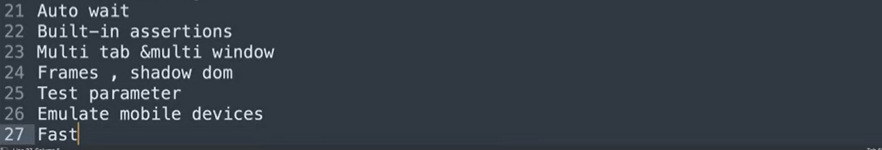
**A screenshot of a computer program

AI-generated content may be incorrect.**

****

**A screenshot of a computer program

AI-generated content may be incorrect.**

**What is Playwright ?**

Playwright Test was created specifically to accommodate the needs of end-to-end testing.

Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox.

Test on Windows, Linux, and macOS, locally or on CI, headless or headed with native mobile emulation of Google Chrome for Android and Mobile Safari.

Free and Open source framework, it is created by Microsoft and used for automation

on web browser applications

Application: Web browser apps | Mobile web apps | API

Languages Supported: Node.JS | Java | Python | .Net (C#)

Browsers: all modern engines chromium, WebKit and Firefox (Headless or headed)

Chromium: Google Chrome, Edge, Opera

WebKit: MAC OS, IOS browsers --> Like Safari

Firefox: All the version of Firefox browser (Desktop Firefox browser, Mobile Firefox browser)

Headless --> No GUI, you will not see anything on the screen, Save time, Save Memory

Headed --> Physical browser on screen

OS Supported: Windows, MAC, Linux

https://playwright.dev/

https://github.com/microsoft/playwright-java

**Features**:

Free | Open Source

Multi-browser | Multi-Language | Multi-OS

Easy Setup and configurations

Functional | API | Accessibility testing

CI CD | Docker support

Recording | Debugging | Explore selectors

Parallel testing

Auto-Wait

Build-in assertions | Less Flaky tests

Tests retry, logs, screenshots, videos

Multi-tab and Multi window

Frames | Shadows DOM

Emulate mobile devise, geolocations

Test parameterization --> Add test data using csv files ,Data driver test

Installation of Playwright:

Pre-requisites:

1. Install node js
   1. Npm --version
   2. Node -- version
2. IDE (VS Code)

Steps:

1. Create a new folder and open it in VS Code
2. Goto terminal and run the command -npm init paylwrite@latest

npx playwright test

run the end-to-end test

npx playwright test –project== chromium

run the tests only on Desktop Chrome

npx playwright test example

run the tests in a specific file

npx playwright test –debug

run the tests in debug mode

npx playwright test codegen

Auto generate tests with Codegen

1. Following will be added
   1. Package. Json --> Node project management file
   2. Playwright.config.ts --> Configuration file
   3. Tests folder --> Basic example test
   4. Tests-examples folder --> details example tests
   5. Gitignore --> to be used during git commit and push
   6. Playwright.yaml --> to be used during ci cd pipeline

A screenshot of a computer program

AI-generated content may be incorrect.

Some commands:

npm playwright -v --> Version of Playwright

npx --> Execute the npm library in local folder

npx playwright –help

**Using VS Code extension:**

1. Create a new folder and open in VS Code
2. Goto extensions section and install playwright extension from Microsoft.
3. Goto view --> Command palette and type playwright --> select install playwright

Command 1: Run the first program

Npm playwright test --> Once test execution completed , a folder playwright-report is created with file name index.html

A screenshot of a computer program

AI-generated content may be incorrect.

Report file path: C:\NewPlayWriteProejct\playwright-report

Run the following command to open the test report file

Command 2: npx playwright show-report

Command 3: npx playwright test --workers 2 --> Runs with 2 workers in parallel

Command 4: npx playwright test one.spec.js --> Runs a specific test file

Command 5: npx playwright test one.spec.js two.spec.js --> Runs the files specified

Command 6: npx playwright test one two --> Runs files that have one or two in the file name

Command 7: npx playwright test -g “check title” --> Runs test with title

Command 8: npx playwright test --project=chromium --> Runs on a specific browser

Command 9: npx playwright test --headed --> Runs tests in headed mode

Command 10: npx playwright test --debug --> Runs tests in debug mode

Command 11: npx playwright test example.spec.js --debug --> debug specific test file

Command 12: npx playwright test example.spec.js:21 –debug --> debug starting from specific line where test (.. starts)

Write first test in Playwright

Step 1: create a file in **test** folder

Step 2: Add module Playwrite/test

const{test,expect}= require('@playwright/test')

**Playwright with Java**

Playwright is a Node.js library to automate Chromium, Firefox and WebKit with a single API. Playwright is built to enable cross browser web testing.

WebKit? --> WebKit is a browser engine developed by Apple and used in its Safari web browser as well as all iOS web browsers.

* Cross – Browser (Chromium, Firefox and WebKit)
* Cross – Platform (Windows, Linux and macOS , locally or on CI , headless or headed
* Cross -Language(JS, TS, Java , Python and .Net
* Test Mobile web – Native mobile emulation of Google Chrome for Android and Mobile Safari

No Flaky tests:

* Auto-wait
* Web first assertion
* Tracing (DOM Scapshot)

No Limits:

* Multiple everything -Multiple tabs, browser , user
* Trusted events – drag & drop, hover and etc
* Frames Shadow support

Powerful Tooling

* Codegen
* Playwright inspector
* Trace viewer

Setup Java Maven --> Create Maven Project --> Add playwright dependency --> Write and run first test case

Create project structure --> Add TestNG --> Run with Testng.xml --> Recording --> POM -->Reports --> CMD

Installation:

1. Download and install JDK
2. Download Maven
3. Add Java bin path and Maven bin path in system environment variables
4. Download and install/extract Eclipse editor

Create project:

1. Open Eclipse and create a Maven project

Group id: Like a folder name

Artifactory Id: Like Project name

Fill other detail and click on Save

1. Open POM.xml and add playwright dependencies

<https://playwright.dev/java/docs/next/intro> --> Copy pom.xml

<properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>com.microsoft.playwright</groupId>  
 <artifactId>playwright</artifactId>  
 <version>1.53.0</version>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.10.1</version>  
 *<!-- References to interface static methods are allowed only at source level 1.8 or above -->*  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

A screenshot of a computer

AI-generated content may be incorrect.

Line 17: Creating instance for playwright

Line 18: Launching chrome browser in headed mode

Line 21: Page – page refence (like driver in selenium)

Line 23: Locator --> Find the locators (as Find Element in Selenium)

@Test

public void registerUser()

{

Playwright pw= Playwright.create();

BrowserType browserType= pw.Chromium();

Browser browser= browserType.launch(New BrowserType.LaunchOptions(setHeadless(false)).setSlowM(3000);

Page page= browser.newPage();

page.locator(".subLink").click();

page.getAtPlaceHolder("name").fill("random\_");

page.getAtPlaceHolder("Email").fill("random\_"+ System.CurrentTimeMills()+"@gmail.com");

page.locator("xpath=//input[@type='password']").fill("Pass1234");

PagewrightAssertion.assertType(page.locators("xpath=//label[text()='Cypress']//preceding::input[1]").isChecked();

page.locator("xpath=//label[text()='Cypress']//preceding::input[1]").click();

page.locator("#gender1").click();

page.locator("#state").selectOptions("Goa"); // Single value selection

String hobbies={"Playing","Reading"};

page.locator(#hobbies).selectOptions(hobbies) ; // Multiple values selection

PagewrightAssertion.assertType(page.locator(".submit-btn")).isEnabled();

page.locator(".submit-btn").click();

PagewrightAssertion.assertType(page.locator("xpath=//div[text()='Signup successfully, Please login!']")).isDisplayed(); }

**Playwright with Java Script**