1. Selenium Introduction:

* Platform: Windows, Mac OS, Linux, Solaris
* Browser: Selenium WebDriver supports a diverse range of web browsers such as Firefox, Chrome, Internet Explorer, Opera and many more. It also supports some of the non-conventional or rare browsers like HTML Unit.
* Programming language: [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)), [Java](https://en.wikipedia.org/wiki/Java_(software_platform)), [Perl](https://en.wikipedia.org/wiki/Perl), [PHP](https://en.wikipedia.org/wiki/PHP), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)) and [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language))

2. Advantages & Disadvantages of selenium:

Advantages:

* Selenium is an Open Source Software.
* Selenium supports various programming languages to write programs (Test scripts)
* Selenium supports various operating systems (MS Windows, Linux, Macintosh etc…)
* Selenium supports various Browsers (Mozilla Firefox, Google Chrome, IE, Opera, Safari etc…)
* Selenium supports Parallel Test Execution.
* Selenium uses less Hardware resources.

Disadvantages:

* No reliable Technical Support from anybody.
* It supports Web based applications only.
* Difficult to use, takes more time to create Test cases.
* Difficult to Setup Test Environment when it compares to Vendor Tools like UFT, RFT, Silk Test etc…
* Limited support for Image Testing.
* New features may not work properly.
* No Test Tool integration for Test Management.
* No Built-in Reporting facility.

3. What are the different Selenium components?

Selenium Components:

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

**Selenium IDE:**

Selenium IDE is a simple record and playback kind of tool which comes as an add-on for Mozilla Firefox only. It is used for prototype testing.

**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.

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 java script and responds to a server, which again interprets the command and returns to code in the respective language.

**Selenium WebDriver:**

Selenium WebDriver is the most important tool of the Selenium suite. 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.

**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. There is a Hub which controls the execution on various machines, and there are multiple nodes on which actual implementation is done.

4. What are the different types of locators in Selenium? Give one example for each

Selenium Locators:

* [ID](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid1)
* [Name](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid2)
* [Linktext](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid3)
* [Partial Linktext](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid4)
* [Tag Name](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid5)
* [Class Name](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid6)
* [DOM Locator](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid7)
* [CSS Selector](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid8)
* [Xpath](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid9)

**ID:**

<div class="inputM make\_relative"> <input id="ch\_login\_email" type="text" required="">

Eg: driver.findElement(By.id("ch\_login\_email"))

[**Name**](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid2)**:**

<input type="email" placeholder="Work Email\*" name="email" value="" class="form-control sign-up-input-2 ">

Eg: driver.findElement(By.name("email"));

[**Linktext**](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid3)**:**

<a href="https://www.lambdatest.com/blog" target="\_blank">Blog</a>

Eg: driver.findElement(By.linkText("Blog"));

[**Partial Linktext**](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid4)**:**

<a href="https://accounts.lambdatest.com/register" target="\_blank">START TESTING<i class="fa fa-arrow-circle-right" aria-hidden="true"></i></a>

Eg: driver.findElement(By.PartialLinkText("Testing"));

[**Tag Name**](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid5)**:**

Eg: driver.findElements(By.tagName(a));

[**Class Name**](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid6)**:**

<input type="email" name="email" value="" placeholder="Email" required="required" autofocus="autofocus" class="form-control mt-3 form-control-lg">

Eg: driver.findElement(By.className("form-control mt-3 form-control-lg "));

[**DOM Locator**](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid7)**:**

In document object model(DOM) we locate element in terms of DOM model. As explained above we can identify element via ID and name through methods of the DOM like ‘getElementById’ and ‘getElementsByName’. The method getElementById will locate only one element at a time, whereas the other method is used to provide an array of elements located by that name. In order to access any specified element in case of an array of elements, we can use index.

<input type="checkbox" name="remember" id="remember" class="form-check-input">  
Eg:document.getElementById(“remember”)  
document.getElementsByNames(“remember”)[0]

[**CSS Selector**](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid8)**:**

CSS Selectors can be located through various formats available:

* Tag and ID

Syntax: css=(Html tag )(#) (value of the ID attribute)

Eg: driver.findElement(By.cssSelector("input# ch\_login\_email "))

* Tag and Class

Syntax: css=(HTML tag)(.)(Value of Class attribute)

Eg: driver.findElement(By.cssSelector("input.form-control mt-3 form-control-lg"))

* Tag and Attribute

Syntax: css=(HTML Page)[Attribute=Value]

Eg: driver.findElement(By.cssSelector("input[name= ‘phone’]"))

* Tag, Class and Attribute

Syntax: css=(HTML tag>)(. )(Class attribute value)([attribute=Value of attribute])

Eg: driver. findElement(By.cssSelector(“button. btn sign-up-btn-2 btn-block [type = submit]”))

* Matches (Starts with, Ends with, Contains)

Starts with:

Syntax: css=(HTML tag)([attribute^=start of the string])

Eg: driver.findElement(By.cssSelector("input[name^='em']"))

Ends with:

Syntax: css=(HTML tag)([attribute$=end of the string])

Eg: driver.findElement(By.cssSelector("input[name$=’ail’]"))

Contains:

Syntax: css=(HTML tag)([attribute\*=partial string])

Eg: driver.findElement(By.cssSelector("input[class\*=’control’]"))

* Child elements

In order to locate element, the following syntax would be used:

Syntax: Css= tagname.class name li:nth-of-child(index of the referenced child which in our case is 3)  
  
Eg: driver.findElement(By.cssSelector(“ul. overview-list li:nth-of-child(3)”);

Similarly in order to access responsive we can use, last-child reference as below:

Syntax: Css= ul. overview-list li:last-child  
  
Eg: driver.findElement(By.cssSelector(“ul. overview-list li:last-child”);

[**Xpath**](https://www.lambdatest.com/blog/locators-in-selenium-webdriver-with-examples/#testid9)**:**

* Standard Xpath

Syntax: Xpath: //tagname[@attribute=’value’]

Eg: driver.findElement(By.xpath(“//input[@name= ’email’]”))

* Using Contains

Syntax: Xpath: //tagname[contains(@attribute, ‘partial value of attribute’)]

Eg:driver.findElement(By.xpath(“//input[contains(@class, ‘form-control’)]”))

* Using Xpath with AND & OR

Syntax using OR:

Syntax: Xpath=//input[@id='login\_1' OR @name='login’]

Eg:driver.findElement(By.xpath(“//input[@type='email' OR @name='email’]))

Syntax using AND:

Syntax: Xpath=//input[@id='login\_1' AND @name='login’]

Eg:driver.findElement(By.xpath(“//input[@type='email' AND @name='email’]))

* Using starts-with

Syntax: Xpath=//tagname[starts-with(@attribute,'starting name of the attribute value')]

Eg:driver.findElement(By.xpath(“//input[starts-with(@name,'pass')]”))

* Using text in Xpath

Syntax: Xpath=//div[text()='Logged In']

Eg:driver.findElement(By.xpath(“//p[@text()=’ SIGN UP’]”))

5. What is the difference between assert and verify commands?

**Assert:** Assert command checks if the given condition is true or false. If the condition is true, the program control will execute the next phase of testing, and if the condition is false, execution will stop and nothing will be executed.

**Verify:** Verify command also checks if the given condition is true or false. It doesn't halts program execution i.e. any failure during verification would not stop the execution and all the test phases would be executed.

6. What is a framework?

Selenium framework is a code structure for making code maintenance simpler, and code readability better. A framework involves breaking the entire code into smaller pieces of code, which test a particular functionality.

The code is structured such that, the “data set” is separated from the actual “test case” which will test the functionality of the web application. It can also be structured in a way wherein, the test cases which need to be executed are called (invoked) from an external application (like a .csv).

There are a number of frameworks out there, but 3 commonly used Selenium framework (s) are:

* [Data Driven framework](https://www.edureka.co/blog/selenium-framework-data-keyword-hybrid-frameworks#DataDrivenFramework)
* [Keyword Driven framework](https://www.edureka.co/blog/selenium-framework-data-keyword-hybrid-frameworks#KeywordDrivenFramework)
* [Hybrid framework](https://www.edureka.co/blog/selenium-framework-data-keyword-hybrid-frameworks#HybridFramework)

7. What are the advantages of Automation framework?

* Increased code reusage
* Improved code readability
* Higher portability
* Reduced script maintenance

8. What is Junit?

JUnit is a unit testing framework for Java programming language. It plays a crucial role test-driven development, and is a family of unit testing frameworks collectively known as xUnit.

JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented.

9. What are Junit annotations?

* Before
* After
* BeforeClass
* AfterClass
* Ignore
* RunWith
* Test

Below is the list of important and frequently used annotations:

1. @Before
2. @BeforeClass
3. @After
4. @AfterClass
5. @Test
6. @Ignore
7. @Test(timeout=500)
8. @Test(expected=IllegalArgumentException.class)

see below table to understand more on annotation :

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | | Annotations | Description |
| 1. | @Test | | This annotation is a replacement of org.junit.TestCase which indicates that public void method to which it is attached can be executed as a test Case. |
| 2. | @Before | | This annotation is used if you want to execute some statement such as preconditions before each test case. |
| 3. | @BeforeClass | | This annotation is used if you want to execute some statements before all the test cases for e.g. test connection must be executed before all the test cases. |
| 4. | @After | | This annotation can be used if you want to execute some statements after each[Test Case](https://www.guru99.com/test-case.html)for e.g resetting variables, deleting temporary files ,variables, etc. |
| 5. | @AfterClass | | This annotation can be used if you want to execute some statements after all test cases for e.g. Releasing resources after executing all test cases. |
| 6. | @Ignores | | This annotation can be used if you want to ignore some statements during test execution for e.g. disabling some test cases during test execution. |
| 7. | @Test(timeout=500) | | This annotation can be used if you want to set some timeout during test execution for e.g. if you are working under some SLA (Service level agreement), and tests need to be completed within some specified time. |
| 8. | @Test(expected=IllegalArgumentException.class) | | This annotation can be used if you want to handle some exception during test execution. For, e.g., if you want to check whether a particular method is throwing specified exception or not. |