Automation with Codex and Playwright

Lesson 7 · One durable check beats ten flaky ones

Today’s outcome

- Choose a worthy automation target
- Add a deterministic browser/API assertion
- Use Codex for explanation and drafts
- Avoid flake by design

What deserves automation?

High-value candidates are:

- repeatable
- clear expected outcome
- stable synthetic setup
- expensive/risky if missed
- likely to regress

Test layers

Need	Best first tool
pure rule	unit/domain test
server/data contract	integration/API test
critical user journey	browser E2E test
visual/experience ambiguity	human + bounded AI review

Semantic, deterministic assertions

Prefer role/name or stable test selectors.

Assert a meaningful UI state, URL, network response, or application state—not merely “the click happened.”

Codex pair-programming loop

1. Explain the existing test
2. Propose smallest patch
3. Review intent and scope
4. Run red/green proof
5. Improve the durable playbook

Flake is a quality signal

Do not solve it with unlimited retries.

Look for dynamic time/data, animation, race conditions, weak locators, external dependencies, and shared state.

Lab: Build One Durable Regression Check

Builder: minimal test diff, actual A/B red/green runs, stability rationale and coverage limit.

Foundation: precise assertion design with predicted A/B results; execution explicitly unverified.

The verifier owns review—not just typing.

Quality gate mindset

A green test means one specified behavior passed under known conditions.

It does not prove the whole product is ready.

Exit ticket

What makes your regression check deterministic?