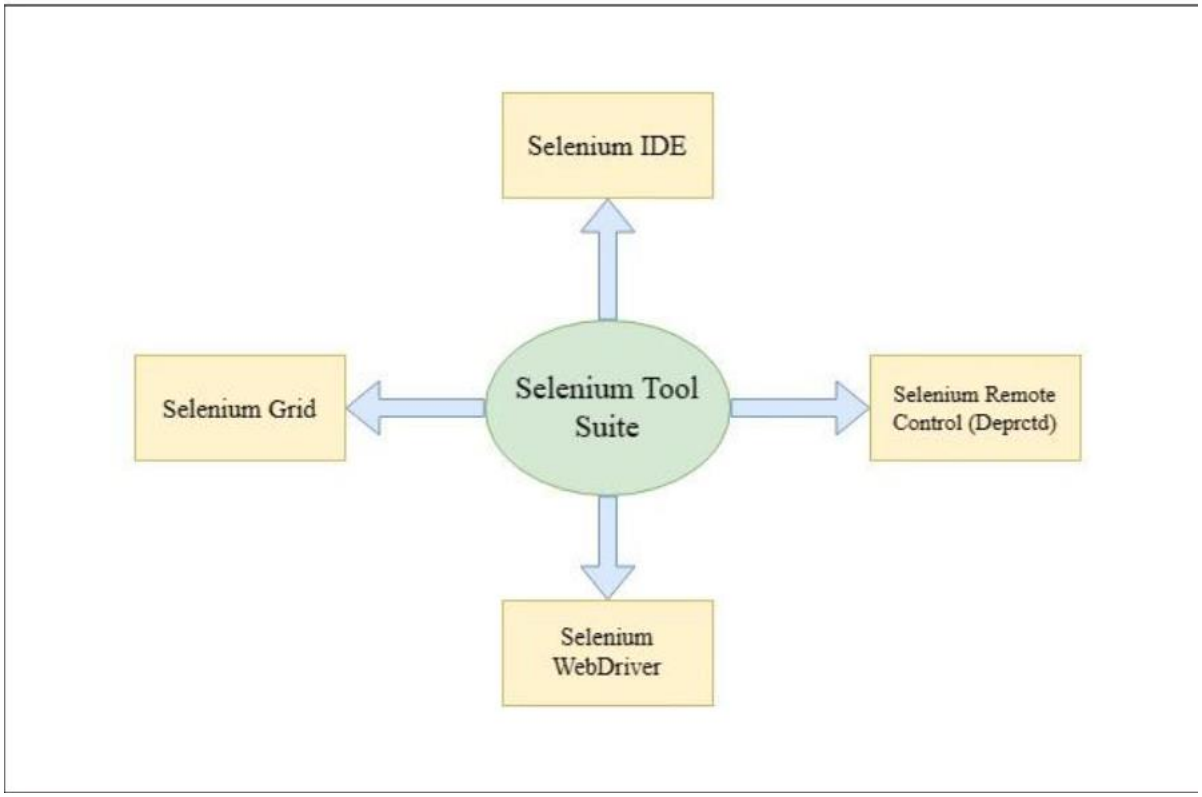
1) What is Selenium, and what is its role in software testing?

Selenium is a popular open-source software testing framework used for automating web applications. It is widely used for functional testing, regression testing, and performance testing. Selenium supports multiple programming languages, including Java, C#, Python, and Ruby, making it accessible to a wide range of developers.

Selenium Tool suite consists of 4 major components

- Selenium IDE (Integrated Development Environment)
- Selenium Remote Control (RC)
- Selenium WebDriver
- Selenium Grid



What is Selenium WebDriver?

Selenium WebDriver is a robust open-source framework for automating web browsers, primarily aimed at easing the testing and verification of web applications. As an important part of the Selenium suite, WebDriver offers a programming interface to interact with web browsers, allowing developers and testers to automate browser actions seamlessly. Unlike its predecessor, Selenium RC (Remote Control), WebDriver directly communicates with the browser, providing a more stable and efficient means of automation.

What is selenium Grid?
Selenium Grid as a tool that lets you do your tests on different computers at the same time. It's like having a master computer (the "Hub") that tells other computers (the "Nodes") what tests to run.

So, if you have a website and you want to make sure it works well on different browsers (like Chrome, Firefox, or Edge) and on different operating systems (like Windows, Mac, or Linux), you can use Selenium Grid to do all these tests at once. This saves you a lot of time because you don't have to do each test one by one on each browser or operating system.

What is Selenium IDE?
Selenium IDE (Integrated Development Environment) is an open-source web automation testing tool under the Selenium Suite1. It's a browser extension that records a user's actions in the browser using existing Selenium commands, with parameters defined by the context of each element.
Selenium IDE is like a tape recorder for your web browser. It records what you do on a website, like clicking buttons or typing in text boxes. Then, it can play back those actions whenever you want. This is useful for testing websites to make sure they work correctly.

What is Selenium Remote Control?
Selenium Remote Control (RC) is a tool used for automated testing of web applications1. It's part of the Selenium Suite, which is a set of different software tools each with a different approach to supporting browser automation

some of its key responsibilities:

- Automated Testing: Selenium is an open-source tool for automation testing of web applications across various browsers and platforms1. It enables the creation of automated test cases1.
- Cross-Browser Testing: Selenium supports multiple programming languages including Java, Python, and C#1, and allows for cross-browser testing2.
- Bug Identification: Selenium helps in efficiently identifying bugs and issues within applications2. This not only saves time but also ensures that software products meet high-quality standards before they are released to end-users2.
- Test Suite Creation: Selenium testers or automation testers may have to write automated test suites, design the BPT components and frameworks3.
- Efficiency and Speed: The time-consuming and repetitive manual tasks are automated with selenium testing, which enhances the efficiency and speed of software testing4.
- Reliability: Automated tests decrease the possibility of human errors and produce consistent results each time a program is run

2) Explain the difference between Selenium WebDriver and Selenium IDE

Selenium WebDriver is a complete system you can use to build your application test requirements, whereas Selenium IDE is an accessory to your testing environment, acting as an extra set of tools for certain testing functions
Selenium IDE offers simple installation and is easy to learn, while Selenium WebDriver is more powerful and flexible but requires more effort to learn

Selenium WebDriver works with all browsers like Firefox, IE, Chrome, Safari, Opera, etc., while Selenium IDE only works within the Mozilla browser

3) What are the different components of Selenium?

The different components of selenium are

- Selenium WebDriver
- Selenium Grid
- Selenium IDE
- Selenium Remote Control

4) How do you set up Selenium WebDriver in your test environment?

5) What is the drawback in Selenium?

Selenium is a powerful tool for automating web application testing, but it does have some limitations12345:

- Limited Support for Desktop and Mobile Applications: Selenium is mainly designed for web application testing and has limited support for testing desktop and mobile apps13.
- Requires Programming Knowledge: Test engineers should be good at any of the programming languages like Java, Python, C#, Ruby, JavaScript etc. to create effective automation tests1.
- No Built-in Reporting: Selenium doesn't have any built-in reporting capability. It has to integrate with TestNG, Extent Reports etc. to get the reporting feature12.
- Image Testing: Selenium doesn't have the capability to test using the images by default. It needs to integrate with tools like Sikuli to perform Image Based Testing12.
- Handling Dynamic Web Elements: Selenium can face challenges when handling dynamic web elements3.
- Dependency on Browser Updates: Selenium's functionality can be affected by browser updates3.
- No Official Support: As Selenium is an open-source and free tool, there is no vendor or official support. In case of issues, users have to rely on community forums24.
- Maintenance and Scalability: Selenium is a maintenance-heavy framework and can be difficult to scale as one grows25.
- No Support for REST and SOAP Platforms: Selenium can't perform automation tests on web services like SOAP or REST2.
- Steep Learning Curve for Beginners: Selenium requires expertise of your team and resources to manage

6) What is the features of Selenium?

Open Source and Portable: Selenium WebDriver is an open source and portable web testing framework that can be used on various platforms and browsers.

Combination of tool and DSL: Selenium WebDriver is a combination of tools and DSL (Domain Specific Language) that allows users to write test scripts in various programming languages, such as Java, Python, C#, Ruby, etc.

Easier to understand and implement: Selenium WebDriver commands are categorized in terms of different classes that make it easier to understand and implement.

Reduce test execution time: Selenium WebDriver supports parallel test execution that reduces the time taken in executing parallel tests.

Lesser resources required: Selenium WebDriver requires lesser resources when compared to its competitors like UFT, RFT, etc.

Supports Multiple Programming Languages: Selenium WebDriver supports multiple programming languages, such as C#, Java, Python, PHP, Ruby, Perl, and JavaScript.

Supports Multiple Operating Systems: Selenium WebDriver supports multiple operating systems, such as Android, iOS, Windows, Linux, Mac, Solaris.

Supports Multiple Browsers: Selenium WebDriver supports multiple browsers, such as Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.

Parallel Test Execution: Selenium WebDriver also supports parallel test execution that reduces time and increases the efficiency of tests.

A flexible language: Once the test cases are prepared, they can be executed on any operating system like Linux, Macintosh, etc.

No installation Required: Selenium WebDriver does not require server installation, test scripts interact directly with the browser.

7) What is the advantage and disadvantage of Selenium?

8) What are the different locators supported by Selenium WebDriver?

The different types of locators supported by Selenium WebDriver include123:

- ID Locator: Efficiently locates an element using its ID attribute.
- Name Locator: Locates an element using its name attribute.
- Class Name Locator: Identifies elements using their class name attribute.
- Tag Name Locator: Locates elements based on their HTML tag name.
- XPath Locator: Uses XPath expressions to find elements.
- CSS Selector Locator: Uses CSS selectors to locate elements.

9) What is difference between Absolute and Relative Xpath?

Absolute xpath gives the complete path of the web element i.e from html tag where as relative xpath, you can simply start by referencing the element you want and go from there.

Relative Xpaths are always preferred as they are not the complete paths from the root element. (//html//body). Because in future, if any webelement is added/removed, then the absolute Xpath changes. So Always use Relative Xpaths in your Automation.

10) How do you handle synchronization issues in Selenium WebDriver?

To handle synchronization issues in Selenium WebDriver, you can1234:

- Avoid using Thread.sleep() and use dynamic waits like implicit wait or explicit wait instead of static waits to improve test efficiency.
- Use explicit wait with WebDriverWait and ExpectedConditions to handle synchronization issues. It provides precise control over waiting conditions and enhances test stability.
- Use ImplicitlyWait(), ExplicitWait(), or FluentWait() to wait for specific conditions or elements to appear on the web page.
- Consider temporarily hard coding a large sleep where you see the issue, and you'll know if adding an explicit wait can help.

11) What is different between types of navigate commands?

The Navigation Command provides four methods: to(), back(), forward(), and refresh().

- to() Command
  Loads a new web page in the current browser window. It accepts a string parameter that specifies the URL of the web page to be loaded.

- back() Command
  Moves back one step in the browser's history stack. It does not accept any parameters and does not return anything.

- forward() Command
  Moves forward one step in the browser's history stack. It does not accept any parameters and does not return anything.

- refresh() Command
  Reloads the current web page in the browser window. It does not accept any parameters and does not return anything.

12) What is the difference between findElement() and findElements() methods in Selenium WebDriver?

Both are the methods of searchContext interface
findElement () returns single element within a web page.whereas findElements() returns a list of webelements that matches the webelement

13) What is Object respository in Selenium?

An Object Repository in Selenium is essentially a centralized storage location where we keep all the web elements that we use in our web application.

In simpler terms, think of it as a map. Just like how a map helps you find locations, an Object Repository helps you find web elements in your application. POM is used to create Object Repository

14) Is Selenium Webdriver is a library?

Selenium is a browser automation library. Most often used for testing web-applications, Selenium may be used for any task that requires automating interaction with the browser.

15) What is assertion in Selenium?

An assertion is used to compare the actual result of an application with the expected result.

16) What are the types of assertions in Selenium?

//will be covered in testng

17) How to hover mouse element in Selenium?

moveToEleement
Action class

18) what other method can be used instead of driverget()?

Instead of driver.get() we can use driver.navigate().to()

19) What are locators and types of locators?

A locator is a way to identify elements on a page. It is the argument passed to the Finding element methods.
Selenium 3 has 8 locators namely ID, Name, class, XPath, CssSelectors, LinkText, PartialLinkText, and TagName which helps us to locate the web elements on DOM.

20) How to use Assert and Verify in Selenium WebDriver

21) What is Dynamic Xpath in selenium?

Dynamic XPath is a powerful feature in Selenium that allows you to locate web elements dynamically[1][2][4]. It's particularly useful when the attributes of a web element, such as `id`, `class`, or `name`, are dynamically generated or change with every page load[1][2].

There are several ways to create Dynamic XPath in Selenium[1][2]:

1. **Using Single Attribute**: `//tagname[@attribute='value']`
2. **Using Multiple Attributes**: `//tagname[@attribute1='value1'][@attribute2='value2']`
3. **Using `contains()` Method**: `//tagname[contains(@attribute,'value')]`
4. **Using `starts-with()` Method**: `//tagname[starts-with(@attribute,'value')]`
5. **Using `text()` Method**: `//tagname[text()='text']`
6. **Using `following` and `preceding` Axes**: These XPath axes methods are used to find elements that dynamically change on refresh or any other operations[5].

For example, if you have a button with an id that changes every time the page is loaded, you could use a dynamic XPath like `//button[contains(@id,'btn')]` to locate the button based on a part of the id that doesn't change[1][2].

22) Can we test API Testing using Selenium?

Selenium can be used for API testing, it's not the most efficient tool for this purpose. Specialized API testing tools offer better capabilities and reporting for API testing tasks.

23) What is POM and Apache POI

Page Object Model, also known as POM, is a design pattern in Selenium that creates an object repository for storing all web elements.

Apache POI is an open-source Java library often utilized to create and handle Microsoft Office-based files. Users can leverage POI to perform various operations (modify, create, display, read) on specific file formats (Excel files being one of them).

24) How can we type text in textbox in Selenium?

Using sendKeys()

25) How do you Handle Dynamic Elements?

Handling dynamic elements in Selenium is crucial for robust test automation. Here are some strategies to handle dynamic elements[4][5][6]:

1. **XPath with dynamic attributes**: XPath is a powerful locator strategy that allows you to locate elements based on their attributes. You can use XPath expressions to identify elements using partial attribute values, regular expressions, or other dynamic patterns. For example, you can use XPath functions like `contains()`, `starts-with()`, or `ends-with()` to locate elements with changing attribute values[4].

2. **CSS selectors with dynamic classes or IDs**: CSS selectors provide another effective way to locate dynamic elements. You can use CSS selectors to target elements with changing classes or IDs by using attribute selectors or other techniques. For example, you can use the `^=` (starts with) or `$=` (ends with) operator to match elements with dynamic classes or IDs[4].

3. **Using relative element positions**: If the dynamic element is located relative to a stable element, you can leverage the relationship between the two elements to locate the dynamic element. For example, you can use XPath or

CSS selectors to identify a stable parent element and then navigate to the dynamic element using sibling, child, or descendant selectors[4].

4. **Explicit waits**: Selenium provides explicit wait mechanisms that allow you to wait for specific conditions to be met before locating an element. You can use methods like `WebDriverWait` and expected conditions like `visibilityOfElementLocated` or `elementToBeClickable` to wait until the dynamic element becomes visible or interactive before locating it[4].

26) How do you Handle Dropdown in Selenium?

Dropdown are handled in two ways

1) Using the select class:
   The Select class is a built-in class in Selenium WebDriver that provides methods for handling dropdowns in Selenium. To use the Select class, you must first locate the dropdown element using a locator method such as findElement().

   Once you have located the dropdown element, you can create a Select object and use its methods to select and deselect options. The Select class provides three methods for selecting options:
   selectByIndex, selectByValue, selectByVisibleText

2) Without Using the select class
   By Storing All the Options in List and Iterating Through It

27) How to get title of a webpage in Selenium?

Using getTitle() method

28) How do you handle dropdowns using Selenium WebDriver?

#above answers

29) Explain the concept of WebDriver's Page Object Model (POM) and its advantages

Page Object Model, also known as POM, is a design pattern in Selenium that creates an object repository for storing all web elements.

30) How do you handle multiple windows and in Selenium WebDriver?

Multiple Windows: Selenium WebDriver provides a getWindowHandle() method to get the handle of the current window and a getWindowHandles() method to get the handles of all open windows1. You can switch between windows using the switchTo().window(handle) method

Frames: Selenium WebDriver allows you to switch between frames using the switchTo().frame() method2. You can pass the index, name, or WebElement of the frame to this method #refer notes

31) What are the different types of waits available in Selenium WebDriver?

In Selenium WebDriver, waits are used to pause the execution of tests until certain conditions are met. This is particularly useful in dealing with elements that may take some time to appear or become interactable due to various factors like network latency or AJAX loading. There are three main types of waits available in Selenium WebDriver[12345]:

1. **Implicit Wait**: This type of wait is used to tell the WebDriver to wait for a certain amount of time before it throws a `NoSuchElementException`. The default setting is 0. Once set, the WebDriver will wait for the specified time for elements to appear when trying to locate them[12345].

2. **Explicit Wait**: This type of wait is used when you want to wait for a specific condition to occur before proceeding further with the test execution. It's more intelligent than an implicit wait because it waits for a certain condition to occur before proceeding[12345].

3. **Fluent Wait**: This is a more advanced type of wait that allows you to configure the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Additionally, you can configure this wait to ignore specific types of exceptions while waiting, such as `NoSuchElementExceptions` when searching for an element on the page

32) Explain the concept of Implicit and Explicit waits in Selenium WebDriver

#answer ABove

33) How do you handle browser cookies in Selenium WebDriver?

To handle browser cookies in Selenium WebDriver, you can use the following commands12:

- Get Cookie: Gets the cookies for the current domain.
  driver.manage().getCookies(); // Returns the List of all Cookies
  driver.manage().getCookieNamed(arg0); //Returns the specific cookie according to name
- Add Cookie: Adds a specific cookie into cookies.
- Delete Cookie: Deletes a specific cookie according to name.

34) What is TestNG, and how is it useful in Selenium testing?

TestNg in selenium is used to create test cases and generate HTML report

35) How do you perform data-driven testing using Selenium WebDriver?

For data driven testing we use Apache POI

36) Explain the concept of TestNG annotations and their usage in Selenium tests

#not in syllabus

37) What are the advantages of using TestNG over JUnit in Selenium testing?

#not in syllabus

38) How do you handle screenshots and logging in Selenium WebDriver?

Screenshots: Selenium WebDriver provides a TakesScreenshot interface which has a method getScreenshotAs(). This method can be used to capture a screenshot and store it in a specified location.

```
TakesScreenshot scrShot =((TakesScreenshot)webdriver);
File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);
#check notes once
```

Logging: For logging in Selenium WebDriver, you can use a logging framework like Log4j or SLF4J. These frameworks allow you to log messages to various output targets with different detail levels (like DEBUG, INFO, WARN, ERROR, etc.).

39) What is the role of Desired Capabilities in Selenium WebDriver?

The Desired Capabilities class helps you set the browser properties such as browser name, version, the path to the browser driver, and more to perform automated browser testing.

40) How do you handle keyboard and mouse events using Actions class in Selenium WebDriver?

#notes

41) Explain the concept of cross-browser testing in Selenium WebDriver

Selenium WebDriver allows you to execute tests across different web browsers such as Chrome, Firefox, Safari, Internet Explorer, and others. This ensures that your web application is compatible with a variety of browsers, providing a more reliable assessment of its functionality.

42) How do you handle alerts and pop-ups in Selenium WebDriver?

We use the Alert interface to handle alert and pop ups in selenium
Before interacting with the alert, you need to switch the WebDriver's focus to the alert using driver.switchTo().alert()

The common methods used are :-

1) Accept():This method is used to click on the ‘OK’ button of the alert
   driver.switchTo().alert().accept();
2) Dismiss(): This method is used to click on the “Cancel” button of the alert
   driver.switchTo().alert().dismiss()
3) getText():This method is used to capture the alert message
   driver.switchTo().alert().getText()
4) sendKeys(String stringToBeSent): This method is used to send Keys in the input box
   driver.switchTo().alert().sendKeys(“Text”)

43) What is the purpose of WebDriver's Select class in handling dropdowns?

The Select class is a built-in class in Selenium WebDriver that provides methods for handling dropdowns in Selenium. To use the Select class, you must first locate the dropdown element using a locator method such as findElement().

Once you have located the dropdown element, you can create a Select object and use its methods to select and deselect options. The Select class provides three methods for selecting options:
selectByIndex, selectByValue, selectByVisibleText

44) How do you handle JavaScript alerts and confirmations using Selenium WebDriver?

#above answer

45) Explain the concept of TestNG data providers and their usage in Selenium tests

46) How do you perform mouse hovering and drag-and-drop actions in Selenium WebDriver?

Mouse Hovering:

```
WebElement ele = driver.findElement(By.xpath("<xpath>"));

//Creating object of an Actions class
Actions action = new Actions(driver);

//Performing the mouse hover action on the target element.
action.moveToElement(ele).perform();
```

Drag and Drop

```
WebElement drag
        = driver.findElement(By.id("draggable"));
```

```
WebElement drop
   = driver.findElement(By.id("droppable"));

action.dragAndDrop(drag, drop).build().perform();
```

47) What is the difference between close() and quit() methods in Selenium WebDriver?

Close() will close only the browser window where the driver is pointing whereas quit() will close all the browser windows that were opened for testing

48) How do you handle SSL certificate errors in Selenium WebDriver?

SSL stands for secure socket layer. Handling SSL certificate errors in Selenium WebDriver involves configuring the browser to accept the SSL certificates. Here are some ways to handle SSL certificate errors in different browsers[123]:

1. **Firefox**: You can create a Firefox profile and set the `accept_untrusted_certs` property to `true`. Then you can use this profile when creating the WebDriver instance.
2. **Chrome**: You can add the `--ignore-certificate-errors` argument when creating the WebDriver instance.
3. **Internet Explorer**: You can set the `acceptSslCerts` property to `true` in the DesiredCapabilities.
4. **Safari**: Handling SSL errors in Safari is a bit more complex. You might need to execute a JavaScript piece that will allow the browser to pass through the certificate and navigate to the intended page[1].

49) Explain the concept of parallel test execution in Selenium using TestNG

50) How do you handle browser navigation (forward, backward) in Selenium WebDriver?

#notes

51) What are the different types of locators supported by Selenium WebDriver?

#above answer

52) Explain the concept of CSS selectors and their usage in Selenium WebDriver

CSS (Cascading Style Sheets) Selectors in Selenium are used to identify and locate web elements based on their id, class, name, attributes and other attributes[123]. They are a preferred locator strategy as they are simpler to write and faster as compared to XPath[123].

The `By.cssSelector(String cssSelector)` method in Selenium WebDriver is used to locate the elements. This method accepts a CSS Selector String as an argument which defines the selection method for the web elements[123].

Here are some types of CSS Selectors in Selenium[123]:

1. **ID**: In CSS, we can use `#` notation to select the `id` attribute of an element. For example, `driver.findElement(By.cssSelector("#idValue"));`.

2. **Class**: In CSS, we can use `.` notation to select the `class` attribute of an element. For example, `driver.findElement(By.cssSelector(".className"));`.

3. **Attribute**: We can also select elements based on any attribute. For example, `driver.findElement(By.cssSelector("input[name='email']"));`.

4. **Sub-String**: CSS Selectors also support sub-string matches. For example, `driver.findElement(By.cssSelector("a[href*='google']"));` will match all `a` elements whose `href` attribute contains the substring 'google'.

5. **Combining Attributes**: We can combine multiple attributes to locate a single element. For example, `driver.findElement(By.cssSelector("input[type='submit'][value='Search']"));`.

53) How do you handle dynamic elements in Selenium WebDriver?

Using starts-with, ends-with, contains

54) What is the purpose of GeckoDriver, ChromeDriver, and EdgeDriver in Selenium?

This driver is used to call these browsers to communicate with the respective browser so that we can independently check whether the application is working properly on the browsers or not

55) How do you handle file uploads and downloads in Selenium WebDriver?

56) Explain the concept of TestNG test suites and their usage in Selenium testing

57) How do you handle frames within frames using Selenium WebDriver?

#nested frames

58) What are the different types of assertions available in TestNG for Selenium tests?

59) How do you handle JavaScript execution using Selenium WebDriver?

In Selenium WebDriver, you can handle JavaScript execution using the `JavascriptExecutor` interface[45]. This interface provides two methods: `executeScript` and `executeAsyncScript`[45].

1. **`executeScript`**: This method executes JavaScript in the context of the currently selected frame or window[45]. The script is run as an anonymous function, and it can return values[45]. Here's an example:

```java
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("document.body.style.backgroundColor = 'yellow';");
```

2. **`executeAsyncScript`**: This method is used to execute asynchronous JavaScript in the context of the currently selected frame or window[45]. The JavaScript executed is single-threaded, while the rest of the page continues parsing, which enhances performance[45]. Here's an example:

60) Explain the concept of data-driven frameworks (eg, Data-driven, Keyword-driven) in Selenium

##Apacahe poi

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

To handle browser cookies in Selenium WebDriver, you can use the following commands12:

- Get Cookie: Gets the cookies for the current domain.
  driver.manage().getCookies(); // Returns the List of all Cookies
  driver.manage().getCookieNamed(arg0); //Returns the specific cookie according to name
- Add Cookie: Adds a specific cookie into cookies.
- Delete Cookie: Deletes a specific cookie according to name.

62) Explain the concept of parallel test execution using Selenium Grid

63) How do you handle dynamic waits in Selenium WebDriver?

#explain about the selenium waits

64) Explain the concept of testngxml and its role in Selenium testing

65) How do you handle SSL certificate errors in Selenium WebDriver?

#above answer

66) What is the purpose of WebDriver's TakesScreenshot interface in Selenium?

The purpose of WebDriver's TakeScreenShot is to take screenshot of the web page

67) Explain the concept of data-driven testing using Excel or CSV files in Selenium

Data-driven testing is a technique in Selenium where the test data set is separated from the actual test case. This allows you to execute the same test case with multiple sets of data. The test data is kept in an external data feed like MS Excel Sheets, CSV Files,

68) What is difference between getWindowHandle vs getWindowHandles?

69) What is difference between driverclose() vs driverquit()?

70) Can we able to automate captcha in selenium?

No

71) What is Selenium Grid?

Parallel testing and cross browser testing

72) What is forward and backward window?

73) How to handle window popup?

74) How we select the element by thier attribute value using CSS selector?

By.cssSelector("input[type= 'text']")

75) which method is mostly used to get the check box in selenium?

Click methods is usually used

76) why gettext() is used in selenium?

To get the inner text of the html page

77) How to delete the text in the text box?

Driver.findElement(By.id("")).clear()

78) Can we move to diffrent frames in selenium?

Yes we can

79) How can we handle checkboxes in selenium?

80) What is diffrence between double click and right click?

81) How to fetch the page url in selenium?

82) How to verify the tool tip text?

83) What is the procedure to import a class outside the pacakage?

84) " Write code to invoke different web browsers?

85) "

86) Write code to launch the url

87) Explain about Apachae POI?

88) What is difference between XSSF and HSSF

89) what is the advantage of Webdriver compared to RC

90) What is the difference between assert and verify commands in Selenium WebDriver?

91) How do you handle dynamic waits in Selenium WebDriver?

92) Explain the concept of parallel test execution using Selenium Grid

93) Explain the concept of parameterization in TestNG and how it can be used in Selenium tests

94) Explain the concept of TestNG test suites and their usage in Selenium testing

95) How do you handle file uploads and downloads in Selenium WebDriver?

96) How do you handle browser navigation using Selenium WebDriver?

97) How do you handle dynamic pop-ups and windows in Selenium WebDriver?

98) What is the purpose of the Actions class in Selenium WebDriver?

99) What are the advantages of using CSS selectors over XPath in Selenium WebDriver?

Unidirectional Flow: CSS selectors are unidirectional, meaning they can only traverse from parent to child3. This allows them to search for elements in the DOM tree more efficiently.

XPath Engine Differences: XPath engines are different in each browser, making them inconsistent1. For example, Internet Explorer does not have a native XPath engine, so Selenium injects its own XPath engine for compatibility of its API1. This can lead to slower performance compared to using native browser features

100) How do you handle JavaScript alerts using Selenium WebDriver?

101)