Selenium
Interview Questions

Click here to view the live version of the page

Frequently Asked Selenium Interview Questions

Selenium is an open-source tool used for automation testing, which allows you to test web applications across a range of browsers. It was introduced once manual testing started to pose several challenges, and there was a high demand for a method that automated the testing process. This Selenium interview questions blog has a compiled list of most of the questions that are asked during Selenium job interviews. The most-asked Selenium interview questions for freshers and experienced are mentioned below.

Top Selenium Testing Interview Questions

Basic Selenium Interview Questions for Freshers

Selenium Interview Questions for Experienced (2 to 5 Years)

Selenium Interview Questions for 2 Years Experience

Selenium Interview Questions for 3 Years Experience

Selenium Interview Questions for 4 to 5 Years Experience

Selenium Advanced Interview Questions (6 to 10 Years)

Selenium Interview Questions for 6 to 7 Years

Selenium Interview Questions for 8 to 10 Years Experience

Selenium Coding Interview Questions and Answers

Selenium MCQ Interview Questions

Selenium Tricky Interview Questions

Selenium Scenario-Based Interview Questions

Selenium Tester Salary Trends

Selenium Testing Job Trends

Selenium Tester Roles and Responsibilities

Conclusion

Did you know?

- Snack-Driven Efficiency: To boost the test script, the Selenium tester uses the magical power of late-night snacks. More pizza and energy drinks consumed during coding sessions are better for bug-finding abilities.
- Click, Click, Click Maestros: Selenium testers perform with lightning-fast and precise mouse-clicking skills. In bug-bashing sessions, we might hear a rhythmic “click, click, click” symphony, showcasing their prowess in this unique art form.

- Code Whispers: Selenium testers claim to have a telepathic connection with their code. They speak of understanding the emotions and intentions of their scripts just by staring at the screen, creating a bond that resembles a quirky friendship rather than a coding task.

Basic Selenium Interview Questions for Freshers

1. What is Selenium testing?

Selenium testing is a tool used for automating web applications to verify their functionality. It deals with browsers, clicks buttons, fills forms, and checks if the site works correctly.

Enroll today in our Selenium Course to know more about Selenium!

2. Is Selenium 2.0 different from Selenium 3.0? If so, how?

Selenium 3.0 represents an enhanced iteration of Selenium 2.0, introducing numerous enhancements and novel functionalities. A significant distinction lies in the inclusion of the WebDriver API as an integral part of Selenium 3.0, eliminating the need for separate downloads and installations as required in Selenium 2.0.

Moreover, Selenium 3.0 exhibits various notable modifications, encompassing enhanced browser compatibility, improved management of browser plugins and extensions, and fortified security features. Furthermore, this version of Selenium demonstrates heightened compatibility with contemporary web browser editions and operating systems.

3. Mention some of the popular tools used for Automation testing.

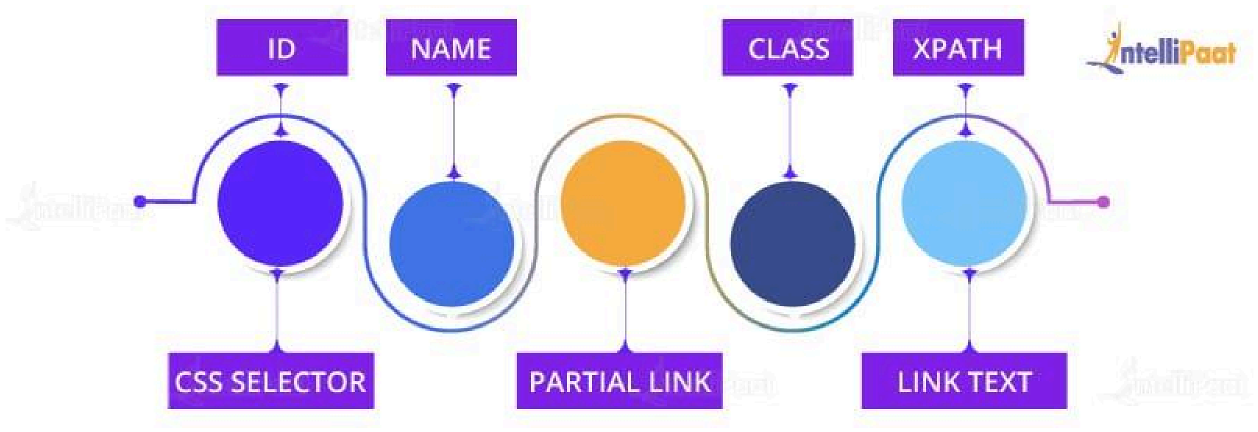
There are several popular tools used for automation testing, including;

- Selenium
- Appium
- TestComplete
- Katalon Studio
- Ranorex
- HP Unified Functional Testing (UFT)
- Apache JMeter, and many more.

These tools help automate the testing process, improve efficiency, and reduce the risk of human error.

4. What is a Locator? How can you find elements in Selenium?

Selenium uses locators to find and match the elements of a web page that it needs to interact with. There are different types of Selenium locators to identify various web elements on a web page:



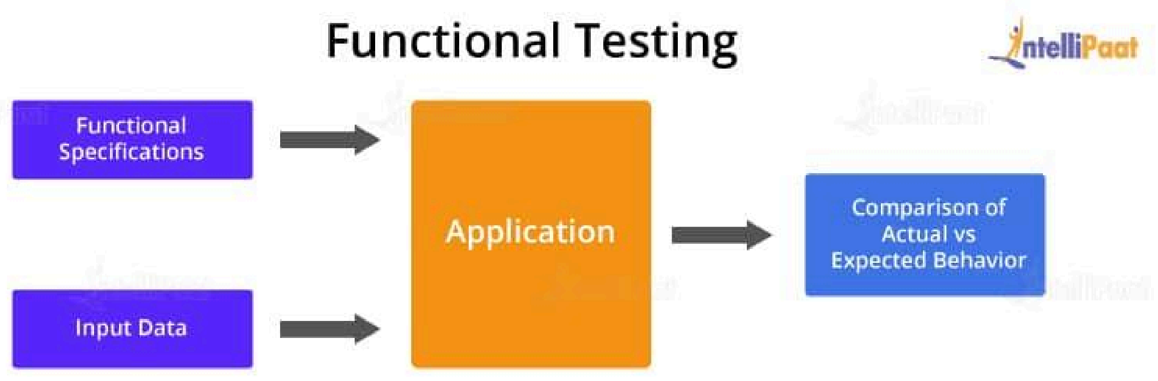
- ID
- Name
- Class
- Partial Link
- XPath
- CSS Selector
- Link Text

5. What are the test types supported by Selenium?

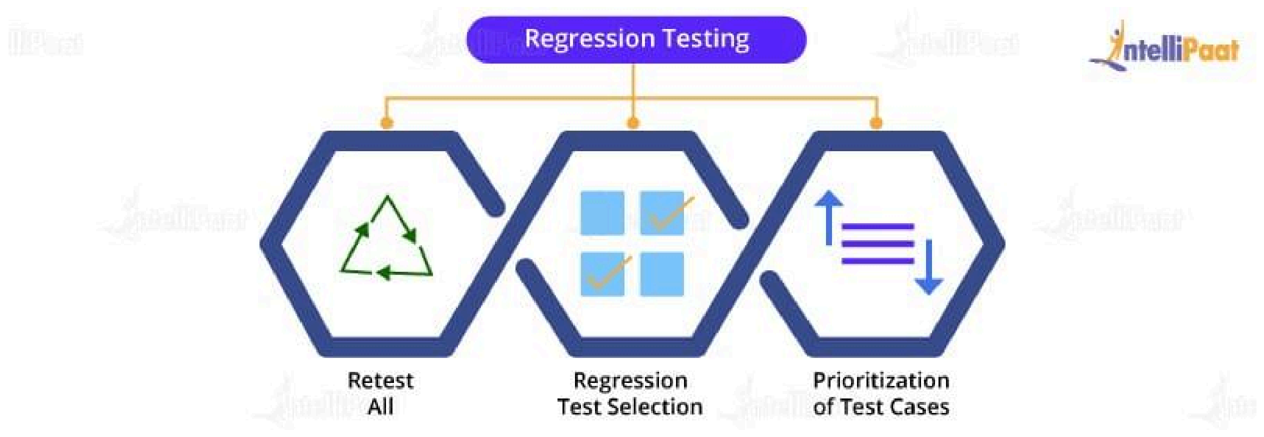
For testing web-based applications, Selenium can be used.

The test types supported by Selenium are:

- Functional testing: It verifies if each function of a software application performs in accordance with specific requirements. This testing primarily involves black-box testing, and it is not concerned with the source code of the application.


Functional Testing
Functional Specifications
Input Data
Application
Comparison of Actual vs Expected Behavior

- Regression testing: It is nothing but a full or partial selection of the already executed test cases to be re-executed to ensure whether the existing functionalities work fine.



6. What is XPath?

While DOM is the recognized standard way for navigating through an HTML element tree, XPath is the navigation tool used to locate a web element based on its XML path.

XML stands for 'Extensible Markup Language' and is used to store, organize, and transport arbitrary data. It stores data in a key-value pair that is very much similar to HTML tags. Both being markup languages and falling under the same umbrella, XPath can be used to locate HTML elements.

The fundamental concept behind locating elements using XPath is traversing between various elements across the entire page and thus enabling a user to find an element with the reference of another element.

7. Explain the difference between single slash and double slash in XPath.

- Single slash (/): Single slash is used to create an XPath with an absolute path. In this case, the XPath would start selection from the document's start node.
- Double slash (//): Double slash is used to create an XPath with a relative path. In this case, the XPath would start selection from anywhere within the document.

Also, check out the blog on Manual testing vs. Automation testing.

8. Why should you use Selenium for test automation?

Selenium should be used for test automation as it:

- Is a free and open-source tool
- Has a large user base and community support
- Has cross-browser compatibility (Firefox, Chrome, Internet Explorer, Safari, etc.)
- Has great platform compatibility (Windows, Mac OS, Linux, etc.)
- Supports multiple programming languages (Java, C#, Ruby, Python, Perl, etc.)
- Has fresh and regular repository developments

- Supports distributed testing

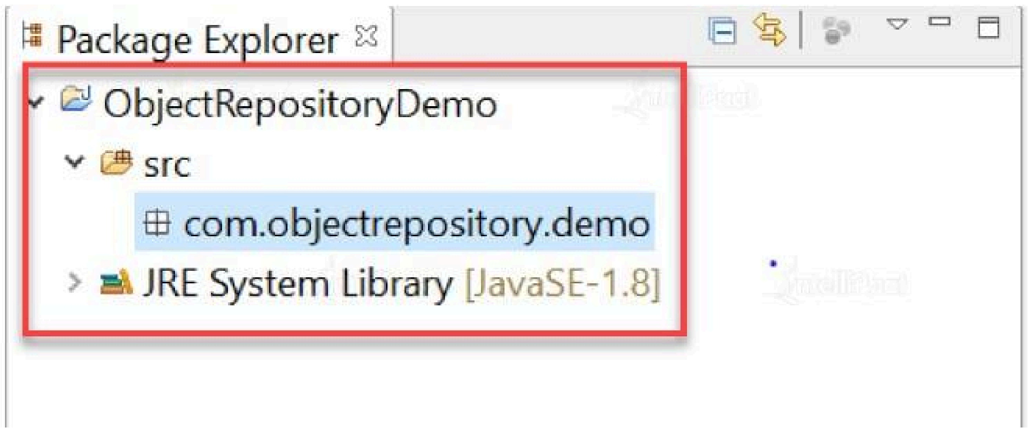
9. Does Selenium have any technical limitations? If so, what are those limitations?

Yes, Selenium has a few limitations:

- Testing of only web applications is possible using Selenium.
- Testing of mobile applications or desktop applications is not possible.
- Captcha and barcode readers cannot be tested using Selenium.
- A third-party tool like TestNG or JUnit should be used to generate reports.
- As Selenium is a free tool, there is no ready vendor support through which users can find various helping communities.
- Prior programming language knowledge is expected from users.

10. What is an object repository?

An object repository allows testers to accumulate web elements of the application under test (AUT), along with their locator values, in one or more centralized locations as restricted to hard-coding them within the test scripts.



11. What is Selenium?

Selenium is a popular open-source software that is used to automate web-based applications. It is a set of multiple software tools, and each tool has a different approach to automated testing.

Selenium has four major components, namely:

- Selenium Integrated Development Environment
- Selenium Remote Control
- Selenium WebDriver
- Selenium Grid

12. What is Selenium WebDriver?

Selenium WebDriver is the most popular component of the Selenium framework. It is a powerful tool that allows you to automate web browsers. With Selenium WebDriver, you can write automation scripts in various programming languages and execute them on different browsers like Chrome, Firefox, Safari, and more.

13. What is the difference between type keys and type commands?

TypeKeys() will trigger JavaScript events, while type() won't. TypesKeys collects different value attributes using JavaScript. Whereas, the type commands imitate an actual user typing.

14. What are the advantages of Selenium?

- Selenium is a purely open-source and portable automation testing tool.
- It supports different languages such as C#, PHP, Java, Perl, Python, JS, and Groovy.
- It also supports different OS, including Windows, Linux, UNIX, and Mac OS.

- It provides powerful methods such as Xpath, DOM, and CSS to locate elements.
- Since it is an open-source tool, developers can customize the code. Also, the developer community is supported by Google.

15. Define automation testing, and list down its advantages.

Automation testing, also known as test automation, involves using tools to automate the testing process, writing and executing test cases without human intervention. It empowers us to develop scripts that can be executed repeatedly and generate comprehensive test reports for the application.

Its advantages are given below:

- It helps with the performance and functional testing of an application.
- It makes the execution of repeated test cases easy.
- It facilitates the concurrent execution of multiple test cases.
- It boosts the accuracy and efficiency of the application by cutting down the chances of human error.
- It efficiently executes tests across an extensive test matrix.
- It saves time and money by reducing the burden of arbitrary tasks.

If you want to become a professional Selenium with Python expert. Learn and master this technology by enrolling in Intellipaat's Selenium with Python Online Training.

16. What are the significant changes/upgrades made to various Selenium versions?

Selenium's first version included only three sets of tools: Selenium IDE, Selenium RC, and Selenium Grid. There was no WebDriver included in the first version. Later, Selenium WebDriver was introduced and hence included in Selenium V2. However, as WebDriver got included, the use of Selenium RC was discouraged with time and has not been much in use since. Selenium 3 is in use. There are some newly added Selenium features such as the IDE and WebDriver. Selenium 4 is the latest released version.

17. How many types of WebDriver APIs are available in Selenium?

The following is a list of WebDriver APIs:

- AndroidDriver
- ChromeDriver
- EventFiringWebDriver
- FirefoxDriver

- HTMLUnitDriver
- InternetExplorerDriver
- iPhoneDriver
- iPhoneSimulatorDriver
- RemoteWebDriver

18. What is an exception test in Selenium?

An exception test is a test that looks forward to an exception being thrown inside a test class. It anticipates the @Test annotation followed by the expected exception name. For example, @Test(expectedException = NoSuchElementException.class) is an exception test for missing elements in Selenium.

Note: Keep in mind the syntax, where the exception is suffixed with .class.

Selenium Interview Questions for Experienced (2 to 5 Years)

19. What is TestNG in Selenium?

TestNG is a popular testing framework that is widely used in Selenium. It is used to manage and run test cases in a more efficient and organized manner. TestNG provides features like grouping, parallel testing, and reporting.

20. What is POM (Page Object Model)? What are its advantages?

The page object model is a design pattern used to create object repositories for the web UI elements. Every web page of an application has a corresponding page class that is responsible for locating the web elements and performing actions on them.

Its advantages are as follows:

- It provides support to separate operations and flows on the UI from verification, hence improving code readability.
- As the object repository is independent of test cases, multiple tests can use the same object repository.
- It increases the reusability of the code

Selenium Interview Questions for 2 Years Experience

21. What are the different types of annotations used in Selenium? Explain the JUnit annotation linked to Selenium.

In Java, a special form of syntactic metadata can be added to the Java source code, which is known as ‘annotations’. Variables, parameters, packages, methods, and classes are annotated. Some of the JUnit annotations are:

- Test
- Before
- After
- Ignore
- BeforeClass
- AfterClass
- RunWith

JUnit annotations linked to Selenium are:

- @Test: The @Test annotation finds a method to be a test method. When used before a test method, it is mentioned as ‘@Test’; it informs the JUnit framework that the following method is a test method.
- @Before: @Before annotation serves the purpose of identifying the method that should be executed prior to running the test method. Its intended use is to establish and configure the test environment before conducting the actual test.
- @After: The “@After” annotation is utilized as a post-execution method following the execution of the test method. This annotation performs

teardown operations, such as deleting temporary data, restoring default values, cleaning up the test environment, and other relevant tasks.

- @BeforeClass: The @BeforeClass method is used only once before the start of all tests. Basically, this is used to perform cumbersome activities, like connecting to a database.
- @AfterClass: The @AfterClass method is used only once after executing all tests. This is used to carry out clean-up activities, like disconnecting from a database.

Download the Selenium Cheat Sheet and use it whenever required, especially during your interviews.

22. Why do testers choose Selenium over QTP?

Selenium is more widely used than QTP since:

- Selenium is an open-source tool, whereas QTP is a profitable tool.
- Selenium is used specifically for testing web-based applications, while QTP can be used for testing client–server applications too.
- Selenium supports multiple browsers like Firefox, IE, Opera, Safari, etc., and has multiple operating systems compatibility too.
  Selenium-supported OS platforms are Windows, Mac, Linux, etc. On the other hand, QTP is limited to Internet Explorer on Windows.

- Selenium supports multi-programming language compatibility. Languages supported by Selenium are Python, Ruby, Perl, etc. But, QTP supports only VBScript.

You can find more on Selenium by visiting our Selenium Community!

23. What are the four elements that you have to pass in Selenium?

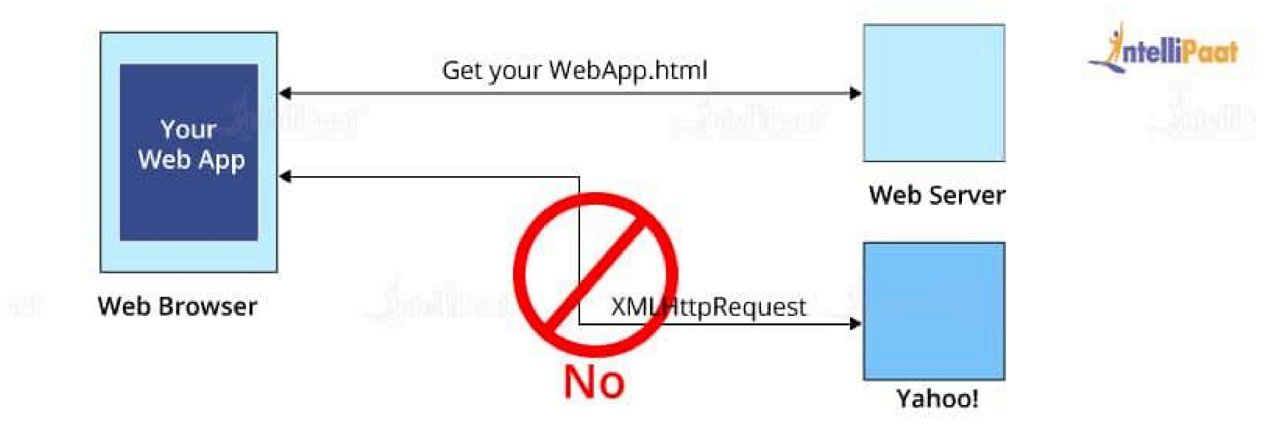
Four parameters that need to be passed in Selenium are:

- Host
- Port number
- Browser
- URL

Selenium Interview Questions for 3 Years Experience

24. What is Same-origin Policy? How can we avoid it?

The 'Same-origin Policy' is introduced for security reasons.



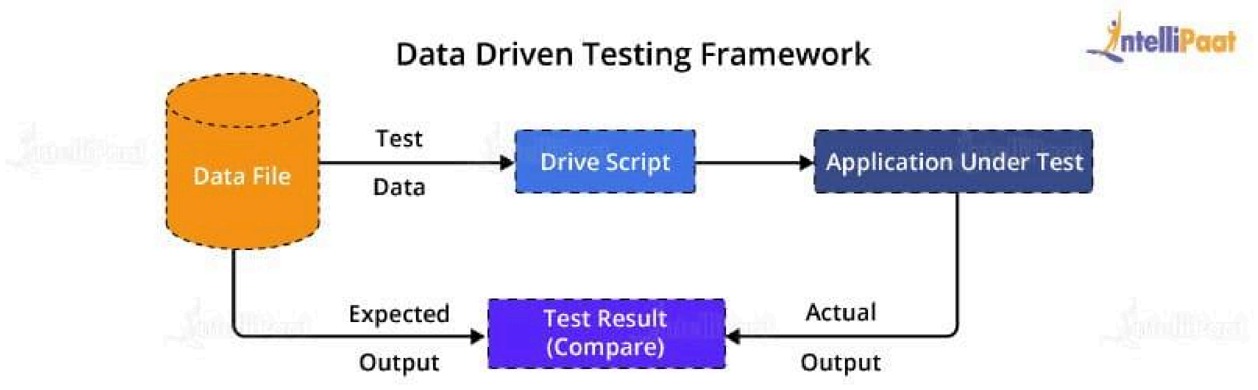
- It ensures that the content of our site will never be accessible by a script from another site.
- As per the policy, any code loaded within the browser can only operate within that website's domain.

To avoid this same-origin policy, the proxy injection method is used. In the proxy injection mode, Selenium Server tricks the browser to be a real HTTP URL, i.e., it acts as a client-configured HTTP proxy, which sits between the browser and the application under test (AUT) and then masks the AUT under a fictional URL.

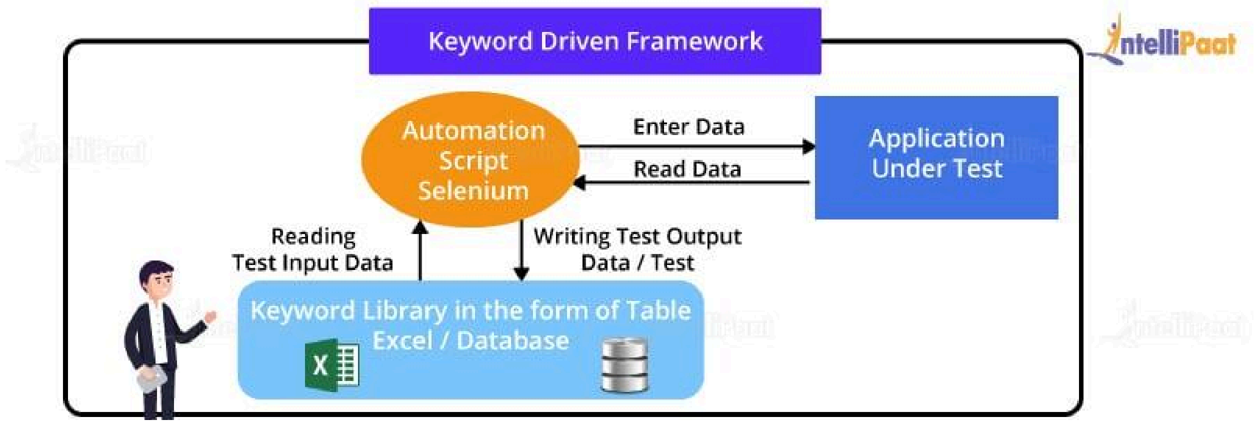
25. What are data-driven framework and keyword-driven framework?

A data-driven framework in Selenium is an approach to separating a 'dataset' from the actual 'test case' (code). This framework is completely dependent on the input test data. The test data is inserted from external sources, such as an Excel file, a

CSV file, or any database. It also allows us to easily control how much data needs to be tested. We can easily increase the number of test parameters by adding more username and password fields to the Excel file (or other sources).



A keyword-driven framework is an extension of the data-driven testing framework in the sense that it not only isolates the test data from the scripts but also keeps the particular section of the code belonging to the test script in an external data file. These sets of code are known as keywords, and hence the framework is so named. Keywords are self-guiding and work based on what actions need to be performed on the application.


Keyword Driven Framework
Automation Script Selenium
Enter Data
Read Data
Application Under Test
Reading Test Input Data
Writing Test Output Data / Test
Keyword Library in the form of Table Excel / Database

26. How will you use Selenium to upload a file?

If the file is on the same machine or on a mapped network drive, it is really straightforward: We have to just type the 'path' of the file in the FileUpload control.

Example:

```
driver = webdriver.Firefox()
element = driver.find_element_by_id("fileUpload")
element.send_keys("C:myfile.txt")
```

Learn more about Selenium from this informative Selenium Tutorial!

27. What is the difference between getwindowhandles() and getwindowhandle()?

- getwindowhandles(): It is used to retrieve the addresses of all currently open browsers. The method returns a Set data type containing the addresses.
- getwindowhandle(): It is used to obtain the current browser's address and its return type is a string data type.

Selenium Interview Questions for 4 to 5 Years Experience

28. What is a Selenium Maven project?

In a Selenium Maven project, developers use Maven as a build tool and Selenium WebDriver to automate browser testing. Maven simplifies the process of managing dependencies and building Java projects, making it easier to configure Selenium WebDriver in the project.

Maven empowers developers to establish a project structure encompassing source code, test code, and resource files while automating the build process. This simplifies project maintenance, distribution, and collaboration for Selenium.

Also, check out the blog on Maven in Selenium.

29. What is a WebElement in Selenium, and how is it used?

A WebElement is an interface in Selenium used to represent a web element on a web page. It is used to interact with the element, such as by clicking a button or entering text into a field.

30. What is Selenese, and what are the three types of Selenese?

Selenese is a set of commands in Selenium used for running a test.

Three types of Selenese are as follows:

- Actions: They are used for performing interactions and operations with the target elements.
- Accessors: They are used for storing values in a variable.
- Assertions: They are used as a checkpoint.

31. If you want to insert a breakpoint in the Selenium IDE, how can you do that?

To insert a breakpoint:

- First, select 'Toggle Breakpoint' by right-clicking on the command in Selenium IDE
- Then, press 'B' on the keyboard and select the command
- The same step should be repeated for deselecting a breakpoint

32. How do you handle alerts in Selenium?

Alerts are pop-up windows that are displayed on a web page. You can handle alerts in Selenium using the Alert interface. The Alert interface provides methods like accept(), dismiss(), and getText() to handle alerts.

33. How do you launch the web browser using WebDriver?

The following syntax can be used to launch the browser corresponding to the system's operating system:

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

Or

```
WebDriver driver = new InternetExplorerDriver();
```

Or

```
WebDriver driver = new ChromeDriver();
```

34. What is the difference between getText() and getAttribute() in Selenium?

getText() returns the visible text of a web element, while getAttribute() returns the value of a specific attribute of the web element.

35. List down some of the technical challenges with Selenium.

- Testing a Windows application: Selenium is just a web-based driver. It does not support Windows-based apps and only supports web apps.
- Testing mobile apps: With the help of Selenium, we can test web apps on any OS and browser that run on desktops. But, we cannot test mobile apps with Selenium because it does not work with OS such as Android and iOS. However, there is an alternative for this, i.e., Appium. It is an open-source automation testing tool that uses the WebDriver protocol to drive native, hybrid, and iOS and Android, which is built specifically for testing mobile apps.
- Limited reporting: It is one of the key challenges. In Selenium, we cannot generate efficient and accurate reports. Accurate reports help developers fix all bugs and errors. We can create reports using TestNG or ExtentReports.
- Handling dynamic elements: With the surge in the use of web apps, the management of dynamic elements should be as much efficient as possible. When a web page loads, the content present on the page

changes depending on the user, location, and other factors. Most of today’s web apps are dynamic in nature for better user experience, e.g., e-commerce websites. In Selenium automation, the handling of dynamic web content is a major challenge. However, Selenium provides an explicit wait feature, where we can set a time interval for the automation testing process to hold the process for the new content to load. Also, another alternative is to utilize the implicit wait feature.

- Handling page load: Some of the web pages in a web app are user-specific. They load elements depending on the user. Also, some elements may be loaded depending on the user’s previous activities. During background processes, the Selenium script might not be able to identify a specific element. To overcome this, we can use explicit waits to provide sufficient time to load and discover the element.
- Handling pop-up windows: Whenever any simple, prompt, or confirmation alert pops up, it is difficult to automate it. Windows-based OS alerts are beyond Selenium’s capabilities as they are part of the OS instead of the browser. However, Selenium WebDriver can utilize multiple windows, and the web-based alerts can easily be handled with the help of the switchTo method. It manages the OS-based pop-ups while keeping the browser running in the background.

36. What is the difference between setSpeed() and sleep() methods?

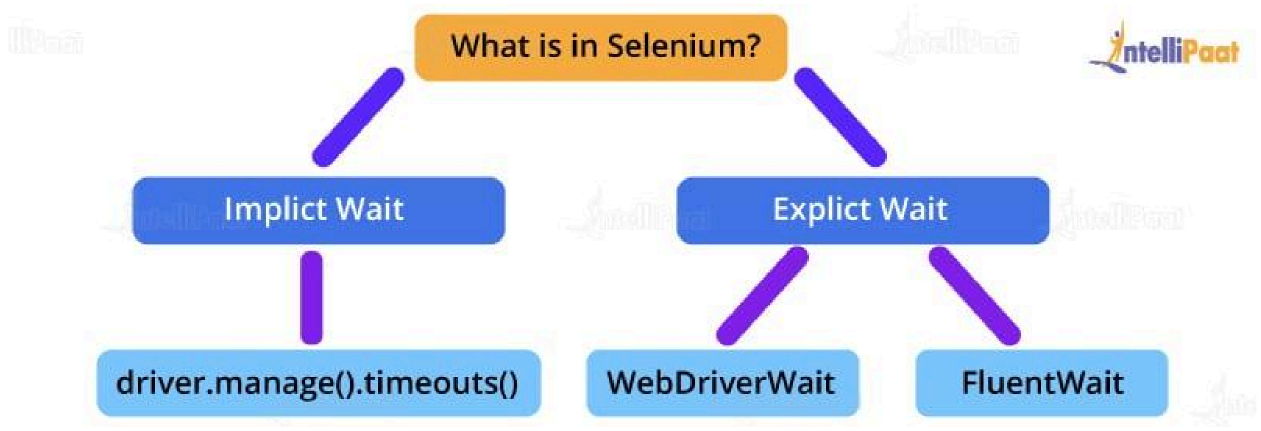
Both setSpeed() and Sleep() in Selenium are used to delay the speed of execution.

- setSpeed: Sets the execution speed with a delay of milliseconds, followed by the Selenium operation. By default, the delay is 0 milliseconds.
- sleep: Causes the suspension of execution of the current thread for a specified period.

Selenium Advanced Interview Questions (6 to 10 Years)

37. What are the different types of waits available in WebDriver?

There are two types of waits available in WebDriver:



- Implicit wait: These waits are used to provide a default waiting time (say, 30 seconds) between the consecutive test steps across the entire test script. Hence, the subsequent test step would only be executed when the 30 seconds are over after executing the previous test step.
- Explicit wait: These waits are used to halt the execution until a particular condition is met or the maximum time has elapsed.

Explicit waits are instantiated for a particular instance only, whereas implicit waits are not.

Want to learn about Selenium with Python! Check out our blog on Selenium Python Tutorial for beginners.

38. How do you handle a frame in WebDriver?

An iframe (an acronym for 'inline frame') is used to insert another document within the current HTML document.

Selecting iframe by ID:

```
driver.switchTo().frame(“ID of the frame“);
```

Locating iframe using the tagName:

```
driver.switchTo().frame(driver.findElements(By.tagName(“ifram
e”).get(0));
```

Locating iframe using index:

- frame(index)

```
driver.switchTo().frame(0);
```

- frame(“Name of the Frame”)

```
driver.switchTo().frame(“name of the frame”);
```

- frame(WebElement element)

  Select Parent Window

```
driver.switchTo().defaultContent();
```

Go through the Selenium Course in London to get a clear understanding of Selenium!

39. How do you set the test case priority in TestNG?

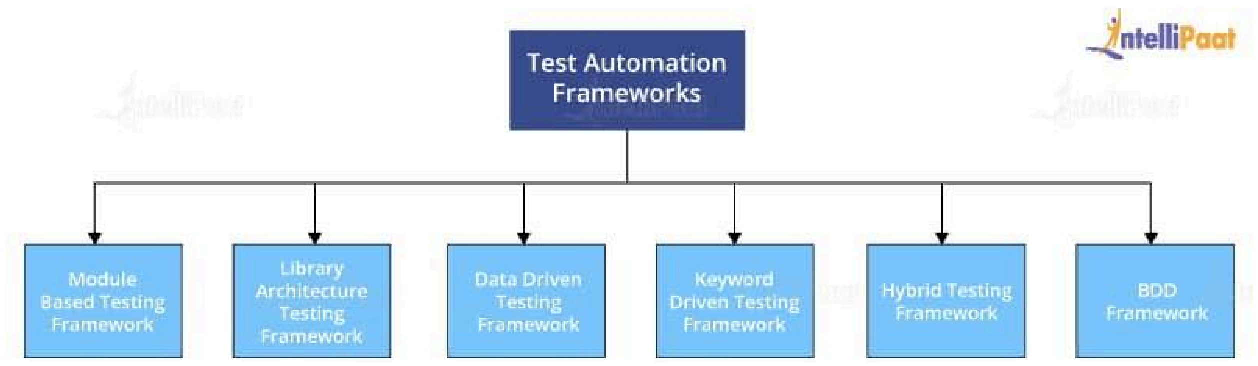
Setting Priority in TestNG:

Test Execution Sequence:

- Method1
- Method2
- Method3

40. What are the different types of frameworks?

The various categories of frameworks are outlined as follows:


Test Automation Frameworks
Module Based Testing Framework
Library Architecture Testing Framework
Data Driven Testing Framework
Keyword Driven Testing Framework
Hybrid Testing Framework
BDD Framework

- Module-based testing framework: This framework divides the entire application under test (AUT) into a number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts are taken together, it builds a larger test script representing more than one module.

- Library architecture testing framework: Instead of dividing AUT into test scripts, with this framework, we segregate the application into functions or rather common functions that can be used by the other parts of the application as well. Thus, we create a common library constituting common functions for AUT. Therefore, these libraries can be called from the test scripts whenever required.
- Data-driven testing framework: The data-driven testing framework helps us segregate the test script logic and the test data from each other. It lets us store the test data into an external database. The data is conventionally stored in 'key–value' pairs. Keys can be used to access and populate the data within the test scripts.
- Keyword-driven testing framework: The keyword-driven testing framework is an extension to the data-driven testing framework in the sense that it not only segregates the test data from the scripts but also keeps a certain set of codes belonging to the test script in an external data file.
- Hybrid testing framework: A hybrid testing framework is a combination of more than one of the above-mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.
- Behavior-driven development framework: The behavior-driven development framework allows the automation of functional validations

in an easily readable and understandable format for Business Analysts, Developers, Testers, etc.

Selenium Interview Questions for 6 to 7 Years Experience

41. Describe the difference between Selenium and QTP.

Feature	Selenium	Quick Test Professional (QTP)
Browser compatibility	It supports almost all popular browsers: Firefox, Chrome, Safari, Internet Explorer, Opera, etc.	QTP supports Internet Explorer, Firefox, and Chrome. It only supports Windows operating system
Distribution	It is distributed as an open-source tool and is freely available	It is distributed as a licensed tool and is commercialized
Application under test (AUT)	It supports the testing of web-based applications only	It supports the testing of both web-based and Windows-based applications

Object repository	Object repository needs to be created as a separate entity in Selenium	QTP automatically creates and maintains the object repository
Language support	It supports multiple programming languages like Java, C#, Ruby, Python, Perl, etc.	It supports only VBScript
Vendor support	As Selenium is a free tool, users would not get the vendor's support in troubleshooting issues	Users can easily get the vendor's support if they face any issues

Learn more about Selenium in this Selenium Training in New York to get ahead in your career!

42. In Selenium, what are breakpoints and start points?

Breakpoints: Breakpoints are used to stall the execution of the test. The execution will stop whenever a breakpoint is implemented, and this will help us check whether the code is working properly or not.

Start points: Start points are the points from where the execution should begin. Start points can be used when we want to run the test script from the middle of the code or after a breakpoint.

43. Mention the need for session handling while working with Selenium.

While working with Selenium, we need session handling. This is because, during test execution, Selenium WebDriver has to interact with the browser all the time to execute the given commands. It is also possible that, before the current execution completes, someone else starts the execution of another script on the same machine and in the same type of browser. So, to avoid such a situation, we need session handling.

Also, check out the blog on how to use Selenium with Java.

44. Mention the types of listeners in TestNG.

The types of listeners in TestNG are:

1. IAnnotationTransformer
2. IConfigurable
3. IConfigurationListener
4. IExecutionListener

5. IHookable
6. IInvokedMethodListener
7. IInvokedMethodListener2
8. IMethodInterceptor
9. IReporter
10. ISuiteListener
11. ITestListener

45. How can we handle Windows-based popups with Selenium?

Selenium only supports web application testing. It does not support the testing of Windows-based applications or mobile applications. To handle Windows-based popups, third-party intervention is required. AutoIt and Robot Class are examples of third-party tools that we can use alongside Selenium to handle Windows-based popups.

Interested in learning Selenium? Check out our Selenium Training in Sydney!

46. How can you take screenshots in Selenium?

Selenium provides a method called getScreenshotAs() that can be used to capture a screenshot of the current browser window. We can then save the screenshot to a file or attach it to the test report.

47. For the database testing in Selenium WebDriver, what API is required?

For the database testing in Selenium WebDriver, we need the JDBC (Java Database Connectivity) API. It allows us to execute SQL statements.

48. How can you identify an object in Selenium?

We can use isElementPresent (string locator) to find an object in Selenium. It takes a locator as the argument and, if found, returns a Boolean.

Become a master of Selenium by taking this online Selenium Course in Toronto!

49. How do you implement data-driven testing in Selenium?

Data-driven testing allows the same test script to be executed with multiple sets of data, making it easier to test different scenarios and edge cases. The candidate should know how to use external data sources, such as Excel or CSV files, to feed the test data into the script and how to handle the output results.

50. Mention the use of XPath in Selenium testing.

XPath is used to define web elements on a web page. The major advantage of XPath is that it helps us identify elements dynamically.

There are two types of XPath:

- Absolute XPath
- Relative XPath

51. Can WebDriver test mobile applications?

No, WebDriver is a testing tool used for web-based applications. So, we cannot test mobile applications with Selenium WebDriver.

52. Explain how Selenium Grid works.

Selenium Grid creates a test suite that works by forwarding test cases to the hub, and from there, the test cases are redirected to Selenium WebDriver. WebDriver will then execute them in the browser. The test suite allows for running tests in parallel.

Hope you find our comprehensive blog on Selenium testing interview questions useful. Do let us know in the comments section if you could make use of these Selenium topics for the interview.

Check out our blog to learn about Types of Software Testing!

53. How do you handle synchronization issues in Selenium WebDriver?

Synchronization issues can occur when the script moves too quickly, and the web page is not ready for the next step. To handle synchronization issues in Selenium WebDriver, we can use different types of waits like Implicit Wait, Explicit Wait, or Fluent Wait.

We can also use the Thread.sleep() method to pause the execution of the script for a specified amount of time. However, this method should be used sparingly as it can make the script slower and less efficient.

54. Explain the difference between driver.close() and driver.quit() commands in Selenium.

The following is the major difference between both commands:

- close() command closes the currently active browser window, which is being used by the user or which is currently accessed by the web driver.
- quit() command closes all the windows opened by the program, unlike the driver.close () command.

Both commands do not take any value and also do not accept any parameter.

55. Explain the difference between findElement() and findElements() in Selenium.

The difference between findElement() and findElements() is as follows:

- findElement(): It finds one particular element within the current page using the locating mechanism. It returns the first element located by the locator.
- findElements(): It finds all the elements within the current page with the help of the locating mechanism. Hence, it returns a list of matching web elements found by the locator.

Get ready for a Manual Testing job by going through these Top Manual Testing Interview Questions and Answers prepared by Industry Experts!

56. Mention the types of navigation commands.

The following are the navigation commands provided by Selenium:

- navigate().back(): It takes the user back to the previous or the last-used web page, according to the history.
- navigate().forward(): It takes the user to the next web page, according to the browser history.
- navigate().refresh(): It allows the user to refresh the current web page by reloading all the web elements.
- navigate().to(): It takes the user to a new web page in a new window, depending on the URL specified.

Selenium Interview Questions for 8 to 10 Years Experience

57. How do you find broken links in Selenium WebDriver?

We can detect whether the given links are broken or not by using the following process:

1. First, accumulate all the links present on a web page using the anchor tag. For each tag, use the attribute ‘href’ value to obtain the hyperlink.
2. Send HTTP requests for each link and verify the HTTP response code

3. Based on the HTTP response code, determine if the link is valid or broken. Then, use the driver.get() method to navigate to a URL, which will respond with a status of 200 – OK (200 – OK indicates that the link is working). If we get any other status, then it indicates that the link is broken
4. Repeat the same process for all the links captured

Preparing for a Software Testing Interview! Check out our Software Testing Interview Questions.

58. What is the use of the 'ExpectedConditions' class in Selenium?

The 'ExpectedConditions' class in Selenium is a utility class that provides a set of predefined conditions that can be used to wait for a certain condition to be met before proceeding with the next step in a test. These conditions can be used in conjunction with the 'WebDriverWait' class to wait for elements to be visible, clickable, or to have a certain text, among other things.

For example, the 'ExpectedConditions.visibilityOfElementLocated(By.id("elementId"))' can be used to wait for an element with a specific ID to be visible on the page. The 'WebDriverWait' class can then be used to wait for this condition to be met before interacting with the element. This helps to ensure that the element is present and visible before the test interacts with it, which can help prevent errors and improve the stability of the test.

59. Explain the significance of the @FindBy annotation in POM.

@FindBy annotation is used to identify web elements in POM. The candidate should be able to explain its importance in improving code readability and maintainability.

60. How do you handle multiple windows in Selenium WebDriver?

In Selenium WebDriver, you can handle multiple windows by actively switching between them using the 'switchTo()' method of the 'WebDriver' interface.

Here are the steps to handle multiple windows in Selenium WebDriver:

1. To acquire the handle of the current window, one can employ the 'getWindowHandle()' method from the 'WebDriver' interface. This method enables access to the unique identifier of the current window.
2. To obtain all window handles in a formal manner, you can utilize the 'getWindowHandles()' method from the 'WebDriver' interface. This method retrieves a set of all currently open window handles.

3. Iterate through the window handles using a loop and check for the window handle of the desired window you want to switch to.
4. After identifying the window handle of the desired window, utilize the 'switchTo()' method from the 'WebDriver' interface by passing the window handle as an argument. This operation allows for the web driver to shift its focus to the designated window.
5. Perform any required actions in the new window.
6. After completing the tasks in the new window, switch back to the original window by using the window handle of the original window that you stored earlier.

Notably, Selenium WebDriver handles the transition to a new window. Upon switching to a new window, the WebDriver automatically adjusts the focus, enabling seamless interaction. Subsequently, when you have completed your tasks in the new window, you can effortlessly switch back to the original window and seamlessly proceed with your testing activities.

61. How do you handle browser cookies in Selenium WebDriver?

When performing web automation using Selenium WebDriver, it is essential to understand how to handle browser cookies effectively. Browser cookies are small data stored by websites on a user's browser, used for various purposes like session management, personalization, and tracking.

- Getting Cookies: To retrieve cookies from the browser using Selenium WebDriver, we can use the get_cookies() method. This method returns a set of dictionaries, each representing a cookie. Each cookie dictionary typically contains attributes such as name, value, domain, path, expiry, and secure flag.
- Adding Cookies: To add cookies in Selenium WebDriver, we utilize the add_cookie() method. This method takes a dictionary containing the cookie attributes as its parameter. The most crucial attributes are the name and value, which must be provided. Other attributes like domain, path, expiry, and secure flag can also be set if required.
- Deleting Cookies: To remove cookies using Selenium WebDriver, we can use the delete_cookie() method, it takes the cookie's name as a parameter.
- Managing Individual Cookies: Selenium WebDriver allows us to manipulate cookies individually. We can access a specific cookie using its name and modify its attributes using the get_cookie() and add_cookie() methods, respectively.
- Working with Expiry: Cookies often have an expiry time. Selenium WebDriver enables us to handle this by setting a cookie's expiry time

using the expiry attribute in the cookie dictionary. We can provide an expiry value in Unix timestamp format to set the desired expiration.

Handling browser cookies in Selenium WebDriver is crucial for scenarios like logging in as a specific user, maintaining a session state, or performing tests that rely on cookie-based functionalities. Utilizing the methods and techniques mentioned above, you can effectively manage and manipulate cookies during your web automation tasks with Selenium WebDriver, ensuring accurate and reliable test execution.

62. How can you perform mouse hover actions in Selenium?

To perform a mouse hover action in Selenium, we can use the Actions class. We first need to create an object of the Actions class and then use the moveToElement() method to move the mouse to the element we want to hover over. We can then perform the required action on the element using click() or other similar methods.

63. What is Cucumber?

Cucumber is a BDD testing tool that enables developers to write test cases in a natural language format that can be understood by non-technical stakeholders. It supports multiple programming languages, including Java, Ruby, and Python, and can be integrated with various testing frameworks, including Selenium WebDriver.

It allows developers to write executable specifications in a readable format and automate the testing process. It enables developers to collaborate and communicate effectively with the team, including product owners, developers, and testers.

64. How do you perform Cross browser testing using Selenium?

Cross browser testing is a critical aspect of software testing that ensures the compatibility and functionality of a web application across different web browsers. Selenium, a popular open-source automation testing framework, provides a comprehensive solution for performing cross browser testing efficiently and effectively.

This process involves creating and executing test scripts using Selenium WebDriver to verify the consistency of the application's behavior across multiple browsers.

To perform cross browser testing using Selenium, the following steps can be followed:

- Test Environment Setup: Start by setting up the necessary testing environment, including installing the required browsers (e.g., Chrome, Firefox, Safari, Internet Explorer) and the corresponding browser drivers (e.g., ChromeDriver, GeckoDriver).

- Choose the Browsers: Determine the target browsers based on the application's target audience and market share. Consider both popular browsers and any specific browsers relevant to your user base.
- Create Test Scripts: Develop test scripts using Selenium WebDriver and the programming language of your choice (e.g., Java, Python, C#). The test scripts should encompass a wide range of test cases, covering various features and functionalities of the application.
- Implement Cross Browser Testing: Within the test scripts, configure the desired browser capabilities and instantiate the WebDriver accordingly. This allows you to run the same tests across different browsers seamlessly.
- Execute the Test Scripts: Run the test scripts on each targeted browser individually, using the appropriate browser driver. Selenium WebDriver will launch the browsers, navigate to the application, and execute the defined test cases.
- Validate Results: Compare the expected results with the actual results obtained from each browser. Identify any inconsistencies, discrepancies, or browser-specific issues and address them accordingly.

- Reporting and Analysis: Generate detailed test reports and analyze the results. This includes documenting any discrepancies or failures encountered during cross browser testing.

By performing cross browser testing using Selenium, you can identify and resolve any browser-specific issues, ensuring consistent functionality and a seamless user experience across different browsers. It is essential to regularly update the browser versions and drivers to stay compatible with the latest updates and changes in the browser landscape.

65. Can you explain how to run Selenium tests in parallel?

Running Selenium tests in parallel can improve the test execution time and reduce the overall test suite duration. The candidate should know how to set up a Selenium grid or use a cloud-based testing service to run tests in parallel on multiple machines or browsers.

66. Can you explain how you would perform drag-and-drop operations in Selenium WebDriver?

Performing drag-and-drop operations in Selenium WebDriver can be done using the ‘Actions’ class. The ‘Actions’ class provides several methods for simulating mouse and keyboard actions, including drag-and-drop.

Here are the steps to perform drag-and-drop operations in Selenium WebDriver:

First, you need to instantiate an instance of the ‘Actions’ class by passing in the ‘WebDriver’ instance as a parameter.

Next, you need to locate the source and target elements on the page. These are the elements that you want to drag and drop.

Once you have located the source and target elements, you can use the ‘drag and drop (WebElement source, WebElement target)’ method of the ‘Actions’ class to simulate a drag-and-drop operation.

You can use the ‘build()’ method and ‘perform()’ method of the ‘Actions’ class to build the drag-and-drop action, and then perform it on the elements.

Here’s an example of how you would perform a drag-and-drop operation using Selenium WebDriver:

```
WebElement sourceElement =
driver.findElement(By.id("source"));

WebElement targetElement =
driver.findElement(By.id("target"));
```

```
Actions actions = new Actions(driver);

actions.dragAndDrop(sourceElement,
targetElement).build().perform();
```

67. How do you handle dynamic web elements in Selenium?

Dynamic web elements are those elements that change continuously on a web page. You can handle dynamic web elements using various methods such as:

- Using regular expressions in locators
- Using XPath functions like contains, starts-with, ends-with
- Using CSS selectors

68. Can you explain how you would automate the login functionality of a web application using Selenium WebDriver?

To automate the login functionality of a web application using Selenium WebDriver, we first need to identify the username and password fields on the login page using locators such as ID, name, or CSS selector.

Once we have located the elements, we can use the sendKeys() method to enter the username and password into the respective fields. We can then use the click() method to click the login button to submit the login form.

69. How do you handle dynamic dropdowns in Selenium WebDriver?

Dynamic dropdowns are those where the options change based on the selection made in another dropdown. To handle dynamic dropdowns in Selenium WebDriver, we first need to locate the parent dropdown and select an option from it. Then, we need to wait for the child dropdown to populate with new options before locating and selecting an option from it.

70. Can you explain how you would automate the testing of a file upload functionality using Selenium WebDriver?

In order to automate the testing of file upload functionality with Selenium WebDriver, the initial step involves locating the file upload button on the web page using locators such as ID, name, or CSS selector.

Once the upload button is successfully located, the sendKeys() method can be utilized to input the file path of the desired file that needs to be uploaded.

Alternatively, the AutoIT tool can be employed to handle the file upload dialog box, as Selenium WebDriver lacks the inherent capability to interact with it.

71. Can you explain how you would automate the testing of a responsive web application using Selenium WebDriver?

The browser window resize function may be used to enlarge the browser window to various sizes in order to evaluate how the program works on various screen sizes during automated testing of a responsive web application using Selenium WebDriver. The browser development tools may also be used to simulate other devices and evaluate how the application operates on them.

72. Can you explain how you would automate the testing of a pop-up window using Selenium WebDriver?

Finding the element that causes the pop-up window using locators like ID, name, or CSS selector is the first step in automating the testing of a pop-up window using Selenium WebDriver.

After that, the pop-up window may be opened by using the click() function. Using WebDriver's switchTo() function, we can switch to the pop-up window and take the appropriate steps there. The same procedure may then be used to return to the main window.

73. How would you automate a web application that uses a CAPTCHA to prevent automated access?

CAPTCHA is a popular security mechanism that requires users to prove that they are humans.

- Use a CAPTCHA-solving service: There are many third-party services that offer CAPTCHA-solving solutions. These services use machine learning algorithms and human workers to solve CAPTCHAs. You can integrate these services into your automation script and use them to automatically solve the CAPTCHA.
- Train a machine learning model: You can use machine learning to train a model to recognize and solve CAPTCHAs. This requires a large dataset of CAPTCHA images and corresponding solutions, as well as expertise in machine learning algorithms. However, once the model is trained, it can be used to automatically solve CAPTCHAs.
- Use OCR technology: Optical Character Recognition (OCR) technology can be used to automatically read and recognize the characters in a CAPTCHA image. This requires a high-quality image and accurate OCR software, but it can be a viable solution for some CAPTCHA types.

- Use human input: In some cases, it may be more efficient to use human input to solve CAPTCHAs. You can hire human workers or use crowdsourcing platforms to solve CAPTCHAs as part of your automation process.

74. How would you test a web application that uses a single-page application (SPA) framework like Angular or React?

To ensure the proper functioning, user-friendliness, and optimal experience of a web application developed using a single-page application (SPA) framework like Angular or React, a comprehensive testing approach is essential. The following explanation outlines the detailed methodology for testing a web application built with a SPA framework:

- Understand the SPA Architecture: Familiarize yourself with the underlying architecture and concepts of the SPA framework, such as Angular or React. Gain knowledge about components, routing, state management, and asynchronous data handling.
- Identify Test Scenarios: Analyze the application's functionality and identify key test scenarios. This includes testing navigation, form submissions, data fetching, dynamic rendering, and user interactions.

- Unit Testing: Begin with unit testing to test individual components, services, and utilities. Utilize testing frameworks specific to the SPA framework, such as Jasmine or Jest, to write and execute unit tests. Focus on testing component rendering, event handling, and data manipulation.
- Integration Testing: Perform integration testing to ensure the correct collaboration between components, services, and modules within the application. Test interactions between components, API integrations, and third-party library integrations. Utilize testing frameworks like Karma or Enzyme to execute integration tests.
- End-to-End Testing: Conduct end-to-end (E2E) testing to simulate user interactions and test the entire application flow. Use tools like Selenium WebDriver or Cypress to write automated E2E tests. Test navigation, form submissions, user authentication, and other critical user flows.
- Accessibility Testing: Ensure the web application adheres to accessibility guidelines by performing accessibility testing. Verify that the application is accessible to users with disabilities and meets WCAG (Web Content Accessibility Guidelines) standards. Use tools like Axe or Lighthouse for automated accessibility testing.
- Compatibility Testing: Test the application across different browsers, devices, and operating systems. Verify that the SPA functions correctly and appears consistent across popular browsers like Chrome, Firefox,

Safari, and Edge. Use browser testing tools or cloud-based testing platforms to perform cross-browser and cross-device testing.

- Security Testing: Assess the application for potential security vulnerabilities. Conduct security testing, including penetration testing, to identify and mitigate security risks. Validate inputs, test against common vulnerabilities, and ensure secure communication.
- Continuous Testing and Automation: Establish a resilient test automation framework to enable continuous testing practices. Automate repetitive tests, including unit tests, integration tests, and end-to-end (E2E) tests. Seamlessly integrate testing into the Continuous Integration/Continuous Deployment (CI/CD) pipeline to facilitate faster feedback loops and ensure the consistent delivery of high-quality software.

75. How would you test a web application that uses WebSocket or real-time communication?

Real-time communication and WebSocket are both becoming more and more common in online applications and testing them calls for a new strategy. You may monitor and examine the WebSocket traffic using tools like Fiddler or Wireshark to evaluate a web application that makes use of WebSockets or real-time communication.

You can also use testing frameworks like JMeter or Gatling to simulate high volumes of WebSocket traffic and test the application's performance under load.

Additionally, manual testing can be done by emulating different scenarios and ensuring that the WebSocket connection is established and maintained properly. Testing the application's error-handling capabilities and security measures is also important in ensuring the reliability and security of real-time communication.

76. How would you handle a scenario where a web application requires user authentication before accessing certain pages?

User authentication is a common requirement in web applications, and testing it requires a different approach. To handle a scenario where a web application requires user authentication before accessing certain pages, the application can implement a login system that verifies user credentials and grants access to authorized users.

This can involve using secure authentication methods like password hashing and using cookies or tokens to maintain user sessions. Testing should ensure that only authenticated users can access restricted pages and that access controls are properly implemented.

Additionally, testing should cover scenarios such as session timeouts, session hijacking, and other security risks associated with user authentication.

77. How would you handle a scenario where a web application uses third-party APIs or services?

Web applications often use third-party APIs or services, and testing them requires a different approach. A hybrid framework in Selenium combines both keyword-driven and data-driven approaches to test automation.

To implement it, you need to define a set of keywords that represent the actions and assertions you want to perform on the application. Then, you create test scripts that use these keywords to interact with the application and verify its behavior.

78. How would you handle a scenario where a web application has dynamic URLs?

Dynamic URLs can be challenging to test, as they change frequently. When dealing with a web application that has dynamic URLs, it is important to implement a consistent and logical URL structure that is easy for both users and search engines to understand.

This can be achieved by using descriptive keywords in the URL and implementing URL rewriting techniques to ensure that the URLs remain clean and user-friendly.

Additionally, using canonical tags and setting up proper redirects can help avoid issues with duplicate content and broken links.

79. What is BDD (Behavior Driven Development)?

Behavior Driven Development (BDD) is an Agile software development approach that focuses on defining the behavior of a software application in a way that is understandable by all stakeholders, including developers, testers, and business analysts.

It is based on the principles of Test Driven Development (TDD), but it emphasizes collaboration and communication between team members. It uses natural language descriptions of features, scenarios, and acceptance criteria to define the expected behavior of the software.

80. How would you handle a scenario where a web application uses complex data structures like trees or graphs?

When working on web applications that rely on complex data structures like trees or graphs, it's essential to grasp their fundamental data model fully and how it impacts application functionality. To reach optimal performance levels, efficient algorithms, and data structures for manipulating or traversing these structures must also be utilized in addition to rigorous testing procedures that protect data integrity while mitigating potential memory allocation or resource issues.

81. How would you handle a scenario where a web application uses a lot of AJAX calls to update its UI?

AJAX calls can make a website dynamic, but they can also pose unique problems in terms of automation. When using third-party APIs or services, it is imperative to test whether the application can effectively communicate with them while handling errors that arise – network outages, rate limiting and invalid responses should all be covered when testing these external services.

Additionally, an application must manage authentication and authorization to securely access external services. Error handling and logging should help diagnose issues with third-party APIs or services; while regular testing will help to ensure that its operation continues as designed should changes be made to those external APIs or services.

82. How would you optimize your Selenium scripts for speed and performance?

Optimizing Selenium scripts for speed and performance is key to decreasing execution time. There are various steps you can take to do so; one approach might be reducing page refreshes/waits in favor of conditional waits that trigger by specific page elements.

Headless browsers or cloud-based services can also help reduce test runtime by simultaneously running tests in parallel, shortening overall run times. Another technique to reduce data transfer between browser and server by employing efficient selectors and eliminating unnecessary requests can help limit data transfers between browser and server; additionally, optimizing test environments to reduce unnecessary logging can boost Selenium script performance as well.

83. How would you handle a scenario where a web application has multiple languages and requires localization testing?

Localization testing is essential when your web app serves users from multiple languages. To effectively conduct localization testing for multiple-lingual web apps, several steps should be taken.

First and foremost, an application should support localization with language packs or resource files which can be tested to ensure accurate and consistent translations across languages. Furthermore, its user interface and layout should also be rigorously scrutinized against various character sets and formats of different languages to ensure compatibility.

Furthermore, testing should cover scenarios like date/time formats, currency symbols, and regional variations.

84. How would you handle a scenario where a web application uses browser-specific features like ActiveX or Java applets?

Browser-specific features can create compatibility issues while automating web applications. To handle a scenario where a web application uses browser-specific features like ActiveX or Java applets, it is important to ensure that the application can properly communicate with the browser and handle any compatibility issues that may arise.

Testing should cover scenarios like browser versions, operating systems, and browser security settings that may impact the use of these features. Additionally, ensuring that alternative solutions are available for users who cannot use these features can help maintain accessibility and functionality for all users.

85. How would you handle a scenario where a web application uses security mechanisms like SSL or OAuth?

Security mechanisms like SSL and OAuth are crucial in web applications. To handle a scenario where a web application uses security mechanisms like SSL or OAuth, it is important to ensure that the application properly implements these mechanisms and handles any security-related issues that may arise.

Testing should cover scenarios like secure communication over HTTPS, token-based authentication using OAuth, and ensuring that user credentials are properly encrypted and stored.

86. How would you handle a scenario where a web application uses third-party plugins like Adobe Flash or Microsoft Silverlight?

Third-party plugins can create compatibility issues while automating web applications. To handle a scenario where a web application uses third-party plugins like Adobe Flash or Microsoft Silverlight, it is important to ensure that the application can properly communicate with the plugins and handle any compatibility issues that may arise. Testing should cover scenarios like browser versions, operating systems, and plugin security settings that may impact the use of these plugins.

87. How do you ensure the stability and reliability of your Selenium tests?

Selenium tests are susceptible to fragility and potential failure caused by factors like varying browser versions, network disruptions, or modifications in the application being tested. To guarantee the steadfastness and dependability of Selenium tests, it is essential to employ resilient and sustainable test code capable of accommodating dynamic changes in the application. This encompasses utilizing efficient selectors, modularizing the test code, and minimizing the reliance on hardcoded values. By adhering to these practices, the stability and reliability of Selenium tests can be enhanced, ensuring smoother test execution even in the face of evolving conditions or alterations in the application under examination.

88. Can you explain how Selenium works under the hood?

A proficient Selenium developer should possess a solid comprehension of the intricate interactions between Selenium and web browsers, along with the mechanisms involved in executing commands and retrieving outcomes using the WebDriver API.

Selenium is a comprehensive suite of tools utilized for automating web browsers. Among these tools, Selenium WebDriver stands out as the most widely employed solution. It establishes communication with the browser through a set of commands and interfaces, thereby emulating user interactions with the web application.

These commands are dispatched to a browser-specific driver, which interprets them into actions that are specific to the particular browser. Selenium WebDriver also encompasses a diverse range of APIs that facilitate interaction with various elements on a web page. These APIs enable tasks such as element discovery, clicking on elements, and entering text into form fields.

In addition to functional and regression testing automation, Selenium can be effectively utilized for a myriad of web automation tasks. Possessing expertise in these aspects equips a Selenium developer with the necessary skills to leverage the full potential of Selenium and deliver robust and efficient test automation solutions.

89. How do you implement a Hybrid Framework in Selenium?

A Hybrid Framework is a combination of different design patterns and approaches, such as Data-Driven, Keyword-Driven, or Behavior-Driven Testing, to improve the modularity, reusability, and maintainability of Selenium tests.

To implement a Hybrid Framework in Selenium, you would first identify the different types of tests required, such as functional, regression, or performance testing.

Then, you would create modular test scripts using a combination of data-driven and keyword-driven approaches.

These scripts can be managed through a test management tool and executed using a test runner. Additionally, implementing a reporting mechanism can provide valuable insights into the test results and aid in debugging and analysis.

90. How do you handle security testing in Selenium?

Security testing involves testing the web application's vulnerabilities and threats, such as XSS, CSRF, or SQL injection, to ensure that it meets the security standards and regulations.

When handling security testing in Selenium, it is important to test for vulnerabilities such as cross-site scripting (XSS), injection attacks, and broken authentication and session management.]

This can be achieved by using security testing tools such as OWASP ZAP or Burp Suite in conjunction with Selenium. Additionally, implementing secure coding practices and ensuring that all inputs and outputs are properly validated can help prevent potential security breaches.

91. How do you integrate Selenium with other testing tools, such as Jenkins or TestNG?

Selenium can be integrated with other testing tools and frameworks to improve the automation workflow and generate useful reports and metrics. Integrating

Selenium with other testing tools such as Jenkins or TestNG involves configuring the testing tools to work with Selenium.

This can be achieved by adding Selenium libraries to the project and setting up test configurations to run Selenium tests. Additionally, integrating with a continuous integration tool like Jenkins allows for automated testing and reporting, while TestNG provides advanced testing features such as grouping and parallel execution.

92. How do you implement performance testing in Selenium?

Performance testing involves measuring the web application's response time, throughput, and scalability under different loads and stress levels. To implement performance testing in Selenium, you would use load testing tools such as JMeter or LoadRunner to simulate heavy user traffic and measure the performance of the web application.

Selenium scripts can be integrated with these tools to simulate real user behavior and generate load on the application. Additionally, using performance profiling tools can help identify bottlenecks and areas for optimization in the application

93. Is it common to implement AI and ML techniques in Selenium? Give one example to explain why.

AI and Machine Learning techniques can improve the efficiency, accuracy, and reliability of Selenium tests, such as using Natural Language Processing or Computer Vision to automate test cases and generate useful insights. It is not common to implement AI and Machine Learning techniques in Selenium, as it is primarily a testing tool.

However, some AI and ML techniques can be used in conjunction with Selenium to enhance the testing process. For example, image recognition can be used to identify UI elements, and natural language processing can be used to generate test cases from user stories.

94. Can you explain how to handle and automate complex web elements, such as iframes, pop-ups, or drag-and-drop elements?

To handle and automate complex web elements such as iframes, pop-ups, or drag-and-drop elements in Selenium, first identify the element using appropriate locators such as ID or XPath.

For iframes, switch to the frame using the driver's switchTo() method. For pop-ups, use the Alert class to handle the pop-up window. For drag-and-drop elements, make use of the Actions class to perform drag-and-drop operations.

95. What is a DesiredCapabilities class in Selenium?

The DesiredCapabilities class is a part of the Selenium WebDriver API and is used to configure the capabilities of a WebDriver instance. It is essentially a set of key-value pairs that specify the various properties of the browser or device that the WebDriver will use to run the tests.

Some examples of the properties that can be set using DesiredCapabilities include; browser type, version, platform, proxy settings, and browser language.

It is particularly useful for cross-browser testing, where you may need to run the same tests on different browsers or platforms. By setting the capabilities using this class, you can ensure that the WebDriver will use the correct settings for each browser/platform, and your tests will run consistently across all environments.

96. What is a WebDriverEventListener in Selenium and how is it used?

A WebDriverEventListener is an interface in Selenium that allows you to listen to various events that occur during the execution of your tests. Some examples of events that can be listened to include beforeNavigateTo, afterNavigateTo, beforeClickOn, afterClickOn, beforeFindBy, and afterFindBy.

By implementing the WebDriverEventListener interface and defining methods for each of these events, you can perform custom actions or log during your tests.

97. What is Gherkin?

Gherkin is a simple, natural language syntax used to describe the behavior of a software application in a way that is understandable by non-technical stakeholders. It is used in conjunction with Cucumber to define the acceptance criteria and scenarios for a software feature.

It uses keywords, such as Given, When, and Then, to describe the steps of a scenario in a clear and concise manner. Gherkin allows developers to write executable specifications that can be easily understood and reviewed by the team.

Selenium Coding Interview Questions and Answers

98. Write a Selenium WebDriver code to open a browser, navigate to a website, and print the page title in Java.

```
WebDriver driver = new ChromeDriver();
driver.get("https://www.intellipaat.com");
String pageTitle = driver.getTitle();
System.out.println("Page Title: " + pageTitle);
driver.quit();
```

99. Implement a function in Selenium WebDriver to find the total number of links on a webpage in Python.

```
WebDriver driver = new ChromeDriver();
driver.get("https://www.intellipaat.com");
Alert alert = driver.switchTo().alert();
String alertText = alert.getText();
System.out.println("Alert Text: " + alertText);
alert.accept();
driver.quit();
```

100. Write a Selenium script to perform a login operation using CSS selectors in Java.

```java
WebDriver driver = new ChromeDriver();
driver.get("https://www.intellipaat.com");
driver.findElement(By.cssSelector("input#username")).sendKeys
("your_username");
driver.findElement(By.cssSelector("input#password")).sendKeys
("your_password");
driver.findElement(By.cssSelector("button#loginButton")).clic
k();
driver.quit();
```

101. Implement a Selenium WebDriver script to capture a screenshot of a webpage in Python.

```python
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("https://www.intellipaat.com")
# Take a screenshot and save it as "screenshot.png"
```

```
driver.save_screenshot("screenshot.png")

driver.quit()
```

102. Create a Selenium script in Java to handle an alert on a webpage.

```
WebDriver driver = new ChromeDriver();

driver.get("https://www.intellipaat.com");

Alert alert = driver.switchTo().alert();

String alertText = alert.getText();

System.out.println("Alert Text: " + alertText);

alert.accept();

driver.quit();
```

Selenium MCQ Interview Questions

103. Which year was selenium created?

a. 2004

b. 2005

c. 2006

d. 2001

2004

104. Who was the creator of selenium?

a. Dan Cuellar

b. Jason Huggins

c. Rossmanith Gmbh

Jason Huggins

105. Which of the following browsers supports selenium?

a. Google Chrome

b. Safari

c. Mozilla Firefox

d. Internet Explorer

e. All of the above

All of the above

106. What do you mean by open-source software?

a. Open-source software is software that circulated across the world with its source code which also means it is available for use, and modification.
b. Open-source software is software that is easy to use and where the code is stored in a public repository also it comes with a distributed license.
c. All of the aboveAll of the above

107. Which of the following is not the alternative to selenium?

a. Cucumber
b. Cypress
c. Puppeteer
d. Mocha

Mocha

Selenium Tricky Interview Questions

108. How do you handle dynamic elements in Selenium, especially when traditional locators don't work?

To handle the dynamic element in selenium especially when traditional locators fail, follow these approach:

Dynamic XPath or CSS Selectors:

- Analyze the dynamic element, identifying consistent attributes or patterns.
- Employ XPath or CSS selectors with functions like contains(), starts-with(), or ends-with() to encompass dynamic elements.

Explicit Waits:

- Implement explicit waits to dynamically pause execution until the element stabilizes.
- Use conditions such as visibilityOf(), elementToBeClickable(), or presenceOfElementLocated() for precise timing.

Retry Mechanism:

- Establish a retry mechanism by catching exceptions and making multiple attempts to locate the element.
- This adaptive approach accommodates scenarios where dynamic changes may momentarily hinder identification.

JavaScriptExecutor:

- Harness JavaScriptExecutor to execute JavaScript code for direct interaction with dynamic elements.
- This technique proves beneficial when conventional WebDriver methods encounter challenges posed by dynamic alterations.

Wait for Page to Stabilize:

- Confirm the full loading and stabilization of the page before initiating interactions with dynamic elements.
- Implement waits for complete page loads or AJAX requests to settle, ensuring a stable environment for automation.

109. What is the purpose of the WebDriverWait class, and how do you use it effectively?

The `WebDriverWait` class in Selenium plays a vital role in handling dynamic elements by enabling explicit waiting. This is crucial for effective synchronization during test automation. To use it optimally, create an instance with a specified timeout and polling interval. Define the expected conditions, such as visibility or clickability, and apply the wait using the `until` method. This approach enhances script reliability by synchronizing with dynamic web page behavior, ensuring a smooth and accurate execution.

Example:

```
WebDriverWait wait = new WebDriverWait(driver, 10);

WebElement element =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id
("exampleId")));
```

110. How would you handle a situation where a web page contains multiple frames, and you need to interact with an element within a specific frame?

When dealing with multiple frames in Selenium, efficiently handle them by identifying the target frame using index, name, or WebElement. Utilize `switchTo().frame()` to navigate, perform actions within the frame, and then return to the default content using `switchTo().defaultContent()`. This ensures precise interaction, enhancing test reliability and effectiveness.

Selenium Scenario-Based Interview Questions

111. You are testing an e-commerce website, and during checkout, the 'Place Order' button is not responding

consistently. How would you approach debugging and resolving this issue using Selenium?

There are several approach for debugging and resolving the issue using Selenium are:

1. Optimize Element Identification: Ensure ‘Place Order’ button has a unique and SEO-friendly identifier, such as a relevant ID or class.
2. Efficient Waiting: Implement precise waits with WebDriverWait to enhance user experience and optimize loading times.
3. Robust Error Handling: Incorporate try-catch blocks with clear error messages for better SEO diagnostics.
4. Detailed Logging for Analysis: Use detailed logging statements to aid developers in analyzing and improving the checkout process.
5. SEO-friendly Naming: Name identifiers and variables descriptively, enhancing code readability and SEO friendliness.
6. Browser DevTools Insights: Leverage browser developer tools for a thorough analysis of the button element, ensuring a seamless user experience.
7. Stay Updated with Latest Technologies: Regularly update Selenium and browser drivers to align with the latest SEO-friendly practices and ensure compatibility.

8. Prioritize User Focus: Confirm browser window focus for improved user interactions and SEO performance.

112. The application you are testing involves uploading files. Explain how you would automate the testing of file uploads using Selenium WebDriver.

We can automate the testing of file uploads using Selenium WebDriver by these mentions approach:

1. Optimize File Input Element Identification: Choose SEO-friendly locators (like ID, XPath, or CSS) for the file input element.
2. Streamlined Interaction with `send_keys()`: Utilize `send_keys()` method for a seamless and SEO-friendly simulation of file selection, providing the file path.
3. Efficient Wait Strategies for Upload Completion: Implement explicit waits to optimize user experience and ensure timely file upload completion.
4. SEO-friendly Verification of Upload Success: Verify upload success using SEO-friendly messages or elements indicating a successful upload.
5. Graceful Handling of Popups (if any): Ensure a smooth transition by gracefully switching to and handling confirmation popups, if presented.

6. Detailed Logging and SEO-friendly Reporting: Incorporate detailed logging statements for transparent progress tracking and integrate with SEO-friendly reporting tools for comprehensive logs.

113. While running your Selenium tests, you encounter intermittent failures due to dynamic elements. Describe strategies to handle dynamic elements effectively and maintain test stability.

Strategies to handle dynamic elements effectively and maintain test stability are:

1. Strategic Element Waits: Implement SEO-friendly explicit waits using WebDriverWait to ensure stability when dealing with dynamic elements.
2. Dynamic XPath/CSS Selectors: Use SEO-friendly dynamic XPath or CSS selectors, focusing on relative paths to enhance test resilience against dynamic content.
3. Stable Parent Elements: Identify SEO-friendly stable parent elements and locate dynamic elements relative to them for increased resilience.
4. Retry Mechanism for Robustness: Enhance test robustness with an SEO-friendly retry mechanism, intelligently rerunning steps in case of intermittent dynamic element issues.

5. Smart Waiting Strategies: Implement SEO-friendly smart waiting strategies, considering combinations of elements, to ensure tests are adaptive to dynamic changes.
6. Page Object Model (POM): Leverage the SEO-friendly Page Object Model for structured organization of locators and methods, promoting readability and maintainability.
7. Dynamic Locator Updates: Dynamically update SEO-friendly locators during runtime to adapt to evolving application changes.
8. Capture SEO-friendly Screenshots and Logs: Capture SEO-friendly screenshots and detailed logs during failures to facilitate debugging and SEO-friendly issue resolution.
9. Regular SEO-friendly Maintenance: Regularly review and update test scripts with SEO-friendly practices to accommodate application changes, ensuring ongoing effectiveness against dynamic elements.

Selenium Tester Salary Trends

Job Role	Average Salary in India	Average Salary in the USA
Selenium Tester	Minimum – ₹3.4 LPA	Minimum – $88,725
(0-9 years of experience)	Average – ₹6.8 LPA	Average – $100,000

	Highest – ₹14 LPA	Highest – $121,500

Selenium Testing Job Trends

According to the Bureau of Labor Statistics US, the employment of Selenium Testers is projected to grow by 25% by 2032.

- Global Demand: With more than 7,000 open jobs on LinkedIn in the United States and more than 5,000 open jobs on LinkedIn in India, the demand for Selenium Testers is increasing.
- Growth Projections: The growth suggested by the Bureau of Labor Statistics of 25% in the field of Selenium Tester, might surpass all other occupations growth by 8%.

Selenium Tester Roles and Responsibilities

Job Role	Description
Test Automation Engineer	Develop and implement automated test scripts using Selenium. Execute automated test suites and analyze the result.

Quality Assurance (QA) Engineer	Design and execute manual test cases. Report and tackle the defects using a big tracking system.
Test Lead/Manager	Oversee the entire testing process and ensure that it aligns with the project. Develop and implement test plans and strategies. Assign tasks to the testing team.
Automation Architect	Design and implement the overall test automation framework. Evaluate and select appropriate tools and technologies for testing.
Test Analyst	Analyze requirements, identify test scenarios, create and execute manual test cases.

According to the job posted on Naukri.com by Publicis Sapient

Role: Senior Quality Engineer Level 1- Automation Testing Selenium

Responsibilities

1. Maintain software quality through an automated QE strategy.
2. Collaborate with the project and business to develop detailed automated scripts.
3. Ability to handle a team of 2-5 people.
4. Skill to enforce defects or other processes in the team.

5. Ability to handle low and medium complexity applications and have used at least one of the estimation techniques.

Skill Required:

1. Experience with QE for distributed and highly scalable systems.
2. Good understanding of OOPS concepts and strong programming skills in Java, Groovy, or JavaScript.
3. Familiarity with the process of test automation tool selection and test approach.
4. Experience in designing and developing automation frameworks and the creation of scripts using best industry practices, such as the Page object model.
5. Understanding of all the aspects of Quality Engineering.
6. Understanding of SOAP and REST principles.
7. Experience in creating test pipeline CI/CD.

Conclusion :

I hope this set of Selenium Tester Interview Questions will help you prepare for your interviews. Best of luck!

If you want to start your career or even elevate your skills in the field of Selenium Tester you can enroll in our Selenium Tester Course and get certified today.

If you want to deep dive into more Selenium Tester interview questions, feel free to join Intellipaat's vibrant Selenium Community and get answers to your queries from like-minded enthusiasts.