**what is playwright?** it is a test automation tool that help us to write ui and api test automation script

**advantage of playwright?**

- autowait capability,

- support all major brower, will work on windows, macos, linux and native mobile emulation for google chrome on android and safari for ios enable

comprehensive testing across different devices and os

- provide laungage binding for js,tcs,java,python,c#

**advance feature?**

- tracing and debuging[record and playback]

- network interception

- browser context management [tranfer data to one browser to another]

- codegen tool[record and automatically write test]

**nodejs?**

it is open source server enviroment

**inilize a new node project?**

npm init

**inilize a new playwright project?**

npm init playwright

**filename should be?**

filename\_in\_camel\_case.spec.js

**test runner in playwright file is?**

playwright.config.js

**step to write basic test?**

import annotation in playwright module called test

require('@playwright/test') - store in object - called test -> const {test}=require('@playwright/test')

test function

test("first playwright test",function(){ });

test object should have two arugument first is "testname" and function where we going to write test

**js is async in nature?**

//step1

//step2

//step3

all above step will execute in a same time, that why js is called async

to stop this we need to tell js explicit wait using "await keyword"

if we use await we need to make the function async

**aynomoues function?**

function that does not contains name

async () => {} , instead of function we have arrow mark

**fixture?**

Browser will get inilize first -> context will create with own session,cookies,storage second -> page will be created mean new tab or window will created

brower -> invoke the browser and use it in function around curly bracket -> async({browser})=>{}

page -> will create a newcontext and newpage to automate -> async({browser,page})=>{}

expect –> will have assertion function

**create context?**

create a freash brower instance with proxy,plugin,cookies

context=await browser.newcontext()

await context.newPage(); -> newPage() method create a new page to automate

**how to get a page?** -> using goto() method on page fixture

Await page.goto(“enter you here”)

**Config file:**

testDir – indicate our test folder

testDir: './tests',

timeout – how many time the test should wait by default it will be 30

  /\* Maximum time one test can run for \*/

  timeout: 30 \* 1000,

expect - it is for assertion

reporter – how do we want to create a report

/\* Reporter to use. See https://playwright.dev/docs/test-reporters \*/

  reporter: 'html',

use – test cases will all the property inside the use property, what browser need to execute, screenshot, retry, etc.

  use: {

    /\* browser need to run \*/

    browserName:'chromium'

/\* mode of running \*/

    headless:false

  },

Except – we can give n times to wait for assertion

expect:{

    timeout: 5000

  }

**By default playwright will run on ?** headless mode

For run playwright test? We need to trigger our test runner config file to run test

**Command to run playwrigth?** Npx playwright test

* Npx playwright – find the playwright in our node module
* Test – find the test runner config file
* It will go to test folder and trigger all two test cases one by one

**Run test on headed mode?** Npx playwright test --headed

**Run only one single test?** Test.only(‘test name’,()=>{})

**Get title from page?** Awiat page.title() function help to get title

**Assert title?** Playwright has default assertion

Await expect(page).toHaveTitle(“Google”)

**Locators:**

It is from page fixture

Most of time we need to use css selector

**If ID is present** = Css -> tagname#id or #id

**If class attribute is present** = Css -> tagname.class or .class

**Write css based on any attribute** = Css -> [attribute=”value”]

**Write css with traversion from parent to child** = Css -> parenttagname >> childtagname

**If need to write the locator based on text** = Text =’ ‘ (or) tagname:has-text(value) give inside the locator

page.locator("h3:has-text('qwerty12')")

Example:-

test.only("third playwright test",async ({page})=>{

    await page.goto("https://rahulshettyacademy.com/loginpagePractise/")

    await page.locator("#username").fill("rahulshetty")

    await page.locator("[id='password']").fill("learning")

    await page.locator("#signInBtn").click()

    //webdriverwait for error machnisms

    console.log("my message:- "+await page.locator("[name='loginForm'] div:nth-child(1) strong").textContent())

})

Enter text on input box: await.locator().fill() – fill method enter the value on text box

How to get the innertext on the element: await.locator.textContent() – get the text form the element

How to assert the value is present on the expected value: await expect(locator).toContains(expected value) – “toContains()” check the expected value is presented

How to erase the value in the inputbox:

 await page.locator("#username").fill("")

 await page.locator("#username").fill("rahulshettyacademy")

How to handle if a css selector has multiple element matches:

We can use “first() or nth(index number)”

console.log(await page.locator('.card .card-body h4 a').first().textContent())

    // or

    console.log(await page.locator('.card .card-body h4 a').nth(1).textContent())

How to get all innerText of the element

Using the allTextContents() return the array, it does not have syncronization

await page.locator('.card .card-body h4 a').allTextContents()

How to handle the wait dynamically in new page

await page.waitForLoadState("networkidle")

**waitForLoadState is used to handle the dynamic load stategery**

networkidle – api call is fully loaded

domcontentloaded – html document is fully loaded

alternative for the waitForLoadSate is waitFor() – but it will wait for only one element

**how to run a specific test file**

npx playwright test tests/testfilename

**how to pause the test execution**

playwright pause() function we can do it

await page.pause(); // it will stop the execution open the playwright inspector

**how to handle static dropdowns**

using selectOption(“value”) we can select

const dropdown=await page.locator("select.form-control")

    await dropdown.selectOption("Teacher")

**how to handle radio button and check it is selected**

toBeChecked() make the assertion value, isChecked() return the boolean value is checked

await page.locator("span.radiotextsty").last().click()

await page.locator("#okayBtn").click()

console.log(await page.locator("span.radiotextsty").last().isChecked())

expect(page.locator("span.radiotextsty").last()).toBeChecked()

**how to handle the checkbox and make a check and uncheck the box**

isChecked() – return boolean value and check the box is checked

click() – check the box

uncheck() – return boolean value and if true uncheck() the box

    await page.locator("#terms").isChecked()

    await page.locator("#terms").click()

    await page.locator("#terms").uncheck()

**how to check if the condition return only return false**

toBeFalsy() – method expect the assertion to fail (or) toBeTruthy() – method expect the assertion to true

expect(await page.locator("#terms").isChecked()).toBeFalsy()

**where should we use await?**

When ever we performing some action we need to do await()

**How to automate blinking text?**

Every blink element has a class=”blinkingText” we can automate with that

await expect(page.locator(".blinkingText")).toHaveAttribute("class","blinkingText")

**How to check the element has this attribute?**

toHaveAttribute(“type”,”value”) method will check attribute is present

**How to do window handling in playwright?**

Three type of promises

Pending,rejected,fulfilled

When do you need this type of action like having a newPromise.all() when performing two action in same time, we need to inform playwright before a new tab is open

const context=await browser.newContext()

    const page=await context.newPage();

    await page.goto("https://rahulshettyacademy.com/loginpagePractise/")

    const documentlinks=page.locator("[href\*='documents-request']")

    const [newPage]=await Promise.all([

        context.waitForEvent("page"), // listen when a new page is opening

        // wait for promises: three type of promisis -pending,rejected,fullfiled

        documentlinks.click(),

    ])

    const email=await newPage.locator('[href\*="mailto"]').textContent()

    console.log(email.split('@'))

if you want to comeback to previous page we can use this,

page.locator("#username").fill(domain)

use parent window name

**what is playright inspector?**

it will help to debug and autocreate the locator

Two way to open playwright using

* Await page.pause()
* Npx playwright test tests/filename.spec.js –debug

**What is codegen?**

It help to record and playback with generated autmation script

Npx playwright codegen base\_url

**How to add screenshot?**

 use: {

    /\* Base URL to use in actions like `await page.goto('/')`. \*/

    // baseURL: 'http://127.0.0.1:3000',

    /\* Collect trace when retrying the failed test. See https://playwright.dev/docs/trace-viewer \*/

    trace: 'on-first-retry',

    /\* browser need to run \*/

    browserName:'chromium',

    /\* mode of running \*/

    headless:false,

    /\* screenshot \*/

    screenshot:'on'

  },

**How to trace the log?**

Trace property under the user will help to track the log

**How to create a reports?**

  /\* reporting \*/

  reporter:'html'

**how to check the item is visible?**

await expect(page.locator("h3:has-text('qwerty12')")).toBeVisible()

or we can have isVisible() return boolean value

**how to enter letter by letter in a word on textbox?**

We can use pressSequentially(word) – it will enter the letter on by one

How to click the element by label?

Happen only for checkbox and radio button and static dropdown

await page.getByLabel("Check me out if you Love IceCreams!").click()

    await page.goto("https://rahulshettyacademy.com/angularpractice/")

    await page.getByLabel("Check me out if you Love IceCreams!").isChecked()

    await page.getByLabel("Employed").click()

    await page.getByLabel("Gender").selectOption("Male")

**how to run the playwright ui runner?**

Npx playwright test tests/testname.spec.js –ui

It has all the step by step screenshot

**How to handle element by placeholder?**

await page.getByPlaceholder("Password").fill("admin")

**How to get element by the role?**

Page.getByRole(“tagname,{attribute:value})

await page.getByRole("button",{name:"Submit"}).click()

**How to get element by the text?**

Page.getByText(“textvalue”)

**Ho w to click by link?**

await page.getByRole("link",{name:"Shop"}).click()

**How to filter the text from the parent tag?**

Page.getByRole(parenttagname).filter({propertyname:value}))

await page.getByRole('app-card').filter({hasText:'iphone X'}).click()

**getByLocators**

    await page.getByLabel("Check me out if you Love IceCreams!").isChecked()

    await page.getByLabel("Employed").click()

    await page.getByLabel("Gender").selectOption("Male")

    await page.getByPlaceholder("Password").fill("admin")

    await page.getByRole("button",{name:"Submit"}).click()

    await page.getByRole("link",{name:"Shop"}).click()

    await page.locator('app-card').filter({hasText: 'Nokia Edge'}).getByRole("button").click()

**How to handle the calender?**

const monthNumber="6"

    const date="15"

    const year="2027"

    const expectedList=[monthNumber,date,year]

    await page.goto("https://rahulshettyacademy.com/seleniumPractise/#/offers")

    await page.locator(".react-date-picker\_\_inputGroup").click()

    await page.locator(".react-calendar\_\_navigation\_\_label").click()

    await page.locator(".react-calendar\_\_navigation\_\_label").click()

    await page.getByText(year).click()

    await page.locator(".react-calendar\_\_year-view\_\_months\_\_month").nth(monthNumber).click()

    await page.locator(".react-calendar\_\_month-view\_\_days\_\_day").nth(date).click()

    const inputs=await page.locator(".react-date-picker\_\_inputGroup input")

    for(let index=0;index<inputs.length;i++){

        const values=inputs[index].getAttribute("values")

        expect(values).toEqual(expectedList[index])

    }

**Goback() and GoForward()?**

    await page.goto("https://www.google.com")

    await page.goBack();

    await page.goForward();

**how to handle element is in hidden mode?**

    await expect(page.locator("#displayed-text")).toBeVisible()

    await page.locator("#hide-textbox").click()

    await expect(page.locator("#displayed-text")).toBeHidden()

**how to handle alert box?**

    page.on("dialog",dialog=>dialog.accept())

    //page.on("dialog",dialog=>dialog.dismiss())

    await page.locator("#alertbtn").click()

**how to handle hover?**

await page.locator("#alertbtn").hover()

**how to handle frame?**

frameLocator() method is help to move from page context to frame

   const framePage=page.frameLocator("courses-iframe")

    await framePage.locator("li a[href\*='lifetime-access]").nth(1).click()