Inside directory, you can run several commands:

npm playwright -v //to check version

npm playwright -help

npm init playwright@latest \\to run playwright

**npx playwright test** Runs the end-to-end tests.

npx playwright show-report

npx playwright test --workers 3 \\run all tests in 3 configured browsers at once

npx playwright test .\tests\example.spec.js \\execute only this test

npx playwright test .\tests\example \\execute all tests with “example” in file name

npx playwright test -g “has title“ \\execute test by test name

npx playwright test --project=chromium \\execute only on chromium

npx playwright test –project chromium --headed \\you can see how Playwright use browser

**npx playwright test alza.spec.js --headed --project="chromium" --config=../playwright.config.js**  [\\run](file:///\\run) with specific config file

**npx playwright test --list --config=../playwright.config.js** //validate loaded config

**npx playwright test --headed //**you can see how Playwright use browser  
**npx playwright test --headed --project chromium**

https://playwright.dev/docs/running-tests#run-tests-in-headed-mode

**npx playwright test --project=chromium --debug** [\\run](file:///\\run) test with playwright inspector

npx playwright test .\tests\example.spec.js --debug

npx playwright test example.spec.js:12 --project=chromium --debug [\\debug](file:///\\debug) specific test on specific line(12)

npx playwright show-report

npx playwright test –ui // Starts the interactive UI mode.

npx playwright test --project=chromium //Runs the tests only on Desktop Chrome.

npx playwright test example

Runs the tests in a specific file.

npx playwright test --debug

Runs the tests in debug mode.

npx playwright codegen

Auto generate tests with Codegen.

We suggest that you begin by typing:

npx playwright test

And check out the following files:

- .\tests\example.spec.ts - Example end-to-end test

- .\tests-examples\demo-todo-app.spec.ts - Demo Todo App end-to-end tests

- .\playwright.config.ts - Playwright Test configuration

Visit https://playwright.dev/docs/intro for more information. ✨

**CODEGEN**

npx playwright codegen --help

npx playwright codegen

npx playwright codegen <https://www.alza.sk/>

npx playwright codegen <https://www.saucedemo.com>

npx playwright codegen --browser firefox

or

npx playwright codegen --browser ff

npx playwright codegen --target javascript -o .\tests\record\_demo.specs.js

npx playwright codegen --target javascript -o record\_demo.specs.js

[\\opened](file:///\\opened) recorder will save script to the defined file in javascript

npx playwright codegen --viewport-size=800,600 [\\run](file:///\\run) with defined resolution

npx playwright codegen --device=”iPhone 14”

console.log(await page.viewportSize()); // Log the viewport size for debugging

npx playwright test iphone14.spec.js --headed --project="chromium" --config=../playwright.config.js

npx playwright codegen --device=”iPhone 14” --color-scheme=dark playwright.dev

**await page.pause() //setting this step into the test will start Playwright inspector on specific step**

**TRACE VIEWER**

playwright.spec.js option: **trace**: 'on-first-retry',

//retries: process.env.CI ? 2 : 0, -commented

retries: 1,

**trace** can also be set to **on, off, retain-on-failure** (only failed)

**npx playwright test --trace on** [\\can](file:///\\can) set in every run

how to decrease default wait timeout - playwright.spec.js - timeout: 10 \* 1000, timeout: 3000

npx playwright test saucedemo1.spec.js --headed --project="chromium" --config=../playwright.config.js

When report is shown after test (or npx playwright show-report), **check Retry tab, at the bottom are “Traces”,** click on and open Trace viewer

or open .zip in “test-results” folder: npx playwright show-trace .\test-results\saucedemo1-saucedemo1-chromium-retry1\trace.zip

trace.playwright.dev [\\open](file:///\\open) trace file manually

to track specific part of test

test('has title', async ({ page, context }) => {  
await context.tracing.start({snapshots: true, screenshots: true})

//test code

await context.tracing.stop({path: 'test1.zip'});

npx playwright show-trace .\test1\_trace.zip //to run trace

**Selectors and Locators /** **using any object property**

https://playwright.dev/docs/api/class-locator

await page.pause() //open playwright inspector in specific step

 await page.click('id=user-name')

 await page.locator('id=user-name').fill(Edison)

or

await page.locator('[id=“user-name“]').fill(Graham)

**CSS Selector**

copy from dev tool

A screenshot of a computer

Description automatically generated

**xpath**

//textarea[@name="q"]

A screenshot of a computer

Description automatically generated

  await page.locator('xpath=//input[@id="user-name"]').fill('Faraday')

  await page.locator('//input[@id="user-name"]').fill('Ramujan')

**TEXT**  // using Text

  await page.locator('text=LOGIN').click()

  await page.locator('input:has-text("LOGIN")').click(); //here we can define „type“ of element with input:

**Actions over elements**

| **Action** | **Method** | **Description** |
| --- | --- | --- |
| **Clicking Elements** | locator.click() | Clicks on an element. |
|  | page.click() | Clicks on a page element directly, often requiring a selector. |
| **Typing into Inputs** | locator.type() | Types text into an input field. |
|  | page.type() | Types text into an input field directly via a selector. |
| **Hovering** | locator.hover() | Moves the mouse over an element. |
|  | page.hover() | Hover over an element on the page via a selector. |
| **Checking and Unchecking** | locator.check() | Checks a checkbox or radio button. |
|  | locator.uncheck() | Unchecks a checkbox or radio button. |
| **Selecting Options** | locator.selectOption() | Selects an option from a <select> dropdown. |
|  | page.selectOption() | Selects an option from a dropdown using a selector. |
| **Filling Forms** | locator.fill() | Fills an input field with a specified value. |
|  | page.fill() | Fills an input field directly via a selector. |
| **Pressing Keys** | locator.press() | Simulates pressing a key (e.g., ENTER, ESC) on a web element. |
|  | page.press() | Simulates pressing a key directly on an element via selector. |
| **Clearing Inputs** | locator.clear() | Clears the content of an input field. |
| **Dragging and Dropping** | locator.dragTo() | Drags an element to a target location. |
| **Waiting for Elements** | locator.waitFor() | Waits for an element to meet a specific condition (e.g., visible, attached). |
| **Action** | Example Code | Description |
| **Click** | element.click(); | Simulates a click action on a web element. |
| **Send Keys (Typing)** | element.sendKeys("Test Input"); | Types the specified text into an input field. |
| **Get Text** | String text = element.getText(); | Retrieves the visible text of a web element. |
| **Mouse Hover** | Actions action = new Actions(driver); action.moveToElement(element).perform(); | Hovers the mouse over a specific element using the Actions class. |
| **Double Click** | Actions action = new Actions(driver); action.doubleClick(element).perform(); | Performs a double-click on a web element using the Actions class. |
| **Right Click (Context)** | Actions action = new Actions(driver); action.contextClick(element).perform(); | Opens the context menu by right-clicking on an element. |
| **Drag and Drop** | action.dragAndDrop(source, target).perform(); | Drags one element and drops it onto another. |
| **Select Option** | Select select = new Select(dropdown); select.selectByVisibleText("Option Text"); | Selects a value from a dropdown by visible text. |
| **Get Attribute** | String value = element.getAttribute("attributeName"); | Fetches the value of a specified attribute from the element. |
| **Wait for Element** | WebDriverWait wait = new WebDriverWait(driver, 10); wait.until(ExpectedConditions.elementToBeClickable(locator)); | Waits until a condition (e.g., element clickable) is met before proceeding. |