

Playwright

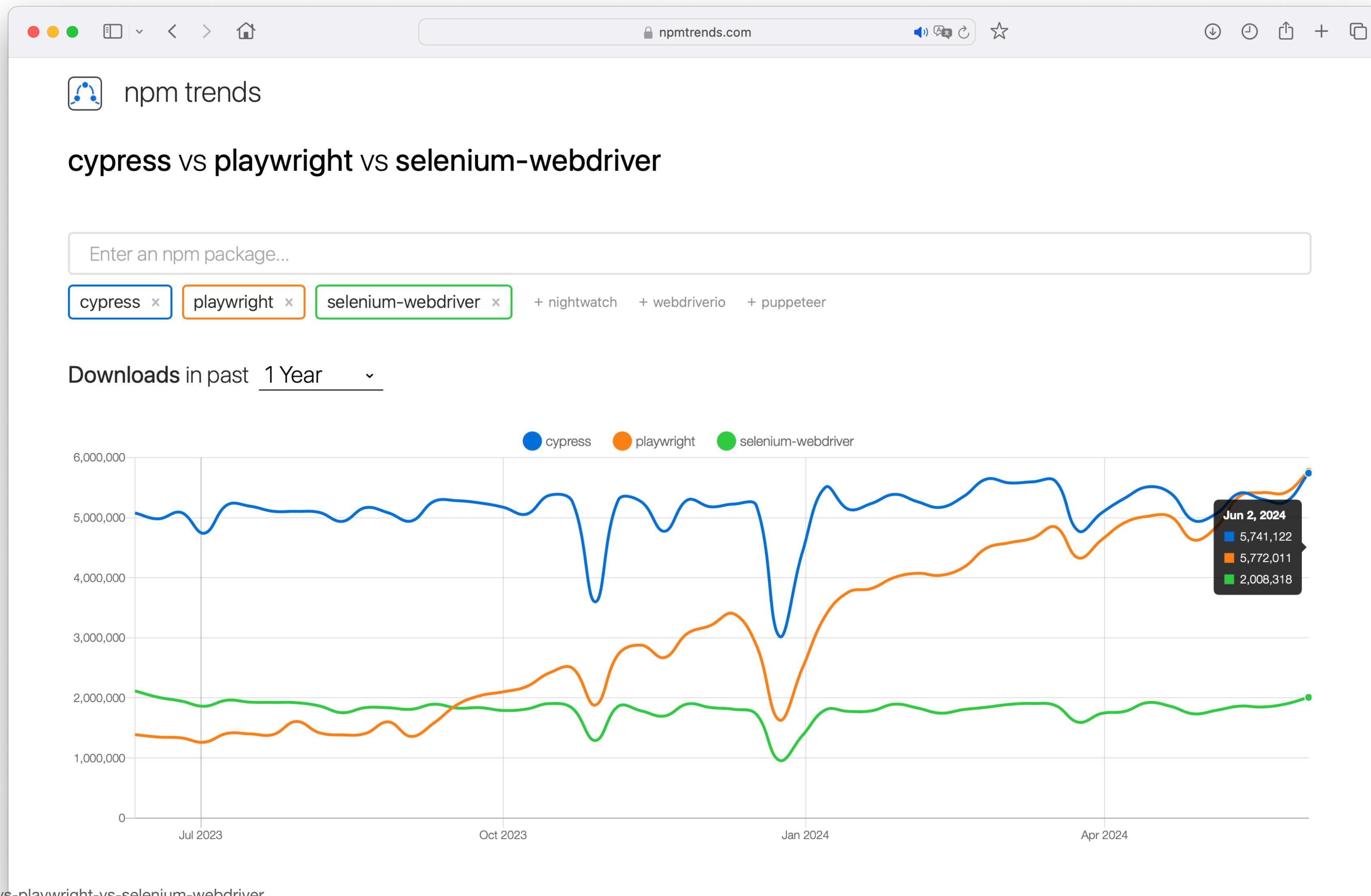
An open-source browser automation library developed by Microsoft.

It supports multiple web browsers, including Chromium, Firefox, and WebKit.

Playwright allows developers to automate browser tasks such as page navigation, form submission, and interaction with web elements.



NPM download trends



Frameworks Comparison



Aspect	Robot Framework	Playwright	Cypress
Type	Keyword-driven test automation framework	Node.js library for browser automation	JavaScript end-to-end testing framework
Language	Uses its own domain-specific language (keywords)	JavaScript, Python, .NET APIs available	JavaScript
Focus	General-purpose, supports various types of testing	Browser automation across different browsers	Web application end-to-end testing
Browser Support	Supports multiple browsers through Selenium	Supports Chromium, Firefox, WebKit	Primarily focused on Chrome
User Interfaces Support	Supports web, mobile, and desktop automation	Web and browser automation only	Web browser automation only
Parallel Execution	Supports parallel execution of tests	Supports parallel execution	Supports parallel execution
Community & Ecosystem	Active community and many libraries	Rapidly growing ecosystem and active development	Active community and plugin ecosystem
Testing Level	Supports unit, integration, and acceptance testing	Primarily focused on end-to-end testing	Primarily focused on end-to-end testing
Flexibility	Highly flexible due to keyword-driven nature	Flexible and powerful due to code-based approach	Flexible and powerful due to code-based approach
Maintenance	Requires maintenance of keyword libraries	Requires maintenance of codebase	Requires maintenance of codebase
Popularity	Widely used in test automation	Gaining popularity, especially for modern web apps	Popular choice for web testing

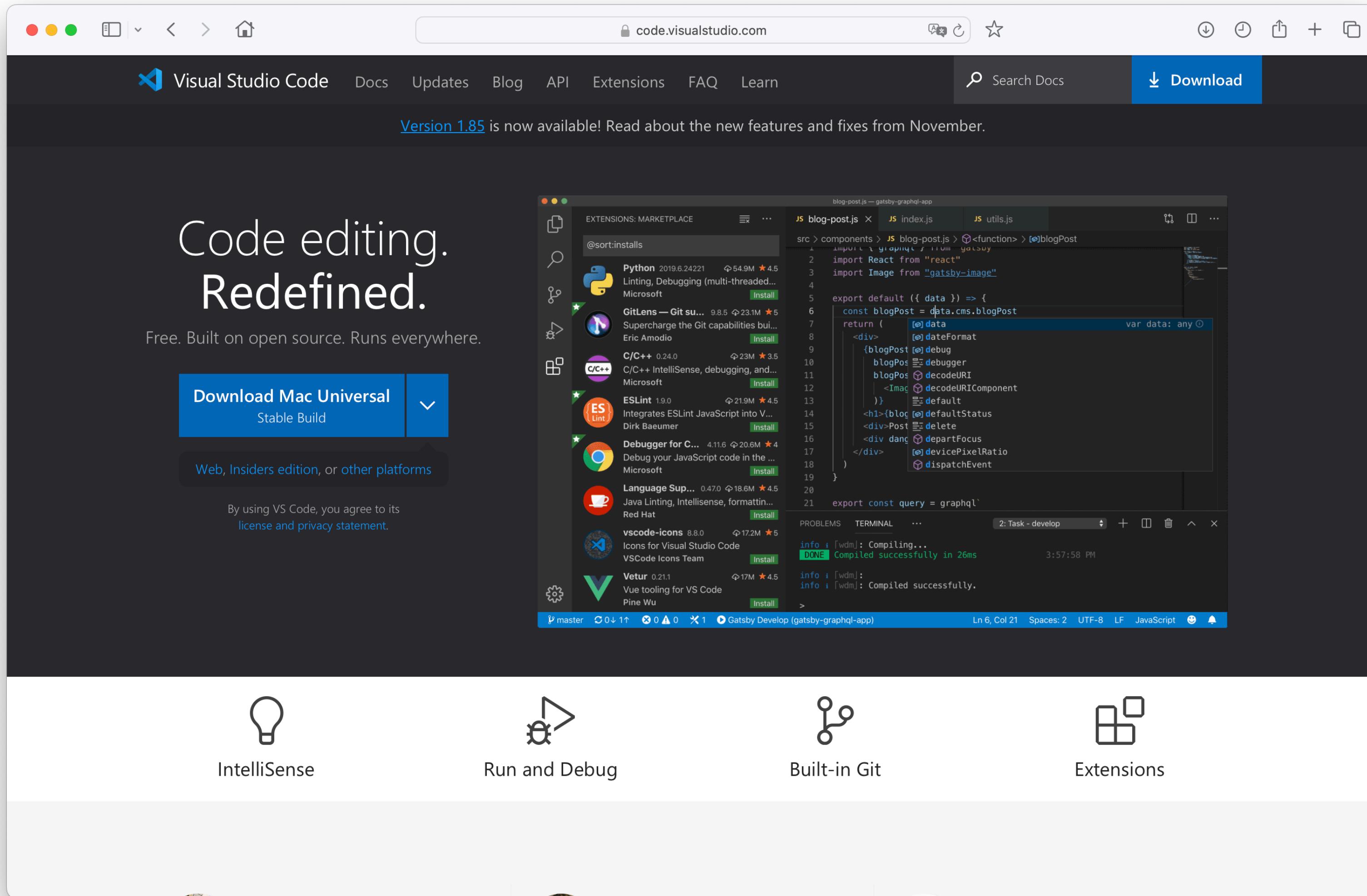
Installations

- VSCode
- Windows
 - Node.js
- Mac
 - Brew
 - Node.js

Install VSCode

Go to this official website => <https://code.visualstudio.com/>

It has already suggested the compatible version with your environment



Windows: Download Node.js

Go to this official website => <https://nodejs.org/en/download> , then download the Windows version

The screenshot shows the Node.js download page. At the top, there are two tabs: "LTS" (Recommended For Most Users) and "Current" (Latest Features). The "LTS" tab is selected. Below the tabs, there are three main download options:

- LTS (Windows Installer):** node-v20.10.0-x64.msi
- Current (macOS Installer):** node-v20.10.0.pkg
- Current (Source Code):** node-v20.10.0.tar.gz

Below these, there is a table showing download links for additional platforms:

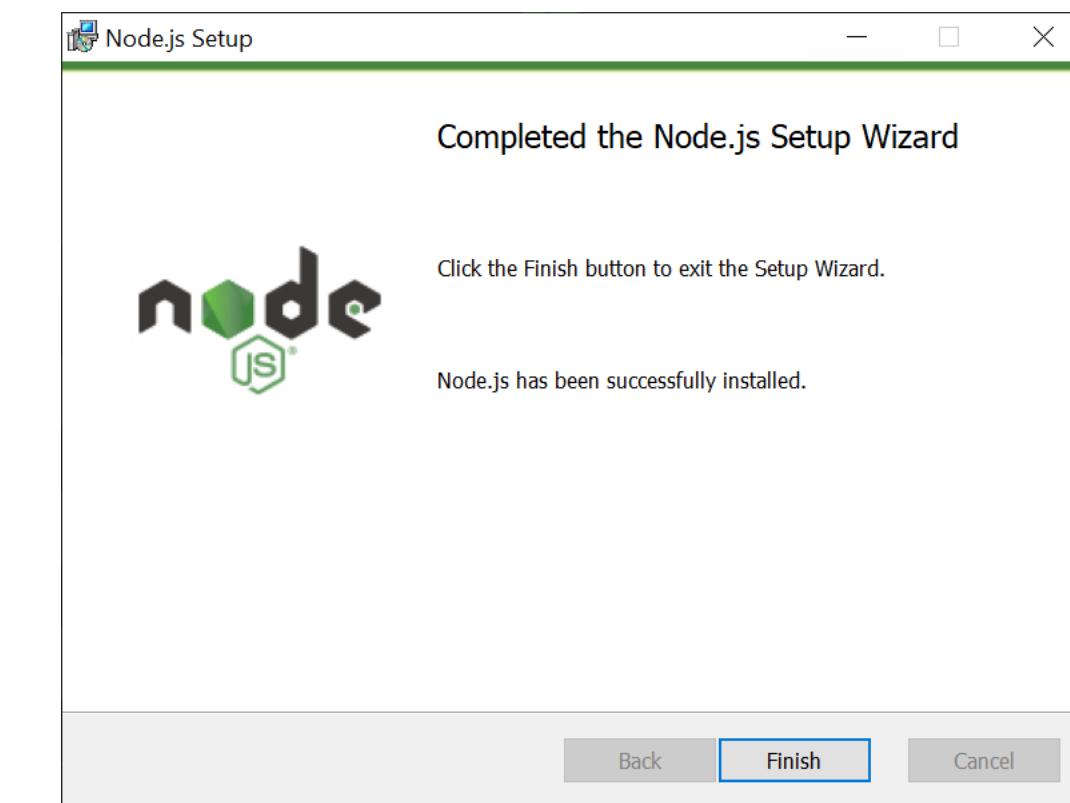
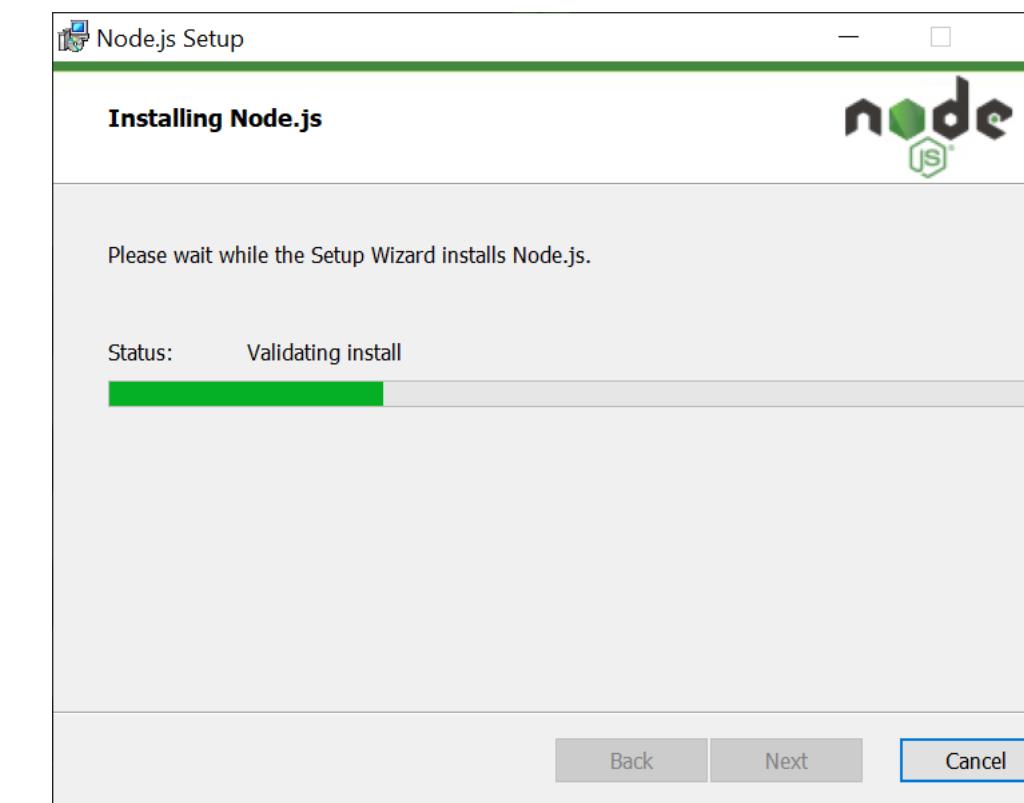
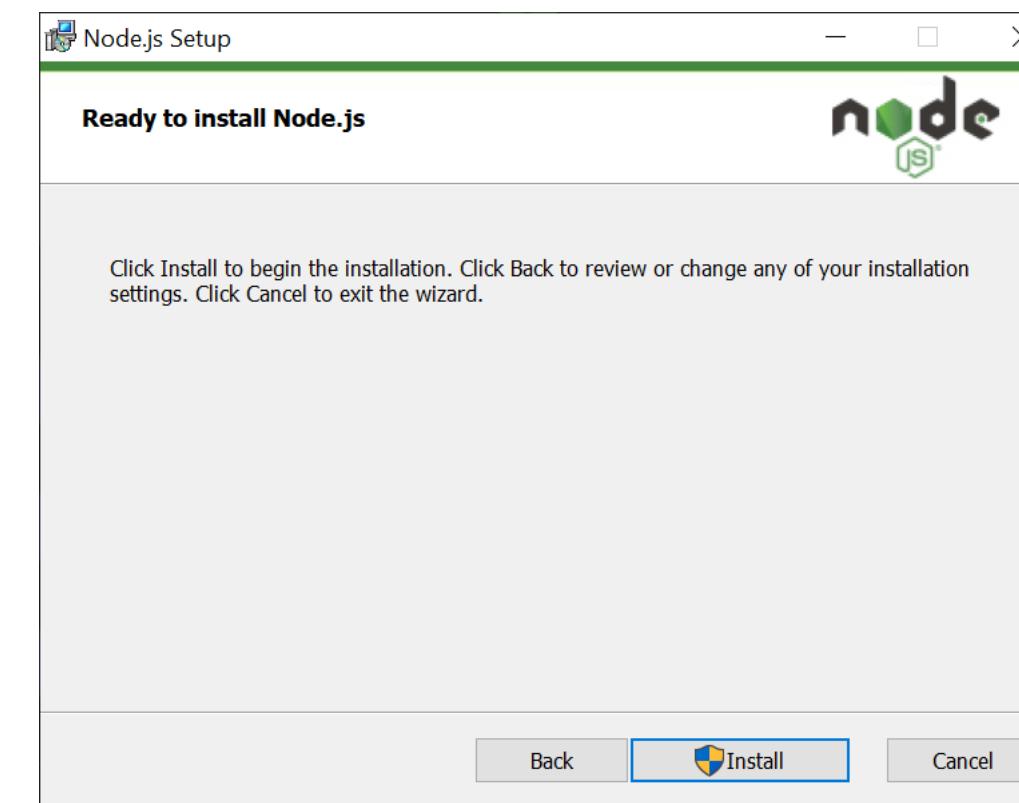
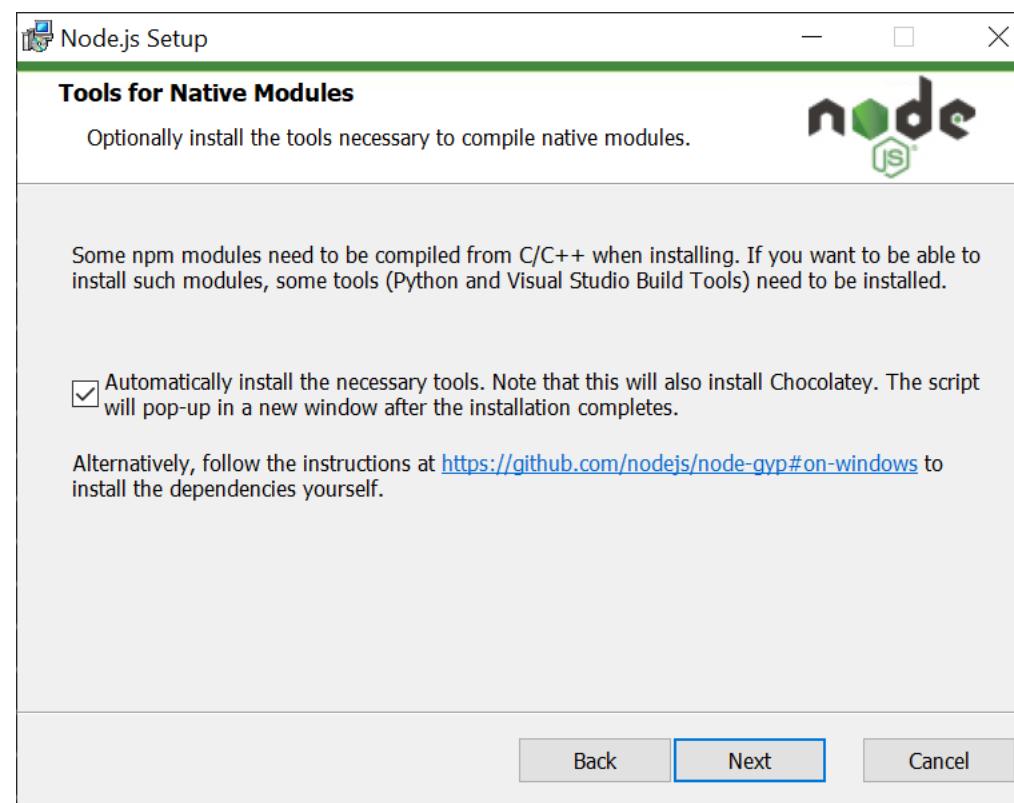
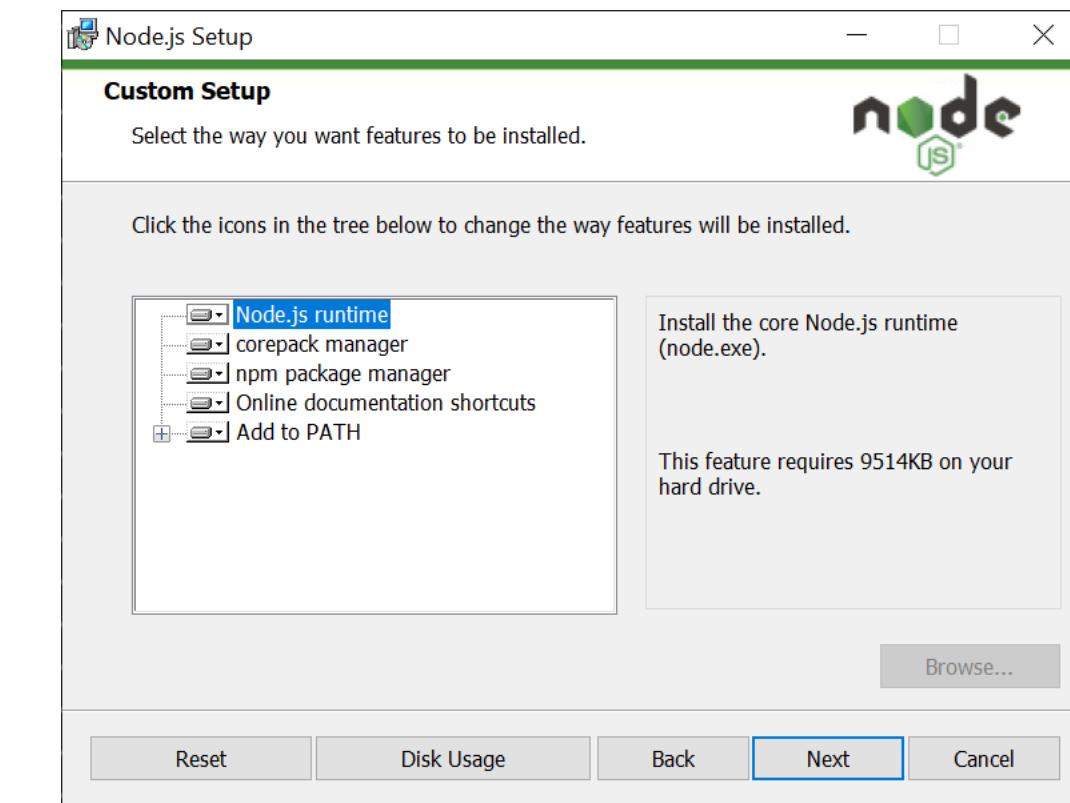
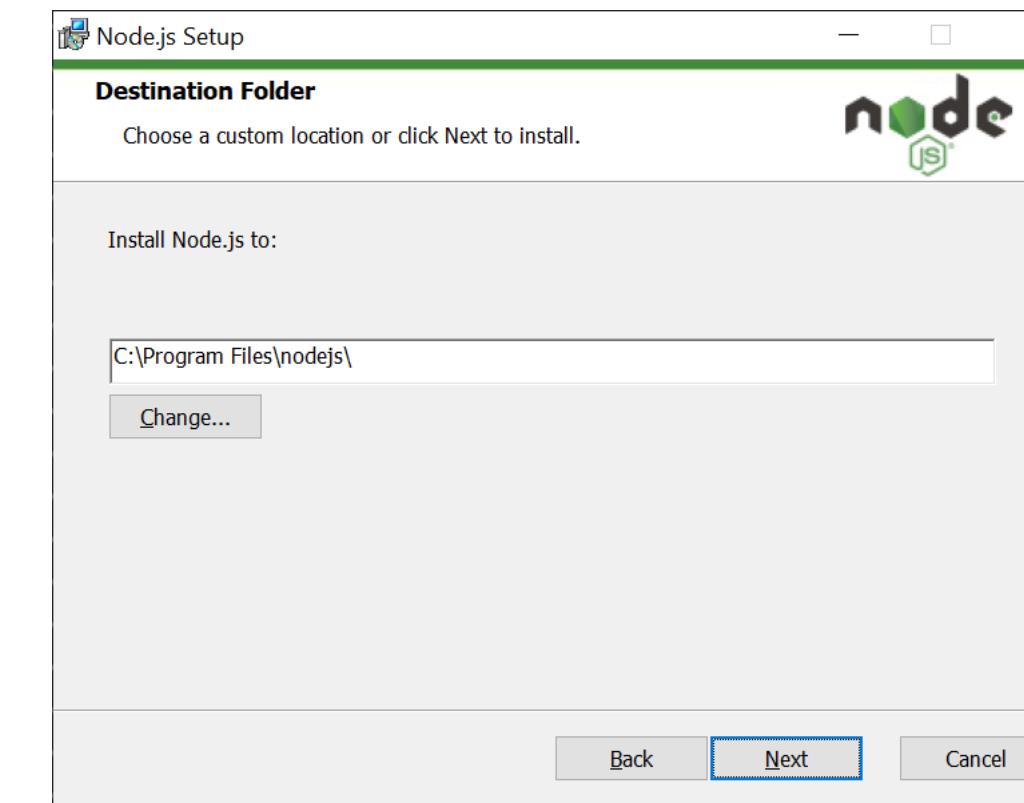
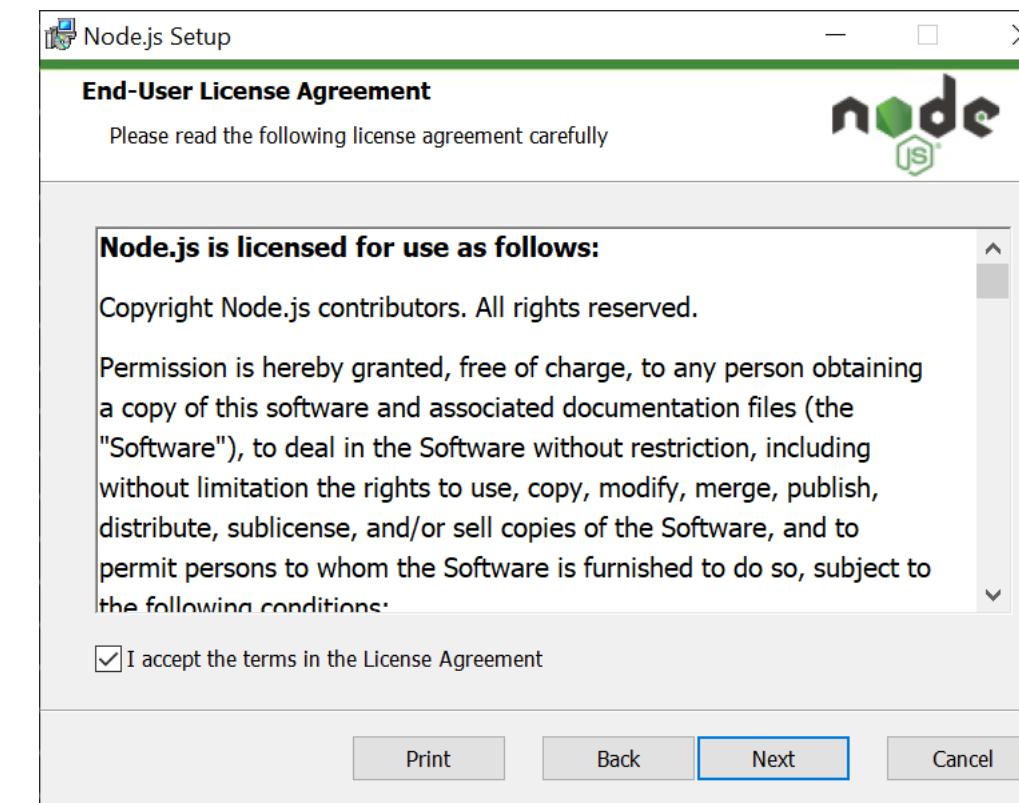
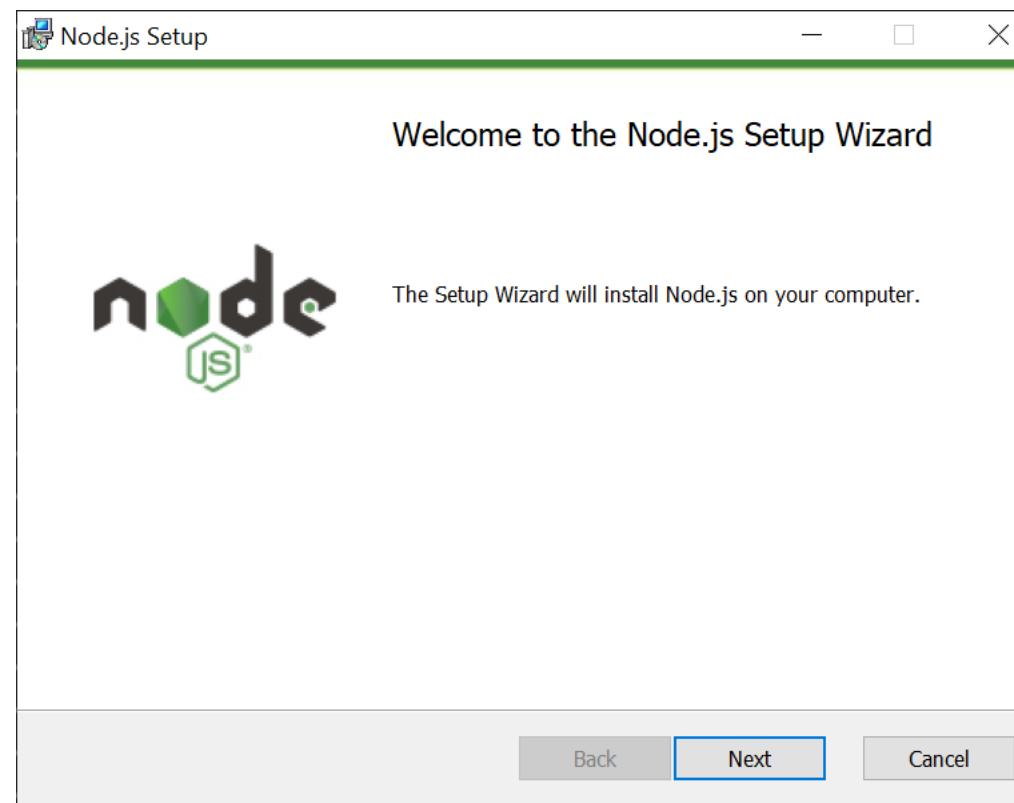
	32-bit	64-bit	ARM64
Windows Installer (.msi)	node-v20.10.0-win32-ia32.msi	node-v20.10.0-win32-x64.msi	node-v20.10.0-win64-ia32.msi
Windows Binary (.zip)	node-v20.10.0-win32.zip	node-v20.10.0-win32-x64.zip	node-v20.10.0-win64-ia32.zip
macOS Installer (.pkg)	node-v20.10.0-macosx-ia32.pkg	node-v20.10.0-macosx-x64.pkg	node-v20.10.0-macos-arm64.pkg
macOS Binary (.tar.gz)	node-v20.10.0-macosx-ia32.tar.gz	node-v20.10.0-macosx-x64.tar.gz	node-v20.10.0-macos-arm64.tar.gz
Linux Binaries (x64)	node-v20.10.0-linux-x64-unknown.tar.gz	node-v20.10.0-linux-x64-unknown-musl.tar.gz	
Linux Binaries (ARM)	node-v20.10.0-linux-armv7-unknown.tar.gz	node-v20.10.0-linux-armv8-unknown.tar.gz	
Source Code	node-v20.10.0.tar.gz		

Below the table, there is a section titled "Additional Platforms" with links to Docker images and other platforms.

At the bottom left, there is a link: <https://nodejs.org/dist/v20.10.0/node-v20.10.0-x64.msi>

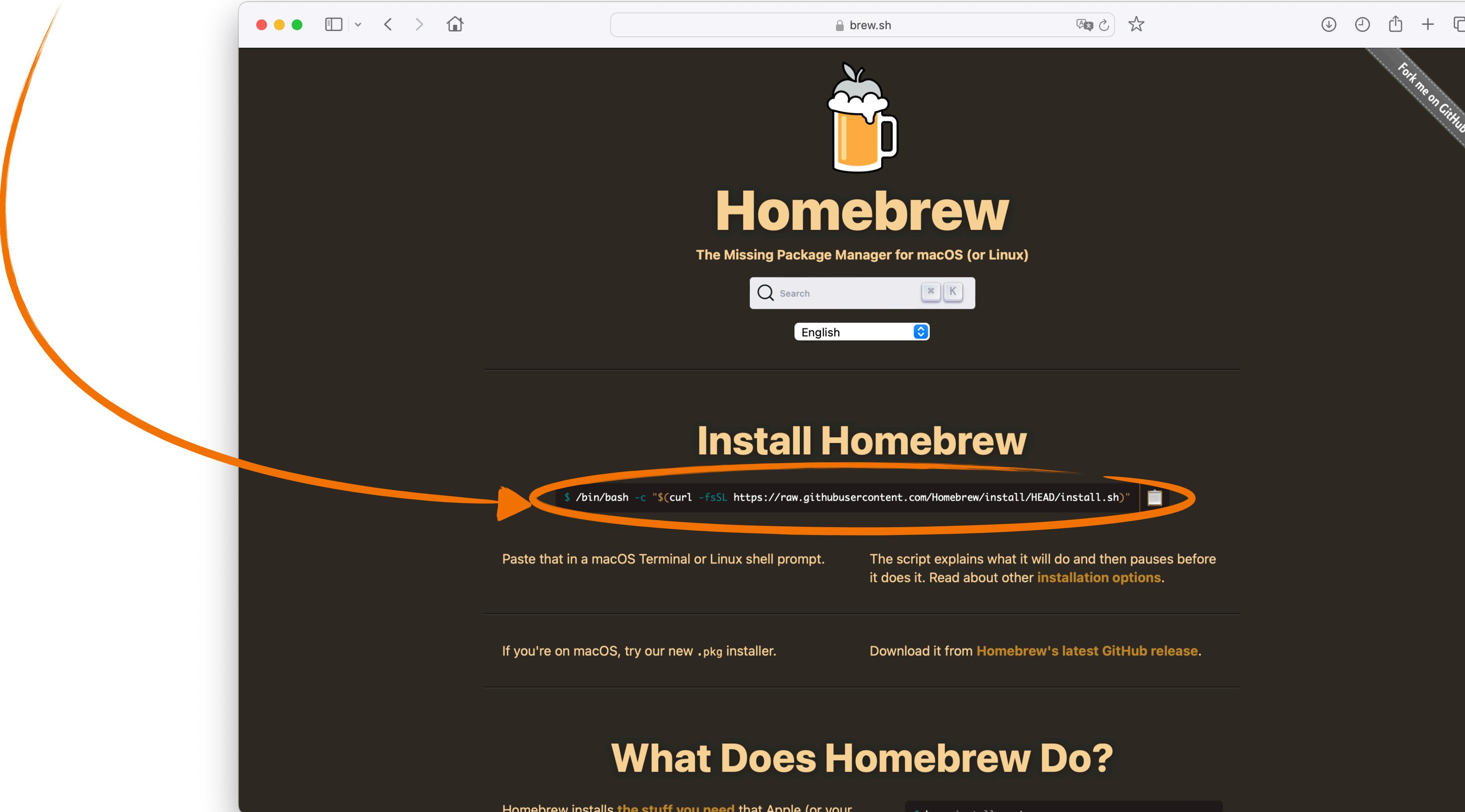
Windows: Install Node.js

So easy! Next, Next, Next, and Finish.



Mac: Homebrew website

At first, it might be a bit challenging, but your life will become more comfortable with Homebrew
Copy that command and paste it to your terminal



Mac: Install Homebrew

At first, it might be a bit challenging, but your life will become more comfortable with Homebrew
Copy that command and paste it to your terminal

After done installation, it will show you the next steps for setting brew path

```
[testuser@Sattawats-MacBook-Air ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"  
==> Checking for `sudo` access (which may request your password)...  
Password:  
[testuser@Sattawats-MacBook-Air ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"  
==> Checking for `sudo` access (which may request your password)...  
| Password:  
| ==> This script will install:  
| /opt/homebrew/bin/brew  
| /opt/homebrew/share/doc/homebrew  
| /opt/homebrew/share/man/man1/brew.1  
| /opt/homebrew/share/zsh/site-functions/_brew  
| /opt/homebrew/etc/bash_completion.d/brew  
| /opt/homebrew  
| Press RETURN/ENTER to continue or any other key to abort:  
[testuser@Sattawats-MacBook-Air ~ % brew untap homebrew/cask  
Warning: /opt/homebrew/bin is not in your PATH.  
Instructions on how to configure your shell for Homebrew  
can be found in the 'Next steps' section below.  
==> Installation successful!  
==> Homebrew has enabled anonymous aggregate formulae and cask analytics.  
Read the analytics documentation (and how to opt-out) here:  
https://docs.brew.sh/Analytics  
No analytics data has been sent yet (nor will any be during this install run).  
==> Homebrew is run entirely by unpaid volunteers. Please consider donating:  
https://github.com/Homebrew/brew#donations  
==> Next steps:  
- Run these two commands in your terminal to add Homebrew to your PATH:  
  (echo; echo 'eval "$(/opt/homebrew/bin/brew shellenv)"') >> /Users/testuser/.zprofile  
  eval "$(/opt/homebrew/bin/brew shellenv)"  
- Run brew help to get started  
- Further documentation:  
  https://docs.brew.sh
```

Mac: Set Homebrew path

Copy that provided 2 lines, and paste to terminal

```
testuser — zsh — 80x24
Warning: /opt/homebrew/bin is not in your PATH.
Instructions on how to configure your shell for Homebrew
can be found in the 'Next steps' section below.
==> Installation successful!

==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run).

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
https://github.com/Homebrew/brew#donations

==> Next steps:
- Run these two commands in your terminal to add Homebrew to your PATH:
  (echo; echo 'eval "$( /opt/homebrew/bin/brew shellenv )"' ) >> /Users/testuser/.zprofile
  eval "$( /opt/homebrew/bin/brew shellenv )"
- Run brew help to get started
- Further documentation:
  https://docs.brew.sh

testuser@Sattawats-MacBook-Air ~ % (echo; echo 'eval "$( /opt/homebrew/bin/brew shellenv )"' ) >> /Users/testuser/.zprofile
```

```
testuser — zsh — 80x24
Instructions on how to configure your shell for Homebrew
can be found in the 'Next steps' section below.
==> Installation successful!

==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run).

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
https://github.com/Homebrew/brew#donations

==> Next steps:
- Run these two commands in your terminal to add Homebrew to your PATH:
  (echo; echo 'eval "$( /opt/homebrew/bin/brew shellenv )"' ) >> /Users/testuser/.zprofile
  eval "$( /opt/homebrew/bin/brew shellenv )"
- Run brew help to get started
- Further documentation:
  https://docs.brew.sh

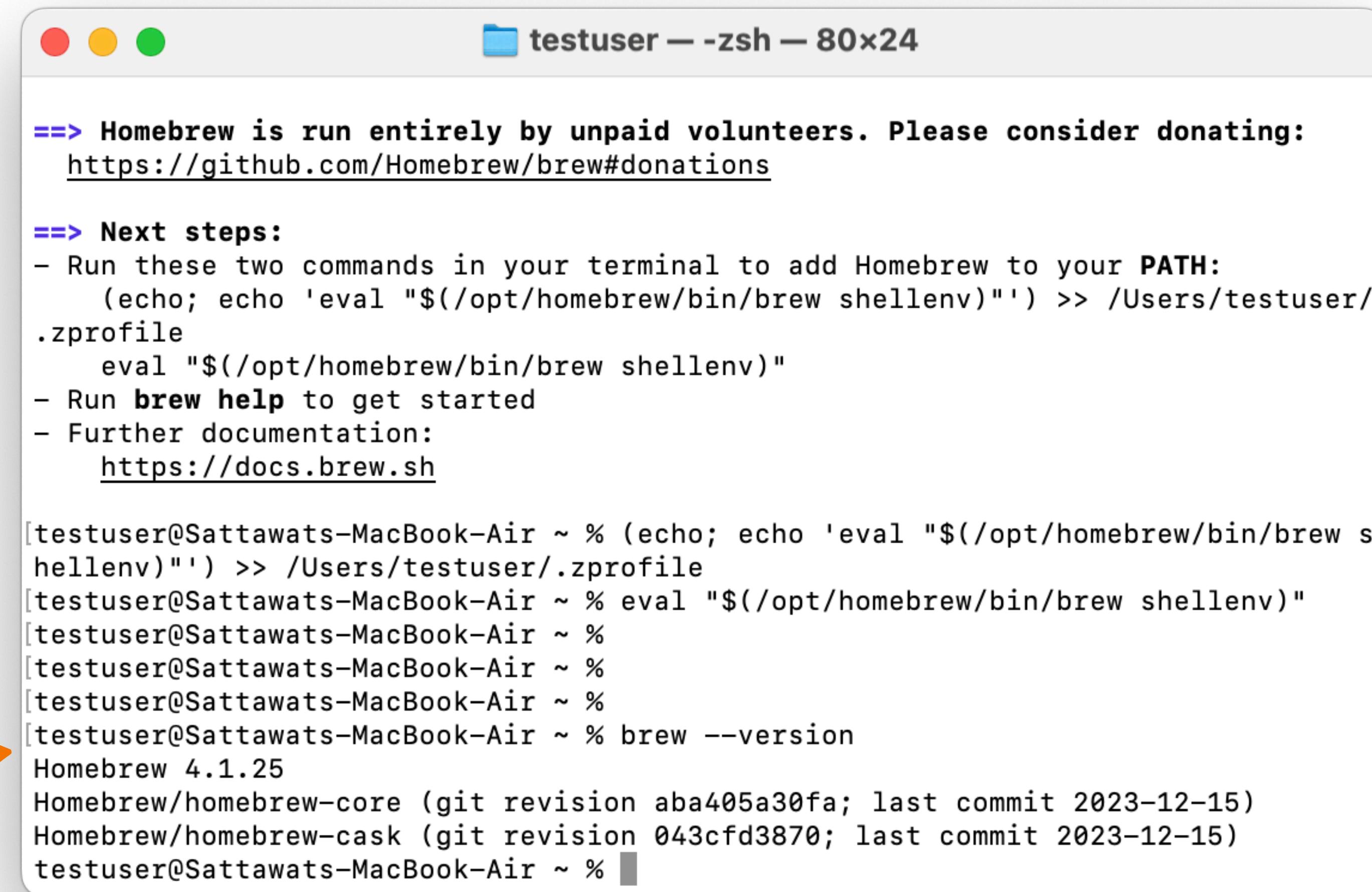
[testuser@Sattawats-MacBook-Air ~ % (echo; echo 'eval "$( /opt/homebrew/bin/brew shellenv )"' ) >> /Users/testuser/.zprofile
testuser@Sattawats-MacBook-Air ~ % eval "$( /opt/homebrew/bin/brew shellenv )"
```

Mac: Check Homebrew version

After set the brew path, check brew version with command "brew --version"

Check brew version:

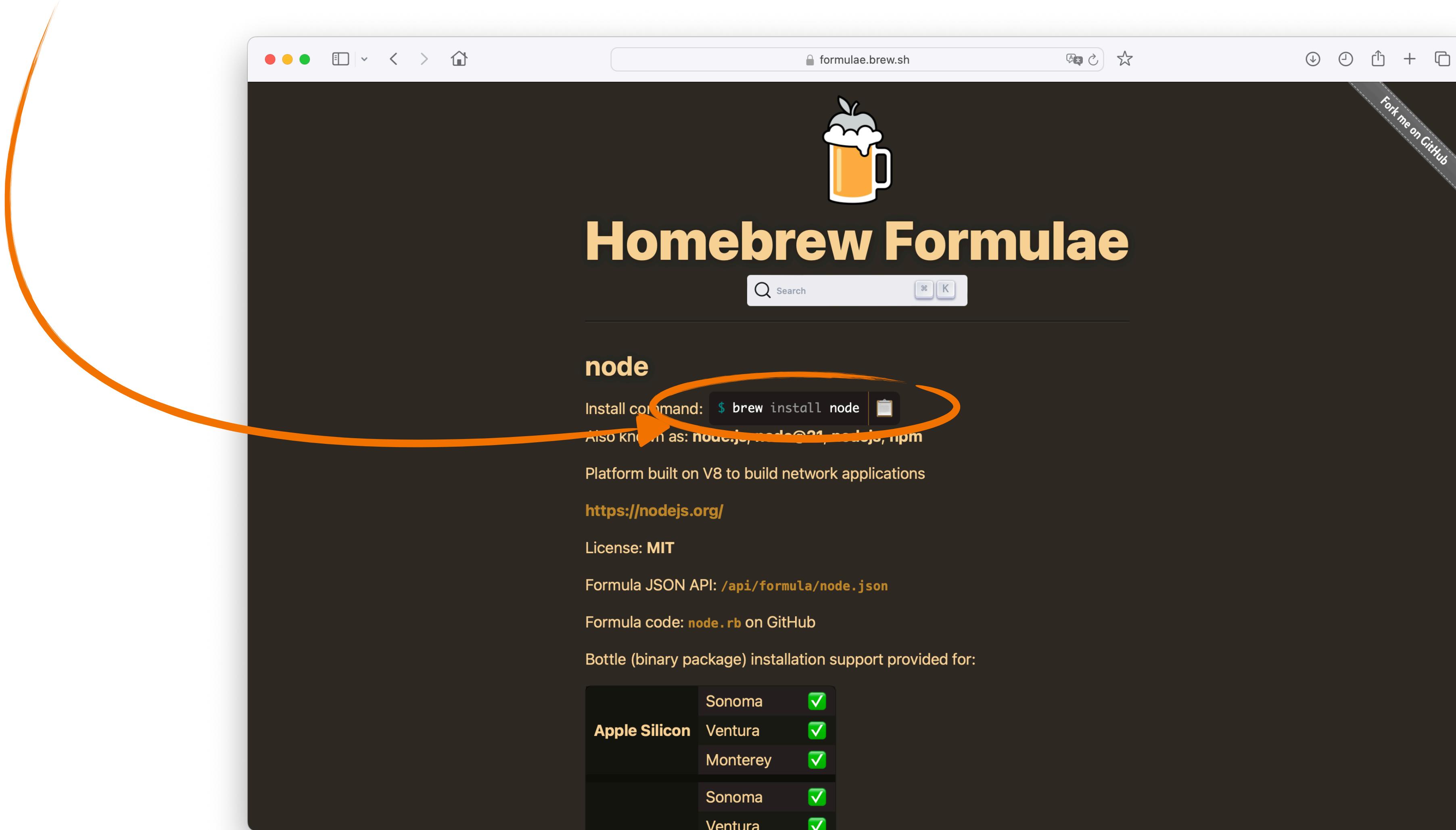
brew --version



```
==> Homebrew is run entirely by unpaid volunteers. Please consider donating:  
https://github.com/Homebrew/brew#donations  
  
==> Next steps:  
- Run these two commands in your terminal to add Homebrew to your PATH:  
  (echo; echo 'eval "$( /opt/homebrew/bin/brew shellenv )"' >> /Users/testuser/.zprofile  
  eval "$( /opt/homebrew/bin/brew shellenv )"  
- Run brew help to get started  
- Further documentation:  
https://docs.brew.sh  
  
[testuser@Sattawats-MacBook-Air ~ % (echo; echo 'eval "$( /opt/homebrew/bin/brew shellenv )"' >> /Users/testuser/.zprofile  
[testuser@Sattawats-MacBook-Air ~ % eval "$( /opt/homebrew/bin/brew shellenv )"  
[testuser@Sattawats-MacBook-Air ~ %  
[testuser@Sattawats-MacBook-Air ~ %  
[testuser@Sattawats-MacBook-Air ~ %  
[testuser@Sattawats-MacBook-Air ~ % brew --version  
Homebrew 4.1.25  
Homebrew/homebrew-core (git revision aba405a30fa; last commit 2023-12-15)  
Homebrew/homebrew-cask (git revision 043cf3d3870; last commit 2023-12-15)  
testuser@Sattawats-MacBook-Air ~ %
```

Mac: Install Node.js

Just run, and wait

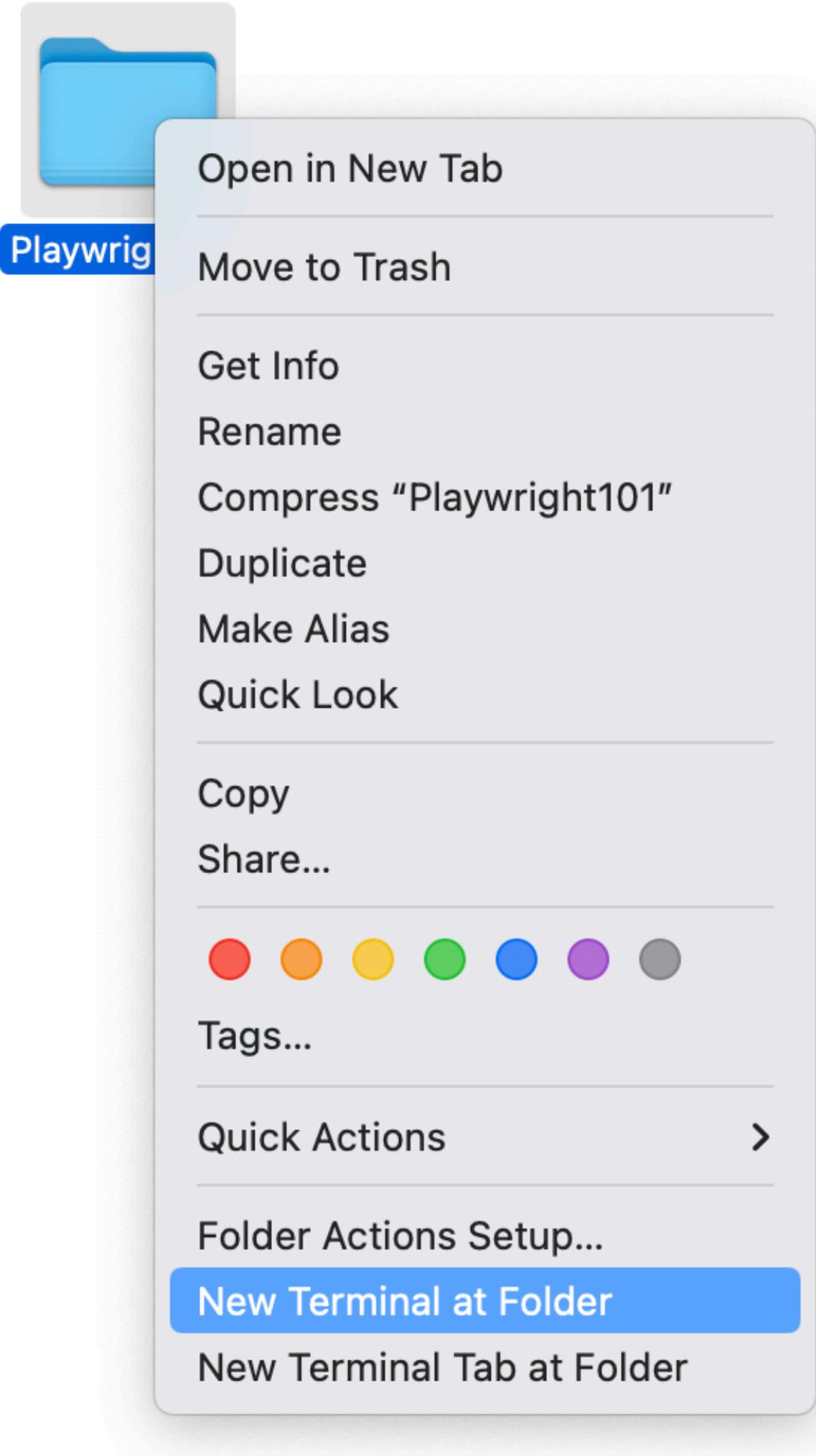


Playwright project structure

and

Example scenarios from init

Initial Playwright project



Run command:

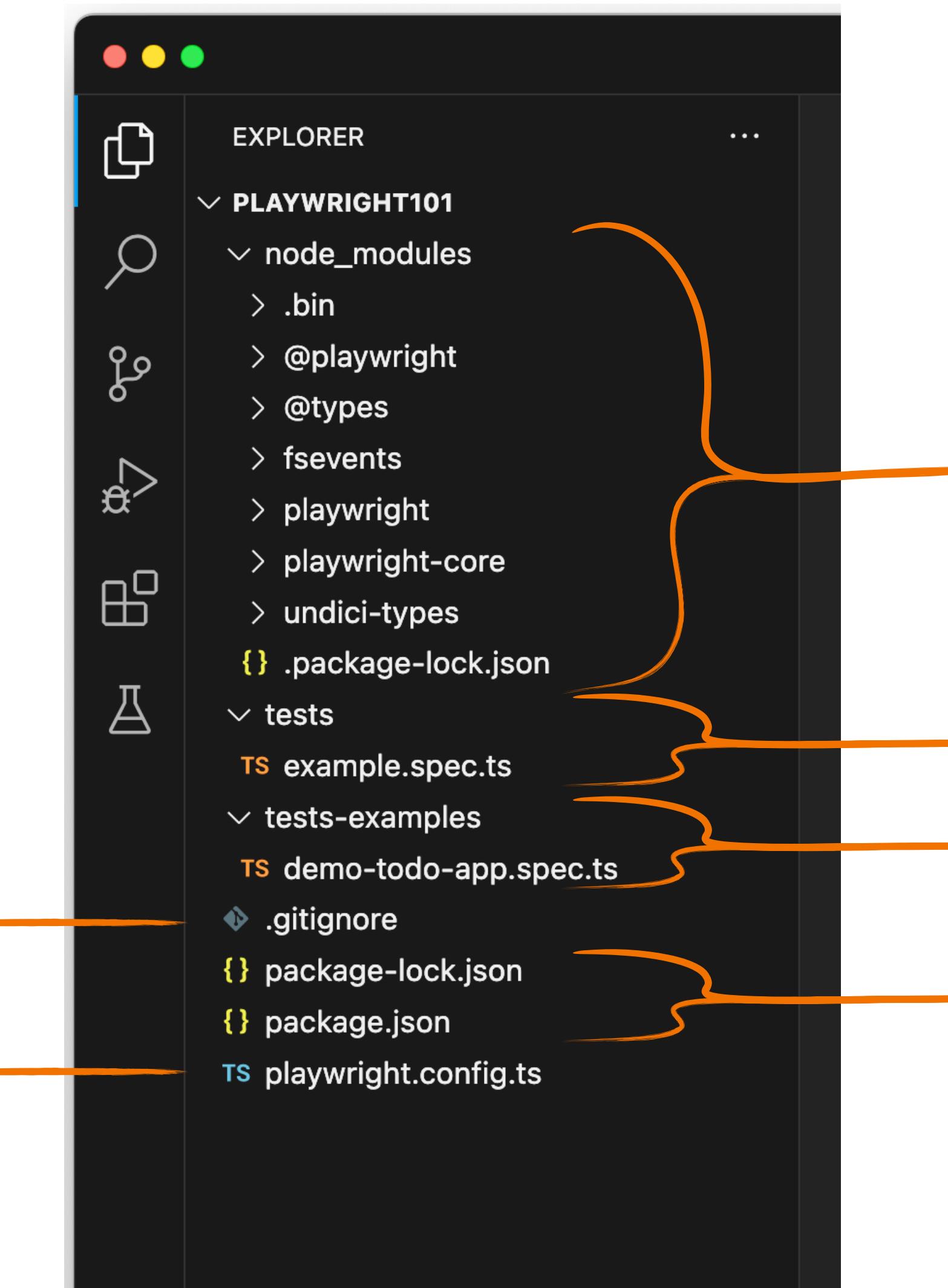
npm init playwright@latest

A screenshot of a terminal window titled "~/Repositories/Playground/Playwright101". The terminal shows the following output:

```
Last login: Wed Nov 15 21:18:09 on ttys014
[jojo@Sattawats-MacBook-Air Playwright101 % npm init playwright@latest
Need to install the following packages:
  create-playwright@1.17.131
Ok to proceed? (y)
Getting started with writing end-to-end tests with Playwright:
Initializing project in '.'
? Do you want to use TypeScript or JavaScript? ...
> TypeScript
  JavaScript
```

Project structure

Specifies intentionally untracked files that Git should ignore
options to configure how your tests are run



Installed Node.js libraries

Our test scenarios will be implemented here

Example test scenarios from Playwright init

All you need to know about what's required in your project's dependency

Example test cases

The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with the following interface elements:

- Explorer View:** On the left, it shows the file structure of a project named "PLAYWRIGHT101". The "tests" folder contains "example.spec.ts", which is currently selected.
- Editor View:** The main area displays the content of "example.spec.ts". The code uses the Playwright Test API to navigate to a page, check its title, and click a link. Two tests are shown, both marked as successful (green checkmarks).

```
1 import { test, expect } from '@playwright/test';
2
3 test('has title', async ({ page }) => {
4   await page.goto('https://playwright.dev/');
5   // Expect a title "to contain" a substring.
6   await expect(page).toHaveTitle(/Playwright/);
7 });
8
9
10 test('get started link', async ({ page }) => {
11   await page.goto('https://playwright.dev/');
12   // Click the get started link.
13   await page.getByRole('link', { name: 'Get started' }).click();
14   // Expects page to have a heading with the name of Installation.
15   await expect(page.getByRole('heading', { name: 'Installation' })).toBeVisible();
16 });
17
18 });
19
```
- Bottom Status Bar:** Shows file statistics (0△0, 0▲0), a "Launch Program (Playwright101)" button, and status information (Ln 19, Col 1, Spaces: 2, UTF-8, LF, {}, TypeScript).
- Bottom Navigation Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TEST RESULTS (which is active), TERMINAL, PORTS, and ROBOT OUTPUT.
- Bottom Right Panel:** Displays the "TEST RESULTS" panel with a summary of the test run: "Test run at 11/21/2023, 9:50:..." (with two green checkmarks for "has title" and "get started link"), and "Test run at 11/21/2023, 9:50:48 PM" (with one orange circle).

Example test cases: has title

The screenshot shows a VS Code interface with the following details:

- EXPLORER:** Shows the project structure under `PLAYWRIGHT101`. A tooltip "First test case named 'has title'" points to the `tests/example.spec.ts` file.
- TEST RESULTS:** Shows a test run from 11/21/2023 at 9:50:48 PM. It includes three test cases:
 - `has title` (passed)
 - `get started link` (passed)
 - `Test run at 11/21/2023, 9:50:48 PM` (info)
- Code Editor:** Displays the `example.spec.ts` file with the following code:1 import { test, expect } from '@playwright/test';
2
3 test('has title', async ({ page }) => {
4 await page.goto('https://playwright.dev/');
5
6 // Expect a title "to contain" a substring.
7 await expect(page).toHaveTitle(/Playwright/);
8 });
9
10 test('get started link', async ({ page }) => {
11 await page.goto('https://playwright.dev/');
12
13 // Click the get started link.
14 await page.getByRole('link', { name: 'Get started' }).click();
15
16 // Expects page to have a heading with the name of Installation.
17 await expect(page.getByRole('heading', { name: 'Installation' })).toBeVisible();
18 });
19

Test scenario structure

test : A function (method) named 'test' requires two arguments: 'title' and 'testFunction'

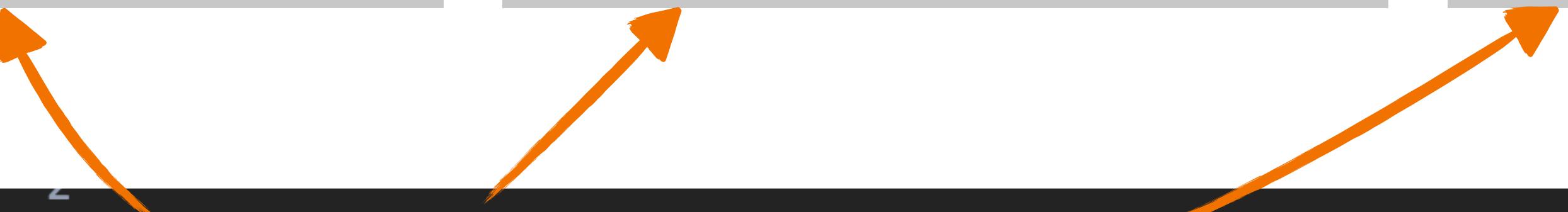
```
3 test(          ,
4
5
6
7
8 );
```

title : A test scenario name must be unique within the same test suite

'has · title'

testFunction : A function that contains our test steps, marked as an **asynchronous (async)** function that will wait for any step marked with '**await**'

```
3           .async .( ) => {
4
5
6
7
8     }
```

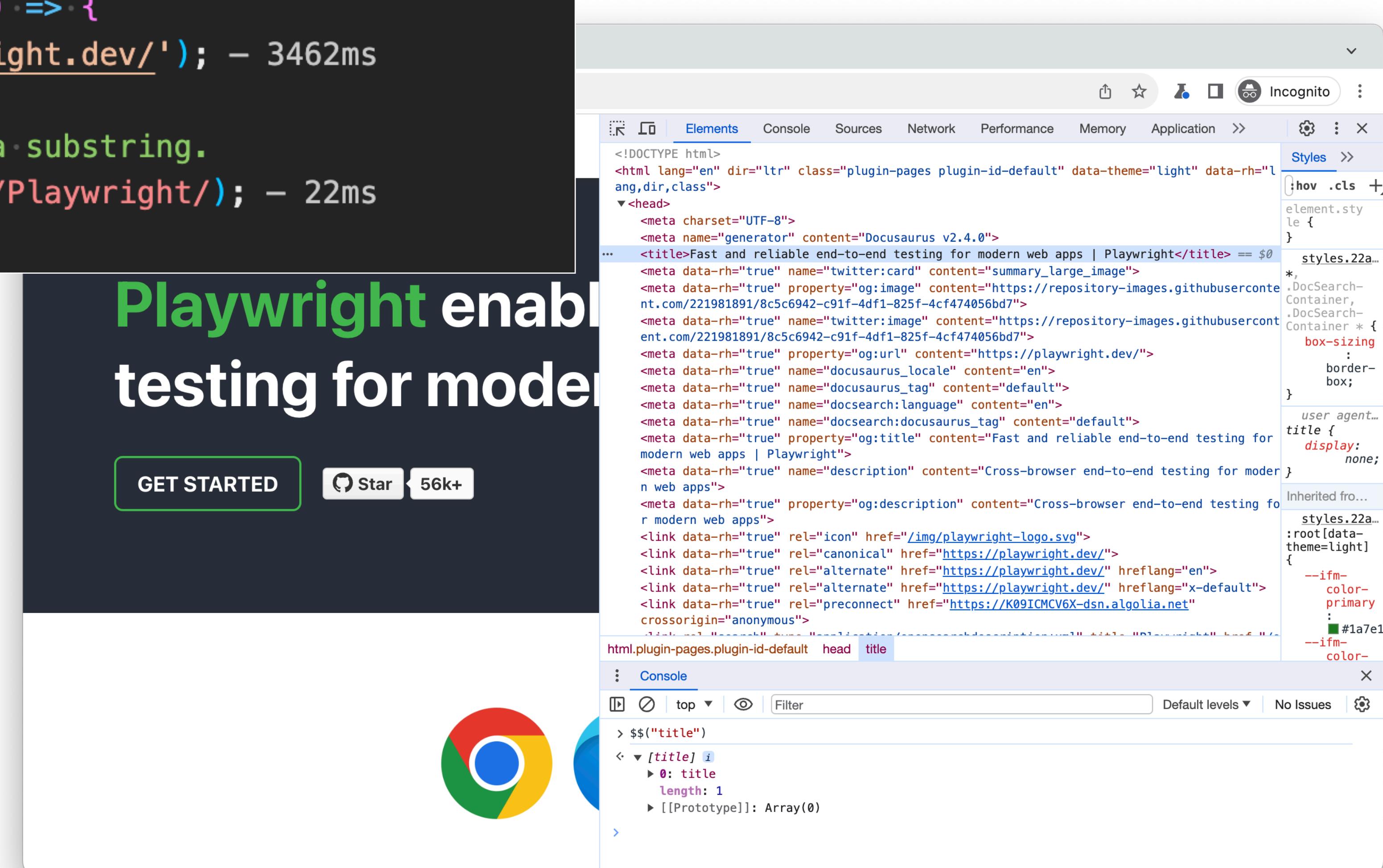
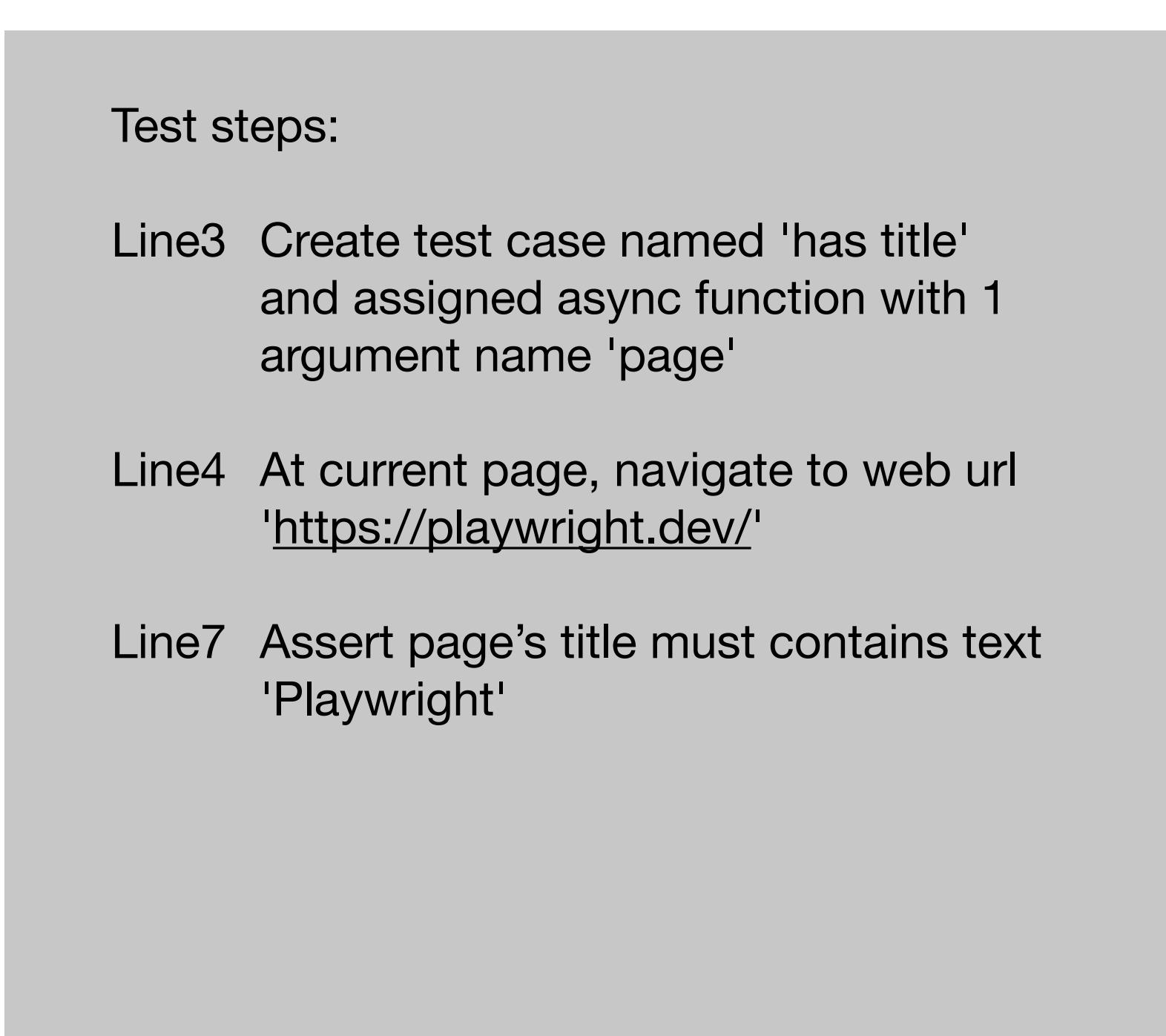


```
2
3 test('has title', async ({ page }) => {
4   await page.goto('https://playwright.dev/'); - 3462ms
5
6   // Expect a title "to contain" a substring.
7   await expect(page).toHaveTitle(/Playwright/); - 22ms
8 });
```

Example test cases: has title

```
2
3  test('has title', async ({ page }) => {
4    await page.goto('https://playwright.dev/'); - 3462ms
5
6    // Expect a title "to contain" a substring.
7    await expect(page).toHaveTitle(/Playwright/); - 22ms
8  });

```



Example test cases: get started link

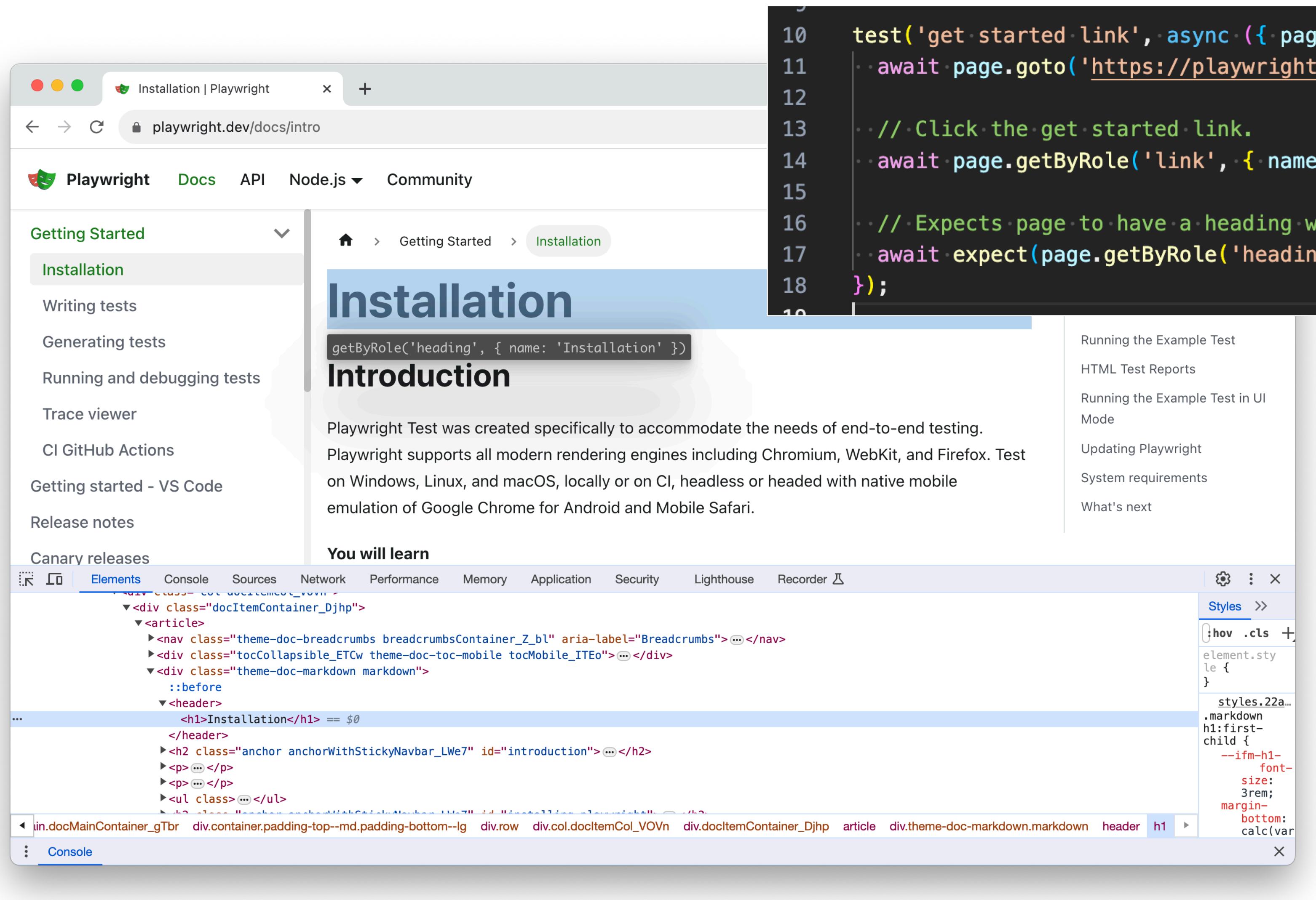


The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure for "PLAYWRIGHT101". A tooltip is displayed over the "tests" folder, containing the text: "Second test case named 'get started link'".
- Code Editor:** The file "example.spec.ts" is open, showing TypeScript code for playwright tests. One specific test case is highlighted:

```
test('get-started-link', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  // Click the get-started-link.
  await page.getByRole('link', { name: 'Get started' }).click();
  // Expects page to have a heading with the name of Installation.
  await expect(page.getByRole('heading', { name: 'Installation' })).toBeVisible();
});
```
- Test Results:** A tooltip in the bottom right corner lists the test results from the most recent run:
 - Test run at 11/21/2023, 9:50:... (Success)
 - has title (Success)
 - get started link (Success)
 - Test run at 11/21/2023, 9:50:48 PM (Success)
- Bottom Status Bar:** Shows file statistics (Ln 19, Col 1), code analysis (0△0), and other development tools.

Example test cases: get started link



The screenshot shows a browser window with the URL `playwright.dev/docs/intro`. The page title is "Installation". The left sidebar has "Getting Started" expanded, with "Installation" selected. The main content area has a heading "Introduction" and some text about Playwright's capabilities. A developer tools panel is open at the bottom, showing the "Elements" tab with the DOM tree and a "Styles" panel on the right. The test code in the console is:

```
10  test('get-started-link', async ({page}) => {
11    await page.goto('https://playwright.dev/'); - 3073ms
12
13    // Click the get-started link.
14    await page.getByRole('link', {name: 'Get started'}).click(); - 66ms
15
16    // Expects page to have a heading with the name of Installation.
17    await expect(page.getByRole('heading', {name: 'Installation'})).toBeVisible(); - 371ms
18});
```

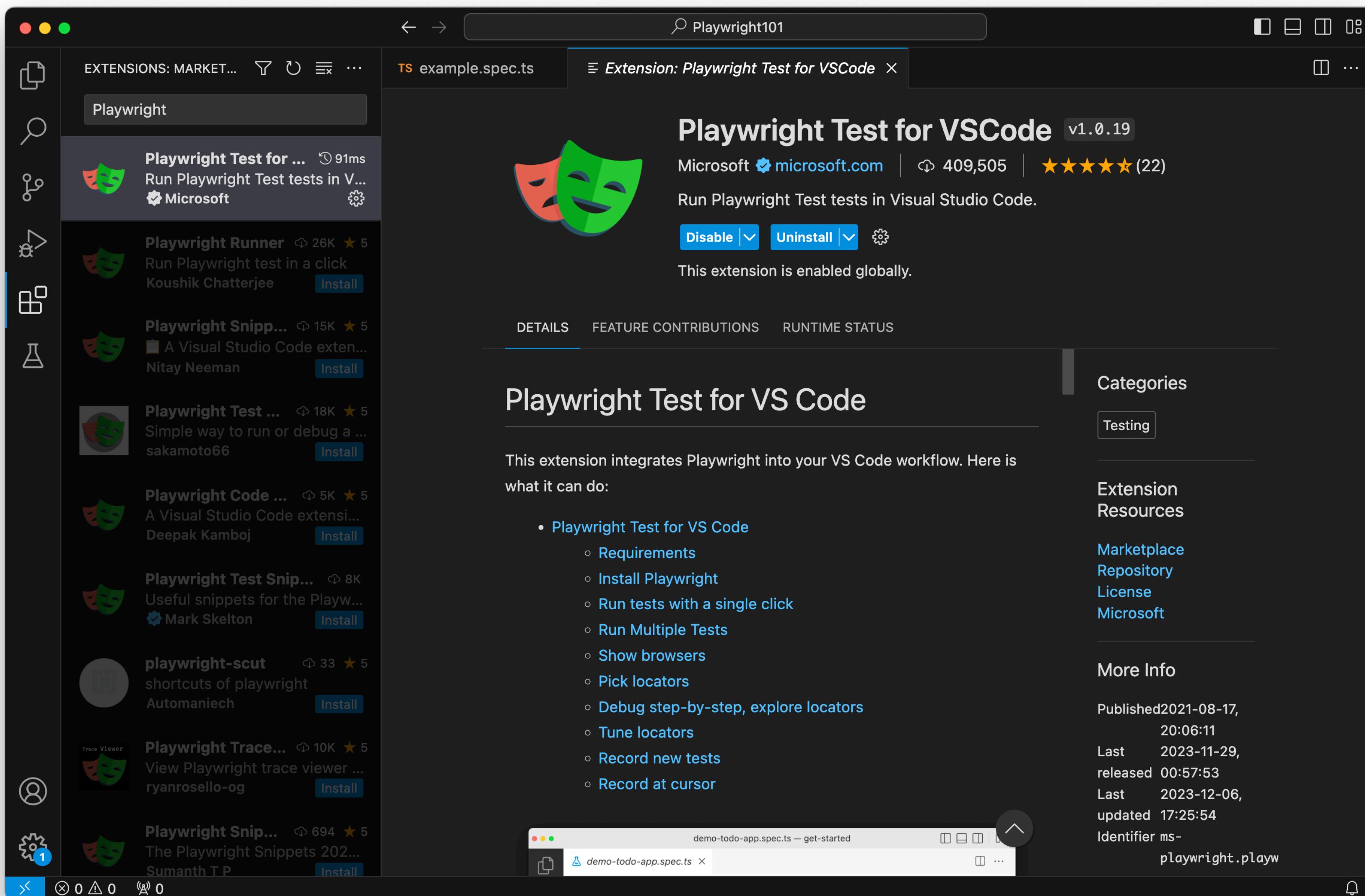
Test steps:

- Line10 Create test case named 'get started link' and assigned `async` function with 1 argument name 'page'
- Line14 At current page, navigate to web url `'https://playwright.dev/'`
- Line17 In expect's bracket, current page find and get page element with role 'heading' (e.g. h1, h2, ...) that contains text 'Installation'. Then it should be visible

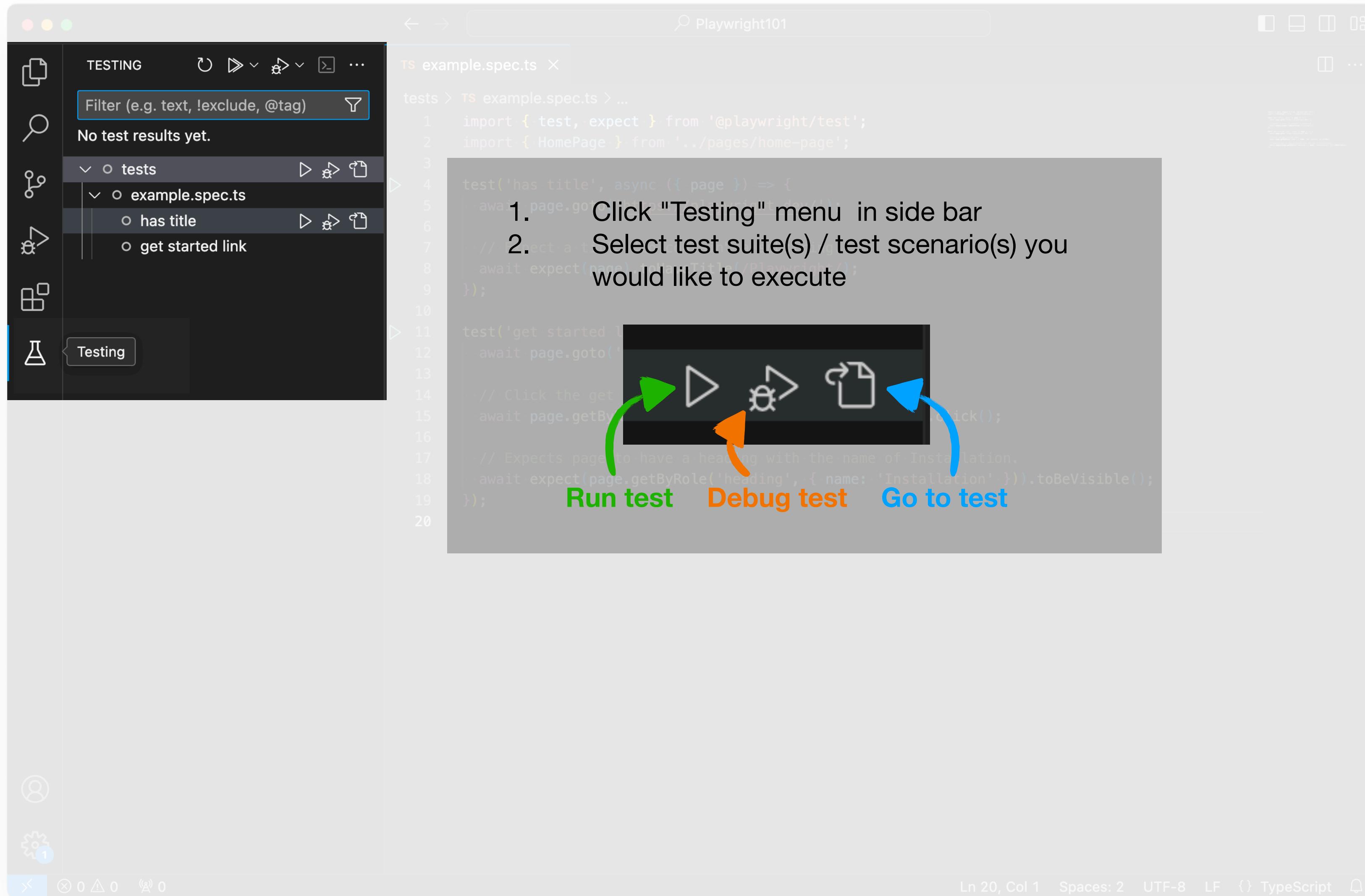
Test execution and Debug mode

- VSCode extension
- Execute / debug test(s) from VSCode
- Execute / debug test(s) from Command

VSCode extension



Execute / debug tests from VSCode



Execute tests from command



A screenshot of a macOS terminal window. The title bar shows the path `~/Repositories/Playground/Playwright101`. The terminal output is as follows:

```
[jojoe@Sattawats-MacBook-Air Playwright101 % npx playwright test
Running 6 tests using 4 workers
  6 passed (10.1s)

To open last HTML report run:
  npx playwright show-report
jojoe@Sattawats-MacBook-Air Playwright101 %
```

Execute all test suite and scenarios:

```
npx playwright test
```

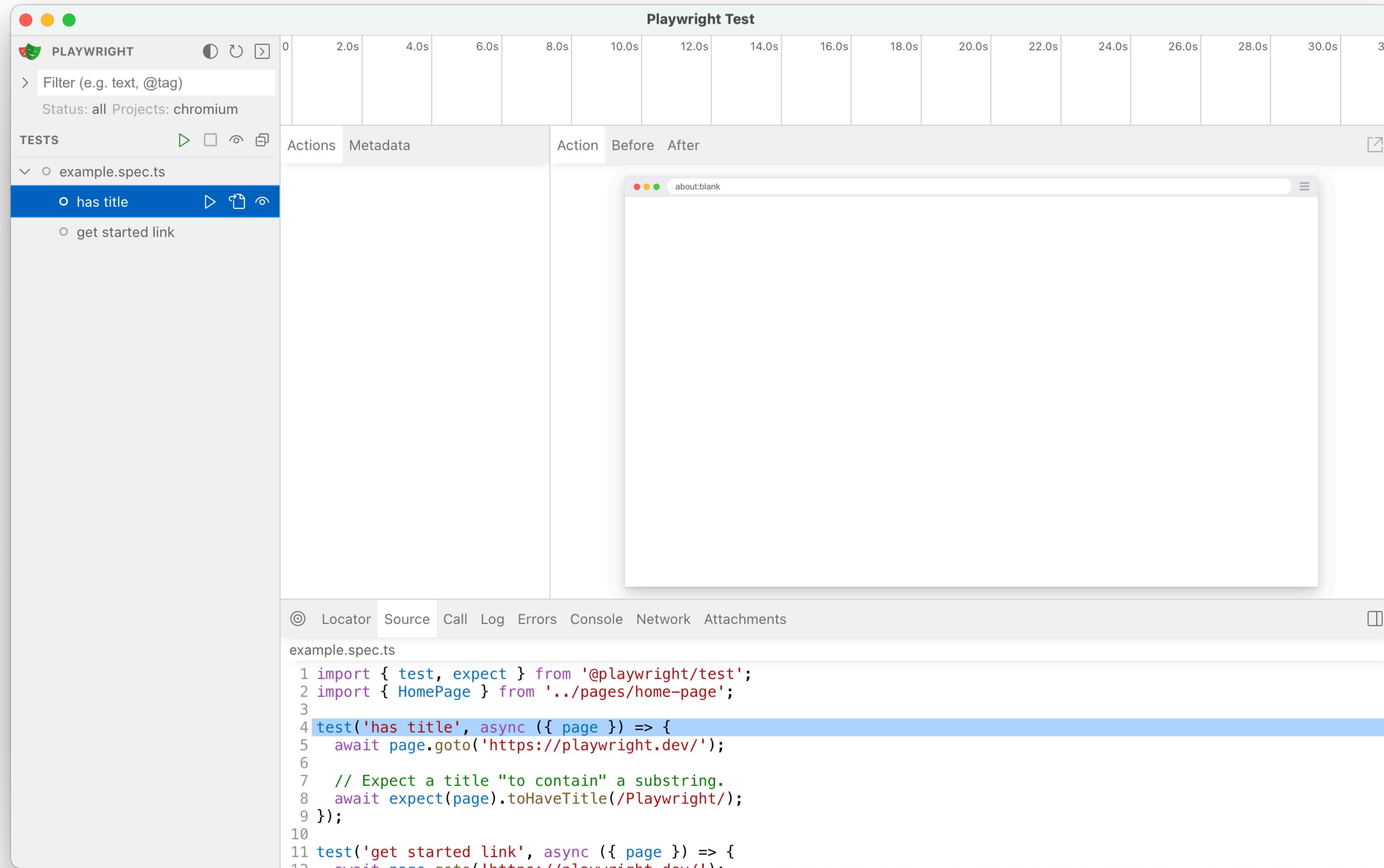
Execute single test suite:

```
npx playwright test tests/example.spec.ts
```

Execute single test scenarios:

```
npx playwright test -g "has title"
```

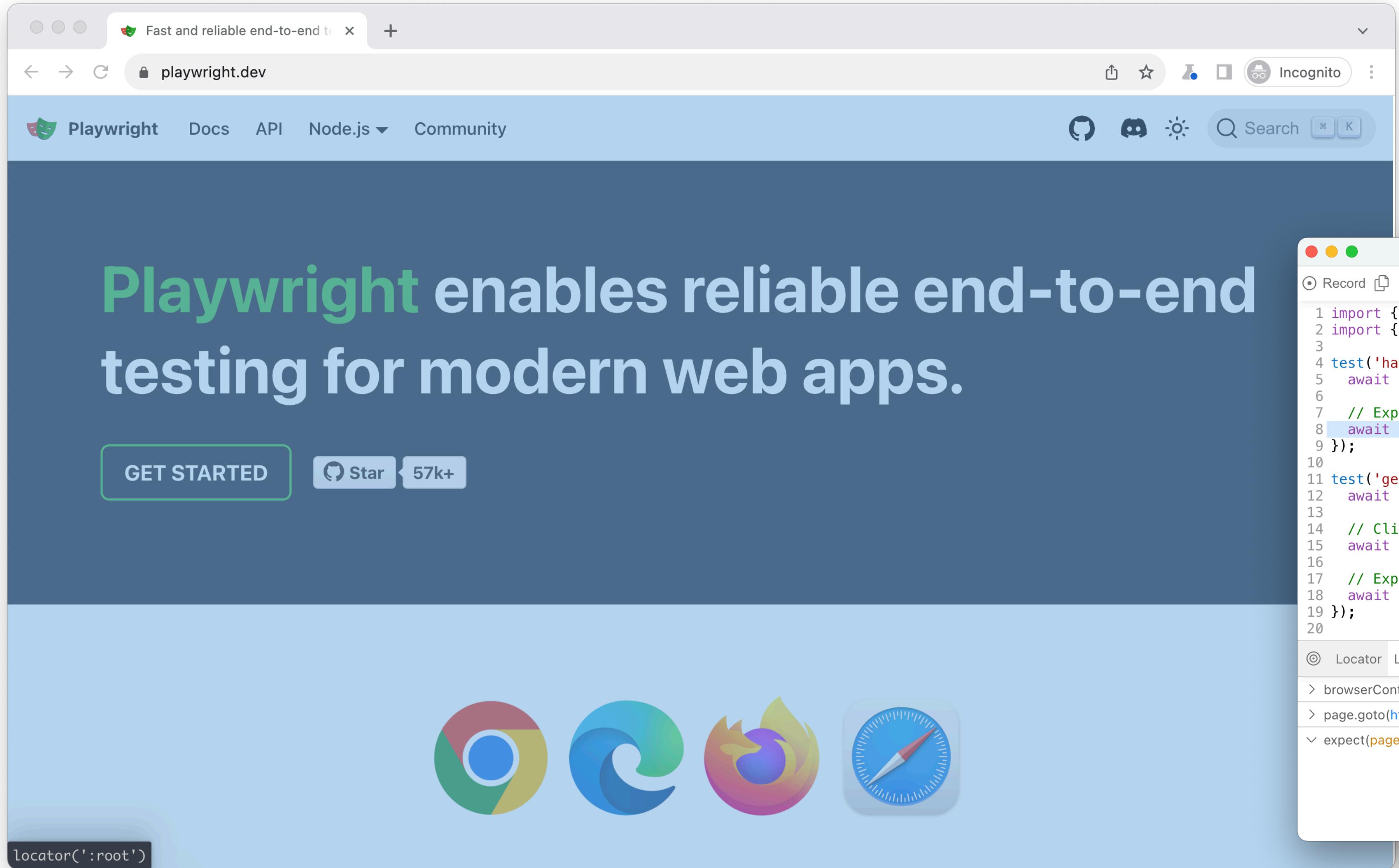
Execute tests in UI mode from command



Open inspector UI mode:

```
npx playwright test --ui
```

Execute tests in debug mode from command



Open inspector debug mode:

```
npx playwright test --debug
```

The screenshot shows the Playwright Inspector tool window. The title bar says 'Playwright Inspector'. The main area is a code editor displaying a TypeScript test file with syntax highlighting. The code is as follows:

```
1 import { test, expect } from '@playwright/test';
2 import { HomePage } from '../pages/home-page';
3
4 test('has title', async ({ page }) => {
5   await page.goto('https://playwright.dev/');
6
7   // Expect a title "to contain" a substring.
8   await expect(page).toHaveTitle(/Playwright/);
9 });
10
11 test('get started link', async ({ page }) => {
12   await page.goto('https://playwright.dev/');
13
14   // Click the get started link.
15   await page.getByRole('link', { name: 'Get started' }).click();
16
17   // Expects page to have a heading with the name of Installation.
18   await expect(page.getByRole('heading', { name: 'Installation' }));
19 });
20
```

Below the code editor is a log panel with the following entries:

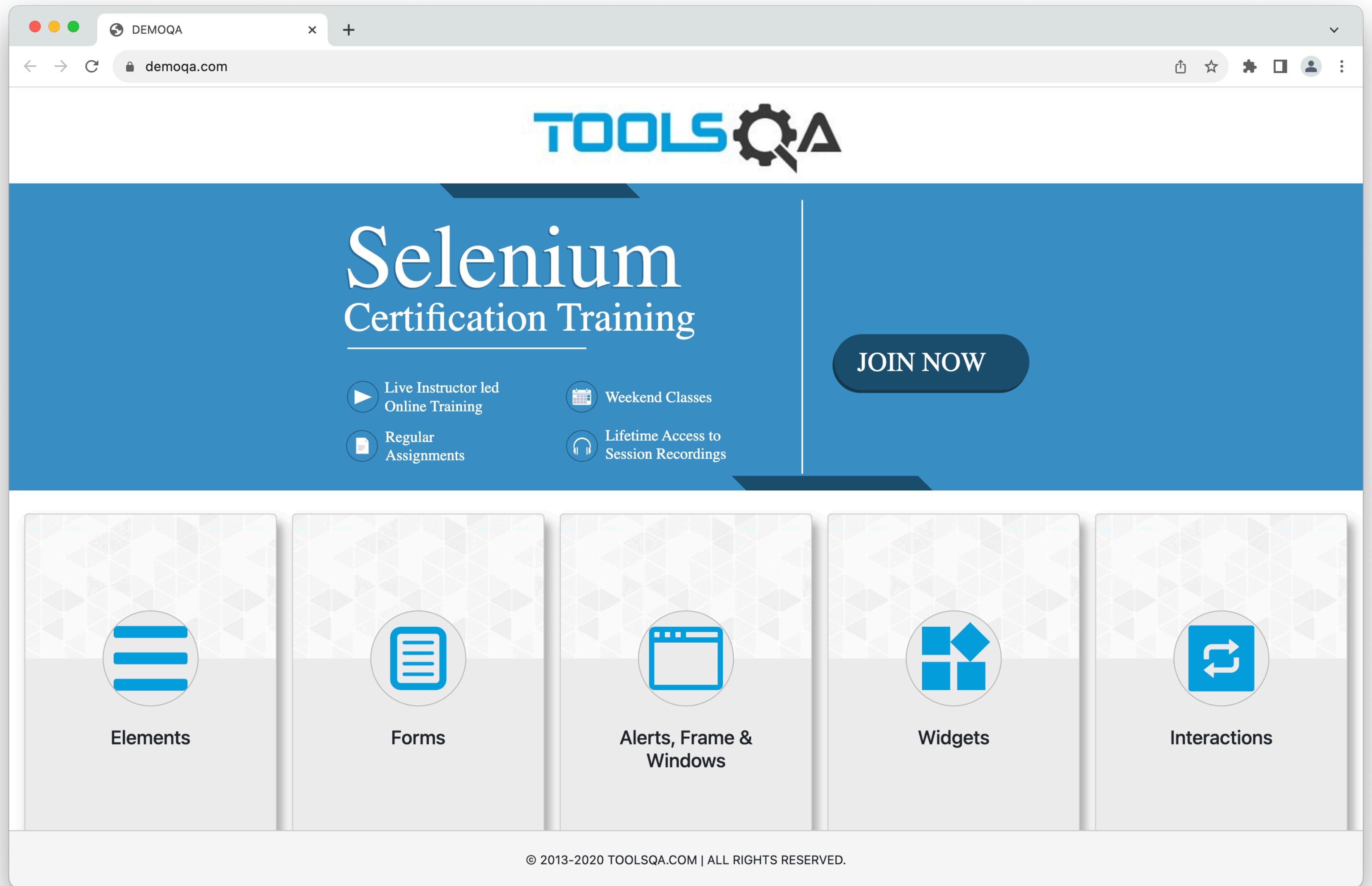
- > browserContext.newPage ✓ — 261ms
- > page.goto('https://playwright.dev/') ✓ — 2.4s
- ✓ expect(page.locator(':root')).toHaveTitle() II

Locators

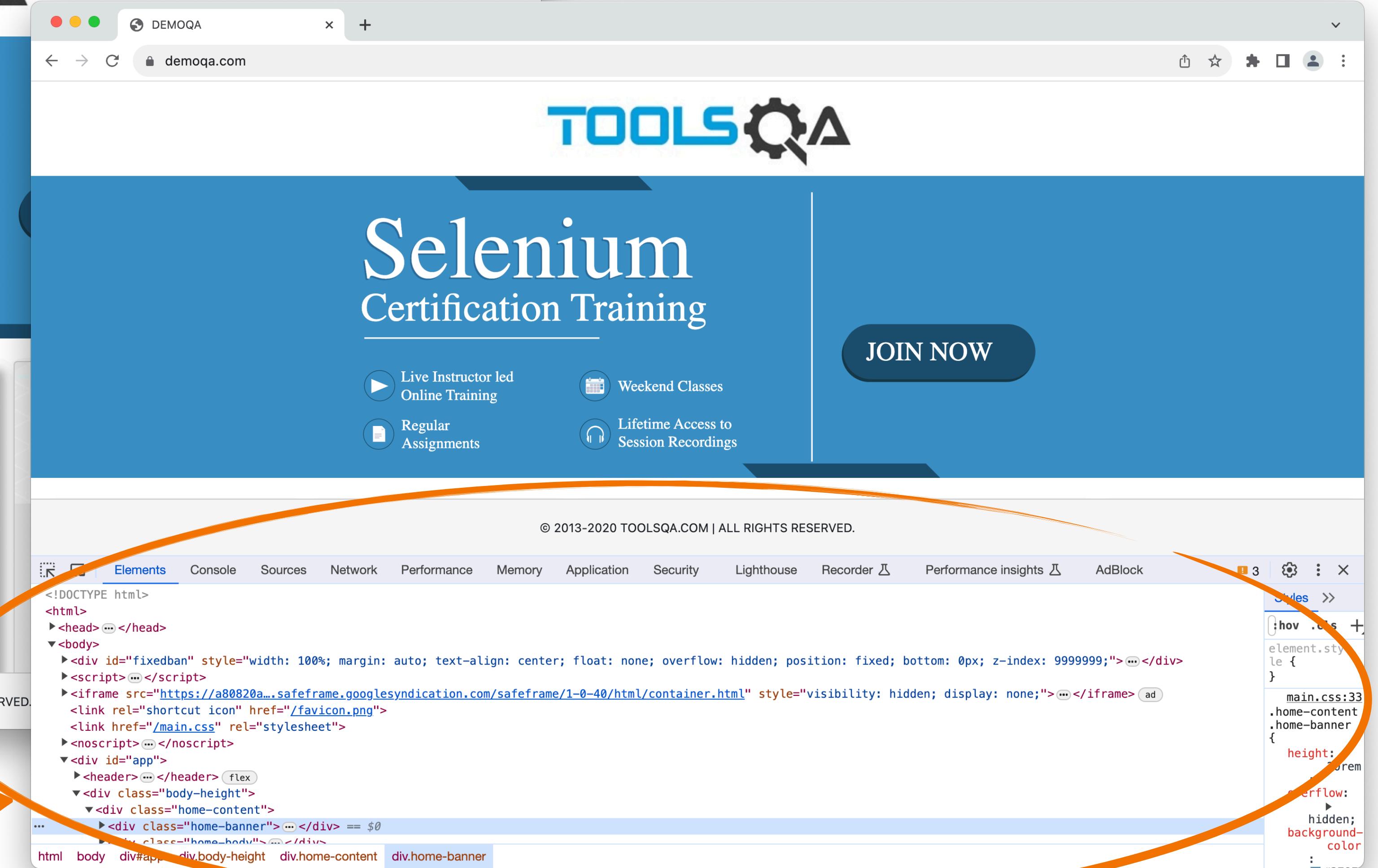
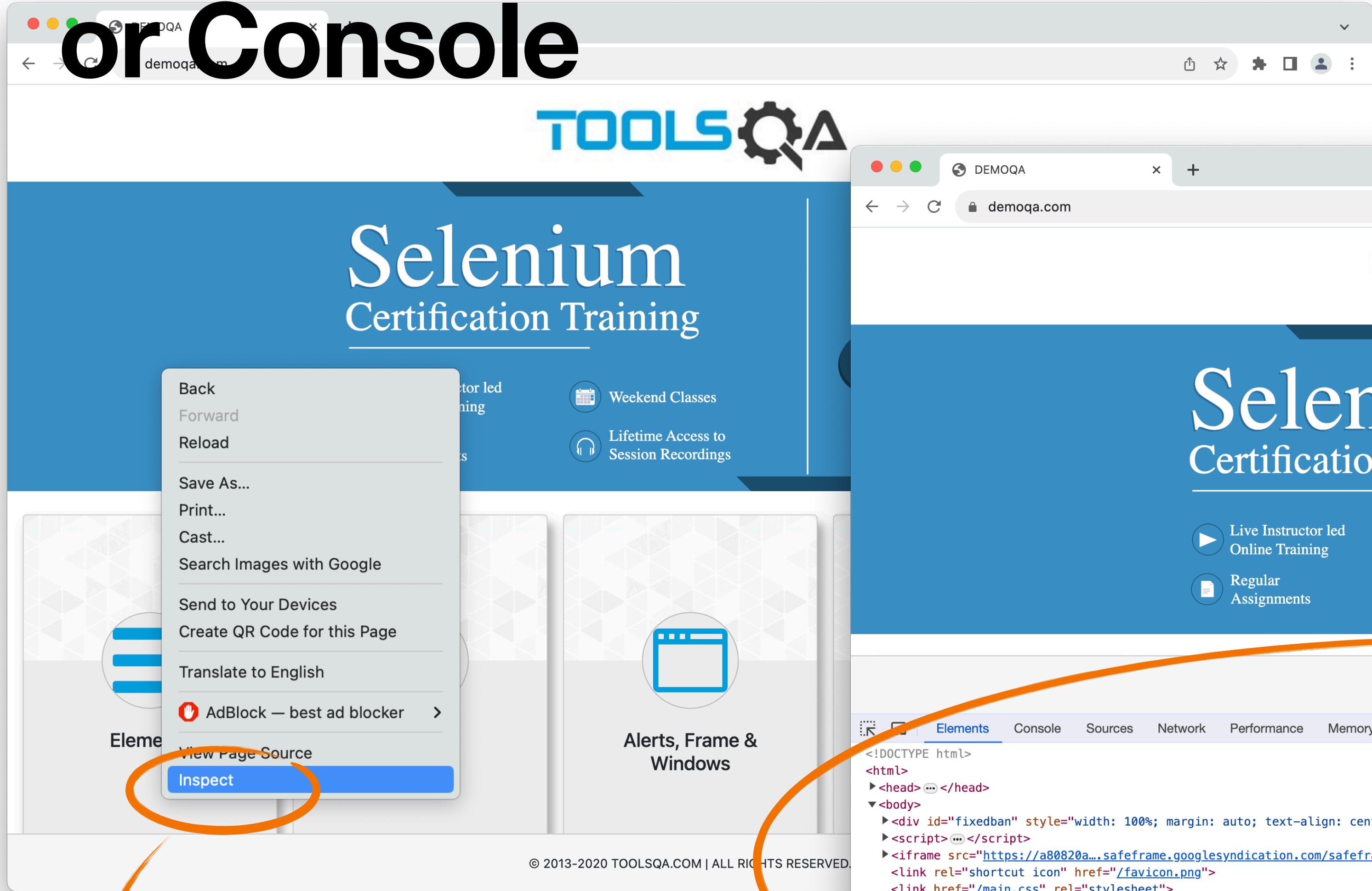
- Chrome inspect and console
- HTML Tags
 - ID
 - CSS
 - XPath
 - data-testid
 - Chain locator

Tools QA : Our playground in this session :)

Open Chrome browser
Go to this url: demoqa.com



Chrome Inspect, Elements, or Console



Open inspect and console steps:

1. Right click on page and select "Inspect" or press "F12"
2. In the inspect tab, select "Elements" or "Console"

Identify the unique locator

The screenshot shows the ToolsQA Selenium Certification Training website. The main heading is "Selenium Certification Training". Below it are four icons: "Live Instructor led Online Training", "Weekend Classes", "Regular Assignments", and "Lifetime Access to Session Recordings". A large blue button labeled "JOIN NOW" is visible. In the bottom left corner, there's a sidebar with "Elements" selected, displaying CSS properties for an "h5" element: Color (#212529), Font (20px -apple-system, "system-ui", "Seg..."), Margin (0px 0px 8px). It also shows "ACCESSIBILITY" settings: Contrast (Aa 13.29), Name, Role, and Keyboard-focusable. The main content area features a "Forms" section with a document icon.

Steps to identify locator of some elements:

1. Open Chrome inspect, and go to Elements tab
2. Use this tool, it is the easiest way

Select an element in the page to inspect it - ⌘ ⌘ C

3. Point the arrow at the element you want, then click

4. In the Elements tab, it will bring you to the line of HTML of the element

However, how do you know that your selected element is unique?

Look closer

The screenshot shows a web browser window with the URL demoqa.com. The page content is a promotional banner for 'Selenium Certification Training'. On the left, there's a large blue box with the 'TOOLSQA' logo at the top. Below it, the text 'Selenium Certification Training' is displayed. To the right of this text is a 'JOIN NOW' button. Further down, there are four categories represented by cards: 'Elements' (highlighted with a purple dashed border), 'Forms', 'Tables', and 'JavaScript'. Each category has a small icon and a brief description. At the bottom of the page, there's a copyright notice: '© 2013-2020 TOOLSQA.COM | ALL RIGHTS RESERVED.' The browser's developer tools are open, with the 'Elements' tab selected. A tooltip from the developer tools points to the 'Elements' card, providing its dimensions: 'div.card.mt-4.top-card 356.8 x 400'. The DOM tree in the developer tools shows the structure of the page, including various divs, spans, and other HTML elements.

What if we want to click this locator, which one is unique?

- A. css = div.avatar.mx-auto.white
xpath = //div[@class='avatar mx-auto white']
- B. css = div.card-body
xpath = //div[@class='card-body']
- C. css = h5
xpath = //h5
- D. CSS can't point to the unique element 'Elements'
xpath = //h5[text()='Elements']

This example shows you two kinds of locators:
CSS and XPath locators

Let's try

Chrome console to identity selectors

The screenshot shows a browser window with the URL `demoqa.com`. The main content is the **ToolsQA Selenium Certification Training** page, featuring sections for **Elements**, **Forms**, **Alerts, Frame & Windows**, and **Widgets**. A specific element in the Widgets section is highlighted with a purple dashed border and labeled `div.avatar.mx-auto.white 100x100`.

The bottom portion of the screenshot shows the **Console** tab of the Chrome DevTools. The console output shows the result of running the CSS selector `$$("div.avatar.mx-auto.white")`, which returns a NodeList with 6 items. An orange circle highlights the first few lines of the console output, and an orange arrow points from this circle to the text "Seem it's not unique selector" located at the bottom left.

```
> $$("div.avatar.mx-auto.white")
< (6) [div.avatar.mx-auto.white, div.avatar.mx-auto.white, div.avatar.mx-auto.white, div.avatar.mx-auto.whi
  div.avatar.mx-auto.white, div.avatar.mx-auto.white] i
    ▷ 0: div.avatar.mx-auto.white
    ▷ 1: div.avatar.mx-auto.white
    ▷ 2: div.avatar.mx-auto.white
    ▷ 3: div.avatar.mx-auto.white
    ▷ 4: div.avatar.mx-auto.white
    ▷ 5: div.avatar.mx-auto.white
    length: 6
    ▷ [[Prototype]]: Array(0)
```

Locator assignment:

In case you would like to fine the CSS selector
use this sign:

```
$$("div.avatar.mx-auto.white")
```

**In case you would like to fine the XPath locator
use this sign:**

```
$x("//div[@class='avatar mx-auto white']")
```

Seem it's not unique selector

This page is the best practice for selector finding

The screenshot shows a web application interface with the following elements:

- Form Fields:**
 - Full Name: Input field containing "Jojoe".
 - Email: Input field containing "name@example.com".
 - Current Address: Text area containing "Current Address".
 - Permanent Address: Text area containing "Permanent Address".
 - Output: A success message box displaying "Name: Jojoe".
- Submit Button:** A blue "Submit" button located at the bottom right of the form.
- Page Footer:** Includes the ZeroStep logo, the text "Build Playwright", and copyright information: "© 2013-2020 TOOLSQA.COM | ALL RIGHTS RESERVED."

The developer tools Elements tab displays the following DOM structure:

```
<div id="aygound-body" class="aygound-body">
  <div class="row">
    <div class="col-12 mt-4 col-md-6">
      <div class="text-field-container">
        <form id="userForm" class="aygound-form">
          <div id="userName-wrapper" class="mt-2 row">
            <div class="col-md-3 col-sm-12">
              <label class="form-label" id="userName-label">Full Name</label>
            </div>
            <div class="col-md-9 col-sm-12">
              <input autocomplete="off" placeholder="Full Name" type="text" id="userName" class="mr-sm-2 form-control">
            </div>
          </div>
          <div id="userEmail-wrapper" class="mt-2 row">
            <div class="col-md-12 col-sm-12">
              <input type="text" id="userEmail" class="form-control">
            </div>
          </div>
          <div id="currentAddress-wrapper" class="mt-2 row">
            <div class="col-md-12 col-sm-12">
              <input type="text" id="currentAddress" class="form-control">
            </div>
          </div>
          <div id="permanentAddress-wrapper" class="mt-2 row">
            <div class="col-md-12 col-sm-12">
              <input type="text" id="permanentAddress" class="form-control">
            </div>
          </div>
          <div class="mt-2 justify-content-end row">
            <div class="text-right col-md-2 col-sm-12">
              <button id="submit" type="button" class="btn btn-primary">Submit</button>
            </div>
          </div>
          <div id="output" class="mt-4 row">
            <div class="border col-md-12 col-sm-12">
              <p id="name" class="mb-1">
                "Name:<br/>"Jojoe<br/></p>
              </div>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
```

id = userForm

css = form#userForm

id = userName

css = input#userName

xpath = //input[@id='userName']

xpath = //input[
@placeholder='Full Name']

data-testid locator

data-testid is an attribute commonly used in web development to facilitate testing of user interfaces. It is not a native HTML attribute, but rather a **custom attribute that developers add to their HTML elements to make it easier to select and interact with those elements in automated tests.**

When writing tests, developers often need a way to uniquely identify elements on a page, especially when using testing libraries or frameworks like Jest, React Testing Library, or others. Adding data-testid attributes to elements allows developers to target specific elements in their tests, regardless of the underlying structure or styling of the page.

```
html
```

 Copy code

```
<button data-testid="submit-button">Submit</button>
```

Chain locator

The screenshot shows a browser window with a form titled "Full Name". The form contains fields for Name (value: Jojoe), Email (value: name@example.com), Current Address, and Permanent Address. Below the form, several locators are listed:

- id = userForm**
- css = form#userForm**
- id = userName**
- css = input#userName**
- xpath = //input[@id='userName']**
- xpath = //input[@placeholder='Full Name']**

The developer tools' Elements tab shows the DOM structure of the form. Two arrows point from the highlighted locators to specific elements in the DOM tree:

- An arrow points from **id = userForm** to the `<form id="userForm" class="mt-2 row">` element.
- An arrow points from **id = userName** to the `<input autocomplete="off" placeholder="Full Name" type="text" id="userName" class="mr-sm-2 form-control">` element.

The developer tools also show the full DOM structure of the page, including other wrapper divs and a submission button.

It is not quite hard to connect more than one selector together

A chain locator is an approach to create a new unique locator by combining different kinds of selectors.

For example, if you cannot find a unique selector:

css = userForm >> xpath = //input[@id='userName']

The sign **>>** is a chain connector that links between:

- 1st locator: css = userForm
- 2nd locator: xpath = //input[@id='userName']

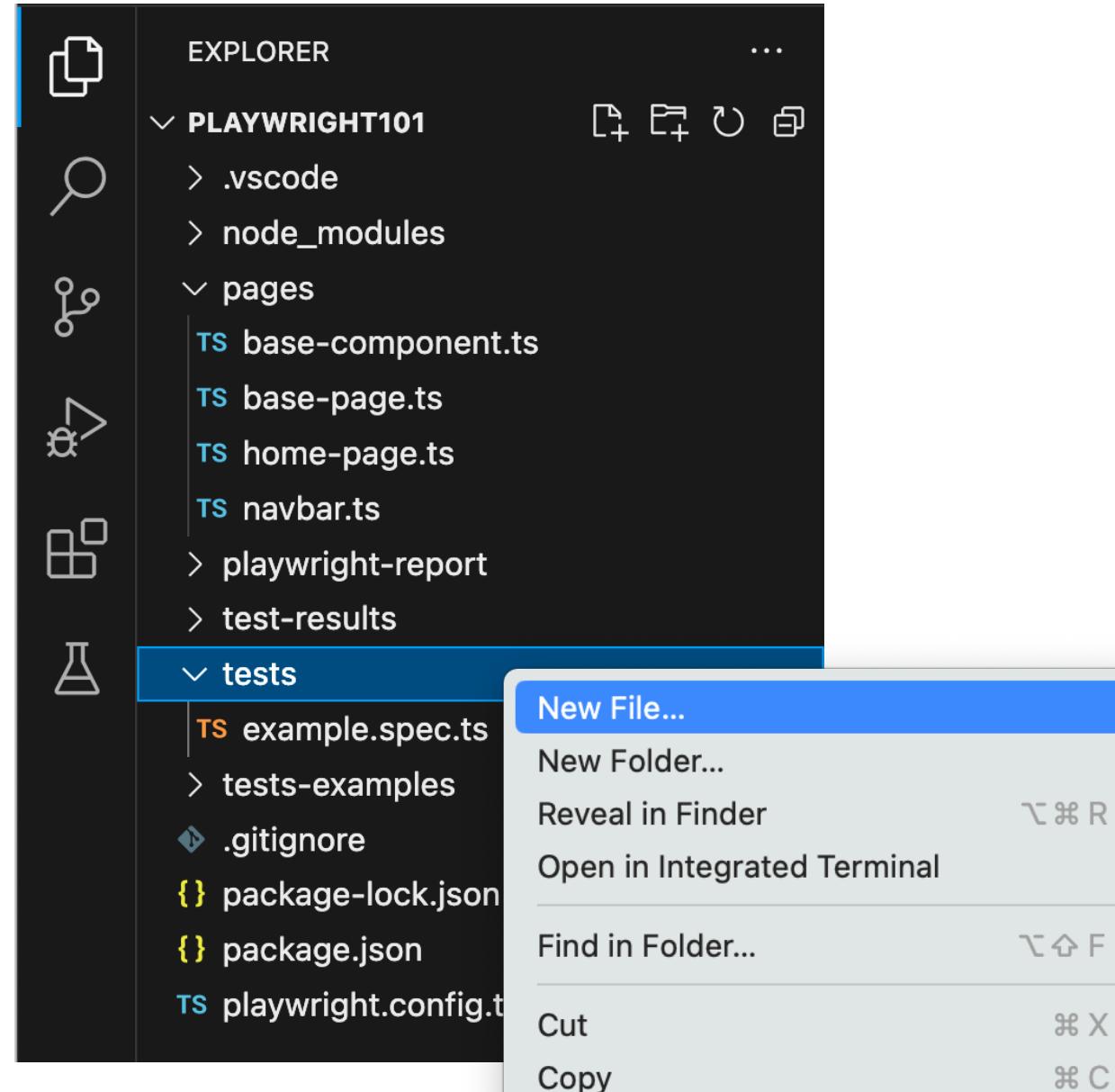
The true form of above chain locator is

```
page.locator("css=userForm")
    .locator("xpath='//input[@id='userName']'")
```

Create first test

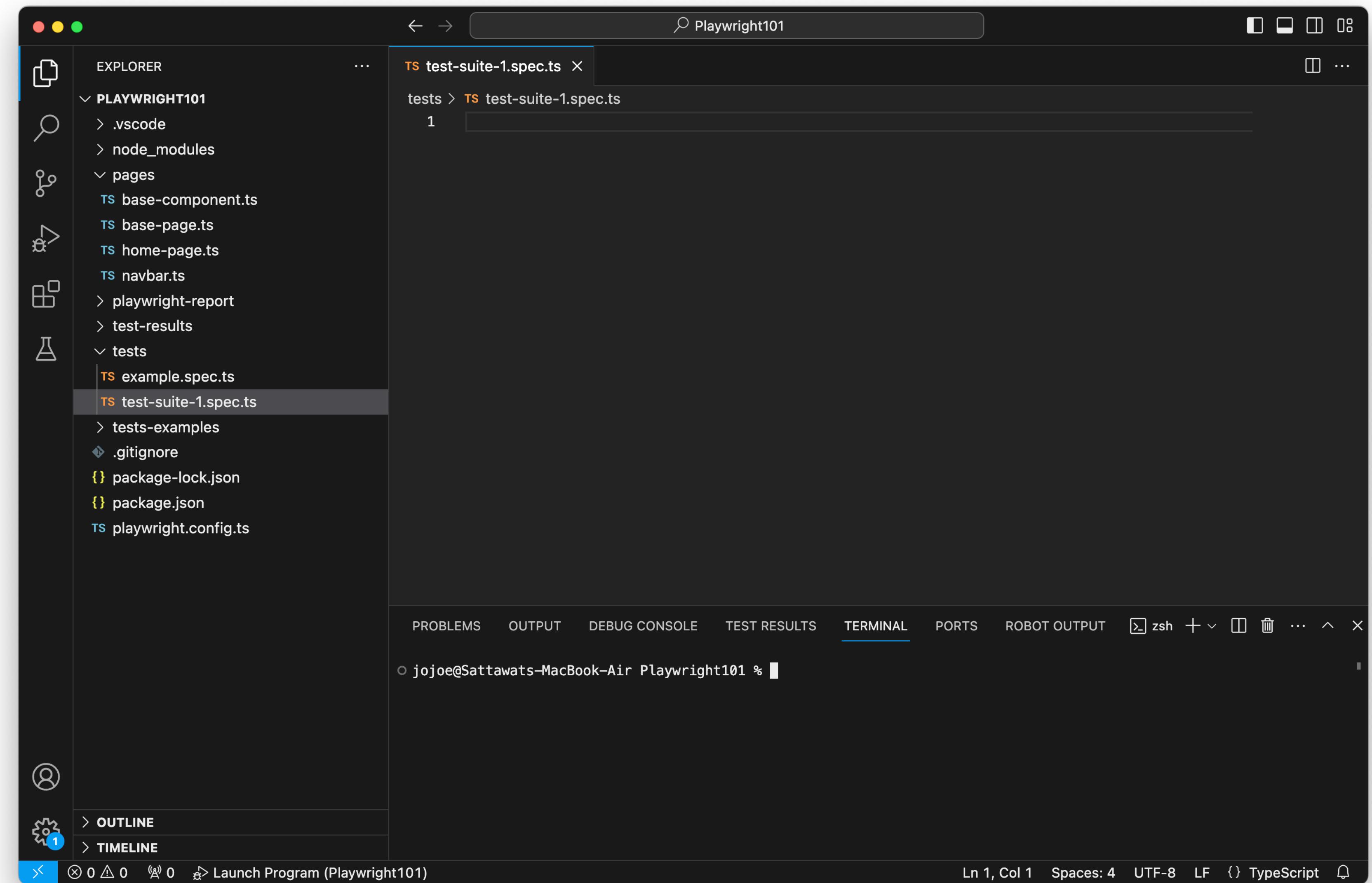
Suite
and
Scenario

Create first test suite file

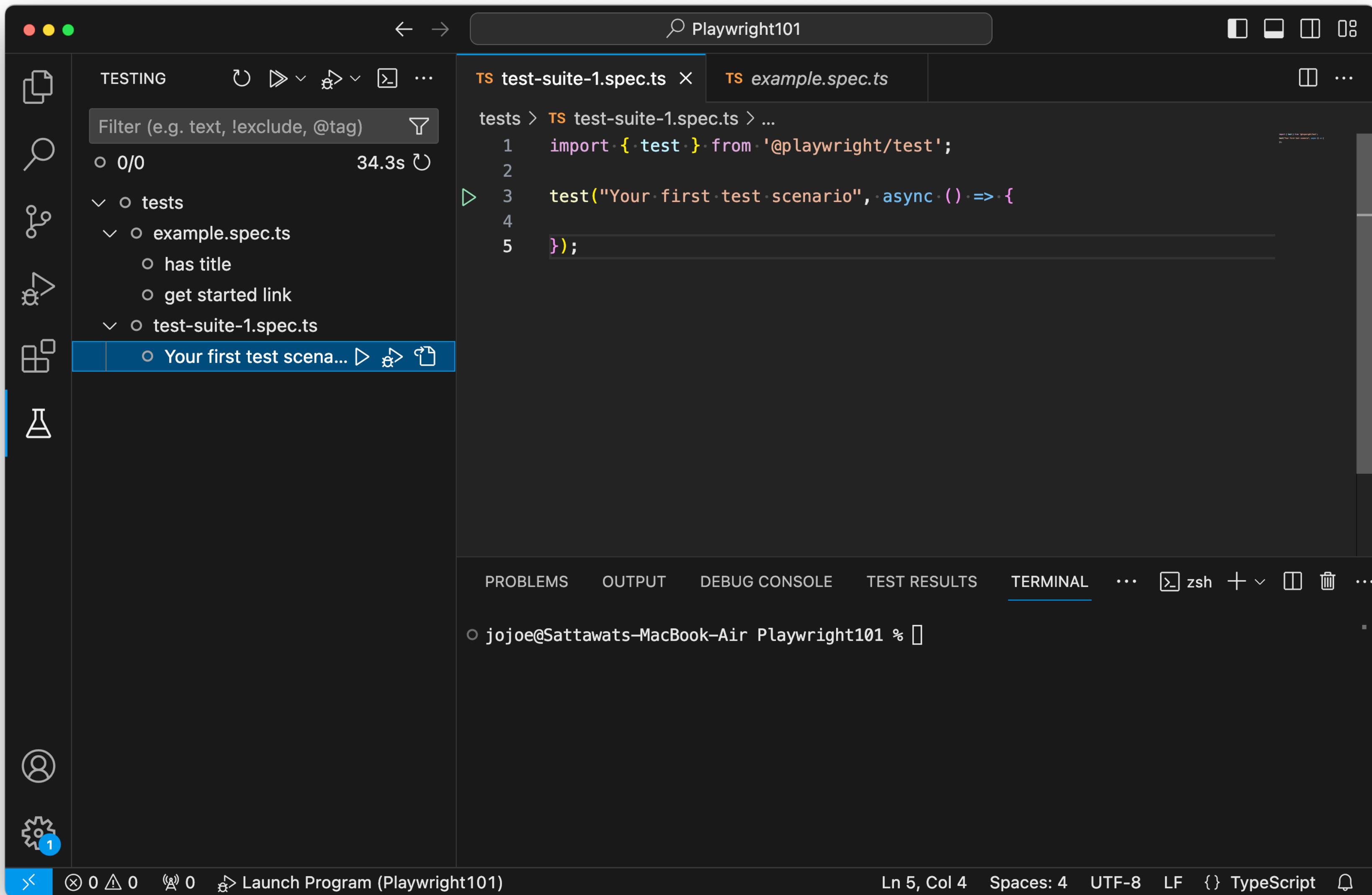


Create new test file (Test suite) steps:

1. Right click at folder "tests"
2. Select "New File..."
3. Naming your test file, this file must ending with .spec.ts



Create first test scenario



- Navigation
- Button
- Mouse click
- Text field
- Keyboard press
- Checkbox
- Radio button
- Dropdown
- Upload / download
- Drag and drop

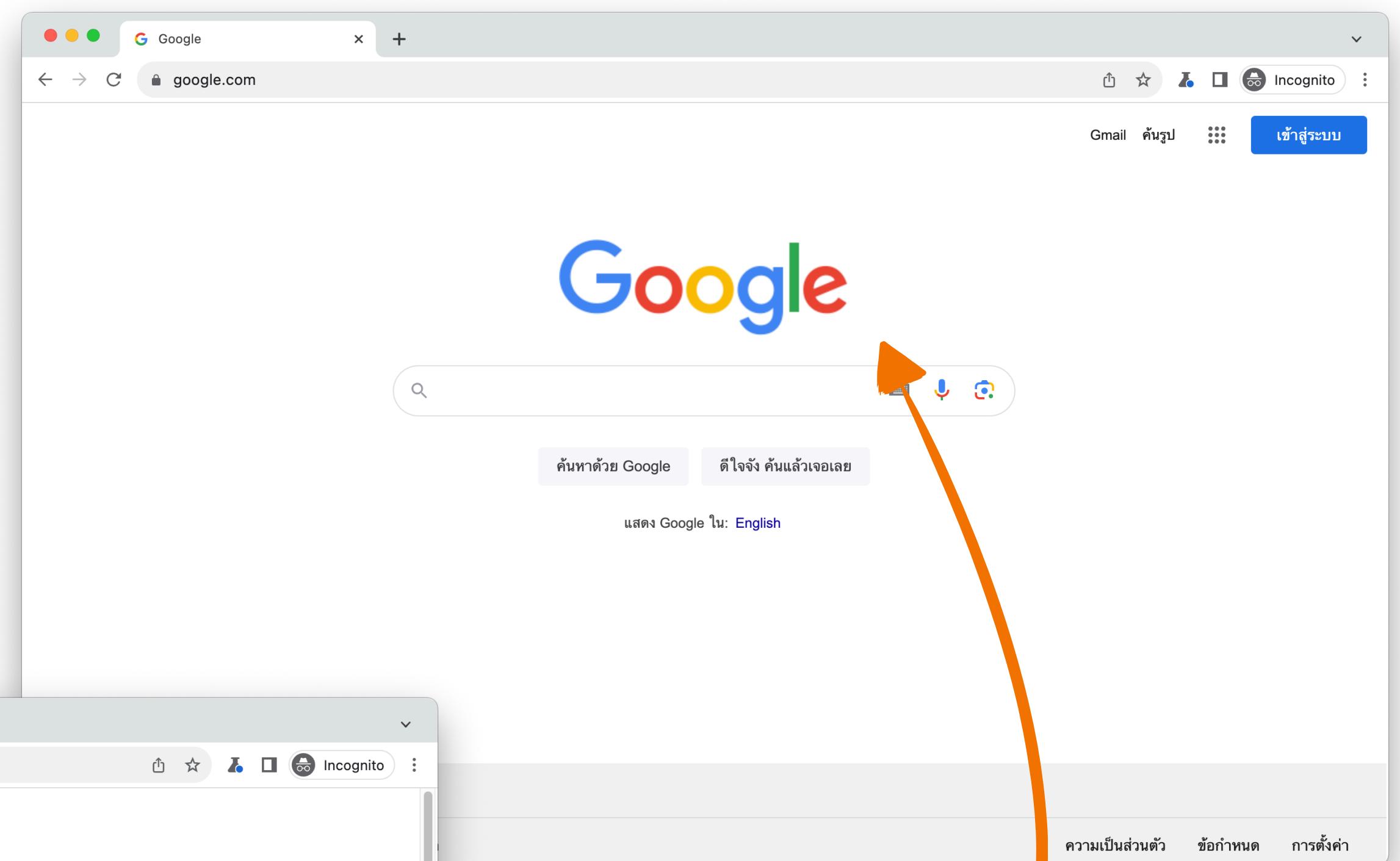
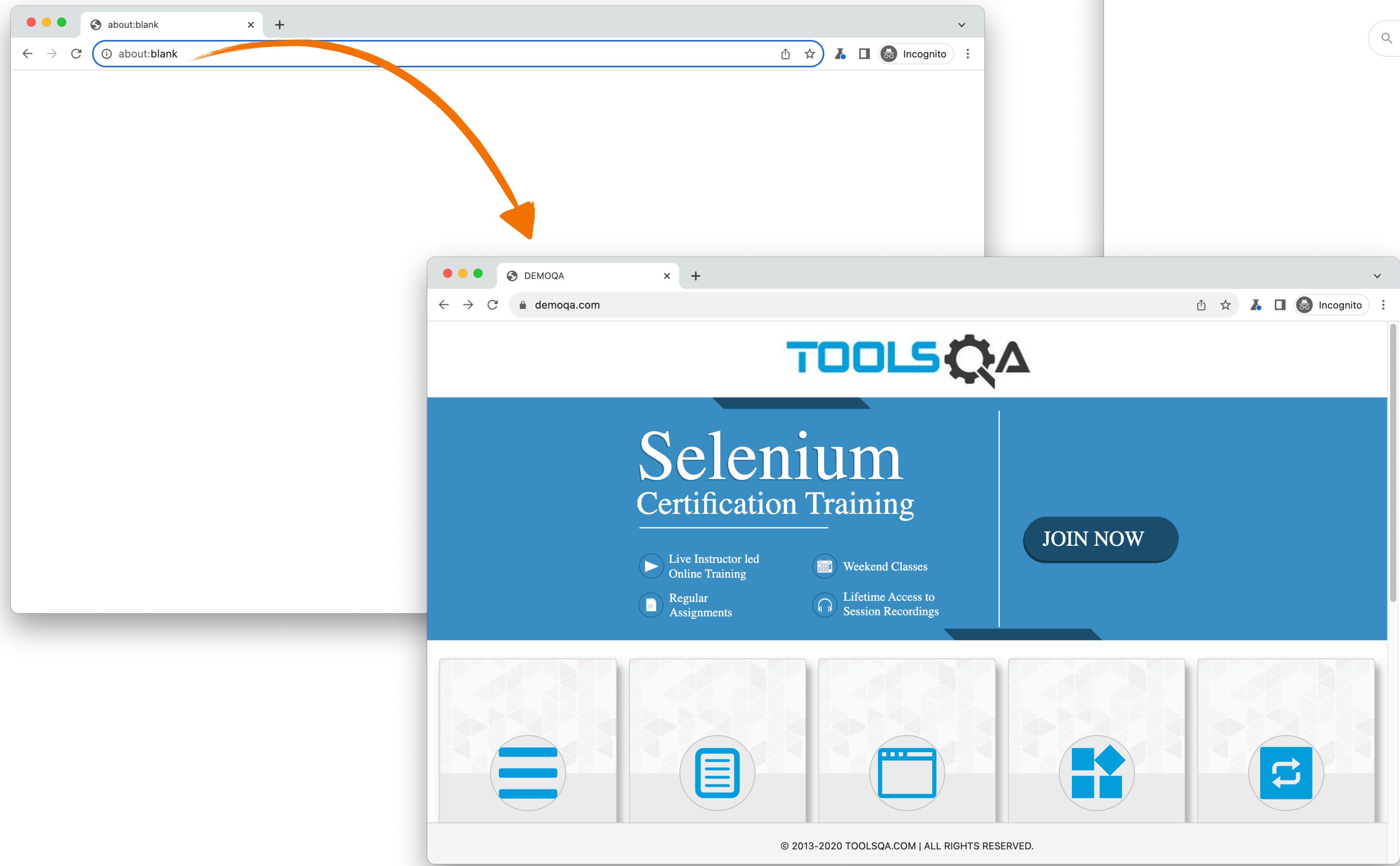
Actions

Scraping

Assertion

Page navigation

Page navigation



Methods using for page navigation

Method & Optionals	Return type	Purpose	Example
.goto()	<code>Promise< null Response ></code>	Navigate to the input url.	<code>await page.goto('https://...');</code>
.goBack()	<code>Promise< null Response ></code>	Navigate to the previous page in history.	<code>await page.goBack();</code>
.goForward()	<code>Promise< null Response ></code>	Navigate to the next page in history.	<code>await page.goForward();</code>

Playwright - Example code

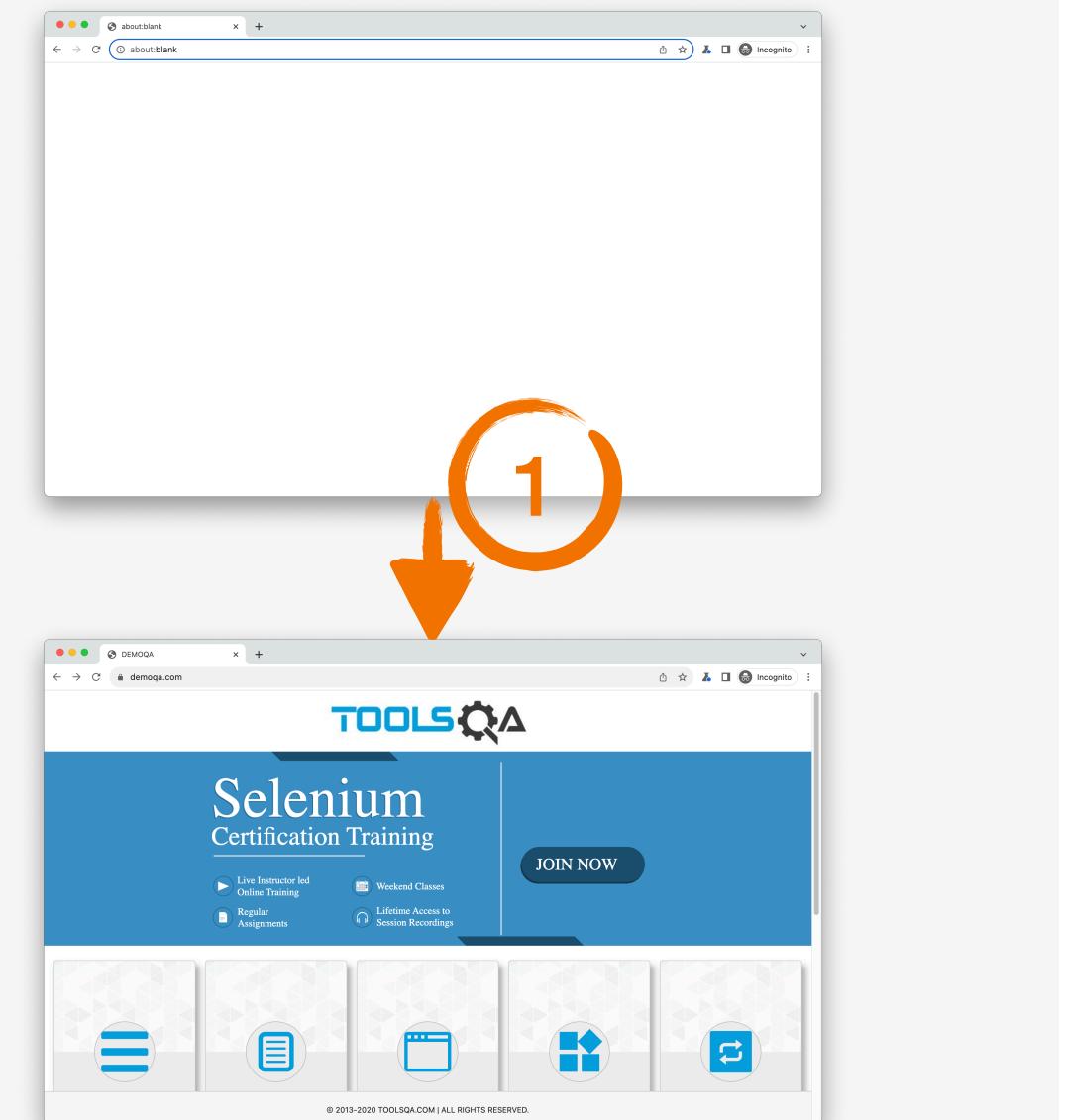
Method & Optionals	Return type	Purpose	Example
.goto()	Promise< null Response >	Navigate to the input url.	await page.goto('https://...');

```
test("Page navigation example - go to", async ({ page }) => {
  // Arrange
  const url = 'https://demoqa.com';

  // Action
  await page.goto(url);

  // Assert
  const currentUrl = page.url();

  expect(currentUrl).toEqual('https://demoqa.com/');
  await expect(page).toHaveURL('https://demoqa.com');
});
```



Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.goBack()	Promise< null Response >	Navigate to the previous page in history.	await page.goBack();

```
test("Page navigation example – backward", async ({ page }) => {
    // Arrange
    const url = 'https://demoqa.com'; ①
    await page.goto(url);

    const url2 = 'https://www.google.com'; ②
    await page.goto(url2);

    // Action
    await page.goBack(); ③

    // Assert
    const currentUrl = page.url();

    expect(currentUrl).toEqual('https://demoqa.com/');
    await expect(page).toHaveURL('https://demoqa.com');
});
```



Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.goForward()	Promise< null Response >	Navigate to the next page in history.	await page.goForward();

```
test("Page navigation example - forward", async ({ page }) => {
  // Arrange
  const url = 'https://demoqa.com'; ①
  await page.goto(url);

  const url2 = 'https://www.google.com'; ②
  await page.goto(url2);

  await page.goBack(); ③

  // Action
  await page.goForward(); ④

  // Assert
  const currentUrl = page.url();

  expect(currentUrl).toEqual('https://www.google.com/');
  await expect(page).toHaveURL('https://www.google.com');

});
```



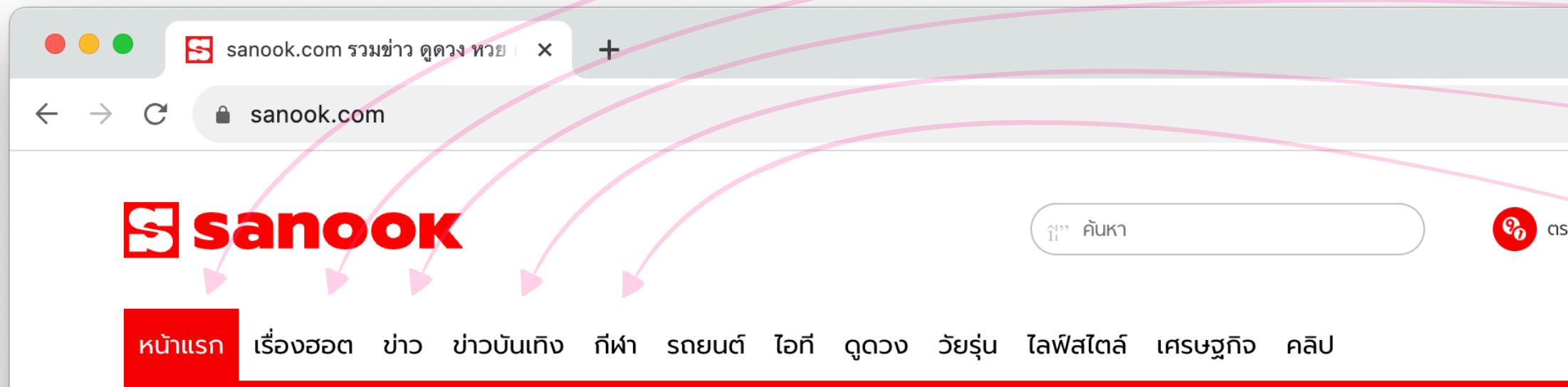
Text

Locate Elements

Text

>> www.sanook.com

Look! Some texts appear on the screen.



เรื่องเด่นวันนี้

วันอาทิตย์ ที่ 5 พฤษภาคม 2567



นายกลัวจะบาน ไม่รู้จะตอบยังไง พร:
ปรึกษาจะพ้องหย่าเมีย ชาวเน็ตอ่านแล้วร้อง

```
<div class="jsx-200851831 jsx-1997710517 MainMenuDesktop">
  ...
  <nav class="jsx-200851831 jsx-1997710517 container">
    <ul class="jsx-200851831 jsx-1997710517">
      <li class="jsx-1560033539 li li--firstpage active">
        <a class="jsx-1560033539" href="https://www.sanook.com/">หน้าแรก</a>
      </li>
      <li class="jsx-1560033539 li li--hot">
        <a class="jsx-1560033539" href="https://www.sanook.com/hot/">เรื่องhot</a>
      </li>
      <li class="jsx-3063308867 li li--news">
        <a class="jsx-3063308867" href="https://www.sanook.com/news/">ข่าว</a>
      </li>
      <li class="jsx-2027901635 li li--entertain">
        <a class="jsx-2027901635" href="https://www.sanook.com/entertain/">ข่าวบันเทิง</a>
      </li>
      <li class="jsx-2271696899 li li--sport">
        <a class="jsx-2271696899" href="https://www.sanook.com/sport/">กีฬา</a>
      </li>
      <li class="jsx-2939420547 li li--auto">
        <a class="jsx-2939420547" href="https://www.sanook.com/auto/">รถยนต์</a>
      </li>
      <li class="jsx-4086506115 li li--it">
        <a class="jsx-4086506115" href="https://www.sanook.com/it/">ไอที</a>
      </li>
      <li class="jsx-757620355 li li--horoscope">
        <a class="jsx-757620355" href="https://www.sanook.com/horoscope/">ดูดวง</a>
      </li>
      <li class="jsx-64817411 li li--campus">
        <a class="jsx-64817411" href="https://www.sanook.com/campus/">วัยรุ่น</a>
      </li>
      <li class="jsx-725620227 li li--lifestyle">
        <a class="jsx-725620227" href="https://www.sanook.com/lifestyle/">ไลฟ์สไตล์</a>
      </li>
      <li class="jsx-1288181763 li li--money">
        <a class="jsx-1288181763" href="https://www.sanook.com/money/">เศรษฐกิจ</a>
      </li>
      <li class="jsx-1560033539 li li--video">
        <a class="jsx-1560033539" href="https://www.sanook.com/video/">คลิป</a>
      </li>
    </ul>
  </nav>
</div>
```

Frequently methods using with text

Method & Optionals	Return type	Purpose	Example
<code>.innerText()</code>	<code>Promise<string></code>	Returns the <code>[element.innerText]</code>	<code>const text = await locator.innerText();</code>
<code>.allInnerTexts()</code>	<code>Promise<Array<string>></code>	Returns an array of <code>'node.innerText'</code> values for all matching nodes.	<code>const text = await locator.allInnerTexts();</code>
<code>.textContent()</code>	<code>Promise<null string></code>	Returns the <code>[node.textContent]</code>	<code>const text = await locator.textContent();</code>
<code>.allTextContents</code>	<code>Promise<Array<string>></code>	Returns an array of <code>'node.textContent'</code> values for all matching nodes.	<code>const text = await locator.allTextContents();</code>

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.innerText()	Promise<string>	Returns the [element.innerText]	const text = await locator.innerText();

```
test('Text : Get inner text', async ({ page }) => {
    // Arrange
    const url = 'https://sanook.com';
    await page.goto(url);

    const textLocator = page.locator('css=div.MainMenuDesktop li.li--news');

    // Action
    const text = await textLocator.innerText();

    // Assert
    expect(text).toContain('ข่าว');
});
```



text = 'ข่าว'

text in add to watch

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.allInnerTexts()	Promise< Array<string> >	Returns an array of 'node.innerText' values for all matching nodes.	const text = await locator.allInnerTexts();

```

test('Text : Get all inner texts', async ({ page }) => {
  // Arrange
  const url = 'https://sanook.com';
  await page.goto(url);

  const textLocator = page.locator('css=div.MainMenuDesktop li[class*="li--"]');

  // Action
  const texts = await textLocator.allInnerTexts();

  // Assert
  expect(texts).toContain('ข่าว');
});

```



```
texts = ['หน้าแรก', 'เรื่องฮอต', 'ข่าว', 'ข่าวบันเทิง', 'กีฬา', 'รถยนต์', 'ไอที', 'ดูดวง', 'วัยรุ่น', 'ไลฟ์สไตล์', 'เศรษฐกิจ', 'คลิป']
```

text in add to watch

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.textContent()	Promise< null string >	Returns the [node.textContent]	const text = await locator.textContent();

```
test('Text : Get text content', async ({ page }) => {
    // Arrange
    const url = 'https://sanook.com';
    await page.goto(url);

    const textLocator = page.locator('css=div.MainMenuDesktop li.li--news');

    // Action
    const text = await textLocator.textContent();

    // Assert
    expect(text).toContain('ข่าว');
});
```

text = 'ข่าว'

text in add to watch

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.allTextContents	Promise< Array<string> >	Returns an array of 'node.textContent' values for all matching nodes.	const text = await locator.allTextContents();

```
test('Text : Get all text contents', async ({ page }) => {
    // Arrange
    const url = 'https://sanook.com';
    await page.goto(url);

    const textLocator = page.locator('css=div.MainMenuDesktop li[class*="li--"]');

    // Action
    const texts = await textLocator.allTextContents();

    // Assert
    expect(texts).toContain('ข่าว');
});
```

```
texts = ['หน้าแรก', 'เรื่องฮอต', 'ข่าว', 'ข่าวบันเทิง', 'กีฬา', 'รถยนต์', 'ไอที', 'ดูดวง', 'วัยรุ่น', 'ไลฟ์สไตล์', 'เศรษฐกิจ', 'คลิป']
```

text in add to watch

Difference between

.innerText()
.getAllInnerTexts()

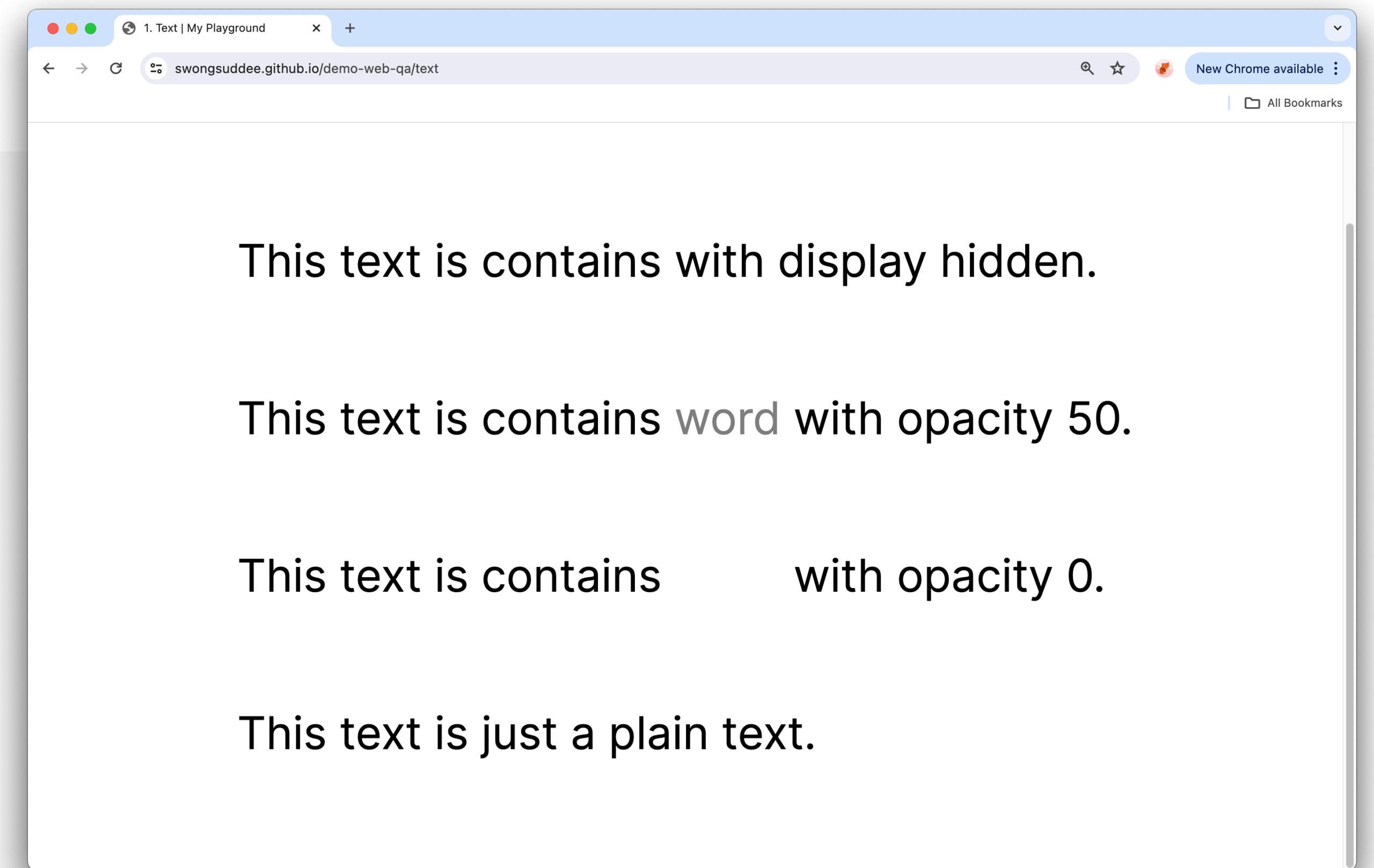
This method **returns the visible text contained within an element**, excluding any hidden text or text within elements that are hidden with CSS. It represents what is visibly rendered on the page.

.textContent()
.getAllTextContents()

This method **returns all the text content contained within an element, including both visible and hidden text**. It returns the text exactly as it is in the HTML source, regardless of its visibility on the page.

Example hidden text

```
<html lang="en">  
  <head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Example</title>  
    <style>  
      .hidden {  
        display: none;  
      }  
  
      .opacity0 {  
        opacity: 0;  
      }  
  
      .opacity50 {  
        opacity: 0.3;  
      }  
    </style>  
  </head>  
  <body>  
    <div id="example">  
      <p data-testid="hidden_p">This is some <span class="hidden">hidden</span> text.</p>  
      <p data-testid="opacity_30_p">This is some <span class="opacity30">dissolve</span> text.</p>  
      <p data-testid="opacity_0_p">This is some <span class="opacity0">invisible</span> text.</p>  
      <p data-testid="normal_p">This is another paragraph.</p>  
    </div>  
  </body>  
</html>
```



Example hidden text

```
test('Text : Getting hidden text', async ({ page }) => {
  const url = 'https://swongsuddee.github.io/demo-web-qa/';
  await page.goto(url);

  const hiddenText = page.getByTestId('hidden_p');
  console.log(await hiddenText.innerText());
  console.log(await hiddenText.textContent());

  const opacity30Text = page.getByTestId('opacity_30_p');
  console.log(await opacity30Text.innerText());
  console.log(await opacity30Text.textContent());

  const opacity0Text = page.getByTestId('opacity_0_p');
  console.log(await opacity0Text.innerText());
  console.log(await opacity0Text.textContent());

  const normalText = page.getByTestId('normal_p');
  console.log(await normalText.innerText());
  console.log(await normalText.textContent());
});
```

Console output

This is some text.

This is some hidden text.

This is some dissolve text.

This is some dissolve text.

This is some invisible text.

This is some invisible text.

This is another paragraph.

This is another paragraph.

Button

`<button> ... </button>`

Mouse click

>> Elements > Buttons

Identify the locator of the element we need and click on it!

Buttons

Double Click Me

Right Click Me

Click Me

You have done a double click

You have done a right click

You have done a dynamic click

© 2013-2020 TOOLSQA.COM | ALL RIGHTS RESERVED.

```
<div>
  <h1 class="text-center">Buttons</h1>
  <div>
    <button id="doubleClickBtn" type="button" class="btn btn-primary">
      Double Click Me
    </button>
  </div>
  <div class="mt-4">
    <button id="rightClickBtn" type="button" class="btn btn-primary">
      Right Click Me
    </button>
  </div>
  <div class="mt-4">
    <button id="PRbG9" type="button" class="btn btn-primary">
      Click Me
    </button>
  </div>
  <p id="doubleClickMessage">
    You have done a double click
  </p>
  <p id="rightClickMessage">
    You have done a right click
  </p>
  <p id="dynamicClickMessage">
    You have done a dynamic click
  </p>
</div>
...  
58
```

Frequently methods using with button

Method & Optionals		Return type	Purpose	Example
<code>.click()</code>	<code>{ button: 'left' }</code>	<code>Promise<void></code>	Click an element.	<code>await locator.click();</code>
	<code>{ button: 'right' }</code>	<code>Promise<void></code>	Right click an element.	<code>await locator.click({ button: 'right' });</code>
<code>.dblclick()</code>	<code>{ button: 'left' }</code>	<code>Promise<void></code>	Double-click an element.	<code>await locator dblclick();</code>
	<code>{ button: 'right' }</code>	<code>Promise<void></code>	Double-right-click an element.	<code>await locator dblclick({ button: 'right' });</code>
<code>.hover()</code>		<code>Promise<void></code>	Hover over the matching element.	<code>await locator.hover();</code>
<code>.isEnabled()</code>		<code>Promise<boolean></code>	Returns whether the element is enabled.	<code>const isEnabled = await locator.isEnabled();</code>
<code>.isDisable()</code>		<code>Promise<boolean></code>	Returns whether the element is disabled.	<code>const isDisabled = await locator.isDisabled();</code>

Playwright - Example code

Method & Optionals		Return type	Purpose	Example
<code>.click()</code>	<code>{ button: 'left' }</code>	<code>Promise<void></code>	Click an element.	<code>await locator.click();</code>
	<code>{ button: 'right' }</code>	<code>Promise<void></code>	Right click an element.	<code>await locator.click({ button: 'right' });</code>

```
test("Mouse click example : Do left click", async ({ page }) => {
    // Arrange
    const url = 'https://demoqa.com/buttons';
    await page.goto(url);

    const clickMeBtn = page.getText('Click Me', { exact: true });

    // Action
    await clickMeBtn.click();

    // Assert
    const clickedText = page.locator('id=dynamicClickMessage');
    await expect(clickedText).toHaveText('You have done a dynamic click');
});
```

Playwright - Example code

Method & Optionals		Return type	Purpose	Example
<code>.dblclick()</code>	<code>{ button: 'left' }</code>	<code>Promise<void></code>	Double-click an element.	<code>await locator.dblclick();</code>
	<code>{ button: 'right' }</code>	<code>Promise<void></code>	Double-right-click an element.	<code>await locator.dblclick({ button: 'right' });</code>

```
test("Mouse click example : Do double click", async ({ page }) => {
    // Arrange
    const url = 'https://demoqa.com/buttons';
    await page.goto(url);

    const doubleClickBtn = page.locator('id=doubleClickBtn');

    // Action
    await doubleClickBtn dblclick();

    // Assert
    const clickedText = page.locator('id=doubleClickMessage');
    await expect(clickedText).toHaveText('You have done a double click');
});
```

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
<code>.click()</code>	<code>{ button: 'left' }</code>	<code>Promise<void></code>	Click an element. <code>await locator.click();</code>
	<code>{ button: 'right' }</code>	<code>Promise<void></code>	Right click an element. <code>await locator.click({ button: 'right' });</code>

```
test("Mouse click example : Do right click", async ({ page }) => {
    // Arrange
    const url = 'https://demoqa.com/buttons';
    await page.goto(url);

    const rightClickBtn = page.locator('id=rightClickBtn');

    // Action
    await rightClickBtn.click({ button: 'right' });

    // Assert
    const clickedText = page.locator('id=rightClickMessage');
    await expect(clickedText).toHaveText('You have done a right click');
});
```

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.hover()	Promise<void>	Hover over the matching element.	await locator.hover();

```
test("Mouse example 1 – hover", async ({ page }) => {
  // Arrange
  const url = 'https://demoqa.com/buttons';
  await page.goto(url);

  const clickMeBtn = page.getText('Click Me', { exact: true });

  // Action
  await clickMeBtn.hover();
});
```

Click Me

```
.btn-primary {
  color: #fff;
  background-color: #007bff;
  border-color: #007bff;
}
```

Click Me



```
.btn-primary:hover {
  color: #fff;
  background-color: #0069d9;
  border-color: #0062cc;
}
```

Button enabled / disabled

>> Elements > Dynamic Properties
Is that button ready to be clicked ?

The screenshot shows a browser window with the title 'DEMOQA' and the URL 'demoqa.com/dynamic-properties'. On the left, a sidebar menu lists various UI elements: Elements, Text Box, Check Box, Radio Button, Web Tables, Buttons, Links, Broken Links - Images, and Upload and Download. The main content area is titled 'Dynamic Properties' and contains several interactive elements:

- A text input field with placeholder 'Type here...'. A pink arrow points from the word 'disabled' in the code block below to this input field.
- A text area containing the random ID 'This text has random Id'.
- A button labeled 'Will enable 5 seconds'.
- A button labeled 'Color Change'.
- A button labeled 'Visible After 5 Seconds'.

On the right, two code snippets are shown side-by-side, each enclosed in a grey box:

Disabled status

```
<button
  id="enableAfter"
  disabled
  type="button"
  class="mt-4 btn btn-primary">
  Will enable 5 seconds
</button>
```

Enabled status

```
<button
  id="enableAfter"
  type="button"
  class="mt-4 btn btn-primary">
  Will enable 5 seconds
</button>
```

At the bottom of the page, a copyright notice reads: © 2013-2020 TOOLSQA.COM | ALL RIGHTS RESERVED.

Locate Elements ➔ Playwright locator

```
<button  
  id="enableAfter"  
  disabled  
  type="button"  
  class="mt-4 btn btn-primary"  
>  
  Will enable 5 seconds  
</button>
```

```
<button  
  id="enableAfter"  
  type="button"  
  class="mt-4 btn btn-primary"  
>  
  Will enable 5 seconds  
</button>
```

Disabled status

Enabled status

```
page.locator('id=enableAfter');  
page.locator('css=button#enableAfter');  
page.locator('xpath=/button[@id="enableAfter"]');  
page.getByRole(  
  'button',  
  { name: "Will enable 5 seconds", exact: true }  
)
```

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.isEnabled()	Promise< boolean >	Returns whether the element is enabled.	const isEnabled = await locator.isEnabled();
.isDisable()	Promise< boolean >	Returns whether the element is disabled.	const isDisabled = await locator.isDisabled();

```

test('Check button is enabled', async ({ page }) => {
    // Arrange
    const url = 'https://demoqa.com/dynamic-properties';
    await page.goto(url);

    const button = page.locator('id=enableAfter');
    const isDisable = await button.isEnabled();

    // Action
    await page.waitForTimeout(6_000);

    // Assert
    expect(isDisable).toBeFalsy();

    const isEnabled = await button.isDisabled();
    expect(isEnabled).toBeTruthy();
});
```

Mouse Hover

>> Widgets > Tool Tips
Something is hiding ('')\

Tool Tips

Practice Tool Tips

Hover me to see You hovered over the Button

Hover me to see

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

```
<div id="buttonToolTopContainer">
<p>Practice Tool Tips</p>
<button
  id="toolTipButton"
  type="button"
  class="btn btn-success"
  aria-describedby="buttonToolTip">
  Hover me to see
</button>
</div>
```

© 2013-2020 TOOLSQA.COM | ALL RIGHTS RESERVED.

Locate Elements ➔ Playwright locator

```
<div id="buttonToolTopContainer">
  <p>Practice Tool Tips</p>
  <button
    id="toolTipButton"
    type="button"
    class="btn btn-success"
    aria-describedby="buttonToolTip">
    Hover me to see
  </button>
</div>
```

```
page.locator('id=toolTipButton');
page.locator('css=button#toolTipButton');
page.locator('xpath=/button[@id="toolTipButton"] ');
page.getByRole(
  "button",
  { name: "Hover me to see", exact: true }
);
```

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.hover()	Promise<void>	Hover over the matching element.	await locator.hover();

```
test("Mouse example – hover", async ({ page }) => {
    // Arrange
    const url = 'https://demoqa.com/tool-tips';
    await page.goto(url);

    const hoverButton = page.locator('id=toolTipButton');

    const before_isVisible = await page.locator(
        'xpath//*[text()="You hovered over the Button"]'
    ).isVisible();

    // Action
    await hoverButton.hover();

    // Assert
    await page.waitForTimeout(1000);

    expect(before_isVisible).toBeFalsy();

    const hoveredMessage = page.locator('xpath//*[text()="You hovered over the Button"]');
    await expect(hoveredMessage).toBeVisible();
});
```

Practice Tool Tips

Hover me to see

You hovered over the Button



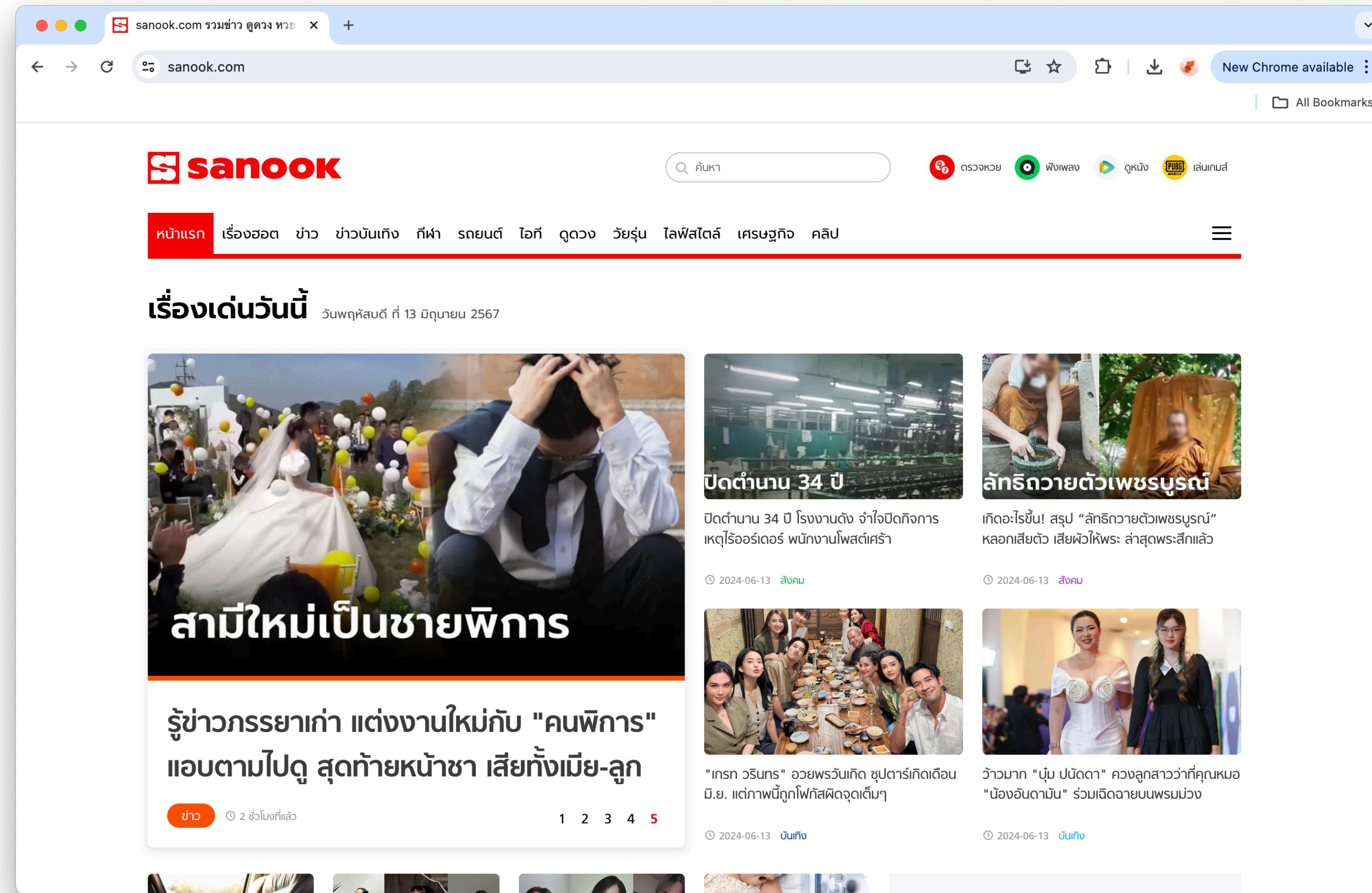
Text input

<input type="text"> & <textarea>

Text input

>> Elements > Text Box

Fill in the text boxes, then click the 'Submit' button.



Frequently methods using with input

Method & Optionals	Return type	Purpose	Example
<code>.clear()</code>	<code>Promise<void></code>	Clears it and triggers an `input` event after clearing.	<code>await locator.clear();</code>
<code>.inputValue()</code>	<code>Promise<string></code>	Returns `input.value` for the selected ` <code><input></code> ` or ` <code><textarea></code> ` or ` <code><select></code> ` element.	<code>await locator.inputValue();</code>
<code>.fill()</code>	<code>Promise<void></code>	Fills it and triggers an `input` event after filling.	<code>await locator.fill('Some text');</code>

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.fill()	Promise<void>	Fills it and triggers an `input` event after filling.	await locator.fill('Some text');
.inputValue()	Promise<string>	Returns `input.value` for the selected `<input>` or `<textarea>` or `<select>`	await locator.inputValue();

```
test("Text input example : Fill text to input field", async ({ page }) => {
    // Arrange
    const url = 'https://www.sanook.com/';
    await page.goto(url);

    const search = page.locator('css=div[class^="search"] input');

    // Action
    await search.fill('Hello World!');

    // Assert
    expect(search).toHaveValue('Hello World!');

    const searchValue = await search.inputValue();
    expect(searchValue).toEqual('Hello World!');
    expect(searchValue).toContain('World!');
});
```

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.clear()	Promise<void>	Clears it and triggers an `input` event after clearing.	await locator.clear();

```
test("Text input example : Clear text from input field", async ({ page }) => {
    // Arrange
    const url = 'https://www.sanook.com/';
    await page.goto(url);

    const search = page.locator('css=div[class^="search"] input');
    await search.fill('Hello World!');

    // Action
    await search.clear();

    // Assert
    expect(search).toHaveValue('');

    const searchValue = await search.inputValue();
    expect(searchValue).toEqual('');
    expect(searchValue).toContain('');
});
```

Checkbox

`<input type="checkbox">`

Checkbox

>> Elements > Check Box

Check! Check! Check! Uncheck!

The screenshot shows a web browser window displaying the w3schools.com/bootstrap5/bootstrap_form_check_radio.php page. The page title is "Bootstrap 5 Checkboxes and Radio Buttons". The main content area is titled "Checkboxes" and contains the following text: "Checkboxes are used if you want the user to select any number of options from a list of preset options." Below this, there is a list of checkboxes:

- Option 1
- Option 2
- Disabled Option

On the left side, there is a sidebar with a navigation menu for "Bootstrap 5 Forms" and "Bootstrap 5 Grid". The "BS5 Checks and Radios" option under "Bootstrap 5 Forms" is highlighted with a green background. The "Example" section contains the following code:

```
<div class="form-check">
  <input class="form-check-input" type="checkbox" id="check1" name="option1" value="something"
checked>
  <label class="form-check-label">Option 1</label>
</div>
```

Below the code is a "Try it Yourself" button. The "Example Explained" section contains the following text: "To style checkboxes, use a wrapper element with `class="form-check"` to ensure proper margins for labels and checkboxes. Then, add the `.form-check-label` class to label elements, and `.form-check-input` to style checkboxes properly inside the `.form-check` container."

Frequently methods using with checkbox

Method & Optionals		Return type	Purpose	Example
<code>.check()</code>		<code>Promise<void></code>	Ensure that checkbox or radio element is checked.	<code>await locator.check();</code>
<code>.uncheck()</code>		<code>Promise<void></code>	Ensure that checkbox or radio element is unchecked.	<code>await locator.uncheck();</code>
<code>.isChecked()</code>		<code>Promise<boolean></code>	Returns whether the element is checked. Throws if the element is not a checkbox or radio input.	<code>const isChecked = await locator.isChecked();</code>
<code>.setChecked()</code>	TRUE	<code>Promise<void></code>	Set the state of a checkbox or a radio element to be checked.	<code>await locator.setChecked(true);</code>
	FALSE	<code>Promise<void></code>	Set the state of a checkbox or a radio element to be unchecked.	<code>await locator.setChecked(false);</code>
<code>.isEnabled()</code>		<code>Promise<boolean></code>	Returns whether the element is disabled, the opposite of enabled.	<code>const isEnabled = await bikeBoxLocator.isEnabled();</code>

Playwright - Example code

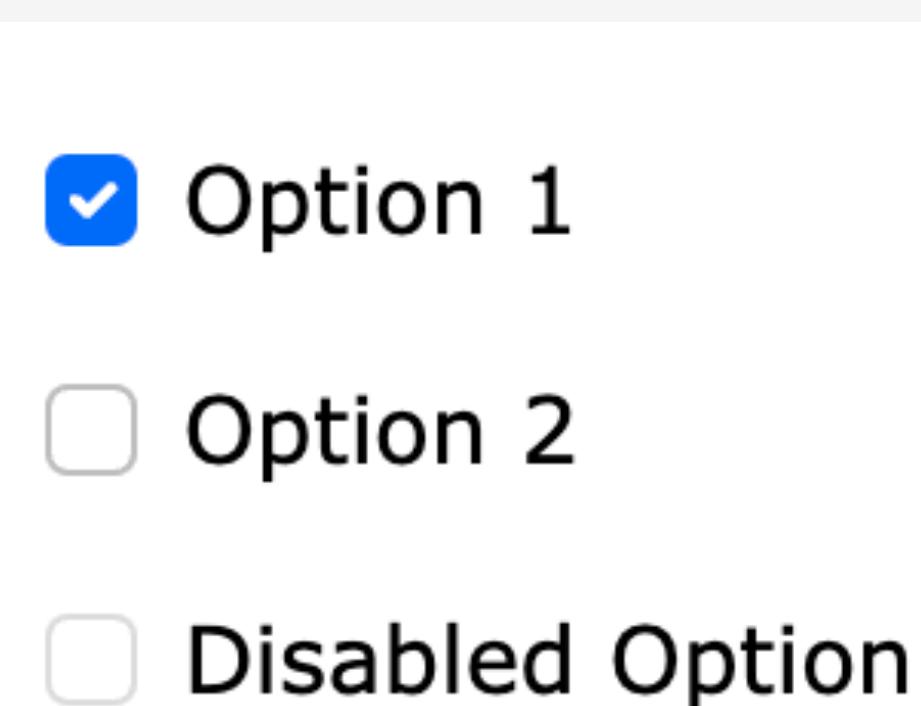
Method & Optionals	Return type	Purpose	Example
.isEnabled()	Promise<boolean>	Returns whether the element is disabled, the opposite of enabled.	<pre>const isEnabled = await bikeBoxLocator.isEnabled();</pre>

```
test("Check box : Is enabled check box", async ({ page }) => {
    // Arrange
    const url = "https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php";
    await page.goto(url);

    const option1 = page.locator('xpath=//label[contains(., "Option 1") and input[@type="checkbox"]]');
    const option2 = page.locator('xpath=//label[contains(., "Option 2") and input[@type="checkbox"]]');
    const option3 = page.locator('xpath=//label[contains(., "Disabled Option") and input[@type="checkbox"]]');

    // Action
    const isOption1Enabled = await option1.isEnabled();
    const isOption3Enabled = await option3.isEnabled();

    // Assertion
    expect(isOption1Enabled).toBeTruthy();
    expect(isOption3Enabled).toBeFalsy();
});
```



Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.check()	Promise<void>	Ensure that checkbox or radio element is checked.	await locator.check();
.isChecked()	Promise<boolean>	Returns whether the element is checked. Throws if the element is	const isChecked = await locator.isChecked();

```
test("Check box : Check the box", async ({ page }) => {
    // Arrange
    const url = "https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php";
    await page.goto(url);

    const option1 = page.locator('xpath=//label[contains(., "Option 1") and input[@type="checkbox"]]');
    const option2 = page.locator('xpath=//label[contains(., "Option 2") and input[@type="checkbox"]]');
    const option3 = page.locator('xpath=//label[contains(., "Disabled Option") and input[@type="checkbox"]]');

    // Action
    await option1.check();
    await option2.check()

    // Assert
    const isOption1Checked = await option1.isChecked();
    const isOption2Checked = await option2.isChecked();
    const isOption3Checked = await option3.isChecked();

    expect({ isOption1Checked, isOption2Checked, isOption3Checked })
        .toEqual({
            isOption1Checked: true, isOption2Checked: true, isOption3Checked: false
        });
});
```

Initial State:

- Option 1:
- Option 2:
- Disabled Option:

Final State (after execution):

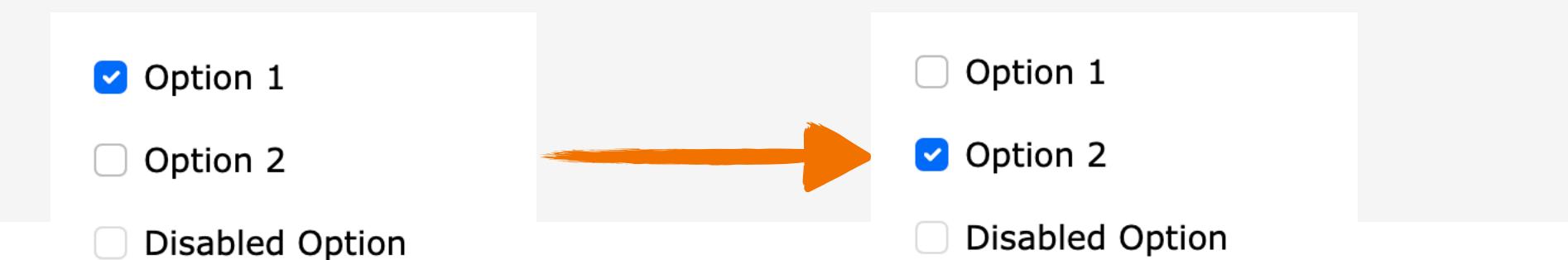
- Option 1:
- Option 2:
- Disabled Option:

Playwright - Example code

Method & Optionals		Return type	Purpose	Example
<code>.setChecked()</code>	TRUE	<code>Promise<void></code>	Set the state of a checkbox or a radio element to be checked.	<code>await locator.setChecked(true);</code>
	FALSE	<code>Promise<void></code>	Set the state of a checkbox or a radio element to be unchecked.	<code>await locator.setChecked(false);</code>

```
test("Check box : Set checked to the box", async ({ page }) => {
    // Arrange
    const url = "https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php";
    await page.goto(url);

    const option1 = page.locator('xpath=//label[contains(., "Option 1") and input[@type="checkbox"]]');
    const option2 = page.locator('xpath=//label[contains(., "Option 2") and input[@type="checkbox"]]');
    const option3 = page.locator('xpath=//label[contains(., "Disabled Option") and input[@type="checkbox"]]');
    
    await option1.setChecked(false);
    await option2.setChecked(true);
});
```



Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.uncheck()	Promise<void>	Ensure that checkbox or radio element is unchecked.	await locator.uncheck();

```

test("Check box : Unchecked to the box", async ({ page }) => {
    // Arrange
    const url = "https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php";
    await page.goto(url);

    const option1 = page.locator('xpath=//label[contains(., "Option 1") and input[@type="checkbox"]]');
    const option2 = page.locator('xpath=//label[contains(., "Option 2") and input[@type="checkbox"]]');
    const option3 = page.locator('xpath=//label[contains(., "Disabled Option") and input[@type="checkbox"]]');

    // Action
    await option1.uncheck();

    // Assert
    const isOption1Checked = await option1.isChecked();
    const isOption2Checked = await option2.isChecked();
    const isOption3Checked = await option3.isChecked();

    expect({ isOption1Checked, isOption2Checked, isOption3Checked })
        .toEqual({
            isOption1Checked: false, isOption2Checked: false, isOption3Checked: false
        });
});

```

Option 1
 Option 2
 Disabled Option



Option 1
 Option 2
 Disabled Option

Radio button

`<input type="radio">`

Radio button

>> Elements >> Radio Button

You can select only one of it.

The screenshot shows a web browser window displaying the w3schools.com/bootstrap5/bootstrap_form_check_radio.php page. The page title is "Radio buttons". A sidebar on the left lists various Bootstrap 5 components, with "BS5 Checks and Radios" highlighted. The main content area contains a heading "Radio buttons" and a paragraph explaining their purpose: "Radio buttons are used if you want to limit the user to just one selection from a list of preset options." Below this is a list of three radio buttons: "Option 1" (selected), "Option 2", and "Option 3". A "Try it Yourself" button is at the bottom. On the right, a code block shows the HTML code for the radio buttons:

```
<div class="form-check">
  <input type="radio" class="form-check-input" id="radio1" name="optradio" value="option1"
checked>Option 1
  <label class="form-check-label" for="radio1"></label>
</div>
<div class="form-check">
  <input type="radio" class="form-check-input" id="radio2" name="optradio"
value="option2">Option 2
  <label class="form-check-label" for="radio2"></label>
</div>
<div class="form-check">
  <input type="radio" class="form-check-input" disabled>Option 3
  <label class="form-check-label"></label>
</div>
```

Frequently methods using with radio button

Method & Optionals		Return type	Purpose	Example
<code>.check()</code>		<code>Promise<void></code>	Ensure that checkbox or radio element is checked.	<code>await locator.check();</code>
<code>.uncheck()</code>		Radio buttons cannot be unchecked.		
<code>.isChecked()</code>		<code>Promise<boolean></code>	Returns whether the element is checked. Throws if the element is not a checkbox or radio input.	<code>const isEnabled = await locator.isChecked();</code>
<code>.setChecked()</code>	TRUE	<code>Promise<void></code>	Set the state of a checkbox or a radio element to be checked.	<code>await locator.setChecked(true);</code>
	FALSE	<code>Promise<void></code>	Set the state of a checkbox or a radio element to be unchecked.	<code>await locator.setChecked(false);</code>
<code>.isEnabled()</code>		<code>Promise<boolean></code>	Returns whether the element is disabled, the opposite of enabled.	<code>const isEnabled = await bikeBoxLocator.isEnabled();</code>

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.isEnabled()	Promise< boolean >	Returns whether the element is disabled, the opposite of enabled.	const isEnabled = await bikeBoxLocator.isEnabled();

```
test('Radio button : Is enabled button', async ({ page }) => {  
    // Arrange  
    const url = 'https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php';  
    await page.goto(url);  
  
    const option1 = page.locator('input#radio1');  
    const option2 = page.locator('input#radio2');  
    const option3 = page.locator('input[type="radio"] [disabled]');  
  
    // Action  
    const isOption1Enabled = await option1.isEnabled();  
    const isOption3Enabled = await option3.isEnabled();  
  
    // Assertion  
    expect(isOption1Enabled).toBeTruthy();  
    expect(isOption3Enabled).toBeFalsy();  
});
```

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.check()	Promise<void>	Ensure that checkbox or radio element is checked.	await locator.check();
.isChecked()	Promise<boolean>	Returns whether the element is checked. Throws if the element is	const isChecked = await locator.isChecked();

```

test('Radio button :Check button', async ({ page }) => {
    // Arrange
    const url = 'https://www.w3schools.com/bootstrap5/bootstrap\_form\_check\_radio.php';
    await page.goto(url);

    const option1 = page.locator('input#radio1');
    const option2 = page.locator('input#radio2');
    const option3 = page.locator('input[type="radio"][disabled"]');

    // Action
    await option2.check();

    // Assertion
    const isOption2Checked = await option2.isChecked();
    expect(isOption2Checked).toBeTruthy();
});

```

Playwright - Example code

Method & Optionals		Return type	Purpose	Example
<code>.setChecked()</code>	TRUE	<code>Promise<void></code>	Set the state of a checkbox or a radio element to be checked.	<code>await locator.setChecked(true);</code>
	FALSE	<code>Promise<void></code>	Set the state of a checkbox or a radio element to be unchecked.	<code>await locator.setChecked(false);</code>

```
test("Radio button : Set checked to button", async ({ page }) => {
    // Arrange
    const url = 'https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php';
    await page.goto(url);

    const option1 = page.locator('input#radio1');
    const option2 = page.locator('input#radio2');
    const option3 = page.locator('input[type="radio"] [disabled]');

    // Arrange
    await option2.setChecked(true);
});
```

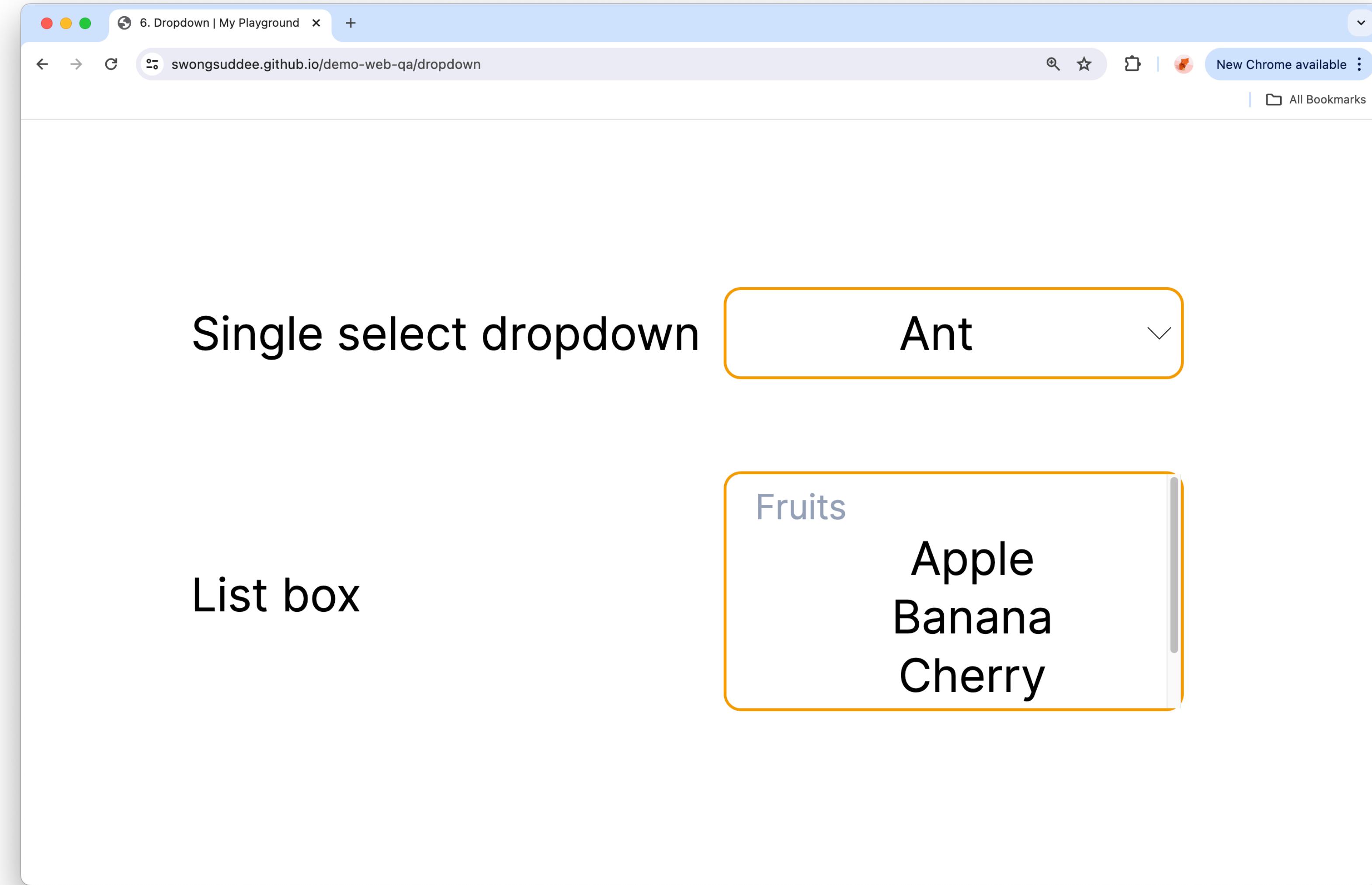
Dropdown

`<select> ... </select>`

Dropdown selection

>> Widgets > Select Menu

The old style "Select Menu" so ez to handle



Frequently methods using with dropdown

Method & Optionals		Return type	Purpose	Example
<code>.selectOption()</code>	Value	<code>Promise<Array<string>></code>	Selects option or options in `<select>` matches by `option.value`. Optional.	<code>const options = await locator.selectOption('red');</code>
	Label		Selects option or options in `<select>` matches by `option.label`. Optional.	<code>const options = await locator.selectOption({ value: 'red' });</code>
	Index		Selects option or options in `<select>` matches by the index. Optional.	<code>const options = await locator.selectOption({ label: 'Blue' });</code>
				<code>const options = await locator.selectOption({ index: 2 });</code>

Index 0 :
Index 1 :
Index 2 :

<option value="red">Red</option>
<option value="1">Blue</option>
<option value="2">Green</option>

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.selectOption()	Value Promise< Array<string> >	Selects option or options in `<select>` matches by `option.value`. Optional.	<pre>const options = await locator.selectOption('red'); const options = await locator.selectOption({ value: 'red' });</pre>

```
test("Select menu : Select from dropdown", async ({ page }) => {
  // Arrange
  const url = "https://swongsuddee.github.io/demo-web-qa/dropdown";
  await page.goto(url);

  const dropdownLocator = page.getByTestId('dropdown-selection');

  // Action
  await dropdownLocator.selectOption('Cat');

  // Assertion
  const selectedOption = await dropdownLocator.inputValue();
  expect(selectedOption).toEqual('value-Cat');
});
```

Ant
Bird
Cat
Dog

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.selectOption()	Label Promise< Array<string> >	Selects option or options in `<select>` matches by `option.label`. Optional.	<pre>const options = await locator.selectOption({ label: 'Blue' });</pre>

```
test("Select menu : Select from dropdown by value", async ({ page }) => {
  // Arrange
  const url = "https://swongsuddee.github.io/demo-web-qa/dropdown";
  await page.goto(url);

  const dropdownLocator = page.getByTestId('dropdown-selection');

  // Action
  await dropdownLocator.selectOption({ value: 'value-Bird' });

  // Assertion
  const selectedOption = await dropdownLocator.inputValue();
  expect(selectedOption).toEqual('value-Bird');
});
```

Ant
Bird
Cat
Dog

Playwright - Example code

Method & Optionals	Return type	Purpose	Example
.selectOption()	Index Promise< Array<string> >	Selects option or options in `<select>` matches by the index. Optional.	<pre>const options = await locator.selectOption({ index: 2 });</pre>

```
test("Select menu : Select from dropdown by index", async ({ page }) => {
  // Arrange
  const url = "https://swongsuddee.github.io/demo-web-qa/dropdown";
  await page.goto(url);

  const dropdownLocator = page.getByTestId('dropdown-selection');

  // Action
  await dropdownLocator.selectOption({ index: 3 });

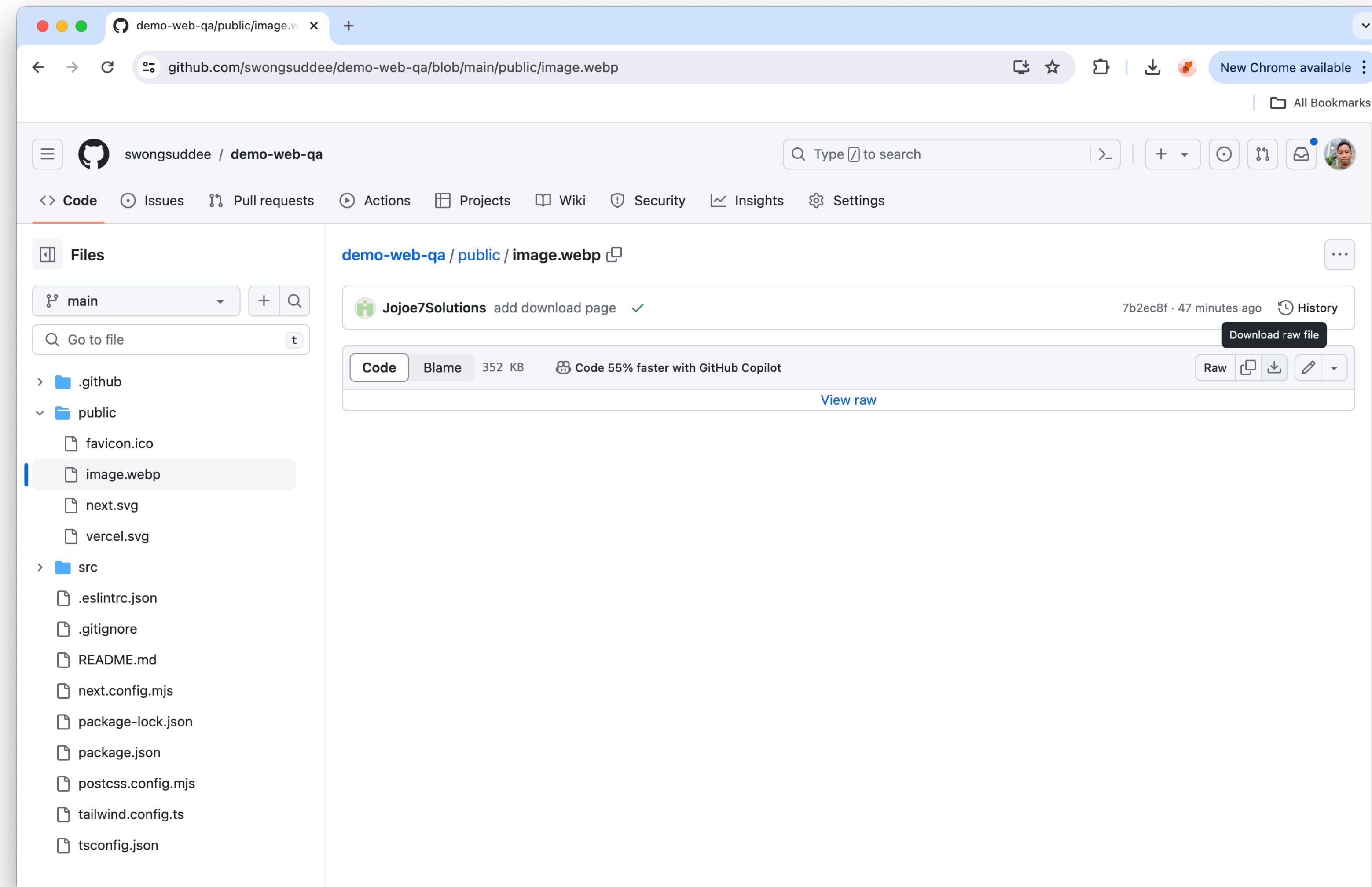
  // Assertion
  const selectedOption = await dropdownLocator.inputValue();
  expect(selectedOption).toEqual('value-Dog');
});
```

Ant
Bird
Cat
Dog

Download & Upload

Download

The example file is in my GitHub >> <https://github.com/swongsuddee/demo-web-qa/blob/main/public/image.webp>



Methods using for download file

Method & Optionals	Return type	Purpose	Example
<code>.waitForEvent()</code>	<code>Promise<Download></code>	Emitted when attachment download started. User can access basic file operations on downloaded content via the passed Download instance.	<pre>// Arrange const downloadPromise = page.waitForEvent('download'); // Action await downloadButton.click(); // Assertion const response = await downloadPromise;</pre>
<code>.suggestedFilename()</code>	<code>string</code>	Returns suggested filename for this download.	<pre>const fileName = response.suggestedFilename();</pre>

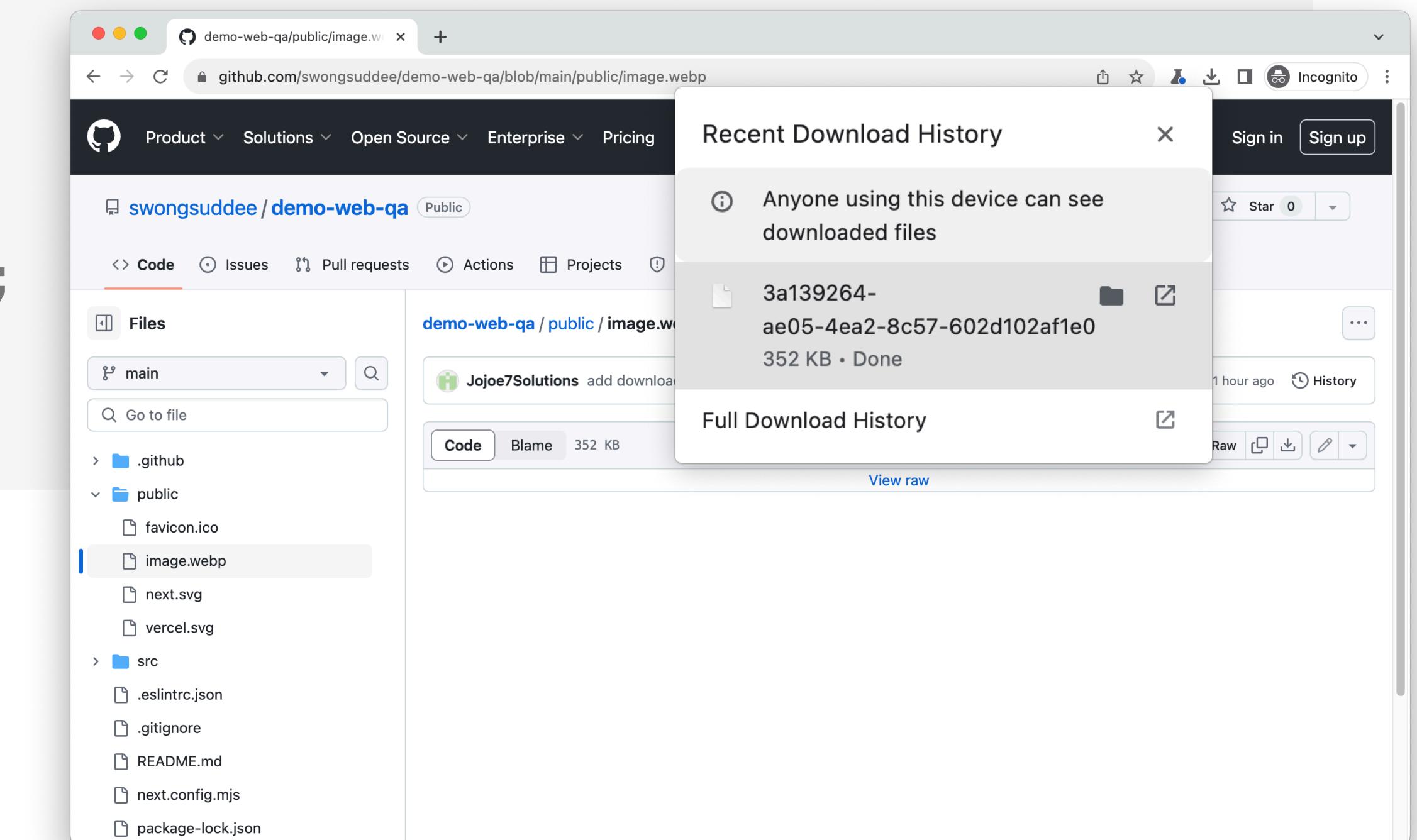
Playwright - Example code

```
test('Download : Download image', async ({ page }) => {
  // Arrange
  const url = "https://github.com/swongsuddee/demo-web-qa/blob/main/public/image.webp";
  await page.goto(url);

  const downloadButton = page.getByTestId('download-raw-button');
  const downloadPromise = page.waitForEvent('download');

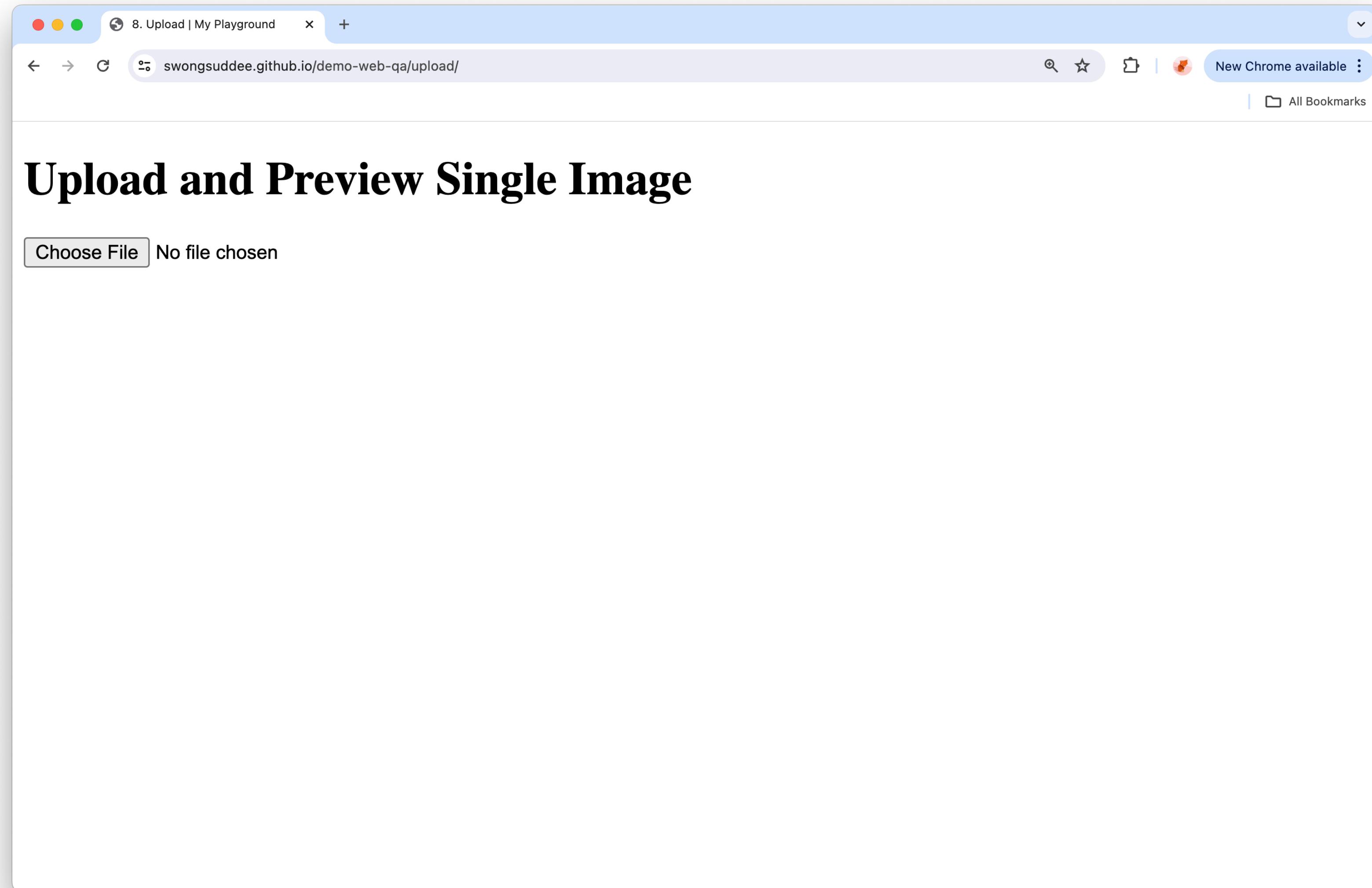
  // Action
  await downloadButton.click();

  // Assertion
  const response = await downloadPromise;
  const fileName = response.suggestedFilename();
  expect(fileName).toEqual('image.webp');
});
```



Upload

Up load some cute image ('')/



Methods using for upload file(s)

Method & Optionals	Return type	Purpose	Example
<code>.setInputFiles()</code>	<code>Promise<void></code>	Upload file or multiple files into `<input type=file>`.	<pre>await locator.setInputFiles('File path')</pre> <pre>await locator.setInputFiles(['File path 1', 'File path 2', ...])</pre>
<code>.inputValue()</code>	<code>Promise< string ></code>	Returns `input.value` for the selected `<input>` or `<textarea>` or `<select>` element.	<pre>await locator.inputValue();</pre>

Playwright - Example code

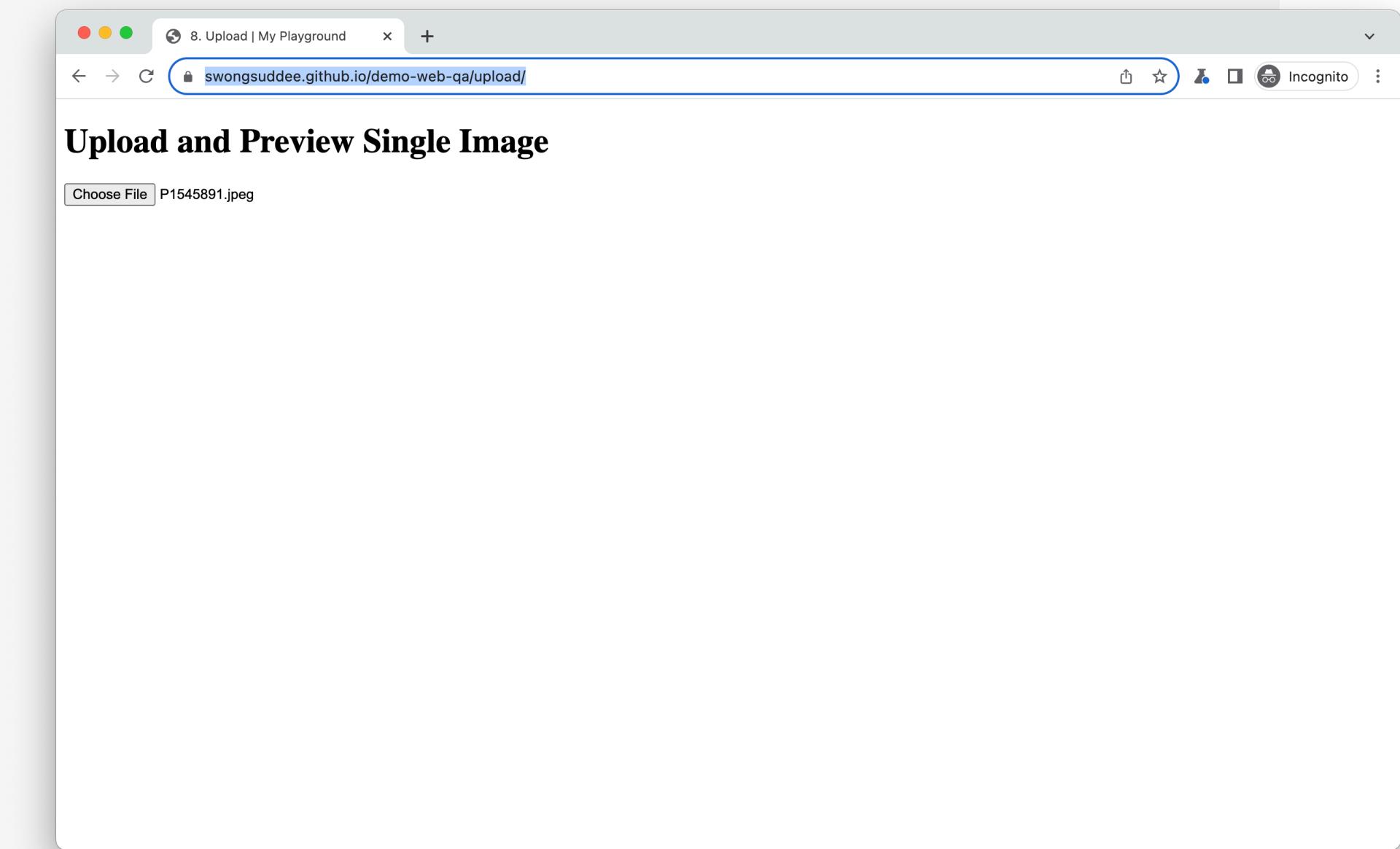
Method & Optionals	Return type	Purpose	Example
<code>.setInputFiles()</code>	<code>Promise<void></code>	Upload file or multiple files into `<input type=file>`.	<code>await locator.check() await locator.setInputFiles('File path');</code>
<code>.inputValue()</code>	<code>Promise<string></code>	Returns `input.value` for the selected `<input>` or `<textarea>` or `<select>`	<code>await locator.inputValue();</code>

```
test('Upload : Upload images', async ({ page }) => {
  // Arrange
  const url = "https://swongsuddee.github.io/demo-web-qa/upload/";
  await page.goto(url);

  const imageDirs = [
    path.join(__dirname, '../assets/P1545891.jpeg'),
  ];
  const inputLocator = page.locator('input');

  // Action
  await inputLocator.setInputFiles(imageDirs);

  // Assert
  const fakepath = await inputLocator.inputValue();
  expect(fakepath).toContain('P1545891.jpeg');
});
```



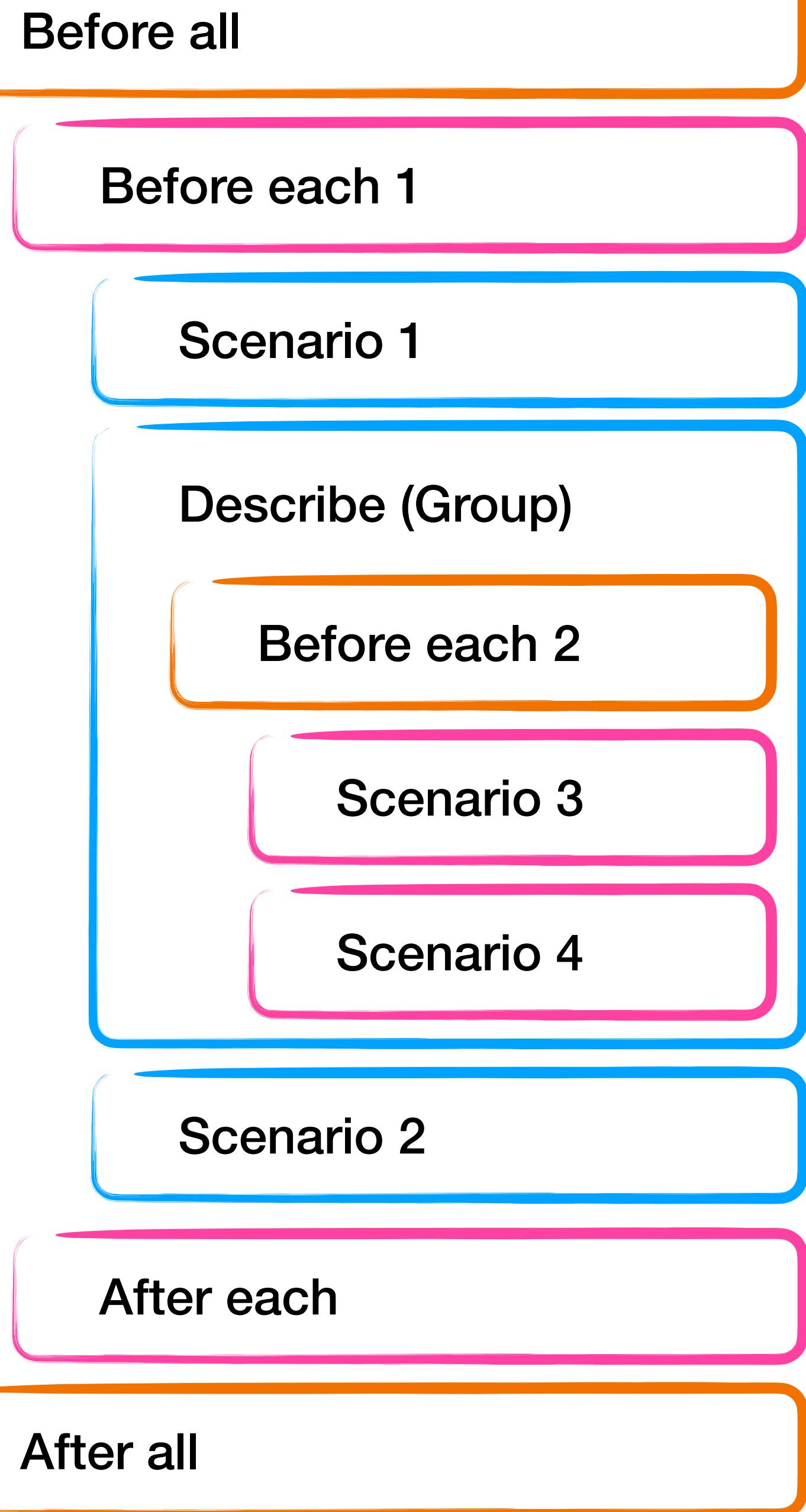
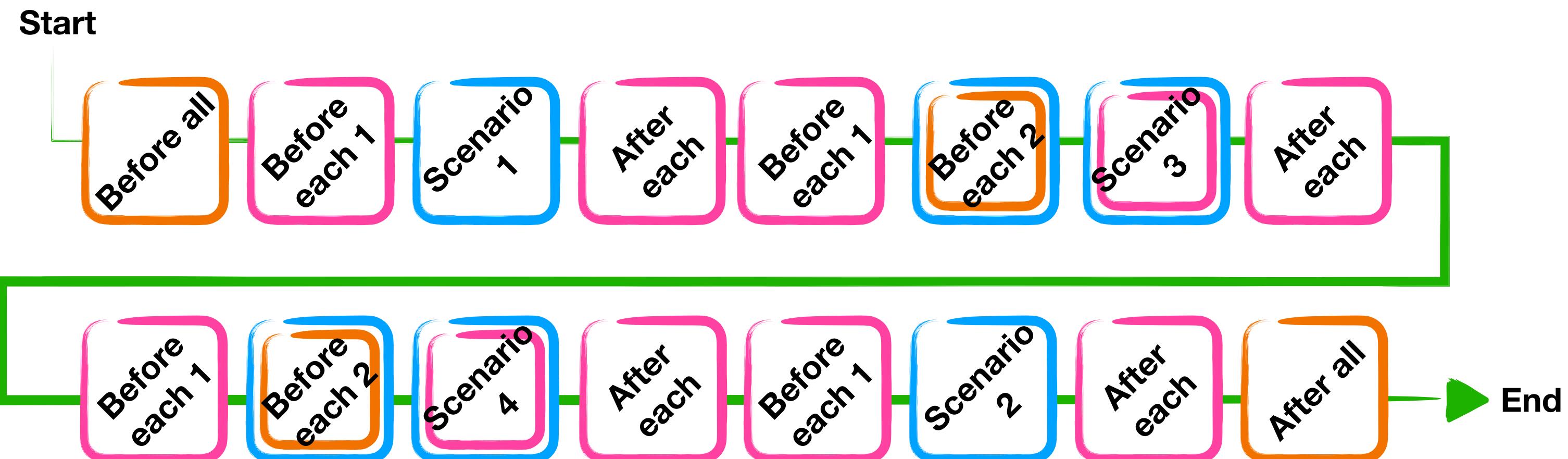
Test suite

- Test setup / teardown
 - `test.beforeAll()`
 - `test.beforeEach()`
 - `test.afterAll()`
 - `test.afterEach()`
- Groups
 - `test.describe()`
- Template

Test setup and teardown

Test setup and teardown are procedures performed before and after the execution of a test or a set of tests to ensure a controlled and **consistent environment for testing**.

These procedures help in preparing the test environment, performing any necessary configurations, and cleaning up after the tests are executed.



Example all test setup and teardown in a single test suite

Similar to the previous slide,
here is the code that shows how the test scenarios,
descriptions, and test setup/teardown run together.

You can see that the **test setup/teardown**
inside `.describe()` block will not affect test scenarios
outside the block.

```
tests > TS test-suite-1.spec.ts > ...
1 import { test } from '@playwright/test';
2
3 test.beforeAll(async () => {
4   // 9ms
5   console.log('Before all');
6 }
7 test.beforeEach(async () => {
8   // 6ms
9   console.log('Before each 1');
10 }
11 test("Scenario 1", async () => {
12   // scenario 1
13 }
14
15 test.describe("Describe", async () => {
16
17   test.beforeEach(async () => {
18     // 0ms
19     console.log('Before each 2');
20   }
21   test("Scenario 3", async () => {
22     // scenario 3
23   }
24
25   test("Scenario 4", async () => {
26     // scenario 4
27   }
28 }
29
30 test("Scenario 2", async () => {
31   // scenario 2
32 }
33
34 test.afterEach(async () => {
35   // 0ms
36   console.log('After each');
37 }
38
39 test.afterAll(async () => {
40   // 5ms
41   console.log('After all');
42 }
```

```
Filter (e.g. text, !exclude)
/opt/homebrew/bin/node ./node_modules/playwright/config.ts /Users/jojo/Repos/tests/test-suite-1\spec\ts --headless
1 --retries 0 --timeout 0 --workers 1
Running 4 tests using 1 worker
Before all
Before each 1
scenario 1
After each
Before each 1
Before each 2
scenario 3
After each
Before each 1
Before each 2
scenario 4
After each
Before each 1
scenario 2
After each
After all
4 passed (434ms)
```

Test suite structure

It's quite straightforward in the naming of their functions

.describe() a group of test scenarios

```
tests > ts test-suite-1.spec.ts > ...
1  import { test } from '@playwright/test';
2
3  test.beforeAll(async () => {
4    console.log("Before all");
5  });
6
7  test.beforeEach(async () => {
8    console.log("Before each 1");
9  });
10
11 test("Scenario 1", async () => {
12   console.log("scenario 1");
13 });
14
15 test.describe("Describe", async () => {
16
17   test.beforeEach(async () => {
18     console.log("Before each 2");
19   });
20
21   test("Scenario 3", async () => {
22     console.log("scenario 3");
23   });
24
25   test("Scenario 4", async () => {
26     console.log("scenario 4");
27   });
28 });
29
30 test("Scenario 2", async () => {
31   console.log("scenario 2");
32 });
33
34 test.afterEach(async () => {
35   console.log("After each");
36 });
37
38 test.afterAll(async () => {
39   console.log("After all");
40 });
41
42
```

.beforeAll() will run at the beginning of the test suite execution

.beforeEach() will run every time before each test scenario is executed

This .beforeEach() will run every time before each test scenario in it's block is executed

.afterEach() will run every time after each test scenario is executed

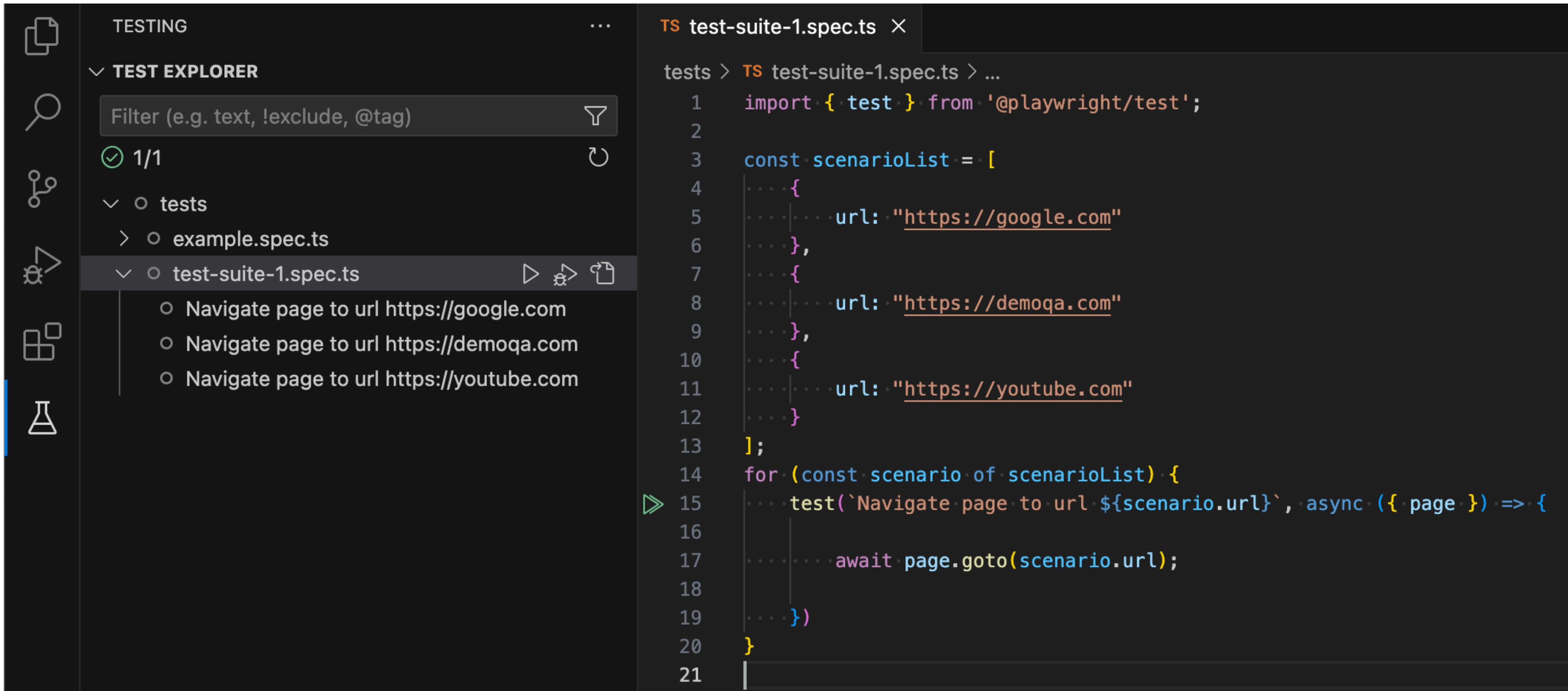
.afterAll() will run at the end of the test suite execution

```
tests > TS test-suite-1.spec.ts > ⚘ test.describe("Describe") callback > ⚘ test.afterAll
10
11  test("Scenario 1", async () => {
12  |... console.log("scenario 1");
13  });
14
15  test.describe("Describe", async () => {
16  |... test.beforeAll(async () => {
17  |...|... console.log("Before all 2");
18  |...});
19
20  |... test.beforeEach(async () => {
21  |...|... console.log("Before each 2");
22  |...});
23
24  |... test.beforeEach(async () => {
25  |...|... console.log("Before each 3");
26  |...});
27
28  |... test("Scenario 3", async () => {
29  |...|... console.log("scenario 3");
30  |...});
31
32
33  |... test("Scenario 4", async () => {
34  |...|... console.log("scenario 4");
35  |...});
36
37  |... test.afterEach(async () => {
38  |...|... console.log("After each 2");
39  |...});
40
41  |... test.afterAll(async () => {
42  |...|... console.log("After all 2");
43  |...});
44
45 });
46
47 test("Scenario 2", async () => {
48 |... console.log("scenario 2");
49 });
50
51 test.afterEach(async () => {
52 |... console.log("After each");
53 });
54
```

What if?
You have more than one test setup/teardown steps

Try add this, this, and this steps to your code
,then see what is going on

Test scenario template



The screenshot shows the Visual Studio Code interface with the following details:

- TESTING** icon in the sidebar.
- TEST EXPLORER** view:
 - Filter bar: "Filter (e.g. text, !exclude, @tag)"
 - Summary: 1/1
 - Tree view:
 - tests
 - > example.spec.ts
 - > test-suite-1.spec.ts
 - Navigate page to url https://google.com
 - Navigate page to url https://demoqa.com
 - Navigate page to url https://youtube.com
- Code Editor**:
 - File: **test-suite-1.spec.ts**
 - Content:

```
ts test-suite-1.spec.ts X

tests > ts test-suite-1.spec.ts > ...

1 import { test } from '@playwright/test';
2
3 const scenarioList = [
4   {
5     url: "https://google.com"
6   },
7   {
8     url: "https://demoqa.com"
9   },
10  {
11    url: "https://youtube.com"
12  }
13];
14 for (const scenario of scenarioList) {
15   test(`Navigate page to url ${scenario.url}`, async ({ page }) => {
16     await page.goto(scenario.url);
17   })
18 }
19 }
20 |
21 |
```

**Are you still alright?
Day 1 end at this page.**

API tests

- Chrome inspect and network
- HTTP header
- Requests
 - GET
 - POST / PUT / PATCH
 - DELETE
- JSON parse

Chrome Inspect and Network

The screenshot shows two instances of the Reqres website (reqres.in) in a browser. The left instance has a context menu open over the header area, with the 'Inspect' option highlighted and circled in orange. The right instance shows the Chrome DevTools Network tab, which is also circled in orange. The Network tab displays a list of network requests, with the first request's response body visible:

```
1 { "data": { "id": 2, "email": "janet.weaver@reqres.in", "first_name": "Janet", "last_name": "Weaver", "avatar": "https://reqres.in/img/faces/2-image.jpg" }, "support": { "url": "https://reqres.in/#support-heading", "text": "To keep BeepBeep free, contributions towards server costs are appreciated!" }}
```

Open inspect and console steps:

1. Right click on page and select "Inspect" or press "F12"
2. In the inspect tab, select "Network"

REQRES : New playground with fake data :/

Open Chrome browser
Go to this url: <https://reqres.in/>

The screenshot shows a Chrome browser window displaying the Reqres API playground at reqres.in. The left side of the interface is a sidebar listing various API endpoints categorized by method (GET, POST, PUT, PATCH, DELETE) and resource type (LIST USERS, SINGLE USER, etc.). On the right, a specific request is being made to `/api/users?page=2`, which returns a `200` status code. The response body is a JSON object representing a paginated list of users, each with properties like id, email, first_name, last_name, and avatar.

Request: /api/users?page=2

Response: 200

```
{ "page": 2, "per_page": 6, "total": 12, "total_pages": 2, "data": [ { "id": 7, "email": "michael.lawson@reqres.in", "first_name": "Michael", "last_name": "Lawson", "avatar": "https://reqres.in/img/faces/7.jpg" }, { "id": 8, "email": "lindsay.ferguson@reqres.in", "first_name": "Lindsay", "last_name": "Ferguson", "avatar": "https://reqres.in/img/faces/8.jpg" }, { "id": 9, "email": "tobias.funke@reqres.in", "first_name": "Tobias", "last_name": "Funke", "avatar": "https://reqres.in/img/faces/9.jpg" }, { "id": 10, "email": "byron.fields@reqres.in", "first_name": "Byron", "last_name": "Fields", "avatar": "https://reqres.in/img/faces/10.jpg" } ] }
```

HTTP header

The screenshot shows the Network tab in the Chrome DevTools developer console. It displays a list of network requests. The first request, labeled "users?page=2", is selected. The "Headers" tab is active, showing the following details:

- Request URL: https://reqres.in/api/users?page=2
- Request Method: GET
- Status Code: 200 OK (from disk cache)
- Remote Address: 172.67.73.173:443
- Referrer Policy: strict-origin-when-cross-origin

Below these, under "Request Headers", there is a note: "⚠ Provisional headers are shown. Disable cache to see full headers. [Learn more](#)". The headers listed are:

Header	Value
Content-Type	application/json
Referer	https://reqres.in/
Sec-Ch-Ua	"Google Chrome";v="119", "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile	?0
Sec-Ch-Ua-Platform	"macOS"
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36

```
tests > TS test-suite-1.spec.ts > ...
1   import { test, request } from '@playwright/test';
2
3   test("HTTP Request", async () => {
4
5     // Create HTTP header
6     const httpContext = await request.newContext({
7       extraHTTPHeaders: {
8         "Authorization": "" // Add your accessToken here if required
9       }
10    });
11  });
12);
13);
```

Here is the **API request header**; most of them are automatically added.

However, in some special cases, you may need to assign a key and value to the header for extras such as

- Authorization (accessToken)
- accepted file types

and so on.

Request GET

```
tests > TS test-suite-1.spec.ts > ...
1   import { test, request } from '@playwright/test';
2
3  ⚡ test("HTTP Request", async () => {
4    // Create HTTP header
5    const httpContext = await request.newContext({
6      extraHTTPHeaders: {
7        "Authorization": "" // Add your accessToken here if required
8      }
9    });
10
11   // HTTP request GET from domain https://reqres.in and path /api/users/2
12   const httpResponse = await httpContext.get("https://reqres.in/api/users/2");
13   console.log(`HTTP request GET response code: ${httpResponse.status()} ${httpResponse.statusText()}`);
14
15
16 });
17
```

Test steps:

- Line5 Create new HTTP context and add http header
- Line8 Add item name "Authorization" with empty string to header (In this case, no need any access token)
- Line13 Sent request GET with created HTTP context to url <https://reqres.in/api/users/2>
- Line14 Log out response status and text to console

PROBLEMS OUTPUT DEBUG CONSOLE TEST RESULTS TERMINAL PORTS ROBOT OUTPUT Filter (e.g. text, !exclude)

```
/opt/homebrew/bin/node ./node_modules/@playwright/test/cli.js test -c playwright.config.ts /Users/jojoee/Repositories/1\spec\ts:3 --headed --project=chromium --repeat-each 1 --retries 0 --timeout 0 --workers 1

Running 1 test using 1 worker
HTTP request GET response code: 200 OK
1 passed (550ms)
```

The screenshot shows a list of API endpoints on the left and a detailed view of a specific request on the right. The list includes:

- GET LIST USERS
- GET SINGLE USER
- GET SINGLE USER NOT FOUND
- GET LIST <RESOURCE>
- GET SINGLE <RESOURCE>
- GET SINGLE <RESOURCE> NOT FOUND
- POST CREATE
- PUT UPDATE
- PATCH UPDATE
- DELETE DELETE
- POST REGISTER - SUCCESSFUL
- POST REGISTER - UNSUCCESSFUL
- POST LOGIN - SUCCESSFUL
- POST LOGIN - UNSUCCESSFUL

The detailed view shows a request to `/api/users/23` with a status of `404` and an empty response body `{}`.

API Request

Sent GET request to this path

After the previous slide show you how to GET user info from url <https://reqres.in/api/users/2>

API Request assignment 1:
Sent request to get the non-exist user with id=23

It should response you with status **404 Not Found**

Request POST

```
tests > TS test-suite-1.spec.ts > ...
1  import { test, request } from '@playwright/test';
2
3  test("HTTP Request", async () => {
4
5    // Create HTTP header
6    const httpContext = await request.newContext({
7      extraHTTPHeaders: {
8        "Authorization": "" // Add your accessToken here if required
9      }
10     });
11
12    // HTTP request GET from domain https://reqres.in and path /api/users
13    const httpResponse = await httpContext.post("https://reqres.in/api/users", {
14      data: {
15        "name": "morpheus",
16        "job": "leader"
17      }
18    });
19    console.log(`HTTP request POST response code: ${httpResponse.status}`);
20
21  });
22
```

Test steps:

- Line5 Create new HTTP context and add http header
- Line8 Add item name "Authorization" with empty string to header (In this case, no need any access token)
- Line13 Sent request POST with created HTTP context to url <https://reqres.in/api/users>
- Line14 Add data payload to request POST
- Line19 Log out response status and text to console

PROBLEMS OUTPUT DEBUG CONSOLE TEST RESULTS TERMINAL PORTS RO
/opt/homebrew/bin/node ./node_modules/@playwright/test/cli.js test -c pl
1\\.spec\\.ts:4 --headed --project=chromium --repeat-each 1 --retries 0 --t

Running 1 test using 1 worker
HTTP request POST response code: 201 Created
1 passed (1.1s)

Request PUT / PATCH

```
tests > TS test-suite-1.spec.ts > ...
1  import { test, request } from '@playwright/test';
2
3  test("HTTP Request", async () => {
4
5    // Create HTTP header
6    const httpContext = await request.newContext({ - 1ms
7    extraHTTPHeaders: {
8      "Authorization": "" // Add your accessToken here if required
9    }
10   });
11
12  // HTTP request GET from domain https://reqres.in and path /api/users/2
13  const httpResponse = await httpContext.put("https://reqres.in/api/users/2", { - 847ms
14  data: {
15    name: "morpheus",
16    job: "leader update"
17  }
18  );
19  console.log(`HTTP request PUT response code: ${httpResponse.status}`);
20
21 });
22 |
```

Test steps:

- Line5 Create new HTTP context and add http header
- Line8 Add item name "Authorization" with empty string to header (In this case, no need any access token)
- Line13 Sent request PUT with created HTTP context to url <https://reqres.in/api/users/2>
- Line14 Add data payload to request POST
- Line19 Log out response status and text to console

PROBLEMS OUTPUT DEBUG CONSOLE TEST RESULTS TERMINAL PORTS RO

```
/opt/homebrew/bin/node ./node_modules/@playwright/test/cli.js test -c pl
1\spec\ts:3 --headed --project=chromium --repeat-each 1 --retries 0 --
Running 1 test using 1 worker
HTTP request PUT response code: 200 OK
1 passed (1.2s)
```

The screenshot shows a list of API endpoints on the left and a detailed view of a PATCH request on the right. The endpoints include:

- GET /api/users LIST USERS
- GET /api/users/{id} SINGLE USER
- GET /api/users/{id} NOT FOUND
- GET /api/users/{resource} LIST <RESOURCE>
- GET /api/{resource} SINGLE <RESOURCE>
- GET /api/{resource}/{id} NOT FOUND
- POST /api/users CREATE
- PUT /api/users UPDATE
- PATCH /api/users UPDATE
- DELETE /api/users DELETE
- POST /api/register REGISTER - SUCCESSFUL
- POST /api/register REGISTER - UNSUCCESSFUL
- POST /api/login LOGIN - SUCCESSFUL
- POST /api/login LOGIN - UNSUCCESSFUL

The detailed view on the right shows a Request to `/api/users/2` with a Response status of 200 OK. The response body is a JSON object:

```
{  
  "name": "morpheus",  
  "job": "zion resident"  
}
```

Guess? How PATCH use Sent PATCH request to this path

After the previous slide show you how to PATCH user info from url <https://reqres.in/api/users/2>

API Request assignment 2:
Sent request to patch the user with id=2

It should response you with status 200 OK

Request DELETE

```
tests > TS test-suite-1.spec.ts > ...
1  import { test, request } from '@playwright/test';
2
3  test("HTTP Request", async () => {
4
5    // Create HTTP header
6    const httpContext = await request.newContext({ - 2ms
7    extraHTTPHeaders: {
8      "Authorization": "" // Add your accessToken here if required
9    }
10   });
11
12   // HTTP request GET from domain https://reqres.in and path /api/users/2
13   const httpResponse = await httpContext.delete("https://reqres.in/api/users/2"); - 808ms
14   console.log(`HTTP request DELETE response code: ${httpResponse.status()} ${httpResponse.statusText()}`);
15
16 });
17
```

Test steps:

- Line5 Create new HTTP context and add http header
- Line8 Add item name "Authorization" with empty string to header (In this case, no need any access token)
- Line13 Sent request DELETE with created HTTP context to url <https://reqres.in/api/users/2>
- Line14 Log out response status and text to console

PROBLEMS OUTPUT DEBUG CONSOLE TEST RESULTS TERMINAL PORTS ROBOT OUTPUT

Filter (e.g. text, !exclude)

```
/opt/homebrew/bin/node ./node_modules/@playwright/test/cli.js test -c playwright.config.ts /Users/jojo/Repositories/1\spec\ts:3 --headed --project=chromium --repeat-each 1 --retries 0 --timeout 0 --workers 1
```

```
Running 1 test using 1 worker
HTTP request DELETE response code: 204 No Content
1 passed (1.2s)
```

JSON parse

Test steps:

Line13 After receive the response of HTTP
GET

Line14 Log out response status and text to
console

Line16 Convert the response body as a
string to JSON form

Line17 Log out the response data

```
tests > TS test-suite-1.spec.ts > ...
1 import { test, request } from '@playwright/test';
2
3 test("HTTP Request", async () => {
4
5     // Create HTTP header
6     const httpContext = await request.newContext({ - 1ms
7     extraHTTPHeaders: {
8         "Authorization": "" // Add your accessToken here if required
9     }
10 });
11
12 // HTTP request GET from domain https://reqres.in and path /api/users/2
13 const httpResponse = await httpContext.get("https://reqres.in/api/users/2"); - 157ms
14 console.log(`HTTP request GET response code: ${httpResponse.status()} ${httpResponse.statusText()}`);
15
16 const data = JSON.parse(await httpResponse.body().toString()); - 1ms
17 console.log("Response data is :", data.data);
18
19 });
20
```

The screenshot shows a browser developer tools interface with several panels:

- Network Tab:** Shows a list of network requests. One request, labeled '2', is selected and expanded. Its response body is displayed in red text:

```
1 {
2     "data": {
3         "id": 2,
4         "email": "janet.weaver@reqres.in",
5         "first_name": "Janet",
6         "last_name": "Weaver",
7         "avatar": "https://reqres.in/img/faces/2-image.jpg"
8     },
9     "support": {
10         "url": "https://reqres.in/#support-heading",
11         "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
12     }
13 }
```
- DEBUG CONSOLE Tab:** Shows the command run in the terminal: `/opt/homebrew/bin/node ./node_modules/@playwright/test/cli.js test -c playwright.config.ts:3 --headed --project=chromium --repeat=1 --retries=0 --timeout=0 --workers=1`. It also displays the test results:

```
Running 1 test using 1 worker
HTTP request PUT response code: 200 OK
Response data is : {"id": 2, "email": "janet.weaver@reqres.in", "first_name": "Janet", "last_name": "Weaver", "avatar": "https://reqres.in/img/faces/2-image.jpg"}
arg1: {"id": 2, "email": "janet.weaver@reqres.in", "first_name": "Janet", "last_name": "Weaver", "avatar": "https://reqres.in/img/faces/2-image.jpg", "email": "janet.weaver@reqres.in", "first_name": "Janet", "id": 2, "last_name": "Weaver", > [[Prototype]]: Object}
```
- PROBLEMS Tab:** Shows a single error message: `1 passed (21.7s)`.

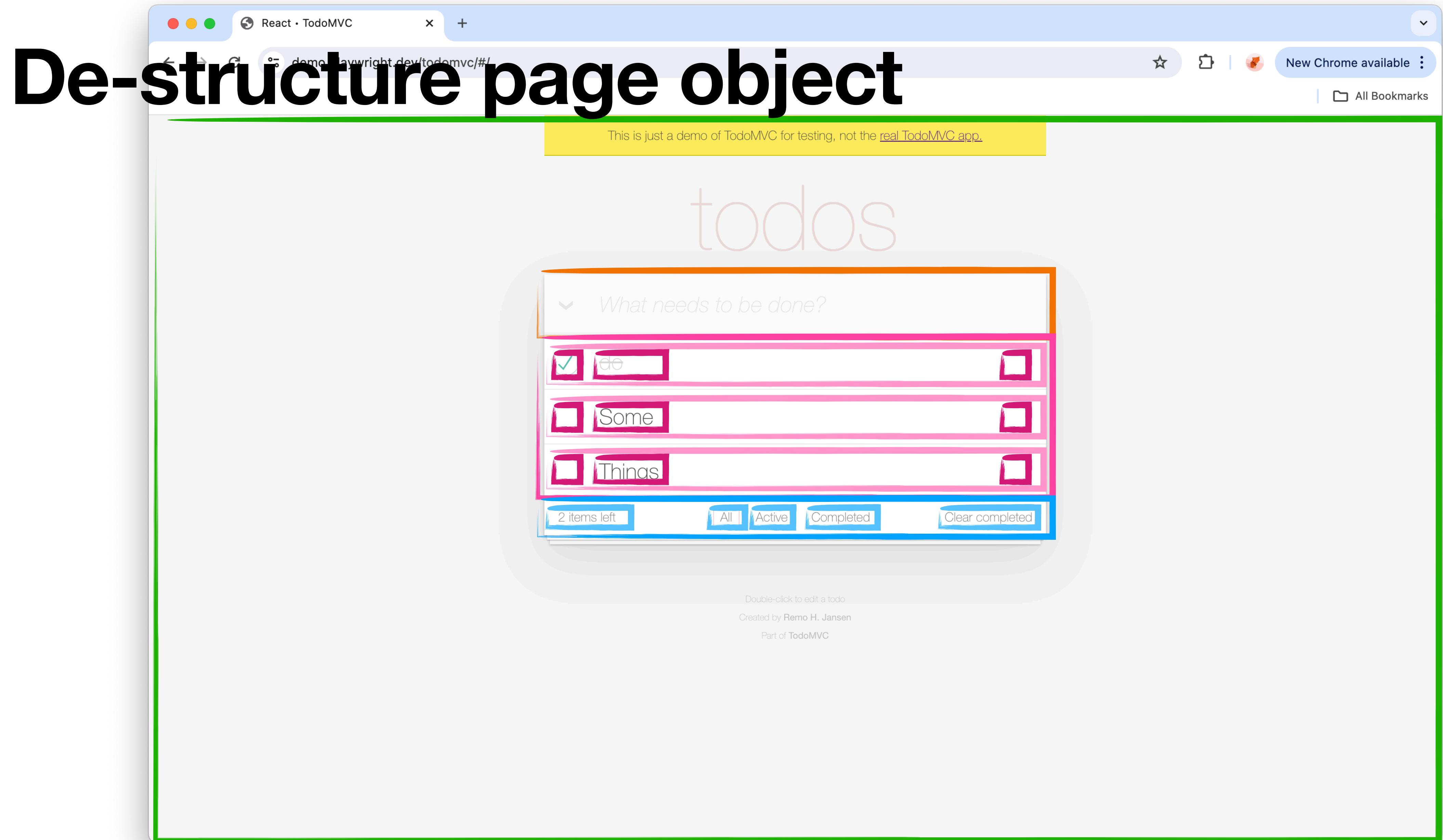
Page object model

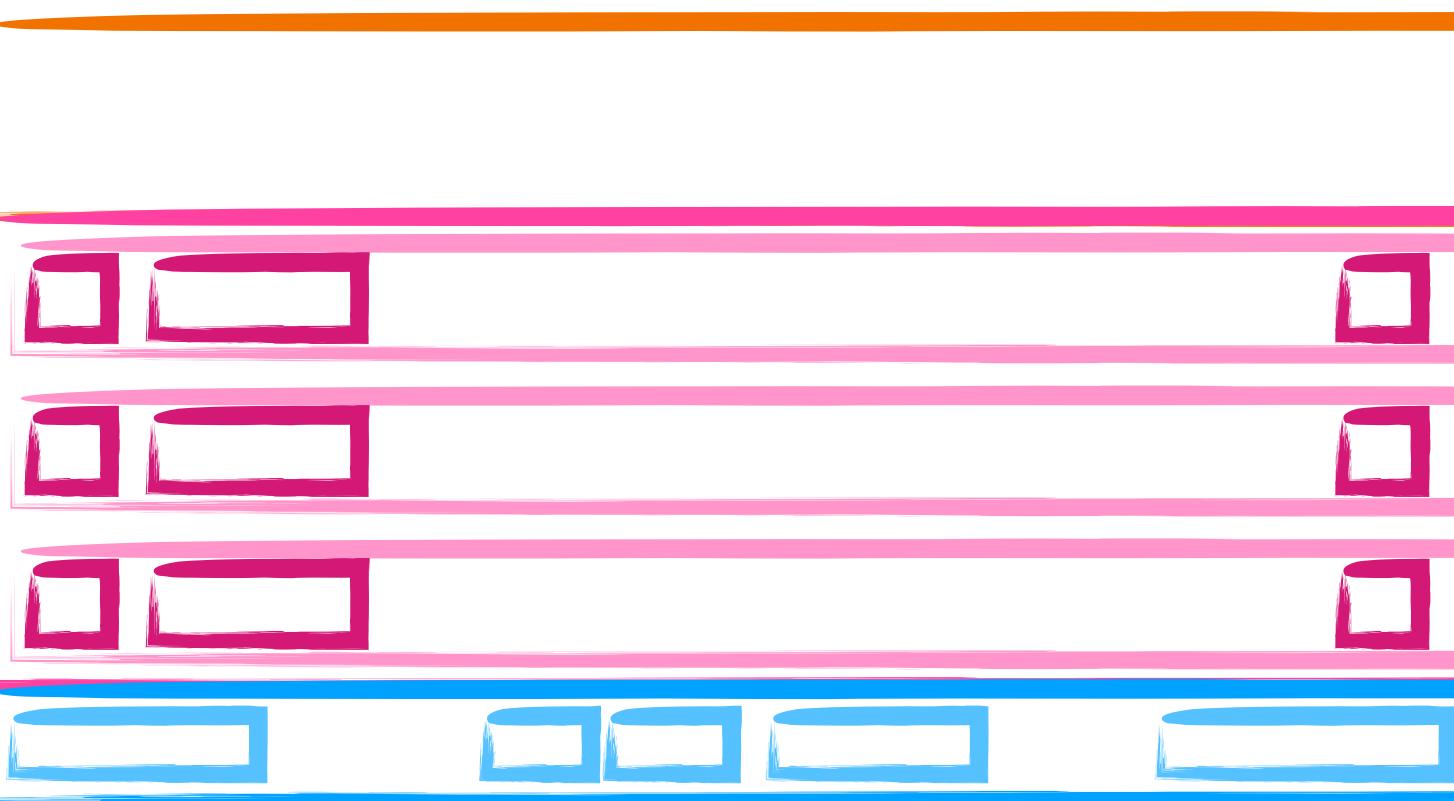
- What is page object?
- De-structure
- Create page object

What is page object?

The Page Object Model (POM) is a design pattern commonly used in test automation to enhance the maintainability and readability of test scripts. While the Page Object Model is not specific to any particular testing framework or tool, it can be applied to various technologies and platforms, including Playwright.

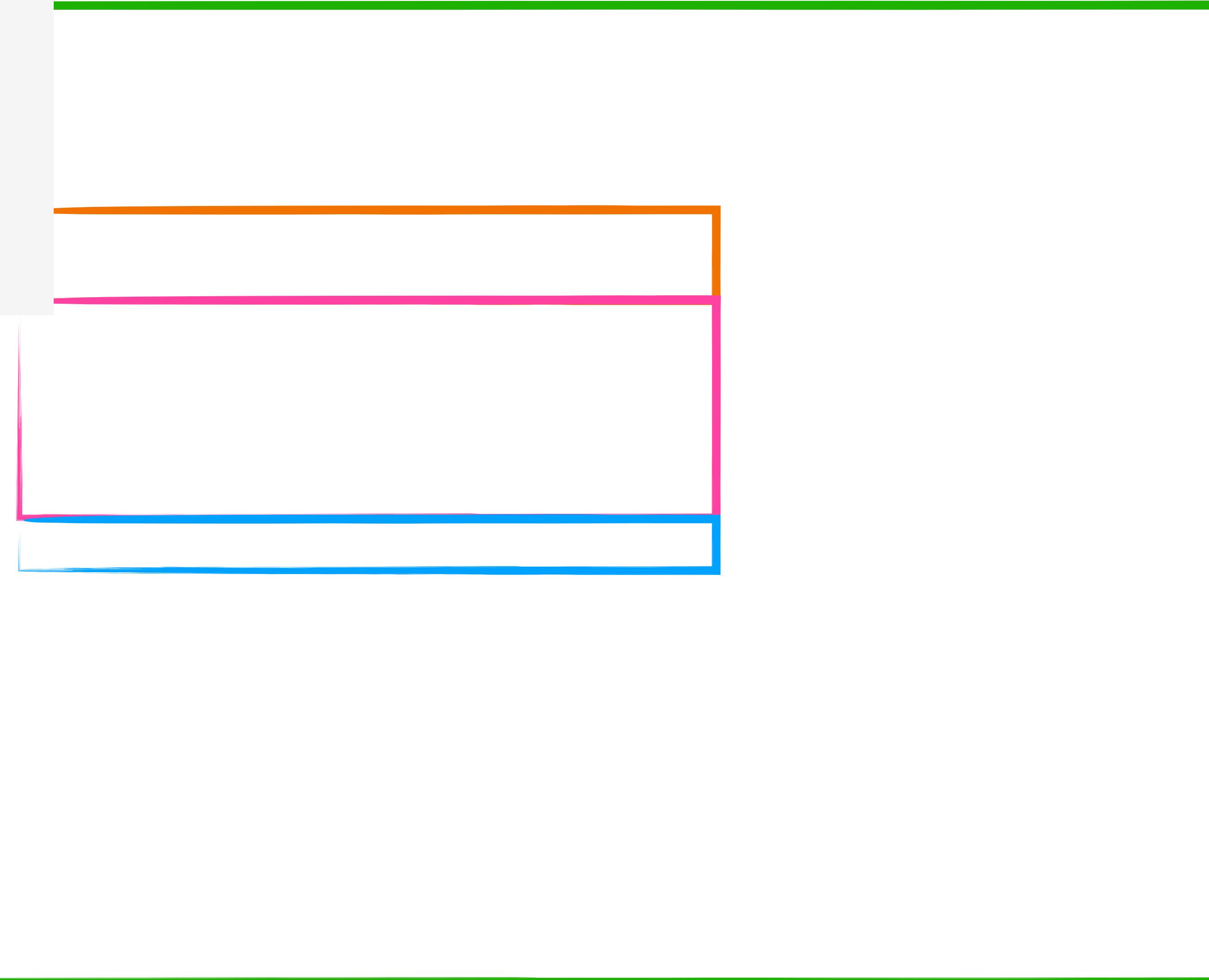
In the context of Playwright, the idea of the Page Object Model involves organizing your test automation code in a way that separates the abstraction of web pages from the actual test logic. This helps to achieve better code organization, reusability, and maintainability.





Page contains 3 objects

- Todo input field
- Todo list
- Filter



Playwright - Example code

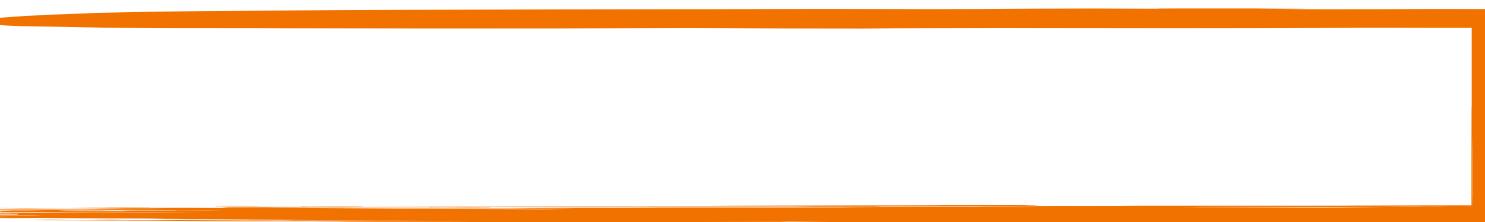
```
class TodoPage {
    private readonly _page: Page;

    public input: InputComponent; 
    public todoList: TodoList; 
    public filter: Filter; 
```

Page contains 3 objects

- Todo input field
- Todo list
- Filter

Todo input field



Todo input field

```
class InputComponent {
    private readonly _page: Page;
    private readonly _element: Locator;

    constructor(page: Page, element: Locator) {
        this._page = page;
        this._element = element;
    }

    public async inputTodo(todo: string) {
        await this._element.fill(todo);
        await this._element.press('Enter');
    }
}
```

Todo list contains

- Todo items



Playwright - Example code

```
class TodoList {  
    private readonly _page: Page;  
    private readonly _element: Locator;  
  
    private readonly _todoItemLocator: Locator; ← Todo list contains  
    • Todo items  
  
    constructor(page: Page, element: Locator) {  
        this._page = page;  
        this._element = element;  
        this._todoItemLocator = this._element.getByTestId('todo-item');  
    }  
  
    public async getTodoCount() {  
        const count = await this._todoItemLocator.count();  
        return count;  
    }  
  
    public async getAllTodos() {  
        const count = await this.getTodoCount();  
        const todoItems = new Array<TodoItem>();  
        for (let index = 0; index < count; index++) {  
            todoItems.push(new TodoItem(this._page, this._todoItemLocator.nth(index)));  
        }  
        return todoItems;  
    }  
}
```

...

Playwright - Example code

```
...  
  
public async completeTodoByTitle(completeTitle: string) {  
    const todos = await this.getAllTodos();  
    for await (const todo of todos) {  
        const title = await todo.getTitle();  
        if (title == completeTitle) {  
            await todo.checkedTodo();  
            break;  
        }  
    }  
  
    public async removeTodoByTitle(removeTitle: string) {  
        const todos = await this.getAllTodos();  
        for await (const todo of todos) {  
            const title = await todo.getTitle();  
            if (title == removeTitle) {  
                await todo.removeTodo();  
                break;  
            }  
        }  
    }  
}
```

Todo item contains

- **Checkbox**
- **Text**
- **Remove button**



Playwright - Example code

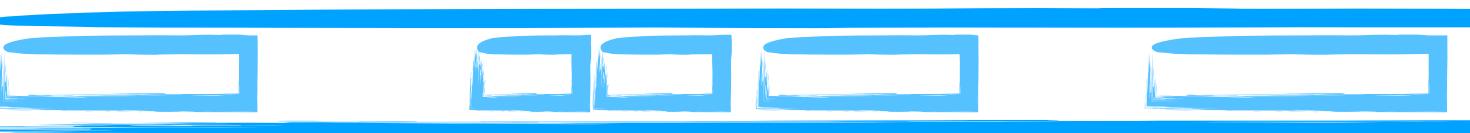
```
class TodoItem {  
    private readonly _page: Page;  
    private readonly _element: Locator;  
  
    private readonly _checkboxLocator: Locator; ←  
    private readonly _textLocator: Locator; ←  
    private readonly _removeBtnLocator: Locator; ←  
  
    constructor(page: Page, element: Locator) {  
        this._page = page;  
        this._element = element;  
        this._checkboxLocator = this._element.locator('css=input.toggle');  
        this._textLocator = this._element.getByTestId('todo-title');  
        this._removeBtnLocator = this._element.locator('css=button.destroy');  
    }  
  
    public async getTitle() {  
        const text = await this._textLocator.innerText();  
        return text;  
    }  
  
    public async checkedTodo() {  
        await this._checkboxLocator.click();  
    }  
  
    public async removeTodo() {  
        await this._textLocator.hover();  
        await this._removeBtnLocator.click();  
    }  
}
```

Todo item contains

- **Checkbox**
- **Text**
- **Remove button**

Filter contains

- Todo count
- Filter all
- Filter active
- Filter completed
- Clear completed



Playwright - Example code

```
class Filter {
    private readonly _page: Page;
    private readonly _element: Locator;

    constructor(page: Page, element: Locator) {
        this._page = page;
        this._element = element;
    }

    // YOUR TURN IN WORKSHOP
}
```

Playwright - Example code

```
test('Example', async ({ page }) => {
    // Arrange
    await page.goto('https://demo.playwright.dev/todomvc/#/');

    const todoPage = new TodoPage(page);
    await todoPage.input.inputTodo('Do');
    await todoPage.input.inputTodo('Some');
    await todoPage.input.inputTodo('Things');

    // Action
    await todoPage.todoList.completeTodoByTitle('Some');
    await todoPage.todoList.removeTodoByTitle('Do');

    // Assertion
    ...
});
```

Scraping data

- Get current page url
- Get text from web
- Get data in table

Scraping from table

It not too hard.. likely you get data from todo list in the previous session.

The screenshot shows a web browser window with the URL w3schools.com/bootstrap5/bootstrap_tables.php. The page content is as follows:

Basic Table

A basic Bootstrap 5 table has a light padding and horizontal dividers.

The `.table` class adds basic styling to a table:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

[Try it Yourself »](#)

Striped Rows

The `.table-striped` class adds zebra-stripes to a table:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

De-structure page object

The screenshot shows a web browser window displaying the [w3schools.com/html/html_tables.asp](https://www.w3schools.com/html/html_tables.asp) page. The page title is "HTML Tables". On the left, there's a sidebar with a "HTML Tutorial" section containing links to various HTML topics. The main content area features a heading "HTML Tables" with "Previous" and "Next" navigation buttons. Below the heading, a text block states: "HTML tables allow web developers to arrange data into rows and columns." A large "Example" section contains a table with the following data:

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

At the bottom of the page, there's a "Try it Yourself..." button.

Page contains Table

Table contains Body

Body contains Rows

**Row contains Company
Contact and Country**

Playwright - Example code

```
class Table {
    private readonly _page: Page;

    public rowLocator: Locator;

    constructor(page: Page) {
        this._page = page;
        this.rowLocator = this._page.locator(
            'xpath=//table[@id="customers"]/tbody/tr[position()>1]'
        );
    }

    public async getCount() {
        const count = await this.rowLocator.count();
        return count;
    }

    public async getAllRows() {
        const count = await this.getCount();
        const rows = new Array<Row>();
        for (let index = 0; index < count; index++) {
            rows.push(new Row(this._page, this.rowLocator.nth(index)));
        }
        return rows;
    }
}
```

...

Playwright - Example code

```
...  
  
public async getRowInfo() {  
    const rows = await this.getAllRows();  
    const infos = new Array<Info>();  
    for await (const row of rows) {  
        const info = new Info();  
        info.company = await row.getCompany();  
        info.contact = await row.getContact();  
        info.country = await row.getCountry();  
        infos.push(info);  
    }  
    return infos;  
}  
}
```

Playwright - Example code

```
class Row {
    private readonly _page: Page;
    private readonly _element: Locator;

    public companyLocator: Locator;
    public contactLocator: Locator;
    public countryLocator: Locator;

    constructor(page: Page, element: Locator) {
        this._page = page;
        this._element = element;
        this.companyLocator = this._element.locator('xpath=//td[1]');
        this.contactLocator = this._element.locator('xpath=//td[2]');
        this.countryLocator = this._element.locator('xpath=//td[3]');
    }

    public async getCompany() {
        const text = await this.companyLocator.innerText();
        return text;
    }

    public async getContact() {
        const text = await this.contactLocator.innerText();
        return text;
    }

    public async getCountry() {
        const text = await this.countryLocator.innerText();
        return text;
    }
}
```

Playwright - Example code

```
class Info {
    public company: string;
    public contact: string;
    public country: string;
}
```

Playwright - Example code

```
test('Scraping data from table', async ({ page }) => {
  // Arrange
  const url = "https://www.w3schools.com/html/html_tables.asp";
  await page.goto(url);

  const table = new Table(page);
  const infos = await table.getRowInfo();

  console.table(infos);
});
```

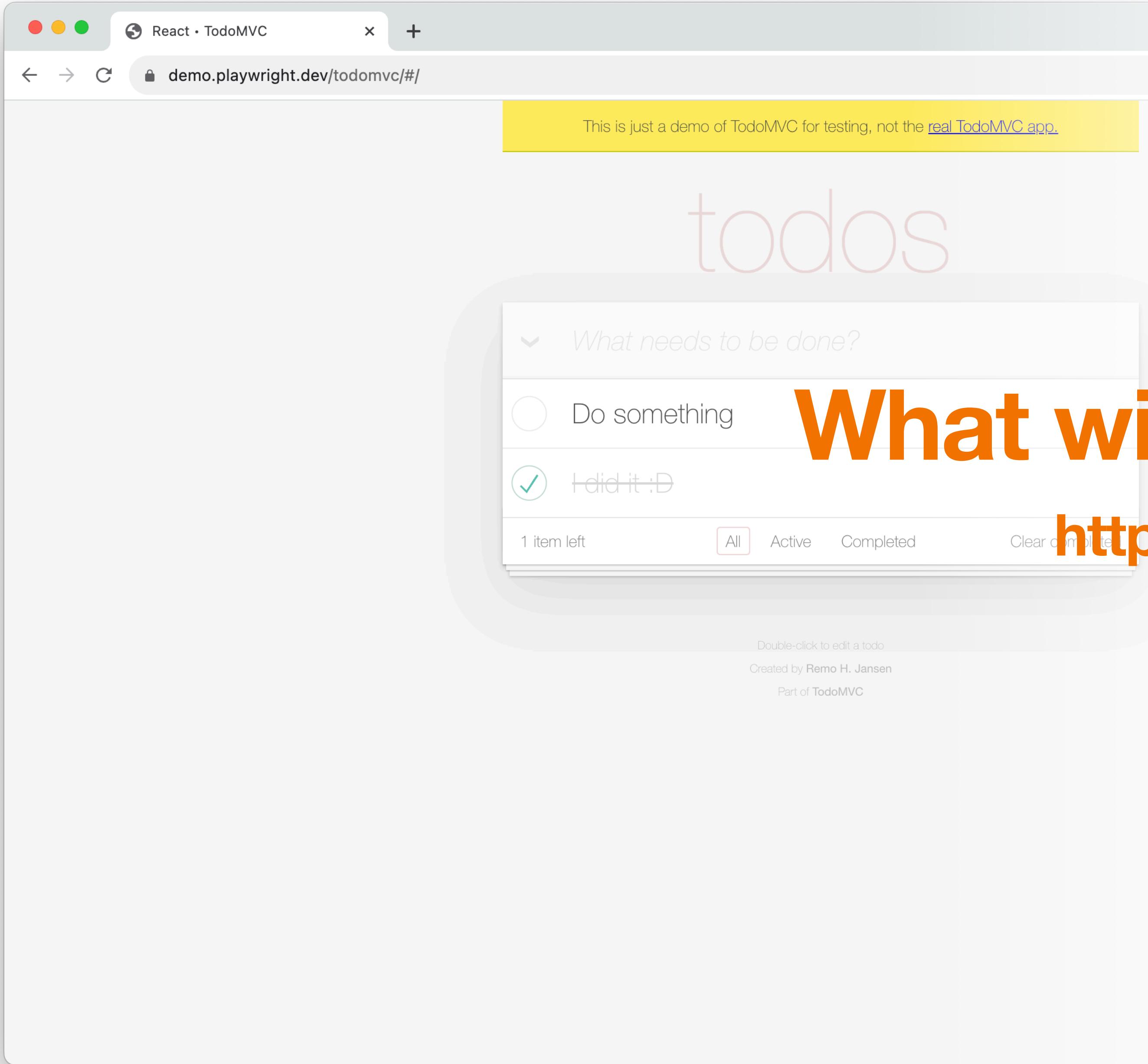
Console output

(index)	company	contact	country
0	'Alfreds Futterkiste'	'Maria Anders'	'Germany'
1	'Centro comercial Moctezuma'	'Francisco Chang'	'Mexico'
2	'Ernst Handel'	'Roland Mendel'	'Austria'
3	'Island Trading'	'Helen Bennett'	'UK'
4	'Laughing Bacchus Winecellars'	'Yoshi Tannamuri'	'Canada'
5	'Magazzini Alimentari Riuniti'	'Giovanni Rovelli'	'Italy'



Don't be afraid
Now we are friend

Playwright Workshop



**As a QA,
What will you do with this site
<https://demo.playwright.dev/todomvc/#/>**

Now it's workshop time

Do anything you want :)

- Go to this
<https://demo.playwright.dev/todomvc/#/>
- Click it
- Fill it
- Check it
- Bla bla bla ~