Skip to main content

PlaywrightDocsAPI
Node.js

- Node.js
- Python
- Java
- .NET

Community
Search⌘K

- Getting Started
  - Installation
  - Writing tests
  - Generating tests
  - Running and debugging tests
  - Trace viewer
  - Setting up CI
- Getting started - VS Code
- Release notes
- Canary releases
- Playwright Test
  - Test configuration
  - Test use options
  - Annotations
  - Command line

Annotations

Introduction

Playwright supports tags and annotations that are displayed in the test report.

You can add your own tags and annotations at any moment, but Playwright comes with a few built-in ones:

- test.skip() marks the test as irrelevant. Playwright does not run such a test. Use this annotation when the test is not applicable in some configuration.
- test.fail() marks the test as failing. Playwright will run this test and ensure it does indeed fail. If the test does not fail, Playwright will complain.
- test.fixme() marks the test as failing. Playwright will not run this test, as opposed to the `fail` annotation. Use `fixme` when running the test is slow or crashes.
- test.slow() marks the test as slow and triples the test timeout.

Annotations can be added to a single test or a group of tests.

Built-in annotations can be conditional, in which case they apply when the condition is truthy, and may depend on test fixtures. There could be multiple annotations on the same test, possibly in different configurations.

Focus a test

You can focus some tests. When there are focused tests, only these tests run.

```
test.only('focus this test', async ({ page }) => {
  // Run only focused tests in the entire project.
});
```

Skip a test

Mark a test as skipped.

```
test.skip('skip this test', async ({ page }) => {
  // This test is not run
});
```

Conditionally skip a test

You can skip certain test based on the condition.

```
test('skip this test', async ({ page, browserName }) => {
  test.skip(browserName === 'firefox', 'Still working on it');
});
```

Group tests

You can group tests to give them a logical name or to scope before/after hooks to the group.

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

test.describe('two tests', () => {
  test('one', async ({ page }) => {
    // ...
  });

  test('two', async ({ page }) => {
    // ...
  });
});
```

Tag tests

Sometimes you want to tag your tests as `@fast` or `@slow`, and then filter by tag in the test report. Or you might want to only run tests that have a certain tag.

To tag a test, either provide an additional details object when declaring a test, or add `@`-token to the test title. Note that tags must start with `@` symbol.

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

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

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

You can also tag all tests in a group or provide multiple tags:

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

test.describe('group', {
  tag: '@report',
}, () => {
  test('test report header', async ({ page }) => {
    // ...
  });
  test('test full report', {
    tag: ['@slow', '@vrt'],
  }, async ({ page }) => {
    // ...
  });
});
```

You can now run tests that have a particular tag with `--grep` command line option.

- Bash
- PowerShell
- Batch

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

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

- Bash
- PowerShell
- Batch

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

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

- Bash
- PowerShell
- Batch

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

Or run tests containing both tags (logical `AND` operator) using regex lookaheads:

```
npx playwright test --grep "(?=.*@fast)(?=.*@slow)"
```

You can also filter tests in the configuration file via testConfig.grep and testProject.grep.

Annotate tests

If you would like to annotate your tests with something more substantial than a tag, you can do that when declaring a test. Annotations have a `type` and a `description` for more context and available in reporter API. Playwright's built-in HTML reporter shows all annotations, except those where `type` starts with _ symbol.

For example, to annotate a test with an issue url:

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

test('test login page', {
  annotation: {
    type: 'issue',
    description: 'https://github.com/microsoft/playwright/issues/23180',
  },
}, async ({ page }) => {
  // ...
});
```

You can also annotate all tests in a group or provide multiple annotations:

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

test.describe('report tests', {
  annotation: { type: 'category', description: 'report' },
}, () => {
  test('test report header', async ({ page }) => {
    // ...
  });

  test('test full report', {
    annotation: [
      { type: 'issue', description:
'https://github.com/microsoft/playwright/issues/23180' },
      { type: 'performance', description: 'very slow test!' },
    ],
  }, async ({ page }) => {
    // ...
  });
});
```

Conditionally skip a group of tests

For example, you can run a group of tests just in Chromium by passing a callback.

example.spec.ts

```
test.describe('chromium only', () => {
  test.skip(({ browserName }) => browserName !== 'chromium', 'Chromium
only!');

  test.beforeAll(async () => {
    // This hook is only run in Chromium.
  });

  test('test 1', async ({ page }) => {
    // This test is only run in Chromium.
  });

  test('test 2', async ({ page }) => {
    // This test is only run in Chromium.
  });
});
```

Use fixme in `beforeEach` hook

To avoid running `beforeEach` hooks, you can put annotations in the hook itself.

example.spec.ts

```
test.beforeEach(async ({ page, isMobile }) => {
  test.fixme(isMobile, 'Settings page does not work in mobile yet');

  await page.goto('http://localhost:3000/settings');
});

test('user profile', async ({ page }) => {
  await page.getByText('My Profile').click();
  // ...
});
```

Runtime annotations

While the test is already running, you can add annotations to [`test.info().annotations`].

example.spec.ts

```
test('example test', async ({ page, browser }) => {
  test.info().annotations.push({
    type: 'browser version',
    description: browser.version(),
  });

  // ...
});
```

Previous
Test use options

Next
Command line

Learn

- Getting started
- Playwright Training
- Learn Videos