the questions that actually get asked ★

50 Playwright Interview Questions

With real answers, grouped by round.
From fundamentals to framework design.

7 sections. 50 answers. Free.

Read this first

Sixty seconds, then the questions.

Answer in your own words. These answers are the shape of a strong response, not a script. Memorised phrasing is obvious to anyone who has run interviews.

Say the tradeoff out loud. Almost every question here has one. Naming it is what separates a mid answer from a senior one.

Bring one real story per section. A suite you sped up, a flake you fixed, a locator strategy you changed your mind about. Stories outrank definitions.

Know your own config. Panels love asking about the setup you actually run. If you have never opened playwright.config, open it before your next interview.

Practice out loud. Reading is revision. Saying it is preparation. The gap between the two is where interviews are lost.

Fundamentals

The warm-up. Get these wrong and the round ends early.

01 What is Playwright and what problem does it solve?

A browser automation framework from Microsoft that drives Chromium, Firefox, and WebKit through one API. It solves the flakiness and setup cost of older tools by bundling auto-waiting, browser binaries, and parallelism by default.

02 How does Playwright differ from Selenium?

Playwright talks to the browser over a persistent connection instead of the WebDriver protocol, auto-waits for elements, ships its own browsers, and runs parallel out of the box. Selenium has a wider language and grid ecosystem and far more legacy adoption.

03 What is a browser context and why does it matter?

An isolated session inside one browser instance, with its own cookies and storage. Contexts are cheap, so each test gets a clean slate without paying to relaunch the browser.

04 Browser, context, and page. How do they relate?

A browser holds many contexts, and a context holds many pages. Browser is the process, context is the incognito-style session, page is a single tab.

05 Are tests parallel by default?

Yes, across worker processes. Tests in a single file run in order by default, and files spread across workers. Control it with the workers option or test.describe.configure.

06 Which languages does Playwright support?

JavaScript and TypeScript, Python, Java, and .NET. The test runner with fixtures and UI mode is richest in the JS/TS version.

07 What is in playwright.config?

Test directory, timeouts, retries, reporters, the projects array for browsers and devices, and the use block for baseURL, tracing, screenshots, and video.

Locators

Where most candidates reveal how much they have actually written.

08 What is a locator and how is it different from a selector string?

A locator is a lazy description of how to find an element. It resolves at the moment you act on it, so it survives re-renders that would break a stale element handle.

09 Which locators should you reach for first?

User-facing ones. getByRole, then getByLabel, getByPlaceholder, getByText. They mirror how a person or screen reader finds the element, so they break less when markup changes.

10 Why is getByRole usually the best default?

It matches the accessibility tree rather than the DOM structure, so it stays valid through styling and layout changes, and it quietly tests accessibility at the same time.

11 When is getByTestId the right call?

When nothing user-facing identifies the element uniquely. It is stable and explicit, but it tests an attribute the user never sees, so it is a second choice, not a first.

12 What is strict mode?

If a locator matches more than one element, Playwright throws instead of silently picking the first. It turns an ambiguous test into a loud failure.

13 How do you narrow a locator that matches several elements?

Chain and filter. Scope to a parent locator, use filter with hasText or has, or use first, last, and nth when position genuinely is the identity.

14 How do you handle elements inside an iframe?

Use frameLocator to scope into the frame, then locate normally inside it.

15 When is XPath acceptable?

As a last resort, usually for legacy markup with no roles, labels, or test ids. It couples the test to DOM structure, which is exactly what you want to avoid.

Assertions & Waiting

The section that separates people who fight flake from people who cause it.

16 What are web-first assertions?

Assertions that retry until they pass or time out. expect(locator).toBeVisible() keeps re-checking, so you do not need a manual wait before it.

17 Why should you avoid waitForTimeout?

It is a fixed sleep. It slows the suite when things are fast and still fails when things are slow. Wait for a state, not a duration.

18 What are actionability checks?

Before acting, Playwright waits for the element to be attached, visible, stable, able to receive events, and enabled. That is why explicit waits are rarely needed.

19 Difference between toHaveText and toContainText?

toHaveText matches the full text, toContainText matches a substring. Reaching for toContainText to make a failure go away is usually hiding a real change.

20 How do you assert a list has finished loading?

toHaveCount on the list locator. It retries, so it doubles as the wait.

21 What does expect.soft do?

Records a failure but lets the test continue, so one run reports several problems. The test still fails at the end.

22 When would you use expect.toPass?

To retry a block of arbitrary code until it stops throwing. Useful for eventual consistency where no single assertion covers the condition.

Test Structure & Fixtures

Framework design questions. Senior signal lives here.

23 What are fixtures?

Playwright's dependency injection. Built-ins like page and request are fixtures, and you can define your own for setup and teardown that runs only for tests that ask for it.

24 Fixtures or beforeEach. Which and why?

Fixtures for anything reusable or stateful, because they are scoped, composable, and only run when requested. beforeEach for simple per-file setup.

25 What is the difference between test-scoped and worker-scoped fixtures?

Test scope runs per test. Worker scope runs once per worker process and is shared by the tests it runs, which suits expensive setup like seeding a database.

26 How do you avoid logging in inside every test?

Authenticate once in a setup project, save storageState to a file, then load that state in the use block. Tests start already signed in.

27 How do you keep tests independent?

No shared mutable data, no ordering assumptions, unique test data per run, and cleanup by the same test that created it. If tests must run in order, they are not independent.

28 What does test.describe.serial do, and what is the risk?

Runs tests in a group in order and skips the rest after a failure. It is a deliberate escape hatch, and it forfeits isolation, so use it rarely.

29 How do you tag and run a subset of tests?

Tag with the tag option or in the title, then filter with grep. A smoke tag that runs on every push is the usual first use.

API & Network

Underrated. Most candidates have not gone past the UI.

30 Can Playwright test APIs directly?

Yes. The request fixture makes HTTP calls with the same cookies and auth as the browser context, so you can set up state or assert backend behavior without the UI.

31 Why set up state through the API instead of the UI?

It is faster and far more stable. Driving five screens to reach the screen you actually want to test adds five ways to fail.

32 How do you mock a network response?

page.route to intercept the request, then fulfill it with your own status and body. Good for error states you cannot trigger on demand.

33 How do you block third-party requests?

Route the pattern and abort it. Cuts noise and speeds up runs, especially for analytics and ad calls.

34 How do you assert on an API response?

Status code, schema shape, the values that matter, and relevant headers. Checking only the status is not API testing.

35 How would you test a file download?

Wait for the download event, then inspect the suggested filename and the saved file. Same idea for uploads with setInputFiles.

Debugging & CI

Practical questions. Vague answers here are obvious.

36 How do you debug a failing test locally?

UI mode for a visual walkthrough, debug mode for the inspector, or page.pause to stop and poke at locators live.

37 What is the trace viewer?

A recorded timeline of the run with DOM snapshots, network, console, and source. Configure traces on first retry and it becomes the main tool for CI failures.

38 A test passes locally and fails in CI. Where do you look?

Timing under load, test data, parallelism collisions, environment differences, and headless rendering. Then open the trace instead of guessing.

39 How do you speed up a slow suite?

More workers, shard across CI machines, push checks down to API and unit layers, cut duplicated coverage, and reuse auth state.

40 What is sharding?

Splitting the suite across machines with the shard flag, then combining the blob reports with merge-reports into one HTML report.

41 Are retries a good idea?

As a safety net for genuine infrastructure noise, yes. As a habit, no. A test that only passes on retry did not pass, and retries hide the failure that would have told you why.

42 What do you capture on failure in CI?

Trace, screenshot, and video, uploaded as artifacts. Without them a red pipeline is just a guess.

Architecture & Strategy

The closing round. These decide senior versus mid.

43 Explain the Page Object Model and one weakness.

It puts page structure and actions behind a class so tests read as intent. The weakness is that page objects grow into god classes that hide logic and get reused badly.

44 What belongs in a page object and what does not?

Locators and small actions belong. Assertions usually do not, because the test should say what it expects. Business logic definitely does not.

45 How do you decide what to automate in Playwright?

Stable, repetitive, high-value paths. Leave exploratory work, constantly changing UI, and anything needing judgment to a human.

46 How do you manage test data?

Generate unique data per run, seed through the API, clean up after, and never depend on a shared record another test can mutate.

47 How do you handle multiple environments?

baseURL per environment through config and env vars, with no hardcoded URLs in tests. Projects can split by browser, environment, or both.

48 Should you use codegen?

As a starting draft, yes. It is a fast way to discover locators. Ship it unedited and you get brittle tests with no structure.

49 How do you know your suite is healthy?

Runtime, pass rate, flake rate, and whether people trust a red build. If the team reruns failures out of habit, the suite is already dead.

50 What would you change first on a suite you inherited?

Measure flake and runtime before touching anything, then fix the loudest source of untrustworthiness. Rewriting before measuring is how people waste a quarter.

One thing before your next interview.

Nobody has ever lost an offer for saying "I have not used that, but here is how I would find out."

People lose offers for bluffing, because the follow-up question always comes, and the panel can tell within about nine seconds.

Know the tool well. Be honest about the edges. That combination is rarer than you think.

Good luck. Go get it. ★

Aston Cook · Senior QA Automation Engineer · AssertHired