**What is Selenium?**

Selenium is purely "**Web Automation Tool**". As simple as that. You can automate user actions and also perform validations like checking existence of an element (image/text/combo-box...) on webpages using selenium. For those who have the following question in mind "Can we automate system applications using Selenium?” the answer is "NO". Selenium is purely for automating web.

**Selenium History and Architecture**

Sources:

http://www.seleniumhq.org/about/history.jsp

http://en.wikipedia.org/wiki/Selenium\_(software)#Selenium\_WebDriver

http://www.aosabook.org/en/selenium.html

The goal of this tutorial is to understand what comprises and defines Selenium. For this we need to look at some of its history and its architecture too. In fact its history is nothing but understanding how it has evolved (architecture) with time and who helped it in its evolution. To be frank with you, to be a Selenium developer it is enough to go through current Selenium WebDriver 2.0 API but we would cover this anyway in our tutorials because we want to pay respect to super heroes who created this awesome tool for us to help in automation and their story be told.

Selenium originally created by Jason Huggins in 2004. Along with contributions from ThoughtWorks engineers and others who are interested in this technology, he released it as an open source under Apache 2.0 license.

Selenium in general is a suite of 3 tools: Selenium IDE, Selenium WebDriver and finally Selenium Grid.

**Selenium Core and Selenium IDE**

Jason Huggins started Selenium project in 2004 while working at ThoughtWorks on one of their in-house application “Time and Expenses” system. He wrote **Selenium core** in JavaScript. Most of the browsers support JavaScript so this worked as a miracle and he was able to automate testing of his application on almost all browsers. Knowing this, his work was much appreciated and the word spread.

Using Selenium core automation was pretty easy to learn even for a non-programmer because it followed a table like syntax. This table operates over the raw Java Script. This table based approach is also called as keyword based approach. Let us see why,

Below we have given an example for Keyword or tabled based programming

Column 1 – ‘type’ is the command (the keyword for expected operation)

Column2 – ‘//div[@id=’welcome text’] ‘ is the element identifier

Column3 – ‘Howdy!!’ is an optional value.

So a user has to remember a bunch of keywords and pass the element locator to perform some action (automated). This keyword based approach took the name of Selenese.

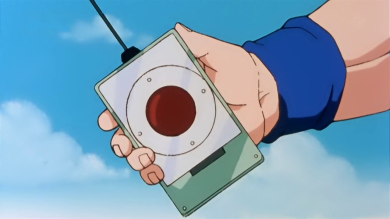
This seems to pretty easy right? But developers and users of Selenium Core sooner found problems in this design; they realized that maintaining a project even of an average size is difficult. Let me put forward some of those problems,

1. Since selenium core is purely written in Selenium, it forced the developers with no option than hosting Selenium Core and tests on server. This is to avoid breaking browser’s security.
2. Giving access to server to host their test is not practical in real time.

In 2006, Shinya Kasatani of Japan became interested in Selenium, and using Firefox extension, he developed a complete integrated development environment-IDE, that’s where the name originated. It is a wrapper around Selenium Core and using this tool one could record, edit and debug tests.

Find Selenium IDE tutorials [🡪here🡨](http://www.questionselenium.com/2012/09/blog-post.html)

**Selenium RC – Remote control**



A Selenium Remote Control is a server, written in Java, which accepts command for the browser via HTTP. This made it possible to write automation for a web application in any programming language that is capable of doing an http request. This means you would write automation in your favorite programming language and kick of an automation run remotely. This is how Selenium RC – Remote Control got its name. Note Selenium RC still uses table/keyword based approach internally. Http proxy internally converts these http requests into appropriate Selenese command.

**WebDriver**

Yes, I did not call it Selenium-WebDriver. This is for a reason. WebDriver is not a part of Selenium initially. WebDriver is another automation framework being developed at the same time frame where Selenium-RC was taking shape. This was developed with intension that tests should be independent of underlying technology. In case of Selenium RC this is not the case. They are dependent on JavaScript support for browser. To eliminate this dependency Selenium WebDriver was developed to make use of browser native language.

**Differences between Selenium RC and WebDriver**

|  |  |
| --- | --- |
| **Selenium RC** | **WebDriver** |
| Uses Selenium Core which makes extensive use of JavaScript | Communicates with a browser using browser native language. |
| Keyword based API | Object based API |
| Supports wide range of languages. Thanks to the idea of http requests. | Supports only Java |
| Need a server to run the test. | We don’t need a server here. |

**Selenium WebDriver**

Selenium WebDriver = Selenium (RC) + WebDriver. Yes you guessed it right; Selenium WebDriver came into existence after mixing two different projects. In 2009, an announcement was made that Selenium RC and WebDriver would merge which eventually called as Selenium WebDriver. This implementation is done by using a browser specific browser driver. Server is no long required here.

Selenium WebDriver is also known as Selenium 2.0 while Selenium RC is known as Selenium 1.0

Directly picked from Wikipedia: Simon Stewart (inventor of WebDriver), who was then with Google and now with Facebook, and David Burns of Mozilla were negotiating with the [W3C](http://en.wikipedia.org/wiki/W3C) to make WebDriver an internet standard. In early 2013, the working draft was released.

Selenium 1.0 provided a rich set of API calls for different browsers. On the other hand Selenium 2.0 aims at provide building blocks on which developers can create their own implementation. Consider this as a specification which is followed by developers. One such project is Watir-webdriver in ruby, which implements the Watir API as a wrapper for Selenium-Webdriver in Ruby.

Watir-WebDriver

Implements

Watir API

Wrapper

Selenium WebDriver - API -Ruby

**What languages are supported by Selenium? And why have we chosen Ruby?**

Selenium is supported by many of the programming languages commonly used. This include: C#, Java, Perl, PHP, Python, Ruby. More details on support for Selenium can be found [here](http://www.seleniumhq.org/about/platforms.jsp).

We have chosen Ruby for following reasons:

* Scripting languages are more convenient to work.
* Learning curve for Ruby is small.
* A survey is conducted by Ken, everyone’s opinion is considered.
* We have not chosen Java, because the learning curve is big.
* We have not chosen Python, because it is very strict on indentation and structure of program which is against our policy.
* We wanted something that is easy to learn even for a person from a non-programming background.

**What are available options in ruby in support of Selenium with Ruby?**

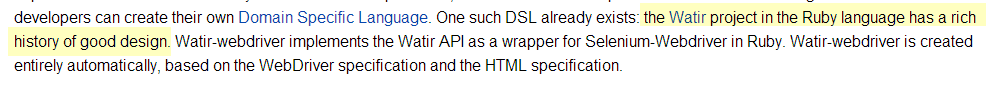
There are various drivers in ruby that provides implementation for Selenium WebDriver specification.

This includes:

1. [Selenium-WebDirver](http://rubygems.org/gems/selenium-webdriver)
2. [Watir-WebDriver](https://rubygems.org/gems/watir-webdriver)
3. [Capybara](http://rubygems.org/gems/capybara)
4. PantomJS

We are currently using Watir-WebDriver for Selenium automation. We always have freedom to choose other options when current option is not meeting our requirement. Headless browser testing is next big thing on list, when the requirement arises or once we have everything settled in Watir WebDriver, we might start exploring this section. From my knowledge, headless browser testing should be done using PantomJS.

See what Wikipedia has to say about Watir Webdriver:



Must read article, I liked the author’s explanation on differences between the options mentioned above:

<http://watirmelon.com/2011/12/03/a-tale-of-three-ruby-automated-testing-apis-redux/>