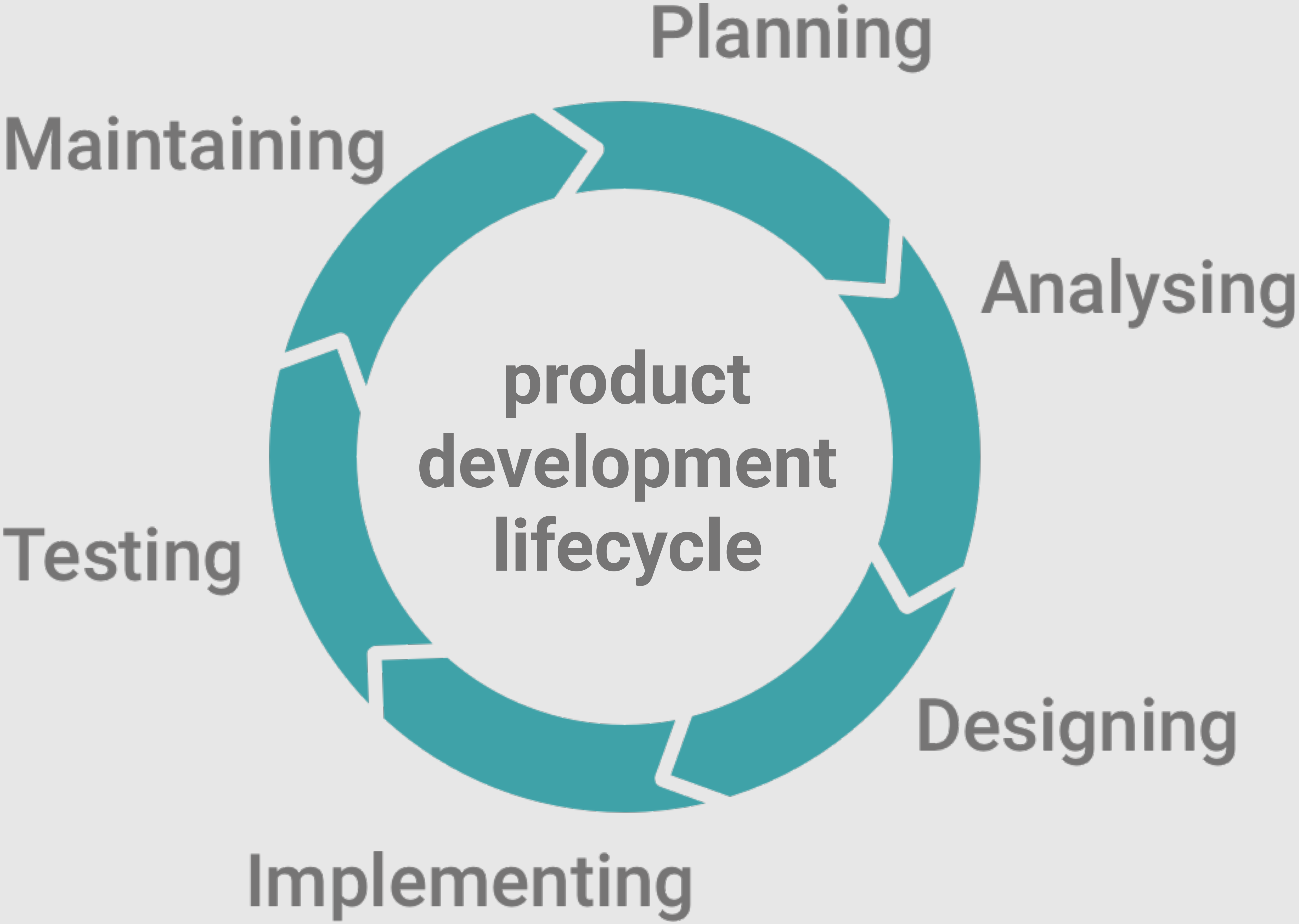
# becoming an accessibility-focused developer

Suzanne Aitchison

accessibility is everyone’s responsibility



product development lifecycle

Planning

Analysing

Designing

Implementing

Testing

Maintaining

# we work in imperfect organisations

accessibility isn’t part of MVP

# we work in imperfect organisations

just put a ticket in the backlog

# we work in imperfect organisations

we don't have any disabled users

# our sphere of control

Implementing

# habits to build

# habits to build

🕵️‍♀️ detect common issues

# habits to build

- 🕵️‍♀️ detect common issues
- 🚀 boost with some quick-win refactoring

# habits to build

- 🕵️‍♀️ detect common issues
- 🚀 boost with some quick-win refactoring
- 🐞 learn by finding bugs in the browser

add accessibility linting with eslint-plugin-jsx-a11y

# jsx-a11y

catch common errors, e.g:

# jsx-a11y

catch common errors, e.g:

- meaningful alt text on images

# jsx-a11y

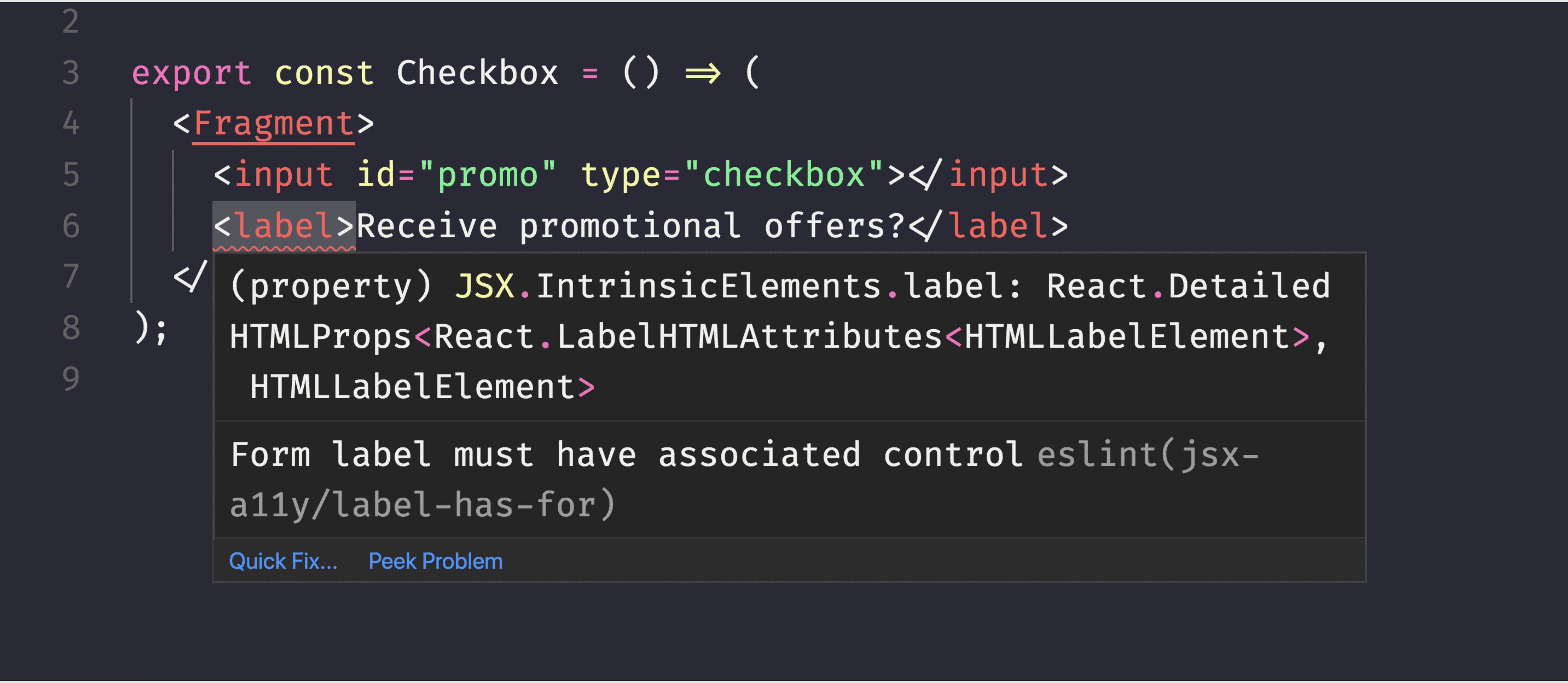
catch common errors, e.g:

- meaningful alt text on images
- association of labels with inputs

# jsx-a11y

catch common errors, e.g:

- meaningful alt text on images
- association of labels with inputs
- no non-interactive elements with click listeners



```
export const Checkbox = () => (
  <Fragment>
    <input id="promo" type="checkbox"></input>
    <label>Receive promotional offers?</label>
  </
);
```

(property) JSX.IntrinsicElements.label: React.DetailedHTMLProps<React.LabelHTMLAttributes<HTMLLabelElement>, HTMLLabelElement>

Form label must have associated control eslint(jsx-a11y/label-has-for) commit to 2 key pieces of refactoring

# refactoring: 1

replace divs with semantic alternatives

❗

Authors are strongly encouraged to view the <div> element as an element of last resort, for when no other element is suitable

*- W3C*

# landmarks

using landmarks correctly allows screen reader users to find and skip to relevant content more easily

# landmarks

<header>

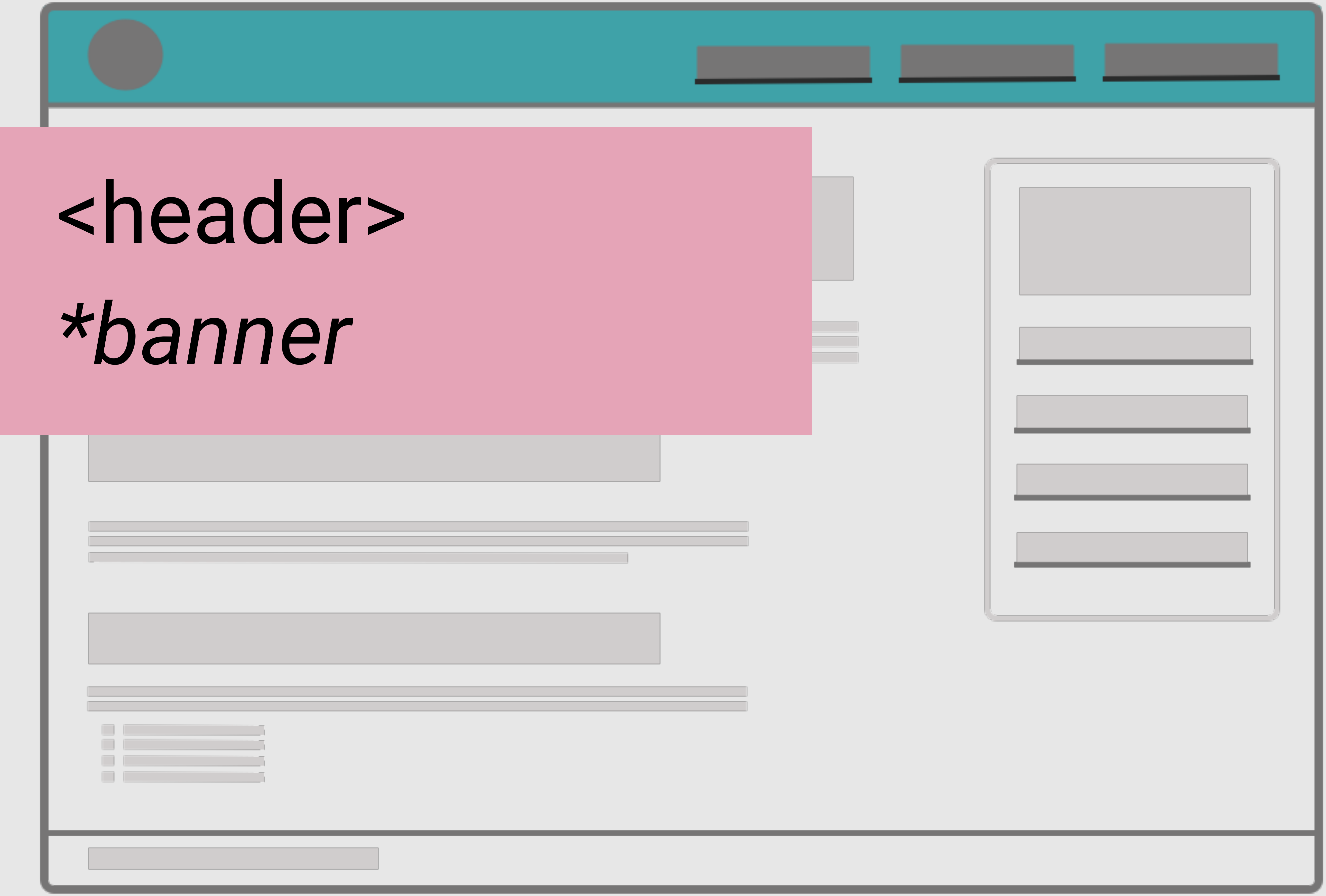
<footer>

<nav>

<main>

<aside>

<section>

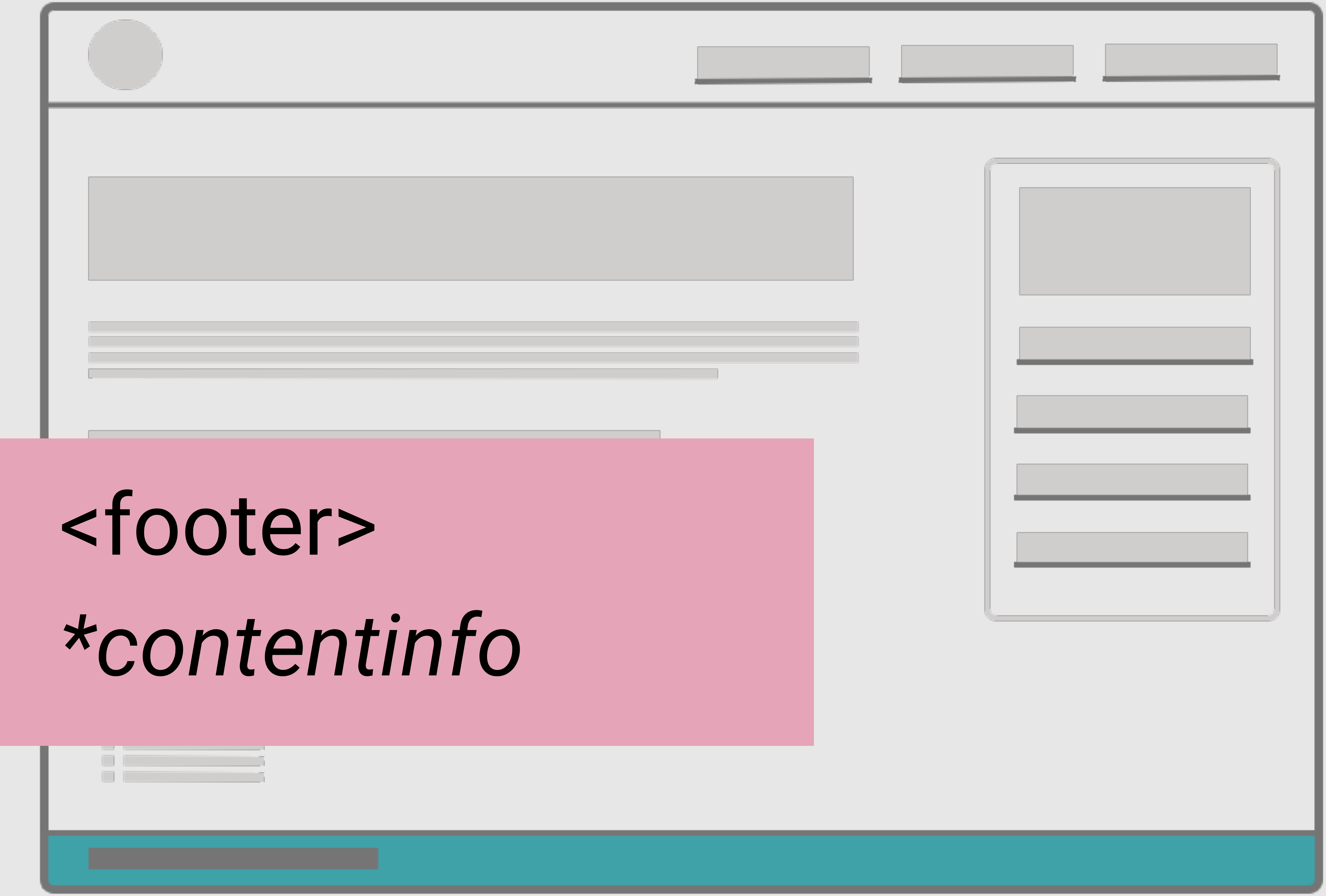


<header>

**banner*

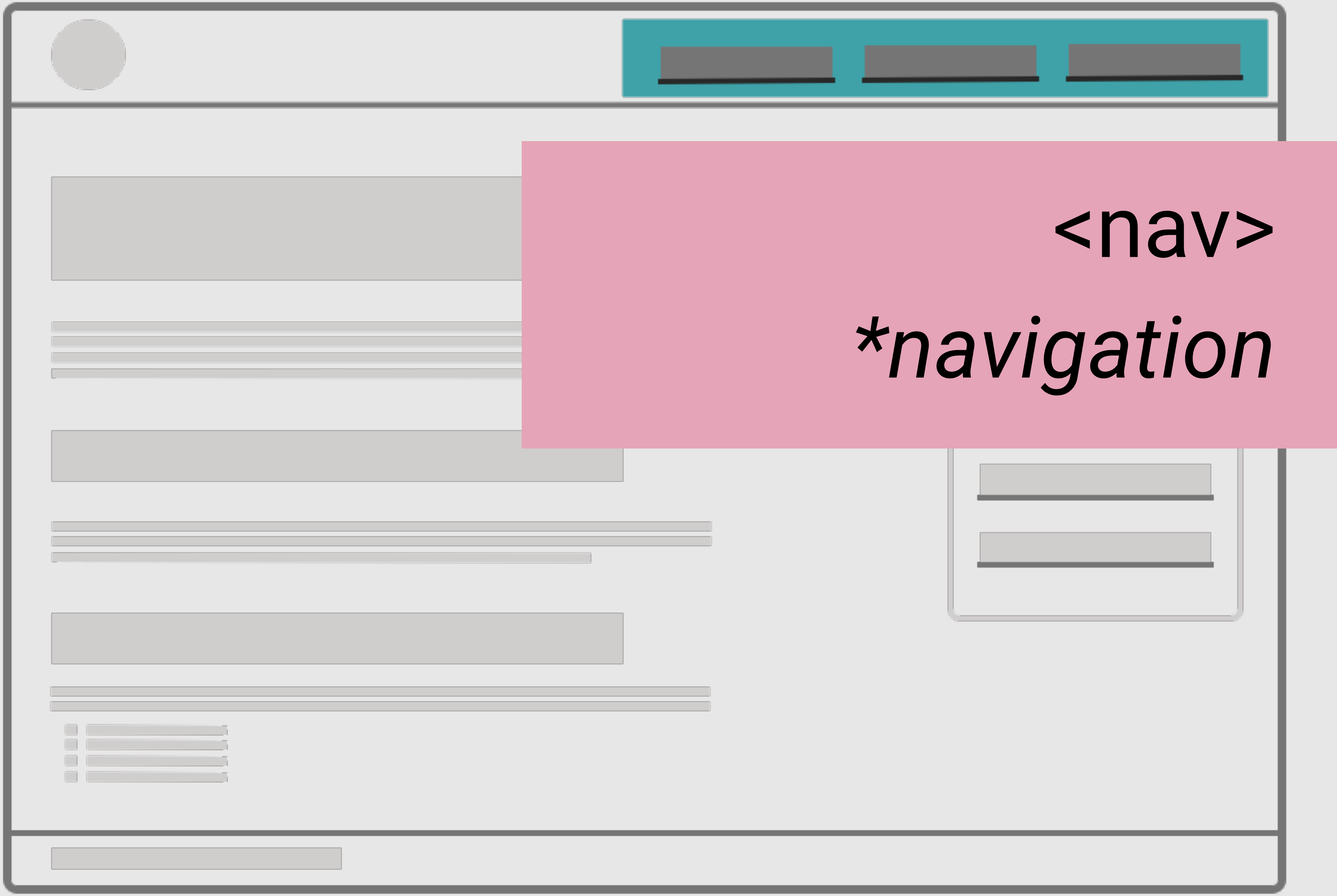
@s_aitchison

bit.ly/a11y-focused-developer



<footer>

**contentinfo*

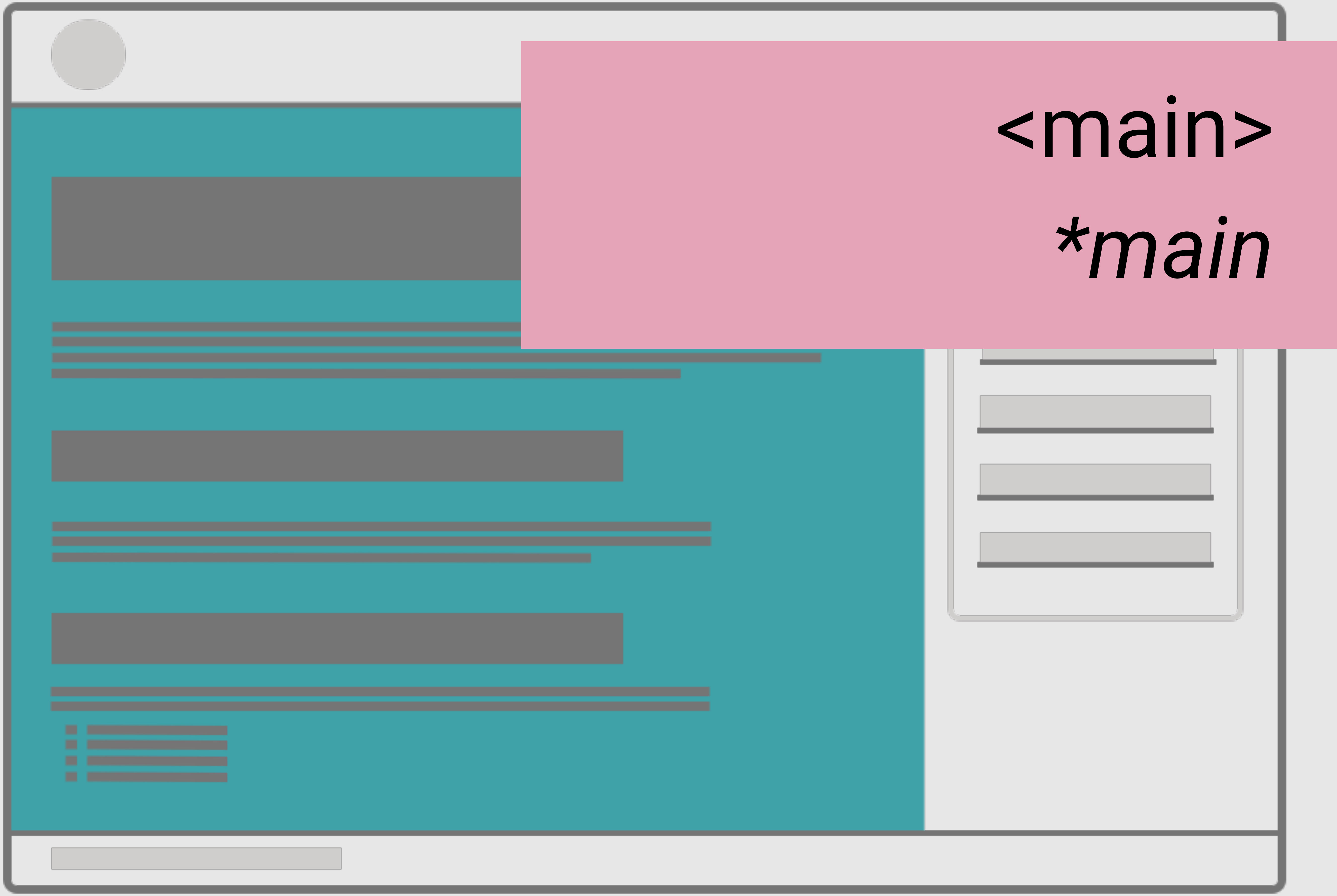


<nav>

*navigation

@s_aitchison

bit.ly/a11y-focused-developer

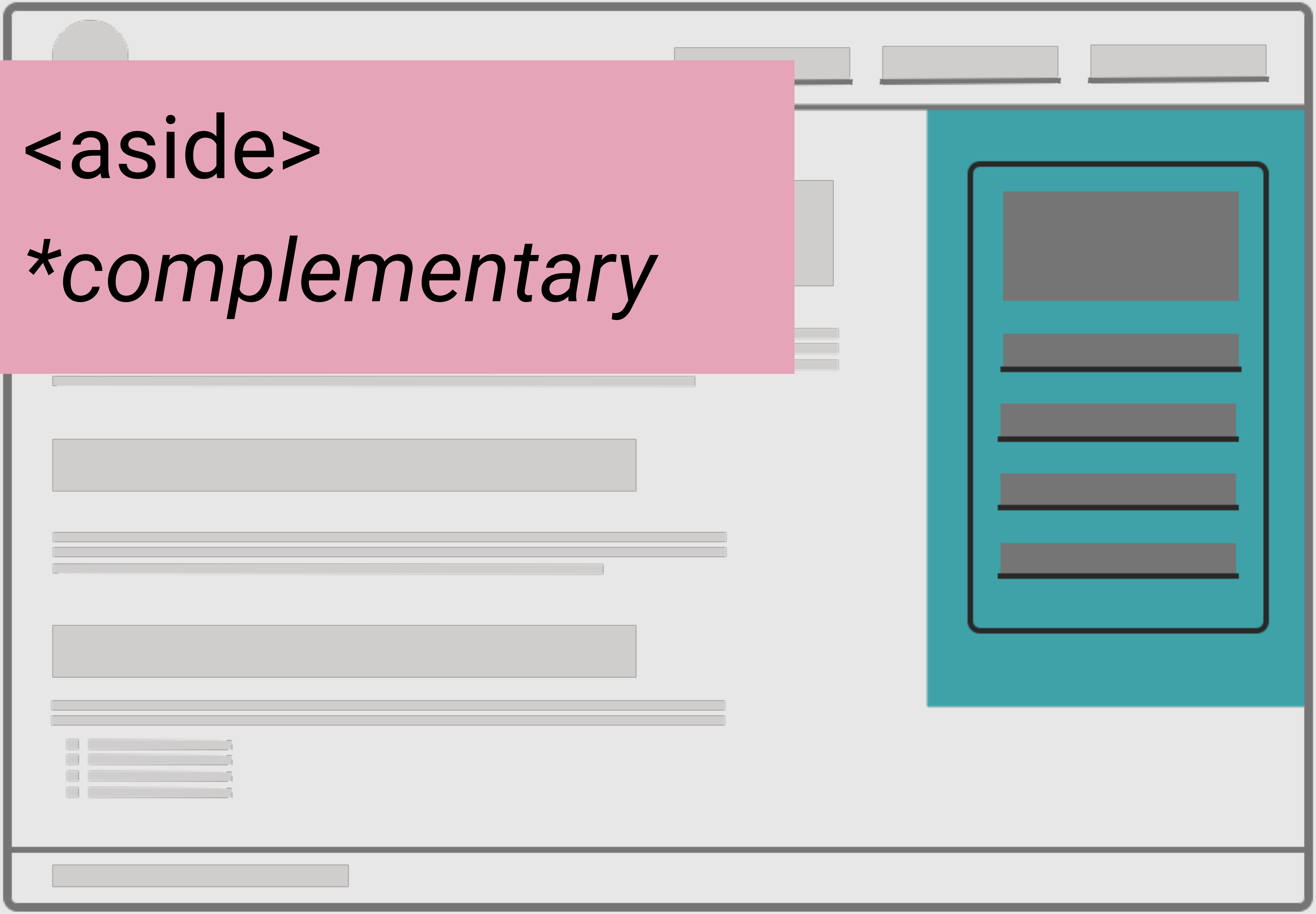


<main>

*main*

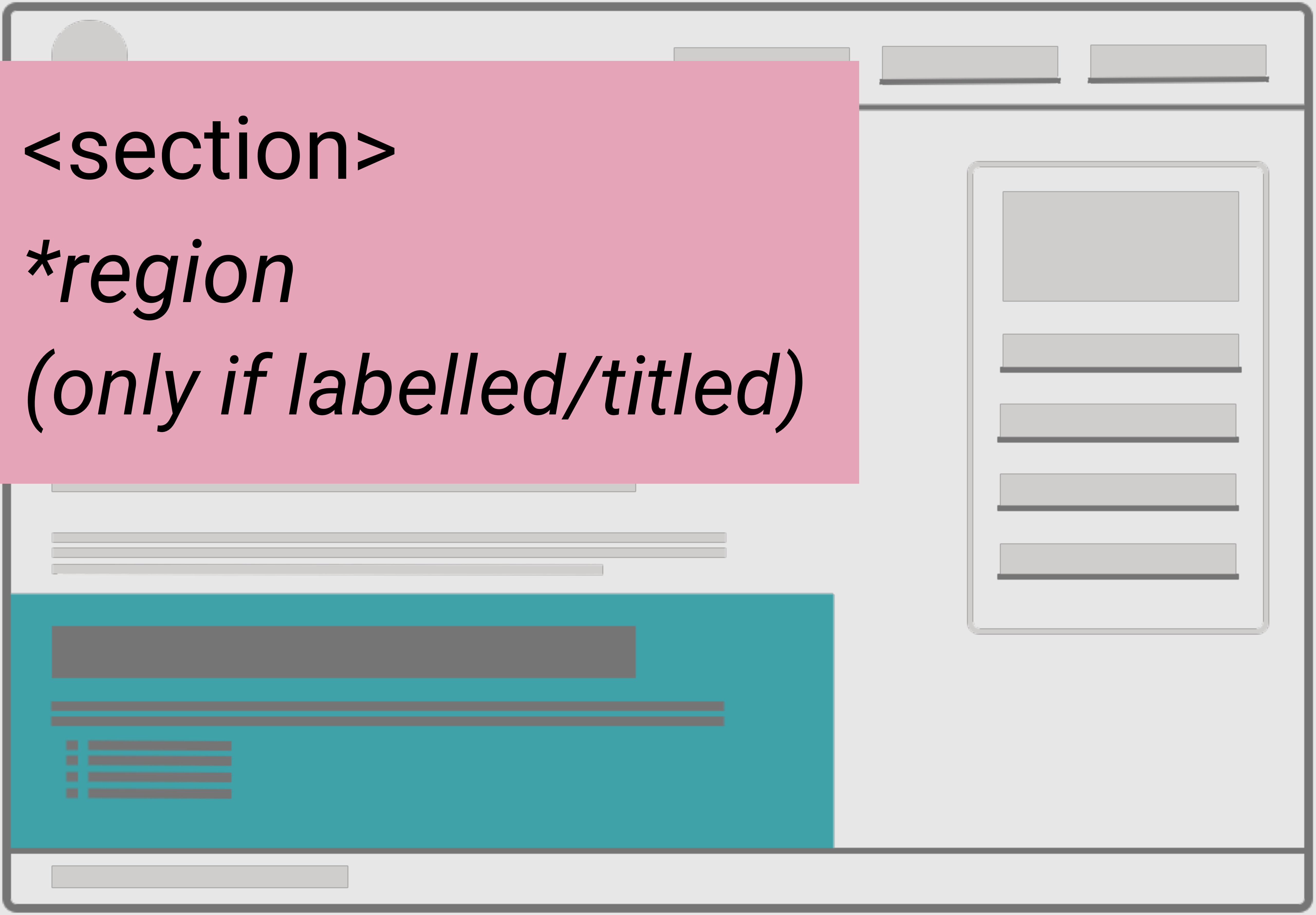
# <aside>

# **complementary*



@s_aitchison

bit.ly/a11y-focused-developer



<section>

**_*region_**

_(only if labelled/titled)_

@s_aitchison

bit.ly/a11y-focused-developer

# lists

ensure lists are marked up using <ul> and <li> to give screen reader users context

!

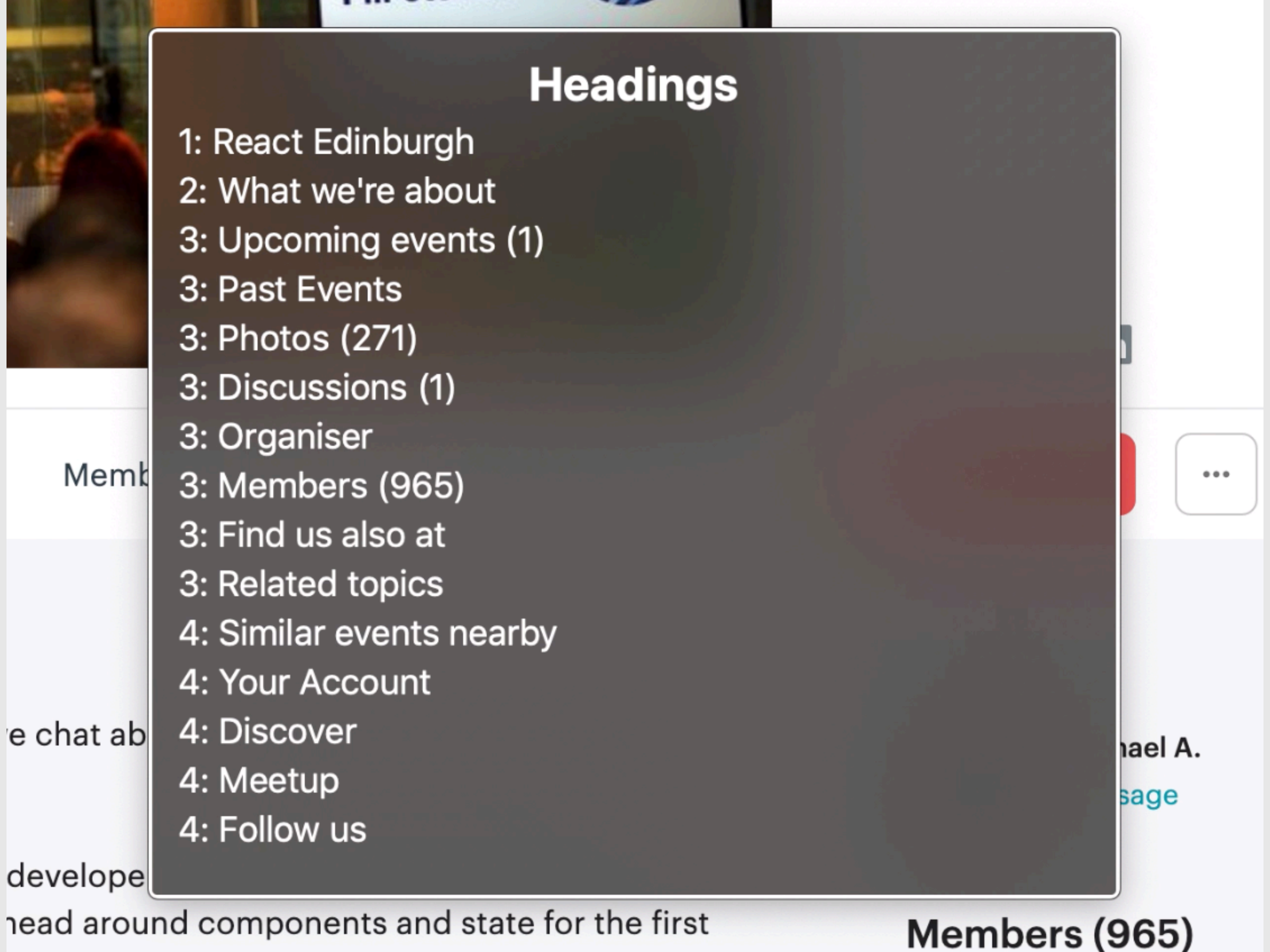
if you used **.map()** to create a set of elements for display, they should probably be **<li>** elements and displayed in a list

# refactoring: 2

audit your headings

**68.8%** of respondents said their first action on a web page is to navigate by heading levels

- *[WebAIM 2019 screen reader user survey]*

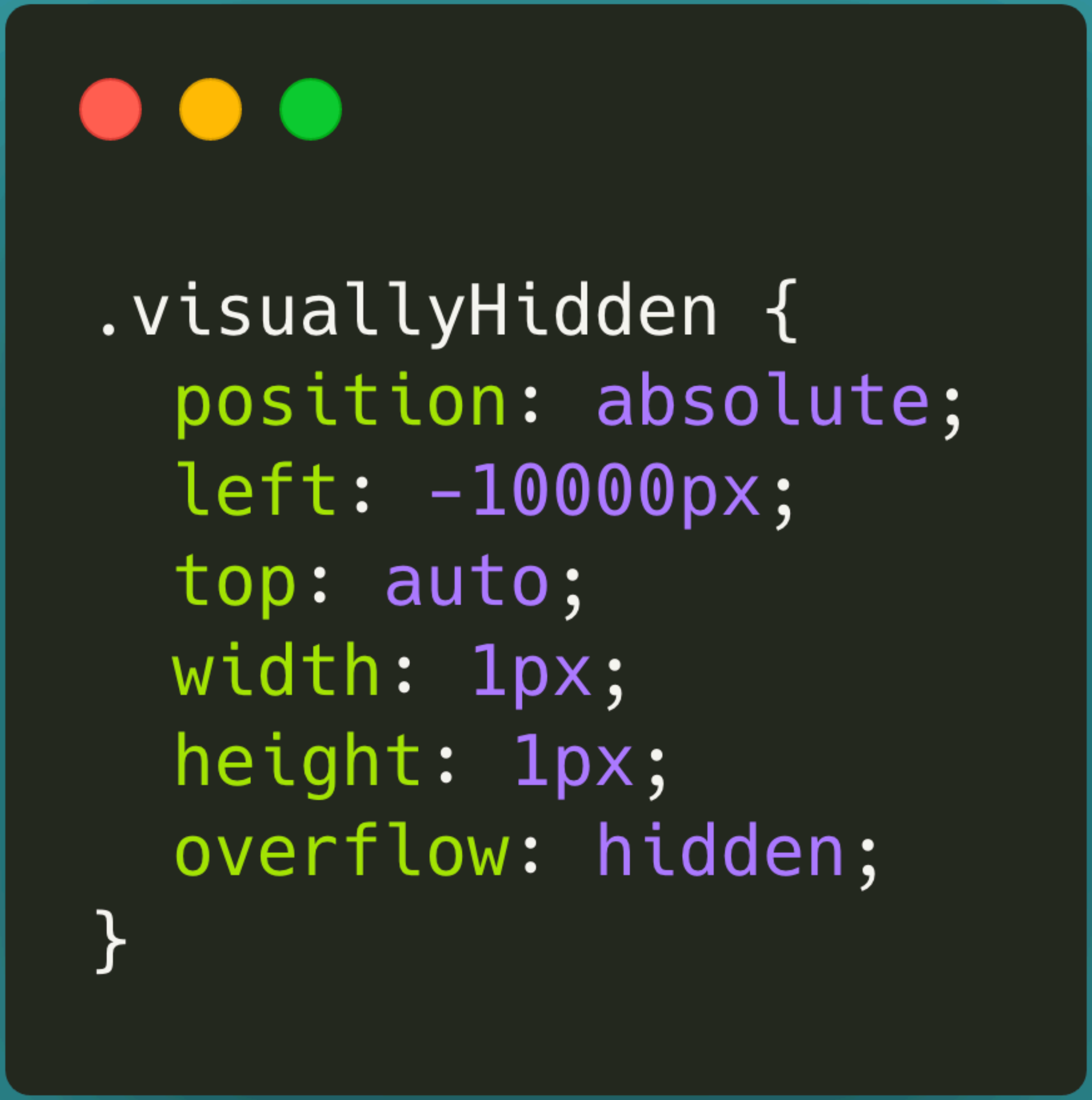


Headings

1: React Edinburgh
2: What we're about
3: Upcoming events (1)
3: Past Events
3: Photos (271)
3: Discussions (1)
3: Organiser
3: Members (965)
3: Find us also at
3: Related topics
4: Similar events nearby
4: Your Account
4: Discover
4: Meetup
4: Follow us

# headings

an invisible heading is better than no headings at all



```css
.visuallyHidden {
  position: absolute;
  left: -10000px;
  top: auto;
  width: 1px;
  height: 1px;
  overflow: hidden;
}
```

# heading levels

think of heading levels in terms of structure, not style

# heading structure

- Find a pet for adoption **<h1>**
  - Meet our dogs **<h2>**
  - Meet our cats **<h2>**
  - New arrivals! **<h2>**

# heading structure

- Find a pet for adoption **<h1>**
  - Meet our dogs **<h2>**
  - Meet our cats **<h2>**
  - New arrivals! **<h2>**

    = 😸 + 🐶 + ?

# heading structure

- Find a pet for adoption **<h1>**
  - Meet our dogs **<h2>**
  - Meet our cats **<h2>**
    - New arrivals! **<h3>**

# heading structure

- Find a pet for adoption **<h1>**
  - Meet our dogs **<h2>**
  - Meet our cats **<h2>**
    - New arrivals! **<h3>**

      = 😸

- only one <h1> per page

- only increase by one level at a time

verify your code in the browser with an accessibility checker, e.g. axe or wave

# axe / wave

catch common errors, e.g:

# axe / wave

catch common errors, e.g:

- colour contrast

# axe / wave

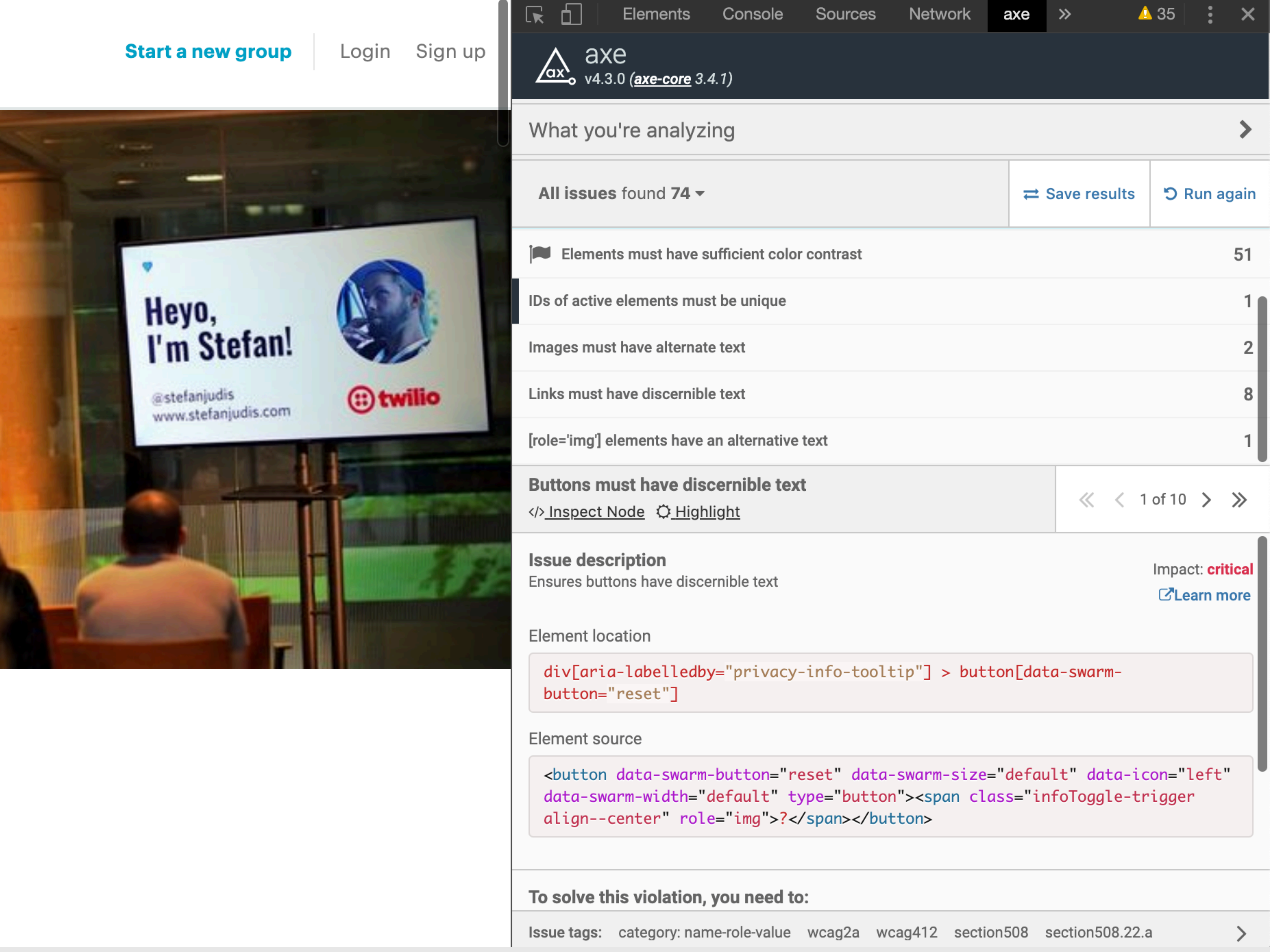
catch common errors, e.g:

- colour contrast
- heading levels skipped

# axe / wave

catch common errors, e.g:

- colour contrast
- heading levels skipped
- duplicate or missing landmarks



What you're analyzing

**All issues** found **74**

Save results

Run again

| | |
|---|---|
| Elements must have sufficient color contrast | 51 |
| IDs of active elements must be unique | 1 |
| Images must have alternate text | 2 |
| Links must have discernible text | 8 |
| [role='img'] elements have an alternative text | 1 |

**Buttons must have discernible text**

</> Inspect Node  Highlight

1 of 10

**Issue description**

Ensures buttons have discernible text

Impact: critical

Learn more

Element location

```
div[aria-labelledby="privacy-info-tooltip"] > button[data-swarm-
button="reset"]
```

Element source

```
<button data-swarm-button="reset" data-swarm-size="default" data-icon="left"
data-swarm-width="default" type="button"><span class="infoToggle-trigger
align--center" role="img">?</span></button>
```

**To solve this violation, you need to:**

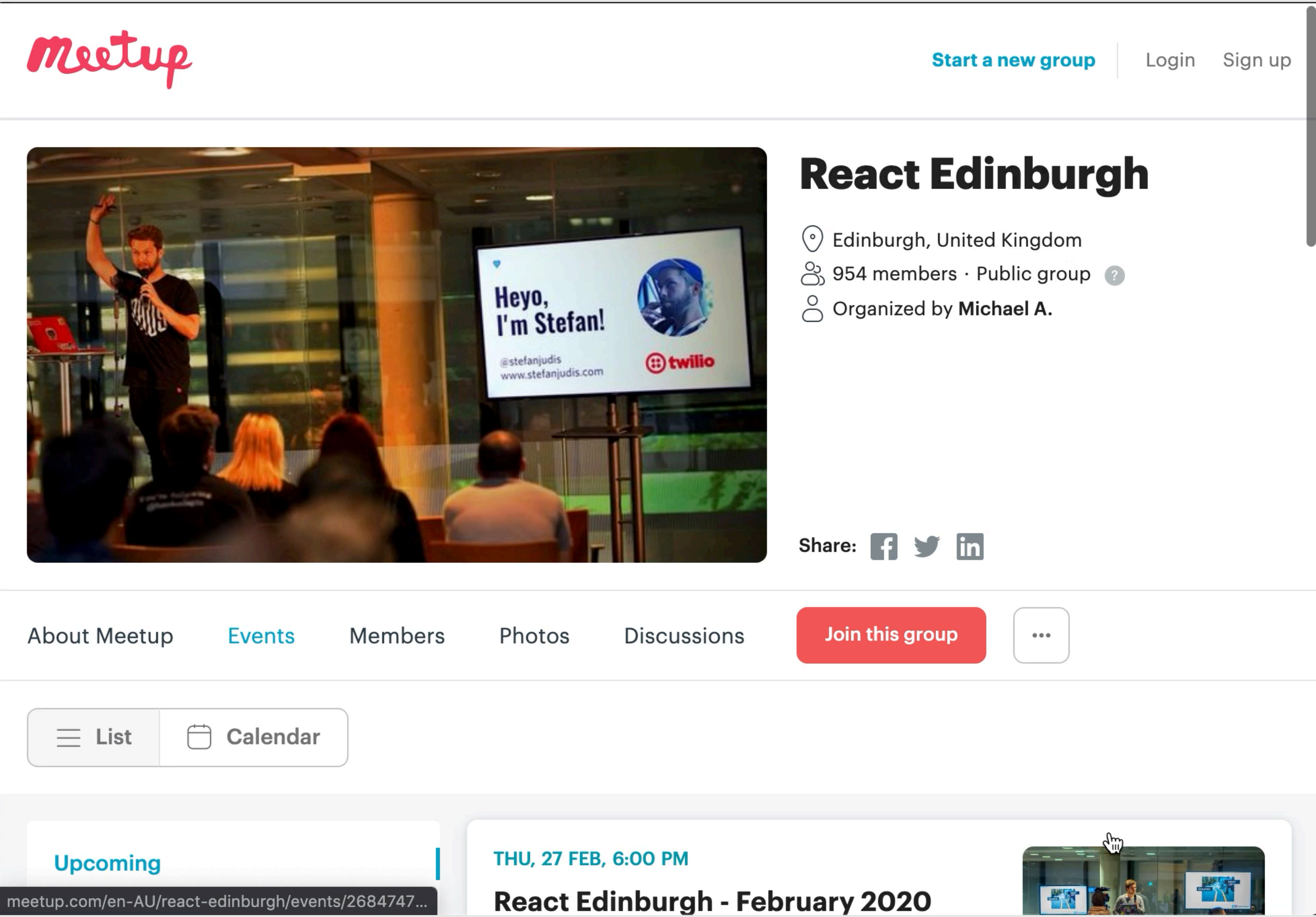
**Issue tags:** category: name-role-value wcag2a wcag412 section508 section508.22.a check your work is operable by keyboard alone

# keyboard operability

- verify you can tab to every interactive element (shift + tab to go backwards)

# keyboard operability

- verify you can tab to every interactive element (shift + tab to go backwards)
- check the current focus is clearly visible



# React Edinburgh

Edinburgh, United Kingdom

954 members · Public group

Organized by **Michael A.**

Share:

Upcoming

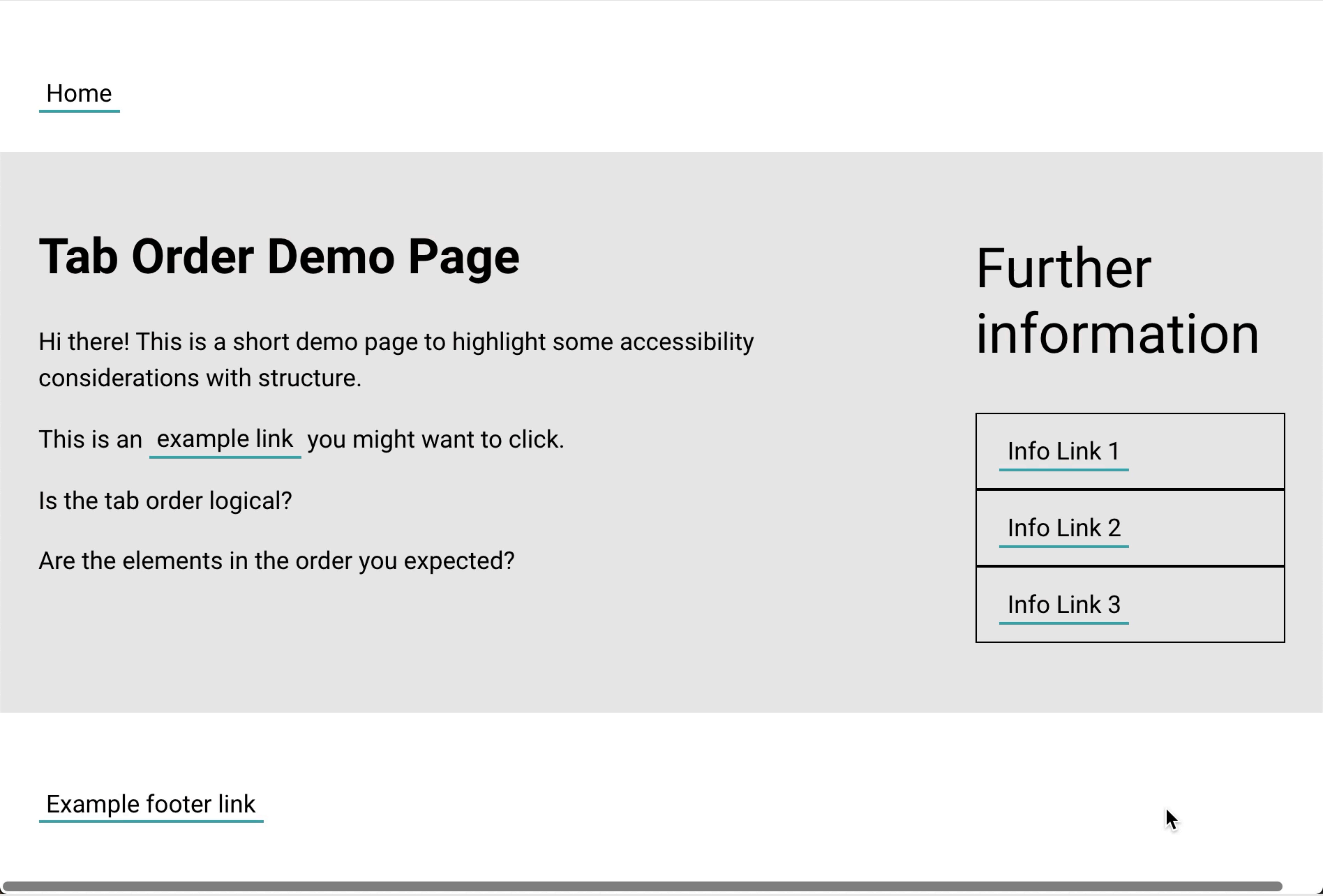
THU, 27 FEB, 6:00 PM

**React Edinburgh - February 2020**

meetup.com/en-AU/react-edinburgh/events/2684747...

# keyboard operability

- verify you can tab to every interactive element (shift + tab to go backwards)
- check the current focus is clearly visible
- ensure the tab order is logical



Home

# Tab Order Demo Page

Hi there! This is a short demo page to highlight some accessibility considerations with structure.

This is an example link you might want to click.

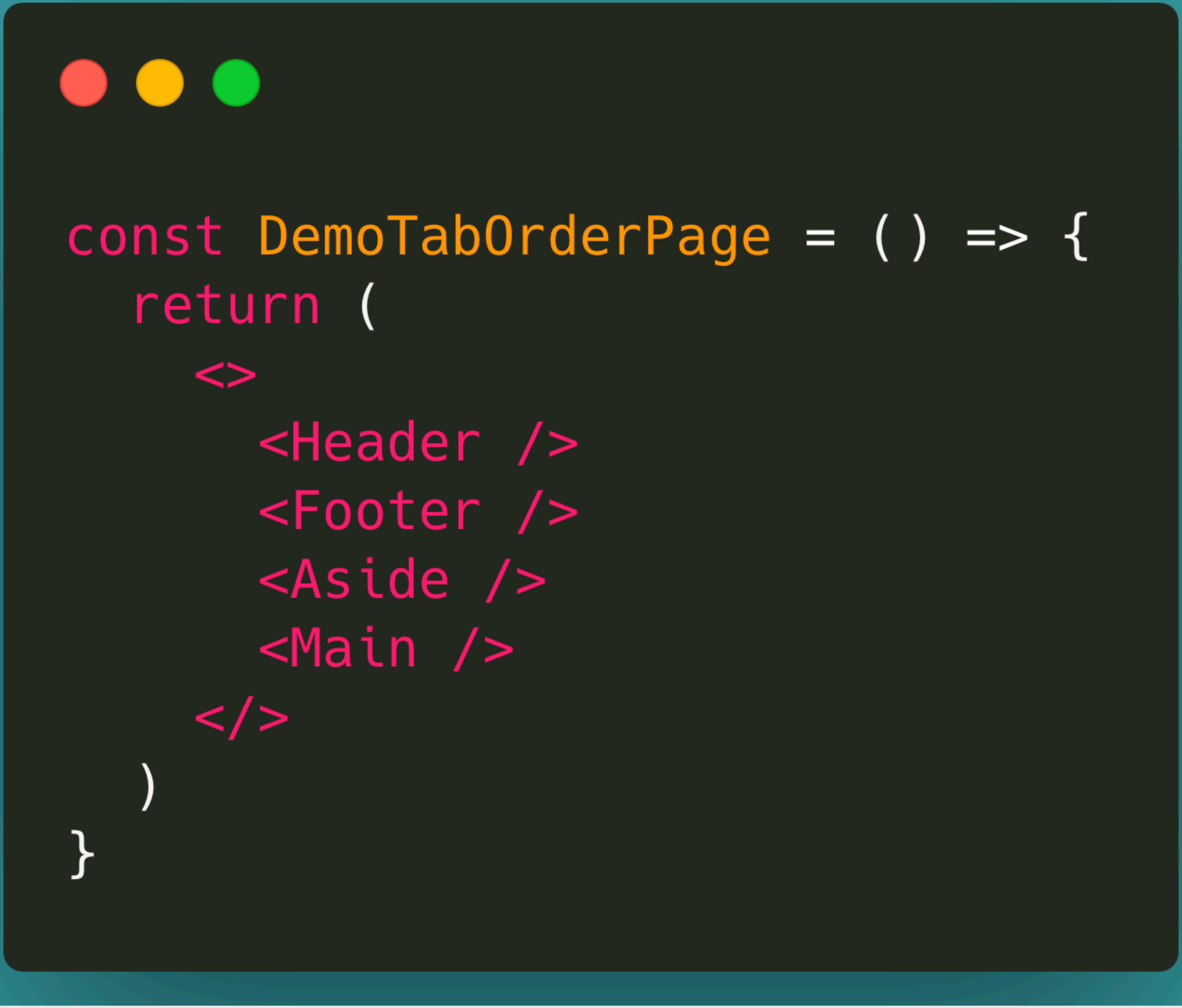
Is the tab order logical?

Are the elements in the order you expected?

## Further information

- Info Link 1
- Info Link 2
- Info Link 3

Example footer link



```
const DemoTabOrderPage = ( ) => {
  return (
    <>
      <Header />
      <Footer />
      <Aside />
      <Main />
    </>
  )
}
```

# keyboard operability

- verify you can tab to every interactive element (shift + tab to go backwards)
- check the current focus is clearly visible
- ensure the tab order is logical
- test interacting with elements

check your code works for screen reader users

❗

a reasonable minimum:

if you used an aria attribute, you should check your code with a screen reader

# screen reader / browser combos

- Windows: NVDA + Firefox
- MacOS: VoiceOver + Safari

*Based on user behaviour reported in the WebAIM 2019 Screen Reader User Survey*

# key screen reader skills

- switch on and off

⌘ + F5

# key screen reader skills

- switch on and off
- read parts of the page

ctrl + ⌥ + right/left arrow

# key screen reader skills

- switch on and off
- read parts of the page
- interact with items

ctrl + ⌥ + spacebar

# your new pre-push checklist

# pre-push checklist

# pre-push checklist

✅ refactor for semantics & heading structure

# pre-push checklist

- ✅ refactor for semantics & heading structure
- ✅ verify in axe/wave

# pre-push checklist

- ✅ refactor for semantics & heading structure
- ✅ verify in axe/wave
- ✅ check with keyboard

# pre-push checklist

- ✅ refactor for semantics & heading structure
- ✅ verify in axe/wave
- ✅ check with keyboard
- ✅ check with screen reader *

# further reading

slides + resource links

bit.ly/a11y-focused-developer

up your a11y

upyoura11y.com