**Selenium**

* [Selenium](https://www.browserstack.com/selenium) refers to a suite of tools that are widely used in the testing community when it comes to [cross-browser testing](https://www.browserstack.com/live).
* Selenium cannot automate desktop applications; it can only be used in browsers.
* It is considered to be one of the most preferred tool suites for [automation testing](https://www.browserstack.com/guide/automation-testing-tutorial) of web applications as it provides support for popular web browsers which makes it very powerful.
* It supports a number of browsers (Google Chrome 12+, Internet Explorer 7,8,9,10, Safari 5.1+, Opera 11.5, Firefox 3+) and operating systems (Windows, Mac, Linux/Unix).
* Selenium also provides compatibility with different programming languages – C#, Java, JavaScript, Ruby, Python, PHP. Testers can choose which language to design test cases in, thus making Selenium highly favorable for its flexibility.

**Note:** It is not mandatory to write Selenium code in the same language as the application.

For example, if the application under test is written in PHP, then testers don’t have to write Selenium code in PHP.

Thus, if a website is written in C#, the Selenium code can be written in PHP too.

### ****Selenium Components****

The Selenium test suite comprises 4 main components:-

1. **Selenium IDE**
2. **Selenium RC**
3. **Selenium Webdriver**
4. **Selenium Grid**
5. **Selenium IDE:**

* Selenium IDE (Integrated Development Environment) is primarily a record/run tool.
* It is an Add-on or an extension available for both Firefox and Chrome that generates tests quickly through its functionality of record and playback.
* You don’t need to learn any test scripting language for authoring any functional tests.

**2.** [**Selenium RC**](https://www.browserstack.com/guide/selenium-rc-tutorial)**:**

* In the case of working with [Selenium RC](https://www.browserstack.com/guide/selenium-rc-tutorial) (Remote Control), one must have good knowledge of at least one programming language.
* This tool allows you to develop responsive design tests in any scripting language of your choice.
* Server and client libraries are the two main components of Selenium RC. Its architecture is complex and it has its limitations.

1. **Selenium WebDriver**

* It is an enhanced version of Selenium RC.
* It was introduced in the market to overcome the limitation faced in Selenium RC.
* Though it is an advanced version of RC, its architecture is completely different from that of RC.
* Just like Selenium RC, Selenium WebDriver too supports multiple programming platforms to provide wider flexibility and requires knowing any one programming language.

1. **Selenium Grid:**

* Selenium Grid is a tool that is used for concurrent execution of test cases on different browsers, machines, and operating systems simultaneously.
* This tool makes Cross-browser compatibility testing very easy.
* There are two versions of the Selenium Grid – the older version is known as Grid 1 and the recent version is known as Grid 2.

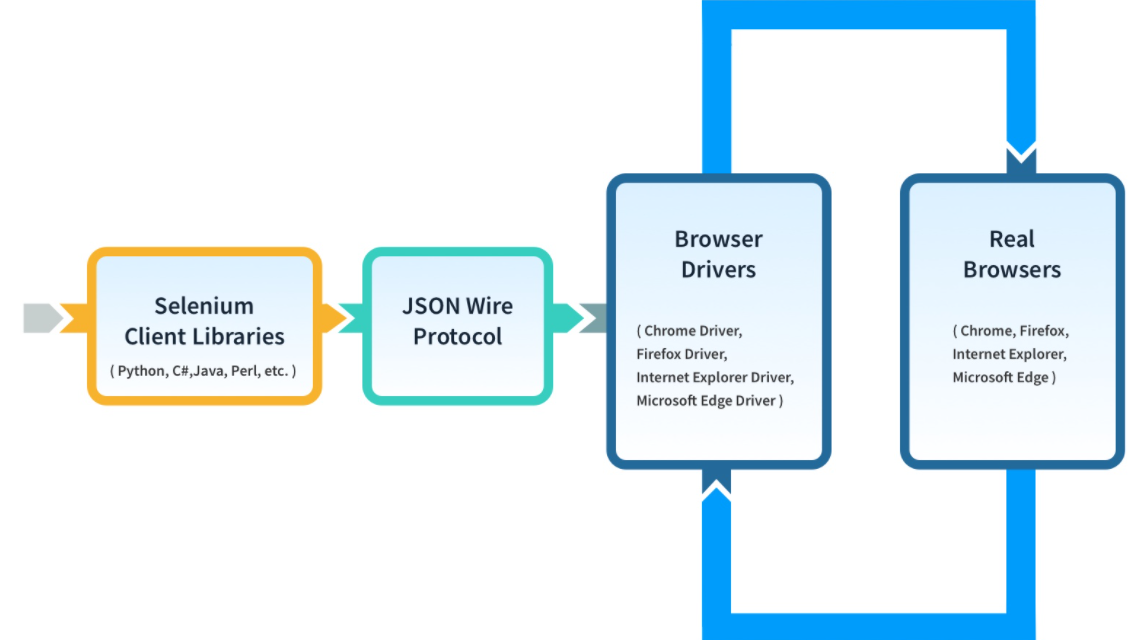
### ****What is Selenium WebDriver?****

* Selenium WebDriver is a web framework that permits you to execute cross-browser tests.
* This tool is used for automating web-based application testing to verify that it performs expectedly.
* Selenium WebDriver allows you to choose a programming language to create test scripts.
* As discussed earlier, it is an advancement over Selenium RC to overcome a few limitations.
* Selenium WebDriver is not capable of handling window components, but this drawback can be overcome by using tools like Sikuli, Auto IT, etc.

**Selenium WebDriver Framework Architecture**

**WebDriver Architecture is made up of four major components:**

1. Selenium Client library
2. JSON wire protocol over HTTP
3. Browser Drivers.
4. 4. Browsers



1. **Selenium Client Libraries/Language Bindings**

Selenium provides support to multiple libraries such as Ruby, Python, Java, etc as language bindings have been developed by Selenium developers to provide compatibility for multiple languages. For instance, if you want to use the browser driver in Python, use the Python Bindings. You can download all the supported language bindings of your choice from the official site of Selenium.

1. **JSON Wire Protocol**

JSON is an acronym for JavaScript Object Notation. It is an open standard that provides a transport mechanism for transferring data between client and server on the web. It provides support for various data structures like arrays and objects which makes it easier to read and write data from JSON.

JSON serves as a REST (Representational State Transfer) API that exchanges information between HTTP servers. Learn more about REST API for accessing Selenium

**3.Browser Drivers**

Selenium provides drivers specific to each browser and without revealing the internal logic of browser functionality, the browser driver interacts with the respective browser by establishing a secure connection. These browser drivers are also specific to the language which is used for test case automation like [C#](https://www.browserstack.com/automate/c-sharp), [Python](https://www.browserstack.com/automate/python), [Java](https://www.browserstack.com/automate/java), etc.

You can download the browser driver of your choice as per your language requirements.

When a test script is executed with the help of WebDriver, the following tasks are performed in the background:

* An HTTP request is generated and it is delivered to the browser driver for every Selenium Command
* The HTTP request is received by the driver through an HTTP server
* All the steps/instructions to be executed on the browser is decided by an HTTP server
* The HTTP server then receives the execution status and in turn sends it back to the automation scripts

**4. Browsers**

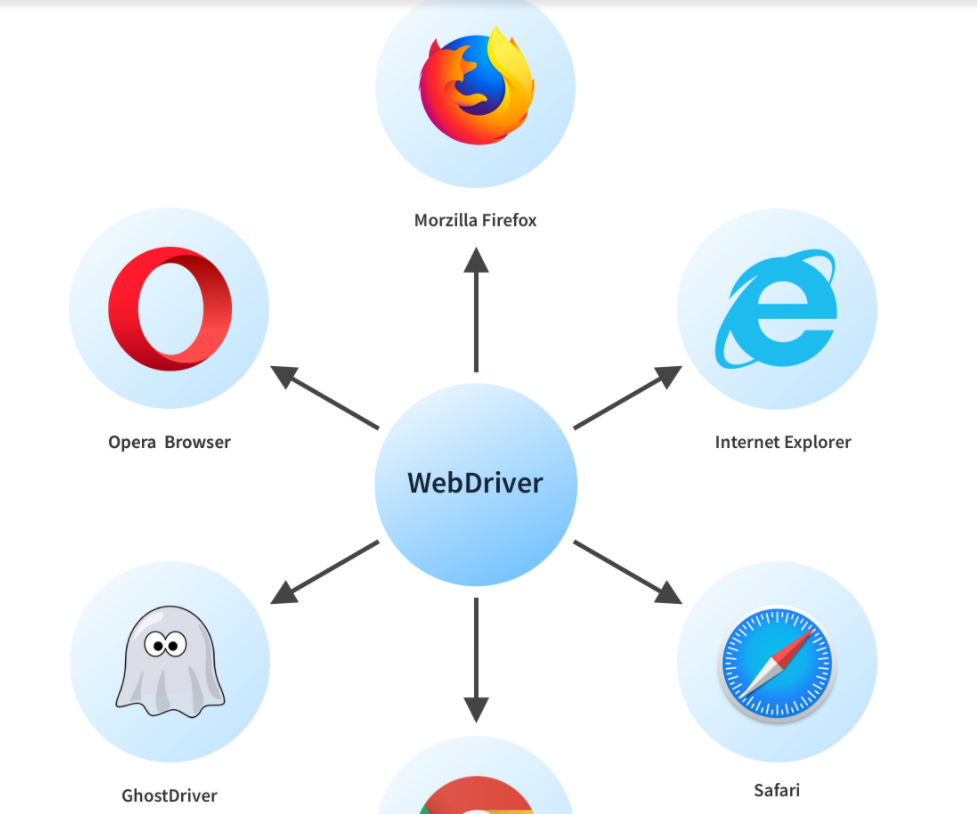
As discussed earlier, Selenium provides support for multiple browsers like Chrome, Firefox, Safari, Internet Explorer etc.

### ****Basic Steps in a Selenium WebDriver Script****

* Create a WebDriver instance.
* Navigate to a webpage.
* Locate a web element on the webpage via [locators in selenium](https://www.browserstack.com/guide/locators-in-selenium).
* Perform one or more user actions on the element.
* Preload the expected output/browser response to the action.
* Run test.
* Record results and compare results from them to the expected output.

### ****Benefits of Selenium WebDriver****

* It is one of the most popular Open-Source tools and is easy to get started with for testing web-based applications. It also allows you to perform [cross browser compatibility testing](https://www.browserstack.com/guide/cross-browser-compatibility-testing-beyond-chrome).
* Supports multiple operating systems like Windows, Mac, Linux, Unix, etc.
* It provides compatibility with a range of languages including Python, Java, Perl, Ruby, etc.
* Provides support for modern browsers like Chrome, Firefox, Opera, Safari, and Internet Explorer.
* Selenium WebDriver completes the execution of test scripts faster when compared to other tools
* More Concise API (Application Programming interface) than Selenium RC’s
* It also provides compatibility with iPhoneDriver, HtmlUnitDriver, and AndroidDriver



**Limitations of WebDriver**

* Support for new browsers is not readily available when compared to Selenium RC
* For the automatic generation of test results, it doesn’t have a built-in command

**How Selenium WebDriver Works?**

On a high-level, Selenium WebDriver works in three steps:

* Test commands are converted into an HTTP request by the JSON wire protocol.
* Before executing any test cases, every browser has its own driver which initializes the server.
* The browser then starts receiving the request through its driver.

Let’s take an example with the code snippet below:

driver = webdriver.Chrome(executable\_path=”…”);

driver. get (https://www.browserstack.com)

As soon as you complete writing your code, execute the program. The above code will result in the launching of the Chrome browser which will navigate to the BrowserStack website.

Now let us understand what goes behind the scene when you click on Run until the launching of the Chrome Browser.

Once the program is executed, every line of code/script will get transformed into a URL. The JSON Wire protocol over HTTP makes this possible. Then this URL is passed to the browser drivers (in our example, the ChromeDriver). At this point, our client library (Python in our example) translates the code into JSON format and interacts with the ChromeDriver.

The URL after JSON conversion looks as follows:

https://localhost:8080/{"url":https://www.browserstack.com"}

To receive the HTTP requests, every Browser Driver uses an HTTP server. Once the browser driver receives the URL, it processes the request by passing it to the real browser over HTTP. And then all your commands in the Selenium scripts will be executed.

The selenium code does the following:

* Create a Selenium WebDriver instance
* Configure browser if required
* Navigate to the required web page and locate the relevant web element
* Perform action on the web element
* Verify and validate the action

Selenium Python Package Set up with diff browsers

#browser exposes an executable file

#Through selenium test we need to invoke the executable file which will then invoke actual browser

# to identify whether our requested URL routed to correct page or not because of cybersecurity attacks we can use below statements

Print(Driver.title)

Print(Driver.current\_url)

Back(), refresh, forward()