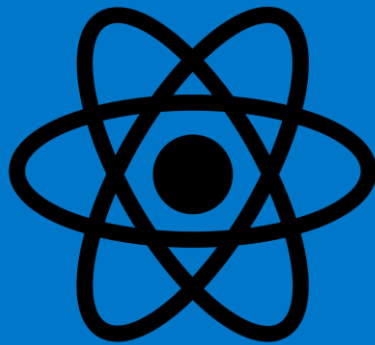




Single Page Applications, One Year Later



508



AUDIT

WCAG

ARIA

ADA

Introductions

-  Mark Steadman
 - Former Fortune 50 Employee, now Accessibility Consultant with Deque.
 - Twitter: @Steady5063
 - Email: mark.steadman@deque.com
-  CB Averitt
 - Principal Accessibility Consultant
 - Twitter: @dive4cb
 - Email: cb.averitt@deque.com

One Year Ago.....

- At CSUN 2018, we presented on the big issues with single page applications, and some remedies for how to deal with some of those issues.
- Reason behind it:
 - We saw a large trend of organizations and websites in general moving to JavaScript frameworks.

One Year Ago.. Within the fortune 50 company

- Then something more happened within the organization
 - Structured framework was taken away and no longer required
 - Fully Accessible framework with consumable components
 - Free to choose whatever framework, language, and design
 - Idea was more freedom for developers. (Less overhead)

Today's Agenda

- Approach to handling the change of multiple single page application frameworks within fortune 50 company
 - Top Frameworks in market
 - The problems within single page frameworks
 - The specific accessibility issues that exist within these frameworks
 - Larger focus on component packages (NPM)
- One year later
 - The results of research and documentation
 - Accessibility support guides

Our approach

- What are the top frameworks being used within fortune 50 company and externally (other industries)
- Overview of Single Page Applications
- What are some of the BIG challenges specifically.
 - Generically across all different frameworks
- Documentation of challenges
- Developer guidance

Top Frameworks

- According to NPM stats (2017-18 timeframe)
 - 59,087,000 ReactJS
 - 11,324,000 AngularJS
 - 7,929,000 VueJS
 - 3,800,000 Ember
- What we found within the fortune 50 company was opposite of data
 - Top ones are EmberJS and AngularJS
 - We decided to address these two frameworks first

Credit: <https://npm-stat.com/>

Overview: Single Page Applications

- A single-page application is a website that re-renders its content in response to navigation actions without making an additional request to the server to fetch new HTML. (Single DOM load)
- Highly interactive applications leveraging client-side rendering techniques
- Primarily driven by the ease of use for mobile development (responsive design)

General Accessibility Challenges

- Page/View Titles
- Keyboard Navigation
- Focus Management
- Dynamically Added Content - ARIA Live and role="alert"
- Component Libraries
 - Document & Heading Structure
 - Buttons, Tables, Links marked up without native html tags
 - Extra added markup (out of box rendering)

Page (View & Route) Titles

- Natural to think “single page application, single page title”
 - And many people assume there is no way to change this.
- Changing view/route (Page) is the same as changing to a new page
- Typical excuse “There’s only one index I can’t change title, view to view”

Keyboard Navigation

- Barriers for keyboard users:
 - Injected Content
 - Bad node packages used (Ex: `<button>` created using `<div>`)
 - Bad document structure
- Many developers think everything has to be accessed via tab.
 - Multiple forms of documentation for single pages applications that say, “Just add a `tabindex=“0”` to make it accessible
 - Packages contain content that have a `tabindex=“0”` inappropriately

Focus Management

- Content not currently present on pages and the way single page applications compile views can make it difficult to maintain a logical focus order
- Great examples:
 - Modals
 - Dynamically added content
 - Status Messages

ARIA-Live and role='alert'

- ARIA-Live (or role="alert") has been a pretty steady way to alert users of status change, or any type of announcement.
 - However, with Single Page frameworks, this can very inconsistent
- With the Async events happening to inject the content, the announcement, whether live or alert, may not be read in time for screen reader.

Component Library/Packages (npm)

- Document Structure
 - Many packages are not accessibility tested
 - Causing major issues in semantics of the page itself
- Often, heading structure and landmarks
- Many components only truly think about that components markup and not the effect it has on the others

Component Library/Packages (npm)

- Buttons, Tables, Links, even Input boxes marked up not using native tags
 - Most packages try to make the most out of using CSS to design “the next best looking item” or the “slick design”
 - Choosing ease of use for user (developer)

Component Library/Packages (npm)

- Buttons made with <div>

```
<div v-on:click="toggle" class="btn btn-primary">Bad Button</div>
```

```
<button v-on:click="toggle" class="btn btn-primary" id="log-in-button"> Good Button</button>
```

Example of bad button



Bad Button

Example of a good button



Good Button

Example made with vue.js

Component Library/Packages (npm)

- Tables made with <div>

```
<h2>Phone Numbers</h2>
```

```
<div class="rTable">
```

```
  <div class="rTableRow">
```

```
    <div class="rTableHead"><strong>name</strong></div>
```

```
    <div class="rTableHead"><span style="font-weight:bold;">Telephone</span></div>
```

```
    <div class="rTableHead">Birthdate</div>
```

```
</div>
```

Component Library/Packages (npm)

- Extra HTML or ARIA added to components
 - Package creators try to make content based on what they find on the internet and then make it “accessible”
 - Example:

```
<div class="welcome-wrapper">  
  <div id="ember387" class="ember-view">  
    <div class="welcome-details-1"> You are now</div>  
  </div>  
  <div id="ember388" class="ember-view">  
    <div class="welcome-details-2 largeText">Logged in! </div>  
  </div>  
</div>
```



One Year Later

Results for our Organization

- Able to create awareness of Single Page Application accessibility challenges
 - Through Lunch and Learns
 - Through developer guides
- Broke down the issues into developer based guides specific to framework
 - Helping with the above issues
 - We began to create guide on accessible packages
- We were able to get our SME's knowledge of Single Page Application terminology

Accessibility Support Part 1

- EmberJS
 - <https://github.com/ember-a11y>
 - <https://www.emberjs.com/blog/2018/06/17/ember-accessibility-and-a11y-tools.html>
- ReactJS
 - <https://reactjs.org/docs/accessibility.html>
 - <https://github.com/reactjs/react-a11y>
 - <https://www.npmjs.com/package/react-aria>
 - <https://www.npmjs.com/package/react-accessible-tooltip>
 - <https://simplyaccessible.com/article/react-a11y/>

Accessibility Support Part 2

- AngularJS-

- <https://docs.angularjs.org/guide/accessibility>
- <https://github.com/dequelabs/ngA11y>



- VueJS-

- <https://medium.com/@emilymears/getting-started-with-web-accessibility-in-vue-17e2c4ea0842>
- <https://alligator.io/vuejs/vuejs-accessible-announcements/>
- <https://github.com/vue-a11y/vue-axe>
- <https://github.com/vue-a11y>
- <https://vuejsexamples.com/accessibility-auditing-for-vue-js-applications/>

Results Cont.

- Began to see a decline in the number of accessibility issues within our single page application teams
- Began to see a large increase in teams using these frameworks being proactive in accessibility and willing to take it as their responsibility to make sure their content is/was accessible.

Any questions?

-  Mark Steadman
 - Former Fortune 50 Employee, now Accessibility Consultant with Deque.
 - Twitter: @Steady5063
 - Email: mark.steadman@deque.com
-  CB Averitt
 - Principal Accessibility Consultant
 - Twitter: @dive4cb
 - Email: cb.averitt@deque.com

Connect with us



[@dequesystems](https://twitter.com/dequesystems)



[dequelabs](https://github.com/dequelabs)



[deque-systems-inc](https://www.linkedin.com/company/deque-systems-inc)



[dequesystemsinc](https://www.youtube.com/dequesystemsinc)

