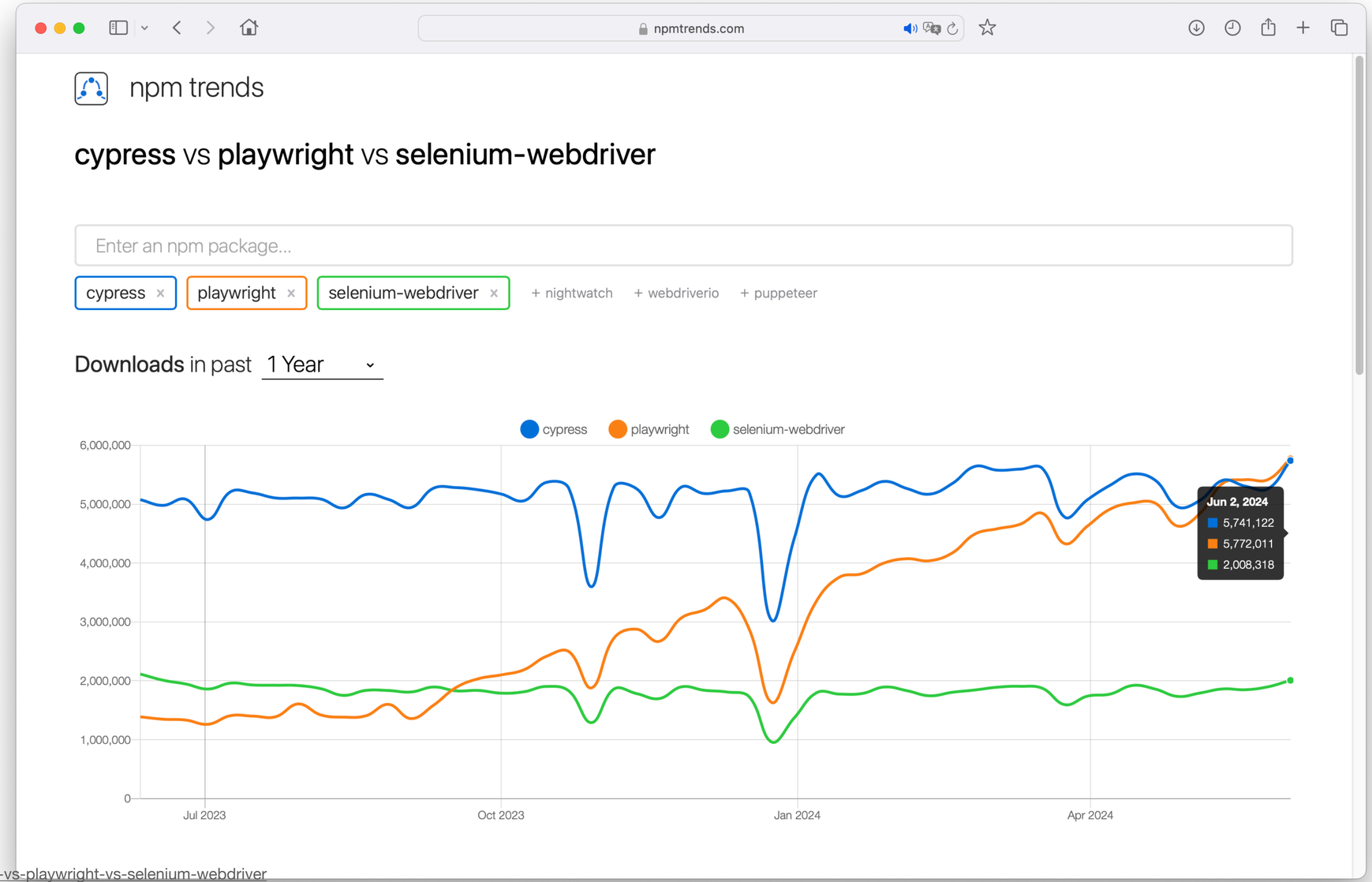
Playwright

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



Ref1: https://npmtrends.com/cypress-vs-playwright-vs-selenium-webdriver
Ref2: https://www.youtube.com/live/qkwIglz3KZc?si=nrWhDngLmUgIHlVc

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

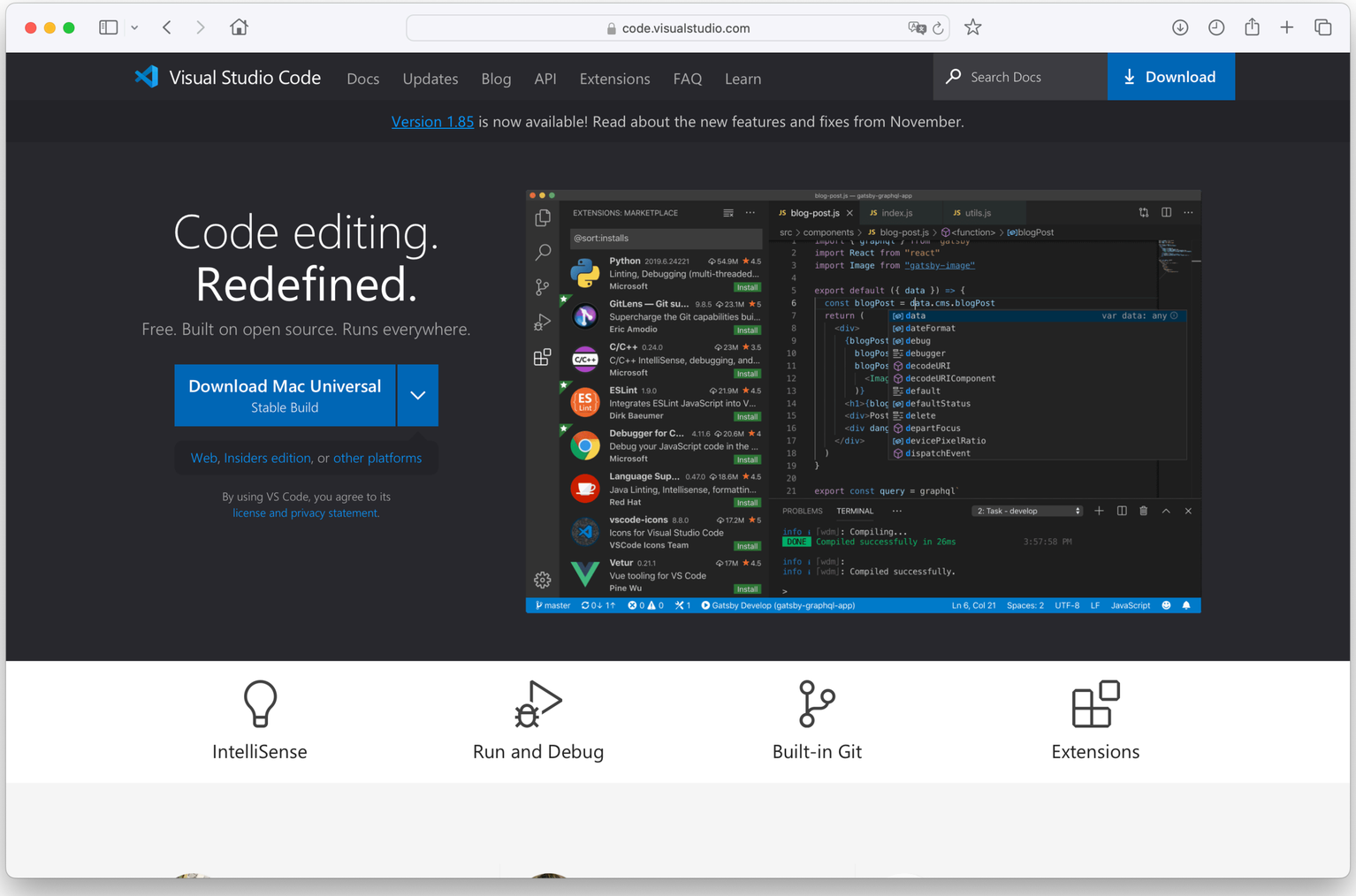
Ref1: "Robot Framework Test Automation" by Sumit Bisht and Manoj Kumar
Ref2: "Mastering Selenium Robot Framework" by Marko Kocic
Ref3: "Mastering Playwright" by Eran Peer
Ref4: "Cypress.io Practical Automation Testing For Everyone" by Filip Hric

Installations

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

Install VSCode

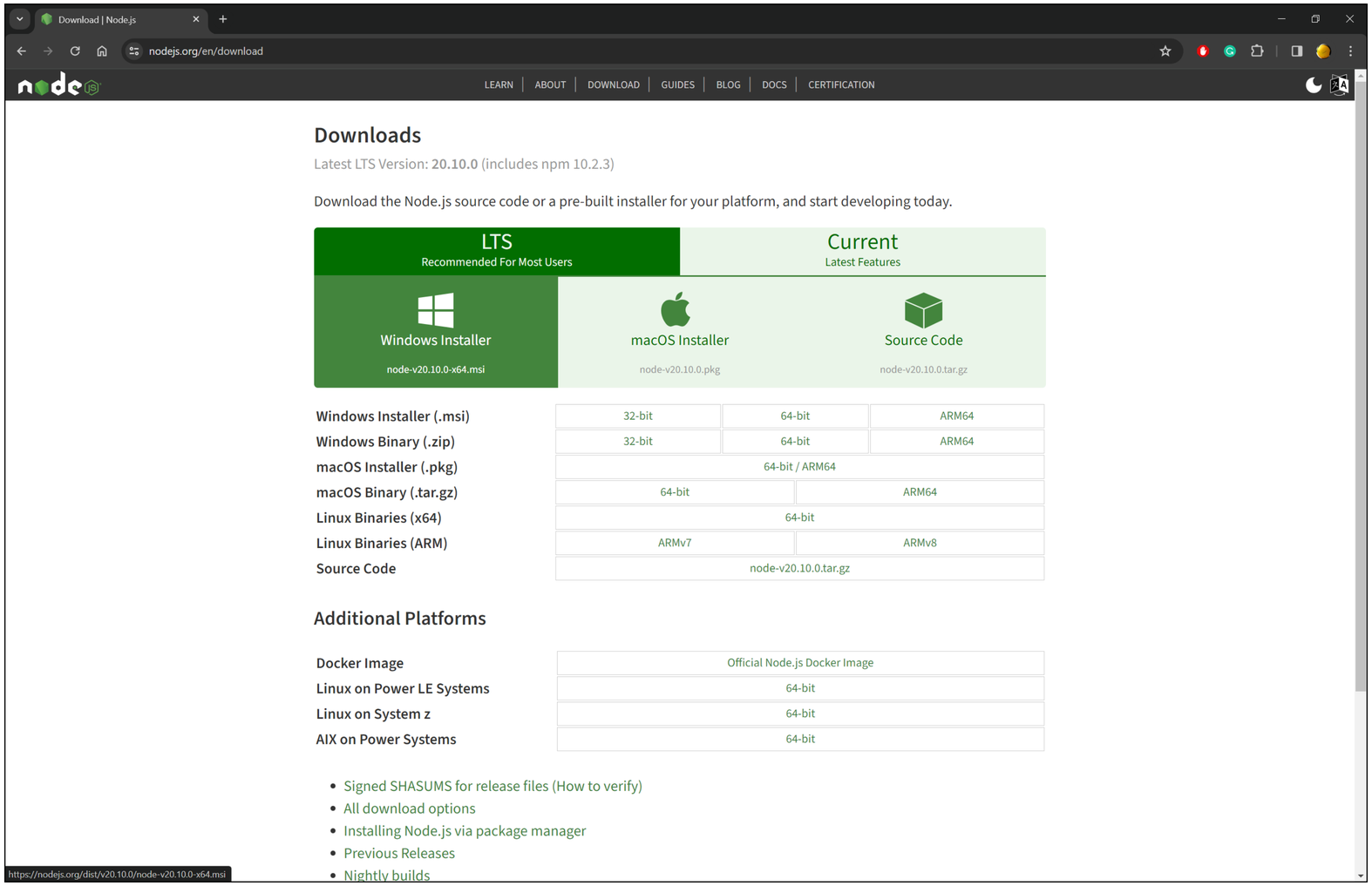
Go to this official website => https://code.visualstudio.com/
It has already suggested the compatible version with your environment



Ref1: https://code.visualstudio.com/

Windows: Download Node.js

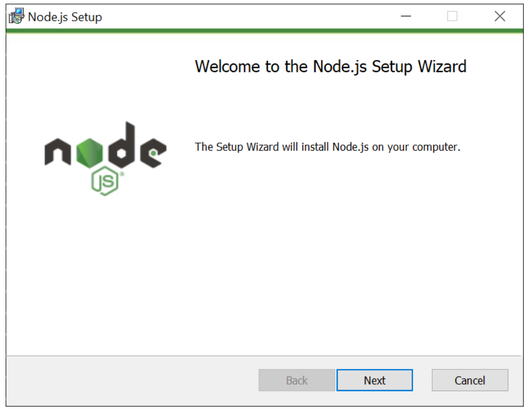
Go to this official website => https://nodejs.org/en/download , then download the Windows version

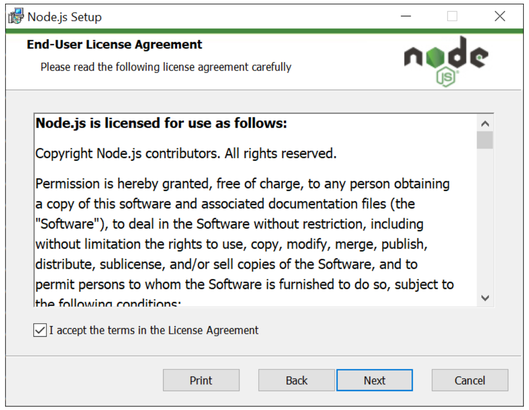


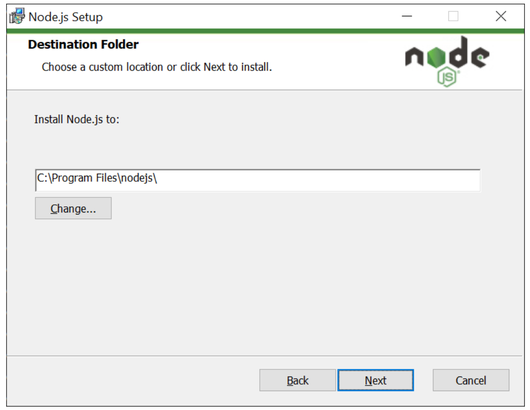
Ref1: https://nodejs.org/en/download

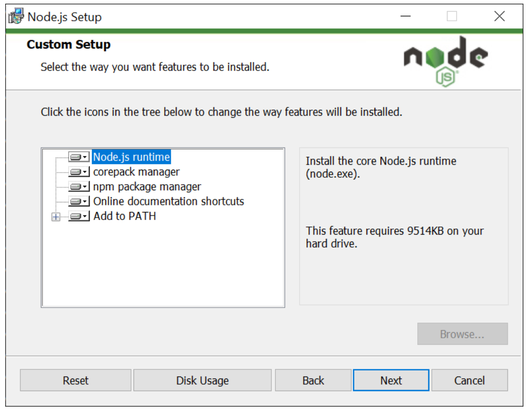
Windows: Install Node.js

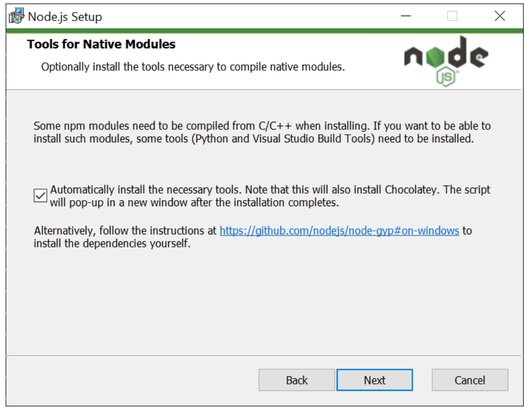
So easy! Next, Next, Next, and Finish.

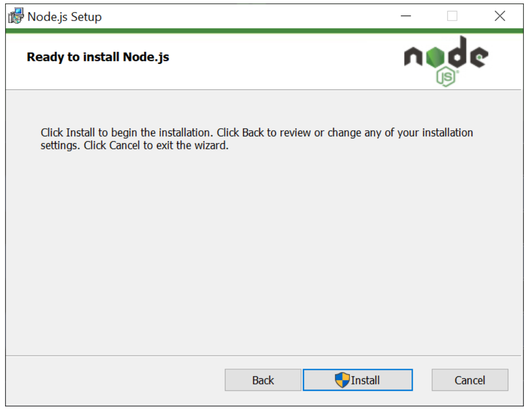


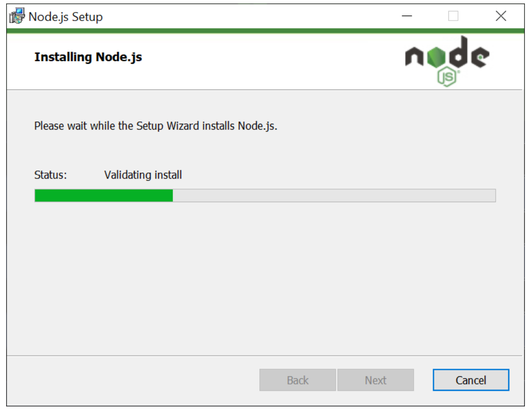


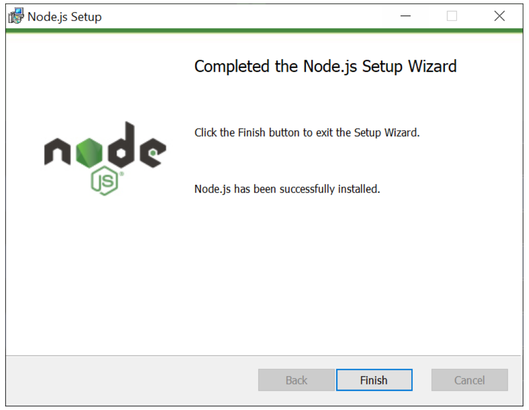






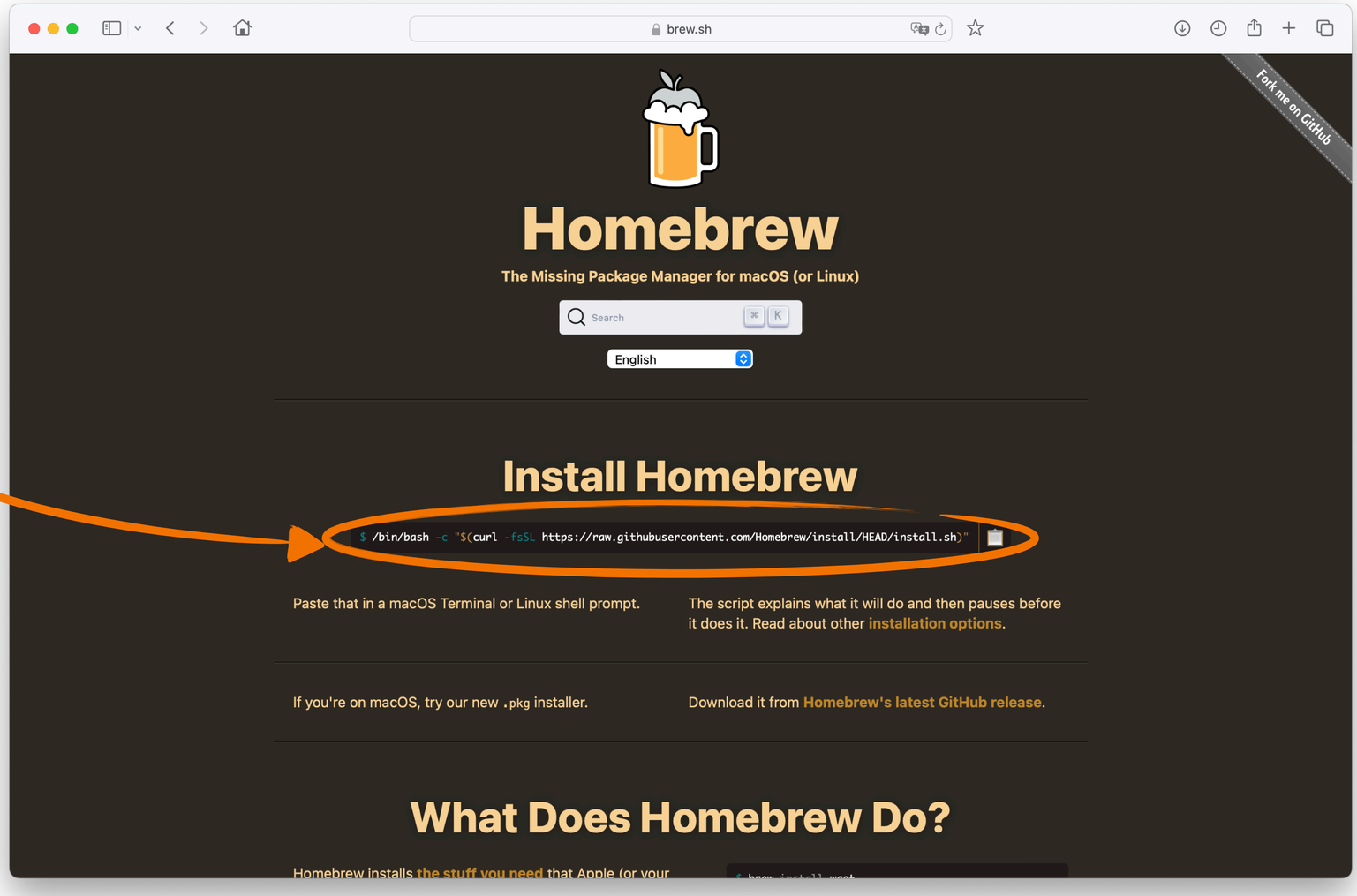






Mac: Homebrew website

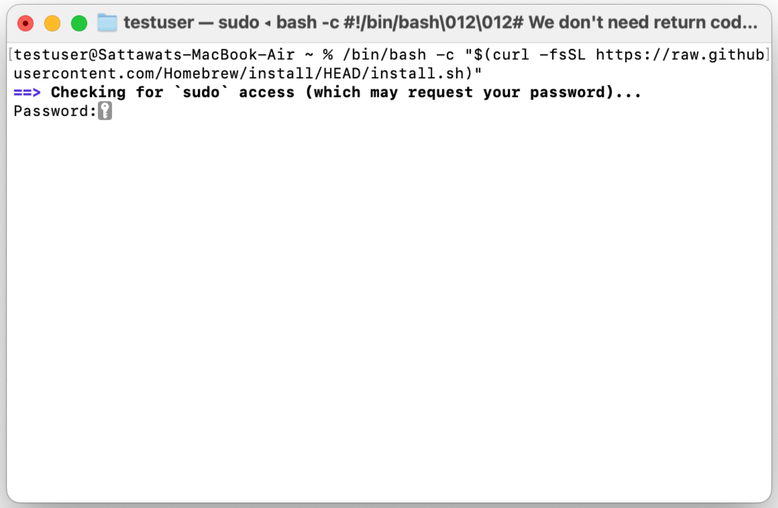
At first, it might be a bit challenging, but your life will become more comfortable with Homebrew
Copy that command and paste it to your terminal

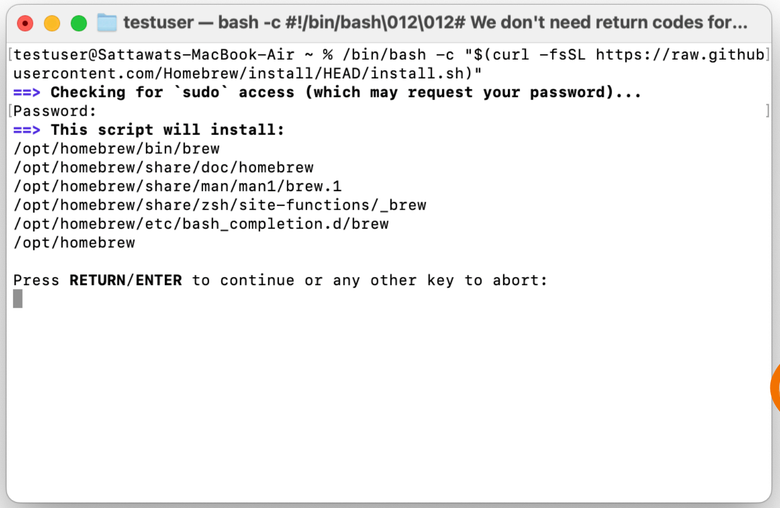


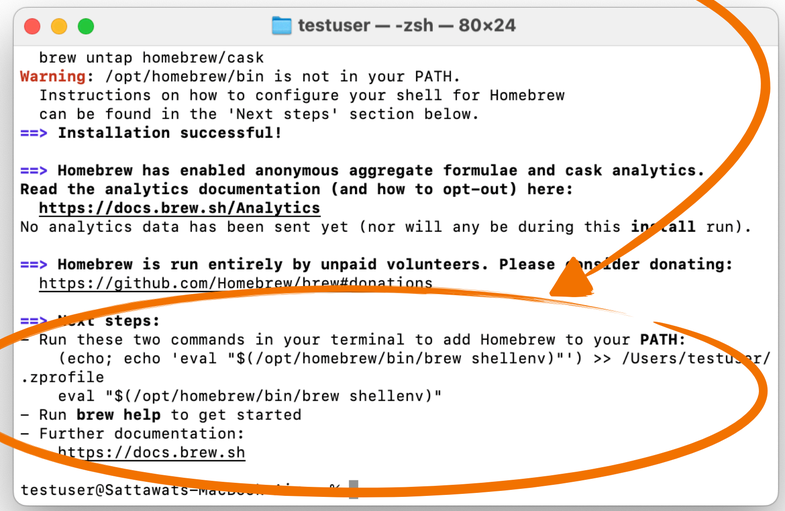
Mac: Install Homebrew

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

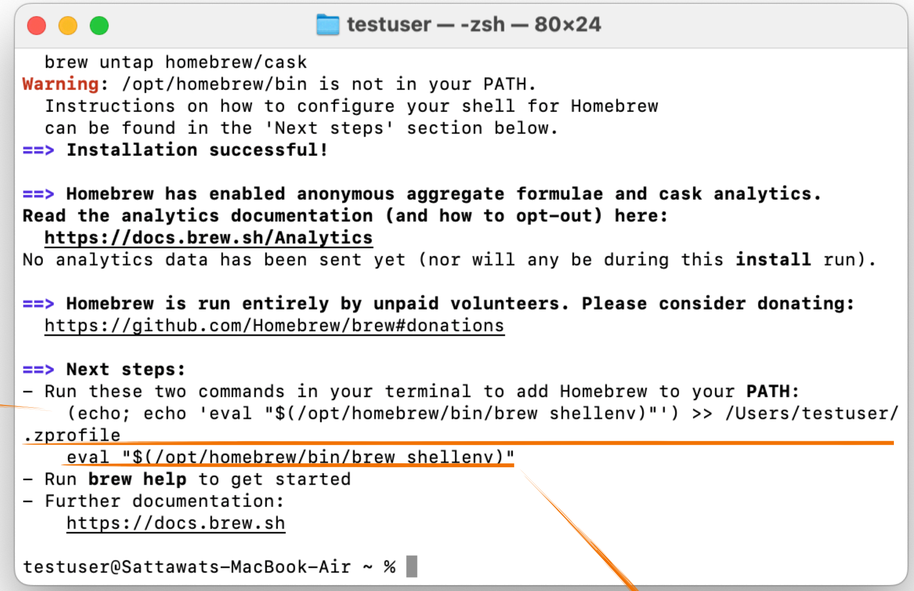


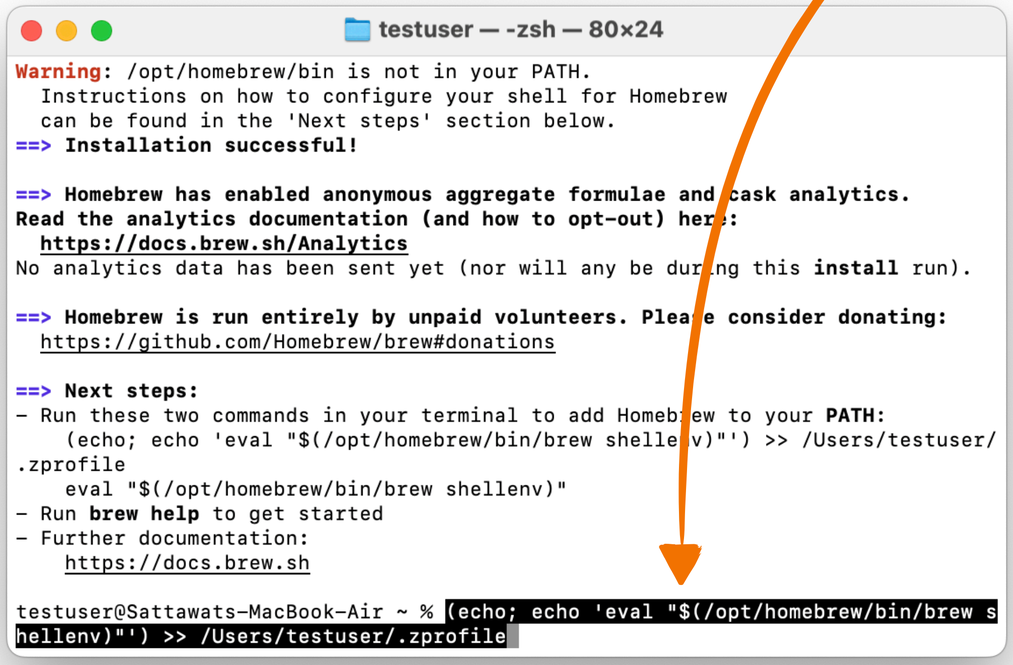


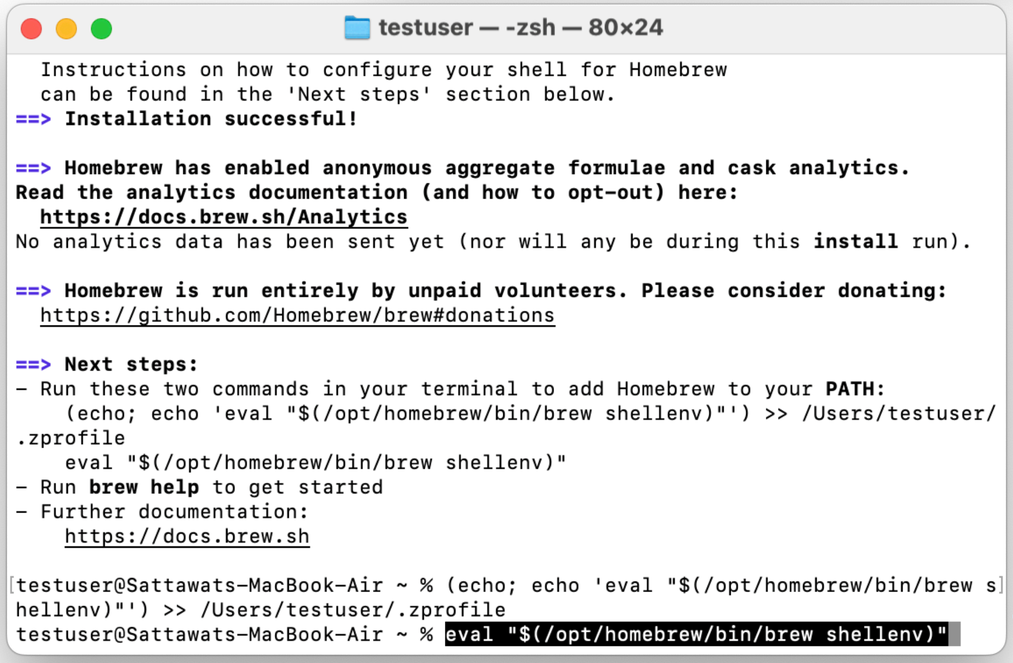


Mac: Set Homebrew path

Copy that provided 2 lines, and paste to terminal





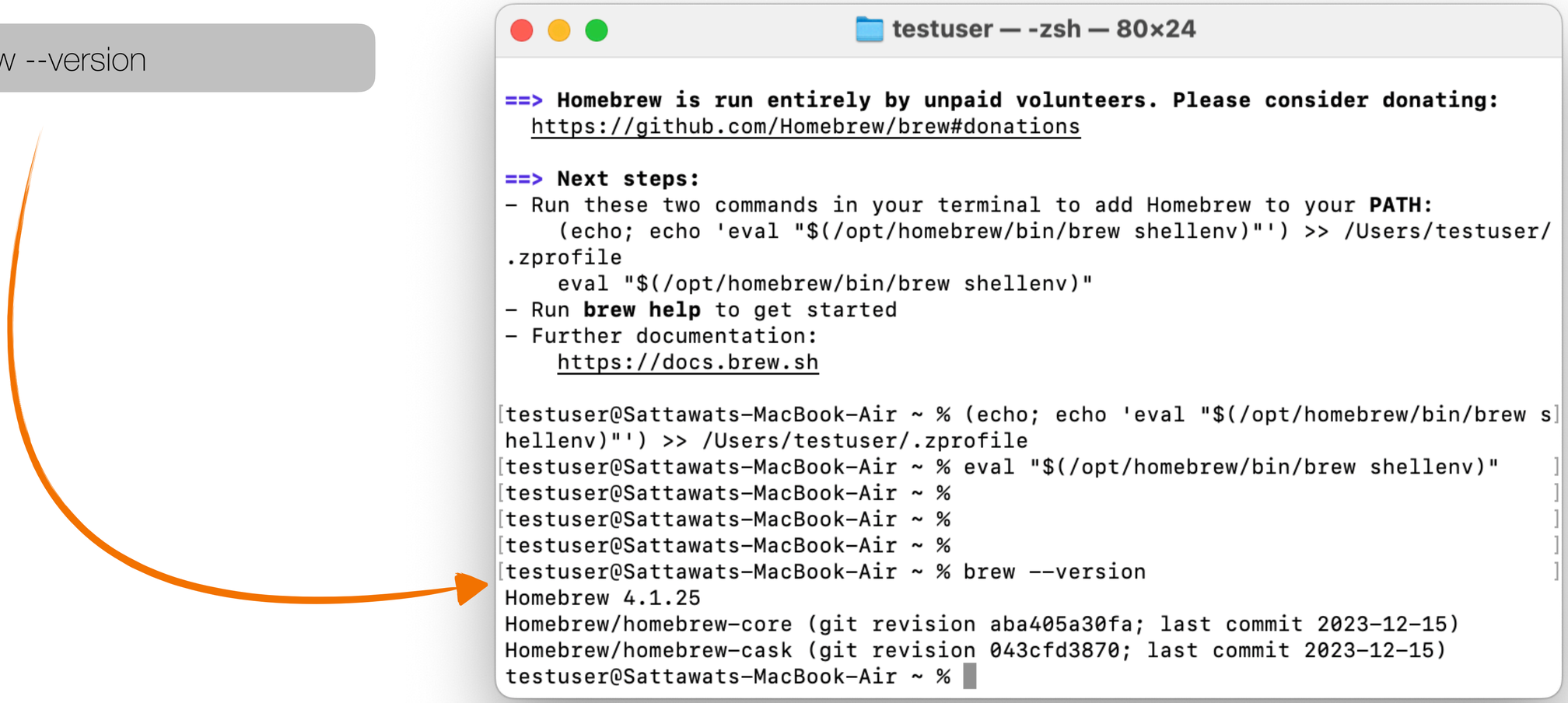


Mac: Check Homebrew version

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

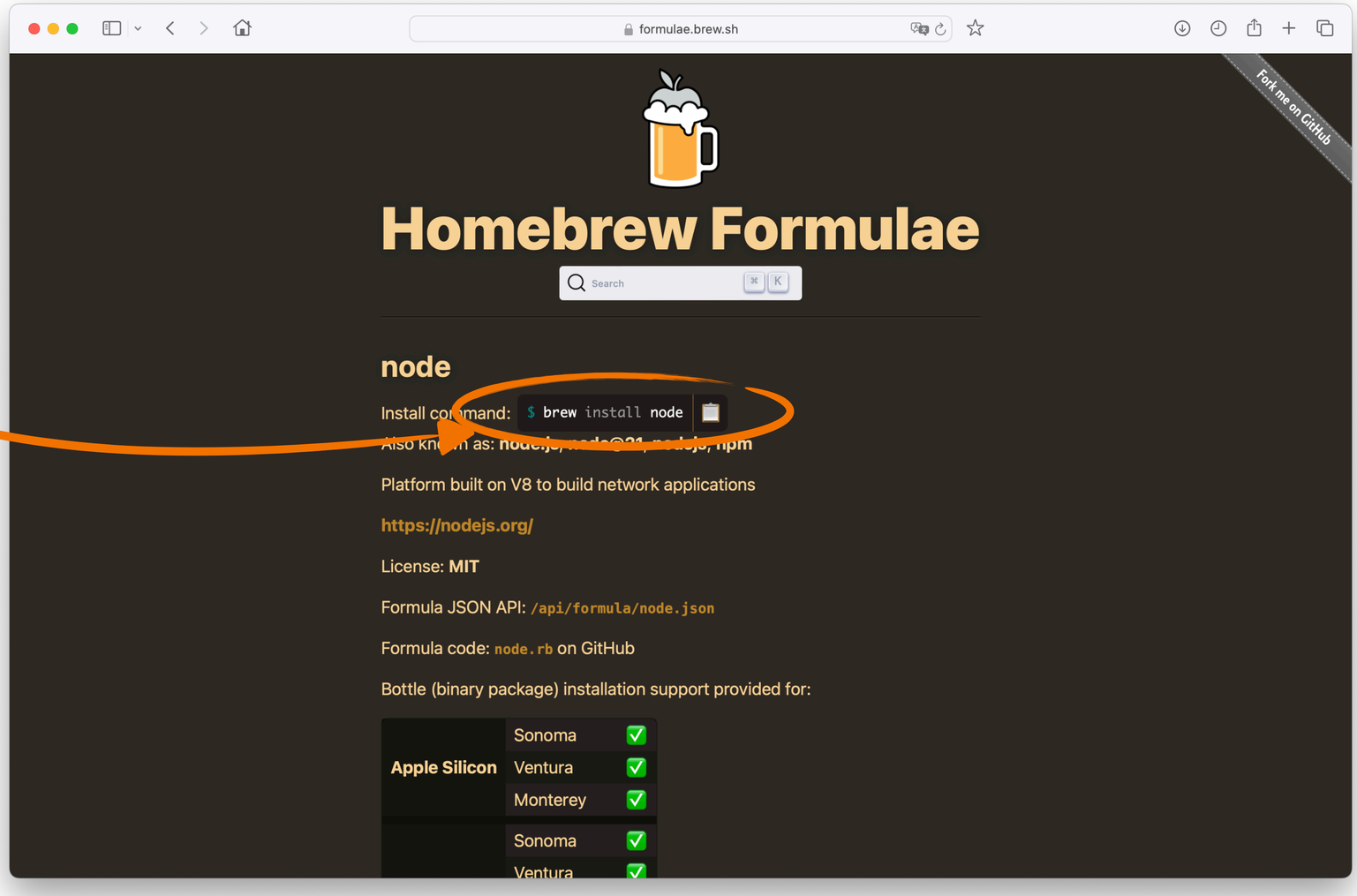
Check brew version:

```
brew --version
```



Mac: Install Node.js

Just run, and wait

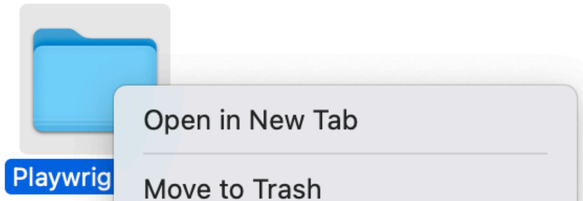


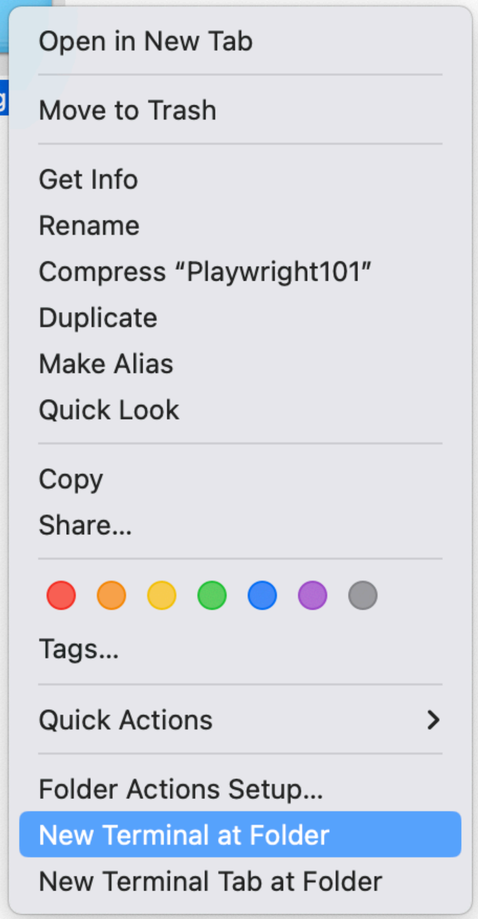
Playwright project structure

and

Example scenarios from init

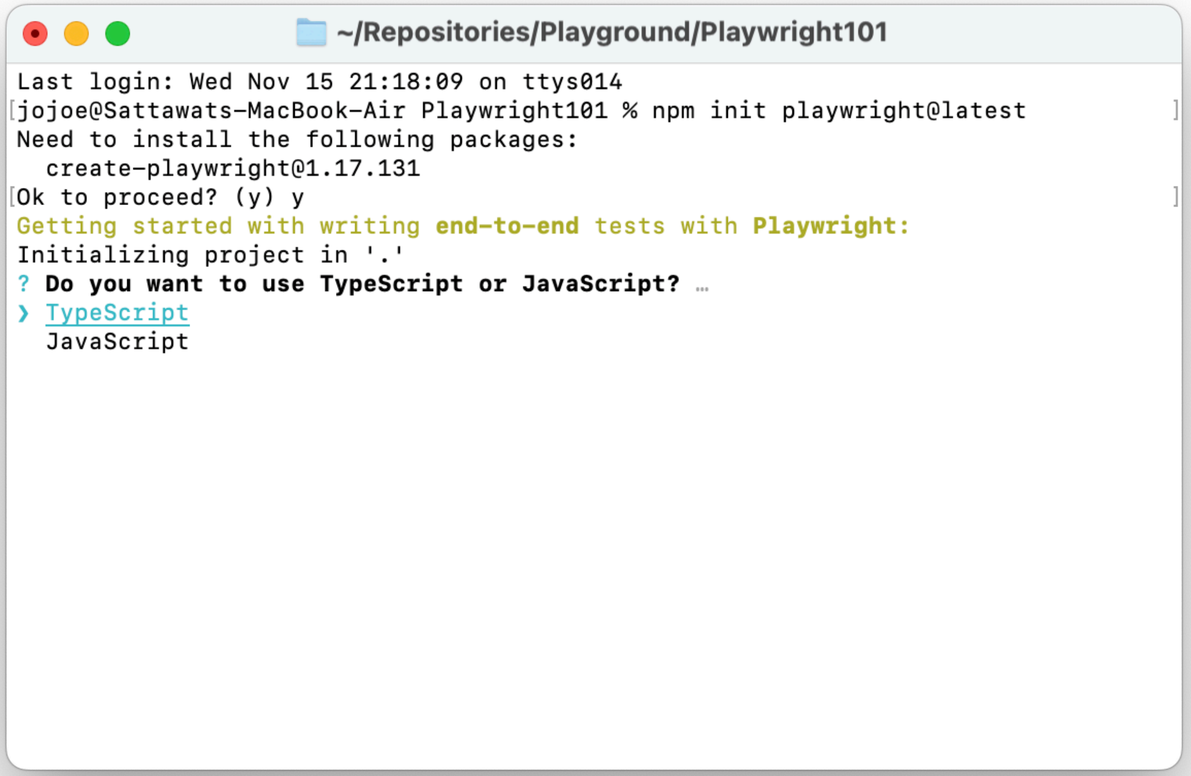
Initial Playwright project



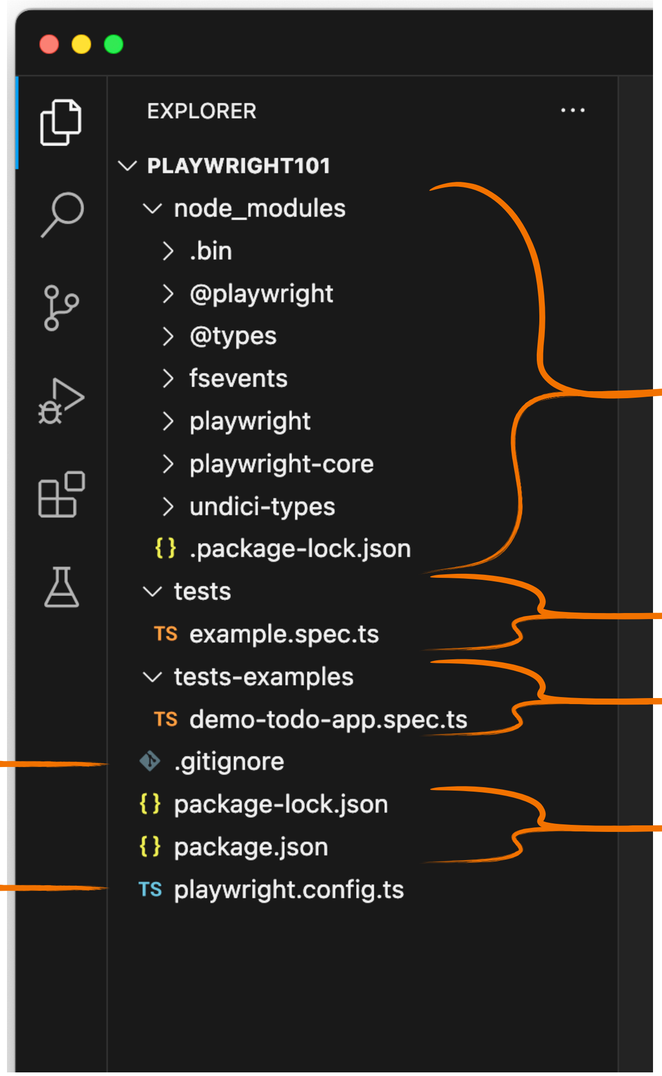


Run command:

```
npm init playwright@latest
```



Project structure



Installed Node.js libraries

Our test scenarios will be implement here

Example test scenarios from Playwright init

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

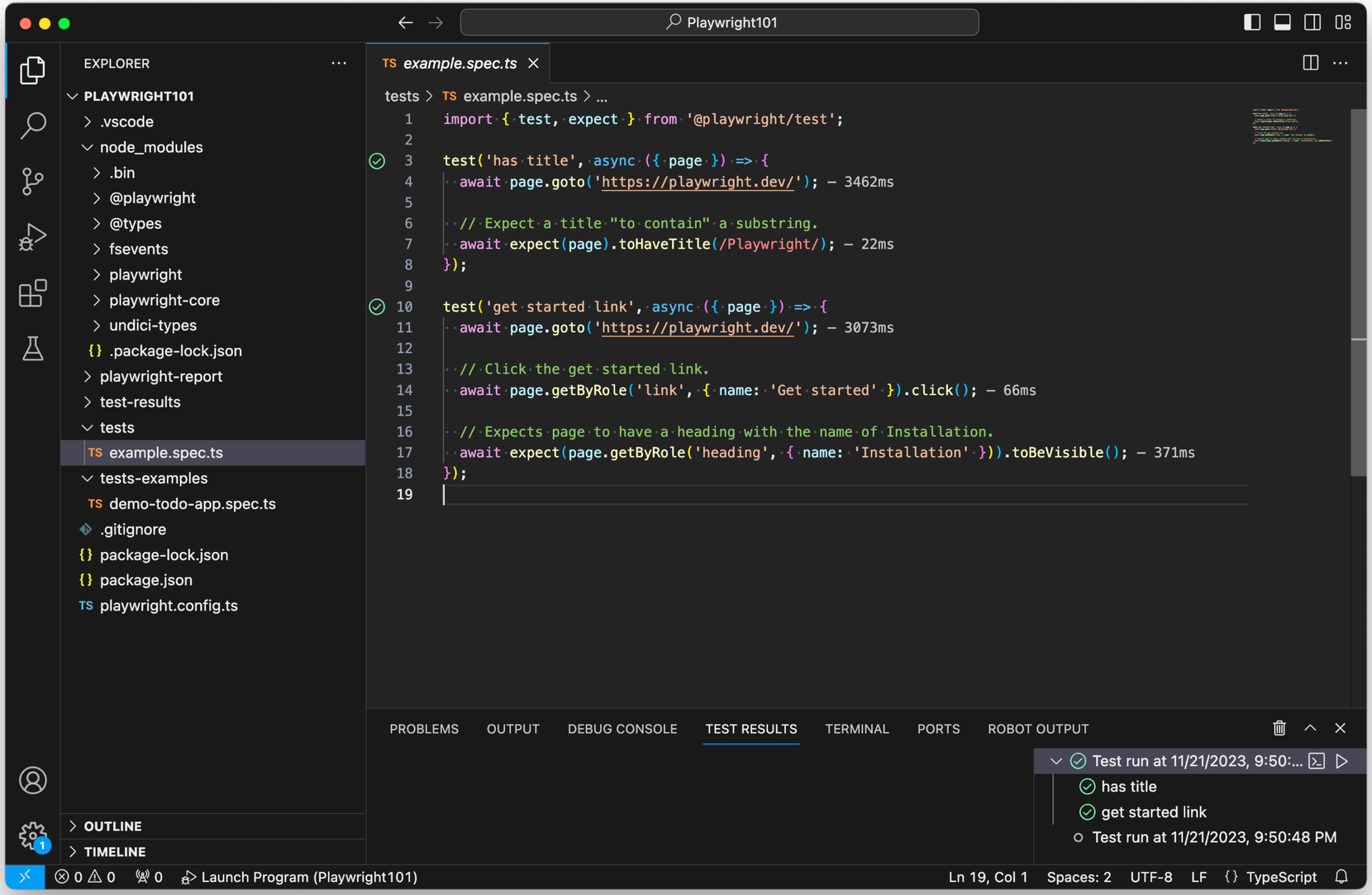
Specifies intentionally untracked files that Git should ignore

options to configure how your tests are run

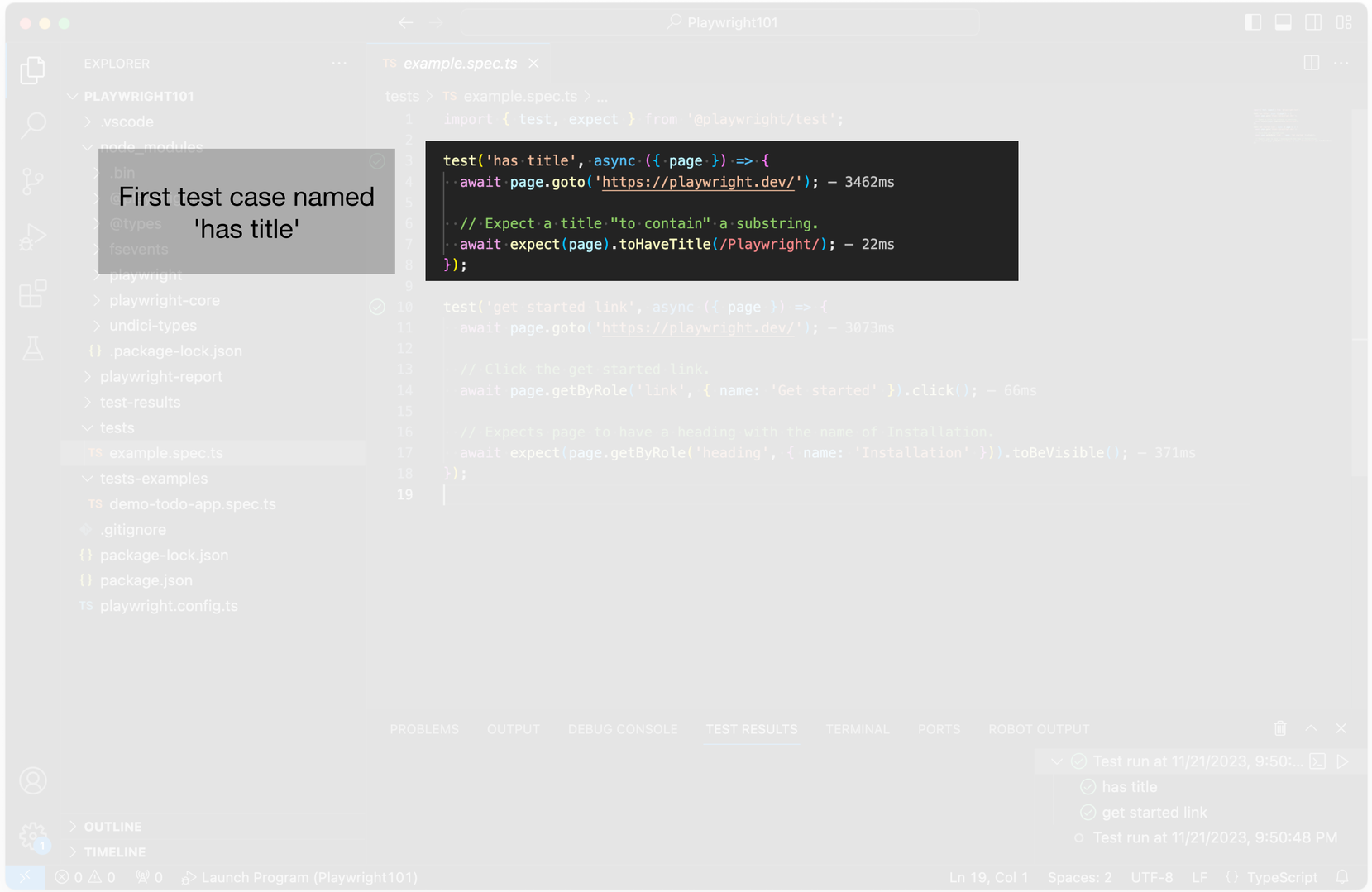
Ref1: package.json: https://docs.npmjs.com/cli/v10/configuring-npm/package-json
Ref2: package-lock.json: https://docs.npmjs.com/cli/v10/configuring-npm/package-lock-json

Ref3: gitignore: https://git-scm.com/docs/gitignore
Ref4: Test configuration: https://playwright.dev/docs/test-configuration

Example test cases



Example test cases: has title



Test scenario structure

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

```
test(          ,
);
```

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**'

```
              async (          ) => {
}
```

```
test('has title', async ({ page }) => {
  await page.goto('https://playwright.dev/'); — 3462ms

  // Expect a title "to contain" a substring.
  await expect(page).toHaveTitle(/Playwright/); — 22ms
});
```

Example test cases: has title

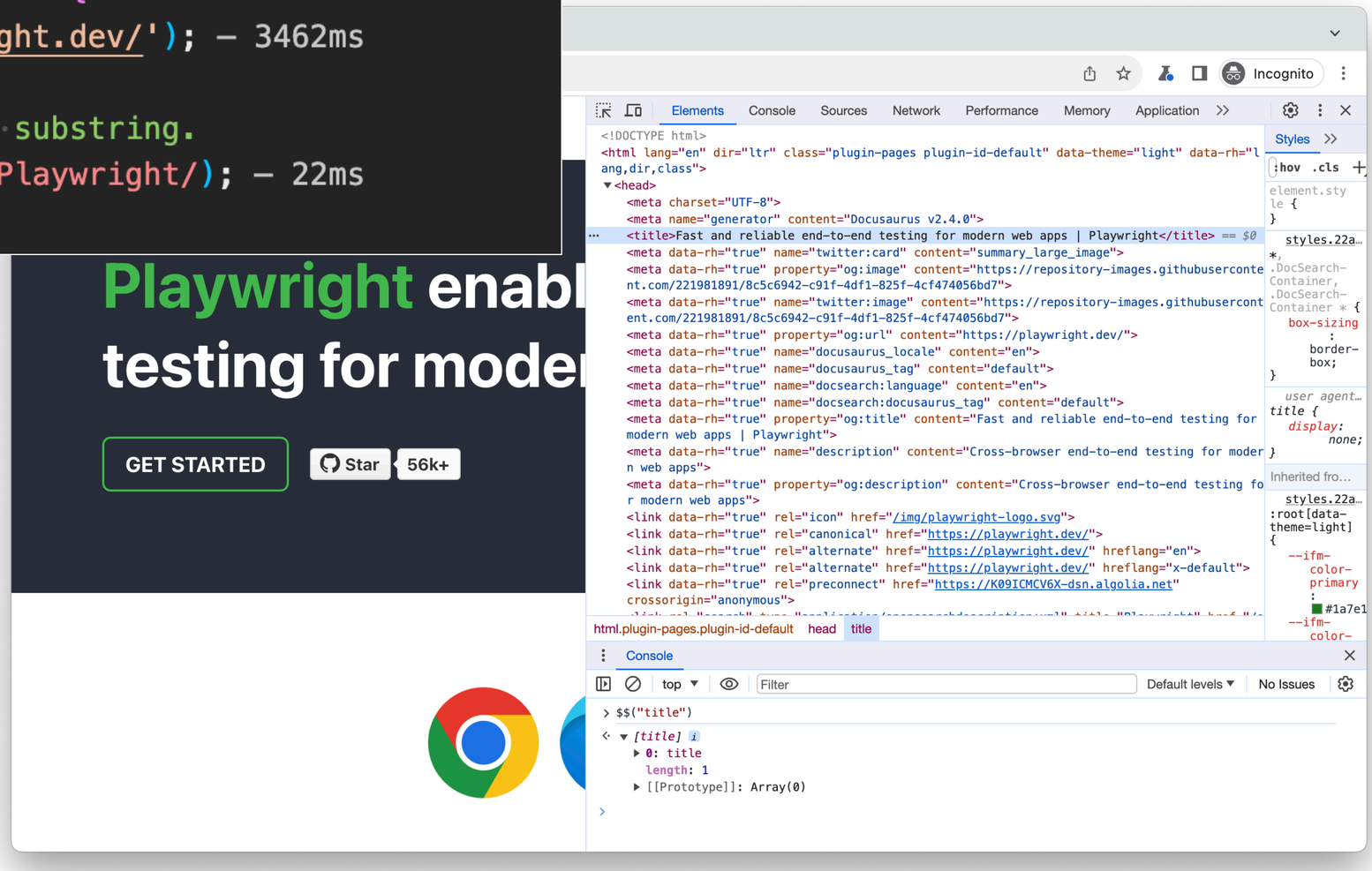
```
3  test('has title', async ({ page }) => {
4    await page.goto('https://playwright.dev/'); — 3462ms
5
6    // Expect a title "to contain" a substring.
7    await expect(page).toHaveTitle(/Playwright/); — 22ms
8  });
```

Test steps:

Line3 Create test case named 'has title' and assigned async function with 1 argument name 'page'

Line4 At current page, navigate to web url 'https://playwright.dev/'

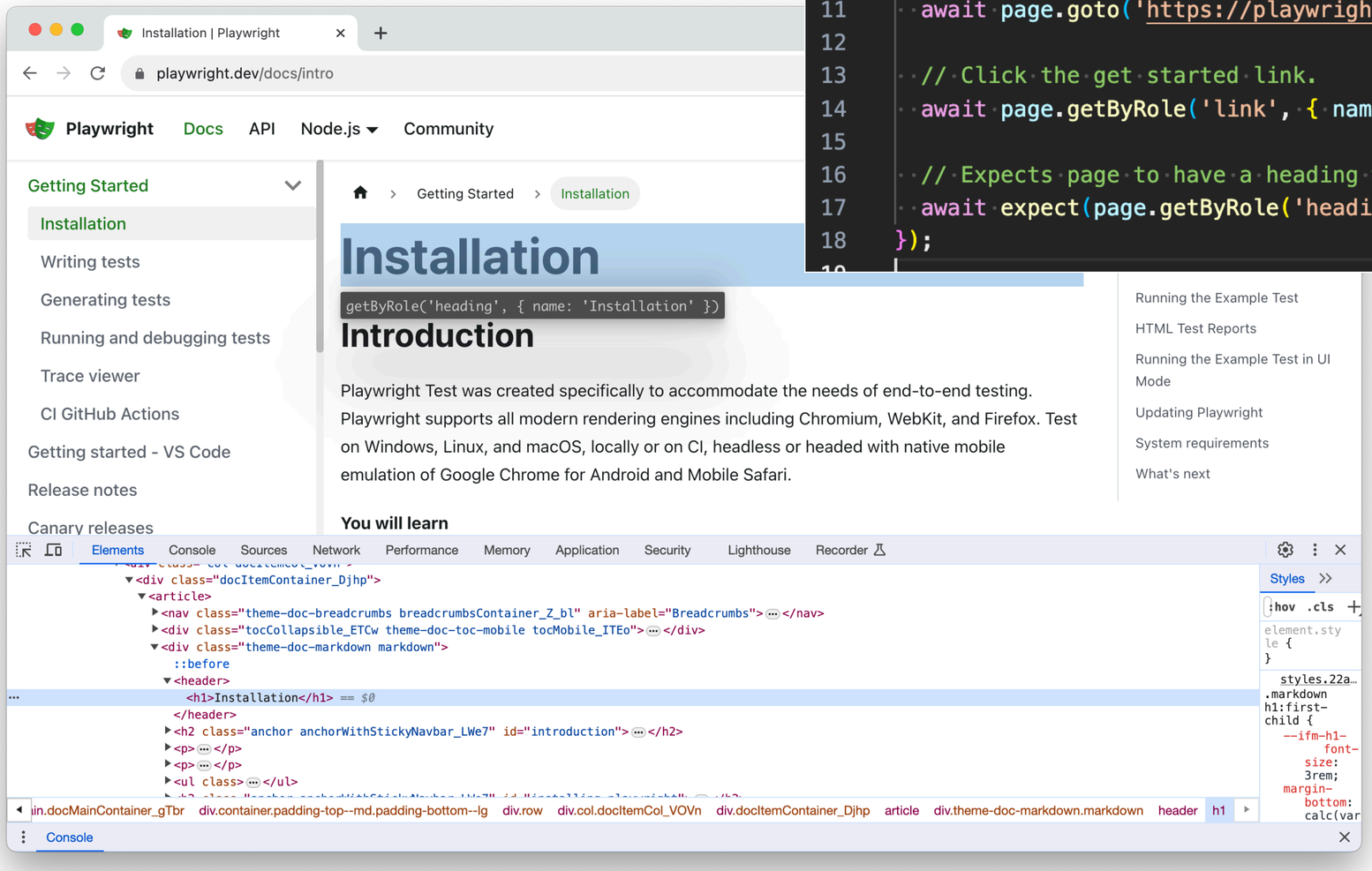
Line7 Assert page's title must contains text 'Playwright'



Example test cases: get started link

Example test cases: get started link

```
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

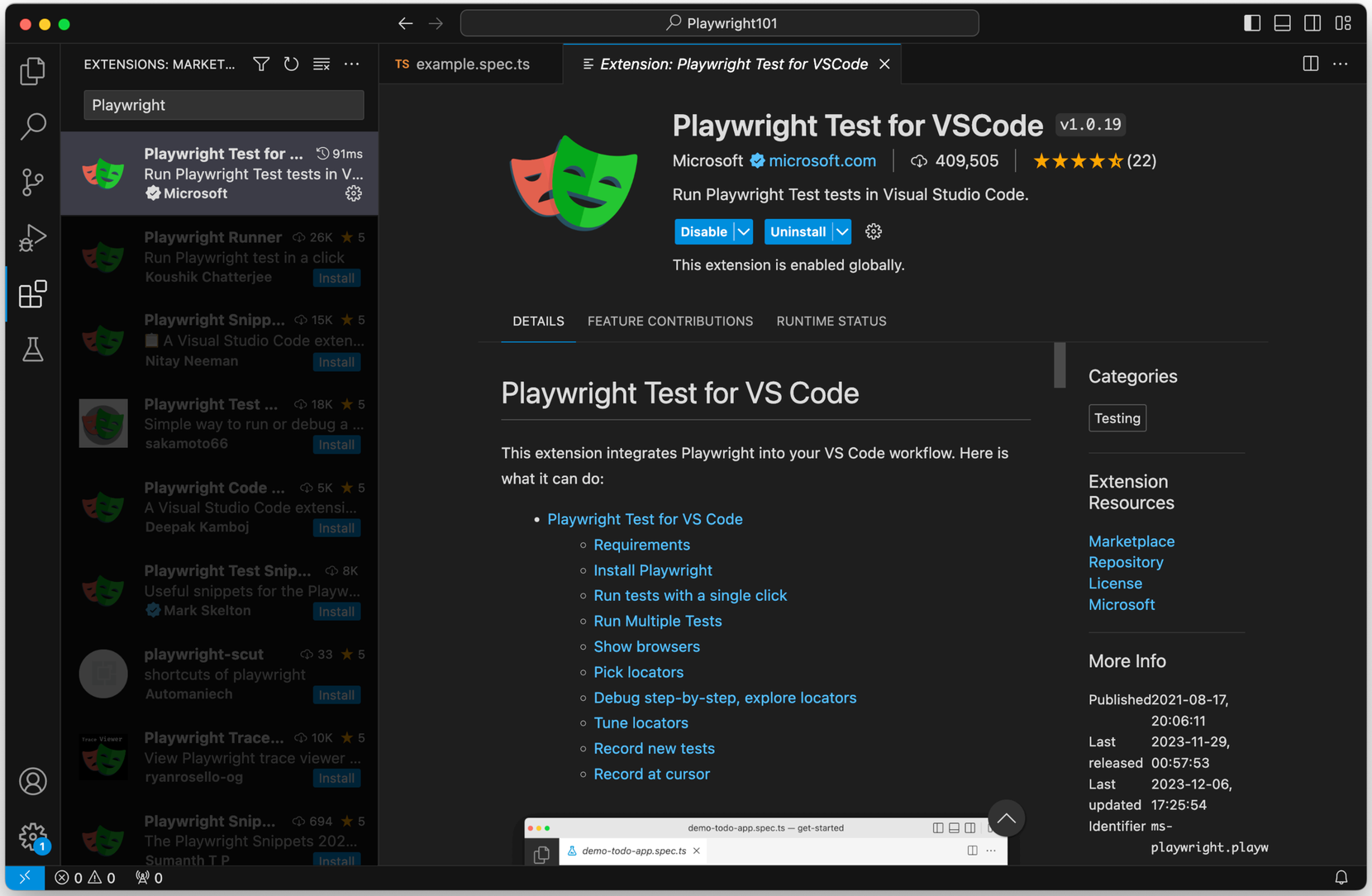
Debug mode

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

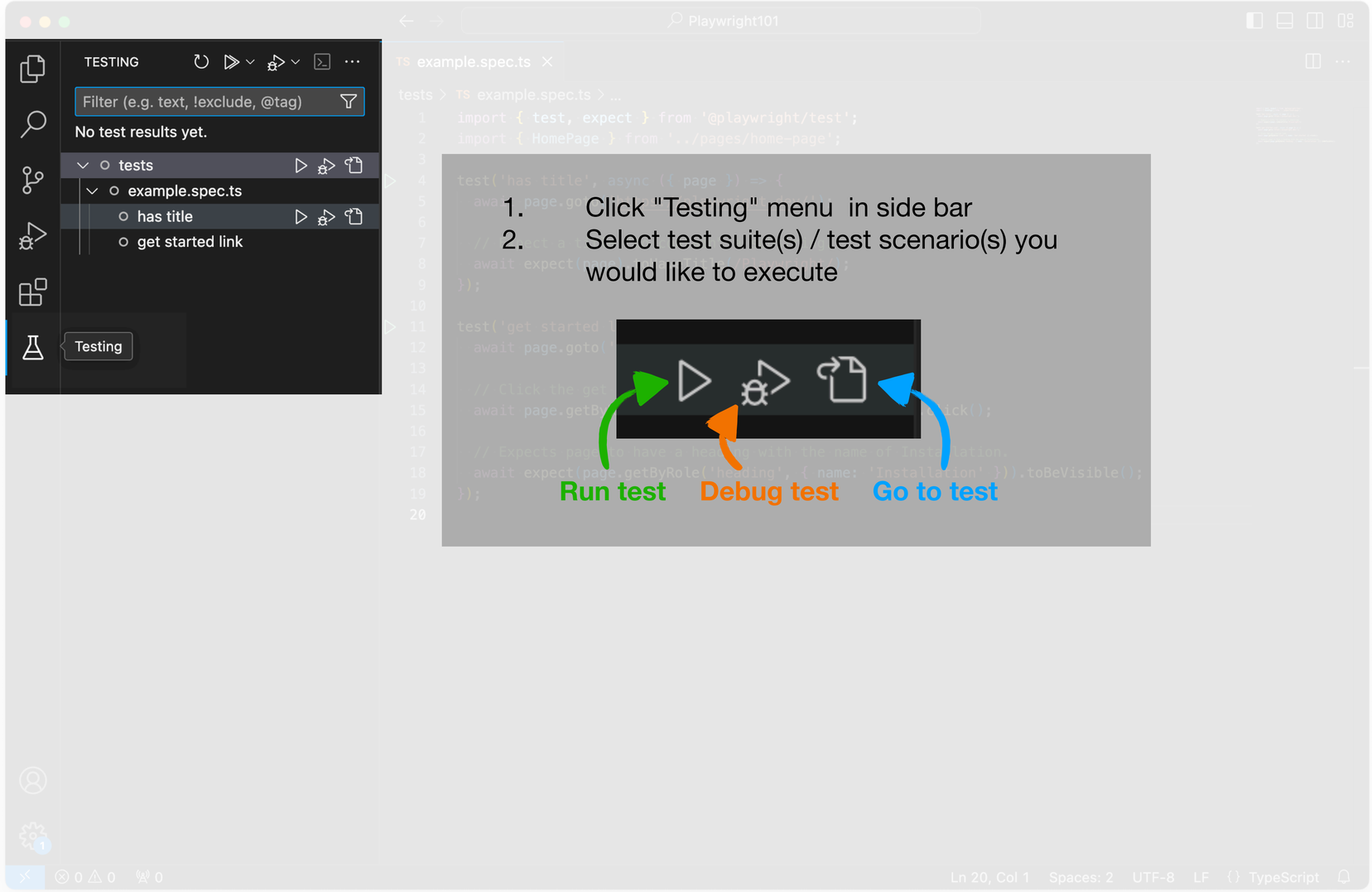
Ref1: https://playwright.dev/docs/intro#running-the-example-test-in-ui-mode
Ref2: https://playwright.dev/docs/test-ui-mode

Ref3: https://playwright.dev/docs/debug

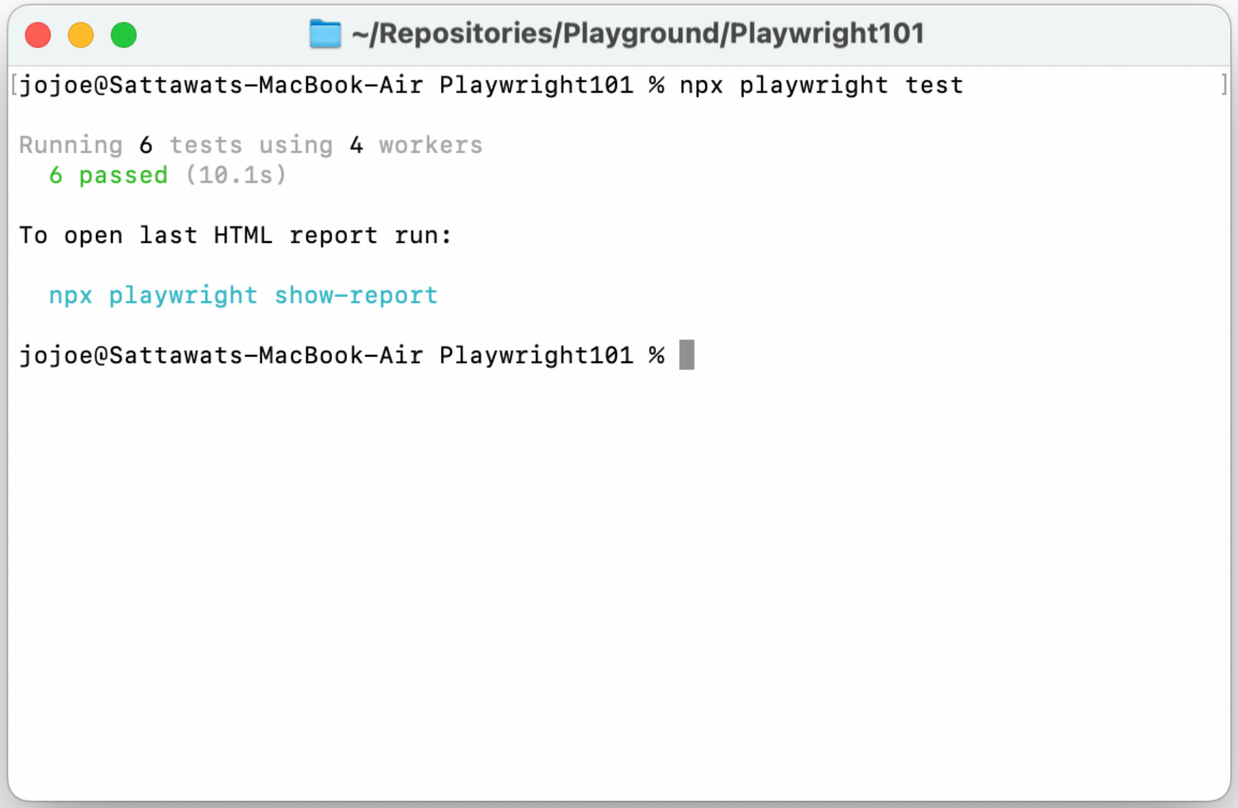
VSCode extension



Execute / debug tests from VSCode



Execute tests from command



Execute all test suite and scenarios:

```
npx playwright test
```

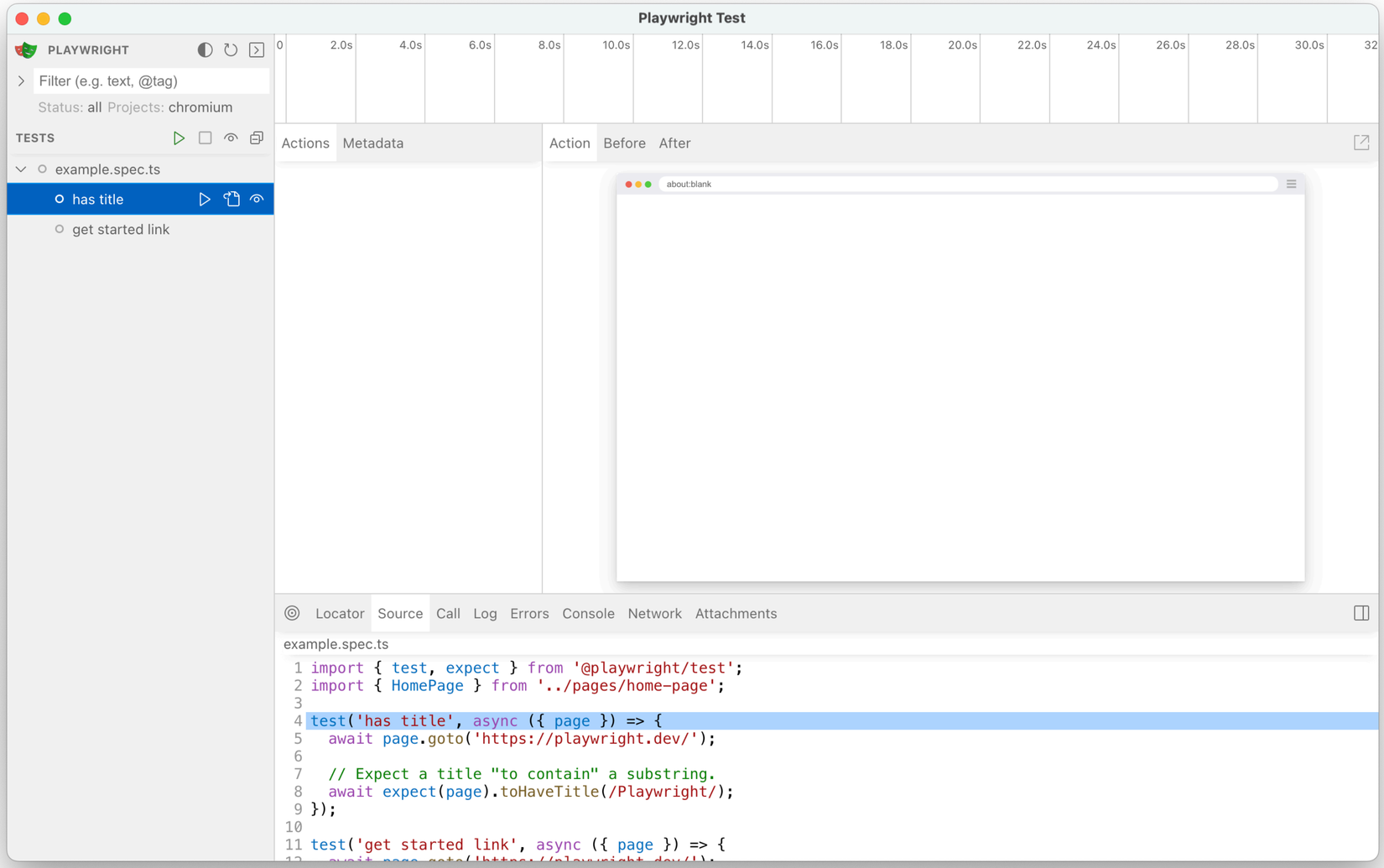
Execute single test suite:

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

Execute single test scenarios:

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

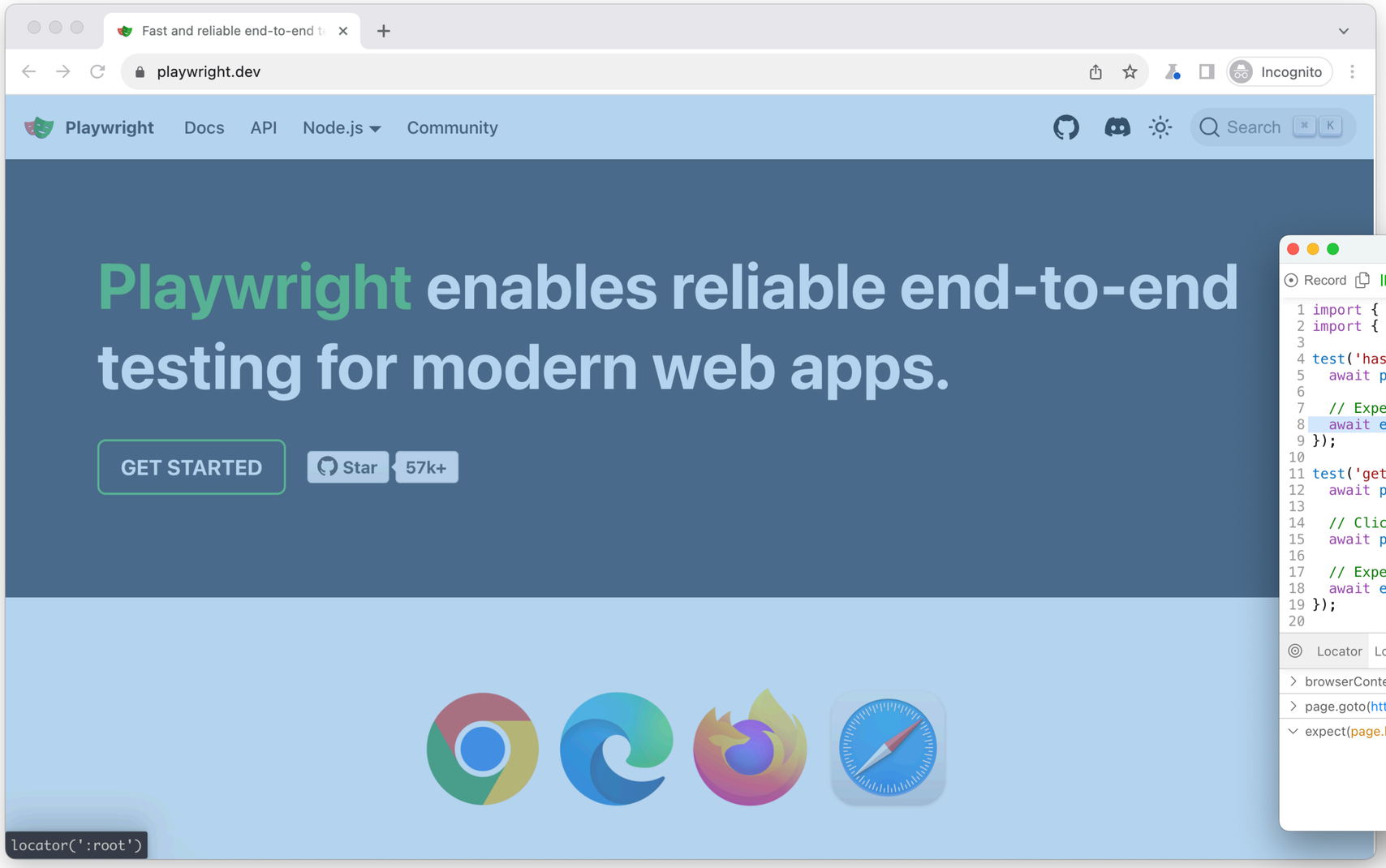
Execute tests in UI mode from command



Open inspector UI mode:

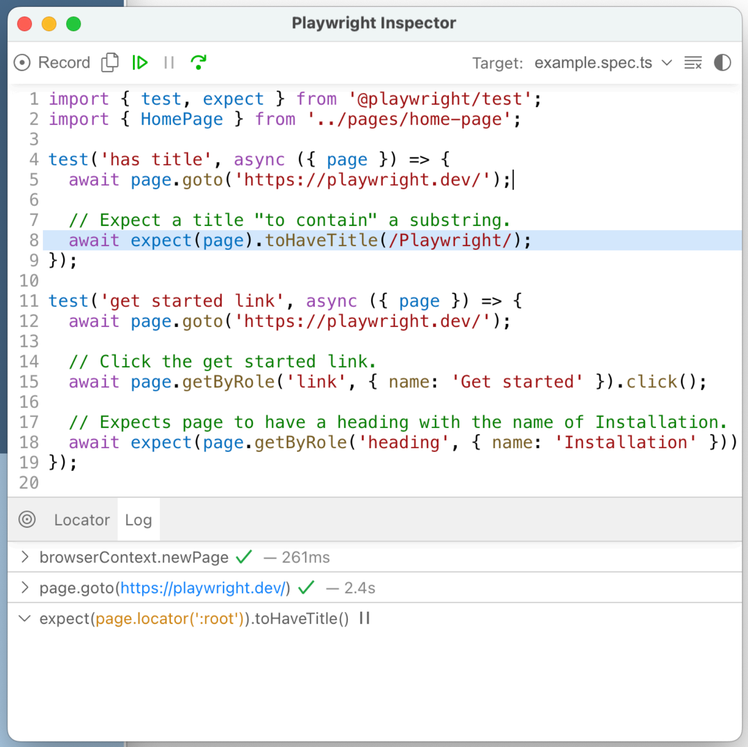
```
npx playwright test --ui
```

Execute tests in debug mode from command



Open inspector debug mode:

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



Locators

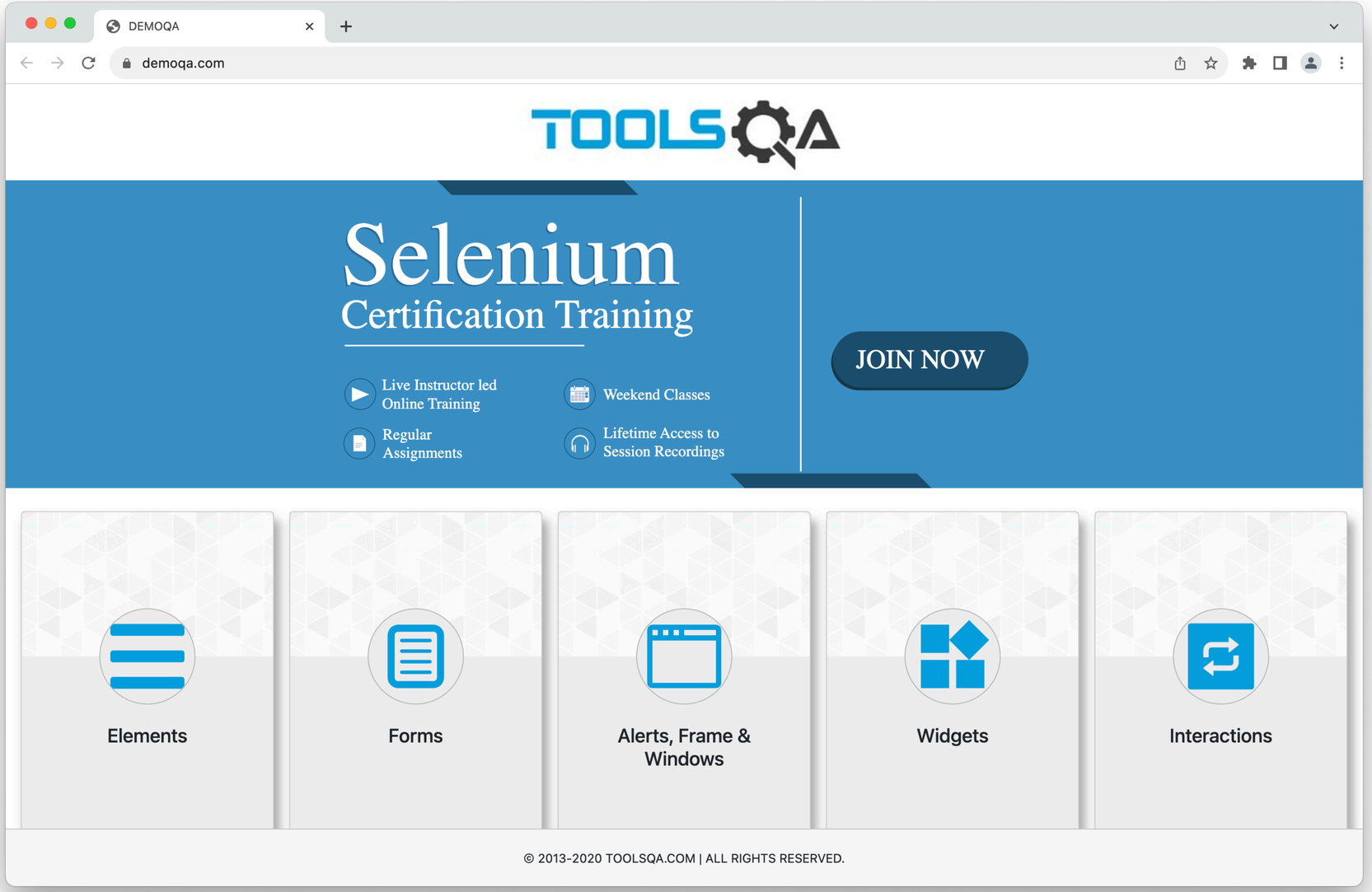
- Chrome inspect and console
- HTML Tags
  - ID
  - CSS
  - XPath
  - data-testid
  - Chain locator

Ref1: https://playwright.dev/docs/other-locators

Tools QA : Our playground in this session :)

Open Chrome browser

Go to this url: demoqa.com

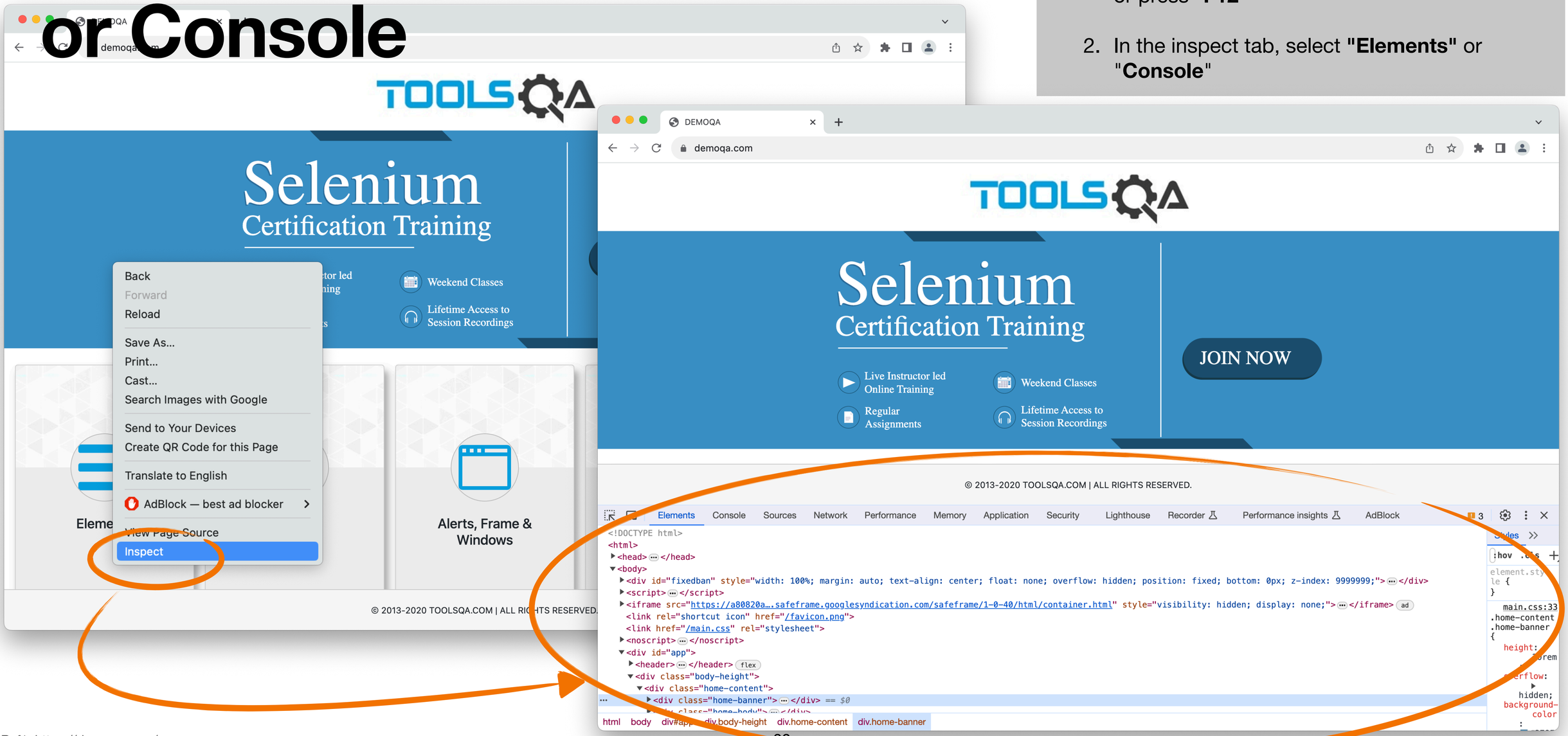


Ref1: https://demoqa.com/

Chrome Inspect, Elements, or Console

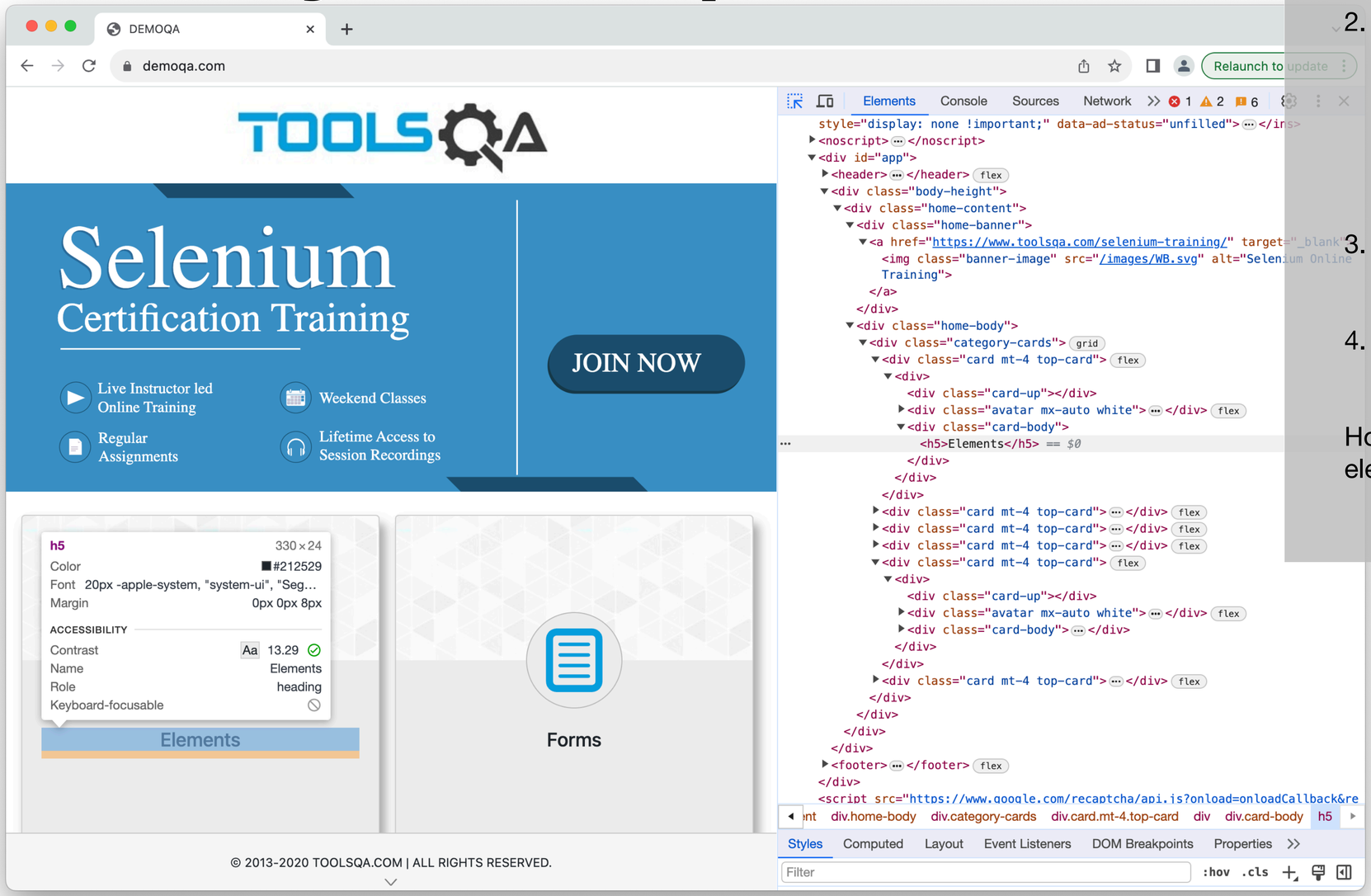
Open inspect and console steps:

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



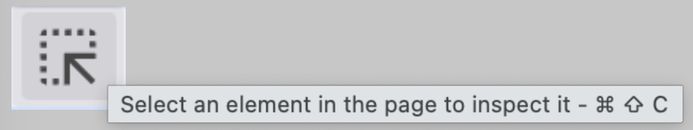
Ref1: https://demoqa.com/

Identify the unique locator



Steps to identify locator of some elements:

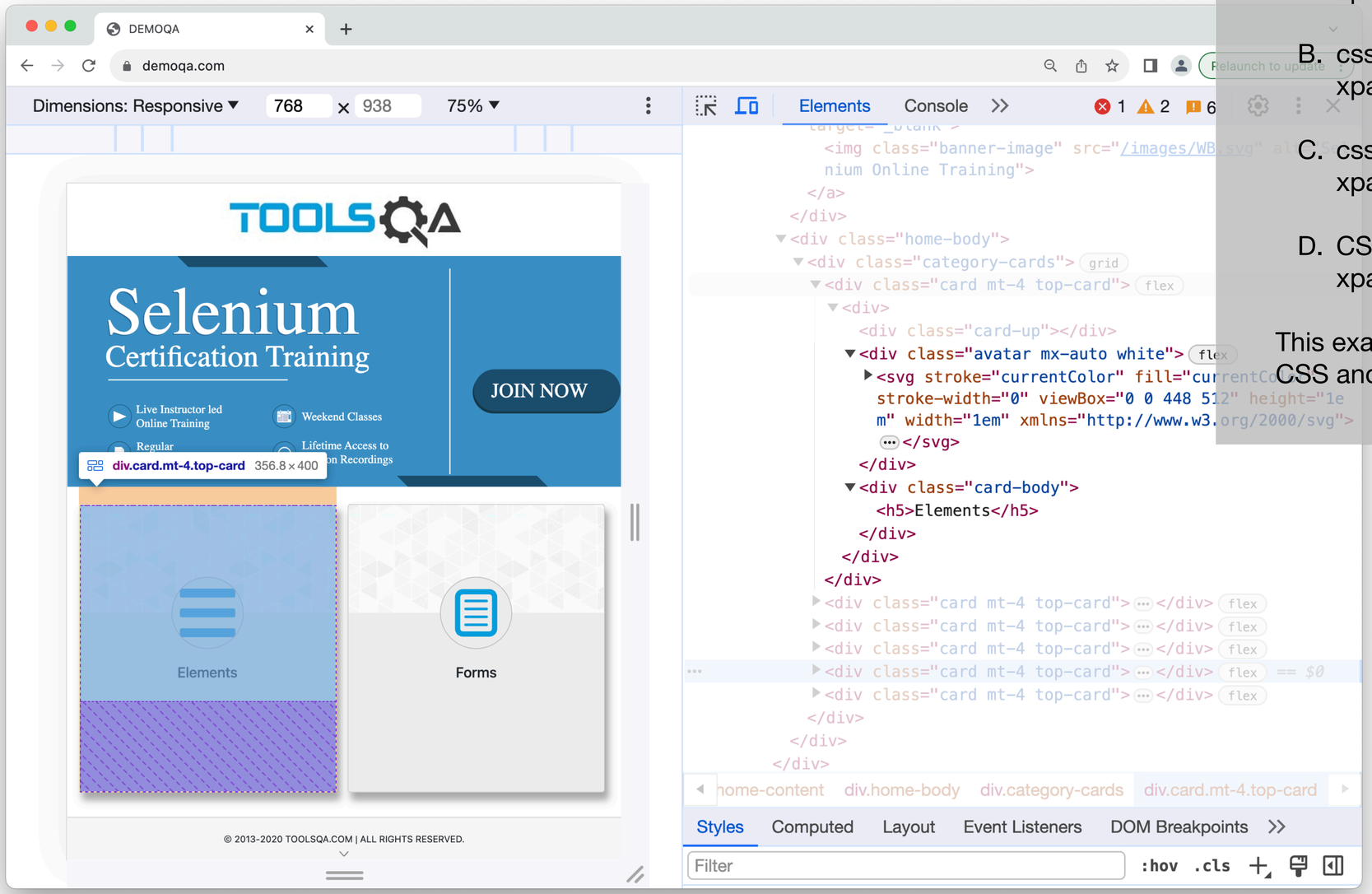
1. Open Chrome inspect, and go to Elements tab
2. Use this tool, it is the easies way



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



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']

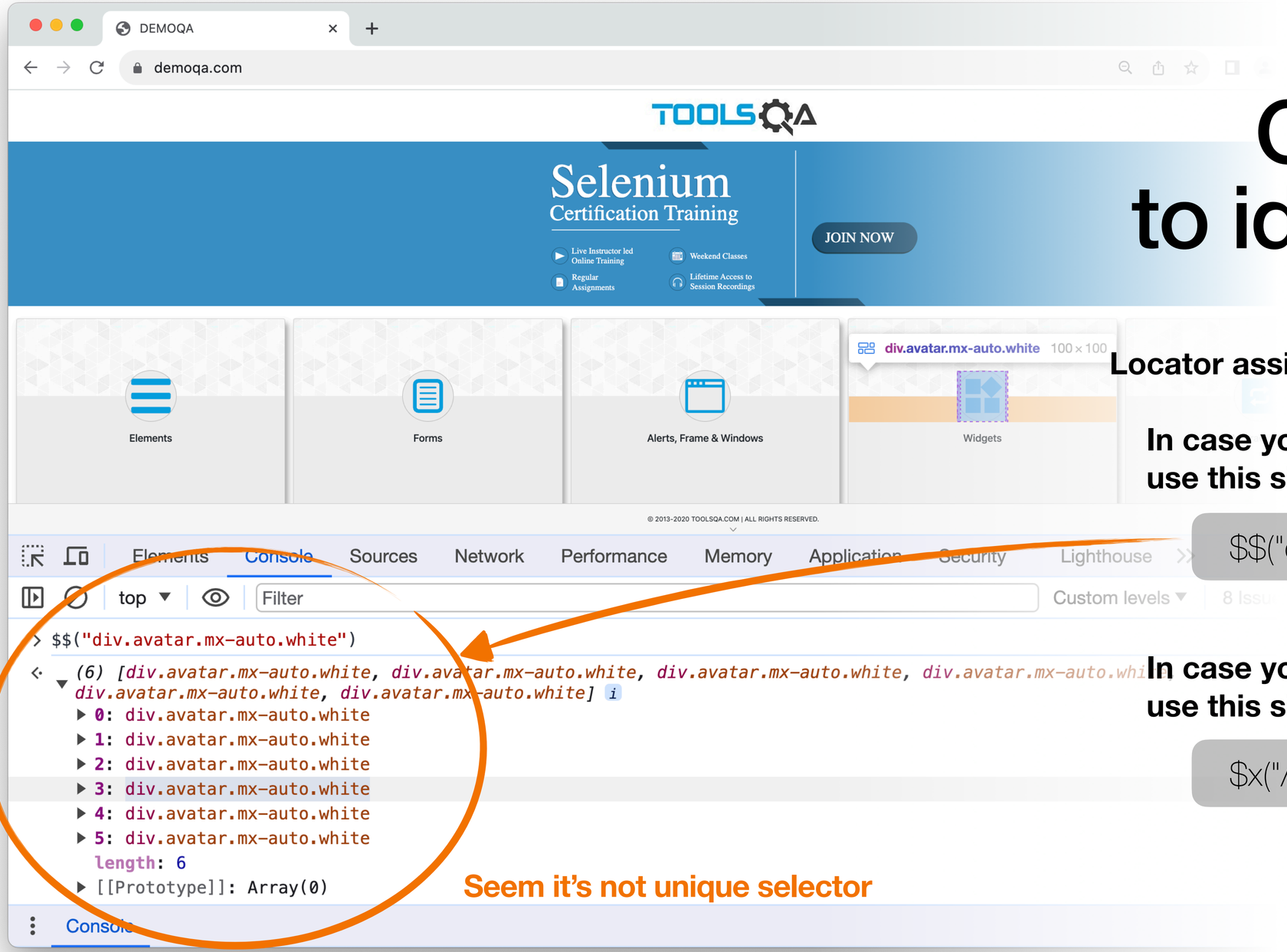
B. css = div.card-body
   xpath = //div[@class='card-body']

C. css = h5
   xpath = //h5

D. CSS is 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



Locator assignment:

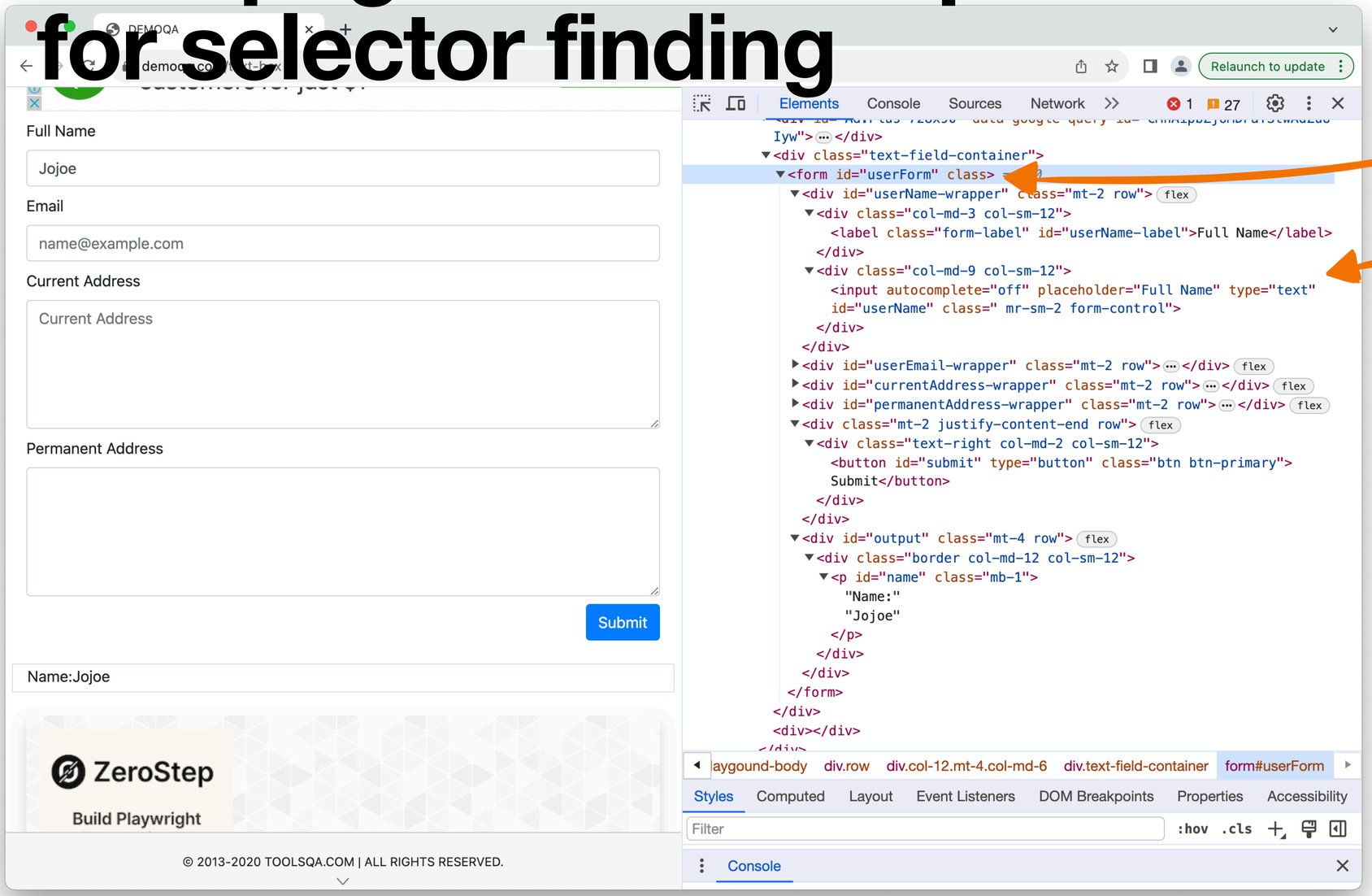
In case you would like to fine the CSS selector use this sign:

```
$$("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']")
```

This page is the best practice for selector finding



id = userForm

css = form#userForm

id = userName

css = input#userName

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

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

data-testid locator

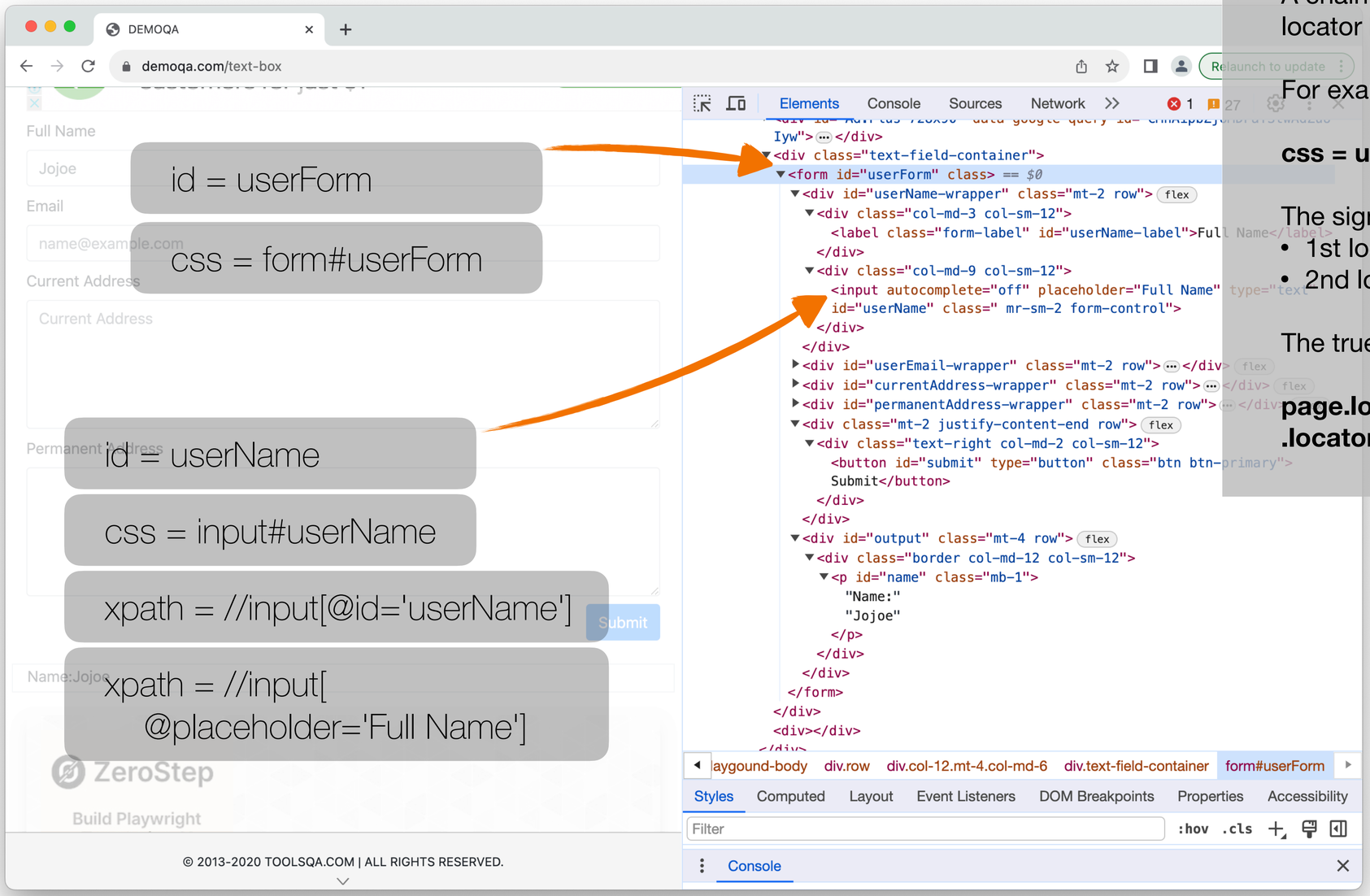
data-testid is an attribute commonly used in web development to facilitate testing of user interfaces. It is not a native HTML attribute, but rather a **custom attribute that developers add to their HTML elements to make it easier to select and interact with those elements in automated tests.**

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

Ref1: OpenAI. (2023). "What is data-testid and how to naming it."

Chain locator



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']'")

Ref1: https://playwright.dev/docs/other-locators#chaining-selectors

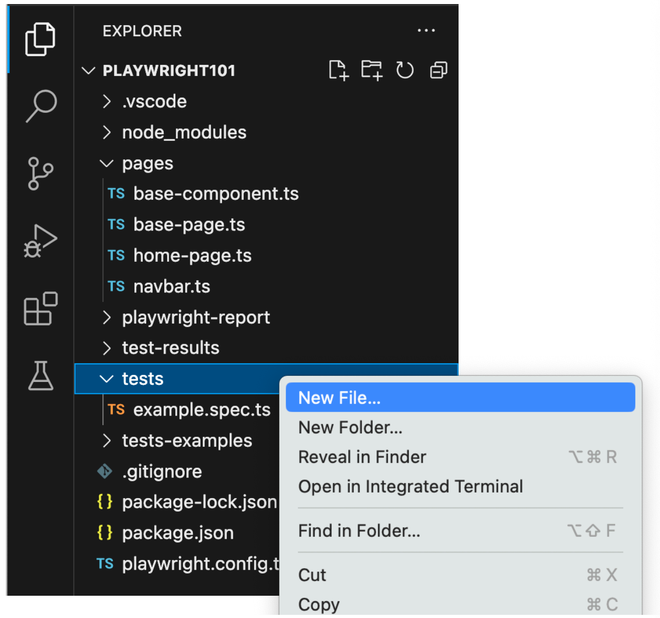
Create first test

Suite

and

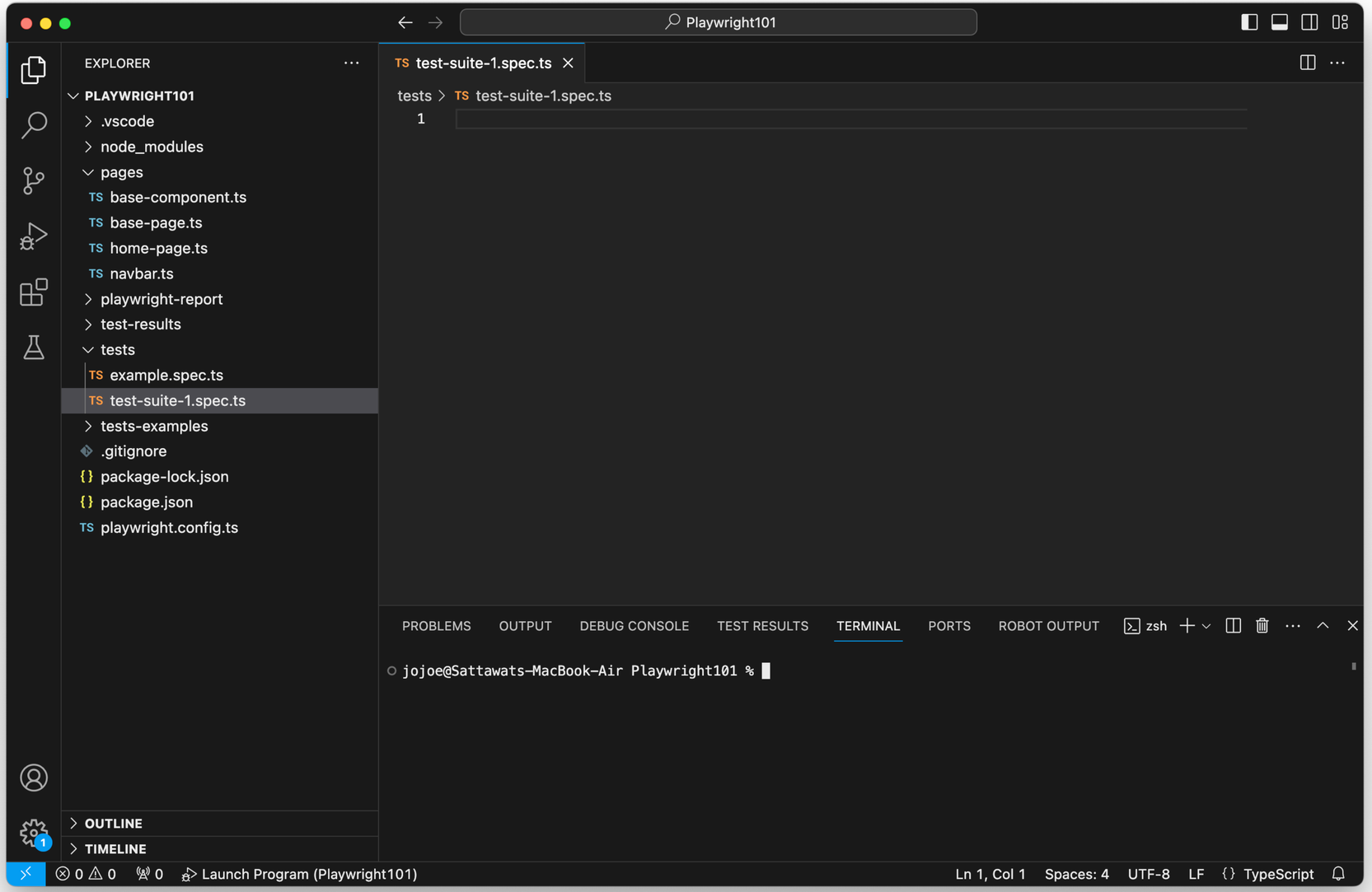
Scenario

Create first test suite file

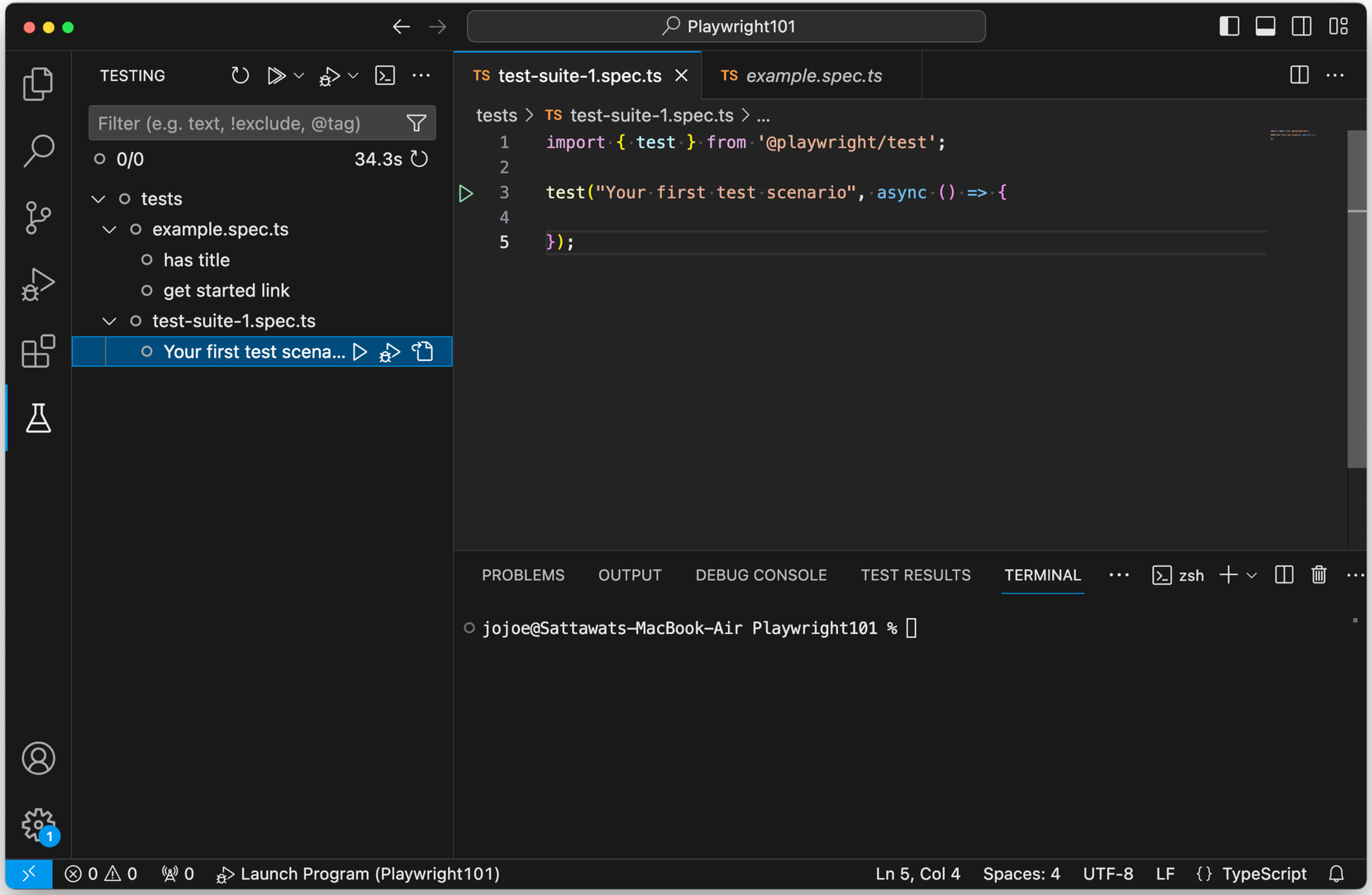


Create new test file (Test suite) steps:

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



Actions

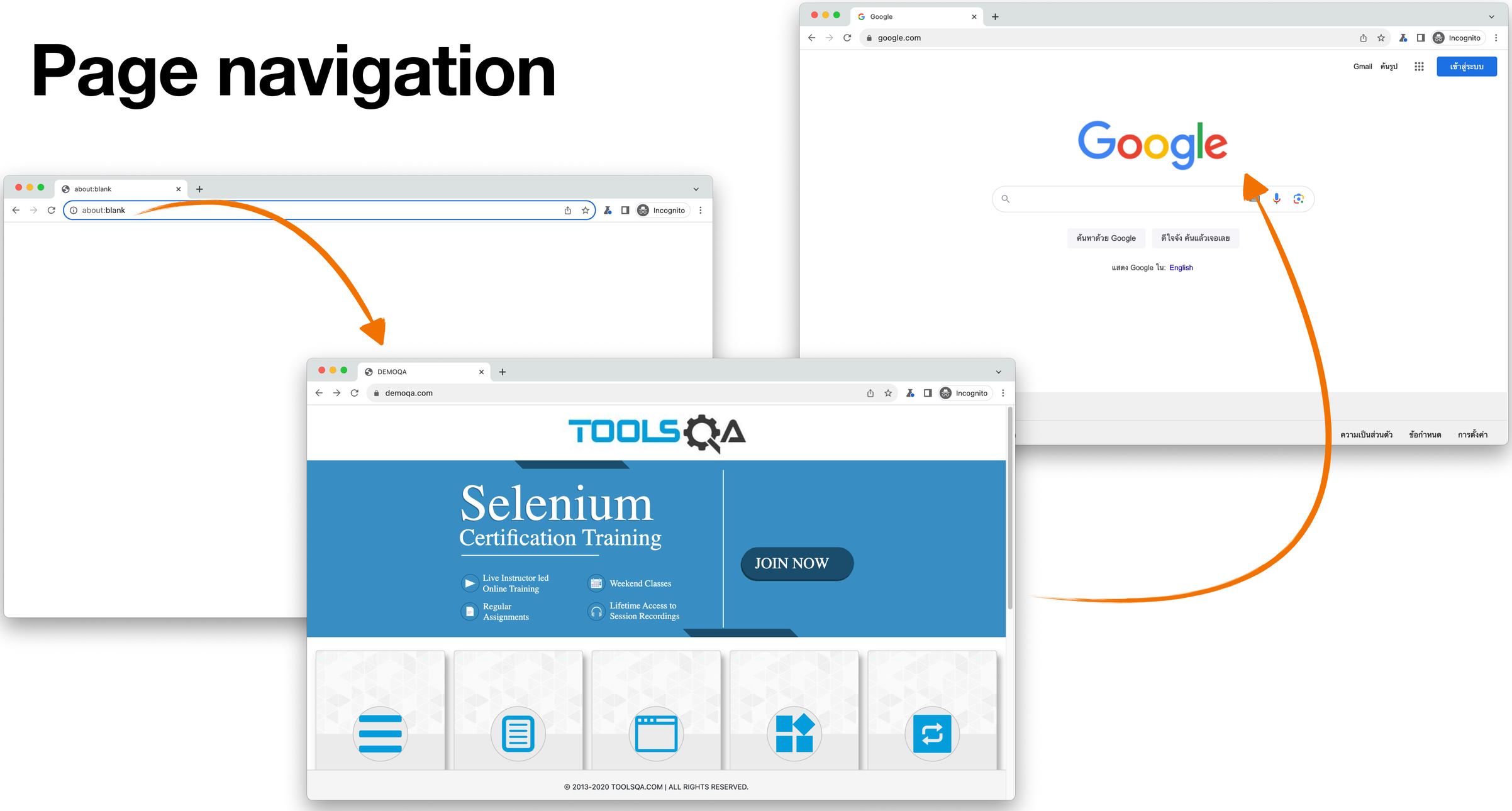
Scraping

Assertion

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

Page navigation

Page navigation



Methods using for page navigation

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

Playwright - Example code

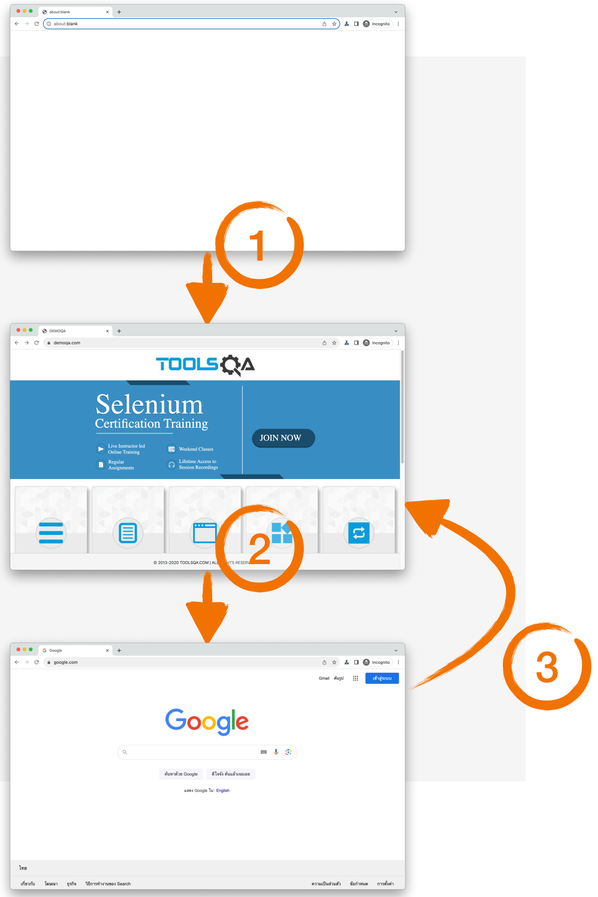
Method & Optionals	Return type	Purpose	Example
.goto( )	`Promise< null|Response >`	Navigate to the input url.	`await page.goto('https://...');`

```
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();`

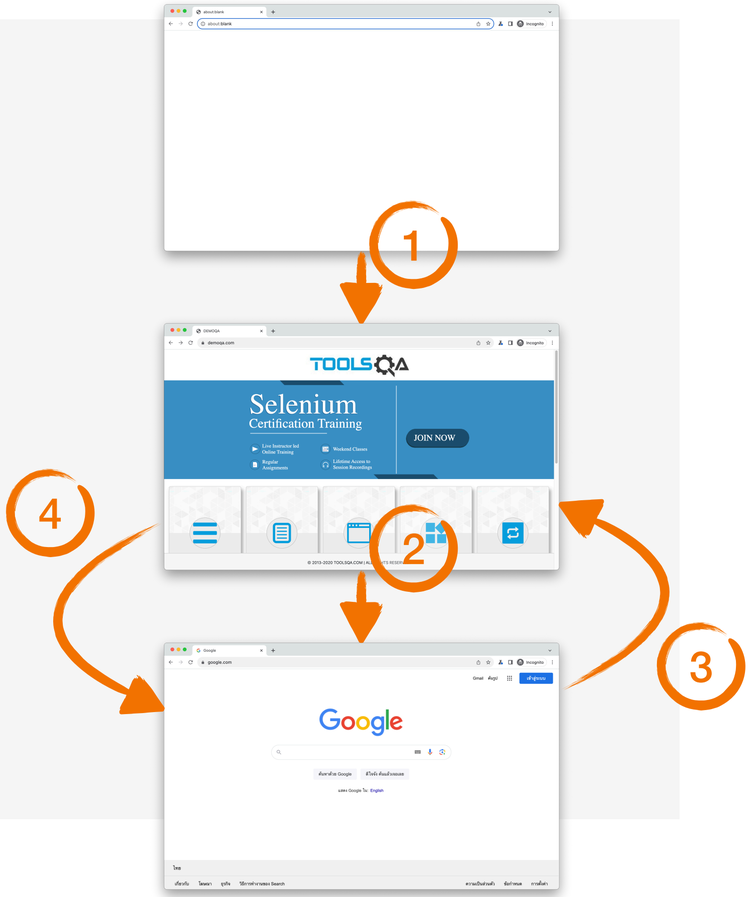
```
test("Page navigation example - backward", async ({ page }) => {
    // Arrange
    const url = 'https://demoqa.com';
    await page.goto(url);

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

    // Action
    await page.goBack();

    // 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();`

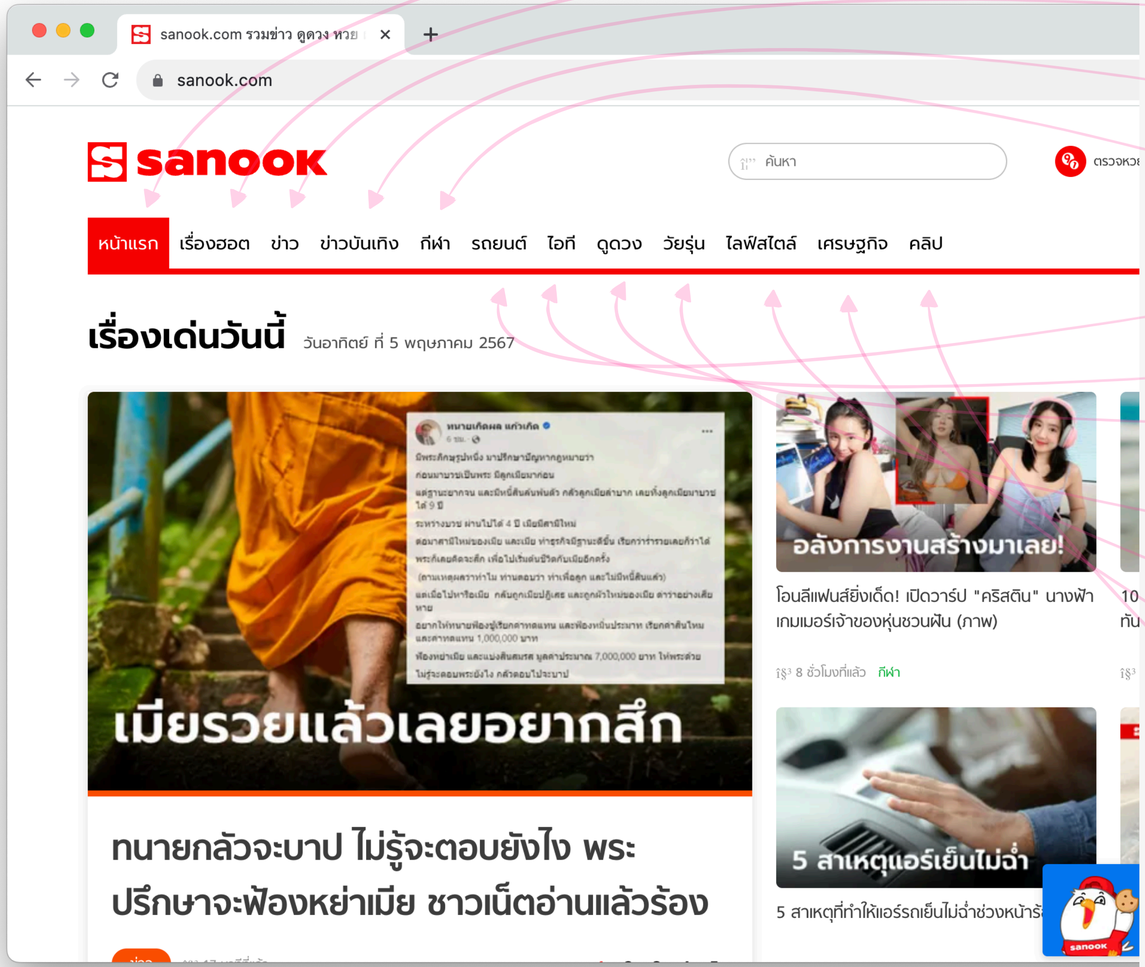
```
test("Page navigation example - forward", async ({ page }) => {
    // Arrange
    const url = 'https://demoqa.com';
    await page.goto(url);

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

    await page.goBack();

    // Action
    await page.goForward();

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

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



Text

Text

>> www.sanook.com
Look! Some texts appear on the screen.



Locate Elements

```
<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/">เรื่องฮอต</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>
```

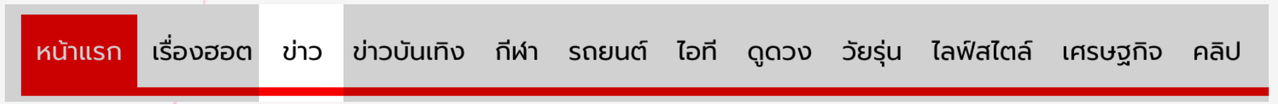
Frequently methods using with text

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

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');
});
```



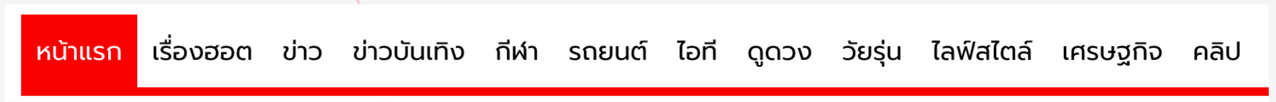

```
text = 'ข่าว'
```

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']
```

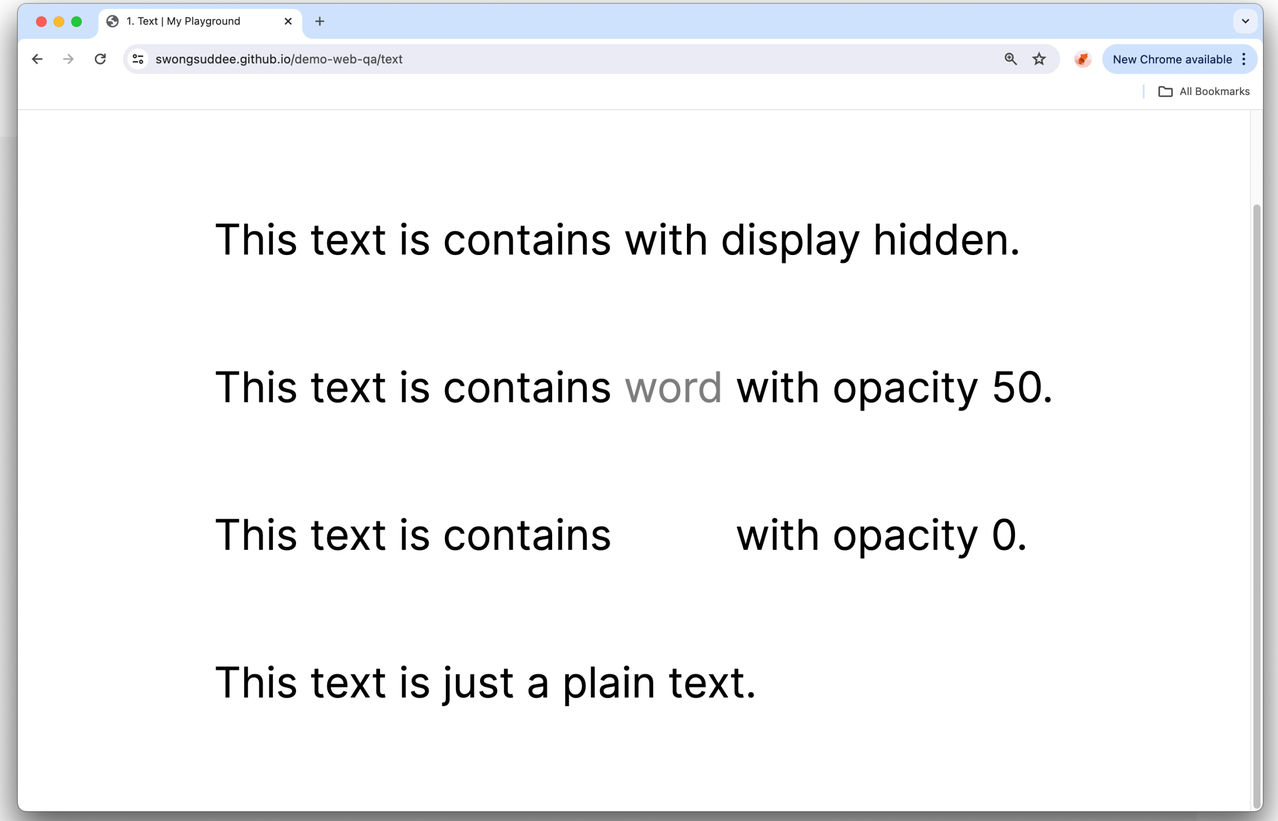
text in add to watch

Difference between

.innerText( ) .getAllInnerTexts( )	.textContent( ) .getAllTextContents( )
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.	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.

Ref1: https://developer.mozilla.org/en-US/docs/Web/API/Node/textContent#
Ref2: https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/innerText

Example hidden text




```html
<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>
```

Ref1: https://swongsuddee.github.io/demo-web-qa/text

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.
```

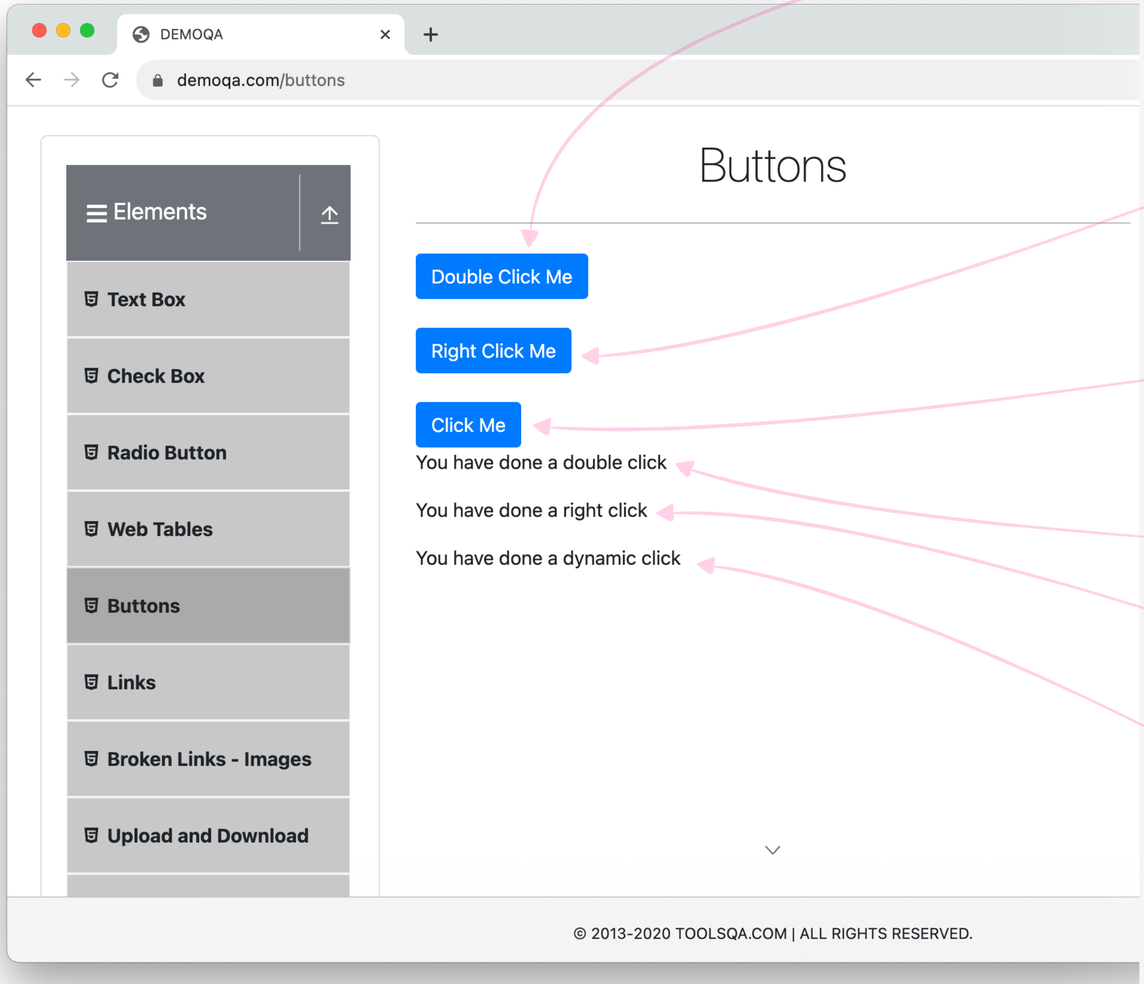
Button

<button> … </button>

Mouse click

>> Elements > Buttons

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



Locate Elements

```
<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>
...
```

Frequently methods using with button

Method & Optionals		Return type	Purpose	Example
.click( )	{ button: 'left' }	`Promise<void>`	Click an element.	`await locator.click();`
	{ button: 'right' }	`Promise<void>`	Right click an element.	`await locator.click({ button: 'right' });`
.dblclick( )	{ button: 'left' }	`Promise<void>`	Double-click an element.	`await locator.dblclick();`
	{ button: 'right' }	`Promise<void>`	Double-right-click an element.	`await locator.dblclick({ button: 'right' });`
.hover( )		`Promise<void>`	Hover over the matching element.	`await locator.hover();`
.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();`

Playwright - Example code

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

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

    const clickMeBtn = page.getByText('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
.dblclick( )	{ button: 'left' }	Promise<void>	Double-click an element.	await locator.dblclick();
	{ button: 'right' }	Promise<void>	Double-right-click an element.	await locator.dblclick({ button: 'right' });

```
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
.click( )	{ button: 'left' }	Promise<void>	Click an element.	await locator.click();
	{ button: 'right' }	Promise<void>	Right click an element.	await locator.click({ button: 'right' });

```
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.getByText('Click Me', { exact: true });

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

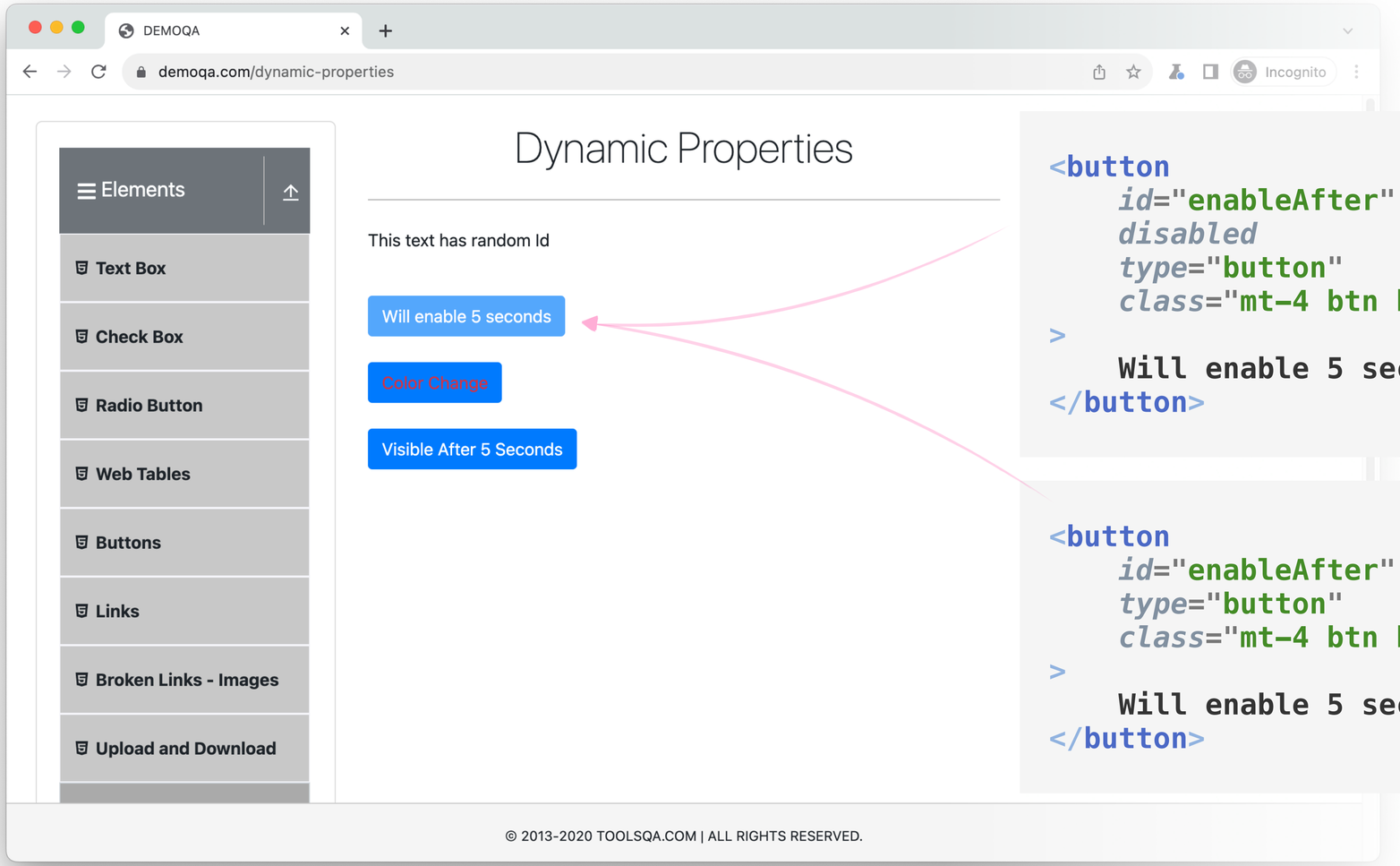
Click Me

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


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

Button enabled / disabled

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



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>
```

Locate Elements ➡ Playwright locator

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

Disabled status

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

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 }
);
```

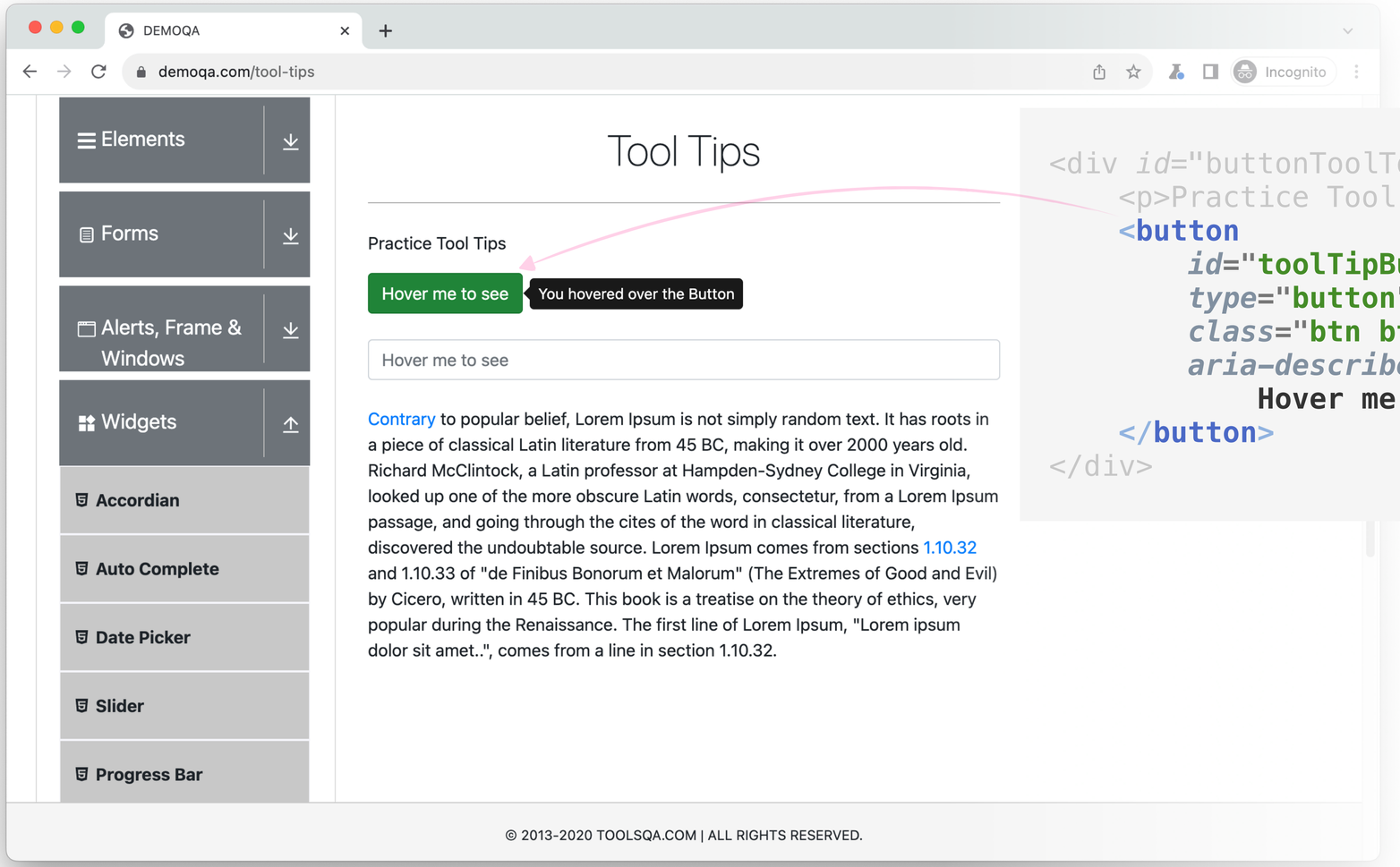
Playwright - Example code

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 (' ')




```
<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>
```

Locate Elements ➡ Playwright locator

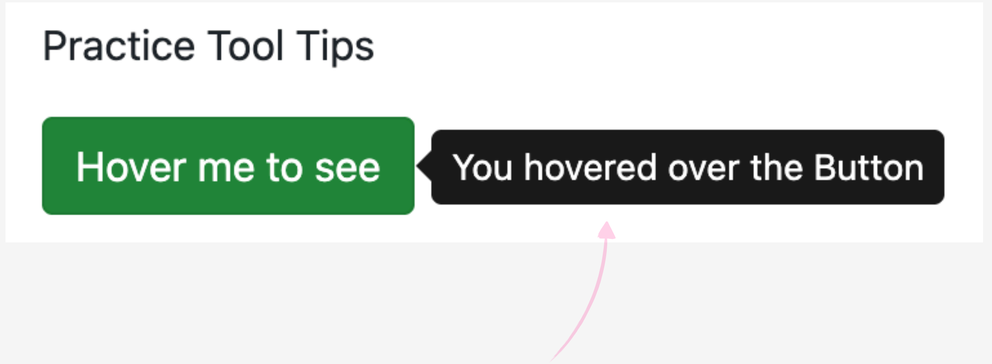
```html
<div id="buttonToolTopContainer">
    <p>Practice Tool Tips</p>
    <button
        id="toolTipButton"
        type="button"
        class="btn btn-success"
        aria-describedby="buttonToolTip">
            Hover me to see
    </button>
</div>
```

```javascript
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();
});
```



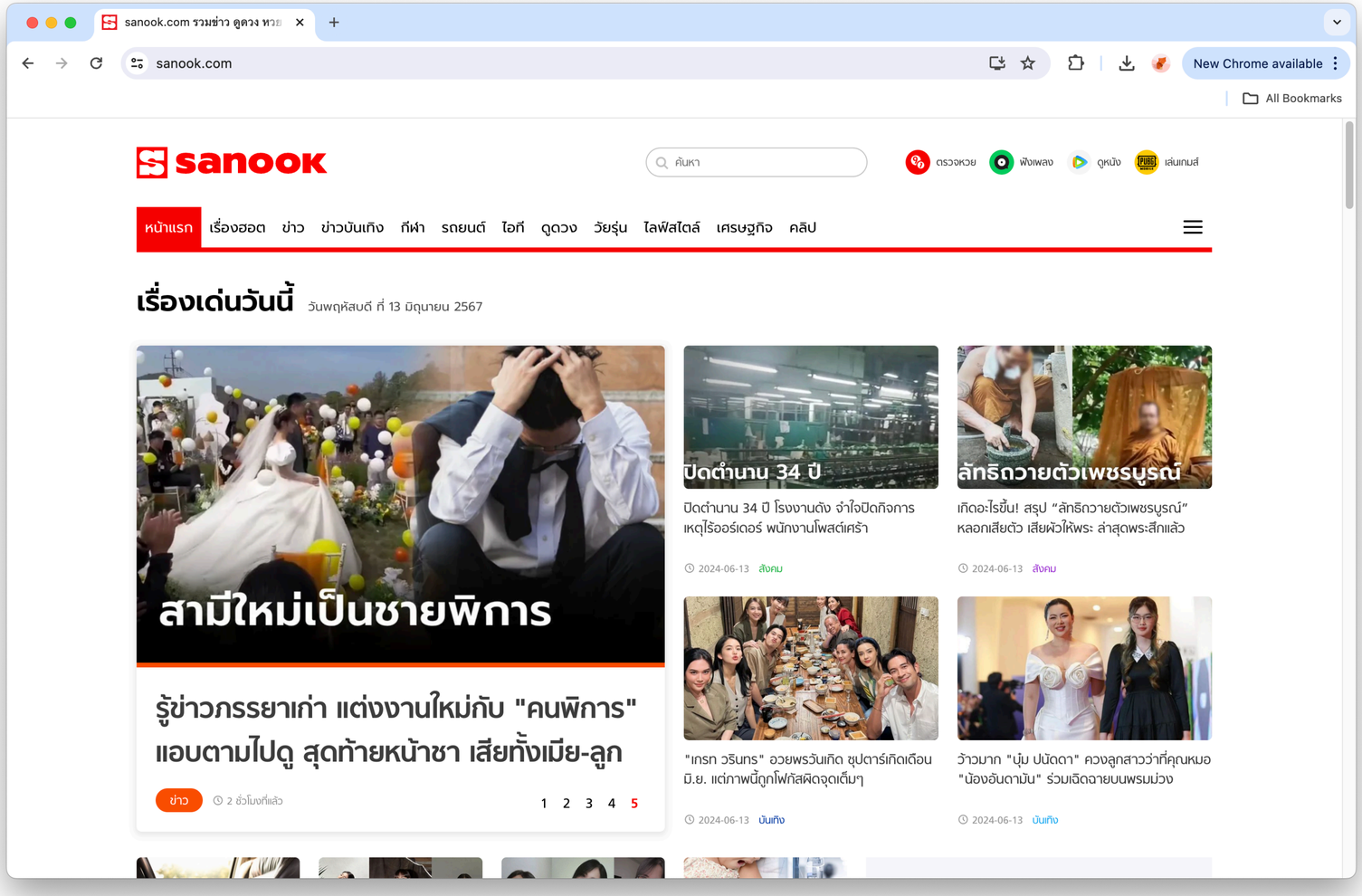
Text input

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

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('');
});
```

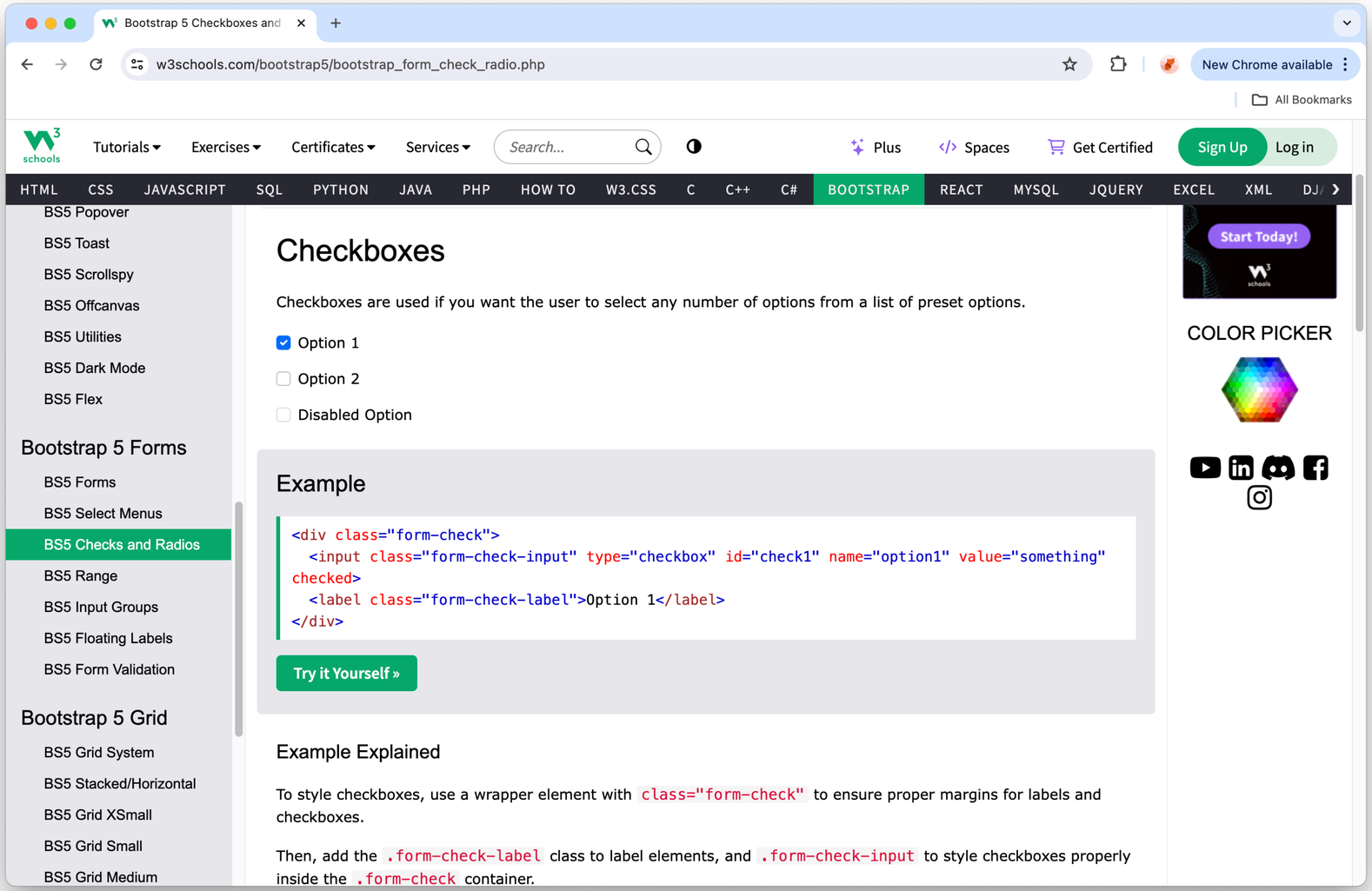
Checkbox

<input type="checkbox">

Checkbox

>> Elements > Check Box

Check! Check! Check! Uncheck! Uncheck!



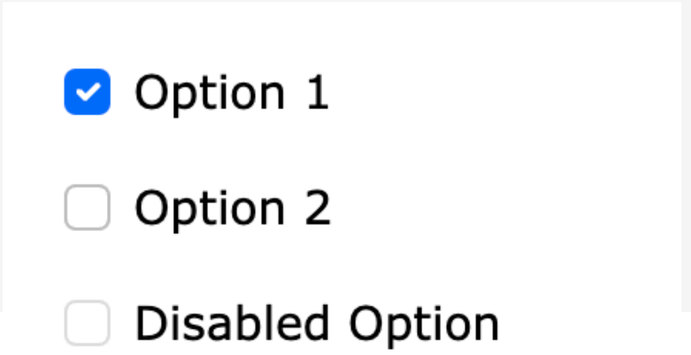
Frequently methods using with checkbox

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

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();`

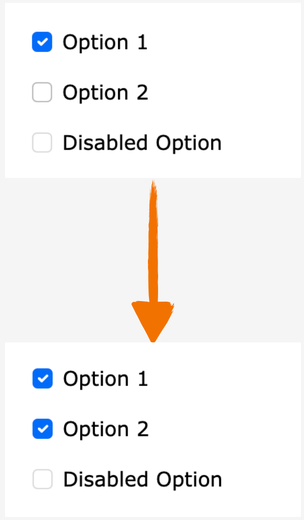
```
test("Check box : Is enabled check box", async ({ page }) => {
    // Arrange
    const url = "https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php";
    await page.goto(url);

    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();`

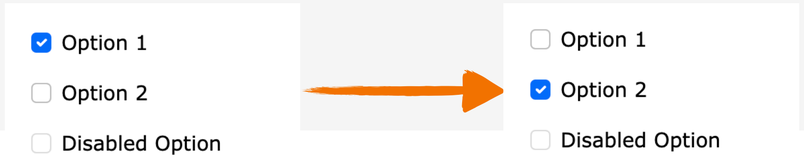
```
test("Check box : Check the box", async ({ page }) => {
    // Arrange
    const url = "https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php";
    await page.goto(url);

    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
        });
});
```



Playwright - Example code

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

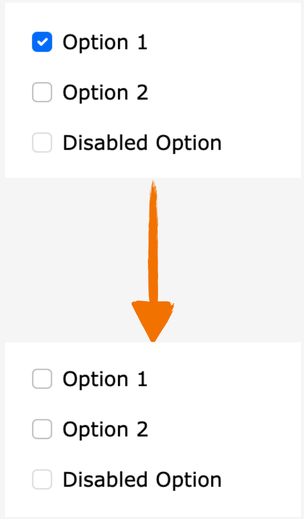
```
test("Check box : Set checked to the box", async ({ page }) => {
    // Arrange
    const url = "https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php";
    await page.goto(url);

    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
        });
});
```



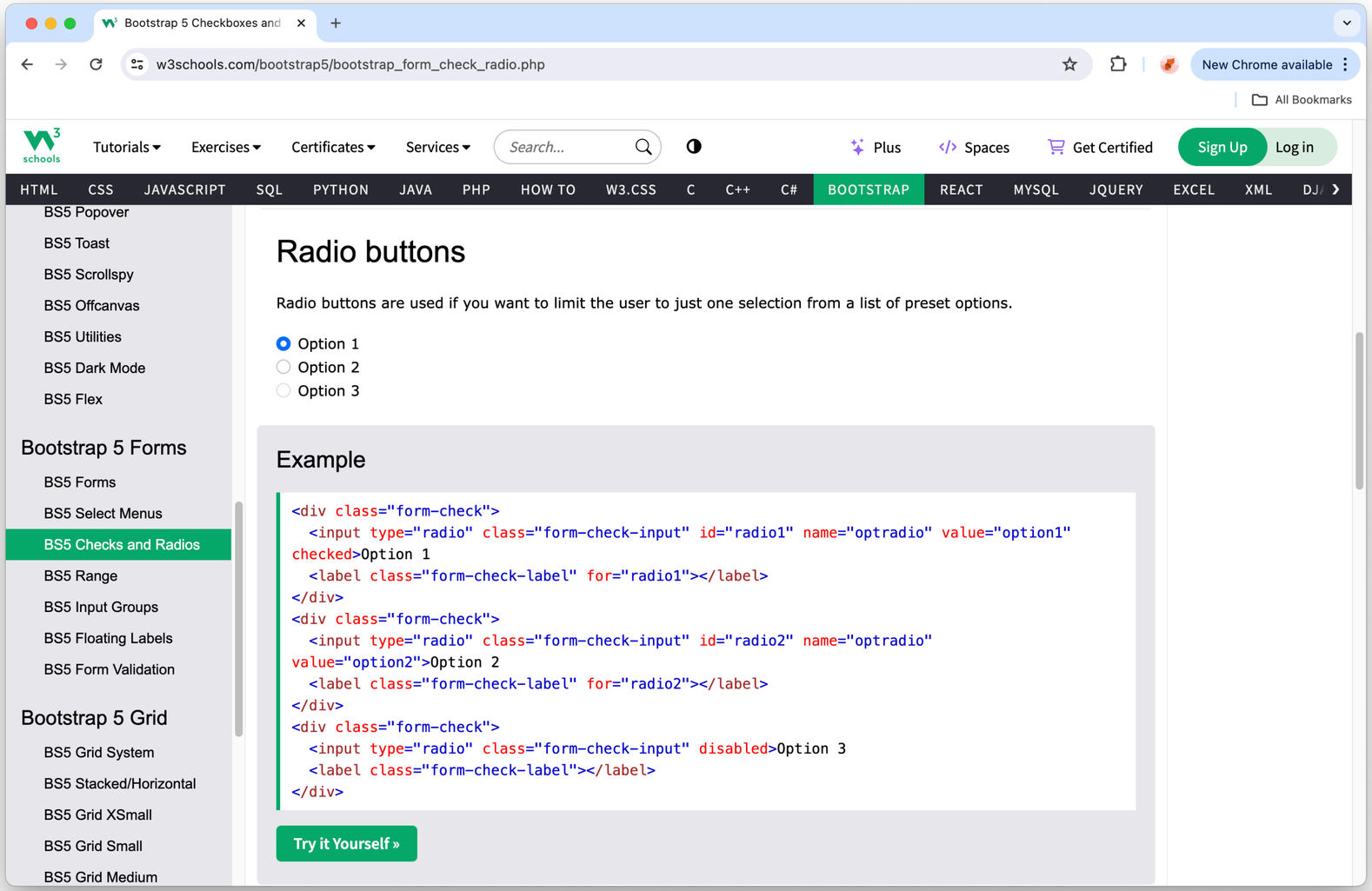
Radio button

<input type="radio">

Radio button

>> Elements >> Radio Button

You can select only one of it.



Frequently methods using with radio button

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

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
.setChecked( )	TRUE	`Promise<void>`	Set the state of a checkbox or a radio element to be checked.	`await locator.setChecked(true);`
	FALSE	`Promise<void>`	Set the state of a checkbox or a radio element to be unchecked.	`await locator.setChecked(false);`

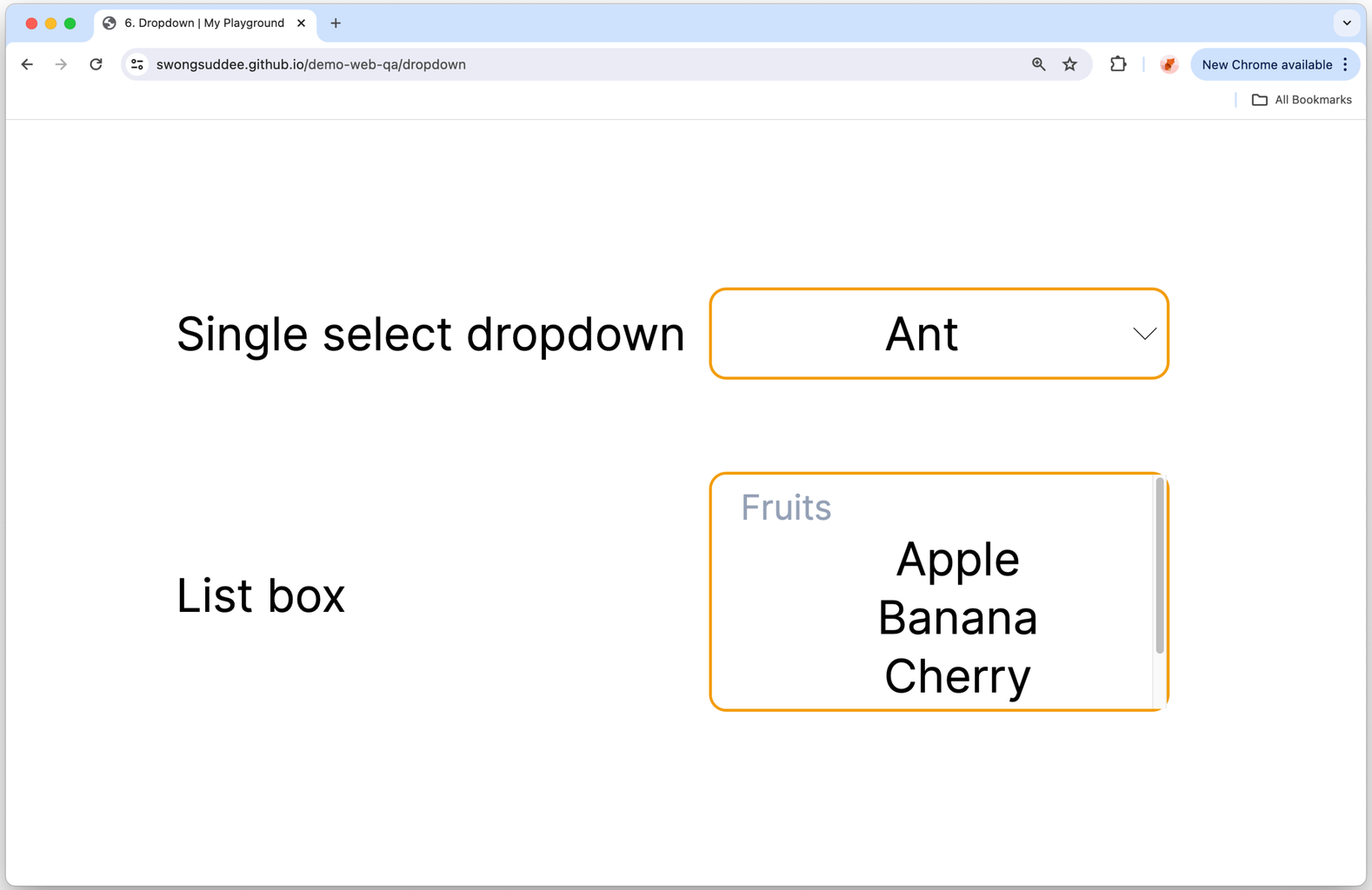
```
test("Radio button : Set checked to button", async ({ page }) => {
    // Arrange
    const url = 'https://www.w3schools.com/bootstrap5/bootstrap_form_check_radio.php';
    await page.goto(url);

    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

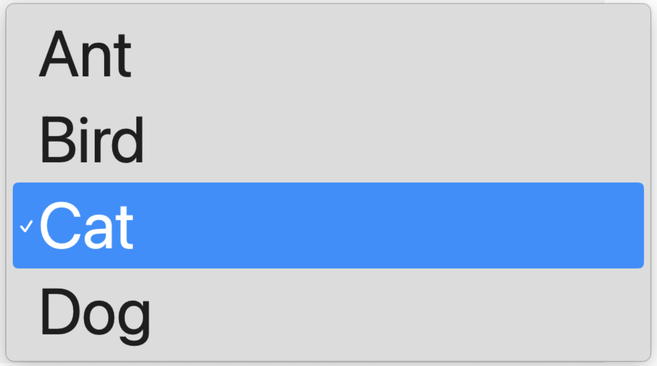
Method & Optionals		Return type	Purpose	Example
.selectOption( )	Value	`Promise< Array<string> >`	Selects option or options in `<select>` matches by `option.value`. Optional.	`const options = await locator.selectOption( 'red' );`
				`const options = await locator.selectOption( { value: 'red' } );`
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

```
Index 0 :    <option value="red">Red</option>
Index 1 :    <option value="1">Blue</option>
Index 2 :    <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.	`const options = await locator.selectOption( 'red' );`
				`const options = await locator.selectOption( { value: 'red' } );`

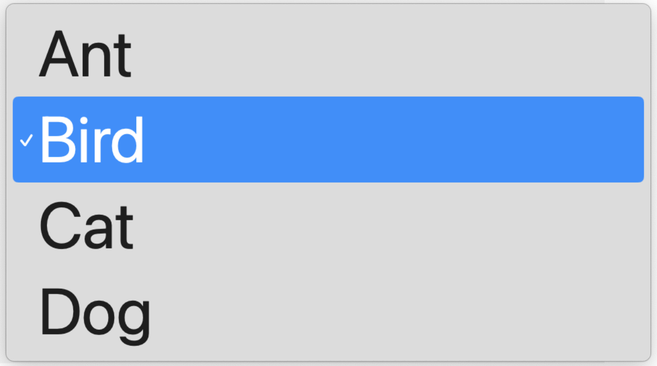
```
test("Select menu : Select from dropdown", async ({ page }) => {
    // Arrange
    const url = "https://swongsuddee.github.io/demo-web-qa/dropdown";
    await page.goto(url);

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

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

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



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.	`const options = await locator.selectOption( { label: 'Blue' } );`

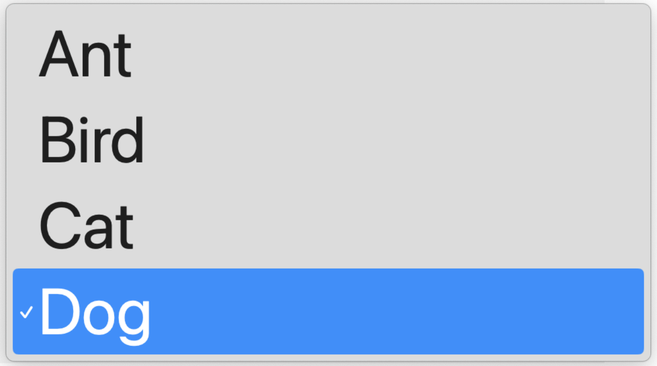
```
test("Select menu : Select from dropdown by value", async ({ page }) => {
    // Arrange
    const url = "https://swongsuddee.github.io/demo-web-qa/dropdown";
    await page.goto(url);

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

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

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



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.	`const options = await locator.selectOption( { index: 2 } );`

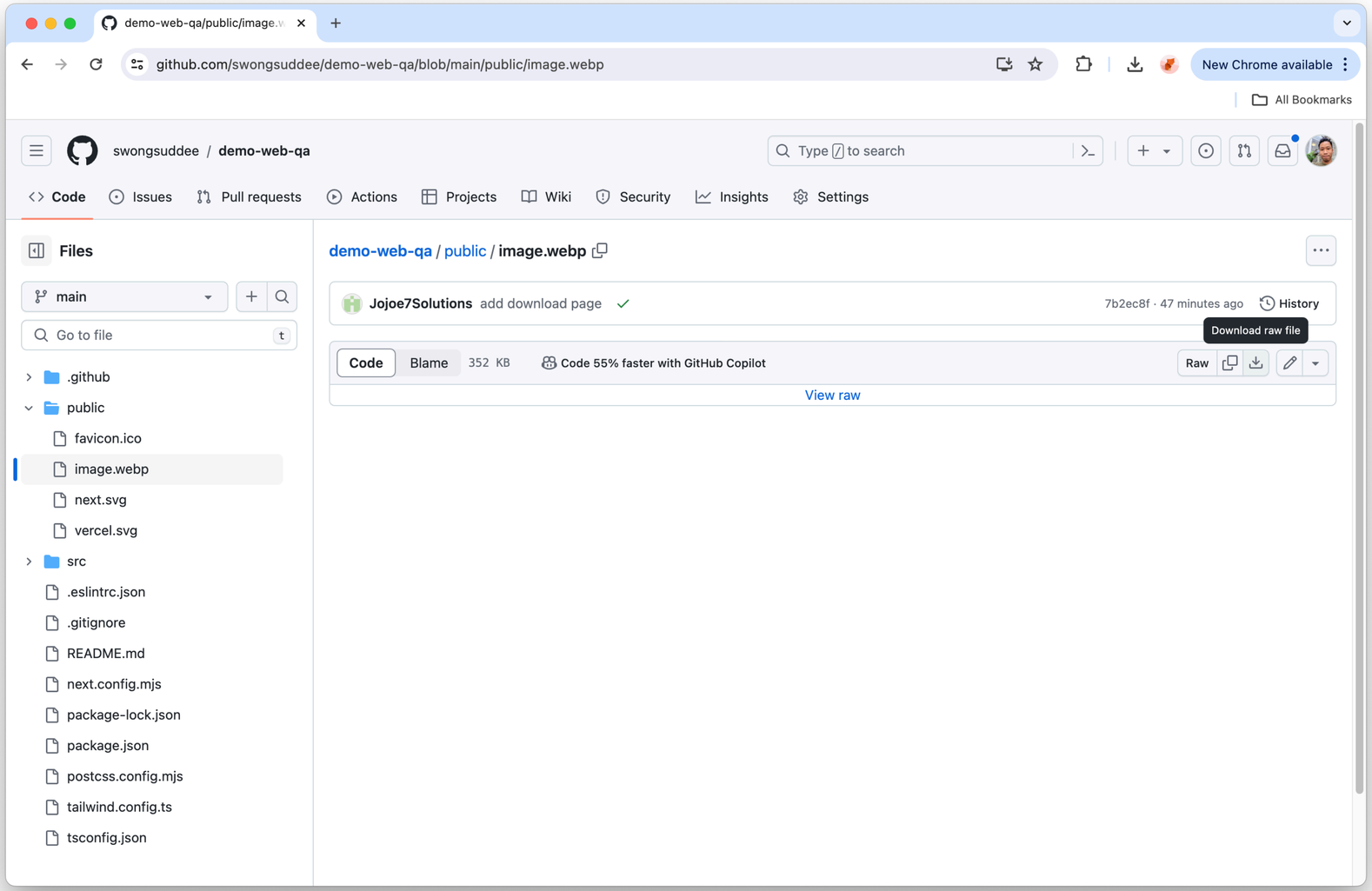
```
test("Select menu : Select from dropdown by index", async ({ page }) => {
    // Arrange
    const url = "https://swongsuddee.github.io/demo-web-qa/dropdown";
    await page.goto(url);

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

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

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



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

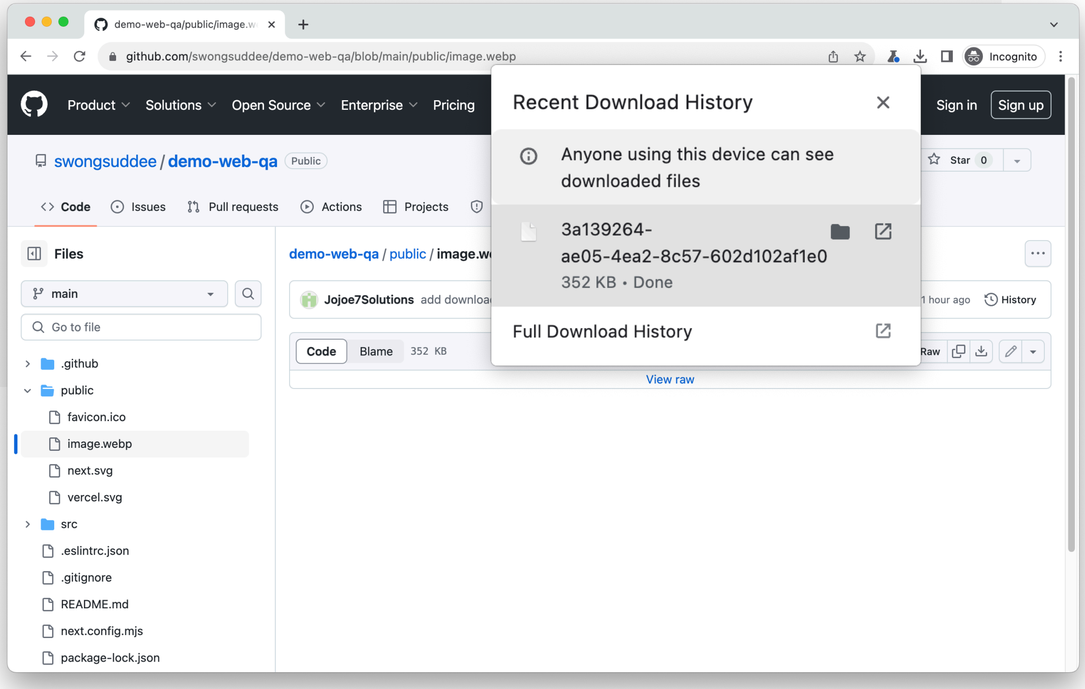
Method & Optionals	Return type	Purpose	Example

.waitForEvent( )	`Promise<Download>`	Emitted when attachment download started. User can access basic file operations on downloaded content via the passed Download instance.	`// Arrange`
			`const downloadPromise =`
			`page.waitForEvent('download');`
			
			`// Action`
			`await downloadButton.click();`
			
			`// Assertion`
			`const response =`
			`await downloadPromise;`

.suggestedFilename( )	`string`	Returns suggested filename for this download.	`const fileName =` `response.suggestedFilename();`

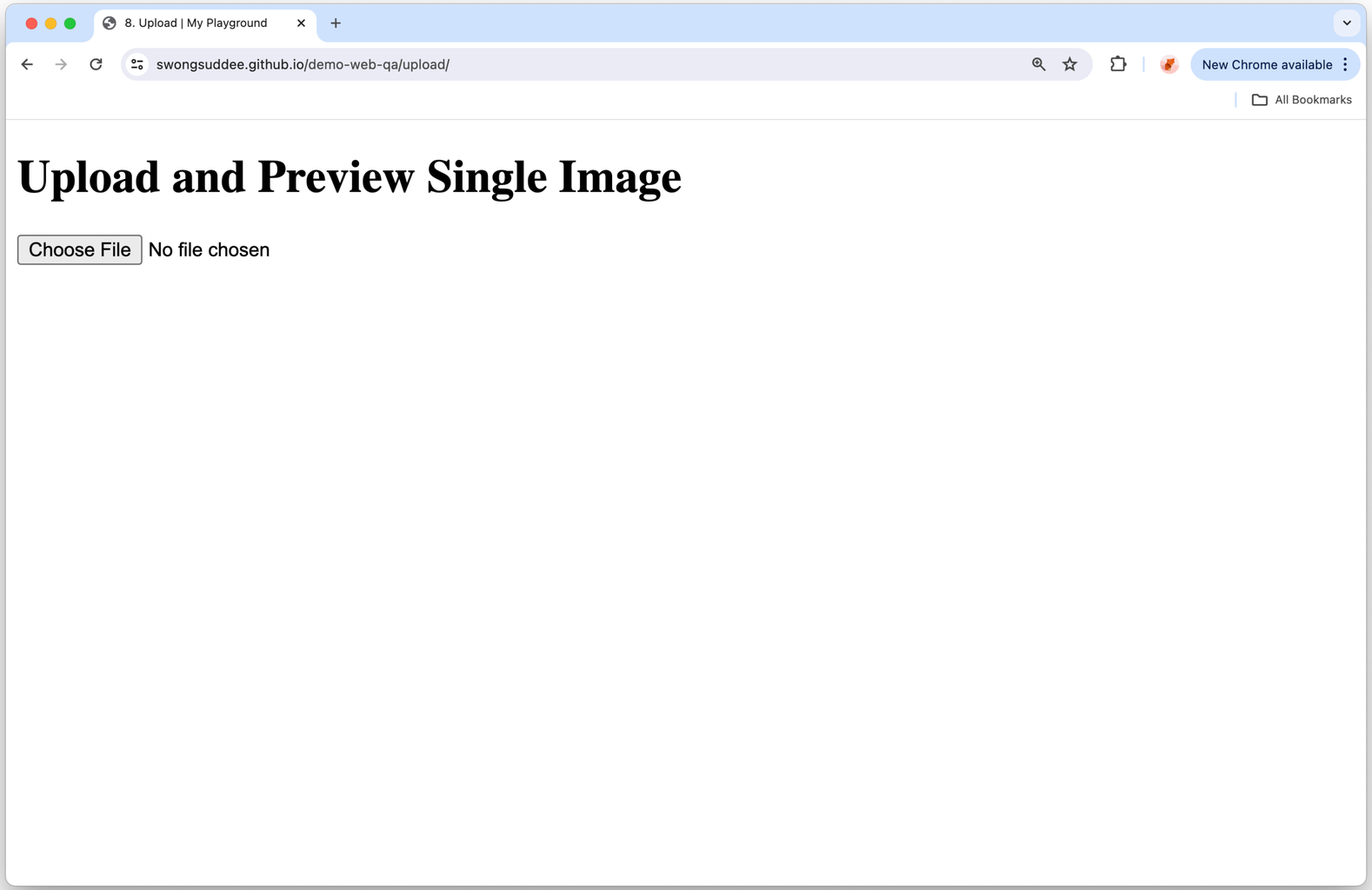
Playwright - Example code

```
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

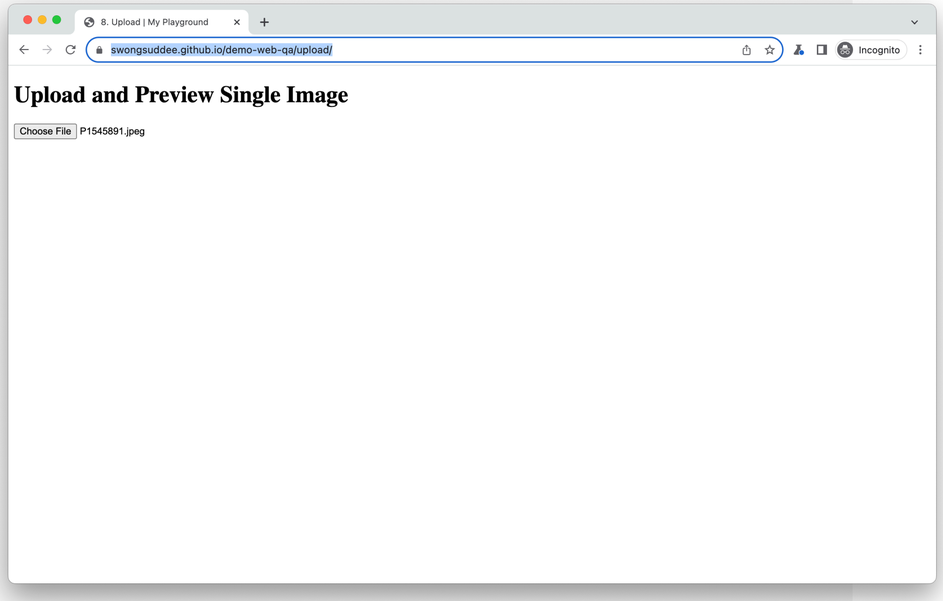
.setInputFiles( )	`Promise<void>`	Upload file or multiple files into `<input type=file>`.	`await locator.setInputFiles( 'File path' );`
			
			`await locator.setInputFiles( [ 'File path 1', 'File path 2', '…' ] );`

.inputValue( )	`Promise< string >`	Returns `input.value` for the selected `<input>` or `<textarea>` or `<select>` element.	`await locator.inputValue();`

Playwright - Example code

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

```
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

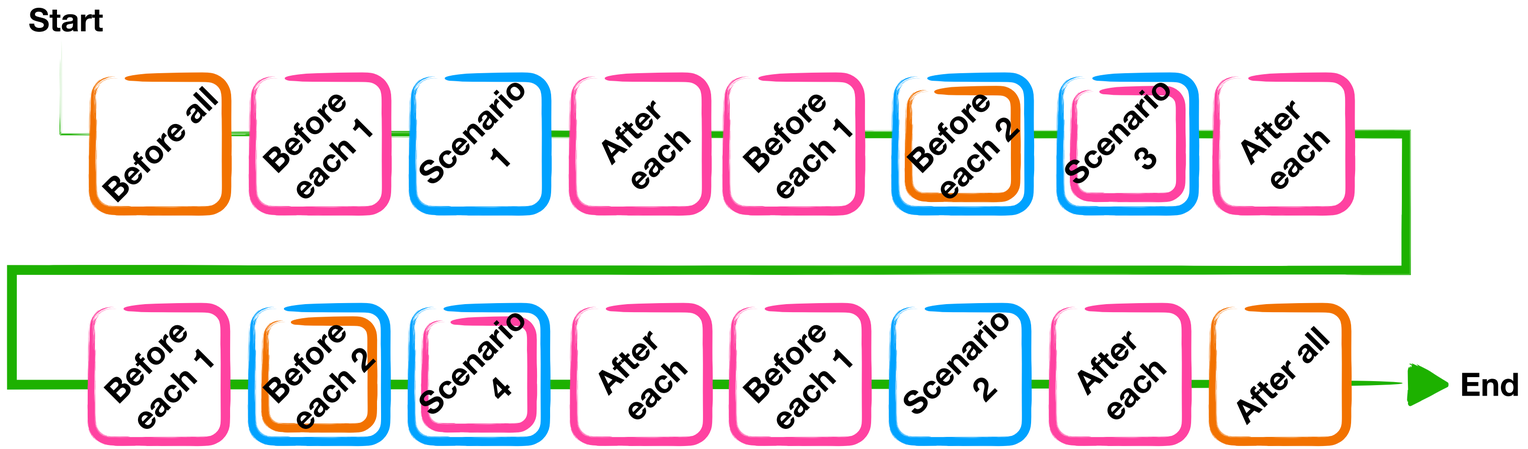
- Test setup / teardown
  - test.beforeAll( )
  - test.beforeEach( )
  - test.afterAll( )
  - test.afterEach( )
- Groups
  - test.describe( )
- Template

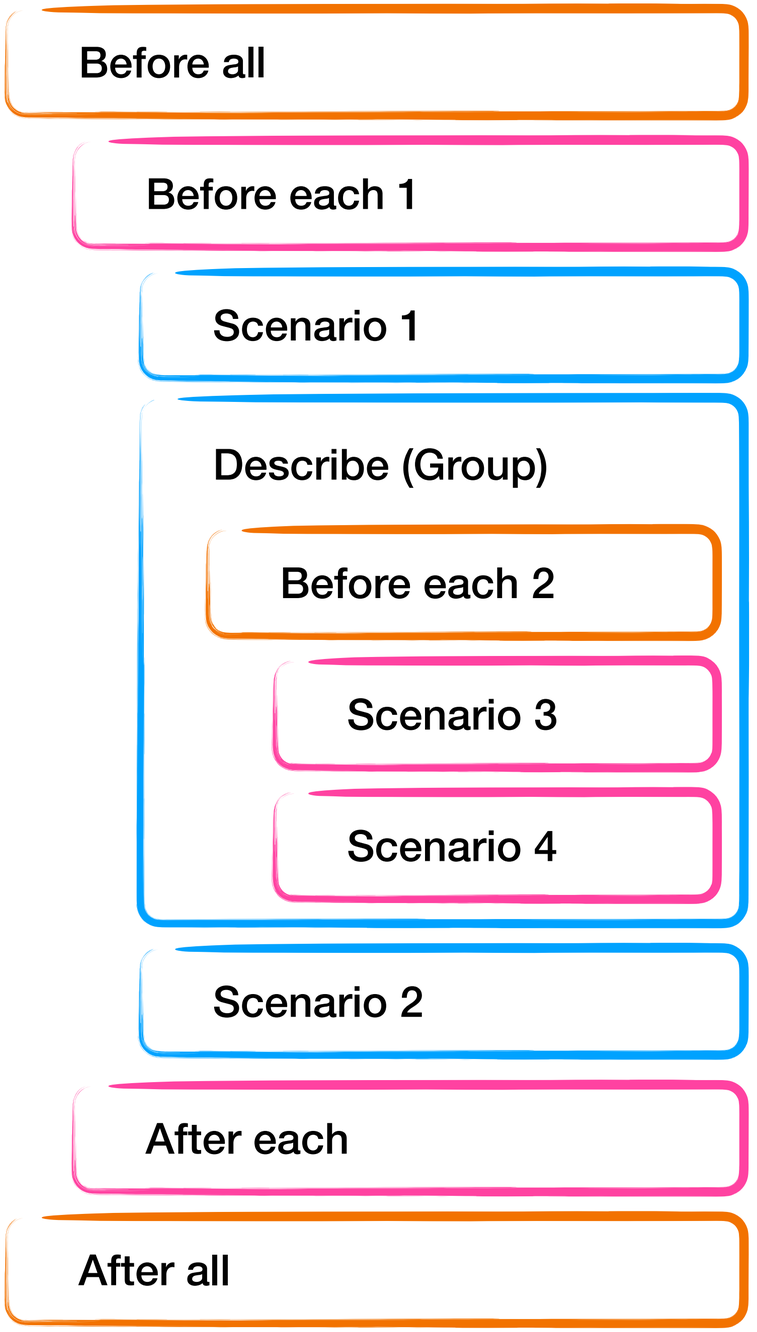
Ref1: https://playwright.dev/docs/api/class-test

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 () => { – 9ms
4      console.log("Before all");
5  });
6
7  test.beforeEach(async () => { – 6ms
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 () => { – 0ms
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
31 test("Scenario 2", async () => {
32     console.log("scenario 2");
33 });
34
35 test.afterEach(async () => { – 0ms
36     console.log("After each");
37 });
38
39 test.afterAll(async () => { – 5ms
40     console.log("After all");
41 });
42
```

Filter (e.g. text, !exclude)

```
/opt/homebrew/bin/node ./node_module
laywright.config.ts /Users/jojoe/Rep
tests/test-suite-1\.spec\.ts --heade
1 --retries 0 --timeout 0 --workers

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 () => { – 9ms
4      console.log("Before all");
5  });
6
7  test.beforeEach(async () => { – 6ms
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 () => { – 0ms
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
31 test("Scenario 2", async () => {
32     console.log("scenario 2");
33 });
34
35 test.afterEach(async () => { – 0ms
36     console.log("After each");
37 });
38
39 test.afterAll(async () => { – 5ms
40     console.log("After all");
41 });
42
```

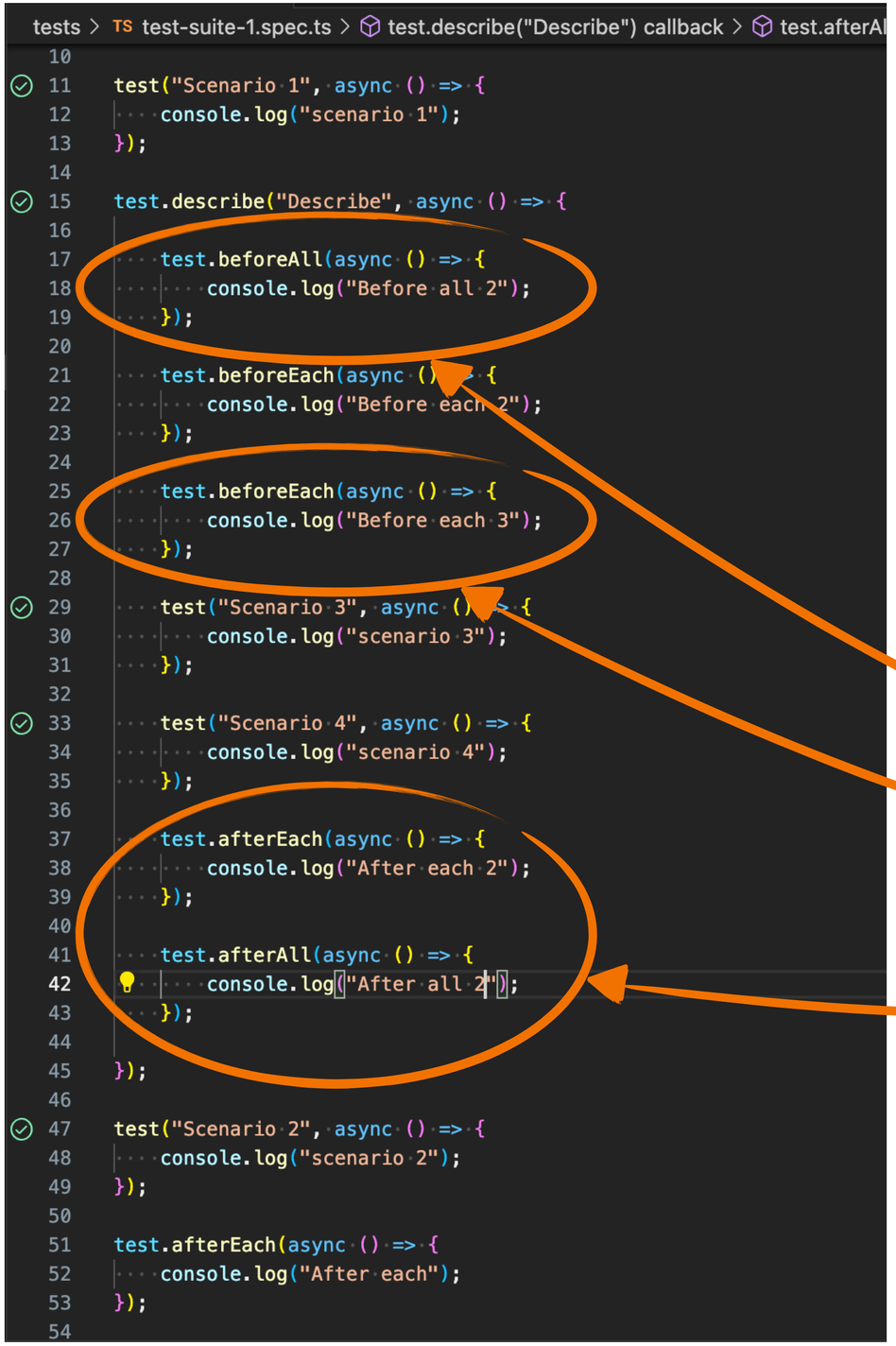
.beforeAll( ) will run at the beginning of the test suite execution

.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



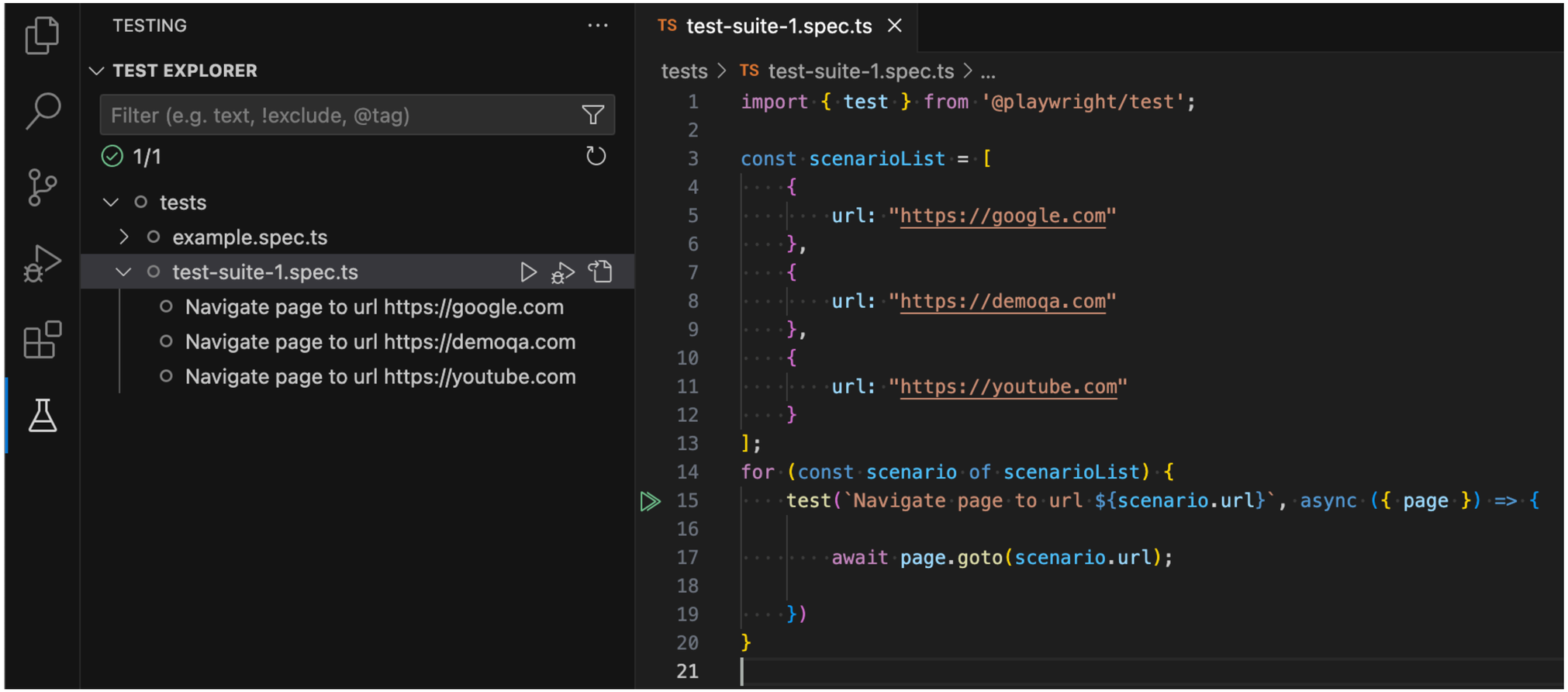
What if?

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



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

API tests

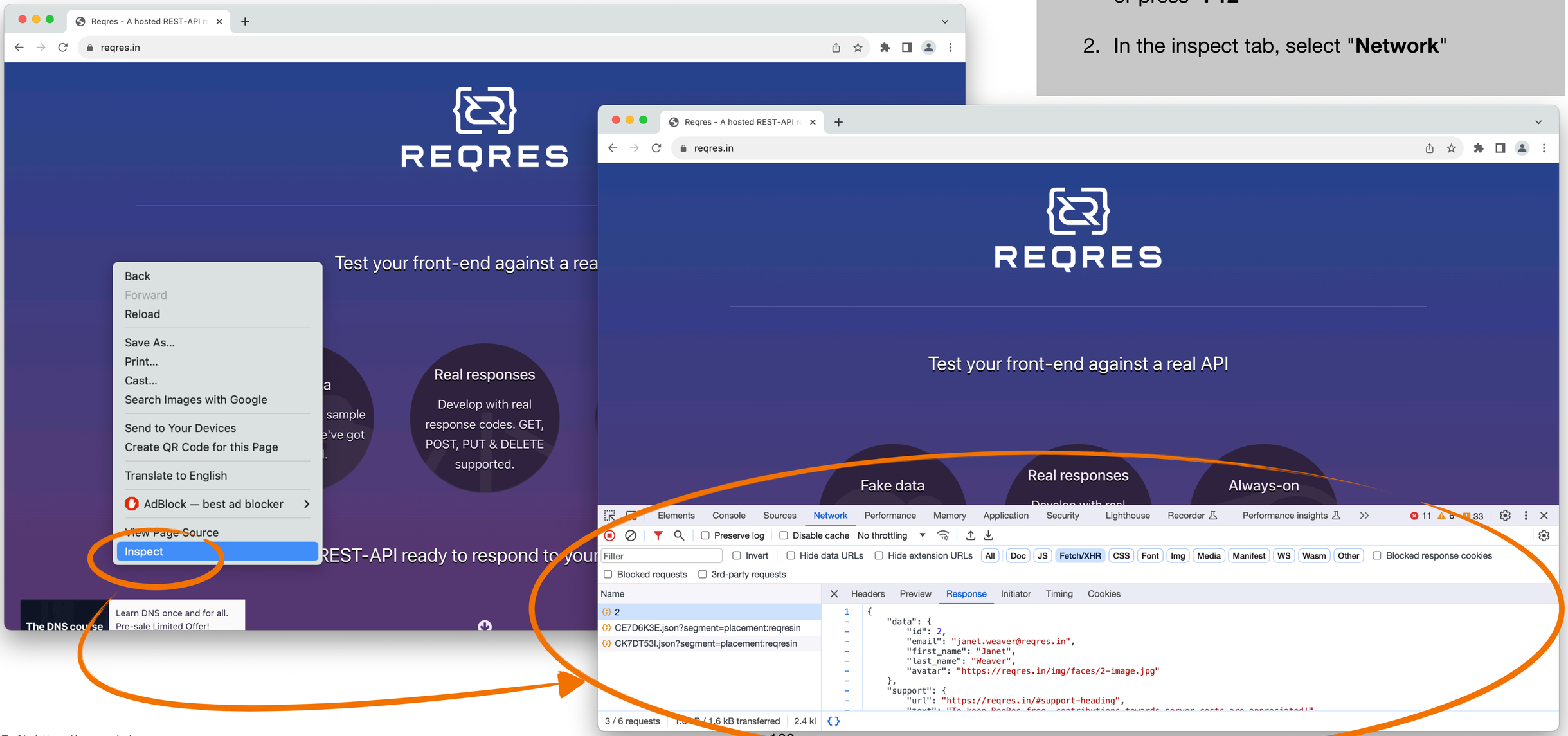
- Chrome inspect and network
- HTTP header
- Requests
  - GET
  - POST / PUT / PATCH
  - DELETE
- JSON parse

Ref1: https://playwright.dev/docs/api-testing

Chrome Inspect and Network

Open inspect and console steps:

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

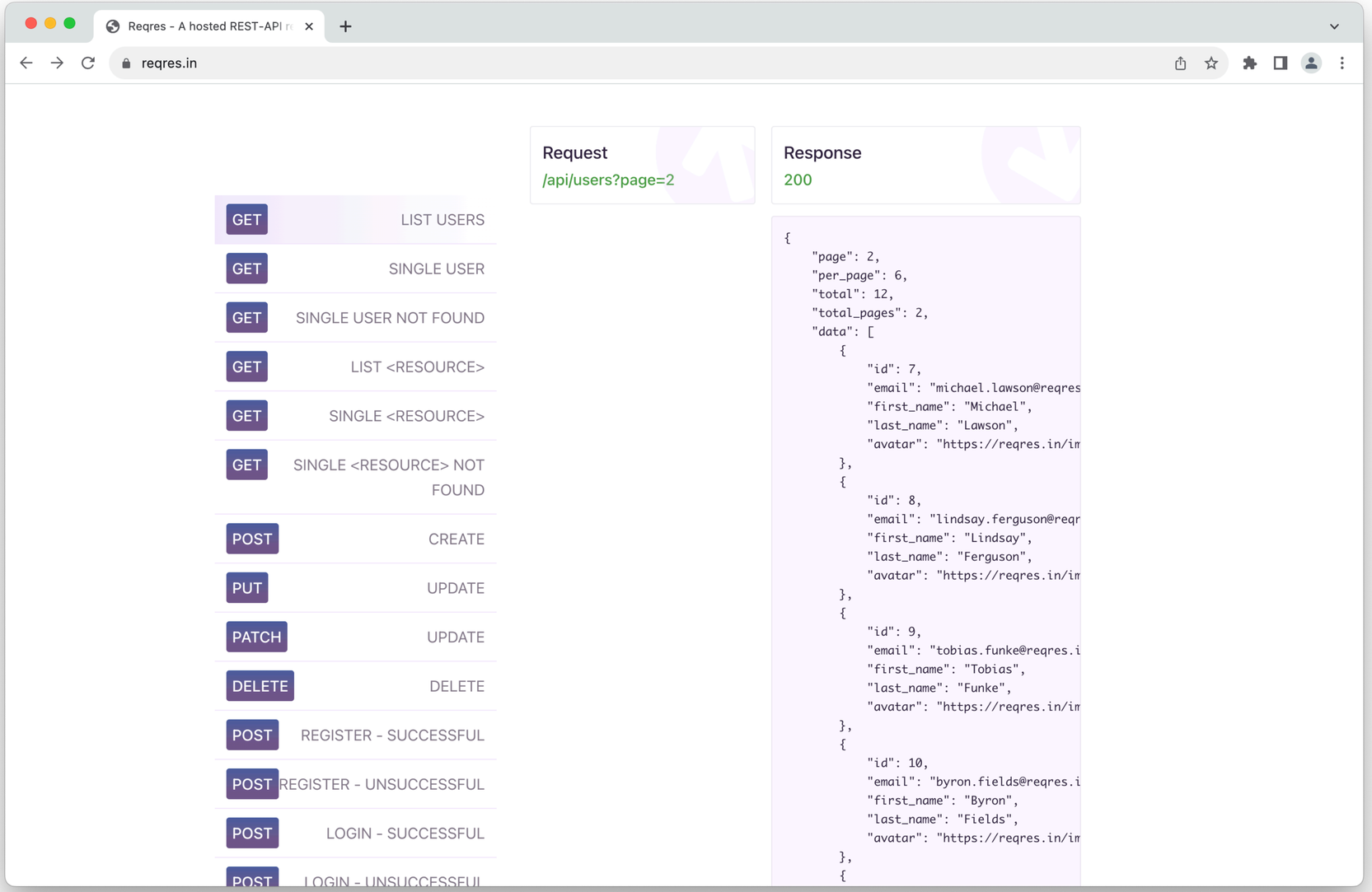


Ref1: https://reqres.in/

REQRES : New playground with fake data :/

Open Chrome browser

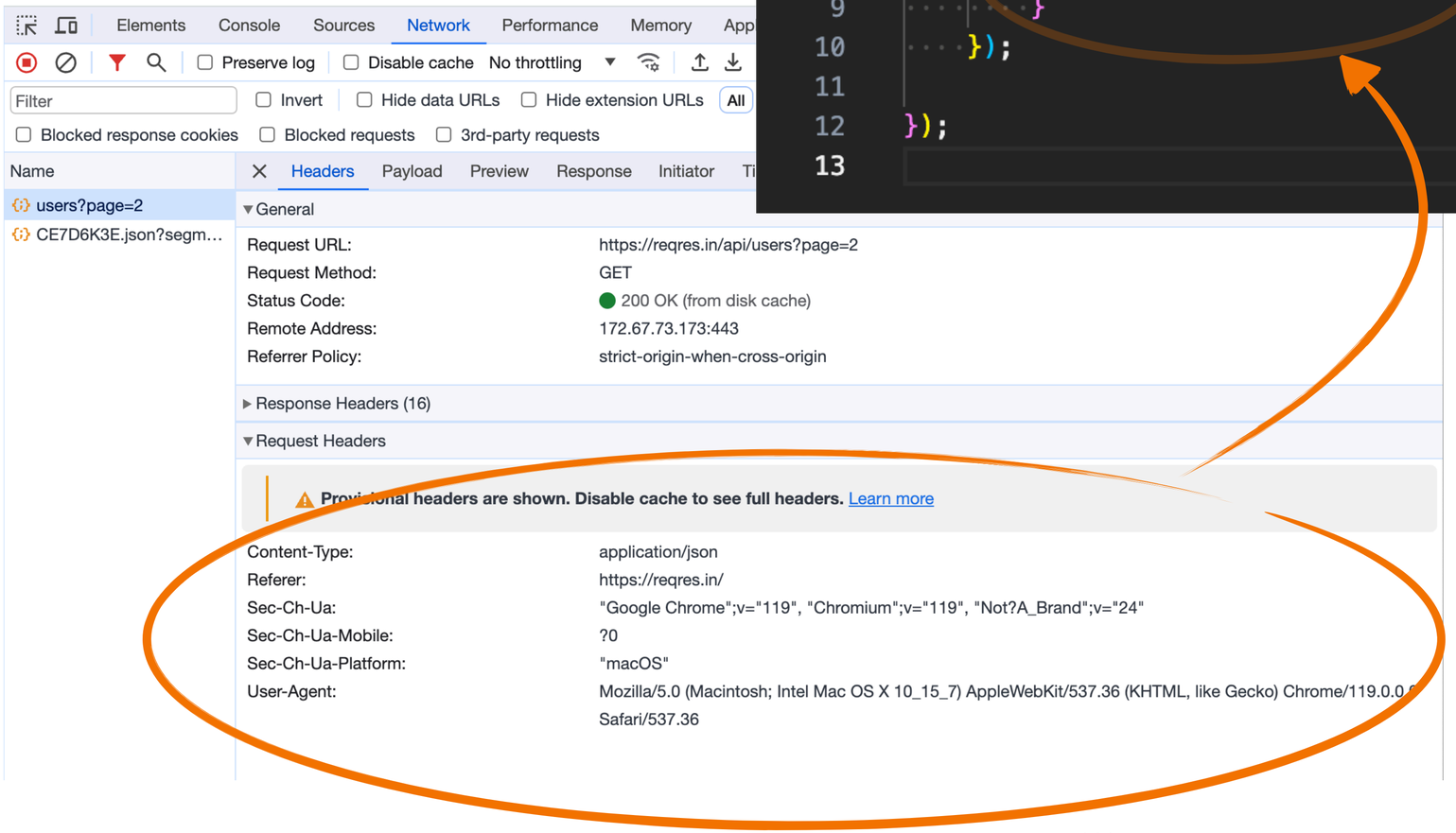
Go to this url: https://reqres.in/



Ref1: https://reqres.in/

HTTP header

```
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
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/2
13      const httpResponse = await httpContext.get("https://reqres.in/api/users/2");
14      console.log(`HTTP request GET response code: ${httpResponse.status()} ${httpResponse.statusText()}`);
15
16  });
17
```

```
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/jojoe/Repositorie
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)
```

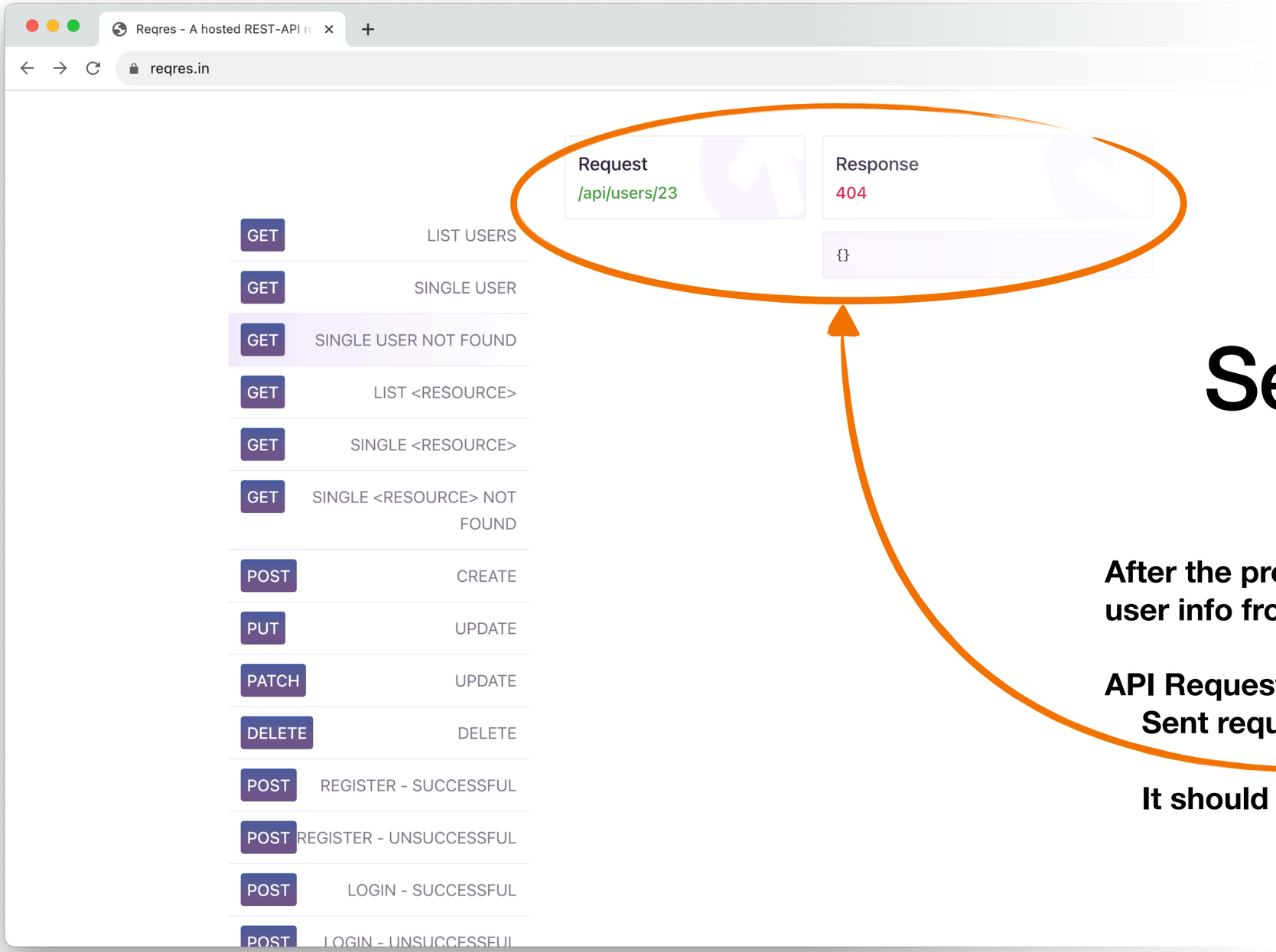
Test steps:

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



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: ${h
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 plo
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: ${ht
20
21  });
22
```

Test steps:

Line5 Create new HTTP context and add http header

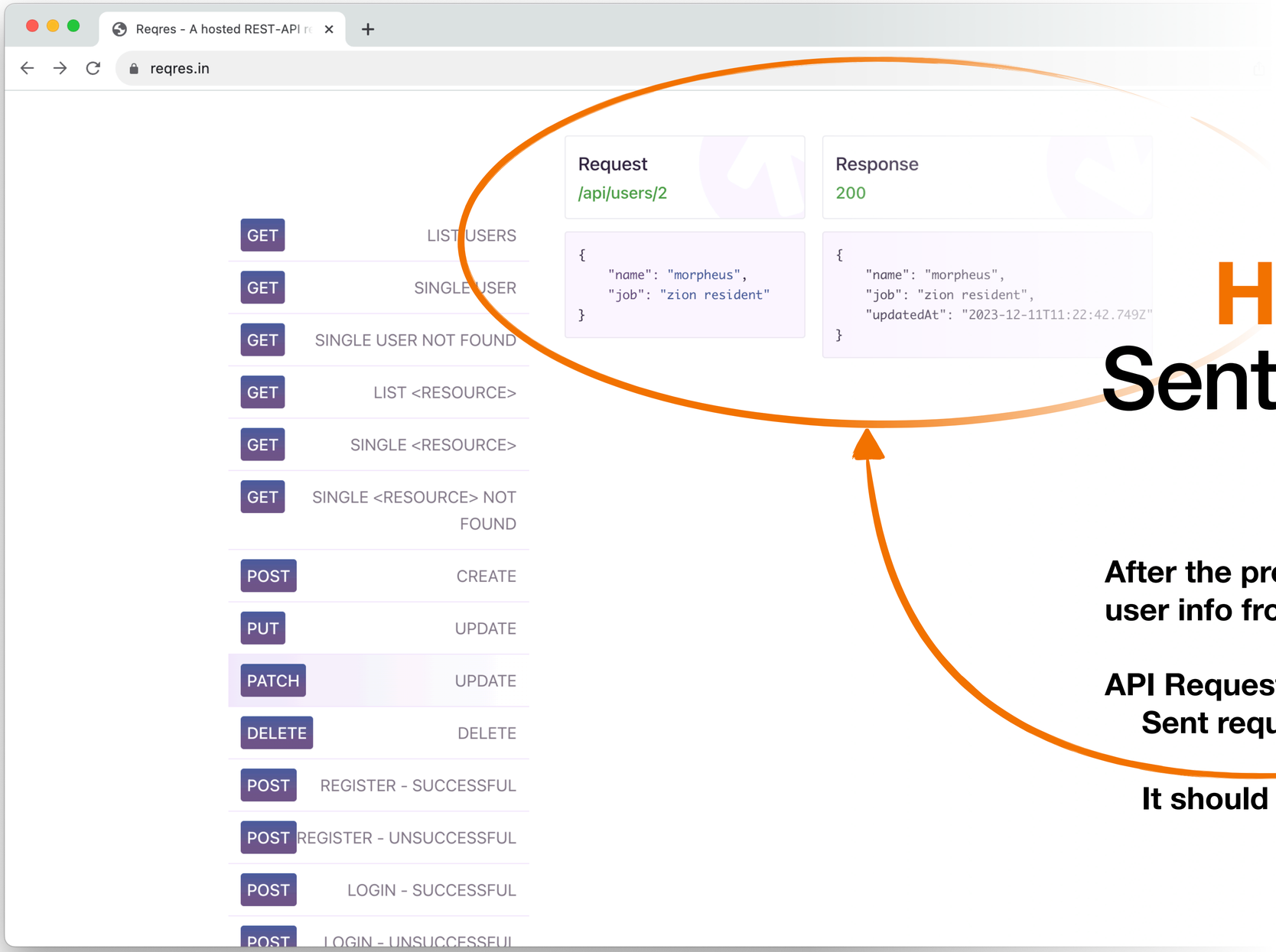
Line8 Add item name "Authorization" with empty string to header (In this case, no need any access token)

Line13 Sent request PUT with created HTTP context to url https://reqres.in/api/users

Line14 Add data payload to request POST

Line19 Log out response status and text to console

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TEST RESULTS  TERMINAL  PORTS  RC

/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)
```



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
```

```
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/jojoe/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)
```

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

JSON parse

Test steps:

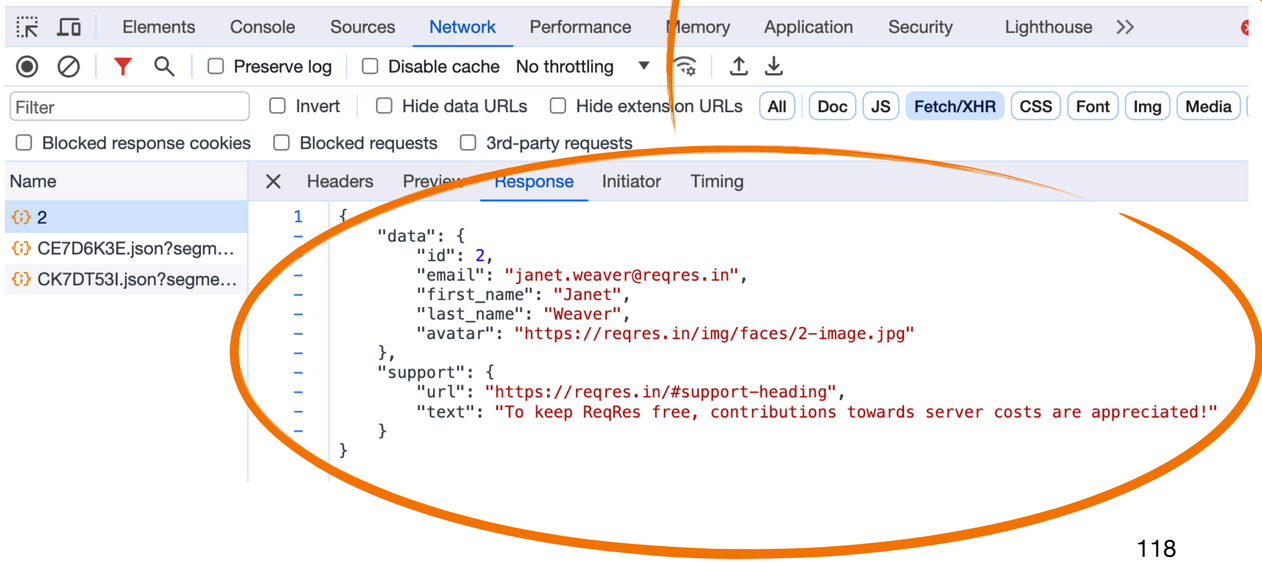
Line13 After receive the response of HTTP GET

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
```



Page object model

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

Ref1: https://playwright.dev/docs/pom

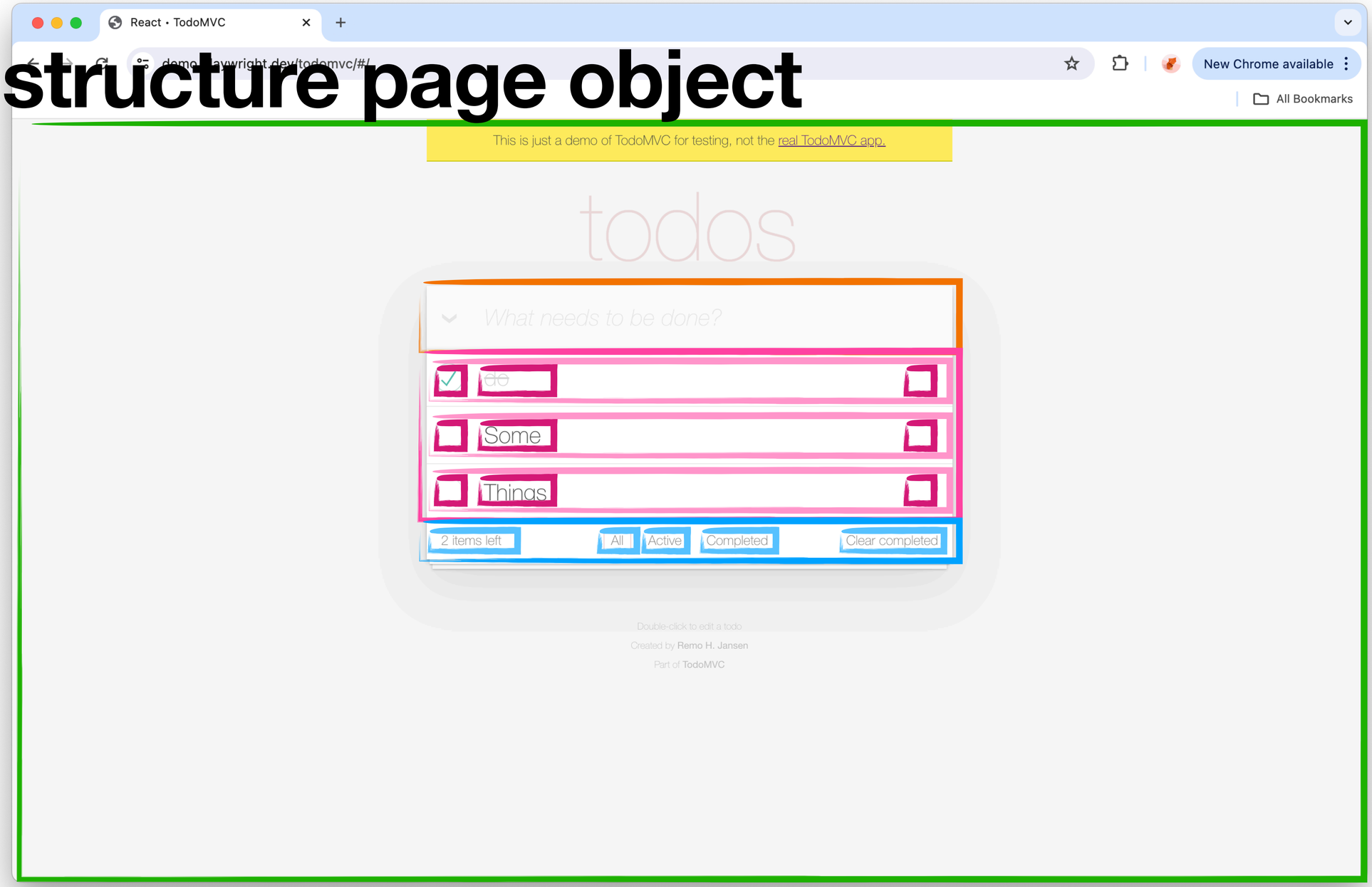
What is page object?

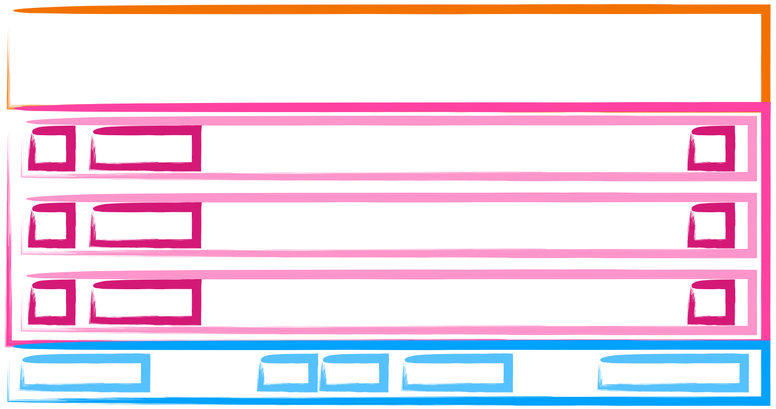
The Page Object Model (POM) is a design pattern commonly used in test automation to enhance the maintainability and readability of test scripts. While the Page Object Model is not specific to any particular testing framework or tool, it can be applied to various technologies and platforms, including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

Ref1: OpenAI. (2023). "What is idea of page object model in Playwright."
Ref2: https://playwright.dev/docs/pom

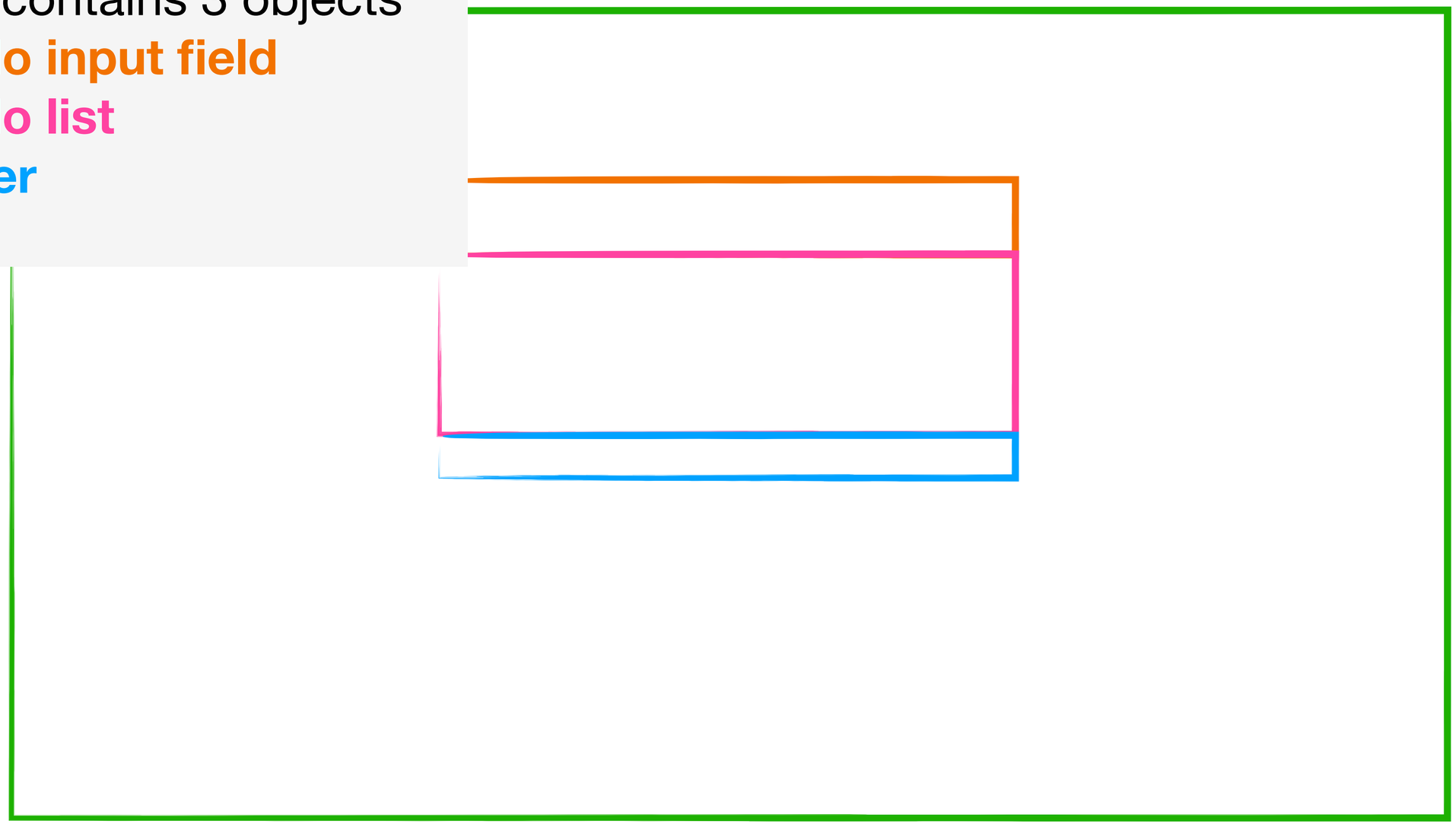
De-structure page object





Page contains 3 objects

- **Todo input field**
- **Todo list**
- **Filter**



Playwright - Example code

Page contains 3 objects

- **Todo input field**
- **Todo list**
- **Filter**

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

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

    constructor(page: Page) {
        this._page = page;
        this.input = new InputComponent(this._page, this._page.locator('css=input.new-todo'));
        this.todoList = new TodoList(this._page, this._page.locator('css=ul.todo-list'));
        this.filter = new Filter(this._page, this._page.locator('css=footer.footer'));
    }
}
```

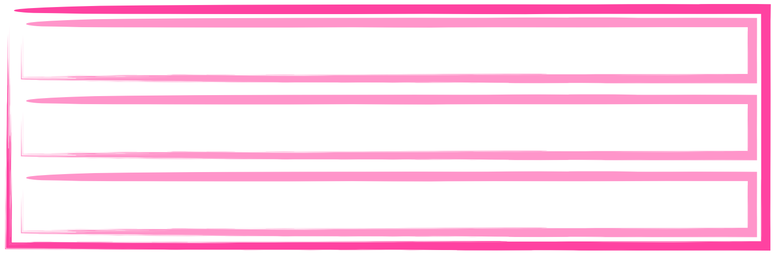
Todo input field

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;

    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;
    }
```

Todo list contains

- Todo items

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.check();
    }

    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
    …
});
```

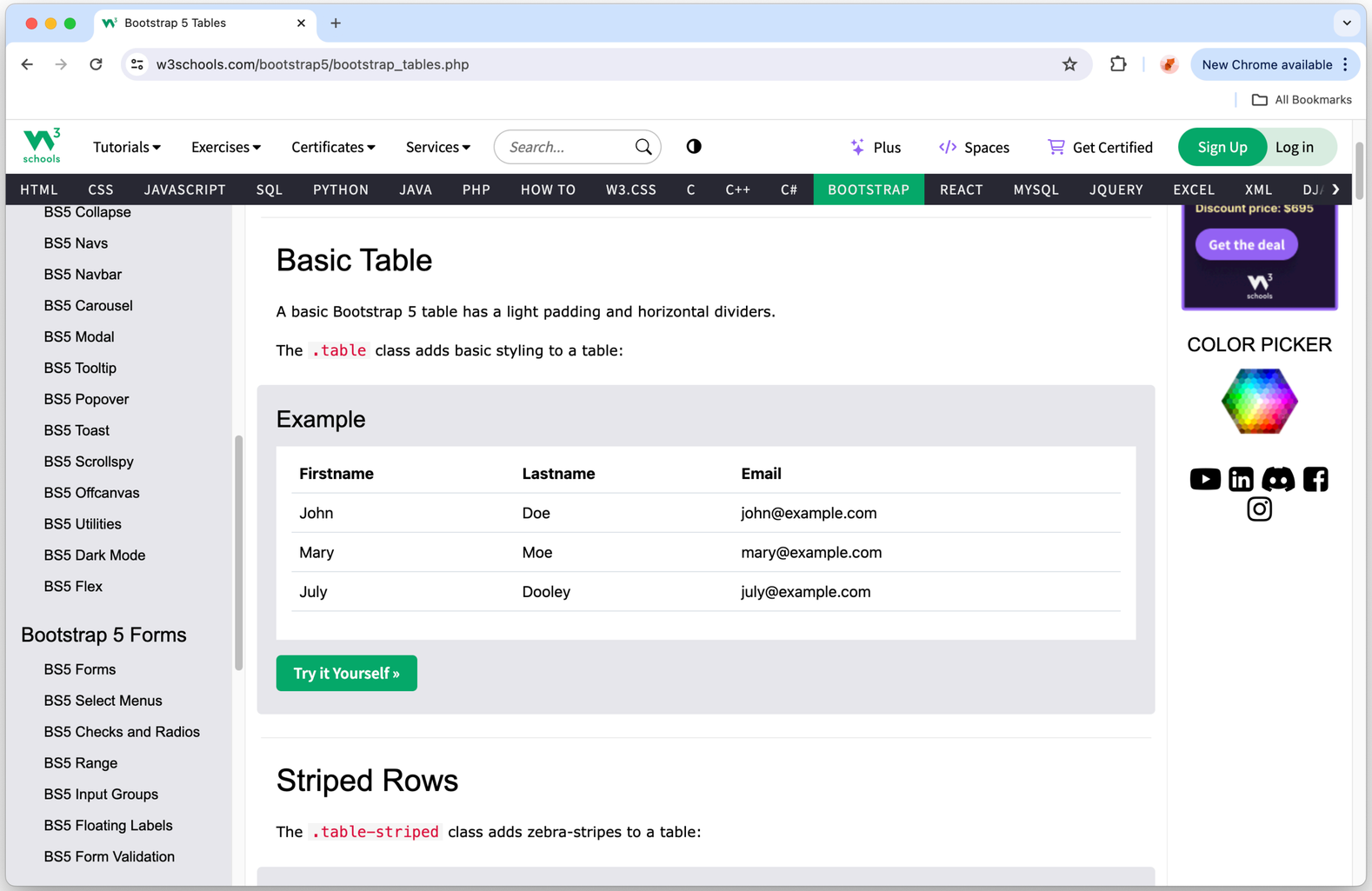
Scraping data

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

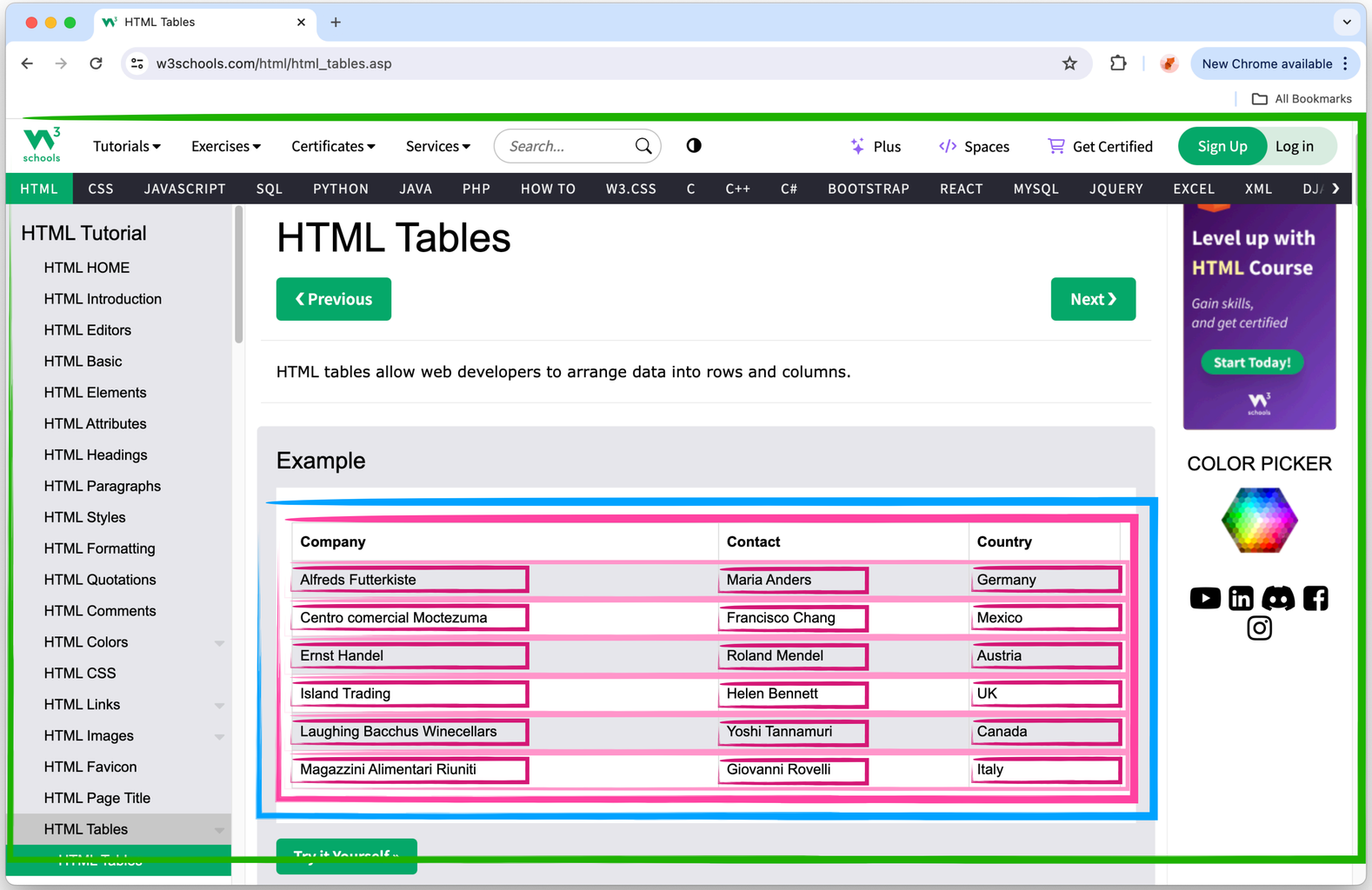
Ref1: https://playwright.dev/docs/api/class-elementhandle

Scrapping from table

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



De-structure page object

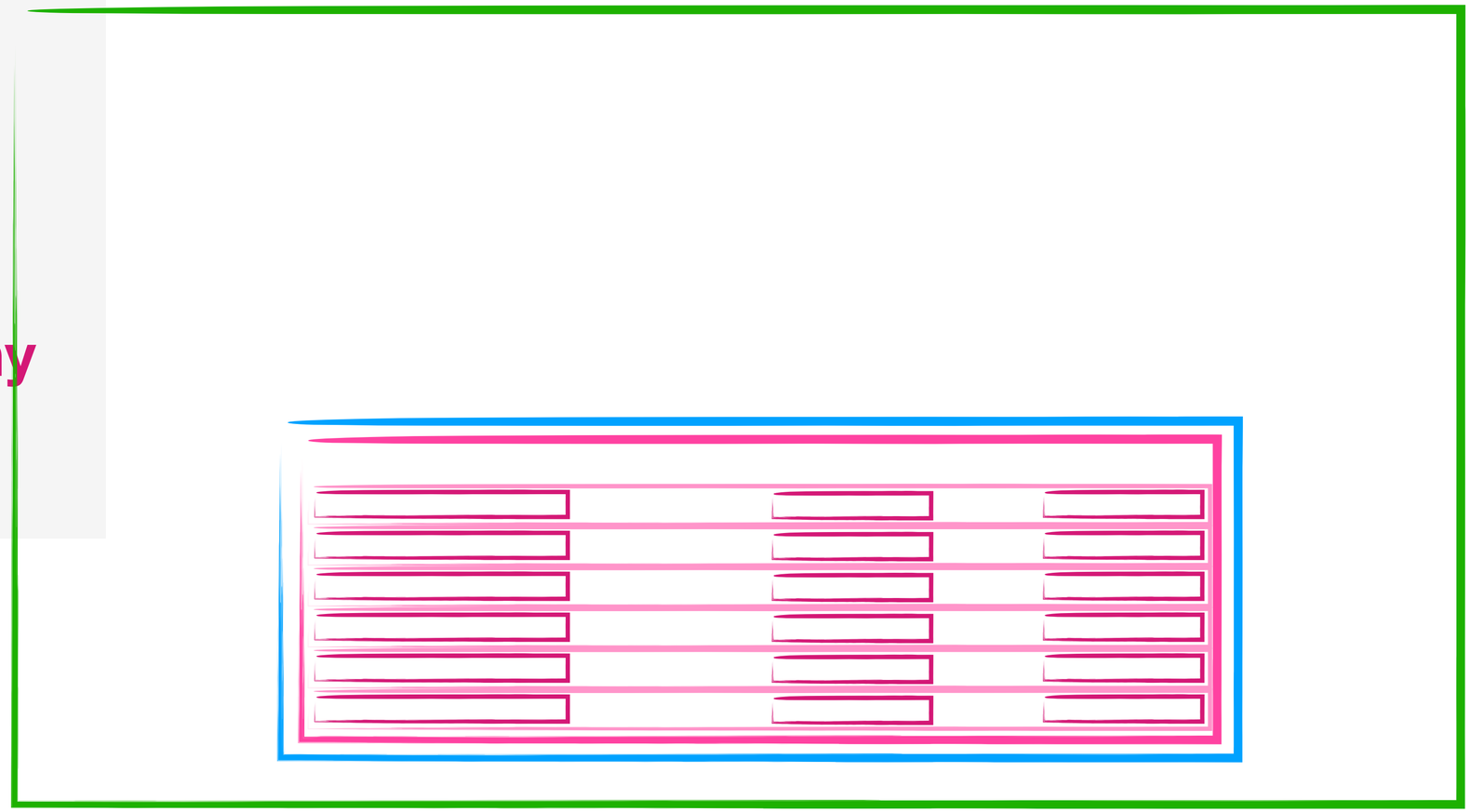


Page contains **Table**

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 getRowCount() {
        const count = await this.rowLocator.count();
        return count;
    }

    public async getAllRows() {
        const count = await this.getRowCount();
        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('Scrapping 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);
});
```

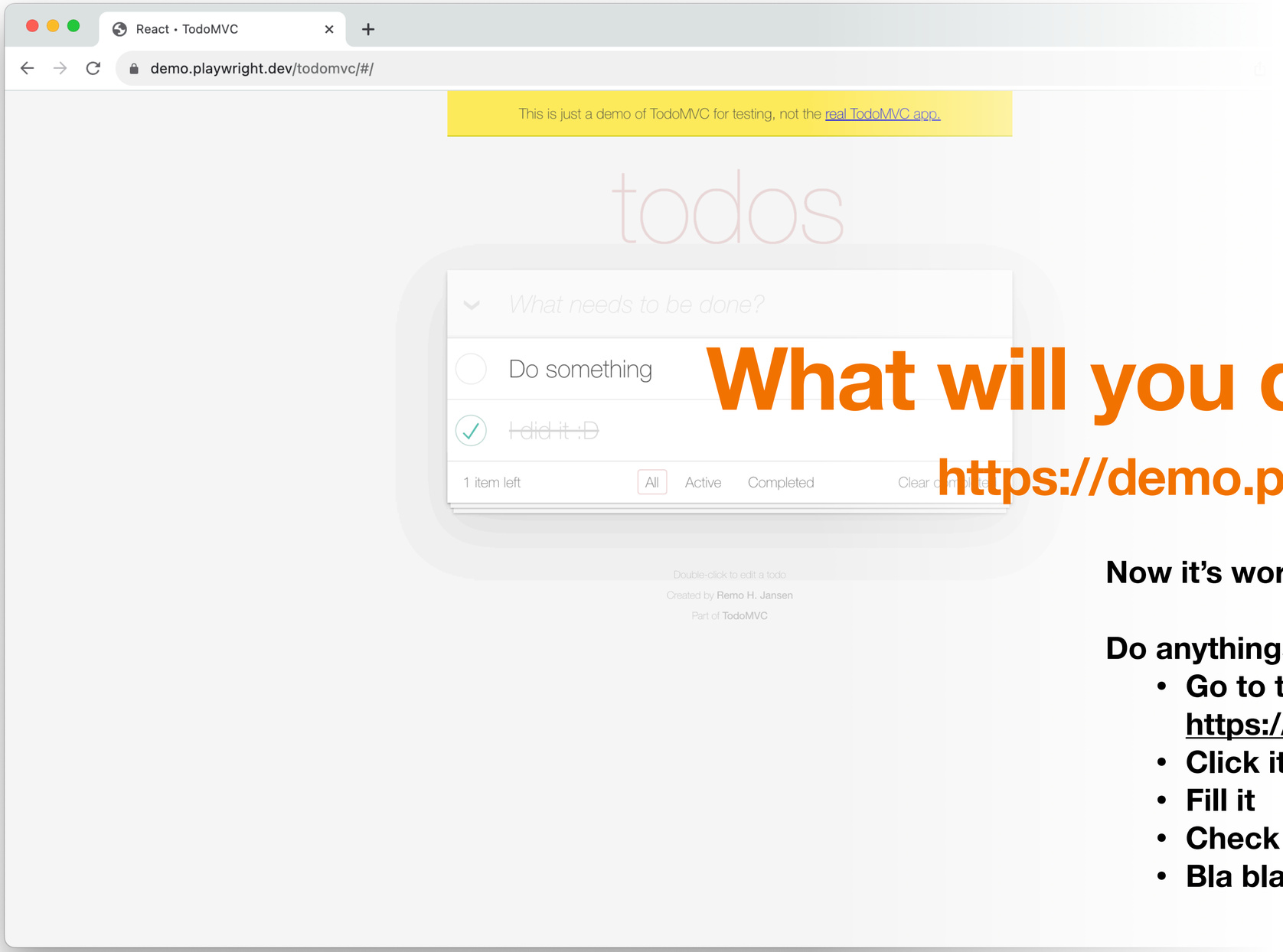
Console output

```
┌─────────┬──────────────────────────────┬──────────────────────┬─────────────┐
│ (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 anythings you want :)

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