1.Visual Comparisons

**Introduction**

Visual testing in Playwright allows you to automatically capture and compare screenshots to ensure that UI changes don’t introduce unexpected differences. Playwright provides the toHaveScreenshot() method to capture screenshots and compare them with reference (golden) images.

* **Feature**: await expect(page).toHaveScreenshot()
  + On the first run, Playwright generates a reference screenshot (golden file).
  + Subsequent runs compare new screenshots to the reference.

**Example:**

typescript

Copy code

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

test('example test', async ({ page }) => {

await page.goto('https://playwright.dev');

await expect(page).toHaveScreenshot();

});

**Generating Screenshots**

When running the visual test for the first time, Playwright creates a new snapshot (golden file) if one does not exist. Any subsequent test run compares the actual screenshot to the golden file.

* **First-Time Behavior**:
  + Playwright throws an error if no golden file is found.
  + It will create a reference snapshot automatically.

**Error Example**:

vbnet

Copy code

Error: A snapshot doesn't exist at example.spec.ts-snapshots/example-test-1-chromium-darwin.png, writing actual.

* **Golden File Creation**:
  + The first successful screenshot is saved for future comparisons.

**Snapshot Naming and Folder Structure**

Snapshot files follow a specific naming convention and are stored in a sub-directory alongside the test file.

* **Naming Convention**:
  + The snapshot name is auto-generated, including the test name, browser, and platform.

**Example**:

* example-test-1-chromium-darwin.png
  + example-test-1.png: Generated name for the snapshot.
  + chromium-darwin: Browser name and operating system.
* **Folder Structure**:
  + Snapshots are stored in a sub-folder named after the test file.

**Example Directory**:

Copy code

example.spec.ts

example.spec.ts-snapshots/

└── example-test-1-chromium-darwin.png

* **Custom Snapshot Name**: You can customize the snapshot name by passing it as an argument.

**Example**:

typescript

Copy code

await expect(page).toHaveScreenshot('landing.png');

**Updating Screenshots**

When the page or UI changes, you need to update the reference screenshots. This can be done using the --update-snapshots flag.

* **Command**:

bash

Copy code

npx playwright test --update-snapshots

* **Custom Snapshot Path**: You can specify a custom path for snapshots using an array of path segments.

**Example**:

typescript

Copy code

await expect(page).toHaveScreenshot(['custom', 'path', 'snapshot.png']);

**Options for Visual Comparison**

**1. maxDiffPixels**

Playwright allows you to define how many pixels can differ between two screenshots before the test fails. By default, comparisons are pixel-perfect.

* **Usage**:
  + Specify maxDiffPixels to tolerate slight pixel differences between screenshots.

**Test Example**:

typescript

Copy code

await expect(page).toHaveScreenshot({ maxDiffPixels: 100 });

* **Global Configuration**: Set this option globally in the Playwright configuration for all tests.

**Config Example**:

typescript

Copy code

export default defineConfig({

expect: {

toHaveScreenshot: { maxDiffPixels: 100 },

},

});

**2. stylePath**

You can apply a custom stylesheet to the page during screenshot comparisons. This is useful for ignoring dynamic or volatile elements like animations or ads.

* **Custom Stylesheet**: Write a CSS file that hides certain elements to ensure more stable screenshots.

**CSS File (screenshot.css)**:

css

Copy code

iframe {

visibility: hidden;

}

* **Test Example**:

typescript

Copy code

import path from 'path';

await expect(page).toHaveScreenshot({ stylePath: path.join(\_\_dirname, 'screenshot.css') });

* **Global Configuration**: Apply the stylePath option globally for all tests.

**Config Example**:

typescript

Copy code

export default defineConfig({

expect: {

toHaveScreenshot: {

stylePath: './screenshot.css'

},

},

});

**Non-Image Snapshots**

Playwright can also handle non-image snapshots, such as text content or binary data, by using toMatchSnapshot().

* **Usage**:
  + Use toMatchSnapshot() to compare text or binary files instead of images.

**Test Example (Text Snapshot)**:

typescript

Copy code

await page.goto('https://playwright.dev');

expect(await page.textContent('.hero\_\_title')).toMatchSnapshot('hero.txt');

* **Snapshot Storage**:
  + Non-image snapshots, like image snapshots, are stored in a sub-directory named after the test file.

**Example Directory**:

Copy code

my.spec.ts

my.spec.ts-snapshots/

└── hero.txt

* **Best Practices**:
  + Commit snapshot files to version control (e.g., Git).
  + Review any snapshot changes during code reviews to ensure intentional updates.

**Conclusion**

Visual comparisons in Playwright provide a robust way to test UI changes by comparing screenshots or other file types. By using various options like maxDiffPixels and stylePath, you can fine-tune your visual tests to ensure consistency and catch unexpected changes. Snapshots, whether images or other data, can be updated and customized to suit the needs of your project.