Accessibility in React Applications
React Zurich Meetup – April 2019

João Figueiredo
@lucalanca
Why should we care about a11y?
1. It affects more people than you think.

1.3 billion people with disabilities.
1. It affects more people than you think.  
   1.3 billion people with disabilities.

2. It will become mandatory.  
   In some countries or fields, it already is.
1. It affects more people than you think.
   1.3 billion people with disabilities.

2. It will become mandatory.
   In some countries or fields, it already is.

3. It’s not as hard as you think.
   Web Technologies and Tools are getting better.
The current state of the web.
The WebAIM Million
An automated accessibility analysis of the top 1,000,000 home pages.
The WebAIM Million
An automated accessibility analysis of the top 1,000,000 home pages.

30% can be verified with automation.
The WebAIM Million
An automated accessibility analysis of the top 1,000,000 home pages.

30% can be verified with automation.

97.8% pages with accessibility failures.
The WebAIM Million
An automated accessibility analysis of the top 1,000,000 home pages.

30% can be verified with automation.

97.8% pages with accessibility failures.

+10.1% errors in React websites.
We are preventing people from using our websites.
We need to do **way** better.
What’s in for today

1. Structure Patterns
2. Interaction Patterns
3. Application Patterns
1

Structural Patterns
Landmarks

They guide screen readers through the website.
<p>| HTML5 | Role |</p>
<table>
<thead>
<tr>
<th>HTML5</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;header /&gt;</code></td>
<td><code>banner *</code></td>
</tr>
</tbody>
</table>

* Does not apply if descendant of `article`, `aside`, `main`, or `section`. 
<table>
<thead>
<tr>
<th>HTML5</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;header /&gt;</code></td>
<td><code>banner *</code></td>
</tr>
<tr>
<td><code>&lt;main /&gt;</code></td>
<td><code>main</code></td>
</tr>
</tbody>
</table>

* Does not apply if descendant of `article, aside, main, or section.`

** Requires labelling (more on that later).
<table>
<thead>
<tr>
<th>HTML5</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;header /&gt;</code></td>
<td><code>banner *</code></td>
</tr>
<tr>
<td><code>&lt;main /&gt;</code></td>
<td><code>main</code></td>
</tr>
<tr>
<td><code>&lt;footer /&gt;</code></td>
<td><code>contentinfo</code></td>
</tr>
</tbody>
</table>

* Does not apply if descendant of `article`, `aside`, `main`, or `section`. 
<table>
<thead>
<tr>
<th>HTML5</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;header /&gt;</td>
<td>banner *</td>
</tr>
<tr>
<td>&lt;main /&gt;</td>
<td>main</td>
</tr>
<tr>
<td>&lt;footer /&gt;</td>
<td>contentinfo</td>
</tr>
<tr>
<td>&lt;nav /&gt;</td>
<td>navigation</td>
</tr>
</tbody>
</table>

* Does not apply if descendant of **article**, **aside**, **main**, or **section**.
<table>
<thead>
<tr>
<th>HTML5</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;header /&gt;</td>
<td>banner *</td>
</tr>
<tr>
<td>&lt;main /&gt;</td>
<td>main</td>
</tr>
<tr>
<td>&lt;footer /&gt;</td>
<td>contentinfo</td>
</tr>
<tr>
<td>&lt;nav /&gt;</td>
<td>navigation</td>
</tr>
<tr>
<td>&lt;section /&gt;</td>
<td>region **</td>
</tr>
</tbody>
</table>

* Does not apply if descendant of *article*, *aside*, *main*, or *section*.

** Requires labelling (more on that later).
<table>
<thead>
<tr>
<th>HTML5</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;header /&gt;</td>
<td>banner *</td>
</tr>
<tr>
<td>&lt;main /&gt;</td>
<td>main</td>
</tr>
<tr>
<td>&lt;footer /&gt;</td>
<td>contentinfo</td>
</tr>
<tr>
<td>&lt;nav /&gt;</td>
<td>navigation</td>
</tr>
<tr>
<td>&lt;section /&gt;</td>
<td>region **</td>
</tr>
<tr>
<td>&lt;form /&gt;</td>
<td>form</td>
</tr>
</tbody>
</table>

* Does not apply if descendant of **article, aside, main, or section.**

** Requires labelling (more on that later).
<table>
<thead>
<tr>
<th>HTML5</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;header /&gt;</code></td>
<td>banner *</td>
</tr>
<tr>
<td><code>&lt;main /&gt;</code></td>
<td>main</td>
</tr>
<tr>
<td><code>&lt;footer /&gt;</code></td>
<td>contentinfo</td>
</tr>
<tr>
<td><code>&lt;nav /&gt;</code></td>
<td>navigation</td>
</tr>
<tr>
<td><code>&lt;section /&gt;</code></td>
<td>region **</td>
</tr>
<tr>
<td><code>&lt;form /&gt;</code></td>
<td>form</td>
</tr>
<tr>
<td><code>&lt;aside /&gt;</code></td>
<td>complementary</td>
</tr>
</tbody>
</table>

* Does not apply if descendant of *article, aside, main, or section*.

** Requires labelling (more on that later).
<table>
<thead>
<tr>
<th>HTML5</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;header /&gt;</td>
<td>banner *</td>
</tr>
<tr>
<td>&lt;main /&gt;</td>
<td>main</td>
</tr>
<tr>
<td>&lt;footer /&gt;</td>
<td>contentinfo</td>
</tr>
<tr>
<td>&lt;nav /&gt;</td>
<td>navigation</td>
</tr>
<tr>
<td>&lt;section /&gt;</td>
<td>region **</td>
</tr>
<tr>
<td>&lt;form /&gt;</td>
<td>form</td>
</tr>
<tr>
<td>&lt;aside /&gt;</td>
<td>complementary</td>
</tr>
</tbody>
</table>

* Does not apply if descendant of *article, aside, main, or section*.

** Requires labelling (more on that later).
HTML5 is a good starting point.
header, main and footer roles are automatically inferred.
Override when the tag is not enough.
Label `nav` when there is more than one in a page.

* using “Secondary Navigation” is redundant.
Visually, a search field doesn’t have a label but we still need to add it.
You can use other elements as labels, when they are also visible on the page.
Forms

The thing everyone hates.

Some tips on how to make them better for visually-impaired users.
1. Always be labelling those inputs.
1. Always be labelling those inputs.

You can label an input by wrapping it in a label with text.
1. Always be labelling those inputs.

```html
<form aria-labelledby="attend-title">
  <h2 id="attend-title">Attend React Zurich</h2>

  <label>
    Your name:
    <input type="text" name="name" placeholder="Michael Scott" />
  </label>

  <label htmlFor="company">Your company</label>
  <input type="text" name="company" id="company" placeholder="ACME Inc." />

  <button>RSVP</button>
</form>
```

An alternative is to explicitly connect a label via `htmlFor` and `id` attributes.
1. Always be labelling those inputs.

```javascript
function InputCompany() {
    return (
        <>
            <label htmlFor='company'>Company</label>
            <input id='company' />
        </>
    )
}

<InputCompany />  
<InputCompany />  
// error: same id is used twice.
```
Even if your designer didn’t put it there.
Even if your designer didn’t put it there.

Your username

```javascript
let style = {
    border: 0,
    clip: 'rect(0 0 0 0)',
    height: '1px',
    width: '1px',
    margin: '-1px',
    padding: 0,
    overflow: 'hidden',
    position: 'absolute'
}

function VisuallyHidden({ Component = 'span', ...rest }) {
    return <Component style={style} {...rest} />
}
```
Even if your designer didn’t put it there.

Your username

```javascript
let style = {
  border: 0,
  clip: 'rect(0 0 0 0)',
  height: '1px',
  width: '1px',
  margin: '-1px',
  padding: 0,
  overflow: 'hidden',
  position: 'absolute'
}

function VisuallyHidden({ Component = 'span', ...rest }) {
  return <Component style={style} {...rest} />
}
```
2. Don’t forget to also label forms.

```html
<form aria-labelledby="attend-title">
  <h2 id="attend-title">Attend React Zurich</h2>

  <label>
    Your name:  
    <input type="text" name="name" placeholder="Michael Scott" />
  </label>

  <label htmlFor="company">Your company</label>
  <input type="text" name="company" id="company" placeholder="ACME Inc." />

  <button>RSVP</button>
</form>
```
Decorative link icons

See all Attendees ➔
Decorative link icons

See all Attendees ➔

```html
<a href="/attendees">
  See all Attendees
  <svg aria-hidden="true"
       xmlns="http://www.w3.org/2000/svg"
       viewBox="0 0 24 24">
  ...
  </svg>
</a>
```
Icon-only links

FOLLOW US ON

<a href="https://www.facebook.com/ACME-Inc"
aria-label="Find us on facebook">
  <svg aria-hidden="true">...</svg>
</a>
Interaction Patterns
Form Validation
Formik is a package to handle form validation.

```jsx
<Formik>
  {({ values, errors, touched }) => (
    <form>
      <label>
        Name:
        <input
          name="name"
          value={values.name}
          aria-describedby='nameError'
        />
        <span id='nameError' aria-live='polite'>
          {errors.name || null}
        </span>
      </label>
    </form>
  )}
</Formik>
```
1. Associate inputs with their error elements.

```jsx
<Formik
  initialValues={{} as any}
  validate={values => {
    const errors: any = {}
    if (!values.name) errors.name = 'Name is required'
    return errors
  }}
}>{({ values, errors, touched }) => {
  <form
    <label>
      Name:
      <input
        name="name"
        value={values.name}
        aria-describedby='nameError'
      />
      <span id='nameError' aria-live='polite'>
        {errors.name || null}
      </span>
    </label>
  </form>
}}
</Formik>
```
2. Show errors as the user types.

```jsx
<Formik
  initialValues={{ name: '' }}
  validate={values => (errors, setFieldError) => {
    if (!values.name) {
      setFieldError('name', 'Name is required');
    }
    return errors;
  }}
  handleSubmit={(values, formik) => {
    // Handle form submission
  }}
>
  <form>
    <label htmlFor='name'>Name:</label>
    <input name='name' value={values.name} onChange={formik.handleChange} />
    <span id='nameError' aria-live='polite'>
      {errors.name || null}
    </span>
  </form>
</Formik>
```
3. OR, show errors when the user finishes.

```jsx
<Formik>
  {({ values, errors, touched }) => (
    <form>
      <label>
        Name:
        <input
          name="name"
          value={values.name}
          aria-describedby='nameError'
        />
        <span id='nameError' aria-live='assertive'>
          {errors.name && touched.name ? errors.name : null}
        </span>
      </label>
    </form>
  )}
</Formik>
```

*Emphasize the importance of the message and causes screen readers to interrupt their current tasks to read aloud this message.*
Icon Toggles
Icon Toggles
Icon Toggles

```javascript
export function MuteButton({ muted = false, onMutedChange }) {
  return (
    <button
      aria-pressed={muted}
      onClick={onMutedChange}
      aria-label='Mute'
    >
      <svg aria-hidden='true'>
        {/*...*/}
      </svg>
    </button>
  );
}
```
export function MuteButton({ muted = false, onMutedChange }) {
    return (;
        <button
            aria-pressed={muted}
            onClick={onMutedChange}
            aria-label='Mute'
        >
            <svg aria-hidden='true'>
                {/*...*/}
            </svg>
        </button>
    );
}
export function MuteButton({ muted = false, onMutedChange }) {
  return (  
    <button  
      aria-pressed={muted}  
      onClick={onMutedChange}  
      aria-label='Mute'  
    >  
      <svg aria-hidden='true'>  
        {/*...*/}  
      </svg>  
    </button>  
  );
}
Application Patterns
Routing

DEMO TIME
Routing Focus Management

When the new page mounts, focus on its outer container.
Routing Focus Management

When the new page mounts, focus on its outer container.

```javascript
function MyPage() {
  const pageRef = useRef(null);
  useEffect(() => {
    pageRef.current.focus();
  }, []);
  return (
    <div tabIndex={-1} aria-labelledby="title" ref={pageRef}>
      <h1 id='title'>My Title</h1>
      {/* ... */}
    </div>
  );
}
```
Routing Focus Management

When the new page mounts, focus on its outer container.

```javascript
function MyPage() {
  const pageRef = useRef(null);
  useEffect(() => {
    pageRef.current.focus();
  }, []);
  return (
    <div tabIndex={-1} aria-labelledby="title" ref={pageRef}>
      <h1 id='title'>My Title</h1>
      {/* ... */}
    </div>
  );
}
```
Routing Focus Management

When the new page mounts, focus on its outer container.

```javascript
function MyPage() {
  const pageRef = useRef(null);
  useEffect(() => {
    pageRef.current.focus();
  }, []);
  return (
    <div tabIndex={-1} aria-labelledby="title" ref={pageRef}>
      <h1 id='title'>My Title</h1>
    </div>
  );
}
```
Routing Focus Management

When the new page mounts, focus on its outer container.

```javascript
function MyPage() {
  const pageRef = useRef(null);
  useEffect(() => {
    pageRef.current.focus();
  }, []);
  return (
    <div tabIndex={-1} aria-labelledby="title" ref={pageRef}>
      <h1 id='title'>My Title</h1>
      {/* ... */}
    </div>
  );
}
```
Notifications
Notifications

<VisuallyHidden role="status" aria-live="polite">
One item was added to the card.
</VisuallyHidden>

new content will be announced without user interaction.
Flash Messages

✅ Your information was updated.
Flash Messages

✅ Your information was updated.

```javascript
function Announcements({ announcements }) {
  return (突发事件
    <div role="status" aria-live="polite" aria-atomic="true">
      {announcements.map((item => (
        <p>
          <strong>
            <svg aria-hidden="false" />
            <VisuallyHidden>{item.type || 'Success'}</VisuallyHidden>
          </strong>
          {item.message}
        </p>
      )))
    </div>
  );
}
```

Have a region of the website for these messages.
Flash Messages

✔ Your information was updated.

```javascript
function Announcements({ announcements }) {
  return (
    <div role="status" aria-live="polite" aria-atomic="true">
      {announcements.map((item => (
        <p>
          <strong>
            <svg aria-hidden="false" />
            <VisuallyHidden>{item.type || 'Success'}</VisuallyHidden>
          </strong>
          {item.message}
        </p>
        ))})
    </div>
  );
}
```

Mark icon as decorative.
Flash Messages

☑️ Your information was updated.

```javascript
function Announcements({ announcements }) {
  return (
    <div role="status" aria-live="polite" aria-atomic="true">
      {announcements.map((item => {
        <p>
          <strong>
            <svg aria-hidden="false" />
            <VisuallyHidden>{item.type || 'Success'}</VisuallyHidden>
          </strong>
          {item.message}
        </p>
      )))
    </div>
  );
}
```

Provide text alternative.
Flash Messages

✅ Your information was updated.

```javascript
function Announcements({ announcements }) {
    return (
        <div role="status" aria-live="polite" aria-atomic="true">
            {announcements.map((item => (
                <p>
                    <strong>
                        <svg aria-hidden="false" />
                        {item.type || 'Success'}</strong>
                    {item.message}
                </p>
            ))})
        </div>
    );
}
```

Only announce newly added content.
That’s all we got time for today.
I’ll be sharing accessibility tips over the next months.