



Adobe

Case Study: CloudUI

Michael Jordan | Accessibility Engineer | @majornista



- Adobe has a cross-product accessibility team
- Supports accessibility in
 - Product requirements
 - Product development
 - Standards committees
 - Relationships with assistive technology vendors
 - Information for end users and authors

ADOBE ACCESSIBILITY



What's this all about?

CloudUI

A system of styles, components, and UI patterns for Adobe Cloud products





Adobe® Marketing Cloud

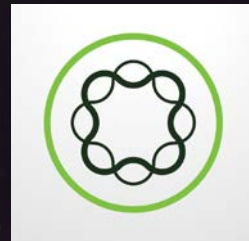
CORE SERVICES



ANALYTICS



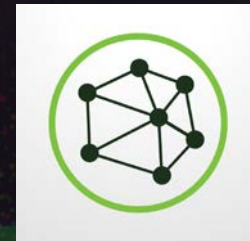
CAMPAIGN



EXPERIENCE
MANAGER



MEDIA
OPTIMIZER



SOCIAL



TARGET



PRIMETIME



Key benefits

- **Provides a fresh and consistent experience for Adobe's customers**
 - Touch Friendly
 - Modular
 - Easy to integrate
 - Heavily unit tested and UI tested
 - Supported by active community
 - Approved by XD
- **Accessible**

Skeptical?

Let's be honest.

We have a long way to go before we meet our accessibility goals for CoralUI or the products that use it.

But...

Engineers are thinking about, talking about, and *taking ownership of* product accessibility.





Open Development

- CoralUI's name is a metaphor for Open Development
- A growing body built, on a strong foundation, by a community of organisms.





Open Development at Adobe

- Similar to the way open source projects work
- Fosters collaboration
- Distributed teams
- Shared Technologies across teams
- Not open source



Open Development in short

- The code is discoverable and openly available, including activity stream
 - Internal git repository
- Discussions happen on an open and archived mailing list
 - Internal discussion lists
- Commits are backed by issues in an openly accessible tracker
 - JIRA
- Wikis and blogs are used for durable information
 - Wiki as opposed to email
- Meritocracy
 - Encourage contribution from outside core team

Open Development is good for accessibility

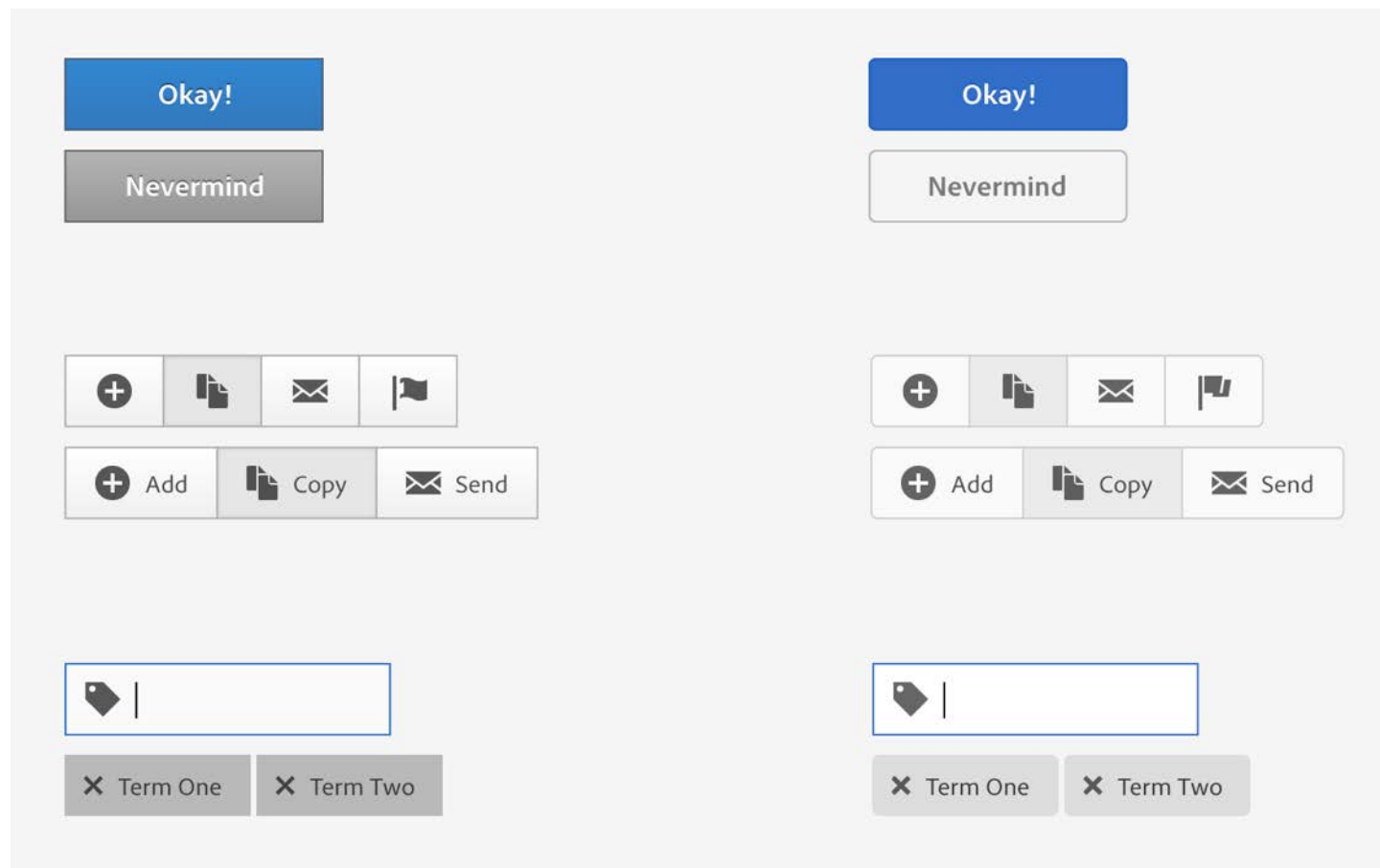
- Issues no longer need languish in a bug base unaddressed
- Accessibility Engineers must earn the title
 1. Fork the code
 2. Fix the issue
 3. Make a pull request
 4. Discuss in the open the merits of a change
 5. Improve fix based on feedback from code review
 6. Merge change
 7. Close the issue
 8. Release

Open development raises accessibility awareness

- Discussion of accessibility issues takes place on open bug tracker and list
- Many more developers become aware of an issue
 - versus reaching out to an individual developer of a feature or control
- Awareness results in more, and better, questions for accessibility team

This has lead to...

- Better color contrast
 - If you can convince a designer to use a different color blue, you must be doing something right.



- WAI-ARIA design pattern implementations for CoralUI components including:
 - Accordion, Alert, Autocomplete, Button, Button Group, Character Count, Checkbox, Dialog, NumberInput, Popover, Progress, Radio Button, Select, SelectList, Selector, Slider, Switch, TabPanel, TagList, TextField, Tooltip, Wait, NavigationView

Definition of Done

- Work is tracked in JIRA
- API changes are approved by the community
- Code is complete
 - follows coding conventions
 - tests added or updated
 - **meets accessibility standards**
 - updated to current visual specification
 - cleaned up or refactored where needed
 - no TODOs or other cruft left in code
- All tests passing
- Verified working on all supported browsers
 - Chrome, FF, Safari on OSX
 - IE9+, FF, Chrome on Windows
 - iOS7+
 - Android 3+
- Peer reviewed via pull request
 - pull request announced on the CoralUI mail list
 - adequate time for feedback allowed
 - feedback addressed, code updated as needed
- Documentation reviewed and updated

What “meets accessibility standards” means for Coral UI

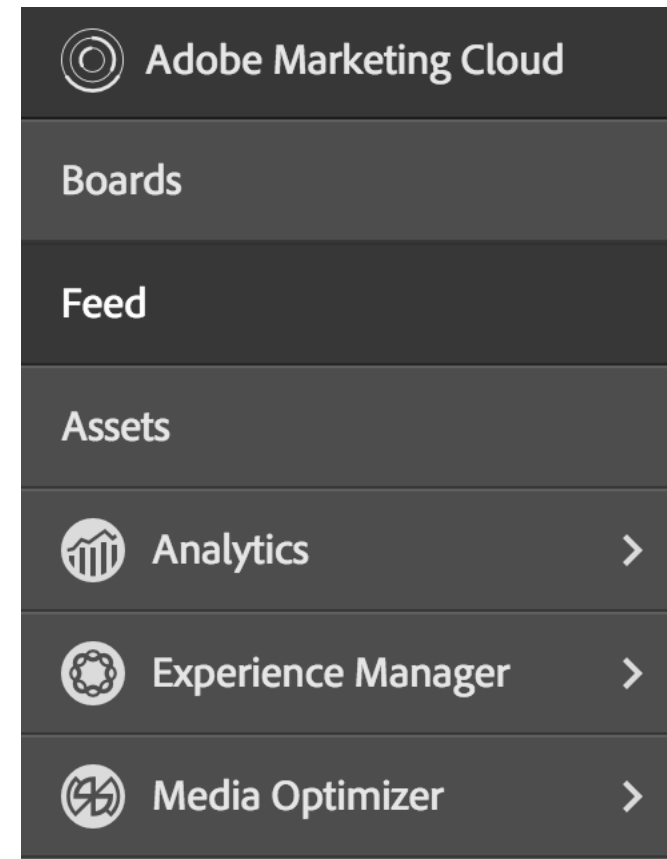
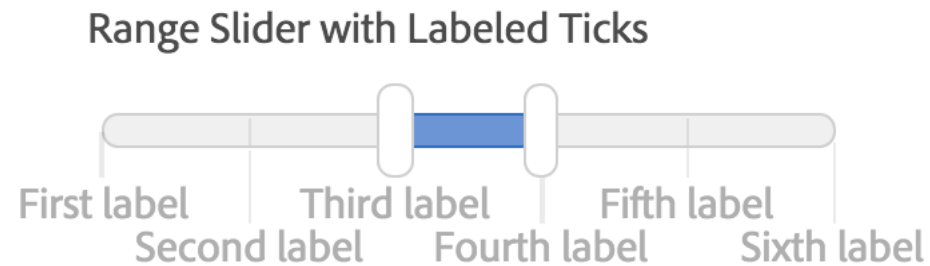
- Developer’s responsibility:
 - Working implementation of appropriate WAI-ARIA design pattern for control
 - Unit tests to verify accessibility implementation
 - Control is expected to work with assistive technology:
 - Windows in IE and Firefox with JAWS or NVDA
 - OSX with Safari and VoiceOver
- Accessibility teams responsibility:
 - Peer review
 - Evaluate and correct behavior across platforms and AT including:
 - iOS with VoiceOver
 - Android with Talkback
 - Windows 8 with Narrator
 - Chrome with ChromeVox

Why division of labor?

- All developers can't be expected to be experts at testing with AT
- For most contributors, work on Coral UI secondary to work on their core products
- Accessibility team has "skin in the game"
 - Keeps discussion of accessibility open and active on the lists
 - Testing, fixing inconsistencies, and submitting pull requests for peer review and acceptance adds to our institutional knowledge

WAI-ARIA design patterns with mobile screen readers

- Design patterns predate touch screen readers on mobile
- Working implementation on the desktop may not work on mobile device



Evolution: Web Components

- Coming in next version of Coral UI
- Components will be built using custom elements
 - Part of web components standard
 - Components will have their own HTML tags
 - Abstracts implementation details away from developer using the component
- Makes accessibility easier for a developer using Coral UI to implement in his/her application



How?

- In the past, usage of Coral UI was prescriptive
- Documentation provides a code sample
- Example:

```
<div class="coral-Slider coral-Slider--ticked coral-Slider--filled" data-init="labeled-slider" data-alternating="true">
  <fieldset>
    <legend>Range Slider with Labeled Ticks</legend>
    <label>Minimum <input type="range" value="14" min="10" max="20" step="2"></label>
    <label>Maximum <input type="range" value="16" min="10" max="20" step="2"></label>
  </fieldset>
  <ul class="coral-Slider-tickLabels">
    <li>First label</li>
    <li>Second label</li>
    <li>Third label</li>
    <li>Fourth label</li>
    <li>Fifth label</li>
    <li>Sixth label</li>
  </ul>
</div>
```


With Web Components

- Developer need not know the inner markup used to make component accessible
- Just needs to provide a label using `labelledby` attribute
- Example:

```
<label id="label-range-1">  
  Range Slider with Labeled Ticks<br>  
</label>  
<coral-slider ticks filled tooltips alternatingTickLabels range values="[14,16]" min="10" max="20" step="2" labelledby="label-range-1">  
  <coral-slider-tick>First label</coral-slider-tick>  
  <coral-slider-tick>Second label</coral-slider-tick>  
  <coral-slider-tick>Third label</coral-slider-tick>  
  <coral-slider-tick>Fourth label</coral-slider-tick>  
  <coral-slider-tick>Fifth label</coral-slider-tick>  
  <coral-slider-tick>Sixth label</coral-slider-tick>  
</coral-slider>
```

Thanks!

- Adobe's Accessibility Resource Center
<http://www.adobe.com/accessibility>
- Adobe's Accessibility Blog
<http://blogs.adobe.com/accessibility>
- Web Components
<http://webcomponents.org/>
- Follow us on Twitter
 - [@AdobeAccess](https://twitter.com/AdobeAccess)
 - [@majornista](https://twitter.com/majornista)



Adobe