



# Installation

## Introduction

Playwright Test was created specifically to accommodate the needs of end-to-end testing. Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox. Test on Windows, Linux, and macOS, locally or on CI, headless or headed with native mobile emulation of Google Chrome for Android and Mobile Safari.

### You will learn

- [How to install Playwright](#)
- [What's Installed](#)
- [How to run the example test](#)
- [How to open the HTML test report](#)

## 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
```

Run the install command and select the following to get started:

- Choose between TypeScript or JavaScript (default is TypeScript)
- Name of your Tests folder (default is tests or e2e if you already have a tests folder in your project)
- Add a GitHub Actions workflow to easily run tests on CI
- Install Playwright browsers (default is true)

# What's Installed

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
```

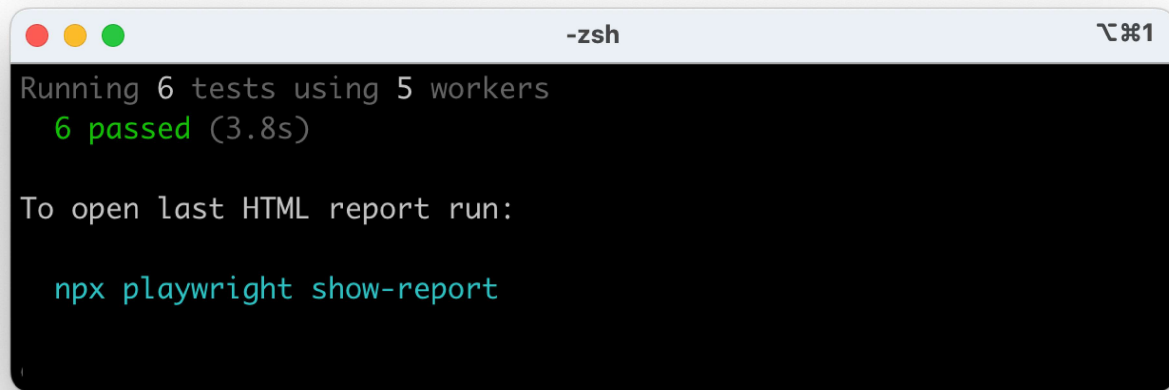
The `playwright.config` is where you can add configuration for Playwright including modifying which browsers you would like to run Playwright on. If you are running tests inside an already existing project then dependencies will be added directly to your `package.json`.

The `tests` folder contains a basic example test to help you get started with testing. For a more detailed example check out the `tests-examples` folder which contains tests written to test a todo app.

## Running the Example Test

By default tests will be run on all 3 browsers, chromium, firefox and webkit using 3 workers. This can be configured in the `playwright.config` file. Tests are run in headless mode meaning no browser will open up when running the tests. Results of the tests and test logs will be shown in the terminal.

```
npx playwright test
```

A terminal window with a light blue title bar containing three colored window control buttons (red, yellow, green) on the left, the text '-zsh' in the center, and a window icon on the right. The terminal has a black background with white text. It displays the output of a Playwright test run: 'Running 6 tests using 5 workers', '6 passed (3.8s)', 'To open last HTML report run:', and 'npx playwright show-report'.

```
Running 6 tests using 5 workers
6 passed (3.8s)

To open last HTML report run:

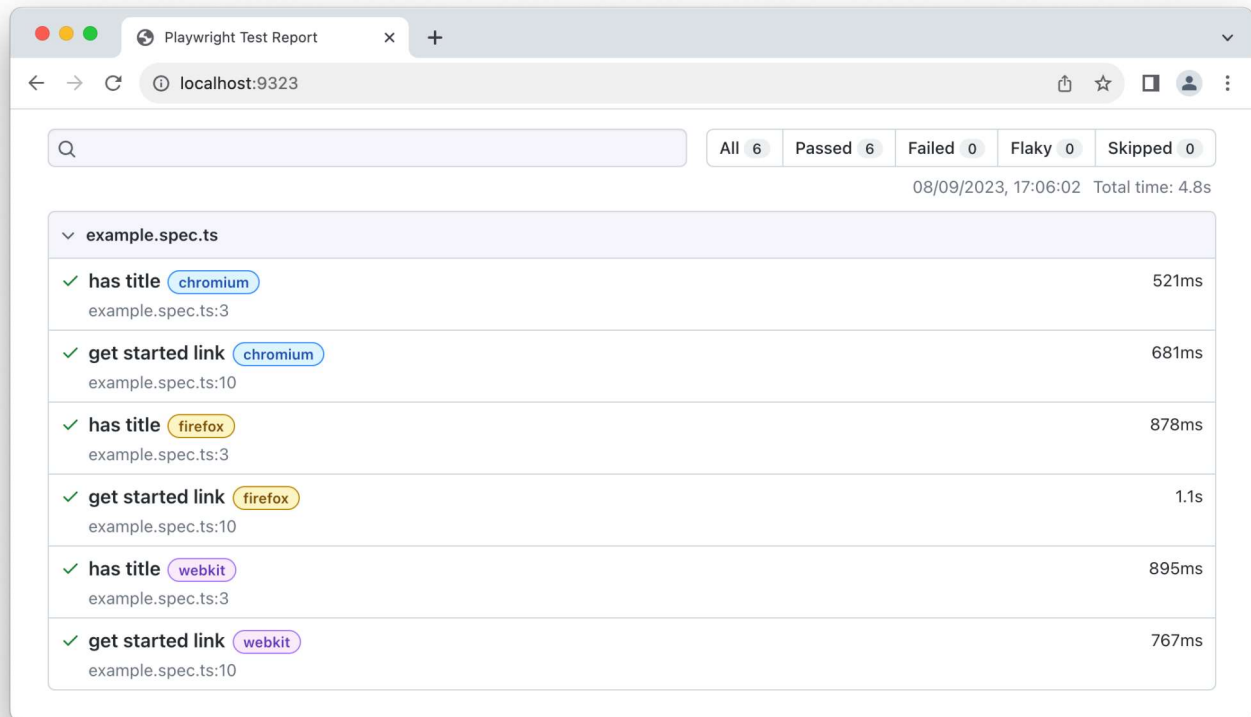
npx playwright show-report
```

See our doc on [Running Tests](#) to learn more about running tests in headed mode, running multiple tests, running specific tests etc.

## HTML Test Reports

After your test completes, an [HTML Reporter](#) will be generated, which shows you a full report of your tests allowing you to filter the report by browsers, passed tests, failed tests, skipped tests and flaky tests. You can click on each test and explore the test's errors as well as each step of the test. By default, the HTML report is opened automatically if some of the tests failed.

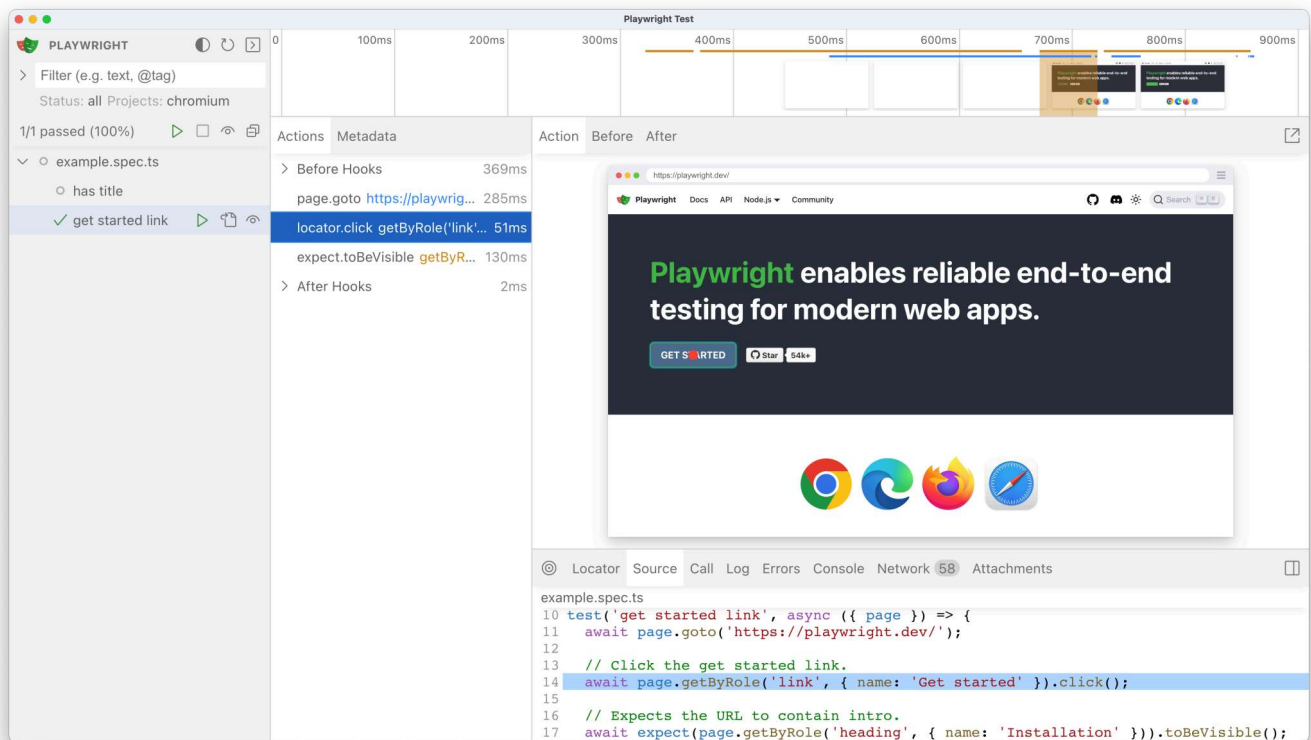
```
npx playwright show-report
```



## Running the Example Test in UI Mode

Run your tests with **UI Mode** for a better developer experience with time travel debugging, watch mode and more.

```
npx playwright test --ui
```



Check out or [detailed guide on UI Mode](#) to learn more about its features.

## Updating Playwright

To update Playwright to the latest version run the following command:

```
npm install -D @playwright/test@latest
# Also download new browser binaries and their dependencies:
npx playwright install --with-deps
```

You can always check which version of Playwright you have by running the following command:

```
npx playwright --version
```

## System requirements

- Node.js 18+
- Windows 10+, Windows Server 2016+ or Windows Subsystem for Linux (WSL).
- MacOS 12 Monterey, MacOS 13 Ventura, or MacOS 14 Sonoma.
- Debian 11, Debian 12, Ubuntu 20.04 or Ubuntu 22.04, with x86-64 or arm64 architecture.

# What's next

- Write tests using web first assertions, page fixtures and locators
- Run single test, multiple tests, headed mode
- Generate tests with Codegen
- See a trace of your tests