Building an Accessible Component Library

Ari Rizzitano and Mark Sadecki
CSUN Assistive Technology Conference
March 23, 2018

Introductions

Mark Sadecki
@cptvitamin

Ari Rizzitano
@arizzitano

Why Components?

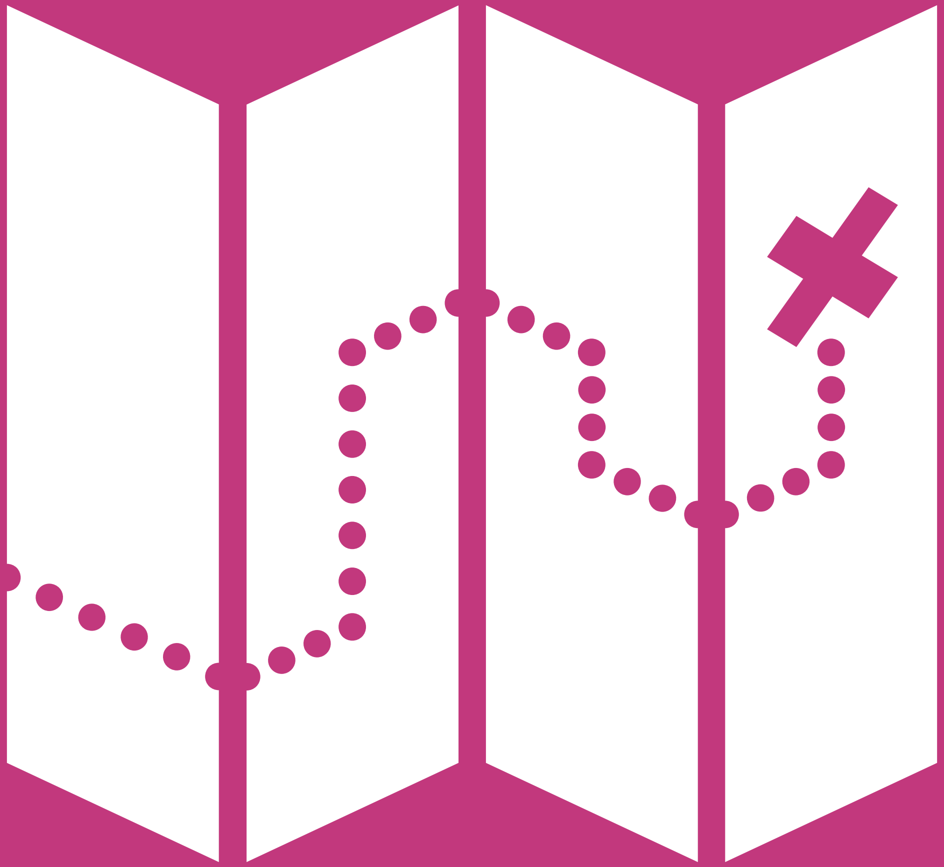
Best Practices

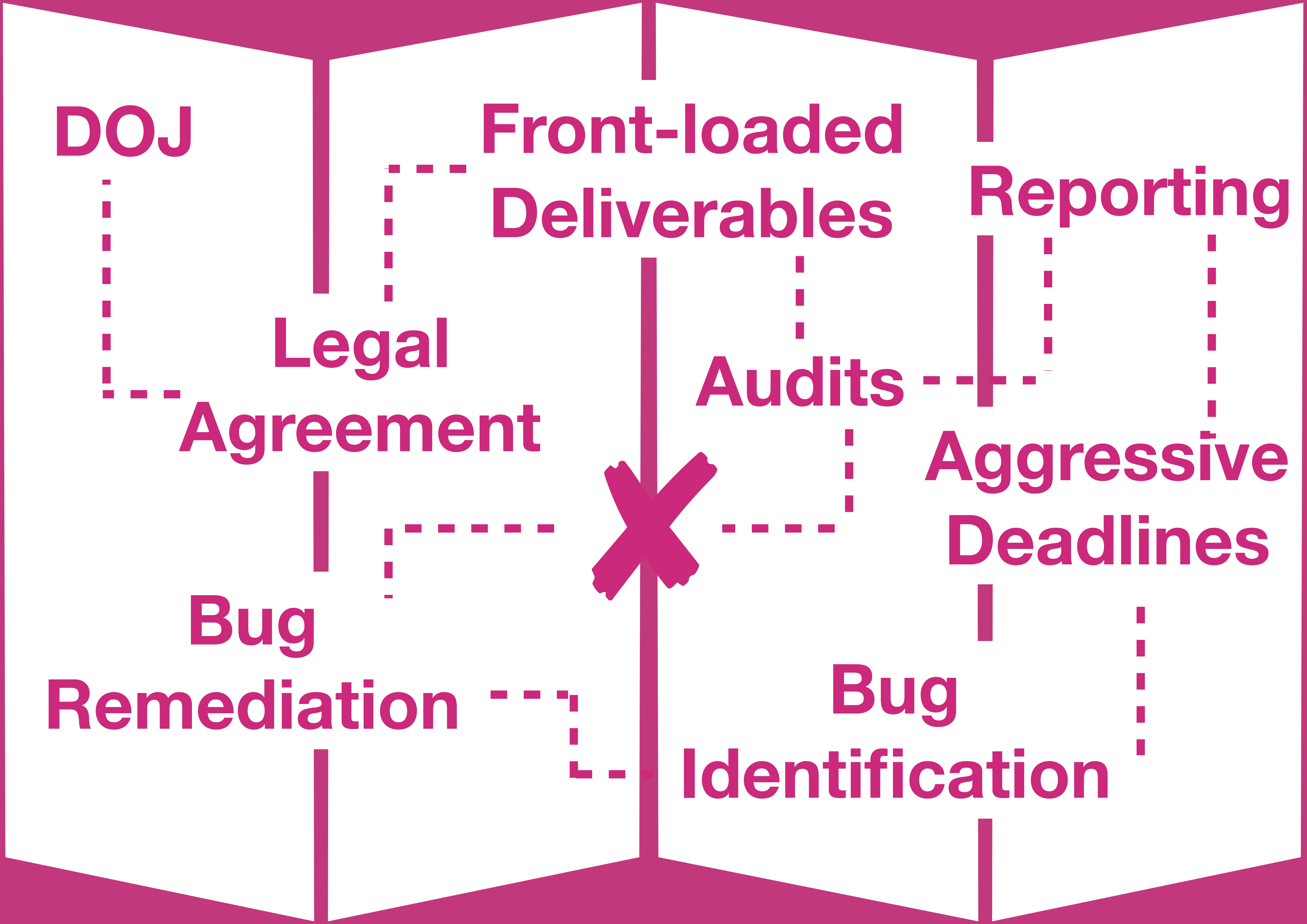
Developer Toolkit

Driving Adoption

Why Components?

Our Accessibility Journey




DOJ
Legal Agreement
Front-loaded Deliverables
Reporting
Audits
Aggressive Deadlines
Bug Remediation
Bug Identification

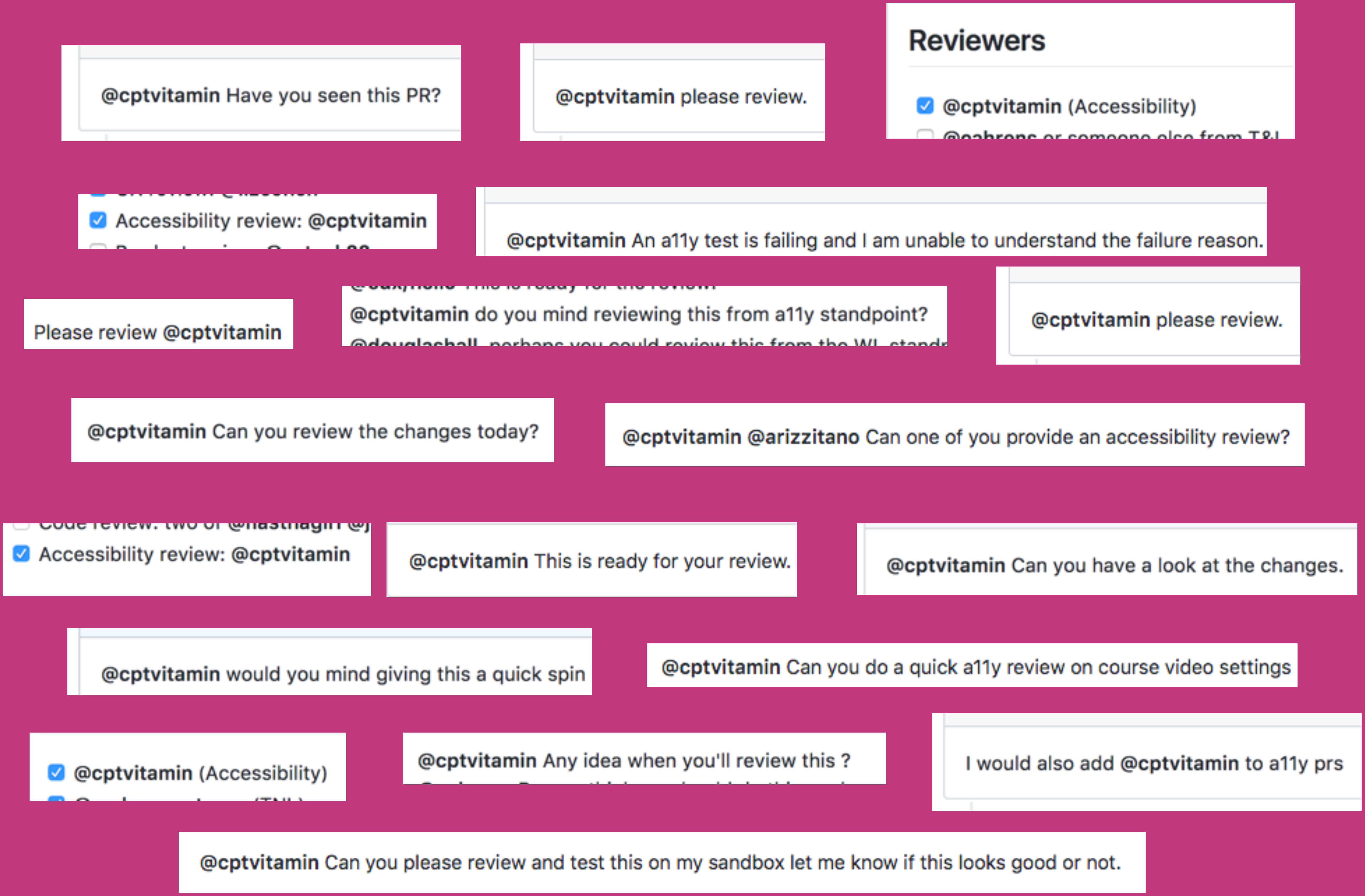
Phase 1:
Achieve Conformance
Phase 2:
Maintain Conformance

“We ship too many a11y bugs.”

“A11Y fixes are hard and take too long.”

“Developers don’t know how to write compliant code.”

“Can’t we just hire one expert to do it all?”



@cptvitamin Have you seen this PR?

@cptvitamin please review.

Reviewers

☑ @cptvitamin (Accessibility)

☑ Accessibility review: @cptvitamin

@cptvitamin An a11y test is failing and I am unable to understand the failure reason.

Please review @cptvitamin

@cptvitamin do you mind reviewing this from a11y standpoint?

@cptvitamin please review.

@cptvitamin Can you review the changes today?

@cptvitamin @arizzitano Can one of you provide an accessibility review?

☑ Accessibility review: @cptvitamin

@cptvitamin This is ready for your review.

@cptvitamin Can you have a look at the changes.

@cptvitamin would you mind giving this a quick spin

@cptvitamin Can you do a quick a11y review on course video settings

☑ @cptvitamin (Accessibility)

@cptvitamin Any idea when you'll review this ?

I would also add @cptvitamin to a11y prs

@cptvitamin Can you please review and test this on my sandbox let me know if this looks good or not.

“Why do we have 6 different modals?”

“Wash, Rinse, Repeat”

Enter: the component library

Modularity

Modularity

- UX and brand consistency
- Faster, easier bugfixes
- No reinventing the wheel

Encapsulation

Encapsulation

- Easier to develop and test
- Consumable by multiple applications or services
- Highly dynamic functionality lives in one place

Learning

Learning

- Lower barrier to entry
- Code shared across teams
- Motivation to help other devs
- Easier to review

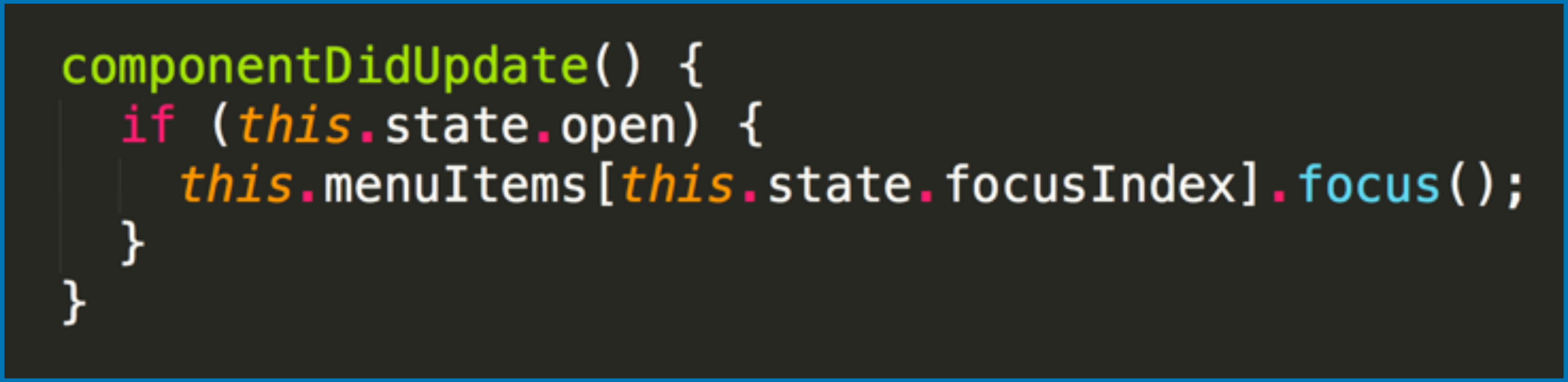
Best Practices

Genericize as much accessible functionality as possible.

A component library is a great way to do this.

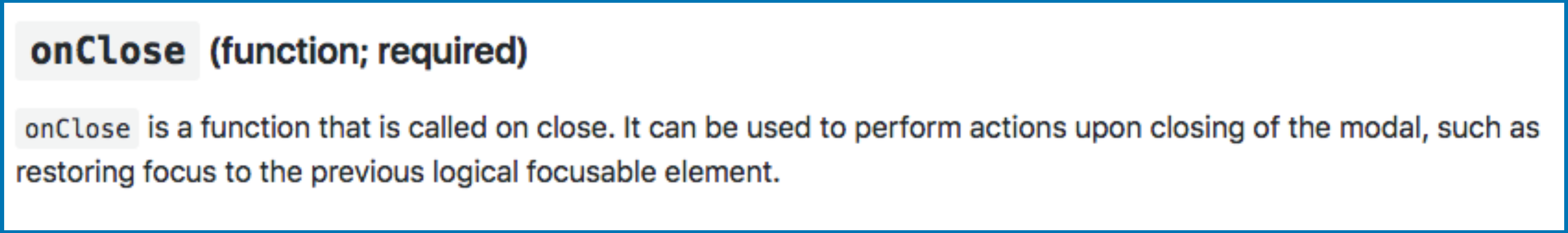
Best practice for focus is best practice for perf

Encourage focus management within components.



```
componentDidUpdate() {
  if (this.state.open) {
    this.menuItems[this.state.focusIndex].focus();
  }
}
```

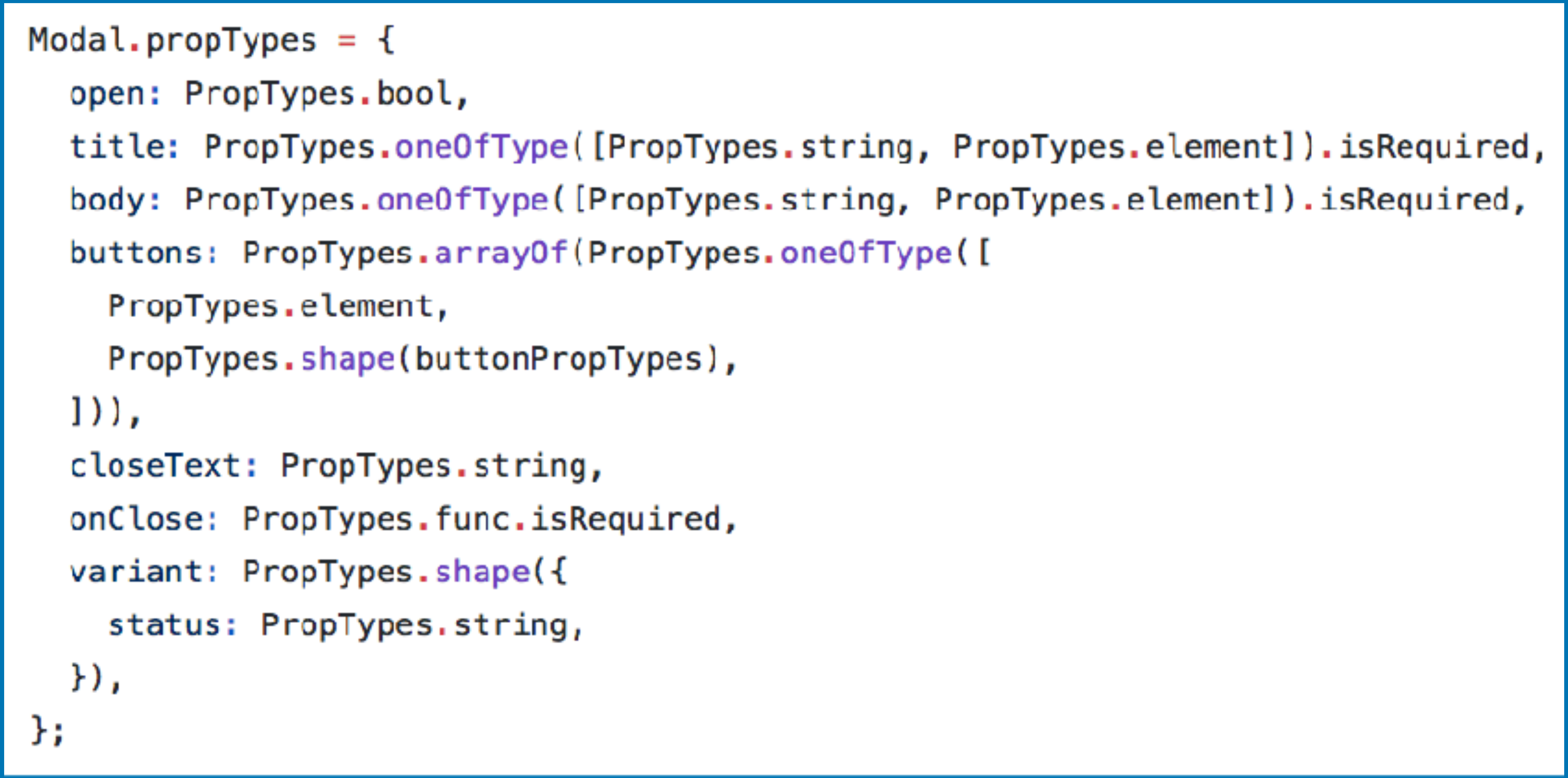
Trap focus when appropriate


`onClose` (function; required)

`onClose` is a function that is called on close. It can be used to perform actions upon closing of the modal, such as restoring focus to the previous logical focusable element.

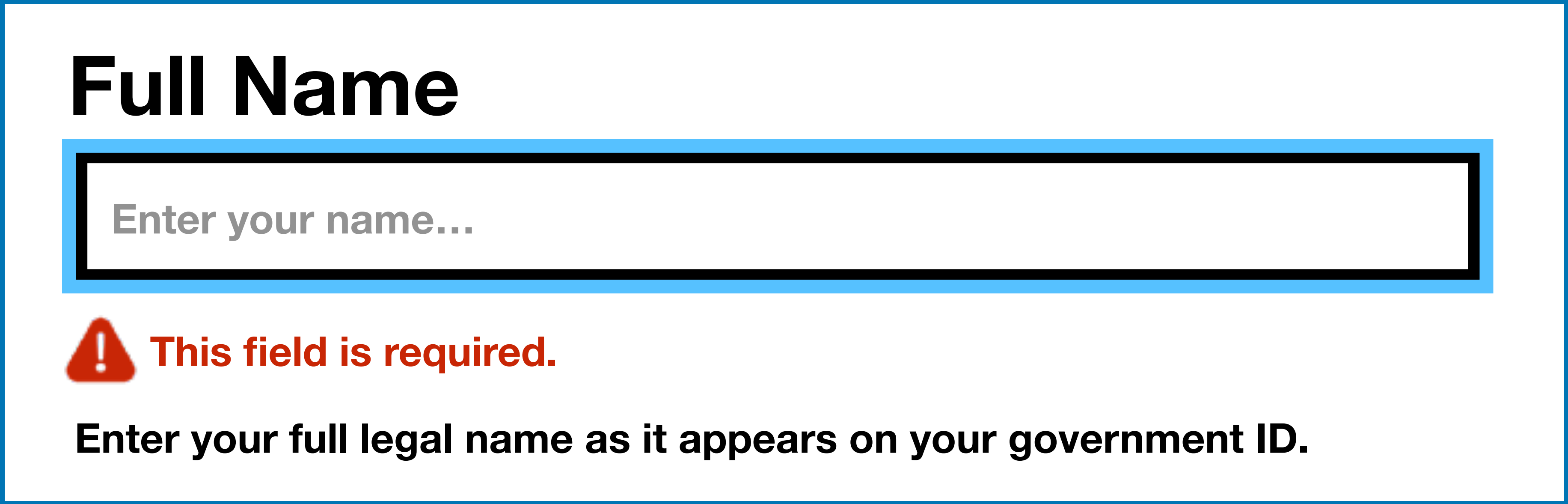
Use API to enforce best practices outside components.

Enforce best practices



```
Modal.propTypes = {
  open: PropTypes.bool,
  title: PropTypes.oneOfType([PropTypes.string, PropTypes.element]).isRequired,
  body: PropTypes.oneOfType([PropTypes.string, PropTypes.element]).isRequired,
  buttons: PropTypes.arrayOf(PropTypes.oneOfType([
    PropTypes.element,
    PropTypes.shape(buttonPropTypes),
  ])),
  closeText: PropTypes.string,
  onClose: PropTypes.func.isRequired,
  variant: PropTypes.shape({
    status: PropTypes.string,
  }),
};
```

Required Props

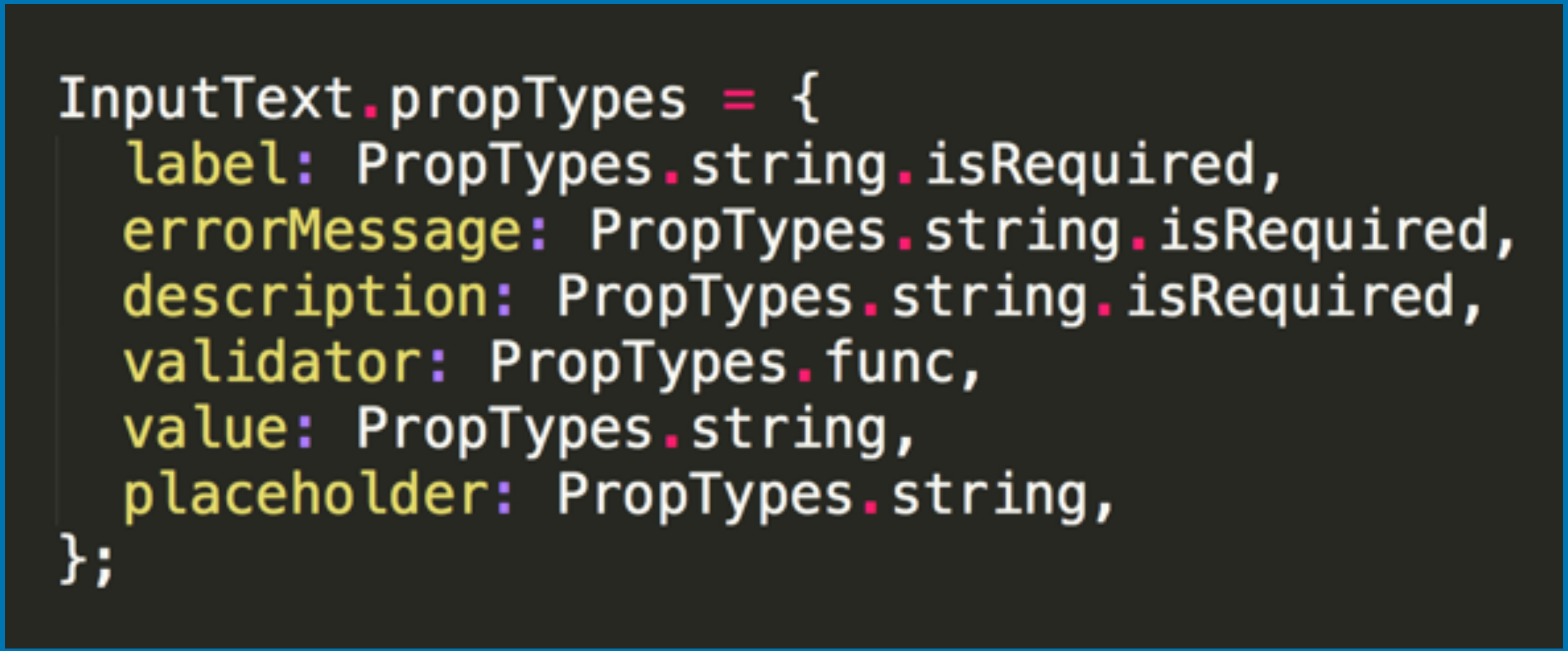


Full Name

Enter your name...

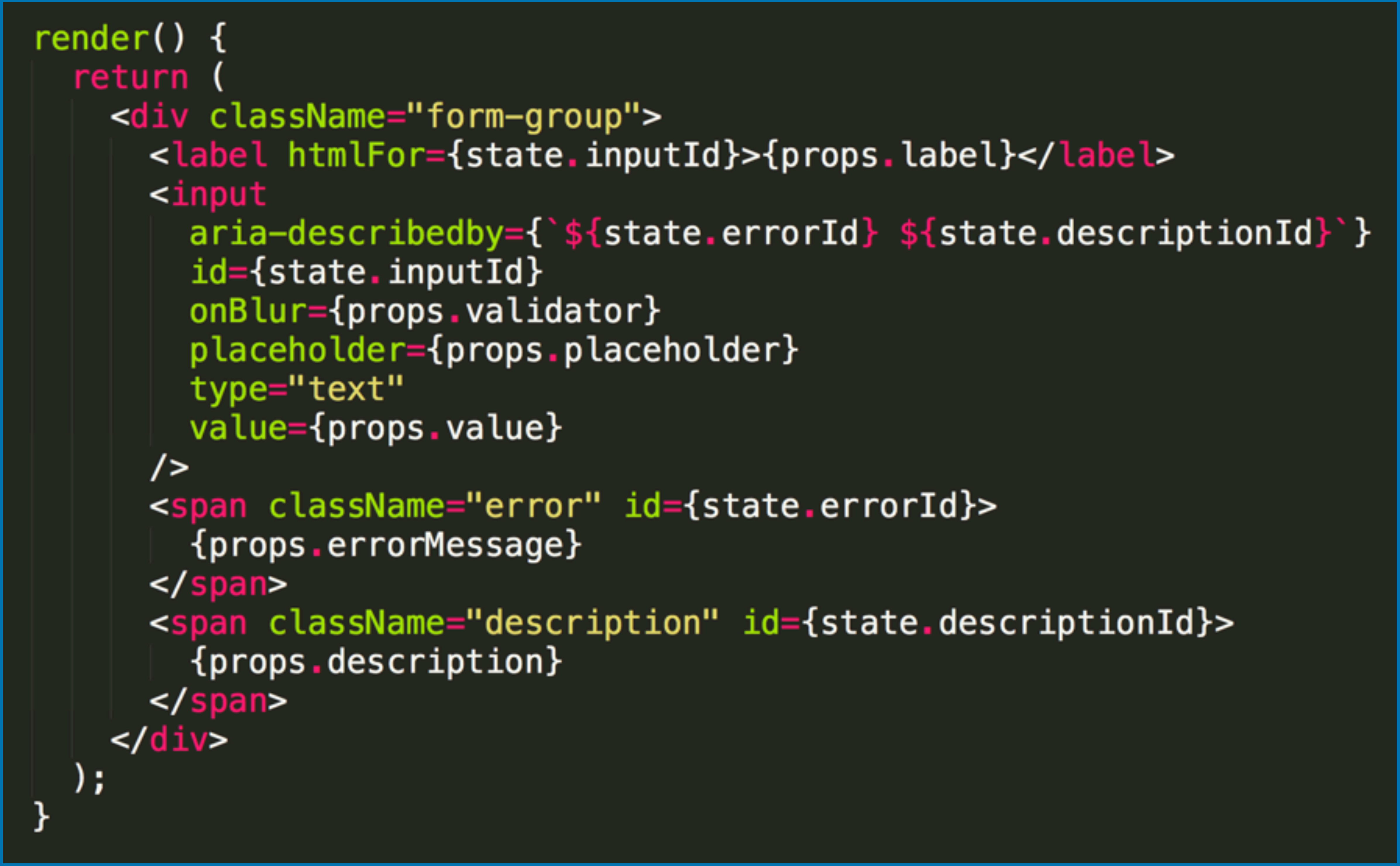
This field is required.

Enter your full legal name as it appears on your government ID.



```
InputText.propTypes = {
  label: PropTypes.string.isRequired,
  errorMessage: PropTypes.string.isRequired,
  description: PropTypes.string.isRequired,
  validator: PropTypes.func,
  value: PropTypes.string,
  placeholder: PropTypes.string,
};
```

Enforce best practices



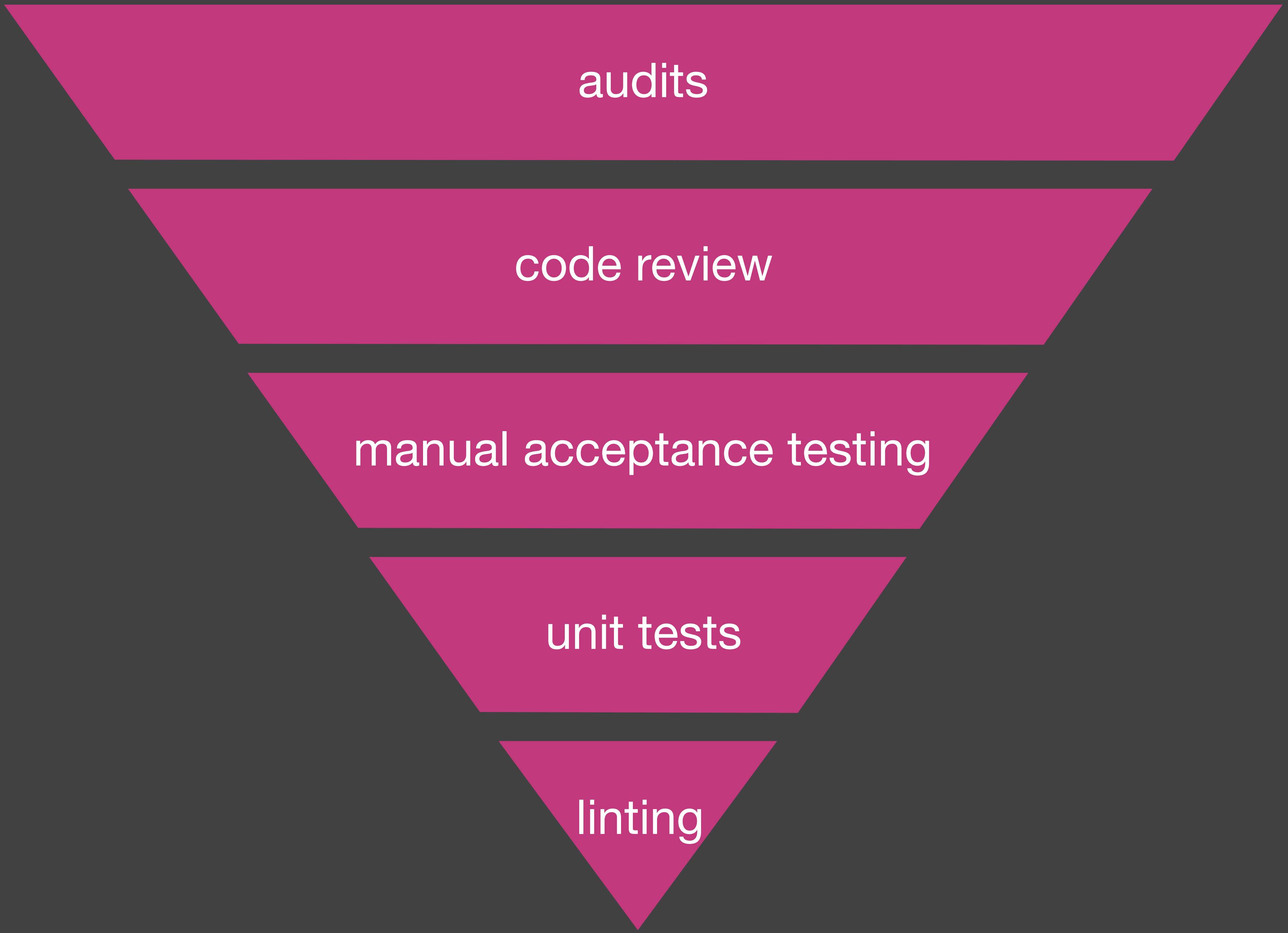
```
render() {
  return (
    <div className="form-group">
      <label htmlFor={state.inputId}>{props.label}</label>
      <input
        aria-describedby={`${state.errorId} ${state.descriptionId}`}
        id={state.inputId}
        onBlur={props.validator}
        placeholder={props.placeholder}
        type="text"
        value={props.value}
      />
      <span className="error" id={state.errorId}>
        {props.errorMessage}
      </span>
      <span className="description" id={state.descriptionId}>
        {props.description}
      </span>
    </div>
  );
}
```

Reuse behavior & markup

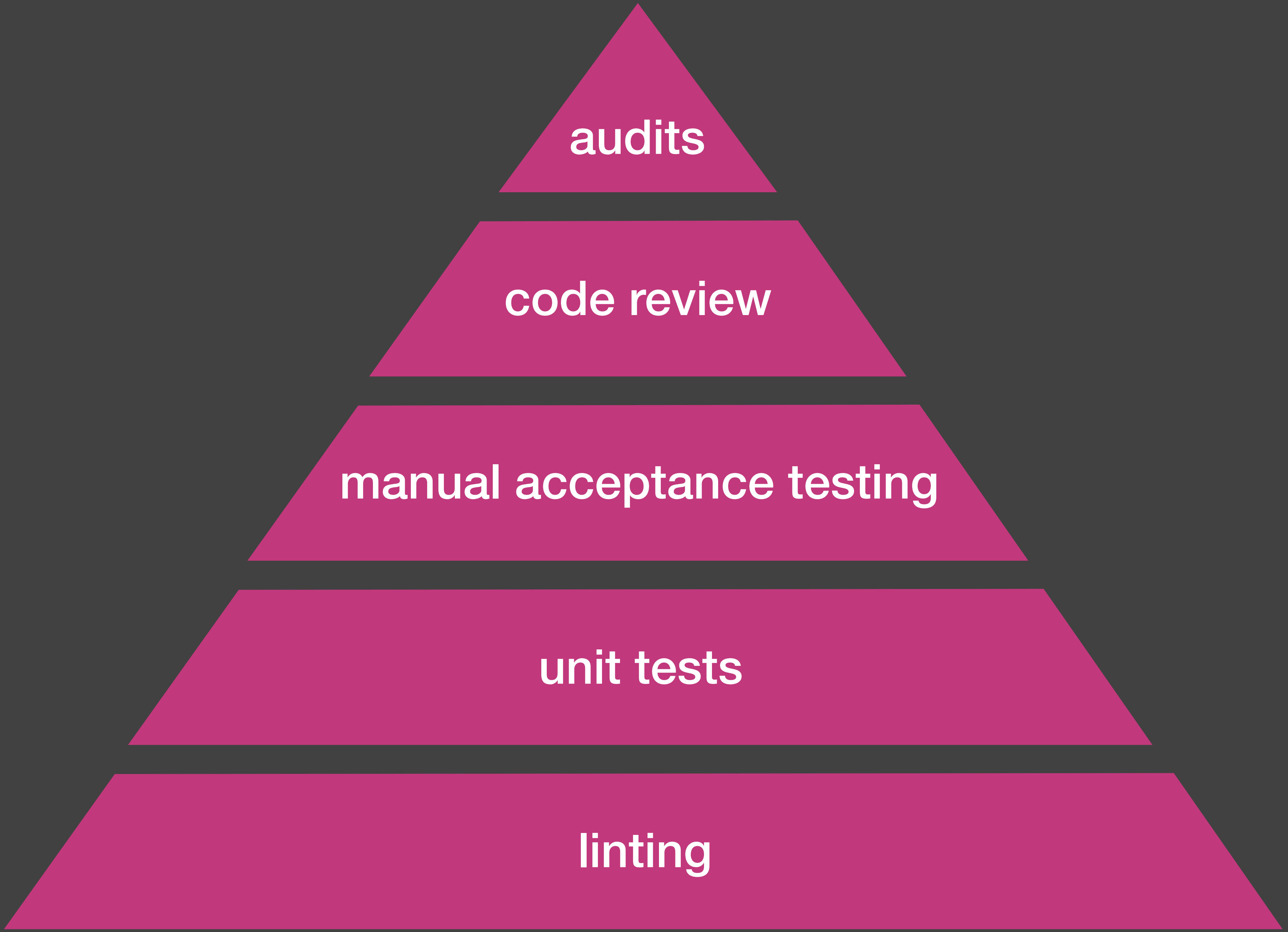
e.g. for different input types

Developer Toolkit

Accessibility Testing



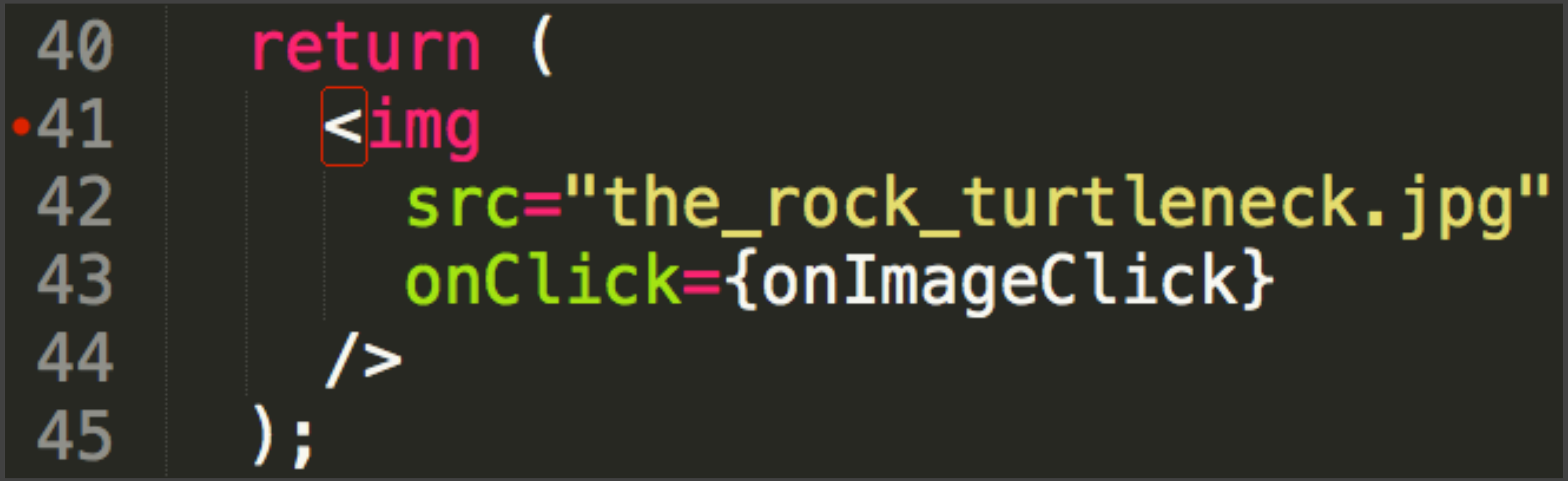
Accessibility Testing



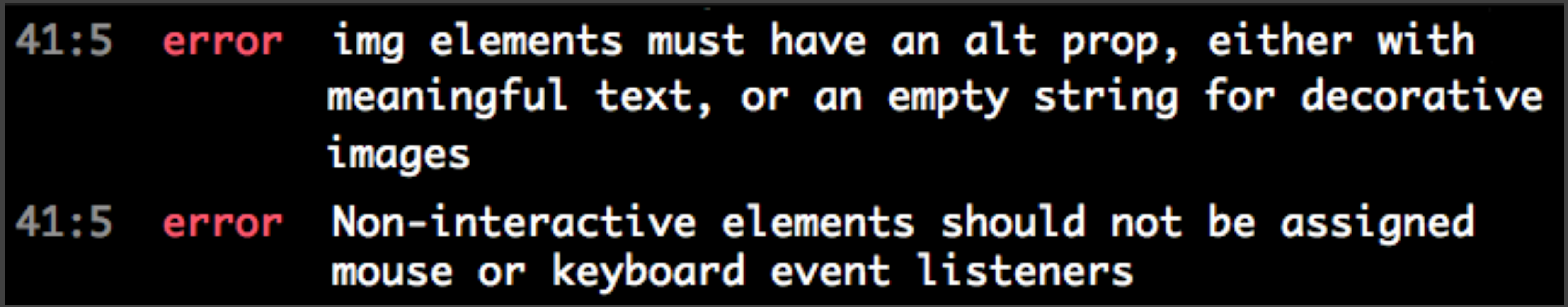
Accessibility linting

Immediate linter feedback helps devs internalize best practices

eslint-plugin-jsx-a11y



```
40    return (
41      <img
42        src="the_rock_turtleneck.jpg"
43        onClick={onImageClick}
44      />
45    );
```



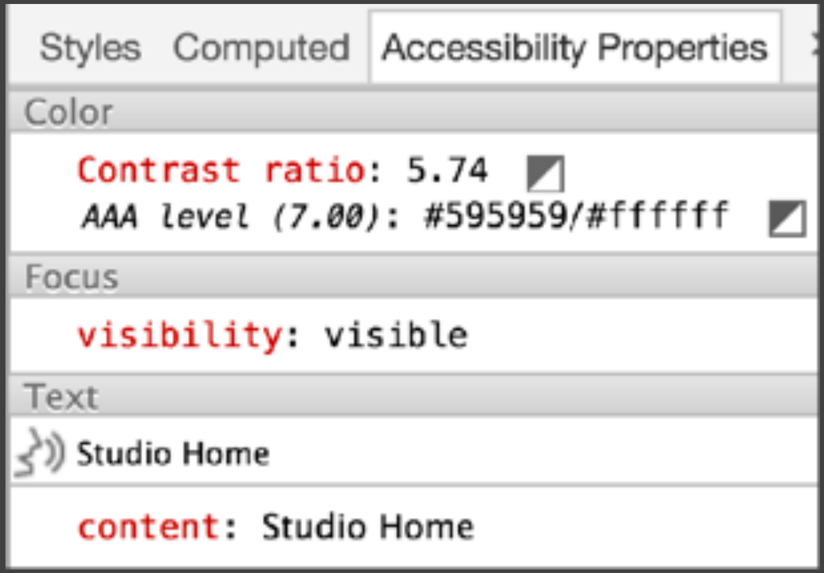
```
41:5   error   img elements must have an alt prop, either with
               meaningful text, or an empty string for decorative
               images
41:5   error   Non-interactive elements should not be assigned
               mouse or keyboard event listeners
```

Solid unit tests

(this one is just common sense)

Dogfood!

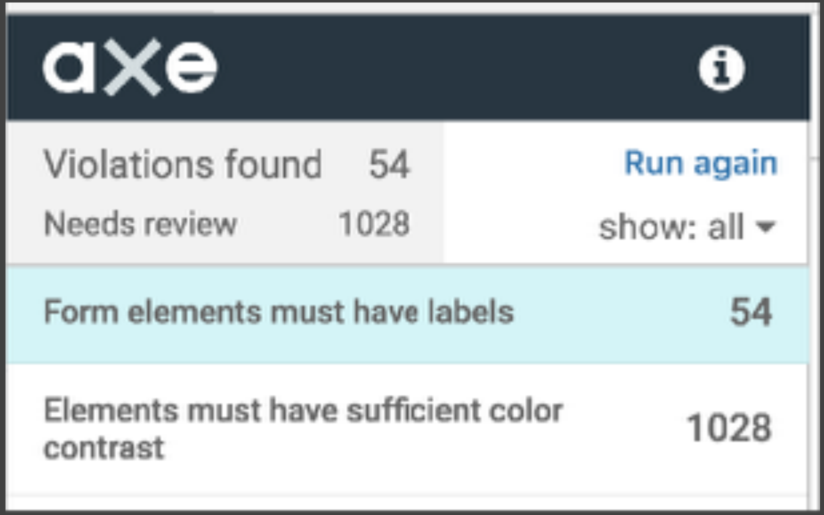
Test and interact with your component library the way people with disabilities might use it.



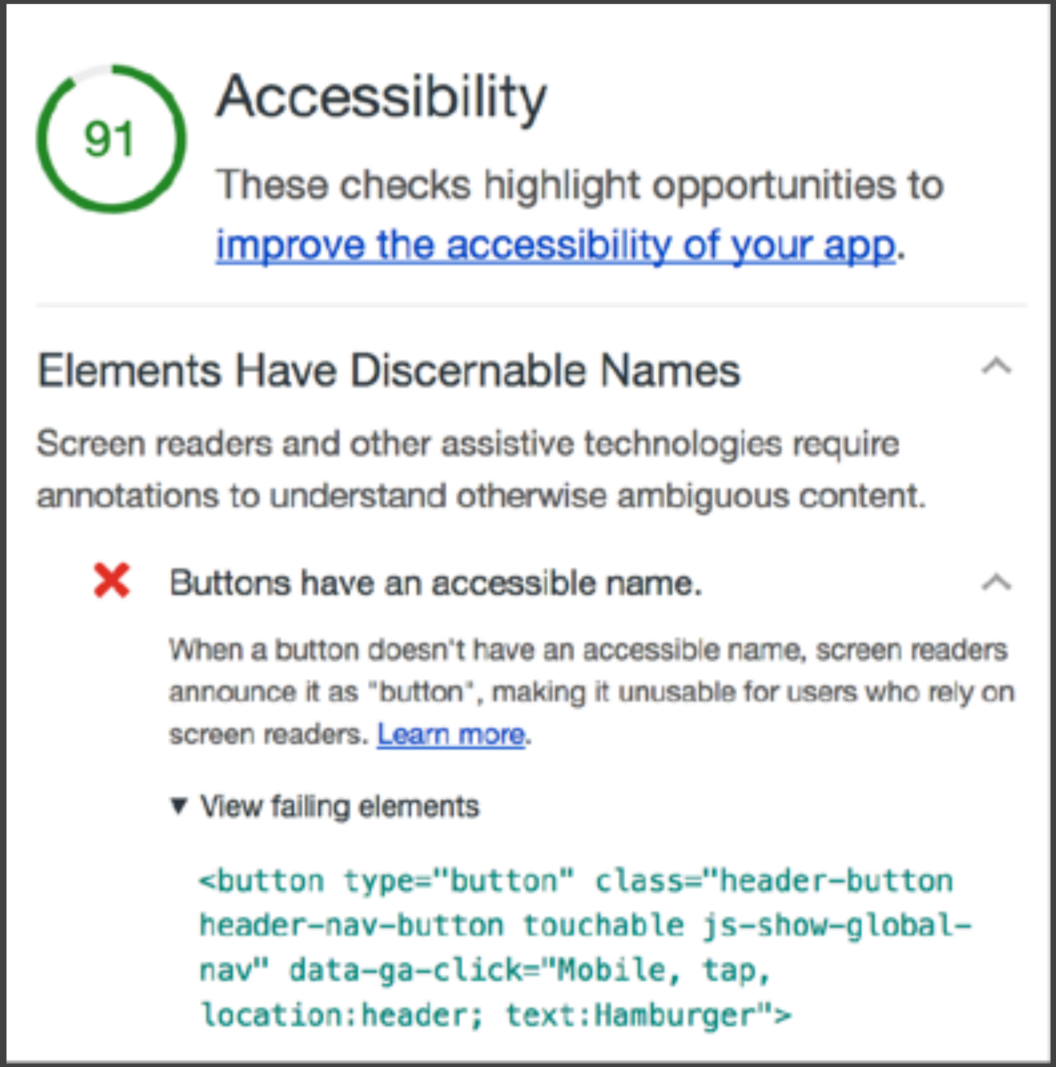
```
Contrast ratio: 5.74
AAA level (7.00): #595959/#ffffff
visibility: visible
```

Studio Home

```
content: Studio Home
```



Violations found	54
Needs review	1028
Form elements must have labels	54
Elements must have sufficient color contrast	1028


91

Accessibility

These checks highlight opportunities to improve the accessibility of your app.

Elements Have Discernable Names

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

Buttons have an accessible name.

When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. Learn more.

▾ View failing elements

```
<button type="button" class="header-button
header-nav-button touchable js-show-global-
nav" data-ga-click="Mobile, tap,
location:header; text:Hamburger">
```

Browser extensions

Explore the accessibility tree, check color contrast, audit your code while working on it

Driving Adoption

Many devs view a11y as a chore. How do we fix this?

Paragon

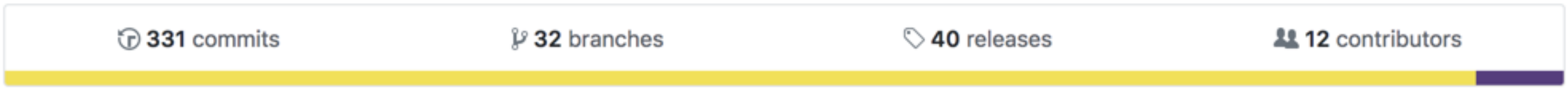
build passing · coverage 100% · Greenkeeper enabled · npm v2.4.1 · downloads 202k · license MIT · 📦🚀 semantic-release

- edX’s (work-in-progress) component library
- Implemented with React and Bootstrap, open source and released via npm
- 15 components (and growing). a11y team built the first 5 components; ***developers did the rest***
- https://github.com/edx/paragon

Make it easy

- Provide a simple development environment/playground
- Automate build & release
- Maintain an issue board
- Supportive code reviews
- ***Extensive documentation & demos***

Paragon



331 commits | 32 branches | 40 releases | 12 contributors

- Sandbox environment doubles as doc site (https://storybook.js.org)
- 100% test coverage
- Release process is fully automated, so devs can focus on what they do best

Make it fun

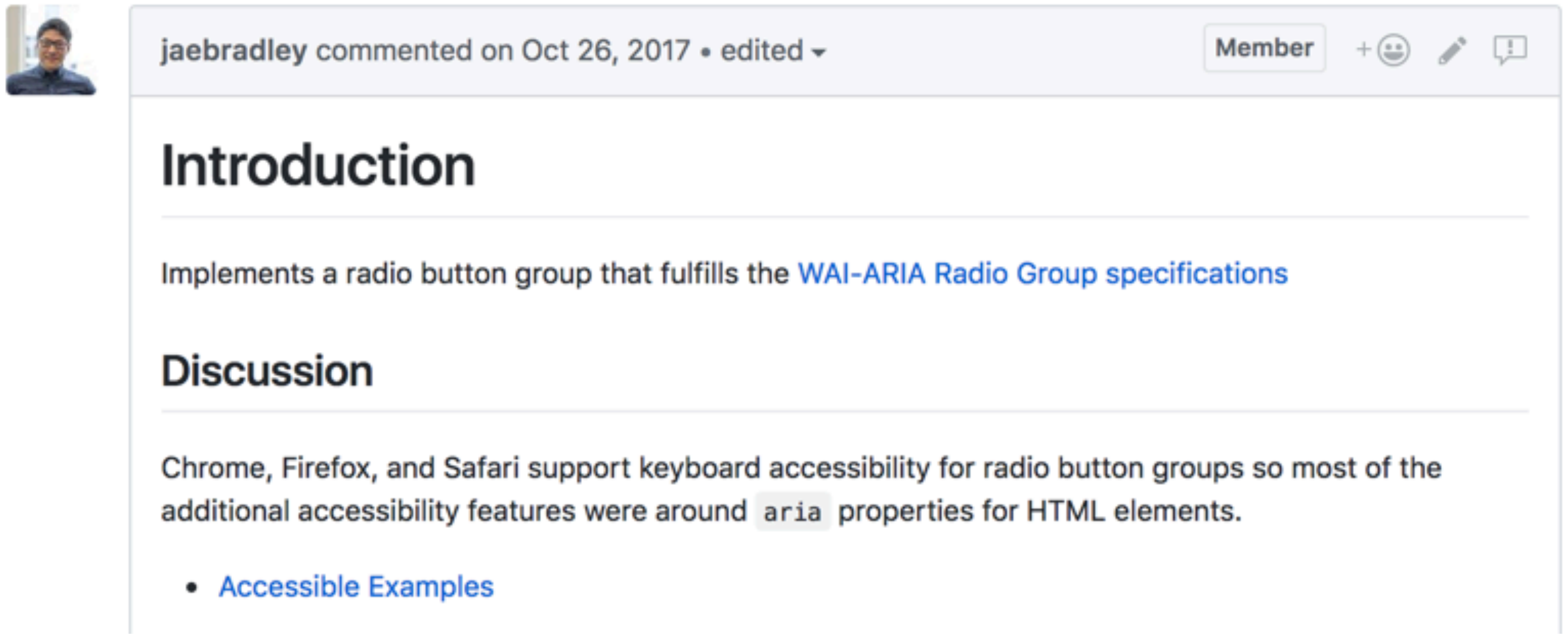
- Reward first-time contributors
- Pair program as necessary
- Encourage collaboration
- Showcase contributions
- “Accessible code is good code”

Party Time

Tricks & treats

The benefits

Proactive



jaebradley commented on Oct 26, 2017 • edited

Member

Introduction

Implements a radio button group that fulfills the WAI-ARIA Radio Group specifications

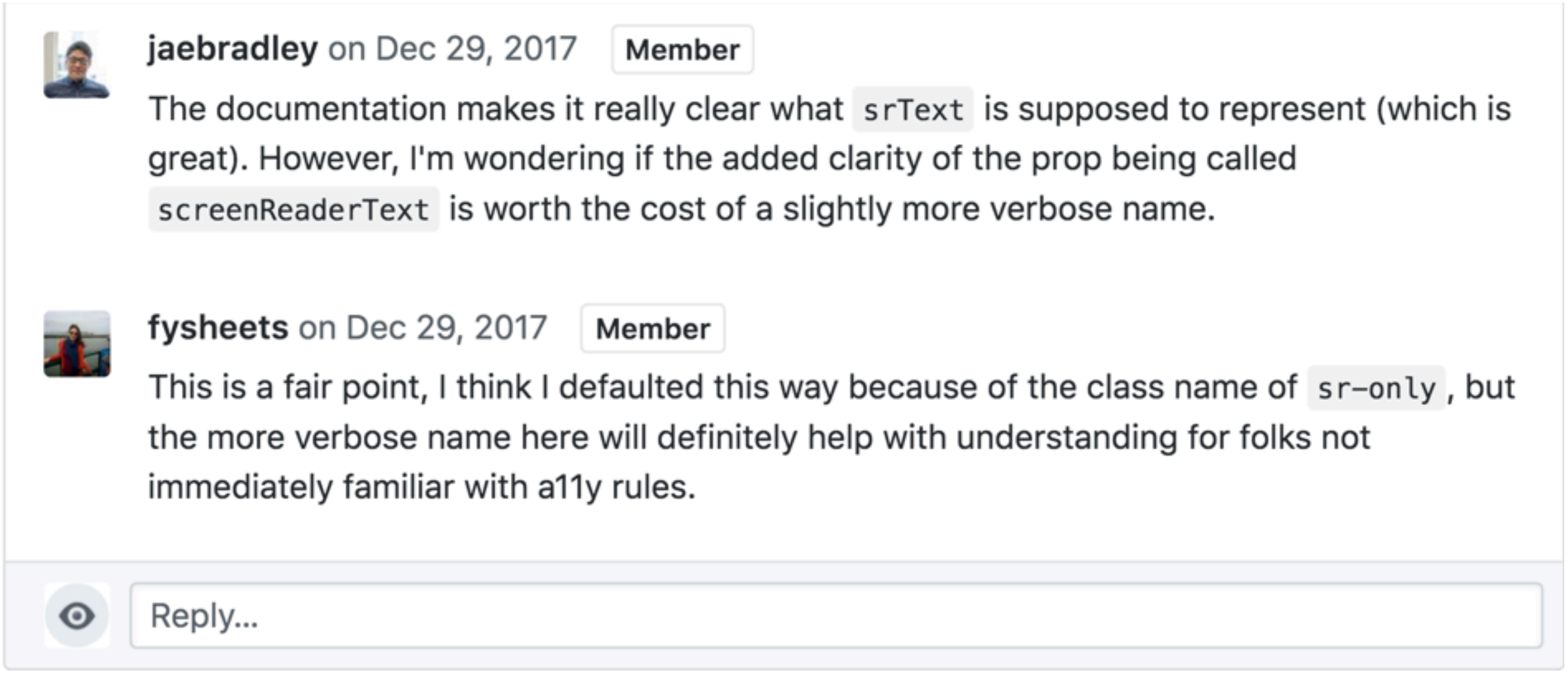
Discussion

Chrome, Firefox, and Safari support keyboard accessibility for radio button groups so most of the additional accessibility features were around `aria` properties for HTML elements.

- Accessible Examples

Developers go straight to the source.

Collaborative



jaebradley on Dec 29, 2017 Member

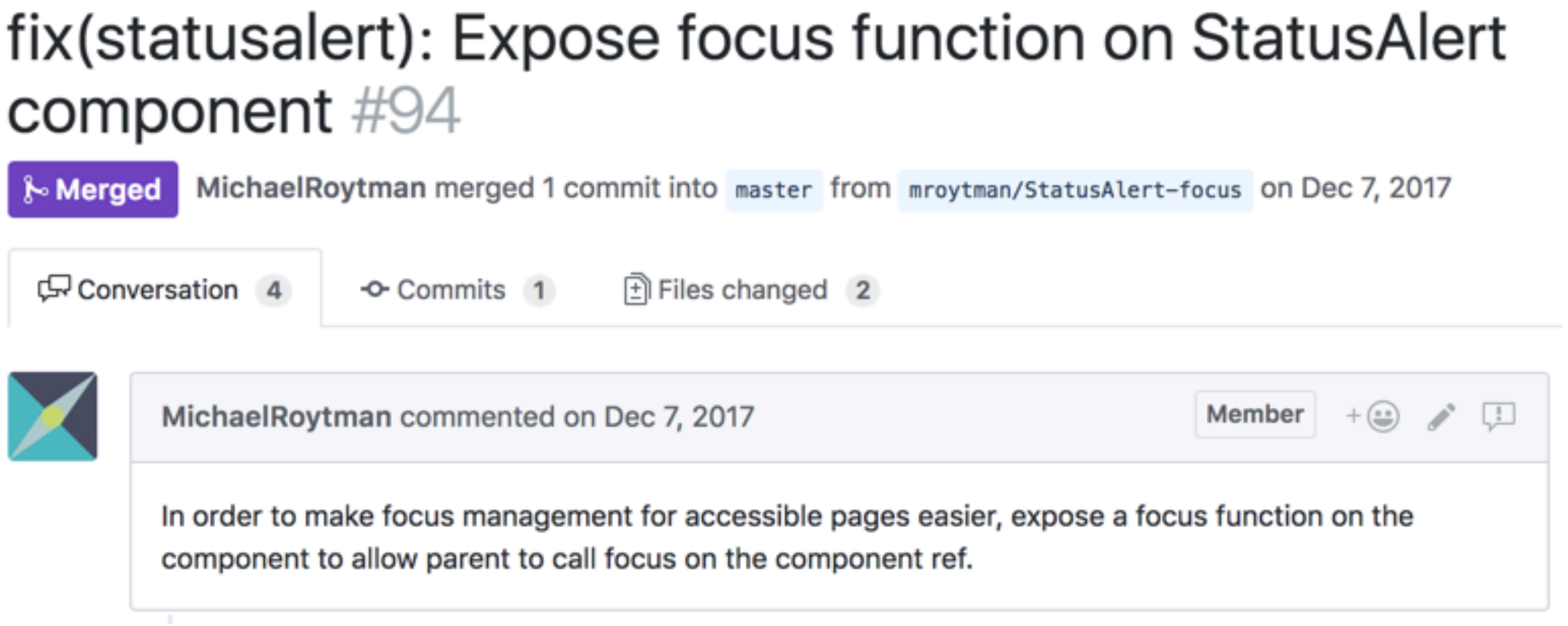
The documentation makes it really clear what `srText` is supposed to represent (which is great). However, I'm wondering if the added clarity of the prop being called `screenReaderText` is worth the cost of a slightly more verbose name.

fysheets on Dec 29, 2017 Member

This is a fair point, I think I defaulted this way because of the class name of `sr-only`, but the more verbose name here will definitely help with understanding for folks not immediately familiar with a11y rules.

Developers optimize for clarity and strive to help colleagues learn.

Conscious



fix(statusalert): Expose focus function on StatusAlert component #94

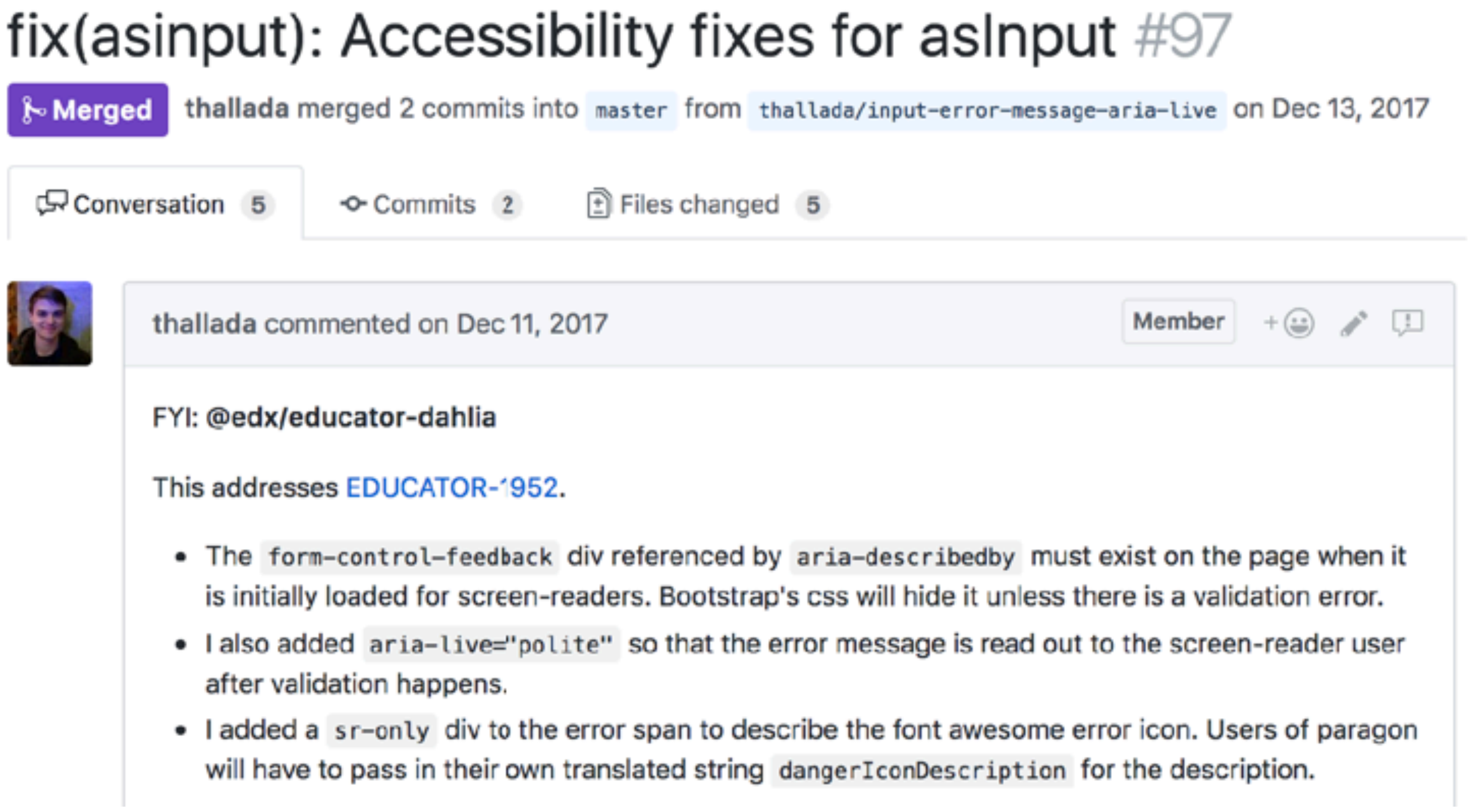
Merged MichaelRoytman merged 1 commit into `master` from `mroytman/StatusAlert-focus` on Dec 7, 2017

MichaelRoytman commented on Dec 7, 2017 Member

In order to make focus management for accessible pages easier, expose a focus function on the component to allow parent to call focus on the component ref.

a11y use cases become second nature.

Thorough



fix(asinput): Accessibility fixes for asInput #97

Merged thallada merged 2 commits into `master` from `thallada/input-error-message-aria-live` on Dec 13, 2017

thallada commented on Dec 11, 2017

FYI: **@edx/educator-dahlia**

This addresses EDUCATOR-1952.

- The `form-control-feedback` div referenced by `aria-describedby` must exist on the page when it is initially loaded for screen-readers. Bootstrap's css will hide it unless there is a validation error.
- I also added `aria-live="polite"` so that the error message is read out to the screen-reader user after validation happens.
- I added a `sr-only` div to the error span to describe the font awesome error icon. Users of paragon will have to pass in their own translated string `dangerIconDescription` for the description.

Different integration environments encourage feature completeness.

Be soft on people, hard on code.

Web accessibility is built on best practices that benefit *everyone*

Thank you.

@arizzitano
@cptvitamin