

What is automation testing?

Those in the technology sector will be familiar with testing. That is, creating and developing a product or piece of software, and frequently putting it through its paces until you get it right.

It's part of the product development cycle and is vitally important for ensuring your product is as good as you want it to be.

Nowadays, testing has become more sophisticated and more advanced. We have started to utilize automation testing to help ease the workload of the rest of the team and provide clear and accurate results.

Here, we'll delve further into automation testing, including what it is, and how to start using it in your business.

What is automation testing?

Automation testing is the process of testing software and other tech products to ensure it meets strict requirements. Essentially, it's a test to double-check that the equipment or software does exactly what it was designed to do. It tests for bugs, defects, and any other issues that can arise with product development.

Although some types of testing, such as regression or functional testing can be done manually, there are greater benefits of doing it automatically. Automation testing can be run at any time of the day. It uses scripted sequences to examine the software. It then reports on what's been found, and this information can be compared with earlier test runs. Automation developers generally write in the following programming languages: C#, JavaScript, and Ruby.

Many software businesses will have an appointed [QA \(quality assurance\) automation tester](#). They design and write the test scripts in the beginning stages. The QA automation tester will work with automation test engineers and product developers to actually test the software and products. They will form a team and control the test automation initiatives, and use different types of test automation frameworks to establish the best one for successful test automation.

When starting to work with unit testing frameworks, the team needs to know about its attributes, runners, assertions, screen shots, test suites, and CI (continuous integration). Popular user testing frameworks include JUnit for Java and Pytest for Python.

Adhering to testing protocols is important in the [technology industry](#). It's critical for continuous delivery (CD) and continuous testing (CT). DevOps and agile software development teams will use both CD and CT in their testing strategies.

By choosing automation testing, businesses are able to streamline their testing procedures to deliver the maximum return on investment (ROI). Why? Because automated testing can shorten development life cycles, eliminate the possibility of human error, and automate mundane and monotonous tasks.

Why is automation needed?

Some teams simply don't have the time or resources to be manually testing software. Automation can help with this. It can significantly reduce the time it takes to test products because it runs quickly and efficiently. This puts time back into the hands of developers and production managers, who can divert their efforts into other aspects of the project. It can greatly [boost productivity](#) as a result.

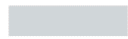
Using automation technology also means that testing can be done more frequently, improving overall functionality. Software development cycles call for repeated testing, often the same test over and over again. Automation testing makes this possible, without taking team members away from other work. It can also deliver more accurate and reliable results than manual testing alone. Further ensuring that the product is ready for market or to move onto the next stage of development. This validation gives the team a boost to continue developing.

Most importantly, automation benefits product development. That's because when software, an app, or another product can be designed and produced more efficiently, it makes way for continuous development once it's been launched. Essentially, the business will be able to work on more software and products, even with the same amount of team members, thanks to automation. Not only does this mean they perfect the final products they put out, but it also means they are creating new software all the time.

What are the benefits of automation testing?

Software testing has many benefits, which is why SaaS businesses across the globe are utilizing automation technology. Here are some of the biggest benefits of using automation testing for software development:

- **Detailed reporting capabilities** - Automation testing uses well-crafted test cases for various scenarios. These scripted sequences can be incredibly in-depth, and provide detailed reports that simply wouldn't be possible when done by a human. Not to mention providing them in a shorter amount of time.
- **Improved bug detection** - One of the main reasons to test a product is to detect bugs and other defects. Automation testing makes this process an easier one. It's also able to analyze a wider test coverage than humans may be able to.



55% passed



14 of 29 Testers

Apple iPhone 11



3 of 3 passed

Apple iPad Pro



2 of 3 passed

Samsung Galaxy S9



3 of 3 failed

Samsung Galaxy S8



1 of 3 passed

- **Simplifies testing** - Testing is a routine part of the operations of most SaaS and tech companies. Making it as simple as possible is key. Using automation is extremely beneficial. When automating test tools, the test scripts can be reused. Manual testing, meanwhile, calls for a single code line to be written for the same test case, each time it needs to be run
- **Speeds up the testing process** - Machines and automated technology work faster than humans. Along with improved accuracy, this is why we use them. In turn, this shortens your software development cycles.
- **Reduces human intervention** - Tests can be run at any time of day, even overnight, without needing humans to oversee it. Plus, when it's conducted automatically, this can also reduce the risk of human error.
- **Saves time and money** - Testing can be time-consuming. Though automation may require an initial investment, it can save money in the long run to become more cost-effective. Team members use their time in other areas and are no longer required to carry out manual testing in many situations. This improves their workflow.

How can enterprises utilize automation?

Enterprises should be [utilizing automation](#) in order to improve their business processes and operating systems, particularly those in the technology industry. Automation provides valuable tools for businesses to use to their advantage, whether that be for improving product delivery times or to meet increasing security standards.

Once you have established which test you'll be using, you need to set goals as a benchmark to see how it performs. Without setting goals, it's going to be difficult for you to utilize the test result to its full potential. Keep focused on this one objective, and don't be afraid to run separate tests where needs be. Consider what it is you're trying to achieve, and how this test can help you do that.

Divide your tests into logical, smaller tests. Larger, more complex tests, are more difficult to run. Team members that aren't writing test code can be moved to other areas of the product development process to better utilize their time. Utilizing automation is all about making testing easier and improving business practices.

This graph gives us a glimpse into the future, showing how machines may soon be able to complete work previously carried out by humans:

[Image source](#)

Which tests should be automated?

It's not possible to automate all testing at once. So, you need to decide which tests to automate first. Let's look at the kind of tests that can benefit from automation and therefore should be automated:

- Tests that can lead to failure because of human error
- Repetitive and monotonous tests
- Extensive tests that require multiple data sets
- Tests that can't be performed manually
- Tests that would take a lot of time manually
- High-risk tests
- Tests that need to run on several different hardware and/or software platforms

What are the types of automation testing?

There are five key types of automation testing. Each can be used in different circumstances, depending on the application under test. You can analyze each one to see which would be best for you, or you can test them out with a trial. This is sometimes the best way to know which tool you should be using. Here are the main types of automation testing tools:

Code analysis

Code analysis consists of different testing tools, including dynamic analysis and static analysis. You can apply different ones to tackle separate tests. For instance, some look for possible security flaws, while others check for usability. To run these tests, the developer will need to write code. Once this has been done, though, there's no human interference for the rest of the testing process.

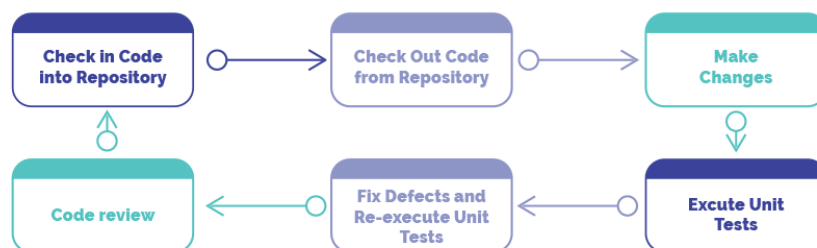
Unit tests

Unit testing is all about checking individual components of the software or product, as you would developing iOS technology for an iPhone or Android for Samsung. This means that each element of the software is fully tested before the finished version is. These tests can be written by developers, but now that automation testing has come into play, there's no need.

Businesses would typically deploy these types of tests in the software development phase of the production process.

This graph shows the unit test life cycle:

UNIT TEST LIFE CYCLE



Integration tests

Integration tests, also known as end-to-end tests, are often more complicated to set up than some other tests. The application models are integrated and tested as a group. This means that communication between each module can be tested, to figure out how well they work as a whole.

Automated accepted tests

Automated accepted tests (AAT) are similar to behavior-driven development (BDD) and automated acceptance test-driven development (AATDD). The acceptance test is created before a new feature is developed. It sets a precedent for the feature to meet and is usually written by developers, the business, and [quality assurance \(QA\)](#) in tandem. In future, they can also be used as regression tests.

Smoke tests

This type of testing is used to check whether the product is stable or not. If it's not stable, it gets sent back to the developers marked as an 'unstable build'. They can then run further tests if needed to identify the root cause of the problem. This diagram shows how the smoke test process works:

The automation test process

The automation test process should be something the team is familiar with tech pros, and particularly developers. Wondering how to start automation testing from scratch? Here's a rundown of the automation test process, from selecting the test tool to execution.

Select the test tool to utilize

We've been through the different types of automation testing. Now's the time to select which one best suits your needs. For instance, if your goal is to detect a specific bug within the software, you may be more inclined to use code analysis automation testing.

You can select from a wide range of test automation tools and [web apps](#) on the market, such as Selenium IDE, WebDriver, UFT, Ranorex, Cucumber, Testcomplete, and Appium. You should be able to access some on Microsoft with many offering a tutorial on how to use them. Some are even open source. So you'll need to have a solid understanding of each tool and how it could benefit your testing.

Establish the scope of automation

This refers to the area test that will be automated, and how big it's going to be. In this step, you will walk through your team's state, prepare the test data and the environment in which the test is to take place. Automated testing means that human interference is rarely needed, so you can leave the test to run when you're away.

Here are some additional factors to consider when determining the scope of your automation testing:

- The complexity of the test
- The main goals of the testing
- How many resources and business components are used
- Technical feasibility

Planning stage

The planning stage will look at creating a testing strategy. This includes your automation test tool and how you're going to use it, your framework design and features, and your goals. What do you want to discover by the end of the testing? You will also need to create a detailed timeline for scripting and executing test cases with the development team who will be carrying out the tests. You will also have to consider in-scope and out-of-scope items of automation.

The execution stage

This is the penultimate stage in the automation testing process. Now you have your test tool and strategy, it's time to run your test. Depending on the test you chose, your developers may need to write code, learn Web Services Description Language (WSDL), and run the test automatically from the get-go with the tool's API or user interface.

[API testing](#) may also be necessary before you begin. Your automated tests will also generate a report for you to analyze with the rest of your team. It provides a summary of the testing so far. This can be used in future tests to compare.

Ongoing maintenance

Ongoing maintenance should be part of your automation testing process. This is particularly needed if you plan on running tests in the future with your reusable test scripts. Because even though you'll have a script ready to use, they will still need to be updated by the time you run your next test if your tool has changed.

Ongoing maintenance also provides reassurance as the team makes their way to the next stage, or backtracks to run another test. When a method has been repeatedly tried and tested, you're more likely to be provided with an accurate outcome.

Finding the right tools to make automation a breeze

Automation can truly change the way you run your business. Now is the time to embrace new technology, and learn methodologies to make the working day run more effectively and efficiently. Don't be afraid to trial a few different tools to see which one works best for you. After all, each business is different, and so is the software or products you are creating.

Remember the following steps:

- Selecting the test tool
- Establishing the scope of the automation
- Planning and strategy
- Execution
- Ongoing test maintenance

Using automated testing is the best way to ensure your business stays on top of debugging, defects, and issues that can quickly turn the production process sour if not ironed out as soon as possible.

KNOW ABOUT THE BEST AUTOMATION TESTING TOOLS FOR 2022

Test automation can't succeed without the help of effective test automation tools. As per the user behavior, test automation trends have changed a lot such as machine learning, and artificial intelligence providers' advanced techniques to test automation. It will be more beneficial to know which automation testing tools are the best as per these trends.

Here's a list of the top 10 automation testing tools :

- Selenium
- Appium
- Katalon Studio
- Cucumber
- HPE Unified Functional Testing (UFT)
- SoapUI
- TestComplete
- Worksoft
- IBM Rational Functional Tester (RFT)
- **Telerik Test Studio**

What are the Top 3 Things you Should Consider Before Selecting the Best software Automation Testing Tools?

1. List out your current requirement
2. Think about your project budget
3. Select the appropriate automation testing team, who can use any kind of tool

Selenium

The best free automation testing tools for web application testing. Selenium tool currently takes first place among all the automation testing tools. Selenium is among the most used automated testing tool for web applications. It is an open-source platform that is compatible with numerous browsers, operating systems, and

programming languages. Selenium features detailed and advanced automation scripts, supports the execution of parallel tests, and integrates other software testing tools. As the best web testing tool, The main thing is the Selenium Framework, it will help you to make code maintenance easy.

Selenium Website: <http://www.seleniumhq.org/>

License: Free

Appium

If you search a mobile automation testing tools list then Appium will always be at the top. Appium is designed on a server, and the user gains access to its automation framework through vendors. Appium has a high compatibility rate, enabling it to automate different mobile apps in any language using any test framework. Appium features high compatibility, testing doesn't require recompiling or SDK tools and can run on mobile and computer operating systems using WevDriver protocols. With the **appium studio**, you can easily analyze, debug, and execute tests on real devices.

Website: <http://appium.io/>

License: Free

Katalon Studio

Katalon Studio can integrate with both Selenium and Appium. The automation testing tool can simplify the automation testing of API, web apps, and mobile applications. Katalon Studio also works with other tools like JIRA, Git, Slack, and such. The main features include compatibility with Windows, Mac, and Linux, an easy user-friendly interface, and numerous keywords for building test cases.

Katalon Studio Website: <https://www.katalon.com/>

License: Free

Cucumber

Cucumber is an open-source platform for Behavior Driven Development (BDD). It showcases an interesting clientele list including Canon and Paypal. Cucumber is

designed to support only web access, focusing on enhanced user satisfaction. Major features include code compatibility with other automation frameworks, uses Gherkin code (which is in simple English), and supports various languages such as Ruby, Java, Groovy, etc.

Cucumber Website: <https://cucumber.io/>

License: Free

HPE Unified Functional Testing (UFT)

Previously known as QuickTest Professional (QTP), HPE Unified Functional Testing (UFT) is highly preferred among users for its cross-platform automation testing tools. It improves a central platform for developers and testers to improve the quality of results while optimizing expenditure. The UFT can automate numerous applications including Web, Desktop, Mobile, Oracle, Java, and SAP. The main features include scripting language is in VBScript, data-driven testing, and high compatibility across numerous browsers, platforms, and operating systems.

UFT Website: <https://www.microfocus.com/en-us/products/uft-one/overview>

License: Paid

SoapUI

SoapUI is designed by Smartbear and provides runs a creative API Test Automation Framework for Representational State Transfers (REST) and Service-Oriented Architectures Program (SOAP). The main features include simple and reusable scripts, asynchronous testing, and creative test generation components.

SoapUI Website: <https://www.soapui.org/>

License: Free and Paid

TestComplete

The final tool on the list is **TestComplete**. It is the best automation testing tool for desktop applications. TestComplete tool supports the testing of mobile, desktop, and web applications. It supports different scripting languages, keyword-driven testing,

regression, and distributed testing. It also integrates well with SmartBear. Main features include reusable scripts, building blocks for scripts, and user-friendly recording and playback tools.

TestComplete Website: <https://smartbear.com/product/testcomplete/overview/>

License: Paid

Worksoft

Worksoft is an automation testing tool that is specially designed for SAP. Worksoft is equipped to tackle all of the most difficult automation problems. WorkSoft functions on a code-free continuous automation testing module. Main features include compatibility with all stages of an SAP project, possesses an integrated test management tool, and finally it can automate SuccessFactors, Ariba Network, Concur, Syclo, and SAP Fiori User Experience.

IBM Rational Functional Tester (RFT)

Rational Functional Tester (RFT) is designed by IBM as a commercial and best automation testing tool for 2022. RFT provides efficient solutions to automating tests for functionality, regression, GUI, and data-driven tests. It has high compatibility with different development frameworks such as Siebel, Java, Flex, Net, and SAP. Major features of RFT include supporting two scripting languages (Java and VB.Net), integrating with IBM Rational Quality Manager, and using natural language and application screenshots for the monitoring testing process.

Telerik Test Studio

Telerik Test Studio automation tools are highly compatible to test applications on desktop, web, and mobile devices. The automation tool is designed to support GUI, functionality, and API tests. It features video recording and playback tools for monitoring progress and is compatible with different browsers. The main features include compatibility with real coding languages, runs on two scripture languages, parallel tests, and creative recording features.

Conclusion

If you are still confused while selecting the best automation software testing tools then you can discuss it with our team of experienced QA Testers. **Testrig Technologies** is the award-winning software testing company offering all most all testing services including **automation testing services**

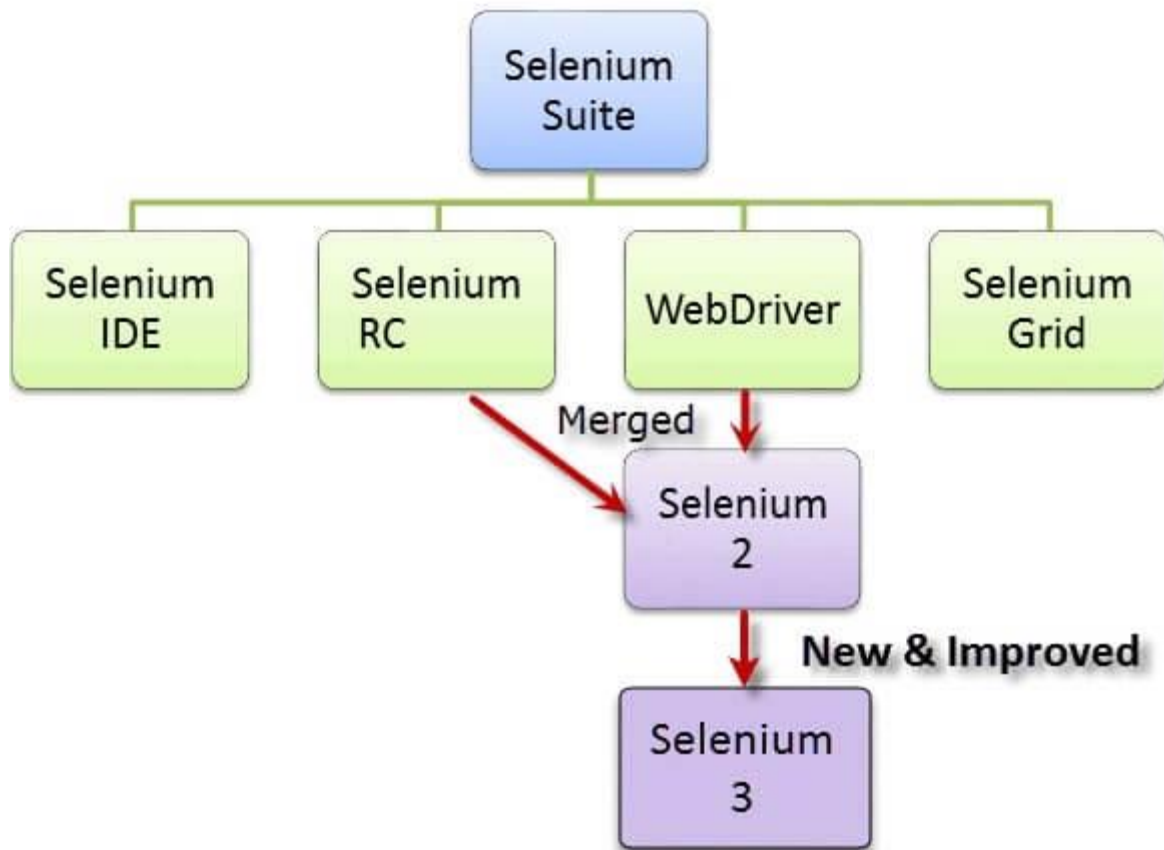
What is Selenium? Introduction to Selenium Automation Testing

What is Selenium?

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

Selenium Software is not just a single tool but a suite of software, each piece catering to different Selenium QA testing needs of an organization. Here is the list of tools

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- WebDriver
- Selenium Grid



Introduction to Selenium

At the moment, Selenium RC and WebDriver are merged into a single framework to form **Selenium 2**. Selenium 1, by the way, refers to Selenium RC.

In this tutorial, you will learn:

- [What is Selenium?](#)
- [Who developed Selenium?](#)
- [The Same Origin Policy Issue](#)
- [Birth of Selenium Remote Control \(Selenium RC\)](#)
- [Birth of Selenium Grid](#)
- [Birth of Selenium IDE](#)
- [Birth of WebDriver](#)
- [Birth of Selenium 2](#)
- [So, Why the Name Selenium?](#)
- [Brief Introduction Selenium IDE](#)
- [Brief Introduction Selenium Remote Control \(Selenium RC\)](#)
- [Brief Introduction WebDriver](#)
- [Selenium Grid](#)
- [Note on Browser and Environment Support](#)

- [How to Choose the Right Selenium Tool for Your Need](#)
- [A Comparison between Selenium and QTP\(now UFT\)](#)
- [Advantages of QTP over Selenium](#)

Who developed Selenium?

Since Selenium is a collection of different tools, it had different developers as well. Below are the key persons who made notable contributions to the Selenium Project

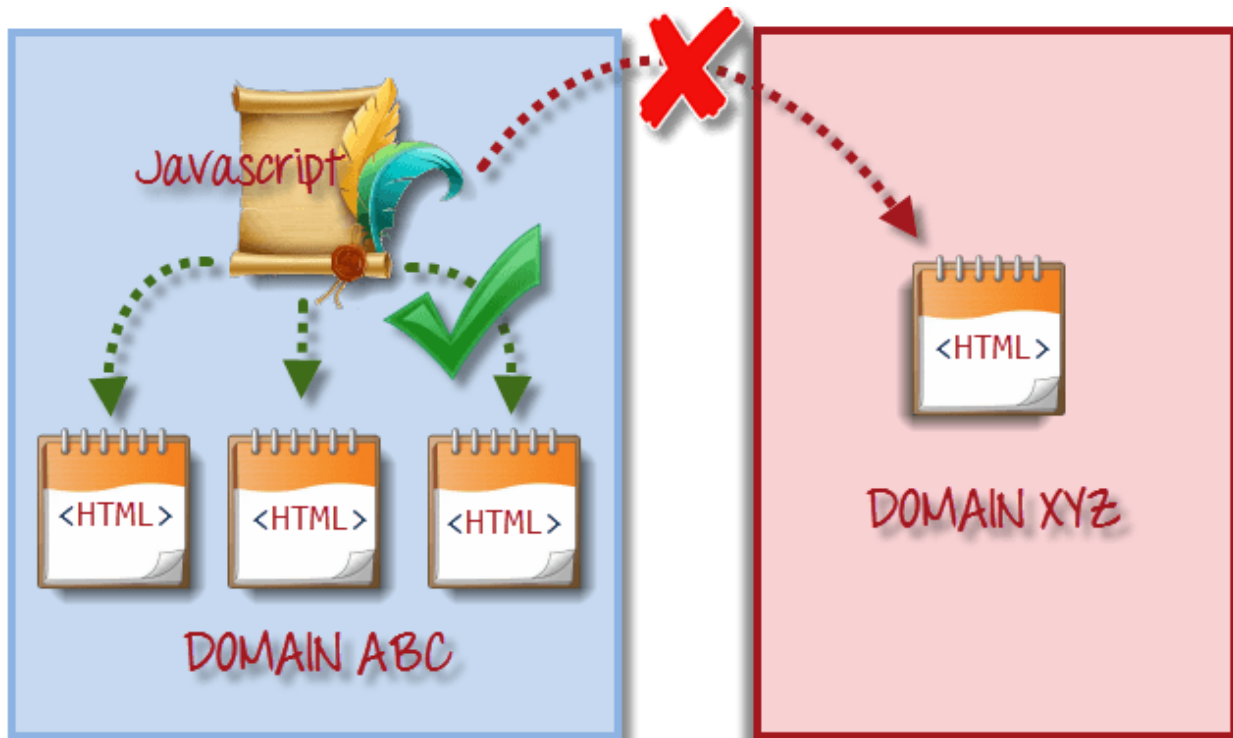


Primarily, Selenium was **created by Jason Huggins**. As an engineer at ThoughtWorks, he was working on a web application that required frequent testing. Having realized that the manual [Testing](#) of their application was becoming more and more tedious, he created a [JavaScript](#) program that would automate the browser's actions. He named this program as the "**JavaScriptRunner**."

Seeing potential in this idea to help automate web applications, he made JavaScriptRunner open-source. It was later re-named as **Selenium Core**.

The Same Origin Policy Issue

Same Origin policy prohibits JavaScript code from accessing elements from a domain that is different from where it was launched. Example, the HTML code in www.google.com uses a JavaScript program "randomScript.js". The same origin policy will only allow randomScript.js to access pages within google.com such as google.com/mail, google.com/login, or google.com/signup. However, it cannot access pages from different sites such as yahoo.com/search or guru99.com because they belong to different domains.



under same origin policy, a JavaScript program can only access pages on the same domain where it belongs. It cannot access pages from different domains

This is the reason why prior to Selenium RC, testers needed to install local copies of both Selenium Core (a JavaScript program) and the web server containing the web application being tested so they would belong to the same domain

Birth of Selenium Remote Control (Selenium RC)



Paul Hammant

Unfortunately; testers using Selenium Core had to install the whole application under test and the web server on their own local computers because of the restrictions imposed by the **same origin policy**. So another ThoughtWork's engineer, **Paul Hammant**, decided to create a server that will act as an HTTP proxy to "trick" the browser into believing that Selenium Core and the web application being tested come

from the same domain. This system became known as the **Selenium Remote Control** or **Selenium 1**.

Birth of Selenium Grid



Patrick Lightbody

Selenium Grid was developed by **Patrick Lightbody** to address the need of minimizing test execution times as much as possible. He initially called the system “**Hosted QA**.” It was capable of capturing browser screenshots during significant stages, and also of **sending out Selenium commands to different machines simultaneously**.

Birth of Selenium IDE



Shinya Kasatani of Japan created **Selenium IDE**, a Firefox extension that can automate the browser through a record-and-playback feature. He came up with this idea to further increase the speed in creating test cases. He donated Selenium IDE to the Selenium Project in **2006**.

Birth of WebDriver



Simon Stewart

Simon Stewart created WebDriver circa **2006** when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core. **It was the first cross-platform testing framework that could control the browser from the OS level.**

Birth of Selenium 2

In **2008**, the whole Selenium Team decided to merge WebDriver and Selenium RC to form a more powerful tool called **Selenium 2**, with **WebDriver being the core**. Currently, Selenium RC is still being developed but only in maintenance mode. Most of the Selenium Project's efforts are now focused on Selenium 2.

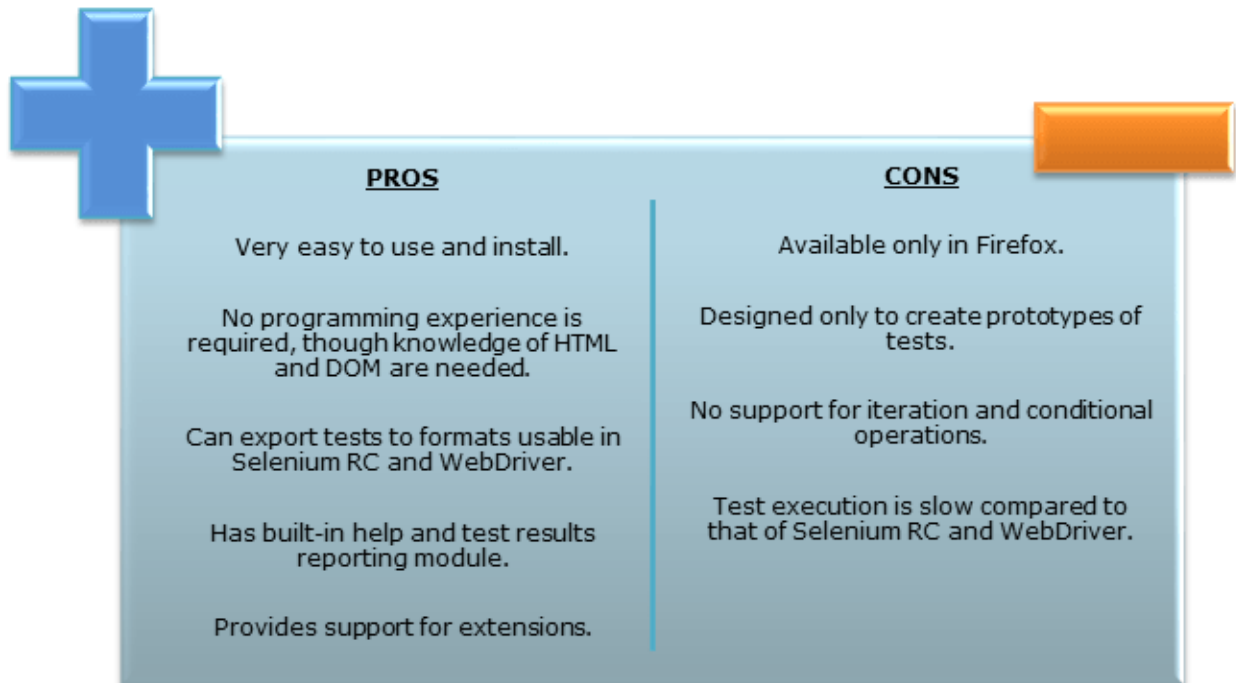
So, Why the Name Selenium?

The Name Selenium came from a joke which Jason cracked once to his team. During Selenium's development, another [automated testing](#) framework was popular made by the company called **Mercury Interactive** (yes, the company who originally made QTP before it was acquired by HP). Since Selenium is a well-known antidote for Mercury poisoning, Jason suggested that name and his teammates took it. So that is how we got to call this framework up to the present.



Brief Introduction Selenium IDE

Selenium Integrated Development Environment (IDE) is the **simplest framework** in the Selenium suite and is **the easiest one to learn**. It is a **Firefox plugin** that you can install as easily as you can with other plugins. However, because of its simplicity, Selenium IDE should only be used as a **prototyping tool**. If you want to create more advanced test cases, you will need to use either Selenium RC or WebDriver.



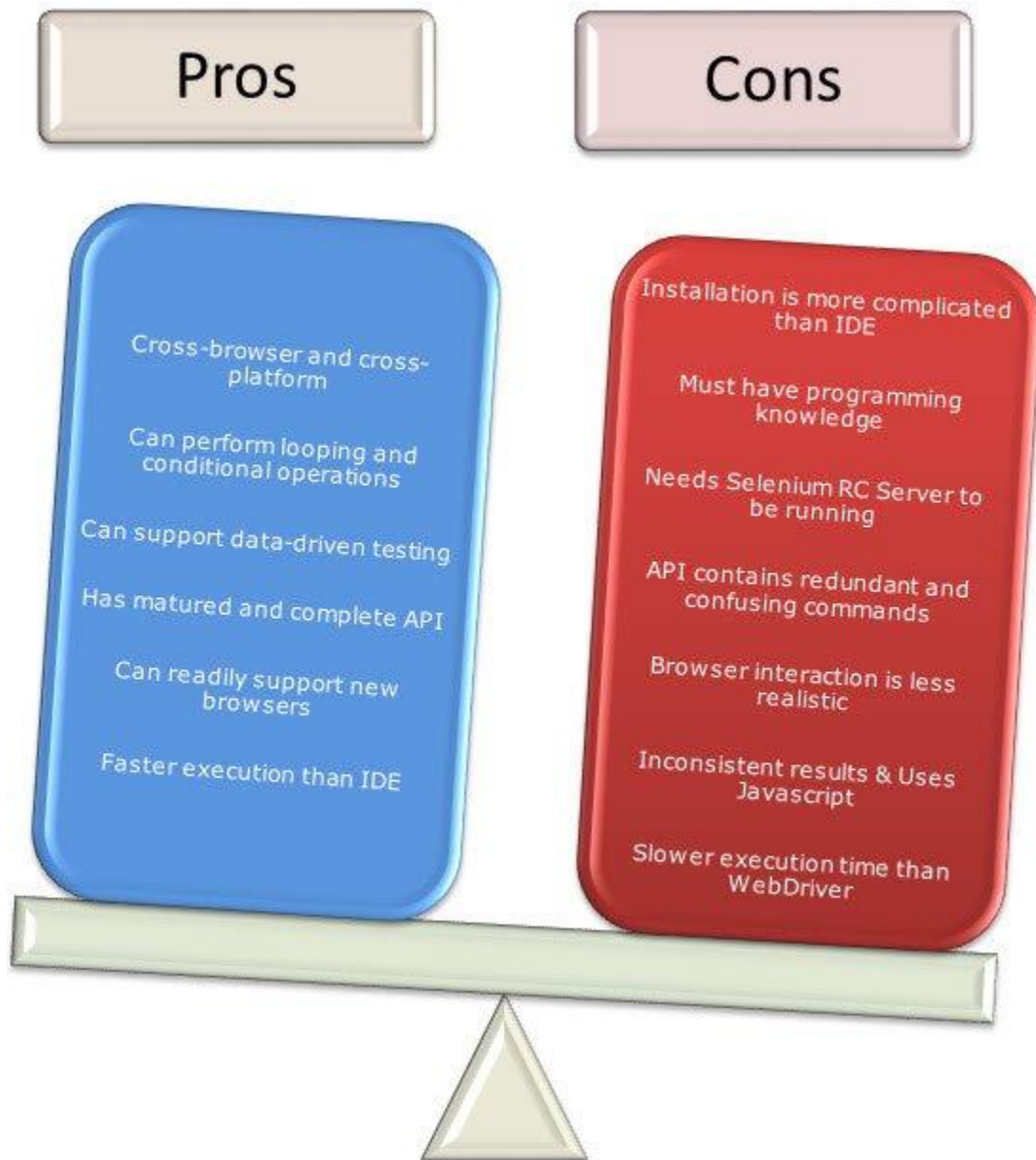
<u>PROS</u>	<u>CONS</u>
Very easy to use and install.	Available only in Firefox.
No programming experience is required, though knowledge of HTML and DOM are needed.	Designed only to create prototypes of tests.
Can export tests to formats usable in Selenium RC and WebDriver.	No support for iteration and conditional operations.
Has built-in help and test results reporting module.	Test execution is slow compared to that of Selenium RC and WebDriver.
Provides support for extensions.	

Brief Introduction Selenium Remote Control (Selenium RC)

Selenium RC was the **flagship testing framework** of the whole Selenium project for a long time. This is the first automated [web testing](#) tool that **allowed users to use a programming language they prefer**. As of version 2.25.0, RC can support the following programming languages:

- [Java](#)
- [C#](#)
- [PHP](#)
- Python
- Perl

- Ruby



Brief Introduction WebDriver

The WebDriver proves itself to be **better than both Selenium IDE and Selenium RC** in many aspects. It implements a more modern and stable approach in automating the browser's actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for Selenium Automation Testing. **It controls the browser by directly communicating with it.**

The supported languages are the same as those in Selenium RC.

- Java
- C#
- PHP
- Python
- Perl
- Ruby

Pros	Cons
Simpler installation than Selenium RC	Installation is more complicated than Selenium IDE
Communicates directly to the browser	Requires programming knowledge
Browser interaction is more realistic	Cannot readily support new browsers
No need for a separate component such as the RC Server	Has no built-in mechanism for logging runtime messages and generating test results
Faster execution time than IDE and RC	

Selenium Grid

Selenium Grid is a tool **used together with Selenium RC to run [parallel tests](#)** across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

Features:

- Enables **simultaneous running of tests in multiple browsers and environments.**
- **Saves time** enormously.
- Utilizes the **hub-and-nodes** concept. The hub acts as a central source of Selenium commands to each node connected to it.

Note on Browser and Environment Support

Because of their architectural differences, Selenium IDE, Selenium RC, and WebDriver support different sets of browsers and operating environments.

	Selenium IDE	WebDriver
Browser Support	Mozilla Firefox	Internet Explorer versions 6 to 11, both 32 and 64-bit
		Microsoft Edge version 12.10240 & above (partial support some functions in development)
		Firefox 3.0 and above
		Google Chrome 12.0. and above
		Opera 11.5 and above
		Android – 2.3 and above for phones and tablets (devices & emulators)
Operating System	Windows, Mac OS X, Linux	iOS 3+ for phones (devices & emulators) and 3.2+ for tablets (devices & emulators)
		HtmlUnit 2.9 and above
		All operating systems where the browsers above can run.

Note: Selenium WebDriver is termed as the successor of Selenium RC which has been deprecated & officially announced by SeleniumHQ.

How to Choose the Right Selenium Tool for Your Need

Tool	Why Choose?
Selenium IDE	<ul style="list-style-type: none"> To learn about concepts on automated testing and Selenium, including: Selenese commands such as type, open, clickAndWait, assert, verify, etc. Locators such as id, name, xpath, css selector, etc. Executing customized JavaScript code using runScript Exporting test cases in various formats. To create tests with little or no prior knowledge in programming. To create simple test cases and test suites that you can export later to RC or WebDriver. To test a web application against Firefox and Chrome only.
Selenium RC	<ul style="list-style-type: none"> To design a test using a more expressive language than Selenese To run your test against different browsers (except HtmlUnit) on different operating systems. To deploy your tests across multiple environments using Selenium Grid. To test your application against a new browser that supports JavaScript. To test web applications with complex AJAX-based scenarios.
WebDriver	<ul style="list-style-type: none"> To use a certain programming language in designing your test case. To test applications that are rich in AJAX-based functionalities.

Tool	Why Choose?
	<ul style="list-style-type: none"> To execute tests on the HtmlUnit browser. To create customized test results.
Selenium Grid	<ul style="list-style-type: none"> To run your Selenium RC scripts in multiple browsers and operating systems simultaneously. To run a huge test suite, that needs to complete in the soonest time possible.

A Comparison between Selenium and QTP(now UFT)

Quick Test Professional(QTP) is a proprietary automated testing tool previously owned by the company **Mercury Interactive** before it was **acquired by Hewlett-Packard in 2006**. The Selenium Tool Suite has many advantages over QTP as detailed below –

Advantages and Benefits of Selenium over QTP

Selenium	QTP
Open source, free to use, and free of charge.	Commercial.
Highly extensible	Limited add-ons
Can run tests across different browsers	Can only run tests in Firefox, Internet Explorer and Chrome
Supports various operating systems	Can only be used in Windows
Supports mobile devices	QTP Supports Mobile app test automation (iOS & Android) using HP solution Mobile Center
Can execute tests while the browser is minimized	Needs to have the application under test to be visible on the desktop
Can execute tests in parallel.	Can only execute in parallel but using Quality Center which is again a paid

Advantages of QTP over Selenium

Advantages of QTP over Selenium	
QTP	Selenium
Can test both web and desktop applications	Can only test web applications
Comes with a built-in object repository	Has no built-in object repository
Automates faster than Selenium because it is a fully featured IDE.	Automates at a slower rate because it does not have a native third party IDE can be used for development

Data-driven testing is easier to perform because it has built-in global and local data tables.	Data-driven testing is more cumbersome since you have programming language's capabilities for setting values f
Can access controls within the browser (such as the Favorites bar, Address bar, Back and Forward buttons, etc.)	Cannot access elements outside of the web application u
Provides professional customer support	No official user support is being offered.
Has native capability to export test data into external formats	Has no native capability to export runtime data onto exte
Parameterization Support is built	Parameterization can be done via programming but is di
Test Reports are generated automatically	No native support to generate test /bug reports.

Though clearly, [QTP](#) has more advanced capabilities, Selenium outweighs QTP in three main areas:

- **Cost**(because Selenium is completely free)
- **Flexibility**(because of a number of programming languages, browsers, and platforms it can support)
- **Parallel testing**(something that QTP is capable of but only with use of Quality Center)

Summary

- The entire Selenium Software Testing Suite is comprised of four components:
 - Selenium IDE, a Firefox add-on that you can only use in creating relatively simple test cases and test suites.
 - Selenium Remote Control, also known as Selenium 1, which is the first Selenium tool that allowed users to use programming languages in creating complex tests.
 - WebDriver, the newer breakthrough that allows your test scripts to communicate directly to the browser, thereby controlling it from the OS level.
 - Selenium Grid is also a tool that is used with Selenium RC to execute parallel tests across different browsers and operating systems.
- Selenium RC and WebDriver was merged to form Selenium 2.
- Selenium is more advantageous than QTP in terms of costs and flexibility. It also allows you to run tests in parallel, unlike in QTP where you are only allowed to run tests sequentially.