Playwright vs Selenium vs Cypress PDF

Selenium is the long-standing standard with wide language and browser support through WebDriver, Cypress runs inside the browser with a friendly runner mainly for JavaScript, and Playwright controls browsers over a modern protocol with auto-waiting and several languages. The best choice depends on your team skills, browsers, application and existing tests.

Key takeaways

- Selenium has the widest language and grid ecosystem and a large existing user base.
- Cypress offers a developer-friendly runner but follows a different architecture with some limits.
- Playwright combines multi-browser support, auto-waiting and strong debugging tools.
- Team skills and existing code often matter more than benchmark claims.
- Run a small pilot with your own application before deciding.

What is comparison of Playwright, Selenium and Cypress?

Playwright, Selenium and Cypress are open-source tools for automating web browsers to test applications, differing in architecture, supported languages, waiting behaviour, tooling and ecosystem.

Why compare the three?

Teams starting automation or considering a switch need a clear view of the trade-offs. Marketing pages tend to praise one tool, so use documentation and a pilot with your own application. This page lists differences that stay fairly stable, and tells you to check current release notes for details that change quickly.

How do they work internally?

Selenium uses the WebDriver standard: your code sends commands to a driver, which controls the browser. **Cypress** runs its test code inside the browser alongside the application, with a Node process helping. **Playwright** controls browsers through a protocol connection from an external process and creates isolated browser contexts. These designs explain many differences in speed, limits and debugging.

What does a side-by-side table show?

Aspect	Selenium	Cypress	Playwright
Languages	Java, Python, C#, JavaScript and more	JavaScript, TypeScript	JavaScript, TypeScript, Python, Java, .NET
Browsers	Many via drivers	Chromium family, Firefox, WebKit support varies	Chromium, Firefox, WebKit
Waiting	Explicit waits needed	Automatic retries	Auto-waiting on actions
Parallel runs	Grid or runner setup	Via paid or manual setups	Built-in workers and sharding
Debugging	IDE, logs	Time-travel runner	Trace viewer, inspector, UI mode
Mobile web emulation	Possible with setup	Viewport emulation	Device emulation built in
Real mobile apps	Via Appium	No	No (web only)
Check each project's current documentation, as features change between releases.			

When does Selenium fit best?

Choose it when your team uses Java or C# heavily, you have a large existing suite, you need a broad grid with many browser and OS combinations, or you pair it with Appium for mobile apps. Its ecosystem and community knowledge are very large.

When does Cypress fit best?

It fits front-end teams working in JavaScript who value an interactive runner and quick feedback while building. Check its limits on multiple tabs, cross-origin flows and browser coverage against your needs before choosing.

When does Playwright fit best?

It suits teams wanting one tool for several browsers, built-in parallelism, API testing, network mocking and rich debugging, and who can use TypeScript, Python, Java or .NET. It is a common pick for new projects.

How should I decide?

1. List your browsers, languages and special needs such as file downloads, multiple tabs or iframes.
2. Pick three representative flows and automate each in the candidate tools.
3. Compare readability, stability over repeated runs, run time and debugging.
4. Consider hiring, training and long-term maintenance.
5. Decide and write down why.

What about migration?

Migrating a large suite takes planning. Move critical flows first, keep both suites during the change, share test data and reuse page object ideas. Do not rewrite for fashion; migrate when the old tool causes measurable problems.

What misconceptions are common?

That one tool is always faster, that old tools are obsolete, or that a tool alone fixes flaky tests. Reliability comes from isolation, good locators and stable data. Learn concepts that transfer across tools.

How do I run a fair pilot?

Pick the same three flows for each tool, give every tool the same time box and the same tester skill level, and run each suite fifty times to see stability. Record the lines of code, the number of flaky runs, the time to write the first version and the time to debug a deliberate failure. Present the numbers in a table to the team and discuss them openly. A pilot based on your application is much more reliable than opinions from the internet.

What non-technical factors should I weigh?

Consider who will maintain the suite in two years, how easy it is to hire or train people for the chosen tool, how well it fits your current CI system, what your security team allows and the licence or support terms you need. A tool that fits your team culture and skills will be used well, while a tool chosen only for features may be abandoned. Write the decision and its reasons so that it can be revisited calmly when needs change.

What is one sentence summary for each tool?

Selenium is the broad and established standard, Cypress is a developer-friendly runner for JavaScript teams, and Playwright is a modern multi-language tool with strong built-in features. Verify each claim against current releases.

When should I revisit the decision?

Review it yearly or when the application, team or tool releases change significantly.

Frequently asked questions

Which is better, Playwright or Selenium?

Neither is best for all teams. Selenium has the widest ecosystem, while Playwright offers modern features, so decide with a pilot on your own application.

Does Cypress support multiple browsers?

Cypress supports several browsers, but coverage and limits differ from Selenium and Playwright, so check the current documentation for your needs.

Can Playwright test mobile apps?

Playwright tests web apps and can emulate mobile devices, but native mobile apps usually need a tool such as Appium.

Should I migrate my Selenium suite?

Only if it causes measurable problems. Migrate critical flows first and keep both suites running during the transition.

How do I choose a tool for my team?

List your languages, browsers and special needs, automate three real flows in each candidate and compare stability, readability and debugging.

Sources and further reading

- Playwright documentation
- Selenium documentation
- Cypress documentation

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550