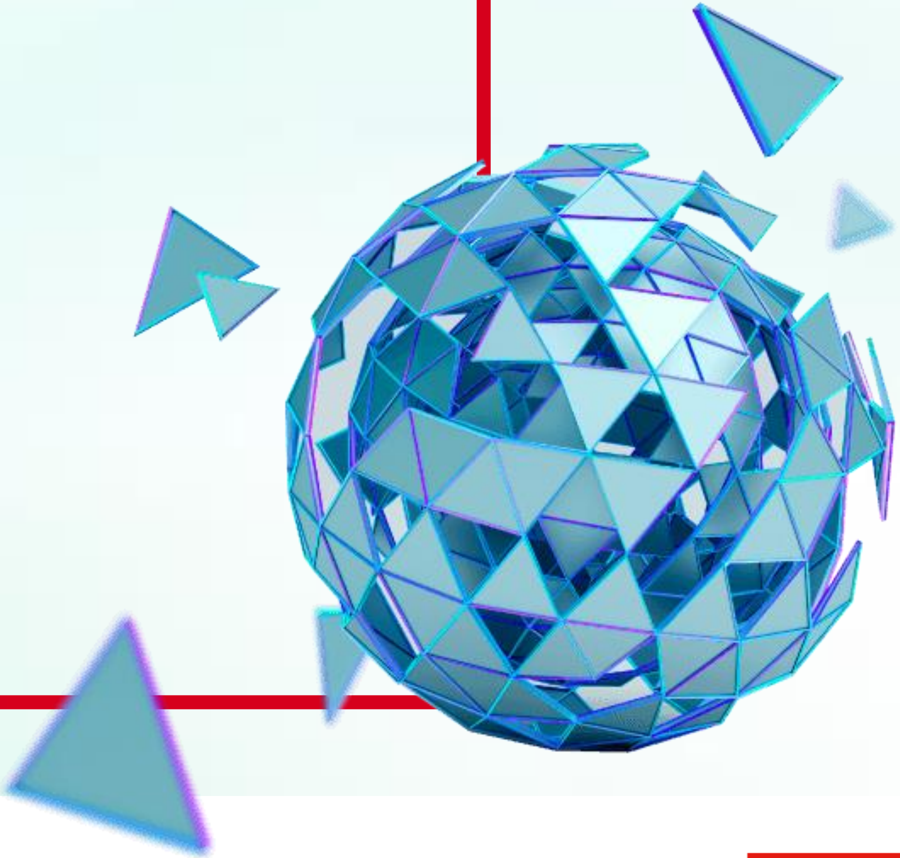
Automation with Playwright



Tien Nguyen

Nash Tech.

Agenda

Playwright Introduction

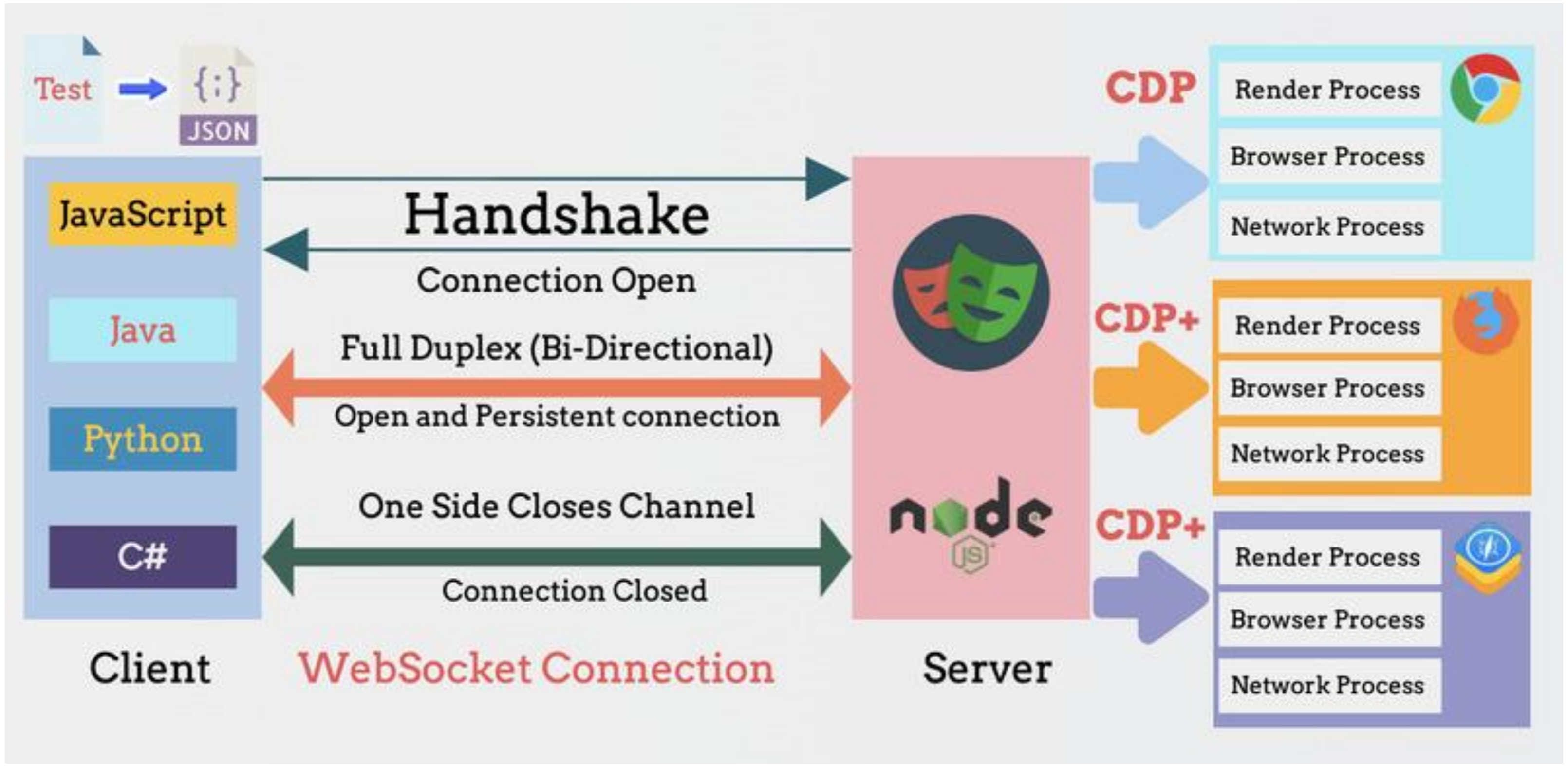
Playwright Introduction

- Playwright overview
- Playwright solution
- Playwright vs Selenium
- The first test case with Playwright

Playwright Overview

- Open-source NodeJS-based framework developed by Microsoft
- Programming languages: **TypeScript, JavaScript**, Python, .NET, Java
- Cross-browser: Chromium, WebKit, and Firefox
- Cross-platform: Windows, Linux, and macOS, headless and headed, mobile web
- Auto wait
- Tracing
- Power tooling: Codegen, Playwright Inspector, Trace viewer
- No trade-offs

Playwright Solution

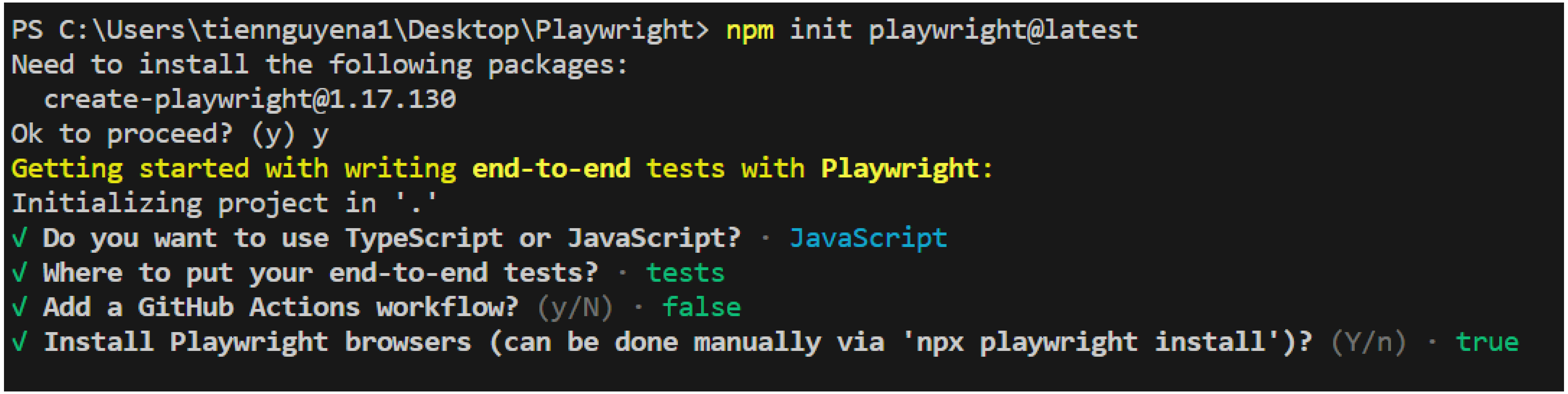


Playwright vs Selenium

	Playwright	**Selenium**
Solution	Using DevTool Protocol	Using WebDriver API
Open-source	Yes	Yes
Browser Support	Chromium, Firefox, and WebKit , Safari, Edge	Chrome, Firefox, Edge, Safari...
Platform	Windows, Linux, and macOS	Windows, Linux, and macOS
Programming language	Java, Python, .NET C#, TypeScript and JavaScript.	Java, Python, C#, Ruby, Perl, PHP, and JavaScript
Features	Provide more modern features like smartwait, working with shadow dom, switch to iframe easily, test retry, visual validation...	Need to implement these features by ourselves.
Locator	Introduce many new kinds of selector	Basic selector
Code Generation	Support Codegen to generate code	Can use Selenium IDE
Building automation framework	Easy to build with a lot of built-in features like running test in parallel, report, capturing screenshot and video, cross-browser	Need to have knowledge about many libraries and combine all of them to create a new framework.
CICD Integration	Yes	Yes
Community	Smaller community	Larger community

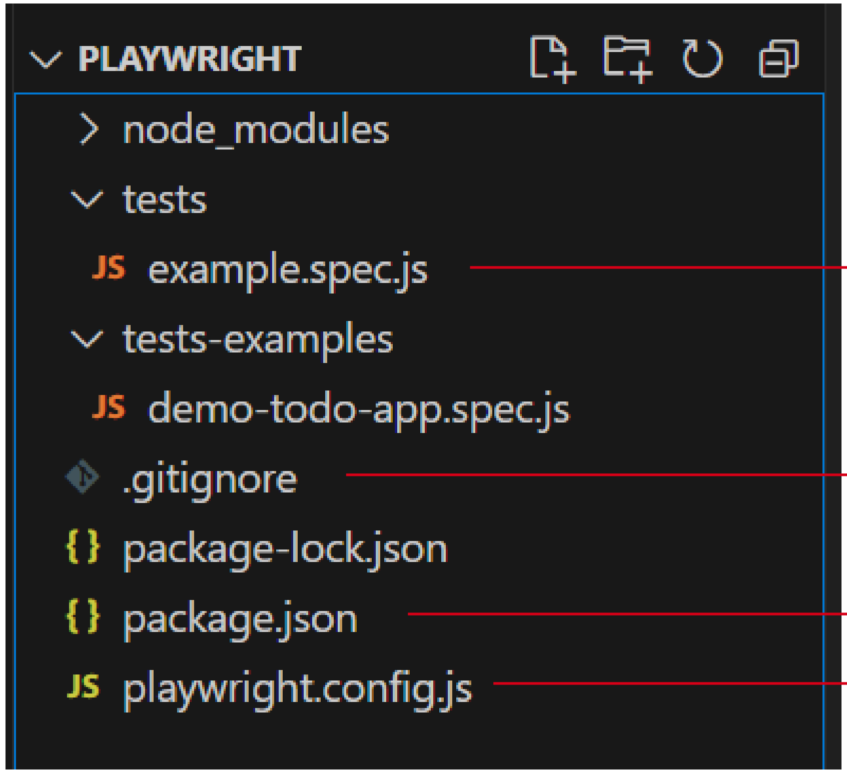
The first test case with Playwright

- Install NodeJs
- Install VS Code and Playwright extension(Playwright Test for VS Code)
- Run “npm init playwright@latest” and choose below options



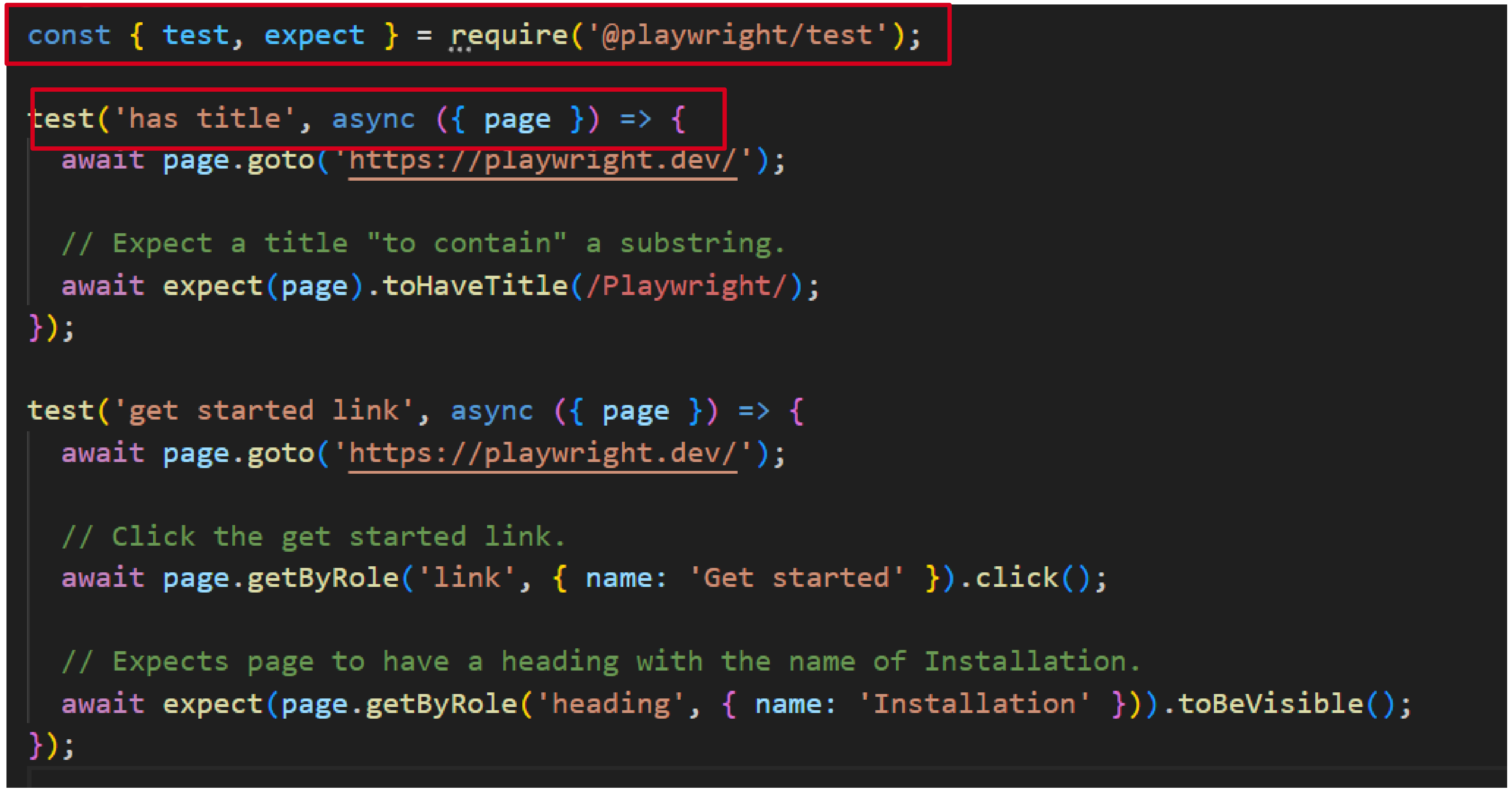
```
PS C:\Users\tiennguyena1\Desktop\Playwright> npm init playwright@latest
Need to install the following packages:
  create-playwright@1.17.130
Ok to proceed? (y) y
Getting started with writing end-to-end tests with Playwright:
Initializing project in '.'
√ Do you want to use TypeScript or JavaScript? · JavaScript
√ Where to put your end-to-end tests? · tests
√ Add a GitHub Actions workflow? (y/N) · false
√ Install Playwright browsers (can be done manually via 'npx playwright install')? (Y/n) · true
```

The first test case with Playwright



```
PLAYWRIGHT
  > node_modules
  v tests
      example.spec.js
  v tests-examples
      demo-todo-app.spec.js
    .gitignore
    package-lock.json
    package.json
    playwright.config.js
```

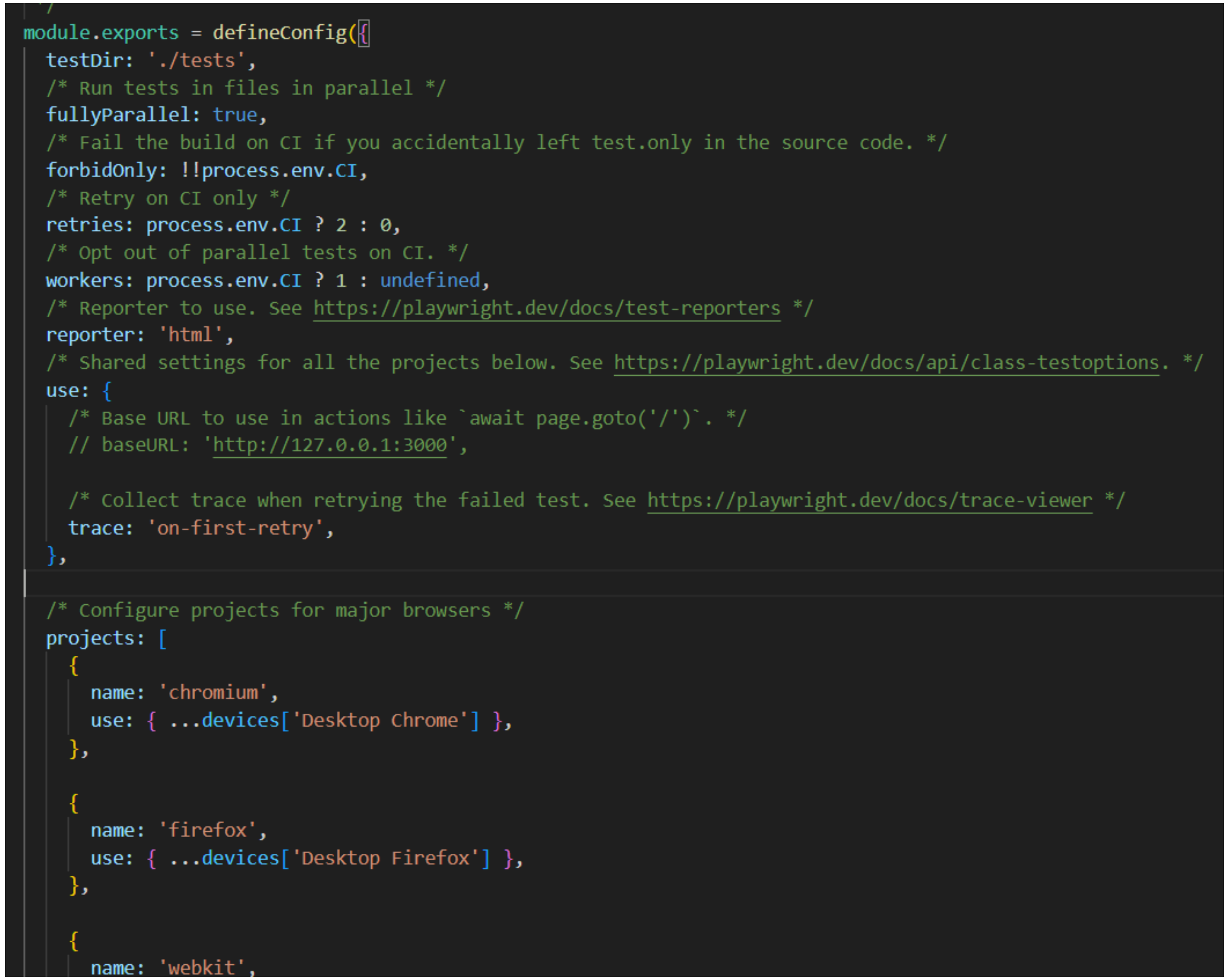
The first test case with Playwright



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

test('has title', async ({ page }) => {
  await page.goto('https://playwright.dev/');

  // Expect a title "to contain" a substring.
  await expect(page).toHaveTitle(/Playwright/);
});

test('get started link', async ({ page }) => {
  await page.goto('https://playwright.dev/');

  // Click the get started link.
  await page.getByRole('link', { name: 'Get started' }).click();

  // Expects page to have a heading with the name of Installation.
  await expect(page.getByRole('heading', { name: 'Installation' })).toBeVisible();
});
```

Playwright Test Case

The first test case with Playwright



```
module.exports = defineConfig({
  testDir: './tests',
  /* Run tests in files in parallel */
  fullyParallel: true,
  /* Fail the build on CI if you accidentally left test.only in the source code. */
  forbidOnly: !!process.env.CI,
  /* Retry on CI only */
  retries: process.env.CI ? 2 : 0,
  /* Opt out of parallel tests on CI. */
  workers: process.env.CI ? 1 : undefined,
  /* Reporter to use. See https://playwright.dev/docs/test-reporters */
  reporter: 'html',
  /* Shared settings for all the projects below. See https://playwright.dev/docs/api/class-testoptions. */
  use: {
    /* Base URL to use in actions like `await page.goto('/')`. */
    // baseURL: 'http://127.0.0.1:3000',

    /* Collect trace when retrying the failed test. See https://playwright.dev/docs/trace-viewer */
    trace: 'on-first-retry',
  },

  /* Configure projects for major browsers */
  projects: [
    {
      name: 'chromium',
      use: { ...devices['Desktop Chrome'] },
    },

    {
      name: 'firefox',
      use: { ...devices['Desktop Firefox'] },
    },

    {
      name: 'webkit',
```

Playwright Config

The first test case with Playwright

- Run the test case

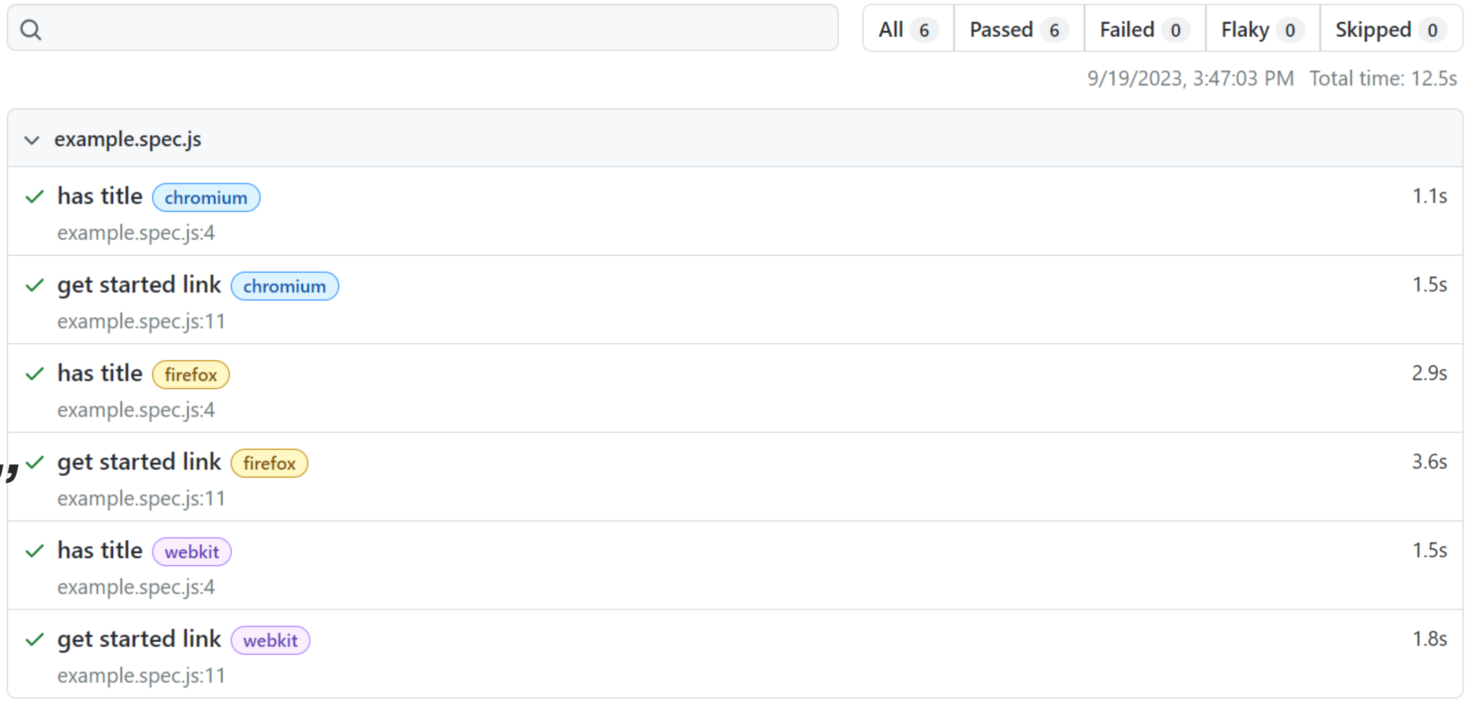
  "npx playwright test"

- Run test in UI mode:

  "npx playwright test --ui"

- Show the report

  "npx playwright show-report"



All 6 | Passed 6 | Failed 0 | Flaky 0 | Skipped 0

9/19/2023, 3:47:03 PM Total time: 12.5s

example.spec.js

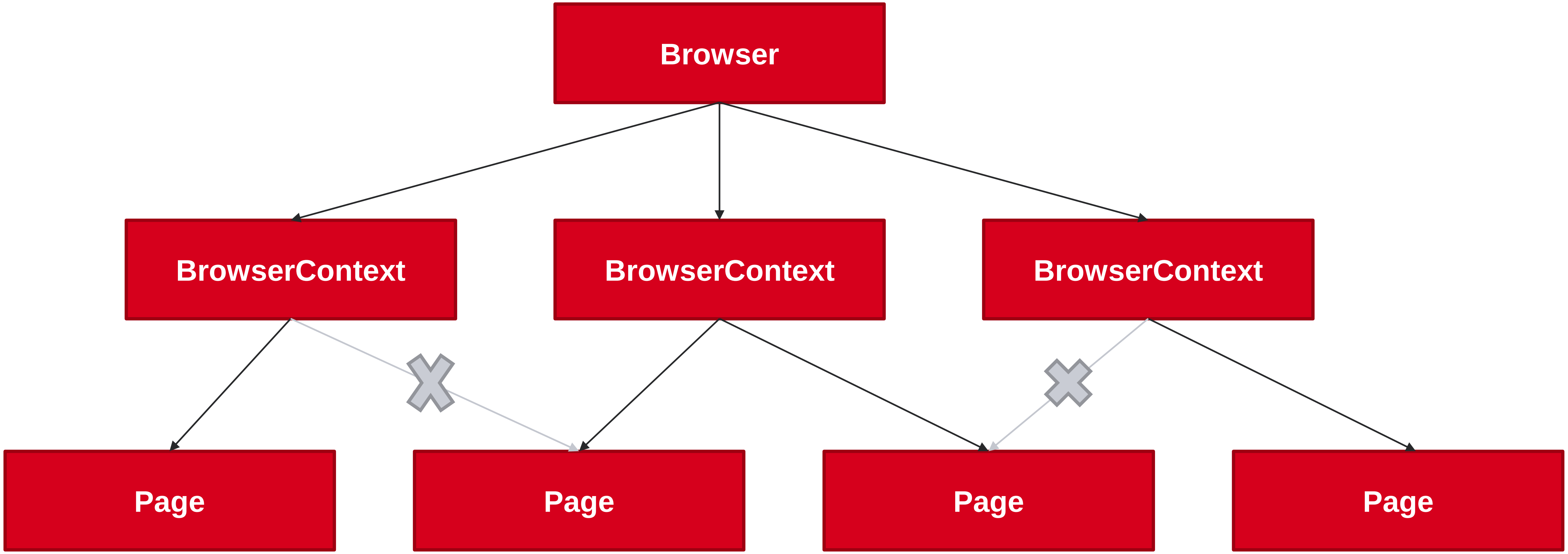
Test	Browser	Location	Time
✓ has title	chromium	example.spec.js:4	1.1s
✓ get started link	chromium	example.spec.js:11	1.5s
✓ has title	firefox	example.spec.js:4	2.9s
✓ get started link	firefox	example.spec.js:11	3.6s
✓ has title	webkit	example.spec.js:4	1.5s
✓ get started link	webkit	example.spec.js:11	1.8s

Playwright basic functions

2. Playwright basic functions

1. Browser, Browser Context and Page
2. Locator
3. Element Interaction
4. Wait
5. Assertion
6. Data-driven
7. Report
8. Fixtures
9. Test Hook
10. Playwright configuration
11. Filter test case for running test
12. API Request
13. Playwright commands

2.1 Browser, Browser Context & Page



2.1.1 Browser

browserType

close

contexts

isConnected

newBrowserCDPSession

newContext

newPage

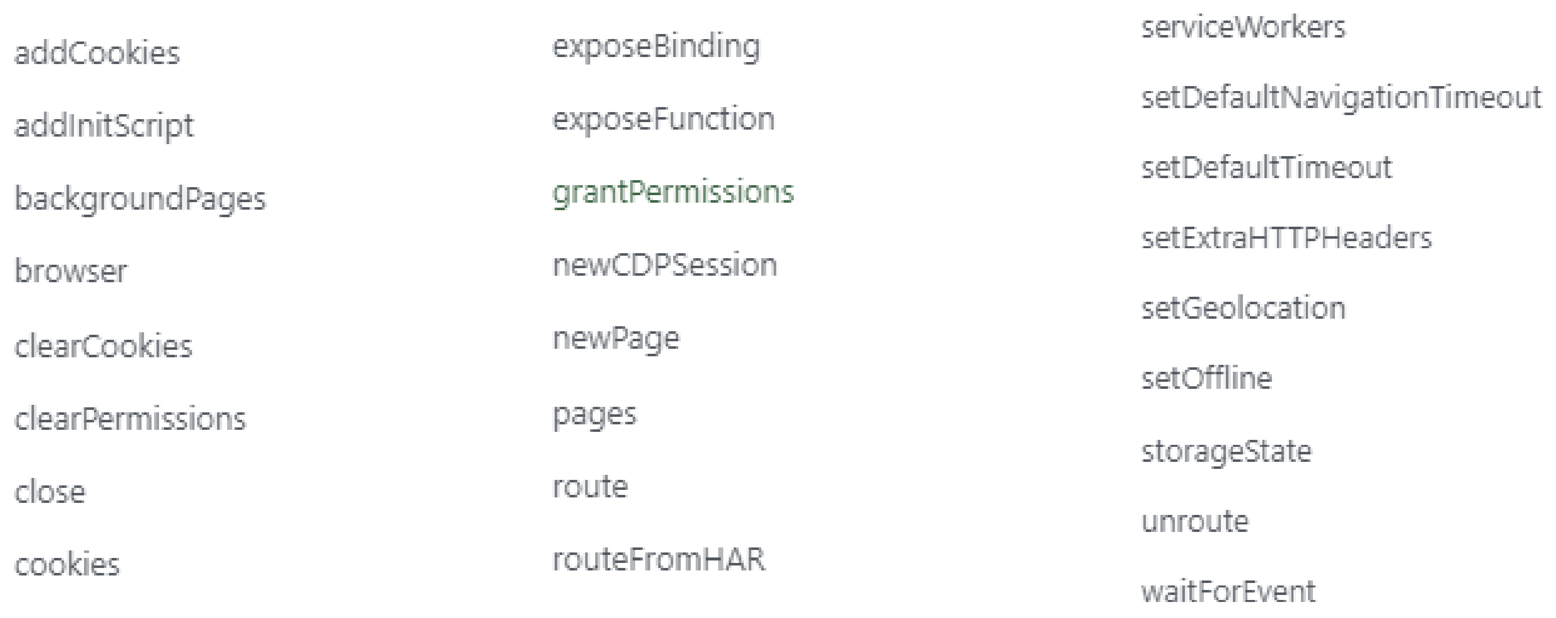
startTracing

stopTracing

version

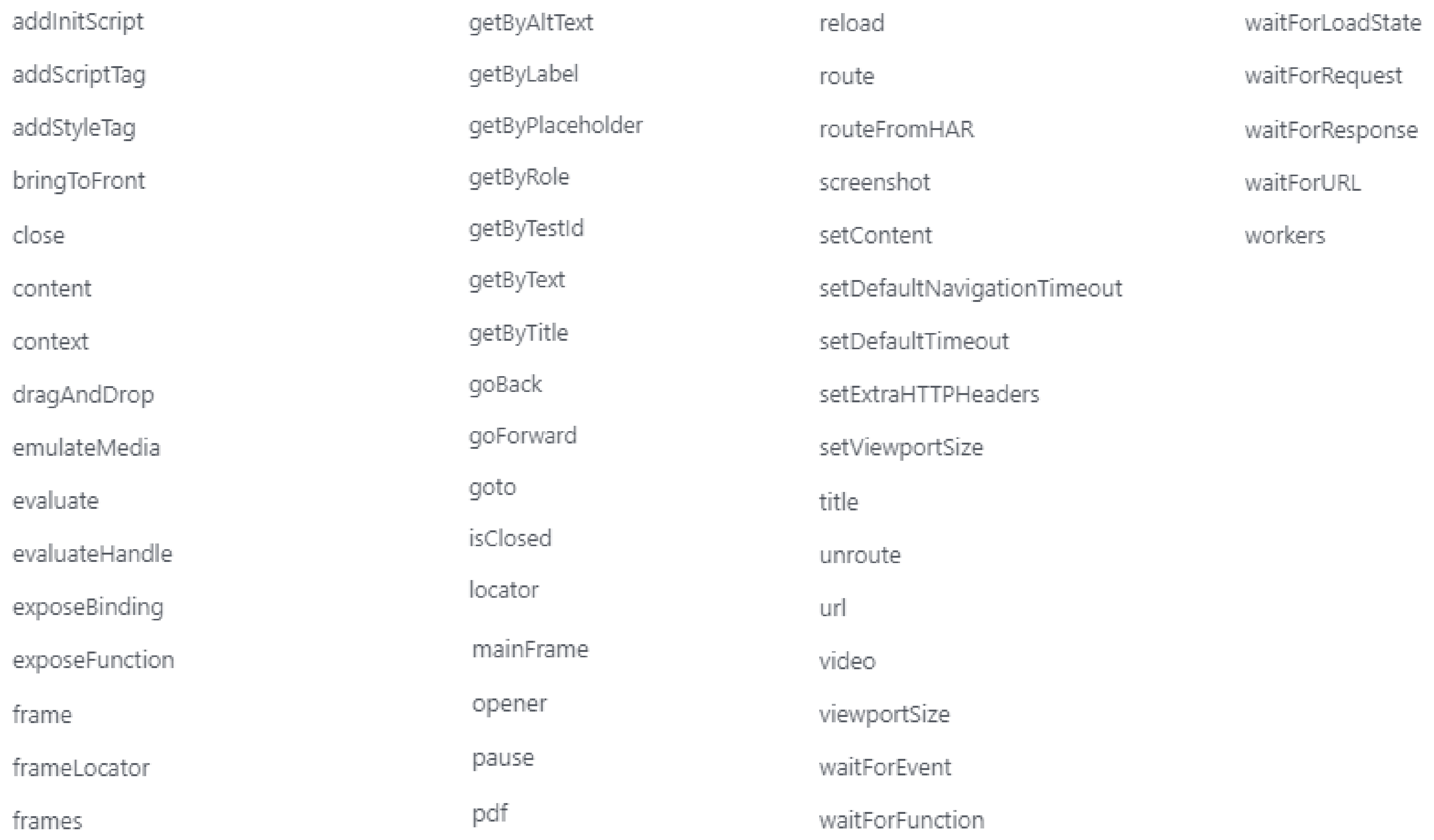
```javascript
(async () => {
  const browser = await playwright.firefox.launch();  // Or 'chromium' or 'webkit'.
  // Create a new incognito browser context.
  const context = await browser.newContext();
  // Create a new page in a pristine context.
  const page = await context.newPage();
  await page.goto('https://example.com');

  // Gracefully close up everything
  await context.close();
  await browser.close();
})();
```

2.1.2 Browser Context



addCookies
addInitScript
backgroundPages
browser
clearCookies
clearPermissions
close
cookies
exposeBinding
exposeFunction
grantPermissions
newCDPSession
newPage
pages
route
routeFromHAR
serviceWorkers
setDefaultNavigationTimeout
setDefaultTimeout
setExtraHTTPHeaders
setGeolocation
setOffline
storageState
unroute
waitForEvent

2.1.3 Page



- addInitScript
- addScriptTag
- addStyleTag
- bringToFront
- close
- content
- context
- dragAndDrop
- emulateMedia
- evaluate
- evaluateHandle
- exposeBinding
- exposeFunction
- frame
- frameLocator
- frames
- getByAltText
- getByLabel
- getByPlaceholder
- getByRole
- getByTestId
- getByText
- getByTitle
- goBack
- goForward
- goto
- isClosed
- locator
- mainFrame
- opener
- pause
- pdf
- reload
- route
- routeFromHAR
- screenshot
- setContent
- setDefaultNavigationTimeout
- setDefaultTimeout
- setExtraHTTPHeaders
- setViewportSize
- title
- unroute
- url
- video
- viewportSize
- waitForEvent
- waitForFunction
- waitForLoadState
- waitForRequest
- waitForResponse
- waitForURL
- workers

2.2 Locator

New locators introduced by Playwright

- page.getByRole() to locate by explicit and implicit accessibility attributes.
- page.getByText() to locate by text content.
- page.getByLabel() to locate a form control by associated label's text.
- page.getByPlaceholder() to locate an input by placeholder.
- page.getByAltText() to locate an element, usually image, by its text alternative.
- page.getByTitle() to locate an element by its title attribute.
- page.getByTestId() to locate an element based on its `data-testid` attribute (other attributes can be configured).

```
await page.getByLabel('User Name').fill('John');

await page.getByLabel('Password').fill('secret-password');

await page.getByRole('button', { name: 'Sign in' }).click();

await expect(page.getByText('Welcome, John!')).toBeVisible();
```

2.2 Locator

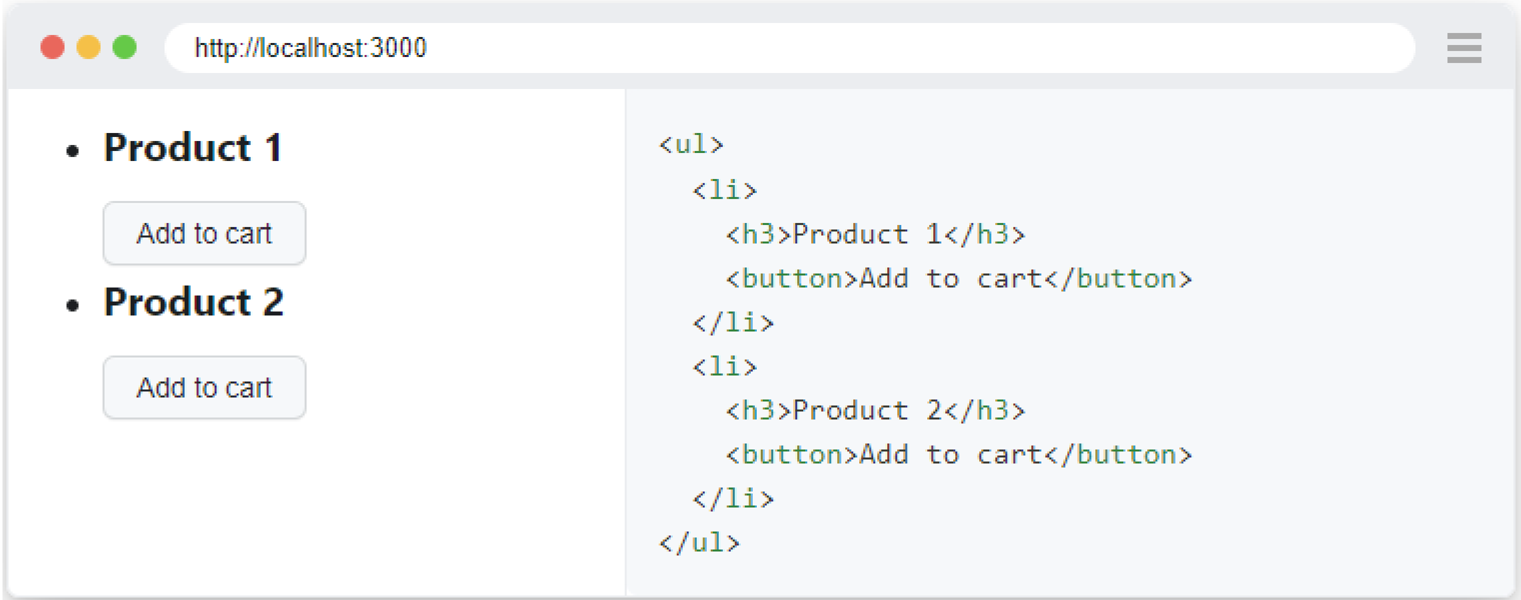
- `Page.locator('text=ming')`
- `Page.locator('css=[aria-hidden=true]')`
- `Page.locator('xpath=//html/body/a')`
- `Page.locator('.something >> visible=true >> nth=2') // nth: selector engine -> 0-based`
- `Page.locator(':nth-match(:text("Buy"), 3)') // CSS pseudo-class -> 1-based`
- `Page.locator('role=checkbox[checked][include-hidden])`
- `Page.locator('id=username')`
- `Page.locator('data-test-id=submit')`
- `Page.locator('data-testid=1234')`
- `Page.locator('data-test=test')`

Old style of locator, but not recommend

2.2 Locator

Filtering Locators

- By text
- By not have text
- By child/descendant
- by not having child/descendant

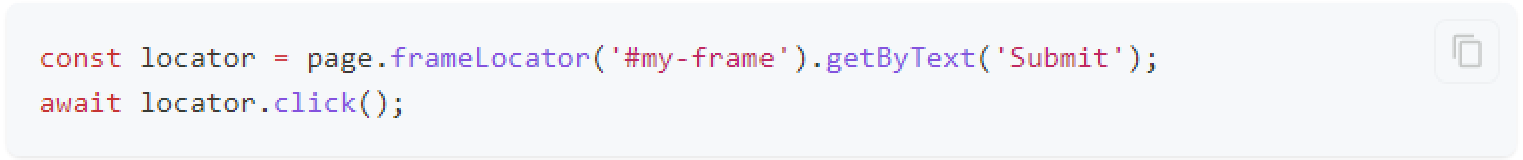


```
<ul>
  <li>
    <h3>Product 1</h3>
    <button>Add to cart</button>
  </li>
  <li>
    <h3>Product 2</h3>
    <button>Add to cart</button>
  </li>
</ul>
```

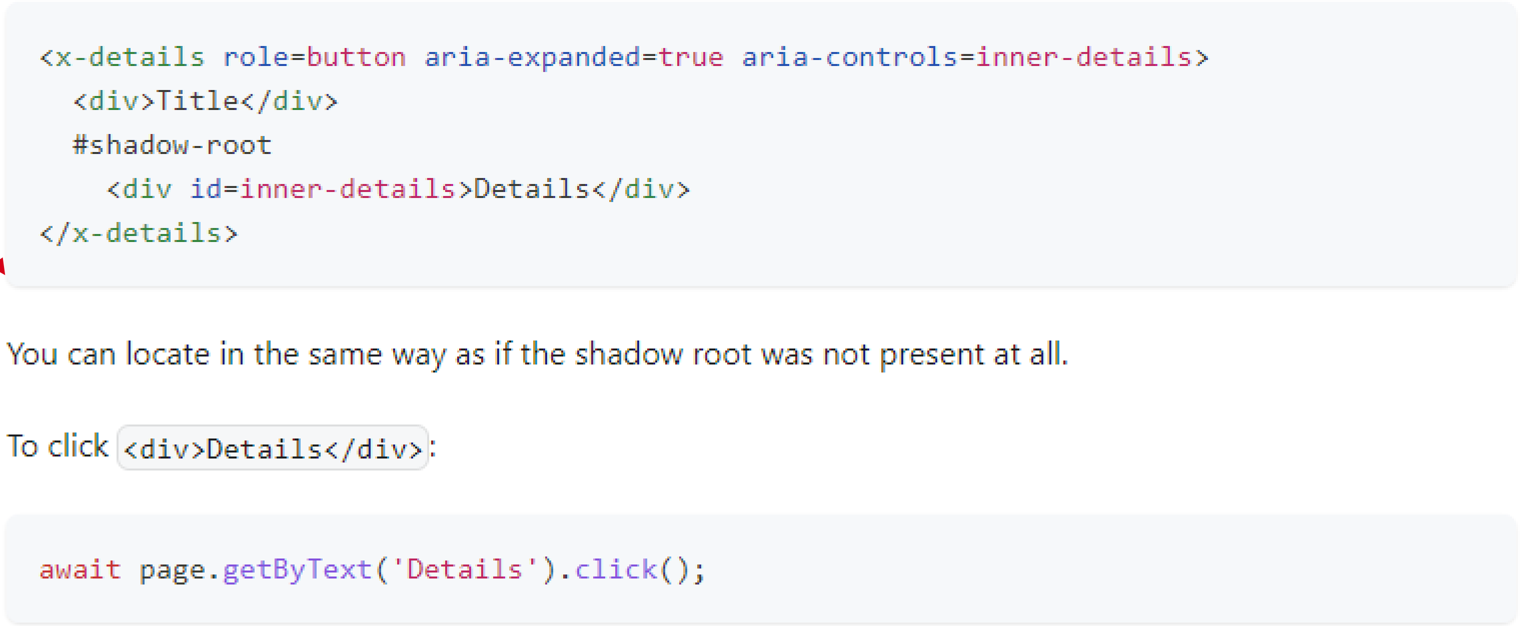
```
await page
    .getByRole('listitem')
    .filter({ has: page.getByRole('heading', { name: 'Product 2' }) })
    .getByRole('button', { name: 'Add to cart' })
    .click();
```

2.2 Locator

- Frame locator
- Shadow DOM locator
- Vue locator
- React locator
- Angular locator
- Custom selector engines



```
const locator = page.frameLocator('#my-frame').getByText('Submit');
await locator.click();
```



```
<x-details role=button aria-expanded=true aria-controls=inner-details>
  <div>Title</div>
  #shadow-root
    <div id=inner-details>Details</div>
</x-details>
```

You can locate in the same way as if the shadow root was not present at all.

To click `<div>Details</div>`:

```
await page.getByText('Details').click();
```

2.3 Element interaction

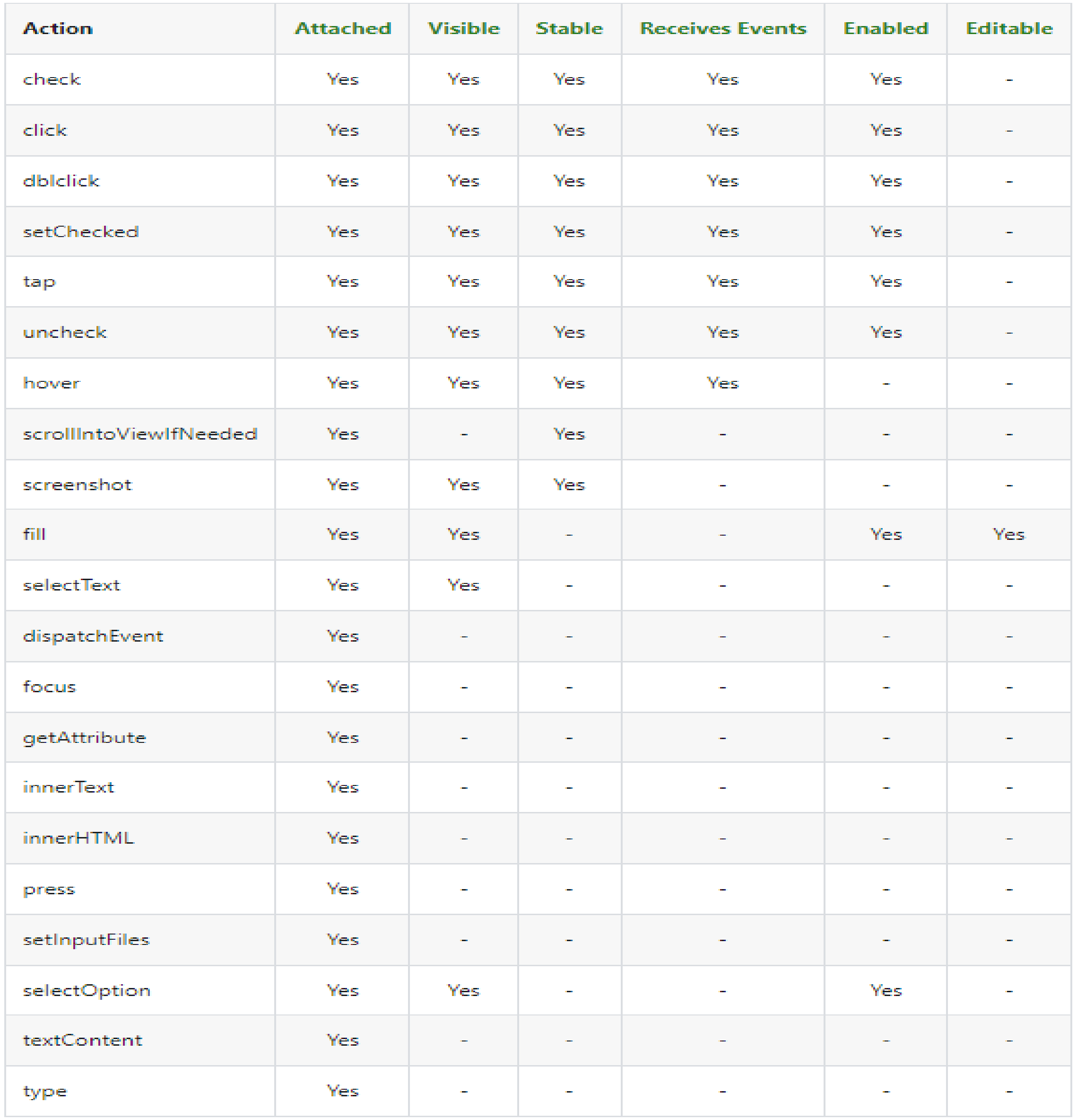
- fill
- check
- isChecked
- selectOption
- click
- dblclick
- hover
- press
- setInputFiles
- focus
- dragTo

```
// Single selection matching the value
await page.getByLabel('Choose a color').selectOption('blue');

// Single selection matching the label
await page.getByLabel('Choose a color').selectOption({ label: 'Blue' });

// Multiple selected items
await page.getByLabel('Choose multiple colors').selectOption(['red', 'green', 'blue']);
```

2.4 Wait

- Auto Wait
- Wait Function

2.4.1 Auto Wait



Action	Attached	Visible	Stable	Receives Events	Enabled	Editable
check	Yes	Yes	Yes	Yes	Yes	-
click	Yes	Yes	Yes	Yes	Yes	-
dblclick	Yes	Yes	Yes	Yes	Yes	-
setChecked	Yes	Yes	Yes	Yes	Yes	-
tap	Yes	Yes	Yes	Yes	Yes	-
uncheck	Yes	Yes	Yes	Yes	Yes	-
hover	Yes	Yes	Yes	Yes	-	-
scrollIntoViewIfNeeded	Yes	-	Yes	-	-	-
screenshot	Yes	Yes	Yes	-	-	-
fill	Yes	Yes	-	-	Yes	Yes
selectText	Yes	Yes	-	-	-	-
dispatchEvent	Yes	-	-	-	-	-
focus	Yes	-	-	-	-	-
getAttribute	Yes	-	-	-	-	-
innerText	Yes	-	-	-	-	-
innerHTML	Yes	-	-	-	-	-
press	Yes	-	-	-	-	-
setInputFiles	Yes	-	-	-	-	-
selectOption	Yes	Yes	-	-	Yes	-
textContent	Yes	-	-	-	-	-
type	Yes	-	-	-	-	-

2.4.1 Wait Function

- waitForEvent
- waitForFunction
- waitForLoadState
- waitForRequest
- waitForResponse
- waitForUrl

```
const { webkit } = require('playwright');  // Or 'chromium' or 'firefox'.

(async () => {
  const browser = await webkit.launch();
  const page = await browser.newPage();
  const watchDog = page.waitForFunction(() => window.innerWidth < 100);
  await page.setViewportSize({ width: 50, height: 50 });
  await watchDog;
  await browser.close();
})();
```

2.5 Assertion

Generic Assertion

- toBe
- toBeGreaterThan
- toBeLessThan
- toEqual
- toContain
- toMatch
- …

Locator Assertion

- toBeDisabled
- toBeChecked
- toBeFocused
- toContainText
- toHaveText
- toHaveValue
- …

Page Assertion

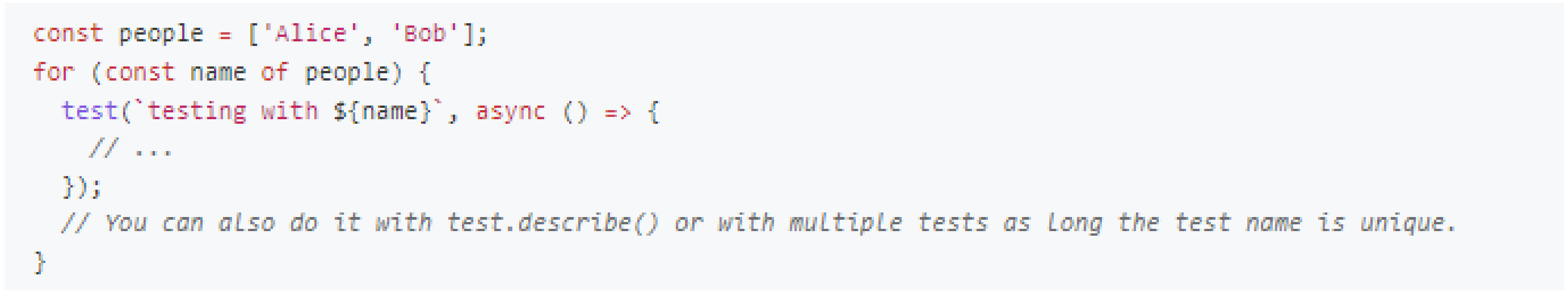
- toHaveScreenshot
- toHaveTitle
- toHaveUrl

SnapshotAssertions

- toMatchSnapshot

2.6 Data Driven

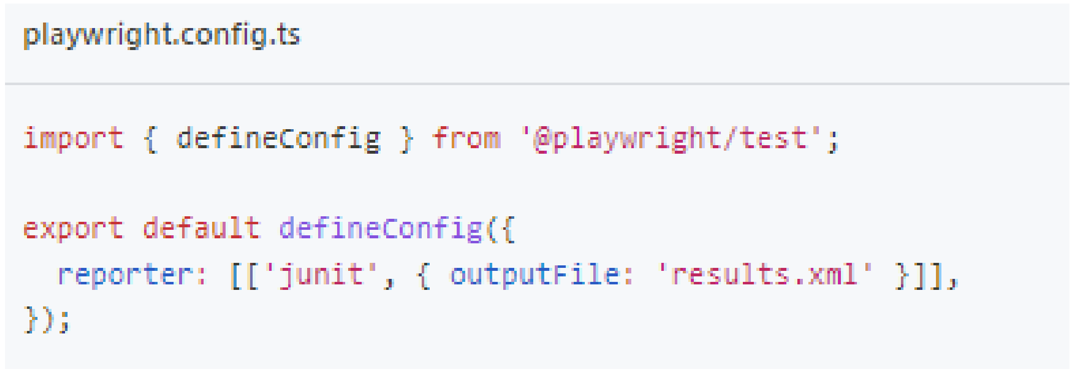
- Parameter Test
- Parameter Project
- Environment Variable



```
const people = ['Alice', 'Bob'];
for (const name of people) {
  test(`testing with ${name}`, async () => {
    // ...
  });
  // You can also do it with test.describe() or with multiple tests as long the test name is unique.
}
```

2.7 Report

- Built-in report: HTML, Json, Junit,…
- Allure report
- ReportPortal report

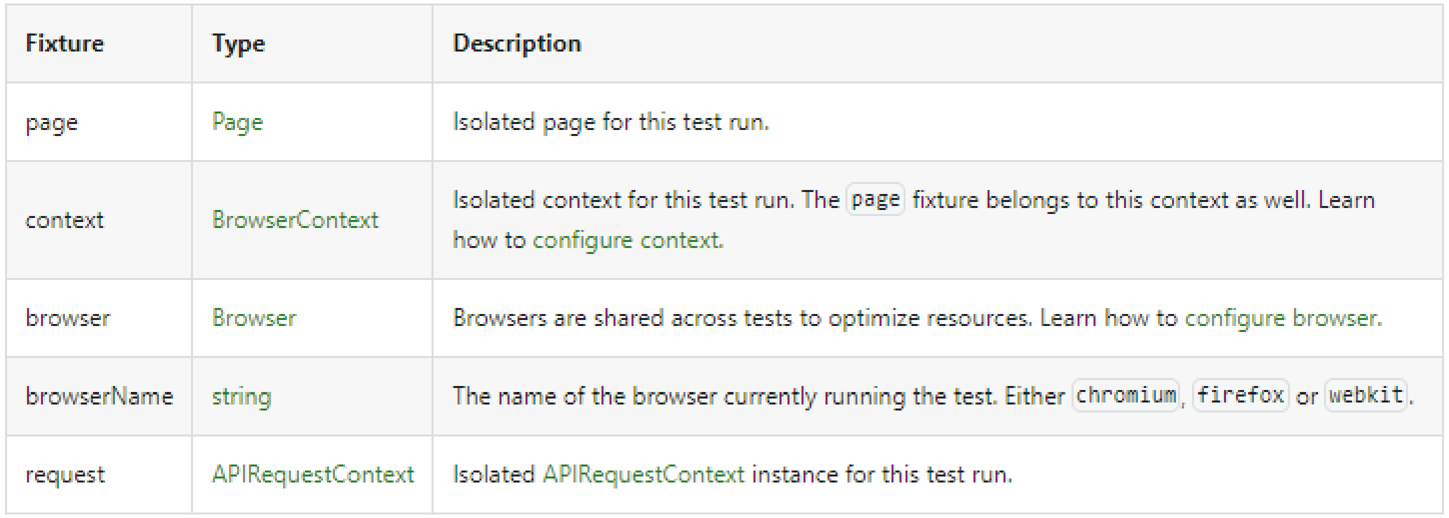


playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  reporter: [['junit', { outputFile: 'results.xml' }]],
});
```

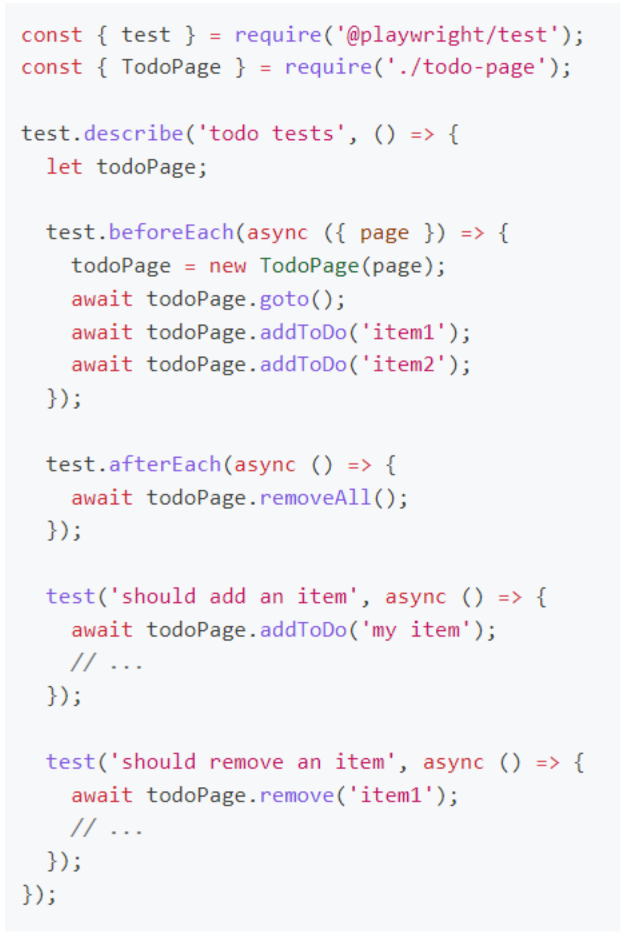
2.8 Fixtures

Test fixtures are used to establish environment for each test, giving the test everything it needs and nothing else. Test fixtures are isolated between tests. With fixtures, you can group tests based on their meaning, instead of their common setup.



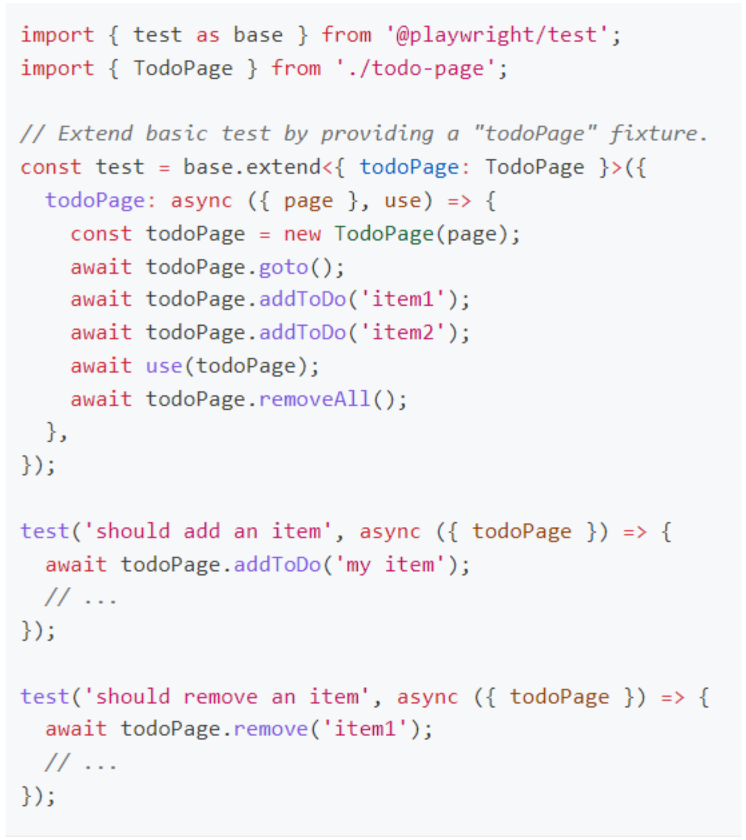
Fixture	Type	Description
page	Page	Isolated page for this test run.
context	BrowserContext	Isolated context for this test run. The `page` fixture belongs to this context as well. Learn how to configure context.
browser	Browser	Browsers are shared across tests to optimize resources. Learn how to configure browser.
browserName	string	The name of the browser currently running the test. Either `chromium`, `firefox` or `webkit`.
request	APIRequestContext	Isolated APIRequestContext instance for this test run.

2.8 Fixtures



```
const { test } = require('@playwright/test');
const { TodoPage } = require('./todo-page');

test.describe('todo tests', () => {
  let todoPage;

  test.beforeEach(async ({ page }) => {
    todoPage = new TodoPage(page);
    await todoPage.goto();
    await todoPage.addToDo('item1');
    await todoPage.addToDo('item2');
  });

  test.afterEach(async () => {
    await todoPage.removeAll();
  });

  test('should add an item', async () => {
    await todoPage.addToDo('my item');
    // ...
  });

  test('should remove an item', async () => {
    await todoPage.remove('item1');
    // ...
  });
});
```

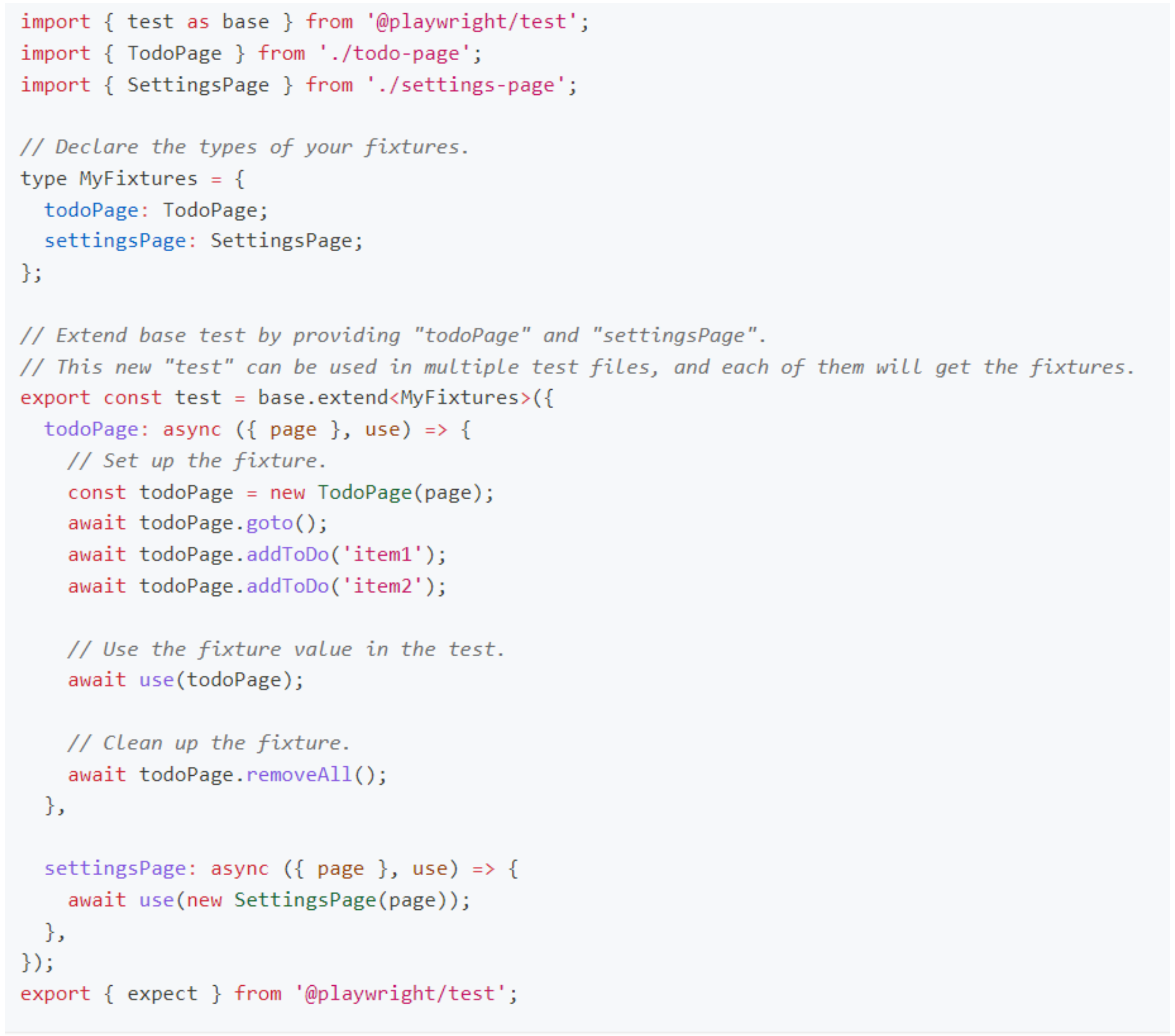
Without fixtures



```
import { test as base } from '@playwright/test';
import { TodoPage } from './todo-page';

// Extend basic test by providing a "todoPage" fixture.
const test = base.extend<{ todoPage: TodoPage }>({
  todoPage: async ({ page }, use) => {
    const todoPage = new TodoPage(page);
    await todoPage.goto();
    await todoPage.addToDo('item1');
    await todoPage.addToDo('item2');
    await use(todoPage);
    await todoPage.removeAll();
  },
});

test('should add an item', async ({ todoPage }) => {
  await todoPage.addToDo('my item');
  // ...
});

test('should remove an item', async ({ todoPage }) => {
  await todoPage.remove('item1');
  // ...
});
```

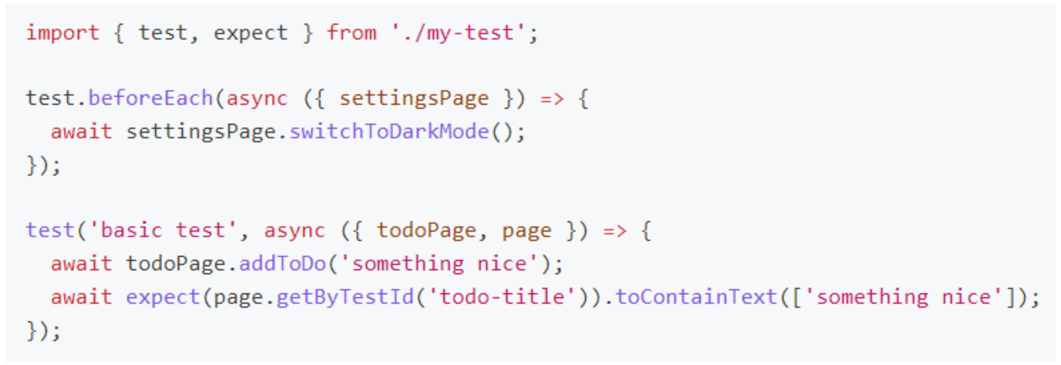
With fixtures

2.8 Fixtures



```
import { test as base } from '@playwright/test';
import { TodoPage } from './todo-page';
import { SettingsPage } from './settings-page';

// Declare the types of your fixtures.
type MyFixtures = {
  todoPage: TodoPage;
  settingsPage: SettingsPage;
};

// Extend base test by providing "todoPage" and "settingsPage".
// This new "test" can be used in multiple test files, and each of them will get the fixtures.
export const test = base.extend<MyFixtures>({
  todoPage: async ({ page }, use) => {
    // Set up the fixture.
    const todoPage = new TodoPage(page);
    await todoPage.goto();
    await todoPage.addToDo('item1');
    await todoPage.addToDo('item2');

    // Use the fixture value in the test.
    await use(todoPage);

    // Clean up the fixture.
    await todoPage.removeAll();
  },

  settingsPage: async ({ page }, use) => {
    await use(new SettingsPage(page));
  },
});
export { expect } from '@playwright/test';
```

Define fixtures

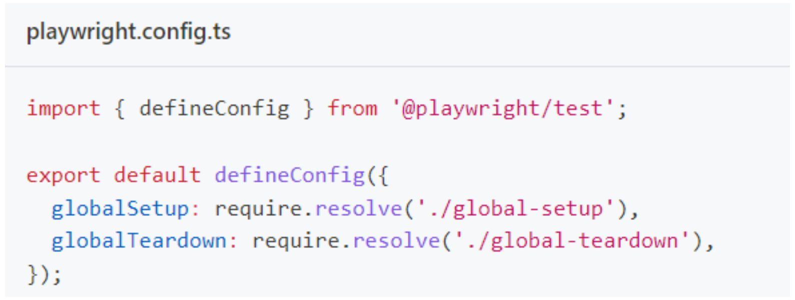


```
import { test, expect } from './my-test';

test.beforeEach(async ({ settingsPage }) => {
  await settingsPage.switchToDarkMode();
});

test('basic test', async ({ todoPage, page }) => {
  await todoPage.addToDo('something nice');
  await expect(page.getByTestId('todo-title')).toContainText(['something nice']);
});
```

Use fixtures

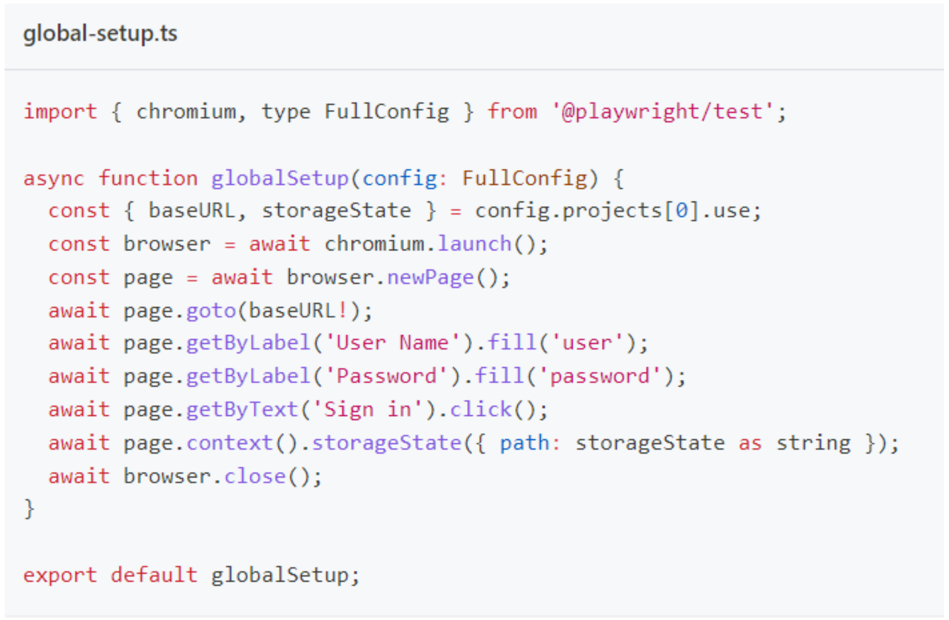
2.9 Test Hook

- beforeAll
- beforeEach
- globalSetup
- afterAll
- afterEach
- globalTeardown



playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  globalSetup: require.resolve('./global-setup'),
  globalTeardown: require.resolve('./global-teardown'),
});
```

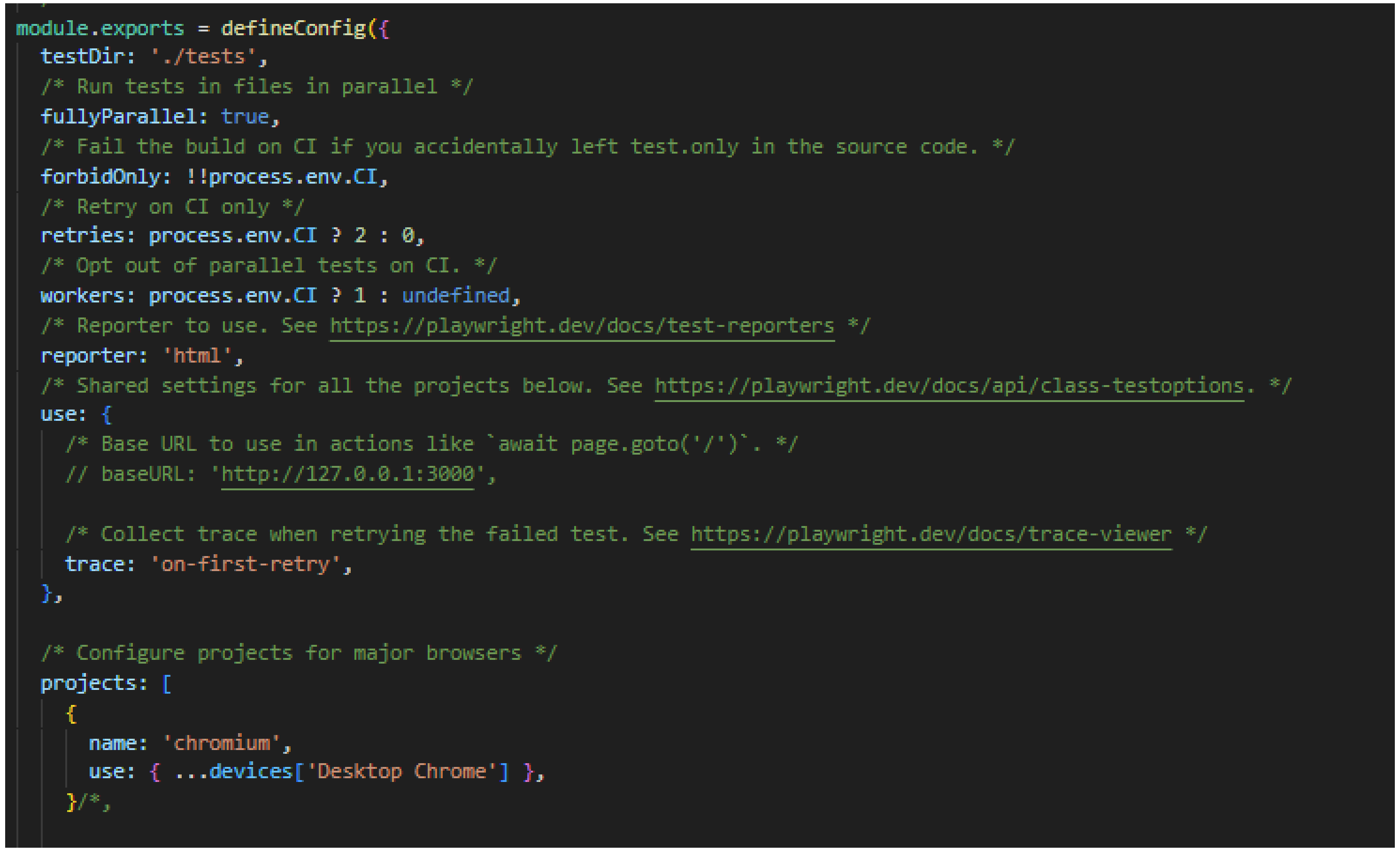


global-setup.ts

```
import { chromium, type FullConfig } from '@playwright/test';

async function globalSetup(config: FullConfig) {
  const { baseURL, storageState } = config.projects[0].use;
  const browser = await chromium.launch();
  const page = await browser.newPage();
  await page.goto(baseURL!);
  await page.getByLabel('User Name').fill('user');
  await page.getByLabel('Password').fill('password');
  await page.getByText('Sign in').click();
  await page.context().storageState({ path: storageState as string });
  await browser.close();
}

export default globalSetup;
```

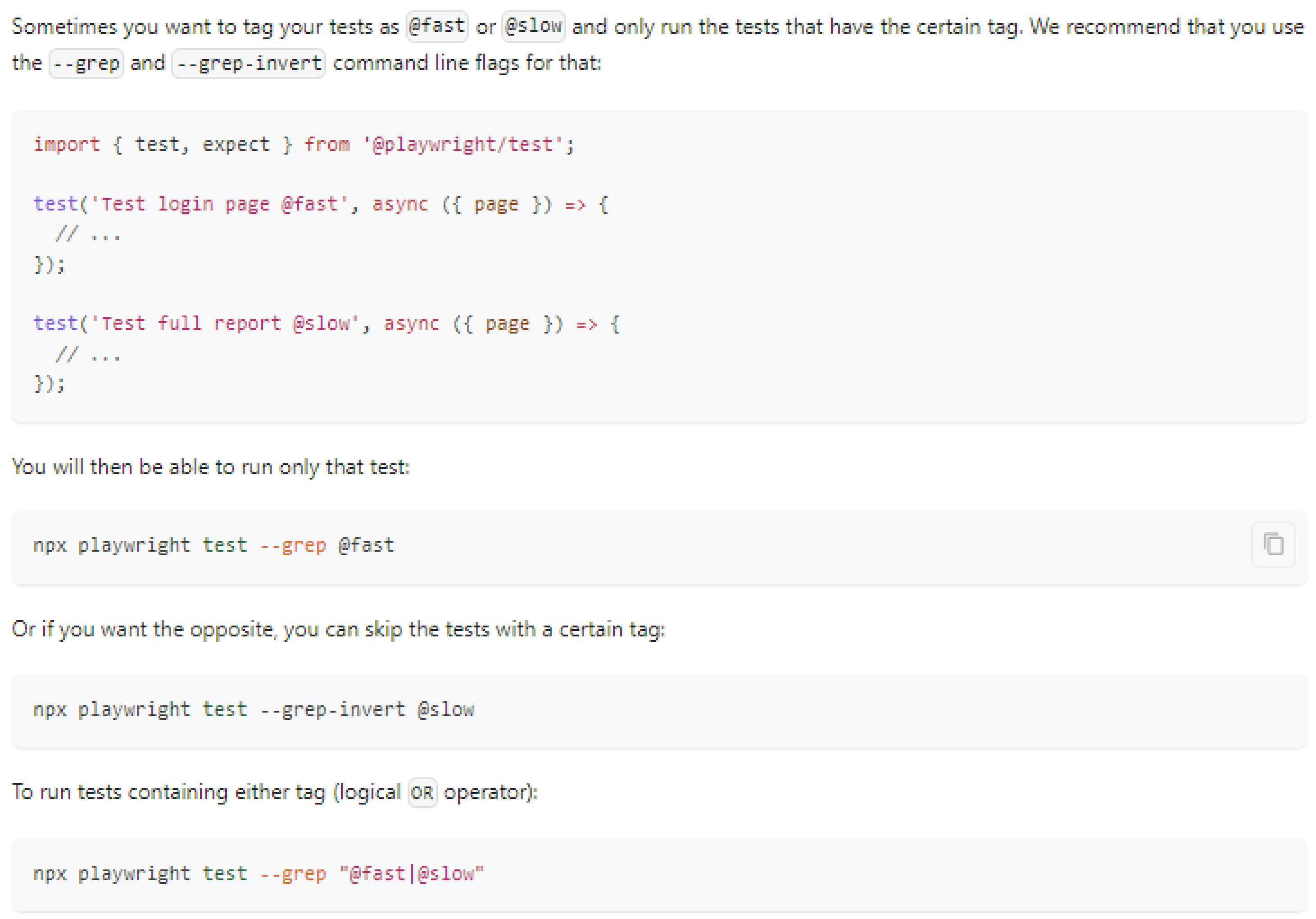
2.10 Playwright configuration

- Set test directory
- Running test in parallel
- Choose browser
- Set report type
- Global Setup/ Teardown



```
module.exports = defineConfig({
  testDir: './tests',
  /* Run tests in files in parallel */
  fullyParallel: true,
  /* Fail the build on CI if you accidentally left test.only in the source code. */
  forbidOnly: !!process.env.CI,
  /* Retry on CI only */
  retries: process.env.CI ? 2 : 0,
  /* Opt out of parallel tests on CI. */
  workers: process.env.CI ? 1 : undefined,
  /* Reporter to use. See https://playwright.dev/docs/test-reporters */
  reporter: 'html',
  /* Shared settings for all the projects below. See https://playwright.dev/docs/api/class-testoptions. */
  use: {
    /* Base URL to use in actions like `await page.goto('/')`. */
    // baseURL: 'http://127.0.0.1:3000',

    /* Collect trace when retrying the failed test. See https://playwright.dev/docs/trace-viewer */
    trace: 'on-first-retry',
  },

  /* Configure projects for major browsers */
  projects: [
    {
      name: 'chromium',
      use: { ...devices['Desktop Chrome'] },
    }/*,
```

2.11 Filter test case for running test



Sometimes you want to tag your tests as `@fast` or `@slow` and only run the tests that have the certain tag. We recommend that you use the `--grep` and `--grep-invert` command line flags for that:

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

test('Test login page @fast', async ({ page }) => {
  // ...
});

test('Test full report @slow', async ({ page }) => {
  // ...
});
```

You will then be able to run only that test:

```
npx playwright test --grep @fast
```

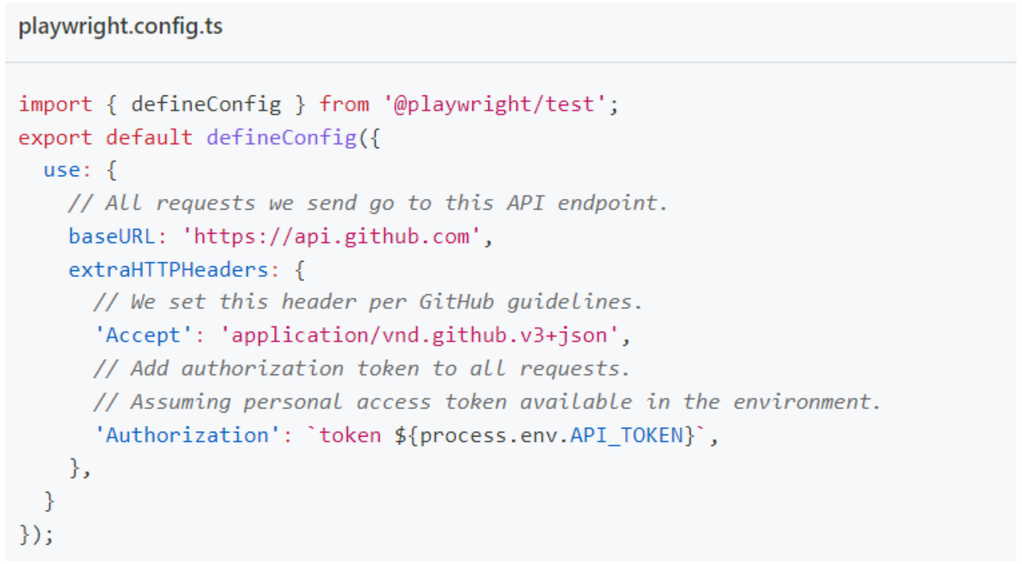
Or if you want the opposite, you can skip the tests with a certain tag:

```
npx playwright test --grep-invert @slow
```

To run tests containing either tag (logical `OR` operator):

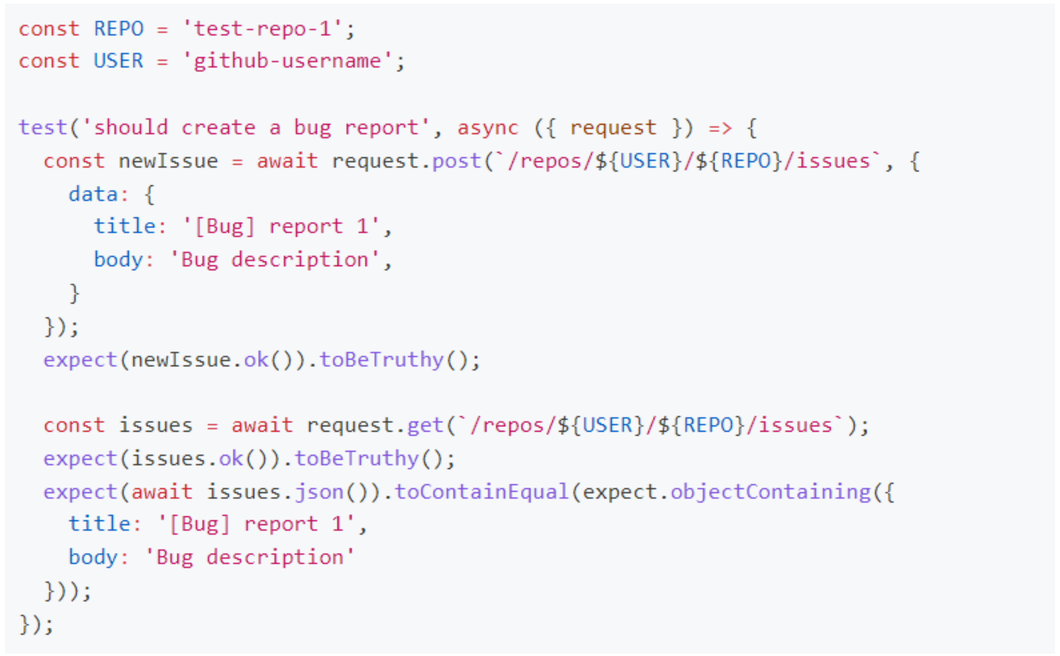
```
npx playwright test --grep "@fast|@slow"
```

2.12 API Request



playwright.config.ts

```
import { defineConfig } from '@playwright/test';
export default defineConfig({
  use: {
    // All requests we send go to this API endpoint.
    baseURL: 'https://api.github.com',
    extraHTTPHeaders: {
      // We set this header per GitHub guidelines.
      'Accept': 'application/vnd.github.v3+json',
      // Add authorization token to all requests.
      // Assuming personal access token available in the environment.
      'Authorization': `token ${process.env.API_TOKEN}`,
    },
  }
});
```



```
const REPO = 'test-repo-1';
const USER = 'github-username';

test('should create a bug report', async ({ request }) => {
  const newIssue = await request.post(`/repos/${USER}/${REPO}/issues`, {
    data: {
      title: '[Bug] report 1',
      body: 'Bug description',
    }
  });
  expect(newIssue.ok()).toBeTruthy();

  const issues = await request.get(`/repos/${USER}/${REPO}/issues`);
  expect(issues.ok()).toBeTruthy();
  expect(await issues.json()).toContainEqual(expect.objectContaining({
    title: '[Bug] report 1',
    body: 'Bug description'
  }));
});
```

2.13 Playwright commands

npx playwright test	⇒ Run all the tests in all projects
npx playwright test tests/todo-page.spec.ts	⇒ Run a single test file
npx playwright test tests/todo-page/ tests/landing-page/	⇒ Run a set of test files
npx playwright test my-spec my-spec-2	⇒ Run files that have my-spec or my-spec-2 in the file name
npx playwright test -g "add a todo item“	⇒ Run the test with the title
npx playwright test –headed	⇒ Run tests in headed browsers
npx playwright test --project=firefox,chrome	⇒ Run tests in particular configuration (project)
npx playwright test --workers=1	⇒ Disable parallelization, run test with 1 worker
npx playwright test --reporter=dot	⇒ Run test with dot reporter
npx playwright test –debug	⇒ Run in debug mode with Playwright Inspector
npx playwright test --help	⇒ Ask for help

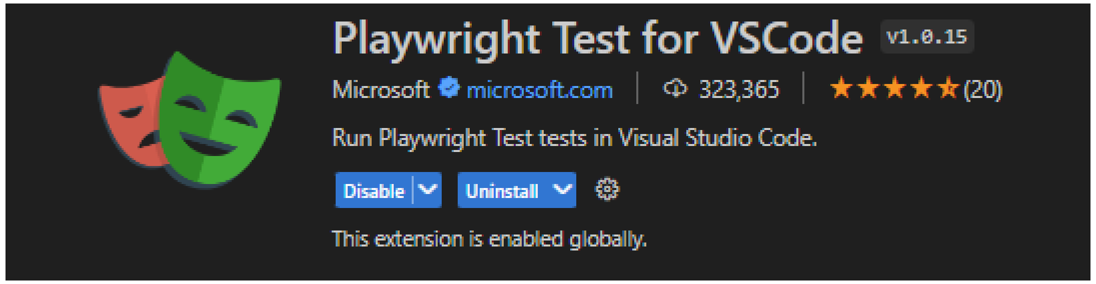
Playwright additional features

3. Playwright additional features

1. VS Code Extension
2. Playwright Tracing
3. Networks
4. Emulation

3.1 VS Code Extension

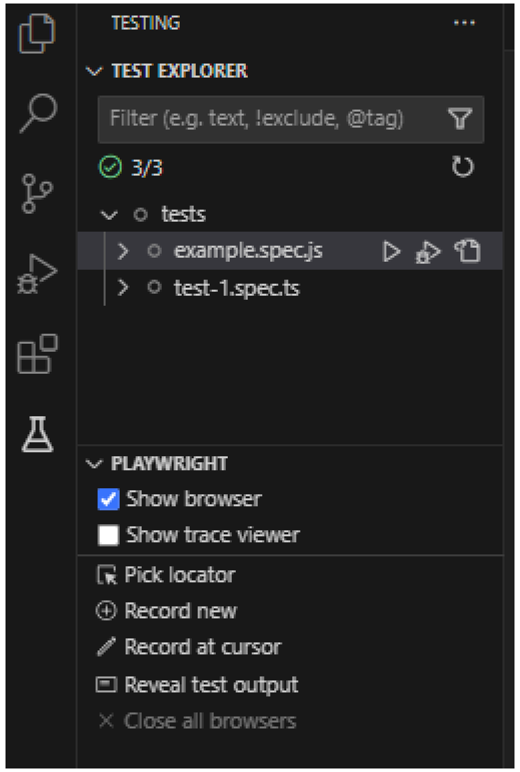
- Running single/multiple test
- Select browser
- Pick locator
- Debug
- Record test case


Playwright Test for VSCode v1.0.15

Microsoft microsoft.com | 323,365 | ★★★★★ (20)

Run Playwright Test tests in Visual Studio Code.

This extension is enabled globally.


TESTING

TEST EXPLORER

Filter (e.g. text, !exclude, @tag)

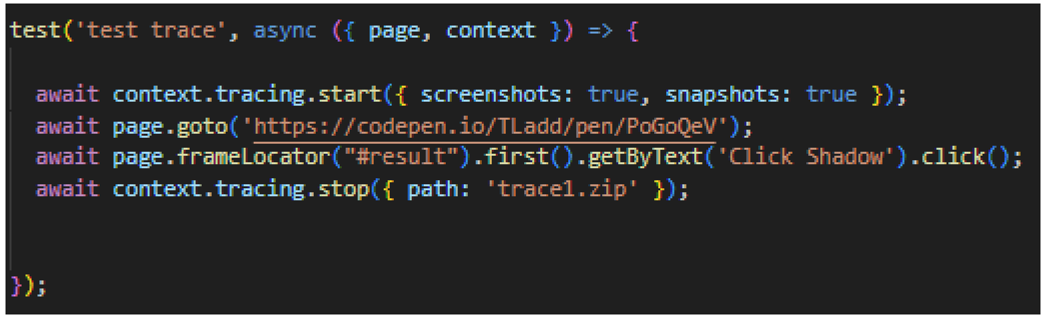
3/3

tests
- example.spec.js
- test-1.spec.ts

PLAYWRIGHT

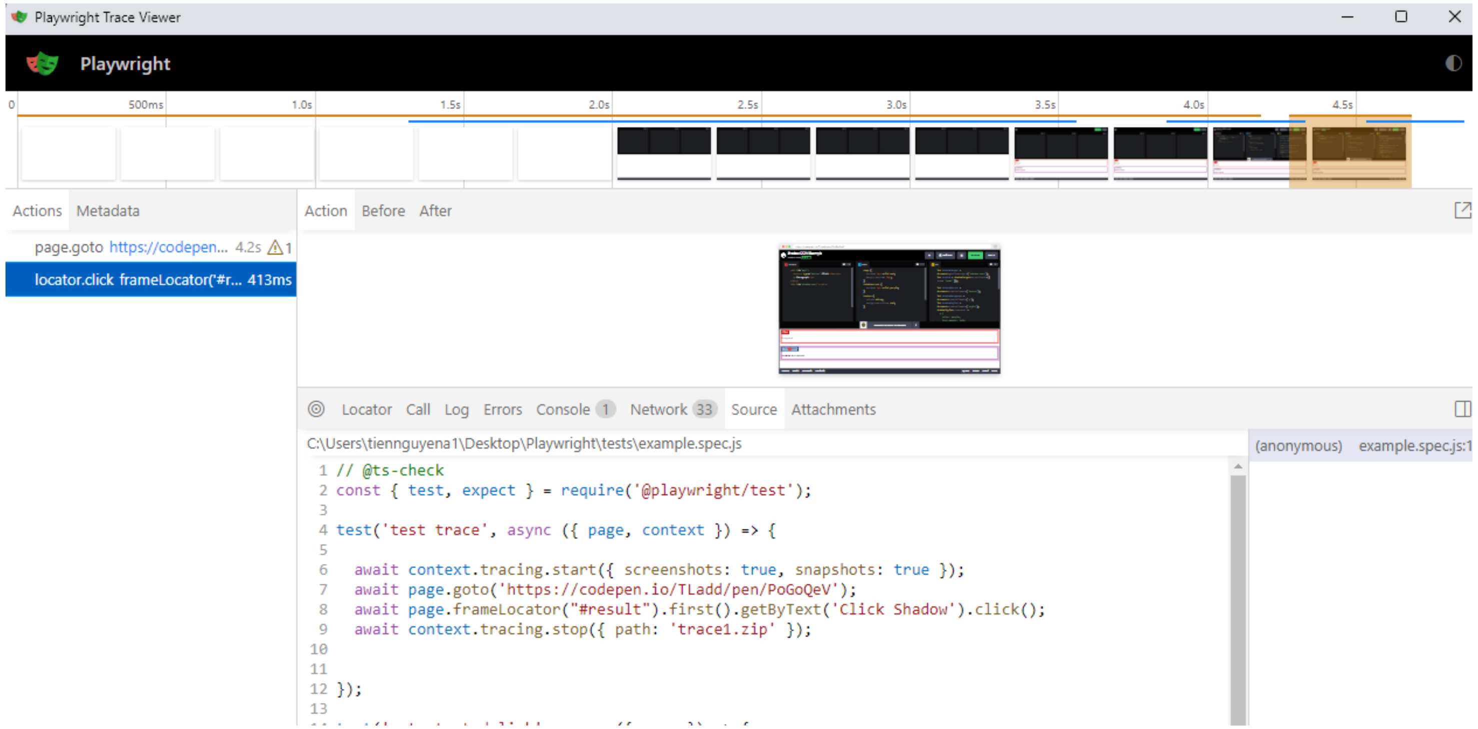
- [✓] Show browser
- [ ] Show trace viewer

3.2 Playwright Tracing



```
test('test trace', async ({ page, context }) => {

  await context.tracing.start({ screenshots: true, snapshots: true });
  await page.goto('https://codepen.io/TLadd/pen/PoGoQeV');
  await page.frameLocator("#result").first().getByText('Click Shadow').click();
  await context.tracing.stop({ path: 'trace1.zip' });


});
```

Useful tool for debugging



```
page.goto https://codepen... 4.2s ⚠1
locator.click frameLocator('#r... 413ms
```

C:\Users\tiennguyena1\Desktop\Playwright\tests\example.spec.js

```
// @ts-check
const { test, expect } = require('@playwright/test');

test('test trace', async ({ page, context }) => {

  await context.tracing.start({ screenshots: true, snapshots: true });
  await page.goto('https://codepen.io/TLadd/pen/PoGoQeV');
  await page.frameLocator("#result").first().getByText('Click Shadow').click();
  await context.tracing.stop({ path: 'trace1.zip' });


});

```

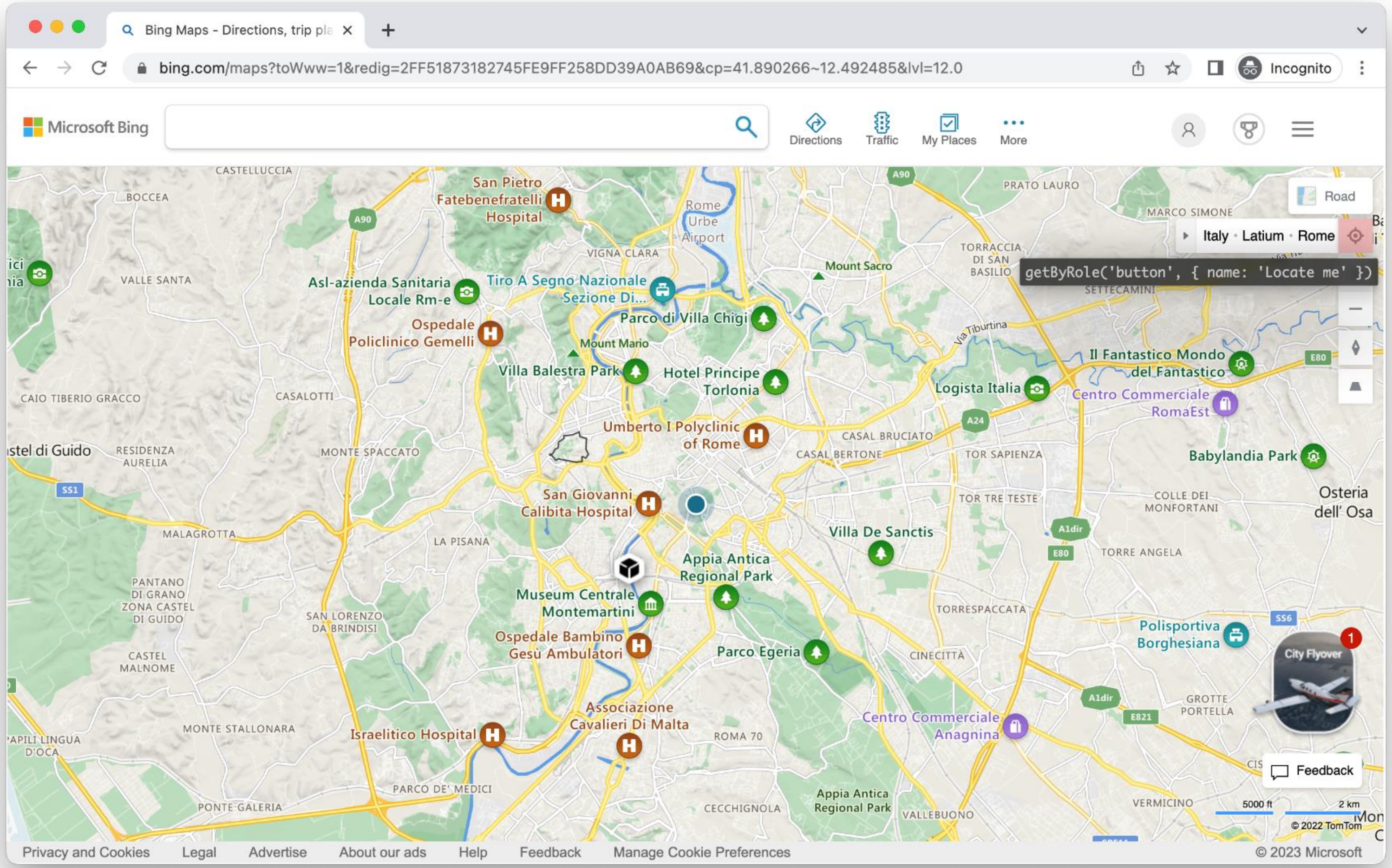
3.3 Networks

- Mocking API
- Authentication
- Proxy
- Modify request
- Abort request
- Modify response
- …

```
// Delete header
await page.route('**/*', route => {
  const headers = route.request().headers();
  delete headers['X-Secret'];
  route.continue({ headers });
});

// Continue requests as POST.
await page.route('**/*', route => route.continue({ method: 'POST' }));
```

3.4 Emulation

- Device
- Viewport
- Locale & Time zone
- Permission
- Geolocation
- Color schema and media



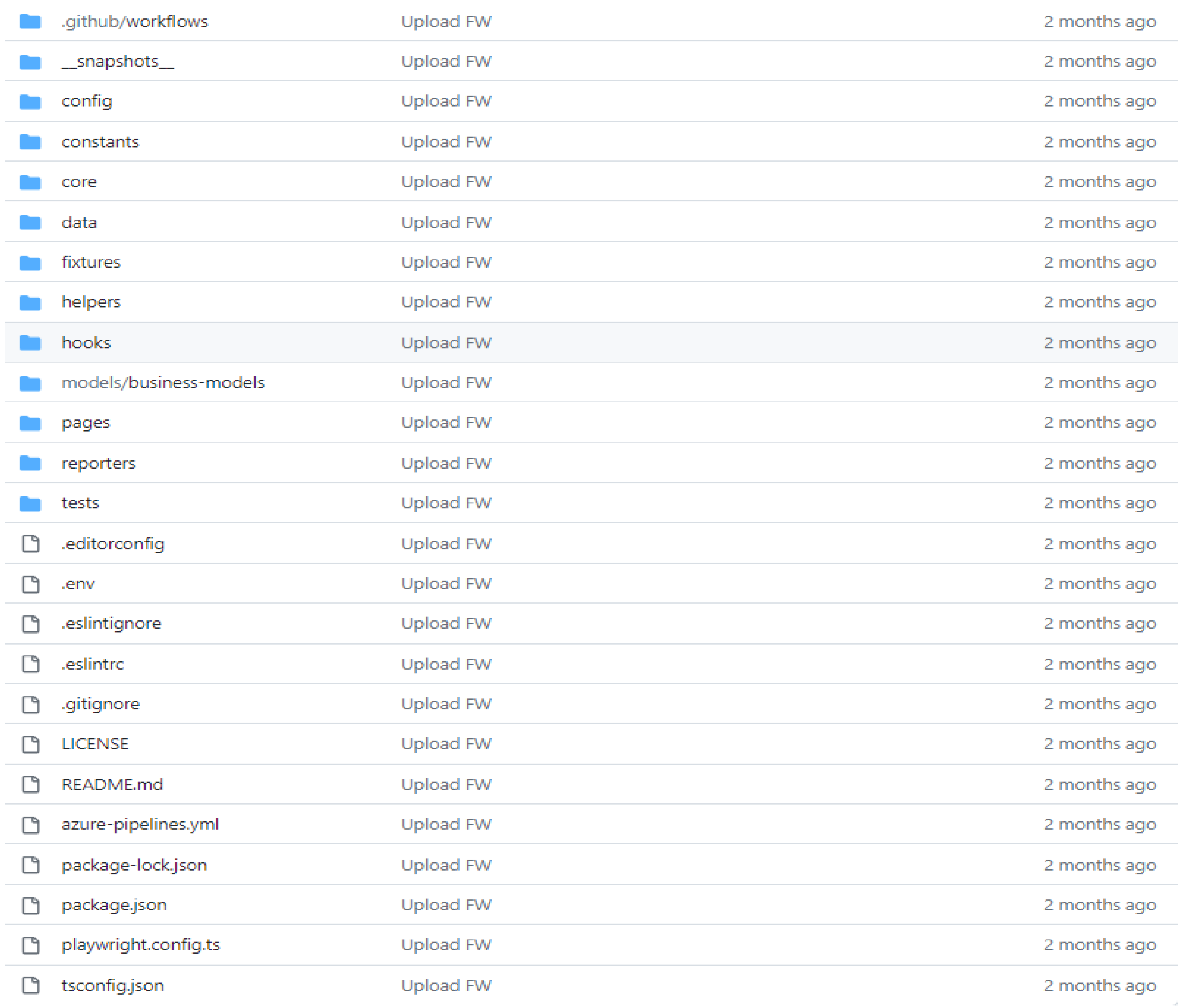
bing.com/maps?toWww=1&redig=2FF51873182745FE9FF258DD39A0AB69&cp=41.890266~12.492485&lvl=12.0

```
getByRole('button', { name: 'Locate me' })
```

Bing Maps - Directions, trip pla
Incognito
Microsoft Bing
Directions
Traffic
My Places
More
Road
Italy · Latium · Rome
San Pietro Fatebenefratelli Hospital
Rome Urbe Airport
Asl-azienda Sanitaria Locale Rm-e
Tiro A Segno Nazionale Sezione Di...
Parco di Villa Chigi
Mount Sacro
Ospedale Policlinico Gemelli
Mount Mario
Villa Balestra Park
Hotel Principe Torlonia
Il Fantastico Mondo del Fantastico
Logista Italia
Centro Commerciale RomaEst
Umberto I Polyclinic of Rome
Babylandia Park
San Giovanni Calibita Hospital
Osteria dell' Osa
Villa De Sanctis
Appia Antica Regional Park
Museum Centrale Montemartini
Ospedale Bambino Gesù Ambulatori
Parco Egeria
Polisportiva Borghesiana
City Flyover
Associazione Cavalieri Di Malta
Israelitico Hospital
Centro Commerciale Anagnina
Feedback
© 2022 TomTom
Privacy and Cookies
Legal
Advertise
About our ads
Help
Feedback
Manage Cookie Preferences
© 2023 Microsoft

Framework Template

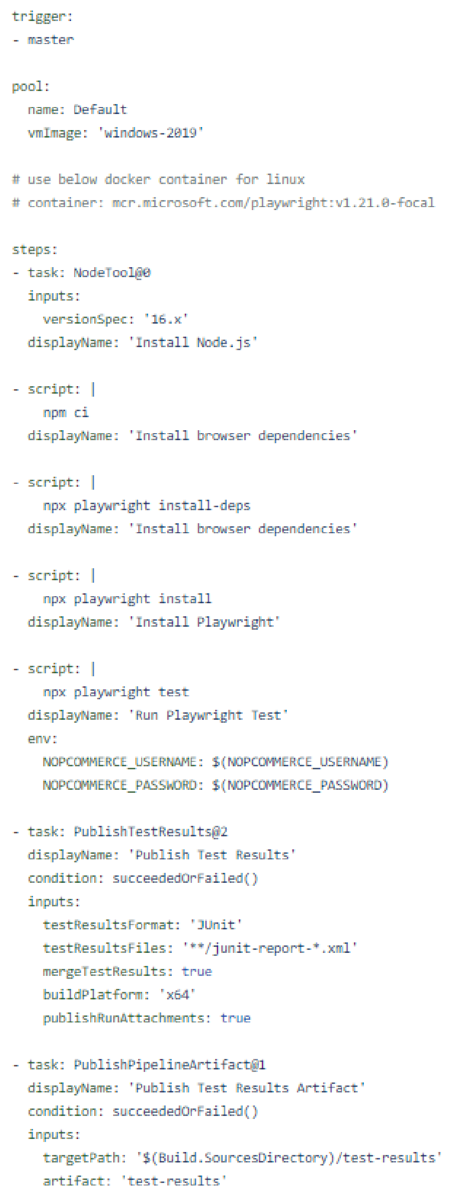
4. Framework Template



.github/workflows	Upload FW	2 months ago
__snapshots__	Upload FW	2 months ago
config	Upload FW	2 months ago
constants	Upload FW	2 months ago
core	Upload FW	2 months ago
data	Upload FW	2 months ago
fixtures	Upload FW	2 months ago
helpers	Upload FW	2 months ago
hooks	Upload FW	2 months ago
models/business-models	Upload FW	2 months ago
pages	Upload FW	2 months ago
reporters	Upload FW	2 months ago
tests	Upload FW	2 months ago
.editorconfig	Upload FW	2 months ago
.env	Upload FW	2 months ago
.eslintignore	Upload FW	2 months ago
.eslintrc	Upload FW	2 months ago
.gitignore	Upload FW	2 months ago
LICENSE	Upload FW	2 months ago
README.md	Upload FW	2 months ago
azure-pipelines.yml	Upload FW	2 months ago
package-lock.json	Upload FW	2 months ago
package.json	Upload FW	2 months ago
playwright.config.ts	Upload FW	2 months ago
tsconfig.json	Upload FW	2 months ago

4. Framework Template

Azure DevOps yml file



```yaml
trigger:
- master

pool:
  name: Default
  vmImage: 'windows-2019'

# use below docker container for linux
# container: mcr.microsoft.com/playwright:v1.21.0-focal

steps:
- task: NodeTool@0
  inputs:
    versionSpec: '16.x'
  displayName: 'Install Node.js'

- script: |
    npm ci
  displayName: 'Install browser dependencies'

- script: |
    npx playwright install-deps
  displayName: 'Install browser dependencies'

- script: |
    npx playwright install
  displayName: 'Install Playwright'

- script: |
    npx playwright test
  displayName: 'Run Playwright Test'
  env:
    NOPCOMMERCE_USERNAME: $(NOPCOMMERCE_USERNAME)
    NOPCOMMERCE_PASSWORD: $(NOPCOMMERCE_PASSWORD)

- task: PublishTestResults@2
  displayName: 'Publish Test Results'
  condition: succeededOrFailed()
  inputs:
    testResultsFormat: 'JUnit'
    testResultsFiles: '**/junit-report-*.xml'
    mergeTestResults: true
    buildPlatform: 'x64'
    publishRunAttachments: true

- task: PublishPipelineArtifact@1
  displayName: 'Publish Test Results Artifact'
  condition: succeededOrFailed()
  inputs:
    targetPath: '$(Build.SourcesDirectory)/test-results'
    artifact: 'test-results'
```

Reference

- https://playwright.dev/docs/intro
- https://www.programsbuzz.com/article/playwright-architecture

5. Exercise

- NT_TD_001_Template_FinalExamination_CourseCode_AutomationWithPlaywright.docx

Thank you

Nash Tech.