CYDEO

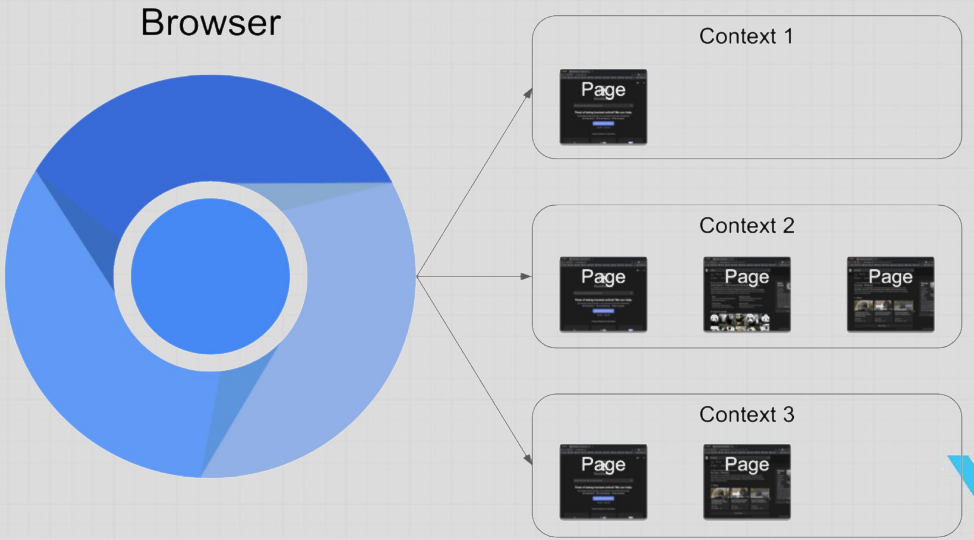
# Playwright Automation Day02

# Content

- The locator object methods
- Test Groups
- Hooks
- Playwright Assertions

# Fixtures

- Playwright uses fixtures to provide reusable, isolated test environments
- Simplifies the test setup and teardown process, making the tests more reliable and maintainable
- Common fixtures includes:
  - page
  - browser
  - context
  - viewport
  - browserName
  - request
  - baseURL



# The page Fixture

- A powerful tool for web automation and testing.
- An isolated page instance for each test
- Provides a clean, consistent starting point for each test
- Automatically created and destroyed
- Provides methods for navigation, interaction, and assertions

# Common Methods of page Object

| Method Name | Description |
| --- | --- |
| goto(url) | Navigate the browser to a specified URL |
| title() | Returns the title of the current page as a string |
| url() | Returns the current URL of the page as a string |
| setViewportSize({w, l}) | sets the size of the browser viewport to specified width and height values |
| setDefaultTimeout(milliseconds) | sets the default maximum time (in milliseconds) the test on the page can take before timing out |
| page.locator(selector) | Creates a locator for the given selector, which can be used to perform actions like click, type |

# Locators

- Powerful tools for element interaction and assertion.
- An object representing a way to find element(s) on the page
- Provides a robust and reliable way to interact with page elements
- Locator Selectors:
    - CSS selectors: page.locator('button.primary')
    - XPath: page.locator('//button[contains(text(), "Submit")]')
    - Text content: page.locator('text=Submit')
    - TestID: page.locator('data-testid=submit-button')

# Common Methods of Locator Object

| Methods - Actions | Methods - Retrieval | Methods - State |
|---|---|---|
| click() | textContent() | isVisible() |
| fill() | innerText() | isEnabled() |
| type() | inputValue() | isChecked() |
| press() | getAttribute() | isDisabled() |
| check() | | |
| uncheck() | | |
| selectOption() | | |

# Test Groups

- The test.describe() is used for creating test groups.
- Test groups allow us to organize related tests together
- It improves readability and maintainability of our test suite
- Test Groups can be nested for further organization
- Test groups can  have their own **hooks** that apply to all tests within the group

```
test.describe('Group name', () => {

  test('Test 1', async ({fixtureName}) => {
    // Test1 codes
  });

  test('Test 2', async ({fixtureName}) => {
    // Test2 codes
  });

});
```

# Hooks

- Allow us to setup and teardown test environments for specific groups of tests.
- Helps with organizing and managing test setup and cleanup more efficiently.
- Reduces code duplication and improves readability
- Enables efficient management of resources for related tests
- Types of hooks in test groups are:
  - beforeEach(): Runs before each test in the test group
  - afterEach(): Runs after each test in the test group
  - beforeAll(): Runs once before all tests in the test group
  - afterAll(): Runs once after all tests in the test group

```
test.describe('User Authentication', () => {

  test.beforeAll(async ({ browser }) => {
    // code that runs one time before all tests
  });

  test.afterAll(async () => {
    // code that runs one time after all tests
  });

  test.beforeEach(async ({ page }) => {
    // code that runs before each test
  });

  test.afterEach(async ({ page }) => {
    // code that runs after each test
  });

  test('Test 1', async ({ page }) => {
    // Test2 codes
  });

  test('Test 2', async ({ page }) => {
    // Test1 codes
  });

});
```

# Flow of Test Hooks and Structures

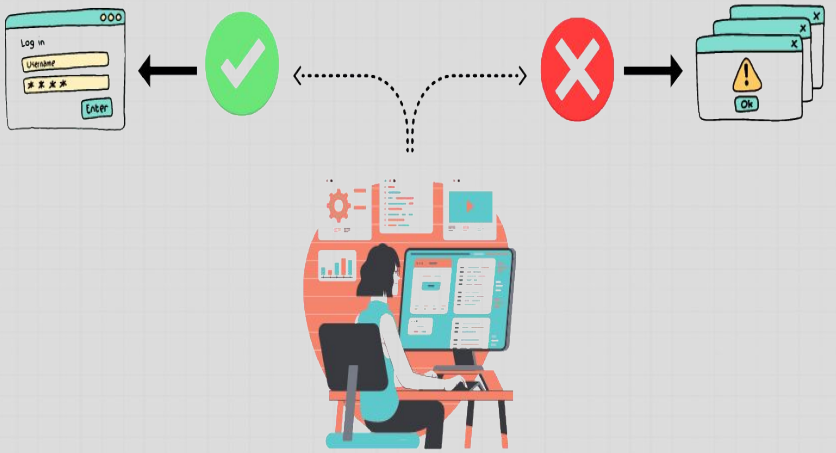
**Execution order:**

1. test.beforeAll()
2. For each test:
    - 2.1 test.beforeEach()
    - 2.2 Test Code
    - 2.3 test.afterEach()
3. test.afterAll()

```
test.describe('Group', () => {
    ├── test.beforeAll(() => { ... });
    ├── test('Test 1', async ({ page }) => { ... });
    │   ├── test.beforeEach(() => { ... });
    │   ├── Actual test code
    │   └── test.afterEach(() => { ... });
    ├── test('Test 2', async ({ page }) => { ... });
    │   ├── test.beforeEach(() => { ... });
    │   ├── Actual test code
    │   └── test.afterEach(() => { ... });
    └── test.afterAll(() => { ... });
});
```

# Playwright Assertions

- Assertions are statements in our test code that check if a condition is true.
- Helps us to validate that the software behaves as expected.
- Assertions catch errors and bugs early in the development process
- Playwright provides built-in assertion methods to make tests more reliable:



# Assertion Methods – with values

| Methods Name | Description |
|---|---|
| expect(value).toBe(expected) | Checks if value is strictly equal to expected |
| expect(value).toBeTruthy() | Checks if value is truthy |
| expect(value).toBeFalsy() | Checks if value is falsy |
| expect(value).toBeGreaterThan(number) | Checks if value is greater than number |
| expect(value).toBeGreaterThan(number) | Checks if value is greater than or equal to number |
| expect(value).toBeGreaterThanOrEqual(number) | Checks if value is less than number |
| expect(value).toBeLessThan(number) | Checks if value is less than number |
| expect(value).toBeLessThanOrEqual(number) | Checks if value is less than or equal to number. |
| expect(value).toBeNull() | Checks if value is null |
| expect(value).toContain(substring) | Checks if value contains substring |
| expect(value).toEqual(expected) | innerText() |

# Assertion Methods – with Elements

| Methods Name | Description |
|---|---|
| expect(element).toBeVisible() | Verifies that the element is visible. |
| expect(element).toBeHidden() | Verifies that the element is hidden. |
| expect(element).toBeEnabled() | Verifies that the element is enabled. |
| expect(element).toBeDisabled() | Verifies that the element is disabled. |
| expect(element).toBeEmpty() | Verifies that the element is empty. |
| expect(element).toBeChecked() | Verifies that the element is checked. (checkbox, radio button) |
| expect(element).toContainText(text) | Verifies that the element contains the specified text. |
| expect(element).toHaveText(text) | Verifies that the element's text is equal to the specified text. |
| expect(element).toHaveAttribute(name, value) | erifies that the element has the specified attribute with the specified value. |
| expect(element).toHaveCSS(name, value) | Verifies that the element has the specified CSS property with the specified value. |