**Playwrights Automation**

**Install the Playwright with command:**

npm init playwright@latest

befor writing test case we need to import some thing like

* const {test, expect} = require('@playwright/test');
* import {test, expect} from '@playwright/test'

**test:** for Test case

**expect:** for assertion

test('Home Page',async ({page})=>{

await page.goto(‘www.google.com’)

})

It test case block and it have anonymous function and we need to pass the **fixture** it called Page.

**Fixture:** is contains so many functions like find element and click element and etc.

**JavaScript promise:** async and wait is used to next step depend on previous because JavaScript have the parallel execution.

**Run command:** (npx playwright test) it is run all the test cases file.

(npx playwright test Homepage.spec.js) it is run specific the test case file.

(npx playwright test –project=chromium) it is run on only the chromium other wise it is run on all the browsers.

(npx playwright test --headed) it is run the test cases Visually otherwise it is run on headless mode.

(npx playwright test Homepage.spec.js –project=chromium –headed --debug) it is help to debug the test cases.

**Locators:**

* Property
* CSS
* XPath

For Click::

* await page.locator(‘locator’).click()
* await page.click (‘locator’)

input box:

* await page.locator(‘locator’).fill(‘value’)
* await page.locator(‘locator’).type(‘value’)
* await page.fill(‘locator’, ‘value’)
* await page.type(‘locator’, ‘value’)

Lucatemultiple web elements:

* const element = await page.$$(locator)

Builtin Locators:

* [page.getByRole()](https://playwright.dev/docs/locators#locate-by-role) to locate by explicit and implicit accessibility attributes.
* [page.getByText()](https://playwright.dev/docs/locators#locate-by-text) to locate by text content.
* [page.getByLabel()](https://playwright.dev/docs/locators#locate-by-label) to locate a form control by associated label's text.
* [page.getByPlaceholder()](https://playwright.dev/docs/locators#locate-by-placeholder) to locate an input by placeholder.
* [page.getByAltText()](https://playwright.dev/docs/locators#locate-by-alt-text) to locate an element, usually image, by its text alternative.
* [page.getByTitle()](https://playwright.dev/docs/locators#locate-by-title) to locate an element by its title attribute.
* [page.getByTestId()](https://playwright.dev/docs/locators#locate-by-test-id) to locate an element based on its data-testid attribute (other attributes can be configured).

Code Gen:

* npx playwright codegen
* npx playwright codegen -o tests/mytestfile.spec.js
* npx playwright codegen --device “iphone 13”

Hard Assertions:

Hard assertion failed assertion will terminate test execution.

Soft Assertions:

soft assertions: failed soft assertions do not terminate test execution, but mark the test as failed.

CheckBox::

await page.locator(‘locator’).check()

await page.locator(‘locator’).toBeChecked()

await page.locator(‘locator’).isChecked() return True False{.toBeTruthy(), .toBeFalsy()}

await page.locator(‘locator’).unCheck()

Dropdown Menu:

* **By label:**

await page.locator('#country').selectOption({label: 'India'})

* **By Text:**

await page.locator('#country').selectOption('India')

await page.selectOption("#country", 'India')

* **By Value:**

await page.locator('#country').selectOption({value: 'uk'})

* **By Index:**

await page.locator('#country').selectOption({index: 1})

* **Multi select:**

await page.selectOption('#colors', ['Blue', 'red', 'Yellow'])

**Alert:**

By default, dialogs are auto-dismissed by Playwright, so you don't have to handle them. However, you can register a dialog handler before the action that triggers the dialog to either **dialog**.**accept()** or **dialog.dismiss()** it.

**Frames:**

* await page.frame({url: 'https://ui.vision/demo/webtest/frames/frame\_1.html'}).fill("[name='mytext1']", 'Aakash')
* await page.frameLocator("frame[src='frame\_1.html']").locator("[name='mytext1']").fill('Aakash')

**Table:**

* const table = await page.locator('#productTable')
* const colums = await table.locator('thead tr th')
* const rows = await table.locator('tbody tr')

**Mouse Actions:**

* await page.locator("//a[normalize-space()='Desktops']").hover()
* await page.locator("//span[@class='context-menu-one btn btn-neutral']").click({button: 'right'})
* await page.locator("//button[normalize-space()='Copy Text']").dblclick()
* await page.locator("//button[normalize-space()='Copy Text']").press('Control+Shift+KeyR')
* await page.locator('#item-to-be-dragged').dragTo(page.locator('#item-to-drop-at'))

**Upload Files:**

* Select one file

await page.getByLabel('Upload file').setInputFiles(path.join(\_\_dirname, 'myfile.pdf'));

* Select multiple files

await page.getByLabel('Upload files').setInputFiles([ path.join(\_\_dirname, 'file1.txt'), path.join(\_\_dirname, 'file2.txt'),]);

**Playwright Hooks:**

* **beforeEach:** This hook is executed before each individual test.
* **afterEach:** This hook is executed after each individual test.
* **beforeAll:** This hook is executed once before any of the tests start running.
* **afterAll:** This hook is executed once after all the tests have been run.