Created by: Lamhot Siagian

Top 111 Playwright Interview Questions & Hints

12 Cards | Questions + Hints | Playwright Test

1. Playwright Architecture — 10 Qs

- **Multi-language support?** Node.js, Python, Java, .NET bindings share same driver protocol.
- **Browsers supported?** Chromium, WebKit, Firefox; bundled channel management.
- **Browser vs BrowserContext vs Page.** Browser is engine; contexts = isolated sessions; pages = tabs.
- **Why contexts matter?** Parallel isolation, cookie/storage separation, faster than new browser.
- **Headless vs headed?** Headless for CI speed; headed for local debugging and visuals.
- **Test runner vs library?** Playwright Test adds fixtures, reporters, retries; library is core API.
- **Workers/parallelism model?** Test files shard across workers; contexts/pages per test.
- **GPU/CPU considerations.** Video/trace slowdowns; CI containers may need flags.
- **Web-first assertions?** Auto-waiting baked into locator & expect APIs.
- **When to prefer Playwright vs Selenium?** Built-in waits, tracing, cross-browser reliability; trade-offs in ecosystem/tooling.

2. Locators & Selectors — 10 Qs

- **Locator API advantage?** Lazy resolution + auto-waiting + retries; chainable filters.
- **Role & text locators.** `getByRole()`, `getByText()` improve stability & a11y alignment.
- **CSS vs XPath.** Prefer CSS/role; XPath only when structure demands.
- **:has(), :has-text(), :nth-match().** Powerful filters for relations and occurrences.
- **Combining filters.** `locator('.row').filter({ hasText:'Paid' }).nth(0)`
- **Waiting for visibility.** `await locator.waitFor()` or `expect(locator).toBeVisible()`.
- **File upload/download.** `setInputFiles`, `page.on('download')` with suggested path.
- **Shadow DOM & frames.** Use locators that pierce shadow & `frameLocator()` for iframes.
- **Avoid brittle selectors.** Prefer test IDs (`getByTestId`) over CSS classes.
- **Stable patterns.** Use semantic roles, labels, accessible names.

3. Auto-Waiting & Synchronization — 9 Qs

- **What is auto-waiting?** Playwright waits for actionable state (attached, visible, stable).
- **Recommended waits.** `locator.waitFor()`, `page.waitForURL()`, `expect` with assertions.
- **Avoid false waits.** No `Thread.sleep()`; use web-first patterns.
- **Network idle caveats.** SPAs may keep sockets open; prefer UI state waits.
- **Handling animations.** Wait for `toBeVisible` / `toBeHidden` or disable animations.
- **Race conditions.** Use `Promise.all([page.waitForNavigation(), click])` pattern.
- **Request finished.** `page.waitForResponse()` with predicate for specific calls.
- **Timeout strategy.** Tune default timeout and per-action timeouts.
- **Retries vs waits.** Prefer deterministic waits; retries for flake control.

4. Authentication & Context Reuse — 9 Qs

- **Login once, reuse?** Persist with `storageState`; load in `use.storageState` or new context.
- **Where to save state?** CI-safe path (e.g., `auth/state.json`) generated by a setup project.
- **Protect secrets.** Use env vars/CI secrets; avoid committing tokens.
- **Multi-user roles.** Generate different states (admin, user, guest) per worker/fixture.
- **SSO/MFA flows.** Prefer API sign-in or test-only backdoors; otherwise manual seed.
- **Context vs page reuse.** Reuse contexts per test file; create fresh page per test.
- **Test data coupling.** Decouple login from app data to reduce flakiness.
- **Cross-domain cookies.** Storage state scoped per domain; sign-in per origin if needed.
- **Logout testing.** Clear storage or new context to validate auth boundaries.

5. Fixtures & Test Hooks — 9 Qs

- **What are fixtures?** Reusable setup like `page`, `browser`, `context`, custom API clients.
- **Per-test isolation.** Each test gets a fresh page; avoid cross-test state.
- **Project/expect config.** Define projects (browsers/devices) and `expect` options in config.
- **Hooks order.** `beforeAll`, `beforeEach`, `afterEach`, `afterAll` lifecycles.
- **Global setup/teardown.** Use `globalSetup` for auth seeding; `globalTeardown` for cleanup.
- **Extending fixtures.** Create `test.extend<>()` for domain-specific helpers.
- **Tagging tests.** Use `@smoke`, `@regression` via `test.describe` or CLI filtering.
- **Serial vs parallel.** Use `test.describe.serial` for dependent flows only.
- **Resource cleanup.** Dispose handles, close servers to prevent leaks.

6. Configuration & Parallel Execution — 9 Qs

- **Key config fields?** `projects`, `use`, `reporter`, `retries`, `timeout`, `fullyParallel`.
- **Sharding/CI speedups.** Split tests across CI nodes; cache browsers; `--max-workers`.
- **Retries policy.** Set `retries` with `forbidOnly` in CI; annotate flaky tests.
- **Trace/video/screenshot mode.** Configure in `use` (e.g., `trace:'retain-on-failure'`).
- **Base URL usage.** Use `use.baseURL` and `page.goto('/path')`.
- **Timeout strategy.** Global vs action vs expect timeouts; avoid overly long defaults.
- **Reporters.** html, list, json, junit, allure; combine multiple reporters.
- **Device emulation.** Define device in project via `devices['iPhone 13']`.
- **Fail-fast and grep.** `--max-failures`, `--grep` / `--grep-invert` to scope runs.

7. Assertions & Expect API — 9 Qs

- **Web-first assertions?** `expect(locator).toBeVisible()`, `toHaveText`, `toHaveURL` wait implicitly.
- **Soft vs hard expects.** Use `expect.soft` to continue after failures.
- **Polling options.** Custom polling for async conditions via `expect` options.
- **Assertions on APIResponse.** `expect(res.ok()).toBeTruthy()`, status, JSON body.
- **Locator count & state.** `toHaveCount`, `toBeEnabled`, `toBeEditable`.
- **Screenshots in expect.** `toHaveScreenshot` for visual regression (opt-in).
- **Timeout overrides.** Per-assert timeout via options.
- **Custom matchers.** Extend `expect` for domain-specific checks.
- **Flake diagnosis.** Review trace/video and annotate unstable assertions.

8. API Testing with Playwright — 9 Qs

- **When to use request context?** Create `request.newContext()` for authenticated API calls.
- **Mix UI + API.** Seed data via API, verify via UI for faster, stable flows.
- **Handle auth tokens.** Inject headers in request context; reuse across tests.
- **Response assertions.** Status, headers, JSON schema; negative cases.
- **File uploads via API.** Multipart `form` helpers; validate content-type.
- **Rate limiting tests.** Burst/cooldown patterns; expect 429 with Retry-After.
- **Contract checks.** Validate response shape before UI depends on it.
- **API fixtures.** Provide reusable `apiClient` via extended fixtures.
- **Mock external APIs.** Use `page.route` to isolate third-party dependencies.

9. Screenshots, Videos & Traces — 9 Qs

- **Screenshot types.** Full-page, element, clip region; mask sensitive areas.
- **Video capture.** Enable in `use`; store only on failure to save space.
- **Trace viewer.** `npx playwright show-trace trace.zip` for time travel debug.
- **Artifacts retention.** Configure per CI job; upload as artifacts for PRs.
- **Console/network logs.** Listen to `page.on('console')` and `'requestfailed'`.
- **Visual comparisons.** Baseline images; diff thresholds; per-device snapshots.
- **Debug mode.** `PWDEBUG=1`, `slowMo`, headed mode to step through actions.
- **Failure triage.** Check trace → network → console to identify root cause.
- **PII handling.** Mask fields; avoid uploading sensitive screenshots/logs.

10. Cross-Browser & Device Testing — 9 Qs

- **Why test WebKit?** iOS/Safari parity; different rendering/quirks.
- **Firefox nuances.** Input handling and focus differences; validate critical flows.
- **Device descriptors.** Use `devices` for DPR, viewport, user agent.
- **Geo & locale.** Override `geolocation`, `locale`, `timezoneId`.
- **Permissions.** Grant `camera`, `clipboard-read`, etc., per context.
- **Network conditions.** Throttle (proxy/devtools) to simulate 3G/latency.
- **Feature detection vs UA.** Prefer capability checks over UA sniffing.
- **Responsive checks.** Test critical breakpoints; use per-project devices.
- **OS-specific flake.** Font rendering & timers differ; verify visuals per OS.

11. Network Interception & Mocking — 9 Qs

- **page.route basics.** Intercept by URL pattern; `fulfill` / `abort` / `continue`.
- **Selective mocking.** Mock only unstable endpoints; let static assets pass through.
- **Recording fixtures.** Record/serialize responses for deterministic tests.
- **GraphQL interception.** Match by operation name in POST body.
- **Latency/fault injection.** Simulate 5xx, timeouts to test resilience paths.
- **Bypass auth domains.** Don't mock IdP; prefer test doubles for third parties.
- **Service worker caveats.** SW caching may mask requests; clear context between tests.
- **Data seeding.** Return canned JSON matching real contracts.
- **Contract drift alerts.** Fail tests when provider shape changes unexpectedly.

12. CI/CD Integration — 10 Qs

- **Headless runs.** `npx playwright test --reporter=html` in CI for artifacts.
- **Jenkins/GitHub/Azure examples.** Use official action or CI containers with browser deps cached.
- **Caching browsers.** Cache `~/.cache/ms-playwright` to speed CI.
- **Matrix builds.** Split by OS, browser, device projects.
- **Sharding tests.** `--shard` across nodes to reduce wall time.
- **Artifacts & uploads.** Upload HTML reports, traces, videos for PR review.
- **Secrets & envs.** Use CI secrets; mask logs; rotate tokens.
- **Flake strategy.** Retries, quarantine, ownership rotation.
- **Quality gates.** Block merges on failing smoke/critical suites.
- **Containerization.** Use base images with all browser deps preinstalled.