

Home > Blogs > Selenium Testing: Everything You Need To Know



Selenium Testing: Everything You Need To Know

May 6, 2022 by  Kazuaki Matsuo

GUIDES

Testing is a vital part of the development cycle and is essential for ensuring the quality and stability of your applications. By [performing application testing](#), you can find and fix bugs in your code before they have a chance to cause problems for your users. Additionally, application

Join HeadSpin at Mobile World Congress Barcelona from 27 Feb - 2 Mar [Schedule Meeting](#)

headspin

Products Resources Company HeadSpin University Pricing

[Log in](#)

[Book Free Trial](#)

test your applications quickly and efficiently. Selenium Testing is a popular tool for automated testing, as it allows you to write tests in various programming languages and run them across different browsers and operating systems.

Share this

Hi there! Welcome 😊

English

This guide will cover everything you need to know about Selenium Testing, including what it is, how it is supposed to work, and how you can use it to test your web applications.

What is Selenium?

Selenium is a popular open-source testing tool used for web application testing. It enables testers to write automated tests in various programming languages to test the functionality of web applications. Selenium tests can be run on many different browsers and operating systems.

Selenium Testing Tool is an excellent tool for automating **web application testing**. It can be used to test the functionality of web applications on various browsers and operating systems. Selenium Software Testing is a great way to automate your web application testing.

Components of Selenium

Selenium is a collection that consists of three major components:

1. Selenium IDE: Selenium IDE is a Firefox add-on that enables testers to record and playback the recorded automated tests. Selenium IDE is easy to use and lets you create and run automated tests quickly. Selenium IDE Chrome also includes a built-in debugger that enables you to troubleshoot your tests. To use Selenium IDE, you first need to install the Selenium add-on for Firefox or Chrome. You can then open Selenium IDE by clicking on the "Selenium" icon in the Firefox or Chrome toolbar.

Once Selenium IDE is open, you can start recording your tests by clicking on the "Record" button. Selenium will then begin recording all of your actions as you perform them in the browser. To stop the test recording, click on the "Stop" button. You can then playback your tests by clicking on the "Play" button. Selenium will then replay all of the actions that you recorded.

2. Selenium WebDriver: Selenium WebDriver is an open-source tool used for automating web browser interaction from a user perspective. With Selenium WebDriver, you can write tests that simulate user interactions with a web application. Selenium WebDriver is available for many different programming languages, such as Java, C#, Python, and Perl.

WebDriver provides a powerful and flexible **test automation framework** that enables you to create automated tests for your web applications easily. It also includes several over-the-top features, such as automatically discovering elements on a web page and capturing screenshots of your tests.

3. Selenium Grid: Selenium Grid distributes your tests across multiple machines or virtual machines (VMs). Selenium Grid enables you to test parallelly on various devices or VMs, allowing you to scale your test automation quickly. Selenium Grid is a crucial part of the overall Selenium testing suite and will enable you to execute your automated tests much faster.

The Selenium Server is a part of Selenium Grid that can be installed locally on your device or hosted on a separate system/server.

English

How does Selenium work?

Selenium WebDriver works by emulating the actions of a user. When you write a Selenium test, you specify the actions you want the user to take, and Selenium will automatically execute those actions on the browser.

For instance, if you need to test a login functionality, you would write a Selenium test to enter the username and password into the appropriate fields and click on the login button. Selenium would automatically execute those actions on the browser and report any errors.

How to get started with Selenium Testing Tool

With Selenium Testing Tool For Automation, you can write automated tests in various programming languages. This enables you to test the functionality of your web application on different browsers and operating systems. Selenium Testing is a great way to automate your web application testing.

To use the Selenium Testing Tool, you must install the Selenium WebDriver. The Selenium WebDriver is a browser automation tool that enables you to run Selenium tests on different browsers and operating systems. Once you have successfully installed the Selenium WebDriver, you can start writing your Selenium tests.

Installing Selenium is easy. You can install Selenium by following these steps:

1. Go to the Selenium website: <https://www.selenium.dev/>
2. Click on the "**Download**" button from the menu.
3. Select the "**Selenium IDE**" option and choose the browser from all the listed browsers.
4. Click on the "**Download**" button.
5. Run the "**SeleniumIDE_4.0.0.exe**" file or the latest build.
6. Follow the prompts to install Selenium IDE.
7. The Selenium will be installed, and you can start writing your Selenium tests.

When writing Selenium tests, you should always keep in mind the following:

- The goal of your Selenium test is to find bugs in your web application.
- Your Selenium tests should be concise.
- You should only use the Selenium WebDriver When you are sure about how to use the tools and scripts.

English

Once you have written your Selenium tests, you can run them on different browsers and operating systems. To do this, you will need to use a Selenium Grid. A Selenium Grid is a server that enables you to run multiple Selenium tests simultaneously on different browsers and operating systems.

Also check: [How to Write an Automated Test Script Using Selenium](#)

The Selenium benefits include:

- Efficient and accurate web application testing.
- The ability to test your web application on multiple browsers and operating systems.
- The ability to run more than one test at the same time.

With a Selenium Grid, you can significantly reduce your time to test your web application. And by using a Selenium Grid, you can ensure that your web application is fully functional before releasing it to users. So if you want to improve your web application testing, consider using a Selenium Grid. It's one of the best ways to automate your web application testing!

Selenium in an Agile Environment

Selenium is an all-in-one tool that can help you streamline your agile testing process. By following the tips below, you can ensure that your automated tests are practical and efficient and that they play a valuable role in your agile development cycle:

Automated tests should be run frequently as part of the continuous integration process.

- Tests should be written to allow them to be run quickly and easily.
- Tests should be designed to test a specific functionality or behavior and should not be too complex.
- New features and changes should be accompanied by automated tests to ensure that the application's functionality remains intact.
- Automated tests should supplement manual testing rather than replace it altogether.

In agile development, Selenium Testing is typically used in the following ways:

- As part of the [regression testing](#) process, to ensure that existing features continue to work as expected after new code has been added.
- To verify that new features are working as expected before they are released to production.
- To help identify and troubleshoot bugs in web applications at both business and development levels before releasing the app.

Also read: [Tips to Know About Selenium Automation Testing](#)

Different types of Selenium Tests

English

Selenium Testing is a popular open-source testing tool used for web application testing. It enables testers to write automated tests in various programming languages to test the functionality of web applications. Selenium tests can be run on many different browsers and operating systems.

There are different types of Selenium Tests that you can write. The most common types of Selenium Tests are:

- **Unit Tests**
- **Functional Tests**
- **Integration Tests**
- **Regression Tests**

Unit Tests

Unit Tests are the simplest type of Selenium tests. A Unit Test verifies that a single unit of code behaves as expected.

When writing a Unit Test, you should first create a test case. A test case is a set of instructions that tests a particular feature or function of your code. To create a test case, you will need to:

- Define the expected outcome of the test.
- Verify that the code under test behaves as expected.

Once your test case is written, you can run it using the Selenium WebDriver. Open your browser and go to the page where your test is located. Then, enter your test case into the browser's address bar and press "Enter." Your Unit Test should automatically run, and you will be able to see if the code under test behaved as expected.

If you seem to be having any issue with a specific unit of code or need help troubleshooting a bug in your code, many online resources can help. Some useful resources include:

- Tutorials and guides on Selenium Testing and Selenium for mobile testing from websites like Selenium.dev
- Online communities and forums dedicated to Selenium Testing
- Commercial support from companies like HeadSpin

Example of writing a selenium unit test:

```
// Define the WebDriver instance and set the desired capabilities.  
let webDriver = new FirefoxDriver();  
webDriver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);  
webDriver.manage().window().maximize();  
  
// Define the expected outcome of the test and verify that the code under test behaves  
let title = webDriver.getTitle();  
Assert.assertEquals("Selenium Testing", title);  
  
// Close the WebDriver instance.  
webDriver.quit();  
</script>
```

We defined the Selenium WebDriver instance in the above example and set the desired capabilities. We then defined the expected outcome of our test and verified that the code under test behaved as expected. Finally, we closed the WebDriver instance. This is an example of how to write a Selenium Unit Test.

Functional Test

Functional Tests are similar to Unit Tests, but they test the functionality of an entire web application. When writing a Functional Test, you should always keep in mind the following:

- The goal of your Functional Test is to find bugs in all the functional components of your web application.
- Your Functional Test should be concise but have to be written for each functional module of your app.
- You should only use the Selenium WebDriver functional tests when all your components are passed from the development stage.

When writing a Functional Test, you will first need to identify the different areas of your web application that you want to test. Once you have selected these areas, you can create a test case for each one. Once your test cases are written, you can run them using the Selenium WebDriver. Open your browser and go to the page where your tests are located. Then, enter your test case into the browser's address bar and press "Enter." Your Functional Test should automatically run, and you will be able to see if the code under test behaved as expected.

Example of writing a functional test:

In the Selenium WebDriver, a special function called "webDriverWait" can wait for a particular condition to occur before executing a test. This function can wait for a page to load, for a particular element to be visible on the page, or for a certain condition to be met. The following example will show you how to use the "webDriverWait" function to wait for a particular element to be visible on the page:

```
<script>
let webDriver = new FirefoxDriver();
webDriver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

// Define the expected outcome of the test and verify that the code under test behaves
let title = webDriver.getTitle();
Assert.assertEquals("Selenium Testing", title);

// Wait for a specific element to show on the page before continuing with the test.
webDriverWait(driver, 30) // Define timeout in milliseconds
.until(function() { return $('navbar_link').isDisplayed(); } );
</script>
```

In this example, we first opened the "selenium_testing.html" file in our web browser using the Selenium WebDriver. We then used the "webDriverWait" function to wait for a particular element, the "navbar_link" element, to be visible on the page before continuing with our test. After waiting for 30 seconds (as defined in the "implicitlyWait" timeouts), we verified that this condition occurred and executed our test code as expected.

Integration Tests

Integration Tests are used to test the integration between different parts of your web application. When writing an Integration Test, you should always keep in mind the following:

- The goal of Integration Testing in Selenium is to find bugs in your web application that arise from integrating multiple components.
- Your Integration Test should be short, and it should verify that all individual components work properly when integrated.

Example of writing integration test in Selenium

One useful feature of the Selenium WebDriver is its ability to wait for certain conditions before executing a test. This can be used, for example, to wait for a page to load or an element to show on the page before continuing with the test. You can use the "webDriverWait" function along with one of several different timeouts.

For example, you could use the "implicitlyWait" timeout, which will automatically wait for an indefinite time until a condition occurs before continuing with your test. Alternatively, you could use one of the other timeouts, such as "until," which will only wait a specified amount of time before continuing your test. Using these features can help you improve the accuracy and efficiency of your Integration Tests by ensuring that they are executed properly every time.

To write an Integration Test in Selenium, you will first need to create a new file called "integration_test.html" in your test directory. This file will contain your test code, which will be executed in your web browser using the Selenium WebDriver.

Once you have created the "integration_test.html" file, you can begin writing your Integration Test. You will first need to include the Selenium WebDriver library in your HTML file to do this. You can do this by adding the line of code below to your script:

```
<script src="https://selenium-release.storage.googleapis.com/3.141.59/selenium-server.
```

Next, you will need to write your test code. In this example, we will be testing the "login" functionality of a web application. First, we need to define the expected outcome of our test and then write the code that will verify that this outcome occurs. In this case, we expect the "login" button to be displayed on the page after entering our username and password.

Finally, we will need to use the "**WebDriverWait**" function to wait for the "login" button to be visible on the page before continuing with our test. After waiting for 30 seconds (as defined in the "**implicitlyWait**" timeouts), we verified that this condition occurred and executed our test code as expected.

End-to-End Tests

End-to-End Tests are used to test an entire web application from start to finish. When writing an **end-to-end test**, you should always keep in mind the following:

- The goal of your End-to-End Test is to find bugs in your web application that arise as a result of the correlation between multiple components.
- Your End-to-End Test should be concise, and it should verify that all individual components work properly when integrated.

Once your test cases are written, you can run them using the Selenium WebDriver. Open your browser and go to the page where your tests are located. Then, enter your test case into the browser's address bar and press "Enter." Your End-to-End Test should automatically run, and you will be able to see if the code under test behaved as expected.

End-to-End Selenium Test Example

To write an End-to-End Test in Selenium, you must create a new file called "end_to_end_test.html" in your test directory. This file will contain your test code, which will be executed in your web browser using the Selenium WebDriver.

Once you have created the "end_to_end_test.html" file, you can begin writing your End-to-End Test. You will first need to include the Selenium WebDriver library in your HTML file to do this. You can do it by adding the given line of code:

```
<script src="https://selenium-release.storage.googleapis.com/3.141.59/selenium-server.
```

Next, you will need to write your test code using the Selenium WebDriver API functions such as "click," "type," and "waitForPageToLoad." In this example, we are testing a simple login page that requires users to enter their username and password before accessing the application. We are then verifying that the expected outcome of our test occurred by checking if the login button is visible on the page after we have entered our credentials in the login form field.

Finally, we will use some of Selenium's built-in timeouts to ensure that our End-to-End Test runs . For example, we will use the "implicitlyWait" timeout to wait for 30 seconds before

continuing with our test code to allow any slow or loading pages to load completely. We will also use the "webDriverWait" function to wait for a specified period before executing our test code to ensure that all application components are fully loaded and working properly.

Tips for running successful Selenium Tests

Certain tips can help you write better selenium scripts and automate your application tests more efficiently. Some of these tips are:

1. Always start by writing a simple test case that verifies the functionality of a single component.
2. Keep your tests concise, and verify that all individual components work properly when integrated.
3. Use the Selenium WebDriver API functions such as "click", "type", and "waitForPageToLoad".
4. Make sure to include timeouts in your test code to ensure that your tests run smoothly.
5. Use the Selenium Grid for parallel testing to speed up your tests' execution.
6. Always run your tests on a variety of different browsers and operating systems.
7. Keep your test code well organized and maintainable to be easily reused and modified in the future.

Limitations of Selenium WebDriver

Although Selenium is one of the best tools for automating your tests on multiple devices, it still has limitations. Some of them are mentioned below:

- Selenium WebDriver cannot interact with flash or Java applets.
- Selenium WebDriver is not capable of handling complex animations.
- Selenium WebDriver cannot recognize text inside images.
- Selenium WebDriver has some difficulty dealing with dynamically generated pages.
- Selenium WebDriver can be difficult to use when testing web applications that use Ajax or ReactJS.

Troubleshooting common issues with Selenium Tests

Troubleshooting issues with Selenium Tests can be challenging, but a few techniques can help you identify and resolve these issues efficiently and quickly.

- One of the best ways to troubleshoot Selenium Tests is using a Selenium Grid, which allows you to run your tests parallel on multiple browsers and operating systems. This can help you identify performance or compatibility problems that may not be visible when running your tests on a single browser or device.

- In addition, it is vital to make use of all of the features available with the Selenium WebDriver, such as screenshots and video recording tools. These can be extremely useful for debugging purposes, as they allow you to easily capture detailed information about any errors or bugs that occur during your tests.
- Finally, it is also essential to test for security vulnerabilities using the Selenium Security Testing Framework. This can help you locate and fix any potential security issues in your web applications.

By following the tips mentioned, you can ensure that your Selenium Tests are running smoothly and efficiently.

Selenium WebDriver 4.0

Selenium WebDriver 4.0 was released in October 2021 and included several new features and improvements. Some of the key features include:

- Support for running Selenium tests on mobile devices, including Android and iOS. This allows testers to automate their **mobile app testing** using the same Selenium framework to test web applications.
- The ability to run several tests parallelly on multiple browsers and operating systems using the improved version of Selenium Grid can significantly improve performance and efficiency.
- New security features have been added to help you identify potential vulnerabilities in your web applications, such as SSL verification and certificate pinning.
- Improved debugging tools make it easier to troubleshoot issues with your Selenium Tests and quickly resolve any bugs or errors during execution.
- W3C Standardization: Selenium WebDriver 4.0 has been standardized by the W3C, making it easier to use with other tools and frameworks.
- Relative Locators: This new feature allows you to locate elements using relative coordinates, which can be especially useful when testing dynamic web applications.

Overall, Selenium WebDriver 4.0 is a powerful tool that offers many new features and improvements that make it easier to automate your web application testing.

Additional resources on Selenium Testing

There are many resources available on Selenium Testing. Some of these resources include:

- The official documentation for the Selenium project: <http://docs.seleniumhq.org/>
- The Selenium Blog: <https://www.selenium.dev/blog/>
- The Selenium Wiki: <https://github.com/SeleniumHQ/selenium/wiki>
- Online tutorials and courses on Selenium Automation Testing:
<https://www.udemy.com/topic/selenium-webdriver/free/>

- The Selenium Google Group: <https://groups.google.com/g/selenium-users>

Whether you are new to Selenium Mobile Testing or have been using this tool for some time, there's a load of information available online to help you learn more about this powerful testing tool and improve your web application testing. So make sure you take advantage of these resources!

The Bottom Line

If you are looking for a powerful and flexible tool for automating your web application testing, then Selenium Testing Tool is the right choice. With Selenium automation testing, you can easily find bugs in your web application and perform efficient and accurate testing on multiple browsers and operating systems. So why wait? Start using Selenium Testing today!

Selenium Testing Frequently Asked Questions (FAQs)

Q1: Is Selenium easy to learn?

A: Yes, Selenium is relatively easy to learn compared to other testing tools. There are many online resources available to help you [start learning Selenium](#).

Q2: What languages can be used with Selenium?

A: You can use many different programming languages with Selenium, including Selenium Java, C#, Python, Selenium Javascript, and Ruby.

Q3: How much does Selenium cost?

A: Selenium is free and open-source software.

Q4: Why is Selenium so widely used?

A: Selenium is used by many companies and organizations because it is a powerful testing tool that is easy to learn and use. The Selenium community is more significant than any other tool in the market, with comprehensive support. This makes the tool highly resourceful and cost-effective for those looking to use it.

Q5: What are the benefits of using Selenium?

A: Some of the key benefits of using Selenium include finding bugs in your web application quickly and easily, automating your web application testing, and running tests on multiple browsers and operating systems simultaneously.

Q6: What's the difference between Selenium and other testing tools?

A: Selenium is unique from other testing tools because it is open source and can be used with different programming languages. It's also very versatile in nature, as it is not just one

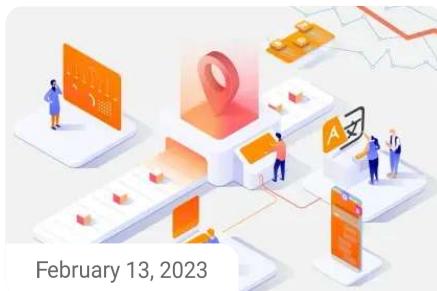
English

tool but a set of tools built into a single framework. This framework can be used in any language that supports DOM, giving it an edge over its other counterparts.

Q7: How can we use the Selenium tool with HeadSpin

A: You can use the Selenium tool with HeadSpin to perform automated web and mobile application testing on multiple browsers and operating systems. Using this powerful combination, you can quickly find bugs in your applications, improve the quality of your applications, and optimize your apps for better performance.

Related blogs

[Browse all blogs →](#)

February 13, 2023

Improving customer experience amidst global recession through localization testing

[Read More →](#)

February 13, 2023

Gaps to avoid in UI test automation strategy amidst economic distress

[Read More →](#)

February 10, 2023

Future of Test Automation with Latest Trends in Software Testing

[Read More →](#)

headspin

[Products](#)[Solutions](#)[HeadSpin for Every](#)[Support](#)[Documentation](#)[Resource Center](#)[Company](#)[About HeadSpin](#)[Why Choose HeadSpin?](#)[English](#)

HeadSpin Platform	Mobile App Testing	Industry	Global Device Infrastructure	HeadSpin University	Press Resources
Add-on Products	Cross Browser Testing	HeadSpin for Telcos	Repository	Webinars & Events	Partners
Audio-Visual Platform	Performance Optimization	Testing Solution for Banking Apps	FAQS	Podcast	Leadership Team
Regression Intelligence	Experience Monitoring	Testing Solution for Retail	Integrations	Converge	Careers
Create Your Own Lab	Android Testing	Industry	Helpdesk	Blogs	Awards
	iOS App Testing	HeadSpin for Gaming Companies		Tutorials	
	Appium – Mobile Test Automation	Testing Solution for Digital Natives		Case Studies	
	Smart TV Testing	HeadSpin Automobile Testing Solution		Latest Feeds	



Copyright © 2023 HeadSpin, Inc. All Rights Reserved. | [Cookies](#) [Privacy](#) [Terms](#)



English