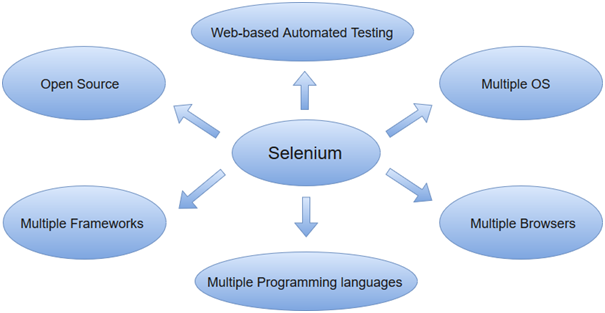
**What is Selenium?**

* Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite.
* It supports automation across different browsers (google chrome, fire fox etc..), platforms (windows, Linux, etc..) and programming languages (java, python, c#, PHP etc..).
* It is used for automation testing.
* It is web based automation testing.
* Selenium supports a variety of programming languages through the use of **drivers** specific to each language.
* 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.
* It 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.



**Automation Testing**

* Manually when testers write the test cases and execute them repeatedly, known as manual testing which is time-consuming and the test results are not sure.
* So, to recover these drawbacks, automation testing came into existence. With automation testing, automation tools are used to execute the test cases.
* It can be used to test the software without human power.
* Automation testing is best suited for areas like load testing, regression testing, repeated execution and performance testing.
* Automation helps companies take new features to market instantly in the testing world and ensure a bug-free user experience.

***"Automation testing refers to the automatic testing of the software in which developer or tester write the test script once with the help of testing tools and framework and run it on the software. The test script automatically test the software without human intervention and shows the result (either error, bugs are present or software is free from them)."***

**Manual and Automation**

|  |  |
| --- | --- |
| Manual testing | Automation testing |
| Testing in which a human tester executes test cases | In automation testing, automation tools are used to execute the test cases |
| In this testing, human resources are involved, that's why it is time-consuming | It is much faster than the manual testing |
| It is repetitive and error-prone | Here automated tools are used that make it interesting and accurate |
| BVT (build verification testing) is time-consuming and tough in manual testing | It's easy to build verification testing |
| Instead of frameworks, this testing use checklist, guidelines, and stringent process for drafting test cases. | Frameworks like keyword, hybrid, and data drive to accelerate the automation process. |
| The process turnaround time is higher than the automation testing process (one testing cycle takes lots of time) | It completes a single round of testing within record time; therefore, a process turnaround time is much lower than a manual testing process. |
| The main goal of manual testing is user-friendliness or improved customer experience. | Automation testing can only guarantee a positive customer experience and user-friendliness. |
| It is best for usability, exploratory and adhoc testing | It is widely used for performing testing, load testing and regression testing. |
| Low return on investment | The high return on investment |

**Advantage of Selenium**

* It is open-source software.
* It supports a wide range of browsers including Chrome, Firefox, Internet Explorer, Edge, and Safari.
* It supports multiple programming languages such as Java, C#, Python, Perl, and Ruby, which allows developers to write test scripts in their preferred language.
* It can be used to automate the testing process of web applications, by simulating user interactions and providing support for test case organization, test execution, test reporting, and test execution management.
* Selenium provides a WebDriver API that allows developers to interact with web pages, simulate user interactions, and automate web browsers.
* Selenium Grid allows you to run test scripts on different machines and browsers in parallel, which can significantly reduce the overall execution time of tests.
* Selenium IDE is a browser extension that allows you to record and playback test scripts, this makes it easy to create test scripts without having to write any code.
* Selenium also provides support for Plug-ins; this allows you to extend its functionality and add new features.

**Dis-advantage of Selenium**

* Only supports web-based applications
* Limited support for mobile browsers
* Requires programming skills
* Limited support for handling dynamic web pages
* Limited support for handling browser plugins
* Limited support for handling browser-specific features
* Limited support for handling browser-specific bugs

**Components or tools**

It is four types. They are

1. Selenium IDE (Integrated Development Environment)
2. Selenium RC (Remote Control)
3. Selenium WebDriver
4. Selenium Grid

**Selenium IDE**

* It is the tool you use to develop your Selenium test cases.
* This part is an add-on for your browser that lets you record and replay test scripts.
* You can construct test scripts with Selenium IDE's record and playback features without writing any code.
* It is a Firefox plug-ins.
* It can be used for non-technical and technical person.
* It is used to create and edit test cases.
* It is also used for debugging.
* It is used for export the different programming language.

**Drawback:**

* It only use Mozilla Firefox.
* We cannot use dynamic web application and data driven testing.

**Selenium Rc**

* This component allows you to write test scripts in different programming languages such as Java, C#, Python, Perl, and Ruby.
* There is no interaction between test script and web browser. So, uses Rc server and this server is very slow.
* It allows for the execution of the test scripts on a remote machine.
* This component only uses the technical people.

**Selenium Web Driver**

* This is the main component of Selenium that controls a browser and simulates user interactions with web pages.
* Much faster than Rc.
* Each browser has different driver.
* WebDriver offers an API for navigating between pages, pressing buttons, and filling out forms on online sites.
* For several browsers, including Chrome, Firefox, Internet Explorer, and Edge, the WebDriver is offered.
* Supports all web browsers and all languages.
* It overcome file upload and popup.

**Drawback:**

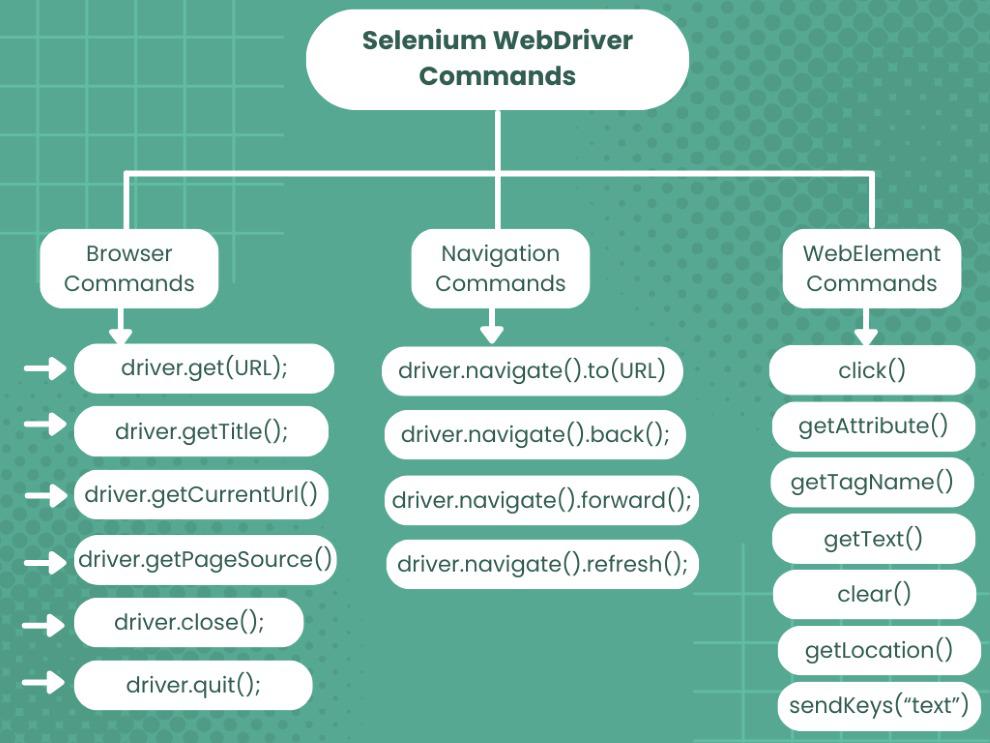
* It generates the reports but cannot generate detailed report.

**Selenium Grid**

* It does parallel execution because this component allows you to run test scripts on different machines and browsers in parallel.
* It allows you to distribute test execution across multiple machines, thus reducing the overall execution time.
* It supports Rc and web driver.
* It uses an architecture name as hub node used for run parallel execution.

**Selenium Commands**

* Selenium commands are the set of commands that are used to run our Selenium tests.
* we have an entirely different set of commands for performing different operations.



**1.Browser commands**

* The techniques used to run Selenium test automation scripts are known as Selenium WebDriver commands.
* Since we are using Selenium WebDriver with Java, commands are simply methods written in Java language.
* A java method is a collection of statements that are grouped together to perform a specific operation
* Insatiate a chrome driver class.

WebDriver driver=new ChromeDriver();

|  |  |  |
| --- | --- | --- |
| commands | use | syntax |
| maximize() | Used to maximize the window. | driver.manage().window().maximize(); |
| get(String url) command | It loads a new web page in the current browser window | driver.get(URL); |
| getTitle() Command | It gets the title of the current web page displayed in the browser.  It does not accept any parameters. It returns the title of the specified URL as a string. | driver.getTitle(); |
| getCurrentUrl() Command | It gets the URL of the current web page shown in the browser.  It does not accept any parameters and returns the URL as a string. | driver.getCurrentUrl() |
| getPageSource() Command | It gets the entire page source of the current web page loaded in the browser.  It does not accept any parameters, but it does return the page source as a string. | driver.getPageSource(); |
| close() Command | It closes the current browser window or tab.  Also, this command does not accept any types of parameters.  It also does not return anything. | driver.close(); |
| quit() Command | t closes all the browser windows and tabs for a particular WebDriver session.  This command does not accept any parameters and not return anything. | driver.quit(); |

**2.Navigation commands**

* This commands provide an efficient way to manage a browser's history and perform actions like going back and forward between pages and refreshing the current page.

|  |  |  |
| --- | --- | --- |
| commands | use | syntax |
| back() Command | Moves back one step in the browser’s history stack.  It does not accept any parameters and does not return anything. | driver.navigate().back(); |
| forward() Command | Moves forward one step in the browser’s history stack.  It does not accept any parameters and does not return anything. | driver.navigate().forward(); |
| to() Command | Loads a new web page in the current browser window.  It accepts a string parameter that specifies the URL of the web page to be loaded. | driver.navigate().to(URL); |
| refresh() Command | Reloads the current web page in the browser window.  It does not accept any parameters and does not return anything. | driver.navigate().refresh(); |

**3.WebElement commands**

* An HTML element on a website serves as the Selenium WebElement.
* HTML elements are used in HTML documents.
* The Web Element can be recognized using various attributes like id, class name, link text, XPath, etc.
* Every HTML element contains both a start tag and an end tag.
* Between the tags is the content.
* To find web elements **:**

Syntax:

WebElement element\_name=driver.findElement(By.Locator(“locator value));

|  |  |  |
| --- | --- | --- |
| **Commands** | **Uses** | **Syntax** |
| sendKeys() commands | The sendKeys method is used to send any keyboard key to an element on the webpage.  This command is used to type something in a text box/input and also press any keys like enter.  The command expects an argument of the keys to be sent and does not return anything. | // Create WebElement WebElement temp = driver.findElement(By.id("TextBox"));  // Perform sendKeys operation temp.sendKeys("GeeksForGeeks"); |
| click() command | Used to perform click operation on a web element such as a button, link, or checkbox. | / Create WebElement WebElement ele = driver.findElement(By.id("GeeksForGeeks"));  // Perform click operation ele.click(); |
| isSelected() command | It used on some link, radio, checkboxes, dropdowns to check whether element is selected or not.  It returns a boolean value of “true” if element is selected and “false” if it is not selected. | WebElement element = driver.findElement(By.id("software-testing"));  boolean status = element.isSelected(); |
| isDisplayed() command | It verifies whether a web element is present and visible on the web page.  Returns true if the element is displayed and false if not. | WebElement element = driver.findElement(By.id("gfg"));  boolean status = element.isDisplayed(); |
| isEnabled() command | Used to checks if an element is enabled for interaction on the web page or not.  It returns true if the element is enabled and false if not. | / Create WebElement WebElement element = driver.findElement(By.id("GFG"));  // Perform isEnabled operation element.isEnabled(); |

**LOCATORS**

* Locators are used to locate the web element on the basis of HTML tags, attributes and HTML texts.
* There are 8 types of locators in Selenium WebDriver:

|  |  |  |
| --- | --- | --- |
| locator | uses |  |
| **id()** | This locator has the highest precedence when searching for web elements in a web page.  Its value is always unique for the particular web element in the entire web page and hence you won’t get multiple matches when you use this locator.  Whenever there is an id attribute in the HTML code we use this. |  |
| **name()** | Wherever there is a name attribute in the HTML code for any web element then we use this locator. |  |
| **className()** | Whenever there is a class attribute in the HTML code we use this locator. |  |
| **tagName()** | Whenever there is a web element with a unique HTML tag we use this locator. |  |
| **linkText()** | Whenever there is a link with a unique HTML text associated with it we use this locator. |  |
| **partialLinkText()** | Whenever there is a link in a webpage with a lengthy text associated with it, we use this locator by using partial HTML text from the link. |  |
| **cssSelector()** | In comparison to the XPath, the cssSelector is a much faster locator and is much more widely used.  It is more complex than the remaining locators but is most effective as in the absence of certain html tags, we can use this to locate the web element. |  |
| **xpath()** | It is a locator which is used to locate a web element using tags, attributes, and text.  We can use the X Path for HTML documents as well as XML documents.  There are two types of XPath, a. Absolute XPath, b. Relative X Path. |  |

**Absolute and relative Xpath**

**Absolute XPath**: This XPath locates the web element from the root element to the required child node. In real-time automation scripts, we should not use absolute XPath.   
**Note:** For dynamic applications such as an Ajax application, we cannot use absolute XPath.

**Relative XPath**: It is the customized XPath, which finds the elements by using tags, attributes or text.

**DropDown**

* The 'Select' class in Selenium WebDriver is used for selecting and deselecting option in a dropdown.
* The objects of Select type can be initialized by passing the dropdown webElement as parameter to its constructor.

WebElement testDropDown = driver.findElement(By.id("testingDropdown"));

Select dropdown = **new** Select(testDropDown);

|  |  |  |
| --- | --- | --- |
| command | use | syntax |
| selectByVisibleText() | It is used to select an option based on the text over the option. | dropdown.selectByVisibleText  ("Database Testing"); |
| selectByIndex | It is used to select an option based on its index, beginning with 0. | dropdown.selectByIndex(5); |
| selectByValue | It is used to select an option based on its 'value' attribute. | dropdown.selectByValue("Database"); |
| isMultiple() | returns TRUE if the drop-down element allows multiple selection at a time; FALSE if otherwise |  |
| getOptions() | Get all the options from the dropdown list.  (return type-list of web element)  List<WebElement> | Select dropdown =new Select(testDropDown);  List<WebElement> drop\_downlist= dropdown.getOptions() |
| getFirstOption() | Get the first option from the dropdown list. | WebElement firstselector\_country= dropdown.getFirstSelectedOption(); |
| getAllSelectedOption() | Get the all selected options from the drop down list.  (return type-list of web element) |  |

**Action Class in Selenium**

* The Action class in Selenium WebDriver is a utility class that enables advanced user interactions such as mouse and keyboard events.
* This class provides a way to simulate complex user interactions like double-clicking, right-clicking, holding down a key, etc.
* The Actions class provides a fluent interface for building and performing actions, which can be performed on specific web elements.

Methods of Actions Class

* Actions class is broadly divided into two categories:

1) Mouse actions

2) Keyboard actions

**Mouse actions**

* Mouse actions in selenium are the actions that can be performed using a mouse, such as clicking, double-clicking, right-clicking, dragging and dropping, etc.
* These actions simulate a user’s interactions with a website through the mouse.
* The Actions class in Selenium WebDriver provides the following mouse action:

|  |  |  |
| --- | --- | --- |
| methods | use | syntax |
| click() | performs a single mouse click on the specified element. | Webelementname.click(); |
| contextClick() | performs a right-click on the specified element. | Actions action=**new** Actions(driver);  action.contextClick(right\_click\_field).  build().perform(); |
| doubleClick() | performs a double-click on the specified element. | Actions action=**new** Actions(driver);  action.doubleClick(double\_click\_field).  build().perform(); |
| moveToElement() | moves the mouse cursor to the middle of the specified element. | Actions action=**new** Actions(driver);    action.moveToElement(mainitem2\_field).  build().perform(); |
| dragAndDrop(source, target) | performs a drag and drop operation between two elements. | Actions action=**new** Actions(driver);  action.dragAndDrop(dragfield, dropfield).build().perform(); |
| dragAndDropBy(source, x-offset, y-offset) | When there is no drop position. | Actions action=**new** Actions(driver);  action.dragAndDropBy(dragmefield, 150, 100).build().perform(); |
| // clickAndHold() | holds down the left mouse button on the specified element. |  |
| //release() | releases the left mouse button on the specified element. |  |

**What is the use of build () and perform () in actions?**

* Build and Perform are methods in the Selenium Actions class.
* They are used together to create and execute complex user interactions on a web page.
* Build is used to create an instance of the Actions class and associate it with a specific WebDriver instance.
* The methods click, doubleClick, contextClick, moveToElement, and dragAndDrop can be chained together to create a series of actions.
* Once you have specified all the actions you want to perform, you can call the build method to build the Actions object.
* Perform is used to execute the actions that were built using the build method.
* Once you call to perform, Selenium will execute the actions in the order that they were specified.

**build()** − This method is used to create a combination of actions having all the actions to be carried on.

**perform()** − This method is used to perform actions without invoking the build() first.

**Keyboard actions**

* The Robot class in Selenium is a Java-based utility class that provides a way to simulate keyboard events on the screen.
* Allows testers to automate tasks that cannot be done using Selenium's built-in methods, such as pressing a key on the keyboard.
* The Robot class in Selenium works by creating a virtual robot on the screen that can perform keyboard.
* The Robot class in Selenium provides several methods for simulating keyboard actions, including key presses and key releases. Here are some of the most commonly used methods:

|  |  |  |
| --- | --- | --- |
| methods | use | syntax |
| keyPress(int keycode) | Simulates a key press of the specified keycode. |  |
| keyRelease(int keycode) | Simulates a key release of the specified keycode. |  |
| keyPress(KeyEvent.VK\_SHIFT) | Simulates a key press of the Shift key. |  |
| keyRelease(KeyEvent.VK\_SHIFT) | Simulates a key release of the Shift key. |  |
| keyPress(KeyEvent.VK\_CONTROL) | Simulates a key press of the Ctrl key. |  |
| keyRelease(KeyEvent.VK\_CONTROL) | Simulates a key release of the Ctrl key. |  |

**Keyboard Functionalities**

Simulating key presses

* + The keyPress(int keycode) method is used to simulate a key press on the keyboard.
  + The keycode parameter specifies which key to press, and can be one of the constants defined in the KeyEvent class.
  + For example, KeyEvent.VK\_ENTER can be used to simulate the Enter key being pressed.

Simulating key releases

* + The keyRelease(int keycode) method is used to simulate a key release on the keyboard.
  + The keycode parameter specifies which key to release, and can be one of the constants defined in the KeyEvent class.
  + For example, KeyEvent.VK\_ENTER can be used to simulate the Enter key being released.

Simulating key combinations

* + The keyPress(int keycode) and keyRelease(int keycode) methods can be used together to simulate key combinations, such as Ctrl+C for copying text.

For example:

Robot robot=**new** Robot();

robot.keyPress(KeyEvent.***VK\_CONTROL***);

robot.keyPress(KeyEvent.***VK\_T***);

robot.keyRelease(KeyEvent.***VK\_CONTROL***);

robot.keyRelease(KeyEvent.***VK\_T***);