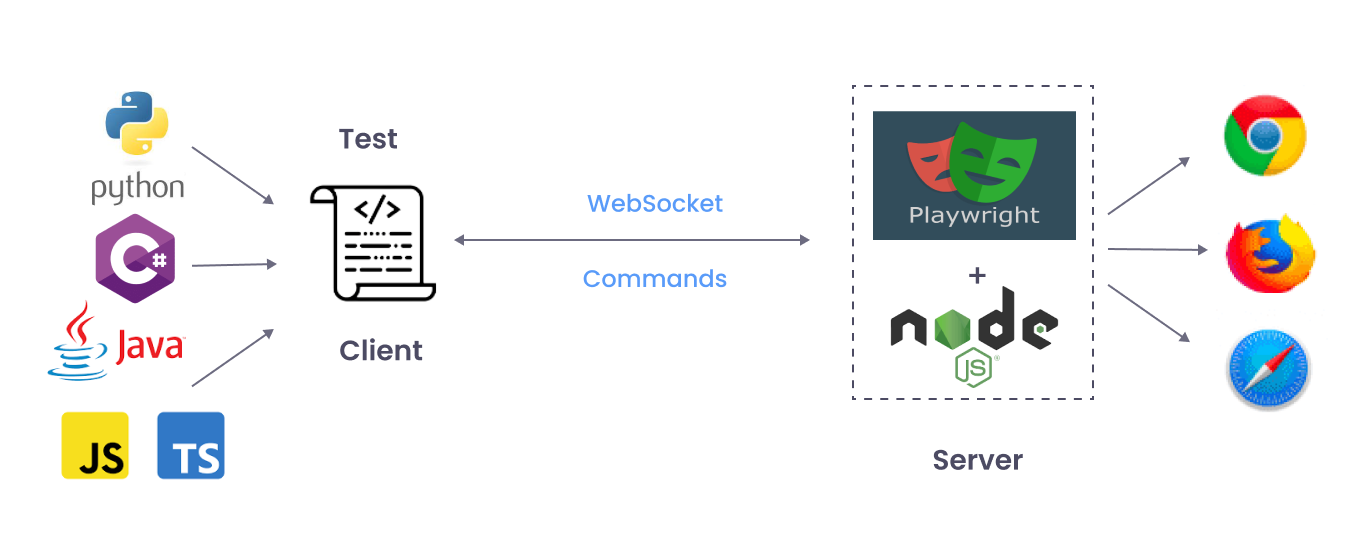
**Playwright**

Playwright is an open-source Node.js library that enables efficient E2E [testing](https://testomat.io/blog/how-to-start-with-javascript-unit-testing-the-best-frameworks-for-process-automation/) of web apps, making Playwright automation a key player in modern software testing



**Key Features**

* **Cross-browser Compatibility.** With Playwright, automation is possible for all modern browsers, including the Chromium family (Chrome and Edge), Firefox, and Webkit (Safari).
* **Cross-platform Capability.** The framework allows tests to be run locally, on CI, and on different platforms: Windows, Linux, and macOS.
* **Multilingual Support**. The tool eliminates the need for the automation engineer to be tied to a specific programming language. It lets you write tests in TypeScript, [JavaScript](https://testomat.io/blog/javascript-testing-frameworks-you-must-know/), Python, .NET, C#, and Java. All about supported [Playwright](https://testomat.io/blog/playwright-locators-handle-elements-inputs-buttons-dropdown-frames-etc/) languages in [Docs](https://playwright.dev/docs/languages)

**Functions Of Playwright That Make It The Best Choice For Web Testing**

1. **Codegen:** This feature allows you to create tests by recording every user action. Importantly, you can write such tests in any supported programming language.
2. **Playwright inspector:** With this feature, you can step-by-step monitor test execution, create selectors, view click points, and much more.
3. **Trace Viewer:** This tool makes it easy to understand why a test failed and debug it. With it, you can access the recordings of your [Playwright tests](https://testomat.io/blog/grouping-playwright-tests-for-improved-framework-efficiency/), enabling you to navigate from one action to another and see what happened during each of them.
4. **Automatic Waiting:** One of the key features of Playwright automation which sets it head and shoulders above more traditional alternatives like Selenium. This feature involves Playwright waiting for elements to be ready before executing any action. This helps eliminate a significant problem faced by testers – unstable tests.
5. **Test Tagging:** With this feature, users can group and run test cases together.
6. **Built-in Reports:** Reporting on test results plays a crucial role in the QA process. [Playwright](https://testomat.io/blog/python-playwright-tutorial-for-web-automation-testing/) comes with various default reporters, including List, Line, Dot, HTML, JSON, and JUnit. Additionally, the framework allows the creation of custom reports and supports integration with third-party reporters such as Allure, Monocart, Tesults, ReportPortal, Currents, and Serenity/JS.
7. **Video and Screenshot Support:** The framework provides the capability to use photos and video recordings of test results, enabling QA automation engineers to quickly identify issues through Playwright automation.
8. **Test Retries:** This feature automatically retries failed tests, which is especially useful for tests prone to occasional failures.
9. **Parallel Test Execution:** The tool supports parallel test execution by running multiple worker processes simultaneously. Moreover, [Playwright](https://testomat.io/blog/playwright-locators-handle-elements-inputs-buttons-dropdown-frames-etc/) can achieve even greater test parallelization by running tests on different machines.
10. **Browser Context Creation:** The tool supports running tests in isolated browser contexts. Creating these elements takes a few milliseconds, significantly faster than launching a new browser. This is another reason why testing with Playwright is known for its high speed.

| **Criteria** | **Playwright** | **Selenium** |
| --- | --- | --- |
| Browser Support | Chromium, Firefox, and WebKit (note: Playwright tests [browser projects](https://applitools.com/blog/lightning-fast-playwright-tests-cross-browser/#:~:text=Playwright%20tests%20browser%20projects), not stock browsers) | Chrome, Safari, Firefox, Opera, Edge, and IE |
| Language Support | Java, Python, .NET C#, TypeScript and JavaScript. | Java, Python, C#, Ruby, Perl, PHP, and JavaScript |
| Test Runner Frameworks Support | Jest/Jasmine, AVA, Mocha, and Vitest | Jest/Jasmine, Mocha, WebDriver IO, Protractor, TestNG, JUnit, and NUnit |
| Operating System Support | Windows, Mac OS and Linux | Windows, Mac OS, Linux and Solaris |
| Architecture | Headless browser with event-driven architecture | 4-layer architecture (Selenium Client Library, JSON Wire Protocol, Browser Drivers and Browsers) |
| Integration with CI | Yes | Yes |
| Prerequisites | NodeJS | Selenium Bindings (for your language), Browser Drivers and Selenium Standalone Server |
| Community Support | Smaller but growing set of community resources | Large, established collection of documentation and support options |
| Open Source | Free and open source, backed by Microsoft | Free and open source, backed by large community |