如何測試與保護你的 Web Application with Playwright



tw MVC

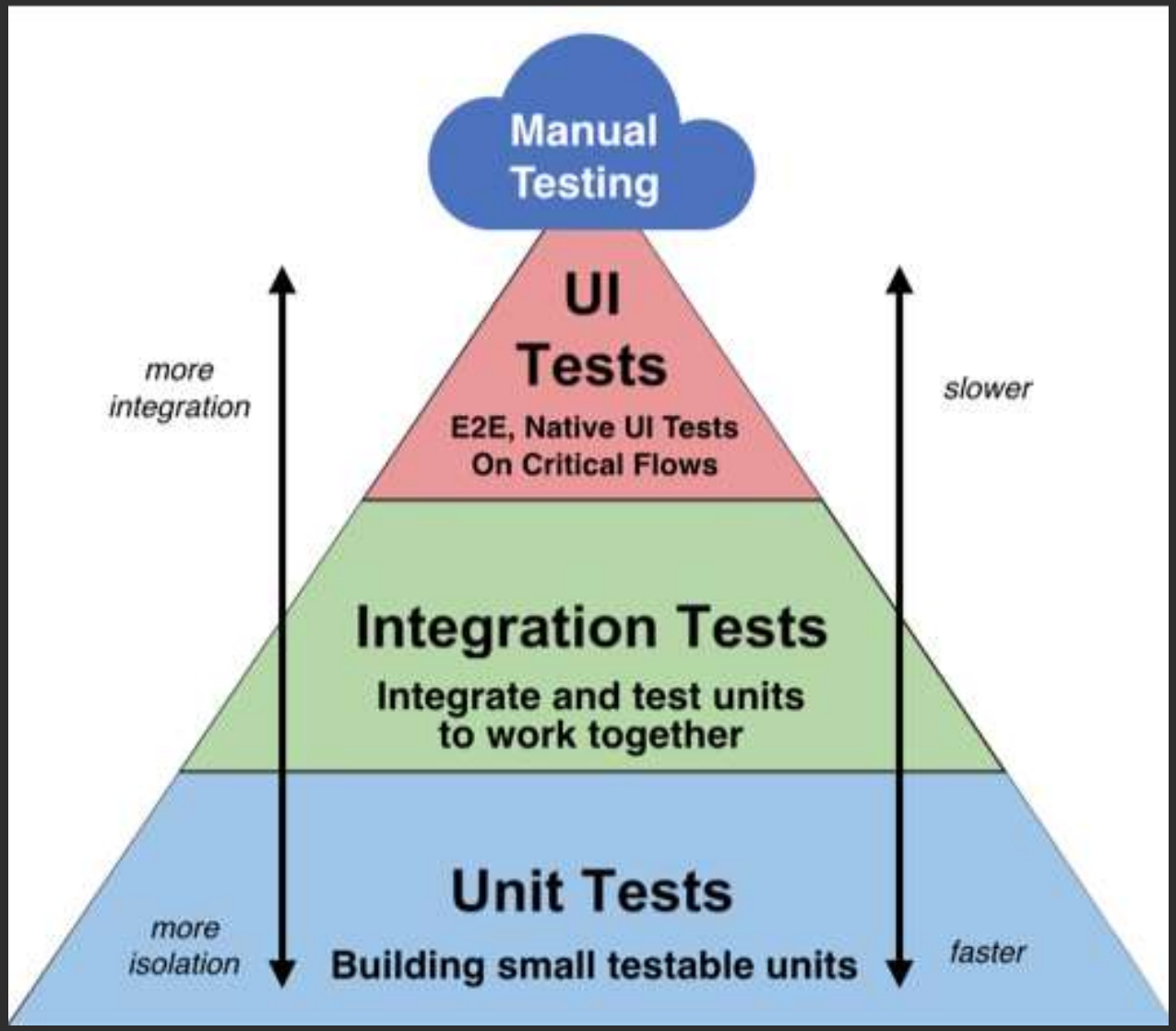
歡迎參加我們的每週四固定聚會

阿磊 Cheng-Ju Wu

開始前...如果我是測試工程師

End-to-End Test ?

End-to-End Test



End-to-End Test

工人智慧

環境問題

Timeout

Trace Log

脆弱性

耗時

重複工作

成本

End-to-End Test

- 繁瑣又缺乏創造力
- 時間成本
- 開發維護成本
- 執行慢
- 測試覆蓋率低
- 可靠性



如何改善
提升效率

切入主題



Agenda

- What is Playwright
- Why choose Playwright
- How to use Playwright
- Use cases

What is Playwright



What is Playwright

Playwright

Playwright Test

github.com/microsoft/playwright

Playwright is

- Bowser Automation Framework
- Node.js Library
- Open Source
- Puppeteer
- By Microsoft

github.com/microsoft/playwright

Playwright is

- Cross-platform (Windows, Linux, macOS)
- Cross-browser testing with single API
- Supports 3 browser engines
  - Chromium：Google Chrome / Microsoft Edge / Opera
  - Firefox
  - WebKit：Apple Safari



	Linux	macOS	Windows
Chromium 100.0.4863.0	✅	✅	✅
WebKit 15.4	✅	✅	✅
Firefox 96.0.1	✅	✅	✅

Playwright can't

- 與實際落差
  - 無法操作手機，只能開啟手機瀏覽器模擬模式
  - WebKit 與 Safari 差異

Playwright is

- Cross-language
  - TypeScript / JavaScript
  - .NET
  - Python
  - Java

```
import {chromium, firefox, webkit} from 'playwright';

for (const browserType of [chromium, firefox, webkit]) {
  const browser = await browserType.launch();
  const page = await browser.newPage();
  await page.goto('https://playwright.dev');
  await page.screenshot({path: `image-${browserType.name()}.png`});
  await browser.close();
}
```

Playwright

JavaScript

```python
from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    for browser_type in [p.chromium, p.firefox, p.webkit]:
        browser = browser_type.launch()
        page = browser.new_page()
        page.goto("https://playwright.dev")
        page.screenshot(path="image-" + browser_type.name + ".png")
        browser.close()
```

Playwright

Python

```java
public static void main(String[] args) {
  try (Playwright playwright = Playwright.create()) {
    List<BrowserType> browserTypes = Arrays.asList(
      playwright.chromium(), playwright.webkit(), playwright.firefox()
    );
    for (BrowserType browserType : browserTypes) {
      try (Browser browser = browserType.launch()) {
        Page page = browser.newPage();
        page.navigate("http://playwright.dev");
        page.screenshot(new Page.ScreenshotOptions()
            .setPath(Paths.get("image-" + browserType.name() + ".png")));
      }
    }
  }
}
```

Playwright

Java

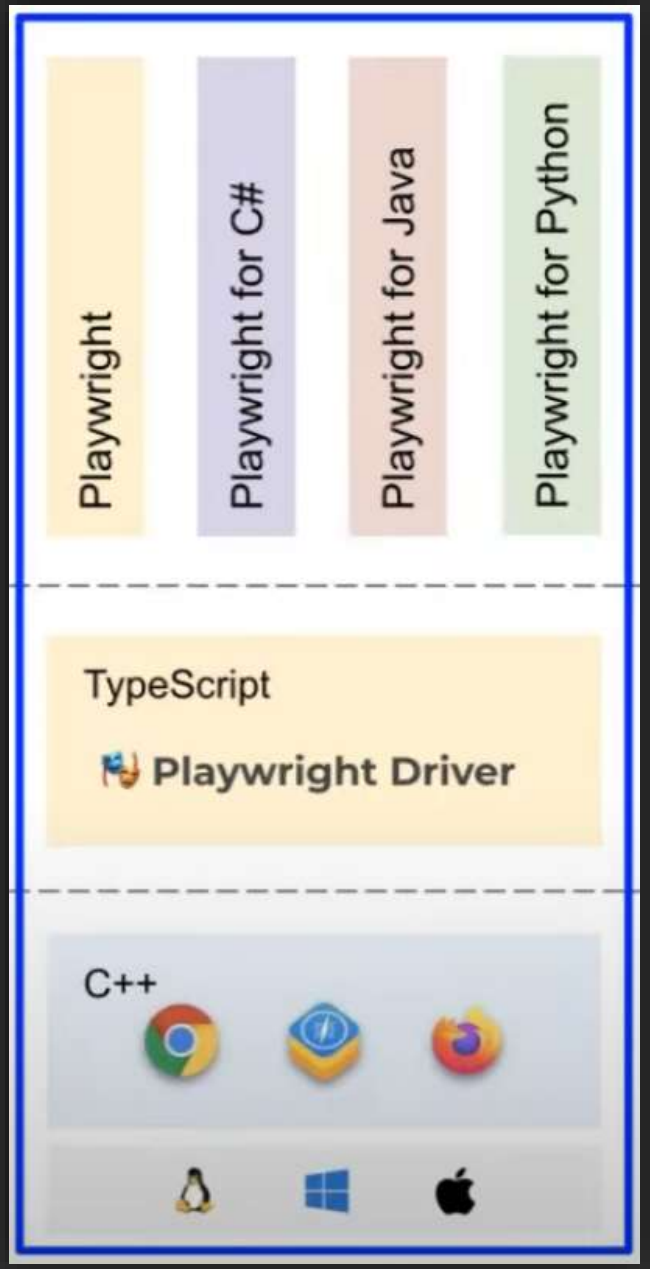
```
public static async Task Main()
{
    using var playwright = await Playwright.CreateAsync();
    var browserTypes = new List<IBrowserType>
    {
        playwright.Chromium,
        playwright.Firefox,
        playwright.Webkit,
    };

    foreach (var browserType in browserTypes)
    {
        await using var browser = await browserType.LaunchAsync();
        var page = await browser.NewPageAsync();
        await page.GotoAsync( url: "https://playwright.dev/dotnet");
        await page.ScreenshotAsync(new PageScreenshotOptions { Path = $"screenshot-{browserType.Name}.png" });
    }
}
```

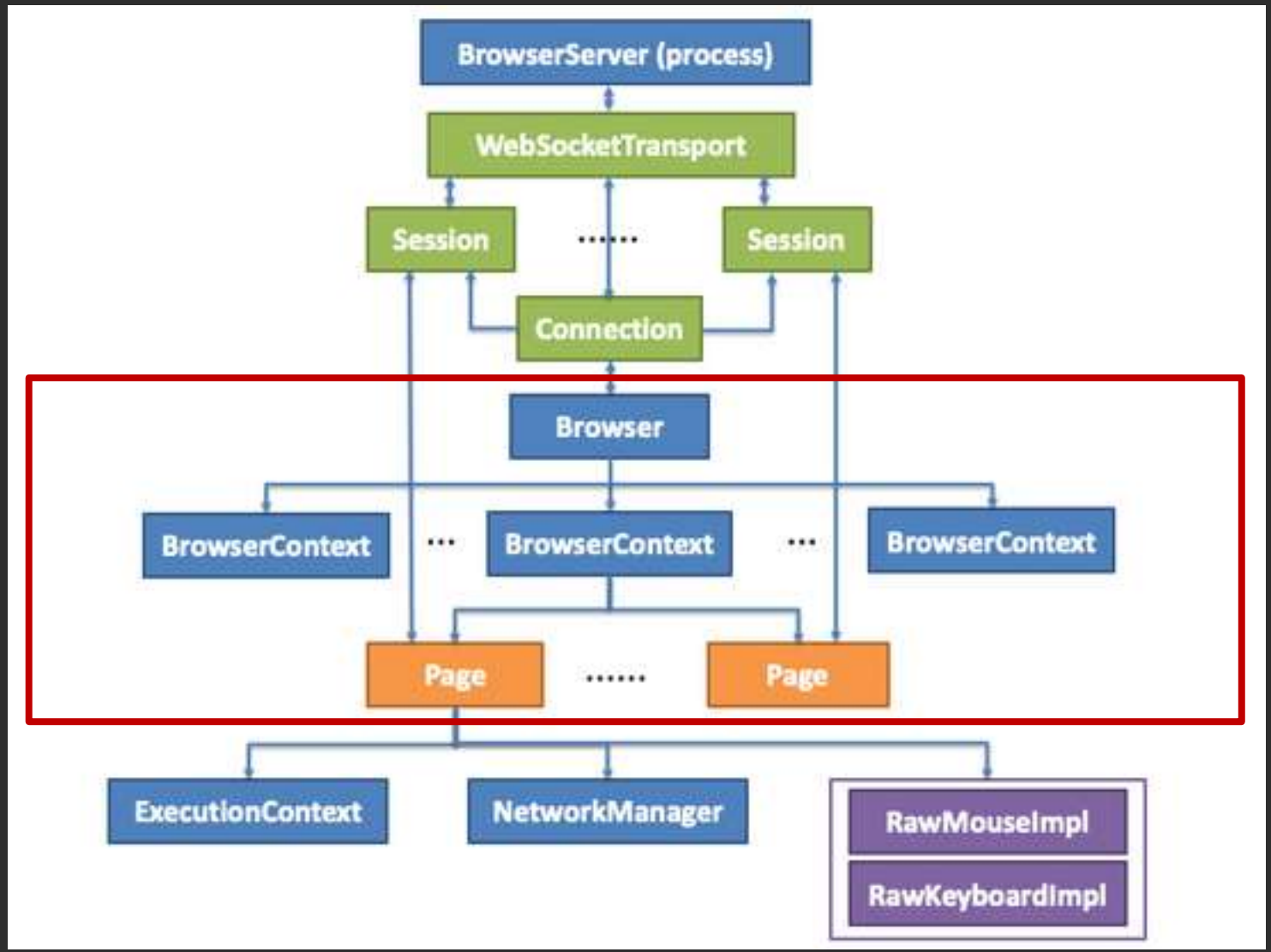
Playwright

.NET

Playwright is



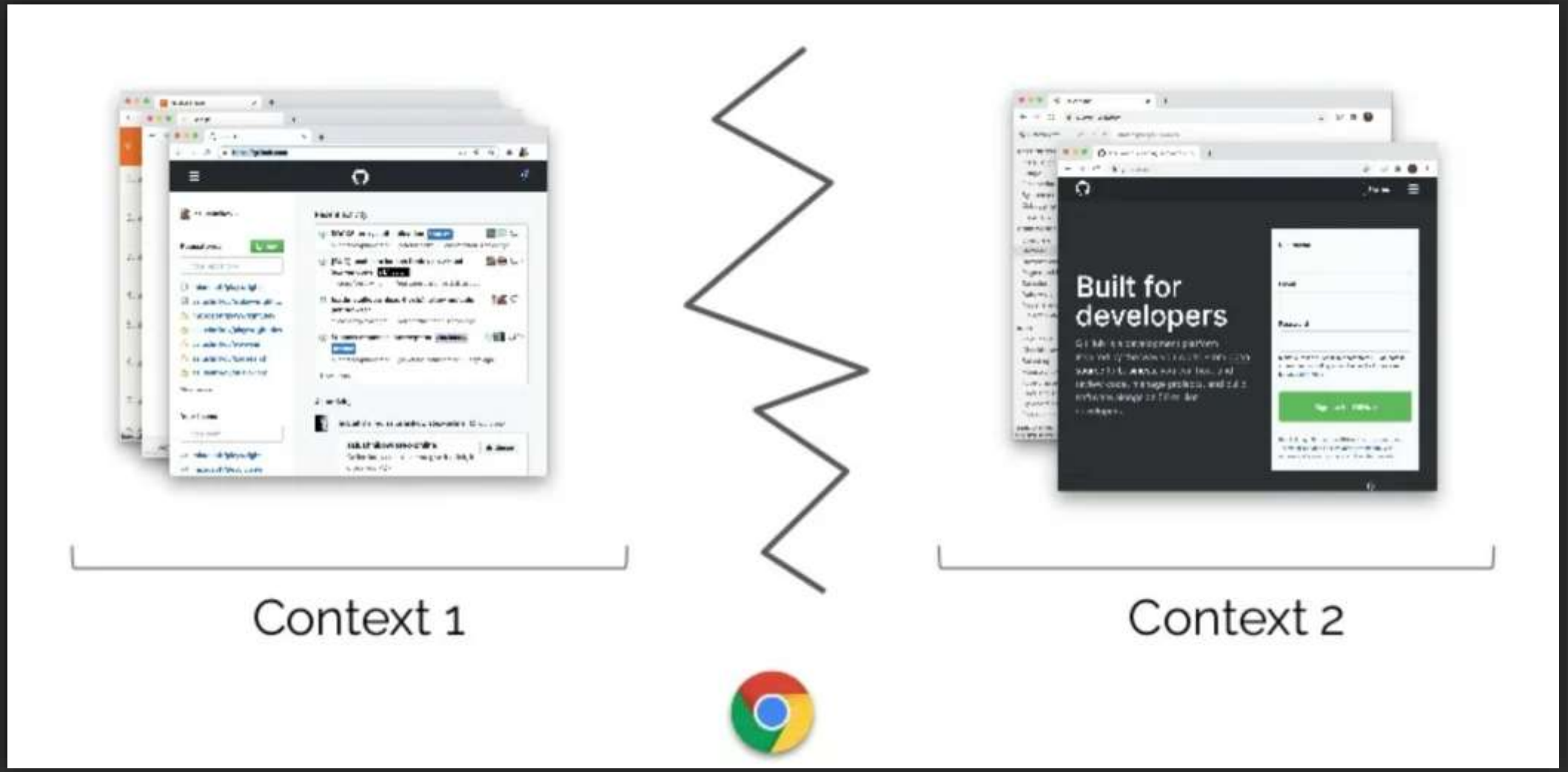
Playwright is



Playwright Test

- Test Runner
- Reporters
- Parallelism
- Test Retries
- …

Playwright: Browser Contexts



Playwright: Browser Contexts

- Not Restart a Browser
- 完整隔離的環境
- 快速實例化
- 較低的資源開銷

```
import {webkit, chromium, firefox} from 'playwright';

const browser = await chromium.launch();

for (let i = 0; i < 10; ++i) {
  // Fast, simple, configurable!
  const context = await browser.newContext()
  const page = await context.newPage();
  await page.goto('https://playwright.dev');
  await context.close();
}
```

Playwright: Browser Contexts

Multiple contexts

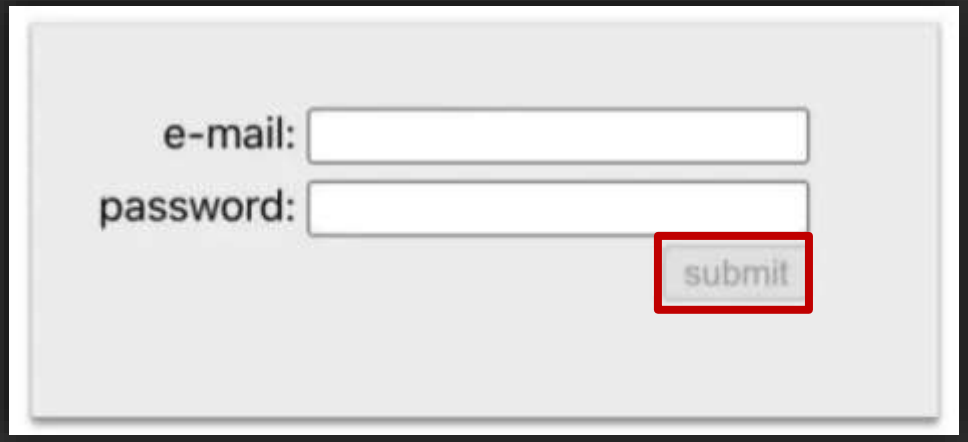
Browser contexts are isolated environments on a single browser instance. Playwright can create multiple browser contexts within a single scenario. This is useful when you want to test for multi-user functionality, like chat.

```
using Microsoft.Playwright;
using System.Threading.Tasks;

class Program
{
    public static async Task Main()
    {
        using var playwright = await Playwright.CreateAsync();
        // Create a Chromium browser instance
        await using var browser = await playwright.Chromium.LaunchAsync();
        await using var userContext = await browser.NewContextAsync();
        await using var adminContext = await browser.NewContextAsync();
        // Create pages and interact with contexts independently.
    }
}
```

Playwright: Auto-waiting

- 簡化測試程式，有助於聚焦實際任務
  - 不太需要手動寫下 `wait` 來確保前端收到回應
- 提高測試可靠性
  - 因應現今網站，頻繁存取後端 `API` 的特性
  - 執行操作前對 `Element` 可操作性檢查




```
setTimeout(async () => {
  await page.click('#submit');
}, 1000);
```

Time Does Not Exist in the Cloud!

Playwright: Auto-waiting

Action	Attached	Visible	Stable	Receives Events	Enabled	Editable
check	Yes	Yes	Yes	Yes	Yes	-
click	Yes	Yes	Yes	Yes	Yes	-
dblclick	Yes	Yes	Yes	Yes	Yes	-
setChecked	Yes	Yes	Yes	Yes	Yes	-
tap	Yes	Yes	Yes	Yes	Yes	-
uncheck	Yes	Yes	Yes	Yes	Yes	-
hover	Yes	Yes	Yes	Yes	-	-
scrollIntoViewIfNeeded	Yes	Yes	Yes	-	-	-
screenshot	Yes	Yes	Yes	-	-	-
fill	Yes	Yes	-	-	Yes	Yes
selectText	Yes	Yes	-	-	-	-

Why choose Playwright



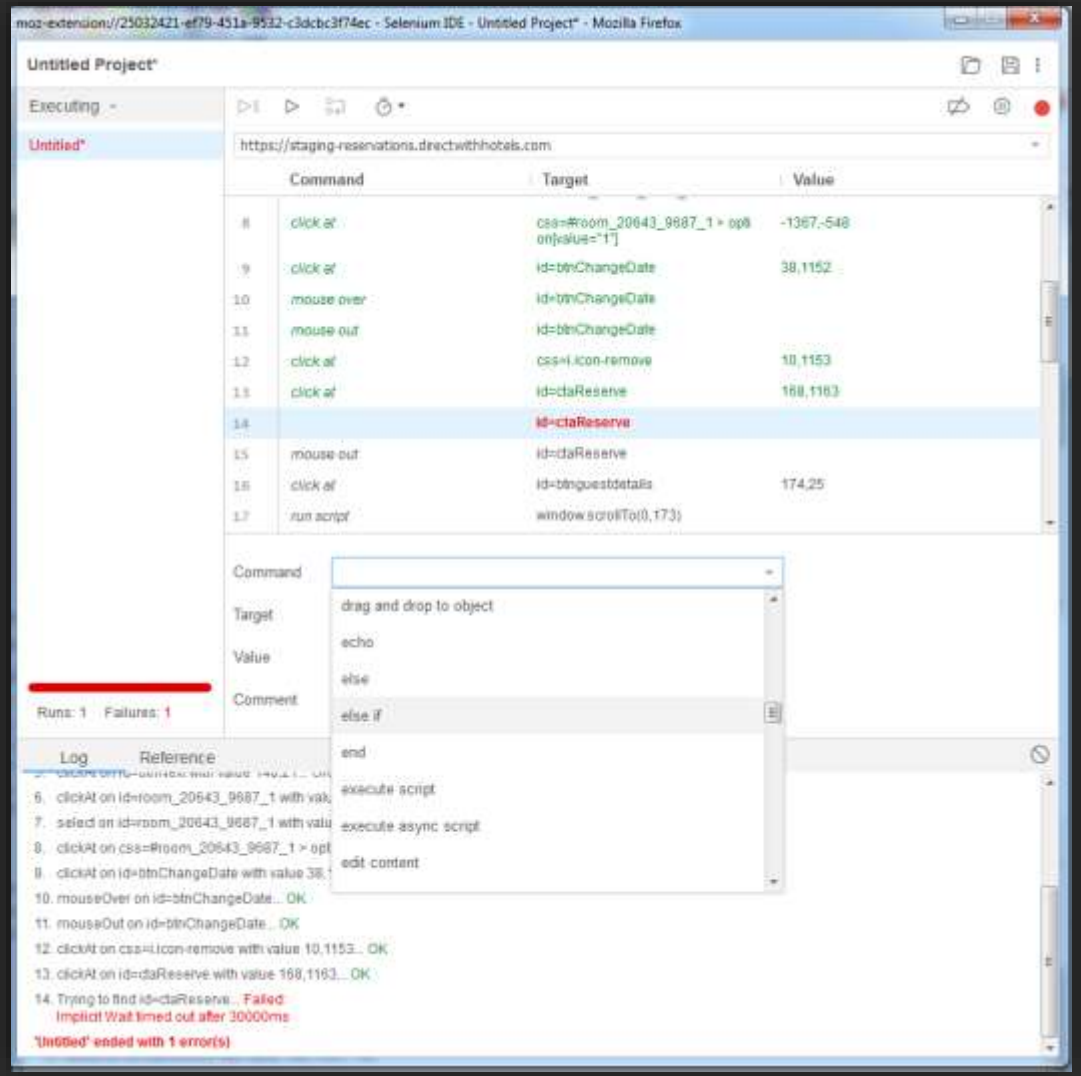
Why choose Playwright

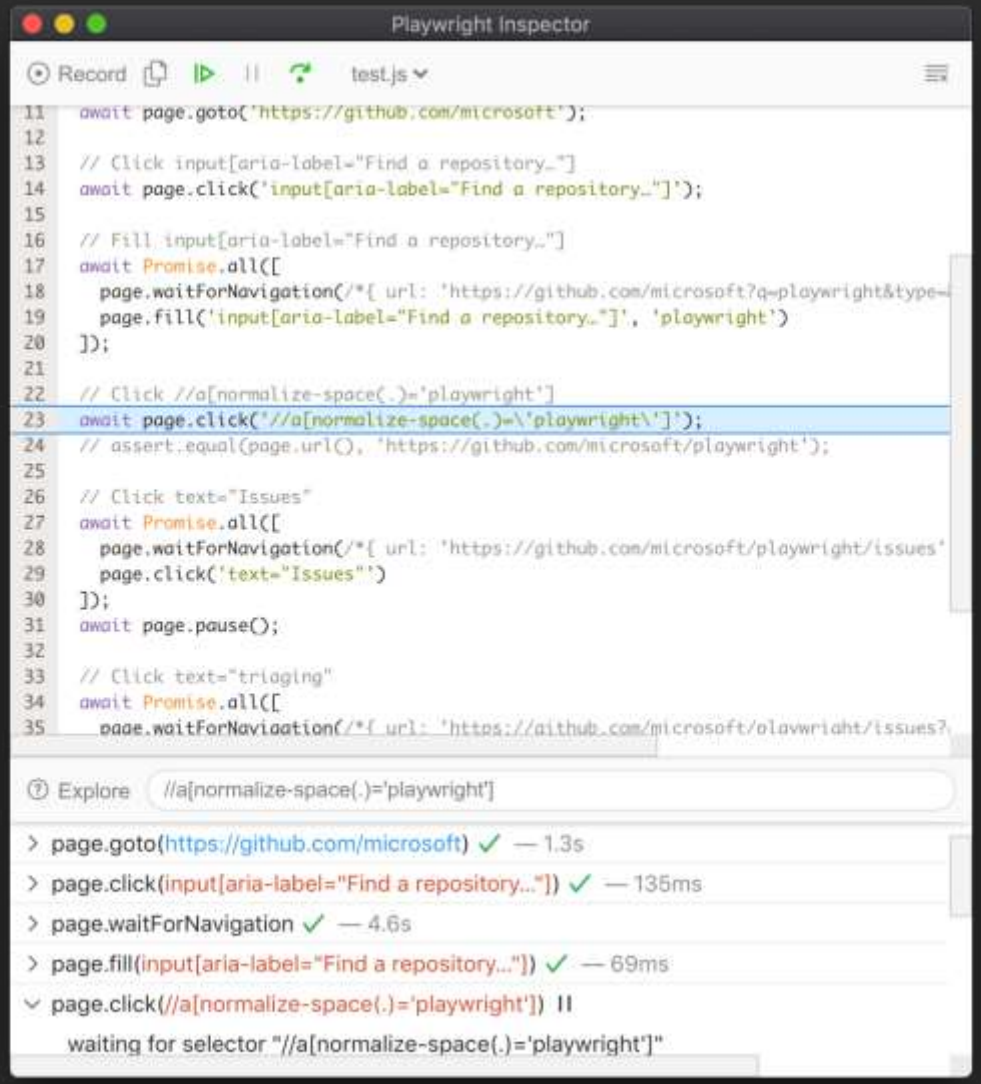




	Playwright	Selenium
Browsers	Chromium, Firefox, and WebKit	Chrome, Firefox, IE, Edge, Opera, Safari, and more
Language	Java, Python, C# JavaScript, Typescript	Java, Python, C# JavaScript, Ruby, Perl, PHP
Installation	Easy to Install	Easy to Install
Prerequisites	NodeJS	Java, Browser Drivers
OS Supported	Windows, Linux, and Mac OS	Windows, Linux, Solaris, and Mac OS
Open Source	Open Source and Free	Open Source and Free
Architecture	Headless Browser with event-driven architecture	Layered Architecture based on JSON Wire Protocol
Real Devices Support	Does not support real devices but supports emulators	Supports real device clouds and remote servers

IDE


Untitled Project*
Executing
Untitled*
https://staging-reservations.directwithhotels.com
Command
Target
Value
Comment
drag and drop to object
echo
else
else if
end
execute script
execute async script
edit content
Runs: 1 Failures: 1
Log
Reference
'Untitled' ended with 1 error(s)


Playwright Inspector
Record
test.js
await page.goto('https://github.com/microsoft');
// Click input[aria-label="Find a repository…"]
await page.click('input[aria-label="Find a repository…"]');
// Fill input[aria-label="Find a repository…"]
await Promise.all([
page.waitForNavigation(/*{ url: 'https://github.com/microsoft?q=playwright&type=
page.fill('input[aria-label="Find a repository…"]', 'playwright')
]);
// Click //a[normalize-space(.)='playwright']
await page.click('//a[normalize-space(.)=\'playwright\']');
// assert.equal(page.url(), 'https://github.com/microsoft/playwright');
// Click text="Issues"
await Promise.all([
page.waitForNavigation(/*{ url: 'https://github.com/microsoft/playwright/issues'
page.click('text="Issues"')
]);
await page.pause();
// Click text="triaging"
await Promise.all([
Explore
//a[normalize-space(.)='playwright']
page.goto(https://github.com/microsoft) — 1.3s
page.click(input[aria-label="Find a repository..."]) — 135ms
page.waitForNavigation — 4.6s
page.fill(input[aria-label="Find a repository..."]) — 69ms
page.click(//a[normalize-space(.)='playwright'])
waiting for selector "//a[normalize-space(.)='playwright']"

How to use Playwright



Playwright Install

- Node.js

```
# Run from your project's root directory
npm init playwright
# Or create a new project
npm init playwright new-project
```

- .NET

```
# Create project
dotnet new console -n PlaywrightDemo
cd PlaywrightDemo

# Add project dependency
dotnet add package Microsoft.Playwright
# Build the project
dotnet build
# Install required browsers - replace netX with actual output folder name, f.ex. net6.0.
pwsh bin\Debug\netX\playwright.ps1 install
```

Playwright

- Download browsers

```
npm i -D @playwright/test
# install supported browsers
npx playwright install
```

```
# Add project dependency
dotnet add package Microsoft.Playwright
# Build the project
dotnet build
# Install required browsers - replace netX
pwsh bin\Debug\netX\playwright.ps1 install
```

```
PS C:\Users\[redacted]\DevProjects\CodeTest\twMVC44\Dotnet\PlaywrightDemo> playwright install
Playwright build of chromium v956323 downloaded to C:\Users\[redacted]\AppData\Local\ms-playwright\chromium-956323
Playwright build of firefox v1313 downloaded to C:\Users\[redacted]\AppData\Local\ms-playwright\firefox-1313
Playwright build of webkit v1596 downloaded to C:\Users\[redacted]\AppData\Local\ms-playwright\webkit-1596
```

Playwright

- Record videos

```
var context = await browser.NewContextAsync(new BrowserNewContextOptions
{
    RecordVideoDir = "videos/"
});
// Make sure to close, so that videos are saved.
await context.CloseAsync();
```

- Screenshots

```
await page.screenshot({ path: 'screenshot.png' });
```

```
await page.GotoAsync("https://playwright.dev/dotnet");
await page.ScreenshotAsync(new PageScreenshotOptions { Path = "screenshot.png" });
```

Playwright

- Code generator and debugger

- Intercept network

```
public static async Task Main()
{
    using var playwright = await Playwright.CreateAsync();
    await using var browser = await playwright.Chromium.LaunchAsync();

    var page = await browser.NewPageAsync();
    await page.GotoAsync( url: "https://www.example.com");
    await page.Locator( selector: "text=Log in").ClickAsync(); //Text selector
    await page.Locator( selector: "button").ClickAsync(); //CSS selector
    await page.Locator( selector: "xpath=//button").ClickAsync(); //XPath selector
    await page.Locator( selector: ":nth-match(:text('Buy'), 3)").ClickAsync(); //Pick n-th match
    await page.Locator( selector: "input:right-of(:text(\"Username\"))").ClickAsync(); //Layout

    //experimental
    await page.Locator( selector: "_react=ListItem[text *= 'milk' i]").ClickAsync(); //React selector
    await page.Locator( selector: "_vue=list-item[text *= 'milk' i]").ClickAsync(); //Vue selector
}
```

CSS Seletors

Sample Code

CSS Selectors

CSS Selectors

In CSS, selectors are patterns used to select the element(s) you want to style.

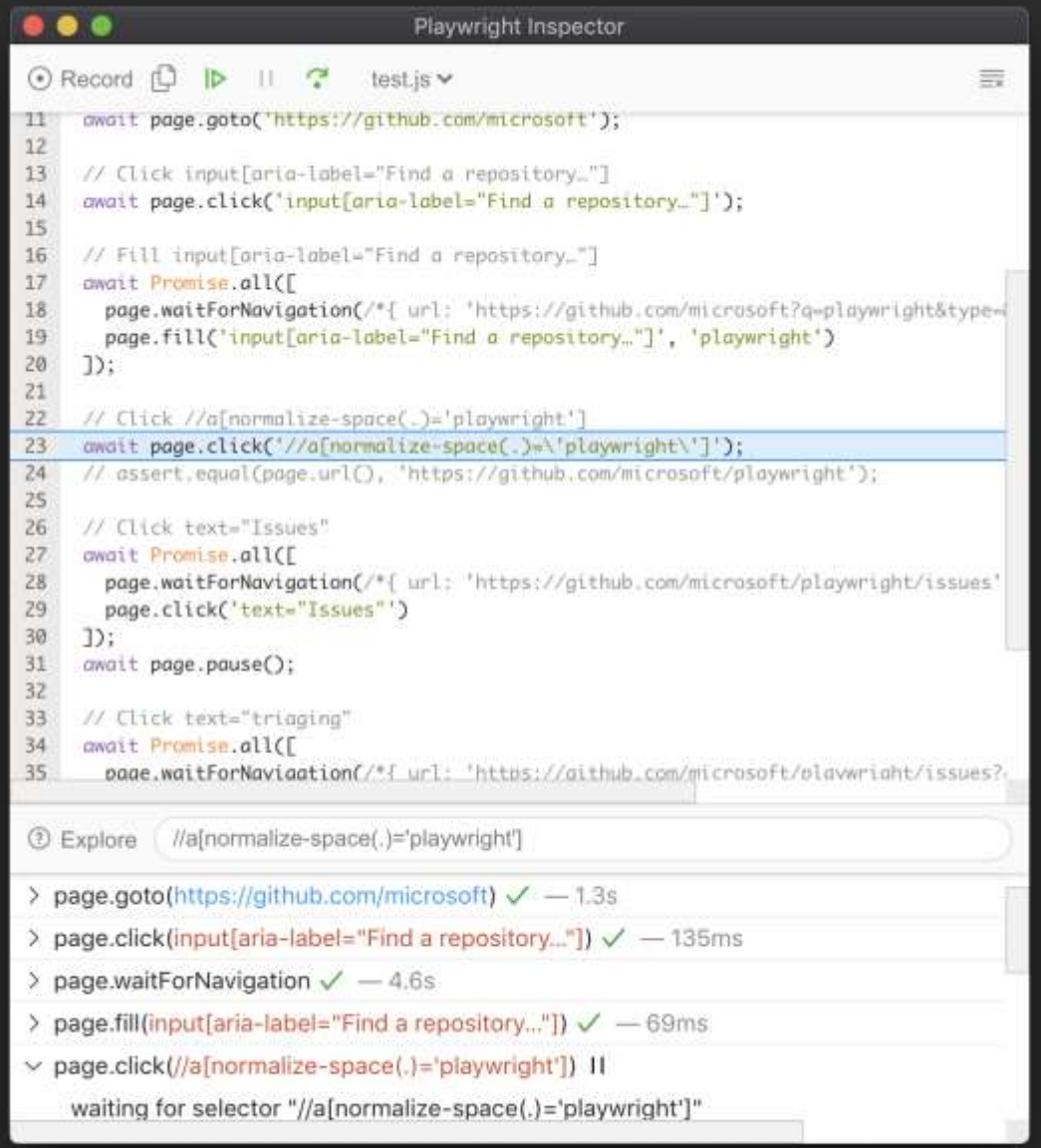
Use our CSS Selector Tester to demonstrate the different selectors.

Selector	Example	Example description
.class	.intro	Selects all elements with class="intro"
.class1.class2	.name1.name2	Selects all elements with both *name1* and *name2* set within its class attribute
.class1 .class2	.name1 .name2	Selects all elements with *name2* that is a descendant of an element with *name1*
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element.class	p.intro	Selects all <p> elements with class="intro"
element,element	div, p	Selects all elements and all <p> elements
element element	div p	Selects all <p> elements inside elements
element>element	div > p	Selects all <p> elements where the parent is a element
element+element	div + p	Selects the first <p> element that is placed immediately after elements
element1~element2	p ~ ul	Selects every element that is preceded by a <p> element
[*attribute*]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[*attribute|=value*]	[lang|=en]	Selects all elements with a lang attribute value equal to "en" or starting with "en-"

w3schools CSS Selectors

Playwright

- Inspector
  - Debugging Playwright scripts





Demo



是否發生過，一句話惹怒...

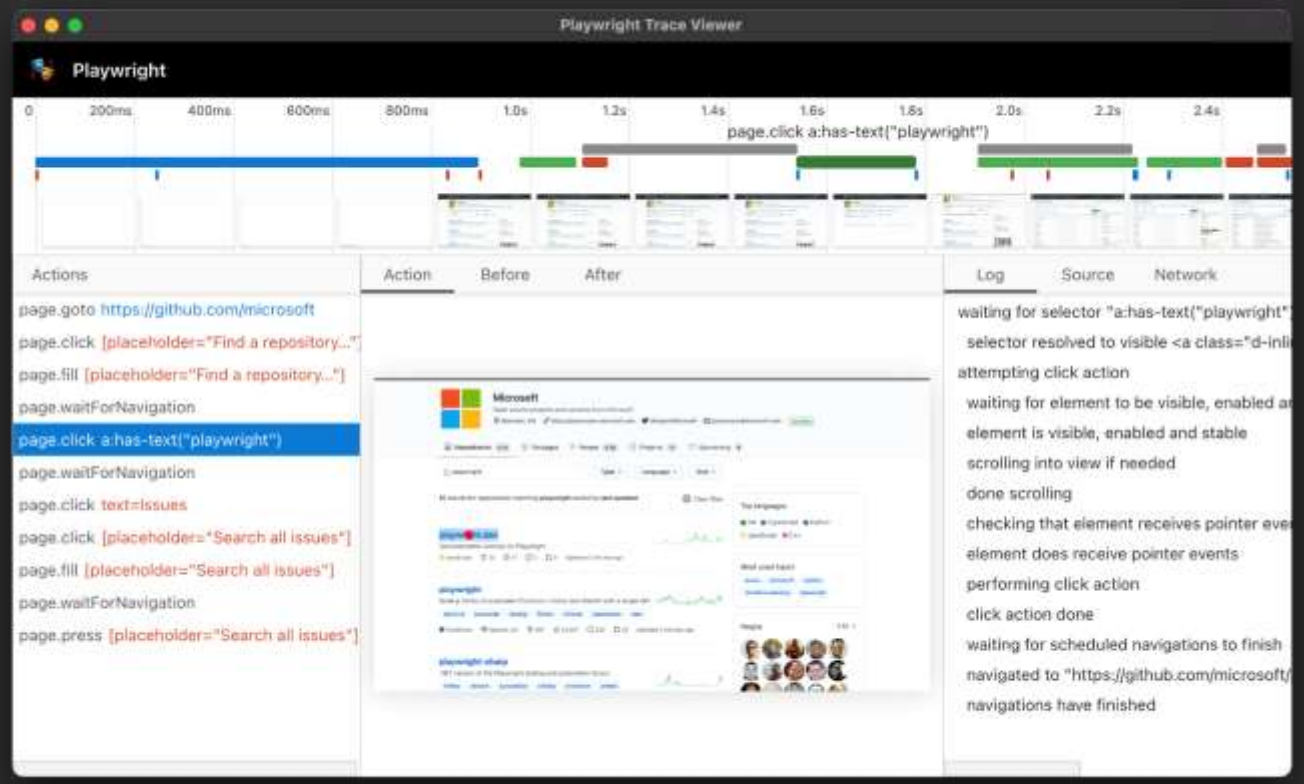
網站反應很慢

怎麼慢法 ?
大約回應多久



Playwright

- Trace Viewer
  - Recording trace
  - Viewing trace
  - Action
  - Screenshots
  - Snapshots





Demo



Getting started with .NET



Getting started with .NET

```
# Create project
dotnet new console -n PlaywrightDemo
cd PlaywrightDemo

# Add project dependency
dotnet add package Microsoft.Playwright
# Build the project
dotnet build
# Install required browsers - replace netX with actual output folder name, f.ex. net6.0.
pwsh bin\Debug\netX\playwright.ps1 install
```

Demo Code



Getting started with .NET

```
using Microsoft.Playwright;
using System.Threading.Tasks;

class Program
{
    public static async Task Main()
    {
        using var playwright = await Playwright.CreateAsync();
        await using var browser = await playwright.Chromium.LaunchAsync();
        var page = await browser.NewPageAsync();
        await page.GotoAsync("https://playwright.dev/dotnet");
        await page.ScreenshotAsync(new PageScreenshotOptions { Path = "screenshot.png" });
    }
}
```

Use Cases



Network – Abort Requests

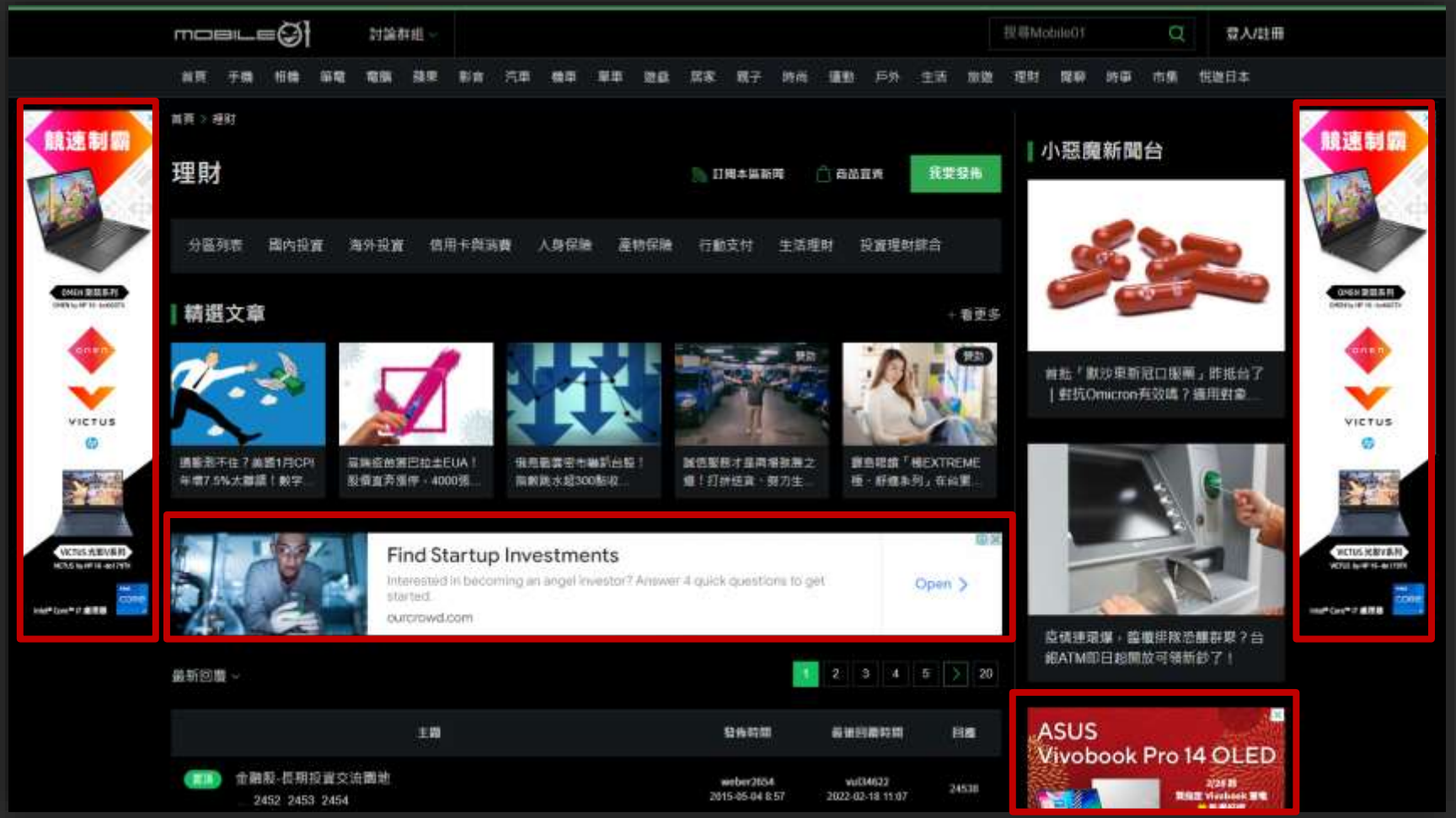


什麼情況下需要用到 Abort Requests



Network – Abort Requests

- Ads-blocker



Demo Code



```
using Microsoft.Playwright;

class Program
{
    public static async Task Main()
    {
        using var playwright = await Playwright.CreateAsync();
        await using var browser = await playwright.Chromium.LaunchAsync(new BrowserTypeLaunchOptions
        {
            Headless = false,
            SlowMo = 100000,
        }); // Task<IBrowser>

        var page = await browser.NewPageAsync();
        await page.RouteAsync( url: "**/*", async handler: route =>
        {
            if (route.Request.Url.StartsWith("https://securepubads.g.doubleclick.net"))
            {
                await route.AbortAsync();
            }
            else
            {
                await route.ContinueAsync();
            }
        }); // Task

        await page.GotoAsync( url: "https://www.mobile01.com/forumtopic.php?c=37");
    }
}
```

Network Abort Requests

Ads-blocker

Sample Code

Network - Abort Request

- 網站新手提示





novice-guide-demo From 黑暗執行緒

Demo Code



```
using Microsoft.Playwright;

class Program
{
    public static async Task Main()
    {
        using var playwright = await Playwright.CreateAsync();
        await using var browser = await playwright.Chromium.LaunchAsync(new BrowserTypeLaunchOptions
        {
            Headless = false,
            SlowMo = 100000,
        }); // Task<IBrowser>

        var page = await browser.NewPageAsync();
        await page.RouteAsync( url: "**/afet.novice-guide.js",  handler: async route => {
            await route.AbortAsync();
        }); // Task

        await page.GotoAsync( url: "https://darkthread.github.io/novice-guide-demo/");
    }
}
```

Network Abort Requests

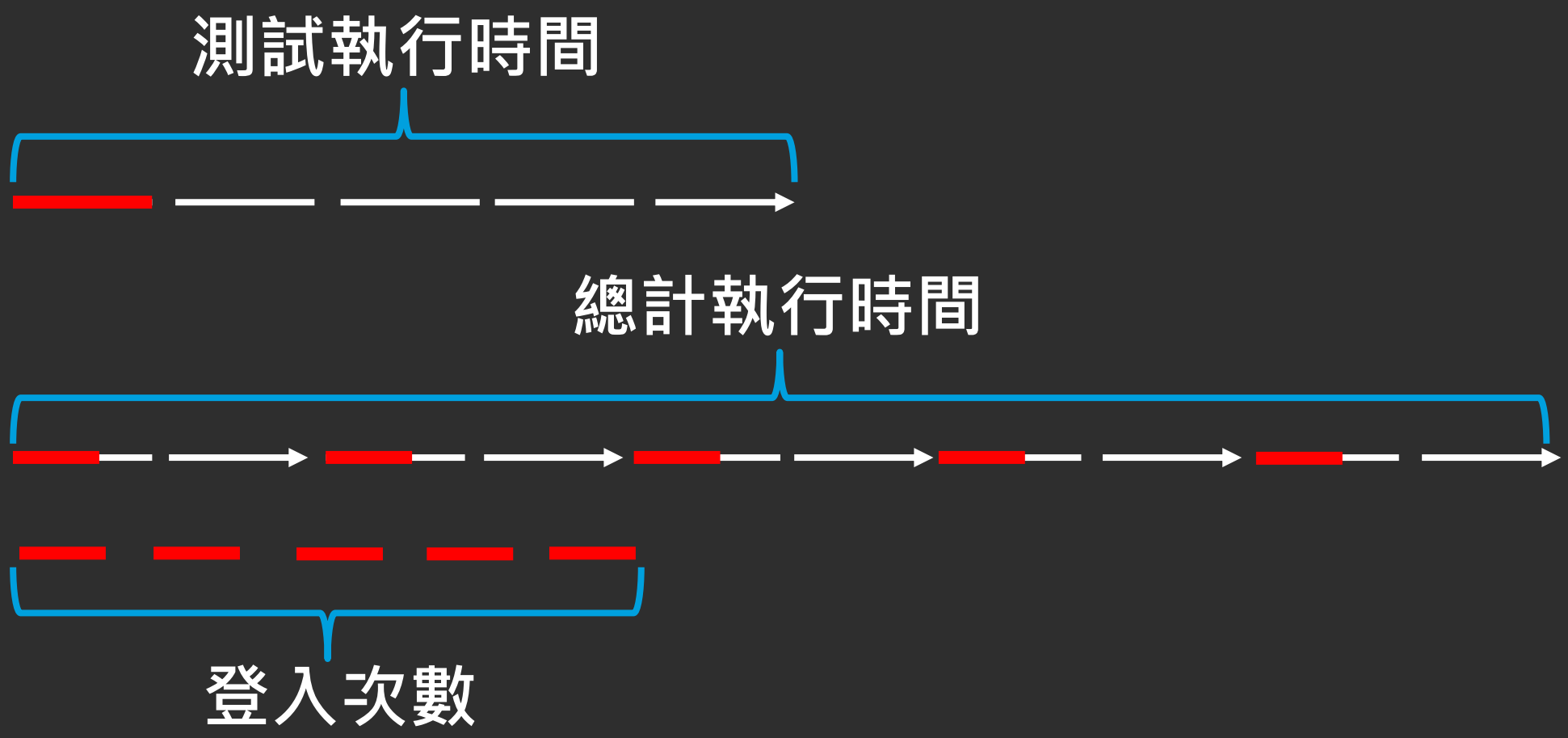
網站新手提示

Sample Code

Authentication



Authentication





tw MVC
https://mvc.tw

Authentication

```
var page = await context.NewPageAsync();
await page.NavigateAsync("https://github.com/login");
// Interact with login form
await page.ClickAsync("text=Login");
await page.FillAsync("input[name='login']", USERNAME);
await page.FillAsync("input[name='password']", PASSWORD);
await page.ClickAsync("text=Submit");
// Verify app is logged in
```

```
// Save storage state into the file.
await context.StorageStateAsync(new BrowserContextStorageStateOptions
{
    Path = "state.json"
});
```

Demo Code



總結



總結 End-to-End Test (RD & QA)

- 提升開發效率 => 透過現代化工具
- 提升溝通效率 => 透過 Trace Log
- 提升測試效率 => 透過 Network Abort Requests
  - 目標關注在真正需要測試的內容: Ads-blocker / 網站新手提示
  - 降低重複性過多的步驟: Authentication

參考資料



參考資料

- Playwright.dev
- microsoft/playwright

Blog 是記錄知識的最佳平台



https://dotblogs.com.tw

SkillTree 為了確保內容與實務不會脫節，我們都是聘請企業顧問等級並且目前依然在職場的業界講師，我們不把時間浪費在述說歷史與沿革，我們並不是教您考取證照，而是教您如何上場殺敵，拳拳到肉的內容才是您花錢想要聽到的，而這也剛好是我們擅長的。



https://skilltree.my

天瓏資訊圖書

