**Selenium interview questions and answers**

1. **What is Selenium?**

Selenium is a software program which automates web applications and browsers, not desktop applications. It supports various OS platforms, including Linux, Mac, and Windows. Its tools are used basically for Regression and Functional Testing.

1. **What are the Components of Selenium?**

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

1. **What is Selenium IDE?**

Selenium IDE comes as a Firefox plugin. It is an integrated development platform to develop and deploy Selenium Test series. It is used to record, playback, edit, and debug the test cases.

1. **What is Selenium RC?**

* Selenium RC (Remote Control) is a testing tool to write automated UI tests of web applications in any language for any HTTP website through the JavaScript-based browser.

1. **What is Selenium WebDriver?**

Selenium WebDriver creates Test cases to run on several browsers, through WebDriver API commands and element locators. With WebDriver, we can use Testing Framework Annotations and Programming features to enhance Test cases.

It is built to provide more concise, easier programming UI along with facing some issues in Selenium-RC API. This tool is developed to support dynamic pages where elements may change without even reloading the page.

**6.What is Selenium Grid?**

With Selenium Grid, we can run Tests on various environments against several browsers. In other words, it supports distributed execution of Tests. We can run multiple tests against various machines at the same time in different operating systems and browsers.

1. **What is Apache Maven? How is it used in Selenium?**

Maven is a Java build management and project management component. With Maven, we can easily manage Java project builds. As a Java build management component, Maven enables us to manage the Selenium build with ease. It manages the documentation, build compilation, and other project related tasks of Selenium test project. With it, we can also create the accurate project structure, manage and add jar files in build path, etc.

1. **What is TestNG? What are its Usage in Selenium?**

Like NUnit or JUnit in Group Test cases TestNG is also a Testing Framework used to run parameterized Tests, Testbatches, and to provide in-depth HTML test reports.

1. **What are the Pros of Selenium?**

* Supports several programming languages to create Test scripts
* Open Source program
* Multi-browser support to conduct Test Scripts/Test Cases.
* Supports several OS platforms to initiate Testing.
* Enables Parallel Test execution.

**10.What are the Cons of Selenium?**

* No built-in support for Reporting Results
* No Tech Support
* Only for Browser/Web-based applications
* No support for new features

**11.how to open browser in selenium?**

**Syntax:**syntax to open browser using selenium is

WebDriver d=new ChromeDriver();//any browser

**12.how to navigate to a specific website?**

d.get(“link”);

13.how to maximize the opened website?

d.manage().window().maximize();

**14.how to find an element in selenium?**

Syntax:syntax to find a element is

d.findElement(“xpath”);

**15.how to enter text in the textbox?**

We can enter text in the textbox using sendkeys method

d.findElement(“xpath”).sendKeys(“data”);

16.what is thread.sleep()? in selenium

* As per the official definition from the [Oracle Java Documentation](https://docs.oracle.com/javase/tutorial/essential/concurrency/sleep.html), **Thread.sleep()** causes the current thread to suspend execution for a specified period.
* **Thread.sleep()** is not a Selenium wait, it is provided by Java.
* **Thread.sleep()** does not have a return type, and it returns void.

**Thread.sleep** methods throws **InterruptedException** when any other thread interrupts the current thread and should be handled by the throws method or try catch block.

Example:

public void threadTest() throws InterruptedException {

Thread.sleep(2000);

}

**17.what are the Limitations of Thread.sleep()**

Using **Thread.sleep()** frequently in an automation framework is not a good practice. If the applied sleep is of 5 secs and the web element is displayed in 2 secs only, the extra 3 secs will increase the execution time. And if you use it more often in the framework, the execution time would increase drastically.

You always have to guess and apply **Thread.sleep()** seconds in advance, as there is no guarantee that the web element would be discoverable under that specified time.

# 18. **What are the different types of wait available in Selenium?**

**Implicit wait:**

* Implicit [wait](https://www.selenium.dev/documentation/en/webdriver/waits/) tells the**web driver to wait** for a certain amount of time before throwing  an exception. In implicit wait, we give wait time **globally** and it will remain applicable to entire test script. WebDriver will wait for the element to load on the basis of time provided in wait condition. However, if web driver is unable to find an element in a given time, it will throw “**ElementNotVisibleException**“.

:

**Syntax:**

driver.manage().timeouts().implicitlyWait(12, TimeUnit.SECONDS);

**example:**

**public** **static** **void** main(String[] args) {

WebDriver d=**new** ChromeDriver();

d.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(6));

d.manage().window().maximize();

d.get("https://www.hyrtutorials.com/p/waits-demo.html");

d.findElement(By.*id*("btn1")).click();

d.findElement(By.*id*("txt1")).sendKeys("hyr");

}

**explicit wait:**

* Explicit [waits](https://www.selenium.dev/documentation/en/webdriver/waits/) are used to halt script execution untill a particular condition is met or the maximum time has elapsed. Unlike Implicit waits, Explicit waits are applied to a particular web element only.
* WebDriver will wait untill element became visible **(visibilityOfElementLocated(By.xpath(“Enter xpath”))** on the basis of time provided in wait condition **(WebDriverWait(driver, Duration.ofSeconds(20)))** . However, if web driver is unable to find an element in a given time, it will throw “**ElementNotVisibleException**“.

**Syntax:**

WebDriverWait w = new WebDriverWait(driver,);

w.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("<<xpath expression>>")))

**example:**

**public** **static** **void** main(String[] args)

{

WebDriver d=**new** ChromeDriver();

d.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

d.manage().window().maximize();

WebDriverWait w=**new** WebDriverWait(d,Duration.*ofSeconds*(30));

WebElement username=w.until(ExpectedConditions.*elementToBeClickable*(By.*xpath*("//input[@name='username']")));

username.sendKeys("Admin");

}

**fluent wait:**

The **Fluent Wait in Selenium** is used to define maximum time for the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an “ElementNotVisibleException” exception. It checks for the web element at regular intervals until the object is found or timeout happens.Syntax:

Wait<WebDriver> w = new

FluentWait<WebDriver>(driver).withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(3)).ignoring(NoSuchElementException.class);

Example:

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(5))

.ignoring(NoSuchElementException.class);

WebElement foo = wait.until(driver -> {

return driver.findElement(By.id("foo"));

});

**Handling elements in selenium**

**19.write an example code to handle textboxes in selenium?**

Handling textboxes in selenium can be done as

**public** **static** **void** main(String[] args) **throws** Exception

{

WebDriver d=**new** ChromeDriver();

//Hadndling TextBoxes

d.get("https://github.com//login");

Thread.*sleep*(2000);

d.manage().window().maximize();

By usrtxt1=By.*id*("login\_field");

WebElement usrtxt=d.findElement(usrtxt1);

**if**(usrtxt.isDisplayed())

{

**if**(usrtxt.isEnabled())

{

usrtxt.sendKeys("Renuka.Guggilla");

Thread.*sleep*(2000);

String enteredtext=usrtxt.getAttribute("value");

usrtxt.getAttribute("value");

System.***out***.println(enteredtext);

usrtxt.clear();

}

**else**

{

System.***out***.println("usernametxt is not enabled");

}

}

**else**

{

System.***out***.println("username text is not displayed");

}

**20.write an example code to handle dropdowns in selenium?**

Handling dropdowns can be done as

**public** **static** **void** main(String[] args) **throws** Exception {

WebDriver d=**new** ChromeDriver();

d.manage().window().maximize();

d.get("https://www.hyrtutorials.com/p/html");

Thread.*sleep*(2000);

WebElement CourseElement=d.findElement(By.*id*("course"));

Select CourseNameDropDown=**new** Select(CourseElement);

List<WebElement> CourseNameDropDownOptions=CourseNameDropDown.getOptions();

**for**(WebElement Option:CourseNameDropDownOptions)

{

System.***out***.println(Option.getText());

}

CourseNameDropDown.selectByIndex(1);

Thread.*sleep*(2000);

CourseNameDropDown.selectByValue("net");

Thread.*sleep*(2000);

CourseNameDropDown.selectByVisibleText("javaScript");

String selectedtext=CourseNameDropDown.getFirstSelectedOption().getText();

System.***out***.println("selected visible text-"+selectedtext);

}

21.write a code to handle windows in selenium?

Handling windows in selenium can be done as

WebDriver d=**new** ChromeDriver();

d.manage().window().maximize();

d.get("https://www.hyrtutorials.com/p/window-handles-practice.html");

Thread.*sleep*(4000);

String parentWindowHandle=d.getWindowHandle();

System.***out***.println("parentwindow handle"+parentWindowHandle);

Thread.*sleep*(2000);

d.findElement(By.*xpath*("//button[@id='newWindowBtn']")).click();

Set<String> WindowHandles=d.getWindowHandles();

**for**(String WindowHandle:WindowHandles)

{

**if**(!WindowHandle.equals(parentWindowHandle))

{

d.switchTo().window(WindowHandle);

Thread.*sleep*(2000);

d.manage().window().maximize();

Thread.*sleep*(2000);

d.findElement(By.*id*("firstName")).sendKeys("renuka");

Thread.*sleep*(3000);

d.close();

Thread.*sleep*(3000);

}

d.switchTo().window(parentWindowHandle);

}

d.switchTo().window(parentWindowHandle);

Thread.*sleep*(2000);

d.findElement(By.*id*("name")).sendKeys("guggilla");

Thread.*sleep*(3000);

d.quit();

**22.write a example program to handle mouse actions in selenium and explain bout each term**

* **Click** − Performs a Click. We can also perform a click based on coordinates.

Syntax: void click(WebElement onElement)

* **contextClick** − Performs a context click/right-click on an element or based on the coordinates

Syntax:void contextClick(WebElement onElement)

* **doubleClick** − Performs a double-click on the webelement or based on the coordinates. If left empty, it performs double-click on the current location.

**Syntax:** void doubleClick(WebElement onElement)

* **mouseDown** − Performs a mouse-down action on an element or based on coordinates.

**Syntax:** void mouseDown(WebElement onElement)

* **mouseMove** − Performs a mouse-move action on an element or based on coordinates.

**Syntax:** void mouseMove(WebElement toElement)

* **mouseUp** − Releases the mouse usually followed by mouse-down and acts based on co-ordinates.

**Syntax:** void mouseUp(WebElement onElement)

**Example program:**

public class Action

{

public static void main(String[] args)

{

WebDriver d=new ChromeDriver();

d.manage().window().maximize();

d.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));

d.get("https://book.spicejet.com/retrievebooking.aspx");

Actions act=new Actions(d);

act.moveToElement(d.findElement(By.xpath("//a[@id='Login']"))).build().perform();//Move the cursor

act.doubleClick(d.findElement(By.xpath("//a[@id='TravelAgentLoginText']"))).build().perform();//double click

act.contextClick(d.findElement(By.xpath("//a[@href=\"GetPage.aspx?pg=https://corporate.spicejet.com/Airports.aspx\"]")))

.build().perform();//RightClick

}

}

**23. What is the major difference between driver.close() and driver.quit()?**

driver.close()

This command closes the browser’s current window. If multiple windows are open, the current window of focus will be closed.

driver.quit()

 When quit() is called on the driver instance and there are one or more browser windows open, it closes all the open browser windows.

**24. What is an alternative option to driver.get() method to open an URL in Selenium Web Driver?**

An alternative option to the driver.get() method to open an URL in Selenium Web Driver is to use the driver.navigate().to() method.

### 25.Mention different ways of locating an element in Selenium?

The various ways of locating an element in Selenium are: by ID, by name, by classname, by tagname, by link text, by partial link text, by CSS selector, and by XPath**.**

**26.How to handle tables in selenium?**

Table handling in selenium can be done as

**public** **static** **void** main(String[] args) {

WebDriver d=**new** ChromeDriver();

d.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(40));

d.get("https://money.rediff.com/indices");

d.manage().window().maximize();

d.findElement(By.*id*("showMoreLess")).click();

WebElement webtable=d.findElement(By.*xpath*("//table[@id='dataTable']//tbody"));

List<WebElement> rows=webtable.findElements(By.*tagName*("tr"));

**int** rowscount=rows.size();

System.***out***.println("the number of rows present in the table are"+rowscount);

**for**(**int** i=0;i<rowscount;i++)

{

List<WebElement> columns=rows.get(i).findElements((By.*tagName*("td")));

**int** columnscount=columns.size();

**for**(**int** j=0;j<columnscount;j++)

{

String CellText=columns.get(j).getText();

**if**(CellText.equals("S&P BSE 500"))

{

System.***out***.println("prev close value is:"+columns.get(1).getText());

}

}

}

**27.What is the TestNG Framework?**

TestNG framework is a testing framework to perform tests in the java programming language. Moreover, the "**NG**" in TestNG abbreviates for "**Next *Generation***". Cedric Beust developed it and inspired by the JUnit and NUnit testing framework.

28. **How do you run the TestNG script?**

TestNG script is run by right-click on the **TestNG class -> Run As -> TestNG Test**.

**29.What are the types of annotations used in TestNG (In the sequence of execution/hierarchy)?**

There are nine types of annotations used in TestNG. In order of their execution sequence, they are as follows:

* *@BeforeSuite*
* *@BeforeTest*
* *@BeforeClass*
* *@BeforeMethod*
* *@Test*
* *@AfterMethod*
* *@AfterClass*
* *@AfterTest*
* *@AfterSuite*

**30**. **What are the categories of annotations in TestNG?**

TestNG annotations divide into three categories:

* **Precondition Annotations**: The annotations under this category execute before the test. It consists of the following annotations:
* @BeforeMethod
* @BeforeClass
* @BeforeSuite
* @BeforeTest
* **Test Annotations**: The annotations under this category are defined just before the test methods. Moreover, it consists of the following annotations:
* @Test
* **Postcondition Annotations**: The annotations under this category execute after the test methods. Additionally, it consists of the following annotations:
* @AfterMethod
* @AfterClass
* @AfterTest
* @AfterSuite

**31.Where is the index report generated and saved in TestNG?**

* The index report generates under the project folder and test-output subfolder. Moreover, this report is available as "***index.html***" by default.

**32.Why do we create the XML file in TestNG?**

We use the XML file in TestNG for many purposes. The TestNG XML file helps us:

* To run multiple tests in a single execution.
* it also helps us to include and exclude the test methods and groups.
* it also helps us to add dependencies in groups.
* it helps to run the test case methods through parameters.
* it assists in the execution of the parallel test execution

**33.what is data-driven testing**

**Definition:**  is a software testing method in which test data is stored in table or spreadsheet format. Data driven testing allows testers to input a single test script that can execute tests for all test data from a table and expect the test output in the same table. It is also called table-driven testing or parameterized testing.

**Explanation:**Data driven testing  is an automation testing framework in which input values are read from data files and stored into variables in test scripts. It enables testers to build both positive and negative test cases into a single test. Input data in data driven framework can be stored in single or multiple data sources like .xls, .xml, .csv and databases.

**Example :**

**To readdata**

**public** **static** **void** main(String[] args) **throws** Exception

{

FileInputStream fis=**new** FileInputStream("C:\\excel\_handling\\students\_info.xlsx");

XSSFWorkbook wb=**new** XSSFWorkbook(fis);

XSSFSheet sheet=wb.getSheetAt(0);

**int** rows=sheet.getLastRowNum();

**int** cols=sheet.getRow(0).getLastCellNum();

List<List<String>> list=**new** ArrayList<List<String>>();

String [][] arr=**new** String[rows+1][cols];

**for**(**int** r=0;r<=rows;r++)

{

List<String>rowlist=**new** ArrayList<String>();

**for**(**int** c=0;c<cols;c++)

{

String value=sheet.getRow(r).getCell(c).getStringCellValue();

//System.out.println(value);

rowlist.add(value);

arr[r][c]=value;

}

//System.out.println(list);

System.***out***.println(rowlist);

}

//System.out.println(list);

System.***out***.println("the data in the form of an array is");

//System.out.println(arr);

**for**(**int** r=0;r<rows;r++)

{

**for**(**int** c=0;c<cols;c++)

{

System.***out***.println(arr[r][c]);

}

}