WebDriver BiDi

The Future of Browser Automation is Now

Quality Beacon – DSTB's conference

Copenhagen, Denmark

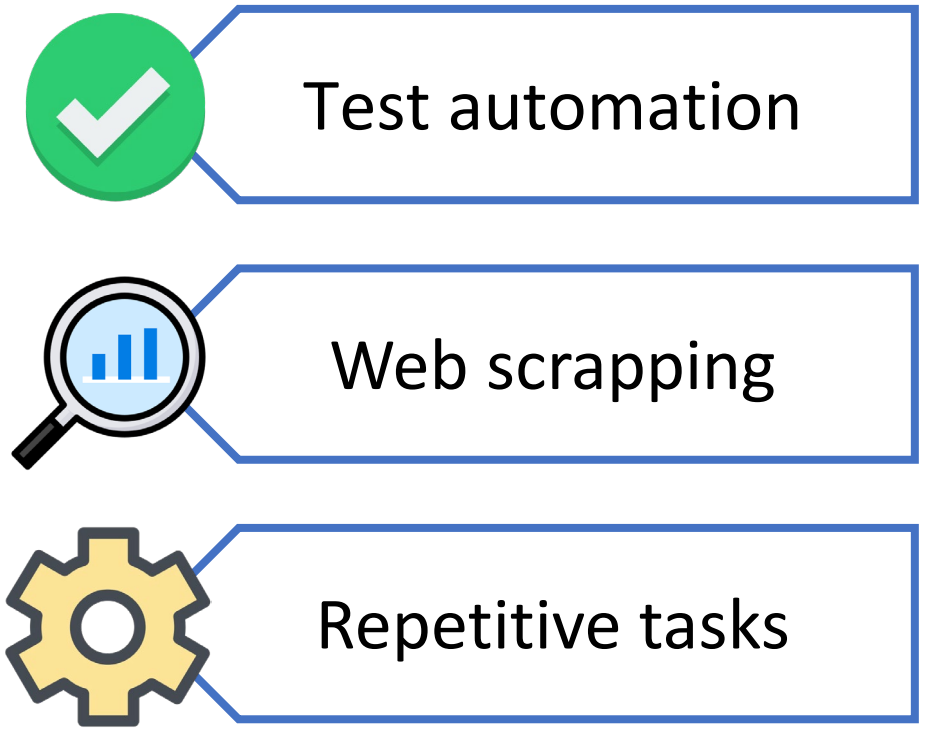
October 21, 2025

Boni García

https://bonigarcia.dev/

Introduction – What is browser automation?

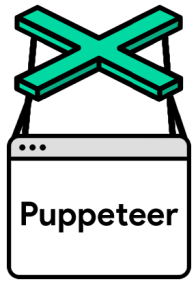
- **Browser automation** is the process of using software or scripts to control a web browser and perform tasks automatically, without manual human intervention



Introduction – What is WebDriver BiDi?

- **WebDriver BiDi (Bidirectional)** is a work-in-progress browser automation protocol that enables two-way, real-time communication between a browser and automation scripts

https://www.w3.org/TR/webdriver-bidi/



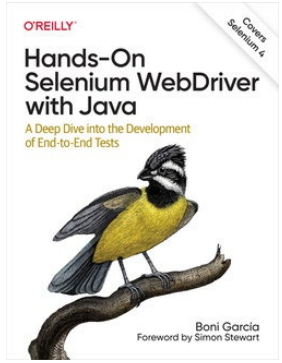
Introduction – What BiDi means for end users?

- More reliable automation: Bidirectional communication leads to more efficient tests
- Better browser control: Access to modern browser features like network interception, log capturing, and more
- Standardized: A single, standard protocol across all major browsers and automation tools
- Simplified tooling: No more need for tool-specific workarounds to access browser capabilities

Introduction – About me

- Associate Professor at UC3M (Spain)
- Tech lead at the Selenium project
- Open-source maintainer
- Author, speaker

https://bonigarcia.dev/




Selenium

What is Selenium?

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

https://selenium.dev/

  - Maintained as open-source 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 Hello World

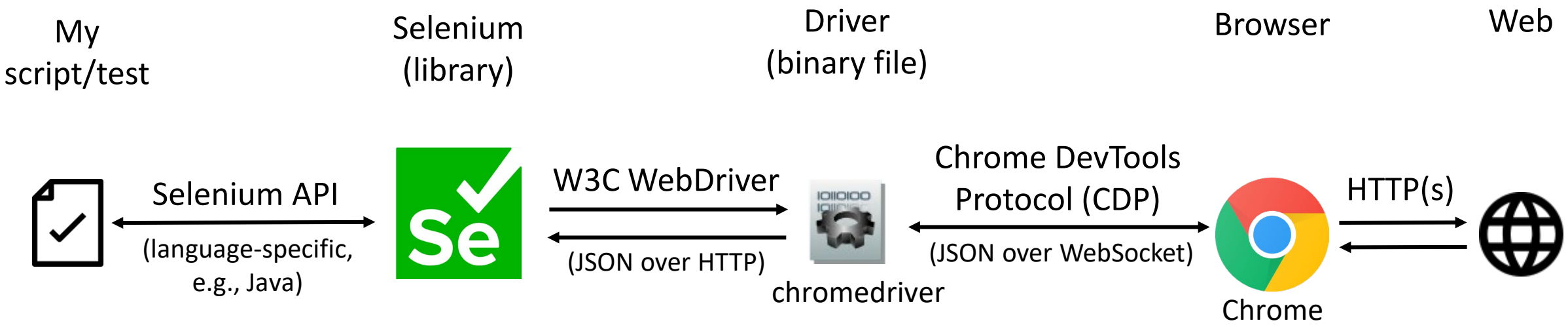

```java
public class HelloWorldSelenium {

    public static void main(String[] args) {
        // Open Chrome
        WebDriver driver = new ChromeDriver();

        // Navigate to web page
        String url = "https://bonigarcia.dev/selenium-webdriver-java/";
        driver.get(url);

        // Check page title
        String title = driver.getTitle();
        System.out.println(String.format("The title of %s is %s", url, title));

        // Close Chrome
        driver.quit();
    }

}
```


```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();
    }

}
```

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

Selenium Architecture

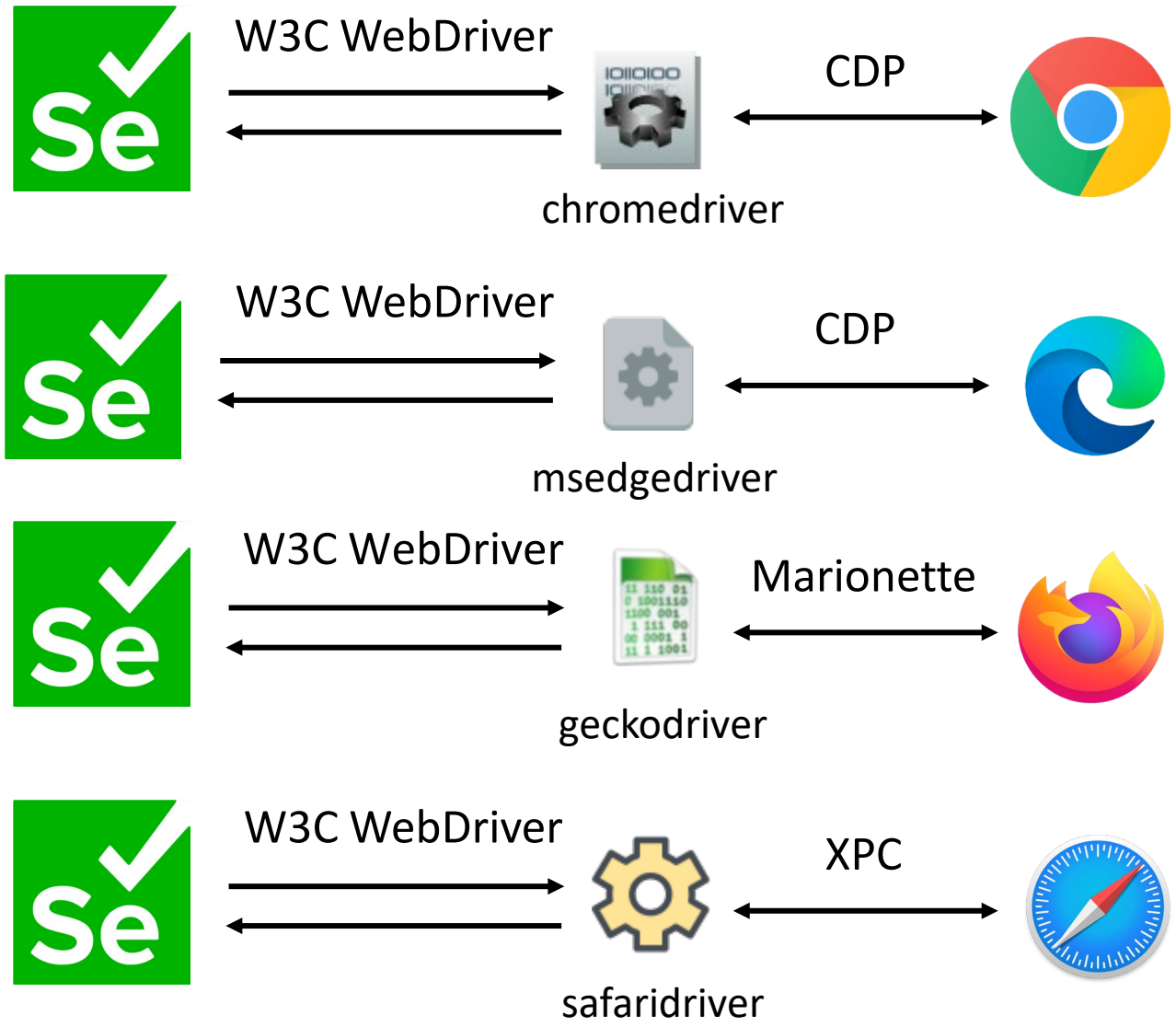
```
WebDriver driver = new ChromeDriver();
```



Driver management (download, setup, and maintenance) is no longer a problem thanks to Selenium Manager

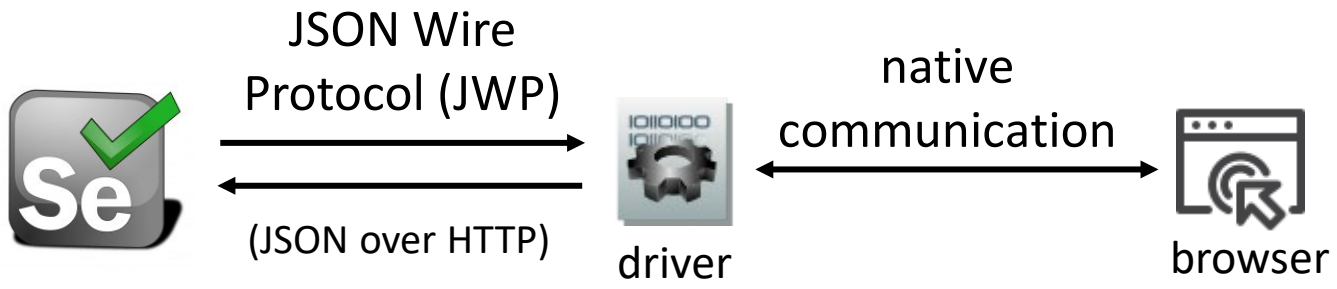
https://www.selenium.dev/documentation/selenium_manager/

Selenium Architecture



Each driver talks the W3C WebDriver protocol with Selenium and a native protocol with the browser

W3C WebDriver

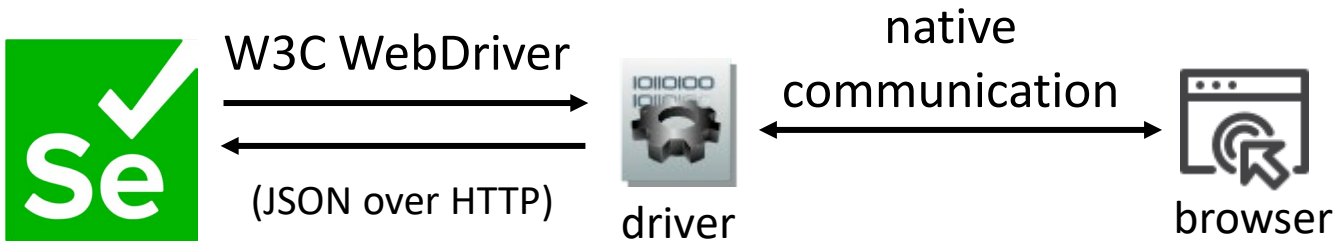


Selenium 2 to 3
(from 2011 to 2016)

- The Browser Testing and Tools Working Group took the concepts from JWP and formalized them into a vendor-neutral, web standard
- WebDriver became a W3C Recommendation in 2018

W3C WebDriver

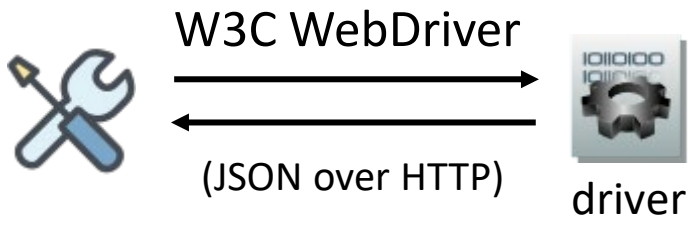
https://www.w3.org/TR/webdriver2/



Selenium 4
(from 2021 to today)

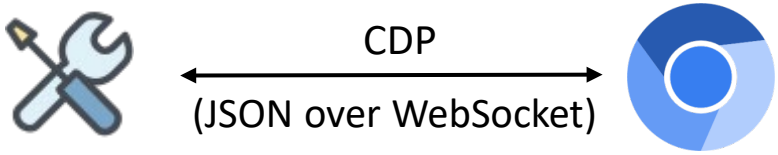
W3C WebDriver – Limitations

- The W3C WebDriver protocol works on top of HTTP (request-response model)
  - Unidirectional communication always initiated by the client
  - The script tells the browser what to do
  - The browser cannot spontaneously send information back to the script
  - This model makes it difficult to capture events happening in the browser in real-time, such as console logs, network requests, or JavaScript exceptions



Chrome DevTools Protocol (CDP)

- The **Chrome DevTools Protocol** (CDP) is a communication protocol that allows us to instrument, inspect, debug, and profile Chromium-based browsers (e.g., Chrome, Edge)
  - CDP use JSON message over **WebSocket** as communication channel
  - WebSocket provide persistent, **bi-directional**, full-duplex connection



https://chromedevtools.github.io/devtools-protocol/

CDP is used by browser automation tools (e.g., chromedriver, Puppeteer, and others), but it is not standard

W3C WebDriver BiDi

- The W3C Browser Testing and Tools Working Group started the **W3C WebDriver BiDi** specification in 2020
  - The goal is to create a new, standardized browser automation protocol that combines the stability of W3C WebDriver with the bidirectional, event-driven capabilities of the CDP
- WebDriver BiDi features:
  - Bidirectional communication using a WebSocket (like CDP)
  - Event-driven architecture (e.g., for log gathering)
  - Support for modern web features (e.g., network interception)

https://www.w3.org/TR/webdriver-bidi/

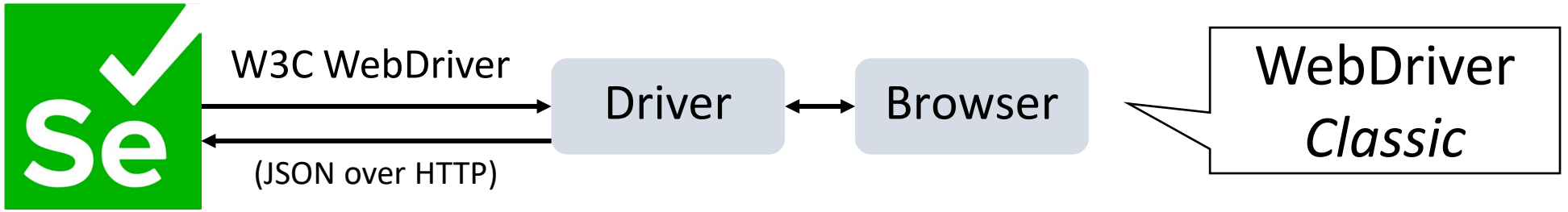
W3C WebDriver BiDi

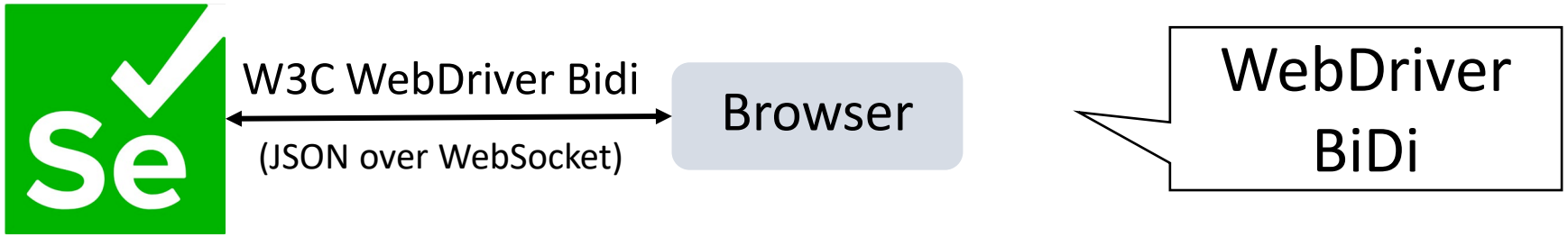
- W3C WebDriver BiDi is organized into several modules:

Module	Functionalities	Examples
Session	Manages connections between scripts and browser	Start/terminate BiDi sessions
Browser	Manage browser process	Maximize browser, manage user context
Browsing context	Controls tabs/windows (called “browsing contexts”)	Open a tab, navigate to URL, get DOM
Emulation	Emulate browser APIs	Geolocation, locale, time zone, theme
Network	Intercept monitor, and manipulate network traffic	Observe HTTP requests and responses
Script	Executes JavaScript in the browser	Run JavaScript and return result
Storage	Manage persistence storage	Create/read/delete cookies
Log	Manage browser logging	Listen to console logs and exceptions
Input	Simulates user input: keyboard, mouse, touch	Send key presses, mouse click
Web extension	Managing and interacting with web extensions	Install/uninstall web extension

https://www.w3.org/TR/webdriver-bidi/

Selenium and WebDriver BiDi





Selenium and WebDriver BiDi

- Some of the WebDriver BiDi modules are already available in the latest versions of Selenium 4

```
@BeforeEach
void setup() {
    ChromeOptions options = new ChromeOptions();
    options.enableBiDi();
    driver = new ChromeDriver(options);

}
```

```
@BeforeEach
void setup() {
    FirefoxOptions options = new FirefoxOptions();
    options.enableBiDi();
    driver = new FirefoxDriver(options);

}
```

```
@BeforeEach
void setup() {
    EdgeOptions options = new EdgeOptions();
    options.enableBiDi();
    driver = new EdgeDriver(options);

}
```

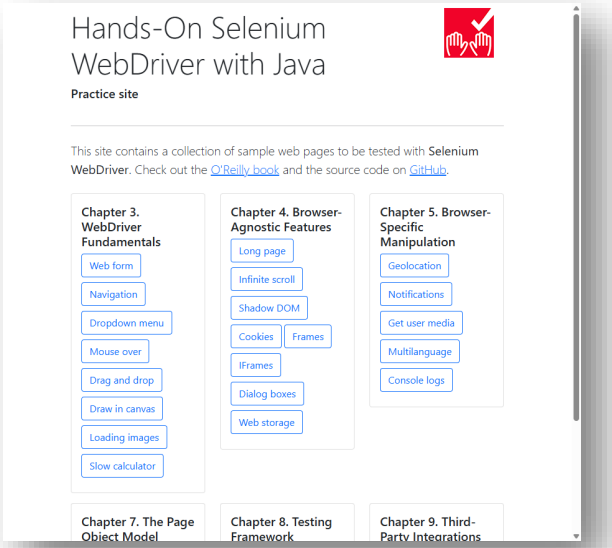
To use WebDriver BiDi in Selenium, first we need to enable it using browser options

https://www.selenium.dev/documentation/webdriver/bidi/w3c/

Selenium and WebDriver BiDi


```
@Test
void testBrowsing() {
    BrowsingContext context = new BrowsingContext(driver,
            driver.getWindowHandle());
    context.navigate("https://bonigarcia.dev/selenium-webdriver-java/");

    String screenshot = context.captureScreenshot(); // Base64
    assertThat(screenshot).isNotBlank();
}
```

Browsing context module



Selenium and WebDriver BiDi


```
@Test
void testNavigation() throws Exception {
    try (BrowsingContextInspector inspector = new BrowsingContextInspector(
            driver)) {
        CompletableFuture<NavigationInfo> future = new CompletableFuture<>();
        inspector.onBrowsingContextLoaded(future::complete);

        BrowsingContext context = new BrowsingContext(driver,
                driver.getWindowHandle());
        context.navigate("https://bonigarcia.dev/selenium-webdriver-java/",
                ReadinessState.COMPLETE);

        NavigationInfo navigationInfo = future.get(5, TimeUnit.SECONDS);
        assertThat(navigationInfo.getUrl())
                .contains("selenium-webdriver-java");
    }
}
```

Browsing context module

Selenium and WebDriver BiDi


```
@Test
void testInput() {
    driver.get(
            "https://bonigarcia.dev/selenium-webdriver-java/web-form.html");

    WebElement inputText = driver.findElement(By.name("my-text"));
    String textValue = "Hello World!";
    Input input = new Input(driver);
    Actions actions = new Actions(driver);
    Actions sendKeys = actions.sendKeys(inputText, textValue);
    input.perform(driver.getWindowHandle(), sendKeys.getSequences());
    assertThat(inputText.getDomProperty("value")).isEqualTo(textValue);

    inputText.clear();
    assertThat(inputText.getDomProperty("value")).isEmpty();
}
```

Input module

Selenium and WebDriver BiDi

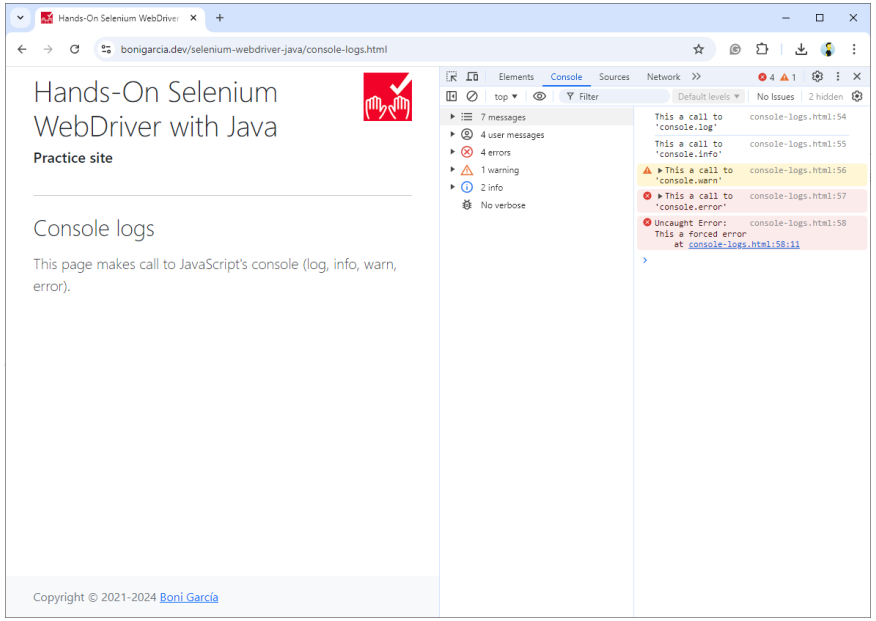

```
@Test
void testScript() {
    String id = driver.getWindowHandle();
    try (Script script = new Script(id, driver)) {
        EvaluateResult result = script.callFunctionInBrowsingContext(id,
                "()=>{return 1+2;}", false, Optional.empty(),
                Optional.empty(), Optional.empty());
        EvaluateResultSuccess successResult = (EvaluateResultSuccess) result;

        assertThat((Long) successResult.getResult().getValue().get())
                .isEqualTo(3);
    }
}
```

Script module

Selenium and WebDriver BiDi

```
@Test
void testLog() {
    List<GenericLogEntry> logs = new ArrayList<>();
    try (LogInspector logInspector = new LogInspector(driver)) {
        logInspector.onGenericLog(logs::add);
        logInspector.onConsoleEntry(logs::add);
        logInspector.onJavaScriptException(logs::add);
    }

    driver.get(
            "https://bonigarcia.dev/selenium-webdriver-java/console-logs.html");

    new WebDriverWait(driver, Duration.ofSeconds(5))
            .until(_d -> logs.size() > 3);

    for (GenericLogEntry log : logs) {
        System.out.println(log.getText());
    }
}
```

Log module

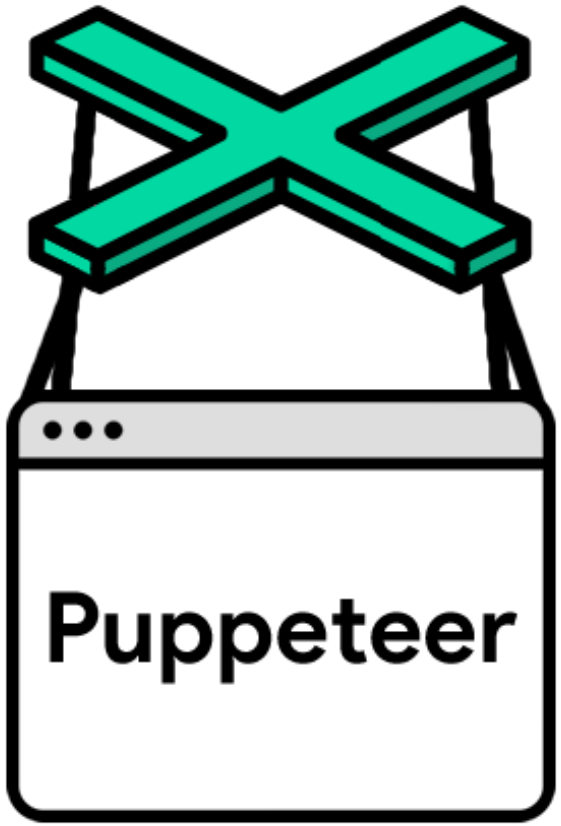


Selenium and WebDriver BiDi


```java
@Test
void testNetwork() throws Exception {
    try (Network network = new Network(driver)) {
        CompletableFuture<ResponseDetails> future = new CompletableFuture<>();
        network.onResponseCompleted(future::complete);
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");

        ResponseDetails response = future.get(5, TimeUnit.SECONDS);

        assertThat(response.getRequest().getMethod()).isEqualTo("GET");
        assertThat(response.getResponseData().getStatus()).isEqualTo(200);
    }
}
```

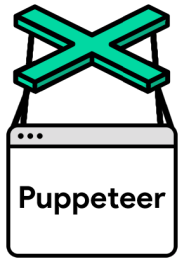
Network module

High-level Selenium API for BiDi is currently in progress (planned for Selenium 5)


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 Hello World


```
const puppeteer = require('puppeteer');

(async () => {
   // Launch Chrome
   const browser = await puppeteer.launch();
   const page = await browser.newPage();

   // Navigate to web page
   const url = 'https://bonigarcia.dev/selenium-webdriver-java/';
   await page.goto(url);

   // Check page title
   const title = await page.title();
   console.log(`The title of ${url} is ${title}`);

   // Close Chrome
   await browser.close();
})();
```

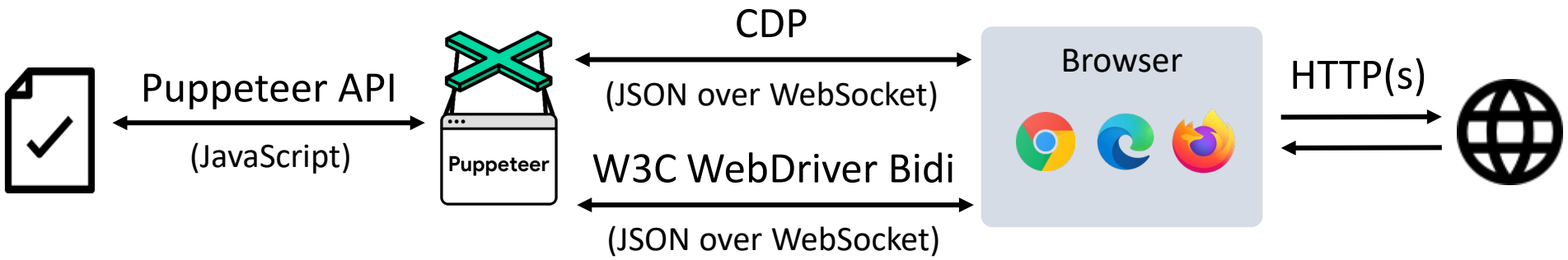
JS

```
const puppeteer = require('puppeteer');

describe('Hello World with Puppeteer', () => {
   let browser;
   let page;

   beforeAll(async () => {
       browser = await puppeteer.launch();
       page = await browser.newPage();
   });

   it('Open sample web page and check title', 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');
   });

   afterAll(async () => {
      await browser.close();
   });

});
```

Jest

Puppeteer Architecture

- Puppeteer is based on the **Chrome DevTools Protocol (CDP)**
- Firefox and CDP:
  - Firefox 82 provided partial support to CDP (February 2021)
  - Firefox 129 deprecated the support for CDP (May 2024)
  - Firefox 141 removed completely the CDP support (May 2025)
- Puppeteer 23 started to provide stable support for Firefox through WebDriver BiDi (May 2024)



Puppeteer and WebDriver BiDi

- The support of WebDriver BiDi in Puppeteer is in progress:

Puppeteer features fully supported over WebDriver BiDi	Puppeteer features not supported over WebDriver BiDi
• Browser automation • Page automation • Script evaluation • Selectors and locators except for ARIA • Input • JavaScript dialog interception • Screenshots • PDF generation • Permissions • Request interception	• Emulation • CDP-specific features • Accessibility • Coverage • Tracing • Other methods

https://pptr.dev/webdriver-bidi

Puppeteer and WebDriver BiDi

```
const puppeteer = require('puppeteer');

describe('Hello World with Puppeteer and BiDi', () => {
   let browser;
   let page;

   beforeAll(async () => {
       browser = await puppeteer.launch({
          browser: 'firefox',
          protocol: 'webDriverBiDi',
       });
       page = await browser.newPage();
   });

   afterAll(async () => {
      await browser.close();
   });

   it('Open sample web page and check title', 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');
   });
});
```

WebDriver BiDi is now the default protocol for Firefox as of Puppeteer 24

Puppeteer and WebDriver BiDi


```
const puppeteer = require('puppeteer');

describe('Log gathering with Puppeteer and BiDi', () => {
   let browser;
   let page;

   beforeAll(async () => {
       browser = await puppeteer.launch({
          browser: 'firefox',
          protocol: 'webDriverBiDi',
       });
       page = await browser.newPage();
   });

   it('Capture console logs via BiDi', async () => {
      const messages = [];
      page.on('console', msg => {
         messages.push(msg.text());
         console.log(`Console message: ${msg.text()}`);
      });

      await page.goto('https://bonigarcia.dev/selenium-webdriver-java/');
      const message = 'Hello from the page!';

      await page.evaluate((text) => console.log(text), message);
      expect(messages).toContain(message);
   });

   afterAll(async () => {
      await browser.close();
   });

});
```

Log gathering

```
const puppeteer = require('puppeteer');

describe('Network interception with Puppeteer and BiDi', () => {
   let browser;
   let page;

   beforeAll(async () => {
       browser = await puppeteer.launch({
          browser: 'firefox',
          protocol: 'webDriverBiDi',
       });
       page = await browser.newPage();
   });


   it('logs all network requests', async () => {
      const urls = [];
      page.on('request', req => urls.push(req.url()));

      await page.goto('https://bonigarcia.dev/selenium-webdriver-java/');

      expect(urls.some(url => url.includes('bonigarcia.dev'))).toBe(true);
   });

   afterAll(async () => {
      await browser.close();
   });

});
```

Network interception


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

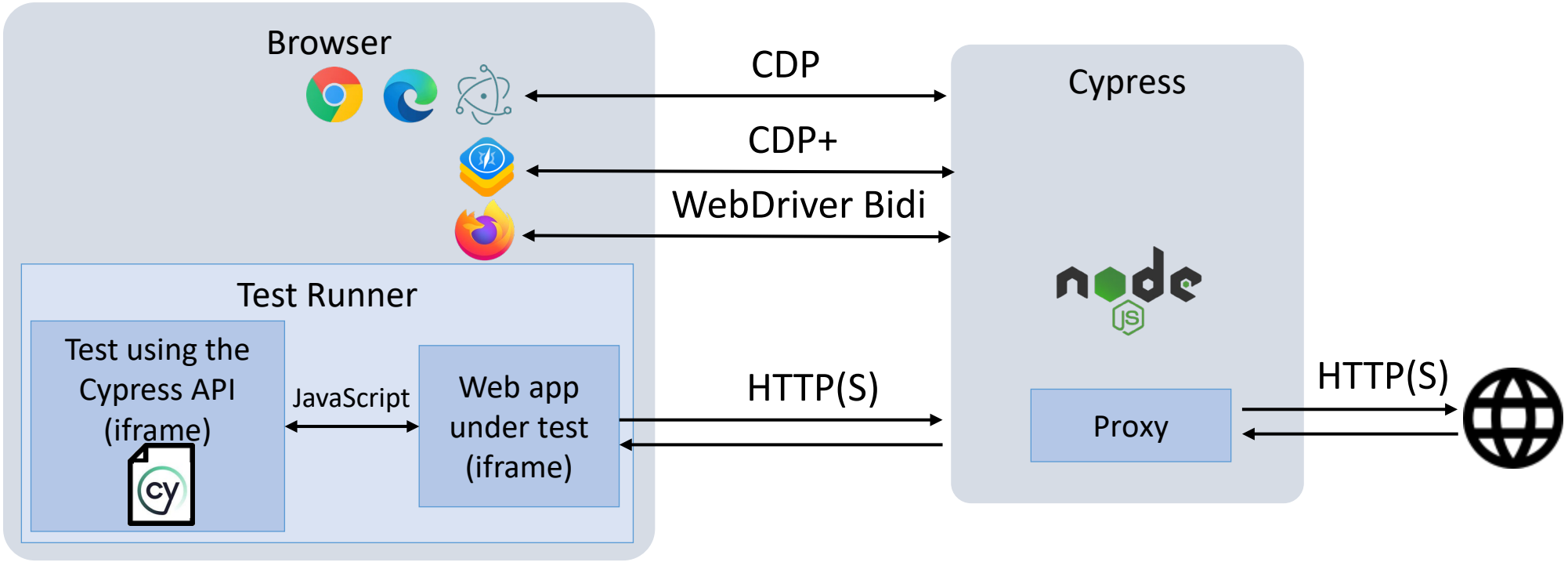
  - Browsers: Chromium-based browsers (like Chrome, Edge, or Electron), Firefox, and WebKit (experimental)

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');
  });
});
```

Cypress Architecture

- Cypress uses CDP to control Chromium-based browsers
- Cypress uses Playwright automation protocol (CDP+) to control WebKit
- Cypress uses WebDriver BiDi to control Firefox 135+ starting with Cypress 14.1.0 (February 2025)



Cypress and WebDriver BiDi

- Current status:
  - Cypress defaults to automating Firefox with WebDriver BiDi
  - Cypress no longer supports CDP in Firefox as of Cypress 15 (August 2025)
- In theory, Cypress users should not notice the specific protocol used internally (CDP or Bidi)
  - The Cypress API (`cy.visit`, `cy.get`, `cy.intercept`, etc.) acts as abstraction layer, and the underlying protocol should not be noticed

https://docs.cypress.io/app/references/changelog

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 Hello World

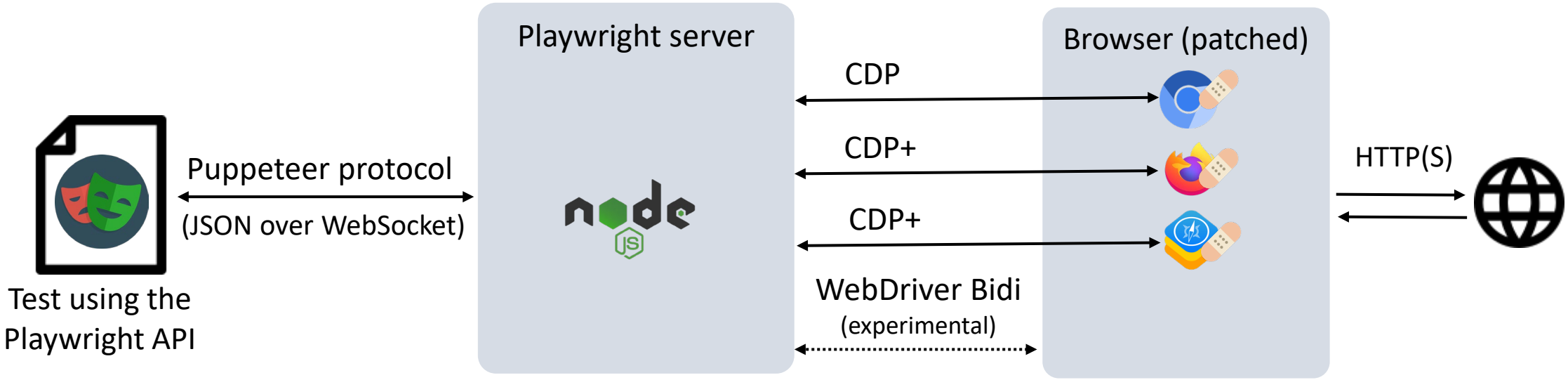
```
const { test, expect } = require('@playwright/test');

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/

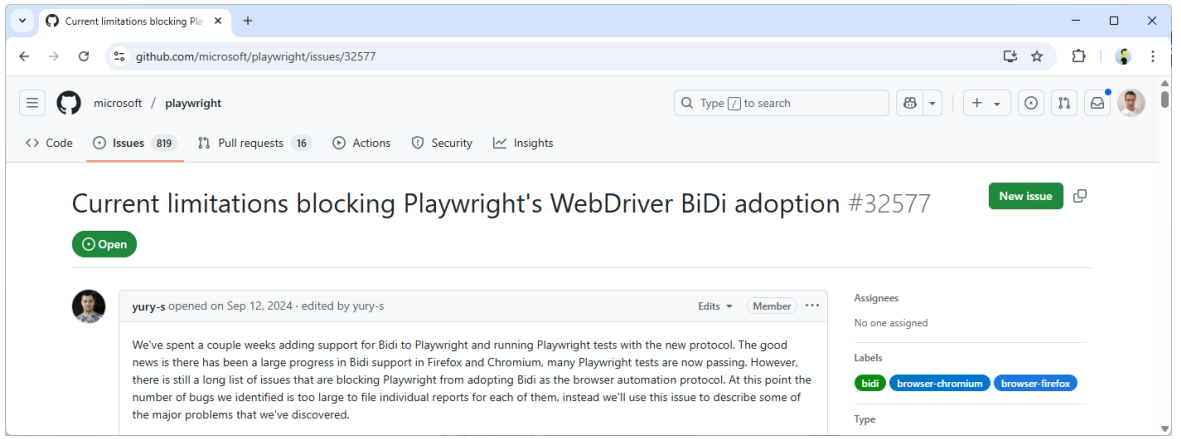
Playwright Architecture

- Playwright maintains patched versions of Chromium, Firefox, and WebKit (to enable automation and cross-browser consistency)
- Playwright uses an extended version of CDP to implement to control uniformly across these browsers



Playwright and WebDriver BiDi

- Playwright's WebDriver BiDi support is still experimental
  - There is active work to support WebDriver BiDi, but it is not yet complete
  - Currently, it is not currently possible for a user to configure Playwright to use WebDriver BiDi
  - The transition to WebDriver BiDi will likely happen automatically in a future version of Playwright once the protocol is mature enough to support all of Playwright's features



https://github.com/microsoft/playwright/issues/32577

Conclusions

- WebDriver BiDi is an in-progress W3C standard for the next generation of browser automation
- It combines the stability of WebDriver with the power of CDP, offering a single, standard way to automate browsers
- Major tools like Selenium, Puppeteer, Cypress, and Playwright are actively integrating WebDriver BiDi
  - Selenium: BiDi low-level features available in Selenium 4, high-level API is in development (planned for Selenium 5)
  - Puppeteer: BiDi support for Firefox since v23
  - Cypress: BiDi support for Firefox since v14.1.0
  - Playwright: BiDi support is still experimental

Conclusions

- How to track the evolution of WebDriver BiDi?
  - W3C WebDriver issues
  - W3C WebDriver BiDi roadmap
  - W3C WebDriver BiDi planning
  - WPT dashboard
  - Implementation of WebDriver BiDi for Chromium
  - Chrome for developers blog about BiDi
  - Communication with the Firefox team about WebDriver BiDi

W3C® WebDriver BiDi

https://www.w3.org/TR/webdriver-bidi/

WebDriver BiDi

The Future of Browser Automation is Now

Thank you so much!

Get these slides at:

https://bonigarcia.dev/

Boni García
boni.garcia@uc3m.es

Read this story at:

https://medium.com/@boni.gg