Contents

[Selenium Components 2](#_Toc425238822)

[Selenium IDE (Integrated Development Environment) 2](#_Toc425238823)

[Limitations of Selenium IDE 2](#_Toc425238824)

[Selenium Builder 3](#_Toc425238825)

[Sauce for Selenium Builder Integration 3](#_Toc425238826)

[Limitations of Selenium Builder 3](#_Toc425238827)

[WebDriver (Selenium 2) 4](#_Toc425238828)

[Foreword 4](#_Toc425238829)

[Introducing WebDriver 4](#_Toc425238830)

[Limitations of WebDriver 5](#_Toc425238831)

[Architecture 6](#_Toc425238832)

[Remote WebDriver (Grid 2) 6](#_Toc425238833)

[Limitations of Grid 7](#_Toc425238834)

[Architecture 8](#_Toc425238835)

[What is the Difference between Selenium ID, Selenium RC, and Web Driver 8](#_Toc425238836)

[What testing Frameworks can be used along with Selenium? 9](#_Toc425238837)

Selenium is a robust set of tools that supports test automation for web-based applications.

Selenium first came to life in 2004 when Jason Huggins was testing an internal application at ThoughtWorks.

Selenium is a JavaScript based framework (JavaScript is implemented into web browsers) which means that it can interact only with web page elements. You can only functional test[[1]](#footnote-1) HTML based websites with Selenium, different things impossible to do.

# Selenium Components

Selenium was **composed of three major tools**. Each one has a specific role in aiding the test automation.

## Selenium IDE (Integrated Development Environment)

**It operates as a Firefox add-on** and provides an easy-to-use interface for developing and running individual test cases or entire test suites.

**Selenium IDE records your browser activity** in various selenium dialects.

### Limitations of Selenium IDE

Selenium IDE does not support:

* any non-web based applications
* any other browser than Firefox
* there is no direct support
* condition statements
* iteration or looping
* logging and reporting of test results
* error handling, particularly unexpected errors
* database testing
* test case grouping
* re-execution of failed tests
* test case dependency
* screenshot capture of test failures
* results Report generations
* tests run against other browsers

The reason is IDE supports only HTML language. Using HTML, we cannot achieve the above-mentioned requirements. **HTML does not support conditional, looping and external source connectives.**

To overcome the mentioned problems WebDriver is used.

For further information go to <http://seleniumhq.org/docs/02_selenium_ide.html>

## Selenium Builder

**It also operates as a Firefox add-on** and provides an easy-to-use interface for developing and running individual test cases or entire test suites.

This is a **simple alternative to Selenium IDE.**

The **first release was in the 2011** as an open source tool as well. This is **using record and playback capabilities**, and then **export those scripts to several programming languages**.

**Selenium Builder 2.0** was released in 2013 with [**Sauce for Selenium Builder plugin**](https://saucelabs.com/builder) to can run **cross-browser testing in the cloud**.

**The most important feature to come along in Selenium Builder is built-in support for Webdriver.** This is particularly handy for switching from Selenium RC to WebDriver. Since Builder uses the same interface for this as it does for Selenium 1, we can use a mixture of both technologies to build scripts. Selenium Builder can also translate between the two versions with reasonable accuracy.

### Sauce for Selenium Builder Integration

**With this plugin, you can run scripts on Sauce Labs directly from Builder’s interface.**

Simply export the tests to a supported programming language of choice then pick from the supported 100+ browser/OS combos.

**Selenium 2 scripts can be exported** into a wide variety of languages:

* Java,
* Ruby,
* Python,
* PHP,
* Node.js,
* C#

Selenium Builder 2 also **defines a clean JSON-based format for storing scripts**.

### Limitations of Selenium Builder

* **So all limitations that apply for Selenium IDE applies for Builder too.**
* For Selenium 2, Builder stores scripts in its own JSON format. So far, the following languages are partially supported. (Not all Selenium step types are exportable):
  + Java
  + Python

For further information go to <http://sebuilder.github.io/se-builder/>

## WebDriver (Selenium 2)

### Foreword

“**Selenium WebDriver is the successor of Selenium Remote Control (Selenium 1)** which has been officially deprecated.”

“The primary new feature in Selenium 2.0 is the integration of the WebDriver API.”…”**Selenium-WebDriver was developed to better support dynamic web pages where elements of a page may change without the page itself being reloaded.** WebDriver’s goal is to supply a well-designed object-oriented API that provides improved support for modern advanced web-app testing problems.”…

“Selenium-WebDriver makes direct calls to the browser using each browser’s native support for automation… **Selenium 1 was worked the same way for each supported browser. It ‘injected’ javascript functions into the browser when the browser was loaded** and then used its javascript to drive the AUT within the browser. **WebDriver does not use this technique; it** **drives the browser directly using the browser’s built in support for automation.**” (seleniumhq.org)

### Introducing WebDriver

WebDriver is the solution for tests that need a little more than just simple browser actions and a linear execution. **WebDriver uses the full power of programming languages**, creating tests that can do things like read and write external files (data-driven testing[[2]](#footnote-2)), make queries to a database and practically anything else a user can do with a normal application.

So with WebDriver can be added programming logic to tests. **Program flow is controlled by using of condition statements and iteration.**

WebDriver provides an API (Application Programming Interface) and library for each of its supported languages:

* Ruby
* Java
* Groovy
* Perl
* PHP
* Python
* C#
* JavaScript

The **WebDriver can be used for post release validation** with continuous integration tools like [Jenkins](https://wiki.jenkins-ci.org/display/JENKINS/Selenium+Plugin)

### Limitations of WebDriver

WebDriver can't handle:

1. any non-web based (Like Win 32, Java Applet, Java Swing, .Net Client Server…etc.) applications, for example:
   1. desktop based applications
   2. Flash
   3. Silverlight
   4. native OS modal windows[[3]](#footnote-3)
   5. file upload
   6. file download
2. file or image content
3. there is no inbuilt reporting capability; so need to apply third-party plugins like
   1. JUnit
   2. TestNG
4. there is no direct support

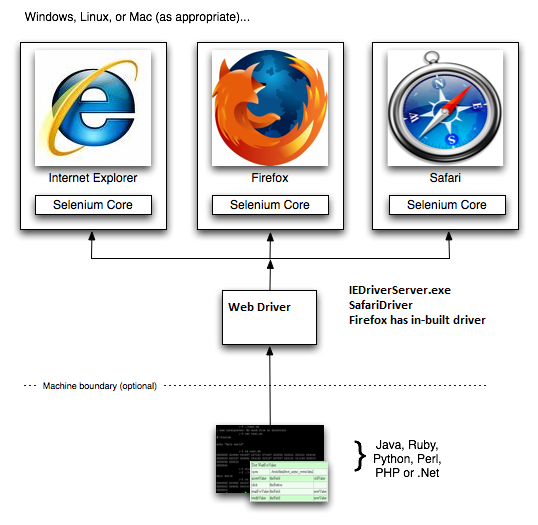
WebDriver configures the LAN connection settings directly, therefore **only a single test can be run on a single machine.**

For further information go to

<http://seleniumhq.org/docs/03_webdriver.html>

<http://www.seleniumframework.com/introduction/what-is-selenium/>

### ****Architecture****



## Remote WebDriver (Grid 2)

“Selenium Grid is a part of the Selenium Suite that specializes on running multiple tests across different browsers, operating systems, and machines in parallel.

Selenium Grid has 2 versions - the older Grid 1 and the newer Grid 2. We will only focus on Grid 2 because Grid 1 is gradually being deprecated by the Selenium Team.”

**Selenium Grid uses a** **hub-node concept:**

**Hub:** The test are running on a single machine called hub.

**Node:** Different machines called nodes do the execution.

**It allows to easily running large test suites:**

* scale by distributing tests on several machines (**parallel execution**)
* **manage multiple environments** from a central point, making it easy to run the tests against a vast combination of browsers / OS
* minimize the maintenance time for the grid by allowing you to implement custom hooks to leverage virtual infrastructure for instance

With Remote WebDriver you can run multiple instances of WebDriver on various operating systems and browser configurations. Remote Controls appear as single ones, so do not have to worry about the actual infrastructure.

### Limitations of Grid

* It has WebDriver at its core. **So all limitations that apply for WebDriver applies for Grid too.**
* Selenium grid by itself cannot run multiple tests in parallel. We **need to use third-party tools like TestNG or JUnit to provide multiple tests to the Grid.**
* **You can only run a limited number of concurrent tests on the same remote control before seriously affecting its stability.** Practically speaking, launching more than 6 browsers on the same WebDriver is not advisable.

The limitations are even more drastic for Internet Explorer.

* Your tests can target multiple nodes to work around the limitation on the number of parallel tests that you can run on a single WebDriver. It can easily be done at the continuous integration build level (one for Internet Explorer, one for Firefox, one for Safari). However, allocating a node to a specific test quickly becomes a nightmare if you want to run your selenium tests in a highly parallelized fashion. **Your tests also become excessively aware of the WebDriver infrastructure, which makes it difficult to evolve the infrastructure transparently.**
* Selenium GRID goes on a network; **we might have to work with network teams in corporate environments** to see if specific ports are blocked and NOT assume that if it works locally, then it works on a network.
* **Setting up a selenium GRID and managing it (creating logs, monitors for health etc.) is a significant time consuming task.**

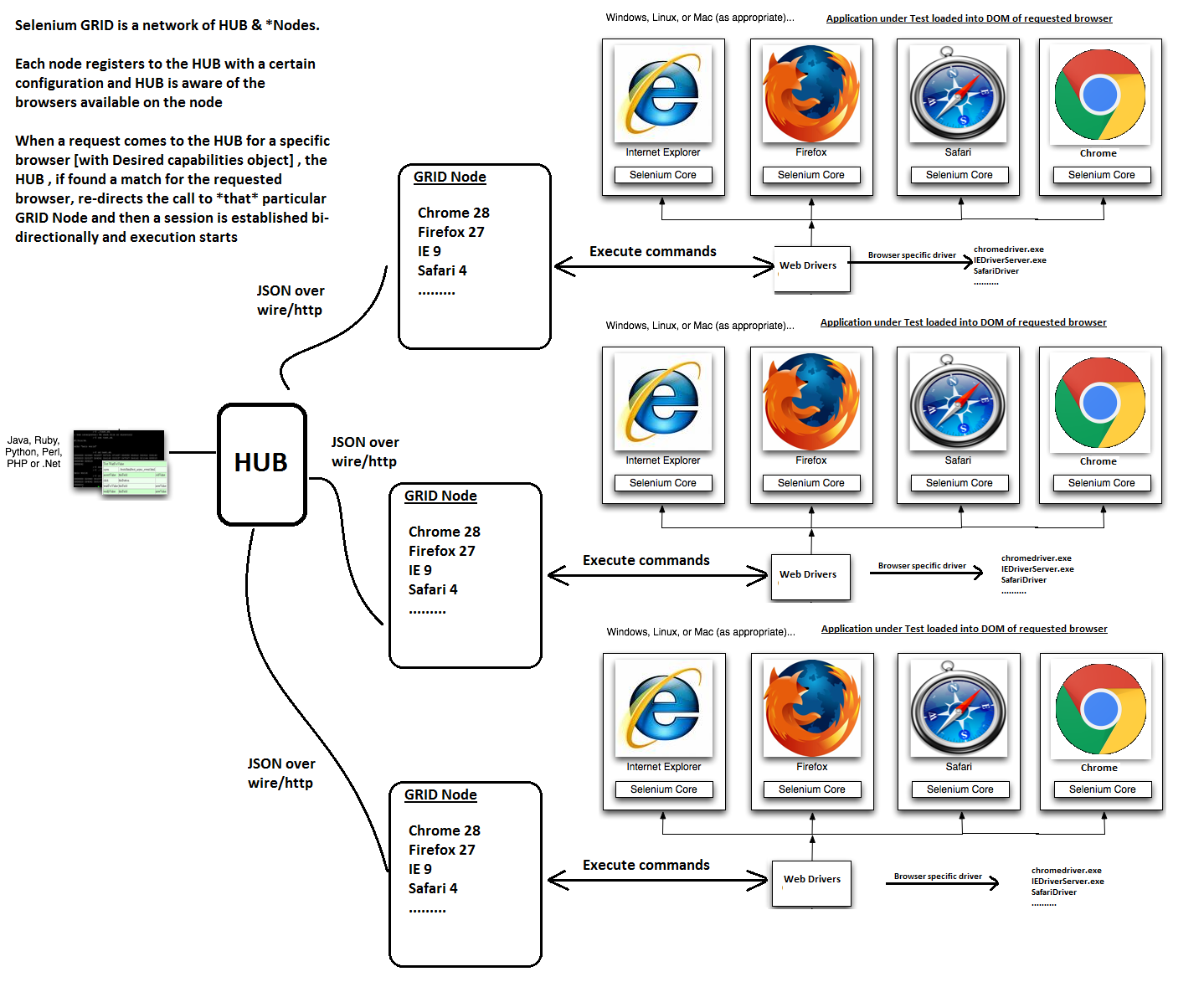
Hence, if you have a corporate environment and you would NOT want to manage it on premise, you have other options like Sauce Labs that of course comes at a cost. So need a cost-benefit analysis before jumping into implementation right away.

For further information go to

<http://code.google.com/p/selenium/wiki/Grid2>

<http://www.seleniumframework.com/intermediate-tutorial/what-is-selenium-grid/>

### ****Architecture****



# What is the Difference between Selenium ID, Selenium RC, and Web Driver

|  |  |  |
| --- | --- | --- |
| Selenium IDE | Selenium RC | Selenium WebDriver |
| It only works in Mozilla browser. | It supports with all browsers like Firefox, IE, Chrome, Safari, Opera etc. | It supports with all browsers like Firefox, IE, Chrome, Safari, Opera etc. |
| It supports Record and playback | It doesn’t supports Record and playback | It doesn’t supports Record and playback |
| Doesn’t required to start server before executing the test script. | Required to start server before executing the test script. | Doesn’t required to start server before executing the test script. |
| It is a GUI Plug-in | It is standalone java program which allow you to run Html test suites. | It actual core API which has binding in a range of languages. |
| Core engine is Javascript based | Core engine is Javascript based | Interacts natively with browser application |
| Very simple to use as it is record & playback. | It is easy and small API | As compared to RC, it is bit complex and large API. |
| It is not object oriented | API’s are less Object oriented | API’s are entirely Object oriented |
| It doesn’t supports of moving mouse cursors. | It doesn’t supports of moving mouse cursors. | It supports of moving mouse cursors. |
| Need to append full xpath with ‘xpath=\\’ syntax | Need to append full xpath with ‘xpath=\\’ syntax | No need to append full xpath with ‘xpath=\\’ syntax |
| It does not supports listeners | It does not supports listeners | It supports the implementation of listeners |
| It does not support to test iphone/Android applications | It does not support to test iphone/Android applications | It support to test iphone/Android applications |

For further information go to

<http://www.softwaretestingclass.com/difference-between-selenium-ide-rc-webdriver/>

# What testing Frameworks can be used along with Selenium?

When we use selenium, then **we make scripts like Script1, Script2, Script3… etc.** **Sometimes** to execute script **we have to get the test data** from XLS file or user. To read the data from XLS file, the process of reading the data from XLS file **is called Parameterization**.

Along with this, **you have to generate the test reports**, we need to know what happened after executing the script, is **script Passed or Failed**?

Also along with the reports, **you need to add logging as well**. If your script is taking more time to execute script then **you need to know what happened at each and every minute**, you need to log each and everything to get idea what script is doing and also at what point script is failed and why.

**So we need a centralized controller which will read the test data, execute the test cases, generate reports and do the logging as well.** In the market, **two frameworks are available** for testing. So that the centralize testing controller are **TestNG or JUnit framework**. You can use selenium with TestNG or JUnit framework. These frameworks will execute the test scripts. They will read the data from XLS file generate the reports and do the logging while executing the scripts.

See more at: <http://www.softwaretestingclass.com/difference-between-selenium-ide-rc-webdriver>

1. **Functional testing** means testing the application against business requirements. Functional testing is executed using the functional specifications given by the client or by the design specifications according to use cases given by the design team. Role of functional testing is to validating the behavior of an application. [↑](#footnote-ref-1)
2. **Data-driven testing (DDT)** is creation of test scripts where test data and/or output values are read from data files instead of using the same hard-coded values each time the test runs. This way, testers can test how the application handles various inputs effectively. It can be any of the data files: Datapools, Excel files, ADO objects, CSV files, ODBC sources [↑](#footnote-ref-2)
3. **Modal window** is a child window that requires the user to interact with it before they can return to the parent application.

   Selenium is not a browser automation tool. It is a web-application automation tool. So testing the print, export etc. dialog boxes are out of the scope and not a capability of the Selenium test engine. [↑](#footnote-ref-3)