# Visual Testing in Playwright

**Visual Testing** ensures that the UI looks the same as expected by comparing current screenshots of pages or elements with **baseline images** (golden images).

**Purpose:**

To detect unintended visual changes like layout shifts, colour mismatches, or broken components in a web application.

## Full Page Visual Comparison

```
expect(await page.screenshot()).toMatchSnapshot("homepage.png");
```

- Takes a **screenshot of the entire page**.
- Compares it with the baseline image named homepage.png.
- Fails the test if there's any visual difference.

**First run:** Playwright saves the snapshot.
**Later runs:** It compares new screenshots with the saved ones.

## Element-Level Snapshot Comparison

```
const logo = page.locator("img[alt='Tricentis Demo Web Shop']");
expect(await logo.screenshot()).toMatchSnapshot('logo.png');
```

- Targets a **specific element** (the website logo).
- Captures its screenshot and compares it with the baseline logo.png.

**Notes:**

- **Baseline images** are stored in the __screenshots__ folder.
- Run with --update-snapshots to update baseline images if changes are expected.
- Alternative way:
- await expect(page).toHaveScreenshot();
    - Automatically handles screenshot capturing and comparison.

# Accessibility Testing in Playwright

## What is Accessibility Testing?

Accessibility testing ensures that your web application is usable by people with disabilities. It checks for compliance with **WCAG** (Web Content Accessibility Guidelines), covering issues like:

- Missing ALT text on images
- Poor colour contrast
- Missing labels on forms
- Inaccessible keyboard navigation

## Setting Up Accessibility Testing in Playwright

To get started, install the accessibility testing plugin:

```
npm install @axe-core/playwright
```

This plugin integrates **Axe-core**, a powerful accessibility engine, into your Playwright tests.

**Example: Accessibility Test in Playwright**

```
import { test, expect } from '@playwright/test';
import AxeBuilder from '@axe-core/playwright';

test('accessibility test', async ({ page }, testInfo) => {

  await page.goto('https://www.w3.org');

   // Run full accessibility scan using Axe
  const accessibilityScanResults = await new AxeBuilder({ page }).analyze();

  // Log results to console
  console.log(accessibilityScanResults);

  // Attach results to the test report (useful in CI)
  await testInfo.attach('accessibility-scan-results', {
    body: JSON.stringify(accessibilityScanResults, null, 2),
    contentType: 'application/json',
  });

  // Assert that there are no accessibility violations
  expect(accessibilityScanResults.violations.length).toEqual(0);
});
```

## Optional Configurations

You can customize the scan with:

- **Tags filtering** (e.g., ['wcag2a', 'wcag21aa']) to target specific **WCAG** levels
- **Disabling rules** (e.g., disableRules(['duplicate-id'])) to ignore known issues

**Only scan for specific WCAG tags**

```
new AxeBuilder({ page }).withTags(['wcag2a', 'wcag2aa']);
```

**Disable certain rules from scan**

```
new AxeBuilder({ page }).disableRules(['duplicate-id']);
```