Playwright Test Runner Configuration PDF

The playwright.config file controls how Playwright Test runs: which folder holds tests, timeouts, retries, parallel workers, reporters, the base URL, browser projects and an optional web server that starts your app. Set shared options under use, define projects for each browser or device, and keep environment values outside the file.

Key takeaways

- One config file controls timeouts, retries, workers, reporters and projects.
- The use block sets shared options such as baseURL, trace and screenshot.
- Projects run the same tests across browsers, devices or environments.
- webServer can start your app before tests and stop it afterwards.
- Read secrets and URLs from environment variables, not from the file.

What is Playwright configuration?

The Playwright configuration file is a TypeScript or JavaScript module that exports settings for the test runner, including test location, timeouts, retries, reporters, browser projects and shared browser options.

What does the config file do?

`playwright.config.ts` is read when you run `npx playwright test`. It tells the runner where tests live, how long to wait, how many to run at once, how to report and which browsers to use. Having a single place for these choices keeps tests short and makes behaviour the same on every machine.

What does a typical file look like?

```
import { defineConfig, devices } from '@playwright/test';

export default defineConfig({
  testDir: './tests',
  timeout: 30000,
  retries: process.env.CI ? 2 : 0,
  fullyParallel: true,
  reporter: [['html'], ['list']],
  use: {
    baseURL: process.env.BASE_URL ?? 'http://localhost:3000',
    trace: 'on-first-retry',
    screenshot: 'only-on-failure',
  },
  projects: [
    { name: 'chromium', use: { ...devices['Desktop Chrome'] } },
    { name: 'firefox', use: { ...devices['Desktop Firefox'] } },
    { name: 'mobile', use: { ...devices['Pixel 7'] } },
  ],
});
```

Which top-level options matter most?

Option	Purpose
`testDir`	Folder holding test files
`timeout`	Maximum time for one test
`expect.timeout`	Wait time for assertions
`retries`	Re-run failed tests
`workers`	Number of parallel worker processes
`fullyParallel`	Run tests in one file in parallel too
`reporter`	Output format such as list, html, json, junit
`forbidOnly`	Fail if test.only is left in code

Use `forbidOnly: !!process.env.CI` so an accidental `.only` cannot silently skip tests on the build server.

What goes inside the use block?

Options here apply to every test: `baseURL`, `viewport`, `locale`, `timezoneId`, `storageState`, `extraHTTPHeaders`, `ignoreHTTPSErrors`, `trace`, `screenshot` and `video`. Setting `baseURL` lets tests call `page.goto('/login')`. Setting `trace: 'on-first-retry'` records a trace only when a retry happens, which keeps storage small.

What are projects?

A project is a named configuration. Use projects to run the same tests in Chromium, Firefox and WebKit, on desktop and mobile sizes, or against different environments. Projects can depend on others, for example a `setup` project that logs in and saves state before the browser projects start. Select a project from the command line with `--project=firefox`.

How do I start the app automatically?

```
webServer: {
  command: 'npm run start',
  url: 'http://localhost:3000',
  reuseExistingServer: !process.env.CI,
},
```

Playwright launches the command, waits until the URL responds, runs tests and stops the server. This removes manual steps from local runs and CI.

How should I choose retries and workers?

Retries are a safety net for rare infrastructure noise, not a way to hide flaky tests. Use zero locally so failures are visible and one or two on CI while you track tests that needed retries. Workers default to a share of CPU cores; reduce them if the application under test cannot handle the load.

How do I handle environments and secrets?

Read values with `process.env`, load a `.env` file with a small loader if needed, and keep real credentials in the CI secret store. Create separate config files or projects for staging and local runs. Never commit passwords or tokens.

What mistakes appear often?

Very high timeouts that hide slowness, retries that mask flakiness, missing `forbidOnly`, traces always on in large suites and one giant project for everything. Review the config with the team the way you review application code.

How do I organise config for several environments?

Keep one shared base config and extend it for each environment, or use projects whose `use` blocks point to different base URLs. Pass the environment name through a variable such as TEST_ENV and choose settings from a small map. This avoids copying files and keeps differences visible in one place, which makes reviews simpler.

What should I check when the config seems ignored?

Confirm that you run the command from the folder containing the config, that the file name is correct, that the project filter is not excluding tests and that command line flags are not overriding

the file. Run `npx playwright test --list` to see which tests the runner finds, and print `process.env` values inside the config temporarily to confirm they arrive as expected.

What is a good review checklist for the config?

Check that forbidOnly is on for CI, that retries are small, that trace and screenshot options save only on failure, that the base URL comes from an environment variable, that projects match the browsers you support and that the reporter outputs a format your pipeline can read.

How do I practise reading and tuning the config?

Create a small sample project, change one option at a time, such as workers, retries or reporter, and observe how the run output changes. Keeping notes on each experiment builds a clear mental model of what every setting does.

Frequently asked questions

What is playwright.config.ts?

It is the configuration file that sets test location, timeouts, retries, reporters, browsers and shared options for the Playwright test runner.

What is a Playwright project?

A project is a named set of options, often used to run the same tests across browsers, devices or environments.

How do I set a base URL?

Set baseURL in the use block, then call page.goto with relative paths such as /login so environments can change without edits.

When should I use retries?

Use retries mainly on CI as a safety net, and track retried tests so real flakiness is fixed rather than hidden.

Can Playwright start my app before tests?

Yes. The webServer option runs a command, waits for a URL to respond and stops the server when the run ends.

Sources and further reading

- Playwright documentation
- Playwright API reference
- Playwright test configuration

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