Playwright enables reliable end-to-end testing for modern web apps.

Any browser • Any platform • One API

Cross-browser. Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox.

Cross-platform. Test on Windows, Linux, and macOS, locally or on CI, headless or headed.

Cross-language. Use the Playwright API in TypeScript, JavaScript, Python, .NET, Java.

Test Mobile Web. Native mobile emulation of Google Chrome for Android and Mobile Safari. The same rendering engine works on your Desktop and in the Cloud.

Playwright has impressed me by its simplicity and efficiency, and that's why I'm here with you today.

Playwright is a framework that enables reliable end-to-end testing for modern web applications.

It supports multiple languages

Resilient • No flaky tests

Auto-wait. Playwright waits for elements to be actionable prior to performing actions. It also has a rich set of introspection events. The combination of the two eliminates the need for artificial timeouts - the primary cause of flaky tests.

Web-first assertions. Playwright assertions are created specifically for the dynamic web. Checks are automatically retried until the necessary conditions are met.

Tracing. Configure test retry strategy, capture execution trace, videos, screenshots to eliminate flakes.

No trade-offs • No limits

Browsers run web content belonging to different origins in different processes. Playwright is aligned with the modern browsers architecture and runs tests out-of-process. This makes Playwright free of the typical in-process test runner limitations.

Multiple everything. Test scenarios that span multiple **tabs**, multiple **origins** and multiple **users**. Create scenarios with different contexts for different users and run them against your server, all in one test.

Trusted events. Hover elements, interact with dynamic controls, produce trusted events. Playwright uses real browser input pipeline indistinguishable from the real user.

Test frames, pierce Shadow DOM. Playwright selectors pierce shadow DOM and allow entering frames seamlessly.

Full isolation • Fast execution

Browser contexts. Playwright creates a browser context for each test. Browser context is equivalent to a brand new browser profile. This delivers full test isolation with zero overhead. Creating a new browser context only takes a handful of milliseconds.

Log in once. Save the authentication state of the context and reuse it in all the tests. This bypasses repetitive log-in operations in each test, yet delivers full isolation of independent tests.

Powerful Tooling

Codegen. Generate tests by recording your actions. Save them into any language.

Playwright inspector. Inspect page, generate selectors, step through the test execution, see click points, explore execution logs.

Trace Viewer. Capture all the information to investigate the test failure. Playwright trace contains test execution screencast, live DOM snapshots, action explorer, test source, and many more.

Installing Playwright

Get started by installing Playwright using npm or yarn. Alternatively you can also get started and run your tests using the VS Code Extension.

npm yarn pnpm

```
npm init playwright@latest
```

Playwright will download the browsers needed as well as create the following files.

```
playwright.config.ts
package.json
package-lock.json
tests/
  example.spec.ts
tests-examples/
  demo-todo-app.spec.ts
```