

# Jak wdrożyć od zera testowanie a11y frontendów

Łukasz Nowak

Senior software developer



WARSZAWSKIE  
DNI INFORMATYKI

Prelekcja wybrana w wyniku selekcji przez Radę Programową złożoną z uznanych liderów obszaru IT oraz Data Science.

Warszawa,  
04.04.2025 - 05.04.2025



OFICJALNA PRELEKCJA WARSZAWSKICH DNI INFORMATYKI





# Łukasz Nowak

Senior frontend dev

---



Łukasz Nowak



LukaszNowakPL

---

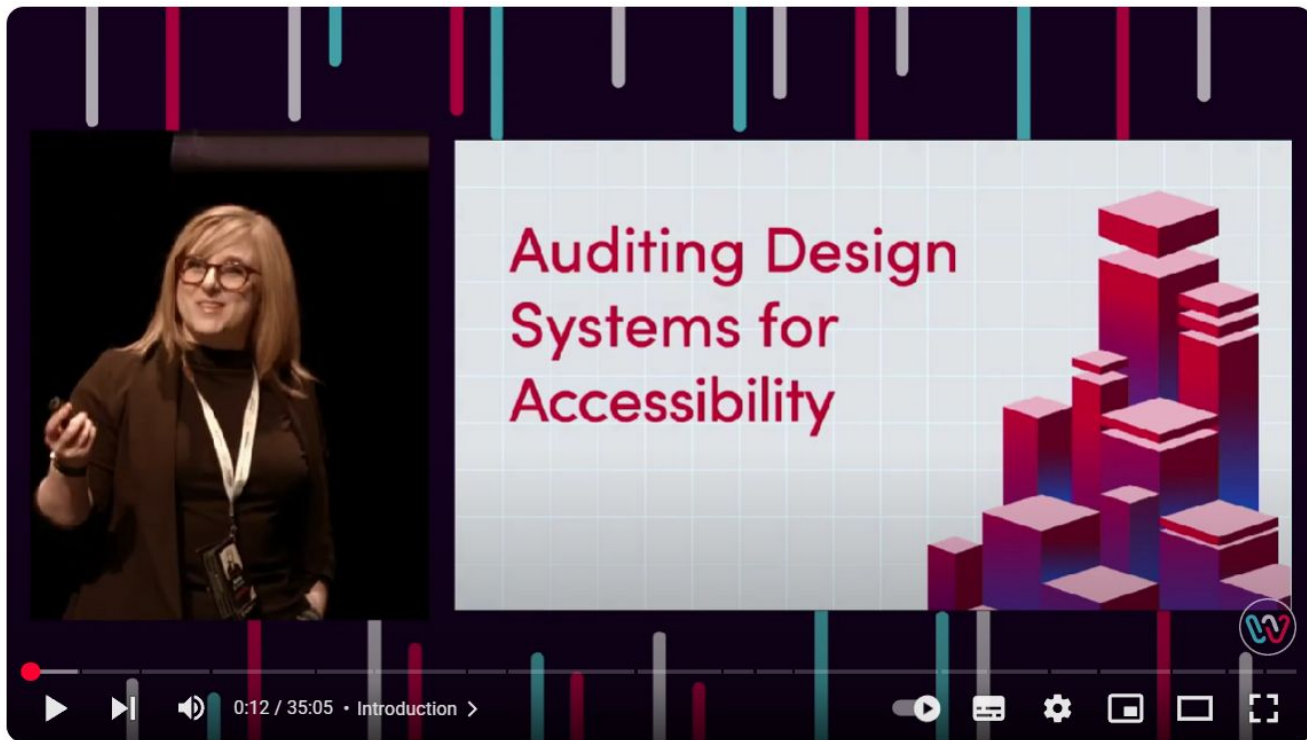
# Aktualny stan a11y w web developmencie

- Temat często poruszany na konferencjach / meetupach
- 28.06.2025 - w życie wchodzi regulacja European Accessibility Act wymagająca spełnienie zaleceń WCAG 2.1 na poziomie AA <sup>(1)</sup>
- Większość serwisów nie pokrywa w całości wymogów dostępności <sup>(2)(3)</sup>
- Bo każdy z nas ma dobre wymówki





Mój zespół był w większości.  
Do pewnego momentu.




<https://www.youtube.com/watch?v=L0PRzlyP1Zc>

**Auditing Design Systems for Accessibility by Anna E. Cook**




“Accessibility is not about being perfect. It’s about trying.”

Anna E. Cook, Wey Wey Web 2023



“It’s best to try a11y testing on new applications. You start small and grow as the project grows.”

Źródło nieznane



Czym tak naprawdę  
jest to accessibility?



# A11y jest niezbędne dla klientów doświadczających<sup>(4)</sup>

- Problemów ze wzrokiem lub słuchem
- Zaburzeń poznawczych i rozumienia treści
- Problemów ruchowych
- Zaburzeń mowy
- Ograniczeń neurologicznych



# Najczęstsze wyzwania a11y w webdev

- Mały kontrast kolorów - do pokrycia przez design
- Skomplikowana treść lub struktura usługi - do pokrycia przez UX
- Niedostateczna identyfikacja elementów na stronie - nasz obszar
- Problematyczna interakcja z elementami - to też



# Niedostateczna identyfikacja elementów na stronie

- Wiele sposobów na zaprezentowanie treści strony
  - Wizualna prezentacja (ekran)
  - Odczyt audio (czytnik ekranu)
  - Poprzez zaawansowaną technologię wspierającą (np. czytnik braille-a)
- Brak/słaby opis elementów typu obrazki, linki, przyciski i in.
- Wszystko rozbija się o jakość produkowanego HTML



# Problematic interaction with elements

- Using page elements to:
  - Selecting an element
    - By searching on the page (e.g. `getByRole('button', {name: 'Send'})`)
    - By a list of elements on the page (navigation using the Tab key)
  - Performing an action dedicated to an element (typing data, clicking, etc.)
- Everything boils down to the quality of the produced HTML



# Punkt startowy do testowania

- Dostępność zapewnia semantyczny i dobrze opisany HTML <sup>(4)</sup>
- Nie trzeba testować na każdym typie assistive technology
- Testowanie na poziomie produkcyjnego bundle
- Playwright (choć inne narzędzia też powinny dać radę)



# Przykładowe repozytorium



## Add airport

### Name

### IATA Code

### Country

### Regions

- ☐ Aegean Islands
- ☐ Corfu Island (Kerkyra)
- ☐ Ha Long Island
- ☐ Rhodes Island

- ☒ Balearic Islands
- ☐ Crete Island
- ☐ Ionian Islands
- ☐ Tenerife Island

- ☐ Canary Islands
- ☐ Dodecanese Islands
- ☒ Maiorca Island

### Vaccination Notes

# Przykładowe repozytorium

- W pełni funkcjonalna React SPA + backend
- Pełne scenariusze testowe
- Mikroframework + reużywalności
- Ważne: Aplikacja posiada błędy w dostępności!  
(Ale o tym przeczytacie w raportach z testów)



<https://github.com/LukaszNowakPL/web-a11y-testing>





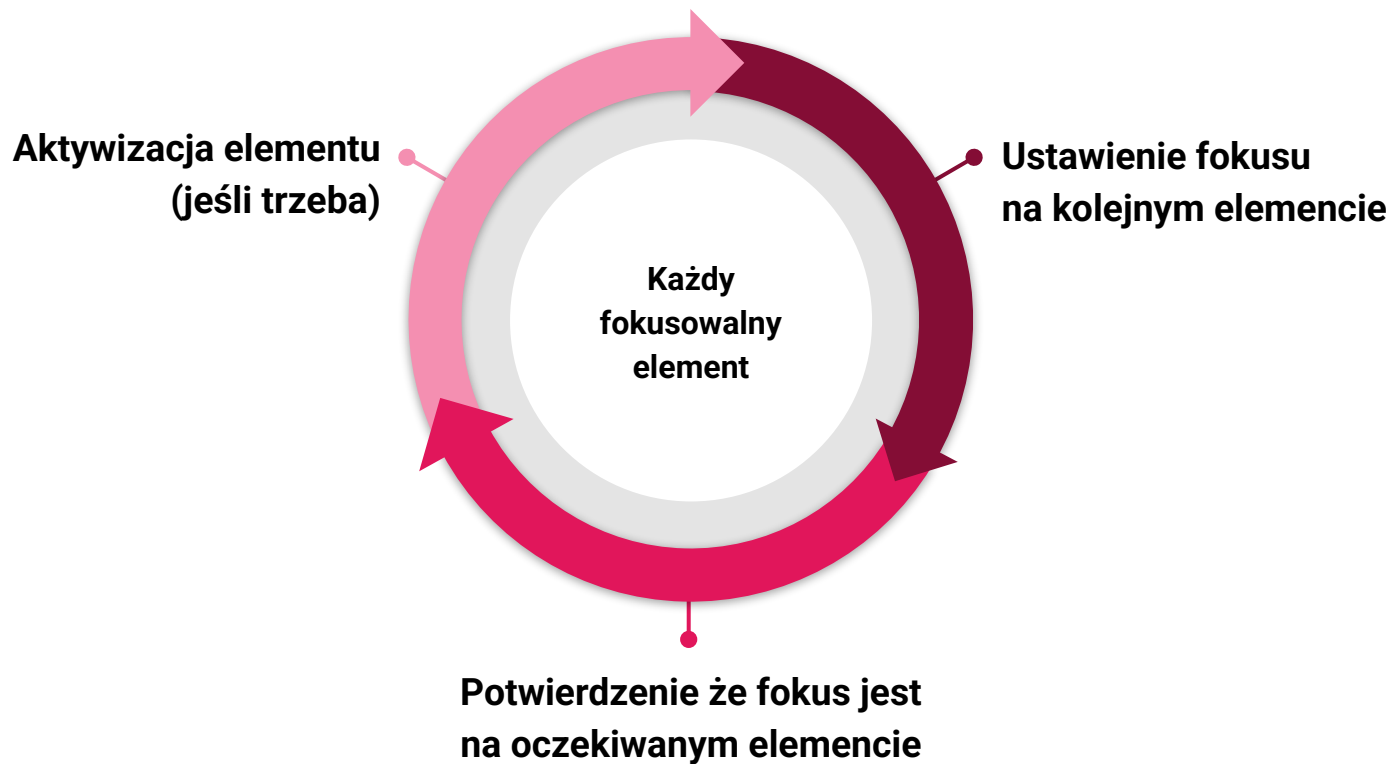
W jaki sposób automatycznie  
testować a11y?

# User journey z klawiaturową nawigacją

- To nic więcej jak kolejny scenariusz funkcjonalny
- Plan minimum: pokryć najprostszą happy path



# Założenie scenariusza



# Wycinek z testu

```
// Entering the page
public async proceedThroughPage(airport: AirportModel) {
    await this.assertNoFocus();
    await this.navigateToNextElement(); // ...

    // Filling airport name field
    await this.assertFocusedElement(this.page.getByRole('textbox', {name: /name/i}));
    await this.fill(airport.name);
    await this.navigateToNextElement(); // ...

    // Clicking submit button
    await this.assertFocusedElement(this.page.getByRole('button', {name: /send/i}));
    await this.performAction();
}
// Asserting data sent to backend service
```

# Reużywalne funkcje

```
private getFocusedElement () {  
    return this.page.locator('*:focus') as NotClickableLocator;  
}  
  
public async assertFocusedElement (expectedElement: Locator) {  
    expect(await this.getOuterHtml(this.getFocusedElement())).toStrictEqual(await this.getOuterHtml(expectedElement));  
}  
  
private async getOuterHtml (element: Locator) {  
    return await element.evaluate ((node) => node.outerHTML);  
}  
  
public async performAction () {  
    await this.page.press('body', this.KEYS.action);  
}  
  
public async selectFocusedElement () {  
    await this.page.press('body', this.KEYS.selection);  
}  
  
public async navigateToNextElement () {  
    await this.page.press('body', this.KEYS.navigateToNextElement);  
}
```

# Audyt finalnego HTML

# Wiele twarzy jednego widoku

## Provide piece of data

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Piece of data

Submit

## Provide piece of data

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Piece of data

This data is obligatory

Submit

## Provide piece of data

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Piece of data

Data required

Submit

# Plan scenariusza





# Wycinek z testu

```
// Then I wait until the page is ready  
await addAirportPage.assertReady();  
  
// When I perform snapshot analysis for initial view  
await addAirportPage.performSnapshotAnalysis('Initial view', {takeLighthouseSnapshot});  
  
// And I fulfill the form with correct data  
await addAirportPage.fulfillForm(airport);  
  
// Then I can see the form data is valid and Submit button is enabled  
await addAirportPage.assertSubmitEnabled();  
  
// When I perform snapshot analysis for provided data  
await addAirportPage.performSnapshotAnalysis('Form filled', {takeLighthouseSnapshot});  
// ...
```

# Proces audytowania

```
public async assertSnapshots(snapshotTitle: string, config?: AssertSnapshotConfig) {  
    await Promise.all([  
        this.assertAxeChecks(snapshotTitle, Boolean(config?.skipAxeViolationChecks)),  
        this.performAriaSnapshotComparison(snapshotTitle),  
        config?.takeLighthouseSnapshot !== undefined  
            && this.assertLighthouseSnapshot (snapshotTitle,  
config.takeLighthouseSnapshot ),  
    ]);  
}
```

# Audyt Lighthouse

- playwright-lighthouse - zintegrowana automatyzacja
- Fajnie craftowy audyt accessibility:
  - Best practices, WCAG, semantyka HTML
  - Analiza wizualna (kontrasty kolorów, odległości itp.)
- Prosty w zrozumieniu i pomocny raport końcowy



<https://www.npmjs.com/package/playwright-lighthouse>



# Zapięcie Lighthouse Flow

```
// And Lighthouse flow
const flow = await LighthousePlaywrightFlow
    .startFlow('Airport addition journey', page);

// And take snapshot helper to be passed to each page
const takeLighthouseSnapshot: TakeLighthouseSnapshot = async (name) =>
    await flow.snapshot(name,
        {onlyCategories: Object.keys(lighthouseThresholds), formFactor: 'mobile'});

// ...
// When I perform snapshot analysis for initial view
await addAirportPage.performSnapshotAnalysis('Initial view',
    {takeLighthouseSnapshot, skipAxeViolationChecks: true});

// ...
// Then I generate snapshot report
await flow.generateReports(testInfo, 'lighthouse-test-add-airport-happy-path-flow.html',
    lighthouseThresholds);
```

# Tworzenie raportu z Lighthouse flow

```
const res = await this.flow.createFlowResult();

// Attach Lighthouse report to Playwright report
await testInfo.attach(filename, {body: reports.toString()});

const lighthouseThresholds = {'best-practices': 100, accessibility: 90, seo: 50};

const categories = Object.keys(lighthouseThresholds)
  as ('accessibility' | 'best-practices' | 'seo')[];
let thresholdFailures: string[] = [];

res.steps.forEach((step) => { /* ... */ });

expect(thresholdFailures).toStrictEqual([]);
```

# Axe accessibility testing engine

- Statyczna analiza HTML u rynkowego lidera
- Raporty z testów - słodko-gorzki format
- Możliwość zawężania skanowania do wybranych wytycznych (np. WCAG 2.1 AA)



<https://playwright.dev/docs/accessibility-testing>



# Wywołanie statycznej analizy HTML w Axe

```
private async assertAxeChecks (snapshotTitle: string, skipViolationChecks: boolean) {
  const axeChecks = await new AxeBuilder({page: this.page})
    .withTags(['wcag2a', 'wcag2aa', 'wcag21a', 'wcag21aa', 'wcag22aa'])
    .analyze();

  await this.testInfo.attach(this.composeAlllyReportName (snapshotTitle), {
    body: JSON.stringify(axeChecks, null, 2),
    contentType: 'application/json',
  });

  if (!skipViolationChecks) {
    expect(axeChecks.violations).toEqual([]);
  }
}
```

# Aria snapshot

- Idea drzewa dostępności <sup>(5)</sup>
- Coś jak DOM dedykowany dla a11y
- Funkcjonalność Aria snapshots w Playwright
- Wzorzec regresji accessibility



<https://playwright.dev/docs/aria-snapshots#aria-snapshots>





# Wywołanie Aria snapshot

```
private async performAriaSnapshotComparison (snapshotTitle: string) {
    expect(await this.page.locator('body').ariaSnapshot()).toMatchSnapshot([
        ARIA_SNAPSHOTS_DIRECTORY,
        `${this.composeSnapshotName(snapshotTitle)}.yaml`,
    ]);
}

private composeSnapshotName (snapshotTitle: string) {
    return slugify(`${this.pageTitle} ${this.testSuite} ${snapshotTitle}`);
}
```

# Przykład Aria snapshotu

- `banner`:
  - `link "Airports dashboard"`:
    - `img`
    - `text`: Airports dashboard
  - `navigation`:
    - `list`:
      - `listitem`:
        - `link "Airports"`
      - `listitem`:
        - `link "Add airport"`
  - `main`:
    - `alert`: Sorry, there is some connectivity error
    - `paragraph`:
      - `text`: We were unable to fetch some of data necessary to display the page. You can
      - `button "restart data fetching"`
      - `text`: manually.
    - `paragraph`: If it won't help, try to restart the app.

# To samo, tylko wizualnie



**Airports dashboard**

Airports [Add airport](#)

## **Sorry, there is some connectivity error**

We were unable to fetch some of data necessary to display the page. You can [restart data fetching](#) manually.  
If it won't help, try to restart the app.

# Nieznana super-moc aria snapshot

```
- listitem:  
  - link "Airports"  
- listitem:  
  - link "Add airport"  
- main:  
  - heading "Add airport" [level=1]  
  - text: Name  
  - textbox "Name"  
  - text: IATA Code  
  - textbox "IATA Code"  
  - text: Country  
  - combobox "Country": Select one  
  - text: Regions
```

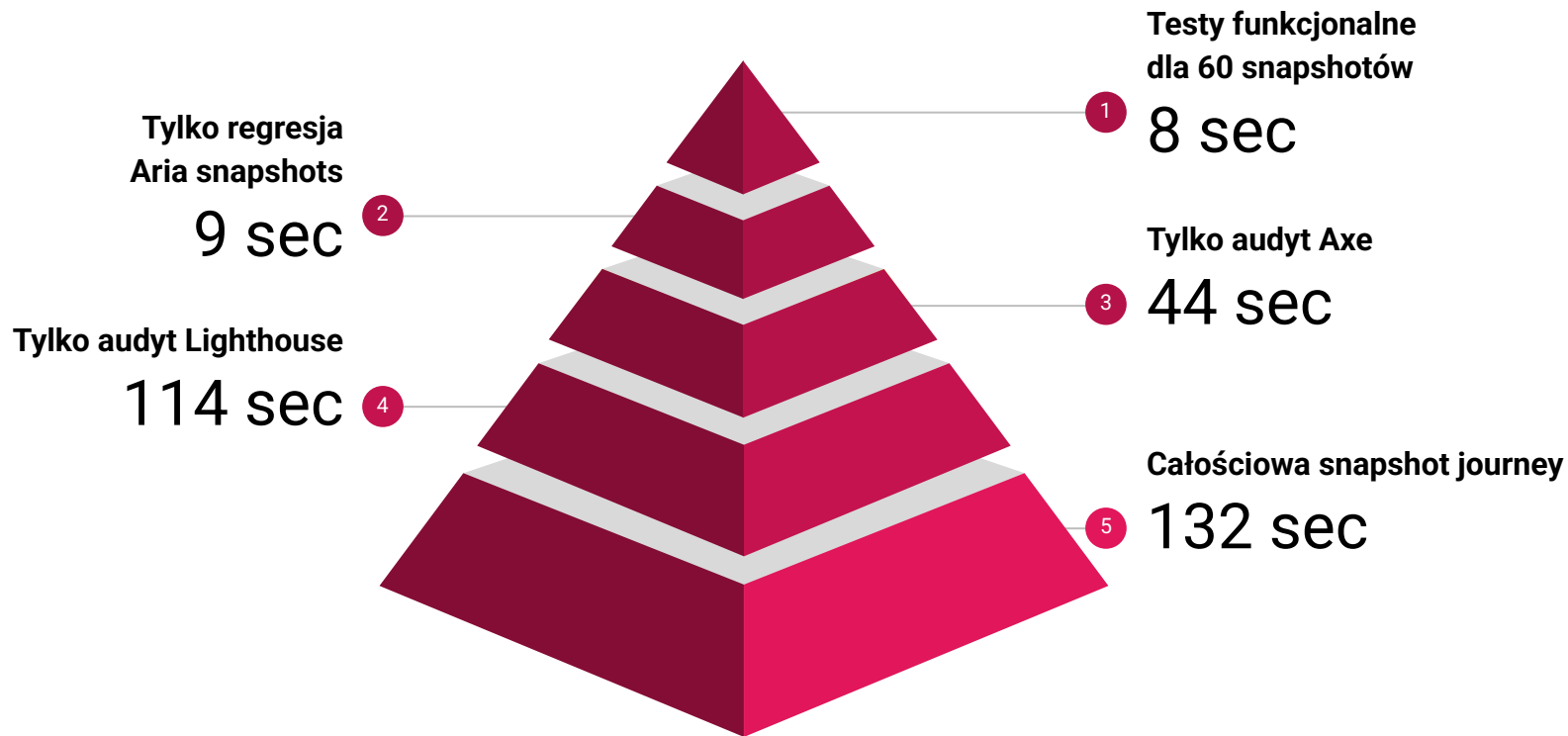
# Uwagi i spostrzeżenia

# Uwagi i spostrzeżenia

- Testy snapshotów są długie



# Czasochłonność testów snapshotów

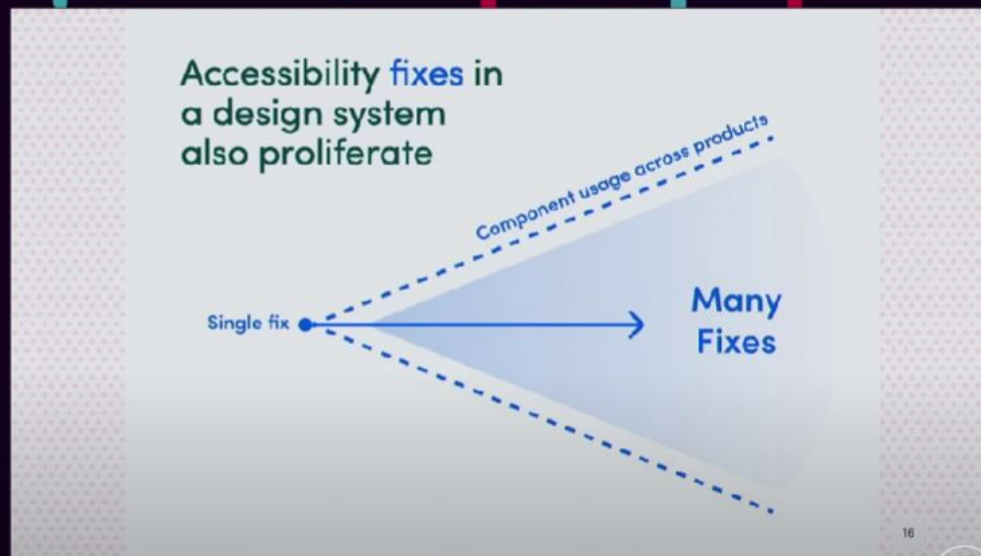


# Uwagi i spostrzeżenia

- Testy snapshotów są długie
- Pokrywajcie a11y na niskim poziomie, np. w design systemie







8:39 / 35:05 • Components >



Auditing Design Systems for Accessibility by Anna E. Cook

# Uwagi i spostrzeżenia

- Testy snapshotów są długie
- Pokrywajcie a11y na niskim poziomie, np. w design systemie
- Ale testujcie też na poziomie aplikacji / produktu
- Testowanie automatyczne to temat relatywnie młody, brakuje mu całościowych rozwiązań, dobrych praktyk i narzędzi
- Uważajcie na opcje płatne



# Podsumowanie

# Podsumowanie

- A11y is not about being perfect. It's about trying
- Pisanie testów pomogło w tworzeniu kultury a11y-aware
- Testy automatyczne nie znajdują wszystkich błędów
- Robią to prawdziwi użytkownicy z assistive technologies



# Disclaimer of Playwright

## DISCLAIMER

Automated accessibility tests can detect some common accessibility problems such as missing or invalid properties. But many accessibility problems can only be discovered through manual testing. We recommend using a combination of automated testing, manual accessibility assessments, and inclusive user testing.

For manual assessments, we recommend [Accessibility Insights for Web](#), a free and open source dev tool that walks you through assessing a website for [WCAG 2.1 AA](#) coverage.

Source: <https://playwright.dev/docs/accessibility-testing>

# Feedback

Zeskanuj kod i zostaw  
swoją opinię



Jak wdrożyć od zera testowanie a11y frontendów - historia prawdziwa.

Łukasz Nowak

<https://warszawskiedniinformatyki.pl/user.html#!/lecture/WDI25-63dd/rate>

# Referencje

1. The EAA: Technical Aspects of Compliance  
<https://www.wcaq.com/compliance/european-accessibility-act/>
2. The state of Web Accessibility in 2024 (Research Report)  
<https://www.accessibilitychecker.org/research-papers/the-state-of-web-accessibility-in-2024-research-report/>
3. The WebAIM Million (2025 accessibility report)  
<https://webaim.org/projects/million/>
4. Common Website Accessibility Issues  
<https://reciteme.com/news/common-website-accessibility-issues/>
5. Accessibility tree  
[https://developer.mozilla.org/en-US/docs/Glossary/Accessibility\\_tree](https://developer.mozilla.org/en-US/docs/Glossary/Accessibility_tree)

