PlaywrightAssertions

* **APIResponseAssertions:**

The APIResponseAssertions class provides assertion methods that can be used to make assertions about the APIResponse in Playwright tests.

**Example Usage:**

javascript

Copy code

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

test('navigates to login', async ({ page }) => {

// ...

const response = await page.request.get('https://playwright.dev');

await expect(response).toBeOK();

});

**Methods**

**toBeOK**

* **Added in:** v1.18
* Ensures the response status code is within the 200-299 range.

**Usage:**

javascript

Copy code

await expect(response).toBeOK();

* **Returns:**  
  Promise<void>

**Properties**

**not**

* **Added in:** v1.20
* Makes the assertion check for the opposite condition. For example, this code tests that the response status is **not** successful:

**Usage:**

javascript

Copy code

await expect(response).not.toBeOK();

* **GenericAssertions**

The GenericAssertions class provides various assertion methods to validate values in your Playwright tests. You can use this class to perform deep equality checks, pattern matching, and value comparisons. An instance of GenericAssertions is created by calling expect().

**Example Usage:**

typescript

Copy code

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

test('assert a value', async ({ page }) => {

const value = 1;

expect(value).toBe(2);

});

**Methods:**

**any()**

Added in: v1.9  
Matches any object instance created from the specified constructor or primitive type. This method can be used with expect(value).toEqual() to perform pattern matching.

**Usage:**

typescript

Copy code

// Match an instance of a class

class Example {}

expect(new Example()).toEqual(expect.any(Example));

// Match any number

expect({ prop: 1 }).toEqual({ prop: expect.any(Number) });

**anything()**

Added in: v1.9  
Matches everything except null and undefined. Useful inside expect(value).toEqual() for pattern matching.

**Usage:**

typescript

Copy code

const value = { prop: 1 };

expect(value).toEqual({ prop: expect.anything() });

**arrayContaining()**

Added in: v1.9  
Matches arrays that contain all elements in the expected array, in any order. The received array can include extra elements.

**Usage:**

typescript

Copy code

expect([1, 2, 3]).toEqual(expect.arrayContaining([3, 1]));

**closeTo()**

Added in: v1.9  
Compares floating-point numbers for approximate equality. Useful when small floating-point errors are expected.

**Usage:**

typescript

Copy code

expect({ prop: 0.1 + 0.2 }).toEqual({ prop: expect.closeTo(0.3, 5) });

**objectContaining()**

Added in: v1.9  
Matches objects that contain and match the properties of the expected object. Additional properties are allowed.

**Usage:**

typescript

Copy code

expect({ foo: 1, bar: 2 }).toEqual(expect.objectContaining({ foo: 1 }));

**stringContaining()**

Added in: v1.9  
Matches strings that contain the expected substring.

**Usage:**

typescript

Copy code

expect('Hello world!').toEqual(expect.stringContaining('Hello'));

**stringMatching()**

Added in: v1.9  
Matches strings that match a given regular expression pattern.

**Usage:**

typescript

Copy code

expect('123ms').toEqual(expect.stringMatching(/\d+m?s/));

**toBe()**

Added in: v1.9  
Compares values with Object.is and checks for strict equality.

**Usage:**

typescript

Copy code

expect(value).toBe(value);

**toBeCloseTo()**

Added in: v1.9  
Compares floating-point numbers for approximate equality, useful for ensuring small differences in floating-point calculations don't affect tests.

**Usage:**

typescript

Copy code

expect(0.1 + 0.2).toBeCloseTo(0.3, 5);

**toBeDefined()**

Added in: v1.9  
Ensures that a value is not undefined.

**Usage:**

typescript

Copy code

expect(value).toBeDefined();

**toBeFalsy()**

Added in: v1.9  
Ensures that a value is false in a boolean context (false, 0, '', null, undefined, NaN).

**Usage:**

typescript

Copy code

expect(value).toBeFalsy();

**toBeGreaterThan()**

Added in: v1.9  
Ensures that a value is greater than the expected number.

**Usage:**

typescript

Copy code

expect(42).toBeGreaterThan(10);

**toBeGreaterThanOrEqual()**

Added in: v1.9  
Ensures that a value is greater than or equal to the expected number.

**Usage:**

typescript

Copy code

expect(42).toBeGreaterThanOrEqual(42);

**toBeInstanceOf()**

Added in: v1.9  
Ensures that a value is an instance of a class.

**Usage:**

typescript

Copy code

expect(page).toBeInstanceOf(Page);

**toBeLessThan()**

Added in: v1.9  
Ensures that a value is less than the expected number.

**Usage:**

typescript

Copy code

expect(42).toBeLessThan(100);

**toBeLessThanOrEqual()**

Added in: v1.9  
Ensures that a value is less than or equal to the expected number.

**Usage:**

typescript

Copy code

expect(42).toBeLessThanOrEqual(42);

**toBeNull()**

Added in: v1.9  
Ensures that a value is null.

**Usage:**

typescript

Copy code

expect(value).toBeNull();

**toBeTruthy()**

Added in: v1.9  
Ensures that a value is true in a boolean context.

**Usage:**

typescript

Copy code

expect(value).toBeTruthy();

**toContain()**

Added in: v1.9  
Ensures that a string contains the expected substring or an array contains the expected item.

**Usage:**

typescript

Copy code

expect('Hello World').toContain('World');

expect([1, 2, 3]).toContain(2);

**toEqual()**

Added in: v1.9  
Performs a deep equality check on objects, arrays, and primitive values.

**Usage:**

typescript

Copy code

expect({ prop: 1 }).toEqual({ prop: 1 });

**toThrow()**

Added in: v1.9  
Ensures that a function throws an error.

**Usage:**

typescript

Copy code

expect(() => { throw new Error('Error'); }).toThrow('Error');

* **LocatorAssertions:**

The LocatorAssertions class in Playwright provides several methods for asserting the state of a Locator within tests. Below is an explanation of some common methods used for testing with Playwright.

**Example: Status Becomes Submitted**

javascript

Copy code

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

test('status becomes submitted', async ({ page }) => {

// Click a button and assert status becomes 'Submitted'

await page.getByRole('button').click();

await expect(page.locator('.status')).toHaveText('Submitted');

});

**Methods**

**1. toBeAttached**

Ensures the Locator points to an element connected to the DOM or a ShadowRoot.

**Usage:**

javascript

Copy code

await expect(page.getByText('Hidden text')).toBeAttached();

**Arguments:**

* options: Object (optional)
  + attached: boolean (optional)
  + timeout: number (optional) — Time to retry in milliseconds.

**Returns:** Promise<void>

**2. toBeChecked**

Asserts that the Locator points to a checked input.

**Usage:**

javascript

Copy code

const locator = page.getByLabel('Subscribe to newsletter');

await expect(locator).toBeChecked();

**Arguments:**

* options: Object (optional)
  + checked: boolean (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**3. toBeDisabled**

Checks if the Locator points to a disabled element.

**Usage:**

javascript

Copy code

const locator = page.locator('button.submit');

await expect(locator).toBeDisabled();

**Arguments:**

* options: Object (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**4. toBeEditable**

Asserts the Locator points to an editable element.

**Usage:**

javascript

Copy code

const locator = page.getByRole('textbox');

await expect(locator).toBeEditable();

**Arguments:**

* options: Object (optional)
  + editable: boolean (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**5. toBeEmpty**

Asserts that the Locator points to an empty or textless DOM node.

**Usage:**

javascript

Copy code

const locator = page.locator('div.warning');

await expect(locator).toBeEmpty();

**Arguments:**

* options: Object (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**6. toBeEnabled**

Ensures that the Locator points to an enabled element.

**Usage:**

javascript

Copy code

const locator = page.locator('button.submit');

await expect(locator).toBeEnabled();

**Arguments:**

* options: Object (optional)
  + enabled: boolean (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**7. toBeFocused**

Ensures that the Locator points to a focused DOM node.

**Usage:**

javascript

Copy code

const locator = page.getByRole('textbox');

await expect(locator).toBeFocused();

**Arguments:**

* options: Object (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**8. toBeHidden**

Ensures that the Locator either points to no DOM node or a non-visible one.

**Usage:**

javascript

Copy code

const locator = page.locator('.my-element');

await expect(locator).toBeHidden();

**Arguments:**

* options: Object (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**9. toBeInViewport**

Ensures the Locator points to an element that intersects with the viewport.

**Usage:**

javascript

Copy code

const locator = page.getByRole('button');

await expect(locator).toBeInViewport();

**Arguments:**

* options: Object (optional)
  + ratio: number (optional) — Minimal intersection ratio.
  + timeout: number (optional)

**Returns:** Promise<void>

**10. toBeVisible**

Ensures that the Locator points to a visible DOM node.

**Usage:**

javascript

Copy code

await expect(page.getByText('Welcome')).toBeVisible();

**Arguments:**

* options: Object (optional)
  + timeout: number (optional)
  + visible: boolean (optional)

**Returns:** Promise<void>

**11. toContainText**

Checks that the Locator points to an element containing the specified text.

**Usage:**

javascript

Copy code

const locator = page.locator('.title');

await expect(locator).toContainText('substring');

**Arguments:**

* expected: string | RegExp | Array<string | RegExp>
* options: Object (optional)
  + ignoreCase: boolean (optional)
  + timeout: number (optional)
  + useInnerText: boolean (optional)

**Returns:** Promise<void>

**12. toHaveAccessibleDescription**

Asserts that the Locator points to an element with a given accessible description.

**Usage:**

javascript

Copy code

const locator = page.getByTestId('save-button');

await expect(locator).toHaveAccessibleDescription('Save results to disk');

**Arguments:**

* description: string | RegExp
* options: Object (optional)
  + ignoreCase: boolean (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**13. toHaveAccessibleName**

Checks that the Locator points to an element with the given accessible name.

**Usage:**

javascript

Copy code

const locator = page.getByTestId('save-button');

await expect(locator).toHaveAccessibleName('Save to disk');

**Arguments:**

* name: string | RegExp
* options: Object (optional)
  + ignoreCase: boolean (optional)
  + timeout: number (optional)

**Returns:** Promise<void>

**toHaveAttribute(name, value)**

**Ensures:** Locator points to an element with the specified attribute and value.

* **Usage:**

js

Copy code

const locator = page.locator('input');

await expect(locator).toHaveAttribute('type', 'text');

* **Arguments:**
  + name (string): Attribute name.
  + value (string | RegExp): Expected attribute value.
  + options (optional):
    - ignoreCase (boolean): Case-insensitive match.
    - timeout (number): Time in ms for retrying the assertion.

**toHaveClass**

**Ensures:** Locator points to an element with the given CSS classes (supports full match or regex).

* **Usage:**

js

Copy code

const locator = page.locator('#component');

await expect(locator).toHaveClass('selected row');

* **Arguments:**
  + expected (string | RegExp | Array<string | RegExp>): Expected class.
  + timeout (optional): Retry time in ms.

**toHaveCount**

**Ensures:** Locator resolves to a specified number of DOM nodes.

* **Usage:**

js

Copy code

const list = page.locator('list > .component');

await expect(list).toHaveCount(3);

* **Arguments:**
  + count (number): Expected count.
  + timeout (optional): Retry time in ms.

**toHaveCSS**

**Ensures:** Locator resolves to an element with the specified CSS style.

* **Usage:**

js

Copy code

const locator = page.getByRole('button');

await expect(locator).toHaveCSS('display', 'flex');

* **Arguments:**
  + name (string): CSS property name.
  + value (string | RegExp): Expected CSS value.
  + timeout (optional): Retry time in ms.

**toHaveId**

**Ensures:** Locator points to an element with the specified DOM Node ID.

* **Usage:**

js

Copy code

const locator = page.getByRole('textbox');

await expect(locator).toHaveId('lastname');

* **Arguments:**
  + id (string | RegExp): Expected ID.
  + timeout (optional): Retry time in ms.

**toHaveJSProperty**

**Ensures:** Locator points to an element with the specified JavaScript property.

* **Usage:**

js

Copy code

const locator = page.locator('.component');

await expect(locator).toHaveJSProperty('loaded', true);

* **Arguments:**
  + name (string): Property name.
  + value (object): Expected property value.
  + timeout (optional): Retry time in ms.

**toHaveRole**

**Ensures:** Locator points to an element with a specific ARIA role.

* **Usage:**

js

Copy code

const locator = page.getByTestId('save-button');

await expect(locator).toHaveRole('button');

* **Arguments:**
  + role (string): ARIA role.
  + timeout (optional): Retry time in ms.

**toHaveScreenshot**

**Ensures:** Two consecutive locator screenshots yield the same result, and then compares the last one.

* **Usage:**

js

Copy code

const locator = page.getByRole('button');

await expect(locator).toHaveScreenshot('image.png');

* **Arguments:**
  + name (string | Array<string>): Snapshot name.
  + options (optional): Includes options like animations, mask, omitBackground, etc.

**toHaveText**

**Ensures:** Locator points to an element with the given text.

* **Usage:**

js

Copy code

const locator = page.locator('.title');

await expect(locator).toHaveText(/Welcome, Test User/);

* **Arguments:**
  + expected (string | RegExp | Array<string | RegExp>): Expected text.
  + options (optional): Includes options like ignoreCase, useInnerText, timeout.

**toHaveValue**

**Ensures:** Locator points to an element with the specified input value.

* **Usage:**

js

Copy code

const locator = page.locator('input[type=number]');

await expect(locator).toHaveValue(/[0-9]/);

* **Arguments:**
  + value (string | RegExp): Expected value.
  + timeout (optional): Retry time in ms.