**Automation Testing** : **Automation Testing** is a software testing technique that performs using special automated testing software tools to execute a test case suite.

**Selenium : Selenium is a suite of tools and libraries used for automating web browsers. Selenium supports automation across different browsers, platforms and programming languages.**

Selenium can be easily deployed on platforms such as Windows, Linux, Solaris and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile and android.

Selenium supports a variety of programming languages through the use of drivers specific to each language.Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby.Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

Selenium can be used to automate functional tests and can be integrated with automation test tools such as **Maven**, **Jenkins**, **& Docker** to achieve continuous testing. It can also be integrated with tools such as **TestNG**, & **JUnit** for managing test cases and generating reports.

**Difference Between Manual Testing and Automated Testing**

| **Parameters** | **Manual Testing** | **Automation Testing** |
| --- | --- | --- |
| **Definition** | In manual testing, the test cases are executed by the human tester. | In automated testing, the test cases are executed by the software tools. |
| **Processing Time** | Manual testing is time-consuming. | Automation testing is faster than manual testing. |
| **Resources requirement** | Manual testing takes up human resources. | Automation testing takes up automation tools and trained employees. |
| **Exploratory testing** | Exploratory testing is possible in manual testing. | Exploratory testing is not possible in automation testing. |
| **Framework requirement** | Manual testing doesn’t use frameworks. | Automation testing uses frameworks like Data Drive, Keyword, etc. |
| **Reliability** | Manual testing is not reliable due to the possibility of manual errors. | Automated testing is more reliable due to the use of automated tools and scripts. |
| **Investment** | In manual testing, investment is required for human resources. | In automated testing, investment is required for tools and automated engineers. |
| **Test results availability** | In manual testing, the test results are recorded in an excel sheet so they are not readily available. | In automated testing, the test results are readily available to all the stakeholders in the dashboard of the automated tool. |
| **Human Intervention** | Manual testing allows human observation, thus it is useful in developing user-friendly systems. | Automated testing is conducted by automated tools and scripts so it does not involve assurance of user-friendliness. |
| **Performance testing** | Performance testing is not possible with manual testing. | Performance testing like load testing, stress testing, spike testing, etc. |
| **Batch Testing** | In manual testing, batch testing is not possible. | You can batch multiple tests for fast execution. |
| **Programming knowledge** | There is no need for programming knowledge in manual testing. | Programming knowledge is a must in case of automation testing as using tools requires trained staff. |
| **Documentation** | In manual testing, there is no documentation. | In automation testing, the documentation acts as a training resource for new developer. He/ She can look into unit test cases and understand the code base quickly. |
| **When to Use?** | Manual testing is usable for Exploratory testing, Usability testing, and Adhoc testing. | Automated testing is suitable for Regression testing, Load testing, and Performance testing. |

Advantages and Disadvantages of Selenium

|  |  |
| --- | --- |
| **Advantage** | **Disadvantage** |
| Open source software | Only suited for web applications.Not suited for desktop applications |
| Support multiple programming language | No built in reporting |
| Support all operating systems | Limited support for image based testing |
| Support multiple browsers | Atleast one programming language knowledge required |
| Support parallel test execution | It doesnot provide any test tool for test management. |
| Selenium can be easily integrated with various testing frameworks like TestNG, JUnit, NUnit, etc. |  |
| test scripts directly interact with web browsers to automate testing activities. |  |

**Selenium Components**

1. **Selenium IDE :**

* Selenium IDE (Integrated Development Environment) is the major tool in the Selenium Suite.
* It is implemented as a Firefox Add-On and as a Chrome Extension.
* It provides record and playback functions.
* Create, edit,execute and debug test cases.
* Export test case in different programming languages.

Drawbacks

* Supports only firefox browser
* Not suited for dynamic web applications.
* No need of programming logic
* Data driven testing not possible.
* Centralized maintenance of objects not possible.

1. **Selenium RC**

* It allowed testers to write automated web application tests in various programming languages like Java, C#, Python, etc.
* The key feature of Selenium RC was its ability to interact with web browsers using a server, which acted as an intermediary between the testing code and the browser.

Drawbacks

* Communication possible through RC server. So Rc is time consuming

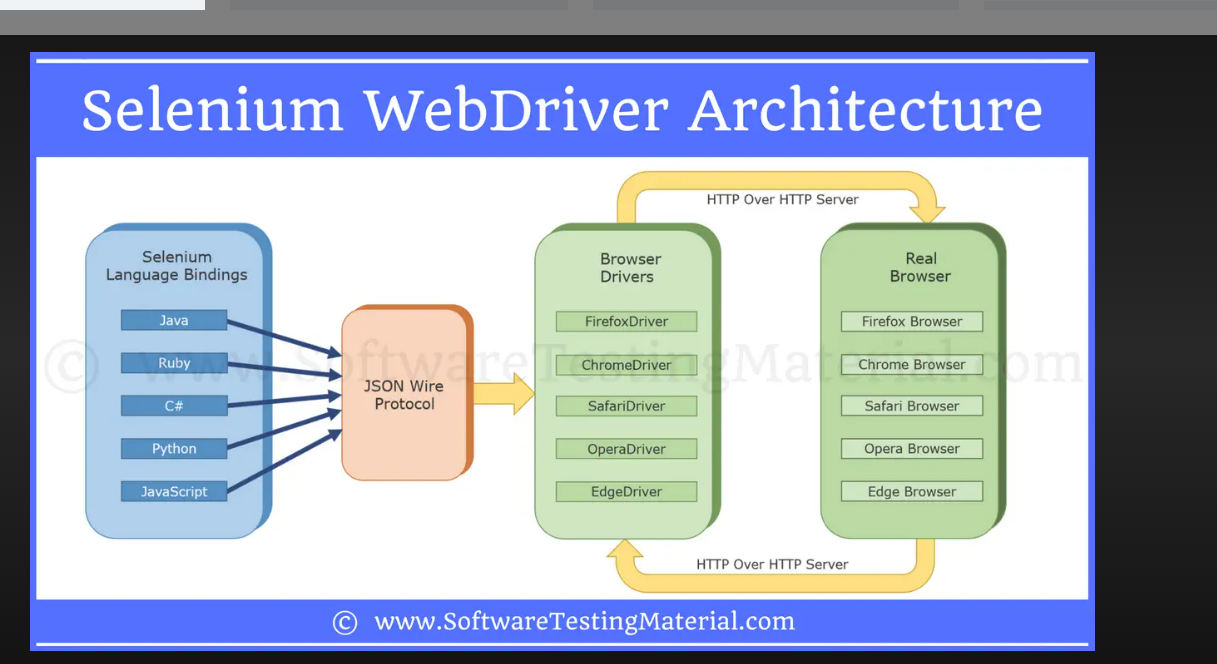
1. **Selenium Web Driver**
2. **Selenium Grid**

* Parallel execution of test cases. Parallel execution done with the help of hub- node architecture.
* Cross-Browser Testing: Test your application across different browsers simultaneously.
* Cross-Platform Testing: Test your application on different operating systems concurrently
* Testcases write with the help of Rc and web Driver.

**Drawbacks**

* Execution done with the help of RC and Web Driver

**Selenium Web Driver Architecture**

****

### Components of Selenium 3 WebDriver**:**

1. **Selenium Client Library:** This component provides language-specific bindings or APIs (e.g., Java, Python, Ruby) that allow users to write test scripts and interact with the WebDriver.
2. **JSON Wire Protocol over HTTP:** The JSON Wire Protocol is a standardized protocol used for communication between the Selenium Client Library and the Browser Drivers. It defines a set of commands and responses in JSON format exchanged over HTTP requests.
3. **Browser Drivers:** These are executable files that establish a communication channel between the WebDriver and the actual web browsers such as Chrome, Firefox, Safari, etc. Each browser requires its specific driver (e.g., ChromeDriver, GeckoDriver, etc.) to enable WebDriver to control and automate browser actions.
4. **Real Browsers:** These are web browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, etc., where the actual testing and automation take place. The WebDriver interacts with these browsers through their respective browser drivers to perform actions like clicking elements, filling forms, navigating pages, and validating content.

### **Working of**Selenium 3 WebDriver**:**

1. Write a test script using the Selenium client library in your preferred language.
2. The test script sends commands through the client library to interact with the browser.
3. The client library converts commands into JSON format and sends them via HTTP request.
4. The browser driver decodes JSON commands and interacts with the real web browser.
5. Browser performs actions (e.g., clicking buttons, entering text) on the web page based on the received commands.

**Commands in Selenium**

1. Webdriver Commands
2. Navigation Commands
3. WebElements

|  |  |  |  |
| --- | --- | --- | --- |
| **Commands** | **Usage** | **Syntax** | **Example** |
| getTitle() | For retrieve the title of the currently opened web page | String title= driver.getTitle(); |  |
| getCurrentUrl() | For retrieve the url of currently opened web page | String url= driver.getCurrentUrl(); |  |
| getWindowHandle() | For retrieve the Handleid of currently opened web page | String url= driver.getWindowHandle(); |  |
| getPageSource() | For retrieve the html code of currently opened web page | String url= driver.getPageSource(); |  |
| close() | For closing currently opened web page | driver.close(); |  |

**Navigation Commands**

|  |  |  |  |
| --- | --- | --- | --- |
| **Commands** | **Usage** | **Syntax** | **Example** |
| **back()** | go back to the previous page that you visited. | driver.navigate().back(); |  |
| **forward()** | redirect to the page you were in before clicking on the back button. | driver.navigate().forward (); |  |
| **refresh()** |  | driver.navigate().refresh(); |  |
| **to()** | It creates a new browser window with a new web page. It takes a String parameter and returns a void value. | driver.navigate().to(“http://google.com”); |  |

WebElement elementname= driver.findElement(By.Locator(“Locator value”));

[**https://www.saucedemo.com/**](https://www.saucedemo.com/)

**Web Element**

The term web element refers to a HTML element. The HTML documents are composed of HTML elements. It consists **a start tag, an end tag** and the **content** in between.

**Web Element Commands**

**Find Element**

find Element command is used to uniquely identify a (one) web element within the web page.

Return type – WebElement.

Syntax for finding WebElement

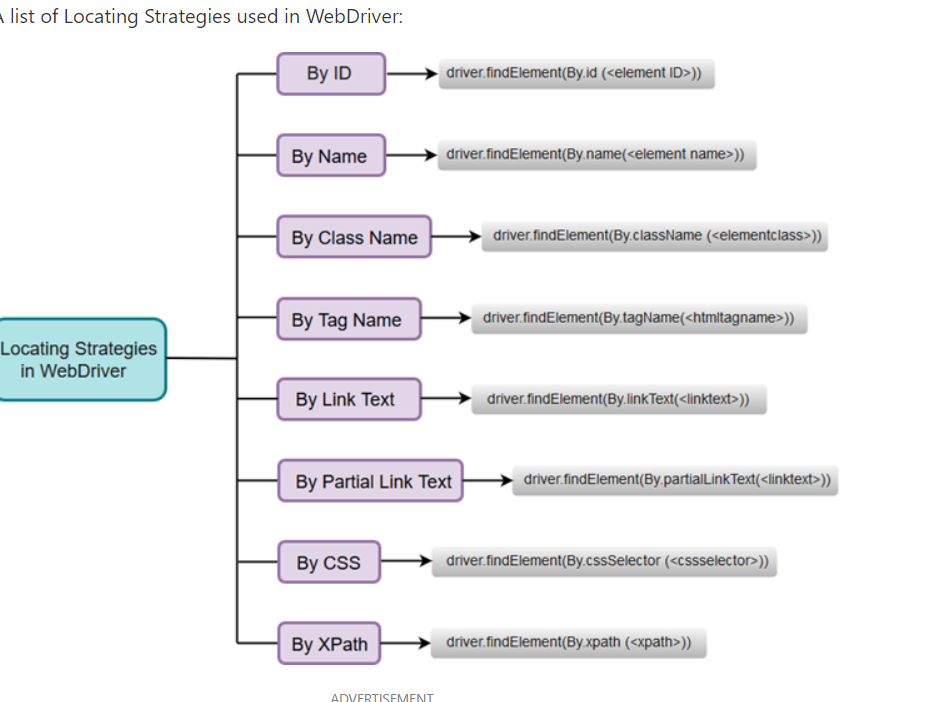
**WebElement elementname= driver.findElement(By.locator(“locator value”));**

WebElement -interface

By-class(locators are in By class)

**Locators**

Locator Value is the unique value using which a web element can be identified



* Most preferred locator is id
* Second preferred – xpath,css Selector(it can be customised).

**Web Element Commands**

|  |  |  |  |
| --- | --- | --- | --- |
| Commands | Usage | Syntax | example |
| click() |  |  |  |
| sendKeys() | The sendKeys method is used to send any keyboard key to an element on the webpage. | element.sendKeys(“type value” | z |
| getText() | This command gives us the text of the particular element visible on the webpage. | element.getText(); |  |
| isSelected() | This method tells if an element is selected or not. It returns a boolean value | element.isSelected(); |  |
| isEnabled() | This method tells if the element is enabled or not and returns a boolean value | element.isEnabled(); |  |
| isDisplayed() | This method tells if an element is present or not and returns a boolean value | element.isDisplayed(); |  |

**close() :** the close() method closes the currently opened window

**quit() :** the quit() method quits the driver

**Alerts**

Alerts are small message box which appears on the webpage to convey some informations to user.

3 types of Alerts

1. Simple Alerts : displays some information or warning on the screen
2. Confirmation Alerts : This confirmation alert asks permission to do some type of operation.
3. Prompt Alerts : This Prompt Alert asks some input from the user

|  |  |
| --- | --- |
| **Commands** | **Uses** |
| alert.dismiss() | For click cancel button in alert box |
| alert.accept() | For click ok button in alert box |
| alert.getText() | For get text in alert box. |
| alert.sendKeys() | To input some data to alert box |

**Syntax for sitching to alert window**

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