Accessibility Evaluation of Open Web Repositories

Juliette Waltregny

October 2025

1 Introduction

This project aimed to develop and test a computational metric for assessing web accessibility across open web repositories. The study implemented a scoring model that quantifies accessibility-related HTML and ARIA features and aggregates them into a global measure termed the **Accessibility Evaluation Index (AEI)**. The analysis was conducted across ten repositories to compare accessibility practices in modern web development.

Our stakeholder emphasized the importance of measurable and transparent accessibility metrics that developers could interpret and improve iteratively. This encouraged the inclusion of proportional scoring and penalty reduction to make the AEI both educational and diagnostic.

2 Methodology

Each repository was cloned locally and analyzed using a custom TypeScript script. The tool parsed HTML (or embedded JSX/TSX) files to extract accessibility-related tags and attributes. A minimum threshold was applied to ensure meaningful computation: files with fewer than 50 total tags were excluded from AEI calculation to avoid bias from partial templates or test files.

The metric computation considered five primary accessibility indicators and one penalty term. Each indicator was normalized per 100 tags to provide proportional weighting.

2.1 Metrics Definition

- **Alternative Text Count (altCount)**: Number of `img` elements with non-empty `alt` attributes. This reflects support for screen readers and non-visual users.
- **ARIA Count (ariaCount)**: Number of elements with valid `aria-*` attributes. ARIA properties enhance accessibility semantics, especially for interactive or dynamic components.
- **Semantic Tag Count (semanticCount)**: Occurrences of semantic HTML5 tags (`<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, etc.). These tags contribute to logical document structure and assistive navigation.
- **Label Count (labelCount)**: Number of correctly associated `<label>` elements. This metric captures form accessibility, ensuring that input fields are identifiable.

- **Keyboard Support Count (keyboardCount)**: Instances of explicit keyboard navigation or focus management (e.g., `tabindex`, key event handlers). This supports users who cannot use a mouse.
- **Penalties**: Deductions for missing attributes, misuse of tags, or redundant ARIA roles. Each penalty reduces the AEI proportionally.

2.2 Accessibility Evaluation Index (AEI) Computation

The **AEI** was computed for each file as:

$$AEI = clamp_{0-100}\left(\frac{(alt + aria + semantic + label + keyboard)}{totalTags/100} - 10 \times penalties\right)$$

The values were clamped between 0 and 100, ensuring comparability across projects regardless of size. The repository-level AEI was the mean of individual file scores (excluding files below the tag threshold).

2.3 Indicators

The following indicators were defined:

- Between 85 and 100 : Green
- Between 65 and 85 : Orange
- Below 65 : Red

3 Repositories Analyzed

Ten repositories were selected to ensure diversity in structure and content richness:

1. **a11yproject.com**
2. **bootstrap**
3. **docusaurus**
4. **gatsby**
5. **hugoDocs**
6. **next.js**
7. **react.dev**
8. **readthedocs.org**
9. **static-accessibility** (subset of Hugo templates)
10. **website** (standalone Hugo website)

Some of these are sub-repositories of larger frameworks (e.g., *Next.js* and *React*), but they were chosen because they include user-facing templates or documentation pages. This ensured that the computed AEI reflects accessibility as experienced by end users rather than internal system code.

4 Results

The computed AEI averages are summarized below:

Repository	Average AEI	Grade
a11yproject.com	72	Yellow
bootstrap	34	Red
docusaurus	45	Red
gatsby	17	Red
hugoDocs	9	Red
next.js	14	Red
react.dev	31	Red
readthedocs.org	38	Red
static-accessibility	9	Red
website	15	Red

Table 1: Accessibility Evaluation Index results per repository

5 Discussion

The results presented in Table 1 indicate a broad disparity in accessibility practices across repositories. While the *a11yproject.com* repository achieved a relatively high Accessibility Evaluation Index (AEI) of 72, most other projects scored considerably lower, with averages ranging between 9 and 45. This suggests that even well-established frameworks and documentation platforms tend to lack consistent accessibility considerations across their codebases.

One important factor influencing these results is the nature and size of the repositories. Several of the analyzed projects—such as *Next.js*, *React.dev*, and *Bootstrap*—are large, multi-module frameworks containing both core logic and demonstration files. Many of these files are not directly user-facing, meaning they were not designed with accessibility as a primary concern. Consequently, the inclusion of such internal or technical files in the analysis may lower the overall AEI, even though the user-facing components of these frameworks may perform better in practice.

Conversely, repositories such as *a11yproject.com* explicitly focus on accessibility awareness and education, which explains their superior performance. Their content is primarily web-facing and structured around accessible design examples, resulting in higher semantic tag usage, more consistent use of `alt` text, and a well-defined document hierarchy.

Future iterations of this analysis could refine the AEI by distinguishing between user-facing templates and backend or testing files to provide a more representative evaluation of practical accessibility quality.

6 Conclusion

The Accessibility Evaluation Index (AEI) provides a scalable and interpretable measure of accessibility quality within web repositories. While none of the tested repositories achieved a perfect score, most showed moderate accessibility awareness. This suggests

progress toward inclusion, but also indicates the need for stronger community emphasis on ARIA, labeling, and keyboard support.

Future improvements could include weighting metrics differently per content type and expanding the analysis to dynamic accessibility events. Moreover, integrating qualitative feedback from developers and users could further refine the AEI toward real-world accessibility outcomes.