# Note to the Reader

Hello, and welcome! The Documentation Team would like to welcome you, and to thank you for being interested in Selenium.

We are currently updating this document for the Selenium 2.0 release. This means we are currently writing and editing new material, and revising old material. While reading, you may experience typos or other minor errors. If so, please be patient with us. Rather than withholding information until it’s finally complete, we are frequently checking-in new writing and revisions as we go. Still, we do check our facts first and are confident the info we’ve submitted is accurate and useful. Still, if you find an error, particularly in one of our code examples, please let us know. You can create a new issue (<https://github.com/SeleniumHQ/www.seleniumhq.org/issues>).

We have worked very, very hard on this document. And, as just mentioned, we are once again working hard, on the new revision. Why? We absolutely believe this is the best tool for web-application testing. We feel its extensibility and flexibility, along with its tight integration with the browser, is unmatched by available proprietary tools. We are very excited to promote Selenium and, hopefully, to expand its user community. In short, we really want to “get the word out” about Selenium.

We believe you will be similarly excited once you understand how Selenium approaches test automation. It’s quite different from other automation tools. Whether you are brand-new to Selenium, or have been using it for awhile, we believe this documentation will truly help to spread the knowledge around. We have aimed our writing so that those completely new to test automation can use this document as a stepping stone. However, at the same time we have included a number of advanced, test design topics that should be interesting to the experienced software engineer. In both cases we have written the “Sel-Docs” to help test engineers of all abilities to quickly become productive writing your own Selenium tests. Experienced users and “newbies” alike will benefit from our Selenium User’s Guide.

# Introduction

## Test Automation for Web Applications

Many, perhaps most, software applications today are written as web-based applications to be run in an Internet browser. The effectiveness of testing these applications varies widely among companies and organizations. In an era of highly interactive and responsive software processes where many organizations are using some form of Agile methodology, test automation is frequently becoming a requirement for software projects. Test automation is often the answer. Test automation means using a software tool to run repeatable tests against the application to be tested. For regression testing this provides that responsiveness.

There are many advantages to test automation. Most are related to the repeatability of the tests and the speed at which the tests can be executed. There are a number of commercial and open source tools available for assisting with the development of test automation. Selenium is possibly the most widely-used open source solution. This user’s guide will assist both new and experienced Selenium users in learning effective techniques in building test automation for web applications.

This user’s guide introduces Selenium, teaches its features, and presents commonly used best practices accumulated from the Selenium community. Many examples are provided. Also, technical information on the internal structure of Selenium and recommended uses of Selenium are provided.

Test automation has specific advantages for improving the long-term efficiency of a software team’s testing processes. Test automation supports:

* Frequent regression testing
* Rapid feedback to developers
* Virtually unlimited iterations of test case execution
* Support for Agile and extreme development methodologies
* Disciplined documentation of test cases
* Customized defect reporting
* Finding defects missed by manual testing

## To Automate or Not to Automate?

Is automation always advantageous? When should one decide to automate test cases?

It is not always advantageous to automate test cases. There are times when manual testing may be more appropriate. For instance, if the application’s user interface will change considerably in the near future, then any automation might need to be rewritten anyway. Also, sometimes there simply is not enough time to build test automation. For the short term, manual testing may be more effective. If an application has a very tight deadline, there is currently no test automation available, and it’s imperative that the testing get done within that time frame, then manual testing is the best solution.

## Introducing Selenium

Selenium is a set of different software tools each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium’s key features is the support for executing one’s tests on multiple browser platforms.

## Brief History of The Selenium Project

Selenium first came to life in 2004 when Jason Huggins was testing an internal application at ThoughtWorks. Being a smart guy, he realized there were better uses of his time than manually stepping through the same tests with every change he made. He developed a Javascript library that could drive interactions with the page, allowing him to automatically rerun tests against multiple browsers. That library eventually became Selenium Core, which underlies all the functionality of Selenium Remote Control (RC) and Selenium IDE. Selenium RC was ground-breaking because no other product allowed you to control a browser from a language of your choice.

While Selenium was a tremendous tool, it wasn’t without its drawbacks. Because of its Javascript based automation engine and the security limitations browsers apply to Javascript, different things became impossible to do. To make things worse, webapps became more and more powerful over time, using all sorts of special features new browsers provide and making these restrictions more and more painful.

In 2006 a plucky engineer at Google named Simon Stewart started work on a project he called WebDriver. Google had long been a heavy user of Selenium, but testers had to work around the limitations of the product. Simon wanted a testing tool that spoke directly to the browser using the ‘native’ method for the browser and operating system, thus avoiding the restrictions of a sandboxed Javascript environment. The WebDriver project began with the aim to solve the Selenium’ pain-points.

Jump to 2008. The Beijing Olympics mark China’s arrival as a global power, massive mortgage default in the United States triggers the worst international recession since the Great Depression, The Dark Knight is viewed by every human (twice), still reeling from the untimely loss of Heath Ledger. But the most important story of that year was the merging of Selenium and WebDriver. Selenium had massive community and commercial support, but WebDriver was clearly the tool of the future. The joining of the two tools provided a common set of features for all users and brought some of the brightest minds in test automation under one roof. Perhaps the best explanation for why WebDriver and Selenium are merging was detailed by Simon Stewart, the creator of WebDriver, in a joint email to the WebDriver and Selenium community on August 6, 2009.

*“Why are the projects merging? Partly because WebDriver addresses some shortcomings in selenium (by being able to bypass the JS sandbox, for example. And we’ve got a gorgeous API), partly because selenium addresses some shortcomings in WebDriver (such as supporting a broader range of browsers) and partly because the main selenium contributors and I felt that it was the best way to offer users the best possible framework.”*

## Selenium’s Tool Suite

Selenium is composed of multiple software tools. Each has a specific role.

### Selenium 2 (aka. Selenium WebDriver)

Selenium 2 is the future direction of the project and the newest addition to the Selenium toolkit. This brand new automation tool provides all sorts of awesome features, including a more cohesive and object oriented API as well as an answer to the limitations of the old implementation.

As you can read in [*Brief History of The Selenium Project*](#_Brief_History_of), both the Selenium and WebDriver developers agreed that both tools have advantages and that merging the two projects would make a much more robust automation tool.

Selenium 2.0 is the product of that effort. It supports the WebDriver API and underlying technology, along with the Selenium 1 technology underneath the WebDriver API for maximum flexibility in porting your tests. In addition, Selenium 2 still runs Selenium 1’s Selenium RC interface for backwards compatibility.

### Selenium 1 (aka. Selenium RC or Remote Control)

As you can read in [*Brief History of The Selenium Project*](#_Brief_History_of), Selenium RC was the main Selenium project for a long time, before the WebDriver/Selenium merge brought up Selenium 2, the newest and more powerful tool.

Now Selenium 1 is deprecated and is not actively supported (mostly in maintenance mode).