**Selenium Interview Question**

1. What is selenium?

Ans - Selenium is a UI automation tool which will help us to automate the GUI. It will not support desktop application.

1. Does Selenium support automation of pages make in PHP and HTML?

Ans- It is independent of language, it can automate any pages made in any language.

1. Does selenium support automation of application made in only java?

Ans- No, it can only automate web applications.

1. Does selenium have execution engine?

Ans- No. It means selenium can not execute its own script by itself. Because for the execution of selenium scripts, we need to rely on 3rd party tool like TestNG or JUnit sometime Java main() method. We usually don’t prefer main() method for their execution as we cannot avail many testing feature via it and also will not get report for their execution.

1. Does selenium has excel reading methods?

Ans- No, Selenium API doesn’t have any such methods. If you want to read data from external source like property files or excel file, you need to use apache-poi or JExcel (apache poi is easy to use than JExcel) for excel and properties class for property file.

1. What are components of selenium?

Ans- It mainly has 4 components- Selenium IDE, Selenium RC, Selenium Grid and Selenium WebDriver.

1. Selenium IDE is a simple record and playback kind of tool which comes as an add-on for Mozilla Firefox and Chrome. It is used for prototype testing. Test cases written in IDE can be exported in many programming languages like Ruby, Java, C#, etc. Edit and Debug options along with record are also available. It is an excellent tool for beginners to understand the syntax of Selenium WebDriver. Selenium IDE records multiple locators for each element it interacts with. If one locator fails during playback, the others will be tried until one is successful. Selenium IDE requires no additional setup other than installing the extension on your browser.
2. Selenium RC: Selenium RC (Remote Control) was the first tool of Selenium Suite. Earlier it was known as JavaScript Executor. RC was the tool which made Selenium famous in the market.

It was the first tool which provided the support for multiple programming languages (JAVA, Ruby, Perl, PHP, Python, and C#).

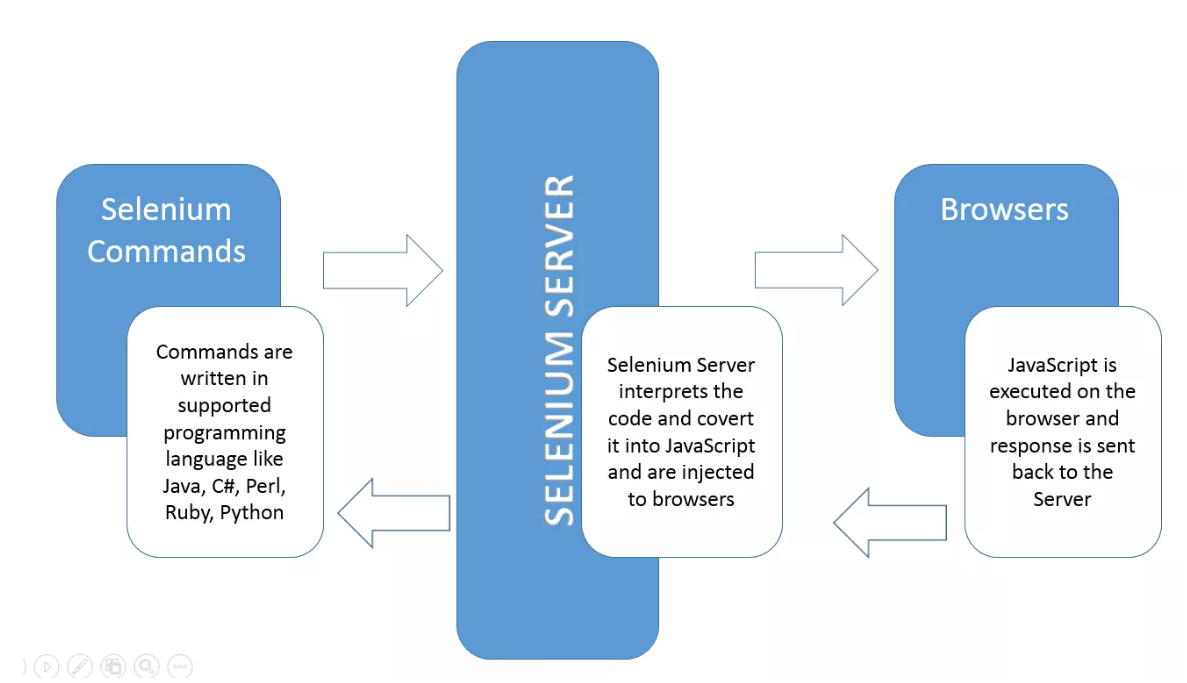
It also supported almost all the major vendors of Browsers like Mozilla Firefox, Google Chrome, Internet Explorer. All the browsers which support JavaScript can be automated using this tool.

Selenium RC is also known as Selenium 1.

The architecture of Selenium RC:

In Selenium RC, there is a manual process called **Selenium Server** is mandatory to start before execution,which acts as a middleman between the code and the browser.

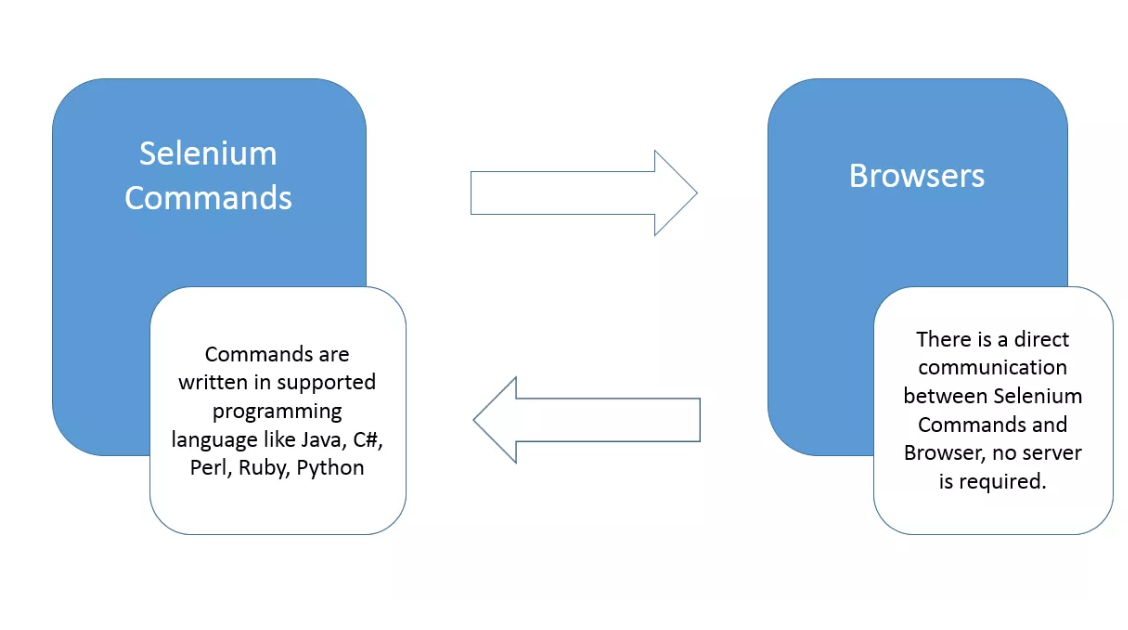
The commands (API’s) are sent to Server. It interprets the command and converts it into JavaScript and then JavaScript is injected to the browser. Now the browser executes the javascript and responds to a server, which again interprets the command and returns to code in the respective language.



Selenium RC is deprecated now.

1. Selenium WebDriver: It is very powerful. Because of many limitations with RC, WebDriver was developed. It does not require any manual process like Selenium Server. There is a direct communication between code and browser.

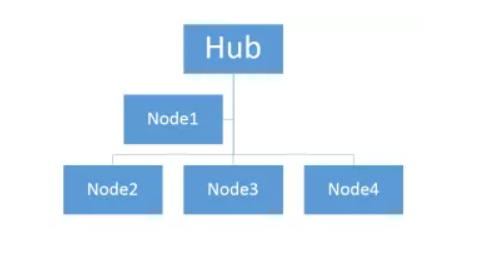
* Features of Selenium WebDriver:
* Open source
* Supports all the key vendors of the browser like Mozilla Firefox, Internet Explorer, Google Chrome, Safari, etc.
* Support Multiple languages like C#, JAVA, Ruby, Perl, Python, and PHP.
* Supports multiple platforms like Linux, Windows, MAC, etc.
* No middleman like Selenium RC server is required.
* Easy to remember API’s.
* Easy to integrate with Testing frameworks.
* Framework Development.
* Parallel Testing capabilities.



1. Selenium Grid: It is the last component of the selenium suite and is used for parallel testing or distributive testing. It allows us to execute test scripts on different machines at same time on different browser from your machine. There is a Hub which controls the execution on various machines, and there are multiple nodes on which actual implementation is done.

It is the last component of the selenium suite and is used for parallel testing or distributive testing. It allows us to execute test scripts on different machines at same time. There is a Hub which controls the execution on various machines, and there are multiple nodes on which actual implementation is done. This is master- slave relationship.

The Architecture of Selenium Grid: In Grid one of the systems is created as Hub. Hub works as a central point controlling all the nodes. Nodes are an actual machine on which execution is done.



1. What are the limitations of selenium?

Ans:

a) It supports testing of only web-based applications.

b) We cannot directly test mobile applications using it, we need to use 3rd party tool like Appium integration with selenium. Appium is “cross-platform”: It facilitates cross platform testing for both iOS and Android using the same API. Appium is open source. You don’t need source code of app to automate it. It is server written in NodeJS. Through Appium you can test automatically your mobile application either on emulator or on real device. [Appium](http://appium.io/) server reads the command coming in from test Java code and executes that command on real device as well as on emulator.

Basic configuration for it-

Install at least JAVA 8, Microsoft .Net framework 4.5, Android SDK (API level 17 or higher), Developer options enabled and Install Appium on your machine.

Jar needed-

Selenium java, Appium Java client and Gson jar.

1. We created a new Java project in Eclipse and added/build above mentioned JAR files in your project.
2. Add the Android apk file path in our code.
3. Set the above-mentioned device capabilities.
4. Initialize WebDriver instance with AndroidDriver as the path for Appium server through which they can connect to each other.
5. Now find elements and perform action on those elements. You can find an element’s id by launching the uiautomationviewer tool available in Android SDK tools folder. You need to take a screenshot of the device and hover mouse on that particular element.
6. Quit your driver.

File app= **new** File("apk-file-path");

DesiredCapabilities capabilities= **new** DesiredCapabilities();

capabilities.setCapability("deviceName", "your-device-name");

capabilities.setCapability("platformVersion", "platform-version");

capabilities.setCapability("platformName", "platform-name");

capabilities.setCapability("app", app.getAbsolutePath());

Webdriver driver = **new** AndroidDriver(**new** URL("http://127.0.0.1:4723/wd/hub"),capabilities);

driver.findElement(By.id("username-element")).sendKeys("username");

driver.findElement(By.id("password-element")).sendKeys("password");

driver.findElement(By.id("password-element ")).click();

driver.quit();

}

Steps to execute automation script:

1. Launch Appium server.
2. Execute the above java code from Eclipse.
3. Now it will start executing script on real device or on emulator

C) Captcha and Barcode is not supported by selenium. At the time of registration in the application, they have captcha or Barcode which can not be read by selenium.

D) For report generation, we need to use 3rd party tools like TestNG and Junit or extents report.

E) As selenium is a free tool, you get very less support whenever you get any error or exception or configuration issue as there is no ready vendor support though the user can find numerous helping communities.

F) User is expected to possess some prior language experience.

1. What are the different types of locators?

Ans- ID, ClassName, Name, TagName, Linktext, Partial linktext, XPath and CSS selector.

1. What does mean by implicit wait?

Ans- If webdriver cannot find an element in the document object model(DOM), then it will wait for defined amount of time for the element to appear in the DOM. An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available. The default setting is 0.

Once set, the implicit wait is set for the life of the WebDriver object instance.

It is applicable for the entire page. Driver will wait till 60sec to make sure that all element on page get loaded. If the element is not available within the specified Time an NoSuchElementException will be raised.

***driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS)***

1. What does mean by explicit wait?

Ans- Unlike implicit wait, you can write custom code or conditions for wait before proceeding further in the code. WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully.

**When to use:** If element takes a long time to load. Also, used to check property of an element (presence, clickability. etc).

•There can be instance when a particular element takes more than a minute to load.

•In that case you definitely not like to set a huge time to Implicit wait, as if you do this your browser will going to wait for the same time for every element.

•To avoid that situation you can simply put a separate time on the required element only.

•By following this your browser implicit wait time would be short for every element and it would be large for specific element.

We can wait for particular element on the page to get displayed.

***WebDriverWait wait= new WebDriverWait(driver, 60);***

***Wait.until(ExpectedConditions.elementToBeSelected(driver.findElement(By.xpath(“”))));***

Let’s say you have situation you have page and that has been loaded, there is an 3rd party advertisement which can comes on your page after 5ses, with this explicit wait we can wait for that advt. but can’t be possible with implicit wait.

***ExpectedConditions class has following methods-***

***elementToBeClickable(By locator),***

***elementToBeSelected(Webelement element),***

***textToBePresentInElement(By locator, String text)***

***titleContains(String title)***

Note: Explicit wait will not wait for 60 sec, the moment when that advt. appears on that page it will come out from wait time and start executing the script onwards. Same is the case with implicit wait, it will come out once the page gets loaded.

1. What is pooling interval?

Ans- Another kind of WebDriverWait. Where we can say that what is the frequency of pooling. In case of explicit wait, your driver will check for web element for every 500 milliseconds. But in the pooling interval-

***WebDriverWait wait= new WebDriverWait(driver, 60);***

***Wait.pollingEvery(5, TimeUnit.SECONDS)***

***Wait.until(ExpectedConditions.elementToBeSelected(driver.findElement(By.xpath(“”))));***

It will check for webelement for every 5 seconds. Check at 5 sec, 10 sec ,15sec…60sec.

1. What is the fluent wait?

Ans- •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 with polling mechanism, as this will try to find element again and again until it finds it or until the final timer runs out.

•It tries to find the web element repeatedly at regular intervals (as specified in polling period) of time until the timeout or till the object gets found. It can define the maximum amount of time to wait for a specific condition and frequency with which to check the condition before throwing an “ElementNotVisibleException” exception.

There is another explicit wait mechanism. It can be achieved in 2 ways-

1. WebDriverWait 2) Fluent wait.

It will find the web element repeatedly at regular intervals of time until the timeout or till the objects get found.

***Wait wait = new FluentWait(WebDriver reference)***

***.withTimeout(Duration.ofSeconds(SECONDS))***

***.pollingEvery(Duration.ofSeconds(SECONDS))***

***.ignoring(Exception.class);***

/below one is deprecated code. Use the above one

***Boolean wait= new FlunetWait<WebDriver>(driver)***

***.withTimout(60, TimeUnit.SECONDS)***

***.pollingEvery(5, TimeUnit.SECONDS)***

***.ignoring(NoSuchElementException.class)***

***.until(ExpectedConditions.elementToBeSelected(driver.findElement(By.xpath(“”))))***

To poll for every 5 seconds upto 60 seconds to find the web element. When ever we poll, we will get exception till the element is not present and we should ignore such exception.

1. Why synchronization in a test is required?

Ans- When tests are run, the application may not always respond with the same speed. For example- it might take s few seconds for progress bar to reach 100percent, a status message to appear, a button to become enabled and a window or pop-up message to open.

You can handle these anticipated timing problems by synchronizing your test to ensure that selenium webdriver will wait until your application is ready before performing a step.

1. Disadvantage of using implicit wait?

Ans- Suppose you need to test app where ”getting the list of hotels” in city take 5 seconds but “getting list of hotels in country” takes 15 seconds. Due to this we will increase our implicit wait time from 5 sec to 15 seconds. Now in future, if due to some technical glitches or some performance coding issues, your wait time for scenario- ”getting the list of hotels” increases from 5 sec to 10-13 sec, then we are not able to catch this bugs via our test scripts. Later on it will be questioned by our QA manager. And if we reduce the wait time, “getting list of hotels in country” will get fail. So this issue can be resolved using explicit wait. Implicit wait define wait time globally. Using explicit wait, we will be maintaining wait time as per the specification given by developer. And if a wait time for any page increase, then it can be easily identified.

However implicit wait will slow down your tests if application responds normally as it will wait for each element appearing in the DOM and increase the overall execution time. It is recommended to avoid or minimize the use of it. Try to handle synchronization issues with an explicit wait which provides more control when compared with implicit wait.

In big framework, combination of implicit and explicit wait is best solution for synchronization issues.

1. How to implement custom wait code?

Ans***- WebElement message= (new WebDriverWait(driver, 5))***

***.until(new ExpectedCondtion<WebElement>(){***

***public WebElement apply(WebDriver d){***

***Return d.findElement(By.id(“page4”)***

***}})***

Here we created a custom conditions, which returns the webelement object once the inner findElement() method locates the element within a specified timeout.

Useful in some scenario like

1. based on events and action performed, the value of element attribute might change at runtime like a disabled textbox gets enabled based on user’s rights.

***Return d.findElement(By.id(“username”)).getAttribute(“readOnly”).contains(“true”);***

***Here ExpectedConditions will wait for Boolean return value based on the attribute value of an element.***

1. Waiting for DOM events. The web application may be using a JavaScript framework such as jQuery for AJAX and content manipulation. For ex- jQuery is used to load the big JSON file from server asynchronously on the page. While jQuery is reading and processing this file,

A test can check its status using ACTIVE attribute. Here custom wait is implemented.

***JavaScriptExecutor js= (JavaScriptExecutor) d;***

***Return (Boolean) js.executeScript(“return jQuery.active== 0”)***

1. When we have to use the fluent wait?

Ans- Now in some case, HTML is same for

1. Thread.sleep?

Ans- your sleeping your thread for particular time. Thread.sleep(2000) will pause the execution for complete 2 sec even though the page is ready or will load in 1sec. It is not part of a webdriver, part of java. Advised not to use in the framework.

1. What is page load timeout?

Ans- When you are doing driver.get(), we load the URL. Here we can supply the timeout. It can achieved

•The pageLoadTimeout limits the time that the script allots for a web page to be displayed.

•If the page loads within the time then the script continues.

•If the page does not load within the timeout the script will be stopped by a TimeoutException.

1. What is JavaScript wait?

Ans***- ExpectedConditions<Boolean> javaScriptWait= new ExpectedConditions<Boolean>(){***

***Public Boolean apply(Webdriver driver){***

***Return ((JavaScriptExecutor)driver).executeScript(“return document.readyState”).equals(“complete”);***

***}***

***}***

***Try{***

***System.out.printlin(“Waiting for page to get load..”);***

***WebDriverWait wait= new WebDriverWait(driver,60);***

***Wait.until(javaScriptWait);***

***}Catch(Throwable error){***

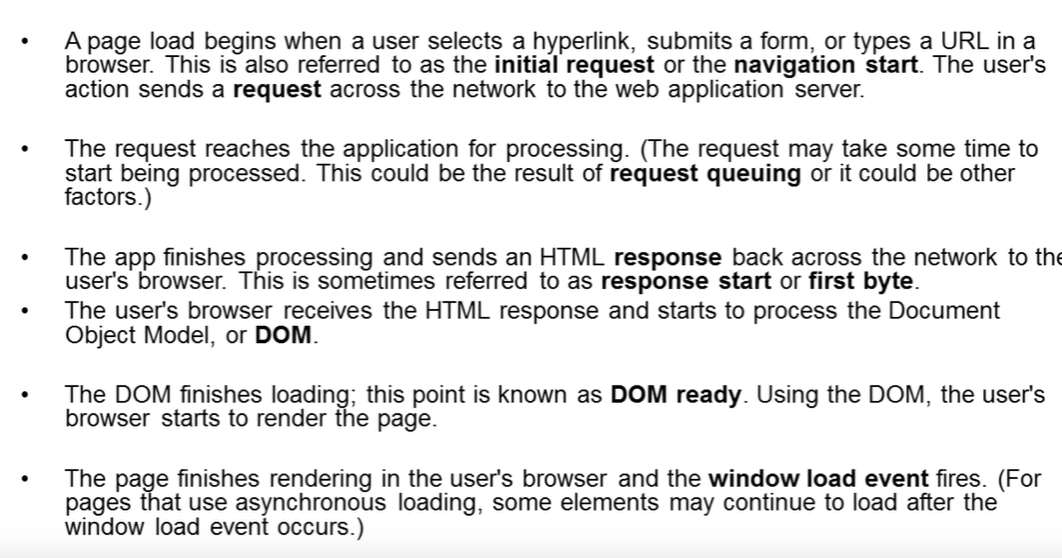
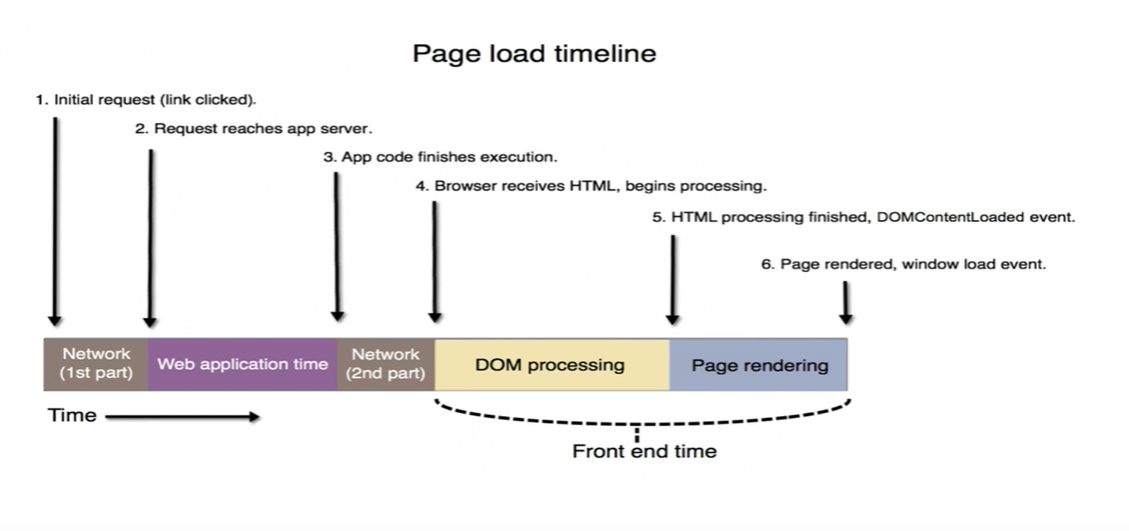
***System.out.printlin(“timeout as page is not loaded after “+60+”seconds”);***

***Asser.assertTrue(true,” timeout as page is not loaded after “+60+”seconds”);***

***}***

1. What is DOM?

Ans- It is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.



When we call a webpage, we get a HTML which is then converted into DOM so that we can call or work with web element.

Now when a Javascript source file is included in a webpage via HTMK <script> tag, there are some script tag which are calling some file. The processing /loading of included file has to be complete first before anything happens on the page. The script is supposed to complete first then any more of such included file is rendered/executed. This we call synchronous loading.

Asynchronous loading-> Sometime we specifically want Javascript source files to load asynchronously. Means while some page is getting rendered, at the same time specified included file in your javascript is also get processed but we will not wait for this script to complete its processing but we will do both the things simultaneously or we may wish that once my page is loaded/rendered then you can run the script later on. This may be handy for files from external sources where we don’t want our including pages to suffer the consequences of: slow connection speed, heavily loaded external servers or waiting for timeouts on external servers that aren’t even there at the moment. Here we want files which are from external sources to get executed inside the JavaScript later on once the page gets rendered then use asynchronously loading.

For these cases, HTML <script> element has ‘async’ and ‘defer’ attributes. Defer requests means loading of specified file occurs after the included page has finished loading. While Async request means that loading of specified file occurs concurrently (or at least independent of) included page.

***Implicit wait: Will wait for the element which are getting loaded asynchronously (i.e. after page has been loaded/rendered)***

***Page Load time: Will wait until complete page gets loaded, not for those elements which are getting loaded asynchronously.***

1. How to enter data in text box

Ans- locate the textbox using driver.findElement() and then use ***sendkeys()*** to enter the text.

1. How to click on link?

Ans- - locate the textbox using driver.findElement() and then use ***click()***

1. How to check the element is displayed or not

Ans- - locate the textbox using driver.findElement() and then use ***isDisplayed().*** It returns true if element is present but if it is not present, it will throw an exception, we need to handle using try-catch exception. Write “return false” in catch block.

1. How to make sure checkbox is selected or not?

Ans- locate the checkbox using driver.findElement() and there is a method ***“isSelected()”*** which will return true if it is selected else false. But if element is not present, it will throw an exception, we need to handle using try-catch exception. Write “return false” in catch block.

2nd option is to get attribute whose values get changes if checkbox is getting selected. So locate the checkbox using ***driver.findElement(By.xpath(“”)).getAttribute(“class”).contains(“checked”)-*** it is checked then its class name should have “checked” text. If yes then return true else false. It can be any text. Just verify in HTML.

1. What is method to find all elements on UI?

Ans- ***driver.findElements()*** is used. Like getting all anchor tag , use By.tagname(“//a”)

1. How to get the current URL of window?

Ans- ***driver.getCurrentUrl()***

1. How to get the title of page?

Ans- ***driver.getTitle();***

1. How to get window id on runtime? Means how to get the link of all the windows. And how to manage multiple windows?

Ans***- Set<String> windowsId= driver.getWindowHandles();*** //no duplicate entry is possible with SET Return a set of window handles which can be used to iterate over all open windows of this WebDriver instance by passing them to [switchTo()](eclipse-javadoc:%E2%98%82=Selenium/C:%5C/Users%5C/atupadhy%5C/.m2%5C/repository%5C/org%5C/seleniumhq%5C/selenium%5C/selenium-api%5C/3.14.0%5C/selenium-api-3.14.0.jar%3Corg.openqa.selenium(WebDriver.class%E2%98%83WebDriver~getWindowHandles%E2%98%82%E2%98%82switchTo%E2%98%82).[Options.window()](eclipse-javadoc:%E2%98%82=Selenium/C:%5C/Users%5C/atupadhy%5C/.m2%5C/repository%5C/org%5C/seleniumhq%5C/selenium%5C/selenium-api%5C/3.14.0%5C/selenium-api-3.14.0.jar%3Corg.openqa.selenium(WebDriver.class%E2%98%83WebDriver~getWindowHandles%E2%98%82Options%E2%98%82window%E2%98%82)

***Iterator<String> itr= windowsId.iterator();***

***ArrayList<String> ids=new ArrayList<String>(); //to store all the windows id. Other wise we again need to traverse***

***While(itr.hasNext()){***

***Ids.add(itr.next());***

***}***

// Suppose I clicked on link and it opens3 window, I want to go to the 3rd link.

**driver.switchTo().window(ids.get(2));**

now we can perform some task on 3rd window.

//to close 3rd window and move to 2nd window.

***Driver.close();***

***driver.switchTo().window(ids.get(1));***

//come to parent window.

***Driver.close();***

***Driver. switchTo().window(ids.get(0));***

1. How to maximize the window?

Ans- ***driver.manage().window().maximize();***

1. How to get co-ordinate of window?

Ans- ***Point point= driver.manage().window().getPosition();***

***Int x= point.getX();***

***Int y=point.getY();***

Similarly, we can get the position of web element on the UI.

Locate that element using ***Point point= driver.findElement(By.xpath()).getLocation()*** with respect to left window.

***Int x= point.getX(); Int y=point.getY();***

1. How to do the mouseover?

Ans- We have to make the object of action class-

***Action action=new Action(driver);***

***Action.moveToElement(driver.findElement(By.xpath())).build().perform()*** -> now action will be peform

1. How to do drag and drop

Ans- ***action. dragAndDrop( Source, target)* -**> both are the webelement.

2nd option is **action.clickAndHold(source)** -> this is the location of the element where I will go and click the element and hold. After that ***action.moveToElement(target)*** -> where you want to move. After move to target then release it on the target then build and perform. ***action.release(target).build().perform()***

***Action.clickAndHold(source).moveToElement(target).release(target).build().perform()***

3rd option is using keyUp and KeyDown()-

***Action.moveToElement(source).keyDown(Keys.CONTROL).moveToElement(target).keyUp(Keys.CONTROL).build().perform()***

1. How to the double click in selenium?

Ans- ***action.moveToElement(driver.findElement(By.xpath(“”))).doubleClick();***

1. How keys class works?

Ans- What ever function you know in keyboard, can be performed by selenium. You have keys class in selenium which will perform keyboard related function.

***Like Press enter -> action.sendKeys(Keys.ENTER);***

***Press F6 -> action.sendKeys(Keys.F6)***

1. How to work with alert?

Ans- ***Alert alert= driver.switchTo().alert();***

***Alert.accept();***

***Alert.dismiss(); //dismiss the alert***

1. How to work with iframe?

Ans- Count no of frames presents in your application-

Driver.findElements(By.tagName(“iframe”)).size();

// can switch to frame based on the index, name and webelement.

***Driver.switchTo().frame(0) // switch to 1st frame.***

***Driver.switchTo().frame(“auto”) // by the name***

***Driver.switchTo().frame(driver.findElement(By.xpath())) //Based on the web element***

1. How to work with select?

Ans- ***Select select=new Select(driver.findElement(By.xpath())*** //make a object of this class , pass the webelement on which select box you want to work.

There are also 3 methods

***Select.selectByIndex(0) /***/select by 1st index

**Select.selectByValue(“valuesName”); //**see the values in HTML tag

***Select.selectByVisibleText(“textname”);*** //values visible on the UI.

We have deselect also.

***Select.deselectAll();*** //everything will get the deselected.

***Select.deselectByValue(“valueName”)***

***Select.deselectByIndex(0);***

***Select.deselectByVisibleText(“textName”)***

1. how to execute the java script?

Ans- there are two ways for this

1. *EventFiringWebDriver*is a class and is used to wrap your *webdriver*around to throw events

***EventFiringWebDriver eventFiringWD= new EventFiringWebDriver(driver);***

***eventFiringWD. executeScript(“document.getElementById(\”idName\”)”);***

***eventFiringWD. executeScript(“document.getElementById(“\inputSuccess\”).value=\”text\””);***

Selenium WebDriver provides an API for tracking the various events that happen when test scripts are executed using WebDriver. Many navigation events get fired before and after a WebDriver internal event occurs (such as before and after navigating to a URL and before and after browser back-navigation), and these can be tracked and captured.

1. Via JavaScriptExecutor

***JavaScriptExecutor js= ((JavaScriptExecutor)driver).excuteScript((“document.getElementById(“\inputSuccess\”).value=\”text\””);***

1. How do you read values from the properties file?

Ans-a) create one property file ***“or.properties”***

b) I will make the object of property class-

***Properties OR = new Properties();***

1. Create the object of the file as your properties is a file

***File file= new File(System.getProperty(“user.dir”)+”Relative path to your properties file”);***

User.dir will give you path for current project.

1. Create a object of FileInputStreamReader as I want to read the all data as a stream of data in one go.

***FileInputStreamReader reader= new FileInputStreamReader(file);***

1. ***OR.load(reader) ->*** will load the input stream and Reads a property list (key and element pairs) from the input byte stream.
2. To read that, supply the keys

***OR.getAttribute(“username”);***

1. What are the java script functions for selenium object location?

Ans- In xpath, you have following JS functions

***Following-sibling, preceding-sibling, starts-with(), ends-with(), contains(), following, preceding***.

In CSS, you have these functions-

***Nth-child(1), $ (start-with), ^ (ends-with)***

1. How many test scripts do you write in a day?

Ans- We can say that if the methods i.e. page Functions are designed, then we can write 5-6 scripts. If there is only UI verification (we have to just the verify the element on the UI, assertion verification or some validation on UI pages) then we can write 8-10 scripts. We write means we make sure that script will execute in all the browsers supported by the application. But if there is end -to-end scenario need to be implemented then we can write max 1-2 scripts (Start from the homepage and go till the last page, let say in ecommerce- start from the homepage, select the product, go to the basket , go to the payment page and do the payment because methods are not ready, we need to write the page functions. then we call the methods and then run the scripts.

1. What are your roles and responsibilities?

Ans-

1. Involved in evaluating Selenium for Web UI Automation. ( In any company, when ever we start the automation, first we evaluate the tool whether this tool is fit for our application or not)
2. Involved in designing and implementing a Selenium web Driver automation framework built using Selenium web Driver + TestNG as Execution Engine. (Involved in the framework design)
3. Involved in designing automation scripts.
4. Performed automated Regression testing of the designed scripts, in coordination with manual testing team so whenever there is any failure happens through the automation, will inform to manual team and they do cross check)
5. Updated and maintained various test artifacts in the Test Automation Framework depending on the changes required. ( So whenever any requirement change, your locator will changes, your scripts will change, I do maintain the scripts and the locators)
6. How many team members do you have in your team?

Ans- This can be 1,2 or 10. Any ans can be fine

1. Is it possible to automate in sprint?

Ans- Yes, These days companies are following agile method where generally they make a sprint of 2 weeks. In this you will get a new feature. So you need to test the feature, once there is no bug then you go ahead with the automation of that feature. Then you can run automated sprint. So within two weeks, we can test the feature and then automate it and make sure that it can work for other sprints also.

1. How do you execute scripts?

Ans- There could be lot of ans for that.

1. You can execute the script through testNG.
2. You can schedule the jobs through Jenkins that will the trigger the script based on time which you have scheduled. Generally company follows it. And whenever there is build deployment, the job will get triggered automatically.
3. You can execute though bat file. You can make the bat file for the script and you can trigger the bat file.
4. What is source code management for you?

Ans- Github, Jenkins or other SVN. 99percent of the company keep their code in central repository like Github.

1. Do you do parallel testing?

Ans- ***<suite name="basicAnnotaions" parallel="classes" thread-count="3">***

Through testNG, we do parallel testing. Here now I can run 3 classes under this suite in 3 different methods.

1. What challenges you face in automation?

Ans- Say that-

1. Challenge was how do you write method that should work irrespective of changes in the UI. To defining the proper method
2. To handle dynamic objects like ID, class that keep on changing. To handle them, I need to use selenium functions /CSS functions like startsWith/endsWith/ following-preceding concepts. If you done some hardcoding, it will not work in the future
3. Execution was a challenged. How do you make sure that all of your test scripts get executed.
4. The way you define the structure for the framework. Whether your framework is good enough to accept any changes or not. Lets say today you are using selenium v3.0, suddenly selenium 5.0 comes in the market. Now whether the framework is flexible enough to adopt the challenge or not. You should always design the framework where you should not touch the test scripts, you should touch only the methods, your supporting functions or UI page functions.
5. To make sure that 100% of test scripts should pass. That was a challenge because in automation, when you run test scripts in the bulk, you could not guarantee all of the script will pass. That all depends on how well you have designed your scripts Whether you have handled proper wait conditions or proper assertion or not.
6. What are the methods of excel reader you have used?

Ans- In any excel workbook , it has lot of sheet. Here we are using apache-poi.

1. 1s we create the object of workbook by writing

***FileInputStream inputStream= new FileInputStream(“relative path to the excel file”)***

***XSSFWorkbook workbook= new XSSFWorkbook(inputStream)***

1. Get the sheet, create an object of sheet.

***Int index= workbook.getSheetIndex(“loginTest”);***

***XSSFSheet sheet= workbook.getSheetAt(index);***

1. Get the 1st row.

***XSSFRow row= sheet.getRow(0)***

count no of active rows (having content) in the sheet.

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

count the no of active columns in rows. Will Return the last logical cell in

the row PLUS ONE, or -1 if the row does not contain any cells.

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

1. To get the cell, use this ***XSSFCell cell= row.getCell(3)***
2. To get the data from excel file-

***If(cell.getCellType()= Cell.CELL\_TYPE\_STRING)***

***Cell.getStringCellValue()***

***Else if (cell.getCellType()= Cell.CELL\_TYPE\_NUMERIC)***

***Cell.getNumericCellValue()***

***Else if (cell.getCellType()= Cell.CELL\_TYPE\_BOOLEAN)***

***Cell.getBooleanCellValue()***