Accessibility Guide for Interactive Web Maps

November 2019

Contents

Section 1: Summary and Key Take-aways

Summary

This document provides interactive web map designers and developers strategic and tactical with guidance on accessibility best practices and standards for custom-built interactive web maps.

Interactive web maps allow a user to interact with a map using a mouse, keyboard, voice, or other mechanism, to select areas of interest, enter coordinates, or toggle layers on and off. The user can typically zoom in and out, and pan around the map while data changes. Interactive web maps can be static or complex in nature.

The map's purpose and content determine whether or not it should be an interactive web map, and if so, drives the requirements to make an interactive web map accessible. What works for one map may not work for another. When creating an interactive web map, ensure that all users can access its content.

By considering accessibility from the start, an interactive web map has a better opportunity to serve all audiences. Creating accessible information should not be an exception to the rule. It should be there when people need it, not by request.

The requirements to make an interactive web map accessible are driven by the purpose and content of the map itself. The "Accessible Examples" section contains examples of how accessible components differ between each map, and how they are based on the subject matter and functionality built into the interactive web mapping application.

State of Minnesota Statute and United States Federal Law

By statute, the State of Minnesota's accessibility standard references the Federal Section 508 and World Wide Web Consortium's Web Content Accessibility Guidelines (WCAG) 2.0, levels A and AA. Section 508 is driven by a federal law that requires electronic and information technology that is developed, procured, maintained, or used by the federal government to be accessible to people with disabilities.

Key Take-aways

This list is only a summary. For a more in-depth look into interactive web map accessibility, reference the “Checklist” section in this document.

- **Integrate color contrast** ratios of at least 4.5:1 for normal text and 3:1 for large text (WCAG 2.1 1.4.3 Contrast, Minimum).
- **Test without the mouse** and only using your keyboard with your interactive web map (WCAG 2.1 2.1 Keyboard Accessible).
- **Avoid keyboard traps** such as a pop-up dialog that prevents the user from entering and interacting with its contents via a keyboard.
- **Use color and texture** to distinguish elements in your interactive web mapping application.
- **Organize the reading order** of elements in your interactive web mapping application into a flow that makes sense.
- **Format the text** content and length for readability.
- **Recognize technological constraints** that impact the delivery of information, such as a framework that doesn’t automatically add <alt> text to basemap tiles. This can enable you to provide additional content, or let the user know of any constraints when accessing your interactive web map.

Section 2: Checklist

General Web Accessibility

- ✓ Visit the WCAG 2.1 quick reference guide.
    - In addition, visit Web accessibility for developers. The supplemental checklist provides a level A (beginner), level AA (intermediate), and level AAA (advanced) WCAG 2.1-specific checklist.
- ✓ Visit the State of Minnesota Office of Accessibility Web and Apps page for further information and resources on web/application accessibility. Includes links to guidelines, training resources, checklists, and tools.
- ✓ A few more introductory resources are available here:
    - A11y Project's Web Accessibility Checklist – "A beginner's guide to web accessibility"
    - World Wide Web Consortium (W3C):
        - Introduction to Web Accessibility
        - Tips and tutorials for writing, designing, and developing for web accessibility
        - Accessibility Standards Overview
- ✓ The map's essential information and purpose must be clearly explained in details of the map's content when a user visits the page. This is also a good location to share accessible limitations and contact information to obtain content in a differing accessible format.

Map Elements

- ✓ Headings: If content is added to your map, ensure the logical order on the page goes from Heading Level 1 <h1>, Heading Level 2 <h2>, Heading Level 3 <h3>, etc.
- ✓ Navigation elements (+/-): If navigation elements such as zoom in (+) or zoom out (-) buttons are on your page, ensure they are keyboard accessible and fit the logical tab order of the page.
- ✓ Buttons: Like navigation elements, ensure buttons are keyboard accessible, fit the logical tab order of the page, and have sufficient color contrast between the button's text and background.
- ✓ Links: Links must be recognizable and differentiated from other text content in the page (e.g., underlined text, or using another method other than color alone). In addition, all links should be focusable when navigating through the map application with a keyboard.
- ✓ Navigation/reading order: Ensure the map application's content and design has a logical structure. To check the reading order, tab through the application after loading it to see where the map's focus takes you to. The order should have the same flow as seen visually, particularly with forms and search dialogs.

- ✓ Map navigation: Map navigation should be possible without the use of a mouse. For instance, the user can use keyboard navigation to find a zoom in, zoom out, and initial extent button.
- ✓ Plain language: If there are components of the map that describe the application in detail, it needs to be clear and concise, without any use of jargon or undefined acronyms.
- ✓ Color contrast: All components of the map application must meet the following contrast ratios defined by WCAG 2.1 1.4.3 Contrast (Minimum):

Font type	**WCAG 2.1 Level AA (Minimum)**	**WCAG 2.1 Level AAA**
Normal text (< 14-pt.font)	4.5:1	7:1
Large text (14-pt. font and larger)	3:1	4.5:1

Figure 1. WCAG 2.1 color contrast ratios for Level AA (Minimum) and Level AAA.

- ✓ Line length: Line length must be considered, either too short or too long in length can be confusing to both sighted and non-sighted users. Ensure you keep text content concise but distinguishable.

Map Symbology

- ✓ Consider using textures if contrast among many map elements is difficult to achieve.
- ✓ Use a color contrast analyzer to evaluate the map symbology to verify that the points or features of interest have enough contrast compared to other maps contents. Keep in mind that individuals with low vision may use magnification, and test accordingly.
- ✓ Colors must be defined clearly in alternative text and with a legend/key.
- ✓ Consider using powerful colors for conveying important information.
- ✓ Select colors that do not have extreme color changes, as they can cause sensory overload for those with autism spectrum disorders.
- ✓ If symbology is key for the map, a map legend should be accompanied either in the mapping application, or adjacent to the map (if the map is embedded on a webpage) so users can verify the information and symbology. It is strongly recommended the map legend be visible when the

map loads, especially when viewing on a desktop or large tablet device. The legend should be easy to access if hidden or disabled in the map view at any time, including views on mobile devices.

Map Symbology Labels

- ✓ Verify the color contrast of the label text with the underlying map symbology color.
- ✓ Analyzing Map Labels for Contrast:
  - If color contrast cannot be met with one color, **try to combine two colors together using a halo** around the label's text. For example: Use white text with a black halo. The combined colors could meet ratios, where used alone they may not.

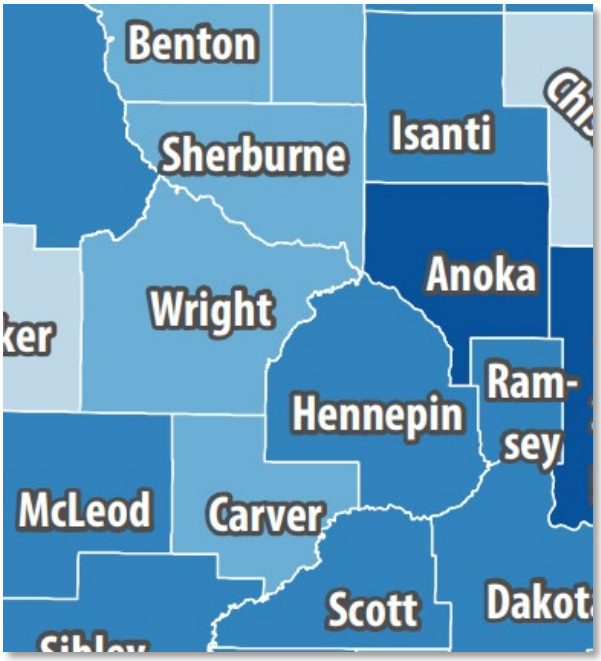


Figure 2. Map label to map symbology contrast example.

Background Color, Sequential Blues	Label Halo Color, Black (#000)	Label Color, White (#FFF)
#BDD7E7	Pass (WCAG 2.1, AAA)	Fail
#6BAED6	Pass (WCAG 2.1, AAA)	Fail
#3182BD	Pass (WCAG 2.1, AAA)	Fail
#08519C	Fail	Pass (WCAG 2.1, AAA)

Figure 3. Table of background colors contrast ratings compared to labels with two colors, the label color, and halo label color.

Map Elements and Color (e.g., Buttons)

- ✓ Colors cannot be used alone to depict important information.
- ✓ Consider using textures if contrast among many map elements if difficult to achieve.
- ✓ Use a color contrast analyzer to evaluate colors. This helps verify that the items of interest have enough contrast compared to other elements in the mapping application. Keep in mind that individuals with low vision may use magnification, and test accordingly.

Forms and Labels

- ✓ Whenever possible, use the label element to associate text with form elements explicitly. The "for" attribute of the label must exactly match the id of the form control.

```
<!--Label example-->
<label for="mapSearch">Search in the map</label>
<!--Search input-->
<input type="text" id="mapSearch"></input>
```

- ✓ Visually position labels to the right of radio buttons and checkboxes, and to the left of or directly above other form fields. Maintaining this practice increases predictability and understandability of your form for all users.
- ✓ For button elements, the label is set inside the element and can include markup. When using the **input** element to create buttons, the label is set in the **value** attribute of the element.

```
<!--Either button example below fits accessibility-->
<button type="submit">Submit</button>
<input type="submit" value="Submit">
```

- ✓ View more explicit guidelines and examples on the w3.org website.

Context Changes

- ✓ Look for items that require a change of context, and ensure the content change is clear to sighted and non-sighted users. For example, a map popup that changes its content when an item is selected. Using the aria-live attribute ensures that when the content is updated inside the element, those changes are communicated to users of assistive technology. View more examples for using aria-live.

```
<!--Context change example for a popup using the ARIA-live attribute-->
<div id="popup" aria-live="assertive">
<p>Map item description</p>
</div>
```

- ✓ Ask assistive technology to announce the dynamic changes using roles, such as error or warning message that appear on the screen. Alerts are particularly important for client-side validation notices to users.

```
<!--ARIA alert role-->
<h2 role="alert">
      The form was not be submitted because the "name" field is blank.
</h2>
```

Testing the Visible Focus

- ✓ Tab through the map/page to verify all interactive web map elements can receive focus and is visually indicated to the user.
- ✓ The visual focus order should match the intended interaction order.
- ✓ Interact with all controls, links, and menus using only the keyboard.
- ✓ Do not "trap" keyboard focus within subsections, except for modal dialogs.
- ✓ Keyboard focus should stay inside a modal dialog until it is dismissed. Once the modal dialog is dismissed, the keyboard focus should be restored to the previously focused element, e.g. Modal dialog example (WCAG 2.1 2.1.2 No Keyboard Trap).
- ✓ Off-screen content, such as responsive navigation, should not receive focus when it is not displayed. Do not hide content by positioning off-screen or behind other items. Use display:none in CSS to hide content from **all users**.

```
<!--Off screen content should use display:none to hide content from all
users-->
<div id="hiddenContentExample" style="display:none;">
<!--Hidden content to all users-->
</div>
```

- ✓ Ensure after interactions with controls that focus is where it should be in logical order or set to an element that contains the result information created by the interaction. Example: A search query being run and a data table being populated with the results, focus should be set to the results table.

Map Popups

- ✓ Can a user access the popup content without using a mouse?

- ✓ Can a user tab through and access all of the popup content when it is displayed on the screen?
- ✓ Does assistive technology know when a popup has opened and is displayed on the screen?
    - Use either aria-live=assertive **or** role=alert to interrupt anything assistive technology is currently reading or has to read. This way they know a popup has been displayed in the map. Be careful not to add both aria-live=”assertive” and role=”alert” as this will cause double speaking in VoiceOver and iOS. See use case examples on the W3C website (WCAG 2.1 3.3.1 Error Identification). After reading the popup to the user, assistive technology will continue reading where it left off.

Section 3: Solutions

Address the key components from the law on accessibility, and work your map around those components. For example, keep in mind keyboard-only navigation while developing the map application, so a user can navigate through the application using navigation keys. Do what you can to make the interactive web map accessible, to the best of your ability.

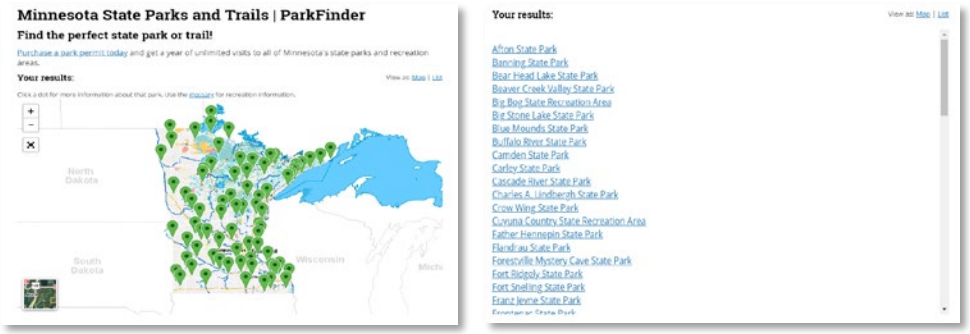
General Solutions

1. How to Meet WCAG 2.1, a quick reference using the four POUR principles of:

    - **P**erceivable: information and user interface components must be presentable to users in ways they can perceive;
    - **O**perable: user interface components and navigation must be operable;
    - **U**nderstandable: information and the operation of user interface must be understandable; and
    - **R**obust: content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

2. Focus on the map's intent/purpose.

    - Ask what the intent or purpose is, then focus on delivering key elements when designing your interactive web map.
    - For example, if you are creating an interactive web map to display Minnesota state parks, you could provide a visual of where the state parks are located around the state in a map and provide a list of Minnesota state parks.



    - Focus on the map delivery, and don't add in every component if you don't need to. Just because you can add in every component, doesn't mean you should. Provide the basic components needed for the map and its message.

3. Start with foundational cartography principles, such as Ordnance Survey's Cartographic design principles:

    - Understand user requirements;

- Consider the display format;
- Create a clear visual hierarchy;
- Focus on simplicity;
- Ensure legibility;
- Consider consistency;
- Ensure accessibility;
- Maintain good composition

4. Use color and texture (WCAG 2.1 1.3.1 Info and Relationships; WCAG 2.1 1.4.1 Use of Color).

    - Provide good color contrast through: navigation elements, buttons, links, text, and the map symbology (WCAG 2.1 1.4.1 Use of Color; WCAG 2.1 1.4.3 Contrast, Minimum; WCAG 2.1 1.4.6 Contrast, Enhanced);
    - Don't rely on color alone as users may be unable to distinguish, or may override page colors. Color cannot be the only way information is conveyed (WCAG 2.1 1.3.1 Info and Relationships; WCAG 2.1 1.4.1 Use of Color).

5. Do not assume your user is a power user.

    - Focus on plain language and a basic level of understanding to provide a better user experience and more accessible format to all of your users.

6. Use the best available tools or technology to get your message across. This may not necessarily be the latest version available.

    - For example, upgrading from the Esri JavaScript API 3.x series to the 4.x series can add out-of-the-box assistive technology support.

7. Recognize technological constraints.

    - For example, assistive technology reads the document object model (DOM), making it important for developers to use semantic HTML as much as possible, and other code, such as JavaScript and/or ARIA to describe what is taking place on the page/map.

Interactive Web Map Solutions

1. Consider whether the functional requirements of the application can be met without an interactive web map and, if so, consider incorporating the ability to use the application with or without the map present.

    - For example: See the Order Help Me Grow Materials accessible example.

2. Add a splash screen with text when the map loads alerting the user of potential constraints, and who to contact for alternative products (WCAG 2.1 2.4 Navigable).

3. Consider the reading order of the map and how users will navigate through the map (WCAG 2.1 3.2.3 Consistent Navigation)

    - Using keyboard-only tools, such as the arrow keys, tab, spacebar, and/or enter keys (WCAG 2.1 2.1.1 Keyboard).
    - A tabindex of 0, allowing assistive technology to access the element. Try to avoid any tabindex above 0, as it can interfere with the order of elements on your page, or map.
    - The order of elements should be in the same order as they appear on the screen as they are in your DOM/HTML. (WCAG 2.1 1.3.2 Meaningful Sequence)
    - Consider element sizing (e.g. buttons, lists, etc.). The recommended minimum font size is 12-pt for readability.

4. Consider text layout in the navigation elements, buttons, and text (WCAG 2.1 1.3.2 Meaningful Sequence) including:

    - Use true text (HTML text), where possible. True text enlarges better, loads faster, and is easier to translate. For visual styling, use CSS.
    - Try to stay away from using ALL CAPS. All caps can be difficult to read and can be read incorrectly, or misinterpreted by assistive technology (WCAG 2.1 1.4.8 Visual Presentation; WCAG 2.1 3.1 Readable).
    - Use adequate font size, as sizing can vary based on the font chosen and the device. Twelve-point (12 pt.) font, or 1em, is the recommended minimum font size (WCAG 2.1 1.4.4 Resize text; WCAG 2.1 1.4.8 Visual Presentation).
    - Test text layout and contrast under magnification to ensure that it is readable (WCAG 2.1 3.1 Readable), including icons and/or legends depicted in an image format.

5. Consider the text layout, including:

    - Use clear semantics where content should have good semantic structure (WCAG 2.1 1.3.1 Info and Relationships).

```
<!--Example where the role is specified so assistive technology
knows the function of the content-->
<li tabindex="0" class="checkbox" role="checkbox" checked aria-
checked="true">
Correct example of a checkbox as a list item
</li>
```

    - Ensuring a link's text makes sense without supporting text, avoid using "click here," "more," or "continue" and use phrases related to the link, such as "Learn more about map accessibility" (WCAG 2.1 2.4.4 Link Purpose, In Context; WCAG 2.1 2.4.9 Link Purpose, Link Only).
    - Remember line length, don't make lines of copy too long or too short (WCAG 2.1 1.4.8 Visual Presentation).

- Make links recognizable by differentiating them in the body of the page with underlines and/or a method besides color (WCAG 2.1 1.4.1 Use of Color; WCAG 2.1 1.4.8 Visual Presentation; WCAG 2.1 2.4 Navigable).
- A visible focus should be obvious for all focusable elements on the page, including buttons. For example, once focused, a link could have a dotted border, background, or a non-color designator (WCAG 2.1 1.4.1 Use of Color; WCAG 2.1 2.4.7 Focus Visible).
- Design accessible form controls by ensuring descriptive labels and instructions are included. Pay close attention to form validation errors and recovery mechanisms (WCAG 2.1 3.3.2 Labels or Instructions; WCAG 2.1 3.3.1 Error Identification; WCAG 2.1 3.3.3 Error Suggestion; WCAG 2.1 3.3.5 Help).

6. Design a "Skip to" content link, or bypass block, for keyboard users to skip navigation, the map, and the map navigation controls. The "Skip to" content link should navigate keyboard users to the content that makes up your map and gives users the ability to get to the purpose without having to navigate through all of the map (WCAG 2.1 2.4.1).

    - While a "Skip to table" / "Skip to data" option is handy, if you provide a table, it is useful to all audiences, and can be a supplemental piece of your interactive web map application, even when the map is the main focus.
    - You can bypass navigation and the map using "Skip to content." You could also include a "Skip to table," "Skip to data," etc. where a data table is provided with the map.
    - If selected, the "Skip to" link takes the user to the next focusable element in the page.

7. Avoid blinking, flashing, or strobing content or colors as they can cause seizures (WCAG 2.1 2.3 Seizures) and be difficult for those on the Autistic spectrum (WCAG 2.1 2.2.2 Pause, Stop, Hide). Be wary of libraries and functions, especially when working with WebGL and/or 3D that can cause jerking or rapid movements. If such movements are necessary, do not enable them by default and try to slow down movements so users can adjust to the changes. Examples: Uber traffic patterns, video and Esri JS v4 Web Scene slide tour.

8. Accessible Rich Internet Applications (ARIA) live regions provide a way to programmatically expose dynamic content changes in a way that can be announced by assistive technology. However, use ARIA sparingly. If you can perform the accessible component with HTML, do that first, and ARIA second.

    - aria-live region – used to alert the assistive technology user to a change in content in for example a popup window. In the example below using the assertive option of the aria-live attribute will immediately read the updated contents of the region to the user whenever the regions content changes.

```
<!--ARIA: aria-live region example-->
<div role="region" id="info" aria-live="assertive">
  <p id="description">Select to view its description</p>
</div>
```

- aria-describedby attribute - used to indicate the IDs of the elements that describe the object. It is used to establish a relationship between widgets or groups and text that described them. This is very similar to aria-labelledby: a label describes the essence of an object, while a description provides more information that the user might need. For example:

```
<!--ARIA: aria-describedby attribute example-->
<div role="application" aria-describedby="info">
      <h1 id="map">Map Example</h1>
      <p id="info">This map shows...</p>
</div>
```

- View more practical ARIA examples

9. Apply static map techniques, where possible.

Section 4: Accessible Examples

Interactive Web Maps (Static)

Order Help Me Grow Materials – Minnesota Department of Education

Purpose

Allow educators to obtain contact information in order to receive Help Me Grow outreach materials to share with parents. The Help Me Grow Program helps with young children's development, learning, and growing by developing milestones and encouraging healthy development.

Accessibility components

1. Contains a "keyboard-only" option allowing tabbing to the Address search field or County dropdown. The "Enter" key submits the search queries.
2. A popup appears upon selection and receives focus that contains the contact information for that County.
3. A "Skip Navigation" link jumps straight to the main content.
4. The visual focus order matches the intended interaction order.

More information

Access the interactive web mapping application.

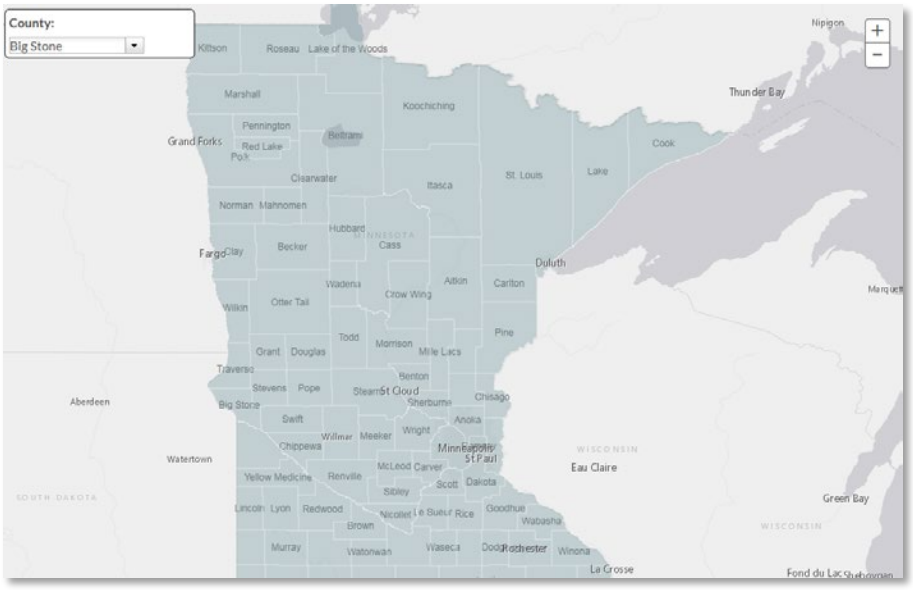


Figure 4. Order help me grow materials county interactive web map.

Minnesota Public Health Data Access Portal – Minnesota Department of Health

Purpose

A web portal featuring interactive web maps dedicated to visualizing and making public health data available to public health professionals and the public. The portal was created with all devices and audiences in mind. The data are accompanied in both a visual and tabular format so all users have access in a meaningful way. Accessibility testing has not been conducted for mobile devices at this time.

Accessibility components

The application includes some of the following functionality:

1. A "Skip to table" link is provided in the map so those accessing the application can skip the map page elements and head straight to a full data table.
2. The map is accompanied by a full data table accessible to all users, but was created with keyboard and assistive technology users in mind.
3. All map components, elements, and buttons are tab-able using the 'Tab' and 'Enter' or 'spacebar' keys.
4. Consider keyboard only users. When a user accesses the application using their mouse or keyboard all map components, elements, and buttons show focus (:focus) throughout the application.

More information

While many different maps are showcased throughout the Portal, the Asthma Hospitalizations interactive web map is one of the more popular maps on the portal.

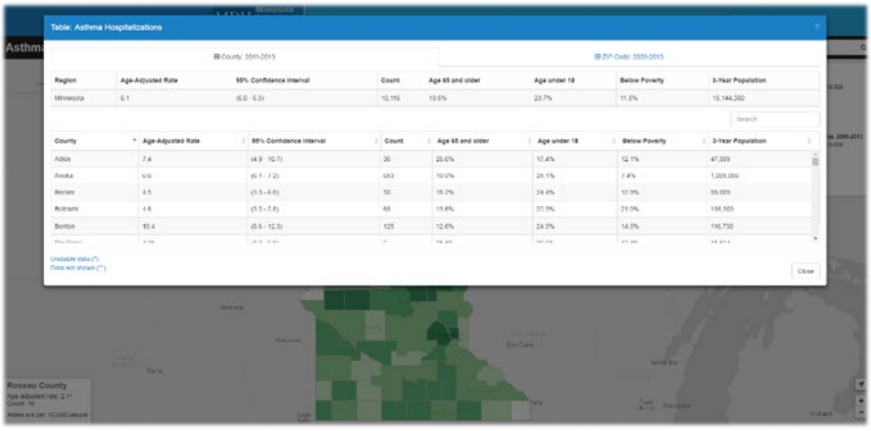


Figure 5. Asthma hospitalizations interactive web map with accompanying data table.

Interactive Web Maps (Complex & Wayfinding)

a11y map – Esri

Purpose

Use as a starting point to discuss ways for Esri to handle map accessibility. **However, be cautious in the implementation of the features for Esri's a11y map, as testing has not yet been fully conducted using speech recognition software (e.g., Dragon) and assistive technology (e.g., JAWS and NVDA). In addition, while assistive technology is being actively used, assistive technology shortcut keys and map navigation shortcut keys can interfere with each other.** Please see the GitHub submitted issue for more information on the testing of the a11y mapping application. A prototype interactive web mapping application to test adding keyboard interaction, similar to Google's a11y behavior, with Esri's ArcGIS API for JavaScript version 4.

Accessibility components

The application includes some of the following functionality:

1. All map components, elements, and buttons are tab-able using the 'Tab' and 'Enter' or 'spacebar' keys.
2. Once a user begins keyboard navigation, or uses the 'tab' key, a new keyboard navigation announcement button is shown in the map, allowing an assistive technology user to read information. **This feature has not been verified to be deactivated within 3 seconds as required by Web Content Accessibility Guidelines 1.4.2 (Audio Control).**

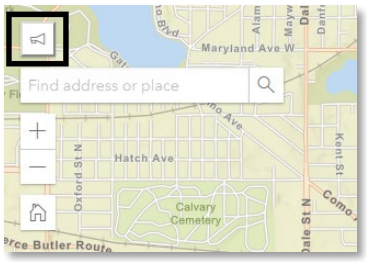


Figure 6. a11y map's keyboard navigation announcement button snapshot.

3. Once keyboard navigation is enabled, the user can navigate features in the map. The map can display a box centered on the map that can be changed using keyboard navigation buttons. **This feature currently interferes with JAWS and NVDA assistive technology when active.**

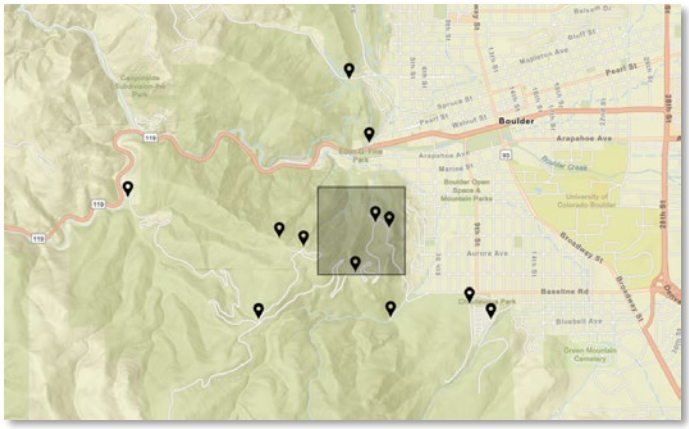


Figure 7. a11y map's selection box for assistive technology.

4. Features can be accessed via the box using the number pad (e.g. 1 to open the popup for Panorama Point).

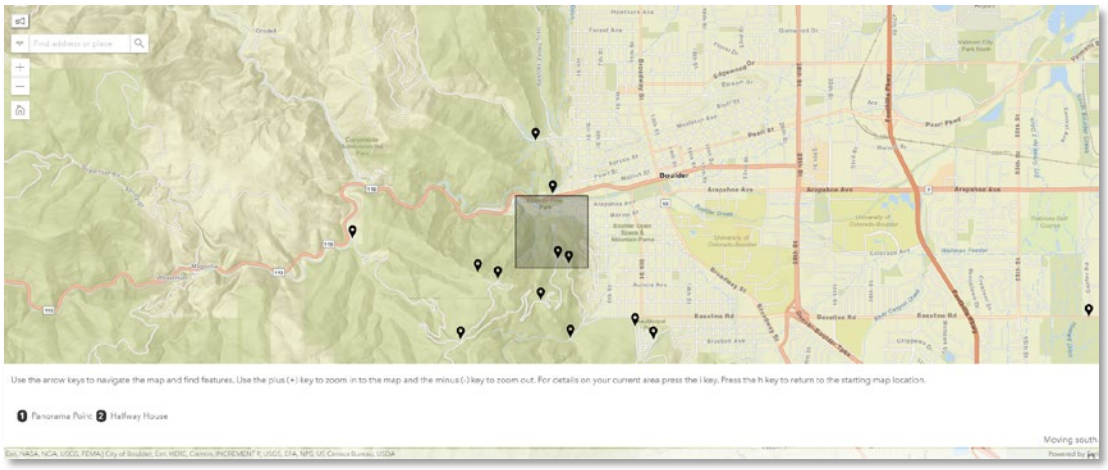


Figure 8. a11y map's feature access is keyboard accessible.

5. All popup information is keyboard accessible, including the highlighted result name, the ability to dock the popup, close the popup, read the attributes information, and zoom to the selected attribute.

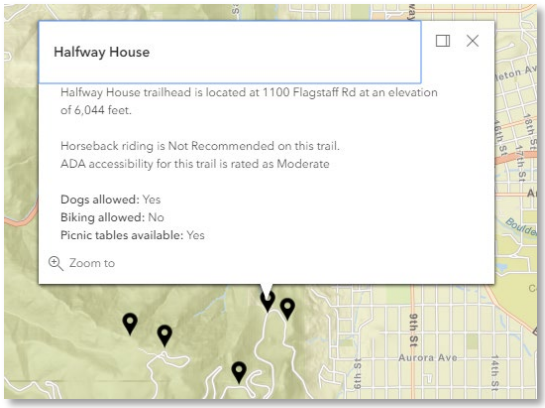


Figure 9. a11y map's keyboard accessible popup screenshot.

More information

Esri staff have hosted the [code repository] and the [interactive web mapping application] on GitHub.

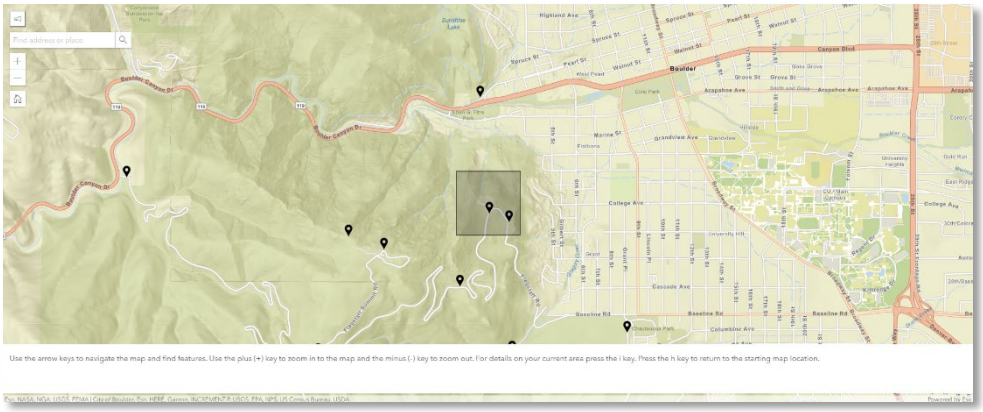


Figure 10. a11y map screenshot.

Public Comment Crowdsource Map – Minnesota Department of Transportation

Purpose

The results of the Minnesota Department of Transportation study were used to help guide future transportation project development and investment in this corridor. The study aimed to develop a vision for the future of the Interstate 94 corridor between St. Paul (Hwy 61) and Minneapolis (Broadway Avenue).

Accessibility components

The application includes some of the following functionality:

1. All map components, elements, and buttons are tab-able using the 'Tab' and 'Enter' or 'spacebar' keys.
2. If you are unable to access the map, a dialog appears on-load that is accessible, and allows the user to key through and provide feedback.
3. Assistive technology will read information about the study area and why comments are being pursued after the map loads.
4. "Unable to use this map" link allowing users to provide feedback if they are unable to use the map.
5. An accessible PDF version of the map is also provided on the project website.
6. Can use a touch pad or mouse, and drag an arrow on the map and add a comment into the map.

More information

Access the rethinking I-94 phase I project page.

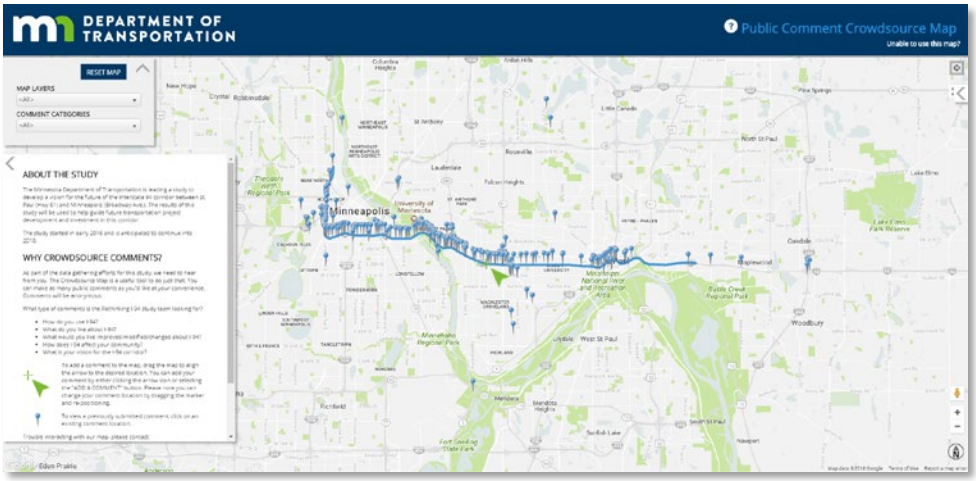


Figure 11. Public comment crowdsource map desktop screenshot.

GeoPDF Downloader – Minnesota Department of Natural Resources

Purpose

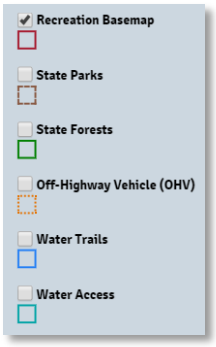
An interactive web mapping application containing geo-enabled PDF maps with digital coordinates. Users can download a geo-enabled PDF map for an area of interest on their smartphone device using the Avenza Maps application (or similar) to view their location on the map, even when airplane mode is active or cellular service is unavailable.

The application was created to provide updated digital maps in the form of PDFs to the public and Department of Natural Resources staff when in areas of low connectivity with all devices and audiences in mind. Accessibility testing has not been conducted for mobile devices at this time.

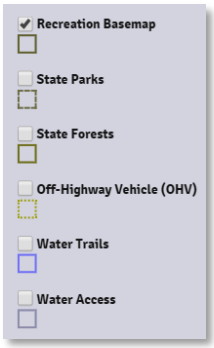
Accessibility components

The application includes some of the following functionality:

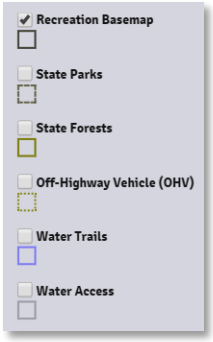
1. GeoPDF map types are distinguished by color and outline. Differing colors with styles pass distinction for: Deuteranopia, Protanopia, and Tritanopia color blindness.



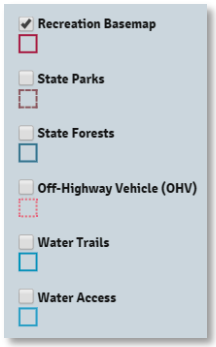
Normal vision



Deuteranopia vision

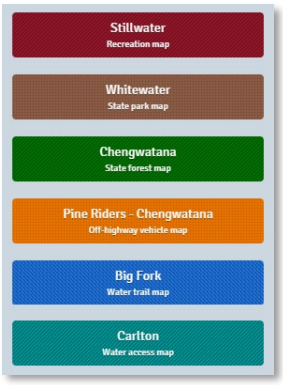


Protanopia vision



Tritanopia vision

2. Similar to the GeoPDF map types, map button types are distinguished by color, texture, and text. The differing colors with styles pass distinction for: Deuteranopia, Protanopia, and Tritanopia color blindness. Additionally, each map indicates its type (e.g. Recreation map).



Normal vision

Deuteranopia vision

Protanopia vision

Tritanopia vision

3. The maps provided are wayfinding maps but each are assigned a name. To provide access to non-sighted users and an easier way to provide geo-enabled PDF maps to sighted users, a search accompanied by the name and type of map is listed so all users are able to access the data in a meaningful way (e.g. Angle Inlet, Recreation map).
4. All map components, elements, and buttons are tab-able using the 'Tab' and 'Enter' or 'spacebar' keys.
5. When a user accesses the application using their keyboard all map components, elements, and buttons show focus (:focus) throughout the application.

More information

Access the interactive web mapping application.

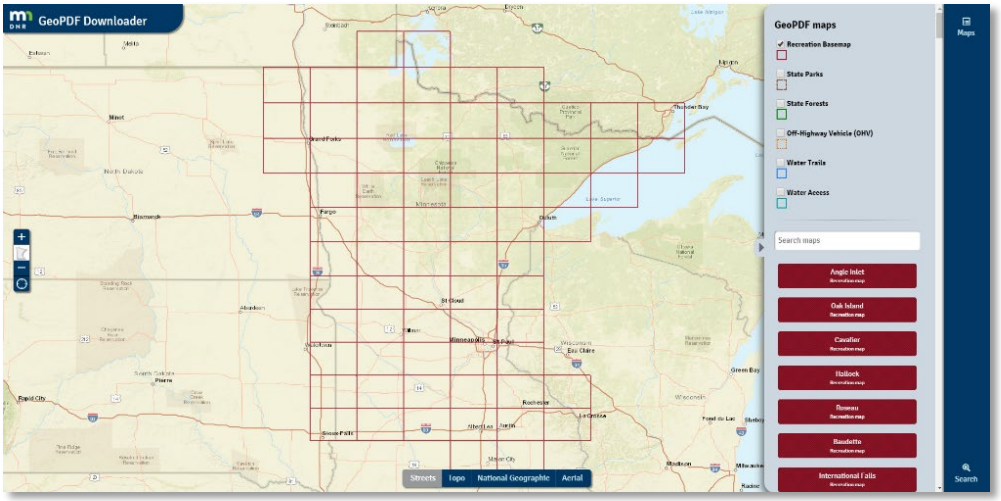


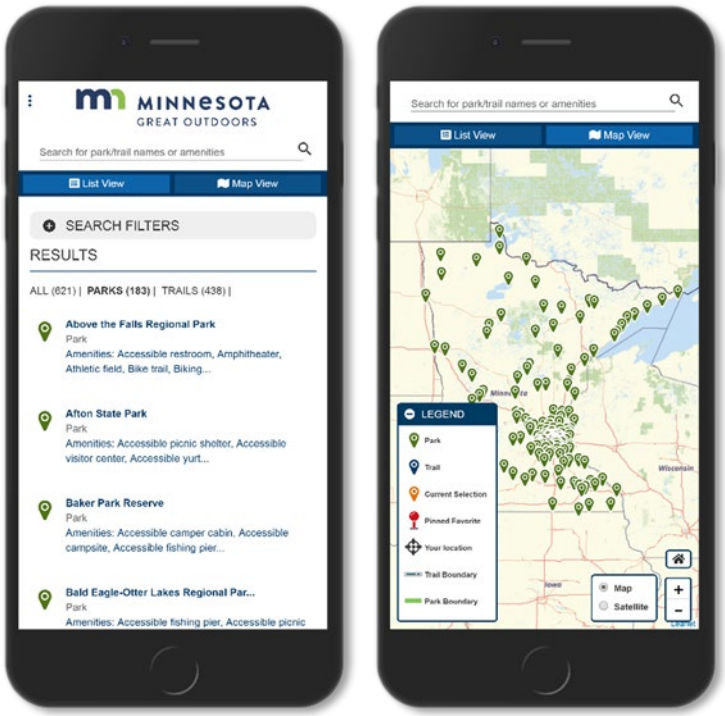
Figure 12. GeoPDF Downloader desktop screenshot.

Minnesota Great Outdoors – Minnesota Department of Natural Resources, et al.

Purpose

An interactive web mapping application showcasing all state and regional parks and trails that are eligible to receive Clean Water Land and Legacy funding. Users can obtain maps, full details, and the most current information about any location.

The application was created to showcase all state and regional parks and trails that are eligible to receive Clean Water Land and Legacy funding.



Accessibility components

The application includes some of the following functionality:

1. Simplicity is one of the key guiding principles of the application. Keeping the structure simple makes it easier to sustain and maintain accessibility.
2. A list compliments the map and could be used as a standalone alternative, showcasing the data. Additionally:

    - There is a list-only option on mobile devices, and
    - The list comes first in the document structure when a user is navigating via the keyboard.

3. Colors meet contrast ratio recommendations and the basemap is highlighted by a streets-typed basemap which has a higher contrast ratio than satellite/imagery.

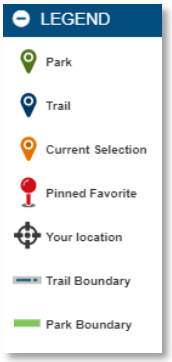


Figure 13. Minnesota Great Outdoors application legend.

4. Material design accessibility was incorporated improving accessibility and usability for all users and includes some of the following components:
    - Alternative text, <alt>, tags that are smart;
    - Screen reader classes that can be added;
    - Easy to read buttons; and
    - Color contrast ratios throughout the interactive web mapping application.
5. Pagination allows users to get through the list more effectively and find what they are looking for.
6. The search was given a high order in the focus priority, making it more readily available when a user is navigating via the keyboard.
7. The logical tab order and focus placing the map at the bottom of the tab order allows users to navigate through content first and takes away the ability to tab through pins on the map at the beginning of their experience.

More information

Access the interactive web mapping application.

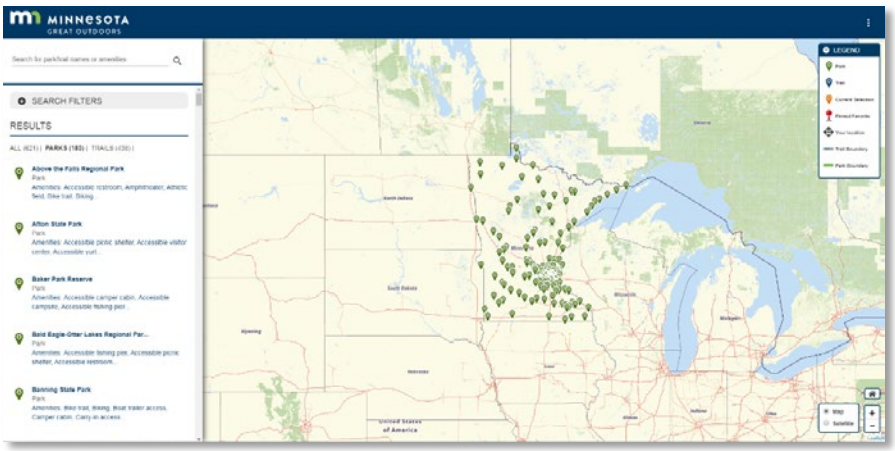


Figure 14. Minnesota great outdoors desktop screenshot.

MapMaster 2.0 – Pearson Education, Inc.

Purpose

An interactive web mapping tool designed to help students develop their geographic literacy, spatial reasoning, and critical thinking skills by examining patterns and relationships between different regional and global data. The tool includes more than 400 cultural, economic, physical, and political datasets.

Accessibility components

The application includes some of the following functionality:

1. The atlas enables keyboard and assistive technology users to access functions through native keyboard-accessible elements with HTML 5, such as <button> and <input> with tabindex.
    - A loading dialog is present while the application loads.
    - The interactive web map is navigable via the tab key, moving focus from one element to the next.
    - Keyboard control of the atlas uses tab-able elements.
    - Element order considered for keyboard-accessible elements.
    - Correct and useful information provided for audio screen reading software.
    - ARIA was only when needed to support functionality.
    - Occasionally used JavaScript to set the keyboard focus.

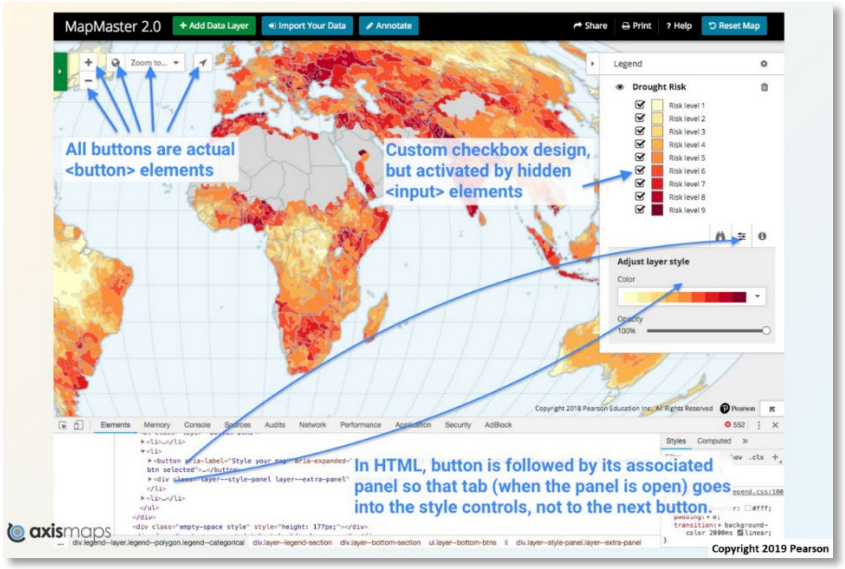


Figure 15. MapMaster screenshot displaying button elements, and when items are referenced, they can be accessed by the user. Copyright 2019 Pearson.

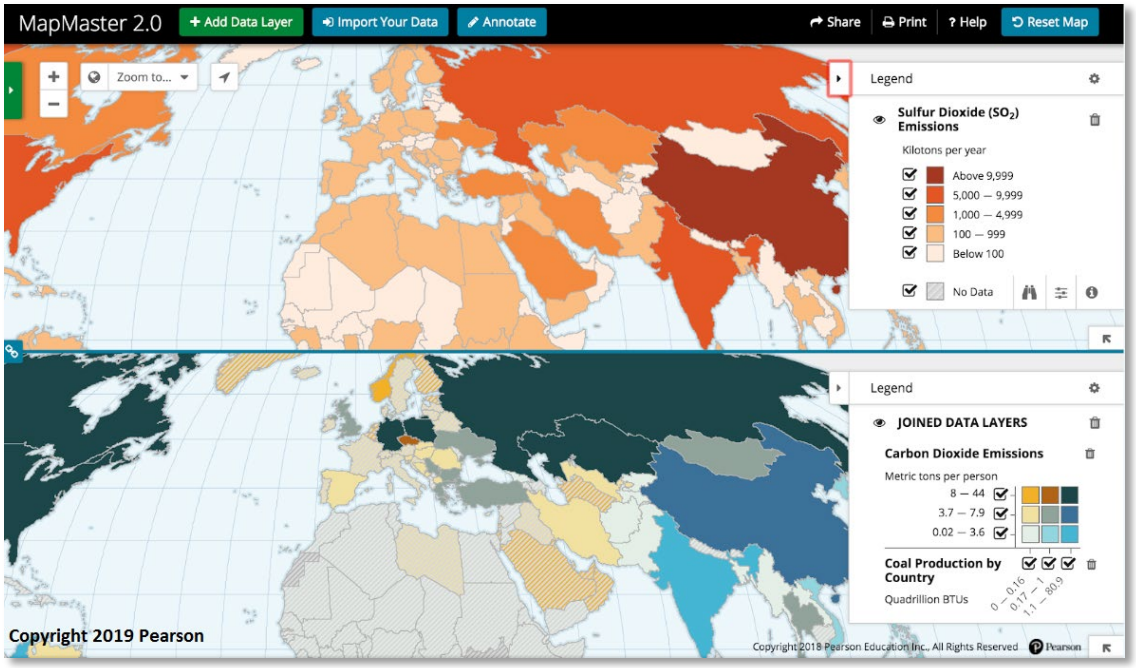


Figure 16. MapMaster screenshot of the focus on the map legend (top left) when users access items throughout the interactive web map. Copyright 2019 Pearson.

2. Map data accompanied by descriptions.

    - The data descriptions enable a blind and low-vision user to "see" an interactive web map with the help of assistive technology. Users should have access to both data values and the broader spatial patterns that others interpret with their eyes.
    - Provide meaningful map descriptions, not just a definition of the data but a summary of what the data show.
    - Provide a data table so users can tab through map steps through an invisible table, from which assistive technology will read names and data values.

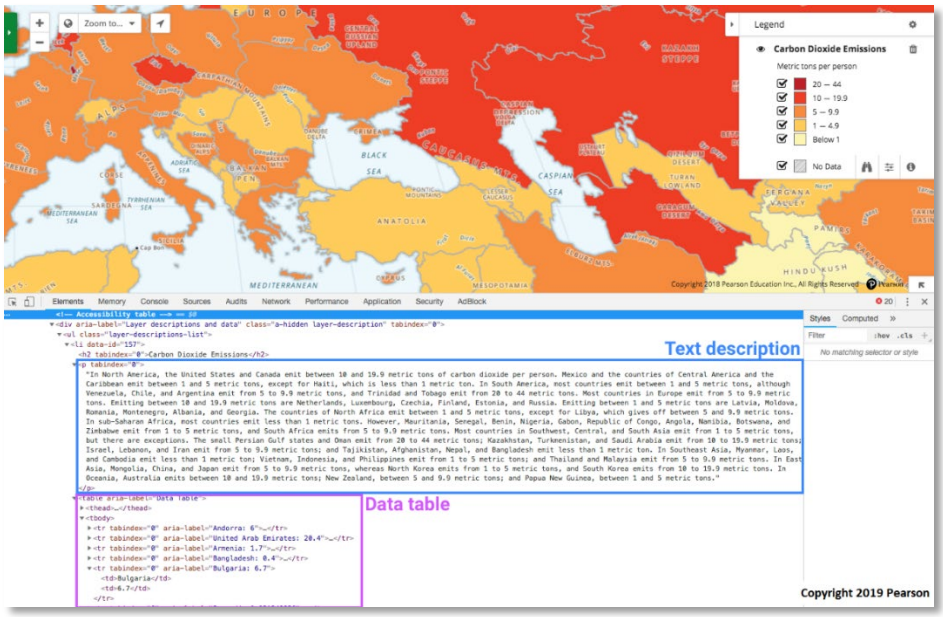


Figure 17. MapMaster screenshot of a text description of the map and a table with the corresponding values for users of assistive technology. Copyright 2019 Pearson.

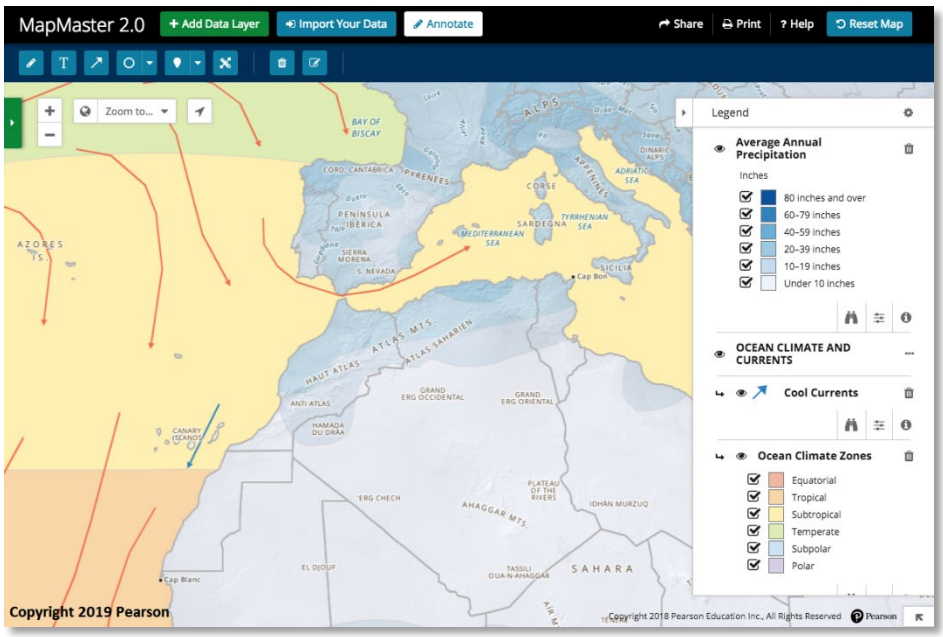


Figure 18. MapMaster screenshot of the legend controls, allowing the user to change the symbology elements one-by-one. Copyright 2019 Pearson.

More information

Access the interactive web mapping application.

Section 5: Technology Recommendations

It is strongly recommended developers create the element you wish to use (e.g. a button as a button, rather than a div as a button), as functionality and accessibility are built in, and require no JavaScript or ARIA and is robust for the future.

```
<!--HTML for button that meets accessibility-->
<button id="searchBtn">Search</button>
```

However, if you must create a element acting as a button, there are a few elements the developer must add in to ensure the element is keyboard accessible, including the: 'role,' 'tab index,' and functionality when a user clicks or accesses via the enter or space keys. The developer must ensure the element can be accessed by a keyboard using the 'tab-index' attribute.

```
<!--HTML for a div mimicking a button that meets accessibility-->
<div role="button" tab-index="0" id="searchBtn">Search</div>

<script>
    /* JavaScript */
    document.getElementById("searchBtn")
        .addEventListener("keyup", function (event) {
            event.preventDefault();
            if ((event.keyCode === 13) || (event.keyCode === 32)) {
              //Add functionality when a user accesses via the Enter/Space keys
              //Can be set to have the same functionality as click events
            }
        });
</script>
```

General Web Recommendations

Generally avoid:

1. CSS outline

    - **Setting a CSS outline of 0 or none should not be implemented** across the interactive web mapping application, as assistive technologies will be unable to navigate through your site.

      ```
      <!--CSS that should NOT be implemented-->
      a {
          outline: 0; /* Do not implement */
          outline: none; /* Do not implement */
      }
      ```

2. aria-hidden

- Only use if hiding content improves the experience for assistive technology users by removing redundant or extraneous content.
- For example: A menu item that has text and a font awesome icon could use aria-hidden on the font awesome icon as it is redundant information to those using assistive technologies.

```
<!--ARIA hidden property-->

<i class="fas fa-camera-retro" aria-hidden></i>
```

3. tabindex greater than 0

    - Setting DOM elements to have tabindex values greater than 0 will change the order of the DOM elements in your mapping application. This can confuse not only those with assistive technologies, but sighted mouse-users, and the developers coding the application. Instead of changing the value of tabindex to a positive integer, change the order of the elements in the application to reflect the order seen/heard in the application.

4. tabindex = -1

    - Setting DOM elements to have a tabindex of -1 hides the content from assistive technology users. While not recommended, it can be implemented for visual elements in the page. However, this is not common, and most applications will not have a need for implementation of tabindex = -1.
    - For example: If displaying a div that contains a Google street view image, this content may not be pertinent to an assistive technology user in which case implementing a tabindex of -1 would be a good implementation.

Mapping Library Recommendations

Esri JavaScript API 4.x

The Esri JavaScript API 4.x includes:

Map navigation can be performed with keyboard shortcuts. **However, these keyboard commands conflict with some JAWS and NVDA keyboard commands.**

Action	Behavior
Arrow keys	Nudge the map to the left, right, up, or down
N	Adjust the view to point north
A	Rotate the view counter clockwise
D	Rotate the view clockwise
+	Incrementally zoom in
-	Incrementally zoom out

SceneView navigation can be performed with keyboard shortcuts (not tested with assistive technology);

Action	Behavior
Arrow keys	Nudge the scene to the left, right, up, or down
P	Move the camera to look perpendicular
N	Adjust the scene to point north
J	Move down closer to the view
U	Move up, higher from the view

The Mapview, SceneView, Popup, and Search classes have a focus method;

```
 /* When the popup is visible, set the focus to the popup
    Else when the popup is not visible, set the focus
    to the search. */
 view.popup.watch("visible"), () => {
    if (view.popup.visible) {
       view.popup.focus();
    } else {
       search.focus();
    }
}
```

Tab order of widgets shown on the page are reflected in the user experience;

Page elements are keyboard accessible via the arrow, tab, spacebar and enter keys; and

The API allows the developer to include accessibility functionality in custom widgets/buttons more readily.

Leaflet Library

Leaflet Library Map tiles: By default leaflet sets alt text to an empty string to prevent assistive technology from reading the URL.

Map navigation:

Interaction	Keyboard Procedure
Pan map up/down, right/left	Arrow keys
Zoom in/out	+/- or tab to controls
Focus on marker	Tab to marker
Open popup	Select ‘enter’ on marker focus

- Map navigation can also be modified using ‘zoomDelta’ options to customize the zoom level change (for both clicking and using the +/- navigation buttons).
- Ability to add ‘<alt>’ tags to Leaflet markers.
- L.marker(latlng, {alt:"text for alt attribute"})
- Tile images use role=presentation and alt="" so assistive technology doesn't read their URLs.
- Map markers use tabindex=0 and popup content containers have an anchor tag to close. This allows the user to access the marker via keyboard or mouse. For Example: Leaflet markers.
- Image overlay, the alt text can be added in to display a single image over specific bounds of the map.
- Zoom controls use role=button and aria-label="Zoom In/Out" on the zoom buttons
- Voiceover issue, with resolution detailed in this Github thread.

Mapbox GL Library

Mapbox GL Library Map Navigation

Interaction	Keyboard Procedure
Pan map up/down, right/left	Arrow keys
Zoom in/out	+/- or tab to controls
Tilt	Shift + up/down arrow keys
Rotate	Shift + right/left arrow keys
Focus on marker	*requires plugin” Tab into marker
Open popup	Select ‘enter’ on marker focus

- No explicit references to map accessibility, except through lightweight plugin (currently in development)
- Canvas contains aria-label=”map”
- Zoom controls use buttons and aria-label=”zoom in/out”
- Ability to tab into markers with plugin that uses tabindex, aria-label, title

Bootstrap

Bootstrap details:

- Create websites and applications fulfilling WCAG 2.1 A/AA/AAA and Section 508 accessibility standards and requirements,
- Apply structural markup including styling and layout to a wide range of markup structures,
- Interactive web map components (modal dialogs, dropdown menus and custom tooltips) are designed to work for touch, mouse and keyboard,
- Color contrast is part of the default palette, and
- Visually hidden content using the 'sr-only' class.

Font Awesome (icon library):

Font Awesome (icon library) details:

Decorative icons: If you are using icons for decorative purposes only, you can add the 'aria-hidden' attribute to ensure the icons are accessible. For example:

```
/* Font awesome 5 decorative icon example */
<i class="fas fa-headphones" aria-hidden></i>
```

Semantic icons: If your icons have semantic meaning, you will need to:

Manually add the 'aria-hidden' attribute

Provide a text alternative inside a ',' or similar element. Also include appropriate CSS to visually hide the element while keeping it accessible to assistive technologies.

Add a 'title' attribute to the icon to provide a tooltip for sighted mouse users.

```
/* Font awesome 5 semantic icon example */
<i class="fas fa-times-circle" aria-hidden title="Close the dialog window">
    <span class="sr-only">Close the dialog window</span>
</i>
```

Section 6: Testing

Testing is the only way to ensure the experience is accessible. It is key to perform a functional test using the WCAG 2.1 Success Criteria to verify how well web content functions compare to the criteria and is less subjective than usability testing.

Overview of WCAG 2.1, and its POUR principles:

Principle	Success Criteria	Level A	Level AA	Level AAA
1. **P**erceivable	1.1 Text alternatives, 1.2 Time-based media, 1.3 Adaptable, 1.4 Distinguishable	1.1.1, 1.2.1 – 1.23, 1.3.1 – 1.3.3, 1.4.1 – 1.4.5	1.2.4 – 1.2.5, 1.3.4 – 1.3.5 1.4.3 – 1.4.5, 1.4.10 – 1.4.13	1.2.6 – 1.2.9, 1.3.6 1.4.6 – 1.4.9
2. **O**perable	2.1 Keyboard accessible, 2.2 Enough time, 2.3. Seizures, 2.4 Navigable, 2.5 Input modalities	2.1.1 – 2.1.2, 2.1.4, 2.2.1 – 2.2.2, 2.3.1 2.4.1 – 2.4.4, 2.5.1 – 2.5.4	2.4.5 – 2.4.7	2.1.3, 2.2.3 – 2.2.6, 2.3.2 – 2.3.3, 2.4.8 – 2.4.10, 2.5.5 – 2.5.6
3. **U**nderstandable	3.1 Readable, 3.2 Predictable, 3.3 Input assistance	3.1.1, 3.2.1 – 3.2.2, 3.3.1 – 3.3.2	3.1.2, 3.2.3 – 3.2.4, 3.3.3 – 3.3.4	3.1.3 – 3.1.6, 3.2.5, 3.3.5 – 3.3.6
4. **R**obust	4.1 Compatible	4.1.1. – 4.1.2	4.1.3	

Tools

1. Hardware:

    - Perform keyboard testing, and/or unplug your mouse while interacting with the interactive web mapping application.

2. Software:

    - Color Contrast Analyzer Tool, Appendix A: The Colour Contrast Analyzer (CCA) helps you to determine the legibility of text and the contrast of visual elements, such as graphical controls and visual indicators. It features:
        - WCAG 2.1 compliance indicators;
        - Several ways to set colors: raw text entry (accepts valid CSS color formats), RGB sliders, color picker (Windows and Mac OS only);
        - Support for alpha transparency on foreground colors; and
        - Color blindness simulator.

3. Websites:

    - WebAIM Color Contrast Checker: This website is a companion to the Colour Contrast Analyzer software. Its most useful feature is a set of lighten and darken links that can be used to make subtle changes to existing colors until a "pass" rating is attained. This tool:
        - Allows the user to enter or select foreground and background colors;
        - Provides functionality to lighten or darken colors to find acceptable contrast levels;
        - Displays the contrast ratio based on the WCAG 2.0 luminosity formula;
        - Indicates if the colors pass the WCAG 2.0 level AA and level AAA requirements for normal and large text sizes; and
        - Shows a sample of the colors specified for quick viewing.

4. Browser extensions:

    - aXe: An open source rules library for accessibility testing. It was developed to empower developers to take automated accessibility testing into their own hands and to avoid common pitfalls of other automated accessibility tools. It does not require any outside server calls, and it can be customized to include custom rules and to integrate with all modern browsers and testing frameworks.
    - Web Accessibility Evaluation Tool (WAVE), Appendix B: A tool to help web developers evaluate their web content and make it more accessible. Using the tool, enter a web address and the web page is presented with embedded icons and indicators giving information about its accessibility. Click on an icon to see a brief overview of what each icon means and view its documentation, or access the documentation panel. People who are not web accessibility experts can also benefit from WAVE.

5. Assistive technology: Assistive technology testing requires training either via a course or with an expert assistive technology user. Testing, as with any other tool, is different than using the tool.

    - JAWS: An assistive technology that allows blind and low vision users to read the screen either with a text-to-speech output or by a refreshable Braille display.
    - Non-Visual Desktop Access (NVDA): A free and open source assistive technology for the Microsoft Windows operating system. It provides feedback via synthetic speech and Braille enabling blind and low vision users to access computers running Windows.
    - VoiceOver: Assistive technology built into Apple Inc.'s Mac OS and related operating systems.

Exercises and simulations

"Testing all the things" is not realistic. It is also critical to test with users that use assistive technology and with the browser zoom/screen magnification on an everyday basis. However, there are certain exercises and functions **you** can test, for example:

1. **Perform keyboard testing**, and/or unplug your mouse on your application.

- Move keyboard focus using the ‘tab’ / ‘shift’ + ‘tab’ keys.
    - Do you see the visual indication of focus?
        - If unable to locate the focus while testing in-browser, type ‘document.activeElement’ in the console to display the active focused element.
    - Do you see the expected behavior while using the ‘Tab’ key in the application?
- While testing, to access links use the ‘enter’ key.
- While testing, to access buttons use the ‘enter’ + ‘space’ keys.
- While testing, to access menus and form controls use the arrow keys.
- Is any content hidden by a ‘mouse-over’ event listener?
- For additional support, see WCAG 2.1 Principle 2, Operable, for a list of success criteria to perform keyboard testing.

2. **Turn off, or disable CSS** on your mapping application to visualize the page without styling.
3. **Turn off, or disable images** in your application to reveal alt text in blank image boxes.
4. **Verify the color patterns** you have implemented meet the WCAG 2.1 Success Criterion 1.4, Distinguishable, for a list of success criteria to achieve color validation.
5. View your page in “high contrast” mode.
6. Verify field labels are coded to the proper form input when brought into focus.
7. Check the DOM’s heading order of <h1>, <h2>, etc. tags.
8. Verify items that require a change of context. For example, a dropdown menu.
9. Check the HTML’s navigation order.
10. Check the skip navigation and landmarks.
11. Check that lists are properly coded as a list. For example, a navigation menu with dropdown benefits as a list.
12. Check link text.
13. Check multimedia components. For example, videos, featured stories, etc.
14. **Carry out simulations** (e.g., assistive technology, low-vision, dyslexia, and distractibility).

    - Assistive technology testing requires training either via a course or with an expert assistive technology user. Testing, as with any other tool, is different than using the tool. When testing using assistive technology, you can test:

        - Navigation
            - Headings

        - Links
        - Landmarks
        - Menus
    - Content
        - Alt text
        - Tables
        - Charts
    - Interaction
        - Forms
        - Dialogs
        - Messages
        - Widgets

15. **Run an automated test**. No automated test tool can prove conformance with WCAG Success Criterion. Automated tests are a good starting point but cannot detect all accessibility issues. The test should run on each page state and are rendered in the browser's DOM.
16. Web Accessibility Evaluation Tool (WAVE)
17. Provides a quick visual summary and the ability to look at the code by selecting a specific accessibility error or warning.
18. Axe automated testing tool
19. Aims at no false positives.
20. Provides issues as violations and its impact (e.g. serious impact), and lets the developer know areas that need additional review.
21. Also provides information on color contrast and on the inspected node Chrome's color picker will prove the contrast ratio and if it meets WCAG 2.1 criteria of 1:4.5.
22. The axe tool itself is accessible and has great documentation.
23. Share with a colleague. Once completed, contact your organization's accessibility point of contact to have them do a final check and provide their feedback.

Section 7: Conclusion

If you have read the entire document, then, congratulations! You should have a good idea on how to make web maps accessible for people with disabilities. A few important points are worth repeating:

- Accessibility testing tools can be an effective way of verifying accessibility for users living with disabilities; however, they are not a catchall for all situations and users. The best accessibility software covers approximately 25% of WCAG 2.1, catching the “big” issues.
- Manual testing, such as with the keyboard and using assistive technology such as screen readers, to run more intensive tests, are critical tools in discovering and isolating issues that automated tests miss.
- A final step, end-user testing in which individuals with disabilities run through your product with their assistive technology tools can help unearth both accessibility and usability issues.

Additionally, other factors can affect testing, such as: technology can be difficult to learn for a user, a user’s investment may differ if they have a disability, and/or a user’s disability type can play a part in their experience.

Lastly, it is critical to focus on all types of disabilities including, but not limited to, visual, cognitive, hearing and mobility disabilities.

Section 8: References

Agafonkin, V. (2018, September 18). Leaflet. Retrieved from https://leafletjs.com.

Avenza (2018, September 18). Avenza Maps. Retrieved from http://www.avenza.com/avenza-maps.

Deque (2018, September 18). aXe: The Accessibility Engine. Retrieved from https://www.deque.com/axe.

Esri (2018, November 1). A11y Map. Retrieved from https://esri.github.io/a11y-map.

Esri (2018, September 18). ArcGIS API for JavaScript. Retrieved from https://developers.arcgis.com/javascript.

Font Awesome (2018, September 18). Retrieved from https://fontawesome.com.

GitHub (2018, September 18). Retrieved from https://github.com.

Henning, S., Zobl, F., Wasserburder, W (2019, June 3). Accessible Web Maps for Visually Impaired Users: Recommendations and Example Solutions. Retrieved from https://cartographicperspectives.org/index.php/journal/article/download/cp88-pr/1575.

Koch, C. (2018, September 21). Chad's Story: IT Accessibility with a Motor Skill Disability [Blog post]. Retrieved from https://theweco.com/maneuvering-internet-motor-skill-disability.

Mapbox (2018, September 18). Mapbox GL JS. Retrieved from https://www.mapbox.com/mapbox-gl-js/api.

Material Design (2018, September 18). Accessibility. Retrieved from https://material.io/design/usability/accessibility.html.

Minnesota Legislature Office of the Revisor of Statues. (2018, November 21). 2018 Minnesota Statues. Retrieved from https://www.revisor.mn.gov/statutes/cite/16E.03.

Ordnance Survey (2018, September 18). Cartographic Design Principles. Retrieved from https://www.ordnancesurvey.co.uk/resources/carto-design/carto-design-principles.html.

Pearson Education, Inc. (2018, November 14). Mapmaster 2.0. Retrieved from https://media.pearsoncmg.com/bc/bc_0media_geo/mapmaster/mm2/shared.php.

Stack Overflow (2018, September 18). Retrieved from https://stackoverflow.com.

United States General Services Administration (2018, November 14). Section 508.gov. Retrieved from https://www.section508.gov.

United States Government Publishing Office (2018, November 21). United States Code Title 29. Retrieved from https://www.gpo.gov/fdsys/pkg/USCODE-2011-title29/html/USCODE-2011-title29-chap16-subchapV-sec794d.htm.

W3C (2018, September 18). The World Wide Web Consortium. Retrieved from https://www.w3.org.

W3C (2018, November 14). Web Content Accessibility Guidelines Overview. Retrieved from https://www.w3.org/WAI/standards-guidelines/wcag.

WebAIM (2018, September 18). Web Accessibility in Mind. Retrieved from https://webaim.org.

WUHCAG (2018, September 18). WCAG 2.0 Checklists. Retrieved from https://www.wuhcag.com/wcag-checklist.

Woodruff, A. (2018, November 14). The academically, administratively, all-around accessible atlas! Retrieved from http://bit.ly/nacis2018.

YouTube (2018, September 18). Retrieved from https://www.youtube.com