In Playwright, **annotations** are special keywords that help define and organize test cases.

These annotations make it easy to create test flows and manage test configurations, such as marking tests to **skip**, **retry**, or **run** with **specific settings**.

**Common Annotations in Playwright:**

1. **test**:
   * The test annotation is the **main function** used to define a test case in Playwright.
   * It takes **two arguments**: the **name of the test** and the **test function** that contains the steps.
   * Example:

|  |
| --- |
| const { test } = require('@playwright/test');  test('First Playwright Test', async ({ page }) => {  await page.goto('https://example.com');  await page.click('text=Get Started');  }); |

* + Here, 'First Playwright Test' is the name of the test, and the function following it contains the actual test code.

1. **test.describe()**:
   * test.describe() is used to **group multiple related tests**.
   * It helps in organizing the test suite and grouping similar tests together.
   * Example:

|  |
| --- |
| **test**.**describe**('Login Feature Tests', () => {  **test**('Valid Login Test', async ({ page }) => {  await page.goto('https://example.com/login');  await page.fill('#username', 'user');  await page.fill('#password', 'password');  await page.click('button[type="submit"]');  });  **test**('Invalid Login Test', async ({ page }) => {  await page.goto('https://example.com/login');  await page.fill('#username', 'invalid\_user');  await page.fill('#password', 'wrong\_password');  await page.click('button[type="submit"]');  });  }); |

* + In this example, test.describe() groups together related tests for the **Login Feature**.

1. **test.beforeEach() and test.afterEach()**:
   * These annotations are used to define **setup** and **teardown** logic that should run **before** or **after** each test.
   * They help in performing common setup tasks like **navigating to a page**, **logging in**, or **clearing test data**.
   * Example:

|  |
| --- |
| **test**.**beforeEach**(async ({ page }) => {  await page.goto('https://example.com');  });  **test**('Check Home Page', async ({ page }) => {  await page.click('text=Home');  // Test steps  }); |

* + Here, test.beforeEach() ensures that **each test** starts from the main page (https://example.com).

1. **test.beforeAll() and test.afterAll()**:
   * These annotations are used to run **setup** or **teardown** tasks **once** for the entire suite.
   * test.beforeAll() runs **before any tests**, and test.afterAll() runs **after all tests**.
   * Example:

|  |
| --- |
| test.**beforeAll**(async () => {  console.log('Setting up test suite');  });  test.**afterAll**(async () => {  console.log('Tearing down test suite');  }); |

1. **test.skip()**:
   * test.skip() is used to **skip** a specific test case.
   * This is useful when a particular test is **not relevant** or is **temporarily disabled**.
   * Example:

|  |
| --- |
| test.skip('This test is skipped', async ({ page }) => {  await page.goto('https://example.com');  }); |

1. **test.only()**:
   * If you want to **run only a specific test**, you can use test.only(). This helps in isolating and debugging a specific test.
   * Example:

|  |
| --- |
| test.**only**('Only this test will run', async ({ page }) => {  await page.goto('https://example.com');  }); |

* + When test.only() is used, **only this test will be executed**, and other tests in the file will be ignored.

1. **test.fixme()**:
   * Similar to test.skip(), but indicates that the test is **known to be failing or incomplete**.
   * Useful for marking **tests that need fixing**.
   * Example:

|  |
| --- |
| test.**fixme**('This test needs fixing', async ({ page }) => {  await page.goto('https://example.com');  }); |

1. **test.retry()**:
   * This annotation helps in **retrying failed tests** automatically. It is useful for **flaky tests** that sometimes fail due to timing issues.
   * Example:

|  |
| --- |
| test('Retry Example', async ({ page }) => {  await page.goto('https://example.com');  }).**retries**(2); // **Retry this test 2 times if it fails** |

1. **test.step()**:
   * Used to **add steps** to a test to make debugging and logs easier to follow.
   * Example:

|  |
| --- |
| test('Test with Steps', async ({ page }) => {  await test.step('Navigate to Homepage', async () => {  await page.goto('https://example.com');  });  await test.**step**('Click Get Started', async () => {  await page.click('text=Get Started');  });  }); |

**Summary of Annotations:**

1. **test()**: Defines a test.
2. **test.describe()**: Groups related tests.
3. **test.beforeEach(), test.afterEach()**: Run before or after each test for setup/teardown.
4. **test.beforeAll(), test.afterAll()**: Run once before/after all tests in a suite.
5. **test.skip()**: Skips a specific test.
6. **test.only()**: Runs only the specified test, ignoring others.
7. **test.fixme()**: Marks a test that needs fixing or is expected to fail.
8. **test.retry()**: Retries a test a specified number of times if it fails.
9. **test.step()**: Adds descriptive steps for easier debugging.

These annotations help you **organize**, **configure**, and **debug** your Playwright test cases more effectively, making it easier to handle different scenarios and maintain cleaner test code.