Playwright TypeScript Automation

Hooks, suites, and execution controls - checklist, definitions, and examples

How to use this handout
Keep this document open during labs. When you write a new test file, confirm you are using the checklist items that apply to your scenario. Each concept includes a short definition and a copy-paste-ready example.

Course-wide Checklist

A) Test Structure and Organization

[ ] Use test.describe() to group tests by feature/module.
[ ] Use consistent naming: Feature - Scenario - Expected Result.
[ ] Keep tests independent (no dependency on previous tests).
[ ] Use tags/annotations when needed (e.g., @smoke, @regression).

B) Hooks and Lifecycle

[ ] Use beforeAll() for one-time setup (seed data, create users, generate auth state).
[ ] Use afterAll() for one-time cleanup (delete test data, close resources).
[ ] Use beforeEach() to prepare state per test (navigate, set mocks, load auth).
[ ] Use afterEach() to capture evidence on failure (screenshot/trace/logs).

C) Execution Control

[ ] Use test.only() only locally for debugging. Never commit it.
[ ] Use test.skip() for blocked tests (include reason and ticket).
[ ] Optional: prefer test.fixme() for known bugs you still want visible in reports.

D) Quality and Stability

[ ] Prefer getByRole(), getByLabel(), getByTestId() locators.
[ ] Avoid waitForTimeout() unless absolutely unavoidable.
[ ] Ensure every test has at least one meaningful assertion.
[ ] Keep UI tests fast: use API setup where possible.

Tip: The fastest way to reduce flaky tests is strong locator strategy + clean setup/teardown.

Definitions and Examples

All examples assume Playwright Test with TypeScript (**@playwright/test**). Copy and adapt these patterns into your project.

1) test.describe() - Group related tests

Definition: Creates a logical suite of tests. Hooks inside describe() apply to tests in that block.
Example:

```
import { test, expect } from "@playwright/test";

test.describe("Login Feature", () => {
  test("should login with valid credentials", async ({ page }) => {
    await page.goto("https://example.com/login");
    await page.getByLabel("Username").fill("demo");
    await page.getByLabel("Password").fill("demo123");
    await page.getByRole("button", { name: "Sign in" }).click();

    await expect(page.getByRole("heading", { name: "Dashboard" })).toBeVisible();
  });
});
```

2) test.beforeAll() - One-time setup

Definition: Runs once before all tests in the current describe() block. Use it for seeding data, creating users via API, or generating auth state once.
Example:

```
import { test, expect } from "@playwright/test";

test.describe("Authenticated Area", () => {
  test.beforeAll(async ({ browser }) => {
    const page = await browser.newPage();
    await page.goto("https://example.com/login");
    await page.getByLabel("Username").fill("demo");
    await page.getByLabel("Password").fill("demo123");
    await page.getByRole("button", { name: "Sign in" }).click();

    // Save login session for reuse
    await page.context().storageState({ path: "storage/auth.json" });
    await page.close();
  });

  test("should open profile page", async ({ page }) => {
    await page.goto("https://example.com/profile");
    await expect(page.getByRole("heading", { name: "Profile" })).toBeVisible();
  });
});
```

3) test.afterAll() - One-time cleanup

Definition: Runs once after all tests in the current describe() block. Use it to delete test data created in beforeAll(), or close external resources.
Example:

```
import { test } from "@playwright/test";

test.describe("Data Setup/Cleanup Example", () => {
  let createdUserId: string;

  test.beforeAll(async ({ request }) => {
    const res = await request.post("https://api.example.com/users", {
      data: { name: "PW User" },
    });
    const body = await res.json();
    createdUserId = body.id;
  });

  test.afterAll(async ({ request }) => {
    if (createdUserId) {
      await request.delete(`https://api.example.com/users/${createdUserId}`);
    }
  });

  test("uses the created user", async ({ page }) => {
    // test steps...
  });
});
```

4) test.beforeEach() - Per-test setup

Definition: Runs before each test in the describe() block. Use it to navigate to a start page, reset state, or set route mocks.
Example:

```
import { test, expect } from "@playwright/test";

test.describe("Cart Module", () => {
  test.beforeEach(async ({ page }) => {
    await page.goto("https://example.com");
    await expect(page.getByRole("navigation")).toBeVisible();
  });

  test("should add item to cart", async ({ page }) => {
    await page.getByRole("link", { name: "Products" }).click();
    await page.getByRole("button", { name: "Add to cart" }).first().click();

    await expect(page.getByRole("link", { name: /Cart/i })).toBeVisible();
  });
});
```

5) test.afterEach() - Evidence and teardown

Definition: Runs after each test. Common use is to capture screenshots or logs when a test fails.
Example:

```
import { test } from "@playwright/test";

test.describe("Evidence Capture", () => {
  test.afterEach(async ({ page }, testInfo) => {
    if (testInfo.status !== testInfo.expectedStatus) {
      await page.screenshot({
        path: `test-results/${testInfo.title}-failed.png`,
        fullPage: true,
      });
    }
  });

  test("example test", async ({ page }) => {
    await page.goto("https://example.com");
    // steps...
  });
});
```

6) test() - The test case

Definition: Defines a test. The callback receives fixtures like page, request, and browser.
Example:

```
import { test, expect } from "@playwright/test";

test("home page should show heading", async ({ page }) => {
  await page.goto("https://example.com");
  await expect(page.getByRole("heading", { name: "Welcome" })).toBeVisible();
});
```

7) test.only() - Run only this test (debugging)

Definition: Runs only the marked test(s). Use locally for debugging. Do not commit test.only() to shared branches because it breaks CI coverage.
Example:

```
import { test, expect } from "@playwright/test";

test.only("debug this test only", async ({ page }) => {
  await page.goto("https://example.com");
  await expect(page).toHaveTitle(/Example/);
});
```

8) test.skip() - Skip a test (blocked or not applicable)

Definition: Skips the test. Always include a reason and a ticket/bug id so the skip is traceable.
Example:

```
import { test } from "@playwright/test";

test.skip("Blocked by BUG-1234: login service returns 500", async ({ page }) => {
  await page.goto("https://example.com/login");
});
```

Rule: test.only() is a local debugging tool. Before you push code, search for 'only(' and remove it.

Complete End-to-End Example (All Together)

This example shows a full suite that uses describe, beforeAll, afterAll, beforeEach, afterEach, test, and skip in one file.

```
import { test, expect } from "@playwright/test";

test.describe("E2E: Purchase Flow", () => {
  test.beforeAll(async ({ request }) => {
    // One-time setup example: seed required data
    await request.post("https://api.example.com/test/seed", {
      data: { scenario: "purchase" },
    });
  });

  test.afterAll(async ({ request }) => {
    // One-time cleanup example
    await request.post("https://api.example.com/test/cleanup", {
      data: { scenario: "purchase" },
    });
  });

  test.beforeEach(async ({ page }) => {
    // Per-test start condition
    await page.goto("https://example.com");
    await expect(page.getByRole("navigation")).toBeVisible();
  });

  test.afterEach(async ({ page }, testInfo) => {
    // Evidence on failure
    if (testInfo.status !== testInfo.expectedStatus) {
      await page.screenshot({
        path: `test-results/${testInfo.title}-failed.png`,
        fullPage: true,
      });
    }
  });

  test("should complete checkout successfully", async ({ page }) => {
    await page.getByRole("link", { name: "Products" }).click();
    await page.getByRole("button", { name: "Add to cart" }).first().click();
    await page.getByRole("link", { name: /Cart/i }).click();
    await page.getByRole("button", { name: "Checkout" }).click();

    await expect(page.getByText("Order confirmed")).toBeVisible();
  });

  test.skip("Blocked by BUG-2222: payment gateway test env unavailable", async ({ page }) => {
    // placeholder test
  });

  // Use locally only, do not commit:
  // test.only("debug checkout issue", async ({ page }) => {
  //   // ...
  // });
});
```

Suggested student practice: copy this file, replace URLs with your training application, and add one additional assertion per test.