

Web Accessibility

February 2025

About Me

- Jon Kuperman
- TC39
- Source Maps
- Signals
- Builds DevTools @ Bloomberg



Introduction to Accessibility

This Course

- What is Accessibility?
- What are the legal requirements for websites
- Assistive technologies
- How to install and use screen readers
- Accessibility standards
- What is semantic HTML?
- Focus management
- Accessibility tricks
- Color and contrast
- Tools and testing

a **accessibility** **y**

1 2 3 4 5 6 7 8 9 10 11



a **11** **y**

What is Accessibility?

When websites and web tools are properly designed and coded, people with disabilities can use them. However, currently many sites and tools are developed with accessibility barriers that make them difficult or impossible for some people to use. Making the web accessible benefits individuals, businesses, and society.

- W3C Web Accessibility Initiative

What is Accessibility?

Web accessibility means that people with disabilities can **use** the Web.

More specifically, Web accessibility means that people with disabilities can **perceive, understand, navigate,** and **interact** with the Web, and that they can **contribute** to the Web.

Similar Fields

- Web Performance
- Internationalization
- UI Design

Disability Statistics

- 26 percent of adults in the United States have some type of disability.
- 2 in 5 adults age 65 years and older have a disability
- 20 percent of people in the US (48 million people) report some degree of hearing loss, and 29 million of them could benefit from using hearing aids.
- 2.3 percent of people in the US (7 million people) report having a visual disability, and 1 million people in the US are legally blind.

Types of Disabilities

- Mobility and physical
- Cognitive and neurological
- Visual
- Hearing

The web is already accessible

(even if your website is not)

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 news](#) , [Frequently Asked Questions](#) .

[What's out there?](#)

Pointers to the world's online information, [subjects](#) , [W3 servers](#) , etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#) ,X11 [Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#))

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

[Getting code](#)

Getting the code by [anonymous FTP](#) , etc.

<https://info.cern.ch/hypertext/WWW/TheProject.html>

Reasons to learn Accessibility!

- It's fun!
- We're the ones making the web inaccessible
- Human Rights
- Legal Issue
- Reach a larger audience
- Impactful
- Makes you a specialist

The Legal Landscape

Web accessibility and the law (US focused)

- web accessibility is a legal requirement under the americans with disabilities act (ada).
- lawsuits & regulations are increasing.
- non-compliance can result in **finances**, **lawsuits**, and **reputational damage**.

Key laws and standards

- ADA Title II & III → applies to government & businesses.
- Section 508 → federal agencies must comply.
- WCAG 2.1 level aa → the standard for web accessibility.
- New DOJ rules (2024) make compliance explicit.

Recent lawsuits and fines

- 2023: lawsuits targeting inaccessible sites rose 62%.
- Whirlpool (Kitchenaid) → sued for barriers on its website.
- Katz's deli (2025) → fined \$20k for ada violations.
- Fines can reach \$150k per violation.





Who is being sued?

Industries facing high legal risk:

- e-commerce
- finance & banking
- hospitality & restaurants
- healthcare

Overlays & widgets don't prevent lawsuits!

How to stay compliant

-  follow wcag 2.1 level aa standards.
-  test with real users & assistive tech.
-  use automated tools (lighthouse, axe, wave).
-  train teams on inclusive design.

AKA, take this course 

Assistive Technologies

Amazing ways people use the web

Keyboard Only



Head wand and mouth stick



Single Switch



Screen Readers



Eye tracker keyboard



The curb cut effect



The Curb-Cut Effect, in its essence, asserts that an investment in one group can cascade out and up and be a substantial investment in the broader well-being of a nation -- one whose policies and practices create an equitable economy, a healthy community of opportunity, and just society.

Exercise 0 - Screen Readers

- <https://github.com/jkup/learn-ally>
- <https://learn-ally.netlify.app/>

Grab the source code, follow the instructions to install a free screen reader on your computer!

Standards and Guidelines

- Web Content Accessibility Guidelines (WCAG)
- The W3C Web Accessibility Initiative (WAI)
- Accessible Rich Internet Applications (WAI-ARIA)

W3C

- The World Wide Web Consortium (W3C)
- The International Standards Organization for the World Wide Web
- Creates the Standards for HTML, CSS and Accessibility
- <https://www.w3.org/>

WAI

- The W3C Web Accessibility Initiative (WAI)
- An Initiative within the W3C focused on Accessibility
- The authors of the official Accessibility Specification (WCAG)
- <https://www.w3.org/WAI/>

WCAG

- Web Content Accessibility Guidelines (WCAG)
- The official standard, published by WAI
- <https://www.w3.org/WAI/standards-guidelines/wcag/>

WAI-ARIA

- Accessible Rich Internet Applications
- Accessibility standard that goes beyond semantic HTML
- <https://www.w3.org/WAI/standards-guidelines/aria/>

WebAIM

- Not an official standards body
- An **independent** advocacy and training group
- Provides easy to understand recommendations for Accessibility
- <https://webaim.org/>

WCAG Conformance Levels

- A (lowest)
- AA (mid range)
- and AAA (highest)
- <https://www.w3.org/WAI/WCAG21/Understanding/conformance#levels>

Level A sets a minimum level of accessibility and does not achieve broad accessibility for many situations. For this reason, UC recommends AA conformance for all Web-based information.

WCAG POUR Principles

POUR Principles

The Four Principles of Web Accessibility
as defined by the WCAG 2.0 Guidelines



Perceivable

Users can recognize the presented information via sight, hearing, or touch.



Operable

Users can navigate and operate the user interface via alternative input methods.



Understandable

Users can make sense of the textual, visual, and audio content and available operations.



Robust

A wide variety of web browsers and assistive technologies can interpret the information.

Screen Readers

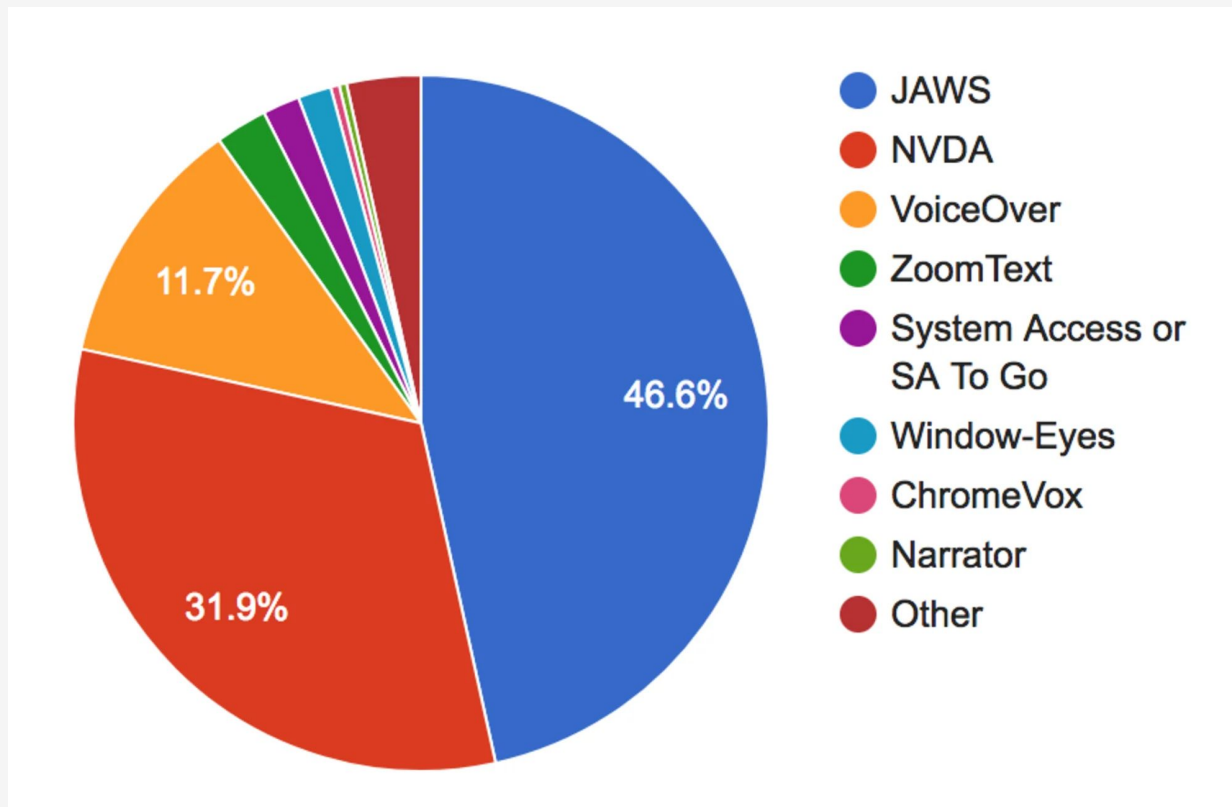
How do they work?

Screen readers convert digital text into synthesized speech. They empower users to hear content and navigate with the keyboard. The technology helps people who are blind or who have low vision to use information technology with the same level of independence and privacy as anyone else.

Screen Readers

1. Read all content
2. Display a list of links
3. Display a list of headings

Popular screen readers



Alternative Text

By default, when a screen reader encounters an image, if it can't find alt text it will read aloud the file's name.

This gets especially tricky for user generated images which often get hashed file names.

```
1 <!-- Flickr -->
2 
5
6 <!-- Facebook -->
7 
13
14 <!-- Twitter -->
15 
16
```

Alt Text

Adding an alt attribute will override that behavior. Screen readers will read the alternative text instead of the file name.

```
  

```

Skipping over images

Sometimes your website will have images that are strictly for decorative purposes. In that case, an empty alt attribute will force the screen reader to skip over the image.



```

```

A note on SEO

Search engines also make use of alternative text. For years SEO shops have suggested stuffing the keywords you want to rank for into alt text wherever possible. This provides a very bad accessibility experience.

Success Criteria	WebAIM's Recommendations
1.1.1 Non-text Content (Level A)	<ul style="list-style-type: none">❑ Images, form image buttons, and image map hot spots have appropriate, equivalent alternative text.❑ Images that do not convey content, are decorative, or contain content that is already conveyed in text are given empty alternative text (alt="") or implemented as CSS backgrounds. All linked images have descriptive alternative text.❑ Equivalent alternatives to complex images are provided in context or on a separate linked page.❑ Form buttons have a descriptive value.❑ Form inputs have associated text labels.❑ Embedded multimedia is identified via accessible text.❑ Frames and iframes are appropriately titled.

Captions for audio

Remember not all content on the web is visual! If your application has video content, be sure to use a captioning service to add closed captioning.

Success Criteria	WebAIM's Recommendations
1.2.1 Prerecorded Audio-only and Video-only (Level A)	<ul style="list-style-type: none">❑ A transcript of relevant content is provided for non-live audio-only (audio podcasts, MP3 files, etc.).❑ A transcript or audio description of relevant content is provided for non-live video-only, unless the video is decorative.
1.2.2 Captions (Prerecorded) (Level A)	<ul style="list-style-type: none">❑ Synchronized captions are provided for non-live video (YouTube videos, etc.).
1.2.3 Audio Description or Media Alternative (Prerecorded) (Level A)	<ul style="list-style-type: none">❑ A transcript or audio description is provided for non-live video. NOTE: Only required if there is relevant visual content that is not presented in the audio.

Semantic HTML

Some elements have semantic meaning but no special functionality.

- `<aside>`
- `<footer>`
- `<header>`

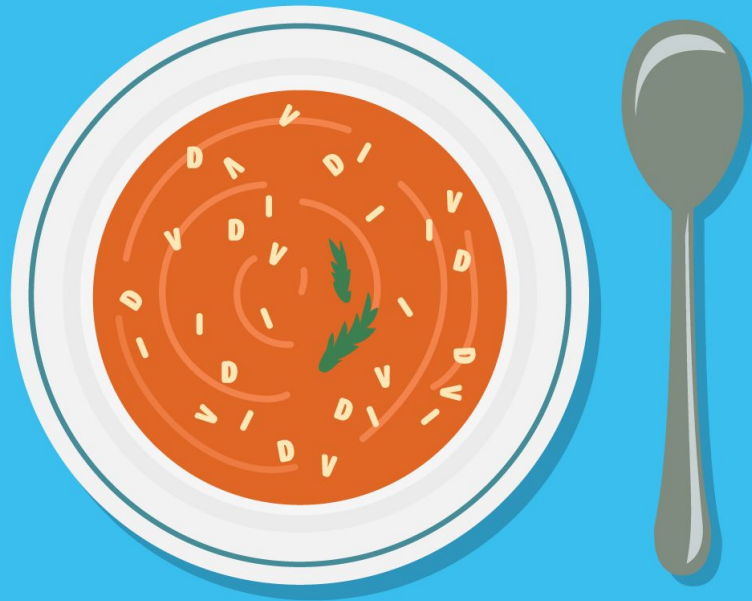
Other's provide a lot of built-in functionality such as:

- `<button>`
- `<input>`
- `<textarea>`

Success Criteria	WebAIM's Recommendations
<u>1.3.1 Info and Relationships</u> (Level A)	<ul style="list-style-type: none">❑ Semantic markup is used to designate headings (<h1>), regions/landmarks, lists (, , and <dl>), emphasized or special text (, <code>, <abbr>, <blockquote>, for example), etc. Semantic markup is used appropriately.❑ Tables are used for tabular data and data cells are associated with their headers. Data table captions, if present, are associated to data tables.❑ Text labels are associated with form input elements. Related form elements are grouped with fieldset/legend. ARIA labelling may be used when standard HTML is insufficient.
<u>1.3.2 Meaningful Sequence</u> (Level A)	<ul style="list-style-type: none">❑ The reading and navigation order (determined by code order) is logical and intuitive.

Div Soup

**ANYONE
FOR DIV
SOUP?**



Form Labels

Form fields can be confusing for screen reader users. There are many ways to label form fields so the label is read out loud whenever the field has focus.

Antipattern - Visual only labels

A common pattern is using div's or paragraph tags to label form fields.

```
<form>
  <p>First Name</p>
  <input type="text" />
  <p>Last Name</p>
  <input type="text" />
  <p>Password</p>
  <input type="password" />
  <input type="submit" value="Submit" />
</form>
```

Antipattern - Visual only labels

A screen reader will read this as “Edit text, blank” to the user.

```
<form>
  <p>First Name</p>
  <input type="text" />
  <p>Last Name</p>
  <input type="text" />
  <p>Password</p>
  <input type="password" />
  <input type="submit" value="Submit" />
</form>
```

HTML Labels

A better option is to use the HTML label tag which will allow screen readers to recite the label when the field is focused.

```
<form>
  <label for="first">First Name</label>
  <input id="first" type="text" />

  <label for="last">Last Name</label>
  <input id="last" type="text" />

  <label for="password">Password</label>
  <input id="password" type="password" />

  <input type="submit" value="Submit" />
</form>
```

HTML Labels

A screen reader will read this as “First Name, edit text, blank” to the user.

```
<form>
  <label for="first">First Name</label>
  <input id="first" type="text" />

  <label for="last">Last Name</label>
  <input id="last" type="text" />

  <label for="password">Password</label>
  <input id="password" type="password" />

  <input type="submit" value="Submit" />
</form>
```

Implicit HTML labels

Another cool trick you can do is wrap your inputs with the label tag. This is called implicit labelling.

```
<form>
  <p>First Name</p>
  <input type="text" />
  <p>Last Name</p>
  <input type="text" />
  <p>Password</p>
  <input type="password" />
  <input type="submit" value="Submit" />
</form>
```

Limitations with the `<label>` tag

The label tag can only works with "labelable" elements. Those include:

- `<button>`
- `<input>`
- `<keygen>`
- `<meter>`
- `<output>`
- `<progress>`
- `<select>`
- `<textarea>`

aria-label

If you ever need to label an element not on that list, use `aria-label` instead.



```
<div aria-label="Interactive div">Hello</div>
```

Week #26

Divs Are Not Buttons

Accessibility Weekly

Brought to You by Equalize Digital

Let's make a button (with a div!)



```
<div onclick="alert('hello')">Click me!</div>
```

Tell screen readers it's clickable




```
<div role="button" onclick="alert('hello')">Click me!</div>
```

Allow users to tab to it



```
<div tabindex="0" role="button" onclick="alert('hello')">  
  Click me!  
</div>
```

Allow keyboard users to interact with it!



```
<div
  tabindex="0"
  role="button"
  onclick="alert('hello')"
  onkeyup="alert('hello')"
>
  Click me!
</div>
```

Tell screen readers what it does!

```
⊙ ○ ●  
  
<div  
  aria-label="Alert the word hello"  
  tabindex="0"  
  role="button"  
  onclick="alert('hello')"  
  onkeyup="alert('hello')"  
>  
  Click me!  
</div>
```

Or, just use a button



```
<button onclick="alert('hello')">Click me!</button>
```

Twitter's Div Soup

```
▼<div class="css-1dbjc4n r-150rngu r-16y2uox r-1wbh5a2 r-rthrr5"> flex
  ▼<div class="css-1dbjc4n r-aqfbo4 r-16y2uox"> flex
    ▼<div class="css-1dbjc4n r-1oszu6l r-1niwhzg r-18u37iz r-16y2uox r-1wtj0ep r-2llsf r-13qz1uu"> flex
      ▼<div class="css-1dbjc4n r-14lw9ot r-jxzhtn r-1ljd8xs r-13l2t4g r-1phboty r-1jgb5lz r-11wrixw r-61z16t r-1ye8kvj r-13qz1uu r-184en5c" data-testid="primaryColumn"> flex
        ▼<div class="css-1dbjc4n"> flex
          ▶<div class="css-1dbjc4n r-aqfbo4 r-gtdqiz r-1gn8etr r-1g40b8q">...</div> flex
          ▼<div class="css-1dbjc4n r-14lw9ot r-184en5c"> flex
            ▼<div class="css-1dbjc4n r-iphfwy"> flex
              ▶<div aria-valuemax="100" aria-valuemin="0" aria-valuenow="0" role="progressbar" class="css-1dbjc4n r-kicko2 r-notknq r-gfcssk r-ludh08x r-u8s1d r-lrvibr r-13qz1uu r-417010">...</div> flex
            ▼<div class="css-1dbjc4n r-14lw9ot r-oyd9sg"> flex
              ▼<div class="css-1dbjc4n"> flex
                ▼<div class="css-1dbjc4n"> flex
                  ▼<div class="css-1dbjc4n r-ymttw5"> flex
                    ▼<div class="css-1dbjc4n r-18u37iz"> flex
                      ▶<div class="css-1dbjc4n r-1hwwag r-18kxxzh r-1b7u577 r-1h8ys4a">...</div> flex
                      ▼<div class="css-1dbjc4n r-1iusrv4 r-16y2uox r-1777fci r-1h8ys4a r-1bylmt5 r-13tjlyg r-7qyjyx r-1ftll1t"> flex
                        ▶<div class="css-1dbjc4n r-184en5c">...</div> flex
                        <div class="css-1dbjc4n r-1awozwy r-18u37iz"></div> flex
                      <div>
                        ▼<div class="css-1dbjc4n"> flex
                          ▼<div class="css-1dbjc4n r-1awozwy r-14lw9ot r-18u37iz r-1w6e6rj r-1wtj0ep r-id7aif r-184en5c" data-testid="toolBar"> flex
                            ▶<div class="css-1dbjc4n r-1awozwy r-18u37iz r-1s2bzc4">...</div> flex
                            ▼<div class="css-1dbjc4n r-1awozwy r-18u37iz r-1s2bzc4"> flex
                              ▶<div aria-disabled="true" role="button" class="css-1dbjc4n r-l5o3uw r-420lwf r-sdzlij r-1phboty r-rs99b7 r-19u6a5r r-2yi16 r-1qi8awa r-icoktb r-1ny4l3l r-ymttw5 r-o7ynqc r-6416eg r-lrvibr" data-testid="tweetButtonInline">...</div> flex
                              </div>
                            </div>
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
```

Screen reader only content

Sometimes you'll want to communicate with a screen reader directly! One cool example is announcing to screen reader users that you offer accessibility features! In that case you can make some HTML and wrap it in a visually hidden class.

```
.visuallyhidden {  
  position: absolute;  
  left: 0;  
  top: -500px;  
  width: 1px;  
  height: 1px;  
  overflow: hidden;  
}
```

Exercise 1 - Screen Readers

Focus Management

Focus Styles

Visible focus indicators are crucial for keyboard users. In this exercise, you'll learn how to create clear, beautiful focus styles that meet WCAG requirements.

Focus Style Requirements

- Focus indicators must have a minimum contrast ratio of 3:1 against adjacent colors
- Focus areas should be at least 2px thick
- Focus styles should be visible in both light and dark modes
- Consider using multiple indicators (outline + background change)
- Animations can enhance focus visibility but shouldn't be distracting

Exercise 2 - Focus Styles

Keyboard only users

- Your entire application should be usable with only the keyboard (forms, buttons, etc)
- Some type of focus indicator should be used to provide a necessary clue as to the currently active item.
- Consider offering a shortcut for keyboard-only users to jump quickly to the main content (a skip link)

Success Criteria	WebAIM's Recommendations
<u>3.2.1 On Focus</u> (Level A)	<ul style="list-style-type: none">❑ When a page element receives focus, it does not result in a substantial change to the page, the spawning of a pop-up window, an additional change of keyboard focus, or any other change that could confuse or disorient the user.
<u>3.2.2 On Input</u> (Level A)	<ul style="list-style-type: none">❑ When a user inputs information or interacts with a control, it does not result in a substantial change to the page, the spawning of a pop-up window, an additional change of keyboard focus, or any other change that could confuse or disorient the user unless the user is informed of the change ahead of time.
<u>3.2.3 Consistent Navigation</u> (Level AA)	<ul style="list-style-type: none">❑ Navigation links that are repeated on web pages do not change order when navigating through the site.

Keyboard Shortcuts

Keyboard shortcuts are another curb cut example. Sites like Twitter and Facebook offer keyboard shortcuts for almost any action which are great for both keyboard only users and and power users!

× Keyboard shortcuts

Navigation

Shortcut help	<code>?</code>
Next post	<code>j</code>
Previous post	<code>k</code>
Page down	<code>Space</code>
Load new posts	<code>.</code>
Home	<code>g + h</code>
Explore	<code>g + e</code>
Notifications	<code>g + n</code>
Mentions	<code>g + r</code>
Profile	<code>g + p</code>
Drafts	<code>g + f</code>
Scheduled posts	<code>g + t</code>
Likes	<code>g + l</code>
Lists	<code>g + i</code>
Direct Messages	<code>g + m</code>
Grok	<code>g + g</code>
Settings	<code>g + s</code>
Bookmarks	<code>g + b</code>
Go to user...	<code>g + u</code>
Display settings	<code>g + d</code>

Actions

New post	<code>n</code>
Send post	<code>⌘ + Enter</code>
New Direct Message	<code>m</code>
Search	<code>/</code>
Like	<code>l</code>
Reply	<code>r</code>
Repost	<code>t</code>
Share post	<code>s</code>
Bookmark	<code>b</code>
Mute account	<code>u</code>
Block account	<code>x</code>
Open post details	<code>Enter</code>
Expand photo	<code>o</code>
Open/Close Messages dock	<code>i</code>

Media

Pause/Play selected Video	<code>k</code>
Pause/Play selected Video	<code>space</code>
Mute selected Video	<code>m</code>
Go to Audio Dock	<code>a + d</code>
Play/Pause Audio Dock	<code>a + space</code>
Mute/Unmute Audio Dock	<code>a + m</code>

All Facebook keyboard shortcuts



Global

Report a problem



Show shortcuts



Show shortcuts



Disabled

Search Facebook



Search

Disabled

Next result



Previous result



News Feed

See more



Disabled

Leave a comment



Jump to the next post



Previous post



Like or unlike a post



Open attachment of post



Create a post



Search Messenger contacts



Share post



Photo albums

Disabled

Enter or exit fullscreen



Previous photo



Next photo



Like photo



Communities

Previous video



Next video



Previous pinned group



Next pinned group



Search Communities



Disabled

Create an event



Single-character shortcuts



Use single-character shortcuts to perform common actions faster on Facebook.

Pin keyboard shortcut help

In the corner of your screen, you'll see shortcuts that relate to what you're doing. They'll change as you use Facebook.



Keyboard shortcuts

PLAYBACK

Toggle play/pause `k`

Rewind 10 seconds `j`

Fast forward 10 seconds `l`

Previous video `P (SHIFT+p)`

Next video `N (SHIFT+n)`

Previous frame (while paused) `,`

Next frame (while paused) `.`

Decrease playback rate `< (SHIFT+,)`

Increase playback rate `> (SHIFT+.)`

Seek to specific point in the video (7 advances to 70% of duration) `0..9`

Seek to previous chapter `OPTION + ←`

Seek to next chapter `OPTION + →`

SUBTITLES AND CLOSED CAPTIONS

GENERAL

Toggle full screen `f`

Toggle theater mode `t`

Toggle miniplayer `i`

Close miniplayer or current dialog `ESCAPE`

Toggle mute `m`

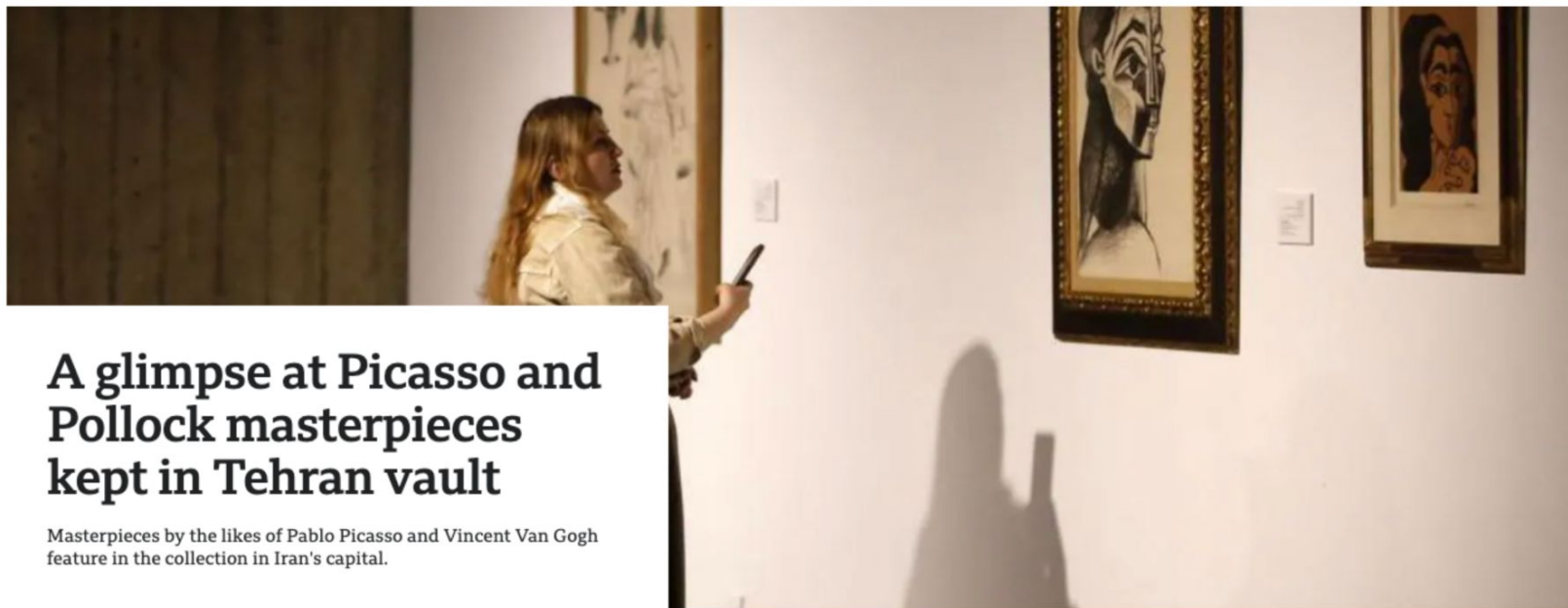
SPHERICAL VIDEOS

[Dismiss](#)

Skip Links

Skip links help users skip over large headers and navigation and jump right into the "main" content of your site. When a user hits tab for the first time, a button will appear and offer users to jump right to the main section.

Arts



A glimpse at Picasso and Pollock masterpieces kept in Tehran vault

Masterpieces by the likes of Pablo Picasso and Vincent Van Gogh feature in the collection in Iran's capital.



Skip to content

BBC

Watch Live

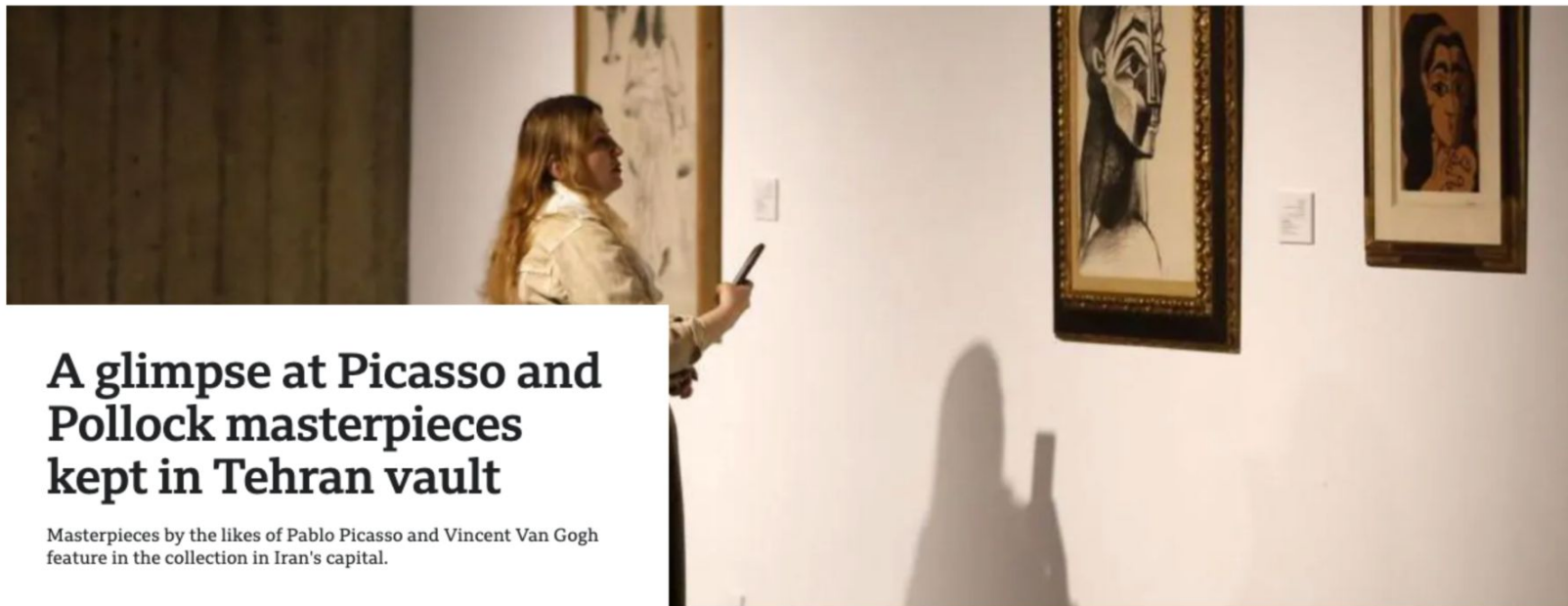
Register

Sign In

Home News Sport Business Innovation Culture Arts Travel Earth Audio Video Live

Arts in Motion

Arts



A glimpse at Picasso and Pollock masterpieces kept in Tehran vault

Masterpieces by the likes of Pablo Picasso and Vincent Van Gogh feature in the collection in Iran's capital.

<https://www.bbc.com/arts#main-content>

How to make a skip link

- Create an anchor with the body "Skip to content"
- Prepend it to the body of your website
- Make it visually hidden
- Give it a focus state which makes it visible

Success Criteria	WebAIM's Recommendations
<u>2.4.1</u> <u>Bypass Blocks</u> (Level A)	<ul style="list-style-type: none">❑ A link is provided to skip navigation and other page elements that are repeated across web pages.❑ A proper heading structure and/or identification of page regions/landmarks may be considered a sufficient technique. Because navigating by headings or regions is not supported in most browsers, WebAIM recommends a "skip" link (in addition to headings and regions) to best support sighted keyboard users.
<u>2.4.2</u> <u>Page Titled</u> (Level A)	<ul style="list-style-type: none">❑ The web page has a descriptive and informative page title.
<u>2.4.3</u> <u>Focus Order</u> (Level A)	<ul style="list-style-type: none">❑ The navigation order of links, form elements, etc. is logical and intuitive.

Tab Navigation

You can use the tab key to navigate to the next tabbable item and shift + tab to navigate to the previous item.

Success Criteria	WebAIM's Recommendations
<u>2.1.1</u> <u>Keyboard</u> (Level A)	<ul style="list-style-type: none">□ All page functionality is available using the keyboard, unless the functionality cannot be accomplished in any known way using a keyboard (e.g., free hand drawing).□ Page-specified shortcut keys and accesskeys (accesskey should typically be avoided) do not conflict with existing browser and screen reader shortcuts.
<u>2.1.2 No</u> <u>Keyboard</u> <u>Trap</u> (Level A)	<ul style="list-style-type: none">□ Keyboard focus is never locked or trapped at one particular page element. The user can navigate to and from all navigable page elements using only a keyboard.
<u>2.1.3</u> <u>Keyboard</u> <u>(No</u> <u>Exception)</u> (Level AAA)	<ul style="list-style-type: none">□ All page functionality is available using the keyboard.

Tabbable elements

- `<a>`
- `<button>`
- `<input>`
- `<select>`
- `<textarea>`
- `<iframe>`

Focusable

The screenshot shows the GitHub repository page for 'focusable' by user 'jkup'. The repository is public and has 70 stars, 18 forks, and 3 watchers. The main content area displays a list of files and their commit history:

File	Commit Message	Commit Date
.gitignore	Initial commit	9 years ago
LICENSE	Initial commit	9 years ago
README.md	Update README.md	8 years ago
index.js	use simple string	6 years ago
package-lock.json	version bump	6 years ago
package.json	version bump	6 years ago

Below the file list, the README is visible, showing the title 'Focusable' and the description 'Returns a string of focusable HTML elements'. The 'Installation' section is also partially visible.

On the right side, the 'About' section provides more details: 'Returns a string of focusable HTML elements', 'Readme', 'MIT license', 'Activity', '70 stars', '3 watching', and '18 forks'. The 'Releases' section shows '1 tags' and a link to 'Create a new release'. The 'Packages' section indicates 'No packages published' with a link to 'Publish your first package'. The 'Contributors' section shows '4' contributors.

Making an element tabbable

You can add the [tabindex](#) attribute to any element like this:



```
<div tabindex="0">I'm focusable</div>
```

TabIndex values

- a **negative** value means that the element should be focusable, but should not be reachable via sequential keyboard navigation;
- **0** means that the element should be focusable and reachable via sequential keyboard navigation, but its relative order is defined by the platform convention;
- a **positive** value means should be focusable and reachable via sequential keyboard navigation; its relative order is defined by the value of the attribute: the sequential follow the increasing number of the tabindex. If several elements share the same tabindex, their relative order follows their relative position in the document.

⚠ Warning: You are recommended to only use `0` and `-1` as `tabindex` values. Avoid using `tabindex` values greater than `0` and CSS properties that can change the order of focusable HTML elements ([Ordering flex items](#)). Doing so makes it difficult for people who rely on using keyboard for navigation or assistive technology to navigate and operate page content. Instead, write the document with the elements in a logical sequence.

Active Element

Sometimes, especially on single page applications, it's helpful to store the currently focused element before a page transition so you can return to it later.

```
// A modal is about to be opened
// Store the current news item
const currentItem = document.activeElement;
// Open the modal
// On modal close, refocus on the news item they had open
currentItem.focus()
```

Tab trapping

Another useful concept is tab trapping. Consider opening a modal on a page which contains a form. A keyboard only user will want to tab around the form but unless we help, tabbing while focused on the last form element will send us all the way back to the main document.

Exercise 3 - Tab Trapping

ARIA

W3C Recommendation

TABLE OF CONTENTS

Abstract

Status of This Document

1. Author requirements for use of ARIA in HTML

2. ARIA semantics that extend and diverge from HTML

3. Author guidance to avoid incorrect use of ARIA

3.1 Avoid overriding interactive elements with non-interactive roles

3.2 Avoid specifying redundant roles

3.3 Be cautious of side effects

3.4 Adhere to the rules of ARIA

3.5 Adhere to the rules of HTML

4. Document conformance requirements for use of ARIA attributes in HTML

4.1 Requirements for use of ARIA attributes to name elements

4.2 Requirements for use of ARIA attributes in place of equivalent HTML attributes

4.3 Requirements for deprecated ARIA role, state and property and attributes

4.4 Case requirements for ARIA role, state and property attributes

5. Allowed descendants of ARIA roles

6. Conformance

Allowed descendants of ARIA roles

Role	Kind of content	Descendant allowances
alert	Flow content	Flow content but with no main element descendants.
alertdialog	Flow content	Flow content
application	Flow content	Flow content
article	<ul style="list-style-type: none"> Flow content Sectioning content Palpable content 	Flow content but with no main element descendants.
banner	<ul style="list-style-type: none"> Flow content Palpable content 	Flow content but with no main , header , or footer element descendants.
blockquote	<ul style="list-style-type: none"> Flow content Palpable content 	Flow content but with no main element descendants.
button	<ul style="list-style-type: none"> Flow content Phrasing content Interactive content Palpable content 	Phrasing content , but with no interactive content descendants, and no descendants with a tabindex attribute specified.
caption	N/A	Flow content but with no main or table element descendants.

WAI-ARIA vs. WCAG

The main difference between WCAG and WAI-ARIA is that **WCAG sets** the overall accessibility **standards** (like requiring sufficient color contrast), while **WAI-ARIA provides techniques** to meet those standards, especially for complex, dynamic content (like using the `aria-label` attribute to give a name to a custom-built component).

Specific example

WCAG: Requires that text and interactive elements have sufficient color contrast. This is a requirement.

WAI-ARIA: Doesn't directly address color contrast. Instead, it helps screen readers understand the role of a complex widget. For instance, if you build a custom slider, WAI-ARIA attributes like `aria-valuenow`, `aria-valuemin`, and `aria-valuemax` help screen readers convey the slider's current value and range to users, fulfilling WCAG's principle of making information and user interface components presentable to users in ways they can perceive.

WAI-ARIA helps make complex components accessible in ways that simple HTML often can't, but it doesn't replace the need for basic accessibility principles like color contrast (which WCAG covers).

Labels, revisited

Earlier, we learned about the `<label>` tag in HTML and how it can be used to label **certain** form elements.

The ARIA spec provides us with great tools for labelling and describing any element we want. They are:

- `aria-label`
- `aria-labelledby`
- `aria-describedby`

labelledby vs. describedby

A label provides essential information about an object, while a description provides extended information that the user might need.

```
<form>
  <label id="quantity-label" for="quantity">Quantity:</label>
  <input type="number"
        id="quantity"
        aria-labelledby="quantity-label product-info"
        aria-describedby="quantity-description"
        value="1"
        min="1"
  >
  <p id="quantity-description">
    Enter the desired quantity (1-10).
  </p>

  <button type="submit">Add to Cart</button>
</form>
```

Roles, States and Properties

ARIA also provides roles which can be applied to any element. Examples include:

- button
- checkbox
- tree
- banner
- aria-autocomplete
- aria-haspopup

ARIA in CSS

ARIA properties can be accessed in CSS!



```
.dropdown[aria-expanded="false"] .icon::after {  
  content: '▶';  
}  
.dropdown[aria-expanded="true"] .icon::after {  
  content: '▼';  
}
```

Live Regions

Applications can become very dynamic. For cases where important information could be coming in at any time, the ARIA spec provides the ability to mark an element as containing live data so that screen readers can read out updates as they come.

Politeness

One of my favorite APIs of all time, the value that you pass in to aria-live is a politeness setting. You can pass in:

- **assertive** - will interrupt whatever it's doing to announce.
- **polite** - will announce the live region update when it next idles.
- **off** - will not read the update.

Imagine a sports application

```
<div aria-live="polite">
  <h3>Live Scores</h3>
  <div id="game1">
    <span id="team1-score">Team A: 10</span> - <span
id="team2-score">Team B: 7</span>
  </div>
  <div id="game2">
    <span id="team3-score">Team C: 14</span> - <span
id="team4-score">Team D: 12</span>
  </div>
</div>

<script>
  // ... (Code that updates scores dynamically) ...

  // Example of updating a score:
  const team1ScoreElement = document.getElementById('team1-
score');
  team1ScoreElement.textContent = "Team A: 12"; // Score has
changed!





  // No need to do anything special for the screen reader.
  // Because the parent div has aria-live, the change is
announced.
</script>
```

Exercise 4 - ARIA

Color and Contrast

Contrast Checker

[Home](#) > [Resources](#) > Contrast Checker

Foreground	Background
Hex Value # 0000FF	Hex Value # FFFFFFFF
Color Picker 	Color Picker 
Alpha 1	
Lightness 	Lightness 

Contrast Ratio

8.59:1

[permalink](#)

Normal Text

WCAG AA: **Pass**

WCAG AAA: **Pass**

The five boxing wizards jump quickly.

Large Text

WCAG AA: **Pass**

WCAG AAA: **Pass**

The five boxing wizards jump quickly.

Graphical Objects and User Interface Components

WCAG AA: **Pass**



Text Input

Foreground

Hex Value

#

Color Picker
Alpha

Lightness

Background

Hex Value

#

Color Picker

Lightness

Contrast Ratio

4.75:1

[permalink](#)

- [Quick](#)
- [Web](#)
- [Acce](#)
- [Web/](#)
- [Evalu](#)
- [Web/](#)
- [Desig](#)
- [Link](#)
- [Cont](#)
- [Book](#)

Normal Text

WCAG AA: Pass

WCAG AAA: Fail

The five boxing wizards jump quickly.

Large Text

WCAG AA: Pass

WCAG AAA: Pass

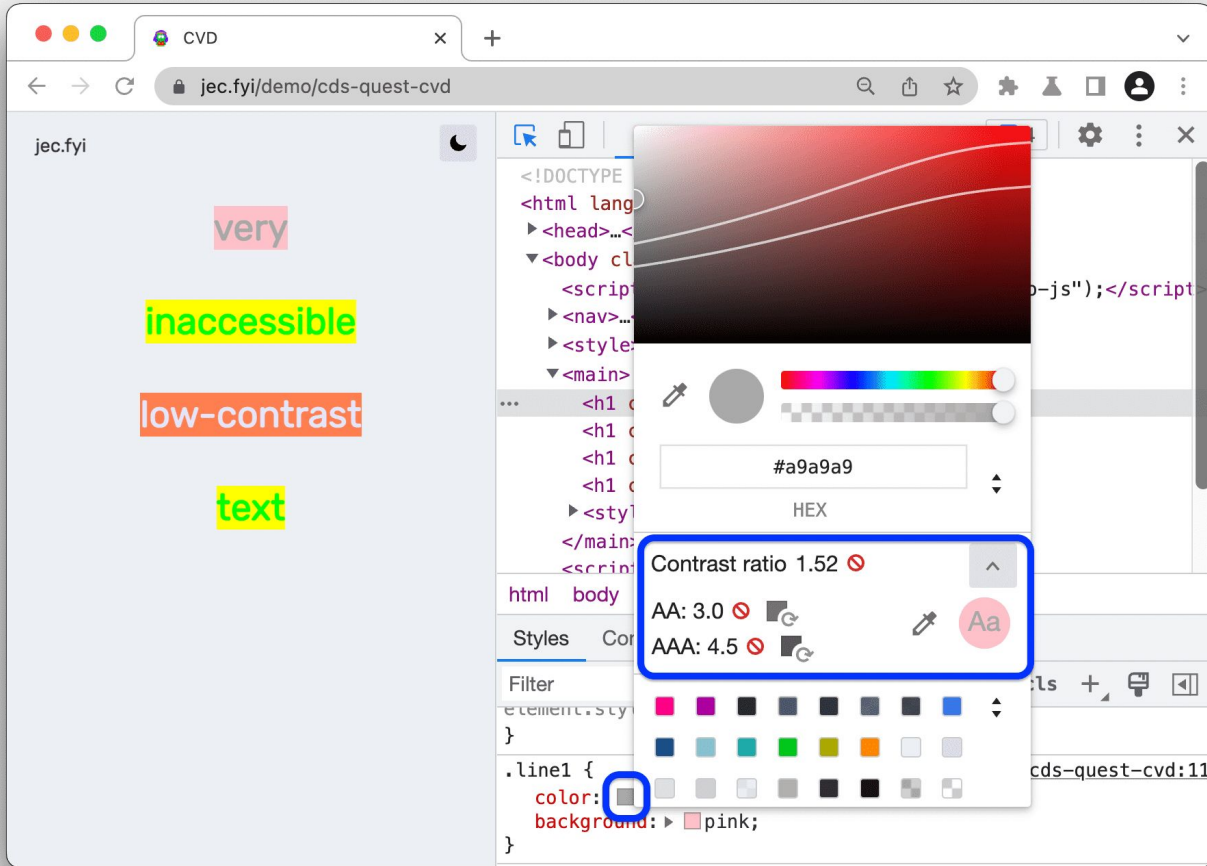
The five boxing wizards jump quickly.

Graphical Objects and User Interface Components

WCAG AA: Pass

★

Text Input

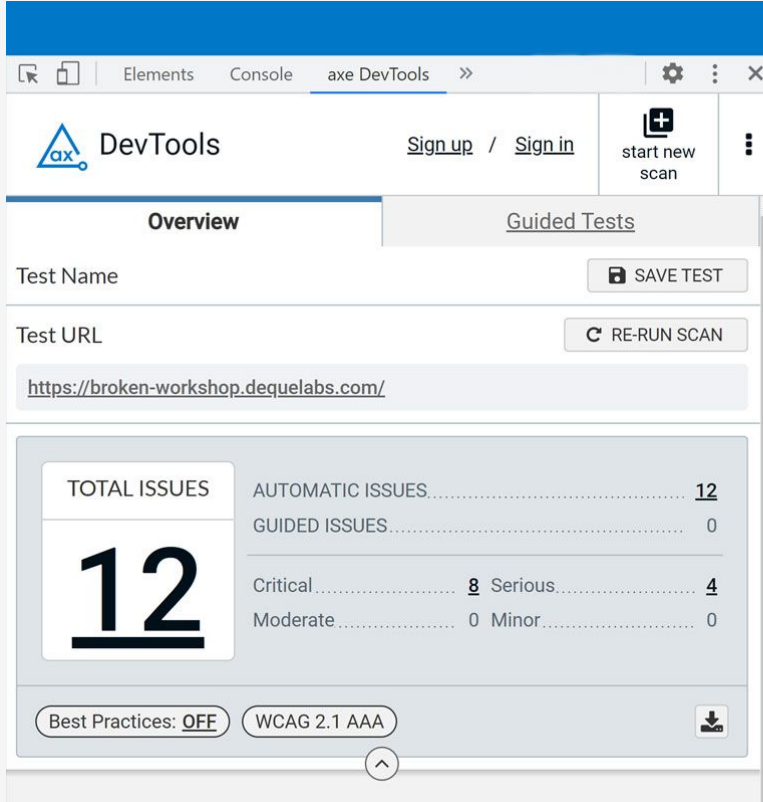


Exercise 5 - Color Contrast

Testing and Tooling

- DevTools
- CI/CD audits
- Linters

AXE DevTools by Deque



The screenshot displays the AXE DevTools interface within a browser window. The browser's address bar shows the URL `https://broken-workshop.dequelabs.com/`. The interface includes a header with the AXE logo, "DevTools" text, and links for "Sign up" and "Sign in". A "start new scan" button is also visible. Below the header, there are two tabs: "Overview" (selected) and "Guided Tests". The "Overview" tab contains a "Test Name" field with a "SAVE TEST" button and a "Test URL" field with a "RE-RUN SCAN" button. The main content area shows a summary of scan results:

TOTAL ISSUES	AUTOMATIC ISSUES.....	12
12	GUIDED ISSUES.....	0
	Critical.....	8
	Serious.....	4
	Moderate.....	0
	Minor.....	0

At the bottom of the interface, there are two toggle buttons: "Best Practices: OFF" and "WCAG 2.1 AAA".

Automatically
catch
accessibility
defects
while you code

Google Lighthouse

68

Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

ARIA

- ▲ [\[role\]](#)s do not have all required [\[aria-*\)](#) attributes
- ▲ Elements with an ARIA [\[role\]](#) that require children to contain a specific [\[role\]](#) are missing some or all of those required children.
- ▲ [\[aria-*\)](#) attributes do not have valid values
- ▲ [button](#), [link](#), and [menuitem](#) elements do not have accessible names.

These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

Wave by WebAIM

The image shows the Wave accessibility evaluation tool interface overlaid on a hotel booking page. The tool's sidebar on the left provides a summary of accessibility issues and features.

Wave Summary:

- powered by [WebAIM](#)
- web accessibility evaluation tool
- Styles: OFF ON
- Summary: 10 Errors, 1 Contrast Error, 39 Alerts, 75 Features, 202 Structural Elements, 1364 ARIA
- Navigation: Summary (selected), Details, Reference, Structure, Contrast
- View details >

Hotel Booking Page Content:

- h4 St. George 4.6 • 701 Reviews aria *role="link"*
- DATES (1 NIGHT) Thu, Jul 28 → Fri, Jul 29
- ROOMS & GUESTS 1 Room, 1 Adult aria *aria-label="Select number of guests" dropdown*
- Map showing locations: St. George Municipal Airport, Harry Reid International Airport, Salt Lake City International Airport. aria *aria-controls="accordion-body"*
- Map labels: Shivwits, Ivins, Santa Clara. aria *role="presentation"*
- Map interaction: aria *aria-label="Open this area"*

Axe-core by Deque

The screenshot shows the GitHub repository page for 'axe-core' by 'dequelabs'. The repository is public and has 174 watchers, 793 forks, and 6.2k stars. The current branch is 'develop', with 74 other branches and 112 tags. The repository contains several files and folders, including '.circleci', '.github', '.husky', '.vscode', 'build', and 'doc'. The most recent commit is by 'dbjorge' with the message 'fix(target-size): do not treat focusable tabpanels as targets (#4702)', made 60 days ago. The repository also has a README, an MPL-2.0 license, a code of conduct, and a security policy.

dequelabs / axe-core

Type [Z] to search

<> Code Issues 355 Pull requests 19 Actions Projects Security Insights

axe-core Public Watch 174 Fork 793 Star 6.2k

develop 74 Branches 112 Tags Go to file Add file <> Code

dbjorge fix(target-size): do not treat focusable tabpanels as targets (#4702) 60d11f2 · last week 5,191 Commits

.circleci	test: resolve nightly failure installing firefox 135 (#4698)	2 weeks ago
.github	chore: don't upgrade glob@11 due to node 18 support (#...	last month
.husky	chore: bump husky from 8.0.3 to 9.0.7 (#4320)	last year
.vscode	chore: set up vscode config to attach to test:debug (#44...	10 months ago
build	docs: Update rule-descriptions.md to include link to ACT ...	last month
doc	chore: sync generated files (#4674)	last month

About

Accessibility engine for automated Web UI testing

www.deque.com/axe/

accessibility a11y axe

Readme

MPL-2.0 license

Code of conduct

Security policy

Activity

Axe-core by Deque

```
axe
  .run()
  .then(results => {
    if (results.violations.length) {
      throw new Error('Accessibility issues found');
    }
  })
  .catch(err => {
    console.error('Something bad happened:', err.message);
  });
```

Lighthouse CI

GoogleChrome / lighthouse-ci

Type to search

<> Code Issues 200 Pull requests 17 Discussions Actions Security Insights

lighthouse-ci Public Watch 79 Fork 653 Star 6.5k

main 21 Branches 56 Tags

Go to file Add file <> Code

dvelasquez docs: added actions-lighthouseci-compare (#1054) de45968 · 4 months ago 988 Commits

.github	chore: bump to node 18 (#989)	2 years ago
docs	chore: bump lhci references to 0.14 (#1050)	8 months ago
packages	fix(cli): assert - destructure lhr from options (#1062)	7 months ago
scripts	misc(release): set npm tag correctly	8 months ago
types	fix(cli): respect collect puppeteerLaunchOptions.headles...	8 months ago
.browserslistrc	refactor: split package into cli and utils	6 years ago
.eslintrc.js	chore(deps): upgrade to storybook 7.6.4, preact 10.19.3, e...	2 years ago
.eslintrc.tests.js	chore(deps): update lint deps and use implicit configurati...	3 years ago

About

Automate running Lighthouse for every commit, viewing the changes, and preventing regressions

webperf lighthouse

Readme Apache-2.0 license Activity Custom properties 6.5k stars 79 watching 653 forks Report repository

Lighthouse CI - GitHub Workflow

```
name: CI
on: [push]
jobs:
  lighthouseci:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: 18
      - run: npm install && npm install -g @lhci/cli@0.14.x
      - run: npm run build
```

Authoring Tools

Linters

- [eslint-plugin-jsx-ally](#)
- [Angular Codelyzer](#)
- [eslint-plugin-vuejs-accessibility](#)

Accessible Design Systems

- [Adobe's React Spectrum](#)
- [Google's Material Design](#)

Accessible Components and Libraries

The screenshot shows the React Spectrum website interface. On the left is a dark sidebar with a navigation menu. The main content area features the title 'React Spectrum' and a subtitle 'A React implementation of Spectrum, Adobe's design system.' Below this are links for 'Get started' and 'GitHub'. A large central illustration depicts a hand interacting with a stylized UI element. On the right is a 'Contents' sidebar listing various component categories.

React Spectrum

INTRODUCTION

- Getting Started
- Versioning

CONCEPTS

- Layout
- Styling
- Testing
- Theming
- Server Side Rendering
- Client Side Routing
- Forms
- Drag and Drop

APPLICATION

- Provider

LAYOUT

- Flex
- Grid

React Spectrum

A React implementation of Spectrum, Adobe's design system.

[Get started](#) • [GitHub](#)

Contents

- Components**
 - Buttons
 - Pickers
 - Collections
 - Date and time
 - Color
 - Overlays
 - Forms
 - Navigation
 - Status
 - Drag and drop

Accessible **Adaptive**

Exercise 6 - Accessibility Audit

Resources

- [Web Content Accessibility Guidelines \(WCAG\)](#)
- [WebAIM](#)
- [Microsoft Inclusive Design](#)
- [Global Accessibility Awareness Day](#)
- [Web App Accessibility \(feat. React\)](#)

Thanks!