Grouping and Tagging in Playwright

Introduction

When you are writing automation tests using **Playwright**, it's important to organize them in a proper way. As your test suite grows, running all tests every time can be slow and unnecessary. That's where **grouping** and **tagging** become useful.

In this document, we will learn:

- Why grouping is needed
- How to group tests in Playwright
- The difference between grouping and tagging
- Best practices to manage your test suite

Why Do We Need Grouping?

Imagine you have 100 test cases. Some of them are **smoke tests**, some are **regression tests** and others are related to different modules like login, dashboard, settings, etc.

It's not practical to run all tests every time. So, we **group** them to:

- Run only the required set of tests
- Organize tests in a cleaner way
- Make debugging and maintenance easier
- Improve the performance of test execution

How to Group Tests in Playwright

Playwright provides the `test.describe()` block, which allows us to **group related tests together**.

You can think of `test.describe()` as a way to make a folder inside your test file, where you can place similar test cases.

Basic Example

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

test.describe('Login Tests', () => {
  test('Should show login page', async ({ page }) => {
    console.log('Test 1: Login page');
  });

  test('Should login with valid credentials', async ({ page }) => {
    console.log('Test 2: Valid login');
  });
});

test.describe('Dashboard Tests', () => {
  test('Should display dashboard', async ({ page }) => {
    console.log('Test 3: Dashboard');
  });

  test('Should logout user', async ({ page }) => {
    console.log('Test 4: Logout');
  });
});
```

In this example:

- Two groups are created: **Login Tests** and **Dashboard Tests**
- Each group has two related test cases

Running Specific Groups

Sometimes you only want to run tests from one group. You can use `.only` to run that group alone:

```
test.describe.only('Dashboard Tests', () => {
  // only this group will run
});
```

You can also use `.skip` to ignore a group:

```
test.describe.skip('Login Tests', () => {
  // this group will be skipped
});
```

Real-Life Scenario

Let’s say your manager wants you to run only **smoke tests** before deployment. You can group all your smoke tests like this:

```
test.describe('Smoke Tests', () => {
  test('Login Test', async ({ page }) => {
    // smoke test
  });

  test('Dashboard Load Test', async ({ page }) => {
    // smoke test
  });
});
```

Then run just this group using `.only` during deployment checks.

Grouping vs Tagging

Both **grouping** and **tagging** help in organizing tests — but they are used differently.

Grouping

- Grouping is done using `test.describe()`
- Helps organize tests by **feature** or **module**
- You can use `.only` or `.skip` inside a group
- It affects how your code is structured

Tagging

- Tagging is used to **mark tests with labels**
- You can run tests using tags from the CLI
- Useful when tests are spread across files but belong to the same type (e.g., `@smoke`)

Example:

```
test('@smoke Login test', async ({ page }) => {
  // smoke test
});
```

To run only smoke tests, use this in terminal:

```
npx playwright test --grep "@smoke"
```

Comparison Table

Feature	Grouping (`test.describe`)	Tagging (`@smoke`, `@regression`)
How it works	Groups multiple tests inside a block	Adds labels to individual tests
Use case	Organize tests by module/feature	Run selected tests from CLI using tag names
Syntax	`test.describe()`	`test('@tag Test name', ...)`
CLI Usage	Not directly used in CLI	Used with `--grep` to filter tests
Example	`test.describe('Login', () => {})`	`test('@smoke should login', ...)`

Best Practices

Here are some tips to follow while grouping and tagging:

- Use **meaningful group names** (e.g., `Login Tests`, `Payment Tests`)
- Don’t overuse `.only` — it should be temporary during development
- Use **tags** like `@smoke`, `@regression`, `@critical` for easy filtering
- Maintain a consistent format for test names and tags
- Document your group and tag structure for team understanding

---

Summary

Concept	Purpose	How to Use
Grouping	Organize tests by feature/module	Use `test.describe()`
Tagging	Mark tests by type	Use `@tag` and `--grep` in CLI
Execution	`.only` to run specific group	`--grep` to run tagged tests
