Playwright

Reliable end-to-end tests in Chromium, Firefox and WebKit.

01 Install & run

```
npm init playwright@latest
npm install -D @playwright/test
npx playwright install --with-deps
npx playwright test
```

init asks for TypeScript or JavaScript, a tests folder, a GitHub Actions workflow and browsers. --with-deps adds the OS libraries the browsers need.

02 Write a test

```
import { test, expect }
  from '@playwright/test'
test('has title', async ({ page }) => {
  await page.goto('/')
  await expect(page).toHaveTitle(/My App/)
})
```

Each test gets a fresh browser context and page, so no state leaks. Relative URLs use baseURL from the config. Name files *.spec.ts or *.test.ts.

03 Hooks, groups & modifiers

```
test.describe('Dashboard', () => {
  test.beforeEach(async ({ page }) => {
    await page.goto('/dashboard')
  })
  test('desktop', async ({ isMobile }) => {
    test.skip(isMobile, 'Desktop only')
  })
})
```

beforeAll/afterAll run once per worker, again after a failure. Also test.only, test.fixme, test.slow() and describe.skip. describe.configure({ mode: 'serial' }) is not recommended.

04 Locators

```
page.getByRole('button', { name: 'Submit' })
page.getByLabel('Email')
page.getByText('Welcome back')
page.getByTestId('submit-btn')
page.getByRole('listitem')
  .filter({ hasText: 'Product 1' })
  .getByRole('button', { name: 'Buy' })
page.locator('.card')  // CSS: last resort
```

Prefer getByRole with a name. getByTestId reads data-testid. Locators are strict: an action fails if several elements match.

05 Actions

```
await page.getByLabel('Name').fill('Ann')
await page.getByLabel('Name').press('Enter')
await page.getByLabel('Terms').check()
await page.getByLabel('Country')
  .selectOption({ label: 'USA' })
await page.getByLabel('Upload')
  .setInputFiles(['a.pdf', 'b.pdf'])
```

fill() sets the value at once; pressSequentially() types key by key. Actions wait until the element is ready.

06 Web-first assertions

```
await expect(page).toHaveURL('/dashboard')
const toast = page.getByRole('status')
await expect(toast).toBeVisible()
await expect(toast).toHaveText('Saved')
const items = page.getByRole('listitem')
await expect.soft(items).toHaveCount(5)
```

They retry until they pass or 5 s run out ({ timeout } overrides). expect.soft keeps going but fails the test.

07 Navigation & waiting

```
await page.goto('/products')
const res = page.waitForResponse('**/api/*')
await page.getByText('Load more').click()
await res
await page.getByText('Loading')
  .waitFor({ state: 'hidden' })
await page.waitForURL('**/products/2')
```

Avoid 'networkidle', waitForTimeout, page.$, type and waitForNavigation: use expect, locators, fill and waitForURL.

08 Visual comparisons

```
await expect(page).toHaveScreenshot()
await expect(page.getByTestId('chart'))
  .toHaveScreenshot({ maxDiffPixels: 100,
    mask: [page.locator('.time')] })
```

```
npx playwright test --update-snapshots
```

The first run writes baselines to <file>-snapshots/ and fails; commit them. Names include the project and platform.

09 Run & debug

```
npx playwright test login.spec.ts
npx playwright test -g "login" --headed
npx playwright test --project=firefox
npx playwright test --ui
npx playwright test --debug
npx playwright show-trace trace.zip
npx playwright codegen localhost:3000
```

--debug opens the Inspector, headed with no timeout (PWDEBUG=1 too); await page.pause() stops there. Record traces with trace: 'on-first-retry' or --trace on.

Playwright

Configuration, network, auth and CI patterns.

10 playwright.config.ts

```
export default defineConfig({
  testDir: './tests',
  fullyParallel: true,
  forbidOnly: !!process.env.CI,
  retries: process.env.CI ? 2 : 0,
  workers: process.env.CI ? 1 : undefined,
  reporter: 'html',
  use: { baseURL: 'http://localhost:3000' },
})
```

Runner options are top level; browser options go in use. workers defaults to half the CPU cores.

11 Projects & web server

```
projects: [
  { name: 'chromium',
    use: { ...devices['Desktop Chrome'] } },
  { name: 'mobile',
    use: { ...devices['iPhone 13'] } },
],
webServer: { command: 'npm run dev',
  url: 'http://localhost:3000',
  reuseExistingServer: !process.env.CI },
```

Each project runs the whole suite. use also takes locale, timezoneId, colorScheme and geolocation (with permissions).

12 Sharding & reports

```
npx playwright test --shard=1/4
npx playwright merge-reports \
  --reporter html ./all-blob-reports
```

Set reporter: 'blob' on CI, then merge every shard's blob-report folder. fullyParallel balances shards by test.

13 Mock the network

```
await page.route('**/api/users', (route) =>
  route.fulfill({ json: [{ name: 'Jo' }] }))
await page.route('**/*.png', (route) =>
  route.abort())
```

Route before goto(). route.fetch() gets the real response to edit for fulfill({ response, json }). ** spans path segments.

14 API testing

```
test('create user', async ({ request }) => {
  const res = await request.post('/users', {
    data: { name: 'John' },
  })
  expect(res.status()).toBe(201)
})
```

request has its own cookies; page.request shares the page's.

15 Reuse login state

```
const file = 'playwright/.auth/user.json'
setup('login', async ({ page }) => {
  await page.goto('/login')  // log in here
  await page.waitForURL('/dashboard')
  await page.context()
    .storageState({ path: file })
})
```

```
{ name: 'setup',
  testMatch: /.*\.setup\.ts/ },
{ name: 'chromium', dependencies: ['setup'],
  use: { storageState: file } },
```

The second part goes in projects, so setup runs first. Git-ignore playwright/.auth/.

16 Timeouts & retries

timeout	30 s per test
expect.timeout	5 s per assertion
actionTimeout	Per action: none
navigationTimeout	Per navigation: none
globalTimeout	Whole run: none

Action and navigation go in use. test.slow() triples one test.

17 Fixtures & page objects

```
type Fixtures = { todoPage: TodoPage }
export const test = base.extend<Fixtures>({
  todoPage: async ({ page }, use) => {
    await use(new TodoPage(page))
  },
})
```

Import test as base. Tests then take ({ todoPage }).

18 GitHub Actions

```
- uses: actions/checkout@v7
- uses: actions/setup-node@v7
  with:
    node-version: lts/*
- run: npm ci
- run: npx playwright install --with-deps
- run: npx playwright test
- uses: actions/upload-artifact@v7
  if: ${{ !cancelled() }}
  with:
    name: playwright-report
    path: playwright-report/
```

Steps of a job with runs-on: ubuntu-latest.