1. What is selenium webdriver?

* It is an API, it is used to test the web based applications.
* It supports multiple browsers like Mozilla, Chrome, IE, HtmlUnit driver, opera etc.
* It supports multiple languages like java, C#, Python, PHP, Pearl etc.
* It supports multiple operation systems like windows, Linux, apple OS etc.

2. What are the limitations of Selenium webdriver?

* It supports web-based applications only, not support window based application.
* It doesn’t have any in-built report generating capability, so we need plug-in’s either JUNIT or TESTNG to generate the test reports.
* At least we should have any one of the supported language in order to automate the test cases.

3. What are the different types of drivers available in selenium?

* Mozilla, Chrome, IE, HtmlUnit Driver, safari driver etc.

4. What are the testing types supported by selenium?

* It supports Functional testing and regression testing.

10. What are the element locaters in selenium?

* Id, name, class name, TagName, Link Text, Partial link Text, xpath, css selector.

5. What are the types of xpath’s in selenium?

* Absolute xpath and Relative xpath.

8. Write any one xpath expression?

* .//\*[@id=”submit Button”]
* .//\*[@id=”submit Button”]

6. What is the difference between absolute xpath and relative xpath?

* Absolute Xpath
* It starts the selection from the document node, it is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.
* Absolute xpath starts with “/ “(single forward slash).
* Relative xpath
* For Relative Xpath the path starts search from the middle of the HTML DOM structure. Which means it can search the element anywhere in the webpage.
* Relative xpath starts with // (double forward slash).

7. What is the difference between “/” [single slash] and “//” [double slash]?

* / is to starts search exact next node of a particular node in the HTML content
* Absolute xpath starts with single slash “/”
* // is to starts search selection anywhere in the HTML content
* Relative xpath starts with double slash “ // “
* Double slash represents the all the elements and single slash represents next element

9. How to handle dynamic xpath and write some expressions?

Ans: We have some possible ways to handle dynamic xpath based element, we can use some methods and we can solve it.

* Contains Method
* //input [contains (text()=’submit Button’)]
* Starts-with method
* //input [Starts-with (@text(), ’btn’)]

10. How to read the data from excel sheet?

Ans: By using JXL or Apache poi jar, Workbook class as well as sheet based on the rows and columns numbers we can get the data from excel sheet.

* FileInputStream Excelfile = new FileInputStream(Path of the excel file);
* ExcelWorkbook = new ExcelWorkbook(Excelfile);
* excelWorksheet sh= ExcelWorkbook.getSheet(sheetname);
* Cell=sh.getrow(rownum).getcell(colnum);
* String str=cell.getStringCellvalue();
* System.out.println(“str”);

11. How to enter the value in the textbox?

* Driver.findElement(By. Xpath(“ “).sendKeys(“12345”);

12. How to return entered data in textbox?

* webElement textBox = driver.findElement(By.Xpath(“ “);
* textbox.sendKeys(“12345”);
* textbox.getAttribute(“value”);

13. How to handle dropdown in the webpage?

* WebElement element=driver.findElement(By.xpath(“ “);
* Select dropDown = new select (element);
* dropDown.selectByvalue(“India”);
* OR
* dropDown.selectByIndex(2);
* OR
* dropDown.selectByvisibleText(“India”);

14. How to take the screen for the webpage using selenium?

TakeScreenshot ts= (TakeScreenshot) driver;

File file=ts.getScreenshotAs(outputType.FILE);

FileUtils.copyFile(sourcefile,destinationfile);

Sourcefile = file

Destinationfile = new File(“./screesnhots/Image.png”) //create a folder with screenshots and store the file in png/desired format

15. How to get particular cell value from the html table or text from the webpage?

* driver.findElement(By.xpath(“ “).getText();

16. How to handle alerts in the webpage?

* We can use Driver.switchTo().alert(); to switch to alert.
* And we can able to perform accept() method to click on ok button
* And we can able to perform dismiss() method to click on cancel button.
* Alert alert = driver.switchTo().Alert();
* alert.Accept();
* alert.dismiss();

17. How to return the text present on the alert window?

* Alert alert=Driver.switchTo().Alert().getText();

(or)

* alert.getText();

18. How to select the radio button, how to verify radio button is selected or not?

* driver.findElement(By.xpath(“ “).click() =======for select radio button
* Boolean RB=driver.findElement(By.xpath(“ “).isSelected();
* System.out.println(“RB”);

19. How to click on hyperlinks?

* Using link Text we can click.
* Driver.findElement(By.linkText(“ “).click();

20. How to find more than one element in the list?

* List <webElement> list=Driver.findElements(By.xpath(“ “);
* Int I = list.size();
* System.out.println(i); OR
* System.out.println(list.size());
* list.get(0).click();

21. How will you find element is displayed in the screen?

* Boolean BL=Driver.findElement(By.linkText(“ “).isDisplayed();
* System.out.println(“BL”);

22. What are the diff types of navigation commands?

* Driver. navigate().to(url);
* Driver. navigate().back();
* Driver. navigate().forward();
* Driver. navigate().refresh();

23. How to run tests using IE, Chrome browsers or How to do cross browser testing?

Ans: Firefox browser is the default browser for selenium so we don’t need to do any settings to execute tests in Firefox.

* System.setProperty(“Webdriver.ie.driver”, “c:\\program files\\IEdriverServer.exe”);
* System.setProperty(“Webdriver.Chrome.driver”, “c:\\program files\\chromedriver.exe”);

24. What are the generic actions in selenium?

* Mouse Hover, drag And Drop

25. How to perform mouse Hover action?

* WebElement element=driver.findElement(By.xpath(“ “);
* Actions act = new Actions(driver);
* Act.moveToElement(element).build().perform(); or click().Perform();

26. What is the difference between testNG and JUNIT?

* TestNG support group test execution but it is not supported in JUnit.
* TestNG support @BeforeTest, @AfterTest, @BeforeSuite, @AfterSuite, @BeforeGroups, @AfterGroups which is not supported in JUnit.
* Test prioritization, parallel testing is possible in TestNG. It is not supported by JUnit.

27. How to perform drag and drop action on the web page?

* WebElement source=driver.findElement(By.xpath("");
* WebElement target=driver.findElement(By.xpath("");
* Actions act=new Actons(driver);
* act.clickAndHold (source).moveToElement(target).release(target).build().

perform(); OR

* Act.dragAndDrop(source,target).build().Perform();

28. How to handle multiple windows?

Parent window to child window:

* String parent=driver.getWindowHandle();
* Set<String> handles=driver.getWindowHandles();
* for (String str:handles)
* driver.switchTo().window(str);

29. How to handle frames?

* Html frames are used to divide our browser window in to multiple sections, where each section can load a separate html document.
* Switch to frame is two ways
* Using frame index and using frame name
* Driver.switchTo().frame(int index);
* Driver.switchTo().frame(String name);

Switch from window to frame (frame name);

* Driver.switchTo().frame(“Frame name”);
* Driver.findElement (By. Xpath(“”)).click();
* Switch to top window:
* Driver.switchTo().defaultContent();

30. How to perform right click on the webpage?

* Actions action = new actions(driver);
* Action.contextClick(productLink).sendKeys(Keys.ARROW\_DOWN).sendKeys(Keys.ARROW\_DOWN).sendKeys(Keys.RETURN).build().perform();

31. What is the difference between implicit wait and Explicit Wait?

Implicit wait:

* It is to tell webdriver to pull the Dom for certain amount of time, when trying to find an element or element if there are not immediately.

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

Explicit Wait:

* Explicit wait define to wait for a certain condition to occur before proceeding the further in the code.
* Webdriverwait wait = new webdriverwait(20, driver);
* WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("someid")));
* Element.click();
* Worst case it is thread. Sleep();
* Thread.sleep(5000);
* This sets the condition to exact time period to wait.

Ex: WebDriverWait wait = **new** WebDriverWait(driver,Duration.*ofSeconds*(30));

WebElement txtbox2 =wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*id*("txt2")));

Fluent Wait**:**

Let’s say you have an element which sometime appears in just 1 second and some time it takes minutes to appear. In that case it is better to use fluent wait, as this will try to find element again and again until it find it or until the final timer runs out.

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

.withTimeout(30, TimeUnit.SECONDS)

.pollingEvery(5, TimeUnit.SECONDS)

.ignoring(NoSuchElementException.class);

32. What is the difference between drivers.Close () and driver. Quit ()?

Driver .Close ()

* It closes the current window where the exactly webdriver is pointing.

Driver .quit ()

* It closes all the associated windows, pages with parent window.

33. Explain about testNG?

* It is a unit testing framework.
* Used to generates the test reports
* It is developed in java language.
* It is a jar file.
* Need testNG to define the order of execution using **priority** keyword.
* We can prioritize the test using **priority** keyword.
* Need testNG to define the controller between different folder structures in the framework.
* It is used for controlling, monitoring and the way of execution of automation scripts in organized way.
* It connects all the folders as packages in an organized order and run in the framework level.
* The order of execution is defined with help of annotations in testNG.
* Annotations are denoted with the help of **@** symbol
* TestNG doesn’t require main method for execute the scripts.

34. What are all the annotations in testNG?

* @Bforeclass @Beforemethod @Beforesuite @Beforegroup
* @Afterclass @Aftermathod @Aftersuite @aftergroup
* @Test @Beforetest @Aftertest @Parameters
* @Listeners @Factory @Dataprovider

35. EXPLAIN ABOUT EACH ANNOTATION?

* @test: it represents the functional test case.
* @Before class: it will be run before the first test method in the current class is invoked
* @After class: it will be run after the last test method in the current class is invoked.
* @beforemethod: it will be run before each test method.
* @aftermethod: it will be run after each test method.
* @before test: it will be run before any test method belongs to the classes.
* @after test: it will run after any test method belongs to the classes.
* @before suite: it will be run before all tests in this suite have run.
* @after suite: it will be run after all tests in this suite have run.

@dataProviders:

* DataProvider: A test method that uses Data Provider will be executed a multiple number of times based on the data provided by the Data Provider. The test method will be executed using the same instance of the test class to which the test method belongs.
* Whereas, dataprovider is used to provide parameters to a test. If you provide dataprovider to a test, the test will be run taking different sets of value each time. This is useful for a scenario like where you want to login into a site with different sets of username and password each time.

Public class DataProviderClass

{

    @BeforeClass

    Public void beforeClass() {

        System.out.println("Before class executed");

    }

    @Test (dataProvider = "dataMethod")

    Public void testMethod(String param) {

        System.out.println("The parameter value is: " + param);

    }

    @DataProvider

Public Object [][] dataMethod() {

        return new Object[][] { { "one" }, { "two" } };

    }

}

@factory:

* Factory in TestNG defines and creates tests dynamically at runtime.
* This is useful if you want to run the test class any number of times.
* For example, if you have a test to login into a site and you want to run this test multiple times, then it’s easy to use TestNG factory where you create multiple instances of test class and run the tests (may be to test any

Memory leak issues).

Public class SimpleTest

{

 Private String param = "";

  Public SimpleTest(String param) {

  this.param = param;

    }

@BeforeClass

    Public void beforeClass() {

        System.out.println("Before SimpleTest class executed.");

    }

@Test

    Public void testMethod() {

        System.out.println("testMethod parameter value is: " + param);

    }

}

 Public class SimpleTestFactory

{

   @Factory

    Public object [] factoryMethod() {

        return new Object[] {

                                new SimpleTest("one"),

                                new SimpleTest("two")

                            };

    }

@parameters:

* Used to pass the test methods.
* Define parameters in the xml.
* It is used for configuration purpose.
* It is used to supply the test data.
* Used in parallel browser execution.

@Listeners:

1. IAnnotationTransformer ,
2. IAnnotationTransformer2 ,
3. IConfigurable ,
4. IConfigurationListener ,
5. IExecutionListener,
6. IHookable ,
7. IInvokedMethodListener ,
8. IInvokedMethodListener2 ,
9. IMethodInterceptor ,
10. IReporter,
11. ISuiteListener,
12. ITestListener.

**ITestListener** has following methods:

* **onStart**: This method is called when any Test starts.
* **onFinish**: This method is called after all Tests are executed.
* **onTestSuccess**: This method is called on the success of any Test.
* **onTestFailure**: This method is called on the failure of any Test.
* **onTestSkipped**: This method is called on skipped of any Test.
* **onTestFailedButWithinSuccessPercentage**: method is called each time Test fails but within success percentage.

**IExecutionListener** has the following methods

It monitors the beginning and end of TestNG run and it includes two methods

* **onExecutionStart():** is run before the TestNG starts running the suites
* **onExecutionFinish()**:is run after TestNG has completed running all the test suites.

The start time is recorded within the method onExecutionStart() and the method onExecutionFinish() is used to print the time TestNG takes to run all the suites.

36. What is the difference is between verify and assert?

Both commands are used for verification purpose.

Assert

* If we use Assert command in the script to verify element in the webpage, the excepted and actual results are not matching properly, it stops the execution of the script.

Verify

* If we use verify command in the script to verify element in the webpage, the excepted and actual results are not matching properly, it will not stop the execution of the script, it continues the script.

37. What is the difference between findeEement() and findelements()?

Ans: Both the methods are used to finds the elements in the webpage.

Find Element ():

* It will find the first matching element in the webpage and if the element is not available or not exists in the webpage it throws noSuchelement exception.

Find Elements ():

* It will find all the matching elements in the webpage, whereas, the findElements method returns an empty list when the element is not available or doesn’t exist on the page.
* It doesn’t throw NoSuchElementException.

38. What are the exceptions majorly faced in selenium and explain?

* nosuchElementException, nosuchwindowException, noSuchframeException, StaleElementReferenceException, timeOutException, webdriverException.

nosuchElementException

* When the element is not present in the Dom (document object model). It throws nosuchelementException.

nosuchwindowException:

* When the driver is trying to switch to the window which is not available, it will throw nosuchwindowexception. So we need to check window id that we pass or wait for some time until the new window is appears.

noSuchFrameException:

* Whenever the driver unable to switch to frame with element, it throws nosuchframeException.

timeOutException:

* When the element was not displayed in the specified time whenever we work with waits, we will see these exceptions.

StaleElementReferenceException:

* This exception is occurs when driver is trying to perform an action on element which is no longer exist or not valid.

WebdriverException or Firefox exception:

This exception may get for if any jars are missing to configure our project or the problem with browser versions selenium will not support latest versions of Firefox browser. It supports below 46 versions like 46, 44, 43, 42 etc.

39. What are the different types of automation frameworks?

* Data Driven Testing Framework
* Keyword Driven Testing Framework
* Modular driven testing framework
* Hybrid Testing Framework

40. What are the advantages of automation framework?

* Reusability of code
* It helps for Maximum coverage
* Recovery scenario
* Low cost maintenance
* Easy Reporting

41. What is page object model framework?

* Page Object Model is a design pattern to create Object Repository for web UI elements.
* Under this model, for each web page in the application there should be corresponding page class.
* This Page class will find the WebElements of that web page and also contains Page methods which perform operations on those WebElements.
* Name of these methods should be given as per the task they are performing i.e., if a loader is waiting for payment gateway to be appear, POM method name can be waitForPaymentScreenDisplay().

42. What is the difference between Excel file and web table?

43. What are the factors we need to consider to develop a particular framework?

44. What are the advantages of JENKINS?

45. How to handle window alerts?

By using autoIt we can handle it. We need to configure autoIt jars to our project.

46. What are the advantages if you have any framework and what are the disadvantages if you don’t have framework?

47. How to send your test results by an email after completion of test execution?

48. How to run your script parallel in all the browsers?

49. What is object repository and how to properties file?

File file = new File("test.properties");

FileInputStream fileInput = new FileInputStream(file);

Properties properties = new Properties();

properties.load(fileInput);

fileInput.close();

50. What is maven and why we use maven?

51. How to perform right click on the web page?

52. How do you send data into textbox without using sendkeys

// JavaScript Executor to enter text

      JavascriptExecutor j = (JavascriptExecutor)driver;

      j.executeScript ("document.getElementById('gsc-i-id1').value='Selenium'");

**Challenges:**

* Handling dynamic xpath.
* Validating the images.
* Handling broken links.
* Handling selenium exceptions.
* Handling browser compatibility issues
* Handling captcha.

**Handling Extent report:**

1. Imported two classes ***Extent Reports*** and ***Extent Test***.
2. ***ExtentReports*:**By using this class we set the path where our reports need to generate.
3. ***ExtentTest*:**By using this class we could generate the logs in the report.
4. Took three methods with @Test annotation such as *passTest*, *failTest* and *skipTest* and a method *startTest with* @BeforeTest annotation and another method *endTest* with @AfterMethod annotation
5. iii. Used object of ***ExtentReports***class (i.e., *extent*) in the *startReport* method which was assigned to @BeforeTest annotation to generate the HTML report in the required path
6. iv. Used object of ***ExtentTest*** class (i.e., *logger*) in the remaining methods to write logs in the report.
7. v. Used ***ITestResult*** class in the @AfterMethod to describes the result of a test.

**CSS locator:**

Following are some of the mainly used formats of CSS Selectors.

* Tag and ID

css=tag#id

* Tag and Class

css=input.inputtext

* Tag and Attribute

css=tag[attribute=value]

css=input[name=Email]

* Tag, Class, and Attribute

css=tag.class[attribute=value]

css=input.inputtext[name=email]

* Sub-String Matches
  + **Starts With (^)**

css=<HTML tag><[attribute^=prefix of the string]>

css=input[id^='Em']

* + **Ends With ($)**

css=<HTML tag><[attribute$=suffix of the string]>

css=input[id$='001']

* + **Contains (\*)**

css=<HTML tag><[attribute\*=sub string]>

css=input[id\*='id']

* Child Elements
  + Direct Child
  + Sub-child
  + nth-child:

**HTML:**

* + <ul id="automation">
  + <li>Selenium</li>
  + <li>QTP</li>
  + <li>Sikuli</li>
  + </ul>

Nth value:

* + css="ul#automation li:nth-of-type(2)"

Last Child:   
 css="ul#automation li:last-child"

Collections in Selenium

\*\*gets all the options belonging to the Select tag

Select objSelect = new Select(driver.findElement(By.id("Search-box")));

List <WebElement> elementCount = oSelect.getOptions();

System.out.println(elementCount.size());

List : to print data from Table

Set: driver.getWindowHandles();

String parentWindowHandle = driver.getWindowHandle();

Set<String> allWindowHandles = driver.getWindowHandles();

Map: Multiple user logins, to store username and password

How to Select Dropdown in Selenium?

You can handle dropdown in Selenium using Select Class and without using Select Class. Below are 5 different methods that can be used to select value in dropdown in Selenium without using Select Class. These methods are:

1. By storing all the options in List and iterating through it
2. By creating Custom Locator and without iterating the List
3. By using [JavaScriptExecutor](https://www.browserstack.com/guide/javascriptexecutor-in-selenium" \t "_blank" \o "How to use JavascriptExecutor in Selenium) class
4. By using [sendKeys](https://www.browserstack.com/guide/sendkeys-in-selenium" \t "_blank" \o "SendKeys in Selenium WebDriver)
5. By using [Actions Class](https://www.browserstack.com/guide/action-class-in-selenium)

Javascript Executor

WebElement dd = driver.findElement(By.xpath("//select"));

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].value='highestprice'", dd);

Sendkeys:

 driver.findElement(By.*xpath*("//select")).sendKeys("highestprice");

API

1. Error codes

200 codes are success codes

200- ok

201-Created

202-Accepted

204-No content

205-Reset Content

400- Client side errors

400- Bad Request

401-Unauthorized

404- Not found

Few 400 and 500 are Informational Codes

500- Internal Server Error

503- Service unavailable

504- Gateway Timeout

Redirectional

301- Moved Permanently

302- Found(Moved Temporarily)

304- Not Modified

Get- Extract/Fetch the data

Post – To create the entity

Put – will update entire record

Patch – Will update few parts of the record

The DELETE command deletes one or more existing records from the table in the database. The DROP Command drops the complete table from the database. The TRUNCATE Command deletes all the rows from the existing table, leaving the row with the column names.

imp:

cmd F/ctrl F - to check your xpath in console

F12 to open consol

to directly copy xpath--->inspect element.. go to the console highlihgted line right click, copy, copy xpath

Xpath:

1.Basic Xpath

Xpath=//tagname[@attribute='value']

Relative xpath

//form/input[3]

2. Contains ()-used when the value of any attribute changes dynamically, for example, login information.

Xpath=//\*[contains(@type,'sub')]

3. OR and AND

OR- identified when anyone condition satisfies

AND - both condition should satisfy

Xpath=//\*[@type='submit' or @name='btnReset']

Xpath=//input[@type='submit' and @name='btnLogin']

4.starts-with

Xpath=//label[starts-with(@id,'message')]

5.Text()

Xpath=//td[text()='UserID']

6. Last()

//form/input[last()-1]

7.position()

descendant::input[position()=2]

it means select second input tag

=========================================

Xpath Axes: used to find the complex or dynamic elements also when no attributes like class,id and name

1. Following (select parallel nodes, child and grandhild of those parallel nodes as well)

Xpath=//\*[@type='text']//following::input

2.Ancestor

Xpath=//\*[text()='Enterprise Testing']//ancestor::div

3. Child

Xpath=//\*[@id='java\_technologies']//child::li

Xpath=//\*[@id='java\_technologies']//child::\*

selects all children of above tag

4. Preceding

Xpath=//\*[@type='submit']//preceding::input

5. Following-sibling

xpath=//\*[@type='submit']//following-sibling::input

//div[@id=""]/following-sbling::div[position()=2]/descendant::\*

6.Parent

Xpath=//\*[@id='rt-feature']//parent::div

7.Self

Xpath =//\*[@type='password']//self::input

8.Descendant (selects child and grandchild)

Xpath=//\*[@id='rt-feature']//descendant::a

Dynamic Radio buttons:

//span[@class='']/input[preceding-sibling::label[text()='custom']]

Read data from excel

//Create an object of File class to open xlsx file

File file = new File(filePath+"\\"+fileName);

//Create an object of FileInputStream class to read excel file

FileInputStream inputStream = new FileInputStream(file);

Workbook guru99Workbook = null;

//Find the file extension by splitting file name in substring and getting only extension name

String fileExtensionName = fileName.substring(fileName.indexOf("."));

//Check condition if the file is xlsx file

if(fileExtensionName.equals(".xlsx")){

//If it is xlsx file then create object of XSSFWorkbook class

guru99Workbook = new XSSFWorkbook(inputStream);

}

//Check condition if the file is xls file

else if(fileExtensionName.equals(".xls")){

//If it is xls file then create object of HSSFWorkbook class

guru99Workbook = new HSSFWorkbook(inputStream);

}

//Read sheet inside the workbook by its name

Sheet guru99Sheet = guru99Workbook.getSheet(sheetName);

//Find number of rows in excel file

int rowCount = guru99Sheet.getLastRowNum()-guru99Sheet.getFirstRowNum();

//Create a loop over all the rows of excel file to read it

for (int i = 0; i < rowCount+1; i++) {

Row row = guru99Sheet.getRow(i);

//Create a loop to print cell values in a row

for (int j = 0; j < row.getLastCellNum(); j++) {

//Print Excel data in console

System.out.print(row.getCell(j).getStringCellValue()+"|| ");

WebDriver driver = new ChromeDriver() explanation

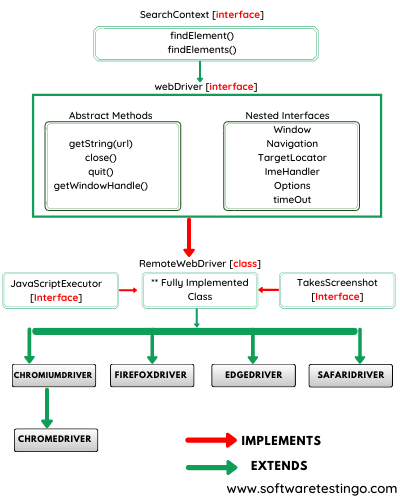
ChromeDriver is a class

ChromeDriver() is the default constructor of ChromeDriver class

New ChromeDriver()=We ae creating object for ChromeDriver class

driver = new ChromeDriver() – above created instance is stored in “driver” local variable

WebDriver= return type



Difference between Navigate and get

Driver.get(url) – will wait until the page loads

Driver.navigate.to(url) – will just launches page and doesnot wait until it loads. You need to use waits implicit/explicit/thread.sleep to wait till the page loads completely It stores the browser history and helps in navigatin to and fro

* Driver. navigate().back();
* Driver. navigate().forward();
* Driver. navigate().refresh();

Difference between selenium 3and selenium4

Ans.1. Protocol change

1. Chromedriver r extendes Chromiumdriver whereas in selenium3 chormedriver diectlyextends remotedriver class
2. Relative locators: above, below, toRightof,toLeftof, near- which help to identify elements ‘relative’ to a particular element in DOM

Examples

* above

By username = RelativeLocator.with(By.id("loginusername")).above(By.id("loginpassword"));

driver.findElement(username).sendKeys("demouser");

* below

By pwd = RelativeLocator.with(By.id("loginpassword")).below(By.id("loginusername"));

driver.findElement(pwd).sendKeys("test");

* Left Of

By CloseBtn = RelativeLocator.with(By.tagName("button")).toLeftOf(By.xpath("//button[contains(text(),\"Log in\")]"));

* Right Of

By LoginBtn = RelativeLocator.with(By.xpath("//button[contains(text(),\"Log in\")]")).toRightOf(By.tagName("button"));

* near

By username = RelativeLocator.with(By.id("loginusername")).near(By.id("loginpassword"));

1. ChromeDevtools
2. DesiredCapabilities is replaced by Options class

**In Selenium 3**

DesiredCapabilities caps = DesiredCapabilities.firefox();

caps.setCapability("platform", "Windows 10");

caps.setCapability("version", "92");

caps.setCapability("build", myTestBuild);

caps.setCapability("name", myTestName);

WebDriver driver = new RemoteWebDriver(new URL(cloudUrl), caps);

**In Selenium 4**

FirefoxOptions browserOptions = new FirefoxOptions();

browserOptions.setPlatformName("Windows 10");

browserOptions.setBrowserVersion("92");

Map<String, Object> cloudOptions = new HashMap<>();

cloudOptions.put("build", myTestBuild);

cloudOptions.put("name", myTestName);

browserOptions.setCapability("cloud:options", cloudOptions);

WebDriver driver = new RemoteWebDriver(new URL(cloudUrl), browserOptions);

1. Actions class: New methods has been introduced ContextClick(), Click(), ClickAndHold(), doubleClick(), and release().

Let’s see a few examples of these methods:

* **clickAndHold**

WebElement clickable = driver.findElement(By.id("clickable"));

new Actions(driver)

.clickAndHold(clickable)

.perform();

* **ContextClick**

WebElement clickable = driver.findElement(By.id("clickable"));

new Actions(driver)

.contextClick(clickable)

.perform();

* **doubleClick**

WebElement clickable = driver.findElement(By.id("clickable"));

new Actions(driver)

.doubleClick(clickable)

.perform();

**Read More:** [How to perform Double Click in Selenium?](https://www.browserstack.com/guide/double-click-in-selenium)

* **Click**

WebElement clickable = driver.findElement(By.id("click"));

new Actions(driver)

.click(clickable)

.perform();

**Also Read:**​​[Understanding Click Command in Selenium](https://www.browserstack.com/guide/selenium-click-command)

* **Release**

WebElement clickable = driver.findElement(By.id("click"));

new Actions(driver)

.click(clickable)

.release()

.perform();

Scrollby :

**A Scroll** is a JavaScript method. The JavaScriptExecutor provides an interface that enables QAs to run JavaScript methods from Selenium scripts.

Using pixels:

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("window.scrollBy(0,250)", "");

### How to scroll down to an element in Selenium until it is visible

//Locating element by link text and store in variable "Element"

WebElement Element = driver.findElement(By.linkText("Try Selenium Testing For Free"));

// Scrolling down the page till the element is found

js.executeScript("arguments[0].scrollIntoView();", Element);

**Scroll to bottom of the page**

js.executeScript("window.scrollBy(0,document.body.scrollHeight)");