UnCodemy

Playwright Trace Viewer and Debugging PDF

Playwright offers several debugging tools: the trace viewer replays a failed run with snapshots, actions, network calls and console logs; the inspector pauses and steps through code; UI mode lets you watch tests and time-travel; and codegen records actions into code. Record traces on first retry in CI and open them locally.

Key takeaways

- Set trace to on-first-retry so CI keeps traces only when needed.
- Open a trace with npx playwright show-trace or from the HTML report.
- UI mode and the inspector help when writing and fixing tests locally.
- Screenshots and video are lighter alternatives for simple failures.
- Start debugging from the first failed action, not from the last error line.

What is Playwright debugging?

The Playwright trace viewer is a tool that opens a recorded trace file and shows each test action with page snapshots, network requests, console messages and source lines so that failures can be investigated after the run.

Why do I need dedicated debugging tools?

UI tests fail for many reasons: a wrong locator, a slow request, a changed element, bad data or a real bug. The error message names the failing line, but not the page state at that time. Playwright records enough to answer "what did the page look like, what was requested and what did the console say?" without rerunning the test again and again.

How do I record a trace?

In the config, set `trace: 'on-first-retry'` under `use`. Other values are `on`, `off`, `retain-on-failure` and `on-all-retries`. From the command line, `npx playwright test --trace on` records every test. Because traces are zip files with snapshots, keep them for failures only in large suites to save disk and upload time.

How do I open and read a trace?

Run `npx playwright show-trace trace.zip` or click the trace link in the HTML report. The viewer shows a timeline of actions, a snapshot of the page before, during and after each action, the locator

used, network requests with responses, console output and the source code line. Click an action to see the page exactly at that point and check whether the target element was present and visible.

What should I look for first?

Question	Where to look
Did the element exist?	Snapshot before the action
Was the locator right?	Locator highlight and action details
Did a request fail?	Network tab
Was there a script error?	Console tab
Was text different?	After snapshot and assertion error
Was it too slow?	Timeline durations
Go to the first failed action; later failures are often side effects.	

How does the inspector help?

Run `npx playwright test --debug` or set `PWDEBUG=1`. A browser and an inspector window open, and the test pauses at the first action. Step through actions, test locators in the picker and watch the log. Add `await page.pause()` in code to stop at an exact spot while you experiment.

What is UI mode?

Start `npx playwright test --ui` to open a window listing tests, running them on demand, showing watch mode, filters, a timeline and the trace for each run. You can pick a locator from the page, see snapshots for every step and rerun on file save. It is excellent while developing new tests.

How do I use codegen safely?

`npx playwright codegen https://example.com` opens a browser and writes code as you click. Use it to discover locators and structure, then clean the output: replace fragile selectors with role-based ones, add assertions and extract page objects. Do not store credentials in generated code.

How do screenshots, video and logs fit in?

Set `screenshot: 'only-on-failure'` and `video: 'retain-on-failure'` for quick evidence. Attach custom logs with `testInfo.attach`. For the browser console, listen with `page.on('console', ...)` while debugging. Use these lighter tools for obvious failures and open the trace for the hard ones.

What debugging habits help teams?

Reproduce locally with the same browser project, run only the failing test with `-g` or the file name, change one variable at a time, and record the root cause in the bug or pull request. Share trace files with developers when the issue is in the application and not in the test.

How do I write a good bug report from a trace?

Include the test name, the browser project, the first failing action, a short sentence on expected and actual behaviour, and the trace file or a screenshot of the failing snapshot. Add the request and response details if the network tab shows an error. A report with this evidence lets a developer reproduce the issue in minutes, and it separates application defects from test defects so each goes to the right owner.

What are the limits of traces?

Traces show what Playwright can observe in the browser. They do not capture problems in external systems unless they surface as network errors, and large traces on long tests can be heavy to open. Keep tests short and focused, record traces only when needed and combine the trace with server logs when the cause appears to be on the backend. Being aware of these limits stops you from expecting one tool to explain every failure.

Frequently asked questions

What is the Playwright trace viewer?

It is a tool that replays a recorded test with page snapshots, actions, network calls and console logs so you can investigate failures.

How do I enable traces?

Set trace to on-first-retry in the use block of the config, or pass the trace option on the command line when running tests.

What is UI mode?

UI mode is a graphical runner that lists tests, shows timelines and snapshots, supports watch mode and helps pick locators while you write tests.

How do I pause a test at a specific line?

Add await page.pause() in the test, or run with the debug flag, then step through actions in the inspector.

Is codegen output ready for production?

No. Treat it as a starting draft, then improve locators, add assertions and structure the code before adding it to the suite.

Sources and further reading

- Playwright documentation
- Playwright API reference
- Playwright trace viewer

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