25 Playwright & Cypress Interview Questions for Automation Engineers

Doc by Aston Cook

This guide combines practical questions and answers on Playwright and Cypress. Perfect for QA and SDET interviews focusing on modern test automation frameworks.

---

1. What is Playwright and how does it differ from Cypress?

Answer:

Playwright is a Node.js library for browser automation that supports multiple browsers and programming languages. Cypress is an end-to-end testing framework focused on JavaScript and Chromium-based browsers. Playwright supports multi-browser testing and native mobile emulation, while Cypress is known for its fast test execution and developer-friendly API.

---

2. Why would you choose Playwright over Selenium?

Answer:

Playwright offers automatic waits, parallel test execution, and supports multiple browsers and languages. It is often faster and requires less configuration for modern apps compared to Selenium.

---

3. How does Cypress handle waits differently from Playwright?

Answer:

Cypress automatically retries commands until elements are ready, removing the need for explicit waits. Playwright also has auto-waiting but allows more granular control with explicit waits when needed.

---

4. What languages do Playwright and Cypress support?

Answer:

Playwright supports JavaScript, TypeScript, Python, Java, and .NET. Cypress primarily supports JavaScript and TypeScript.

---

5. How do you structure tests in Playwright?

Answer:

Use Page Object Model (POM) for reusable locators and methods. Organize specs by feature and use fixtures for setup and teardown.

---

6. How do you assert element visibility in Cypress?

Answer:

Use cy.get('selector').should('be.visible').

---

7. Can Playwright handle multiple tabs or browsers?

Answer:

Yes, Playwright supports multi-context and multi-browser testing natively.

---

8. What is a fixture in Playwright?

Answer:

Fixtures are reusable components that set up state before tests, such as creating test users or initializing data.

---

9. How do you mock API responses in Cypress?

Answer:

Use cy.intercept() to stub API calls and provide mocked responses.

---

10. What is headless mode in Playwright and Cypress?

Answer:

Headless mode runs browsers without a GUI. It is faster and often used in CI pipelines.

---

11. How do you perform API testing with Playwright?

Answer:

Use Playwright’s request API to send HTTP requests and validate responses within tests.

---

12. How do you handle authentication in Cypress?

Answer:

Log in via UI once and save session cookies, or directly set cookies or localStorage to bypass login in tests.

---

13. How do you retry failing tests in Playwright?

Answer:

Configure retries in the Playwright config file with retries: 2 for flaky tests.

---

14. What are Cypress custom commands?

Answer:

Custom commands are reusable methods defined with Cypress.Commands.add() for common actions like login.

---

15. How do you capture screenshots on failure?

Answer:

Both tools automatically capture screenshots and videos if configured. Playwright: screenshot: 'only-on-failure'.

---

16. What are some limitations of Cypress?

Answer:

Limited cross-browser support, no native multi-tab testing, and reliance on JavaScript.

---

17. What is auto-waiting in Playwright?

Answer:

Playwright automatically waits for elements to be ready before interacting, reducing flaky tests.

---

18. How do you run Cypress tests in parallel?

Answer:

Use Cypress Dashboard Service or CI configuration to distribute tests across multiple machines.

---

19. How do you use Playwright test annotations like skip or only?

Answer:

Use test.skip() to skip a test and test.only() to run a specific test.

---

20. What is the difference between .should() in Cypress and expect() in Playwright?

Answer:

.should() is chainable and retries until assertion passes in Cypress. expect() in Playwright works with locators and uses built-in assertions.

---

21. How do you parameterize tests in Playwright?

Answer:

Use test.describe() blocks and pass data arrays to iterate over scenarios.

---

22. How do you handle environment variables in Cypress?

Answer:

Define variables in cypress.env.json or pass them with CLI using --env.

---

23. How do you debug tests?

Answer:

In Playwright: run tests with npx playwright test --debug. In Cypress: use .debug() or the interactive Test Runner.

---

24. How do you run tests across multiple browsers?

Answer:

Playwright supports Chromium, Firefox, and WebKit natively. Cypress supports Chrome and Edge, with limited Firefox support.

---

25. How do you integrate these tools into CI/CD?

Answer:

Add Playwright or Cypress commands in CI pipelines like GitHub Actions or Jenkins. Install dependencies and run tests headless.