Selenium interview questions:

(//div[@id='nav-xshop'])/a[1]/following-sibling::a/parent::div

"One of the challenges I faced was dealing with dynamic web elements whose attributes changed frequently. To overcome this, I used more stable locators like XPath and CSS Selectors based on attributes that remained constant. Additionally, I implemented explicit waits to ensure that elements were fully loaded before interacting with them."

Another challenge was handling unexpected pop-ups and alerts that appeared during test execution. I addressed this by implementing alert handling in my scripts to accept or dismiss alerts as required. I also used try-catch blocks to ensure that the script could handle unexpected alerts without failing."

Managing test data for data-driven tests was a challenge, particularly with large datasets. To overcome this, I used external data sources like Excel and databases to manage test data. I also implemented data validation and sanitization processes to ensure data consistency across different test environments."

How many types of webdriver api's available in selenium?

WebDriver APIs interact with different browsers and devices.

- **Browser-specific WebDrivers** (ChromeDriver, FirefoxDriver, EdgeDriver, etc.).
- **RemoteWebDriver** for remote/cloud-based testing.
- **HtmlUnitDriver** for headless browser testing.
- **EventFiringWebDriver** for event tracking.
- **AppiumDriver** for mobile testing.

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

How do you ensure a page is loaded using selenium webdriver?

Implicit Wait:

Selenium's **Implicit Wait** tells WebDriver to wait for a specific amount of time before throwing an exception if an element is not found. This is applied globally and works for all elements.

```
WebDriver driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); // 10 seconds wait time

driver.get("https://example.com");
```

Explicit Wait (Best Practice):

Use **Explicit Wait** with WebDriverWait to wait for specific conditions, like waiting for a particular element or condition to be true before proceeding.

Example: Wait for an element to be visible (indicating that the page or a portion of the page is loaded):

```
WebDriver driver = new ChromeDriver();

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

driver.get("https://example.com");

// Wait until a specific element is visible

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

Some common ExpectedConditions:

- visibilityOfElementLocated(): Waits until an element is visible.
- elementToBeClickable(): Waits until an element is clickable.
- presenceOfElementLocated(): Waits until the element is present in the DOM.'

Page Load Timeout:

You can set a **page load timeout** to wait for a page to completely load. This tells WebDriver to wait until the page is loaded fully before interacting with the page.

```
WebDriver driver = new ChromeDriver();

driver.manage().timeouts().pageLoadTimeout(20, TimeUnit.SECONDS); // Set 20 seconds timeout for page loading

driver.get("https://example.com");
```

Using findElements() Method:

- The findElements() method returns a list of matching elements. If the element is not present, the list will be empty. This method won't throw an exception if the element is not found.

```
public boolean isElementPresent(By locator) {

   return driver.findElements(locator).size() > 0;

}
```

Status codes and it's meaning

200,500,201

- **200 OK** - Response to a successful GET, PUT, PATCH or DELETE. Can also be used for a POST that doesn't result in a creation.
- **201 Created** - Response to a POST that results in a creation. Should be combined with a Location header pointing to the location of the new resource

- **204 No Content** - Response to a successful request that won't be returning a body (like a DELETE request)
- **304 Not Modified** - Used when HTTP caching headers are in play
- **400 Bad Request** - The request is malformed, such as if the body does not parse
- **401 Unauthorized** - When no or invalid authentication details are provided. Also useful to trigger an auth popup if the API is used from a browser
- **403 Forbidden** - When authentication succeeded but authenticated user doesn't have access to the resource
- **404 Not Found** - When a non-existent resource is requested
- **405 Method Not** Allowed - When an HTTP method is being requested that isn't allowed for the authenticated user
- **410 Gone** - Indicates that the resource at this end point is no longer available. Useful as a blanket response for old API versions
- **415 Unsupported Media Type** - If incorrect content type was provided as part of the request
- **422 Unprocessable Entity** - Used for validation errors
- **429 Too Many Requests** - When a request is rejected due to rate limiting
- **500 Internal Server Error** - This is either a system or application error, and generally indicates that although the client appeared to provide a correct request, something unexpected has gone wrong on the server
- **503 Service Unavailable** - The server is unable to handle the request for a service due to temporary maintenance

Difference between put and patch Rest API Methods? **SearchContext** is the super interface of WebDriver and WebElement interface. So, SearchContext and WebDriver has parent child relationship between them.

What is batch file?

A **batch file** is a script file containing a series of commands that are executed by the Windows command-line interpreter (CMD). It typically has a .bat or .cmd file extension. In the context of **Selenium** or automated testing, batch files can be used to automate various tasks, such as setting up the environment, launching browsers, running Selenium tests, or performing other prerequisite actions before or after tests.

Benefits of Using Batch Files in Selenium:

- **Automate Repetitive Tasks**: Automates the steps of setting up the environment, running tests, and cleaning up after execution.
- **Simplify Test Execution**: Provides a simple way to execute Selenium test suites with a single click.
- **Integration with CI/CD**: Batch files can be easily integrated into CI/CD systems like Jenkins, making it easier to trigger Selenium tests automatically.

In summary, a **batch file** in Selenium can be used to automate various actions such as running tests, setting up configurations, or interacting with external systems, all of which improve the test automation process.

How can we launch a batch file in selenium webdriver project?

To launch a **batch file** from a Selenium WebDriver project, you can use **Java's Runtime or ProcessBuilder** classes to execute the batch file. This is particularly useful for running external scripts or commands during your automated test execution.

1. Using Runtime.getRuntime().exec():

You can use the exec() method from the Runtime class to execute the batch file.

```
import java.io.IOException;

public class LaunchBatchFile {

   public static void main(String[] args) {

      try {

         // Specify the path to the batch file

         String batchFilePath = "C:\\path\\to\\your\\batchfile.bat";

         // Launch the batch file

         Runtime.getRuntime().exec(batchFilePath);



         System.out.println("Batch file executed successfully.");

      } catch (IOException e) {

         e.printStackTrace();

      }

   }

}
```

How do you run a selenium webdriver test from a command line?

Running a **Selenium WebDriver** test from the command line is often done using build tools like **Maven** or **Gradle**. These tools can manage dependencies, compile the project, and run test cases without requiring an IDE. Here's how to do it step by step: **Run the Maven Test Command**: Run the following command to execute the Selenium test using Maven:

mvn clean test

Different exceptions in selenium webdriver:

Exception	Description
NoSuchElementException	Element not found in the DOM.
TimeoutException	Command took longer than the specified wait time.
ElementNotInteractableException	Element is in the DOM but cannot be interacted with.
StaleElementReferenceException	Element reference is no longer valid (e.g., after page refresh).
NoSuchWindowException	Tried to switch to a non-existing window.
NoSuchFrameException	Tried to switch to a non-existing frame.
InvalidSelectorException	Selector syntax is invalid.
WebDriverException	General WebDriver error.
JavascriptException	Error executing JavaScript through Selenium.
ElementNotSelectableException	Tried to select a non-selectable element.
MoveTargetOutOfBoundsException	Tried to m he mouse to an out-of-bounds element.
NoAlertPresentException	Tried to interact with a non-existent alert.
InvalidArgumentException	Invalid argument passed (e.g., incorrect URL).
SessionNotCreatedException	WebDriver session could not be created

Continue generating

WebDriverException	General WebDriver error.
JavascriptException	Error executing JavaScript through Selenium.
ElementNotSelectableException	Tried to select a non-selectable element.
MoveTargetOutOfBoundsException	Tried to move the mouse to an out-of-bounds element.
NoAlertPresentException	Tried to interact with a non-existent alert.
InvalidArgumentException	Invalid argument passed (e.g., incorrect URL).
SessionNotCreatedException	WebDriver session could not be created.
NoSuchSessionException	Tried to interact with a closed or invalid session.
UnhandledAlertException	Tried to interact with the page while an alert is present and unhandled.

TimeoutException

- **Description**: Thrown when a command, like a wait or a page load, takes longer than the defined timeout period.
- **Example**: When an explicit wait times out while waiting for a condition.

NoSuchElementException

- **Description**: Thrown when the element being searched for is not found in the DOM.
- **Example**: When you try to locate an element using findElement() and the element does not exist on the page.

StaleElementReferenceException

- **Description**: Thrown when an element that was previously located becomes stale, usually because the DOM has been refreshed or the element has been removed.
- **Example**: When a page refreshes, or dynamic content changes, causing the reference to the previously located element to become invalid.

What is POM in selenium:

The **Page Object Model (POM)** is a design pattern in Selenium test automation.

It involves creating separate classes (known as page objects) for each web page or part of a web page that your tests interact with. Each page object class contains methods that correspond to actions that can be performed on the page, as well as locators for the elements on that page.

How Page Object Model Works:

1. **Create a Class for Each Page**:
    - For each web page or component, you create a corresponding page object class.

    - The class includes locators for elements on the page and methods to perform actions (e.g., clicking a button, entering text).
2. **Access Page Objects in Tests**:
    - The test scripts interact with the page classes to perform actions on the elements. These scripts do not dire

Advantages of Using Page Object Model:

1. **Improves Code Reusability**:
    - The methods in the page classes can be reused across multiple tests.
2. **Enhances Code Maintainability**:
    - If there are changes in the UI (like element locators), you only need to update the page object class, not every test that interacts with the page.
3. **Improves Readability**:
    - The test cases become more readable and concise since they directly call meaningful methods like loginPage.enterUsername() instead of writing driver.findElement() each time.
4. **Reduces Code Duplication**:
    - By creating reusable page objects, you avoid duplicating code related to interacting with web elements across multiple tests.
5. **Encourages Separation of Concerns**:
    - Test logic is separated from UI interaction logic, making both easier to manage.

How to scroll down to a page using java script selenium?

To scroll down a webpage using JavaScript with Selenium in Java, you can execute JavaScript directly via Selenium's JavascriptExecutor.

```
// Create an instance of JavascriptExecutor

JavascriptExecutor js = (JavascriptExecutor) driver;

// Scroll down by a specific number of pixels (e.g., 1000 pixels) js.executeScript("window.scrollBy(0, 1000);");
```

Explanation:

- window.scrollBy(0, 1000) scrolls the window down by 1000 pixels. The first parameter is for horizontal scrolling (0 in this case), and the second is for vertical scrolling.
- window.scrollTo(0, document.body.scrollHeight) scrolls to the very bottom of the page.

How to scroll down to particular method?

To scroll down to a particular element on the page using Selenium in Java, you can use JavaScript with Selenium's JavascriptExecutor. You'll need to locate the element on the page and then scroll the page until that element is in view.

// Scroll to the element using JavascriptExecutor JavascriptExecutor js = (JavascriptExecutor) driver; js.executeScript("arguments[0].scrollIntoView(true);", element);

arguments[0].scrollIntoView(true);: This JavaScript command scrolls the page until the element is in view.

Which are all file types used as data source for different frameworks?

Testing Frameworks

Frameworks: JUnit, TestNG, Selenium, Cucumber, etc.

- **Properties Files (.properties):** Used for configuration settings (key-value pairs).
- **XML Files (.xml):** Used for structured data and configuration (e.g., TestNG configurations).
- **JSON Files (.json):** Commonly used to store data for API testing or configurations.
- **CSV Files (.csv):** Used for data-driven testing (importing data from spreadsheets).
- **YAML Files (.yml/.yaml):** Often used in configuration (especially in Cucumber).
- **Excel Files (.xls/.xlsx):** Used for data-driven testing in frameworks like Selenium.

What are listeners in selenium?

In Selenium, **listeners** are interfaces that listen to specific events that occur during the execution of a test. They allow you to track or react to events like the start and end of a test, changes to elements, exceptions, and so on.

Types of Listeners in Selenium

There are two primary types of listeners in Selenium:

1. **WebDriver Event Listener**: Used to capture events related to WebDriver actions (e.g., click, change in element, etc.).
2. **TestNG/JUnit Listeners**: Used for capturing test-related events (e.g., when a test starts, passes, fails, or skips).

You need to implement the WebDriverEventListener interface and override its methods to capture different WebDriver events.

@Override

public void beforeClickOn(WebElement element, WebDriver driver) { System.out.println("Before clicking on: " + element.toString()); }

@Override

public void afterClickOn(WebElement element, WebDriver driver)

TestNG Listeners

- TestNG listeners allow you to listen to test-related events such as when a test starts, passes, fails, or skips.

TestNG listeners can be implemented by implementing specific interfaces such as:

- ITestListener
- IInvokedMethodListener
- ISuiteListener

@Override

public void onTestSuccess(ITestResult result)

{ System.out.println("Test Passed: " + result.getName()); }

@Override

public void onTestFailure(ITestResult result)

{ System.out.println("Test Failed: " + result.getName()); }

How to go back and forth to the page in selenium

In Selenium, you can navigate back and forth between pages using the navigate() method provided by the WebDriver interface. This is useful when you're testing scenarios where you need to go back to the previous page or move forward again (like using the browser's back and forward buttons).

Navigating Back and Forth

1. **Navigate Back**: Simulates the browser's back button.
2. **Navigate Forward**: Simulates the browser's forward button.

driver.navigate().back();

driver.navigate().forward();

How do you take screenshots in selenium webdriver?

In Selenium WebDriver, you can take screenshots by using the TakesScreenshot interface. This interface provides a method called getScreenshotAs(), which allows you to capture a screenshot and save it in various formats, typically as a file.

```
// Cast WebDriver object to TakesScreenshot

TakesScreenshot screenshot = (TakesScreenshot) driver;

// Capture the screenshot as a file

File srcFile = screenshot.getScreenshotAs(OutputType.FILE);

// Specify the destination path for the screenshot

File destFile = new File("path/to/screenshot.png");

// Copy the screenshot to the desired location

FileUtils.copyFile(srcFile, destFile);
```

Testing types supported by selenium

Functional testing, regression testing, cross browser testing, cross platform testing, smoke testing, E2E testing, data driven testing, integration testing, sanity testing.

Testing Type	Description
Functional Testing	Validates the functionality of the web application.
Regression Testing	Ensures new changes don't break existing functionality.
Cross-Browser Testing	Tests the application across different browsers.
Cross-Platform Testing	Tests the application on different operating systems.
Smoke Testing	Quick validation of critical functionalities.
End-to-End Testing	Tests the entire application workflow.
Data-Driven Testing	Runs tests with multiple sets of data.
Keyword-Driven Testing	Uses keywords to represent test steps.
BDD Testing	Automates tests using natural language scenarios.
UI Testing	Validates the front-end user interface.
Integration Testing	Ensures different modules work together.
Performance Testing	Basic performance checks like page load time.
Sanity Testing	Tests specific features after minor changes.

What do you mean by assertion in selenium?

In Selenium, **assertions** are used to validate whether the actual result of an action matches the expected result during test execution. Assertions help determine whether a test case has passed or failed by comparing expected and actual outcomes. If an assertion fails, it typically stops the execution of the test and marks it as failed.

Types of Assertions

Hard Assertions:

- **Definition**: A hard assertion causes the test execution to stop immediately if the assertion fails.

Soft Assertions:

- **Definition**: A soft assertion allows the test to continue execution even if the assertion fails. All failures can be reported after the entire test method execution is completed.

Common Assertion Methods in TestNG:

- Assert.assertEquals(actual, expected) - Verifies that two values are equal.
- Assert.assertTrue(condition) - Verifies that a condition is true.
- Assert.assertFalse(condition) - Verifies that a condition is false.
- Assert.assertNotNull(object) - Verifies that an object is not null

Soft Assertions in TestNG

In TestNG, soft assertions are available through the SoftAssert class, which allows multiple assertions in a single test method without immediately halting execution.

What are different build phases in maven

n Maven, the **build phases** are part of the **build lifecycle**, which is a series of steps that define how a project is built. Maven has three built-in **build lifecycles**:

1. **default** (the main lifecycle, used to build the project)
2. **clean** (to clean the project)
3. **site** (to generate project documentation)

Common Maven Commands:

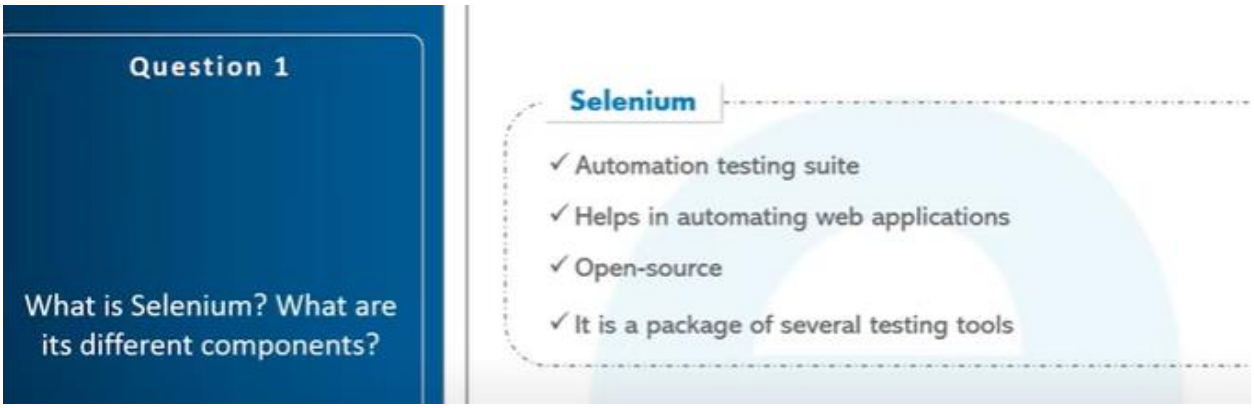
- mvn compile: Executes the compile phase and all preceding phases (like validate and process-resources).
- mvn test: Executes the test phase, which includes all phases up to and including test-compile.
- mvn package: Executes the package phase and all preceding phases, resulting in a JAR, WAR, or other packaged format.
- mvn install: Executes the install phase, installing the package to the local Maven repository.
- mvn clean: Executes the clean lifecycle, removing all previously generated build files.

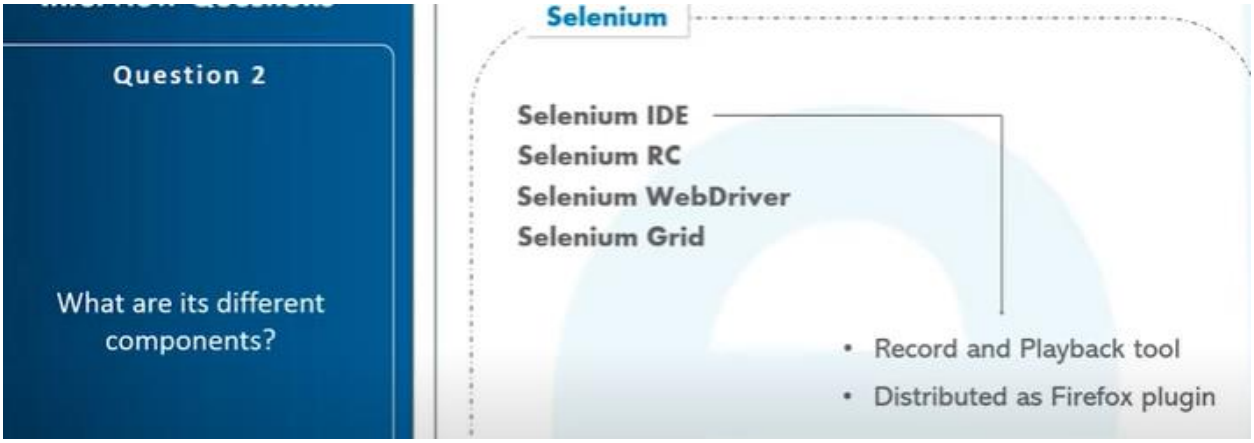
How will you handle keyboard and mouse actions using selenium

In Selenium WebDriver, **keyboard** and **mouse actions** can be handled using the Actions class.

Method	Description
`click()`	Performs a left mouse click.
`doubleClick()`	Performs a double-click.
`contextClick()`	Performs a right-click.
`moveToElement()`	Moves the mouse to a specified element.
`dragAndDrop()`	Performs a drag-and-drop action.
`clickAndHold()`	Clicks and holds the mouse button.
`release()`	Releases the mouse button.
`sendKeys()`	Sends a sequence of keystrokes to an element.
`keyDown()`	Simulates pressing down a key (e.g., `SHIFT`, `CTRL`).
`keyUp()`	Simulates releasing a key.
`scrollToElement()`	Scrolls to the specified element (Selenium 4).
`moveByOffset()`	Moves the mouse cursor by a specific x and y offset.
`perform()`	Executes all actions added to the chain.

`perform()`	Executes all actions added to the chain.
`build()`	Builds the chain of actions before performing them (optional in most cases).





Selenium IDE (Integrated Development Environment)

- **Purpose:** A browser extension (for Firefox and Chrome) that allows users to record and playback scripts.
- **Functionality:** Best for beginners, it helps create test cases through recording user interactions with the browser. The recorded scripts can be exported in various programming languages.
- **Limitations:** It's mainly for simple automation tasks and not suitable for complex tests or large test suites.

Selenium WebDriver

- **Purpose:** Provides APIs to interact with web browsers directly.
- **Functionality:** It allows developers to automate browser actions like clicking, typing, navigating, and more. It works across different browsers (Chrome, Firefox, Safari, Edge, etc.).
- **Language Support:** WebDriver supports multiple programming languages such as Java, C#, Python, Ruby, JavaScript, and more.

Selenium Grid

- **Purpose:** Used for parallel testing across different machines, browsers, and platforms.
- **Functionality:** Selenium Grid allows you to run tests on multiple machines at the same time, significantly reducing test execution time. It supports running tests on different browsers and operating systems in parallel.

Selenium RC (Remote Control)

- **Purpose:** Used to interact with browsers by injecting JavaScript into the web page.
- **Status:** Now deprecated and replaced by WebDriver due to its complex architecture and slower execution times.

Selenium Interview Questions

Question 3

What is a Selenium framework?

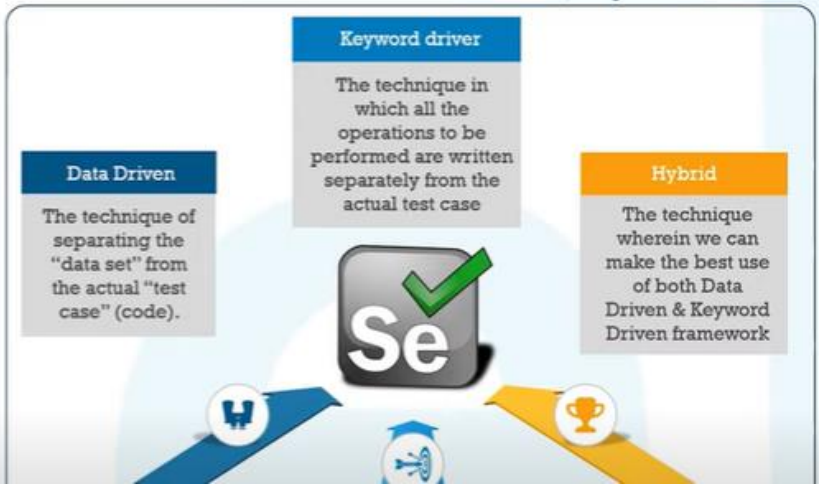
- It is a code structure for making code maintenance simpler, and code readability better.
- This framework involves breaking the entire code into smaller pieces of code, which test a particular functionality.
- It can also be structured in a way wherein, the test cases which need to be executed are called (invoked) from an external application

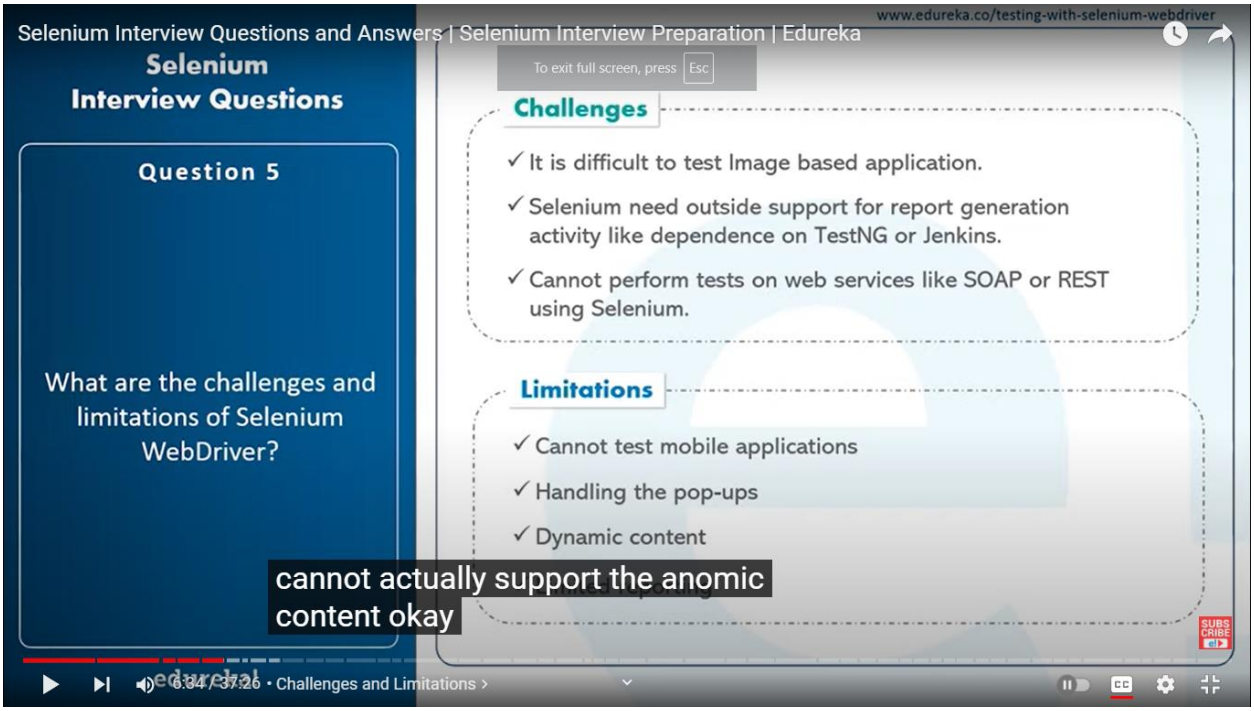
www.edureka.co/testing-with-selenium-webdriver

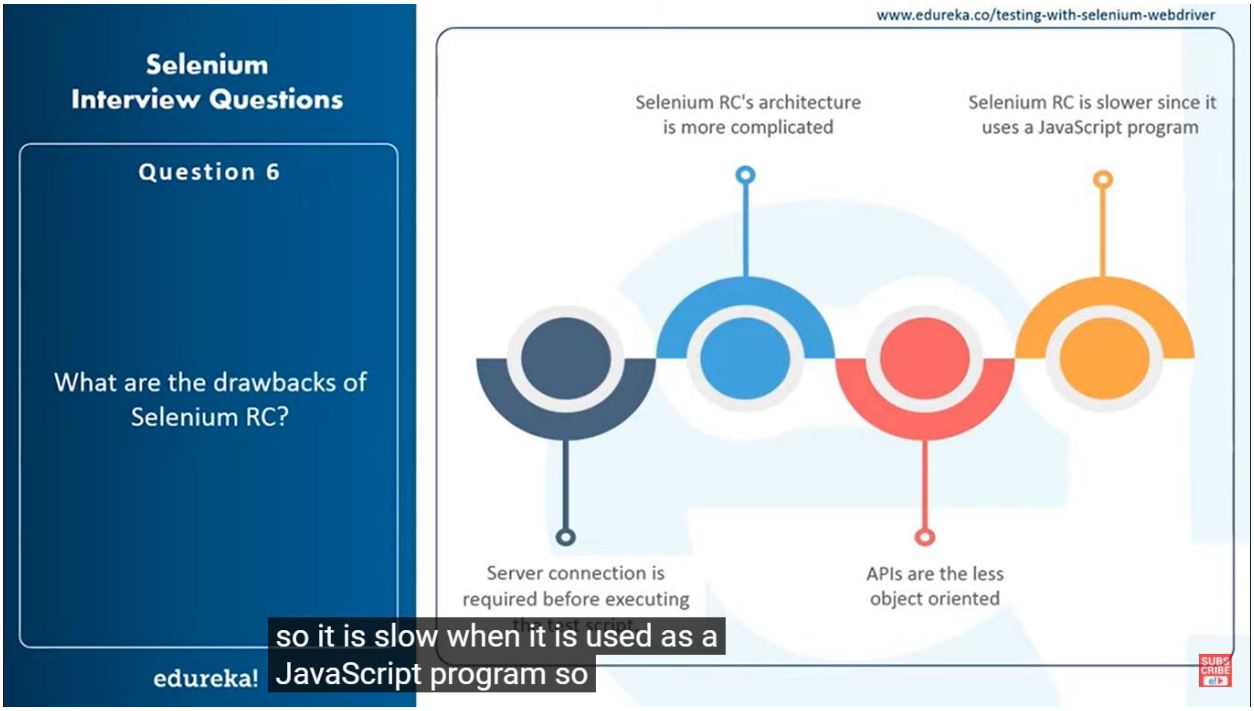
Selenium Interview Questions

Question 4

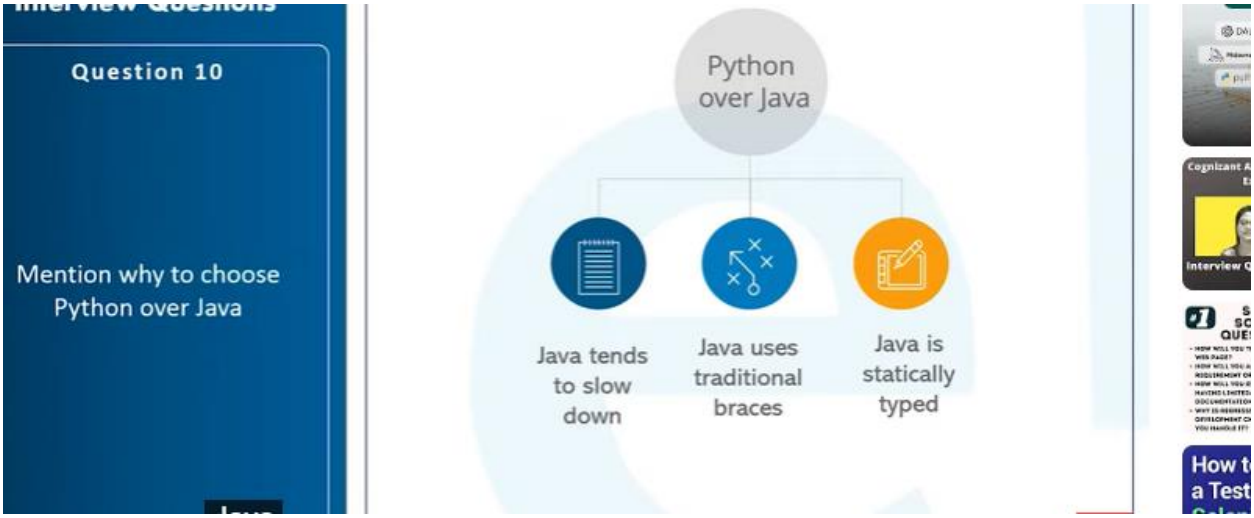
What are the different types of frameworks in Selenium?

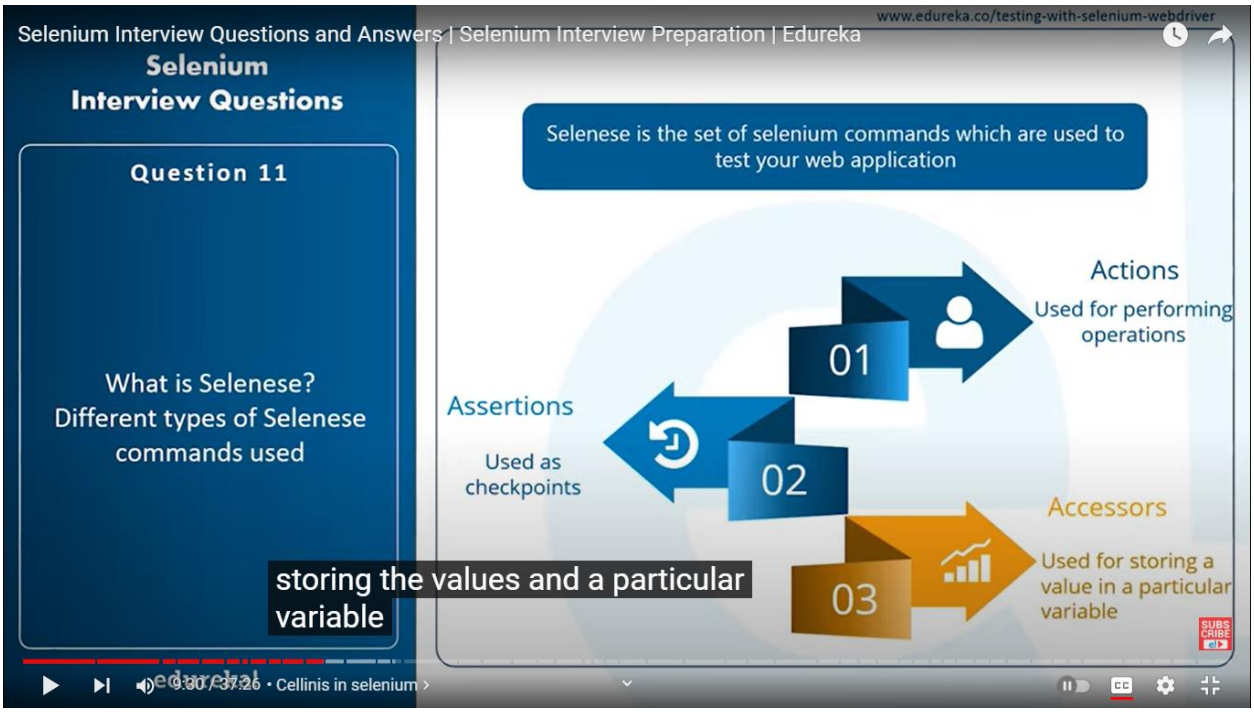


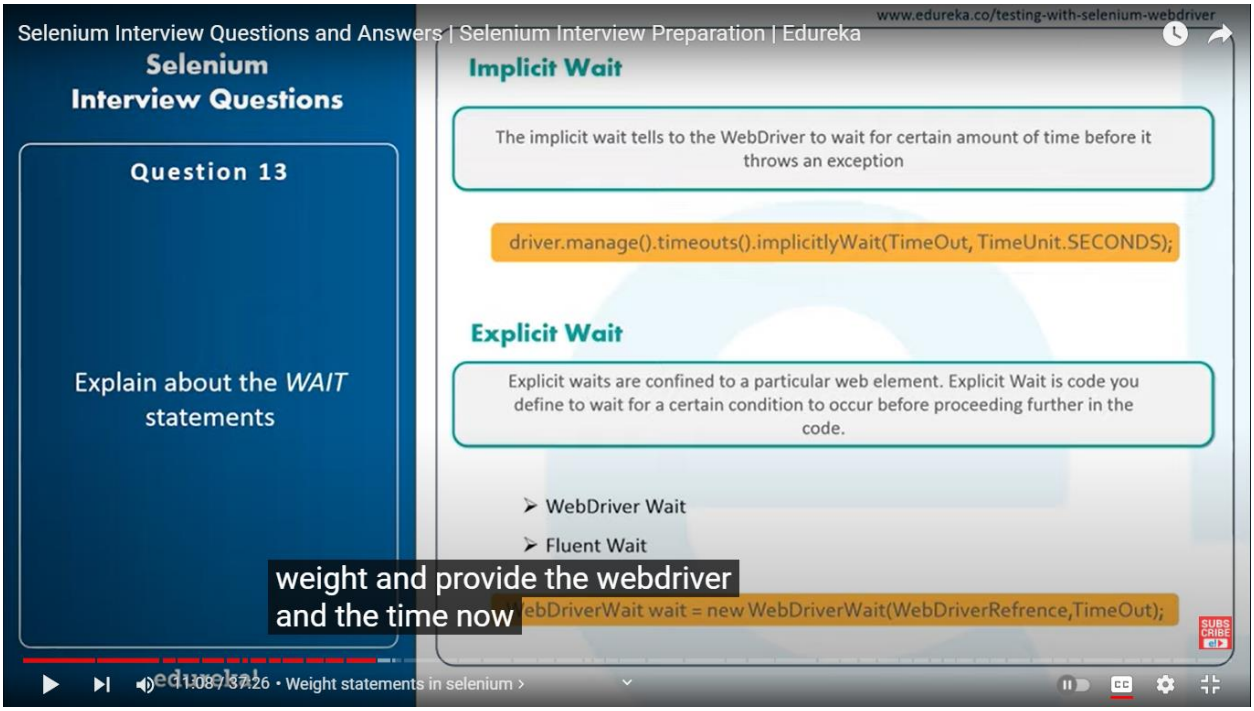

Selenium Interview Questions and Answers | Selenium Interview Preparation | Edureka
www.edureka.co/testing-with-selenium-webdriver
Selenium
Interview Questions
Question 5
What are the challenges and limitations of Selenium WebDriver?
To exit full screen, press Esc
Challenges
✓ It is difficult to test Image based application.
✓ Selenium need outside support for report generation activity like dependence on TestNG or Jenkins.
✓ Cannot perform tests on web services like SOAP or REST using Selenium.
Limitations
✓ Cannot test mobile applications
✓ Handling the pop-ups
✓ Dynamic content
cannot actually support the anomic content okay
6:04 / 37:26 • Challenges and Limitations ›

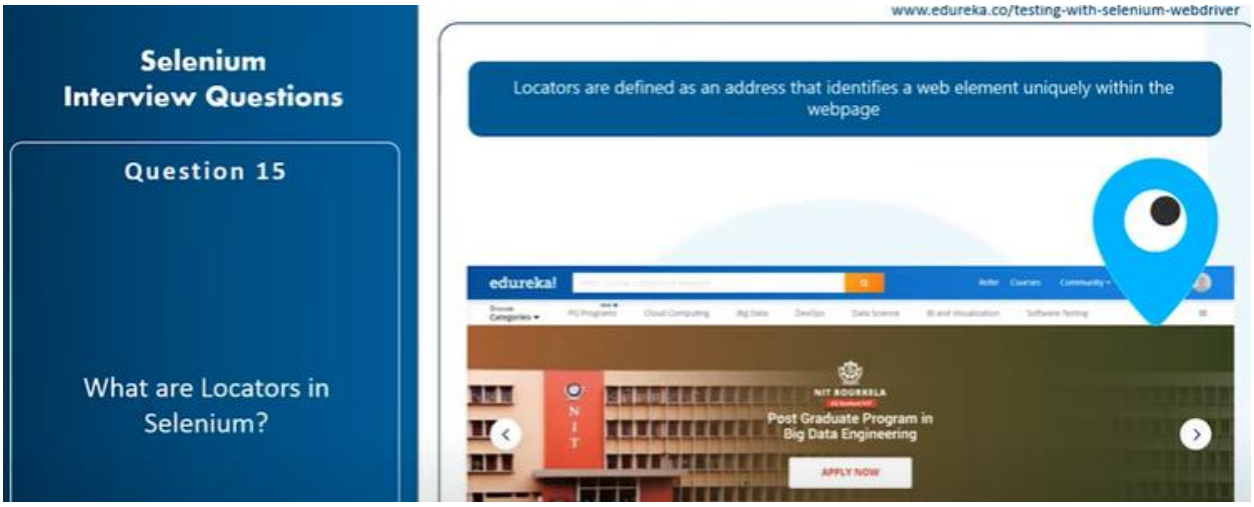

Selenium
Interview Questions
Question 6
What are the drawbacks of Selenium RC?
www.edureka.co/testing-with-selenium-webdriver
Selenium RC's architecture is more complicated
Selenium RC is slower since it uses a JavaScript program
Server connection is required before executing
APIs are the less object oriented
so it is slow when it is used as a JavaScript program so
edureka!

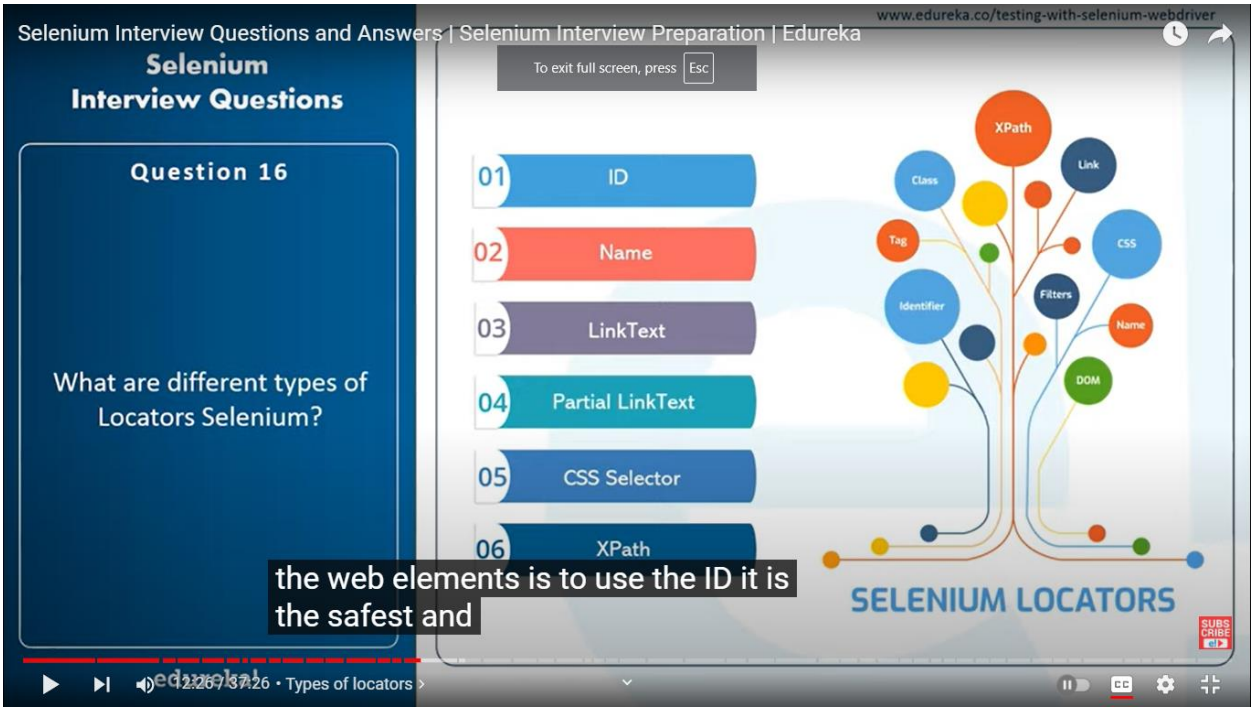

Question 8
Mention the capabilities of Selenium WebDriver
Handling multiple frames
Handling pop-ups
Handling browser windows
Alerts
Helps in Page navigation
Drag and drop
Supports Ajax based applications

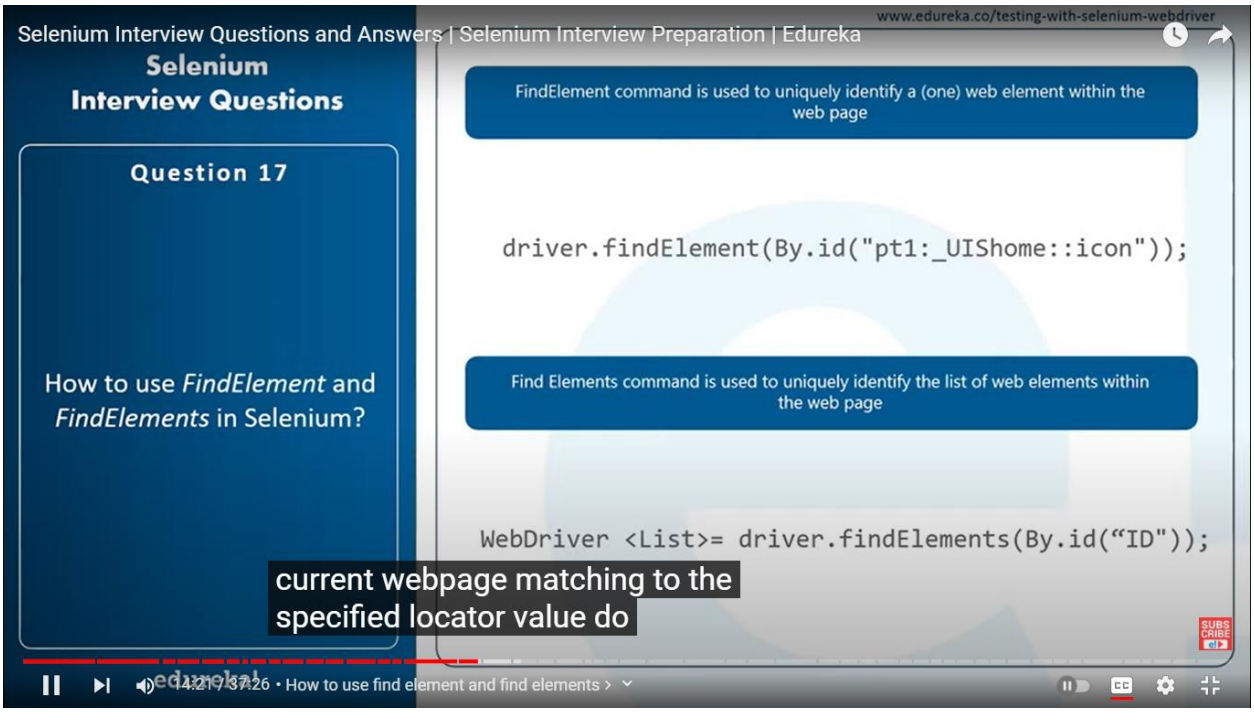

Question 10
Mention why to choose Python over Java
Python over Java
Java tends to slow down
Java uses traditional braces
Java is statically typed

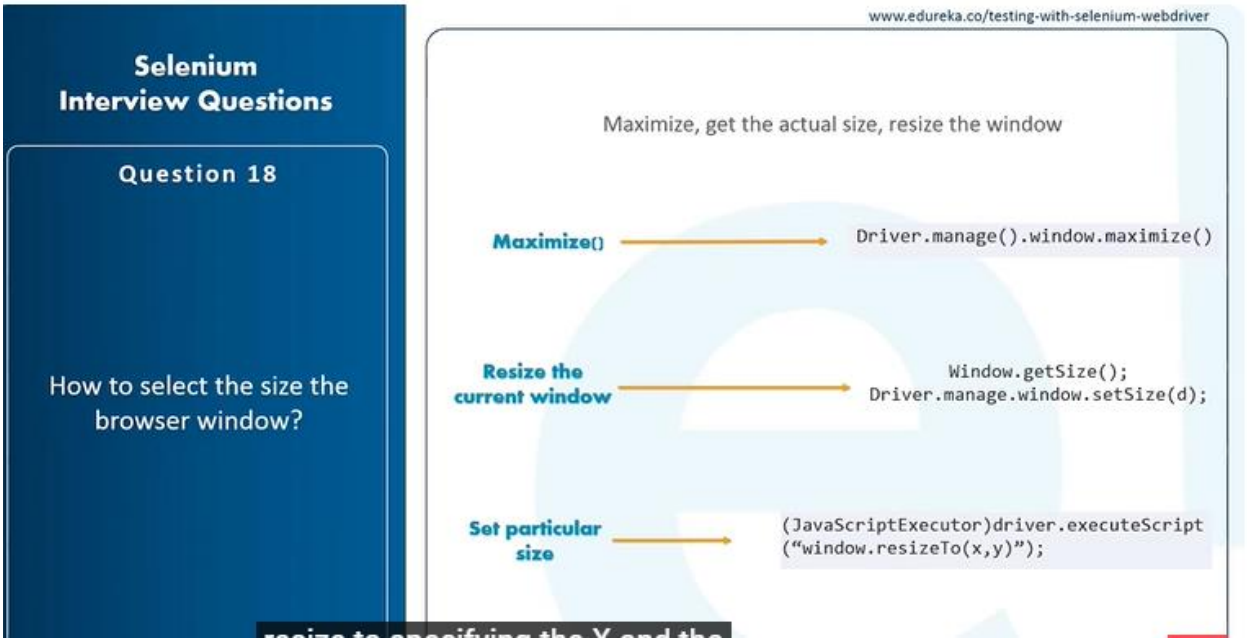

Selenium Interview Questions and Answers | Selenium Interview Preparation | Edureka
www.edureka.co/testing-with-selenium-webdriver
Selenium Interview Questions
Question 11
What is Selenese? Different types of Selenese commands used
Selenese is the set of selenium commands which are used to test your web application
Actions
Used for performing operations
01
Assertions
Used as checkpoints
02
Accessors
Used for storing a value in a particular variable
03
storing the values and a particular variable
0:30 / 37:26 • Cellinis in selenium ›


Selenium Interview Questions and Answers | Selenium Interview Preparation | Edureka
www.edureka.co/testing-with-selenium-webdriver
Selenium
Interview Questions
Question 13
Explain about the WAIT statements
Implicit Wait
The implicit wait tells to the WebDriver to wait for certain amount of time before it throws an exception
driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);
Explicit Wait
Explicit waits are confined to a particular web element. Explicit Wait is code you define to wait for a certain condition to occur before proceeding further in the code.
WebDriver Wait
Fluent Wait
ebDriverWait wait = new WebDriverWait(WebDriverRefrence,TimeOut);
weight and provide the webdriver
and the time now
11:08 / 37:26 • Weight statements in selenium


Selenium
Interview Questions
Question 15
What are Locators in Selenium?
www.edureka.co/testing-with-selenium-webdriver
Locators are defined as an address that identifies a web element uniquely within the webpage
edureka!
NIT ROURKELA
Post Graduate Program in
Big Data Engineering
APPLY NOW


Selenium Interview Questions and Answers | Selenium Interview Preparation | Edureka
www.edureka.co/testing-with-selenium-webdriver
Selenium
Interview Questions
Question 16
What are different types of Locators Selenium?
To exit full screen, press Esc
01 ID
02 Name
03 LinkText
04 Partial LinkText
05 CSS Selector
06 XPath
XPath
Link
Class
Tag
CSS
Identifier
Filters
Name
DOM
SELENIUM LOCATORS
the web elements is to use the ID it is the safest and
12:26 / 37:26 • Types of locators

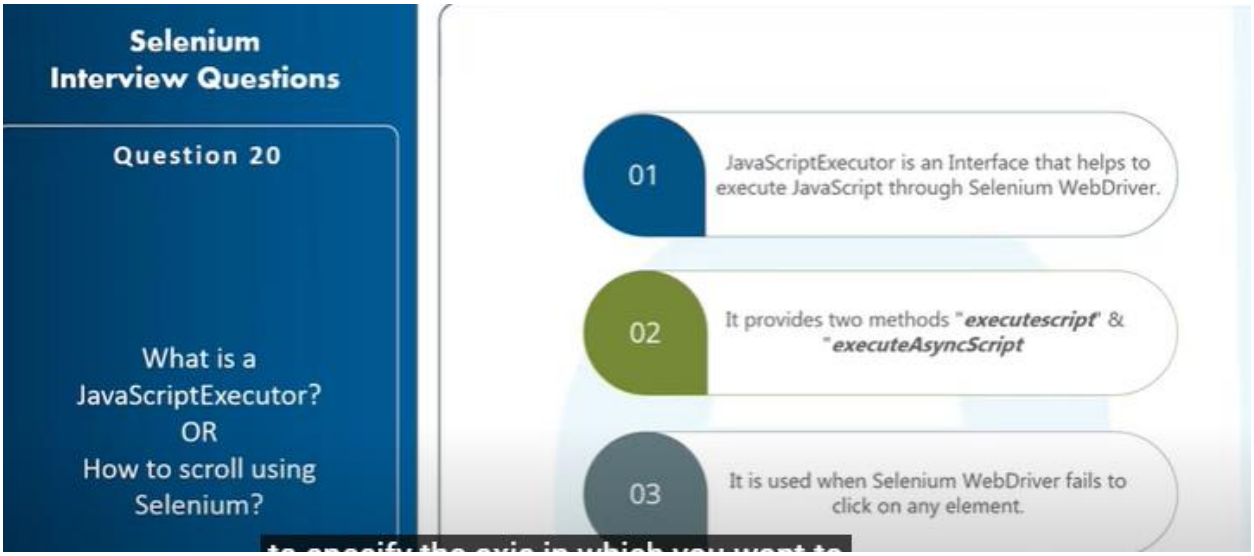

Selenium Interview Questions and Answers | Selenium Interview Preparation | Edureka
www.edureka.co/testing-with-selenium-webdriver
Selenium
Interview Questions
Question 17
How to use FindElement and FindElements in Selenium?
FindElement command is used to uniquely identify a (one) web element within the web page
driver.findElement(By.id("pt1:_UIShome::icon"));
Find Elements command is used to uniquely identify the list of web elements within the web page
WebDriver <List>= driver.findElements(By.id("ID"));
current webpage matching to the specified locator value do
14:21 / 37:26 • How to use find element and find elements

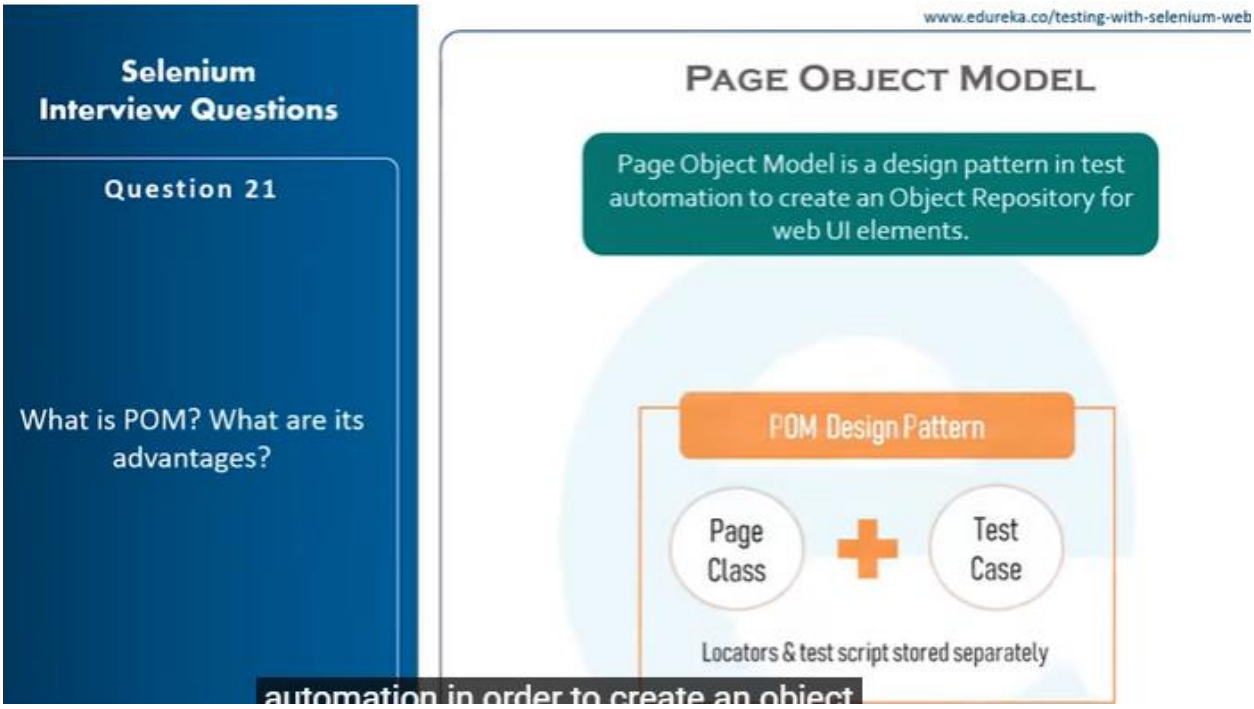


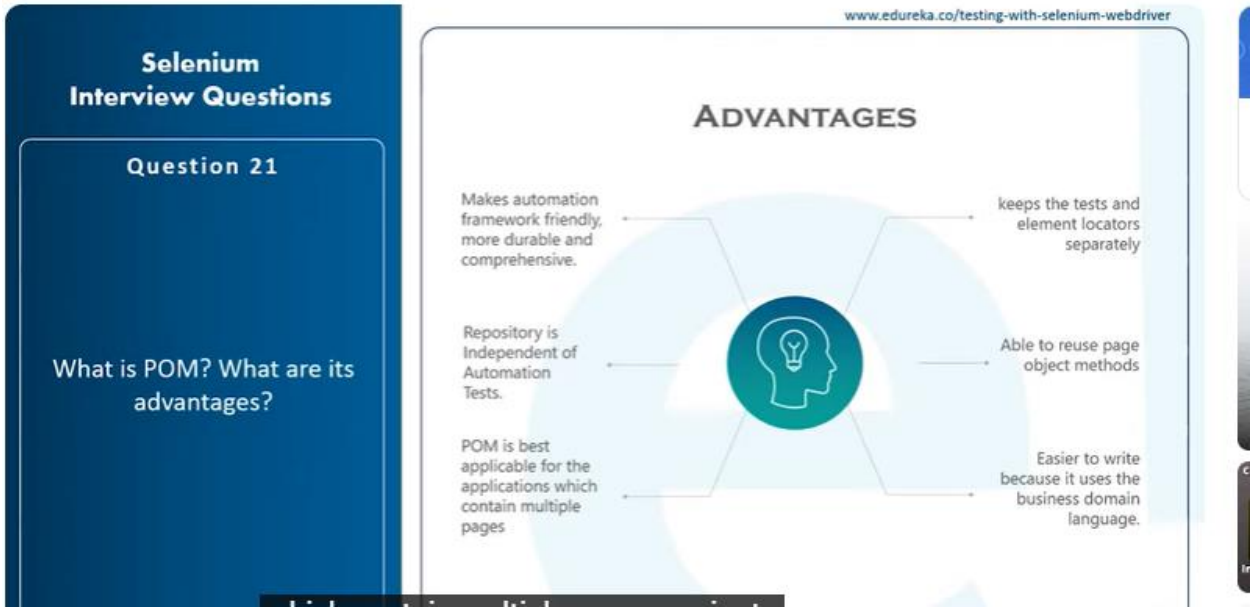
In Selenium, **JXL** (Java Excel API) is a third-party library used to interact with Excel files (specifically **.xls** format, as it does not support **.xlsx** files). It allows developers to read, write, and modify Excel spreadsheets in their Selenium test automation scripts.

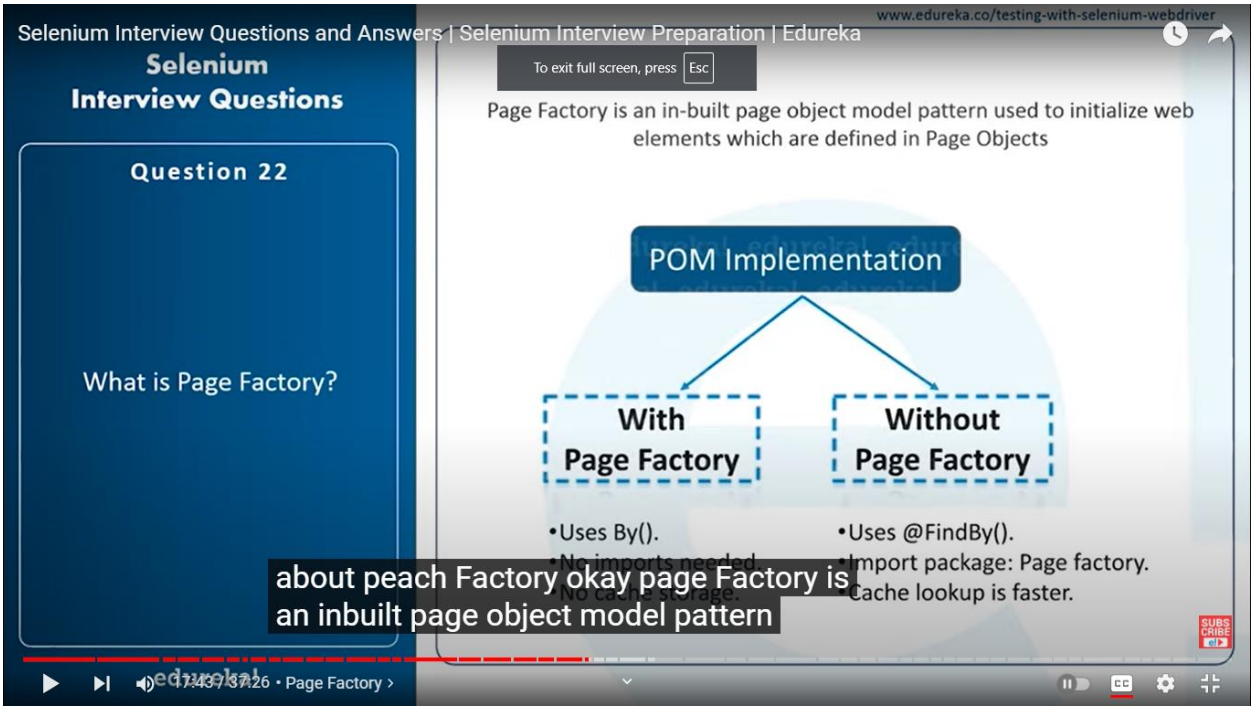
Apache POI (Poor Obfuscation Implementation) is a Java library used for reading, writing, and manipulating Microsoft Office files, such as Excel spreadsheets, Word documents, and PowerPoint presentations. It is widely used in Selenium for handling Excel files, especially for **data-driven testing**

where test data is stored in Excel.






Selenium
Interview Questions
Question 21
What is POM? What are its advantages?
www.edureka.co/testing-with-selenium-webdriver
ADVANTAGES
Makes automation framework friendly, more durable and comprehensive.
Repository is Independent of Automation Tests.
POM is best applicable for the applications which contain multiple pages
keeps the tests and element locators separately
Able to reuse page object methods
Easier to write because it uses the business domain language.


Selenium Interview Questions and Answers | Selenium Interview Preparation | Edureka
Selenium
Interview Questions
Question 22
What is Page Factory?
www.edureka.co/testing-with-selenium-webdriver
To exit full screen, press Esc
Page Factory is an in-built page object model pattern used to initialize web elements which are defined in Page Objects
POM Implementation
With Page Factory
Without Page Factory
•Uses By().
•Uses @FindBy().
•Import package: Page factory.
Cache lookup is faster.
about peach Factory okay page Factory is an inbuilt page object model pattern
17:43 / 37:26 • Page Factory ›

Selenium Interview Questions

Question 23

What is the difference between Page Object Model (POM) and Page Factory?

PAGE OBJECT MODEL	PAGE FACTORY
It is a class which represents the web page and holds the functionalities	It is a way to initialize the web elements within the page object when the instance is created

Selenium Interview Questions

Question 26

Can Selenium handle window pop-ups?

www.edureka.co/testing-with-selenium-webdrive

Selenium does not support handling pop-ups

Alert is used to display a warning message. It is a pop-up window that comes up on the screen

This page says

Question 26

Can Selenium handle window pop-ups?

This method is called when you click on the 'OK' button of the alert.

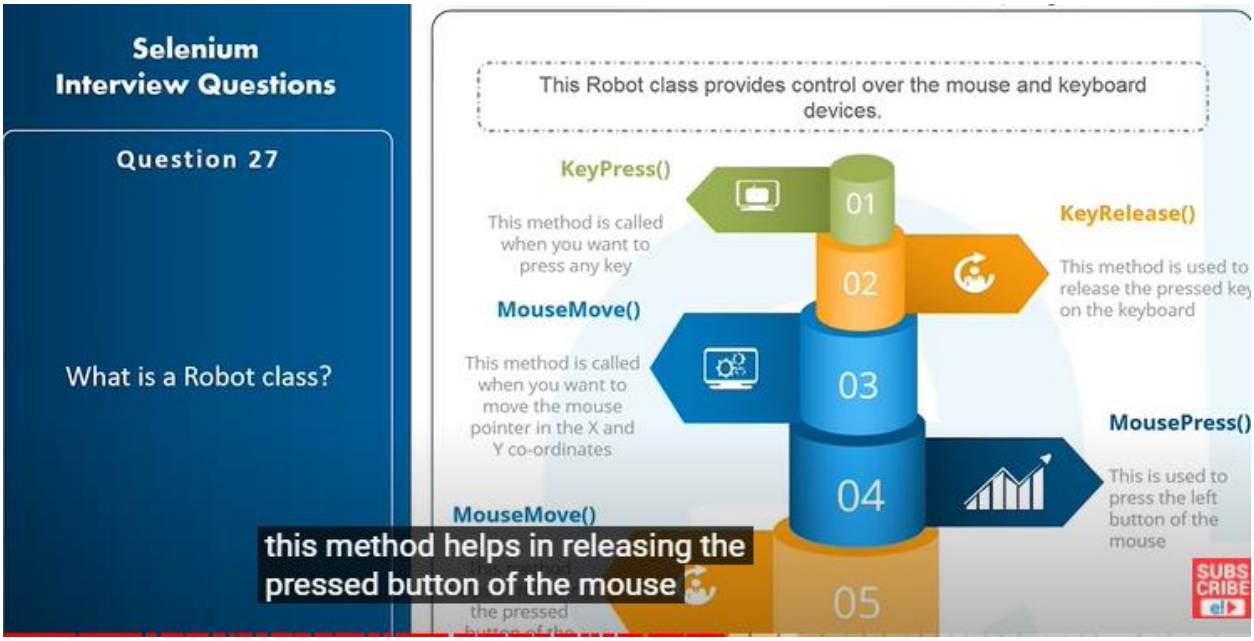
This is called when you want to send some data to alert box.

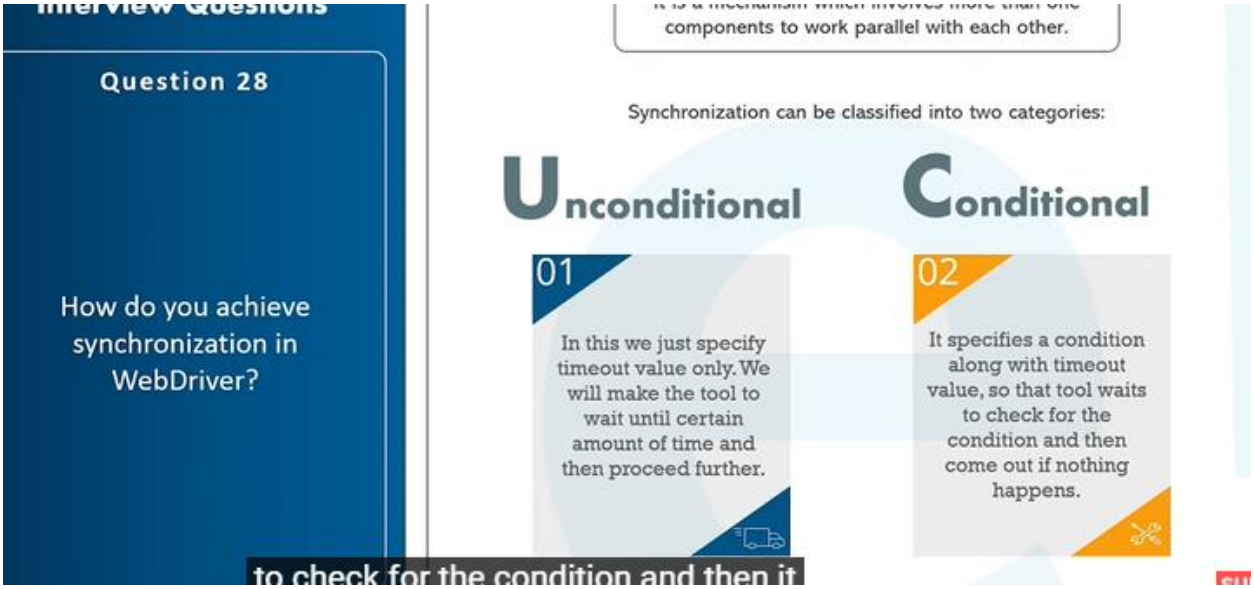
Void dismiss() | Void accept() | String getText() | Void sendKeys(String stringToSend)

This method is called when the 'Cancel' button is clicked in the alert box.

This method is called to capture the alert message.

what is void this


Selenium
Interview Questions
Question 27
What is a Robot class?
This Robot class provides control over the mouse and keyboard devices.
KeyPress()
This method is called when you want to press any key
KeyRelease()
This method is used to release the pressed key on the keyboard
MouseMove()
This method is called when you want to move the mouse pointer in the X and Y co-ordinates
MousePress()
This is used to press the left button of the mouse
MouseMove()
01
02
03
04
05
this method helps in releasing the pressed button of the mouse
SUBSCRIBE


Question 28
How do you achieve synchronization in WebDriver?
components to work parallel with each other.
Synchronization can be classified into two categories:
Unconditional
Conditional
01
In this we just specify timeout value only. We will make the tool to wait until certain amount of time and then proceed further.
02
It specifies a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.
to check for the condition and then it

Selenium Interview Questions

Question 29

How to take a screenshot in Selenium?

www.edureka.co/testing-with-selenium-webdrive

Selenium has provided *TakesScreenShot* interface by which we can use getScreenshotAs method which will help in capturing the entire screenshot in form of file then using FileUtils we can copy screenshots from one location to another location

```
File src= ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
try {
 // now copy the  screenshot to desired location using copyFile //method
FileUtils.copyFile(src, new File("C:/selenium/error.png"));
```

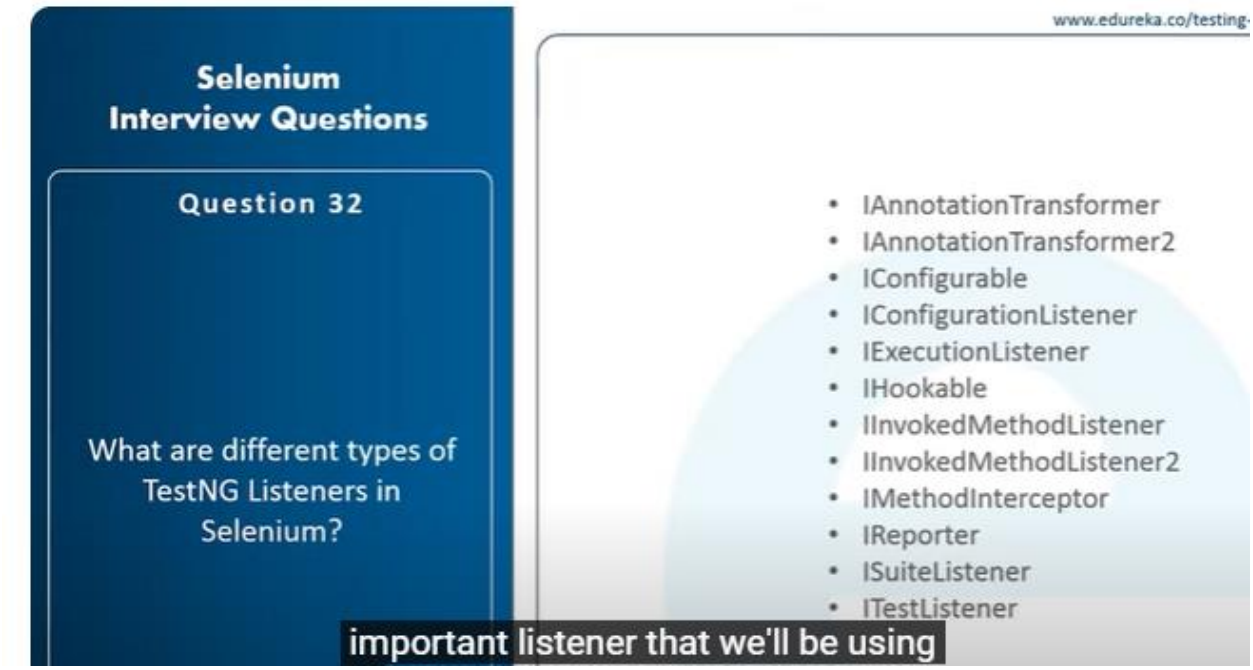
Selenium Interview Questions

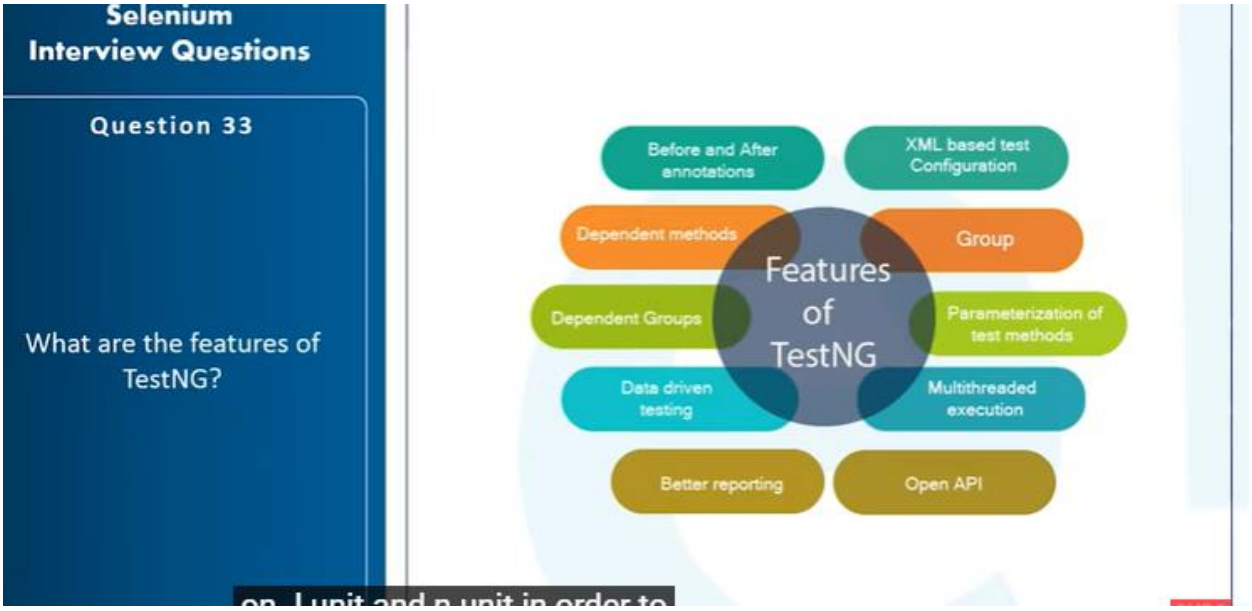
Question 30

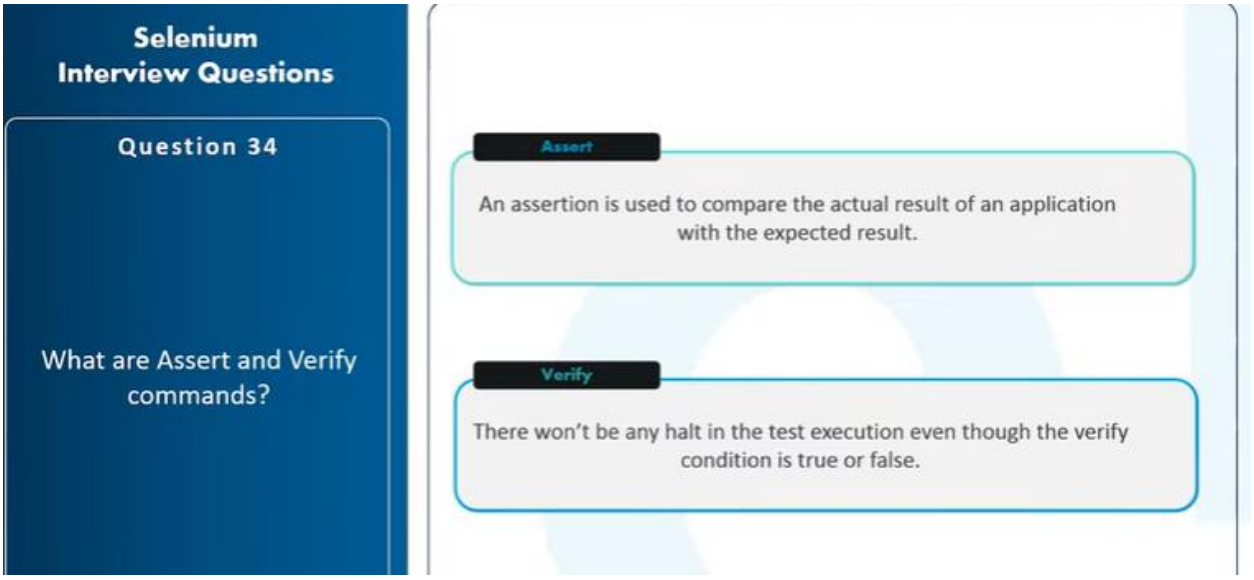
How to handle multiple windows in Selenium?

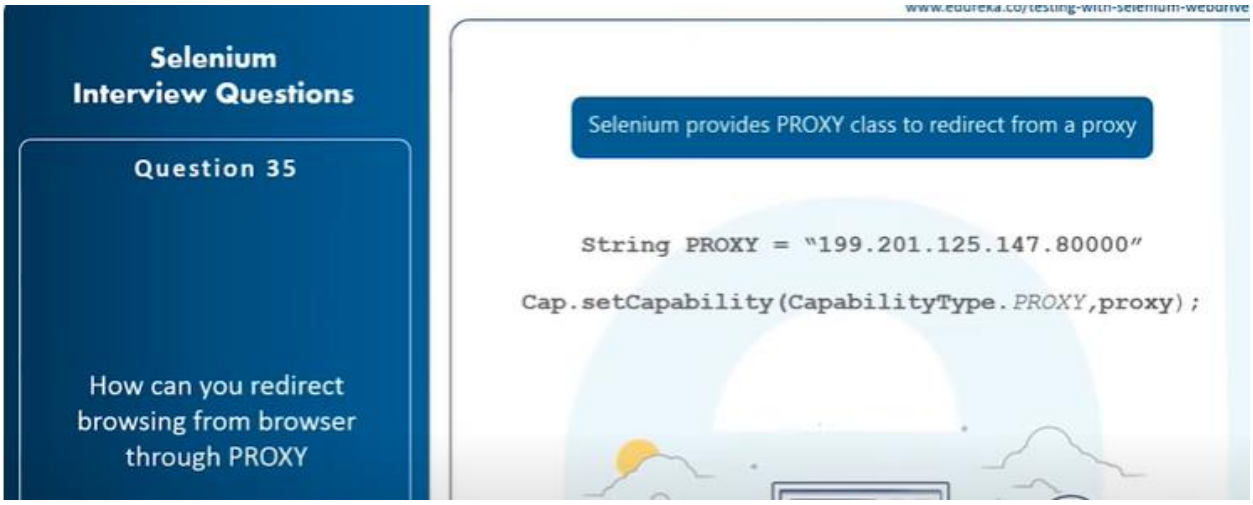
A window handle is a unique identifier that holds the address of all the windows. This is basically a pointer to a window, which returns the string value

- **get.windowhandle()**: helps in getting the window handle of the current window
- **get.windowhandles()**: helps in getting the handles of all the windows opened
- **set**: helps to set the window handles which is in the form of a string.
- **switch to**: helps in switching between the windows
- **action**: helps to perform certain actions on the windows.


Selenium
Interview Questions
Question 32
What are different types of TestNG Listeners in Selenium?
www.edureka.co/testing-
IAnnotationTransformer
IAnnotationTransformer2
IConfigurable
IConfigurationListener
IExecutionListener
IHookable
IInvokedMethodListener
IInvokedMethodListener2
IMethodInterceptor
IReporter
ISuiteListener
ITestListener
important listener that we'll be using


Selenium
Interview Questions
Question 33
What are the features of TestNG?
Before and After annotations
XML based test Configuration
Dependent methods
Group
Features of TestNG
Dependent Groups
Parameterization of test methods
Data driven testing
Multithreaded execution
Better reporting
Open API





How can you debug the tests in selenium ide

Debugging tests in **Selenium IDE** is an essential step in ensuring your test scripts run as expected. Selenium IDE provides several built-in features that help with debugging test cases effectively. Below are various ways to debug tests in Selenium IDE:

1. Using Breakpoints

- **What is it?** Breakpoints allow you to pause the execution of a test at specific steps. This is useful for inspecting the application state at a particular point in the test.

Steps:

1. Open Selenium IDE and load your test case.
2. In the test editor, locate the command where you want to set the breakpoint.

3. Right-click on that command and select **Toggle Breakpoint**. A red dot will appear next to the command, indicating the breakpoint.
4. Run the test. The execution will pause when it hits the breakpoint, allowing you to inspect the state of the web application.
5. You can then **Resume** or **Step through** the test commands to proceed.

What are regular expressions in selenium

In Selenium, **Regular Expressions (RegEx)** are used to match patterns within strings, particularly in commands like verifyText, verifyTitle, assertText, and others. RegEx allows you to create flexible and powerful text patterns to match dynamic content such as varying IDs, text, or URLs. This is especially useful when the exact text isn't known beforehand or when you need to match a portion of a string.

Selenium Interview Questions

Question 40

How can you achieve Database Testing using Selenium?

Selenium does not support Database Testing, still, it can be partially done using JDBC and ODBC

Selenium Interview Questions

Question 43

How can you prepare a customised HTML report in TestNG using Hybrid framework?

www.edureka.co/testing-with-selenium-webdriver

1. Junit with the help of an Ant

2. TestNG using inbuild default file

3. Also use XSL file

file to get the HTML report also you can

SUBS

Selenium Interview Questions

Question 44

How can you switch between the Frames?

```
driver.switchTo.frame
```

1. Number
2. Name or an ID
3. Location of previous element

previously found web element using the

SUBS

	A comparison between various automati		
Parameters	**Selenium**	**QTP**	**W**
Licence cost	Open source software	HP licensed software	O
Scripting language	Java, C#, Python, Ruby, PHP, Perl, JavaScript	VB Script	R
Operating system support	Windows, Linux, Solaris	Windows 8/XP/ Vista	W
Browser support	Firefox, IE, Opera, Safari, Chrome	IE, Firefox, Chrome	Fi S
Object recognition	Selenium IDE, Firebug, Firepath	Using Object Spy	O re
Script execution speed	Low	High	L
Non-browser based app support	No	Yes	N
Device support	Supports iOS and Android	iOS, Android, Blackberry, Windows	iO
Ease of support	User and professional community support is available	Has dedicated HP support	Li op
Script creation time	Time required is much more	Time required is less	Ti

Question 46

Difference between SetSpeed and Sleep method

`Thread.sleep()` — Pauses the execution for a specific period of time

`setSpeed()` — Delays the execution for a specific amount of time

Selenium Interview Questions

Question 47

How to submit a form in Selenium?

You can use the Submit button

You can Click

Selenium Interview Questions

Question 49

Can I navigate back and forth the webpage in Selenium?

Yes. You can navigate in the browser

01 `Driver.navigate.forward`

02 `Driver.navigate.back`

03 `Driver.navigate.refresh`

04 `Driver.navigate.to("url")`

How can we locate elements using their text in XPath?

Using the text() method: xPathExpression = //*[text()='username']

2. What is the fastest web driver implementation?

HTMLUnit Driver implementation is the quickest.

3. Explain how to assert the webpage's text using Selenium 2.0?

WebElement el = driver.findElement(By.id(“ElementID”))

String text = el.getText();

Assert.assertEquals(“Element Text”, text);

4. What is Object Repository?

An object repository is a necessary component of any UI automation. This is because it allows a developer to store all objects used in the scripts in one or more centralized locations rather than scattered throughout the test scripts.

How to send ALT/SHIFT/CONTROL keys simultaneously in Selenium WebDriver?

We hold down these keys and click other buttons to achieve unique functionality. To hold on to these keys, we need to define two more methods apart from keys.ALT or keys.SHIFT or keys.CONTROL functions. To include more than one key to be pressed or released, we can pass ALT, SHIFT, and CONTROL as parameters in the modifier key argument.

Here is the syntax for the same:

```
Actions a = new Actions(driver)

a.keyDown(Keys.ALT).keyDown(Keys.SHIFT) .keyDown(Keys.CONTROL).sendKeys('test') .build()
.perform();
```

6. How can you upload a file using Selenium WebDriver?

We can upload a file using the element.sendKeys (file path) command. But before doing so, we must use the HTML tag ‘input’ with the attribute type as a file. This helps Selenium identify the web element first and upload the file.

7. How can you use regular expressions in Selenium?

Regexp is a keyword that can be prefixed to a text to treat it as a regular expression in Selenium.

8. How can you handle colors in a web driver?

We can use the command getCSSValue(arg0) to fetch the colors by sending the color as the argument.

9. Which command takes you forward by one page in the browser history?

The navigate().forward() command takes you forward by one page (previous page) on the browser's history. This method lets the browser click the forward button in the existing browser window.

10. What is POM (Page Object Model)?

Every application webpage has a corresponding page class responsible for locating and performing actions on the web elements. POM is a design pattern that creates object repositories for web elements. It improves code reusability and readability. You can also run multiple test cases on the object repository.

11. How can you enable a disabled textbox?

We can enable a disabled HTML element by changing the attributes using the .removeAttribute("disabled") method.
Here is a code to allow a text box using the Javascript executor.

String enable ="doc.getElementsByName("name').removeAttribute('disabled');";

javascript.executeScript(enable);

12. How can we type text in a textbox element using Selenium?

We can type text in a textbox using the sendKeys() method.

Here is the code:

```
WebElement searchTextBox = driver.findElement(By.id("srch"));

searchTextBox.sendKeys("searchTerm");
```

13. How can you submit a form in Selenium?

We can submit a form in Selenium using the submit() method.

Here is the code:

```
driver.findElement(By.id("form1")).submit();
```

14. How do you take a screenshot with Selenium Webdriver?

We can use the TakeScreenshot function to capture the screenshot and getScreenshotAs() command to save the screenshot.

15. Which type command should you use if there is a reload event once your typing is completed?

The typeAndWait command can be used when a reload event in the software posts a typing action.

16. What's the difference between close and quit commands?

The distinction between close and quit commands is as follows: -

- driver.close() - Used to close the currently focused browser.
- driver.quit() - This function is used to close all browser instances.

17. How to delete cookies in Selenium?

Using the deleteAllCookies() method.

Here is the code:

driver.manage().deleteAllCookies();

18. Write the code to double-click an element.

Code to double-click an element: -

Actions action = new Actions(driver);

WebElement element=driver.findElement(By.id("elementId"));

action.doubleClick(element).perform();

19. How can we select elements by their attribute value using CSS Selector?

Using [attribute=value], we can select all elements of a specific class. For example, '[type=small]' will select the element with the attribute type of value'small'.

20. What is the fundamental difference between XPath and CSS selectors?

The primary distinction between XPath and CSS selector is that XPaths allow us to traverse up in the document, allowing us to move to parent elements. With CSS selectors, we can only move downwards in the document.

21. What is Desired capabilities of the selenium web driver?

Desired capabilities are a collection of key-value pairs used in browser instances to store or configure browser-specific properties such as version, platform, and so on.

22. How can we find all the links on a web page?

The anchor tag 'a' is used for all of the links. So, by looking for elements of tagName 'a,' we can find all of the links on a page.

Here is the code:

```
List<WebElement> links = driver.findElements(By.tagName("a"));
```

23. How can we capture screenshots in Selenium?

Using getScreenshotAs method of TakesScreenshot interface, we can take screenshots in Selenium

Here is the code:

```
File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(scrFile, new File("D:\\testScreenShot.jpg"));
```

24. How can we check if an element is getting displayed on a web page?

The isDisplayed method enables us to check if an element is displayed on a web page.

driver.findElement(By locator).isDisplayed();

25. Can a captcha be automated?

No, a captcha and barcode reader cannot be automated.

26. What is an Object repository?

An object repository is a centralized location for all test script objects or WebElements. Using the Page Object Model and Page Factory design patterns in Selenium, we can create an object repository.

27. What is a hybrid framework?

A hybrid framework is a combination of one or more frameworks. It is usually associated with a combination of data-driven and keyword-driven frameworks where the test data and test actions are kept in external files (in the form of a table).

28. When do we use Selenium Grid?

We use a selenium grid when we want to run the same or different test scripts on multiple platforms simultaneously. This helps us test under various environments, and the loss of time consumed for testing is reduced.

29. What is a node in a selenium grid?

Nodes are machines that are linked to the Selenium grid hub and host Selenium instances that run the test scripts. The selenium grid, unlike the hub, can have multiple nodes.

30. Explain the line of code Webdriver driver = new FirefoxDriver();

Webdriver driver = new FirefoxDriver(), refers to creating an object of type WebDriver by implementing an object of the FirefoxDriver class.

Selenium Coding Interview Questions for Experienced 10+ Years Developers

1. How can we move to the nth child element using XPath?

Using XPath, there are two ways to get to the nth element:

- Using index position and square brackets: For example, div[2] will locate the second div element.
- Position() is used as follows: For instance, div[position()=3] will locate the third div element.

2. How can the message in the alert box be retrieved?

The command storeAlert() can be used to retrieve the message from the alert pop-up and save it to a variable.

3. How can we move to the parent of an element using XPath?

Using the '..' expression in XPath, we can move to the parent of an element.

4. Can Selenium handle pop-up windows?

Selenium is an automation testing tool that can only work with web applications. As a result, Selenium cannot handle the pop-up in windows. However, by integrating with third-party tools, we can overcome this issue.

5. What exactly is the distinction between getwindowhandle() and getwindowhandles()?

- getwindowhandles(): This function retrieves the addresses of all open browsers and returns the data type <SetString>.
- getwindowhandle() returns the data type String and is used to obtain the address of the currently focused browser window.

6. How can we launch different browsers in the selenium web driver?

By creating an instance of a driver of a particular browser- WebDriver driver = new FirefoxDriver();

7. What are regular expressions, and how can you use them in Selenium?

Regular expressions are special text strings that are used to represent search patterns. Regexp is a keyword that can be used as a prefix to treat a text as a regular expression in Selenium.

8. Write the code to right-click an element.

Code to right-click an element in Selenium: -

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

WebElement element=driver.findElement(By.id("elementId"));

action.contextClick(element).perform();
```

9. How do you confirm the precise position of a web element?

The commands verifyElementPositionLeft and verifyElementPositionTop are used. These make use of pixel comparison to determine the position of the element from the left and top of the web page.

10. How can you redirect browsing from a browser through some proxy?

Selenium facilitates a PROXY class to redirect browsing from a proxy. Look at the example below.

Example:

```
String PROXY = "199.201.125.147:8080";

org.openqa. Selenium.Proxy proxy = new.org.openqa.selenium.Proxy();

proxy.setHTTPProxy(Proxy)

.setFtpProxy(Proxy)

.setSslProxy(Proxy)

DesiredCapabilities cap = new DesiredCapabilities();

cap.setCapability(CapabilityType.PROXY, proxy);
```

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

11. How to type text in a textbox using Selenium?

The sendKeys(“String to be entered”) enter the string in a textbox.

Syntax:

```
WebElement username = drv.findElement(By.id("Email"));

// entering username

username.sendKeys("sth");
```

12. List the automation tools that can be integrated with Selenium to achieve continuous testing.

Here are some of the automation tools to achieve continuous testing:

- Jenkins
- Travis CI
- CircleCI
- AWS CodePipeline
- Azure DevOps
- Bitbucket Pipelines

13. What are the different types of annotations which are used in Selenium?

The different types of annotations used in Selenium include:

- @Test - This is used to mark a method as a test method.
- @BeforeMethod - This annotation can execute a method before each test method.
- @AfterMethod - This is used to run a method following each test method.
- @BeforeClass - This annotation specifies that a method should be executed before the first test method.

14. When do you make use of AutoIT?

Selenium is only used to automate web applications. However, if we want to handle, manage, or maintain the GUI, HTML pop-ups, we must use AutoIT.

15. In the Selenium web driver, which API is used for database testing?

Selenium employs JDBC (Java Database Connectivity) for database testing. This enables us to create and execute SQL (Structured Query Language) queries.

16. Can you explain how the web driver handles colors?

By passing the color as an argument to the command getCssValue(arg0), we can retrieve the colors.

17. Which web driver implementation is the fastest?

HTML Unit Driver is the fastest web driver implementation. This is because this driver does not run the tests in the browser but instead runs only plain HTTP, which is faster than expected.

18. What exactly are enhanced privilege browsers?

Privileges browsers are similar to Proxy Injection in that they allow websites to do things that are not normally permitted.

19. What is the distinction between a get request and a post request?

A get request retrieves data from a server, whereas a post request submits data to a server.

20. How can we determine the value of an element's attributes such as name, class, and value?

Using getAttribute("{attributeName}") method, we can find the value of different attributes of an element e.g.-

String valueAttribute = driver.findElement(By.id("elementLocator")).getAttribute("value");

21. What is an implicit wait in Selenium?

An implicit wait is a type of wait that waits for a specified time while locating an element before throwing NoSuchElementException. By default, selenium tries to find elements immediately when required without any wait. So, it is good to use implicit wait. This wait is applied to all the elements of the current driver instance.

Syntax:

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

22. What expected conditions can be used in Explicit waits?

Some of the commonly used expected conditions of an element that can be used with explicit waits are:

-

- elementToBeClickable(WebElement element or By locator)
- stalenessOf(WebElement element)
- visibilityOf(WebElement element)
- visibilityOfElementLocated(By locator)
- invisibilityOfElementLocated(By locator)
- attributeContains(WebElement element, String attribute, String value)
- alertIsPresent()
- titleContains(String title)
- titleIs(String title)
- textToBePresentInElementLocated(By, String)

23. What is fluent wait in Selenium?

A fluent wait is a type in which we can also specify polling interval(intervals after which the driver will try to find the element) along with the maximum timeout value.

Syntax:

```
Wait wait = new FluentWait(driver) .withTimeout(20, SECONDS) .pollingEvery(5, SECONDS)
.ignoring(NoSuchElementException.class); WebElement textBox = wait.until(new
Function<webdriver,webElement>() {
public WebElement apply(WebDriver driver) { return driver.findElement(By.id("textBoxId")); }
}
);
```

24. What different keyboard operations can be performed in Selenium?

The different keyboard operations that can be performed in Selenium are:-

- .sendKeys(“sequence of characters”) - Used for passing character sequence to an input or textbox element.
- .pressKey(“non-text keys”) - Used for keys like control, function keys, etc, that are non-text.
- .releaseKey(“non-text keys”) - Used with keypress event to simulate releasing a key from keyboard event.

25. Write the code to double-click an element in Selenium?

Code to double-click an element in Selenium: -

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId"));
action.doubleClick(element).perform();
```

26. Write the code to right-click an element in Selenium?

Code to right-click an element in Selenium: -

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

WebElement element=driver.findElement(By.id("elementId"));

action.contextClick(element).perform();
```

27. How to mouse hover an element in Selenium?

Code to mouse hover over an element in Selenium-

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

WebElement element=driver.findElement(By.id("elementId"));

action.moveToElement(element).perform();
```

28. How can we fetch the title of the page in Selenium?

Using driver.getTitle(); we can fetch the page title in Selenium. This method returns a string containing the title of the webpage.

29. How can we fetch the page source in Selenium?

Using driver.getPageSource(); we can fetch the page source in Selenium. This method returns a string containing the page source.

30. How to verify tooltip text using Selenium?

Tooltips webelements have an attribute of type ‘title’. By fetching the value of the ‘title’ attribute, we can verify the tooltip text in Selenium.

Syntax:

```
String toolTipText = element.getAttribute("title");
```