *Behavior Driven Development (BDD)*

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.

### *Domain Driven Development (DDD)*

DDD targets to introduce the business domain concepts into the software architecture. Such a framework could yield the following benefits:

- Deliver a working model matching the expectations of business and IT stakeholders
- More modular, extensible and reduces maintenance cost.
- Highly reusable and can accommodate new business object definitions.

## Q-54. What Is The Principal Difference Between A Data-Driven Framework & A Keyword Driven Framework?

**Data-driven framework (DDF)**

- In 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, a command or the DB URL, etc.
- They remain detached from the test automation suite. Hence, the tests creation can continue with or without the AUT (application under test).
- The keyword-driven tests cover the full functionality of the application under test.

## Q-55. What Benefits Does TestNG Have Over 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.
    1. @BeforeSuite & AfterSuite
    2. @BeforeTest & AfterTest
    3. @BeforeGroup & AfterGroup
3. TestNG doesn't enforce a class to extend.
4. Method names aren't mandatory in TestNG which are in JUnit.
5. We can make a case depend on another in TestNG whereas in JUnit it's not possible.
6. 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.
7. Parallelization of test cases is another crucial feature of TestNG.

## Q-56. 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 this **TestNG tutorial** to learn more about parameters.

## Q-57. 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"})
```

## Q-58. 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.

## Q-59. How Can We Use Selenium RC API Along With Selenium 2.0/3.0?

We can still invoke Selenium 1.0 API (RC) from the Selenium 2.0 code. But there might be a few Selenium 1.0 methods missing.

However, we first need to create Selenium instance from WebDriver and call Selenium functions.

The old methods might slow down the execution due to the API transitioning happening in the background.

## Q-60. 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