Building Accessible React apps

Vikas Parashar
Frontend Engineer, HackerRank

Web Accessibility

“The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.”

- Tim Berners-Lee, W3C Director and inventor of the World Wide Web

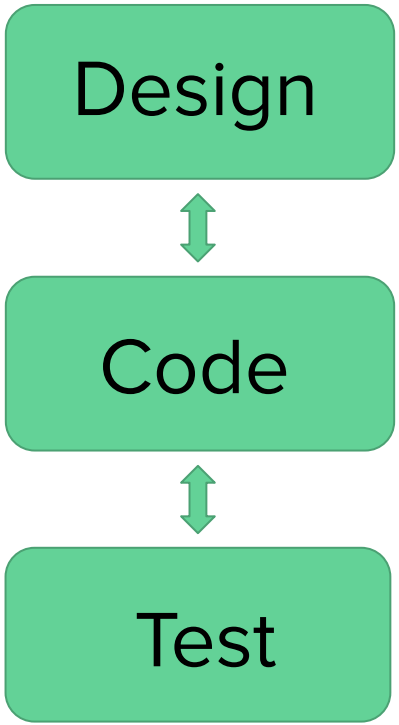
Challenge

Make HackerRank's products accessible to as many people as possible

Why?

- Improves User Experience
- Benefits SEO
- Expand user base
- More enterprise customers
- Legal reasons

How we are doing it?

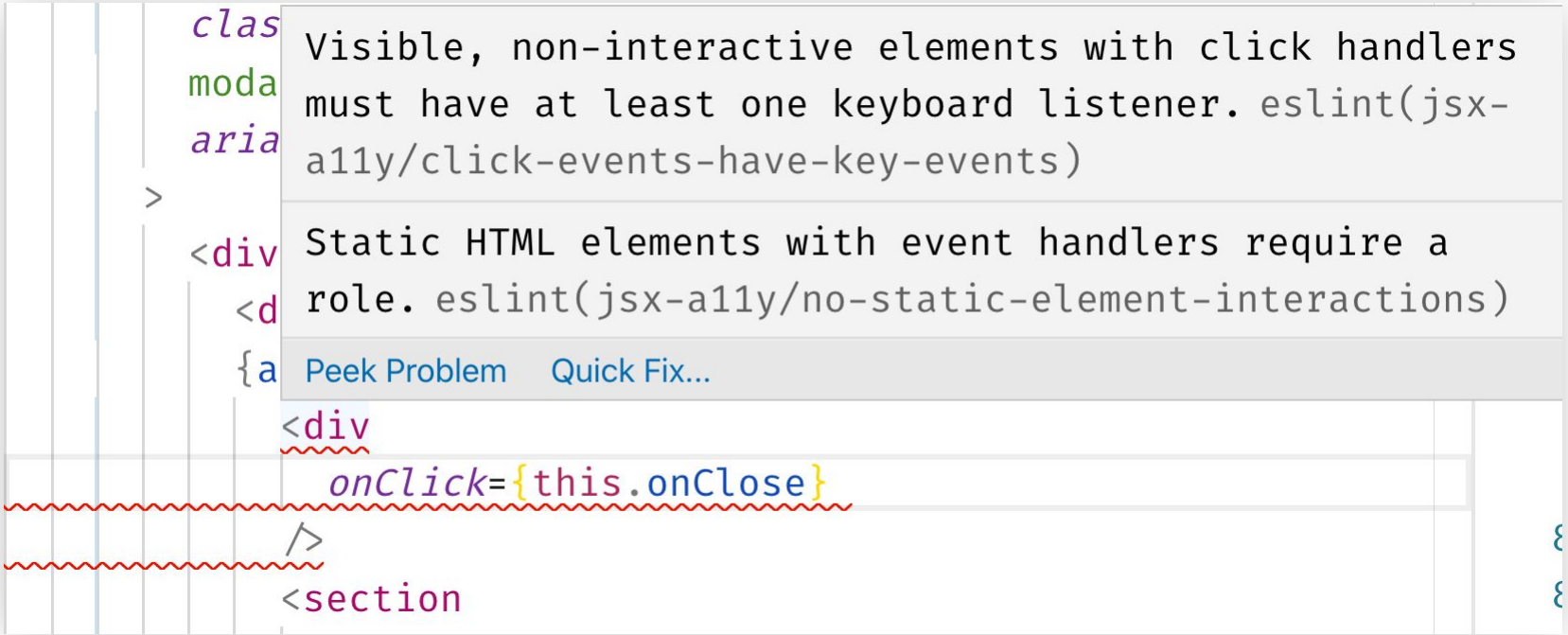

Design
Code
Test

Color contrast checker

Color contrast checker

Or try http://colorsafe.co/

Linter for accessibility rules



Dev tools logger

```
//webpack config
plugins: [
  new webpack.DefinePlugin({
    ...
    SHOW_A11Y_LOGGER: JSON.stringify(
      ENV !== 'production' &&
      process.env.ENABLE_A11Y === '1'
      ),
  }),
   ...
],
```

Dev tools logger

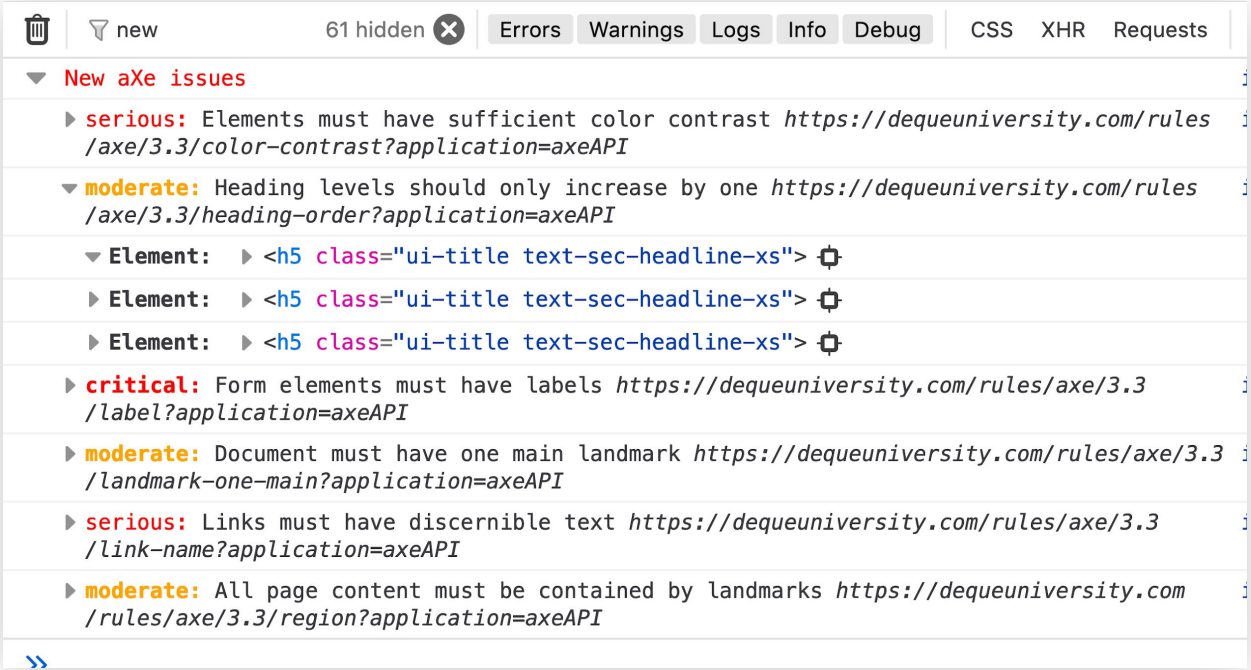
```
//index.js/root component
componentDidUpdate( ... ) {
  if(SHOW_A11Y_LOGGER) {
    axe(React, ReactDOM, 1000); //react-axe
  }
}
```

Dev tools logger

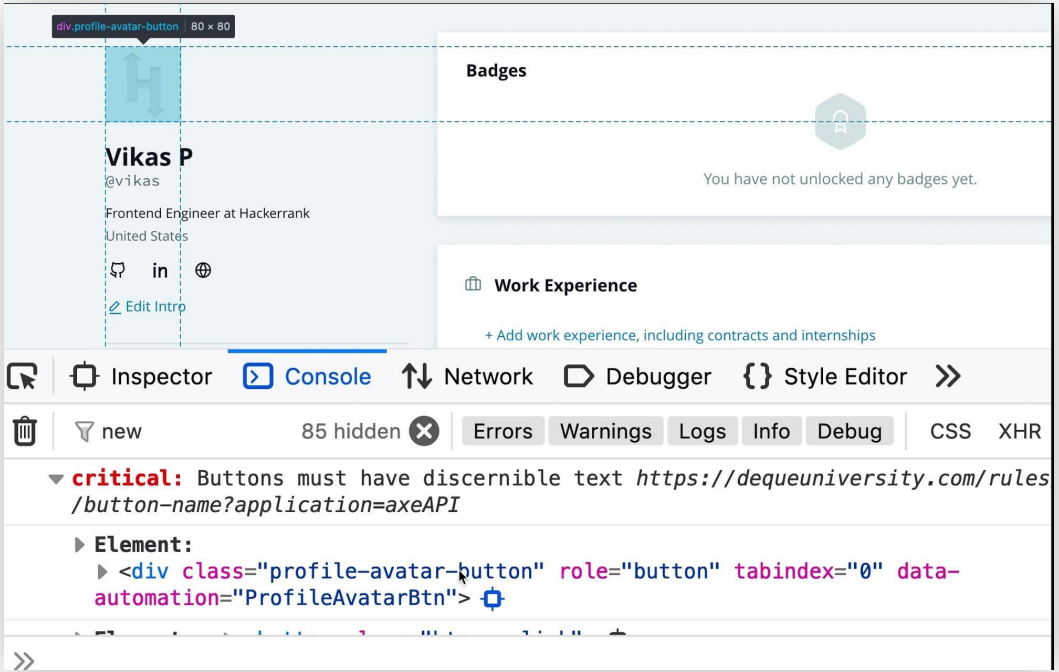
```
//before
> npm start

//with logger
> ENABLE_A11Y=1 npm start
```

Dev tools logger



Dev tools logger



Unit Tests

```
//use:
it('should not violate any a11y rules', async () => {
  const results = await auditA11y(<Button>Test</Button>);
  expect(results).to.be.accessible();
});
```

Unit Tests

```
FAIL src/shared/community/auth/auth_box.spec.js (7.562s)
  ● Test <AuthBox /> › should not violate any a11y rule

    AssertionError: Expected the HTML to have no violations(total violations: 2):


      #1:
      Node: <button><img src="/"></button>
      Rule Description: "Buttons must have discernible text (button-name)"
      Summary: Fix all of the following:
        Element is in tab order and does not have accessible text

      Fix any of the following:
        Element has a value attribute and the value attribute is empty
        Element has no value attribute or the value attribute is empty
        Element does not have inner text that is visible to screen readers
        aria-label attribute does not exist or is empty
        aria-labelledby attribute does not exist, references elements that do not exist or referenc
        Element's default semantics were not overridden with role="presentation"
        Element's default semantics were not overridden with role="none"
      Help URL: https://dequeuniversity.com/rules/axe/3.0/button-name?application=axeAPI
```

Accessible Rich Internet Applications (ARIA)

- Tells AT about the semantics of widgets, structures and behaviours.

ARIA

```
<div
  role="toolbar" aria-label="Settings" //structure
>
  <button
    role="switch"          // widget
    aria-pressed="true" // behaviour/state
  >
    Dark theme
  </button>
</div>
```

UI Kit

Reusable component library

Some commonly used components:

1. Button
2. Dialog
3. Tab

Button

- Most common mixup happens between **button** and anchor tag(**<a>**) elements.
- Using router to handle navigation by adding **onClick** event listener.

Button

```
//Don't do this. ❌
<Link to="/home">
  <Button>Home</Button>
</Link>

//or

//Don't do this.❌
handleClick() {
  //logic to navigate to '/home' using router etc
}

<Button onClick={handleClick}>Home</Button>
```

Button

```
//Fix
import { Link } from 'react-router';

function Button(props) {
  const {role, children, ...rest} = props;

  const isLink = props.role === 'link';
  const Element = isLink ? Link : 'button';

  return (
    <Element {...props}>{children}</Element>
  )
}
```

Button

```
//after

//as link ✅
<Button role="link" to="/home">Home</Button>

// as button ✅
<Button onClick={handleClick}>Submit</Button>

}
```

UI Kit

Reusable component library

Some commonly used components:

1. Button
2. Dialog
3. Tab

Dialog

- Should close on ESC key press.
- Focus should be inside once open.
- Focus should be returned to triggering element when dialog is closed.

Dialog

```
//get the focusable elements so
// they get focus on open
const elm = modal.querySelector('input, select, textarea, button:not
(.ui-dialog-close), a');
```

Dialog

```
//if no focusable element is present, focus on close button
const focusElement = elm || modal.querySelector
('button.ui-dialog-close');
```

Dialog

```
//we call this on modal open and on TAB events
focusTrap() {
  const modal = this.modalRef;
  const activeElement = document.activeElement;

  //if dialog is open and focus is not inside
  if (modal && !modal.contains(activeElement)) {

    //and if have our focus element, focus on it.
    if (focusElement) {
      focusElement.focus();
      return;
    }
```

Dialog

```
//markup
<section
  role="dialog"  //1
  aria-modal="true" //2
  aria-labelledby={GENERATED_ID} //3
>
  {props.onClose &&  //4
    <IconButton Icon={CrossIcon} btnText="Close Dialog"/>
  }
  <h1 id={GENERATED_ID}>{props.title}</h1>
  ...
</section>
```

Dialog

1. **role=‘dialog’**

   Tells assistive tech that the container is for a dialog.

2. **aria-modal=‘true’**

   Tells assistive tech that the windows underneath the dialog are not available for interaction.

Dialog

3. **aria-labelledby=‘idref’**

   Gives the dialog an accessible name by referring to the element that provides the dialog title.

UI Kit

Reusable component library

Some commonly used components:

1. Button
2. Dialog
3. Tab

Tab

1. When focus moves into tablist, focus on active tab.
2. Should tell AT that container is a set of tabs.
3. Should tell AT about the currently active tab.
4. Should tell AT about the what tab controls what content.

Tab

```
//example
<Tab title="Examples">
  <Tab.List tabList={ ... }/>
  <Tab.Content>
    ...
  </Tab.Content>
</Tab>
```

Tab

```
//Tab
<div aria-label={title} role="tablist">
  {children}
</div>
```

Tab

```
//List component
<button
  role="tab"
  aria-selected="true"
  id="example-tablist-0"
  aria-controls="example-0-tab"
  onClick={handleClick} //change tabpanel
>
  ...
</button>
```

Tab

```
//Content
<div
  tabindex="0"
  role="tabpanel"
  id="example-0-tab"
  aria-labelledby="example-tablist-0"
>
  ...
</div>
```

Tab

1. **role=”tablist”**

   Indicates that the element is container for a set of tabs.

2. **role=”tabpanel”**

   Indicates that the element is container for tab panel content.

Tab

3. **role=”tab”**

    - Indicates the element serves as a tab control.
    - Provides title for associated tab panel.
    - Should be used with button element.

What’s next?

Next?

- Integrate react-router 5 for in app focus management.
- CI/CD integration.
- More awareness.

Learnings?

Learnings

- No ARIA is better than Bad ARIA.

```
//Don't do this. ❌
<a href="#" aria-label="Read more">Link Text</a>
```

Learnings

- No ARIA is better than Bad ARIA.

```
<button aria-label="close">X</button>


<button>
  <span class="sr-only">close</span>
  X
</button>
```

Learnings

- No ARIA is better than Bad ARIA.

```
// multiple event handlers,
// needs explicit focus style ❌
<div
  role="button"
  onClick={handleClick}
  onKeyDown={handleKeyDown}
  tabIndex="0"
>
  Submit
</div>
```

Learnings

- No ARIA is better than Bad ARIA.

```
// semantic,
// browser focus styles ✅
<button onClick={handleClick}>Submit</button>
```

Learnings

- No ARIA is better than Bad ARIA.
- Tooling helps catching issues.
- Treat accessibility as requirement and not as feature.
- Automated tests are good start but user testing is a must.

Resources

- react-axe(https://github.com/dequelabs/react-axe)
- axe-core(https://github.com/dequelabs/axe-core)
- eslint-plugin-jsx-a11y(https://github.com/evcohen/eslint-plugin-jsx-a11y)
- Laws and policies(https://www.w3.org/WAI/policies/)
- Color Safe(http://colorsafe.co/)

Many more at https://a11yproject.com/resources

Resources

Demo code(gist): https://bit.ly/2CNLQf8

Thank you.

Twitter: @vicode_in