How to Make JavaScript Accessible

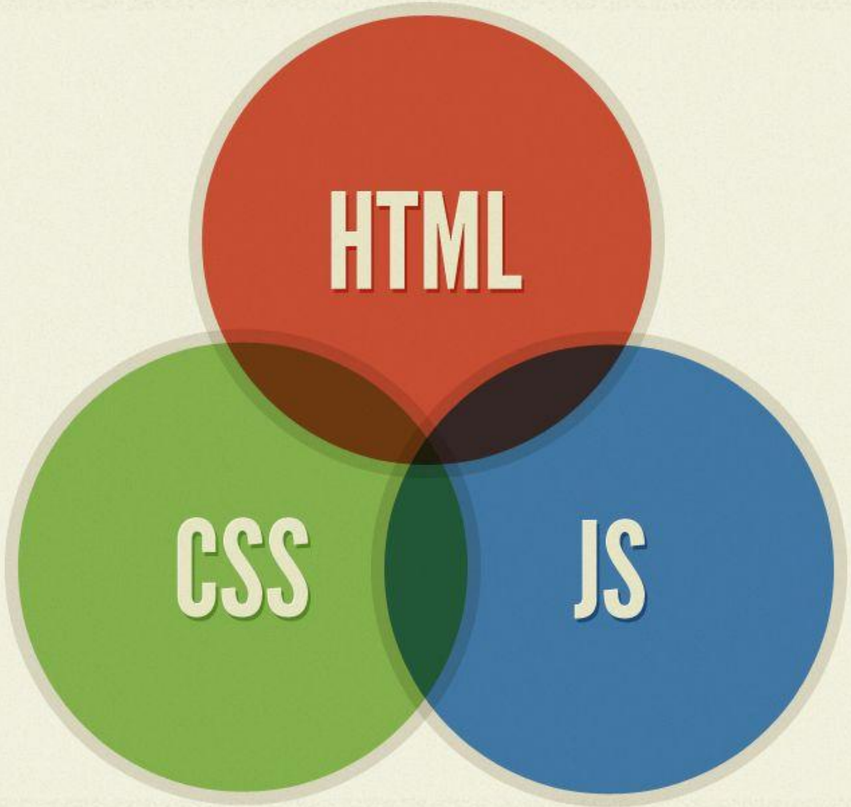
Without Losing Your Sanity (or Your Hair)

10-April-2024 • Todd Libby

dev.nexus 2024

Thanks!


<html>
{.css}


HTML
CSS
JS

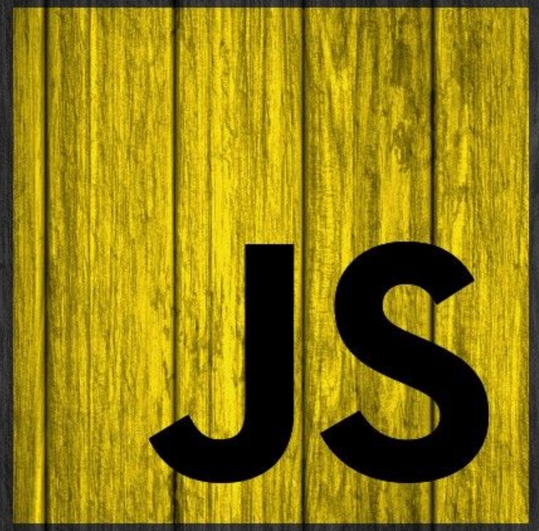

415


JavaScript

Data

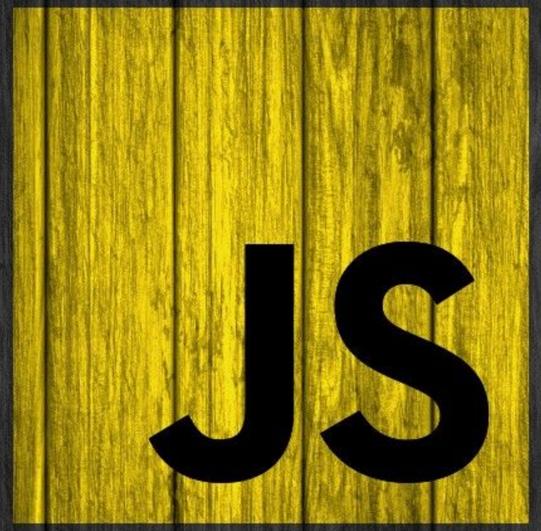
The numbers behind the talk

1.2 Billion



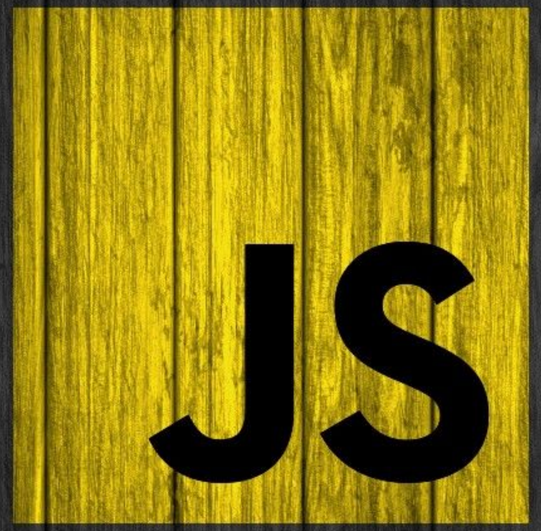
Page elements tested in 2024 by WebAIM

1173



Up from 1050 elements per home page in 2023

95.9%



of home pages had a detected WCAG 2.x failure!

Home pages with most common WCAG 2 failures

WCAG Failure Type	% of home pages in 2024	% of home pages in 2023	% of home pages in 2022	% of home pages in 2021	% of home pages in 2020	% of home pages in 2019
Low contrast text	81.0%	83.6%	83.9%	86.4%	86.3%	85.3%
Missing alternative text for images	54.5%	58.2%	55.4%	60.6%	66.0%	68.0%
Missing form input labels	48.6%	45.9%	46.1%	54.4%	53.8%	52.8%
Empty links	44.6%	50.1%	49.7%	51.3%	59.9%	58.1%
Empty buttons	28.2%	27.5%	27.2%	26.9%	28.7%	25.0%
Missing document language	17.1%	18.6%	22.3%	28.9%	28.0%	33.1%

```
<!DOCTYPE html>
<html lang="en">
<head>
```

Frameworks and Libraries

and JavaScript, oh my!

JavaScript Frameworks

Most Improved in Accessibility

- Stimulus
- Emotion
- React
- Zone.js
- Angular
- styled-components
- MooTools

JavaScript Libraries

Most Improved in Accessibility

- YUI
- Polyfill
- Lodash
- AOS
- jQuery Mobile
- lit-html

Accessible JavaScript

You can make it possible!

- Keyboard & Mouse
- Form validation
- Dynamic content
- Styling
- Performance
- Unobtrusive JavaScript

Examples

Show me the accessibility!

Keyboard & Mouse

```
<ul class=“accordion”>
  <li>
    <h3>Todd Libby</h3>
    <p>Chief Accessibility Officer</p>
    … // More info
  </li>
  …
</ul>
```

<html>

JS

```
const acc = accordion.getElementsByTagName('li');
  for (var i = 0; i < acc.length; i++)
{
  var accTitle = acc[i].getElementsByTagName('h3');
  addEvent(accTitle, 'click', Acc.clickListener);
}
```

```
clickListener: function(event)
{
  var fold = this.parentNode;
  Acc.expand(fold);
  preventDefault(event);
},
```

A. Make non-keyboard-focusable elements keyboard-accessible.

B. Add an element that is keyboard-accessible (e.g., hyperlink) to the document.

```
<ul class=“accordion”>
  <li id=“position-cao”>
    <h3><a href=“#cao”>Todd Libby</a></h3>
    <p>Chief Accessibility Officer</p>
    … // More info
  </li>
  … // Other code
</ul>
```

```
const acc = accordion.getElementsByTagName('li');
for (var i = 0; i < acc.length; i++)
{
  var accLinks = acc[i].getElementsByTagName(‘a');
  var accTitleLink = accLinks[0];
  addEvent(accTitleLink, 'click', Acc.clickListener);
}
```

```
clickListener: function(event)
{
  var fold = this.parentNode.parentNode;
  Acc.expand(fold);
  preventDefault(event);
},
```

tabindex=0

<details>

- Native HTML
- Well supported*
- Supports ***toggle*** event
- Low JS overhead**
- Accessible
- Use with ***<summary>***
- **Very customizable**

Form Validation

Validating Your Forms

- Instant validation (field validated on every value change) - NO!
- Validating after the fact (field validated on blur) - Not a big fan
- Validation on submission of form (Click and validate) - YES!

```
const form = document.querySelector(‘form’);

form.addEventListener(‘submit’, function(event) {
    event.preventDefault(); // Prevent form submission

    // Prevent custom validation logic
    const email = document.getElementById(‘email’).value;
    const password = document.getElementById(‘password’).value;

    if (!emailIsValid(email)) {
      alert(‘Password must be between 8-20 characters long’);
      return;
    }

    …

    // Submit the form if validation passes
    form.submit();
  });
```

```
<p aria-live="assertive" class="sr-only">
  Not a valid email address.
</p>

<p aria-hidden="true">
  Not a valid email address.
</p>
```

HTMLElement.focus()

Form Accessibility

- Inline error messages
- Place all errors **above** the form in long forms
- Clear and concise
- Do not rely on **color** alone
- Focus where important
- Stop disabling buttons
- Don’t wrap everything in a **<label>**

Buttons

<button>

<a>

```
const button =
document.createElement('div')

button.innerText = 'Click
Here…'
```

<button>

```
<div class="menu-button-links">
  <button type="button"
    id="menubutton"
    aria-haspopup="true"
    aria-controls="menu2"
    aria-expanded="false">
    WAI-ARIA Quick Links
    <svg xmlns="http: //www.w3.org/2000/svg"
      …
    </svg>
  </button>
  <ul id="menu2"
    role="menu"
    aria-labelledby="menubutton">
    <li role="none">
      …
    </li>
    …
    <li role="none">
      …
    </li>
  </ul>
</div>
```

https://www.w3.org/WAI/content-assets/wai-aria-practices/patterns/menu-button/examples/js/menu-button-links.js

Styling

Yes! CSS & Accessibility

- Use semantics and style them accordingly
- Apply CSS if you have to with `setAttribute`
- Do **NOT** style headings to look like other headings
- Sensible typography
- Readability & contrast of text
- You do not need a framework for everything!

Performance

Performance of JS

- Consider a simpler solution
- Remove unused code
- Consider built-in user agent features:
  - client-side validation
  - native <video> player
  - CSS animations
- Minification, Gzipping, bundlers

Performance of JS

1. HTML is parsed first.
2. CSS parsed when applicable
    a. Linked assets start to be fetched.
3. JS is parsed by the browser, evaluates the JS and runs it.
4. The user agent decides how each HTML element should be styled.
5. The result of the styling is painted to the screen.

Unobstrusive JS

Mixing is for cooking and drinks.

Benefits of Unobtrusive JS

- Separation of behaviors from the markup
- Reusable code
- Progressive enhancement
- Graceful degradation
- Capability detection

```
<a
 id=“alert”
 href=“javascript:void(0);”
 onClick=“alert(‘You clicked the
 button.’);”
>
 Click this super button!
</a>
```

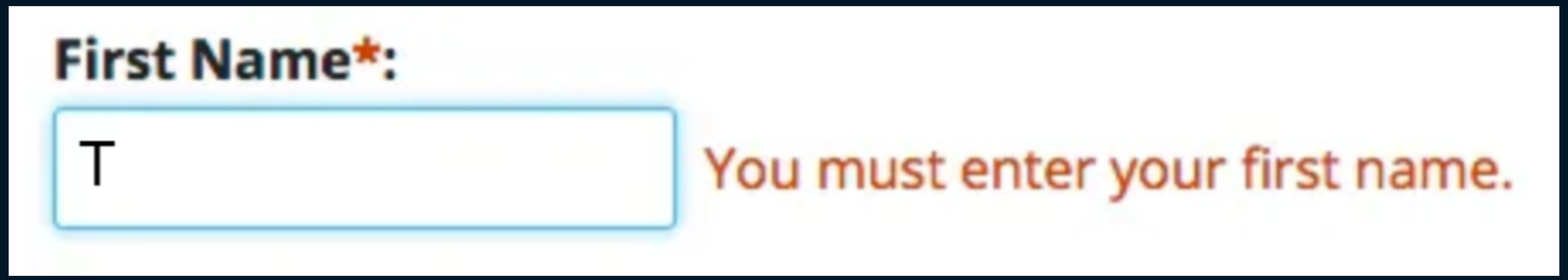
```
<script>
 jQuery(function($) {
  $(“#alert”).click.function() {
   alert(“Thanks for clicking the button!”);
   return false;
  });
 });
</script>

<a id=“alert” href=“#”>Click this button!</a>
```

ARIA - Accessible Rich Internet Application

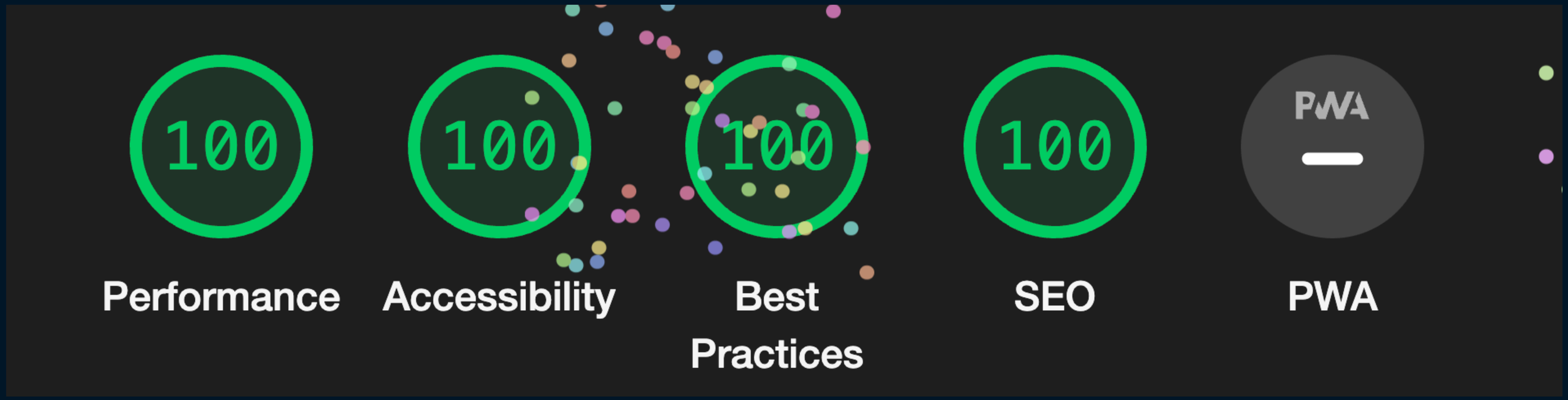
Wrap-up

```
<div
 role="button"
 tabindex="0"
 onClick={this.onClick}
>
 Click Here
</div>
```


First Name*:
T
You must enter your first name.

```
<span
 role="button"
 tabindex="0"
 onClick={this.onchange}
>
 Super Sleek Button!
</span>
```

```
document.getElementById('hero').style =
'font-size: 12rem;';
```


100
100
100
100
PWA
Performance
Accessibility
Best Practices
SEO
PWA

Slides: https://bit.ly/4cStCsF

Thank you!