# Maryland Statewide Mapping Accessibility Guidelines Project

## Statewide IT Accessibility Initiative

The Maryland Department of Disabilities Information Technology (IT) Accessibility Initiative supports state agencies to meet Maryland's Non-Visual Access standards. The IT Accessibility Initiative's purposes are to improve policies and practices in state IT procurement and ensure equal access to state agency information technology for citizens with disabilities.

Email: [mdod.nva@maryland.gov](mailto:mdod.nva@maryland.gov)

Website: [mdod.maryland.gov](https://mdod.maryland.gov/news/Pages/Web-Accessibility.aspx)

Youtube channel: [MDTAPVideo](https://www.youtube.com/user/MDTAPVideo)

## 

## 

## 

## 

## Web Map Accessibility Guidelines

The intention of the mapping accessibility guidelines is to provide GIS administrators and site owners with specific guidance on how to comply with state and federal accessibility standards and provide non-visually accessible means to obtain the presented information.

1. **Perceivable Content**
   1. *All non-text content that provides relevant information must have a text alternative*.
      1. Images must be marked with the "alt" attribute or an equivalent label method. Images that do not convey relevant information or are used for decorative/spacing purposes should be marked with an empty "alt" attribute (otherwise known as a null alt attribute).
         1. Background elements, such as map quadrants, should be marked with a null alt attribute or placed in the background if possible.
      2. Form controls, buttons, and other interactive elements must have a programmatically associated, descriptive label.
      3. Images of text should not be used when it is possible to use formatted text.
      4. Any audio that conveys meaning must have captioning or other text equivalent.
      5. WCAG best practices for alternative text should be followed.
   2. *Visually communicated information must have an alternative accessible to assistive technology.*
      1. Information such as directions, locations, and other data must have a text alternative OR be in text format that is focusable by assistive technology.
      2. Presented information should be in a list or table format available in the widget. If this is not possible, a link to this information should be provided alongside the widget.
   3. *Links must be visually discernable from regular text and sufficiently descriptive of their destination.*
   4. *There must be a visible focus indicator for all focusable elements.*
      1. The visible focus indicator should be plainly visible to all users, following recommended width (3-5px) and contrast ratios (3:1).
   5. *Color must not be the only method of conveying information, except when directly relevant to understanding the content.*
      1. Color used to convey meaning must be accompanied by either texture or symbols.
      2. Exception: Color that is directly relevant to conveying information, such as a heat map using yellow and red to display levels, is exempt from this requirement.
   6. *User interface elements containing text must meet WCAG 2.0 AA Color Contrast standards (4.5:1 for regular text, 3:1 for large print).*
2. **Functionality**
   1. *The user interface must have a logical reading order that is programmatically determined.*
      1. A logical reading order is defined as how a user is intended to move through the interface AND how the order of elements affects meaning.
      2. Reading order must be determined programmatically to ensure assistive technology interprets it correctly.
   2. *All interactive elements must be reachable and usable by keyboard navigation. This includes map navigation functions.*
      1. If focus can be moved to an element via keyboard access, focus must be able to move away from the element (i.e. no keyboard “traps” – WCAG 2.0: 2.1.2)
      2. When a modal dialog is open, focus should be locked to the dialog until is dismissed by the user.
      3. A bypass method (“Skip to Content”) link must be provided to allow users to skip directly to content they are able to interact with.
   3. *All user interface elements must have programmatically assigned names, roles, and values to allow for assistive technology access.*
      1. Through ARIA or other means, all interactive elements must be discernible and operable by assistive technology.
   4. *Data and content must not change without user input (i.e. change should not happen when focus is put on an element).*
   5. *When user input initiates a change of context, this change must be communicated to assistive technology.*
      1. Upon activation, a button, field, or other method of interaction should provide feedback to assistive technology users. This feedback can be either through announced content or forced move of focus to the new element.
      2. Real time updates to information should also be communicated in the same manner.
   6. *An alternative method of input must be available if custom interactions methods are present*.
      1. Standardized control schemes, such as a digital keyboard or a button element, must be available if custom controls, such as a click/drag drop area, are implemented.
      2. Custom keyboard shortcuts should have another method available in the event that said shortcuts are used by assistive technology.
   7. *All content must avoid meet WCAG 2.3.1 “Three Flashes or Below Threshold.”* 
      1. This guideline is found here: <https://www.w3.org/WAI/WCAG21/quickref/?versions=2.0#qr-seizure-does-not-violate>.
   8. *Technological constraints should be recognized and users should be made aware of them.*
      1. If existing framework is unable to support any of the above guidelines AND impacts delivery of information, an accessible alternative must be provided. This alternative can take the form of a downloadable document or new webpage.
      2. If the widget or application mostly supports the above guidelines except for certain elements, such as hiding the tiles forming the map picture, users should be made aware of this through the element’s label OR prior to accessing the widget or application.

## General Techniques & Examples

Please note code examples are incomplete and may not function if copied. They are intended to show the semantics of accessible coding.

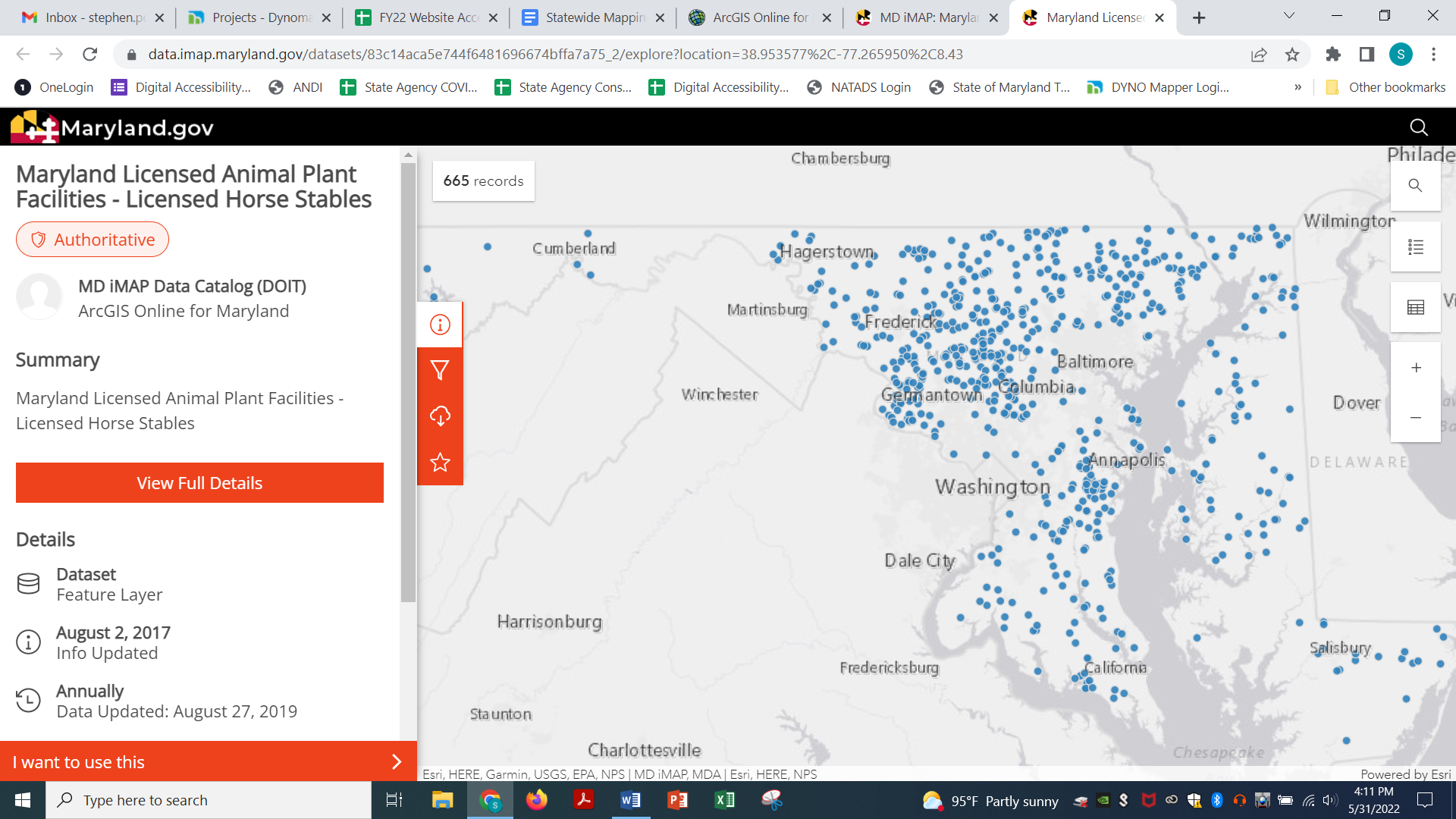
It is important to consider the user experience while developing your dataset and map. If a feature cannot be made accessible or does not provide a meaningful interaction, attempt to warn the user of the issue and direct them to an accessible alternative.

1. Text Alternatives
   1. When using standard HTML, the "alt" attribute is the best method of providing text alternatives to images. When using other languages, ARIA can be used to add an accessible name.
      1. EX: <img src='/map\_legend' alt='agency building'>
      2. EX: <svg aria-label='agency building'>
      3. EX: <svg aria-labelledby='legend\_item1'> <text id='legend\_item1'>Agency Building</text>
   2. To hide decorative or unneeded elements, a null alt attribute can be used. Please note that the attribute must be present in order to function in this capacity.
      1. EX: <img src=/map\_legend' alt="">
      2. EX: <svg aria-hidden='true'>
   3. Form controls must have a visual indicator of their function along with a programmatic label.
      1. (HTML) A text label can be programmatically associated with the input: <label for='next\_button'>Next</label><input type='button' value='next' id='next\_button'>
         * The label tag can also wrap around the input and not use the for attribute.
      2. (Non-HTML) Floating text can be programmatically associated using ARIA. Do not rely on placeholders to provide labeling. Example: <span>User Name:</span> <mat-form-field> <input matinput aria-required='true' id='mat\_input0' aria-label='User Name'> </mat-form-field>
         * Aria-labelledby may be used in the same manner as it is used for images.
   4. Images of text do not respond to accessibility settings such as font-size, contrast, and more. Thus, when it is possible, avoid using them. If for whatever reason images must be used, ensure the alt text description conveys the pictured text and the text itself meets WCAG contrast ratio requirements. It is recommended to use a font that has easily distinguishable characters.
      1. Regular text and links should be easily discernible; font size should be at least 12pt. To be considered large print, font size must be 14pt bold or 18pt regular.
   5. Audio that contains meaning, such as voiced instructions or feedback for a timer, must have captioning or a text equivalent.
      1. Captioning can be placed alongside the control scheme for the audio.
      2. Feedback, such as a ping or chime when a timer is complete, should use a visual indicator of occurrence such as an alert dialog or an icon appearing.
   6. [Please refer to this link for WCAG alternative text best practices.](https://www.w3.org/WAI/WCAG21/quickref/?versions=2.0&showtechniques=111#qr-text-equiv-all)
2. Accessible Alternatives
   1. Provide a text based list alongside any visually presented information, such as directions or building locations. This can be in the widget itself or a separate region on the page but it must be in the same location as the other form of information.
   2. Tables are an acceptable alternative to a list to display data. The important factors for accessibility of tables are to provide header cells (<th>) and provide a title or caption to explain the purpose of the table.  [W3C provides several tutorials and examples for different types of tables.](https://www.w3.org/WAI/tutorials/tables/)
   3. If neither method is available, an accessible document should be available for download. This document can be any kind so long as it adheres to best practices. PDF or Excel documents will be the most common type. [WCAG 2.0 provides guidelines for document accessibility.](https://www.w3.org/WAI/standards-guidelines/wcag/docs/)
   4. [This example is a simple explanation of providing multiple methods to access the same information.](https://docs.google.com/document/d/1D-ILZcl6BEhrUBeHcAjdlTNXVaY6N1BG/edit?usp=sharing&ouid=109684011666381781797&rtpof=true&sd=true)
3. Link Visibility & Descriptions
   1. Links must be visibly discernible from regular text. The simplest way to achieve this is to keep the default underline. So long as color is not the only method used here, other stylizations may work as well.
   2. Link description has two possible standards: WCAG 2.4.4 and 2.4.9. 2.4.4 states that link purpose can be conveyed in context; this is the minimum requirement for NVA and Section 508 compliance. 2.4.9 is the AAA standard and is quite easy to meet: link purpose must be conveyed via the link text alone. Usually, this only requires the name or phrase indicating the destination. If an acronym is used and there is no context for it (such as an agency acronym on a different agency’s website), a title attribute can be used to add the spelled out acronym. Title is the best choice here because it reads *in addition to* the link text. ARIA labels would overwrite the link text.
4. Visible Focus Indicator
   1. This is no different from standard WCAG solutions. [Sufficient techniques can be found here.](https://www.w3.org/WAI/WCAG21/quickref/?versions=2.0&showtechniques=247#qr-navigation-mechanisms-focus-visible)
   2. Please note that the focus indicator must be color-contrast compliant to avoid loss of functionality. The ratio for this is 3:1 with a minimum 1 pixel size.
5. Color Conveying Meaning
   1. Color must be paired with another visual indicator to convey meaning. Typically, this is done with an icon but other methods, such as texture, can be used.
      1. EX: An agency building marker is colored blue but is also represented by an office building icon.
      2. EX: The status of a construction zone is indicated by a color shape next to the location name. A legend is provided in the widget that shows the different shapes and colors and connects them to the relevant status.
      3. Note that these symbols should also have relevant labeling to ensure total accessibility.
   2. An exception is provided when color is the direct meaning of the content being delivered, such as the example of a topographical map indicating temperatures of zones. In these cases, it is best practice to attempt some form of accommodation. A tooltip that appears on focus that delivers pertinent information in text would be an acceptable workaround.
6. Color Contrast
   1. Color contrast can be checked with a number of online tools. [Level Access has one available here](https://www.levelaccess.com/color-contrast-checker/) and also has created a Chrome extension. [Adobe's Color Center also has a contrast analyzer](https://color.adobe.com/create/color-contrast-analyzer) that also has a palette generator for creating a completely compliant color scheme.
7. Logical Reading Order
   1. For HTML documents, the Document Object Model determines the reading order for assistive technology. Ensure that elements are coded in the order that a user is intended to use them.
      1. EX: A navigation menu button that opens and closes the menu is placed above the menu itself in the DOM order.
         * <div> <button id='nav\_menu\_button' aria-expanded='false' aria-controls='main\_menu'> <img src='/element' alt='Nav Menu Access'> </button> </div>
         * <nav id='main\_menu'> <ul> <li> <a> Home </a> </li> <li> <a> About </a> </li> </ul> </nav>
      2. If the elements are not correctly placed after each other in the order, AT will not interpret them correctly. In the below example, AT will focus the navigation menu first, followed by the logo and then the button that opens the menu.
         * <nav id='main\_menu'> <ul> <li> <a> Home </a> </li> <li> <a> About </a> </li> </ul> </nav>
         * <div> <img src='/home' alt='Agency logo'> </div>
         * <div> <button id='nav\_menu\_button' aria-expanded='false' aria-controls='main\_menu'> <img src='/element' alt='Nav Menu Access'> </button> </div>
   2. The best approach is to code pages and widgets as you intend a user to read and interact with them. The goal is to give AT and keyboard only users a streamlined experience.
8. Keyboard Navigation
   1. Use native HTML elements as much as possible. Ensure that the semantic meaning of the chosen HTML tag matches the intended behavior.
      1. If HTML cannot be used, refer to the next section “Names, Roles, and Values” for possible solutions.
   2. If an element is not in TAB order, such as a <div> styled as a button, then tabindex=’0’ may be used to add it to the TAB tree. This will place it according to the elements above and below it in the DOM order. Do not use any number above 0 because it will cause the element to be read before any other element. Please note that tabindex=’-1’ *removes* elements from the TAB order.
   3. Focus being locked to an open modal dialog keeps the dialog in reading order and prevents users from losing its place by accidentally moving focus to underlying elements. W3C provides an example here: <https://www.w3.org/WAI/ARIA/apg/example-index/dialog-modal/dialog>.
      1. The W3C GitHub also has some working examples: <https://w3c.github.io/aria-practices/examples/dialog-modal/dialog.html>.
   4. Bypass methods are typically used to skip repeated sections of websites, such as a header bar and navigation menu. The “Skip To Content” link should be the first focusable element on the page (or at minimum within the first three elements). For mapping applications, the link should move to the most accessible version of the material, bypassing the map controls to reach the results list or table.
      1. The link must be visible when focus is on it; it does not need to be shown until that point. Please note that properties such as ‘display: none’ should not be used so that focus can find the link.
9. Names, Roles, and Values
   1. In HTML, AT is able to interpret native elements and interact accordingly. However, some elements or non-HTML elements will not have the semantically-assigned behavior desired. ARIA is used to assign these behaviors with the 'role' attribute.
      1. Note: HTML native elements should be used where possible.
   2. Name, role, and value is the set of terms used by WCAG for the same purpose. [Sufficient techniques can be found here.](https://www.w3.org/WAI/WCAG21/quickref/?versions=2.0&showtechniques=412#qr-ensure-compat-rsv)
   3. For widgets and APIs, there are other attributes able to communicate states and properties of elements. In these cases, care must be taken to avoid misusing or misattributed ARIA as most elements will require multiple attributes to achieve full interoperability.
      1. [W3C provides guidelines and explanations of ARIA authoring practices.](https://www.w3.org/TR/wai-aria-1.1/) This resource also provides the framework and full list of ARIA attributes and roles and their associated behaviors.
10. Context Changes
    1. Changes in content and data should be initiated by the user activating elements. Change should not happen when focus is moved to an element to prevent users from losing information or being sent to unexpected new windows.
       1. [This is covered by WCAG 2.0 guidelines 3.2.1.](https://www.w3.org/WAI/WCAG21/quickref/?versions=2.0&showtechniques=321#qr-consistent-behavior-receive-focus)
    2. Please note that dynamic information is not covered by this. Real time updates will fall under the 2.4 guideline.
    3. User input requires feedback; when a button is pressed (or similar interaction), it may be necessary to force focus to the content change initiated or announce it with a live region.
       1. Be sure to only match expected behaviors; a user should understand what will happen when an interaction occurs. Most native HTML elements and ARIA roles will provide what is needed.
       2. If focus should be moved, such as opening a entry from a flag on the map, Javascript is able to do this: focusMethod = function getFocus() {document.getElementById("UniqueID").focus();}
          * Please note this should only be done in order to skip elements. If the next element is the related element, it is unnecessary to move focus.
       3. Aria-live regions are useful for announcing when input initiates a change in content. It is also useful for announcing real time updates to content, such as a news ticker. Be careful to avoid spamming users; updates should be fairly spaced apart (20-30 seconds).
          * [Deque has a useful tool for setting up a simple live region.](https://dequeuniversity.com/library/aria/liveregion-playground)
          * Some roles have 'live' built into them; role='alert' is one such example. Please refer to the W3C ARIA authoring guide for more information.
11. Control Schemes
    1. When using custom gestures or areas that require specific cursor movement, such as a drag & drop area for file uploads, a traditional method must be available. This also includes shortcuts as some users with physical disabilities may not be able to press the corresponding keys and those using AT may use that shortcut for another purpose.
       1. EX: Google Drive has a drag & drop region in the center of the screen but also provides an upload new file button in the menu under the New button.
       2. On mobile, this may require having a separate menu for interactions or allowing the use of the digital keyboard.
12. Technological Constraints
    1. Notifications of potential accessibility issues or constraints should be placed prior to the actual widget if embedded in a page. If the widget opens in a new window, it should have this notice within the first three elements. Provide users with links to accessible alternatives if possible.
    2. A good rule of thumb is to add this notice to any descriptions of the content of the widget that are available.
    3. For ESRI, since most of WCAG is partially supported ([their completed Voluntary Product Accessibility Template reports can be found here](https://www.esri.com/en-us/legal/accessibility/conformance-reports)), find what is not supported and use other elements in relative context to convey the necessary information. For example, flags on the map may not be labeled but if the flag opens a tooltip on focus, the tooltip can be read by screen readers.

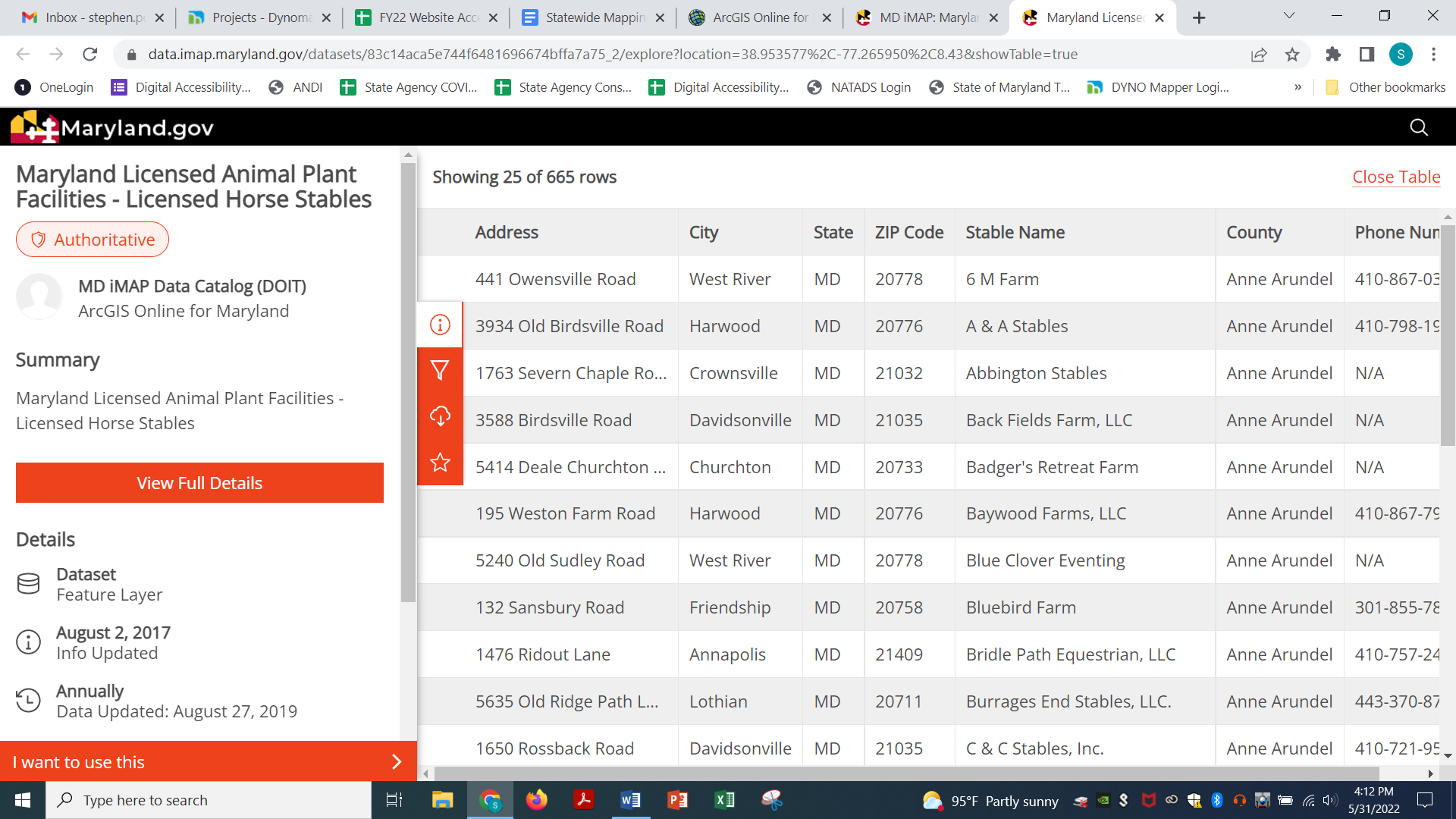
## Accessible Alternatives Example

The following example uses screenshots from opendata.maryland.gov, from the Licensed Horse Stables dataset. It is displayed in the ESRI mapping tool.

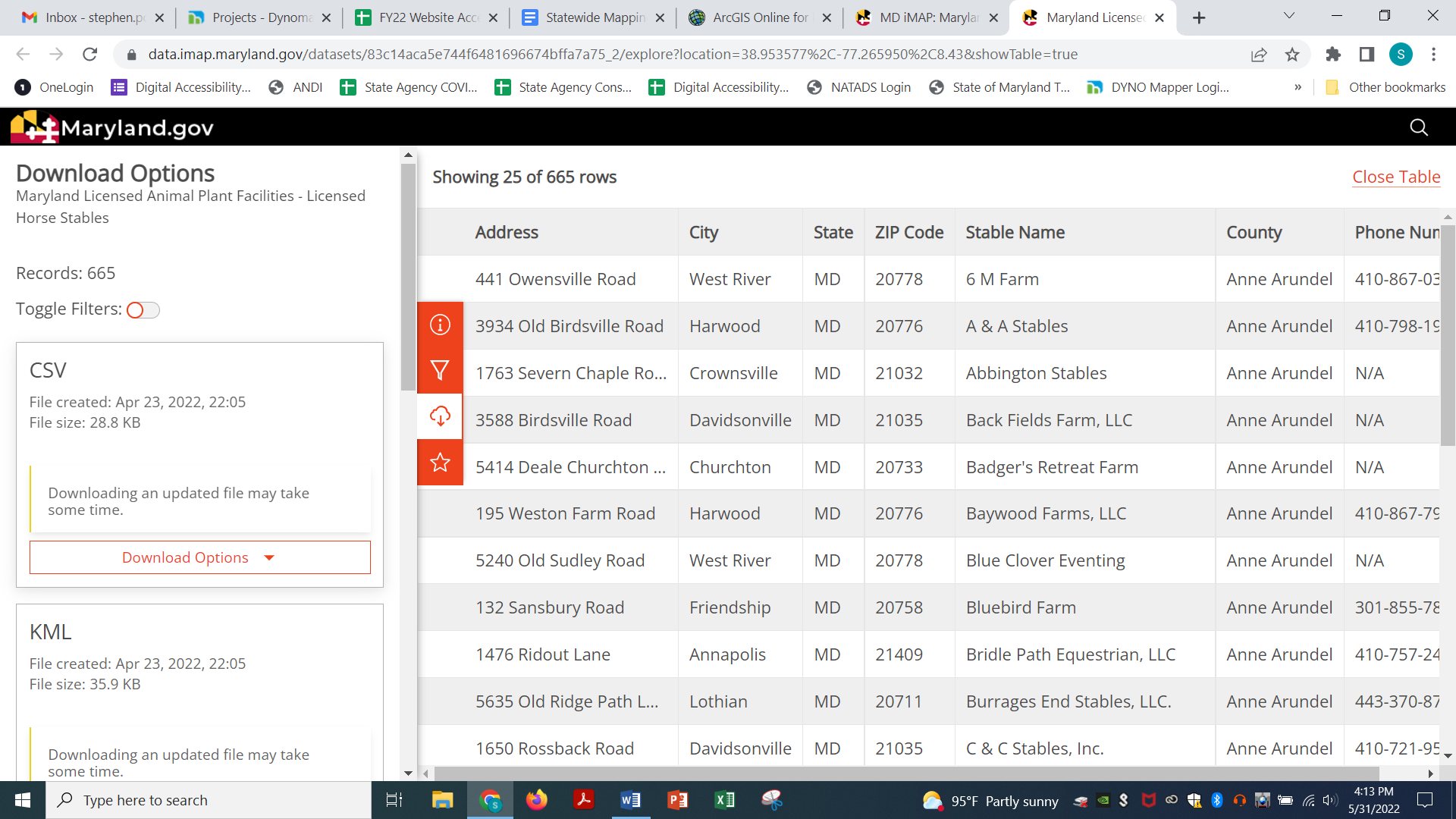
In this example, data points are represented on the map via blue dots. Hovering over a dot shows a tooltip displaying the location’s name and address. Since this dataset is only showing one type of data point, there is no need to distinguish between types with a legend.



For the sake of the example, it is determined that these dots are not labeled correctly for screen readers. Additionally, there small target size and proximity, even when zoomed in, may present an issue for a user with a physical disability. However, the third button on the right opens a table with the dataset. This table is also reachable via a link in the information region on the left off-screen in the screenshot.



This data table contains all of the results shown on the map with different methods of filtering. It is also properly marked-up HTML with ARIA to enable screen readers to move freely inside it, connect data points with headers, and announce filter changes. If for whatever reason a user is unable to use this data table, there is also an option to download the dataset in different formats that may be more accessible.



By providing both of these options, this web app provides accessible alternatives to reach the same information a user can gain from the map itself. Ideally, the blue dots function as buttons to jump to the intended result in the data table. Datasets containing multiple types of information will require linking the legend results to the table as well so users can quickly pick which category they wish to view.

## ESRI Mapping Accessibility Guidance

This section is a summary of various resources available from ESRI (<https://www.esri.com/en-us/accessibility/resources>). Listed below are techniques specific to ESRI’s ARCGIS application and additional resources they provide. Please note some of these techniques may not be applicable depending on what platform you are using. This section should updated as changes are made to ESRI products in use that enhance accessibility. ESRI’s accessibility team tests and reports on their products regularly. For questions or reporting issues, contact [esriaccessibility@esri.com](mailto:esriaccessibility@esri.com).

The state of Minnesota has created several useful resources as well for map accessibility: <https://mn.gov/mnit/about-mnit/accessibility/maps/>. Their [guide for accessible web maps](https://mn.gov/mnit/assets/Accessibility%20Guide%20for%20Interactive%20Web%20Maps_tcm38-403564.pdf) includes a checklist that is helpful when developing new content.

* Image Alt Text: In StoryMaps, click the Properties menu for a media element. The dialog option will show a text field for adding alt text for the element.
  1. ESRI YouTube Guide: <https://www.youtube.com/watch?v=bJ40v5lxy0A>
* Keyboard Navigation: ARCGIS uses Windows built-in Mouse Keys, turning the number pad into a virtual mouse. It also provides a list of keyboard shortcuts.
  1. ESRI provides a set of instructions for using ARCGIS with keyboard navigation and other assistive technology. Make sure to provide this link along with the application to assist disabled users: <https://pro.arcgis.com/en/pro-app/latest/get-started/overview-of-accessibility.htm>
* Accessible Basemaps: ESRI has a number of basemaps that have taken accessibility into consideration. These can be found on their accessibility resources page.
  1. The most recent creation is a high contrast version: <https://www.esri.com/arcgis-blog/products/arcgis-living-atlas/mapping/working-with-enhanced-contrast-basemaps-to-improve-accessibility/>.
  2. Andy Skinner, the author of most of these basemaps, is a major content creator. His profile provides a number of other useful resources: <https://www.esri.com/arcgis-blog/author/ajskinner2/>.
* Legend Information: If icons are unlabeled, it is extremely important to put equivalent information in text. The space next to the icons is focusable even without content. For example, a category in a legend states Zipcodes Serviced. The icon below it shows the border color for the zipcodes but there is no text. Therefore, users with low vision or using assistive technology have no contextual information about what is indicating the serviced zipcodes. At the very least, having text next to each “blank” will communicate to users there are entries for these categories.
  1. ESRI presentation: <https://www.youtube.com/watch?v=miTPT68kjto> – contains useful information on symbology
  2. ESRI also provides accessible basemaps to use
* JavaScript API: ARCGIS Online Web Apps supports the use of semantic HTML and ARIA, if needed. The linked presentation below explains how testing can be conducted using any browser and how to implement fixes.
  1. ESRI presentation: <https://www.youtube.com/watch?v=HJlIdCs5Z54>
* Accessible PDFs ARCGIS Pro provides the ability to create an accessible PDF in it, instead of having to remediate the document in Acrobat. It is extremely important to check the Accessibility tab in the Export menu to include tags; otherwise, all your work will be undone.
  1. Guide: <https://www.esri.com/arcgis-blog/products/arcgis-pro/mapping/creating-accessible-pdfs-in-arcgis-pro/>
* [DATA TABLE CREATION GUIDE IF AVAILABLE]