Playwright **Test**

Playful testing **framework**

What is Playwright Test?

- Cross-browser Web Testing Framework
- Node.js: JavaScript / TypeScript
- Free, Open Source, Sponsored by Microsoft
- Extensively used in the industry

Why yet another test runner?

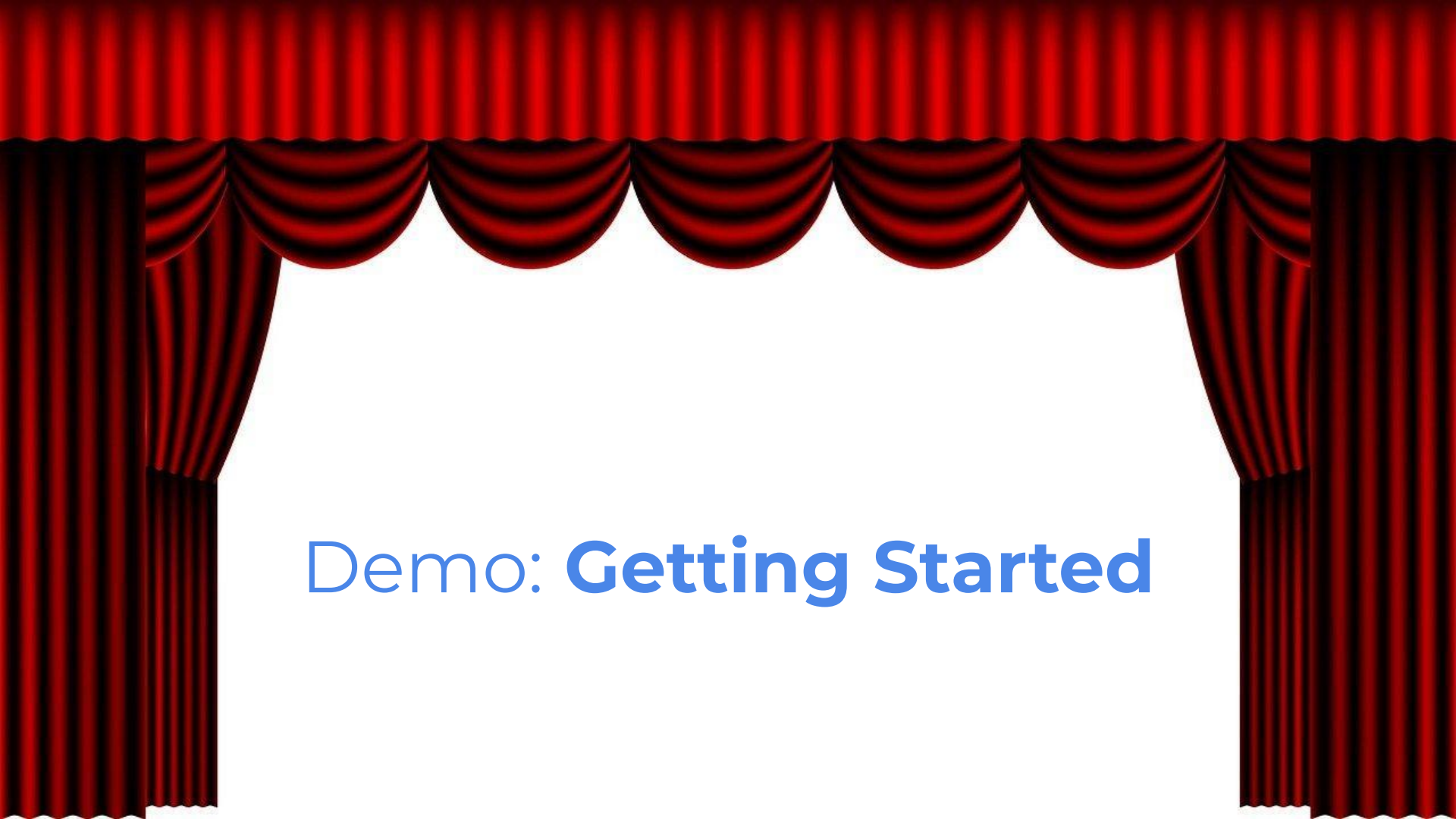
- Historically, JavaScript test frameworks are built for **unit tests**
- Playwright Test is built for **end-to-end tests**:
  - Cross-browser — Chrome, Firefox & Safari
  - Parallelisation — tests are fast
  - Isolation — zero-overhead test isolation
  - Flexibility — pytest-like fixture configuration

Agenda

1. Getting Started
2. Fundamentals
3. Configuration
4. Playwright Inspector & CodeGen
5. Playwright Tracing

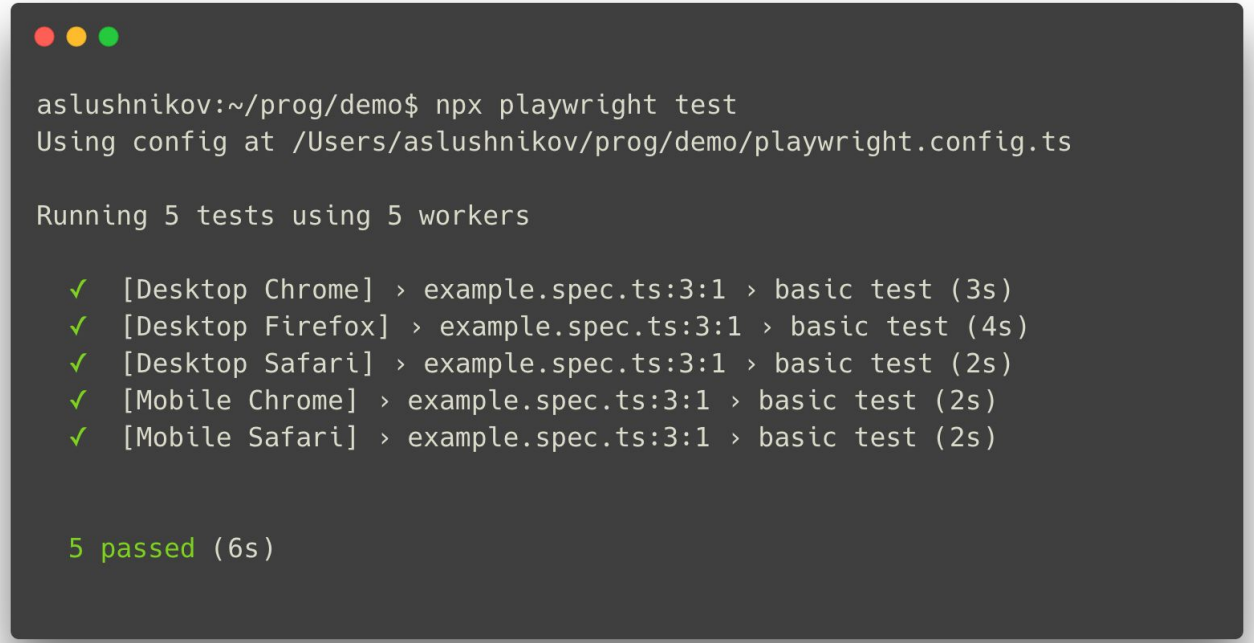
Chapter 1

Getting **Started**


Demo: **Getting Started**

Installation: **npm init playwright**

Running: **npx playwright test**



Test: **e2e/example.spec.ts**

```
1 import { test, expect } from '@playwright/test';
2
3 test('basic test', async ({ page }) => {
4   await page.goto('https://playwright.dev/');
5   await page.locator('text=Get started').click();
6   await expect(page).toHaveTitle(/Getting started/);
7 });
```

Test: **e2e/example.spec.ts**

1. **Test Isolation**

```
1 import { test, expect } from '@playwright/test';
2
3 test('basic test', async ({ page }) => {
4   await page.goto('https://playwright.dev/');
5   await page.locator('text=Get started').click();
6   await expect(page).toHaveTitle(/Getting started/);
7 });
```

Test: **e2e/example.spec.ts**

1. **Test Isolation**

2. **Auto-waiting**

Test: **e2e/example.spec.ts**

1. **Test Isolation**

2. **Auto-waiting**

3. **Web-First Assertions**

Chapter 2

Fundamentals

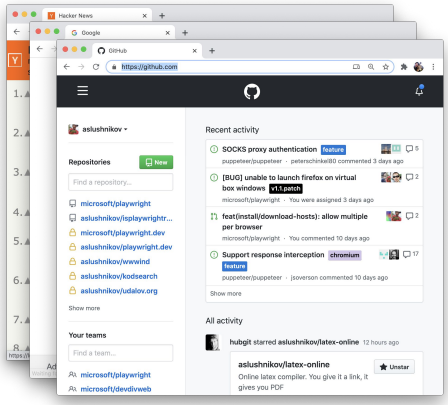
Fundamentals: **Test Isolation**

❌ **Old-School:** Browser Restart

- Slow instantiation (>100ms)
- Huge memory overhead

✅ **Playwright Test:** Browser Contexts

- Full isolation
- Fast instantiation (~1ms)
- Low overhead

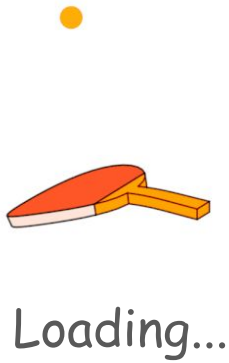


Browser Context

Fundamentals: **Auto-waiting**

❌ **Old-School**: timeouts to await elements

- Time **does not exist** in the cloud
- Timeouts are inefficient



✅ **Playwright Test:** built-in **auto-waiting**

- Just Works!
- Happens for all actions (e.g. `click`, `fill`, `press`)
- No need for `setTimeout` calls

Fundamentals: **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	-
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	-	-	-	-
dispatchEvent	Yes	-	-	-	-	-

Fundamentals: **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	-
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	-	-	-	-
dispatchEvent	Yes	-	-	-	-	-

Fundamentals: **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	-
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	-	-	-	-
dispatchEvent	Yes	-	-	-	-	-

Fundamentals: **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	-
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	-	-	-	-
dispatchEvent	Yes	-	-	-	-	-

Fundamentals: **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	-
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	-	-	-	-
dispatchEvent	Yes	-	-	-	-	-

Fundamentals: **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	-
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	-	-	-	-
dispatchEvent	Yes	-	-	-	-	-

Fundamentals: **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	-
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	-	-	-	-
dispatchEvent	Yes	-	-	-	-	-

Fundamentals: **Web-First Assertions**

❌ **Old-School**: assert current state

- Web Applications are highly dynamic
- State is always in flux

✅ **Playwright Test:** declare expected state

- Wait until the declared state is reached
- Web-First assertions

Fundamentals: **Web-First Assertions**

```
expect(locator).toBeChecked()
expect(locator).toBeDisabled()
expect(locator).toBeEditable()
expect(locator).toBeEmpty()
expect(locator).toBeEnabled()
expect(locator).toBeFocused()
expect(locator).toBeHidden()
expect(locator).toBeVisible()
expect(locator).toContainText(text)
expect(locator).toHaveAttribute(name)
```

```
expect(locator).toHaveClass(expected)
expect(locator).toHaveCount(count)
expect(locator).toHaveCSS(name, value)
expect(locator).toHaveId(id)
expect(locator).toHaveJSProperty(name, value)
expect(locator).toHaveText(expected)
expect(page).toHaveTitle(title)
expect(page).toHaveURL(url)
expect(locator).toHaveValue(value)
```

📍 Locators API

- `Locator := (page, selector)`
- Create locators with `page.locator(selector)`
- Represents a view to the element(s) on the page
- Re-queries page on each method call
- **“strict” by default**
- Useful in POMs

Fundamentals: **Web-First Assertions**

```
await expect(page.locator('.products .item')).toHaveText(['soap', 'rope']);
```

Fundamentals: **Web-First Assertions**

```
await expect(page.locator('.products .item')).toHaveText(['soap', 'rope']);
```

1. Must be **awaited**

Fundamentals: **Web-First Assertions**

```
await expect(page.locator('.products .item')).toHaveText(['soap', 'rope']);
```

1. Must be **awaited**

2. **Re-queries** given locator

Fundamentals: **Web-First Assertions**

Chapter 3

Configuration

```ts
// example.spec.ts
import { test, expect } from '@playwright/test';

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  await page.locator('text=Get started').click();
  await expect(page).toHaveTitle(/Getting started/);
});
```

```ts
// example.spec.ts
import { test, expect } from '@playwright/test';

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  await page.locator('text=Get started').click();
  await expect(page).toHaveTitle(/Getting started/);
});
```

```ts
// example.spec.ts
import { test, expect } from '@playwright/test';

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  await page.locator('text=Get started').click();
  await expect(page).toHaveTitle(/Getting started/);
});
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  projects: [

  ],
};
export default config;
```

```ts
// example.spec.ts
import { test, expect } from '@playwright/test';

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  await page.locator('text=Get started').click();
  await expect(page).toHaveTitle(/Getting started/);
});
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },

  ],
};
export default config;
```


// example.spec.ts
import { test, expect } from '@playwright/test';
Run 1
test('basic test', async ({ page }) => {
await page.goto('https://playwright.dev/');
await page.locator('text=Get started').click();
await expect(page).toHaveTitle(/Getting started/);
});
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
projects: [
{
name: 'Desktop Chrome',
use: { browserName: 'chromium', },
},
],
};
export default config;

```ts
// example.spec.ts
import { test, expect } from '@playwright/test';

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  await page.locator('text=Get started').click();
  await expect(page).toHaveTitle(/Getting started/);
});
```


```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// example.spec.ts
import { test, expect } from '@playwright/test';

test('basic test', async ({ page }) => {
  await page.goto('https://playwright.dev/');
  await page.locator('text=Get started').click();
  await expect(page).toHaveTitle(/Getting started/);
});
```

```
aslushnikov:~/prog/demo$ npx playwright test
Using config at /Users/aslushnikov/prog/demo/playwright.config.ts

Running 3 tests using 3 workers

  ✓  [Desktop Chrome] › example.spec.ts:3:1 › basic test (3s)
  ✓  [Desktop Firefox] › example.spec.ts:3:1 › basic test (4s)
  ✓  [Desktop Safari] › example.spec.ts:3:1 › basic test (2s)

  3 passed (4s)
```

Run 1

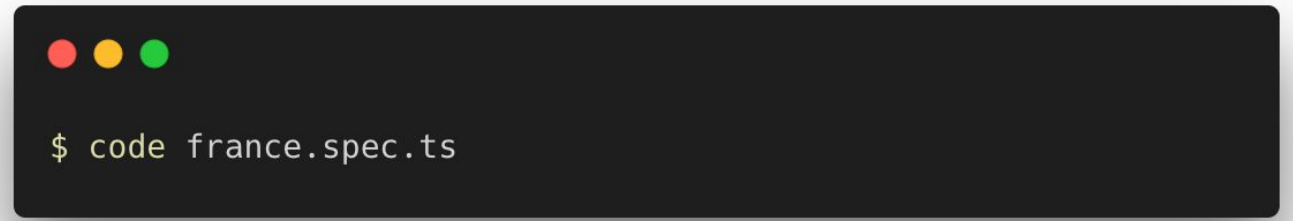
Run 2

Run 3

```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

Granular Configuration: **france.spec.ts**

- Per-file configuration
- Per-suite configuration



```ts
// france.spec.ts
import { test, expect } from '@playwright/test';
```

TypeScript

```ts
// france.spec.ts
import { test, expect } from '@playwright/test';
// per-file configuration
test.use({ locale: 'fr-FR', timezoneId: 'Europe/Paris' });
```

TypeScript

TypeScript

```ts
// france.spec.ts
import { test, expect } from '@playwright/test';
// per-file configuration
test.use({ locale: 'fr-FR', timezoneId: 'Europe/Paris' });

test('should work', async ({ page }) => { /* ... test goes here ... */ });
test('should use euro', async ({ page }) => { /* ... */ });
```

TypeScript

```ts
// france.spec.ts
import { test, expect } from '@playwright/test';
// per-file configuration
test.use({ locale: 'fr-FR', timezoneId: 'Europe/Paris' });

test('should work', async ({ page }) => { /* ... test goes here ... */ });
test('should use euro', async ({ page }) => { /* ... */ });

test.describe('light theme', () => {


});
```

TypeScript

```ts
// france.spec.ts
import { test, expect } from '@playwright/test';
// per-file configuration
test.use({ locale: 'fr-FR', timezoneId: 'Europe/Paris' });

test('should work', async ({ page }) => { /* ... test goes here ... */ });
test('should use euro', async ({ page }) => { /* ... */ });

test.describe('light theme', () => {
  test.use({ colorScheme: 'light' }); // per-suite configuration

});
```

TypeScript

```ts
// france.spec.ts
import { test, expect } from '@playwright/test';
// per-file configuration
test.use({ locale: 'fr-FR', timezoneId: 'Europe/Paris' });

test('should work', async ({ page }) => { /* ... test goes here ... */ });
test('should use euro', async ({ page }) => { /* ... */ });

test.describe('light theme', () => {
  test.use({ colorScheme: 'light' }); // per-suite configuration
  test('should be light', async ({ page }) => { /* ... */ });
});
```

TypeScript

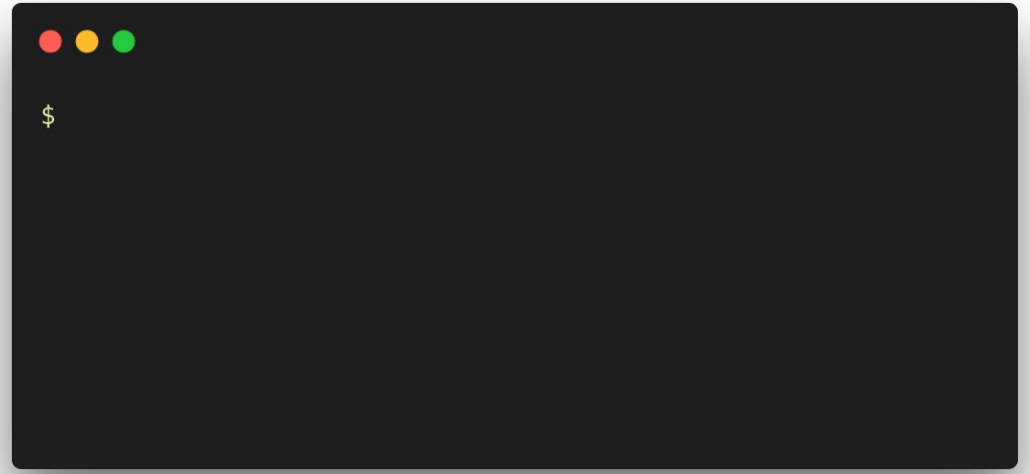
```ts
// france.spec.ts
import { test, expect } from '@playwright/test';
// per-file configuration
test.use({ locale: 'fr-FR', timezoneId: 'Europe/Paris' });

test('should work', async ({ page }) => { /* ... test goes here ... */ });
test('should use euro', async ({ page }) => { /* ... */ });

test.describe('light theme', () => {
  test.use({ colorScheme: 'light' }); // per-suite configuration
  test('should be light', async ({ page }) => { /* ... */ });
});
test.describe('dark theme', () => {
  test.use({ colorScheme: 'dark' }); // per-suite configuration
  test('should be dark', async ({ page }) => { /* ... */ });
});
```

Configuration Options

https://aka.ms/playwright/fixtures

- acceptDownloads
- baseURL
- browserName
- bypassCSP
- channel
- colorScheme
- deviceScaleFactor
- extraHTPHeaders
- geolocation
- hasTouch
- headless
- httpCredentials
- ignoreHTTPSErrors
- javaScriptEnabled
- launchOptions
- locale
- offline
- permissions
- proxy
- screenshot
- storageState
- timezoneId
- trace
- userAgent
- video
- viewport

Configuration: **Data-Driven Tests**



Configuration: **Data-Driven Tests**

Configuration: **Data-Driven Tests**

```
$ cat urls.json
[
  "https://playwright.dev",
  "https://playwright.dev/docs/why-playwright",
  "https://playwright.dev/docs/intro",
  "https://playwright.dev/docs/core-concepts",
  "https://playwright.dev/docs/cli",
  "https://playwright.dev/docs/debug"
]
$ code check-urls.spec.ts
```

TypeScript

```ts
// check-urls.spec.ts
import { test, expect } from '@playwright/test';
```

TypeScript

```
// check-urls.spec.ts
import { test, expect } from '@playwright/test';

const urls = require('./urls.json');
```

TypeScript

```
// check-urls.spec.ts
import { test, expect } from '@playwright/test';

const urls = require('./urls.json');
for (const url of urls) {



}
```

TypeScript

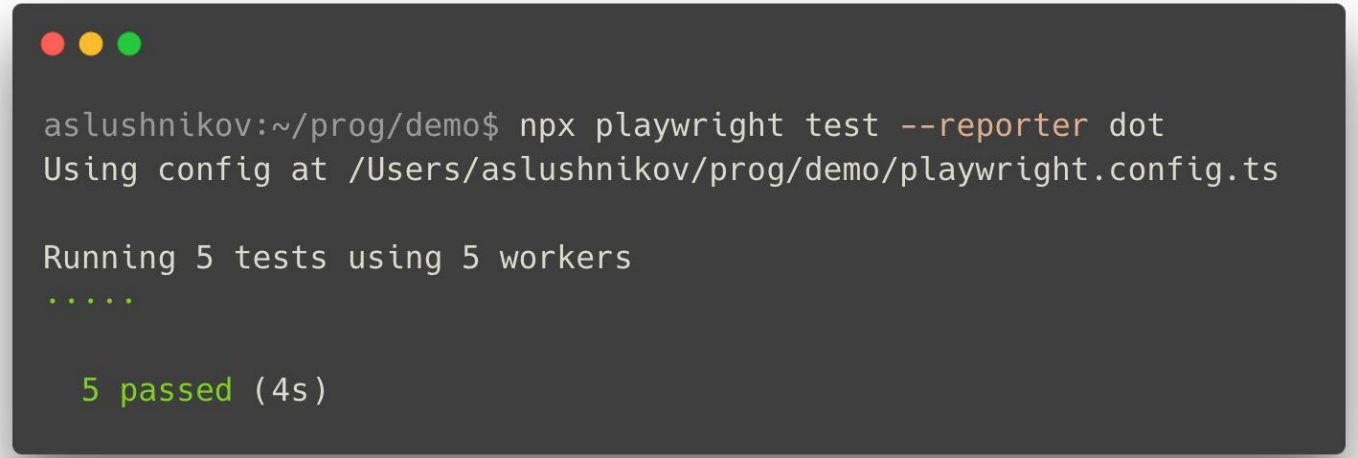
```
// check-urls.spec.ts
import { test, expect } from '@playwright/test';

const urls = require('./urls.json');
for (const url of urls) {
  test(`check ${url}`, async ({ page }) => {
    await page.goto(url);
  });
}
```

TypeScript

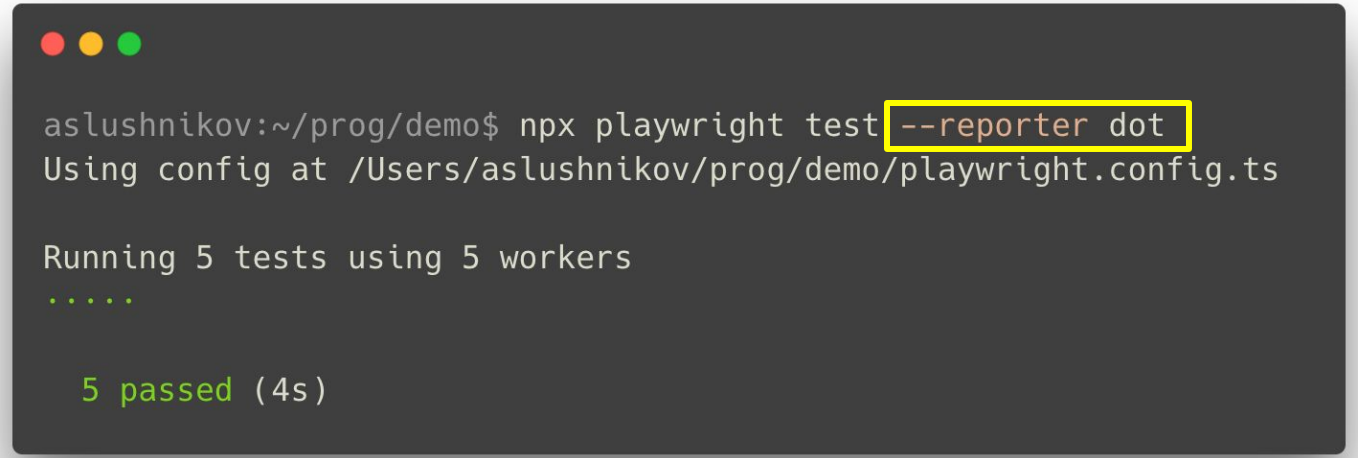
```
// check-urls.spec.ts
import { test, expect } from '@playwright/test';

const urls = require('./urls.json');
for (const url of urls) {
  test(`check ${url}`, async ({ page }) => {
    await page.goto(url);
  });
}
```

NOTE: Make sure to have different test titles

Configuration: **Reporters**



Configuration: **Reporters**



```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: 'dot',
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: process.env.CI ? 'dot' : 'line',
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

Built-in reporters

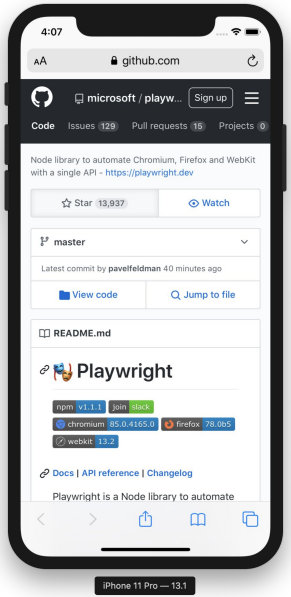
- ➔ dot
- ➔ list
- ➔ line
- ➔ json
- ➔ junit

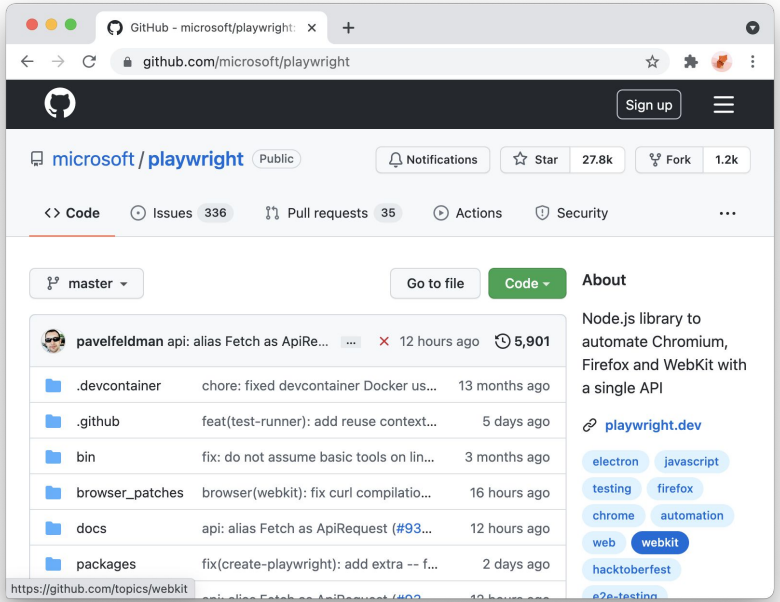
Third-party reporters

- ➔ allure-playwright

https://aka.ms/playwright/reporters

Configuration: **Devices**





```ts
// playwright.config.ts
import { PlaywrightTestConfig } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: { browserName: 'chromium', },
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: devices['Desktop Chrome'],
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: devices['Desktop Chrome'],
    },
    {
      name: 'Desktop Firefox',
      use: { browserName: 'firefox', },
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

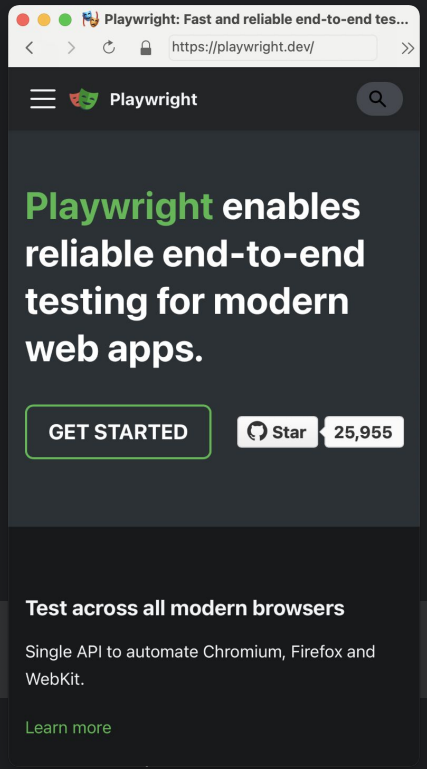
```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: devices['Desktop Chrome'],
    },
    {
      name: 'Desktop Firefox',
      use: devices['Desktop Firefox'],
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: devices['Desktop Chrome'],
    },
    {
      name: 'Desktop Firefox',
      use: devices['Desktop Firefox'],
    },
    {
      name: 'Desktop Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: devices['Desktop Chrome'],
    },
    {
      name: 'Desktop Firefox',
      use: devices['Desktop Firefox'],
    },
    {
      name: 'Mobile Safari',
      use: { browserName: 'webkit', },
    }
  ],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: devices['Desktop Chrome'],
    },
    {
      name: 'Desktop Firefox',
      use: devices['Desktop Firefox'],
    },
    {
      name: 'Mobile Safari',
      use: devices['iPhone 12 Pro'],
    }
  ],
};
export default config;
```

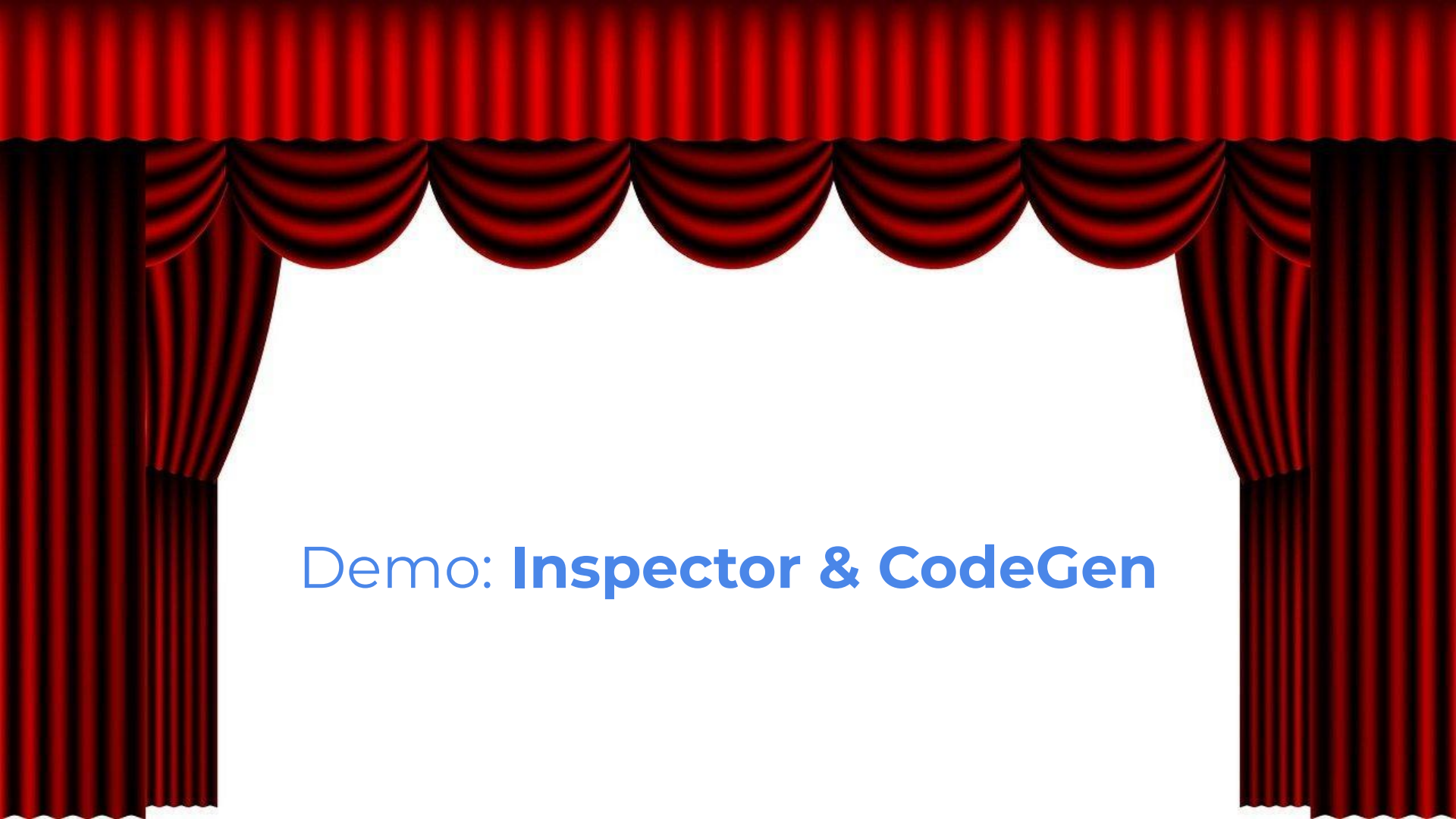
```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: [
    process.env.CI ? ['dot'] : ['list'],
    ['json', { outputFile: 'test-results.json' }],
  ],
  projects: [
    {
      name: 'Desktop Chrome',
      use: devices['Desktop Chrome'],
    },
    {
      name: 'Desktop Firefox',
      use: devices['Desktop Firefox'],
    },
    {
      name: 'Mobile Safari',
      use: devices['iPhone 12 Pro'],
    }
  ],
};
export default config;
```



Chapter 4

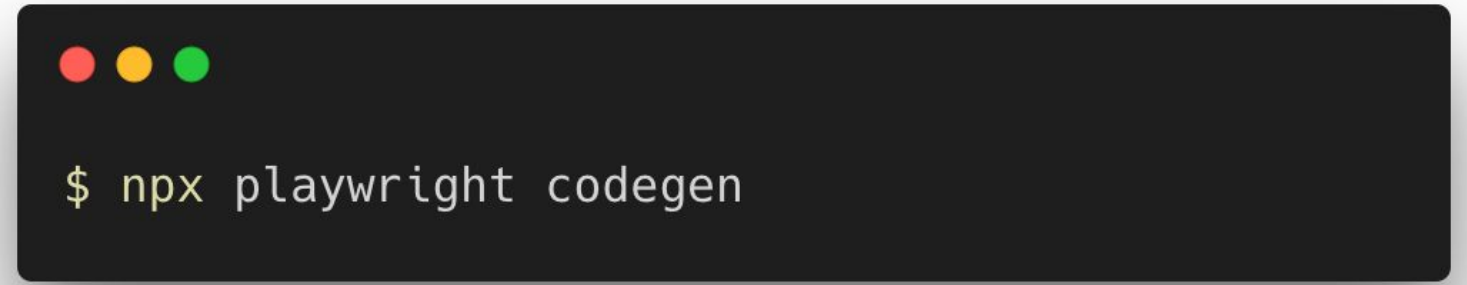
Playwright

Inspector & Codegen


Demo: **Inspector & CodeGen**

Playwright **Inspector**

Playwright **Code Generation**

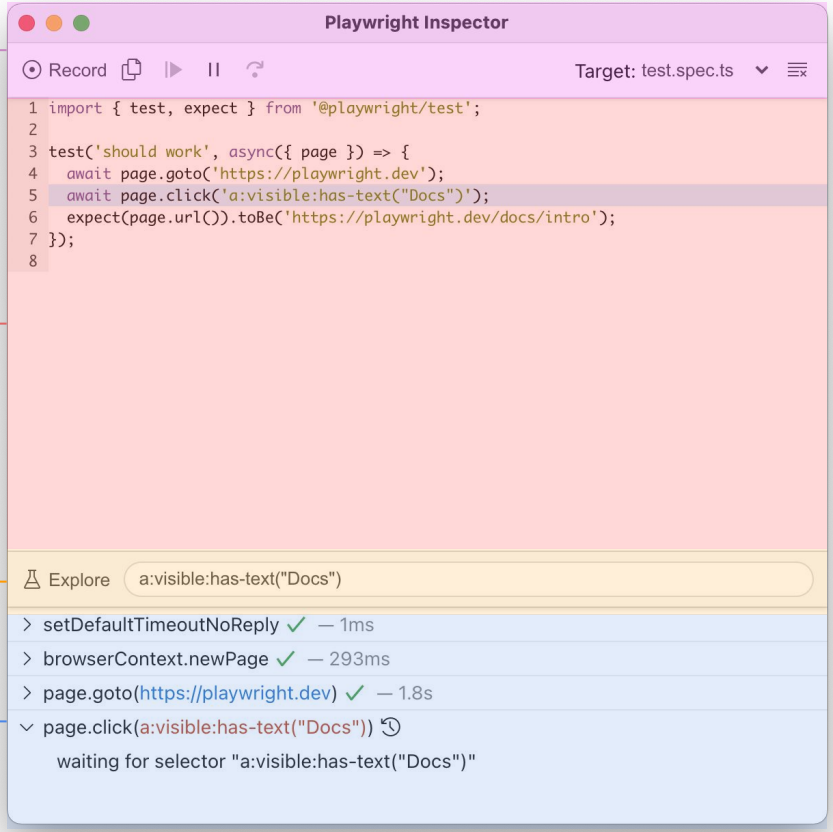


Control Panel

Source Code

Selectors Playground

Actions



Playwright **Inspector**

Chapter 5

Post-mortem Debugging
Playwright Tracing

Post-Mortem Debugging

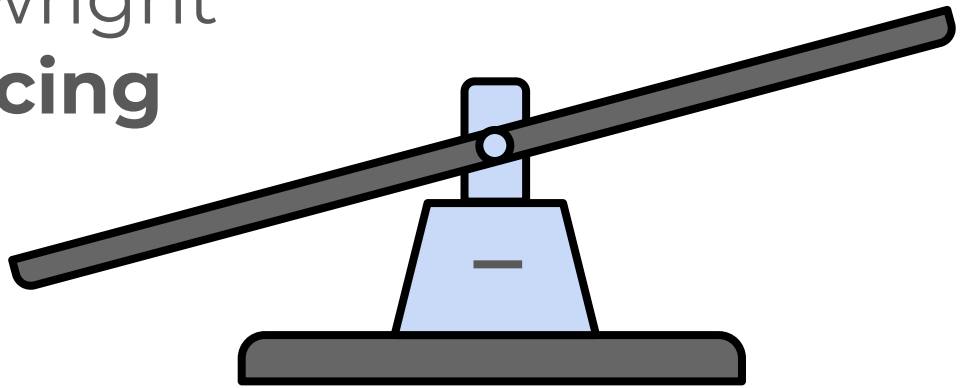
- **Post-Mortem Debugging –** debugging test failures on CI without being able to debug locally.
- **Test Artifacts –** any by-product of test running that helps debug test failures.
  - Logs
  - Screenshots
  - Videos

Unique Artifact: **Tracing**

Playwright
Tracing

videos

screenshots



Playwright **Tracing**

Playwright **Tracing**

trace.zip files

- Playwright actions
- Playwright events
- Screencast
- Network log
- Console log
- DOM snapshots 🔥

Playwright **Tracing**

trace.zip files

- Playwright actions
- Playwright events
- Screencast
- Network log
- Console log
- DOM snapshots 🔥

Trace Viewer

- GUI tool to explore trace files
- Bundled with Playwright

Playwright Tracing: **Workflow**

 **Enable** trace collection in `playwright.config.ts`

 **Setup** CI to upload trace files

 **Download** & **Inspect** trace files with Playwright Trace Viewer

TypeScript

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: process.env.CI ? 'dot' : 'line',


  projects: [{
    name: 'Desktop Chrome',
    use: devices['Desktop Chrome'],
  }, {
    name: 'Desktop Firefox',
    use: devices['Desktop Firefox'],
  }, {
    name: 'Mobile Safari',
    use: devices['iPhone 12 Pro'],
  }],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: process.env.CI ? 'dot' : 'line',
  retries: 2,


  projects: [{
    name: 'Desktop Chrome',
    use: devices['Desktop Chrome'],
  }, {
    name: 'Desktop Firefox',
    use: devices['Desktop Firefox'],
  }, {
    name: 'Mobile Safari',
    use: devices['iPhone 12 Pro'],
  }],
};
export default config;
```

TypeScript

TypeScript

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: process.env.CI ? 'dot' : 'line',
  retries: 2,
  use: {
    trace: 'on-first-retry',
  },
  projects: [{
    name: 'Desktop Chrome',
    use: devices['Desktop Chrome'],
  }, {
    name: 'Desktop Firefox',
    use: devices['Desktop Firefox'],
  }, {
    name: 'Mobile Safari',
    use: devices['iPhone 12 Pro'],
  }],
};
export default config;
```

```ts
// playwright.config.ts
import { PlaywrightTestConfig, devices } from '@playwright/test';
const config: PlaywrightTestConfig = {
  reporter: process.env.CI ? 'dot' : 'line',
  retries: 2,
  use: {
    trace: 'on-first-retry',
  },
  projects: [{
    name: 'Desktop Chrome',
    use: devices['Desktop Chrome'],
  }, {
    name: 'Desktop Firefox',
    use: devices['Desktop Firefox'],
  }, {
    name: 'Mobile Safari',
    use: devices['iPhone 12 Pro'],
  }],
};
export default config;
```

TypeScript

Enabling Trace Collection

```yaml
# .github/workflows/tests.yml
on: [push]
jobs:
  run_tests:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm ci
    - run: npx playwright install --with-deps
    - run: npm run test:e2e
```

Github Actions

```yaml
# .github/workflows/tests.yml
on: [push]
jobs:
  run_tests:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm ci
    - run: npx playwright install --with-deps
    - run: npm run test:e2e
    - uses: actions/upload-artifact@v2
      if: always()
      with:
        name: test-results
        path: test-results
```

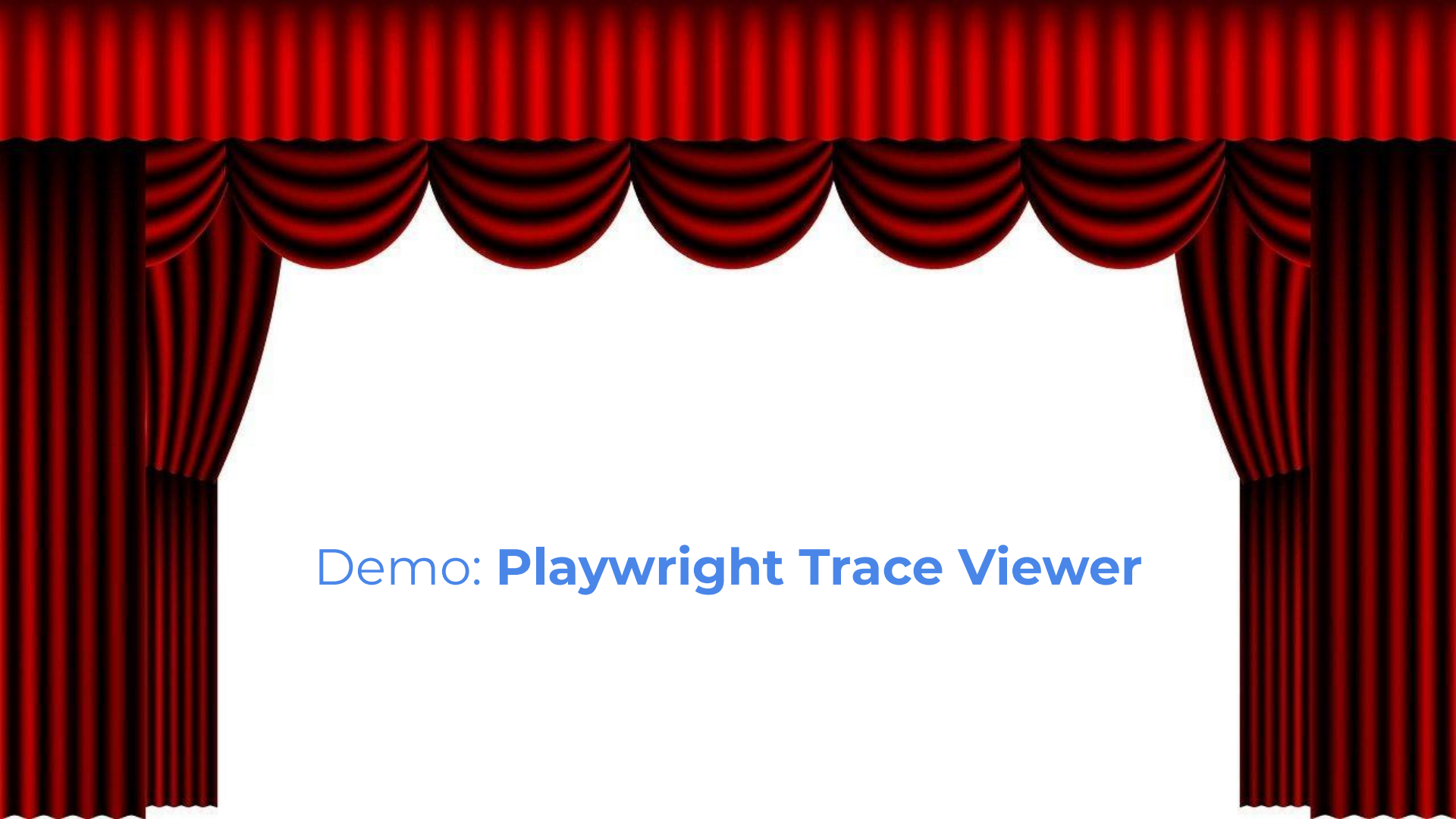
Github Actions

Uploading Artifacts

Github Actions

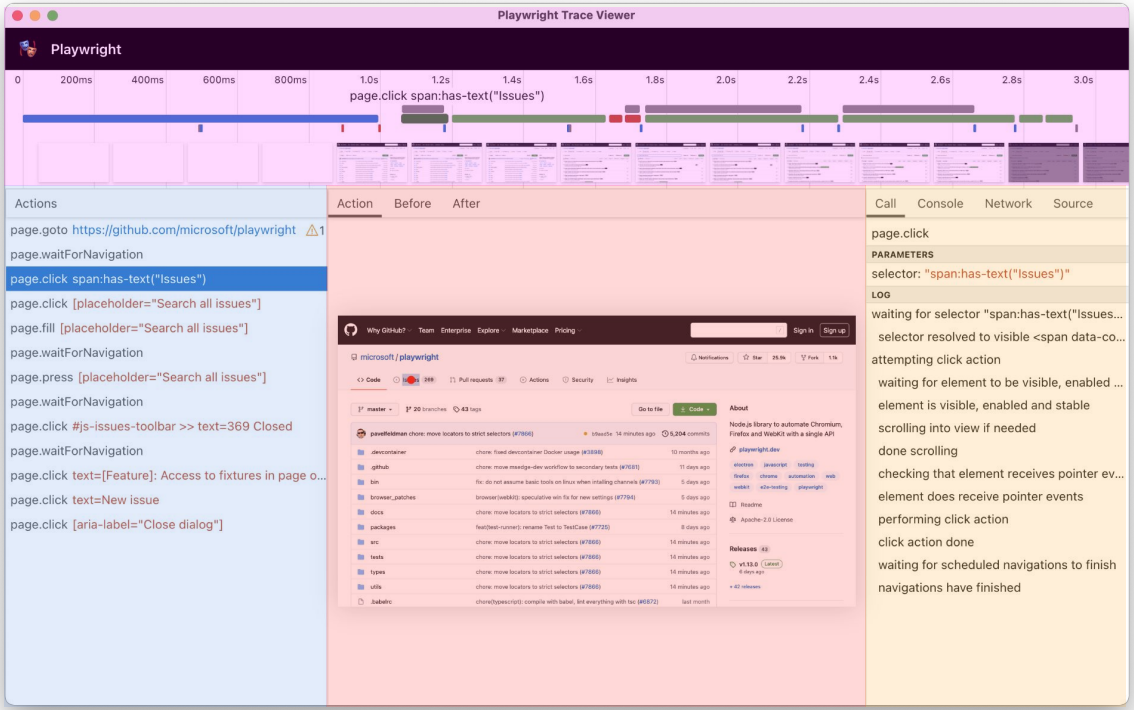
```yaml
# .github/workflows/tests.yml
on: [push]
jobs:
  run_tests:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm ci
    - run: npx playwright install --with-deps
    - run: npm run test:e2e
    - uses: actions/upload-artifact@v2
      if: always()
      with:
        name: test-results
        path: test-results ← default folder with all artifacts
```

Uploading Artifacts


Demo: **Playwright Trace Viewer**

Opening **Playwright Trace Viewer**

Timeline



Actions List

Action Details

DOM Snapshot 🔥

Playwright Test: **Playful Testing Framework**

1. Get started with **npm init playwright**
2. Configure everything at **`playwright.config.ts`**
3. Test **iPhone**, customize reporters, generate tests
4. Debug tests with **Playwright Inspector**
5. Author tests with **Playwright CodeGen**
6. Post-mortem with **Playwright Tracing**

Playwright

- Cross-browser Web Testing and Automation Framework
- Documentation: https://playwright.dev
- Source / Issues: https://github.com/microsoft/playwright
- Social:
  - https://aka.ms/playwright/slack
  - https://aka.ms/playwright/twitter
  - https://aka.ms/playwright/youtube

Questions?

Andrey Lushnikov

@aslushnikov

aslushnikov@gmail.com

Playwright Test

@playwrightweb

https://aka.ms/playwright-slack

microsoft/playwright