Selenium Interview Questions & Answers

1. **Explain the Selenium Architecture**

2. **Difference between Selenium-IDE & Selenium RC , WebDriver**

Selenium IDE:-

- Selenium IDE is a Firefox plugin which is used to create and execute test cases
- It records and plays back the interactions which the user had with the web browser
- Using IDE, you can export the programming code in different languages: Java, Ruby, Python and so on

Selenium Grid:-

- Selenium Grid is used for parallel testing or distributed testing. It allows us to execute test scripts parallely on different machines

Selenium RC:-

- Selenium Remote Control (RC) is used to write test cases in different Programming languages
- In Selenium IDE, we can run the recorded scripts only in Firefox browser, whereas, in Selenium RC, we can run the recorded script in any browser like IE, Chrome, Safari, Opera and so on

Selenium WebDriver:-

- Selenium WebDriver is a tool used to automate testing Scripts for web application
- It allows us to execute tests against different browsers like Firefox, Chrome, IE & Safari
- Selenium WebDriver eliminated the use of Selenium Server thus making it work faster than RC

3. **Difference between**
   WebDriver driver = new FirefoxDriver() ;
   FirefoxDriver driver = new FirefoxDriver() ;

-

WebDriver driver = new WebDriver(); // we cannot create Object of WebDriver Interface

FirefoxDriver driver = new FirefoxDriver(); // we can create object of a Firefox class

We can invoke every methods of Firefox class only by using instance of ***FirefoxDriver***. ***FirefoxDriver*** implements all methods of ***WebDriver*** Interface. By using this instance we can run test script only on Firefox only and not on other browser.

WebDriver driver = new FirefoxDriver(); // WebDriver interface type object reference is given to instance of FirefoxDriver class

By changing the constructor in above statement we can run the test script on different browsers.

4. **Difference between get(), navigate().to()**

-

   ***get()* -** This methods requires the URL of the application. After opening the browser, we need to use this method to open any particular application.

   ***navigate()* –** this method is used to navigate backward and forward & refreshing too. For example *Navigate().forward(), navigate().backward(), navigate().refresh().*

   ***navigate().to("—url-- ")* –** this command also used to open any particular application by passing a url.

5. **What is API , where it is being used**

- An API is the tool that makes a website's data digestible for a computer. Through it, a computer can view and edit data.
- Using the Amazon API, a third party Web site can post direct links to Amazon products with updated prices and an option to "buy now."

6. **Difference between quit() & close()**

   quit() method closes the browser where as ***close()*** method closes current tab opened in Browser if only single tab in opened in a browser then ***close()*** method will closes the whole browser.

7. **How to maximixe & minimize the browser**

   - *driver.manage().window().maximize();*
   - *Dimension n = new Dimension(360,592);*
     Driver.manage().window.setSize(n);
   - *driver.manage().window().setPosition(new Point(-2000. 0)) ;*
   - *driver.manage().window().resize()*

8. **What is Webdriver "interface" or Class**

- Interface

9. **What is Super interface for WebDriver**

- SearchContex

10. **What is WebElement & explain all the Methods available in WebElement**

Selenium Webdriver represents all the **HTML** elements as WebElements. ***WebElement*** is an interface and all the abstract methods in the interface are implemented by the ***RemoteWebElement*** class. Super Interface of WebElement method is ***SearchContext*** and ***TakeScreenShot.***

Some methods of WebElement interface are as follows –

Clear( );	Click () ;	findElement (By by);	findElements (By by);
getAttribute(String) ;	getCssValue(String);	getLocation ( );	getRect ();
getSize ( );	getTagName ( ) ;	getText( ) ;	isDisplayed () ;
isEnabled ();	isSelected () ;	sendKeys();	Submit();

WebElement.findElement() - will use the ***element*** as the scope in which to search for your selector. This means it is generally used for searching for child elements.

WebDriver.findElement() - will use the ***driver*** (i.e the entire page) to search for your given selector.

11. How many locator is available in Webdriver , & which locator is preferred ?

1	className	5	name
2	CSS selector	6	partialLinkText
3	id	7	Tag Name
4	linkText	8	xpath

12. How to check whether object is available in GUI

- *driver.findElement(By.xpath("//------xpath-----")).isDisplayed();// returns Boolean value*
- *driver.findElement(By.xpath("//------xpath-----")).isEnabled();// returns Boolean value*
- *driver.findElement(By.xpath("//------xpath-----")).isSelected();// returns Boolean value*

13. How to check the text from the UI?

- *driver.findElement(By.xpath("//------xpath-----")).getText();// returns String value*

14. How to capture color , height , width , font –size of the Element?

- *WebElement.getSize();* //returns width and height of the rendered element
- *WebElement.getCssValue(Property name); //returns the value of colour and font size*

15. How to get the Location of the Webelement ?

- *driver.findElement(By.xpath("//------xpath-----")).getLocation();*

16.How to check whether object is selected or not ?

- *driver.findElement(By.xpath("//------xpath-----")).isSelected(); // returns Boolean value*

17. How to check whether object is enabled in GUI?

- *driver.findElement(By.xpath("//------xpath-----")).isEnabled();// returns Boolean value*

18. How to delete All Cookies

- *driver.manage().deleteAllCookies();*

19. Do we use any constructor in webdriver

20. How to compare Image

21. How to get the WebElement height & width?

Ans:
```
WebElement elmt=driver.findElement(By.xpath("//div[@class='_5iyx']"));
   System.out.println("height SIze "+elmt.getSize().getHeight());
   System.out.println("width SIze "+elmt.getSize().getWidth());
```

22. What is Synchronization?

Ans: Synchronization meaning: when two or more components involved to perform any action, we expect these components to work together with the same pace. The co-ordination between these components to run parallely is called Synchronization. Synchronization (**Wait**) in Selenium has a great significant value.

23. How to handle Synchronization wait available in Webdriver?

Ans: It is a mechanism which involves more than one components to work parallel with Each other.

Generally in Test Automation, we have two components
1. Application Under Test
2. Test Automation Tool.

Both these components will have their own speed. We should write our scripts in such a way that both the components should move with same and desired speed, so that we will not encounter "Element Not Found" errors which will consume time again in debugging.

Synchronization can be classified into two categories:

1. Unconditional
2. Conditional Synchronization

Unconditional :
In this we just specify timeout value only. We will make the tool to wait until certain amount of time and then proceed further.

Examples: Wait() and Thread.Sleep();

The main disadvantage for the above statements are, there is a chance of unnecessary waiting time even though the application is ready.

The advantages are like in a situation where we interact for third party systems like interfaces, it is not possible to write a condition or check for a condition. Here in this situations, we have to make the application to wait for certain amount of time by specifying the timeout value.

Conditional Synchronization:

We specify a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.

It is very important to set the timeout value in conditional synchronization, because the tool should proceed further instead of making the tool to wait for a particular condition to satisfy.

In Selenium we have implicit Wait and Explicit Wait conditional statements

1. Implicit Wait.

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 when we define the implicit wait, it will set for the life of the WebDriver object instance.

It is a mechanism which will be written once and applied for entire session automatically. It should be applied immediately once we initiate the Webdriver.

Implicit wait will not work all the commands/statements in the application. It will work only for "FindElement" and "FindElements" statements.

If we set implicit wait, find element will not throw an exception if the element is not found in first instance, instead it will poll for the element until the timeout and then proceeds further. We should always remember to add the below syntax immediately below the Webdriver statement.

Syntax:

```
driver.manage().timeouts().impilicitlyWait(10.TimeUnit.SECONDS );
```

Explicit Wait:

We need to define a wait statement for certain condition to be satisfied until the specified timeout period. If the Webdriver finds the element within the timeout period the code will get executed.

Explicit wait is mostly used when we need to Wait for a specific content/attribute change after performing any action, like when application gives AJAX call to system and get dynamic data and render on UI.

Example: Like there are drop-downs Country and State, based on the country value selected, the values in the state drop-down will change, which will take few seconds of time to get the data based on user selection.

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.util(ExcpextedCondition.visibilityOfElementLocated(By.id("stateDropDown")));
```

The above statement waits up to 10 seconds before throwing Exception (TimeoutException - Timed out after 10 seconds waiting for visibility of element) or if it finds the element, it will return in 0 - 10 seconds.

Fluent Wait:

Using FluentWait we can define the maximum amount of time to wait for a condition, as well as the frequency with which to check for the condition. And also the user can configure to ignore specific types of exceptions such as "NoSuchElementExceptions" when searching for an element. NoSuchElement exception is thrown by findElement(By) and findElements(By). When ever it try to find any element it returns the first matching element on the current page else it throws NoSuchElementException.

```
FluentWait wait = new FluentWait (driver);
wait.withTimeout(10, TimeUnit.SECONDS);
wait.pollingEvery(10, TimeUnit.SECONDS);
wait.ignoring(NoSuchElementException.class);
```

24. Which wait statement will be used to wait till page load?

We will used pageLoadTimeout .

```
driver.manage().timeouts().pageLoadTimeout(5, TimeUnit.SECONDS);
```

25. How to handle dynamic object?

In dynamic elements, There is always a Fixed part in dynamic elements attribute either in mid or in last or in start. We need to generate the locator using that fixed part.

- **Creating Dynamic CSS-Selector –**

Tag[attribute = 'value'] ← CSS selector structure

- If fixed part is at starting - use **(^)**

  e.g. ***input[id^ = '---Fixedpart---']***

- If fixed part is at mid - use **($)**

  e.g. ***input[id$ = '---Fixedpart---']***

- If fixed part is at ending - use **(*)**

  e.g. ***input[id* = '---Fixedpart---']***

- **Creating Dynamic xpath –**

  // tag[@attribute = ‘value’] 🡐 CSS selector structure

- If fixed part is at starting - use **starts-with**

  e.g. // ***input [starts-with @id, ‘---Fixed part is at starting---’]***

- If Fixed part is at mid or at Ending

  e.g. // ***input [contains(@id, ‘---Fixed part ---’]***

26. Difference between thread wait , implicitly wait , explicitly wait?

- **Thread.sleep (xxxx) ;** stops the execution of code for a specified time period compulsory. This statement force to wait specified time period unconditionally. Time Period is defined in millisecods.

Sr. No	**Implicit Wait**	**Explicit Wait**
1.	Valids for life time of webdriver instance hence once used applicable to all elements	Can apply to particular web element
2	Achieved by using **implicitlyWait**	Achieved by using **fluentWait** and **WebDriverWait**
3.	Polling time is browser specific and fixed and **timeouts** is **customizable**.	**Polling time** and **timeouts** is **customizable**.
4.	this wait doesn’t provide a feature to wait till a certain condition occur	this wait provides a feature to wait till a certain condition occur
5.	We **can’t ignore** any **exception** using implicit Wait	We **can ignore exception** using explicit Wait

6.	**Syntax** : driver.manage().timeouts.implicitlyWait(XX, TimeUnit.*SECONDS*);	**Syntax : 1) using FluentWait**
		FluentWait **w** = new FluentWait(driver);
		w.withTimeout(10,TimeUnit.*SECONDS*);
		
		Syntax : 1) using WebDriverWait
		WebDriverWait **w** = new WebDriverWait(driver, 20);
		w.until(ExpectedConditions.elementToBeClickable(By.id(“ -------- ”)));

7.	throws **NoSuchElementException** after timeout.	throws **NoSuchElementException** after timeout if not ignored

27. What is fluent wait?
Ans: Fluent wait is not a type of wait . we can achieve explicit wait using Fluent wait. Fluent wait is used to customize the Explicite wait.The fluent wait is used to tell 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.

28. How to perform drop down in selenium

Ans: by using Instance of select class and by performing following any method using driver instance.

```
Select month = new Select (driver.findElement(By.name("month")));

month.selectByVisibleText("Jan");

month.selectByIndex();

month.selectByValue("0");
```

29. List out all methods available in Select class?

1	*selectByVisibleText()*	**5**	*deselectByVisibleText()*
2	*selectByValue()/*	**6**	*deselectByValue()*
3	*selectByIndex()/*	**7**	*deselectByIndex()*
4	*isMultiple()*	**8**	*deselectAll()*

30. How to capture all the value from the dropdown?

***Ans**:* below is the code for getting all the options from drop down. getOption method will returns the all elements from the dropdown in List dd.

```
Select month = new Select (driver.findElement(By.name("month")));

List<WebElement> dd = month.getOptions();  // get all options
System.out.println(dd.size());  // get length
For( int i = 0 ; i<dd.size ; i++)
{
  System.out.println(dd.get(i).getText());
}
```

31. How to print only Selected value from the dropdown?

```
Select month = new Select (driver.findElement(By.name("month")));

List<WebElement> dd = month.getOptions();  // get all options
System.out.println(dd.size());  // get length
For( WebElement webEle : dd )
{
  boolean seleted =webElement.isSelected() ;
  if( selected== true)
  System.out.println("Selected Value : "+webElement.getText());
}
```

32. How to capture only non-selected value from the dropdown?

Ans: below is the code for getting non selected values

```
Select month = new Select (driver.findElement(By.name("month")));

List<WebElement> dd = month.getOptions(); // get all options
System.out.println(dd.size()); // get length
For( WebElement webEle : dd )
{
  boolean seleted =webElement.isSelected() ;
  if( selected== false)
  System.out.println("Non Selected Values : "+webElement.getText());
}
```

34. How to select all the similar value from the dropdown

EG we have multiSelect dropdown, like automation testing , manual testing , sql testing , java , we should all the option which contains "testing" word

35. How to work with custom select dropdown/ auto suggest dropDown?

36. How to take mouse over operation on the element?

Ans: Below is the sample code for mouse over in selenium. We have to use Action class for mouse over.

```
Actions act = new Actions(WebDriver) ;

WebElement ele = WebDriver.findElement(By.xpath("---- xpath ------"));

Act.moveToElement(ele).build().perform();
```

37. How to perform keyboard operation?

Ans: We can perform Keyboard actions using Robot class.

38. How to perform "control+c"

Ans – By using Action class

39. Difference between build() & perform()?

Ans:

The ***build()*** method is used compile all the listed actions into a single step. we have to use build() when we are performing sequence of operations and no need to use only if we are performing single action.

example where ***build() required –***

```
Actions builder = new Actions(driver);
builder.clickAndHold((WebElement)listItems.get(0)).clickAndHold((WebElement)listItems.get(3)).click().build
().perform();
```

example where ***build() is not required –***

```
WebElement draggable = driver.findElement(By.id("draggable"));
WebElement droppable = driver.findElement(By.id("droppable"));
new Actions(driver).dragAndDrop(draggable,droppable).perform();
```

40. How to perform drogAndDrop Operation in selenium?

Ans: we can perform drag and drop operation in selenium using Actions class which have dragAndDrop() method. Below is the code for the same.

```
WebElement From =
driver.findElement(By.xpath(".//*[@id='treebox1']/div/table/tbody/tr[2]/td[2]/table/tbody/tr[1]/td[4]/span"));

WebElement To =
driver.findElement(By.xpath(".//*[@id='treebox2']/div/table/tbody/tr[2]/td[2]/table/tbody/tr[1]/td[4]/span"));

Actions builder = new Actions(driver);

Action dragAndDrop = builder.clickAndHold(From)

.moveToElement(To)

.release(To)

.build();

dragAndDrop.perform();

}
```

61. What is framework ?

Ans: Framework is a set of rules, libraries, tools, utility, along with their best practices in place Which help us to automate an application with minimum effort by using reusable components of writing scripts to automate the application along with proper reporting & debugging features.

Types of Framework:
Data Driven Framework
Keyword Driven Framework
Hybrid Framework
Page Object Model
Behaviour Driven Devlopment

62. Which framework you have used & why?

We created Keyword Driven framework along with Data Driven and Page object model
It's a universal framework it can applicable to any project.

63. Explain Framework with component

Ans : Keywords Package:
We create 2 classes named constants.java and Keywords.java. In constants.java, we write all the constants like drivers, webelements, file. We declared all the elements as a private because we want to achieve encapsulation. To access that element, we use getter, setter method.
In Keywords.java, we write keywords like openBrowser method, openURL, Screenshot etc. Basically common methods.

Page Objects Package:

Here, we find the elements using page factory for particular webpage. For one webpage we write one java class with different action performing methods.

TestCasePackage :

We create one base class java class in that we write one method which we run beforetest. In that method, we open browser and URL. The second method in base class is teardown. In that method we quit the driver. This method we run after test.
Another classes are our test case classes which are related to page object java classes. In test cases classes, we create instance of page object java class and then we execute that action methods which we declare in page object java classes.

Utility Package

ExcelReader, TakeScreenshot,Listners and are common classes which are written at once and used multiple times for every test cases.
We write 3 classes in that 2 classes for read data one for properties (readConfig), and another for read test data from excel file (Xutils).
The third java class is reporting, in that we create html formatted report related to test cases.

Configuration folder (Object Repository Package)

Here there are 2 properties files, config file have details of Dev and QA environment (urls & related credentials). In config.properties file, we write baseURL, username, pwd, driver path. This way we are achieving the abstraction since we are not directly referring the locators by its value instead we are passing these as a part of object of properties files and hence achieving the abstraction since we are hiding actual locators and just using its object reference.

Driver folder

Just copy and paste all the drivers here just for simplicity to have all related things under one project.

Screenshot folder

We take screenshot for failed test cases

Extent config:

Log4j
pom.xml
Testng.xml
Extent.xml

64. What is TestNg why it is required ?

TestNG is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired from JUnit which uses the annotations (@).

- Using TestNG you can generate a proper report, and you can easily come to know how many test cases are passed, failed and skipped.
- You can execute failed test case separately

TestNG is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired from JUnit which uses the annotations (@).

- Using TestNG you can generate a proper report, and you can easily come to know how many test cases are passed, failed and skipped.
- You can execute failed test case separately

65. WithOut Testng what are the challenges u faced ?

Ans : We can't do parallel Testing, We can't set priority to testcase/method, Without Testng we can't genearate html report, Without main method we can't execute the test case

66. Why TestNg over Junit ?

Ans : ***Grouping*** is possible in TestNg, ***Parallel*** **Testing** is possible in TestNg, More annotations used as compare to Junit, ***Parameterization*** is possible in TestNg, Support both ***Java*** and ***.net***

67.What is annotation? Explain all annotation
68. Diff between @BeforeTest and @AfterTest
69. Diff between @BeforeSuite and @AfterSuite

Annotaion: Annotations were formally added to the Java language in JDK 5, and TestNG made the choice to use annotations to annotate test classes.

Sr. No.	Annotations	Description
1	**@Test**	Marks a class or a method as a part of the test
2	**@BeforeMethod**	A method which is marked with this annotation will be executed before every ***@test*** annotated method.
3	**@AfterMethod**	A method which is marked with this annotation will be executed after every ***@test*** annotated method.
4	**@BeforeClass**	A method which is marked with this annotation will be executed before ***first @Test*** method execution. It runs only once per class.
5	**@AfterClass**	A method which is marked with this annotation will be executed after all the test methods in the current class have been run
6	**@BeforeTest**	A method which is marked with this annotation will be executed before ***first @Test*** annotated method.
7	**@AfterTest**	A method which is marked with this annotation will be executed when **all *@Test*** annotated methods complete the execution of those classes which are inside *<test>* tag in *testng.xml* file.
8	**@BeforeSuite**	A method which is marked with this annotation will run **only once before** all tests in the suite have run
9	**@AfterSuite**	A method which is marked with this annotation will run **once after** execution of all tests in the suite have run
10	**@BeforeGroups**	This annotated method will run **before the first test run** of that specific group.
11	**@AfterGroups**	This annotated method will run **after all test methods** of that group completes its execution.
12	**@Parameters**	This annotation is used to pass parameters to test methods.
13	**@Listeners**	This annotation is used with test class. It helps in writing logs and results.
14	**@ Factory**	Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ].
15	**@DataProvider**	we use @DataProvider annotation for data supplier method which return Object[][]. which we can use in @Test annotated method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.

Benefits of Using Annotations

Following are some of the benefits of using annotations −

- TestNG identifies the methods it is interested in, by looking up annotations. Hence, method names are not restricted to any pattern or format.
- We can pass additional parameters to annotations.
- Annotations are strongly typed, so the compiler will flag any mistakes right away.
- Test classes no longer need to extend anything (such as TestCase, for JUnit 3).

70. Explain Hierarchy of Testng annotation ?

Ans :

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

71. What is batch execution and how to achive batch execution?

Ans : Collection of multiple test is called batch ,We can execute multiple test case through testing.xml

72. Write a syntax of testing xml?

Ans :

```
<suite>
   <listeners>
      <listener name="value">
      <test>
         <classes>
             <class>
            <methods>
```

73. Write a grouping execution and How to achive group Execution ?

Ans :

```
@Test(groups={"smokeTest","RegressionTest"})
public void loginTest(){
   Sopln("loginsuccessfully");
}
```

Structure in Testng.xml:

```
<run>
  <groups>
    <define name = "all">
     <include name = "smokeTest"/>
```

```
                                      <include name = “RegressionTest”/>
                                    </define>
                                 <run>
                                 <define name = “all”/>
                                </run>
                              </groups>
```

74. What is parallel execution and how to achive parallel execution?
Ans : We can use “parallel” attribute in testing.xml to accomplish parallel test execution in TestNG.

75. How to achive cross browser Testing using selenium?
Ans : Using @parameter annotation
Syntax: <parameter name = “browser” value= “chrome”/>

76. How to disable in TestNg ?
Ans : There are Three ways
@Test(enabled=false)
@Ignore
Throw Skip exception in method

77. How to execute same test with multiple time ?
Ans : We write **@Test(invocation count =5)**
It means above TestCase run 5 times continuously

78.What is assertion and How many assertion you used in realtime selenium script ?

Ans: Assertions verify that the state of the application is same to what we are expecting. Selenium Assertions can be of three types: “assert”, “verify”, and ” waitFor”. When an “assert” fails, the test is aborted. When a “verify” fails, the test will continue execution, logging the failure. A “waitFor” command waits for some condition to become true. They will fail and halt the test if the condition does not become true within the current timeout setting. Perhaps, they will succeed immediately if the condition is already true.

79. what is @Parameter annotation in Testng & how to send parameter to Testng test ?
Ans :

```
Public class parameterizedTest(){
@Test
@parameters(“browser”)
public void parameterizedTest(String browser){
   If(browser.equals(“chrome”)){
     Sopln(“open chrome browser”);
   }else if(browser.equals(“firefox”))
    {
     Sopln(“open firefox browser”);
    }
```

80. How to Execute same test with multiple data ?
-- Using Data Provider

81. What is the @Listener annotations in TestNG ?
-- Listeners “listen” to the event defined in the selenium script and behave accordingly. The main purpose of using listeners is to create logs. There are many types of listeners such as WebDriver Listeners and TestNG Listeners.

82. How to execute only failed test only , when batch execution is Done?
Ans : After the batch execution “refresh the folder” than automatically we get testing-falied.xml (inside test-output), just the that xml file.

83. Difference between TestNG Listeners and webdriver Listeners
Ans : Listeners are those which will listen to event.

In case of ***WebDriver***, whenever anyone clicks on a button on a web page, then automatically a method will be called (***actionPerformed***() in ActionListener case). Upon Action method will be called.

But in case of ***TestNG***, there are the actual "listeners" which listen (here the method gets called automatically) to the ***test execution events***. A few examples are: ***onStart***(), ***beforeStart***(), ***afterFinish***(), and ***onFinish***() etc. Mostly, the ***TestNG*** automation teams implement their own custom listeners for custom Logging and reporting.

WebDriver listeners too do a similar job...of logging and/or reporting. But then both of them work on different event sets. ***WebDriver works on different automation events whereas TestNG works on different test's related events.*** The point to note is that, with WebDriver listeners, "Logging" happens before/after event occurance such as click/SendKeys etc.

84. How to execute failed test multiple times ?
Ans : using ITestREtry

85. Whenever we get build , which test-scripts you will execute first
Ans : Using grouping concept , we will execute smokeTest first.

Block -2 : SVN interview Questio

86. What is SVN or GitHub?
Ans : GitHub is a **Git** repository hosting service, but it adds many of its own features. While **Git** is a command line tool, GitHub provides a Web-based **graphical** interface. It also provides **access** control and several collaboration features, such as a wikis and basic task management tools for every project.

87. What is the Role of SVN /Github in Automation?
Ans:

88. What store & get File from SVN?
Ans:

89. What is the advantages of SVN?
Ans:

Block- 3 : Maven Interview Question

90. What is Maven/ ANT ?
Ans:

91. What is Role of Maven FrameWork ?
Ans:

92. What is dependencies , how to handle dependencies in framework ?
Ans:

93. What of Project Object Model [POM .xml]
Ans:

94. Explain the maven Plugin you used in your framework
Ans:

95. How to Execute TestNG.xml in Maven POM.xml.
Ans:

Block- 4 : Jenkins Interview Question

96. What is Jenkins?
Ans:

97. What is the Role of Jenkins in Framework?
Ans:

98. What is the Advantages of Jenkins?
Ans:

99. How to configure Jenkines?
Ans:

Selenium miscellaneous interview question

100. Difference between xpath & css-selector?
Ans:

101. Where we have used contractor in selenium?
Ans:

102. How to execute Java-scripts in Selenium , or how to work with browser scroll bar ?
Ans:

103. How to handle SSL popup in IE ?
Ans:

104. What is use Selenium-Grid ?
Ans:

105. What is the Use DEsiredCapabalites in SELENUM ?
Ans:

106. Non automatable test case in your Project ?
Ans: