Progressive Insurance

# Accessibility Toolkit

[Introduction 3](#_Toc3811239)

[What is accessibility? 3](#_Toc3811240)

[What makes a web site accessible? 3](#_Toc3811241)

[Standards 3](#_Toc3811242)

[Accessible design standards overview 4](#_Toc3811243)

[Accessibility for content creators and editors 5](#_Toc3811244)

[Heading levels and document outline 5](#_Toc3811245)

[Correct 5](#_Toc3811246)

[Incorrect 5](#_Toc3811247)

[Links 6](#_Toc3811248)

[Understandable content 6](#_Toc3811249)

[Alt text 6](#_Toc3811250)

[Accessibility for designers 7](#_Toc3811251)

[Proximity 7](#_Toc3811252)

[Fonts 7](#_Toc3811253)

[Color 7](#_Toc3811254)

[Keyboard accessibility 7](#_Toc3811255)

[Resources 7](#_Toc3811256)

[Accessibility for Developers 8](#_Toc3811257)

[Priority 1 8](#_Toc3811258)

[Ensure full keyboard-only interaction 8](#_Toc3811259)

[Provide HTML ALT attributes for all IMG tags 8](#_Toc3811260)

[Provide visual focus indication 8](#_Toc3811261)

[Provide a unique HTML "Title" tag on each page 8](#_Toc3811262)

[Use HTML Header tags, Skip links and/or Landmark elements 8](#_Toc3811263)

[Associate Labels with Form Elements 9](#_Toc3811264)

[Ensure focus on modal dialogs 9](#_Toc3811265)

[Set the Language in the HTML 9](#_Toc3811266)

[Priority 2 10](#_Toc3811267)

[Ensure proper use of color and adequate contrast 10](#_Toc3811268)

[Provide notification of dynamic changes to content 10](#_Toc3811269)

[Links, buttons & lists 10](#_Toc3811270)

[Ensure use of responsive designs 10](#_Toc3811271)

[Tables – associate column headers with row data. 11](#_Toc3811272)

[Ensure accessible error messaging 11](#_Toc3811273)

[Details 11](#_Toc3811274)

[Priority 3 11](#_Toc3811275)

[Ensure functionality and readability with modified Font size and spacing 11](#_Toc3811276)

[Caption and transcript videos 11](#_Toc3811277)

[Avoid timed events 11](#_Toc3811278)

[Avoid blinking elements 12](#_Toc3811279)

[Ensure the Acronyms, Abbreviations & Numbers read appropriately 12](#_Toc3811280)

[Support the use of the ‘autocomplete’ for primary form entry fields 12](#_Toc3811281)

[Quality Assurance Methods 13](#_Toc3811282)

[Priority 1 Ensure full keyboard-only interaction 13](#_Toc3811283)

[Provide HTML ALT attributes for all IMG tags 13](#_Toc3811284)

[Provide a unique HTML "Title" tag on each page 13](#_Toc3811285)

[Use HTML Header tags, Skip links and/or Landmark elements 13](#_Toc3811286)

[Associate Labels with Form Elements 13](#_Toc3811287)

[Ensure focus on modal dialogs 13](#_Toc3811288)

[Set the Language in the HTML 14](#_Toc3811289)

[Priority 2 15](#_Toc3811290)

[Ensure proper use of color and adequate contrast 15](#_Toc3811291)

[Provide notification of dynamic changes to content 15](#_Toc3811292)

[Links, buttons & lists 15](#_Toc3811293)

[Ensure use of responsive designs 15](#_Toc3811294)

[Ensure accessible error messaging 15](#_Toc3811295)

[Priority 3 16](#_Toc3811296)

[Ensure functionality and readability with modified Font size and spacing 16](#_Toc3811297)

[Caption and transcript videos 16](#_Toc3811298)

[Avoid timed events & blinking elements 16](#_Toc3811299)

[Ensure the Acronyms, Abbreviations & Numbers read appropriately 16](#_Toc3811300)

[Support the use of the ‘autocomplete’ for primary form entry fields 16](#_Toc3811301)

[Sample testing protocol 18](#_Toc3811302)

[Mobile 19](#_Toc3811303)

[IOS devices 19](#_Toc3811304)

[Android devices 19](#_Toc3811305)

[ARIA Roles and States 20](#_Toc3811306)

[ARIA Roles 20](#_Toc3811307)

[ARIA States 21](#_Toc3811308)

[ARIA Properties 21](#_Toc3811309)

[Resources 22](#_Toc3811310)

[Reference, checklists, training 22](#_Toc3811311)

[Articles 22](#_Toc3811312)

[Tools 22](#_Toc3811313)

[Accessibility for Testers 23](#_Toc3811314)

[Tools for testing accessibility 23](#_Toc3811315)

[Web aim 23](#_Toc3811316)

[Google 23](#_Toc3811317)

[Progressive 23](#_Toc3811318)

[Other 23](#_Toc3811319)

## Introduction

### What is accessibility?

Accessibility at Progressive means making sure our websites, applications and content are can be used by as many people as possible, without regard to physical, perceptual, or cognitive disability. This includes customers, agents, partners, and employees.

* Users with impaired vision may have difficulty when viewing a specific color spectrum or contrast between colors; that may require a larger text or specific fonts.
* Users who are functionally non-sighted may require the use of an assistive technology such as a screen reader.
* Users with a hearing impairment may require auditory content with subtitles.
* Users who have difficulty using a mouse or keyboard may require alternative hardware or software to complete forms.
* Users who have cognitive challenges may have difficulty interpreting complex statements, paragraphs or vocabulary.

### What makes a web site accessible?

The site must be Perceivable, Operable, Understandable and Robust (POUR) for all users.

* **Perceivable**: Available to the senses (vision and hearing primarily) either through the browser or through assistive technologies (e.g. screen readers, screen magnifiers, etc.)
* **Operable:** Users can interact with all controls and interactive elements using either the mouse, keyboard, or an assistive device.
* **Understandable**: Content is clear and limits confusion and ambiguity.
* **Robust:** A wide range of technologies (including old and new user agents and assistive technologies) can access the content.

### Standards

The Web Content Accessibility Guidelines (WCAG) are the generally recognized standard for accessibility world-wide and are what the majority of businesses follow.

WCAG was developed and is maintained by the World Wide Web consortium (W3C). WCAG was recently updated and the official standard that Progressive seeks to comply with is WCAG 2.1 level AA. The WCAG “levels” are hierarchical; to meet AA standards, a site must first meet all level A standards, in addition to all level AA standards.

The official list of WCAG 2.1 AA standards can be found at the [**W3.org** site](https://www.w3.org/WAI/intro/wcag).

There is a list of accessibility tools and information resources at the end of this document.

### Accessible design standards overview

Below are some specific accessibility standards. These are included in the WCAG standard, though the list is not all-inclusive. As you can see, most of these are common-sense standards that would benefit any user. Some exist to ensure that popular assistive tools such as screen readers will be able to interpret the content and navigate the site.

* Ensure that everything on-screen can be interacted with using only the keyboard.
* Provide a unique HTML title tag for each page.
* Apply HTML headings (e.g., H1, H2, H3, etc.) hierarchically. Heading tags should reflect the organizational structure of the document.
* Use semantic HTML tags to ensure that links and button are correctly identified. Links should be links (i.e. anchor tags) and buttons should be buttons (i.e. button tags).
* Set the page's language using HTML.
* Provide an easily visible indication of which item on-screen has the 'focus'.
* With each page, focus should begin on the header or topmost field.
* When a window pops up, focus should move there.
* Do not display meaningful content using CSS alone. Verify that all text can be read, and elements are visible with CSS ‘off’.
* Avoid blinking elements.
* Use HTML semantically. Don't let style dictate.
* Provide alternative text for screen images. Every image tag must include an alt attribute.
* Don’t hide important text inside images.
* Provide a skip link to allow users to navigate past repetitive page content directly to an anchor or target at the beginning of the main content
* In forms, use the "for" attribute to associate label tags with their form element.
* Design for high contrast between foreground / text and background.
* Error messages must be apparent and accessible.
* Set the language of the page in the HTML tag.
* Text should be resizable up to 200% without loss of content or functionality.
* Do not use color alone to convey meaning, including in text (e.g. “click red button”).
* Link text should have meaningful labels (avoid "click here").
* Provide multiple ways of navigating.
* Videos must include captions and/or transcripts.
* Provide text for any audio information.
* All text on-screen must be accessible by text to speech screen readers.
* Provide the ability to pause timed events.
* Make sure content is clearly written and easy to read.
* Use semantic HTML and ARIA to ensure correct screen-reader interpretation.
* [ARIA roles, states and properties](https://www.progressive.com/dev/training/accessibility/standards/aria/) are dynamically implemented where appropriate.

## Accessibility for content creators and editors

### Heading levels and document outline

Ensure that the page's document outline follows the correct hierarchical order. Just as you would write an outline for a report or paper starting with the main topic, then adding subtopics beneath it, ensure the outline of a web page follows this same hierarchical structure. Users using screen readers depend on the document's HTML headings follow the outline for correct navigation.

* Include one and only one H1 tag.
* Do not skip heading levels.
* There can be as many headings of each level from H2 down as needed to organize the text, but the hierarchical relationships must be maintained.
* All H2 must follow H1, with no lower level tags (H3, H4, etc.) in between.
* The same rule applies at each level: No H4s between an H2 and the next H3.

#### Correct Heading Structure

* H1
  + H2
    - H3
      * H4
    - H3
      * H4
    - H3
  + H2
    - H3

#### Incorrect Heading Structure

* H1
  + H2
    - * H4
  + H2
    - H3
    - H3

There may be times when the correct document outline does not fit the desired visual design. In these cases, use CSS classes with heading levels to modify the visual style while retaining the correct hierarchy.

Progressive.com has a CSS style for each heading type.

You can use the [**ANDI**](https://www.ssa.gov/accessibility/andi/help/install.html) or [**WebAIM WAVE Chrome extension**](http://wave.webaim.org/extension/) to verify that the document order is correct.

### Links

Ensure that every link's purpose is clear from its context. For example, a link that reads "Learn more" has no meaning when taken out of context. Screen reader users see a list of links in their screen reader and can navigate by them; however, if the link text is vague, they won't know what to expect when clicking on the link. Consider changing to, "Learn more about Progressive Homeowners Insurance," for example.

Screen readers may become confused when two links that go to the same URL are adjacent. Therefore, if you have a graphical element and some text that both link to the same URL, combine them into a single linked element when possible.

### Understandable content

Make sure content is clearly written and easy to understand.

### Alt text

Visually impaired users cannot see images, so we must take steps to ensure that screen readers handle images correctly.

Every image tag on a site must include an alt attribute to describe it.

* If an image contains text, quote that text exactly in the alt text.
* If an image contains visual information, explain what is being shown.
* If an image is nothing purely decorative, leave the alt text blank.

## Accessibility for designers

Designers have the following major concerns when it comes to making a page accessible:

### Proximity

* Users with limited field of vision or who use large text work within a narrow field of view. If related information is visually spread out User can an have trouble knowing about it, seeing it, and using it.
* Keep related information such as labels and fields in close proximity.
* Manage user expectations when new information is displayed such as a link that opens a new browser tab

### Fonts

* Text should be larger than 10px, preferably at least 14px.

### Color

* Do not rely on color alone to convey information. For example, instead of all required fields in a form having a red outline, also give them an asterisk next to their labels. That way, people who are color blind can still understand which fields are required.
* WCAG requires a contrast ratio of 4.5:1 for normal text and 3:1 for large text. Normal text is defined as approximately 14px. Large text is defined as approximately 18px or larger or 14px and bold.

### Keyboard accessibility

* Provide visual indication with the proper contrast ratio for all actional elements that receive focus as the user tabs through it.
* Make sure all actionable fields, buttons and links are keyboard-accessible and can be reached and fully interacted with using the appropriate keystrokes depending on the control type.
* Keep field labels outside of the fields, if possible.

### Resources

* Use the [**WebAIM Color Contrast Checker**](http://webaim.org/resources/contrastchecker/) to ensure contrast is correct.
* Review **WebAIM's** [**information on accessible fonts**](http://webaim.org/techniques/fonts/) for more details.
* Review [**WebAIM’s information on Keyboard accessibility**](https://webaim.org/techniques/keyboard/)for general guidance
* Review the [**WAI-ARIA Authoring practices for detailed keyboard accessibility**](https://www.w3.org/TR/wai-aria-practices/)guidance
* [**Design.progressive.com**](file:///C:\Users\THENRY1\Desktop\•%09Design.progressive.com) provides the Progressive branded colors and their color contrast ratios.
* Review the [**W3C Accessibility Requirements for People with Low Vision**](https://www.w3.org/TR/low-vision-needs/#proximity-of-related-information)

## Accessibility for Developers

Occasionally you will have to use ARIA roles and states to allow screen readers to understand for example a tabbed interface, so the user can interact with it using only a keyboard. Progressive’s accessibility standards are based on the WCAG 2.1 Level AA standard. Progressive’s toolkit organizes requirements by priority. All systems should eventually meet all requirements regardless of priority. Priorities are intended only to assist with project prioritization and resource allocation.

### Priority 1

These standards are the most important. Ensure that every page and every site comply with these before being published.

#### Ensure full keyboard-only interaction

* Ensure that all actionable elements are reachable and controllable with keyboard only.
* Do not create keyboard traps (i.e., areas where you get stuck and can no longer reach the rest of the fields).
* Ensure a logical tab order capturing all actionable elements
* No invisible tab stops (e.g. links with no visible content)

##### Details

* Buttons, radio buttons, dropdown menus, and other elements must be accessible using standard keystrokes. Use this **[keystroke cheat sheet](https://webaim.org/techniques/keyboard/" \t "_blank)** **[external]** to code your elements appropriately for keyboard accessibility.
* If an actionable element isn't gaining focus, add tabindex="0".
* Avoid use of tabindex numbering to define the tab order (i.e. tabindex="1", tabindex="2", etc.)
* Use tabindex="-1" only in very rare circumstances when the user should not interact with the element in any way.

#### Provide HTML ALT attributes for all IMG tags

Provide alternative text for screen images. Each image tag must include an alt attribute.

##### Details

* All alt text content should be provided or approved by the designer or copywriter
* For images that are decorative only, add alt="" to the image tag.
* For images that contain useful information or function, enter the provided description/caption in the alt attribute.
* CSS based images should be used only in cases where the image is decorative. Other images must use the proper HTML image tags

#### Provide visual focus indication

Ensure that whenever a screen element gains focus, there is visual indication of the focus through for example a border highlight.  
  
Details

• The method and presentation of the visual focus will be provided by the designers

• Do not disable the property in CSS unless an alternative focus indicator is provided by the designers

• Ensure the visual focus indication provides the required contrast against background elements

#### Provide a unique HTML "Title" tag on each page

Ensure that all pages, including dynamically generated ones, have a unique title tag.

##### Details

* Page title text will be provided or approved by the designer or copywriter.
* Add a unique title tag to the page, preferably hard-coded or server-side — not dynamically creating using JavaScript after the DOM has loaded.

#### Use HTML Header tags, Skip links and/or Landmark elements

People with disabilities need methods to quickly navigate through content. This is accomplished by providing semantic structure to the content that can be navigated using assistive technology:

* All content should have a well-defined structure using semantic **H1**, **H2**, **H3**, etc. tags to define the document outline of the page as provided or approved by the designer or copywriter
* Skips links are encouraged when significant repeated content should be bypassed.
* ARIA landmarks are also encouraged to provide additional navigation of the page structure.  
   Details
* **Headers** - required on each page should have a main header and associated sub-section headers
  + The most important heading of the page should be an H1 tag. The second most important should be an H2, etc.
  + Don't let style dictate the heading order. If the designer wants an H3 tag on the page to look like an H2 for example, don't change the H3 to an H2 just so the style will work.
* **Skip Links**
  + Strongly recommended for pages with complex heading structure and/or navigation prior to the main content
  + Position the "Skip to Main Content" link off the page on page load.
  + Make the skip link the first focusable element, so when the user hits their tab key, the Skip link gets focus and is displayed on the page.
  + The Skip link should take the user to the first heading or form field on the page.
* **Landmarks**
* Recommended when you have significant page structure such as headers, footers, navigational areas and main content to offer enhanced navigation through assistive technology
* All content must fall within a landmark region to avoid orphaned content
* See the [**ARIA Landmark documentation**](https://www.w3.org/TR/WCAG20-TECHS/ARIA11.html)(external) for more information

#### Associate Labels with Form Elements

Associate label tags with their associated form element.

##### Details

* Add a "for" attribute to all form labels. The "for" attribute value of the label tag must match the ID value of the associated form element.
* Ensure the visible name of the control matches the accessible name so that when using a tool such as Dragon Naturally Speaking the user can use the control by speaking the text that appears on the screen.
* Ensure that required fields are announced and visually indicated to the user.
* Supporting labeling such as date mask needs to be visually associated and announced to the user

#### Ensure focus on modal dialogs

When the user opens a modal dialog, place the focus on the modal so the user's screen reader will read it off.

##### Details

* Ensure the modal or an element within the modal such as a header has focus.
* Return focus to the element that activated the modal when the modal is closed using the Esc key or via a close button.
* Ensure that focus stays within the modal until the modal is closed.

#### Set the Language in the HTML

The language attribute is used by screen readers to provide the correct accent and pronunciation

##### Details

* Set the primary language attribute in the HTML tag for all documents
* Set a language attribute for sections of text in a language different from the primary language to ensure correct accent and pronunciation of the content

### Priority 2

#### Ensure proper use of color and adequate contrast

Ensure all meaningful (non-marketing) content and actionable elements meet contrast requirements

##### Details

* Designers are responsible for delivering content and controls with the proper color contrast based on WCAG requirements and the Progressive accessible brand color palette.
* This includes, but is not limited to: all meaningful content including graphical elements, buttons, form fields, focus indicators, and state indicators (hover, focused, checked, selected, current item, etc.)
* Ensure that all content displays and adjusts when High contrast viewing mode is enabled through the OS or browser

• Ensure that color alone is not used to convey meaning or association, indicate an action, prompt a response or as a state indicator.

#### 

#### Provide notification of dynamic changes to content

When dynamic content is updated automatically or in response to a user action, notify the user. This particularly important for content above or prior to the current point of focus.

##### Details

* Designers should specify the workflow and interactions associated with dynamic changes
* One way to achieve this through a limited use of aria-live attributes.

#### Links, buttons & lists

* Ensure links are identified by screen readers as links, buttons are identified as buttons and lists as lists. Use descriptive text on both buttons and links.

##### Details

* Links
  + For an element that navigates to another page or within a page, use HTML link syntax.
  + Links must have a non-empty href attribute to be considered true links and accessible to keyboard users.
  + Links that open in a new window or tab are indicated (also external links)
  + Make links descriptive and unambiguous for use when presented with an assistive technology by referencing the phrase or sentence that refers to the content behind the link. This allows users with sight-disability to navigate the page using only links.
  + Links should be underlined but when not possible, ensure adequate contrast between links and page text. Provide a non-color designator on both mouse hover and keyboard focus, such as an underline, to assist users with color deficiencies in their vision.
  + If the copywriter/legal doesn't want or can't have enough text linked to provide context as to where the link will take you, use aria-label= to better describe the link.
* Buttons
  + For an element that initiates an immediate action, such as form submission, use a button not an image.
  + Use the HTML button tag - DO NOT use a styled hyperlink, DIV or SPAN for buttons.
  + Use CSS to style buttons if required.
  + If you cannot use a button tag for some reason, set an ARIA role of "role=button" on the tag.
* Lists
  + Ensure that lists have the proper semantic code structure so that users can identify grouped items as lists

#### Ensure use of responsive designs

Details:

* This includes supporting users who adjust the page to:
  + Different browser window sizes and resolutions
  + Portrait and landscape modes
  + Zoom text size up to 400%

Tables – associate column headers with row data.   
  
An accessible data table needs to define the relationships between the data cells and their corresponding columns and rows programmatically. Doing so enables assistive technologies to understand the relationships between the data cells and their corresponding columns and rows which can be communicated to a non-sighted user in a way the user can understand.

* In simple tables mark-up tables with the <TH> element in the first row to make the information is usable by assistive technology
* Create header/ID associations by giving each header a unique ID or scope to associate related cell content in complex tables
* Data tables very often have brief descriptive text before or after the table that indicates the content of that table. This text should be associated to its respective table using the <caption> element.

#### Ensure accessible error messaging

Ensure links are identified by screen readers as links and buttons are identified as buttons. All error messages need to be programmatically associated with the form field(s) or focusable within the tab order

##### Details

* Alert the user to the presence of the error in an apparent and accessible manner.
* Ensure that the error text is programmatically associated with the form field
* Allow the user to easily access the form controls that need to be modified such as moving the users focus to the first error
* Allow resubmission and revalidation of the form.

### Priority 3

#### Ensure functionality and readability with modified Font size and spacing

##### Details

##### Ensure that the user can increase font size by 200% without breaking or loss of content.

##### People with low vision or dyslexia may override text spacing to enable readability. Ensure that when users override text spacing via style sheet or other settings, there is no loss of content or functionality when the following adjustments are made:

##### Line height (line spacing) to at least 1.5 times the font size;

##### Spacing following paragraphs to at least 2 times the font size;

##### Letter spacing (tracking) to at least 0.12 times the font size;

##### Word spacing to at least 0.16 times the font size.

#### Caption and transcript videos

Ensure videos include closed-captioning and transcripts.

##### Details

* YouTube allows you to add captions to videos, either manually or automatically.
* Provide a link close to the video that allows the user to read a transcript of the video.

#### Avoid timed events

If any timed events are required, provide the user with a way to pause the timing. People need enough time to read info before they are taken away from it especially those using an assistive technology

##### Details

* Allow the user to click a link to continue instead of automatically redirecting.
* Pause animation on focus.

#### Avoid blinking elements

Do not have anything blinking more than 3 times per second. Quick flashing elements could cause seizures in some people and discomfort in many others.

##### Details

* Control the timing of the animation via CSS/jQuery/JavaScript or avoid the blinking completely.

#### Ensure the Acronyms, Abbreviations & Numbers read appropriately

Screen readers will attempt to read abbreviations as words and will read numbers in total rather than as individual digits.

##### Details

* Ensure that abbreviations and acronyms such “UM” aren’t pronounced as a word but rather each individual letter announced or the meaning “Uninsured motorist” is provided
* Ensure that larger numbers such as policy number are read as individual digits.

#### Support the use of the ‘autocomplete’ for primary form entry fields

Select HTML5 form entry fields should be tagged using autocomplete to assist users in form entry tasks.

##### Details

Here is a list of autocomplete fields that are supported

|  |  |  |
| --- | --- | --- |
| Content type | name attribute | autocomplete attribute |
| Name | name fname mname lname | * name (full name) * given-name (first name) * additional-name (middle name) * family-name (last name) |
| Email | Email | * email |
| Address | address city region province state zip zip2 postal country | * For one address input:   + street-address * For two address inputs:   + address-line1   + address-line2 * address-level1 (state or province) * address-level2 (city) * postal-code (zip code) * country |
| Phone | phone mobile country-code area-code exchange suffix ext | * tel |
| Usernames | Username | * username |

**Note:** Use either only **street-address** or both **address-line1** and **address-line2**. **address-level1** and **address-level2** are only necessary if they're required for your address format.

## Quality Assurance Methods

### Priority 1 Ensure full keyboard-only interaction

Required tool(s): Keyboard only

* Use your tab key to ensure you can reach every actionable item including links, fields, buttons, clickable images and icons.
* Ensure a logical tab order including modal windows
* Ensure there is a visual focus indication on all actionable elements
* Verify there are no keyboard traps
* Verify there are no invisible tab stops (e.g. links with no visible content)
* Use this [**keystroke cheat sheet**](https://webaim.org/techniques/keyboard/) **[external]** to ensure elements, especially custom controls can be correctly interacted with using the keyboard.

#### Provide HTML ALT attributes for all IMG tags

Required tool(s): Tenon, WAVE, ANDI, Chrome Accessibility Developer Tools

* Validate that the alt text matches the requirements provided by the designer or copywriter

#### Provide a unique HTML "Title" tag on each page

Required tool(s): Tenon, WAVE, ANDI, Chrome Accessibility Developer Tools

* Using Andi select the “structures” from the drop down and the more details option to the right to select Page title

#### Use HTML Header tags, Skip links and/or Landmark elements

Required tool(s): Tenon, WAVE, ANDI, Chrome Accessibility Developer Tools

* Using either ANDI:
  + validate heading structure and order for accuracy
  + If present, validate skip link presentation and functionality
  + If present, validate landmark outline and ensure that no content is orphaned

#### Associate Labels with Form Elements

Required tool(s): Tenon, ANDI, Chrome Accessibility Developer Tools, JAWs inspect

* Using ANDI select ”focusable elements” from the drop down and then select “labels” on the right. Using the arrows icons to review label display in ANDI each label for accuracy or errors
* Running the JAWS inspect “full report “provides that associated labels for all form elements
* Validate form masks that appear when a field has focus such as DOB by Running the JAWS inspect “under mouse report” on the field in question

#### Ensure focus on modal dialogs

Required tool(s): Keyboard

* Manual test via keyboard to ensure focus and a method to close the dialog

#### Set the Language in the HTML

Required tool(s): Tenon, ANDI, Chrome Accessibility Developer Tools

ANDI provides and option under the Structure and more information option to display the page language and other language attributes within the page

### Priority 2 Ensure proper use of color and adequate contrast

Required tool(s): Tenon, ANDI, Chrome Accessibility Developer Tools, Chrome Google High Contract extension

* Using ANDI select the contrast option to validate page elements
* Optionally to validate select elements use the eye dropper and the WebAIM Contrast Checker to verify correct contrast ratios.
* Test using High contrast mode using the Chrome extension. Ensure that all images content (images, icons, text and buttons) are visible

#### Provide notification of dynamic changes to content

Required tool(s): JAWS, NVDA

* Manual testing with JAWS

#### Links, buttons & lists

Required tool(s): Tenon, ANDI, Chrome Accessibility Developer Tools, JAWS Inspect

* Using ANDI validate:
  + Links are descriptive of what the link is and where it takes you.
    - That link visually appear as links rather than as a button or other visual element
    - Verify that additional visual indicators or content identifies when a link opens a new tab or page
  + Buttons are descriptive of the action and function as a button to submit information
  + Manually validate that all associated content such as bulleted or numbered lists are coded as a list

#### Ensure use of responsive designs

Required tool(s): Tenon, ANDI, Chrome Accessibility Developer Tools

* Use the Chrome settings option to zoom the page to 400%.
  + Validate that the pages changes viewport and doesn’t require scrolling in more than one direction
* Portrait and landscape modes can be validated by changing browser window dimensions

##### Tables – associate column headers with row data

Required tool(s): JAWS, Tenon, ANDI, Chrome Accessibility Developer Tools

* Validate that cell content is associated with the column reader

#### Ensure accessible error messaging

Required tool(s): JAWS, Tenon, ANDI, Chrome Accessibility Developer Tools

* Use **JAWS** to ensure when error messages fire, the user knows what the error message is, can get to the field to fix the error and can resubmit the form for submission.

### Priority 3

#### Ensure functionality and readability with modified Font size and spacing

Required tool(s): Favlet

* Using the favlet verify that the page still reads correctly looking for things like cutoff text
* Line height (line spacing) to at least 1.5 times the font size;
* Spacing following paragraphs to at least 2 times the font size;
* Letter spacing (tracking) to at least 0.12 times the font size;
* Word spacing to at least 0.16 times the font size.

#### Caption and transcript videos

Required tool(s): Manual verification   
Ensure the video as installed has accessible transcript and captioning.

* Verify any audio-only information can be understood via visual information.

#### Avoid timed events & blinking elements

Required tool(s): Manual verification, Chrome Accessibility Developer Tools

* Verify that all timed events can be controlled if they are not for decorative purposes only.
* Time the animations. Do not have anything blinking more than 3 times per second. Quick flashing elements could cause seizures in some people and discomfort in many others

#### Ensure the Acronyms, Abbreviations & Numbers read appropriately

Required tool(s): JAWS, JAWS inspect

Screen readers will attempt to read abbreviations as words and will read numbers in total rather than as individual digits.

* Ensure that abbreviations and acronyms such “UM” aren’t pronounced as a word but rather each individual letter announced or the meaning “Uninsured motorist” is provided
* Ensure that larger numbers such as policy number are as individual digits.

#### Support the use of the ‘autocomplete’ for primary form entry fields

Required tool(s): Manual Verification

Select HTML5 form entry fields should be tagged using autocomplete to assist users in form entry tasks.

* This can be tested by allowing your browser to save selected fields and then verify they autocomplete on your form

Here is a list of autocomplete fields that are supported

|  |  |  |
| --- | --- | --- |
| Content type | name attribute | autocomplete attribute |
| Name | name fname mname lname | * name (full name) * given-name (first name) * additional-name (middle name) * family-name (last name) |
| Email | Email | * email |
| Address | address city region province state zip zip2 postal country | * For one address input:   + street-address * For two address inputs:   + address-line1   + address-line2 * address-level1 (state or province) * address-level2 (city) * postal-code (zip code) * country |
| Phone | phone mobile country-code area-code exchange suffix ext | * tel |
| Usernames | Username | * username |

### 

## Sample test protocol

1. Open the page you are testing in Chrome and check the results in [**WAVE**](http://wave.webaim.org/extension/) . Note "Errors" and "Alerts"; these are the most severe issues that must be addressed.

2. WAVE ignores background colors and images. To check for color contrast, input the hex code values for the text and background colors into the [**WebAIM Color Contrast Checker**](http://webaim.org/resources/contrastchecker/).

2. Tab through the page you are testing with the keyboard to ensure that:

a. The tab order is correct - top to bottom, left to right and in reverse using shift+tab.

b. Focus is always detectable.

c. Everything can be interacted with using only the keyboard ([**Keyboard Accessibility**](https://webaim.org/techniques/keyboard/) [WebAIM].

d. Links don't open automatically, forms don't submit automatically, there are no automatic pop-ups, focus never jumps to another element automatically on focus, and the page doesn't otherwise change on focus alone.

3. View the page you are testing in Internet Explorer. Turn on the **JAWS** screen reader and tab through the page.

a. Double check everything from step 2.

b. Ensure that the proper [**ARIA roles and states**](https://www.progressive.com/dev/training/accessibility/standards/aria/) [PGR internal] are being read.

c. Use the down arrow key to read through the page and ensure nothing is read which isn't visible and nothing is skipped.

d. Press Insert+F6 to see the structure of headings and ensure they are correct.

e. Press Insert+F7 to see list of links and ensure the meaning makes sense in context.

f. Press 'L' to see order of lists.

g. Press 'F' to see order of form fields.

h. Press 'T' to see order of tables.

4. Open **ZoomText** and ensure the page can be read/interacted with.

5. Inspect all image tags for an alt attribute — purely decorative images need an empty alt attribute.

6. Ensure that information, selection, and error information is never indicated with color alone.

a. Turn off CSS using the [**Web Developer Chrome Extension**](https://chrome.google.com/webstore/detail/web-developer/bfbameneiokkgbdmiekhjnmfkcnldhhm) to make sure page still makes sense and no content is lost.

b. View the page in high contrast mode to ensure everything is still informative and visible.

7. Run an **Accessibility Audit** report in Chrome:

a. Press F12 in Chrome to open the Developer tools.

b. Go to the "Audits" section.

c. Make sure "Accessibility" is selected and click "Run".

d. Check the results for and take note of any errors that might have been missed during manual testing — in particular, ARIA errors.

## Mobile Device Configuration and Gestures

#### IOS devices

1. Go to Settings > General > Accessibility and turn "VoiceOver" on.

2. Access the page you are testing in Safari and ensure you can navigate and interact with all elements using VoiceOver gestures:

a. Tap once - Select item

b. Double tap - Activates the selected item

c. Flick left - Move to previous item (i.e., reverse tabbing)

d. Flick right - Move to next item (i.e., tabbing)

e. Three finger flick down - Scroll up one page

f. Three finger flick up - Scroll down one page

g. Two finger flick down - Read page starting at selected item

h. Two finger flick up - Read page starting at the top

i. There are other gestures, but these are the ones you will use most frequently.

#### Android devices

1. Go to [...] and turn "TalkBack" on.

2. Access the page you are testing and ensure you can navigate and interact with all elements using TalkBack gestures:

a. Drag one finger - Explore your screen and hear audible feedback for what is being touched.

b. Double-tap anywhere on the screen - Opens or activates the item that you last touched.

c. Swipe up or down using two fingers - Scroll within lists.

d. Swipe left or right using two fingers - Change pages and screens.

e. Swipe right (or down) using one finger - Move to the next item.

f. Swipe left (or up) using one finger - Move to the previous item.

g. Swipe down then up using a single motion - Transition to the next reading level when reading blocks of text, then swipe right to read forward or left to go back.

h. Swipe up then down using a single motion - Transition to the previous reading level when reading blocks of text, then swipe right to read forward or left to go back.

i. Swipe right then left using a single motion - Move to the next page.

j. Swipe left then right using a single motion - Move to the previous page.

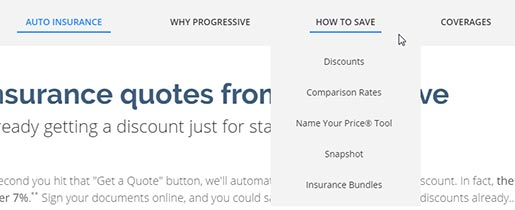
## ARIA Roles & States

Accessible Rich Internet Applications (ARIA) defines ways to make web content and applications more accessible to people with disabilities. ARIA is a set of HTML attributes used by screen readers and other accessibility software to help explain to the user (1) where they are on the page and (2) what is happening on the page.

See **the** [**Mozilla Developer Network**](https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA) for more in-depth information.

Developers apply ARIA attributes to HTML elements when the HTML tags alone are not sufficient in explaining to the user where they are or what's happening with the element. ARIA should only be used as a supplement to semantic HTML tags - it should not be used as a sole solution.

ARIA is used to define an element's role and its current state. For example, the page may have a navigation menu, with submenus



Using ARIA attributes on the menu's HTML, the developer can define this as a menu (role=menu) and the user's screen reader will notify the user that it's a menu, whether or not it has submenu items (aria-haspopup=true), whether or not the submenu items are opened (aria-expanded=true/false) and how many items there are (role=menuitem).

**Demo**: [Screen reader with ARIA](https://youtu.be/mNCMrxfO9Mc) [YouTube]

Although HTML 5 offers semantic tags for things like the footer and navigation, it is recommended that ARIA attributes defining the roles of these tags are still applied, as screen reader software still might not interpret HTML 5 tags correctly; the software just hasn't caught up, yet.

### ARIA Roles

An ARIA role attribute defines what this element is on the page. It is defined in code with "role=*rolename*" as an attribute on the HTML element. Common roles are:

* banner (header)
* navigation
* menubar
* menuitem
* menu
* menuitem
* contentinfo (used for the footer)
* button (when the HTML button tag is not used)

This [**ARIA roles cheat sheet**](http://karlgroves-sandbox.com/CheatSheets/ARIA-Cheatsheet.html) provides a good list of ARIA roles and where they're used.

### ARIA States

ARIA state attributes define when an element has changed on the page. ARIA state attributes are usually set to true/false and change dynamically via JavaScript at the same time the element changes. These dynamic changes are coded by the developer.

For example, if we have an accordion section on a page that is collapsed by default, the aria-expanded attribute on the accordion element is set to false, because it is not expanded. When the user tabs to the element and hits their enter key to open it, the accordion section opens and the aria-expanded attribute is dynamically changed to true. This triggers the screen reader to say to the user "expanded", so they know the section is now opened.

Common ARIA states are:

* aria-checked=true/false (checkboxes, radio buttons)
* aria-expanded=true/false (collapsible sections)
* aria-pressed=true/false (buttons that aren't semantic HTML buttons)

### ARIA Properties

ARIA properties are added to HTML elements to help define how they should be labelled, or to better describe the element. They are similar to ARIA states, except their values don't normally dynamically change.

Common ARIA properties are:

* aria-labelledby (defines a name for the element based on text from another element)
* aria-label (hard-coded label for the element)
* aria-controls (an element that controls the display of other elements on the page for example)

## Resources

### Reference, checklists, training

* [**W3C Accessibility Quick Reference**](https://www.w3.org/WAI/WCAG20/quickref/?currentsidebar=%23col_customize&levels=aaa) [WWW Consortium]
* [**Accessibility Training and Resources**](https://webaim.org/techniques/keyboard/) [WebAIM]
* [**Accessible fonts**](http://webaim.org/techniques/fonts/) [WebAIM]
* [**WCAG Checklist**](https://www.wuhcag.com/wcag-checklist/) [WuhCAG]
* [**ARIA roles and attributes cheat sheet**](http://karlgroves-sandbox.com/CheatSheets/ARIA-Cheatsheet.html) [Karl Groves's Sandbox]
* [**Accessibility testing document**](http://pgr/sites/webx/Shared%20Documents/Documentation/Accessibility/accessibility-testing-doc.xlsx) [PGR Internal]
* [**Branding Color Chart**](https://www.progressive.com/dev/colors/#contrast) [PGR internal]
* [**Keystroke cheat sheet**](https://www.progressive.com/content/downloads/domainprogressive/dev/training/accessibility/keyboard-interactions.pdf)[PDF] [PGR internal]
* [**accessibility.js**](https://www.progressive.com/content/js/domainprogressive/accessibility.js) [PGR internal]

### Articles

* [**Simply Accessible Accessibility articles**](https://simplyaccessible.com/articles/)
* [**Ssbbartgroup: Proper use of ARIA**](http://www.ssbbartgroup.com/blog/how-not-to-misuse-aria-states-properties-and-roles/)
* [**Medium.com: 7 Things Every Designer Needs to Know About Accessibility**](https://medium.com/salesforce-ux/7-things-every-designer-needs-to-know-about-accessibility-64f105f0881b)
* [**Medium.com: Projectors Don't Lie**](https://medium.com/salesforce-ux/projectors-dont-lie-b85ef628b04)

### Tools

* [**Web Accessibility Evaluation Tool (WAVE) Chrome Extension**](http://wave.webaim.org/extension/) [WebAIM]
* [**Color Contrast Checker**](http://webaim.org/resources/contrastchecker/) [WebAIM]
* [**Identifying WCAG compliant color palettes**](http://colorsafe.co/) [Colorsafe]
* [**Accessibility Tester**](https://tenon.io/) [Tenon]
* [**Keyboard Accessibility**](https://webaim.org/techniques/keyboard/) [WebAIM]
* [**ChromeVox Chrome Extension**](https://chrome.google.com/webstore/detail/chromevox/kgejglhpjiefppelpmljglcjbhoiplfn?hl=en) [Google]
* [**Web Developer Chrome Extension**](https://chrome.google.com/webstore/detail/web-developer/bfbameneiokkgbdmiekhjnmfkcnldhhm) [Google]
* [**Chrome Accessibility Developer Tools**](https://chrome.google.com/webstore/detail/accessibility-developer-t/fpkknkljclfencbdbgkenhalefipecmb?hl=en) [Google.com]
* **JAWS Screen Reader** (SRM request)
* **JAWS Inspect** (SRM request)
* **ZoomText** (SRM Request)