# PLAYWRIGHT

## Hands On

@chonla

v1.1.1

# OUTLINE

**CHECK-IN**
System Under Test, **Twittah!**

**GET STARTED**
Setup Playwright

**FOUNDATION**
Test Structure

**WEB TESTINGS**
Playwright API, Locators, Actions, Assertions

**TRY IT YOURSELVES**
with **Twittah!**

**PRACTICES**
Sparkling Structures, Page Object Model, Data Driven Testings, Idempotent, Gherkin, CI/CD

# HELLO

**Twittah**

The popular social network look-alike.

# Setup PLAYWRIGHT

```
~/work/bayo/training/playwright/test (3m 11.19s)
npm init playwright@latest

Need to install the following packages:
  create-playwright@1.17.129
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? · TypeScript
✔ Where to put your end-to-end tests? · tests
✔ Add a GitHub Actions workflow? (y/N) · true
✔ Install Playwright browsers (can be done manually via 'npx playwright install')? (Y/n) · true
```


```
✔ Do you want to use TypeScript or JavaScript? · TypeScript
✔ Where to put your end-to-end tests? · tests
✔ Add a GitHub Actions workflow? (y/N) · true
✔ Install Playwright browsers (can be done manually via 'npx playwright install')? (Y/n) · true
```


https://github.com/chonla/twittah-test [main]

# Run PLAYWRIGHT

```
npx playwright test

Running 6 tests using 5 workers
  6 passed (9.7s)

To open last HTML report run:

  npx playwright show-report
```

example.spec.ts

| | | |
|---|---|---|
| ✓ has title chromium<br>example.spec.ts:3 | 931ms |
| ✓ get started link chromium<br>example.spec.ts:10 | 1.4s |
| ✓ has title firefox<br>example.spec.ts:3 | 936ms |
| ✓ get started link firefox | 1.2s |

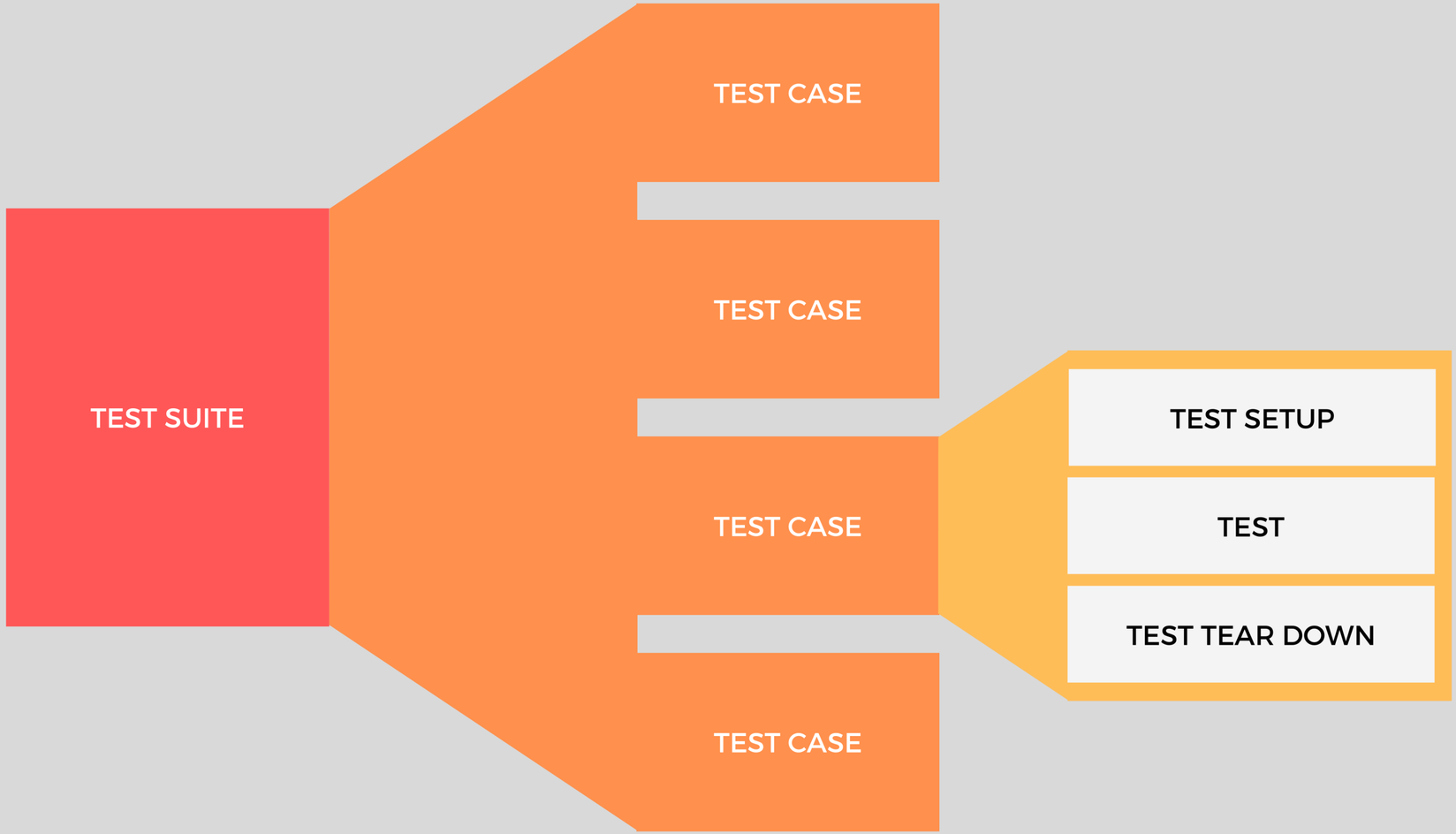
# Configure Browsers

playwright.config.ts

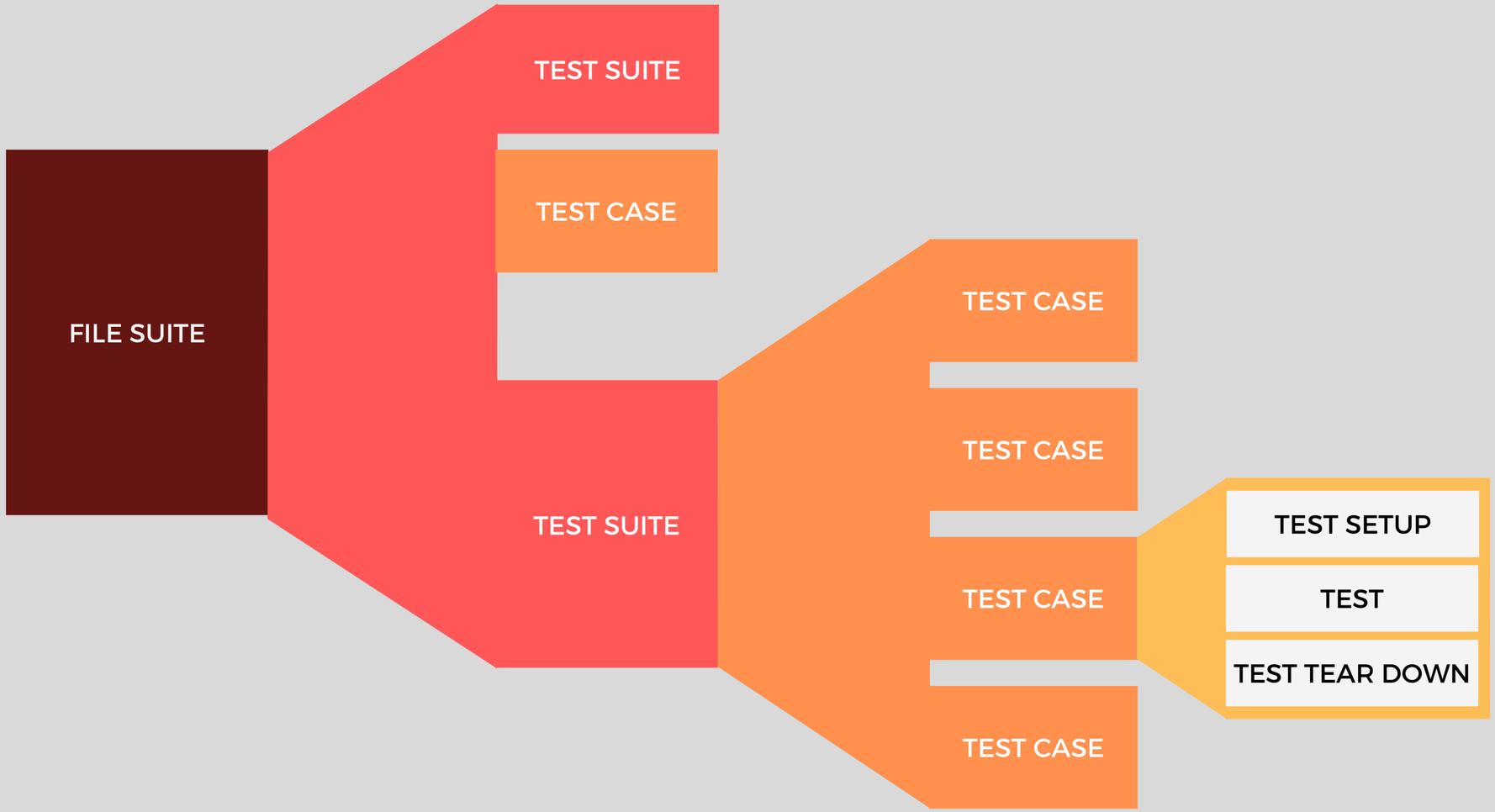
playwright.config.ts > default

```ts
    /* 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'] },
    },

```

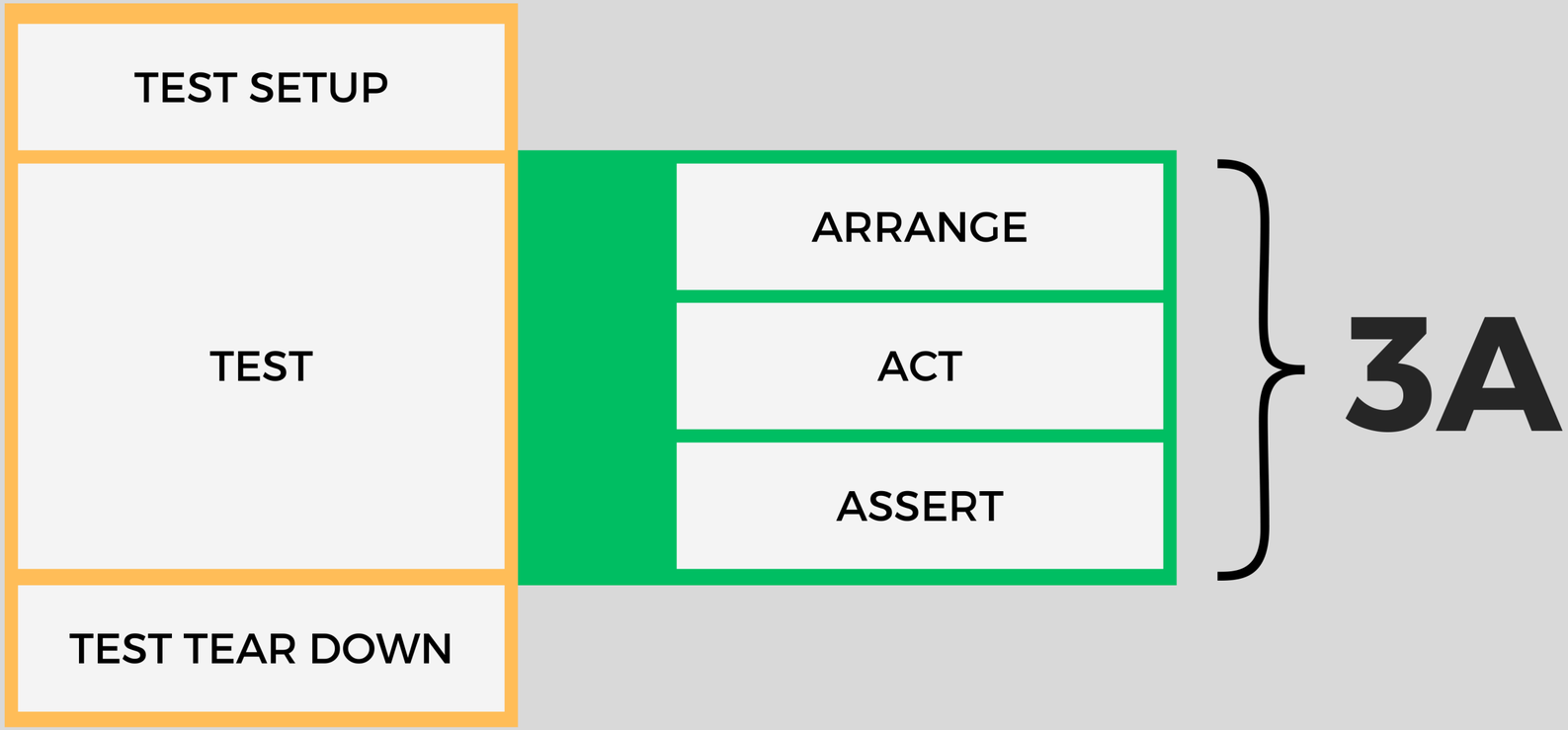
# Test Structures



# PLAYWRIGHT's Structures



# Test Case Anatomy



# Tests in PLAYWRIGHT

tests-examples/demo-todo-app.spec.ts

tests-examples > demo-todo-app.spec.ts > test.describe('New Todo') callback

```
1   import { test, expect, type Page } from '@playwright/test';
2
3   test.beforeEach(async ({ page }) => {
4     await page.goto('https://demo.playwright.dev/todomvc');
5   });
6
7   const TODO_ITEMS = [
8     'buy some cheese',
9     'feed the cat',
10    'book a doctors appointment'
11  ];
12
13  test.describe('New Todo', () => {
14    test('should allow me to add todo items', async ({ page }) => {
15      // create a new todo locator
16      const newTodo = page.getByPlaceholder('What needs to be done?');
17
```

TEST SETUP

TEST SUITE

TEST CASE

# 3A in PLAYWRIGHT

tests/example.spec.ts

```ts
import { test, expect } from '@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();
});
```

TEST CASE

ARRANGE

ACT

ASSERT

# Twittah! revisited #1

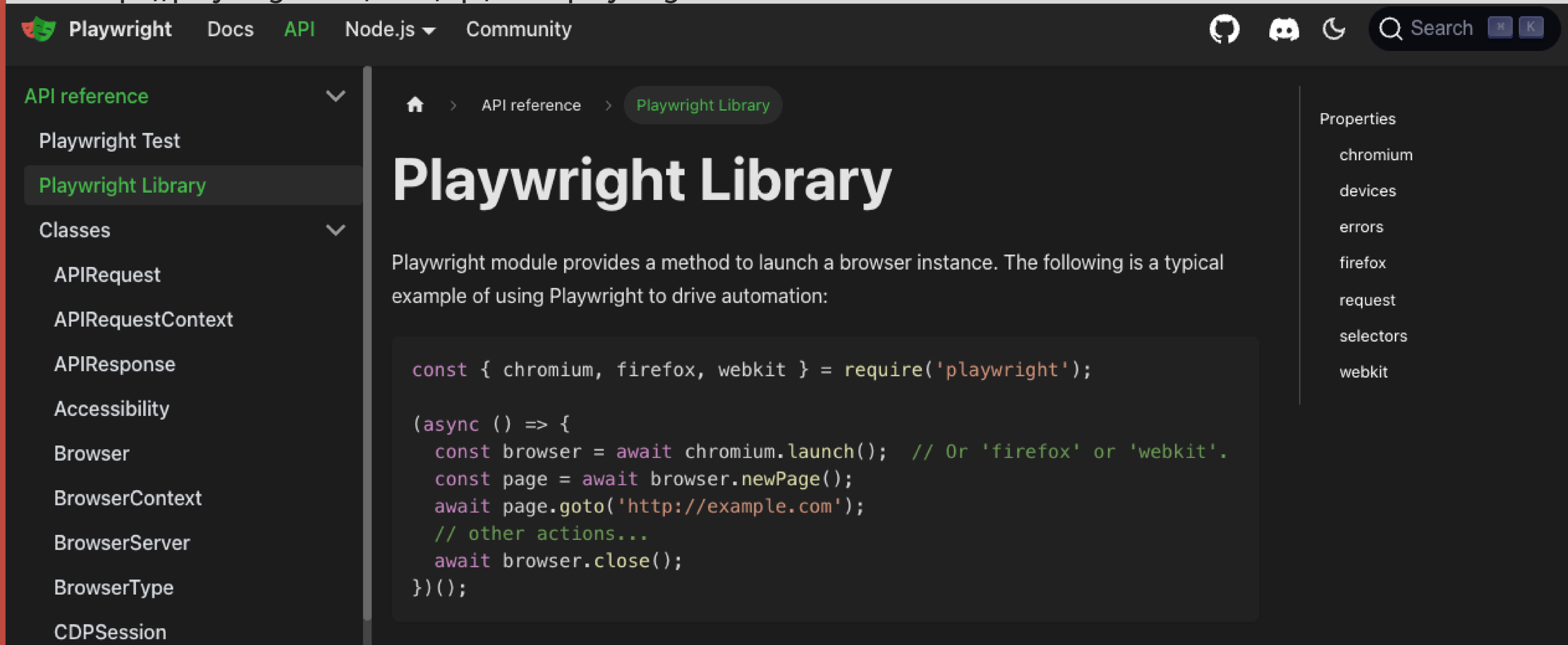
tests/twittah.spec.ts

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

test('Visit Twittah!', async ({ page }) => {
  await page.goto('https://twittah.web.app');
  await expect(page.getByTestId('app-name')).toBeVisible();
  await expect(page.getByTestId('app-name')).toHaveText('Twittah!');
});
```

https://github.com/chonla/twittah-test [visit]

# PLAYWRIGHT API

https://playwright.dev/docs/api/class-playwright



# LOCATORS

```
page.getByRole('role-name');
page.getByText('text');
page.getByLabel('label');
page.getByPlaceholder('placeholder');
page.getByAltText('alt-text');
page.getByTitle('title-text');
page.getByTestId('test-id');
```

They are asynchronous functions. Use with “await” when needed..

```
await page.getByRole('link', { name: 'Get started' }).click();
```

# .getByRole()

Explicit role is defined with attribute “role”, implicit role is defined in W3C spec.

```
<a href="/" role="menu-item">หน้าแรก</a>
<a href="/profile">โปรไฟล์</a>
<a (click)="doLogout()">ล็อกเอาท์</a>
```

role=”menu-item”

role=”link”

role=”generic”

https://playwright.dev/docs/api/class-page#page-get-by-role

# .getByText()

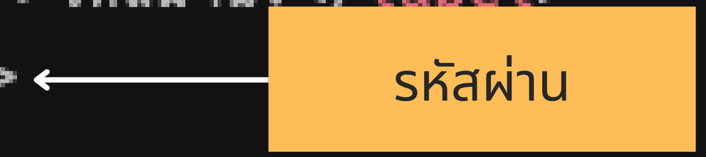
Text, AKA innerText

```
<div>สวัสดี</div>
<div>สวัสดี <span>คุณบัญชา</span></div>
```


https://playwright.dev/docs/api/class-page#page-get-by-text

# .getByLabel()

```
<input aria-label="ล็อกอิน">

<label for="password-field">รหัสผ่าน:</label>
<input id="password-field">
```


https://playwright.dev/docs/api/class-page#page-get-by-label

# .getByPlaceholder()

```
  <div class="control">
    <input type="text" class="input" placeholder="บัญชี Twittah">
  </div>
</div>
<div class="field">
  <label class="label">รหัสผ่าน</label>
  <div class="control">
    <input type="password" class="input"placeholder="รหัสผ่าน">
  </div>
</div>
```

บัญชี Twittah

รหัสผ่าน

https://playwright.dev/docs/api/class-page#page-get-by-placeholder

# .getByAltText()

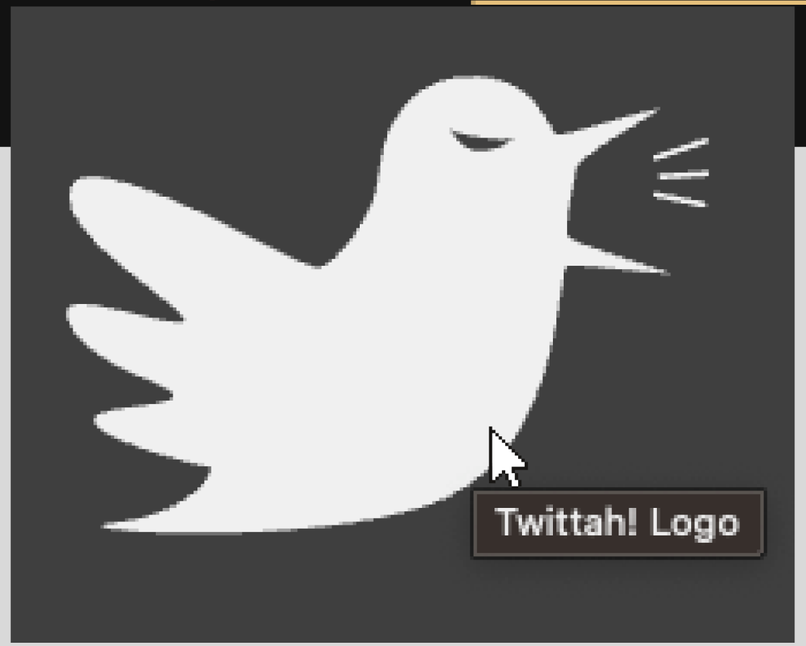
```
ll-height branding">
logo"><img src="./assets/logo.png" alt="Twittah! Logo"></div>
```


Twittah! Logo

https://playwright.dev/docs/api/class-page#page-get-by-alt-text

# .getByTitle()

```
l-height branding">
ogo"><img src="./assets/logo.png" title="Twittah! Logo"></div>
```




Twittah! Logo

https://playwright.dev/docs/api/class-page#page-get-by-title

# .getByTestId()

```
<div class="control">
  <input type="text" class="input" placeholder="บัญชี Twittah" data-testid="login-field">
</div>
```

login-field

HTML attribute prefixed by “data-*” is a custom attribute.

https://playwright.dev/docs/api/class-page#page-get-by-test-id

# .locator()

```
await this._page.locator('button').click();

await this._page.locator('css=button').click();

await this._page.locator('xpath=//body/div/div/div/div[2]/button').click();
```

XPath and CSS are very sensitive
since they are tied to UI layout which easily changes.

https://playwright.dev/docs/other-locators

# Locators with CodeGen

```
npx playwright codegen https://twittah.web.app
```

CodeGen tracks every actions.

Record | Target: Test Runner

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

test('test', async ({ page }) => {
  await page.goto('https://twittah.web.app/');
  await page.getByTestId('login-field').click();
  await page.getByTestId('login-field').fill('bancha');
  await page.getByTestId('login-field').press('Tab');
  await page.getByTestId('password-field').fill('123456');
  await page.getByTestId('password-field').press('Enter');
});
```

# Select items from locators

```
this._page.getByRole('button').filter({ hasText: 'Register' });

this._page.getByRole('button').and(this._page.getByText('Register'));

this._page.getByRole('button').or(this._page.getByText('Register'));

this._page.getByRole('button').first();

this._page.getByRole('button').last();

this._page.getByRole('button').nth(4);
```

https://playwright.dev/docs/api/class-locator#locator-filter

# Select child locators

```
this._page.getByRole('form').getByTestId(/share-button-/);
```

# Perform ACTIONS

```
await page.getByTestId('login-field').fill('bancha');
await page.getByTestId('password-field').fill('123456');
await page.getByTestId('login-button').click();
```

# ACTIONS

| | | | |
|---|---|---|---|
| **click** | Click the elements | **hover** | Move cursor to the elements |
| **check** | Check the elements | **uncheck** | Uncheck the elements |
| **fill** | Fill text to the elements | **clear** | Clear the element texts |
| **type** | Type each characters into the elements | **press** | Press one or multiple elements |
| **focus** | Focus the elements | **blur** | Leave the elements |
| **tap** | Tap the elements | | |

https://playwright.dev/docs/api/class-locator

# ASSERTIONS

To assert states of System Under Test from taken actions.

```
await expect(page.getByTestId('current-user-profile')).toBeVisible();
await expect(page.getByTestId('user-profile-display-name')).toHaveText('บัญชา');
await expect(page.getByTestId('user-profile-login-name')).toHaveText('@bancha');
expect(page.url()).toEqual(`https://twittah.web.app/home`);
```

# ASSERTIONS

## APIResponseAssertions

API Requests are also Async.

Playwright can do simple assertions to API call.

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

test('navigates to login', async ({ page }) => {
  // ...
  const response = await page.request.get('https://playwright.dev');
  await expect(response).toBeOK();
});
```

https://playwright.dev/docs/api/class-apiresponseassertions

# ASSERTIONS

## GenericAssertions

toBeTruthy() !== toBe(true)

toBeFalsy() !== toBe(false)

toBe() !== toEqual()

toEqual() !== toStrictEqual()

There are several APIs for GenericAssertions.

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

test('assert a value', async ({ page }) => {
  const value = 1;
  expect(value).toBe(2);
});
```

https://playwright.dev/docs/api/class-genericassertions

# ASSERTIONS

## LocatorAssertions

**REMINDER**

THEY ARE ASYNC!!

To test if locators have the expected states.

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

test('status becomes submitted', async ({ page }) => {
  // ...
  await page.getByRole('button').click();
  await expect(page.locator('.status')).toHaveText('Submitted');
});
```

https://playwright.dev/docs/api/class-locatorassertions

# ASSERTIONS

## PageAssertions

Page is also Async.

To test if page has the expected states.

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

test('navigates to login', async ({ page }) => {
  // ...
  await page.getByText('Sign in').click();
  await expect(page).toHaveURL(/.*\/login/);
});
```

https://playwright.dev/docs/api/class-pageassertions

# ASSERTIONS

## SnapshotAssertions

You have to create a referenced snapshot. Assertion is about verify the screenshot does match.

If page does not change, the snapshot should not too.

```
expect(screenshot).toMatchSnapshot('landing-page.png');
```

https://playwright.dev/docs/api/class-snapshotassertions

# Twittah! revisited #2

tests/twittah.spec.ts

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

test('Visit Twittah!', async ({ page }) => {
  await page.goto('https://twittah.web.app');
  await expect(page.getByTestId('app-name')).toBeVisible();
  await expect(page.getByTestId('app-name')).toHaveText('Twittah!');
});
```

https://github.com/chonla/twittah-test [visit]

# TRY IT YOURSELVES

```
<form _ngcontent-ng-c2284167468 novalidate class="ng-pristine ng-invalid ng-touched">
  <div _ngcontent-ng-c2284167468 class="field">
    <label _ngcontent-ng-c2284167468 class="label">บัญชี Twittah!</label>
    <div _ngcontent-ng-c2284167468 class="control">
      <input _ngcontent-ng-c2284167468 type="text" formcontrolname="login"
      placeholder="บัญชี Twittah" class="input ng-pristine ng-invalid ng-touched"
      data-testid="login-field"> flex
        #shadow-root (user-agent)
      </input>
    </div>
  </div>
  <div _ngcontent-ng-c2284167468 class="field">
    <label _ngcontent-ng-c2284167468 class="label">รหัสผ่าน</label>
    <div _ngcontent-ng-c2284167468 class="control"> == $0
      <input _ngcontent-ng-c2284167468 type="password" formcontrolname="password"
      placeholder="รหัสผ่าน" class="input ng-untouched ng-pristine ng-invalid" data-
      testid="password-field"> ... </input> flex
```

**Use Playwright to**

1. Login into Twittah!
2. Post a message to Twittah!
3. Logout from Twittah!

# SPARKING STRUCTURE


EXPLORER

PLAYWRIGHT

- .github
- fixtures
  - app.ts
  - users.ts M
- interfaces
  - app.ts
  - credential.ts
  - user.ts
- node_modules

Data

Structure of Data

users.ts M

fixtures > users.ts > ...

You, 1 second ago | 1 author (You)

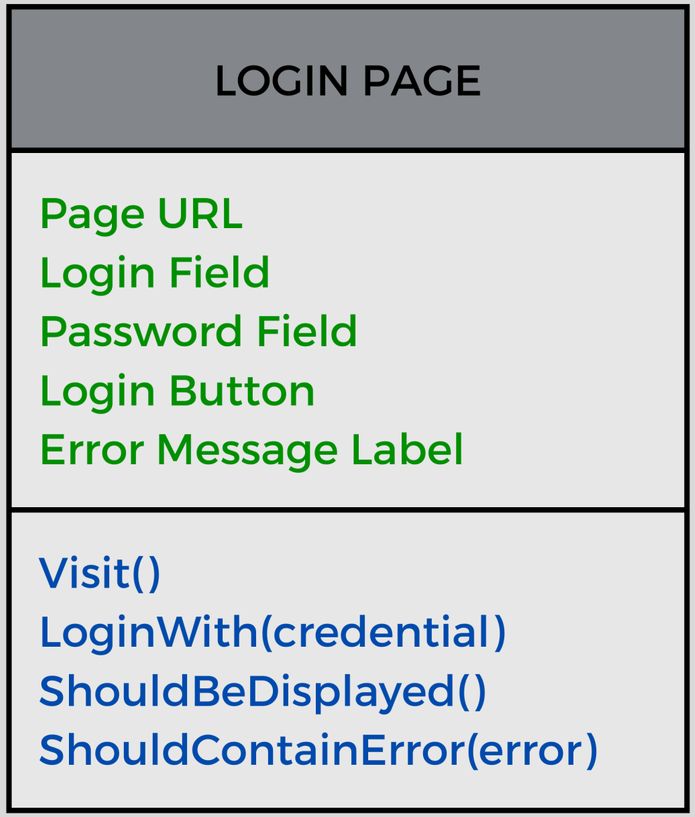
```ts
import { User } from "../interfaces/user";

export const validUser: User = {
    displayName: 'ไมตรี',
    credential: {
        login: 'maitree',
        password: '123456'
    }
};

export const invalidUser: User = {
    displayName: 'คนไม่ดี',
    credential: {
```

https://github.com/chonla/twittah-test [structure]

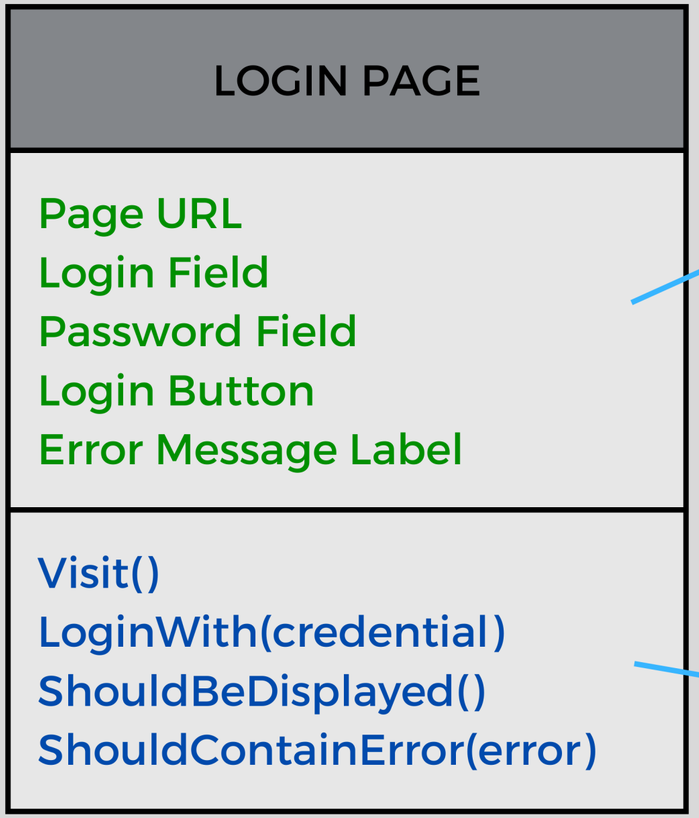
# PAGE OBJECT MODEL

Page object model is a design pattern used in Selenium to collect web elements into objects and wrap complicated actions into higher level of abstractions in order to **simplify maintenance and promote readability.**

# PAGE OBJECT MODEL



# PAGE OBJECT MODEL




```
You, 4 minutes ago | 1 author (You)
export class LoginPage {
    readonly _page: Page;
    readonly _pageUrl: string;
    readonly _loginField: Locator;
    readonly _passwordField: Locator;
    readonly _loginButton: Locator;
    readonly _errorMessageLabel: Locator;
    readonly _appName: Locator;

    constructor(page: Page) {
        this._page = page;
        this._pageUrl = `${app.baseUrl}/`;
        this._loginField = page.getByTestId('login-field');
        this._passwordField = page.getByTestId('password-field');
        this._loginButton = page.getByTestId('login-button');
        this._errorMessageLabel = page.getByTestId('error-message');
        this._appName = page.getByTestId('app-name')
    }

    async visit() {
        await this._page.goto(this._pageUrl);
    }

    async loginWith(credential: Credential) {
        await this._loginField.fill(credential.login);
        await this._passwordField.fill(credential.password);
        await this._loginButton.click();
    }

    async shouldBeDisplayed() {
        await this._page.waitForURL(this._pageUrl);
        await expect(this._appName).toBeVisible();
        await expect(this._appName).toHaveText('Twittah!');
```

https://github.com/chonla/twittah-test [pom]

# TRY IT YOURSELVES #2

Convert your previous tests into

**Page Object Models**

# DATA DRIVEN TESTINGS

Data Driven Testings or Parameterized Testings is a test pattern that use the same test logic to test by given data set.

# DATA DRIVEN TESTINGS

Using for loop through the data in the data set.

```
for (const invalidUser of invalidUsers) {
  test(`Credential ไม่ถูกต้อง: ${invalidUser.credential.login}`, async ({ page }) => {
    await loginPage.loginWith(invalidUser.credential);

    await loginPage.shouldBeDisplayed();
    await loginPage.shouldContainErrorMessage(invalidUser.errorMessage);
  });
}
```

https://github.com/chonla/twittah-test [parameterized]

# IDEMPOTENT

Making tests idempotent is possible by creating a controlled environment using **Test Setup to prepare system state before tests** and **Test Tear Down to reset system state after tests**.

# GHERKIN

Gherkin is a language used in Cucumber, Test Framework, promoting BDD or Behavior-Driven Development by focusing on Business requirement readability in test script.

# GHERKIN in PLAYWRIGHT

```
npm i -D playwright-bdd
```

https://github.com/chonla/twittah-test [gherkin]

# GHERKIN in PLAYWRIGHT

playwright.conf.js

```
1   import { defineConfig, devices } from '@playwright/test';
2   import { defineBddConfig } from 'playwright-bdd';
3
4   const testDir = defineBddConfig({
5     paths: ['features/*.feature'],
6     require: ['steps/**/*.ts'],
7     importTestFrom: './fixtures/fixtures.ts'
8   });
9
10  export default defineConfig({
11    testDir: testDir,
12    /* Run tests in files in parallel */
13    fullyParallel: true,
14    /* Fail the build on CI if you accidentally left test.only in the source co
```

Import

Declare

Apply

# GHERKIN in PLAYWRIGHT

fixtures/fixtures.ts

```ts
import { test as base } from 'playwright-bdd';
import { DefaultPage } from '../pom/default.page';
import { LoginPage } from '../pom/login.page';
import { HomePage } from '../pom/home.page';

export const test = base.extend<{
    defaultPage: DefaultPage;
    loginPage: LoginPage;
    homePage: HomePage;
}>({
    defaultPage: async ({ page }, use) => {
        await use(new DefaultPage(page));
    },
    loginPage: async ({ page }, use) => {
        await use(new LoginPage(page));
    },
    homePage: async ({ page }, use) => {
        await use(new HomePage(page));
    },
});
```

Inject page object models

# GHERKIN in PLAYWRIGHT

steps/default.ts

```ts
import { createBdd } from 'playwright-bdd';
import { test } from '../fixtures/fixtures';

const { Given, When, Then } = createBdd(test);

Given('I open Twittah', async ({ defaultPage }) => {
    await defaultPage.visit();
});

Then('I see the login page', async ({ loginPage }) => {
    await loginPage.shouldBeDisplayed();
});
```

Define steps

Use page object model

# GHERKIN in PLAYWRIGHT

steps/login.ts

```
import { createBdd } from 'playwright-bdd';
import { test } from '../fixtures/fixtures';

const { Given, When, Then } = createBdd(test);

Given('I open Login Page', async ({ loginPage }) => {
    await loginPage.visit();
});

When(/I login with login name ([^ ]+) and password ([^ ]+)/,
    async ({ loginPage }, loginName: string, password: string) => {
        await loginPage.loginWith({ login: loginName, password: password });
    });

Then('I see the home page', async ({ homePage }) => {
    await homePage.shouldBeDisplayed();
});
```

Define steps in RegExp

Capture parameters from step

# GHERKIN in PLAYWRIGHT

features/login.feature

```
Feature: Visit Twittah!

    Background: อยู่ที่หน้า Login ของ Twittah แล้ว
        Given I open Login Page

    Scenario: ล็อกอินสำเร็จต้องไปที่หน้าแรก
        When I login with login name bancha and password 123456
        Then I see the home page
```

Before Each

Test Case

# GHERKIN in PLAYWRIGHT

```
npx bddgen && npx playwright test

Running 6 tests using 5 workers
  6 passed (3.7s)

To open last HTML report run:

  npx playwright show-report
```

# CI/CD

```
docker run -it --rm --ipc=host -v .:/app -w /app mcr.microsoft.com/playwright:v1.37.1-jammy sh -c "npm install ; npm ci ; npx playwright test"

added 89 packages, changed 33 packages, and audited 130 packages in 26s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 9.6.7 -> 10.2.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.5
npm notice Run npm install -g npm@10.2.5 to update!
npm notice

added 129 packages, and audited 130 packages in 3s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Running 6 tests using 5 workers
  6 passed (5.9s)
```

https://github.com/chonla/twittah-test [cicd]

# That’s all … for now

## Thank you

**Test Scripts Repository**

https://github.com/chonla/twittah-test

**@chonla**