Web Accessibility 101

Making Research and Geospatial Applications Inclusive

What will this session cover?

What to expect from this session

- No coding!
  - Concepts, methods, and tools that are programming-language/system agnostic
- Interactivity: anonymous polls/Q&A
  - Using Vevox
  - Can enter the poll using your laptop or mobile
  - No login/account required
- A lot to absorb
  - This is a jumping-off point
  - Links to resources provided in the slides

What to expect from this session

- A lot to absorb
  - This is a jumping-off point
  - Links to resources provided in the slides
  - Don’t expect to be an accessibility expert at the end!

- Key takeaways:
  - Web accessibility is a fundamental part of building a good web application;
  - Be open to advice, suggestions, and critique on web accessibility!

What to expect from this session

- Key words and principles for you to search later!
- Each framework is different, but hopefully this will focus you on what to search…
  - “edit CSS Rshiny app”
  - “Tab focus Python Dash app”
- Embed ideas and approaches, and let you find the specifics for your case!

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **176-296-390**

Or scan the QR code

Have you opened Vevox successfully?

Yes!

0%

Yes again!

0%

Have you opened Vevox successfully?

Yes!

0%

Yes again!

0%

Workshop content

1. What is web accessibility?
2. Where do we start?
3. How do we make web applications and maps more accessible?
4. How do we test our maps/apps for accessibility?
5. What are some hazards to be aware of?
6. Where do we learn more?

Workshop content

1. **What is web accessibility?**
2. Where do we start?
3. How do we make web applications and maps more accessible?
4. How do we test our maps/apps for accessibility?
5. What are some hazards to be aware of?
6. Where do we learn more?

Before we kick off...

- Even though we might be all using different frameworks to build web apps, the underlying logic is usually the same
- How familiar are you with each of these?

HTML, CSS, JS?

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **176-296-390**

Or scan the QR code

How familiar are you with each of these?

Completely unfamiliar — Very familiar

HTML (Hypertext Markup Language)

CSS (Cascading Style Sheets)

JS (JavaScript)

How familiar are you with each of these?

Completely unfamiliar

Very familiar

HTML (Hypertext Markup Language)

CSS (Cascading Style Sheets)

JS (JavaScript)

HTML (Hypertext Markup Language)

- Provides the structure and content of web pages
- Uses elements like `<header>`, `<nav>`, `<main>` to give meaning to content (semantic meaning)
- Screen readers and assistive technologies rely on proper HTML structure
- Like the skeleton of a wood-frame house - it defines what goes where

CSS (Cascading Style Sheets)

- Controls how content looks and is positioned
- Handles colors, fonts, spacing, layout, and responsive design
- Can enhance or hinder accessibility through color contrast, font sizes, focus indicators
- Like the plasterboard, paint, flooring and general decoration of the house

JavaScript

- Interactive layer
- Makes web pages dynamic and responsive to user actions
- Handles clicks, form submissions, animations, and real-time updates
- For accessibility, must work with keyboards, screen readers, and other assistive technologies
- Like an electrical system in your house; sometimes even like a smart-home assistant…
  - Can be very powerful, but also can be insecure!

Using Rshiny, Python etc.

- Generate HTML, CSS, and JavaScript behind the scenes
  - Output standard web technologies
- Your R or Python code creates the HTML structure, styling, and interactive behaviours
- Even though you write in R or Python, users still interact with HTML/CSS/JS in their browsers
- The same accessibility principles apply - your R Shiny or Python app must generate accessible HTML output
  - They don't *always* – we need to check!
- Sort of like a turn-key or flat-pack house…

HTML, CSS, JS - Not the only frameworks

- Many other web frameworks and tools exist (WebAssembly, WebGL, SVG, etc.)
- But! Almost all web applications ultimately rely on these three core technologies
- Accessibility best practices apply regardless of the specific technology
- What you learn about accessible HTML will help with any framework you use later

Good accessibility starts with understanding the fundamentals, not mastering every possible technology!

Some reading if you haven't used HTML and CSS

- HTML, CSS and JS for Rshiny
- HTML and CSS for Python Developers
- CSS and JavaScript accessibility best practices
- HTML Accessibility: Programming with an Inclusive Perspective

What is web accessibility?

How does web accessibility differ from other uses of the term “accessibility”?

Join the Vevox session

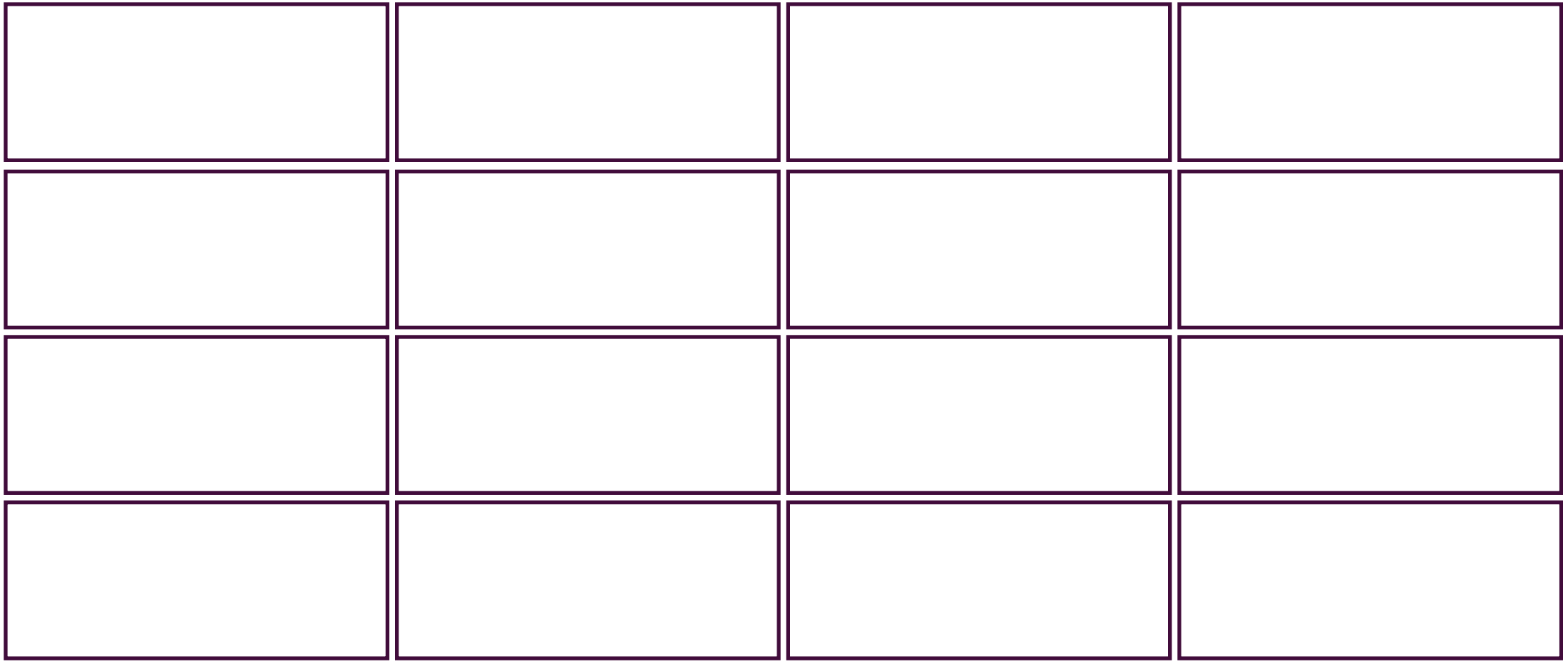
Go to **vevox.app**

Enter the session ID: **176-296-390**

Or scan the QR code

What does accessibility mean to you?

What does accessibility mean to you?



Accessibility

- FAIR Data Principles: Findable, Accessible, Interoperable, Reusable
  - Accessible: (meta)data can be accessed by a user with a standard protocol (e.g. not buried on a laptop, “available upon reasonable request”)
- Ensuring data and results can be accessed, understood, and utilised by a wide, non-specialised audience
- Broadening participation
- **Web accessibility**
  - **Ensures everyone, including people with disabilities, can perceive, understand, navigate, and interact with the web.**
  - **Precise, technical definition**

Related reading:

- WebAIM: Introduction to Web Accessibility
- FAIR Data Principles

A little bit of background

- 1 in 5 people in the UK have a long-term illness, impairment or disability. Many more have a temporary disability.
- Varied and diverse:
  - Health conditions
  - Changing abilities
  - Temporary disabilities
  - Situational limitations

Related reading:

- WebAIM: Introduction to Web Accessibility
- Web Accessibility Initiative: Diverse Abilities and Barriers

A little bit of background

- Visual (Blindness, low vision, colour-blindness)
- Auditory (Deafness, hard-of-hearing)
- Motor (response time, dexterity, fine motor control),
- Cognitive (learning disabilities, processing, memory).

Related reading:

- WebAIM: Introduction to Web Accessibility
- Web Accessibility Initiative: Diverse Abilities and Barriers

A little bit of background

- I’m not a web development expert!
- My experience:
  - Visual (Low vision, visual disturbances),
  - Motor (response time, dexterity, fine motor control),
  - Cognitive (Memory/confusion, processing, aphasia, effect of chronic pain),
  - Situational.

- Adaptations:
  - Voice notes on Whatsapp
  - Printing material in large font sizes
  - Screen-readers

Related reading:
- WebAIM: Introduction to Web Accessibility
- Web Accessibility Initiative: Diverse Abilities and Barriers

Designing with disability in mind

- Other researchers
  - Collaborators, reviewers, peers
- Interested parties and collaborators
  - Local authorities, councils, police force, medical staff
- General public
- Students

Who are you excluding by not intentionally designing for disability?

What are you saying about the role of disabled people in society?

Workshop content

1. What is web accessibility?
2. **Where do we start?**
3. How do we make web applications and maps more accessible?
4. How do we test our maps/apps for accessibility?
5. What are some hazards to be aware of?
6. Where do we learn more?

Where do we start?

The web is an incredible resource that is ever growing; how do we leverage it in an accessible way?

Web Content Accessibility Guidelines (WCAG)

- An international set of guidelines – the web accessibility Bible
- A useful checklist when building a website
- A great resource to reference when trying to decide the best way to implement something
- Used by many governments as a required standard

"A developer can learn the basics of web accessibility in just a few days, but, as with any technical skill, it often takes months to internalize the mindset as well as the techniques."

WebAim, Introduction to Web Accessibility

Related reading:

- WebAIM: Introduction to Web Accessibility
- Web Accessibility Initiative: Diverse Abilities and Barriers

POUR: the 4 principles behind WCAG

- **Perceivable:** can the information be accessed via the browser or assistive technologies?
- **Operable:** can users interact with all elements equally whether using a mouse, the keyboard, or an assistive device?
- **Understandable:** is the content clear, equally understandable whether a user is using assistive technologies, and does it limit confusion and ambiguity?
- **Robust:** does the website work for a wide range of technologies, instead of relying on very specific tech?

Related reading:

- WebAIM: Introduction to Web Accessibility
- Web Accessibility Initiative: Diverse Abilities and Barriers

What are we aiming for?

- For users to get a **full experience** from our website
  - Regardless of whether they are using a vanilla browser, or assistive technology
  - Regardless of whether they are navigating with a mouse, the keyboard, or other assistive technology
- For our website to be **easy to use** and **enjoyable to use**
- To meet all WCAG AA standards at minimum

Workshop content

1. What is web accessibility?
2. Where do we start?
3. **How do we make web applications and maps more accessible?**
4. How do we test our maps/apps for accessibility?
5. What are some hazards to be aware of?
6. Where do we learn more?

How do we make apps and maps more accessible?

How do we ensure the POUR principles are met?

Making sure your content is **perceivable**

- Sizing
- Contrast and colours
- Labels
- Fonts
- Zoomability
- Alt-text
- Screen-reader friendly
- Alternative data formats

Target size

- Ensure controls meet a minimum size/have sufficient spacing so that they are clickable
- Need to be at least 24 X 24 CSS pixels
- Think about touch/mobile use:
  - Touch screens can be an accessibility adjustment
  - It’s more difficult to be precise using a touch screen than a mouse

Contrast

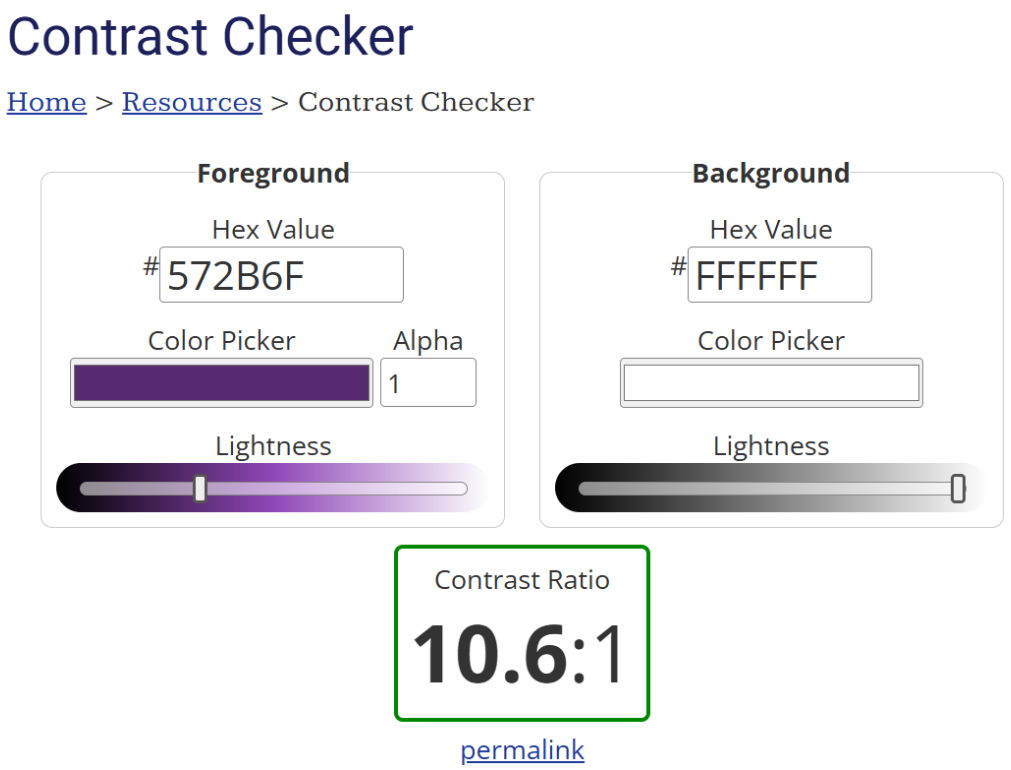
- Ensure sufficient contrast between text/buttons/symbols and the background
- Aim for an absolute minimum contrast of 4.5:1

4.5:1

4.5:1

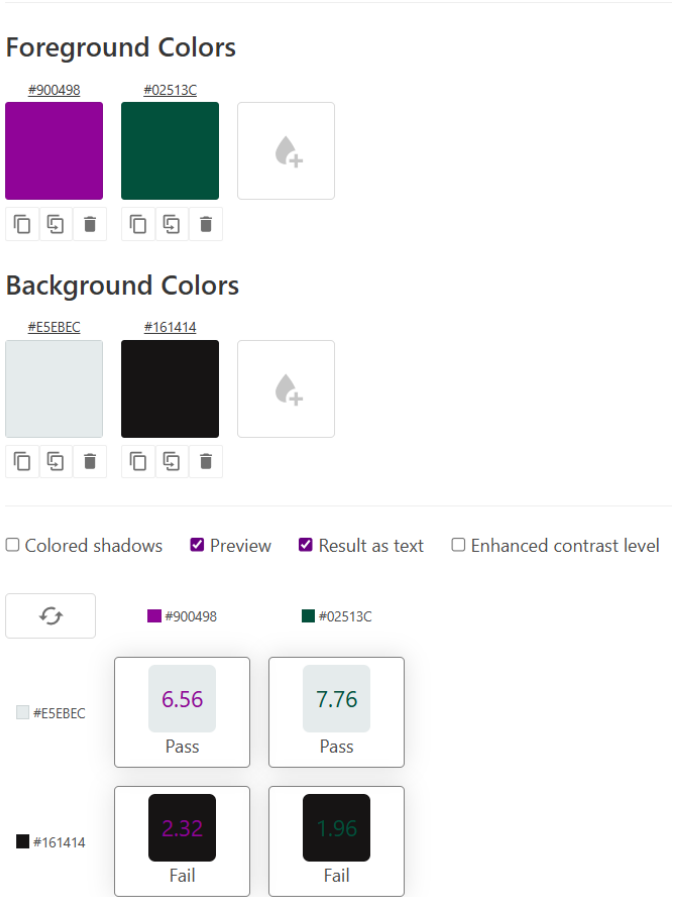
WCAG AA: Pass ☑
WCAG AAA: Fail ☒

Tools for contrast



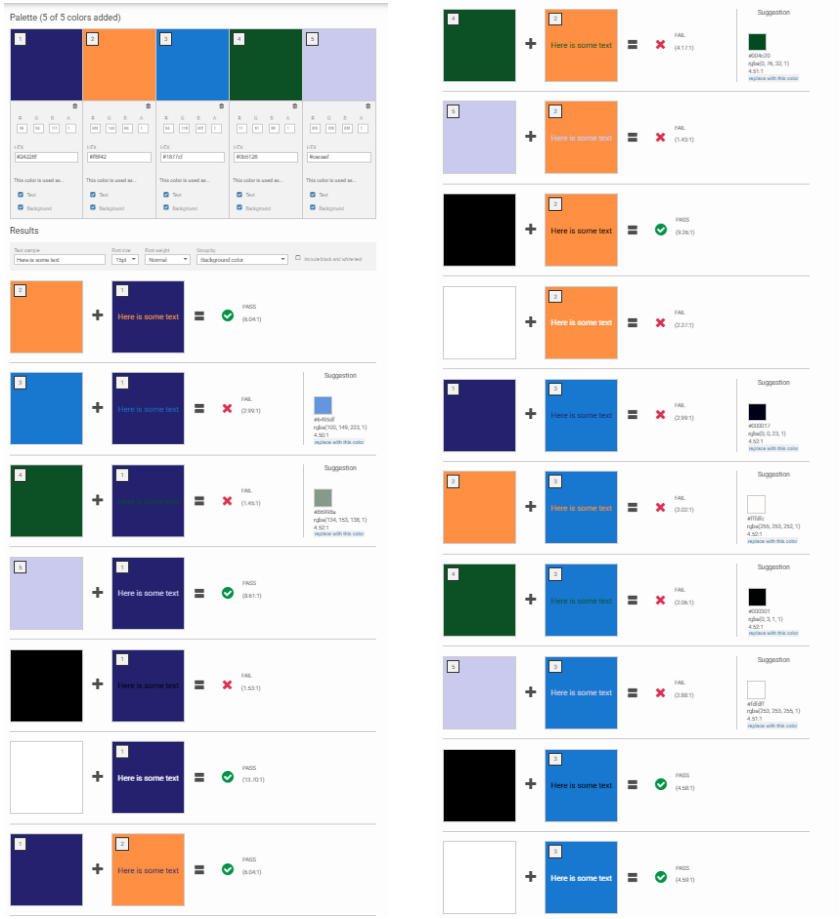
- https://webaim.org/resources/contrastchecker/
- Compare two colours (foreground and background)
- Check against WCAG criteria for normal and large text, and graphical user interface components
- Very well designed, accessible site

Tools for contrast



- https://multiple-contrast-checker.netlify.app/
- Useful for checking selections of colours
- Doesn't give you the same feedback on WCAG AA or AAA guidelines
- Website text/UI is low-contrast

Tools for contrast



- https://color-contrast-checker.deque.com/
- Allows much more complex comparisons (of every colour with every other colour)
- Suggests replacement colours for better contrast
- A bit difficult to see all the comparisons: can be overwhelming

Contrast on maps

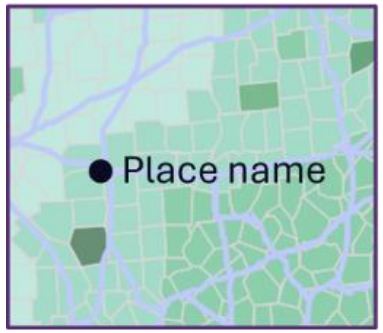
- Treat labels, boundaries, lines as text when looking at contrast ratios (e.g. requiring high contrast)
  - Background colours underneath labels are often varied and not constant
  - Using halos/masks can make labels more legible
- More discussion on typography later

Related reading:

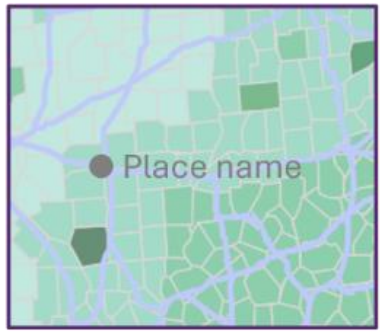
- 11 Ways to Enhance Map Clarity with Strategic Labeling

Which do you think are the WORST labels? Pick up to three

A



B



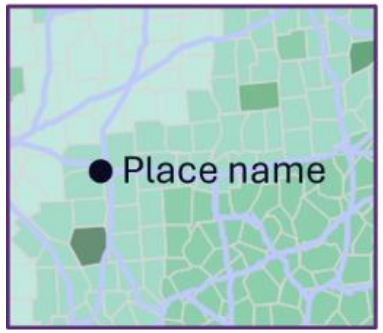
C

D

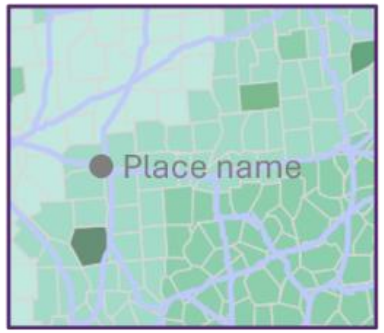
E

Which do you think are the WORST labels? Pick up to three

A



B



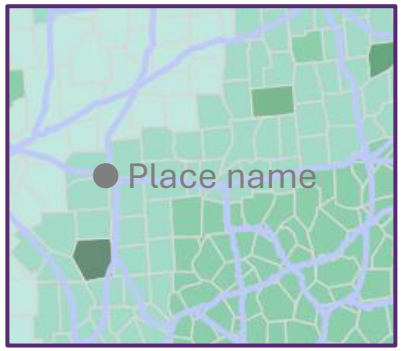
C

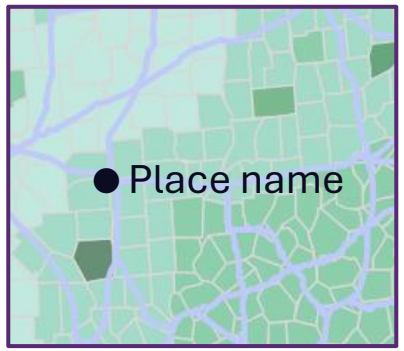
D

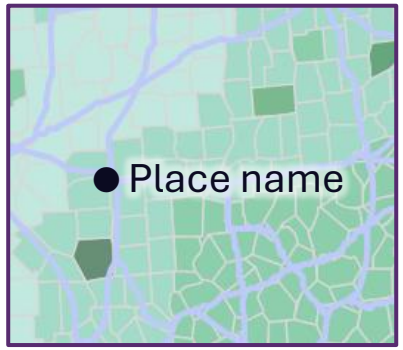
E

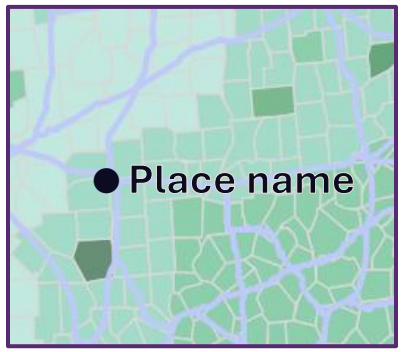
Contrast on maps

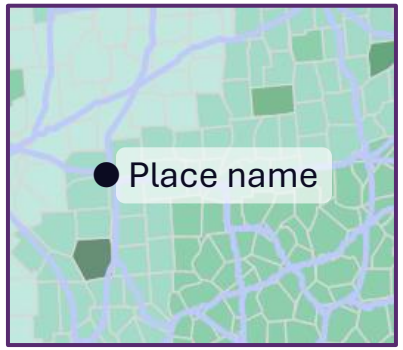
- Treat labels, boundaries, lines as text when looking at contrast ratios (e.g. requiring high contrast)
  - Background colours underneath labels are often varied and not constant
  - Using halos/masks can make labels more legible
- More discussion on typography later











Related reading:

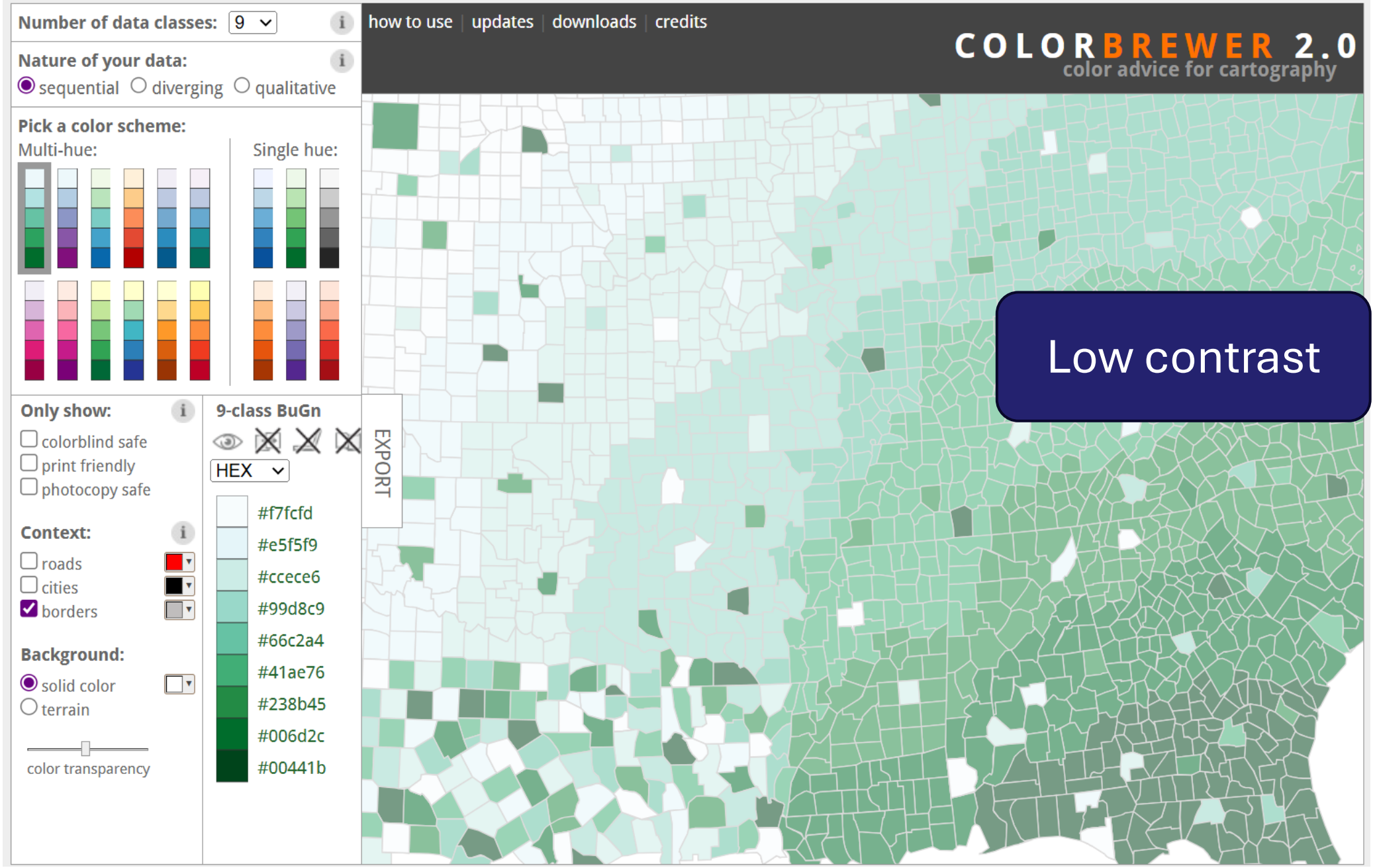
- 11 Ways to Enhance Map Clarity with Strategic Labeling

Contrast on maps

- Colour being used to denote category:
  - Should contrast with lines/markers/labels
  - Should be differentiable from other regions
  - Use an alternative source of information: not *just* colour
    - Labels
    - Symbols
    - Tooltips/popups
- If feasible, provide different basemap options or different colour options
  - No one-size fits all!

Related reading:

- Designing a High Contrast Base Map for Users with Low Vision
- 11 Techniques for Making Maps More Accessible That Enhance Readability


Number of data classes: 9
Nature of your data:
sequential diverging qualitative
Pick a color scheme:
Multi-hue:
Single hue:
Only show:
colorblind safe
print friendly
photocopy safe
Context:
roads
cities
borders
Background:
solid color
terrain
color transparency
9-class BuGn
HEX
#f7fcfd
#e5f5f9
#ccece6
#99d8c9
#66c2a4
#41ae76
#238b45
#006d2c
#00441b
EXPORT
how to use | updates | downloads | credits
COLORBREWER 2.0
color advice for cartography
Low contrast


Number of data classes: 9
Nature of your data:
sequential diverging qualitative
Pick a color scheme:
Multi-hue:
Single hue:
Only show:
colorblind safe
print friendly
photocopy safe
Context:
roads
cities
borders
Background:
solid color
terrain
color transparency
9-class BuGn
HEX
#f7fcfd
#e5f5f9
#ccece6
#99d8c9
#66c2a4
#41ae76
#238b45
#006d2c
#00441b
EXPORT
how to use | updates | downloads | credits
COLORBREWER 2.0
color advice for cartography
Low contrast

Tools for contrast on maps

- https://colorbrewer2.org/ - allows you to pick different colour maps for different data class values
  - Lets you pick terrible options too – ensure to test colour choices with a contrast checker
- https://datavizcontrast.com/ - allows you to find contrast levels for different line thicknesses
  - Useful for choosing colours for linear features
- https://projects.susielu.com/viz-palette - Shows different plots of colours side-by-side with different filters
  - Great for maps and for other visualisations

PICK

Use Chroma.js

Use Colorgorical

Use ColorBrewer

Add

Replace

EDIT

7 Colors

Add

#hex rgb

hsl

1 #ffd700

2 #ffb14e

3 #fa8775

4 #ea5f94

5 #cd34b5

6 #9d02d7

7 #0000ff

GET

String quotes

Object with metadata

#hex rgb

```
["#ffd700",
"#ffb14e",
"#fa8775",
"#ea5f94",
"#cd34b5",
```

Color Population:

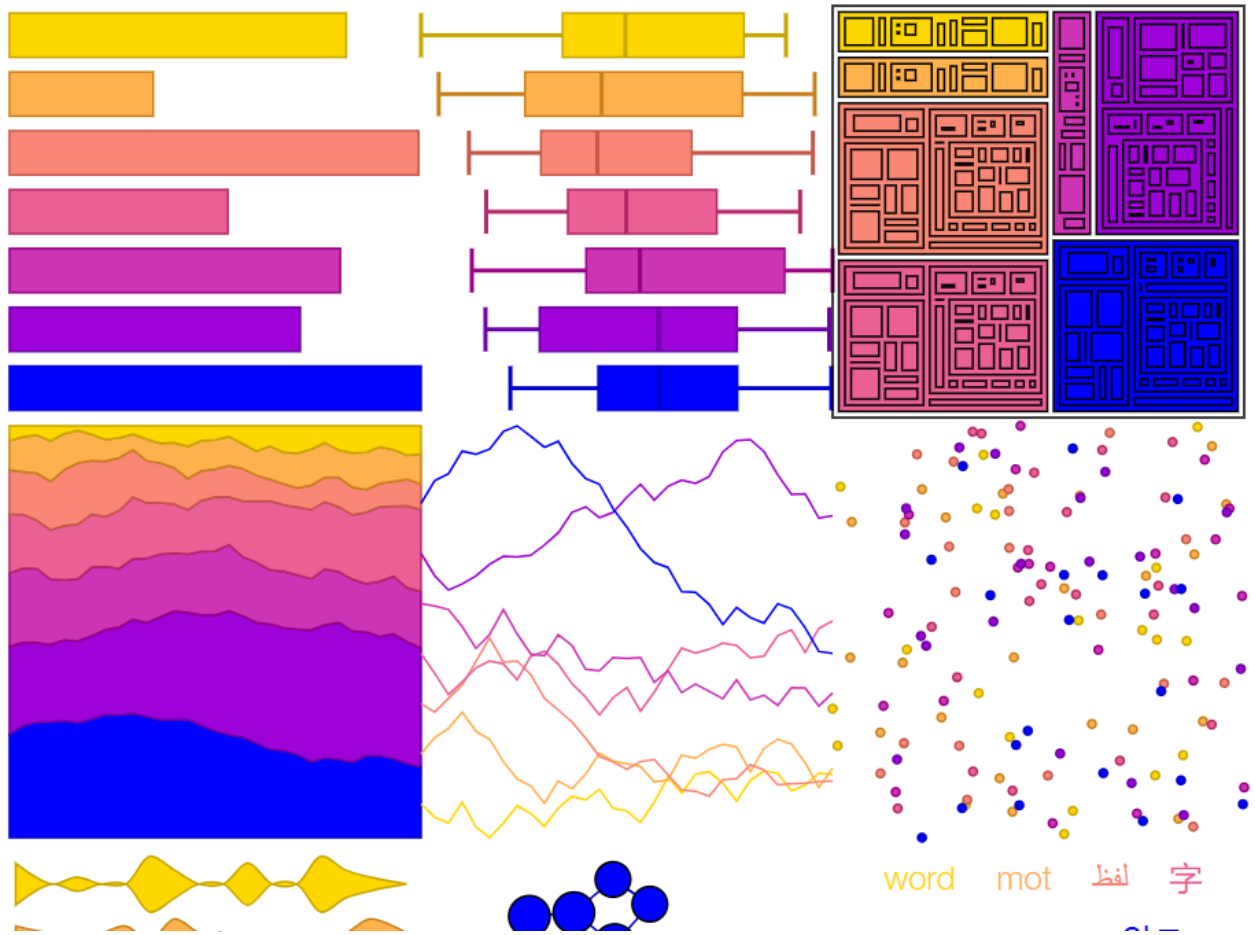
No Color Deficiency - 96% | Deuteranomaly - 2.7% | Protanomaly - 0.66%

Protanopia - 0.59% | Deuteranopia - 0.56% | Greyscale

Sample font

Randomize Data

Stroke: Dark None

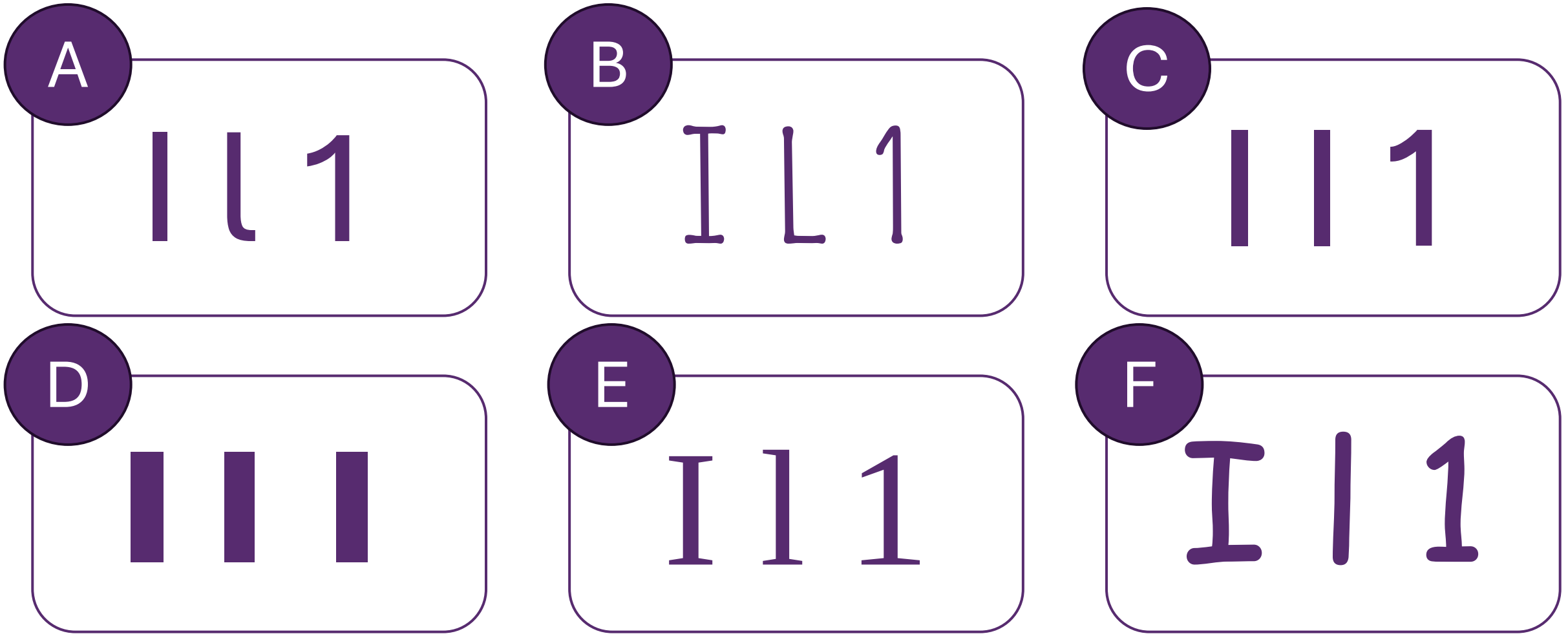


Fonts

- Simple, familiar, easy-to-read
  - Avoid fonts with character complexity or ambiguity
- Limited number of fonts
- Disagreement over serif vs. sans serif
- Minimum ~16 px
- Using system fonts
  - Gives the user control/flexibility
  - Does mean you have a less control over appearance

Related reading:

- Typefaces and Fonts
- Accessible fonts and readability: the basics


A
l l 1
B
I L 1
C
I I 1
D
E
I l 1
F
I I 1

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **176-296-390**

Or scan the QR code

Which font is the most legible in this example?

A

B

C

D

E

F

Which font is the most legible in this example?

A

B

C

D

E

F

A

A long time ago in a galaxy far, far away...

B

A long time ago in a galaxy far, far away...

C

A long time ago in a galaxy far, far away...

D

A long time ago in a galaxy far, far away...

E

A long time ago in a galaxy far, far away…

F

A long time ago in a galaxy far, far away...

Fonts

- Different fonts work best in different settings…
- What if we let the user choose their own?

Fonts - advanced

- Different ways to set font size on the web: rem, em, px…
- Use CSS to set fonts:
  - Can use CSS in a separate `.css` file, or can use *inline* css
  - Can be accessed in plain html, or via various different frameworks in Python, R, etc.
- Em vs. rem vs px?
  - px – exact pixel size of font; overrides browser settings
  - em – scale in relation to the font size of their parent element; can become complex with nesting
  - rem – scale relative to the **root element**'s font size

Related reading:

- How to use rem units in CSS for accessible design

Fonts - advanced

- rem units (root-em):
  - Users can specify a default font size in their browser, and elements that use rem units will respect this
  - A good idea for accessibility!

- Pixel units: will override user's default settings
  - Not good for accessibility!
  - Breaks zooming on desktop!

Related reading:

- How to use rem units in CSS for accessible design

Zoomability

- By using rem units for fonts, you make the page more zoomable
  - Pages *not breaking* when zoomed up to 200% is a key accessibility feature
- Test zooming in and out of your website/webapp
  - Check mobile and desktop, zoom can be inconsistent!
- Raise issues if your favourite framework makes it very difficult to zoom!

Related reading:

- How to use rem units in CSS for accessible design

Font Size Accessibility Examples

Using pixel sizing

This paragraph uses a size 12px font. This fails basic WCAG requirements for being too small. Additionally, it cannot be changed by the user via browser font-size settings.

While this paragraph uses a size 16px font (as recommended in WCAG), because it uses pixel sizing it will override the users' font settings.

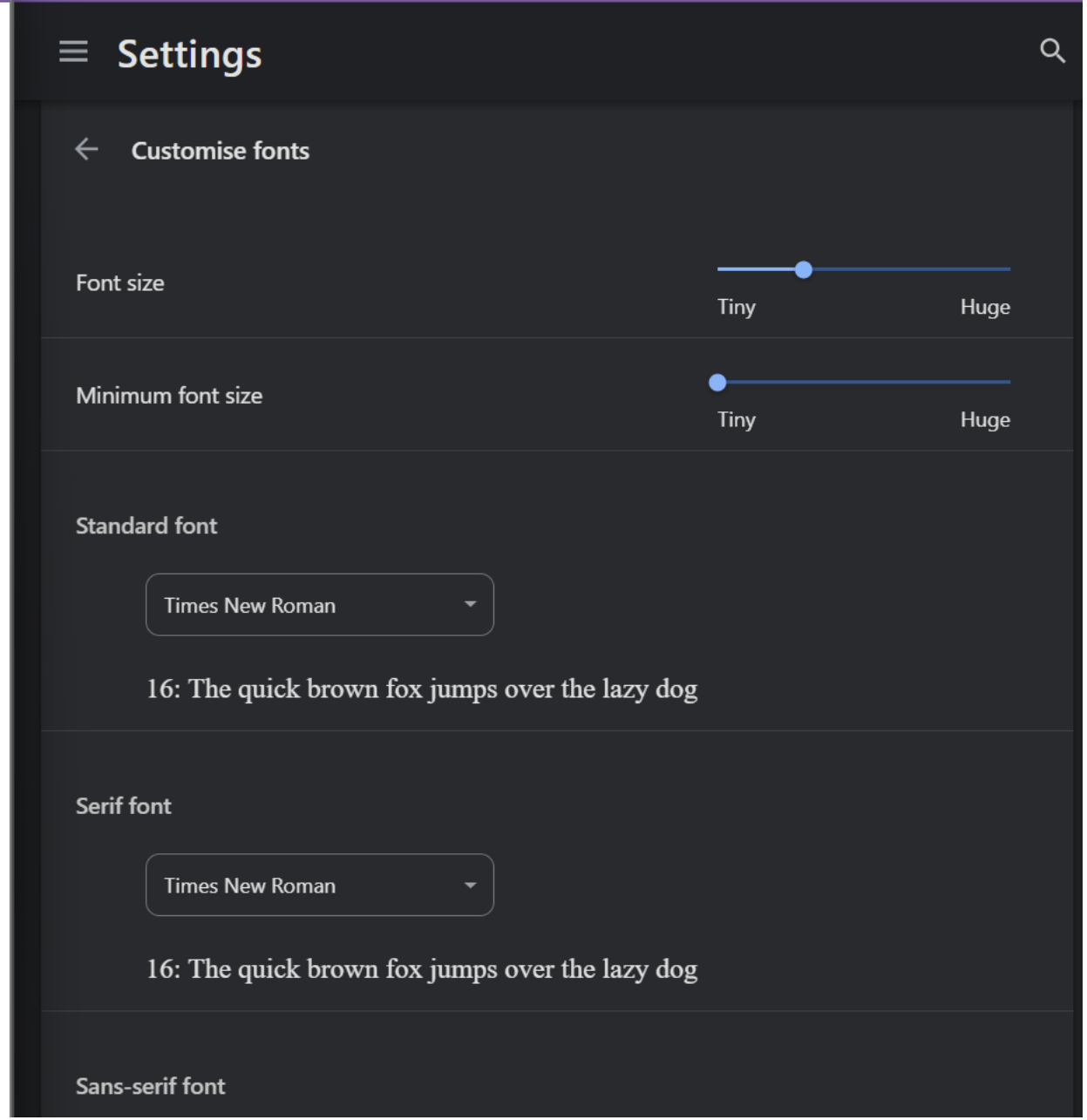
This paragraph uses a 20px font size. While larger font sizes can be useful and improve readability, it is better to leave this to users to decide for themselves.

Using rem sizing

This paragraph uses a 0.9rem font size, which makes it a little smaller than the default browser fontsize. It's usually not a good idea to go below the default fontsize.

While this paragraph uses a 1rem font size, meaning it is set to the default browser fontsize.

This paragraph uses a 1.5rem fontsize. Larger fonts can be used to highlight short snippets of text (like in headings, pull quotes, etc.).



Demo
See source on GitHub to see how to set css variables

Alt text

- Ensure all visual elements such as images have alternative text
- Videos should also have visual descriptions
- But what should we add to a zoomable map that has a lot of complex information?

Take a few minutes to brainstorm ways of making interactive maps more accessible with alternative text

Example: https://leafletjs.com/examples/choropleth/example.html

Example 2: https://www.datawrapper.de/maps/choropleth-map

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **176-296-390**

Or scan the QR code

How might you facilitate alt-text in an interactive map?

How might you facilitate alt-text in an interactive map?

Alt text for maps or charts

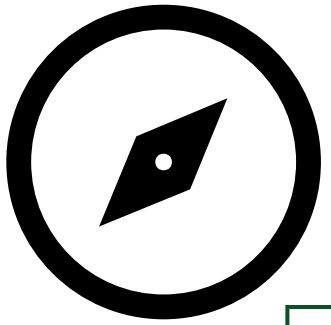
- Providing a more extensive detailed description that explains key points and correlations
- If you were to describe the plot/chart/map over the phone (or on a science podcast!) how would you describe it?
  - What are the key take-home messages?
- While alt-text for images should be brief, for complex visualisations you need more information!
- Keep description neutral and objective
- Provide alternative formats…

Related reading:

- Text descriptions for data visualisations
- Making analytical publications accessible
- Best practice for writing alternative text for complex images

Alt text for maps or charts

- Ensure you communicate the meaning of symbols as opposed to purely describing their appearance
  - “Circle, with a diamond in the centre rotated by 45 degrees from the vertical” vs.
  - “Cartoon compass symbol”
- Consider cognitive load
- Edit for clarity and test with users
- Organise descriptions sensibly:
  - Start with general, broad-scale descriptions
  - Group information logically



Related reading:

- Text descriptions for data visualisations
- Making analytical publications accessible
- Best practice for writing alternative text for complex images

Alt text for maps or charts

- Practical approach: how do we implement?
- For simple images, we use the alt-text HTML tag
- There is no single correct approach:
  - https://www.w3.org/WAI/tutorials/images/complex/
  - https://accessibility.blog.gov.uk/2023/04/13/text-descriptions-for-data-visualisations/
- Sometimes we need to use HTML tags; sometimes it is better to put it in the body text

Screen reader friendly websites

- In addition to alt-text, there are other places to add text descriptions
- Buttons, text inputs etc. have “label” attributes which tell the user what the element is for
  - Use them!
  - Search “accessible label for…” and then the name of the element in your chosen framework
- Navigate your website with a screen reader: while you won’t be able to “emulate” being a screen-reader-user, it’s an important part of testing

Alternative formats

- Alongside your maps or other visual displays, offer an equivalent alternative format
- Downloadable tables of data
- Summary statistics
- Detailed data description
  - Think of the results, discussion, and conclusion section of a paper
- Alternative base maps (to allow for higher contrast)
- Download of map as an image/pdf for printing/zooming
- Should an alternative format replace our chart as the primary format?

Recap: make your content **perceivable**

- Contrast and colours
- Labels
- Fonts
- Zoomability
- Alt-text
- Alternative data formats

Making sure your content is **operable**

- Keyboard navigation
- Avoid rogue custom buttons
- Semantic HTML

Keyboard navigation

- Lots of users navigate using a keyboard or assistive tech that interacts with the web like a keyboard
  - Speed and efficiency
  - Motor-skills and accuracy
  - Fatigue
- Need to ensure that all the same functionality that can be accessed with a mouse is available to users via a keyboard

Exercise on tab order

- Visit content here: https://murphyqm.github.io/foss4g-uk-2025/accessibility.html
- Explore the page by clicking and by tabbing
- Which cause issues?
  - After the course, you can have a look at the underlying code

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **176-296-390**

Or scan the QR code

Which elements are BAD?

Element 1 ##.##%

Element 2 ##.##%

Element 3 ##.##%

Element 4 ##.##%

Element 5 ##.##%

Element 6 ##.##%

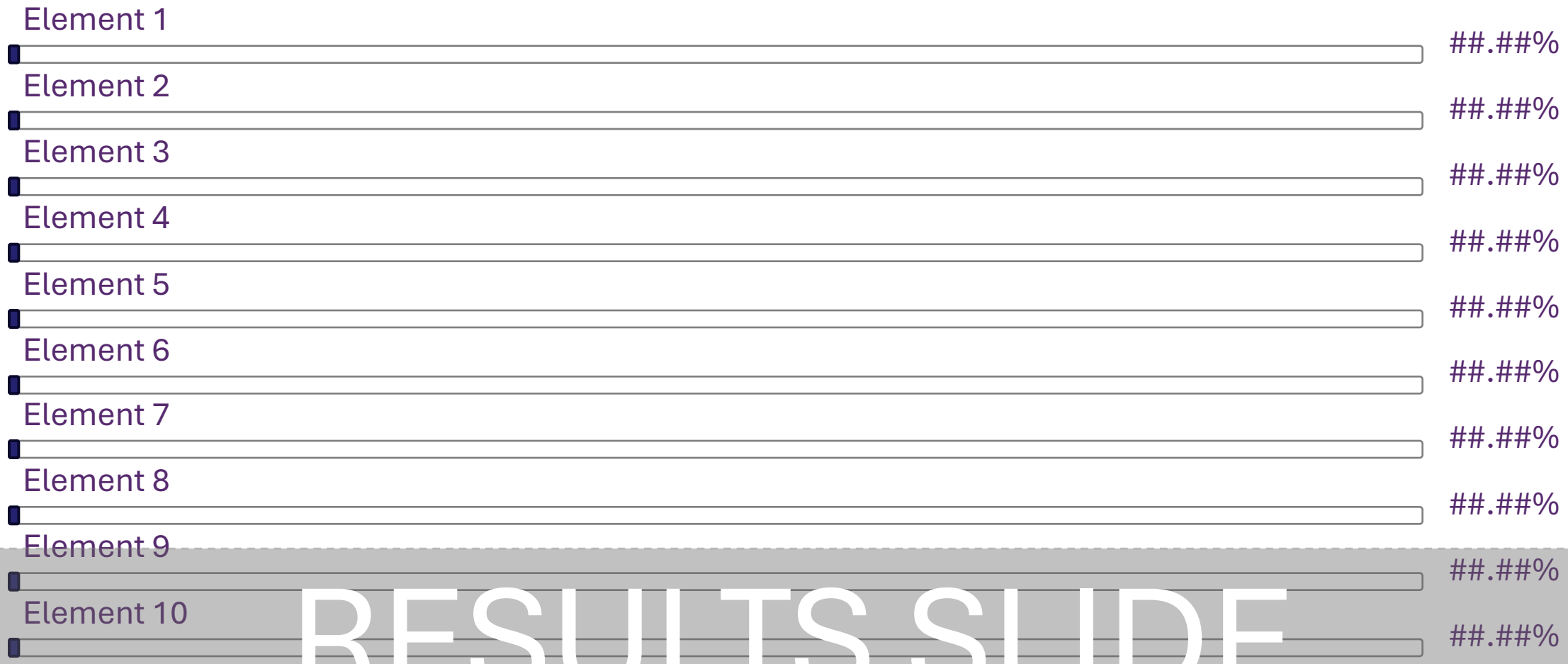
Element 7 ##.##%

Element 8 ##.##%

Element 9 ##.##%

Element 10 ##.##%

Which elements are BAD?



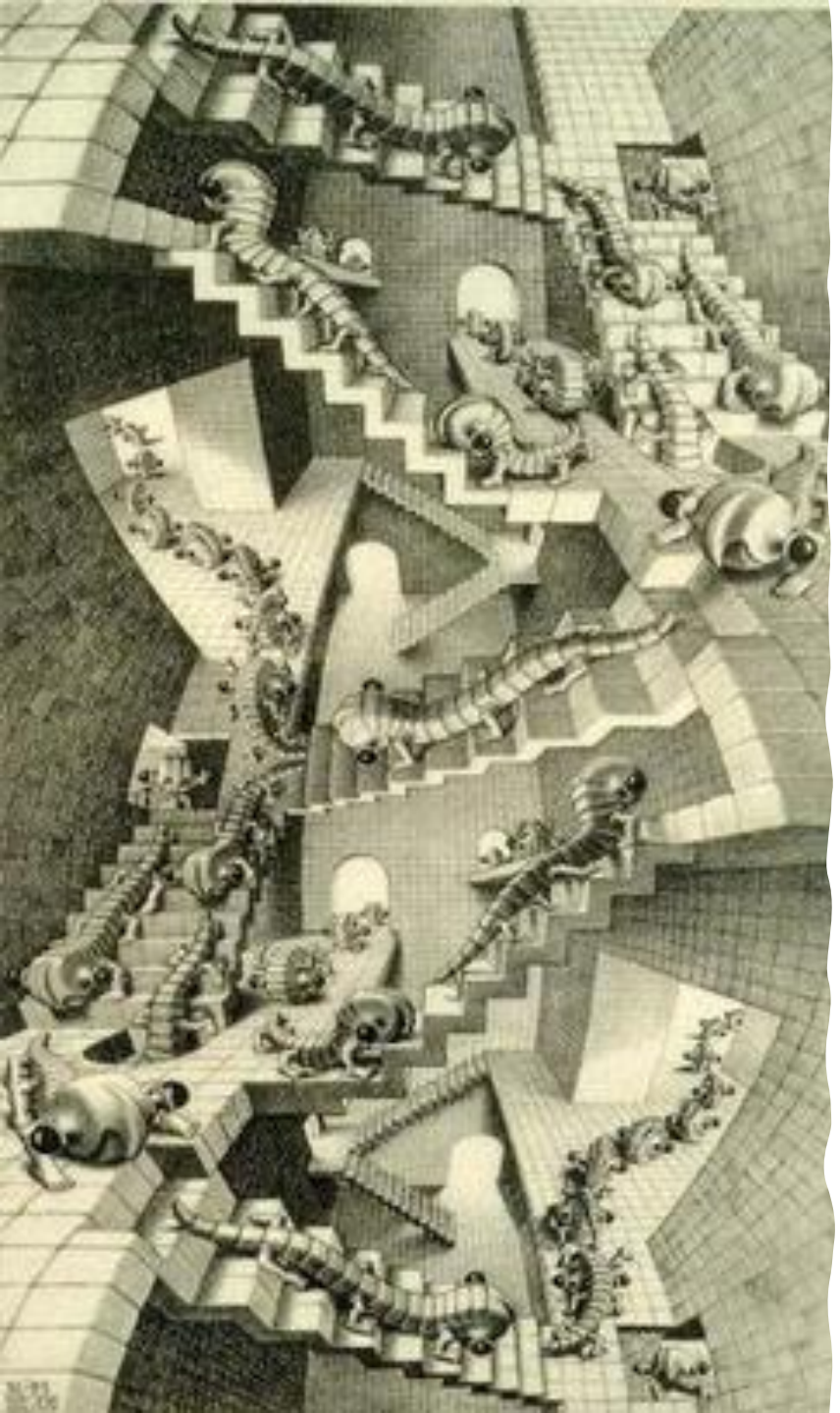
Keyboard navigation

- Do the “no mouse” test on web applications
  - Note any buttons/interactivity that you have issue with
- Use standard buttons/elements wherever possible
  - Avoid rogue custom buttons!
  - HTML elements have been designed with accessibility in mind
  - You can style them how you wish with CSS
  - Using span or div elements as custom buttons often leads to accessibility issues
- Check that map elements can be focused with the keyboard

Keyboard navigation

- Skip links help users navigate busy content
  - Often at the top of pages to skip navigation
  - Hidden from mouse interaction, and only show when using the keyboard
  - Jump to the start of the “main” content
- If you have a map with many, many interactive and focusable elements, this may also be somewhere for a skip link
  - See https://murphyqm.github.io/foss4g-uk-2025/accessibility.html

```
<div style="outline:none" tabindex="-1">
  <a href="#main-content" class="page__content-skip-link">Skip to main content</a>
</div>
```



Semantic html

- HTML is the skeleton of the house
- Trying to navigate a website with poorly structured html is like trying to make your way through an Escher painting or a funny house filled with warped mirrors!
  - Especially when using a screen reader or other assistive technology!
- Need semantic html to avoid this

Fair use, https://en.wikipedia.org/w/index.php?curid=3475224

Semantic HTML

- Semantic elements are elements with **defined meaning**
- Semantic elements:
  - <img>
  - <h1>
  - <button>
  - <select>
  - <header>
  - <nav>
- Non-semantic elements:
  -
  -

Related reading:

- HTML Semantic Elements
- Semantic HTML

Semantic HTML

- Use pre-designed, standard buttons
  - Users know what to expect, how to interact with them
  - Less likely to fail basic accessibility guidelines
- Ensure that you assign labels where required
- Check that your Python or R code is outputting sensible HTML
  - Sometimes there can be issues with inputs/interactive elements

Making sure your content is **understandable**

- Semantic HTML
- Logical choices for design
- Sensible alt-text
- Alternative formats
- Link names

We've already solved a lot of these issues in the P and O sections!

Semantic HTML

- Use heading levels as actual heading levels, not aesthetic choices!
  - Use H1, H2, H3 etc. in order to denote main headings, sub-, and sub-sub-headings
  - Use CSS to style text for emphasis, *not* heading tags!
- Organise the flow of your content sensibly
  - Use main, body, aside etc. tags
  - If designing on a grid, make sure the html order makes sense (as opposed to only making sense when arranged with CSS)

Sensible alt-text

- Outside of purely technical descriptions, alt text should also help the user understand the topic
  - Describe the patterns seen – what are the key take-aways
- Instead of focusing on aesthetic choices, ensure the message is understandable
- Use unambiguous, clear language

Alternative formats

- We briefly covered this before, but…
- Is an image or graphic really the best choice?
- Would a more accessible alternative actually be better?
- Can a user fully understand the content from an alternative format?

Link names

- Link text should make sense when read out of context
  - Screen reader or keyboard navigation users will often skip from link to link or experience all links on the page in a list format
- Examples to avoid (underlined words are the link)
  - Read more
  - Click here to read our docs
  - Studies show that x, y, z…
  - Read through some examples
- Better links:
  - Read more about X
  - Read our docs
  - Browse examples of x, y, z…
  - Studies show x, y, z

Making sure your content is **robust**

- Custom functions vs. semantic HTML
- Does it require a special piece of tech to work?

Avoiding very custom solutions

- Custom-built workarounds to accessibility issues can cause problems if not well-maintained
- Relying on well-established standards is a better idea!
  - Semantic HTML
  - Applications that have accessibility statements/docs/tutorials
- Try to build for both mobile and desktop where possible
  - Don’t assume only small screens are touch-operated

Make sure the content works in different browsers

- When designing for web, ensure you test the webapp on different browsers
- Just because there is a market lead in a specific browser, it doesn't mean that this is reflective of the disabled community

Browser	User share
Chrome	66.45%
Safari	16.61%
Edge	5.35%
Firefox	2.52%
Samsung Internet	2.21%
Opera	2.09%
Android	1.31%
UC Browser	0.9%
Brave	0.85%
Other	1.72%

POUR: the 4 principles behind WCAG

- **Perceivable:** can the information be accessed via the browser or assistive technologies?
- **Operable:** can users interact with all elements equally whether using a mouse, the keyboard, or an assistive device?
- **Understandable:** is the content clear, equally understandable whether a user is using assistive technologies, and does it limit confusion and ambiguity?
- **Robust:** does the website work for a wide range of technologies, instead of relying on very specific tech?

Related reading:

- WebAIM: Introduction to Web Accessibility
- Web Accessibility Initiative: Diverse Abilities and Barriers

Search for accessibility docs for your platform

- Mapping libraries:
  - https://leafletjs.com/examples/accessibility/
- Accessibility testing:
  - https://github.com/dequelabs/axe-core

Workshop content

1. What is web accessibility?
2. Where do we start?
3. How do we make web applications and maps more accessible?
4. **How do we test our maps/apps for accessibility?**
5. What are some hazards to be aware of?
6. Where do we learn more?

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **176-296-390**

Or scan the QR code

How would you test a web app for accessibility?

How would you test a web app for accessibility?

Testing web apps for accessibility

- Use a checklist
  - https://webaim.org/standards/wcag/checklist
  - https://www.digitala11y.com/wcag-checklist/
- Do a no-mouse navigation
  - Check that all functionality is accessible via keyboard
- Explore the page with a screen-reader
  - Only useful if you put the time in to learning how to use a screen reader correctly

Testing tools

- Web accessibility tools:
  - WAVE: https://wave.webaim.org/
  - Example 1: (really good example) https://wave.webaim.org/report#/https://www.gov.uk/guidance/accessibility-monitoring-how-we-test
  - Example 2: (some issues) https://murphyqm.github.io/foss4g-uk-2025/
- Exercise: try to find all the issues with this page
  - Do you manage to find anything that the automated testing missed?
  - https://murphyqm.github.io/foss4g-uk-2025/accessibility/html/bad-design.html
  - https://wave.webaim.org/report#/https://murphyqm.github.io/foss4g-uk-2025/accessibility/html/bad-design.html

Browser Tools

- Firefox and Chrome have extensive developer tools available, which are very useful for accessibility testing
- In Chrome, can emulate phone screens/different screen sizes
  - Can view accessibility tree
  - Can dig into issues raised by WAVE
  - Very useful if you haven't written the HTML (using Rshiny etc.)

Testing web apps for accessibility

- For large-scale, important web applications
  - Hire a professional, disabled, accessibility tester!
  - Simulating disability is not successful or accurate

Workshop content

1. What is web accessibility?
2. Where do we start?
3. How do we make web applications and maps more accessible?
4. How do we test our maps/apps for accessibility?
5. **What are some hazards to be aware of?**
6. Where do we learn more?

Not listening to feedback

- Accessibility is a continuous process, not a destination
- Best practices evolve
- There is no one-size-fits-all
- There will **always** be something to improve
- Welcome feedback!
  - Feedback from one user struggling to use your website is more valuable than feedback/compliments from 50 people who find it easy!

Accessibility overlays: avoid!

- Plugins/add-ons that purport to fix accessibility issues
- Widely disparaged by the disabled community:
  - Don‘t fix basic underlying issues (contrast, messy html, lack of alt-text)
  - Add an additional layer of complication to the website, especially when trying to use assistive technology
- Often centred around compliance instead of user needs
- Broadcasts that accessibility was an afterthought

Related reading:

- Should I Use An Accessibility Overlay?
- Accessibility Overlay Widgets Attract Lawsuits

AI accessibility checker tools

- Do not rely on these

Perfectionism

- There is a LOT of work involved in making the web accessible
  - The same way there is a LOT of work in making research open and reproducible
  - It is absolutely worth the effort
- Don’t let overwhelm make you bury your head in the sand!
- A lot of us are “accidental” web developers without formal training
- Keep “P. O. U. R.” on a sticky note on your monitor
- The same way you spell-check your work, check through the basics

But know when you're out of your depth

- If a key planned output of your work is a dashboard…
  - That you hope will be widely used by the general public
  - That will deliver important information
  - maybe you *do* need someone with formal training in this area!
- Hire web developers with public-service experience
- Hire disabled web developers
- Hire disabled web accessibility auditors

Incorrectly-used ARIA labels

- ARIA or Accessible Rich Internet Applications add extra detail to websites, allowing for improved accessibility
- Complex to implement correctly unless you actively use them to navigate
- No ARIA – better than bad ARIA
- Causes confusion and makes the page unreadable

Related reading:
- ARIA guides

Workshop content

1. What is web accessibility?
2. Where do we start?
3. How do we make web applications and maps more accessible?
4. How do we test our maps/apps for accessibility?
5. What are some hazards to be aware of?
6. **Where do we learn more?**

Tutorials

- Mozilla Accessibility docs: in general, the MDN docs are a fantastic resource; the accessibility docs are no different
- Web Accessibility Initiative: these tutorials are a great step-by-step guide to implementing various accessible patterns
- Automating Accessibility: *How To Test for the 6 Most Common Accessibility Issues on Home Pages*

General resources

- UK Gov web design standards
- UK Gov accessibility strategy
- Web Content Accessibility Guidelines (WCAG) 2.2

Blogs/accounts

- Bluesky accounts:
  - Accessibility Awareness: great random snippets that you learn a lot from!
- Blogs:
  - Sarah L. Fossheim

This is not an endorsement of any of these accounts/views shared!

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **176-296-390**

Or scan the QR code

Question slide

What's one takeaway from today?

What's one takeaway from today?