

AUTOMATING ACCESSIBILITY TESTING WITH PLAYWRIGHT



[WCNL]

rowdy.codes rowdyrabouw rowdyrabouw

ROWDY

Front-End-Focused Senior Software Engineer with a focus on usability and accessibility

rowdy.codes rowdyrabouw rowdyrabouw

ROWDY

Google Developer Expert in Web Technologies

A11Y

Accessibility

Accessibility is a universal right because it is the gateway to other rights, such as voting, education, employment, and healthcare.

If you can’t enter a building or access a website, you are effectively denied your citizenship.

1.300.000.000

1 in 6 people worldwide - World Health Organization (WHO)


D
ATASAN
ATASAM
HASTANESI
H
WEG
HS Datatoe 13:90
K9 Datatoe 23:00
K3 Datatoe 21:00
K8 Datatoe 21:00


ACCESSRBLE

MAZE | POWER PELLETS | GHOSTS | BONUS FRUITS

Welcome to Pac-Man Maze Adventures

Whether you are a fearless maze runner or a careful ghost dodger, the arcade world is packed with glowing pellets, power-ups, and high-score dreams. Grab a power pellet, track the ghosts, and race through the maze for the next big snack-sized victory.

Get the Power Pellet!

Your edge against the ghost pack, tight corners, and last-second chases before the bonus round begins.

- Essential for turning Blinky, Pinky, Inky, and Clyde into easy pickings.
- Great for clearing a maze and stacking points before time runs out.

Bonus Fruits

When exploring the maze, keep an eye out for cherries, strawberries, and other bonus treats that can add a huge score boost if you stay sharp and keep moving.

As Pac-Man says in the arcade:

“ワカ・ワカ”

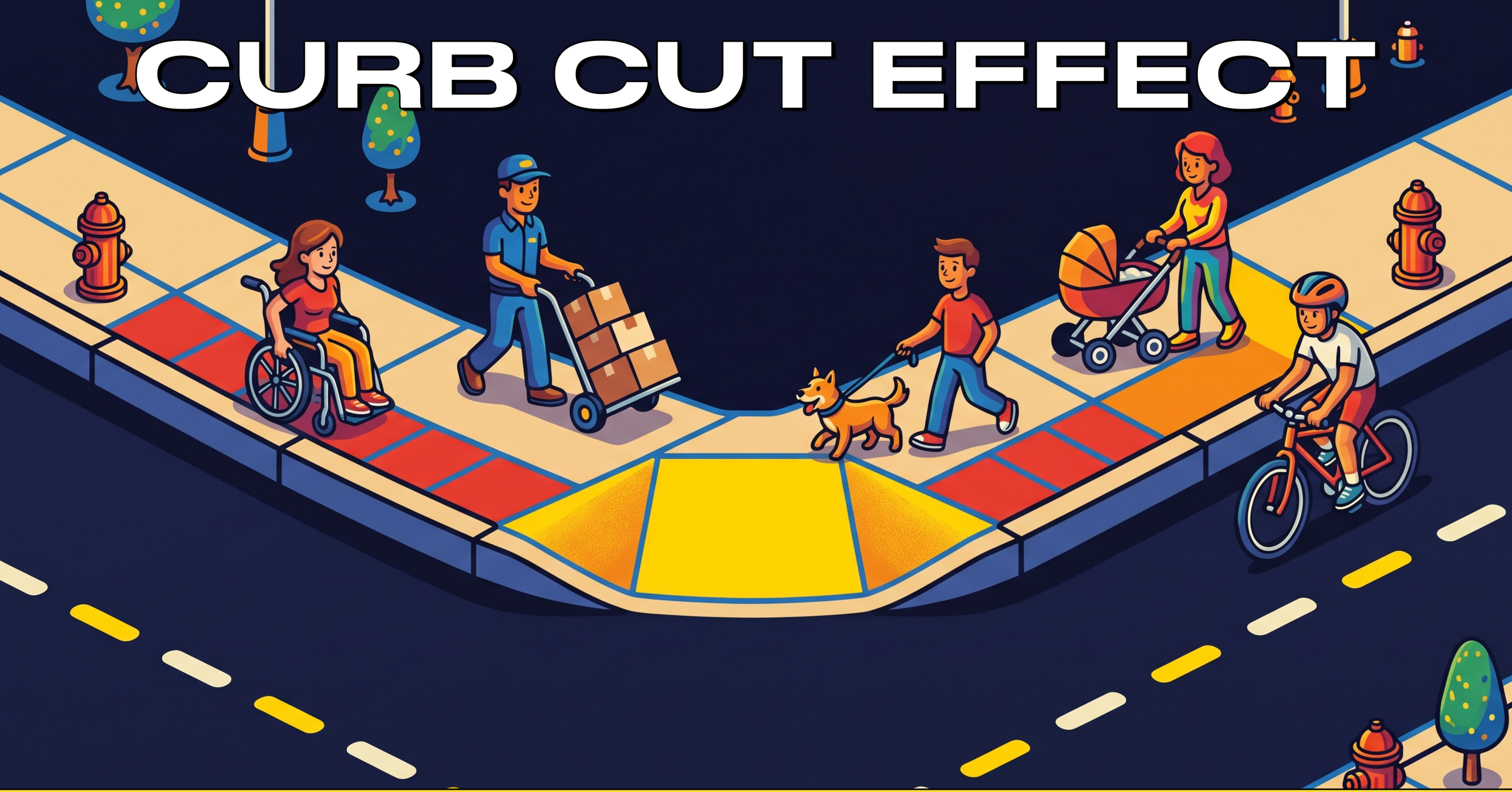


	TOUCH	SEE	HEAR	SPEAK	COGNITIVE
PERMANENT	 Missing limbs	 Blind	 Deaf	 Non-verbal	 Dyslexia
TEMPORARY	 Broken arm	 Eye infection	 Hearing aid	 Throat infection	 Concussion
SITUATIONAL	 Carrying a baby	 Sun on screen	 NC headphones	 Foreign languages	 Tired

CURB CUT EFFECT



CURB CUT EFFECT



“We’re all just temporarily abled.”

Cindy Li

web designer

(1975–2018)

EEA

European Accessibility Act

B2C

Business to Consumer

B2B

Business to Business



2B

Sh*t to Business

WCAG

Web Content Accessibility Guidelines

POUR

Perceivable, Operable, Understandable, and Robust

86

Success Criteria



WCAG Conformance Levels

31 Success Criteria

AA

WCAG Conformance Levels

24 Success Criteria

AAA

WCAG Conformance Levels

31 Success Criteria

How to Meet WCAG (Quickref Reference)

How to Meet WCAG (Quick Reference)

A customizable quick reference to Web Content Accessibility Guidelines (WCAG) 2 requirements (success criteria) and techniques.

W3C Web Accessibility Initiative WAI

Show About & How to Use

Contents | Filter | Hide

1. Perceivable
   - 1.1 Text Alternatives
     - 1.1.1 Non-text Content
   - 1.2 Time-based Media
     - 1.2.1 Audio-only and Video-only (Prerecorded)
     - 1.2.2 Captions (Prerecorded)
     - 1.2.3 Audio Description or Media Alternative (Prerecorded)
     - 1.2.4 Captions (Live)
     - 1.2.5 Audio Description (Prerecorded)
     - 1.2.6 Sign Language (Prerecorded)
     - 1.2.7 Extended Audio Description (Prerecorded)
     - 1.2.8 Media Alternative (Prerecorded)
     - 1.2.9 Audio-only (Live)
   - 1.3 Adaptable
     - 1.3.1 Info and Relationships
     - 1.3.2 Meaningful Sequence
     - 1.3.3 Sensory Characteristics
     - 1.3.4 Orientation
     - 1.3.5 Identify Input Purpose
     - 1.3.6 Identify Purpose
   - 1.4 Distinguishable
     - 1.4.1 Use of Color
     - 1.4.2 Audio Control

Selected Filters: **WCAG 2.2:** all success criteria and all techniques. | Clear filters | Expand **all** sections | Share

Principle 1 – Perceivable

Information and user interface components must be presentable to users in ways they can perceive.

Guideline 1.1 – Text Alternatives

Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.

1.1.1 Non-text Content — Level A

All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. Show full description

Understanding 1.1.1

Show techniques and failures for 1.1.1

SHARE | BACK TO TOP

Guideline 1.2 – Time-based Media

Provide alternatives for time-based media.

1.2.1 Audio-only and Video-only (Prerecorded) — Level A

For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled as such: Show full description

Understanding 1.2.1

Show techniques and failures for 1.2.1

SHARE | BACK TO TOP


WCAG in Plain English - AAArdvark
Provided by AArdvark
Home
Principles
1 Perceivable
2 Operable
3 Understandable
4 Robust
Guidelines
1.1 - Text Alternatives
1.2 - Time-based Media
1.3 - Adaptable
1.4 - Distinguishable
2.1 - Keyboard Accessible
2.2 - Enough Time
2.3 - Seizures and Physical Reactions
2.4 - Navigable
2.5 - Input Modalities
3.1 - Readable
3.2 - Predictable
3.3 - Input Assistance
WCAG in Plain English
Making the Web Content Accessibility Guidelines (WCAG) easy to understand, one success criterion at a time.
Friendly reminder
This is a beginner-friendly guide, not a replacement for the official WCAG. See our full disclaimer.
Search articles
Search
Search
Perceivable
Text Alternatives
WCAG 2.0, 2.1, 2.2
Level A
1.1.1 Non-text Content
All images and other non-text content (like icons, charts, audio, CAPTCHAs, or controls) must have a descriptive text alternative that conveys their meaning. Purely decorative content can be hidden from assistive technologies (e.g. using an empty alt attribute).
Code and Labels, Sensory, Wording
Auditory/Hearing, Cognitive, Visual
Code, Content
View Details
Perceivable
Time-based Media
WCAG 2.0, 2.1, 2.2
Level A
1.2.1 Audio-only and Video-only (Prerecorded)



> pnpm create playwright

```
TypeScript or JavaScript (default: TypeScript)

Tests folder name (default: tests, or e2e if tests already exists)

Add a GitHub Actions workflow (recommended for CI)

Install Playwright browsers (default: yes)
```

> pnpm create playwright

```
playwright.config.ts      → Test configuration

tests/example.spec.ts     → Minimal example test

package.json              → @playwright/test

pnpm-lock.yaml            → package-lock.json / yarn.lock
```

```ts
import { test, expect } from "@playwright/test";

test("has title", async ({ page }) => {
  // some test code
});

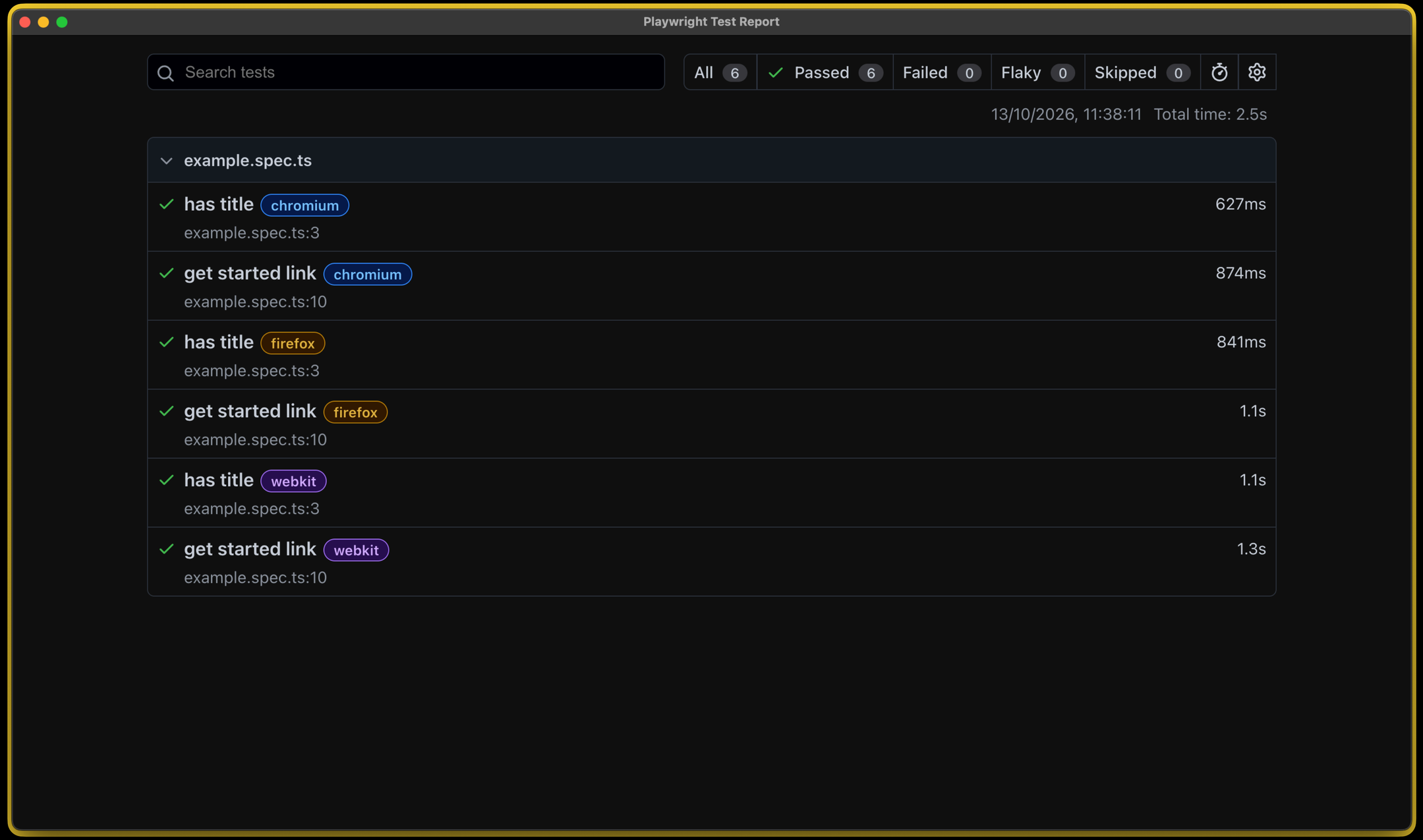
test("get started link", async ({ page }) => {
  // some test code
});
```

```
> npx playwright test

  Running 6 tests using 6 workers
    6 passed (3.8s)
```



```
> npx playwright show-report
```


Playwright Test Report
Search tests
All 6
Passed 6
Failed 0
Flaky 0
Skipped 0
13/10/2026, 11:38:11 Total time: 2.5s
example.spec.ts
has title chromium
example.spec.ts:3
627ms
get started link chromium
example.spec.ts:10
874ms
has title firefox
example.spec.ts:3
841ms
get started link firefox
example.spec.ts:10
1.1s
has title webkit
example.spec.ts:3
1.1s
get started link webkit
example.spec.ts:10
1.3s

AXE

Axe-core by Deque





Performance

Accessibility



Best Practices



SEO



Agentic Browsing

```
> pnpm add -D @axe-core/playwright
```

a11y.basic.spec.ts

```ts
import { test, expect } from "@playwright/test";
```

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";
```

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("page should be accessible", async ({ page }) => {



});
```

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("page should be accessible", async ({ page }) => {
  await page.goto("http://127.0.0.1:8080/");


});
```

MAZE | POWER PELLETS | GHOSTS | BONUS FRUITS

Welcome to Pac-Man Maze Adventures

Whether you are a fearless maze runner or a careful ghost dodger, the arcade world is packed with glowing pellets, power-ups, and high-score dreams. Grab a power pellet, track the ghosts, and race through the maze for the next big snack-sized victory.

Get the Power Pellet!

Your edge against the ghost pack, tight corners, and last-second chases before the bonus round begins.

- Essential for turning Blinky, Pinky, Inky, and Clyde into easy pickings.
- Great for clearing a maze and stacking points before time runs out.

Bonus Fruits

When exploring the maze, keep an eye out for cherries, strawberries, and other bonus treats that can add a huge score boost if you stay sharp and keep moving.

As Pac-Man says in the arcade:

“ワカ・ワカ”



a11y.basic.spec.ts

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("page should be accessible", async ({ page }) => {
  await page.goto("http://127.0.0.1:8080/");
  const axeResults = await new AxeBuilder({ page }).analyze();

});
```

a11y.basic.spec.ts

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("page should be accessible", async ({ page }) => {
  await page.goto("http://127.0.0.1:8080/");
  const axeResults = await new AxeBuilder({ page }).analyze();
  expect(axeResults.violations).toEqual([]);
});
```

```
> npx playwright test tests/a11y.basic.spec.ts --project=chromium
```



a11y.testinfo.spec.ts

```
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("Page should be accessible", async ({ page },          ) ⇒ {
  await page.goto("http://127.0.0.1:8080/");




});
```

```
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("Page should be accessible", async ({ page },          ) ⇒ {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () ⇒ {




  });
});
```

a11y.testinfo.spec.ts

```
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("Page should be accessible", async ({ page },          ) ⇒ {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () ⇒ {
    const axeResults = await new AxeBuilder({ page }).analyze();




  });
});
```

a11y.testinfo.spec.ts

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("Page should be accessible", async ({ page }, testInfo) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page }).analyze();
    await testInfo.attach("axe-results", {
      body: JSON.stringify(axeResults.violations, null, 2),
      contentType: "application/json",
    });

  });
});
```

a11y.testinfo.spec.ts

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";

test("Page should be accessible", async ({ page }, testInfo) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page }).analyze();
    await testInfo.attach("axe-results", {
      body: JSON.stringify(axeResults.violations, null, 2),
      contentType: "application/json",
    });
    expect(axeResults.violations.length).toBe(0);
  });
});
```

```
> npx playwright test tests/a11y.testinfo.spec.ts --project=chromium
```



```
> pnpm add -D axe-html-reporter
```

a11y.html.spec.ts

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";


test("Page should be accessible", async ({ page }, testInfo) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page }).analyze();
    await testInfo.attach("axe-results", {
      body: JSON.stringify(axeResults.violations, null, 2),
      contentType: "application/json",
    });



    expect(axeResults.violations.length).toBe(0);
  });
});
```

a11y.html.spec.ts

```ts
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";


test("Page should be accessible", async ({ page }, testInfo) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page }).analyze();
    await testInfo.attach("axe-results", {
      body: JSON.stringify(axeResults.violations, null, 2),
      contentType: "application/json",
    });



    expect(axeResults.violations.length).toBe(0);
  });
});
```

a11y.html.spec.ts

```
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";


test("Page should be accessible", async ({ page }, testInfo) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page }).analyze();









    expect(axeResults.violations.length).toBe(0);
  });
});
```

a11y.html.spec.ts

```
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";
import { createHtmlReport } from "axe-html-reporter";

test("Page should be accessible", async ({ page }, testInfo) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page }).analyze();









    expect(axeResults.violations.length).toBe(0);
  });
});
```

a11y.html.spec.ts

```
import { test, expect } from "@playwright/test";
import AxeBuilder from "@axe-core/playwright";
import { createHtmlReport } from "axe-html-reporter";

test("Page should be accessible", async ({ page }, testInfo) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page }).analyze();
    const reportHTML = createHtmlReport({
      results: axeResults,
      options: { projectKey: "Playwright A11Y Test" }
    });
    await testInfo.attach("axe-results-html", {
      body: reportHTML, contentType: "text/html"
    });
    expect(axeResults.violations.length).toBe(0);
  });
});
```

```
> npx playwright test tests/a11y.html.spec.ts --project=chromium
```



.withTags

Only run selected rules

Default

WCAG 2.0 Level A + AA (2008)

WCAG 2.1 Level A + AA (2018)

WCAG 2.2 Level A + AA (2023)

Best practices

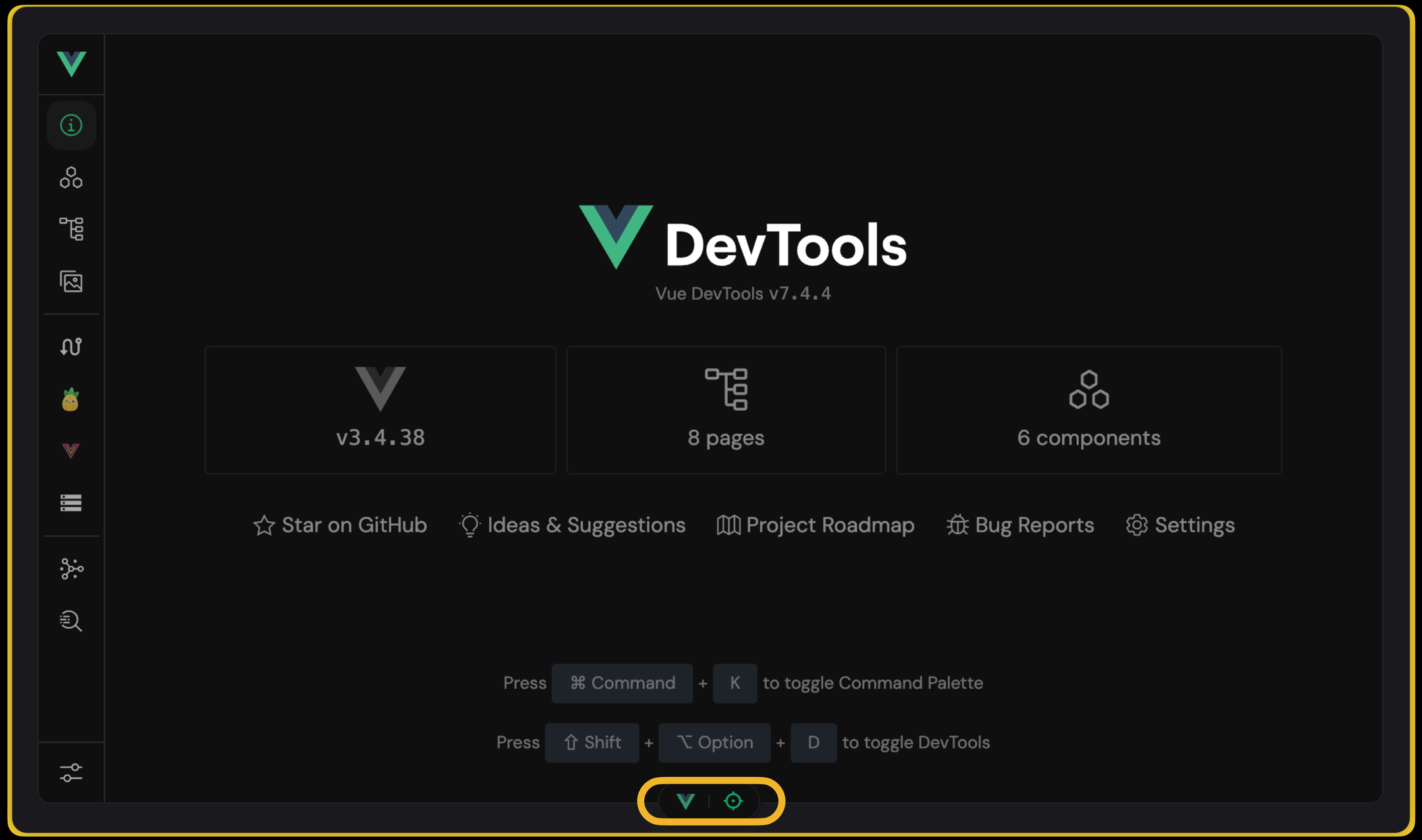
Deque's recommended rules that aren't required by WCAG but improve the overall user experience

```
wcag2a          (WCAG 2.0 Level A)
wcag2aa         (WCAG 2.0 Level AA)
wcag2aaa        (WCAG 2.0 Level AAA)
wcag21a         (WCAG 2.1 Level A)
wcag21aa        (WCAG 2.1 Level AA)
wcag21aaa       (WCAG 2.1 Level AAA)
wcag22a         (WCAG 2.2 Level A)
wcag22aa        (WCAG 2.2 Level AA)
wcag22aaa       (WCAG 2.2 Level AAA)
best-practice
experimental
```

```
test("Page should be accessible", async ({ page }) ⇒ {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () ⇒ {
    const axeResults = await new AxeBuilder({ page })
      .withTags(["wcag2a", "wcag2aa", "wcag21a", "wcag21aa"])
      .analyze();
    expect(axeResults.violations.length).toBe(0);
  });
});
```

.exclude

Prevent all rules from running against the specified elements and all of their descendants


DevTools
Vue DevTools v7.4.4
v3.4.38
8 pages
6 components
Star on GitHub
Ideas & Suggestions
Project Roadmap
Bug Reports
Settings
Press ⌘ Command + K to toggle Command Palette
Press ⇧ Shift + ⌥ Option + D to toggle DevTools

```
test("Page should be accessible", async ({ page }) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page })
      .exclude(".vue-devtools__panel")
      .analyze();
    expect(axeResults.violations.length).toBe(0);
  });
});
```

```
test("Page should be accessible", async ({ page }) => {
  await page.goto("http://127.0.0.1:8080/");
  await test.step("Run accessibility scan", async () => {
    const axeResults = await new AxeBuilder({ page })
      .exclude("#element-with-known-a11y-issue")
      .analyze();
    expect(axeResults.violations.length).toBe(0);
  });
});
```

Axe-core® Accessibility Results for Playwright A11Y Test project

Page URL: http://127.0.0.1:8080/

axe-core found 7 violations

#	Description	Axe rule ID	WCAG	Impact	Count
1	Heading levels should only increase by one	heading-order	Best practice	moderate	1
2	<html> element must have a lang attribute	html-has-lang	WCAG 2 Level A, WCAG 3.1.1	serious	1
3	Images must have alternative text	image-alt	WCAG 2 Level A, WCAG 1.1.1	critical	1
4	Form elements must have labels	label	WCAG 2 Level A, WCAG 4.1.2	critical	1
5	and must only directly contain , <script> or <template> elements	list	WCAG 2 Level A, WCAG 1.3.1	serious	1
6	Zooming and scaling must not be disabled	meta-viewport	WCAG 2 Level AA, WCAG 1.4.4	moderate	1
7	lang attribute must have a valid value	valid-lang	WCAG 2 Level AA, WCAG 3.1.2	serious	1

Failed

1. Heading levels should only increase by one

Learn more

heading-order

Best practice
moderate

Ensure the order of headings is semantically correct

Issue Tags: **cat.semantics** **best-practice**

#	Issue Description	To solve this violation, you need to...
1	**Element location** `h3:nth-child(3)`	Fix any of the following: • Heading order invalid

Heading levels should only increase by one

```
<h1>Welcome to Pac-Man Maze Adventures</h1>

<h3>Get the Power Pellet!</h3>

<h3>Level Select</h3>
```

```
<h1>Welcome to Pac-Man Maze Adventures</h1>

<h2>Get the Power Pellet!</h2>

<h2>Level Select</h2>
```

<html> element must have a lang attribute

```
<!doctype html>

<html>

  <head>
```

```
<!doctype html>

<html lang="en">

  <head>
```

Images must have alternative text

```
<img src="Gemini_Generated_Image_soeanusoeanusoea.jpg" />
```

```
<img src=“Gemini_Generated_Image_soeanusoeanusoea.jpg"
     alt="Retro art of Pac-Man chasing a scared blue ghost,
     with four ghosts trailing behind."/>
```

Form elements must have labels



```
Enter your Player Name:
<input name="player" type="text" />
```



```
<label for="player">Enter your Player Name:</label>
<input type="text" id="player" />
```

 and <ol> must only directly contain <li> elements

```
ul div::before { content: "▪ "; }

<ul>
  <div>Essential for turning ... pickings.</div>
  <div>Great for clearing a ... out.</div>
</ul>
```

```
<ul>
  <li>Essential for turning ... pickings.</li>
  <li>Great for clearing a ... out.</li>
</ul>
```

Lang attribute must have a valid value

```
<p class="quote">As Pac-Man says in the arcade:</p>
<span lang="japanese"><blockquote>ワカ・ワカ</blockquote></span>
```

```
<p class="quote">As Pac-Man says in the arcade:</p>
<span lang="ja"><blockquote>ワカ・ワカ</blockquote></span>

blockquote::before { content: open-quote; }
blockquote::after { content: close-quote; }
```

As Pac-Man says in the arcade:

“ワカ・ワカ”



quotation marks

As Pac-Man says in the arcade:

「ワカ・ワカ」



CJK brackets

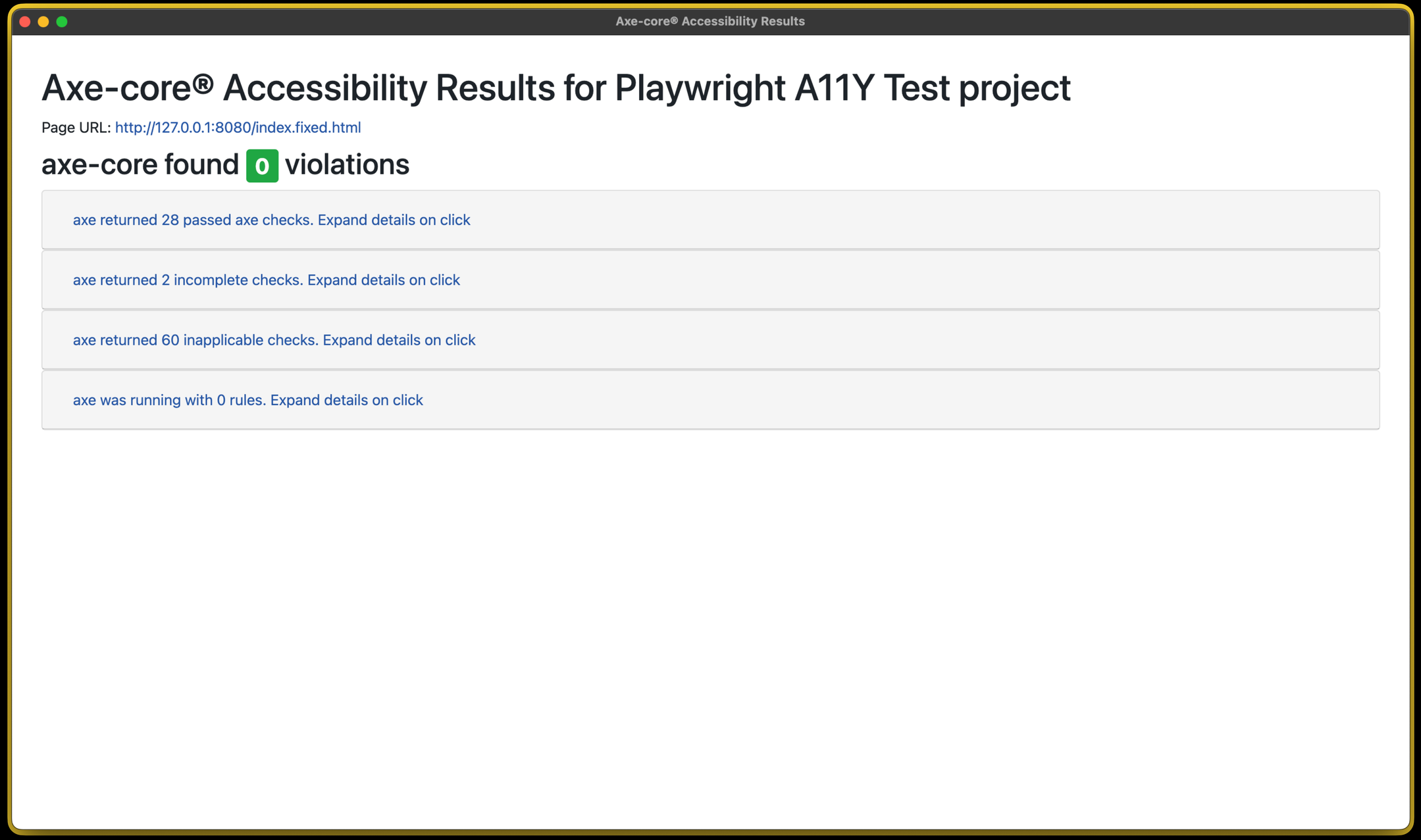
Zooming and scaling must not be disabled



```
<meta name="viewport" content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
```



```
<meta name="viewport" content="width=device-width,
    initial-scale=1.0" />
```


Axe-core® Accessibility Results
Axe-core® Accessibility Results for Playwright A11Y Test project
Page URL: http://127.0.0.1:8080/index.fixed.html
axe-core found 0 violations
axe returned 28 passed axe checks. Expand details on click
axe returned 2 incomplete checks. Expand details on click
axe returned 60 inapplicable checks. Expand details on click
axe was running with 0 rules. Expand details on click

To learn more about the next level, click here.

“Underline your f*cking links you sociopaths”

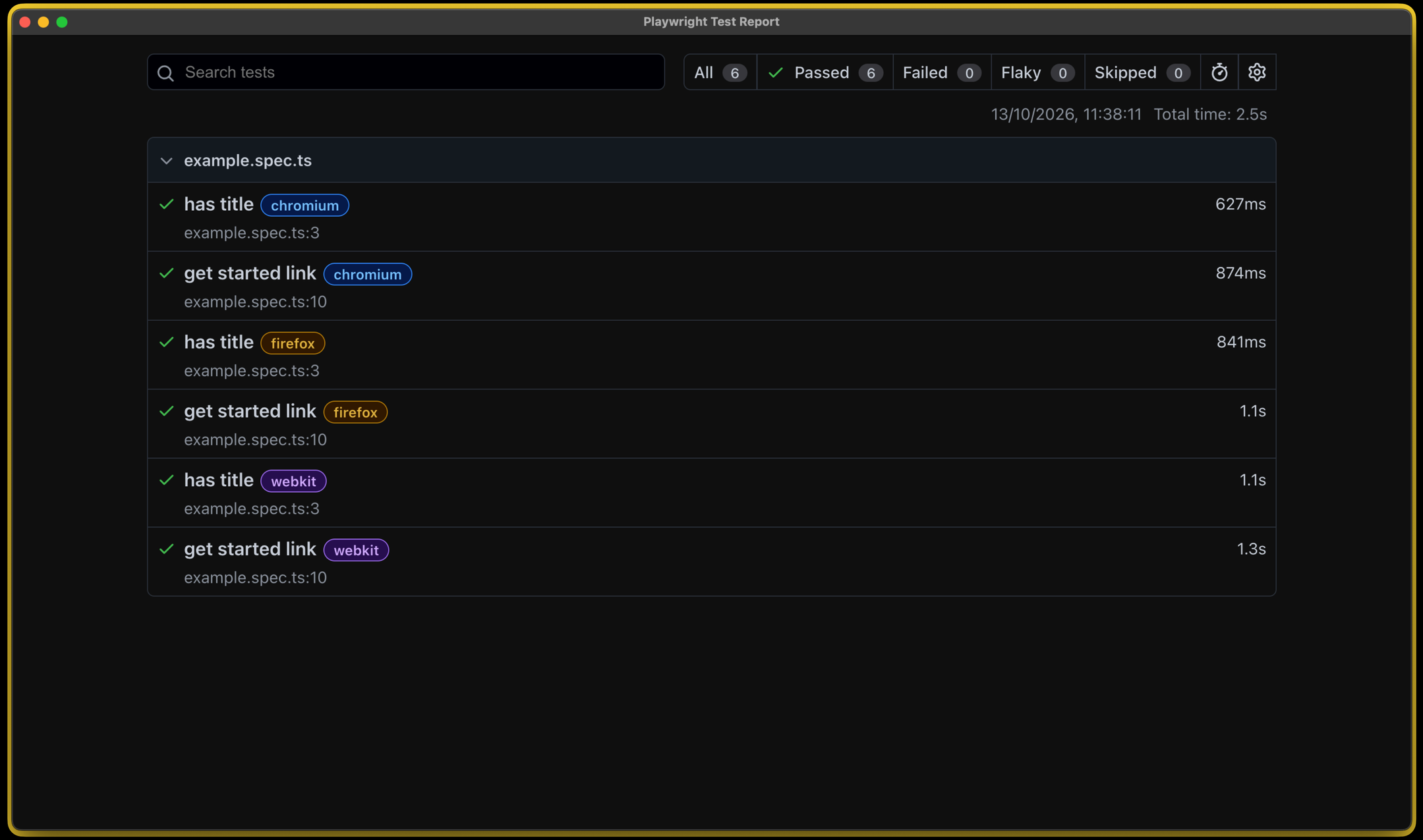
Heydon Pickering

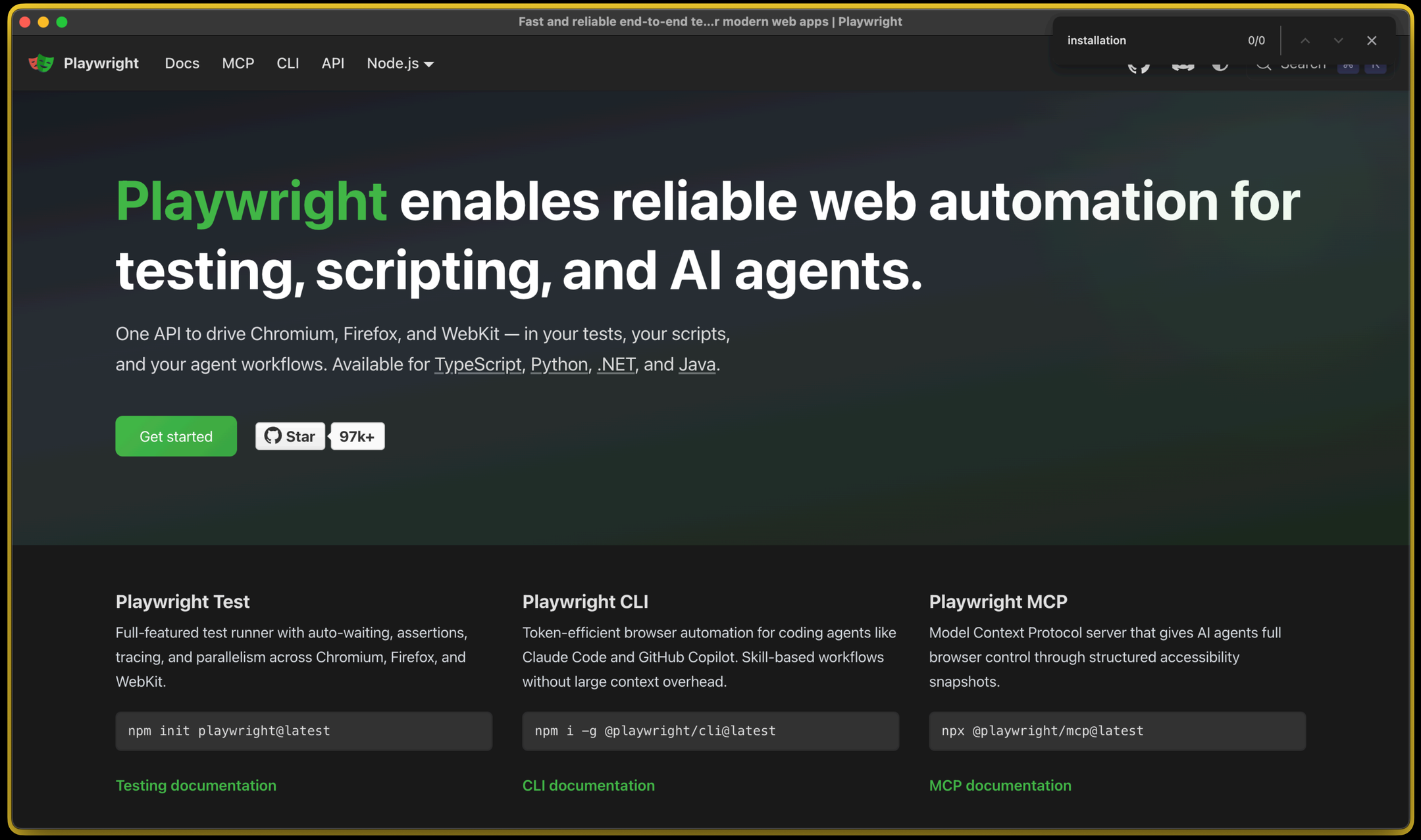
technical writer and designer

Learn more about the next level.

MORE

We can do more ... and should do more


Playwright Test Report
Search tests
All 6
Passed 6
Failed 0
Flaky 0
Skipped 0
13/10/2026, 11:38:11 Total time: 2.5s
example.spec.ts
has title chromium
example.spec.ts:3
627ms
get started link chromium
example.spec.ts:10
874ms
has title firefox
example.spec.ts:3
841ms
get started link firefox
example.spec.ts:10
1.1s
has title webkit
example.spec.ts:3
1.1s
get started link webkit
example.spec.ts:10
1.3s


Fast and reliable end-to-end te...r modern web apps | Playwright
installation
0/0
Playwright
Docs
MCP
CLI
API
Node.js
Search
Playwright enables reliable web automation for testing, scripting, and AI agents.
One API to drive Chromium, Firefox, and WebKit — in your tests, your scripts, and your agent workflows. Available for TypeScript, Python, .NET, and Java.
Get started
Star
97k+
Playwright Test
Full-featured test runner with auto-waiting, assertions, tracing, and parallelism across Chromium, Firefox, and WebKit.
npm init playwright@latest
Testing documentation
Playwright CLI
Token-efficient browser automation for coding agents like Claude Code and GitHub Copilot. Skill-based workflows without large context overhead.
npm i -g @playwright/cli@latest
CLI documentation
Playwright MCP
Model Context Protocol server that gives AI agents full browser control through structured accessibility snapshots.
npx @playwright/mcp@latest
MCP documentation

getByRole

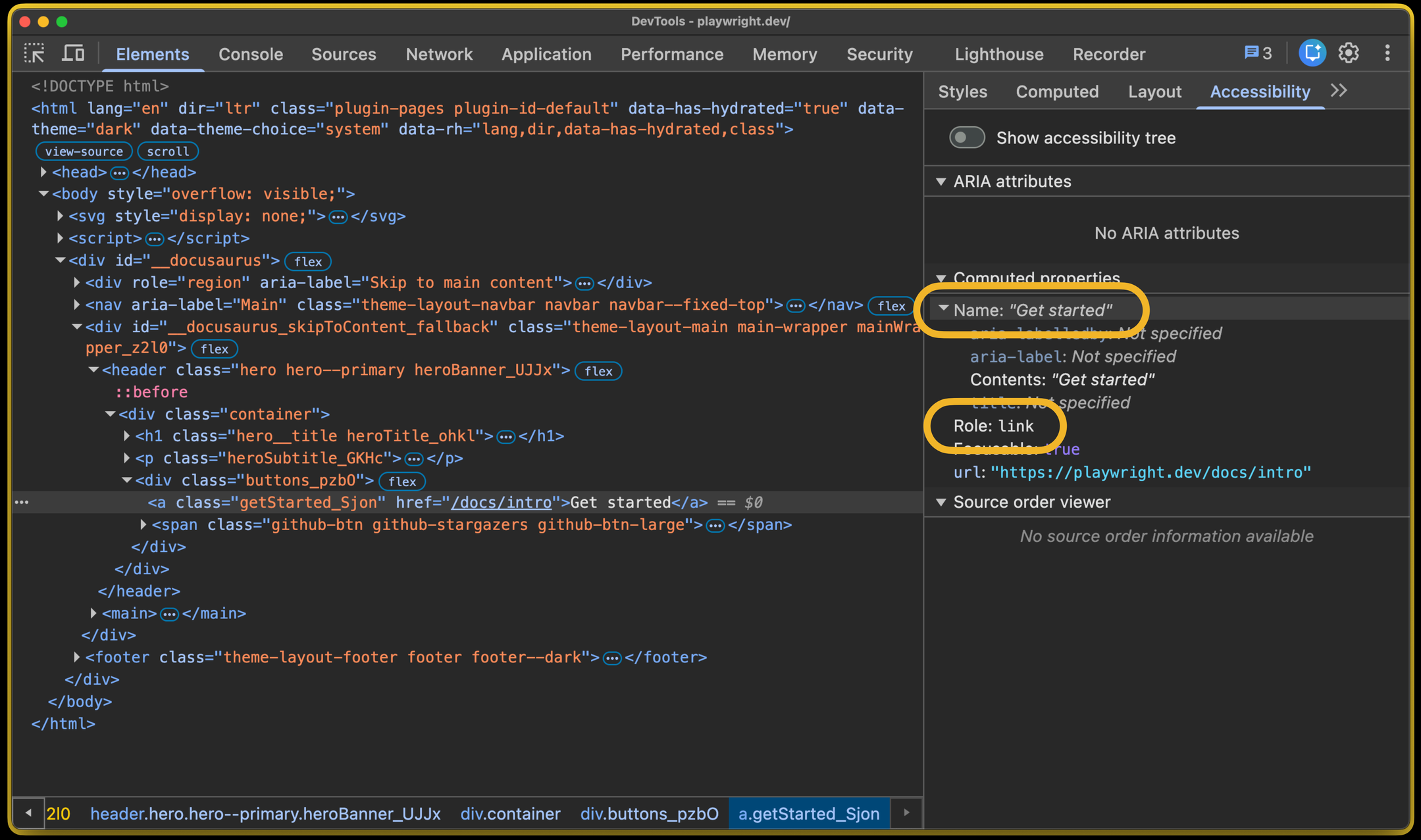
How assistive technology perceives the page

example.spec.ts

```
import { test, expect } from "@playwright/test";

test("get started link", async ({ page }) ⇒ {
  await page.goto("https://playwright.dev/");
  await page.getByRole("link", { name: "Get started" }).click();
  await expect(page.getByRole("heading", { name: "Installation" }))
                                                .toBeVisible();
});
```

alert alertdialog application article banner
blockquote button caption cell checkbox
code columnheader combobox complementary
contentinfo definition deletion dialog directory
document emphasis feed figure form
generic grid gridcell group heading img
insertion link list listbox listitem log main
marquee math meter menu menubar
menuitem menuitemcheckbox menuitemradio
navigation none note option paragraph
presentation progressbar radio radiogroup
region row rowgroup rowheader scrollbar
search searchbox separator slider spinbutton
status strong subscript superscript switch
tab table tablist tabpanel term textbox time
timer toolbar tooltip tree treegrid treeitem


DevTools - playwright.dev/
Elements
Console
Sources
Network
Application
Performance
Memory
Security
Lighthouse
Recorder
3
<!DOCTYPE html>
<html lang="en" dir="ltr" class="plugin-pages plugin-id-default" data-has-hydrated="true" data-theme="dark" data-theme-choice="system" data-rh="lang,dir,data-has-hydrated,class">
view-source
scroll
<head> </head>
<body style="overflow: visible;">
<svg style="display: none;"> </svg>
<script> </script>

flex

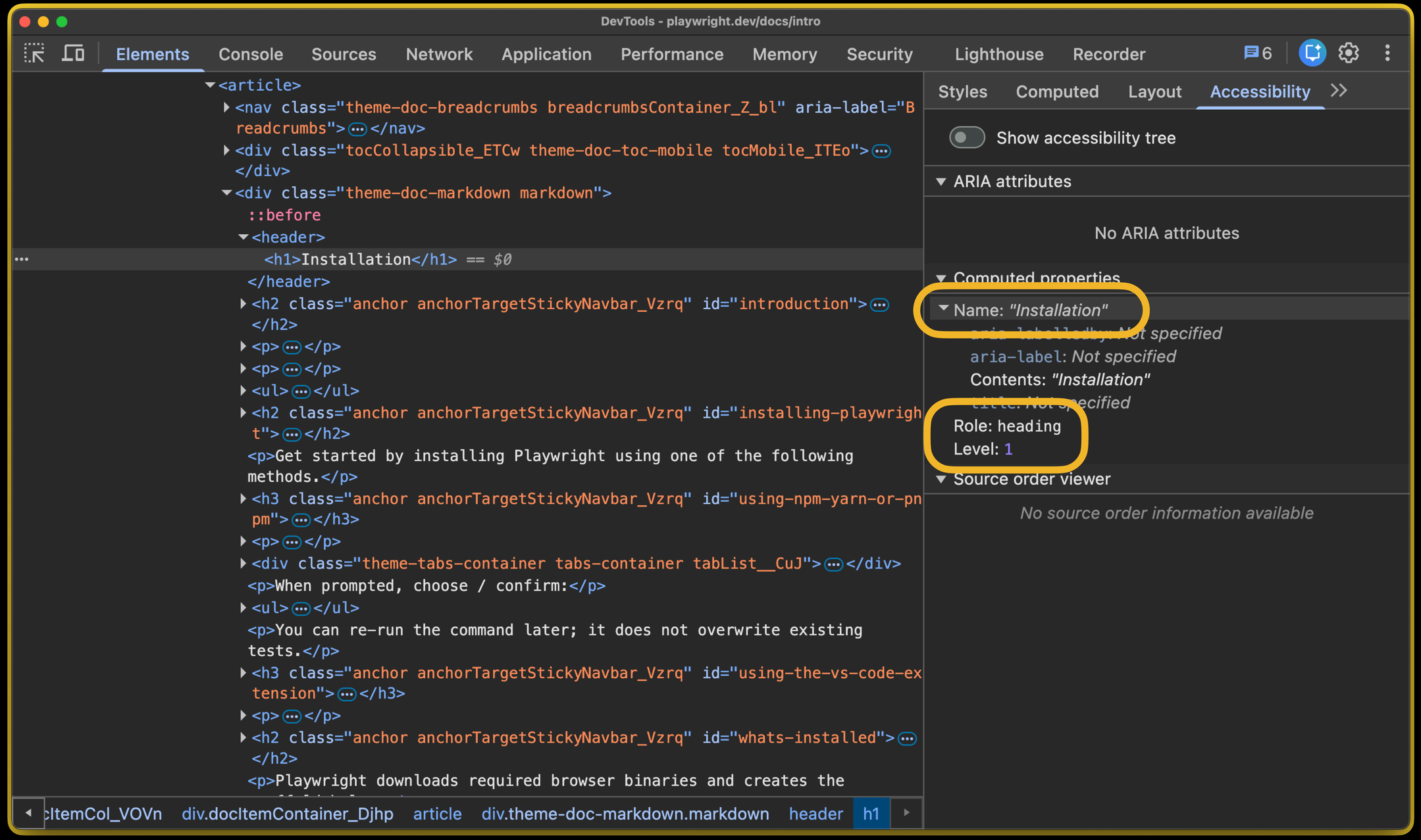
<nav aria-label="Main" class="theme-layout-navbar navbar navbar--fixed-top"> </nav>

<header class="hero hero--primary heroBanner_UJJx">
::before

<h1 class="hero__title heroTitle_ohkl"> </h1>

Get started == $0

<main> </main>
<footer class="theme-layout-footer footer footer--dark"> </footer>
2l0
header.hero.hero--primary.heroBanner_UJJx
div.container
div.buttons_pzbO
a.getStarted_Sjon
Styles
Computed
Layout
Accessibility
Show accessibility tree
ARIA attributes
No ARIA attributes
Computed properties
Name: "Get started"
aria-label: Not specified
Contents: "Get started"
Role: link
url: "https://playwright.dev/docs/intro"
Source order viewer
No source order information available


DevTools - playwright.dev/docs/intro
Elements
Console
Sources
Network
Application
Performance
Memory
Security
Lighthouse
Recorder
6
<article>
<nav class="theme-doc-breadcrumbs breadcrumbsContainer_Z_bl" aria-label="Breadcrumbs">…</nav>
…

::before
<header>
<h1>Installation</h1> == $0
</header>
<h2 class="anchor anchorTargetStickyNavbar_Vzrq" id="introduction">…</h2>
…
…
…
<h2 class="anchor anchorTargetStickyNavbar_Vzrq" id="installing-playwright">…</h2>
Get started by installing Playwright using one of the following methods.
<h3 class="anchor anchorTargetStickyNavbar_Vzrq" id="using-npm-yarn-or-pnpm">…</h3>
…
…
When prompted, choose / confirm:
…
You can re-run the command later; it does not overwrite existing tests.
<h3 class="anchor anchorTargetStickyNavbar_Vzrq" id="using-the-vs-code-extension">…</h3>
…
<h2 class="anchor anchorTargetStickyNavbar_Vzrq" id="whats-installed">…</h2>
Playwright downloads required browser binaries and creates the
docItemCol_VOVn
div.docItemContainer_Djhp
article
div.theme-doc-markdown.markdown
header
h1
Styles
Computed
Layout
Accessibility
Show accessibility tree
ARIA attributes
No ARIA attributes
Computed properties
Name: "Installation"
aria-label: Not specified
Contents: "Installation"
Role: heading
Level: 1
Source order viewer
No source order information available

```
await page.getByRole("link", { name: "Get started", exact: true }).click();

await expect(page.getByRole("heading", { name: "Installation",
                                         exact: true,
                                         level: 1 })).toBeVisible();
```

getByLabel

```
<label for="player">Enter your Player Name:</label>
<input type="text" id="player" />

await page.getByLabel("player").fill("Puck Man");
```


パックマン
PUCK MAN
namco
HIGH SCORE
2UP
CONTROL
1人100円 / 2人200円

getByAltText

```
<img src="namco.jpg" alt="Namco logo"/>

const logo = page.getByAltText("Namco logo");
await expect(logo).toBeVisible()
```

getByText

```
<p class="quote">As Pac-Man says in the arcade:</p>
<span lang=“ja"><blockquote>ワカ・ワカ</blockquote></span>

await expect(page.getByText("ワカ・ワカ")).toBeVisible();
```

getByTitle

```
<iframe title="Map of the Pac-Man maze" src="..."></iframe>

await expect(page.getByTitle("Map of the Pac-Man maze")).isVisible();
```

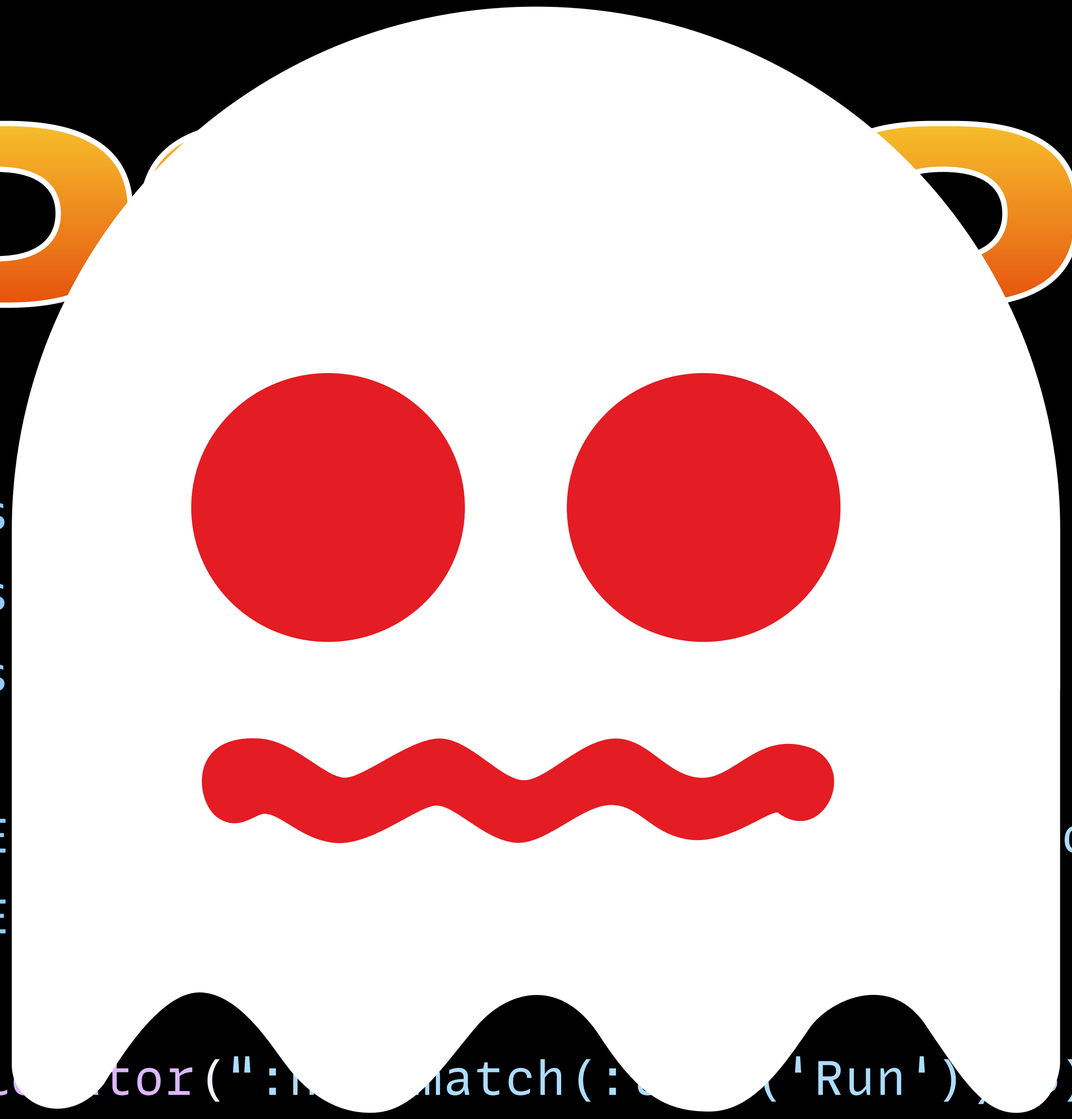
getByTestId



```
<select data-testid="character-select">
  ...
  <option value="blinky">Blinky</option>
  ...
</select>

const dropdown = page.getByTestId("character-select");
await dropdown.selectOption("blinky");
```

lo[illegible]or

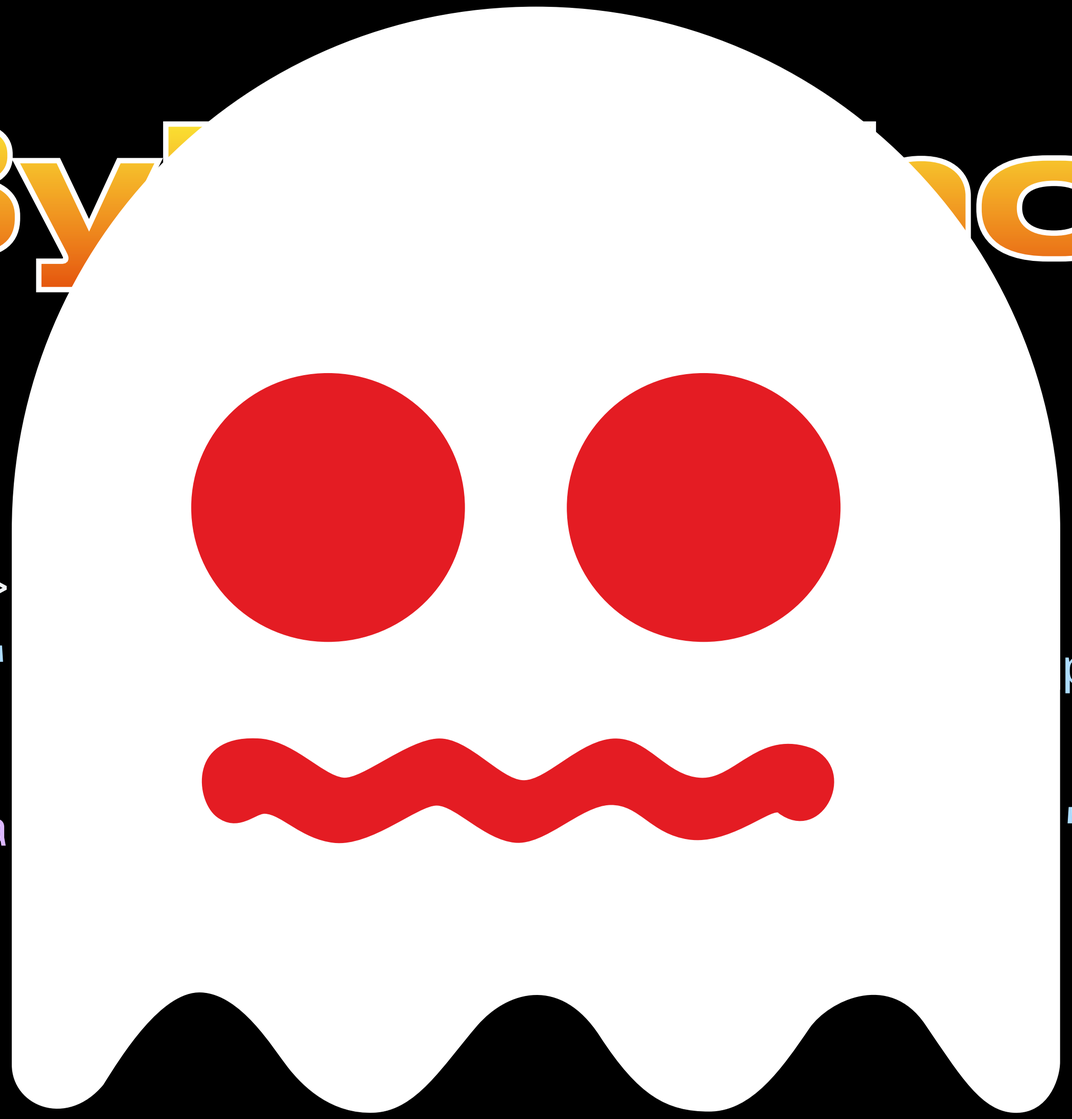


```
<button clas
<button clas
<button clas

const clydeE                                    on-btn");
await clydeE

await page.l    tor(":    match(:    ('Run')    )").click();
```

getBy[illegible]older

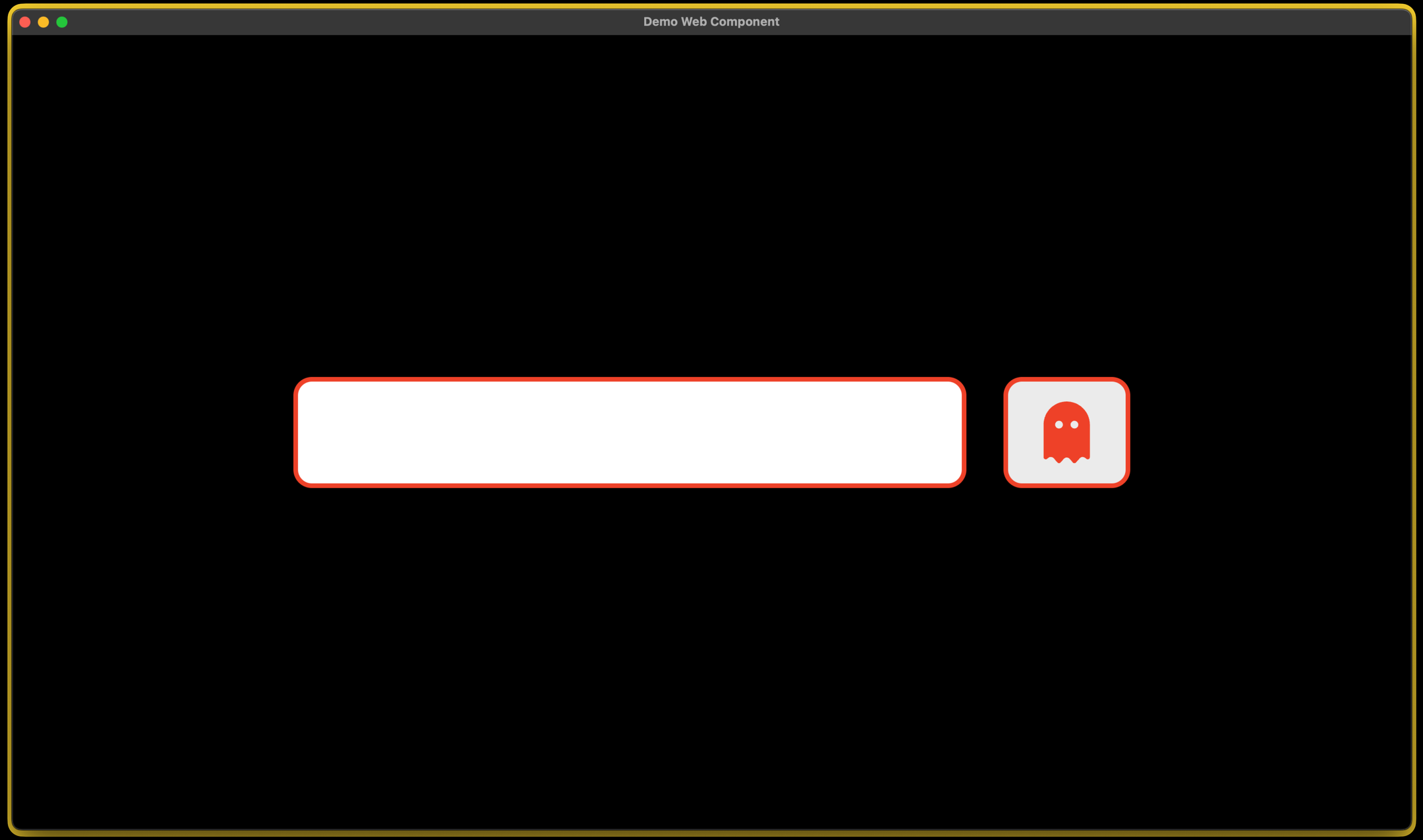


```
<label for="email">
<input type="email"                                    ple.com" />

await page.getByPla                                    "pinky@namco.com");
```

VITEST

Component Testing


Demo Web Component

demo-component.js

```
export class DemoComponent extends HTMLElement {
  constructor() {
    super();
  }
  connectedCallback() {
    this.innerHTML = `
      <form>
        <input />
        <button><svg viewBox="0 0 384 512”> ... </svg></button>
      </form>
    `;
  }
}

customElements.define("demo-component", DemoComponent);
```

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Demo Web Component</title>
    <link rel="stylesheet" href="style.css" />
    <script type="module" src="demo-component.js"></script>
  </head>
  <body>
    <demo-component />
  </body>
</html>
```

demo-component.test.js

```
import "../demo-component.js";
import { describe, it, expect, beforeEach, afterEach } from "vitest";
import { configureAxe } from "vitest-axe";

describe("DemoComponent", () => {
  let element;

  beforeEach(() => {
    element = document.createElement("demo-component");
    document.body.appendChild(element);
  });

  afterEach(() => {
    document.body.removeChild(element);
  });
```

```
  it("should have no accessibility violations", async () => {
    const form = element.querySelector("form");
    const axe = configureAxe({
      rules: {
        // disable landmark rules when testing isolated components
        region: { enabled: false },
      },
    });

    const results = await axe(form);
    expect(results).toHaveNoViolations();
  });
});
```

```
> vitest
```

```
Expected the HTML found at $('input') to have no violations:
<input type="text">

Received:
"Form elements must have labels (label)"

Fix any of the following:
  Element does not have an implicit (wrapped) <label>
  Element does not have an explicit <label>
  aria-label attribute does not exist or is empty
  aria-labelledby attribute does not exist, references elements that do not exist
                                                or references elements that are empty
  Element has no title attribute
  Element has no placeholder attribute
  Element's default semantics were not overridden with role="none" or role="presentation"

You can find more information on this issue here:
https://dequeuniversity.com/rules/axe/4.12/label?application=axeAPI
```

```
Expected the HTML found at $('button') to have no violations:
<button type="submit">

Received:
"Buttons must have discernible text (button-name)"

Fix any of the following:
  Element does not have inner text that is visible to screen readers
  aria-label attribute does not exist or is empty
  aria-labelledby attribute does not exist, references elements that do not exist
                                                or references elements that are empty
  Element has no title attribute
  Element does not have an implicit (wrapped) <label>
  Element does not have an explicit <label>
  Element's default semantics were not overridden with role="none" or role="presentation"

You can find more information on this issue here:
https://dequeuniversity.com/rules/axe/4.12/button-name?application=axeAPI
```



“We’re all just temporarily abled.”

Cindy Li

web designer

(1975–2018)

“Underline your f*cking links you sociopaths”

Heydon Pickering

technical writer and designer

DANK JE WEL

rowdy.codes/wcnl

