

# Playwright — Method Syntax Quick Reference (JavaScript)

A compact quick-reference of common Playwright APIs and method syntax for JavaScript (Node.js). Use this as a cheat-sheet while writing tests.

## Contents

1. Import & Test Basics
2. Page navigation & lifecycle
3. Locator API (selection & actions)
4. Assertions (expect)
5. Browser, Context & Page creation
6. Fixtures & Hooks (test runner)
7. Network & API (request)
8. Debugging, Tracing & Screenshots
9. CLI & Config snippets
10. Useful patterns

# 1. Import & Test Basics

Common imports when using Playwright Test runner in JS.

```
// import (CommonJS)
const { test, expect } = require('@playwright/test');

// import (ESM)
// import { test, expect } from '@playwright/test';

// basic test
test('example', async ({ page }) => {
  await page.goto('https://example.com');
  await expect(page).toHaveTitle(/Example/);
});
```

# 2. Page navigation & lifecycle

Page navigation, waiting and lifecycle helpers.

```
// goto with options
await page.goto('https://example.com/path', { waitUntil: 'load', timeout: 30000 });

// reload / goBack / goForward
await page.reload();
await page.goBack();
await page.goForward();

// wait for load/selector/navigation
await page.waitForLoadState('networkidle');
await page.waitForSelector('#app');
await page.waitForURL('**/dashboard');
```

# 3. Locator API (selection & actions)

Locator is the recommended API for resilient element interaction.

```
// creating locators
const btn = page.locator('button:has-text("Sign in")');
const input = page.getByPlaceholder('Email');
const byRole = page.getByRole('button', { name: 'Submit' });

// common actions
await input.fill('user@example.com');
await btn.click();
await btn dblclick();
await btn.hover();
await input.press('Enter');

// assertions on locator
await expect(btn).toBeVisible();
await expect(input).toHaveValue('user@example.com');

// locator chaining
await page.locator('form').locator('input[name="q"]').fill('playwright');
```

# 4. Assertions (expect)

Playwright's expect supports many matchers with auto-waiting.

```
// page-level
await expect(page).toHaveTitle(/Dashboard/);
await expect(page).toHaveURL(/dashboard/);

// locator-level
await expect(page.getText('Welcome')).toBeVisible();
await expect(page.locator('.count')).toHaveText('42');
await expect(response).toBeOK(); // for APIResponse
```

```
// timeout option
await expect(locator).toHaveText('Done', { timeout: 10000 });
```

## 5. Browser, Context & Page creation

Programmatic browser control (not using test runner fixtures).

```
// using playwright package (not test runner)
const playwright = require('playwright');
(async () => {
  const browser = await playwright.chromium.launch({ headless: true });
  const context = await browser.newContext({ viewport: { width:1280, height:720 } });
  const page = await context.newPage();
  await page.goto('https://example.com');
  await browser.close();
})();

// storageState for authentication
await context.storageState({ path: 'state.json' });
await browserType.launchPersistentContext(userDataDir, { headless: false });
```

## 6. Fixtures & Hooks (test runner)

Built-in fixtures: page, context, browser, request, and test hooks.

```
// hooks
test.beforeAll(async () => { /* global setup */ });
test.afterAll(async () => { /* teardown */ });

// per-test
test.beforeEach(async ({ page }) => { /* runs before each test */ });

// extend fixtures
const base = require('@playwright/test');
const test = base.test.extend({
  authPage: async ({ page }, use) => {
    // perform login once per test
    await page.goto('/login');
    await page.fill('#email', 'user');
    await page.fill('#pass', 'pw');
    await page.click('text=Sign in');
    await use(page);
  }
});

// using the custom fixture
test('uses authPage', async ({ authPage }) => {
  await expect(authPage).toHaveURL('/dashboard');
});
```

## 7. Network & API (request)

Intercepting, mocking requests and APIRequestContext examples.

```
// route interception / mocking
await page.route('**/api/**', route => {
  const req = route.request();
  if (req.method() === 'POST')
    route.fulfill({ status: 200, body: JSON.stringify({ ok: true }) });
  else
    route.continue();
});

// wait for response
const [response] = await Promise.all([
  page.waitForResponse(resp => resp.url().includes('/status') && resp.status() === 200),
  page.click('button#check')
]);
// APIRequestContext (test runner fixture 'request')
```

```
const r = await request.get('https://api.example.com/health');
expect(r.ok()).toBeTruthy();
const json = await r.json();
```

## 8. Debugging, Tracing & Screenshots

Tools to capture failures and debug tests.

```
// launch inspector
// PWDEBUG=1 npx playwright test

// trace
await page.tracing.start({ screenshots:true, snapshots:true });
await page.tracing.stop({ path: 'trace.zip' });

// screenshot and video
await page.screenshot({ path: 'screenshot.png', fullPage: true });
// record video via context
const context = await browser.newContext({ recordVideo: { dir: 'videos/' } });
await context.close();
```

## 9. CLI & Config snippets

Useful CLI commands and playwright.config.js patterns.

```
// run tests
npx playwright test
npx playwright test tests/login.spec.js --project=chromium

// generate code
npx playwright codegen https://example.com

// playwright.config.js (example)
module.exports = {
  timeout: 30000,
  retries: 1,
  use: { headless: true, viewport: { width:1280, height:720 } },
  projects: [ { name: 'chromium', use: { browserName: 'chromium' } } ]
};
```

## 10. Useful patterns

Storage state, authentication helpers, waiting for network, and combining API + UI for fast tests.

```
// save authenticated state
await context.storageState({ path: 'auth.json' });

// reuse storageState in config
module.exports = { use: { storageState: 'auth.json' } };

// hybrid tests: create data via API then validate via UI
await request.post('/api/create', { data: { name: 'item' } });
await page.goto('/items');
await expect(page.getByText('item')).toBeVisible();
```

## Credits

Generated with assistance from ChatGPT — Playwright method syntax quick reference (JavaScript). Review and adapt examples to your project's Playwright version.