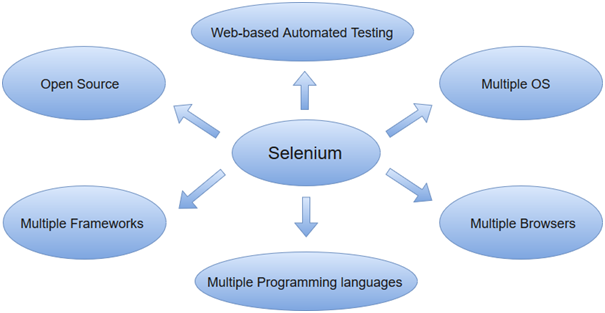
What is Selenium

Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite. It was originally developed by Jason Huggins in 2004 as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms and programming languages.

Well, Selenium is an open-source automation testing tool which is used for automating tests carried out on different web-browsers.

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.

### Selenium WebDriver(Selenium 2.0)

Selenium WebDriver allows you to directly interact with the browsers through your automation test scripts. Java, PHP, C#, Python, Ruby, Perl, and Javascript are some of the programming languages it supports. The browsers Selenium WebDriver supports include Mozilla Firefox, Google Chrome version 12.0.712.0 and above, Internet Explorer, Safari, Opera version 11.5 and above, and HtmlUnit version 2.9 and above. As for operating systems, Selenium WebDriver supports Windows, Linux, Mac OS, and Solaris.

Selenium WebDriver is also known as the Selenium 2.

## What Are The Different Types Of Tools Selenium Is Composed Off?

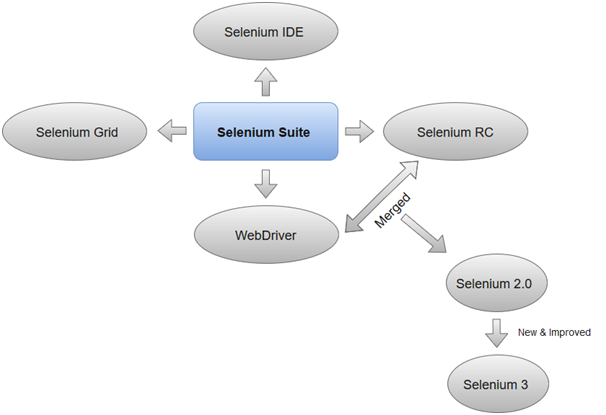
Selenium offers a collection of tools for browser automation. It comprises of 4 Selenium components, each serving a different purpose for web testers.

1. Selenium IDE (Integrated Development Environment)

2. Selenium RC (Remote Control)

3. Selenium WebDriver

4. Selenium Grid



### **1.Selenium Integrated Development Environment (IDE)**

Selenium IDE is implemented as Firefox extension which provides record and playback functionality on test scripts. It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, RSpec, Python, C#, JUnit and TestNG. You can use these exported script in Selenium RC or Webdriver.

>Selenium IDE has limited scope and the generated test scripts are not very robust and portable.

### **2. Selenium Remote Control**

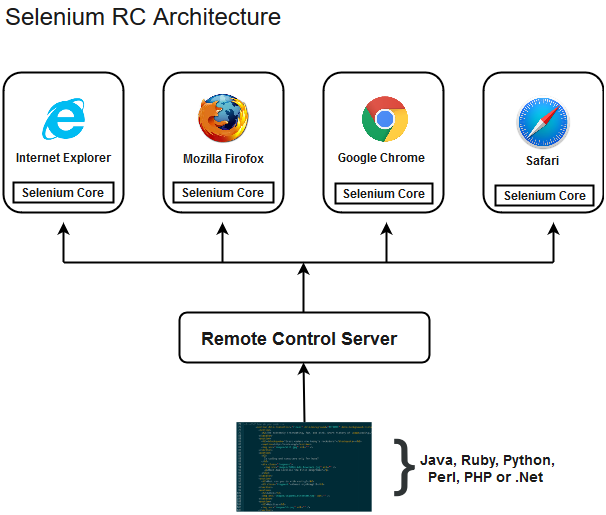
Selenium RC (officially deprecated by selenium) allows testers to write automated web application UI test in any of the supported programming languages. It also involves an HTTP proxy server which enables the browser to believe that the web application being tested comes from the domain provided by proxy server.

**00:00/11:48**

Selenium RC comes with two components.

1. Selenium RC Server (acts as a HTTP proxy for web requests).
2. Selenium RC Client (library containing your programming language code).

The figure given below shows the architectural representation of Selenium RC.



Selenium RC had been considered quite effective for testing complex AJAX-based web user interfaces under a Continuous Integration System.

### **3. Selenium WebDriver**

Selenium WebDriver (Selenium 2) is the successor to Selenium RC and is by far the most important component of Selenium Suite. Selenium WebDriver provides a programming interface to create and execute test cases. Test scripts are written in order to identify web elements on web pages and then desired actions are performed on those elements.

Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the web browser.

Since, WebDriver directly calls the methods of different browsers hence we have separate driver for each browser. Some of the most widely used web drivers include:

* Mozilla Firefox Driver (Gecko Driver)
* Google Chrome Driver
* Internet Explorer Driver
* Opera Driver
* Safari Driver
* HTML Unit Driver (a special headless driver)

#### **Note: Selenium version 2 merged the best features of Selenium RC and Selenium WebDriver into Selenium WebDriver. The latest release Selenium 3 has new added features and functionalities**

### **4. Selenium Grid**

Selenium Grid is also an important component of Selenium Suite which allows us to run our tests on different machines against different browsers in parallel. In simple words, we can run our tests simultaneously on different machines running different browsers and operating systems.

Selenium Grid follows the **Hub-Node Architecture** to achieve parallel execution of test scripts. The Hub is considered as master of the network and the other will be the nodes. Hub controls the execution of test scripts on various nodes of the network.

### Selenium 2 Is WebDriver Combined With Selenium RC

On July 8, 2011, Selenium official blog made an announcement for the release of Selenium 2. Selenium 2 was built by integrating Selenium RC and WebDriver. WebDriver back then was a competitive & probably a more preferred web-application testing framework than Selenium RC. Wondering how? Well, WebDriver allowed testers to interact more closely with various browsers by controlling them directly through in-built browser extensions whereas Selenium RC ran a JavaScript Application inside a browser. Fortunately for testers, the developers from both the projects shook hands as they felt integrating Selenium RC and WebDriver could make the life of testers a lot easier. And so it was named as the Selenium WebDriver.

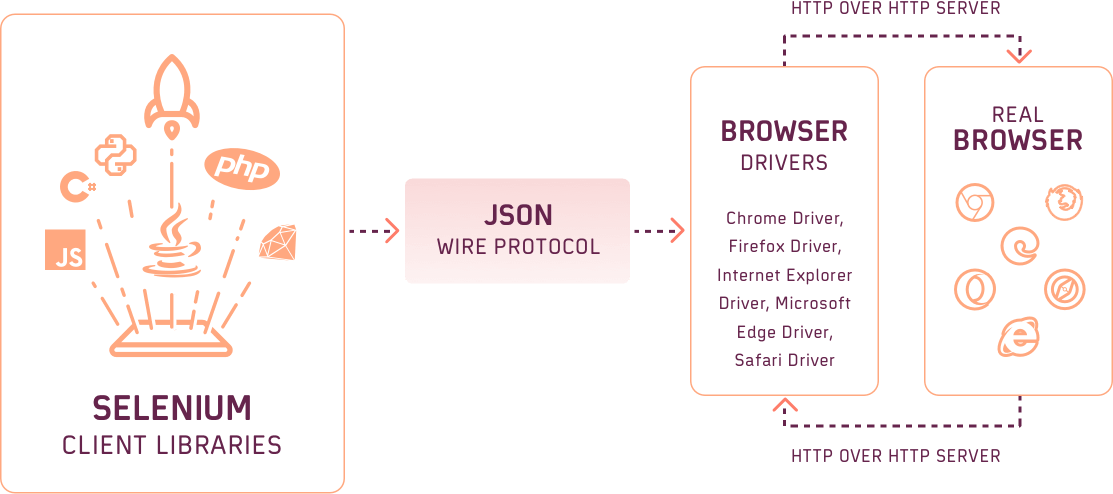
Since then Selenium 2 has been upgraded to a better and improvised version Selenium 3 & recently it got another upgrade Selenium 4.

### What Makes Selenium WebDriver Better Than Selenium RC?

Selenium WebDriver can help you perform test automation of your web-application without any server dependency like Selenium RC. A Selenium WebDriver directly opens up a browser based on the specification you declare through Selenium scripts or the Client APIs. Another key distinguishing feature which makes Selenium WebDriver better than Selenium RC is the fact that it leverages the functionality of your native operating system to control the browser instead of relying over the JavaScript commands to initiate and operate the browser.

Selenium WebDriver also resulted in fewer API interactions when compared with Selenium RC. Using Selenium WebDriver, developers can even create their own Domain Specific Language. [Watir framework](https://www.lambdatest.com/blog/cross-browser-automation-testing-using-watir/" \t "_blank) is one such example for running Selenium testing with Ruby.

### What is Selenium WebDriver Architecture?



Selenium WebDriver comprises of 4 main components:

1. Selenium Client Libraries

2. JSON Wire Protocol Over HTTP Client

3. Browser Drivers

4. Real Browsers

Selenium Client Libraries

Selenium allows you to execute browser automation with these scripts written in any programming language with the help of Selenium Client Libraries. These Client libraries act as an interpreter between your test script & Selenium. They translate a test script written in any programming language to Selenese through language bindings. That way, Selenium is able to follow your given instruction irrespective of what language you chose to write your Selenium test scripts.

The major and most widely used Selenium bindings provided by the Selenium Project are their to help you with:

1. Java Selenium Testing

2. Python Selenium Testing

3. JavaScript Selenium Testing

4. C# Selenium Testing

5. Ruby Selenium Testing

6. PHP Selenium Testing

JSON Wire Protocol OverHTTP Client

Selenium allows you to automate browser testing. With Selenium WebDriver, you can execute browser automation by interacting directly with all of the major browsers such as Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, Opera, and more. Every browser comprises a Browser Driver which has its own HTTP server.

JSON Wire Protocol is responsible for communicating with the browser drivers through their HTTP server. It fetches the information from Selenium Client Libraries and then relays it to the respective Browser Driver.

Browser Drivers

Each browser has a driver which is responsible for controlling the actions performed within that browser. After JSON Wire Protocol relays information to a Browser Driver, the Browser Driver controls the Browser to execute your Selenium test scripts automatically and sends the response in HTTP protocol through a HTTP server. Following are the Browser Drivers for the major web browsers:

1. ChromeDriver

2. FirefoxDriver

3. OperaDriver

4. SafariDriver

5. Ruby Selenium Testing

6. EdgeDriver

Browsers

Selenium WebDriver allows you to interact with various web browsers i.e. Google Chrome, Mozilla Firefox, Safari, Opera, Microsoft Edge, and more. Kindly note, Selenium helps you perform browser testing of your web applications and not native mobile apps. For that purpose, you have another open-source framework called Appium.

In 2012, W3C was approached by Simon Stewart(Founder of WebDriver) & David Burns to add [WebDriver](https://www.w3.org/TR/webdriver/) as an internet standard. After a long period of negotiation,WebDriver was finally declared as a W3C standard protocol in 2019.

### Major Drawback Of Selenium WebDriver

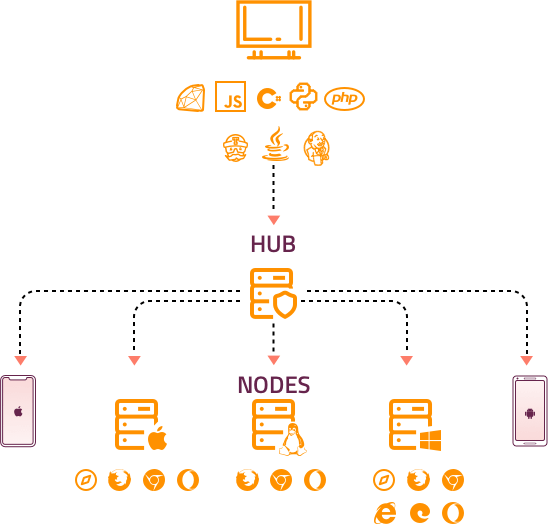
Although Selenium WebDriver allows you to easily automate web application testing over different browsers. It doesn’t allow you to do it in parallel. You can’t run multiple tests simultaneously and this isn’t feasible if you are supposed to automate a large test suite or if you are supposed to run a small test suite over hundreds of browsers & OS combinations. Thanks to Selenium Grid, testers were able to overcome this scalability problem.

### Selenium Grid

Selenium Grid has by far been the most useful component of the Selenium project. [Selenium Grid](https://www.lambdatest.com/blog/why-selenium-grid-is-ideal-for-automated-browser-testing/) allows parallel testing against various browsers & OS combinations through a Client-Server model. Here, the Server is known as the Hub which has multiple Clients to interact with.

With Selenium Grid, you can connect a server to multiple remote machines which can then be used to run a browser automation script over multiple browser + OS configurations, simultaneously.

How Selenium Grid Works?



The Hub and Nodes are the two primary components of the Selenium Grid.

Hub or Server

The WebDriver client grants access requests through the Hub. The Hub, in the Selenium grid, is responsible for sending commands to the Nodes through JSON Wire Protocol over HTTP.

Node or Client

A Node, in Selenium grid, is simply an instance of a Selenium WebDriver over a remote machine. What happens inside a node is similar to what we discussed in the architecture of Selenium WebDriver. Each node is an instance of a Remote Selenium WebDriver responsible for interacting with the browsers based on the commands received from the Hub or server.

How Hub & Nodes Work Together In A Selenium Grid?

When you run a test script over a Selenium Grid, it is first pushed to the Hub/Server where the desired capabilities class is assessed to figure out the test configurations. The desired capabilities class specifies the browser, browser version and operating system over which the test script should be executed. Among the registered nodes, the Hub finds a suitable Node which matches with the desired capabilities. The Hub then sends commands to the Node for executing the script in the remote machine which meets the right browser + OS configuration.

Selenium Grid profoundly minimizes the test runtime as it allows you to run multiple test scripts over different remote devices, simulatenously.

### Major Challenges Of Selenium Grid

Although Selenium Grid is an excellent option to automate browser testing. However, there are a few challenges with Selenium Grid in-house infrastructure.

Collection of Latest & Legacy Browsers

You can only automate web testing with Selenium Grid over the browsers & operating system which are available in your remote machines. Meaning, you will have to ensure that you are always updating or adding more nodes to the Hub for a thorough round of automated browser testing. This could be challenging considering the pace at which browser versions are being released.

Collection of Devices & OS

Similar to browsers you would have to keep a track over the latest devices being launched from different vendors. Not to forget, you would also need to spend a considerable amount of money to add new devices & operating systems in your nodes.

Hardware Expenditure

More nodes you add to your Selenium Grid more expenditure would be required for processing parallel testing. You would need at least 4GB memory RAM to process a Selenium Grid of 2 parallel runs. If you wish to run 3 tests in parallel then a 4GB memory RAM would rarely cut it. Imagine the hardware required to run 10 parallel test sessions.

Maintenance Takes A Toll

Even after you add the latest devices to your Selenium Grid, you would still require dedicated resources to manage the in-house Selenium Grid Infrastructure..

On-premise Restrictions

The Hub & Node should be connected under the same network. Meaning, if you are having a Selenium Grid setup in your corporate office then you can’t access it from anywhere else.

Note:You may be wondering how to overcome these challenges. Well, don’t worry. We got your back! With LambdaTest, you can ditch the hassle of maintaining in-house Selenium Grid as we offer you a cloud-based, online Selenium Grid of 3000+ real browsers for both desktop and mobile. Our Selenium Grid is compatible with all the test automation frameworks & programming languages that are supported by the Selenium Project. What’s more? Well, we also help you collaborate better with your team & build a bug-free deployment pipeline with our third-party integrations to CI/CD tools like Jenkins, CircleCI, etc. & project management tools like Jira, asana, and more.

### Selenium IDE

Selenium IDE also known as the Selenium Recorder was made available for use in 2006. It offers an Integrated Development Environment to record and playback test scripts. [Selenium IDE](https://www.lambdatest.com/blog/selenium-ide-what-is-it-why-is-it-must-for-every-qa/) is available as a Mozilla Firefox add-on and a Google Chrome plugin.

The primary purpose of bringing in Selenium IDE was to ease the process of regression testing for manual testers. With Selenium IDE, all you have to do is hit the record button and run the test manually in your browser and once the test case is executed, you can stop the recording. The next time you need to execute that test case, you would just have to playback the recording and the browser automation would take place.

While Selenium IDE records your test cases, it automatically generates a Selenese script for every interaction that you make with the browser. Selenese is a scripting language exclusive to Selenium. It offers commands for various browser operations such as click a button, select values from drop-down etc.

Note: Selenium IDE 2 was not supported by Mozilla Firefox 51 and greater versions. Fortunately, a new version Selenium IDE 3 was introduced to allow testers to run Selenium tests through IDE on all Mozilla Firefox browsers. However, there were 2 major drawbacks with Selenium IDE which have now been resolved in the latest version.

### The Fragility Of Recorded Test Scripts

If you were to ever commit even the slightest of changes in the UI of your website then the recordings became ineffective due to the change in Selenium locators. For example, previously, if you have recorded an interaction within your browser to click on the Login button and you later decided to change the text of the button to Sign-in then your Selenium IDE script would become ineffective as the locator value of the button would change from what it was when the test was recorded. As a result, you might have to record the entire test script again.

However, the new Selenium IDE 3.x version has overcome this problem by capturing numerous locators while recording the test cases. Now, if any locator is changed due to the UI transformation then the Selenium IDE will look for some other locator. So in this regard, your test would now fail only if all of the locators that were present while recording the test are changed.

### Selenium IDE Lacked Cross Browser Support

When Selenium IDE was introduced in 2006, it was only made as an add-on for Mozilla Firefox back then and Google Chrome was launched in 2008. So it took years for Selenium IDE to be added to Google Chrome Web Store. But even then the relevance of cross browser testing was being missed as you could only test on 2 major browsers. Not anymore though!

Now, the Selenium IDE 3.x version allows you to run your Selenium automation scripts over Selenium WebDriver with the help of a .side file. This .side file is created by the command line test runner of Selenium IDE called SIDE Runner.

SIDE Runner acts as the intermediary between Selenium IDE and Selenium WebDriver, allowing you to perform browser automation with your Selenium IDE test scripts over a variety of browsers.

### A Better Selenium IDE Is Now Available

Due to these reasons, Selenium IDE failed to gain traction in the early stages of development and the project was not actively managed. However, the scenario is different now. The Selenium IDE project is now being managed progressively and has also referred to two other alternatives of itself. One is Kantu which is an open source and the other is Katalon Studio which is a closed-source. Out of all the Selenium tools, Selenium IDE is the easiest to help manual testers take a leap into automation testing with Selenium.

### Selenium RC(Remote Control)

Introduced in 2005, Selenium RC became the first instance of Selenium testing tool, which is why it is often referred to as Selenium 1. It was configured in Java to work on the basis of a client-server model where Selenium RC is the server & browsers were clients. As a server, Selenium RC receives commands from the browser client through HTTP protocol.

Although written in Java, Selenium RC allowed execution of browser automation tests written across any programming language. To further simplify the process, later Selenium Client API were introduced. Selenium Client API lifts the dependency of you having to write Selenese as they help Selenium to interpret the code written in different programming languages. You have Selenium Client drivers for Java, C#, Python, Ruby, PHP. For JavaScript, the same Java driver can be used through Rhino engine.

For parallel testing with Selenium RC, every new test case will need a different instance of server unless you are running parallel testing with Selenium RC through Java & PHP where the server instance runs in a continuous state.

It is important to note that Selenium RC has been deprecated after the launch of Selenium 2 also known as the Selenium WebDriver.

## History of Selenium

It all started in 2004 with Jason Huggins, an engineer at Thoughtworks. While testing, he realized that manual testing was inefficient. So, he created a program using Javascript to automate repetitive tasks. Selenium Core is the name of this open-source program used for automating different web applications.

Selenium: An Open Source Tool

There was a time when only Thoughtworks employees used Selenium internally. But by the end of 2004, Selenium became an open-source tool and was made available for the testing community. I can’t quote the exact date and time for when it became open source. However, it’s safe to say that it was no later than November 2004. During that time Selenium had bugs.

Moreover, many companies were adhering to waterfall development practices which had inefficiencies. Selenium sewed down the reach of bug fixes to users. The fact that Selenium became an open-source was a huge thing. Not only was the tool available for free use, but people could also contribute to the Selenium Project for expanding its features.

Evolution of Selenium

The popularity of Selenium started to grow and within a year. By October 2005, Selenium evolved significantly. Developers started to make big plans for the open-source tool. The original mission was to use Selenium as a tool for web testing. But as it evolved, developers started to envision adding features that would take Selenium beyond what the founders originally intended it for.

In 2007, Jason Huggins moved to Google and continued working on the tool. Some new features included cross-platform testing and framed application testing. But Huggins shifting to Google didn’t rule out ThoughtWorks out of the picture.

History of Selenium WebDriver

In the same year, 2007, Simon Stewart developed the WebDriver. He was also a ThoughtWorks employee. The purpose of this tool was to exercise control on the browser. Moreover, this testing framework could test across different platforms. Selenium lacked some features and WebDriver just fueled the desire for them. In fact, for some time, it emerged as Selenium’s competitor. But the rivalry ended in July 2011 when the two tools joined together to form Selenium 2.0. It paired the WebDriver APIs which we are familiar with today. It had the original feature set of Selenium along with new features.

History of Selenium Grid

Patrick Lightbody developed the Selenium grid for minimizing test execution times. After the release of Selenium 2.0, the server of Selenium had a built-in grid functionality. It facilitated the sending of Selenium commands simultaneously to multiple machines. During crucial stages, a Selenium Grid was also capable of taking browser screenshots.

History of Selenium Remote Control

If a domain is different from where its launch occurred, the JavaScript code can't access elements from it. This is the Same Origin Policy that forced testers using Selenium Core to install the web server and the application under test on their local PCs.

Thanks to Paul Hammant(Engineer at Thoughtworks) who came up with an idea and implemented a server which could act as an HTTP proxy. The trick was to make the browser believe that application under test and Selenium Core originate from the same domain.

History of Selenium IDE

Shinya Kasatani created the Selenium IDE to speed up the creation of test cases. An intuitive interface, quick setup, and rapid results were only some of the major benefits IDE provided. Even with limited knowledge of scripting languages, you could create tests with its point-and-click interface.

That Was All. Any Questions?

As we already know, modern SDLC(Software Development Life Cycle) methodologies demand shorter release times. However, to make sure that the releases are bug-free and function smoothly, deterministic and repeatable testing is a must. Selenium has been aiding business deliver robust web application, faster and has made its way among some of the most vital tools used in the IT industry. All in all, if you wish to make the testing procedure a breeze, Selenium is the way to go!

We hope by far you have a pretty good understanding of what is Selenium testing. If you have any questions, feel free to share them with us. Happy testing!

## Why Is Selenium Important?

Product teams are hesitant in releasing a product hastily. Usually, when the process is manual, exhaustive testing is a part of the SDLC. Depending on the size of the testing team, the process can take a few days to many weeks. No firm can afford the inefficiency that comes with dedicating excess time to development. If you want to supercharge the process, [Selenium automation testing](https://www.lambdatest.com/selenium-automation) is the way to go. Here are the key benefits of using Selenium.

### Faster Cross Browser Testing With Selenium

Your users can visit your web application from any browser of their choice. Which is why it is important to ensure a [cross browser compatible website](https://www.lambdatest.com/blog/how-to-make-a-cross-browser-compatible-website/). That way, you can ensure that your users are always experiencing a seamless UI, regardless of which browser they are using. The process to test your web application over different browsers is called cross browser testing and it can be nightmarish for manual testers as they may have to test over hundreds of browsers, every time a change is pushed into their web application. However, with Selenium you can [automate browser testing](https://www.lambdatest.com/selenium-automation).

Selenium WebDriver can help you perform browser automation across Mozilla Firefox, Opera, Safari, and Google Chrome, Microsoft Edge, and even the notorious Internet Explorer. These are the major players in the browser wars. And a single open-source test automation framework ensuring compatibility with all of them, what more could one ask for?

Browser testing with Selenium has made life easier for web testers around the globe.

### Selenium Supports Different Programming Languages

Selenium supports multiple programming languages providing the flexibility for any tester to get a grip over the test automation framework with ease.

What are the different programming languages supported by the [Selenium WebDriver](https://www.lambdatest.com/blog/selenium-webdriver-tutorial-with-examples/).

Although you can perform browser testing with Selenium through almost any programming language, perks of being popular. The official Selenium project offers bindings for Java, JavaScript, C#, Python, PHP, Ruby.

### Selenium Supports Multiple Operating Systems Too

Selenium supports multiple operating systems such as Windows, Linux, macOS, Solaris. That way, you can aggressively test your web application by executing it over combinations of browsers + OS.

### Selenium Can Be Integrated With CI/CD Pipelines

[Continuous Integration & Continuous](https://www.lambdatest.com/blog/what-is-continuous-integration-and-continuous-delivery/) Delivery demand for a rapid and frequent delivery of a build's new releases. Now imagine yourself being in-charge of ensuring cross browser compatibility of your web-app manually after every build is passed through your CI/CD pipelines. We won't have to tell you how much bandwidth it might cost. Not to forget the probability of human error while doing so. With Selenium, there is a smarter way!

Using Selenium tool for browser automation, the team doesn't have to wait around for iterations running their course. CI engines keep each team member informed of configuration, infrastructure, changes in code, and errors. This way, the team can catch deployment failures, if any, at an early stage.

Selenium automation can help you with performance, functional, and compatibility testing that is recurring in nature. It also facilitates quick debugging by giving almost instant feedback to the developers.

Also Read: [Build An Automated Testing Pipeline With GitLab CI/CD and Selenium Grid](https://www.lambdatest.com/blog/automated-testing-pipeline-with-gitlab-ci-cd-and-selenium/)

### Selenium WebDriver Boosts Productivity Of An Agile Tester

Agile is a methodology that focuses on the continuous improvement of a product by dividing the entire development project into small modules or sprints. After every spring a new feature is added to the product through a release cycle. Testing is performed after each iteration to ensure stakeholders that the web-app or website is fully functional. This gives room for continuous feedback and eliminating bugs in the initial phases itself. However, it can be tough for an agile tester to perform testing over new requirements, while keeping check over the [regression defects](https://www.lambdatest.com/blog/why-understanding-regression-defects-is-important-for-your-next-release/).

Selenium automates browser-based web applications allowing an agile tester to automate repeated test scripts so they can come up with more critical test scenarios. Selenium framework speeds up the test execution process and improves testing performance as a whole.

### Selenium is the most popular test framework suites for automating web browser testing

There are tonnes of open-source software available over the internet. However, before incorporating any open-source framework into our projects we need to think about its credibility. How actively is it being managed and by whom?

Selenium is one of the most popular open-source test automation framework. The Selenium Project is being actively managed and has been improvising the Selenium software since 2004. Selenium teting tool also has a huge user community, so in case you get stuck with something. You can easily troubleshoot it with the help of Selenium community, Selenium tutorials, blogs, and discussion forums.

## Why Is Selenium Used?

Selenium is basically used to automate the testing across various web browsers. It supports various browsers like Chrome, Mozilla, Firefox, Safari, and IE, and you can very easily automate browser testing across these browsers using Selenium WebDriver.

You can see live automated tests being performed on your computer screen. But the question that we’ll be answering in this article still stands as there are many tools available for automation testing.

## Who Uses Selenium Automation?

Selenium automation testing is widely adopted throughout the web development domain. Almost every business that is running online through a website or web-application is having Selenium tool incorporated in their testing workflows.

Although Selenium is a test automation framework it isn't limited to testers alone. Being compatible with multiple programming languages, Selenium has open ways for developers to automate their unit testing efforts too. Even DevOps team integrate Selenium in their CI/CD tools to ensure a rigorous round of browser automation testing every time a code change is committed through pipeline from one stage environment to another.

Being an open-source framework, it wouldn't be wrong to comment that Selenium serves as a lifeline to many freelancer web developers & web testers.

Simply speaking, After each integration, UI engineers can use Selenium for visual regression tests. Even freelance developers can run tests for debugging using Selenium. QA engineers can ensure deterministic scripts and tests with enhanced accuracy and coverage with Selenium.

## Which Testing Can Be Automated Using Selenium WebDriver?

Selenium can help you with browser automation for various purposes.

### Unit Testing

Unit testing is done by the developers when a feature enhancement or a bug-fix is applied to a module of a web-application. Unit testing can be strenuous if a developer has to work over multiple code changes for different modules.

We've multiple unit testing frameworks specific to each programming language such as JUnit for Java, NUnit for Python, and more, to help us automate unit testing. Selenium is compatible with these unit testing frameworks and can help developers quickly validate their unit testing over various browsers + OS combinations.

### System Testing or Black Box Testing

In system testing, testers check the compliance of the system with specific requirements. It involves testing functionality of a module from an end-to-end perspective. Testers neither have any tests executed previously nor do they have any context of the code. A QA professional could draft a script to verify the functionality of the entire system.

Selenium automation can help you with black box testing scripts, saving you the bandwidth to come up with more unique test scenarios.

### Integration Testing

In Integration testing, the QA team performs tests to make sure that components or units working fine independently work well even when put together. Testers verify that the modules give the same output even after they integrate. This takes place when more than one module is combined together to exhibit a functionality.

Selenium automation can help you with black box testing scripts, saving you the bandwidth to come up with more unique test scenarios.

### End-to-End Testing

As a QA engineer, you are responsible for end-to-end testing of your web application from the end-user's perspective. You would have to think of a series of test cases to ensure the smooth functioning of different touchpoints on the web application.

End-to-End testing could take a considerable amount of your time as your web-application or website may have multiple features and web pages to test over a variety of devices, operating systems, and browsers. You may even have to keep tabs over performance of your web pages. Selenium automation can help you do it with much ease. With Selenium, you can run parallel testing for browser automation and prepare reports for evaluating performance parameters, statuses of test cases and more.

### Regression Testing

Regression testing is performed to evaluate the end-to-end functionality of a web-application or website after any code change is pushed into one staging environment from another. The idea is to ensure that the latest addition of code isn't hampering the existing functionality of your web app. This is the main reason why regression testing is an integral part of every release cycle. It ensures that the existing system can flawlessly incorporate new features. However, it is also the most tedious process. As even for a minute change, you would've to test the entire web application again.

With Selenium WebDriver, you can automate regression testing rounds and save your team a considerable amount of time and bandwidth in a narrow release window.

### Automated Browser Testing

As we already discussed, [cross browser testing](https://www.lambdatest.com/) is the primary reason behind the popularity of Selenium. Selenium supports all major browsers and allows you to perform browser automation to test your website over a variety of them, simultaneously through a [Selenium Grid](https://www.lambdatest.com/selenium-automation).

### Performance and Load Testing

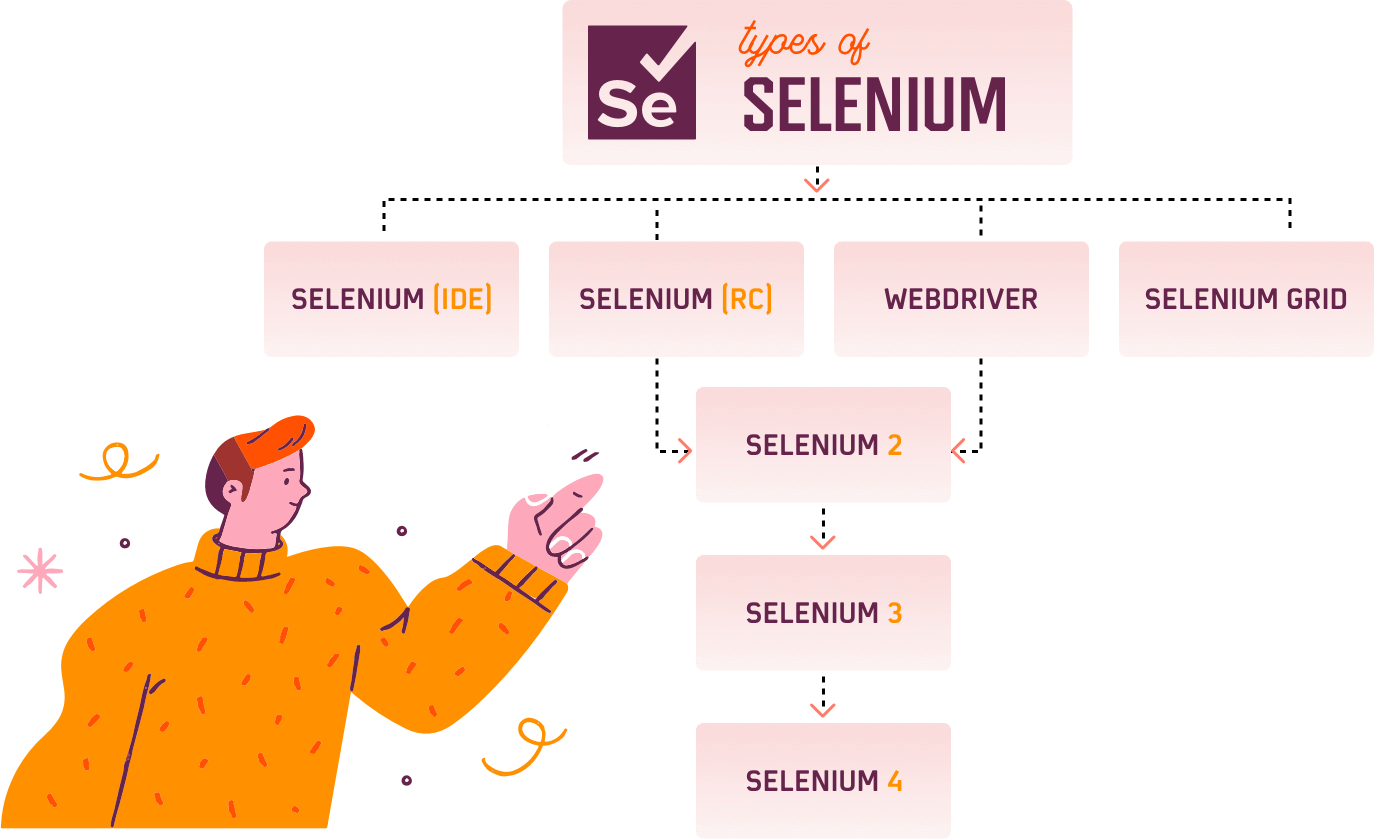
As far as the end-user is concerned, it all comes down to the performance of the web-app. Stakeholders set performance benchmarks and QA professionals run tests to determine whether the application performs as per expectations. For instance, testers check the loading time of the homepage of a website on all browsers.

Instead of manually running your web application over hundred of browsers + OS combinations, you can [measure page load time with Selenium](https://www.lambdatest.com/blog/how-to-measure-page-load-times-with-selenium/). Similarly, you can evaluate other performance metrics for your web-application through Selenium automation testing.

### Monkey Testing

In Monkey testing, we try to break the system by giving a random input to the system. The input could be anything from an end-user’s perspective, it could be a click, scroll, or a dummy text string.

You can generate test scripts to [automate monkey testing with Selenium](https://www.lambdatest.com/blog/monkey-testing-with-webdriverio/). For example, you can have a program that constantly generates random click over a webpage by looking for web locators.



Selenium RC was the first instance of Selenium and is also called as Selenium 1. Selenium WebDriver is called as Selenium 2.0 and is built by combining Selenium RC with WebDriver, a competitor with Selenium back in the days. Selenium WebDriver has been improvised to Selenium 3 & Selenium 4(Alpha release) as well.

## How Do I Get Started With Automation Testing Using Selenium?

Well, you can refer to our [support documentation](https://www.lambdatest.com/support/docs/getting-started-with-lambdatest-automation/) & [GitHub repository](https://github.com/LambdaTest/) to get a step-by-step procedure for running your first Selenium automation testing script over an online Selenium Grid. So you don’t have to worry about the challenges with Selenium infrastructure as we will be providing you a cloud Selenium Grid of zero downtime. Want to know a fun fact? Your first 100 Selenium automation testing minutes are on us, with just a free sign-up. You can also avail benefits of manual cross browser testing, responsive testing, and more with a lifetime free access to LambdaTest, the world’s fastest-growing cloud Selenium Grid.

Our support documentation can help you get started with Selenium in any language that you're comfortable with. By referring to it, you can develop a better functional understanding of Selenium framework, run some sample tests. After creating a WebDriver instance, navigate to any web page. On that web page, locate an HTML element and perform any action on it. After that, predict the response of the browser to that action. With the help of a test framework, run tests and record the results. In the end, conclude the test. We also have [Selenium Tutorials](https://www.lambdatest.com/blog/category/selenium-tutorial/) on our blog page.

Here is a tutorial series to help you become run Selenium testing from a beginner to an expert level.

## Advantages of Using Selenium for Automated Testing

## 1. Language and Framework Support

When someone chooses a tool the first thing that comes to mind is: "Does my tool supports the language that I know?"

Well, this is not the case with Selenium as it supports all major languages like Java, Python, JavaScript, C#, Ruby, and Perl programming languages for software test automation.

You can write your scripts in any of these programming languages and Selenium converts it into Selenium compatible codes in no time. So, there is no need for knowing Selenium only languages. Also, every Selenium supported language has dedicated frameworks which help in writing test script for Selenium test automation. So, when you go for Selenium as a tool for performing automation testing, you don’t have to worry about language and framework support as Selenium does that for you!

## 2. Open Source Availability

One of the many things that adds to the advantages of Selenium is its open source availability. So, being an open source tool, Selenium is a publicly accessible automation framework and is free, with no upfront costs. So, you can save bucks here and use them for other good causes.

The Selenium community is continuously helping developers and software engineers in automating the web browser features and functionalities. Selenium being open source also helps you customize the code for better code management and enhance the functionality of predefined functions and classes. Selenium has become the most reliable web automation tool because of the ease of generating test scripts to validate functionality.

## 3. Multi-Browser Support

“One Selenium script for all browsers” is what the Selenium community has been working on and improvising every day. As per [StatCounter](http://gs.statcounter.com/browser-market-share" \t "_blank), Chrome, Firefox, Safari, Internet Explorer, Opera, and Edge browsers are the most used browsers worldwide and Selenium script is compatible with all the mentioned browsers. You don’t need to rewrite scripts for every browser, just one script for all browsers.

## 4. Support Across Various Operating Systems

Different people use different operating systems and it is necessary that your automation tool supports all of them. Selenium is yet a highly portable tool that supports and can work across different operating systems like Windows, Linux, Mac OS, UNIX, etc.

You can create Selenium test suites over any platform like Windows and can execute the same test suite on another platform, for example, Mac or Linux. This enables developers and software testers to easily write test automation scripts without laying much emphasis on the platform on which it will run.

## 5. Ease Of Implementation

Selenium automation framework is very easy-to-use tool. Selenium provides a user-friendly interface that helps create and execute test scripts easily and effectively. You can also watch while tests are running. You can analyze detailed reports of Selenium tests and take follow-up actions.

And finally, you will never feel alone. A huge Selenium community is always available to help you in case of need. You can ask your queries and perform brainstorming in the community.

## 6. Reusability and Integrations

As mentioned earlier, Selenium automation test suites are reusable and can be tested across multiple browsers and operating systems. However, the twist is if that Selenium is not an all-inclusive web automation testing tool. Hence, it needs third-party frameworks and add-ons to broaden the scope of testing.

For example, you need to integrate Selenium with [TestNG](https://testng.org/)and [JUnit](https://junit.org/junit5/)for managing test cases and generating reports. For achieving continuous testing, you’ll need to integrate it with some CI/CD tools like [Jenkins](https://jenkins.io/), Maven, and Docker. Also, for performing image-based testing, you need to integrate Selenium with tools like Sikuli, and for performing cross-browser testing with cloud-grid. You can integrate Selenium with almost all management tools.

## 7. Flexibility

Test management is what which is very important in testing lifecycle. It becomes easier and more efficient with Selenium features like regrouping and refactoring of test cases. This helps developers and testers in quick changes to the code, reducing duplication, minimizing complications and improving maintainability. These features make Selenium more flexible and usable as compared to other automation testing tools and hence helps Selenium to keep an edge.

## 8. Parallel Test Execution and Faster Go-to-Market

The main aim of automated testing is to save time and efforts. With the help of Selenium Grid, we can execute multiple tests in parallel, hence reducing the test execution time. With the help of cloud-grids for cross-browser testing you can test across as many as hundreds of browsers in parallel using Selenium hence saving you time in multiples of hundreds.

## 9. Less Hardware Usage

If you compare Selenium with other vendor focused automation tools like QTP, UFT, SilkTest, you will find that Selenium requires less hardware as compared to other testing tools.

## 10. Easy to Learn and Use

Selenium scripts are not something like writing hundred-page complex algorithm. Writing Selenium scripts is not more than writing a few pieces of codes to automate functionalities of your website. Also, documentation on the [Selenium website](http://www.seleniumhq.org/) is very helpful for developer and testers to start with Selenium automation testing. With the radically growing community, Selenium tutorials, testing, and development support is just a Google search away.

Also with Selenium IDE extension on Firefox browser, you can use record and play functionality to generate Selenium scripts for future reference.

## 11. Constant Updates

As Selenium is supported by a community and we all know that an active community doesn’t like to stay stagnant, the Selenium community is also constantly releasing constant updates and upgrades. The best part about having a community is that these upgrades are readily available and easy to understand hence you do not need any specific training. This makes Selenium resourceful as compared to other tools and cost-effective as well.

## Selenium Limitations

* Selenium does not support automation testing for desktop applications.
* Selenium requires high skill sets in order to automate tests more effectively.
* Since Selenium is open source software, you have to rely on community forums to get your technical issues resolved.
* We can't perform automation tests on web services like SOAP or REST using Selenium.
* We should know at least one of the supported programming languages to create tests scripts in Selenium WebDriver.
* It does not have built-in Object Repository like UTF/QTP to maintain objects/elements in centralized location. However, we can overcome this limitation using Page Object Model.
* Selenium does not have any inbuilt reporting capability; you have to rely on plug-ins like **JUnit** and **TestNG** for test reports.
* It is not possible to perform testing on images. We need to integrate Selenium with **Sikuli** for image based testing.
* Creating test environment in Selenium takes more time as compared to vendor tools like UFT, RFT, Silk test, etc.
* No one is responsible for new features usage; they may or may not work properly.
* Selenium does not provide any test tool integration for Test Management.

## Selenium Features

* Selenium is an open source and portable Web testing Framework.
* Selenium IDE provides a playback and record feature for authoring tests without the need to learn a test scripting language.
* It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
* Selenium supports various operating systems, browsers and programming languages. Following is the list:
  + Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript
  + Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
  + Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
* It also supports parallel test execution which reduces time and increases the efficiency of tests.
* Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
* Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.
* Selenium requires fewer resources as compared to other automation test tools.
* WebDriver API has been indulged in selenium whichis one of the most important modifications done to selenium.
* Selenium web driver does not require server installation, test scripts interact directly with the browser.
* Selenium commands are categorized in terms of different classes which make it easier to understand and implement.
* Selenium Remote Control (RC) in conjunction with WebDriver API is known as Selenium 2.0. This version was built to support the vibrant web pages and Ajax.