APIs for Browser Automation

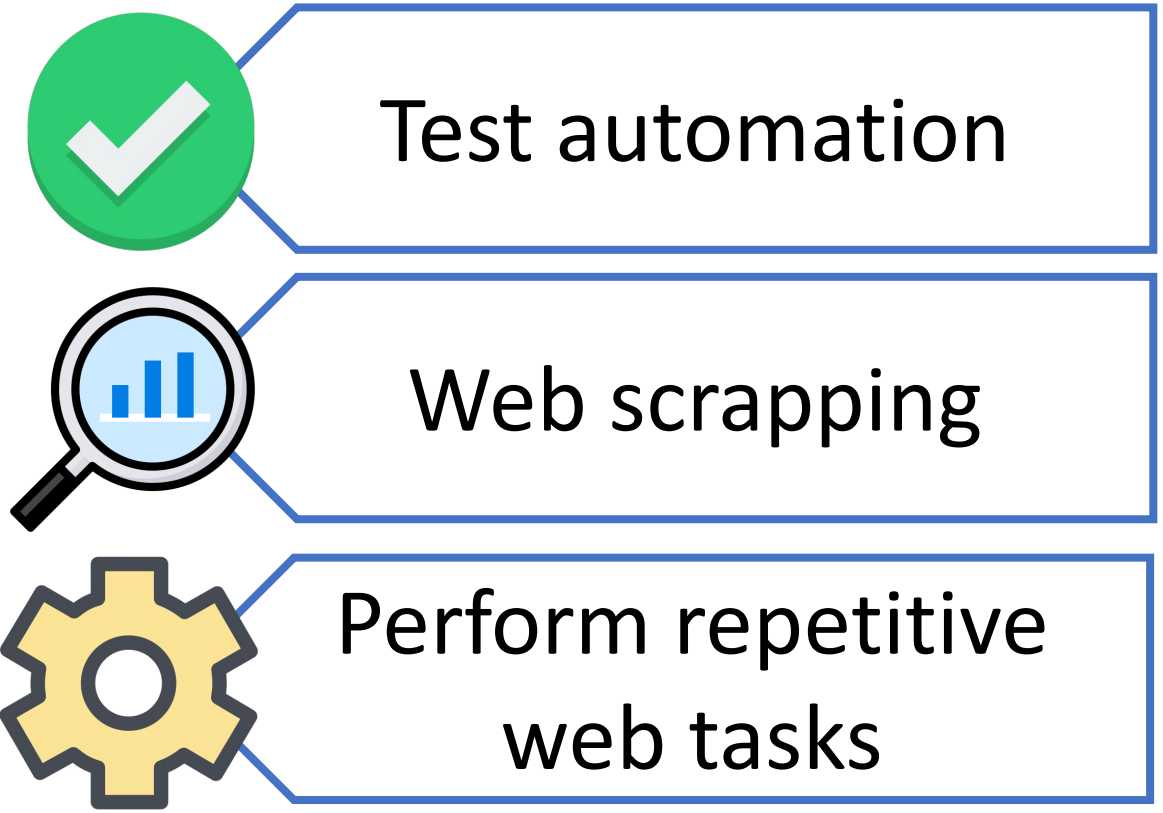
-Selenium, Cypress, Puppeteer, and Playwright-

Ministry of Testing Athens
Meetup on the Beach
30 May 2024

Boni García
https://bonigarcia.dev/

What is “Browser Automation”?

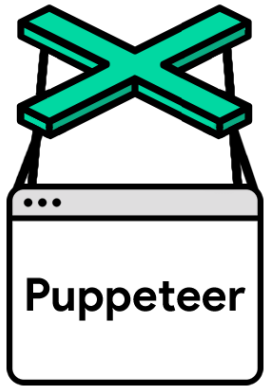
- **Browser automation** refers to the use of software to perform tasks in a web browser automatically
  - A common technique is through an **API** (i.e., programmatically)



Browser Automation – APIs

Selenium

cypress



Playwright

https://github.com/angrykoala/awesome-browser-automation

Selenium – What is Selenium?

- Selenium WebDriver (often known as simply **Selenium**) is a multilanguage **browser automation library**

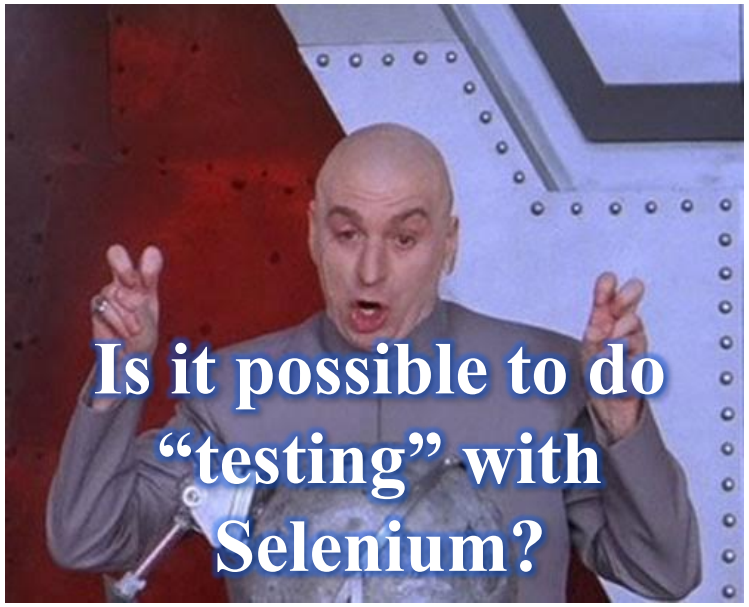
https://selenium.dev/

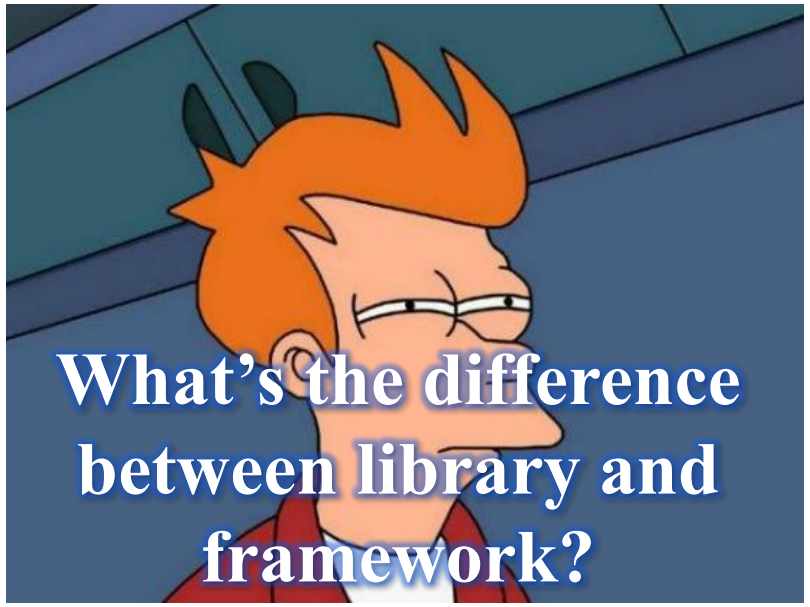
  - Maintained by the Selenium project since 2004
  - Languages: officially supported in Java, JavaScript, Python, .Net, and Ruby

  - Browsers: any browser with a driver compliant with W3C WebDriver

Selenium – What is NOT Selenium?

- Selenium is NOT a testing framework 
- Selenium is NOT a testing library 





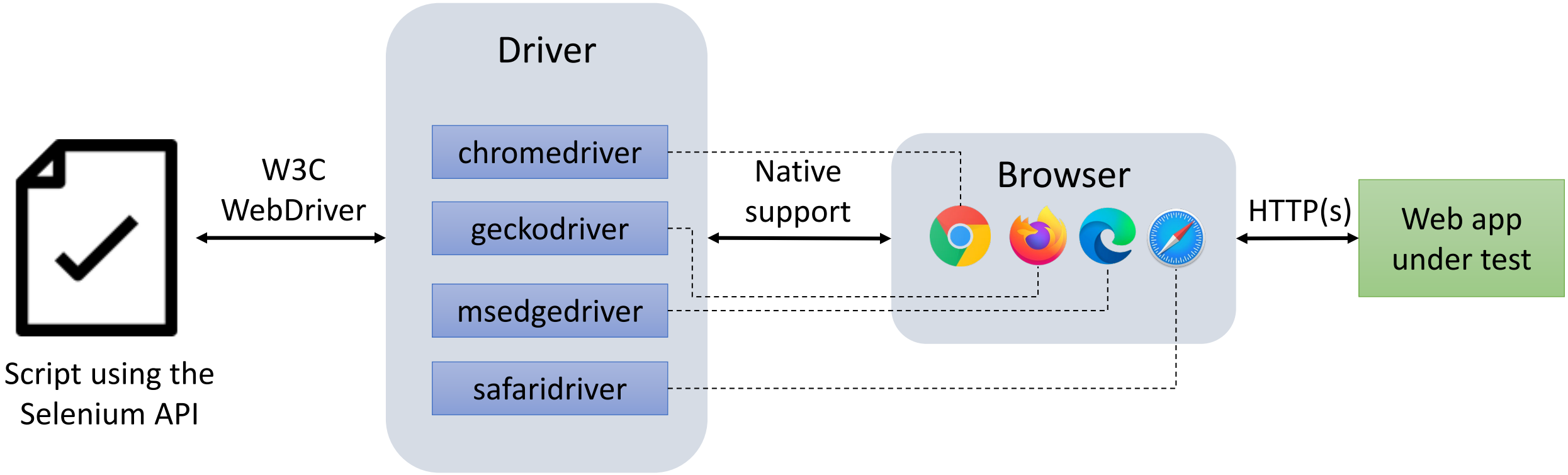
Library vs. Framework

- A **library** is a collection of code that developers can call using an API to solve a given problem

- A **framework** is a library that that provides a foundational structure for developing software applications

Selenium – Architecture

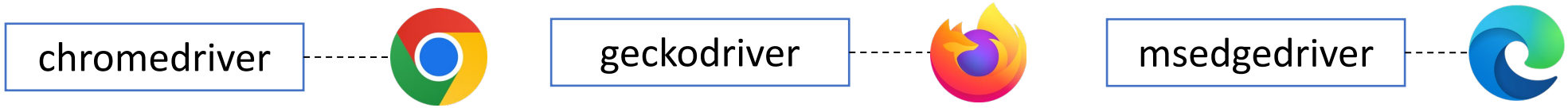
Selenium



Selenium – Setup

- The project setup is language-specific

- Browsers: we need at least a browser and its corresponding driver



Selenium Manager

Selenium – Hello World


```java
class HelloWorldSeleniumTest {

    WebDriver driver;

    @BeforeEach
    void setup() {
        driver = new ChromeDriver();
    }

    @Test
    void test() {
        // Open system under test (SUT)
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");

        // Assert web page title
        String title = driver.getTitle();
        assertThat(title).contains("Selenium WebDriver");
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

}
```

AssertJ

https://github.com/bonigarcia/browser-automation-apis/

Selenium – Ecosystem

https://www.selenium.dev/ecosystem/

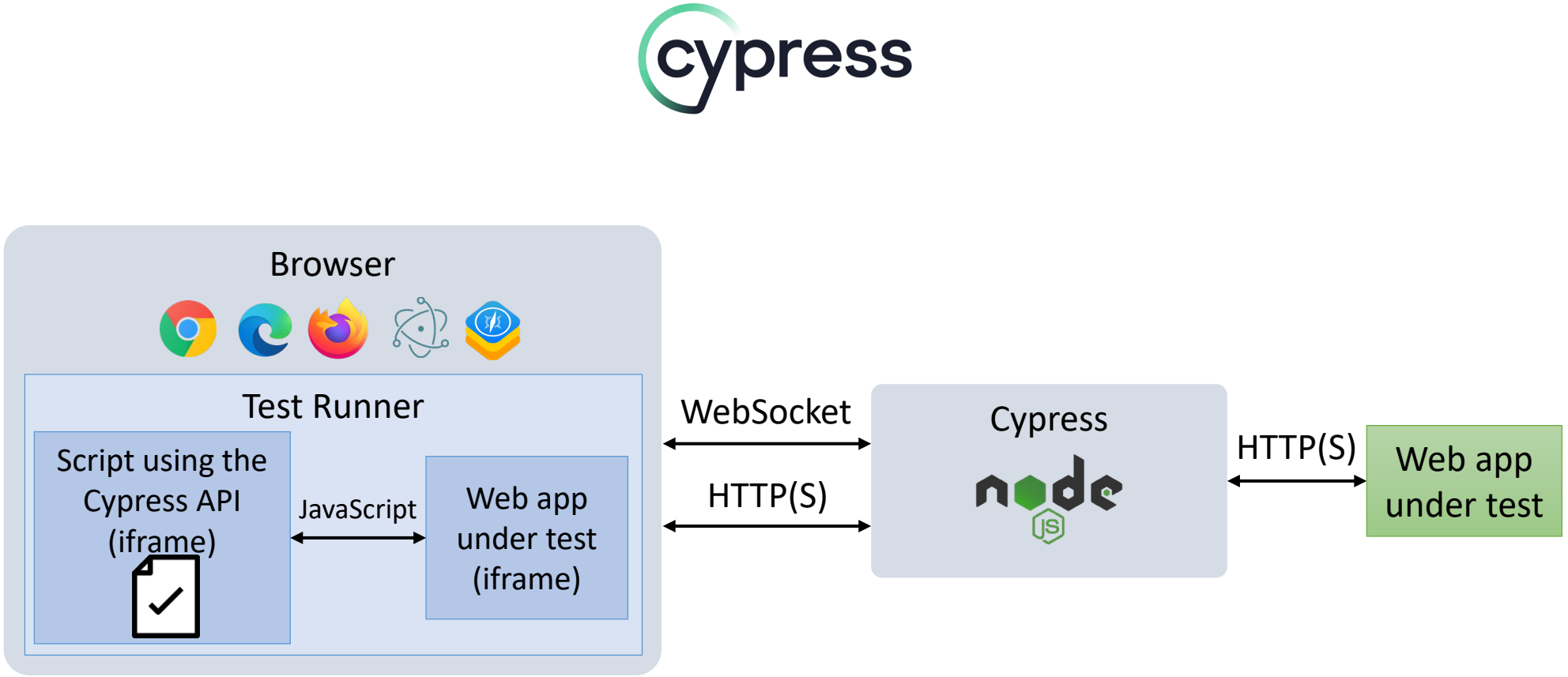
Cypress – What is Cypress?

- **Cypress** is a JavaScript **end-to-end automated testing framework**

https://www.cypress.io/

  - Created as a company in 2014 to provide a seamless experience for automated web testing
  - Language: JavaScript
  - Browsers: Chromium-based browsers (like Chrome and Edge), Firefox, Electron, WebKit (experimental)

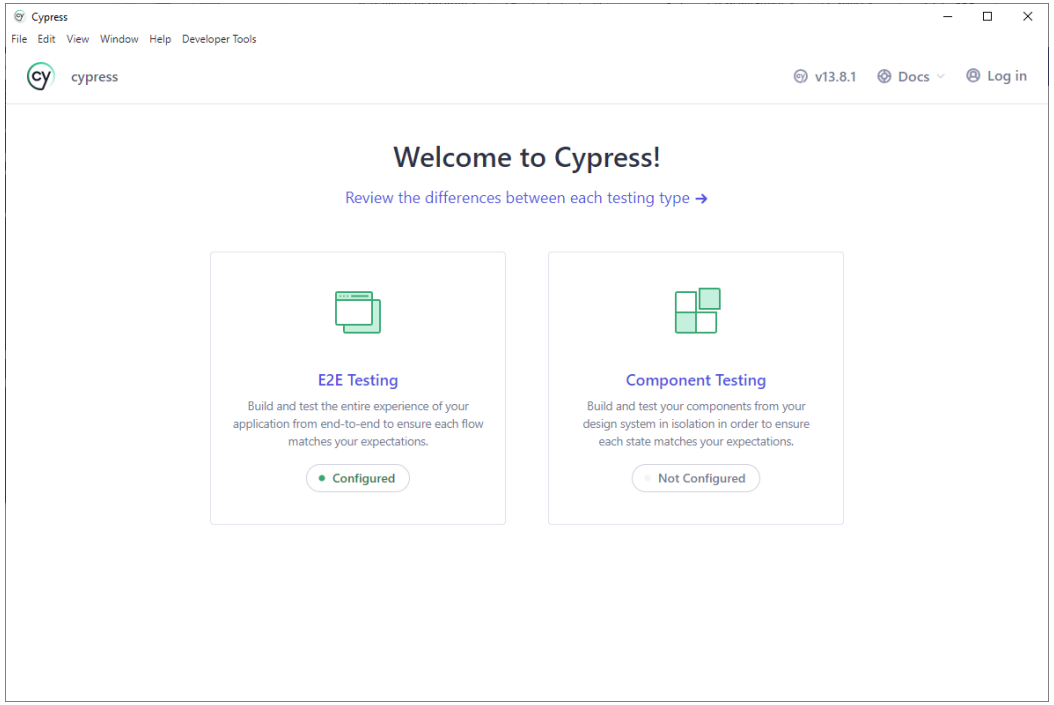
Cypress – Architecture

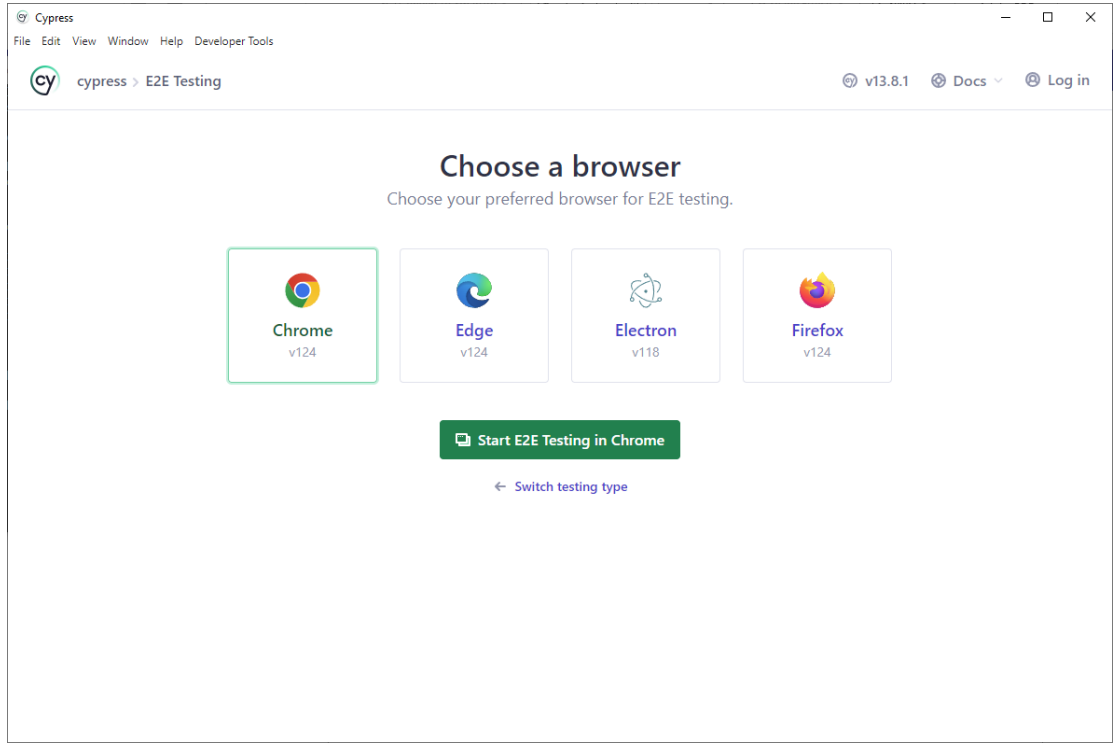


Cypress – Setup

- We use npm (the Node.js package manager) to install Cypress

```
npm install cypress
npx cypress open
```





Cypress – Hello World

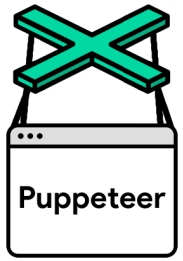
```
describe('Hello World Cypress', () => {
   it('Open sample web page and check title', () => {
      // Open system under test (SUT)
      cy.visit('https://bonigarcia.dev/selenium-webdriver-java/');

      // Assert web page title
      cy.title().should('include', 'Selenium WebDriver');
  });
});
```

https://github.com/bonigarcia/browser-automation-apis/

Puppeteer – What is Puppeteer?

- **Puppeteer** is a Node.js **browser automation library**

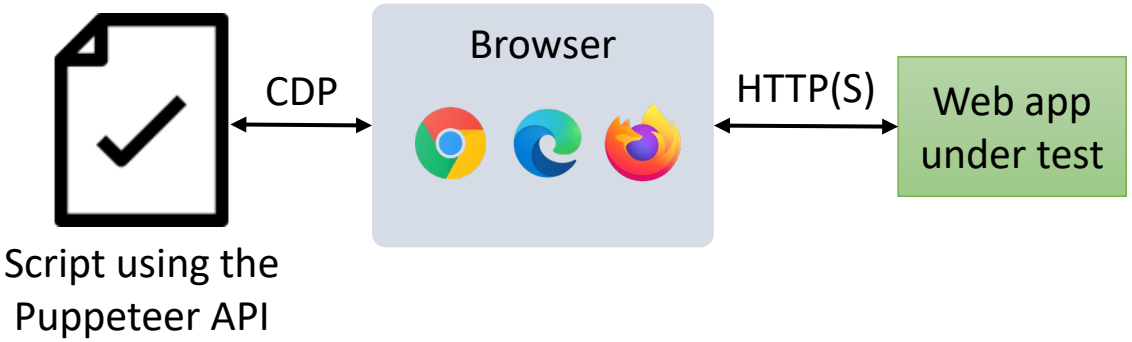


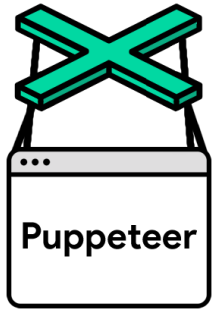
https://pptr.dev/

  - Created and maintained by the Chrome DevTools team at Google since 2017
  - Language: JavaScript or TypeScript

  - Browsers: Chromium-based browsers (like Chrome and Edge) and Firefox (experimental)

Puppeteer – Architecture





Puppeteer – Setup

- We can use npm to install Puppeteer

```
npm install puppeteer
```

  - Puppeteer automatically downloads and uses by default a browser called **Chrome for Testing**

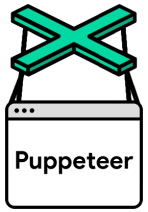
- To implement end-to-end tests, we will need some unit testing framework, like Jest

```
npx jest
```

Puppeteer – Hello World

```
describe('Hello World with Puppeteer', () => {
    it('Login in practice site', async () => {
        // Open system under test (SUT)
        await page.goto('https://bonigarcia.dev/selenium-webdriver-java/');

        // Assert web page title
        const title = await page.title();
        expect(title).toContain('Selenium WebDriver');
    });
});
```



https://github.com/bonigarcia/browser-automation-apis/

Playwright – What is Playwright?

- **Playwright** is a multilanguage **end-to-end automated testing framework**

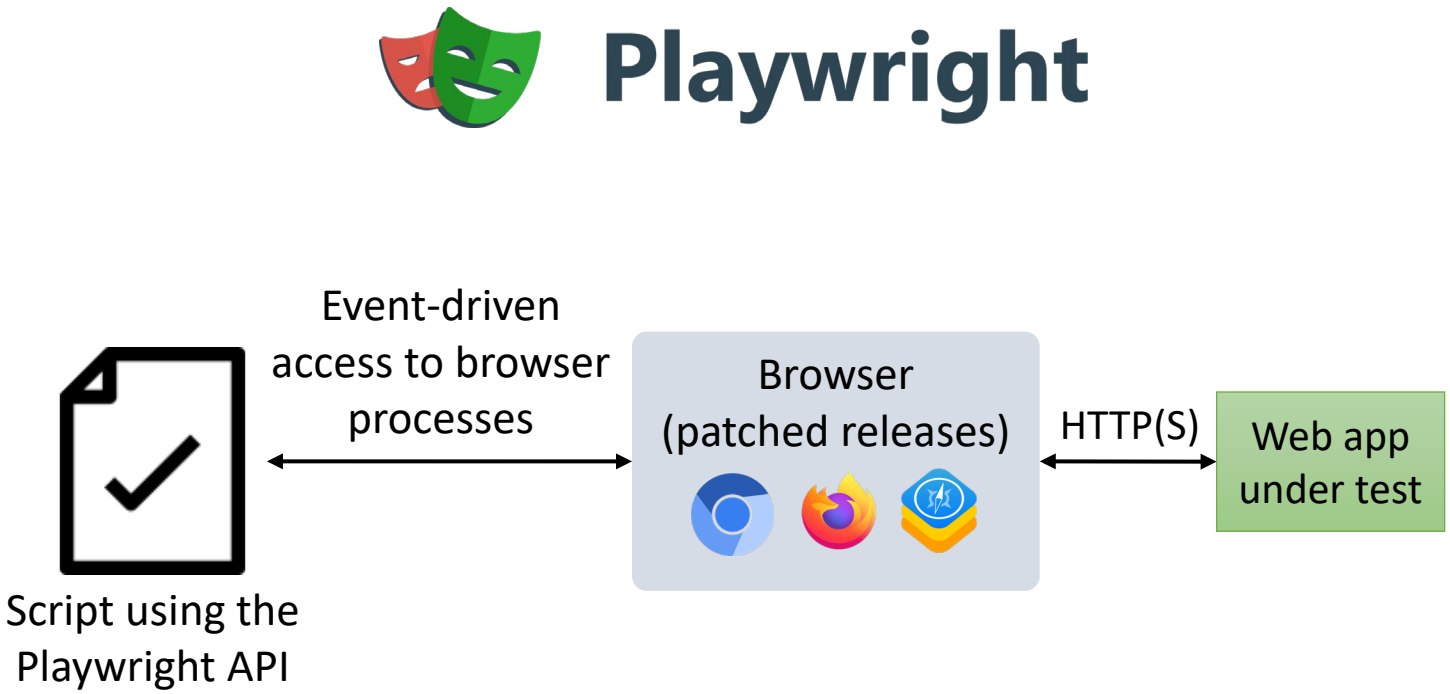
Playwright

https://playwright.dev/

  - Maintained by Microsoft since 2020, when the original team behind Puppeteer moved from Google to Microsoft
  - Languages: JavaScript, TypeScript, Python, .Net, and Java

  - Browsers: Patched releases of Chromium, Firefox, and WebKit

Playwright – Architecture



Playwright – Setup

- The project setup is language-specific

  - For example, we can use npm to create a new Playwright project:

```
$ npm init playwright@latest

> npx
> create-playwright

Getting started with writing end-to-end tests with Playwright:
Initializing project in '.'
? Do you want to use TypeScript or JavaScript? ...
  TypeScript
> JavaScript

? Where to put your end-to-end tests? » tests

? Add a GitHub Actions workflow? (y/N) » false

? Install Playwright browsers (can be done manually via 'npx playwright install')? (Y/n) » true

...

✓ Success! Created a Playwright Test project at ...
```

Playwright – Hello World

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

test('Hello World Playwright', async ({ page }) => {
    // Open system under test (SUT)
    await page.goto('https://bonigarcia.dev/selenium-webdriver-java/');

    // Assert web page title
    const title = await page.title();
    expect(title).toContain('Selenium WebDriver');
});
```

Playwright

https://github.com/bonigarcia/browser-automation-apis/

Comparison – Features

Feature	Selenium	Cypress	Puppeteer	Playwright
DOM elements selection	✓	✓	✓	✓
Keyboard actions	✓	✓	✓	✓
Mouse actions	✓	✓	✓	✓
Web forms	✓	✓	✓	✓
JavaScript execution	✓	✓	✓	✓
Dialog handling	✓	✓	✓	✓
Timeout	✓	✓	✓	✓
Explicit waiting	✓	✓	✓	✓
Screenshots	✓	✓	✓	✓
Window handling	✓	✓	✓	✓
Browser history	✓	✓	✓	✓
Shadow DOM	✓	✓	✓	✓
Cookies	✓	✓	✓	✓
Event listeners	✓	✓	✓	✓
Web authentication	✓	✓	✓	✓
Print page	✓	✓	✓	✓
Headless	✓	✓	✓	✓
Web extensions	✓	✓	✓	✓
Geolocation	✓	✓	✓	✓
Notifications	✓	✓	✓	✓
Web proxy	✓	✓	✓	✓
User media	✓	✓	✓	✓
Handling insecure pages	✓	✓	✓	✓
Localization	✓	✓	✓	✓
Incognito mode	✓	✓	✓	✓
Network traffic interception	✓	✓	✓	✓

Comparison – Features

Feature	Selenium	Cypress	Puppeteer	Playwright
Multilanguage	✓	✗	✗	✓
Cross-browser	✓	P	P	P
Automatic waiting	✗	✓	✗	✓
Tabs handling	✓	✗	✓	✓
Frames and iframes	✓	P	✓	✓
Console log gathering	P	✗	✓	✓
Session recording	✗	P	✓	✓
Assertions	✗	✓	✗	✓
Live reload	✗	✓	✗	✗
Test retries	✗	✓	✗	✗
Visual testing	✗	P	✗	✓
Component testing	✗	✓	✗	P

Testing-specific features

Example – Login

```
Feature: User login

  Scenario: Successful login with valid credentials
    Given the user is on page
       "https://bonigarcia.dev/selenium-webdriver-java/login-form.html"
    When the user enters "user" in the Login field
    And the user enters "user" in the Password field
    And the user clicks the Submit button
    Then "Login successful" is shown as text
    And a screenshot is saved
```

Gherkin Syntax

Example – Login

Selenium

```java
@Test
void test() throws Exception {
    // Open system under test (SUT)
    driver.get("https://bonigarcia.dev/selenium-webdriver-java/login-form.html");

    // Log in
    driver.findElement(By.id("username")).sendKeys("user");
    driver.findElement(By.id("password")).sendKeys("user");
    driver.findElement(By.cssSelector("button[type='submit']")).click();

    // Assert expected text
    WebElement successElement = driver.findElement(By.id("success"));
    assertThat(successElement.getText()).contains("Login successful");

    // Take screenshot
    File screenshot = ((TakesScreenshot) driver).getScreenshotAs(FILE);
    Path destination = Paths.get("login-selenium.png");
    Files.move(screenshot.toPath(), destination, REPLACE_EXISTING);
}
```

Puppeteer

```javascript
describe('User login', () => {
  it('Successful login with valid credentials', async () => {
    // Open system under test (SUT)
    await page.goto('https://bonigarcia.dev/selenium-webdriver-java/login-form.html');

    // Log in
    await page.type('#username', 'user');
    await page.type('#password', 'user');
    await page.click('button[type="submit"]');

    // Assert expected text
    const successElement = await page.$('#success');
    expect(await successElement?.evaluate(el => el.textContent)).toContain('Login successful');

    // Take screenshot
    await page.screenshot({ path: 'login-puppeteer.png' });
  });
});
```

cypress

```javascript
describe('User login', () => {
  it('Successful login with valid credentials', () => {
    // Open system under test (SUT)
    cy.visit('https://bonigarcia.dev/selenium-webdriver-java/login-form.html');

    // Log in
    cy.get('#username').type('user');
    cy.get('#password').type('user');
    cy.get('button[type="submit"]').click();

    // Assert expected text
    cy.get('#success').contains('Login successful');

    // Take screenshot
    cy.screenshot('login-cypress');
  });
});
```

Playwright

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

test('User login', async ({ page }) => {
  // Open system under test (SUT)
  await page.goto('https://bonigarcia.dev/selenium-webdriver-java/login-form.html');

  // Log in
  await page.locator('#username').fill('user');
  await page.locator('#password').fill('user');
  await page.locator('button[type="submit"]').click()

  // Assert expected text
  await expect(page.locator('#success')).toHaveText('Login successful');

  // Take screenshot
  await page.screenshot({ path: 'login-playwright.png' });
});
```

Conclusions – Summary

	Selenium	Cypress	Puppeteer	Playwright
Nature	Browser automation library	End-to-end testing framework	Browser automation library	End-to-end testing framework
Automation mechanism	Web standards (W3C WebDriver)	Custom architecture based on JavaScript	Chrome DevTools Protocol (CDP)	Patched versions of some browsers
Languages	Java, JavaScript, Python, .Net, Ruby	JavaScript	JavaScript or TypeScript	JavaScript, TypeScript, Python, .NET, and Java
Browsers	All major browsers	Chromium-based browsers, Firefox, and WebKit (experimental)	Chromium-based browsers and Firefox (experimental)	Chromium, Firefox, and WebKit
Maintained by	The Selenium project	The Cypress company	Google	Microsoft

Conclusions – Pros and Cons

	Selenium	Cypress	Puppeteer	Playwright
Pros	• Multilanguage • Cross-browser, since it is entirely based on open standards • Rich ecosystem	• The test and app run in the same browser, providing fast execution and automatic waiting • Built-in high-level testing features	• Comprehensive automation capabilities due to direct communication with the browser using CDP	• Multilanguage • Built-in high-level testing features
Cons	• Specific operations (e.g., explicit wait) should be individually handled (or using high-level frameworks belonging to its ecosystem) • Does not provides specific features for testing	• Because the app is run in a iframe, some actions are restricted (e.g. use different browsers or multiple tabs) • Limited cross-browser support • Only supports JavaScript	• Specific operations should be individually handled • Limited cross-browser support • Limited language support • Does not provides specific features for testing	• Rather than actual releases, it uses patched browser versions of Chrome, Firefox, and WebKit

APIs for Browser Automation

-Selenium, Cypress, Puppeteer, and Playwright-

Thank you very much!

Boni García
https://bonigarcia.dev/