Accessibility Training Guide

This training guide is intended for developers, Creative Technologists, designers, Test Automation Developers, Quality Engineers and anyone else who is responsible for ensuring websites are accessible.

Contents

[Introduction 4](#_Toc22036968)

[Color Contrast 5](#_Toc22036969)

[Images 6](#_Toc22036970)

[Decorative Images 6](#_Toc22036971)

[Images Wrapped Within Links 7](#_Toc22036972)

[Scalable Vector Graphics (SVGs) 8](#_Toc22036973)

[Decorative SVGs 8](#_Toc22036974)

[Background Images 8](#_Toc22036975)

[Images of Text 9](#_Toc22036976)

[Links/Buttons 10](#_Toc22036977)

[Links with No Discernible Text 10](#_Toc22036978)

[Buttons with No Discernible Text 11](#_Toc22036979)

[Forms 12](#_Toc22036980)

[Form Inputs with No Visible Label 12](#_Toc22036981)

[Radio Buttons and Checkboxes 13](#_Toc22036982)

[Required Fields for Input and Select 13](#_Toc22036983)

[Required Fields for Radio Button and Checkbox 14](#_Toc22036984)

[Inputs with Supplemental Information 14](#_Toc22036985)

[Keyboard 16](#_Toc22036986)

[Visible Focus 16](#_Toc22036987)

[Tabindex 17](#_Toc22036988)

[Headings 18](#_Toc22036989)

[List Items 19](#_Toc22036990)

[Tables 20](#_Toc22036991)

[Simple Table 20](#_Toc22036992)

[Complex Table 20](#_Toc22036993)

[Language 22](#_Toc22036994)

[Language of the page 22](#_Toc22036995)

[Language of the part 22](#_Toc22036996)

[Title 23](#_Toc22036997)

[Bypass Blocks 24](#_Toc22036998)

[Skip Links 24](#_Toc22036999)

[Landmark regions 25](#_Toc22037000)

[ARIA 27](#_Toc22037001)

[ARIA Roles 28](#_Toc22037002)

[Landmark roles 28](#_Toc22037003)

[Alert 28](#_Toc22037004)

[Dialog 29](#_Toc22037005)

[Tab 29](#_Toc22037006)

[ARIA States 31](#_Toc22037007)

[ARIA Expanded 31](#_Toc22037008)

[ARIA Hidden 31](#_Toc22037009)

[ARIA Selected 32](#_Toc22037010)

[ARIA Properties 35](#_Toc22037011)

[ARIA Required 35](#_Toc22037012)

[ARIA Label 36](#_Toc22037013)

[ARIA Describedby 37](#_Toc22037014)

[ARIA Labelledby 38](#_Toc22037015)

[ARIA Live 39](#_Toc22037016)

[Comply Troubleshooting 41](#_Toc22037017)

[Ensures ARIA attributes are allowed for an element’s role 41](#_Toc22037018)

[Ensures elements with an ARIA role that require child roles contain them 41](#_Toc22037019)

[Ensures elements with an ARIA role that require parent roles are contained by them 41](#_Toc22037020)

[Ensures all elements with a role attribute use a valid value 42](#_Toc22037021)

[Ensures all ARIA attributes have valid values 42](#_Toc22037022)

[Ensures the contrast between foreground and background colors meet WCAG 2 AA contrast ratio thresholds 43](#_Toc22037023)

[Ensures every form element has a label 43](#_Toc22037024)

[Ensures each page has at least one mechanism for a user to bypass navigation and jump straight to the content 44](#_Toc22037025)

[Ensures every HTML document has a lang attribute 44](#_Toc22037026)

[Ensures buttons have discernible text 44](#_Toc22037027)

[Ensures links have discernible text 45](#_Toc22037028)

[Ensures every ID attribute is unique 46](#_Toc22037029)

[Ensures every ID attribute value of active elements is unique 46](#_Toc22037030)

[Ensures every ID attribute value used in ARIA and in labels is unique 47](#_Toc22037031)

[Ensures <meta name="viewport"> does not disable text scaling and zooming 47](#_Toc22037032)

[Ensures that lists are structured correctly 47](#_Toc22037033)

[Ensures <li> elements are used semantically 48](#_Toc22037034)

[Ensures each HTML document contains a non-empty <title> element 49](#_Toc22037035)

[Ensures <iframe> and <frame> elements contain a non-empty title attribute 49](#_Toc22037036)

[Ensures <img> elements have alternate text or a role of none or presentation 49](#_Toc22037037)

[Ensures <input type="image"> elements have alternate text 49](#_Toc22037038)

[Resources 50](#_Toc22037039)

# Introduction

Nationwide strives to meet level A and AA of the [Web Content Accessibility Guidelines](https://www.w3.org/TR/WCAG21/) (WCAG) 2.1, the international standard for creating accessible websites. The code examples provided in this training guide adhere to WCAG 2.1.

The success of creating an accessible, inclusive environment at Nationwide depends upon everyone understanding and embracing its importance.

Accessibility not only impacts our customers, but our associates as well. That is why it is important to take accessibility into account, even when creating emails and electronic documents, including Word, Powerpoint, and PDFs.

Web developers play an important role ensuring that digital accessibility is addressed during the early stages of project development rather than adding on accessibility during the final iterations. Addressing accessibility early in the development process will reduce the risk of costly and timely accessibility remediation efforts late in the project lifecycle.

Web developers are expected to become familiar with the Web Content Accessibility Guidelines (WCAG). It is not necessary to memorize WCAG, but it should be used as a reference throughout the project lifecycle.

The most effective way to reduce the risk of accessibility issues is to create semantic HTML.

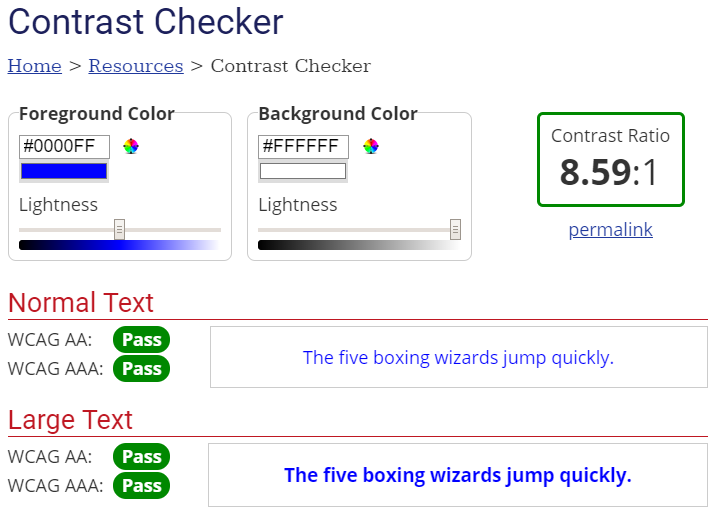
# Color Contrast

The contrast ratio between foreground and background colors needs to be high enough so that people with low vision can read content.

According to WCAG, the minimum contrast ratio between the foreground and background should be 4.5:1 for normal text and 3:1 for large text. Large text is considered 14 point and bold or 18 point.

The ratio between the foreground and background color can be determined using any of Nationwide’s accessibility automation tools. The automation tools include Comply, axe-core and the axe plug-in (available for Chrome and Firefox).

In addition to these tools, the [WebAIM Color Contrast Checker](https://webaim.org/resources/contrastchecker/) can analyze the ratio between color values. Sliders are available to adjust the foreground and/or background colors until the minimum standards are met.



# Images

All images must provide alternative text. Alternative text is important because it provides an alternative to blind and low vision users to interpret an image.

Imagine trying to explain this picture to someone over the phone. How would you do it?

**A herd of bison grazing in a lush green field in Yellowstone National Park

**

You might say “a herd of bison grazing in a lush green field at Yellowstone National Park”. Alternative text works the same way.

<img src="…" alt="a herd of bison grazing in a lush green field at Yellowstone National Park" />

Screen readers will read the alt text so that blind and low vision users understand the purpose of the image.

## Decorative Images

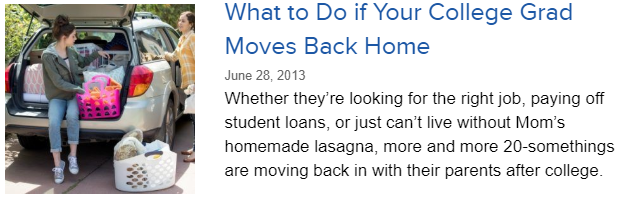
All images need alternative text, but not all images provide meaning. Some images are used for purely decorative purposes, to make the page more visually pleasing. If an image is decorative, adding empty alt text will cause screen readers to ignore it. It’s important to note that simply omitting the alt text is not the same as providing empty alt text. Many screen readers will read the file name of the image if alt text is not provided. The only way to ensure the decorative image will not be read by screen readers is to provide empty alt text.



<img src="…" alt="" />

## Images Wrapped Within Links

If a link surrounds an image, the link’s destination needs to be described rather than the image. In the following example, the image is clickable, and the link will take the user to a web page called “What to Do if Your College Grad Moves Back Home”.



If the alternative text is “girl sitting in back of station wagon unloading luggage”, that would not accurately describe where the link is going to take the user. The more accurate alt text would explain the link’s destination.

<a href="/college-grads-movin-back-home/">

<img src="…" alt="what to do if your college grad moves back home" />

</a>

The screen reader will read “link, graphic, what to do if your college grad moves back home”. This makes much more sense than “link, graphic, girl sitting in back of station wagon unloading luggage”.

## Scalable Vector Graphics (SVGs)

Scalable Vector Graphics (SVGs) work differently than images. By default, screen readers do not interpret SVGs as images. If SVGs are to be read by screen readers, three additional attributes are needed.

An image role needs added to the SVG so that screen readers recognize the SVG as an image.

An aria-labelledby need added to the SVG. The aria-labelledby value needs to match the ID on the title.

A title needs added. The title should be just below the SVG and the ID on the title needs to match the value of the aria-labelledby.

The Nationwide logo is an SVG. The image is read by the screen reader by including the three attributes.



<svg aria-labelledby="NandEagle" role="img">

<title id="NandEagle">Nationwide logo</title>

The screen reader will read “graphic, Nationwide logo”.

## Decorative SVGs

If the SVG is strictly for decorative purposes, omitting the role, aria-labelledby and title attributes will cause the screen reader to ignore it, similar to adding alt="" on an image element.

## Background Images

By default, background images are ignored by screen readers. Background images are controlled through CSS and there is no image attribute.

## Images of Text

The best practice is to use actual text whenever possible. If an image of text is used, the alt text on the image should match the text within the image.



<img src="…" alt="Check the background of this firm with Broker Check" />

# Links/Buttons

Every link must explain its purpose so that screen reader users know where they will be taken when they click the link.

Here’s an example of a typical link.

example of a link

<a href=“…">

Helping Americans address the risk of outliving their retirement income

</a>

The screen reader will read “link, helping Americans address the risk of outliving their retirement income”.

## Links with No Discernible Text

If the link text is empty, screen readers will announce a link, but won’t be able to discern where the link will take the user. This typically happens when background images or icon fonts are used.

link with no discernible text

Since the home icon is a background image and background images don’t have alternative text, the screen reader will read “link” but won’t know where the link will take the user.

<a href="/">

<span class="nw-breadcrumbs--home"></span>

</a>

Adding an aria-label with the appropriate link text will resolve the issue.

<a href="/" aria-label="Nationwide home">

<span class="nw-breadcrumbs--home"></span>

</a>

The screen reader will read “link, Nationwide home”.

## Buttons with No Discernible Text

Like links, if the button text is empty, screen readers will announce a button, but won’t be able to discern the purpose of the button. This typically happens when background images or icon fonts are used.

magnifying glass button with no discernible text

Since the magnifying glass button is an icon font, the screen reader will read “button” but won’t know the button’s purpose.

<button class="btn" type="button" id="ebsearch">

<i class="glyphicon glyphicon-search"></i>

</button>

Adding an aria-label to the button will resolve the issue.

<button class="btn" type="button" id="ebsearch" **aria-label="search"**>

<i class="glyphicon glyphicon-search"></i>

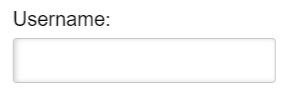
</button>

The screen reader will read “button, search”.

# Forms

Every input, select box, radio button and checkbox must be associated with a label. Without a label, screen reader users won’t know how to interact with the form element.

Here’s an example of a typical input.



There is a visible label, so it’s easy for sighted users to make an association between the input and the label. For screen reader users, however, the association must be created, either implicitly or explicitly.

With an implicit association, the label is wrapped around the input.

<label>Username:

<input type="text" class="text-field-lrg" name="username" value="">

</label>

With an explicit association, the label has a for attribute that matches the ID on the input.

<label for="username">Username:</label>

<input type="text" class="text-field-lrg" id="username" name="username" value="">

## Form Inputs with No Visible Label

Sometimes, there is an input, but no visible label exists.

search input with no visible label

Sighted users can determine that this is a search field, based on the placeholder text and the magnifying glass image. For screen reader users, however, the association must be created.

Since there is no visible label, an aria-label can be used as a substitution to the label element.

<input type="text" placeholder="Search For Your Company" name="company”

**aria-label="search for your company"**>

The screen reader will read “search for your company, edit”.

## Radio Buttons and Checkboxes

Radio buttons and checkboxes need to be associated with labels. In addition, a legend is needed that explains the grouping.

Here’s an example of a typical radio button grouping.



Sighted users can make the association between gender and the labels for the radio buttons. For screen reader users, however, the association must be created. The entire grouping should be wrapped in a fieldset. A legend should immediately follow the fieldset.

<fieldset>

<legend>Gender:</legend>

<input type="radio" value="M" name="gender" id="gender-m" class="text-field-dd">

<label for="gender-m" class="radio-label">Male</label>

<input type="radio" value="F" name="gender" id="gender-f" class="text-field-dd">

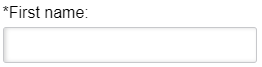
<label for="gender-f" class="radio-label">Female</label>

</fieldset>

When the first radio button receives focus, the screen reader will read “Gender, grouping, male radio button checked, 1 of 2”.

## Required Fields for Input and Select

Required fields can be identified by either using the HTML required attribute or using aria-required=“true”.



<label for="firstName">\*First name:</label>

<input type="text" aria-required="true" name="firstName" id="firstName" value="">

The screen reader will read “first name, edit, required”.

## Required Fields for Radio Button and Checkbox

Required fields on radio buttons and checkboxes are identified on the fieldset element rather than the input.



<fieldset aria-required="true">

<legend>Gender:</legend>

<input type="radio" value="M" name="gender" id="gender-m" class="text-field-dd">

<label for="gender-m" class="radio-label">Male</label>

<input type="radio" value="F" name="gender" id="gender-f" class="text-field-dd">

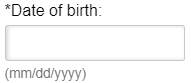
<label for="gender-f" class="radio-label">Female</label>

</fieldset>

The screen reader will read “gender, grouping, required, male radio button not checked, 1 of 2”.

## Inputs with Supplemental Information

Supplemental information, such as hint text or error messages, also need to be associated with the input field.



Sighted users can make the association between the label and the hint text. For screen reader users, however, the association must be created.

The association is created by adding an aria-describedby property on the input and associating it with the ID on the hint text.

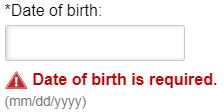
<label for="dateOfBirth">\*Date of birth:</label>

<input type="text" aria-required="true" aria-describedby="dateOfBirth-hint" name="dateOfBirth" id="dateOfBirth" value="">

<span class="hint" id="dateOfBirth-hint">(mm/dd/yyyy)</span>

The screen reader will read “date of birth, edit, required, mm slash dd slash yyyy, blank”.

Error message need associated with inputs as well, using the aria-describedby property. More than one association can be made. If there is hint text and an error message, include both within the aria-describedby property.



<label for="dateOfBirth">\*Date of birth:</label>

<input type="text" aria-required="true" aria-describedby="dateOfBirth-error dateOfBirth-hint" name="dateOfBirth" id="dateOfBirth" value="">

<span class="field-error" id="dateOfBirth-error">Date of birth is required.</span>

<span class="hint" id="dateOfBirth-hint">(mm/dd/yyyy)</span>

The screen reader will read “date of birth, edit, required, date of birth is required, mm slash dd slash yyyy, blank”.

# Keyboard

Believe it or not, not everyone uses a mouse to navigate through websites. Some people rely on the keyboard only, either due to preference or an inability to use a mouse.

All functionality must work using the keyboard only.

* If submenus can only be accessed via a mouseover, keyboard-only users will be excluded.
* If links don’t have an href, keyboard only users won’t be able to tab to the link.
* If buttons are built using a <div> or <span> rather than a <button>, keyboard-only users won’t be able to tab to the element.
* If date input fields require the user to choose a date via a calendar widget rather than typing the date, keyboard-only users will be excluded.

When testing functionality, always make sure to see if it works using the keyboard only. It only takes a minute to check, but it has a profound impact on keyboard-only users.

## Visible Focus

Keyboard-only users use the Tab key to navigate to interactive elements, and they need to know when focus has moved to an element. The focus must always be visible.

Never use outline:0 or outline:none in CSS. This causes the visible focus to be removed for keyboard-only users.

~~outline:0~~

~~outline:none~~

Link without focus:



Link with focus:



If a :hover effect is used in CSS, provide an equivalent :focus effect.

Button without focus:

button without focus

Button with focus:

button with focus

CSS:

.preferred:hover, preferred:focus {}

## Tabindex

By default, interactive elements, such as form inputs, links, and buttons, automatically are included in the tabindex order. The order is determined by the DOM – the first interactive element within the DOM receives focus first, followed by the second, and so on. There is no need to add a tabindex to an interactive element.

If the tab order does not match the DOM, it may be because a tabindex greater than 0 was used, which will interrupt the natural order. A tabindex greater than 0 should never be used.

If an interactive element is not receiving focus, this may indicate a coding error.

* If a link doesn’t receive focus, it may be missing an href or it may have tabindex=“-1”.
* If a button doesn’t receive focus, it may have been coded with a <span> or <div> rather than a <button> or <input type=“button”>.
* If a radio button or checkbox is not receiving focus, it may have been created with a custom input that is not accessible.

# Headings

Headings help to visually segregate the webpage into sections. Headings are important to screen reader users as well because shortcut keys can be used to navigate between headings.

Headings should follow a logical order. A heading level one should be first on the page, followed by heading level 2, heading level 3, etc. Heading levels should not be skipped. It is a best practice to have only one heading level 1 per page.

Don’t use CSS to create fake headings. Styling a heading to look like a heading may be fine visually, but screen readers will not recognize it as a heading. Only headings properly marked up with <h1>, <h2>, <h3>, <h4>, <h5> and <h6> will be recognized as headings.

# List Items

List items are used to group similar items. List items can be either unordered, meaning the order of the items doesn’t matter, or ordered, meaning the order does matter. An example of an unordered list is a grocery list. An example of an ordered list would be a recipe, where the order of the steps matter.

Don’t use CSS to create fake lists. Styling a list to look like a list may be fine visually, but screen readers will not recognize it as a list.

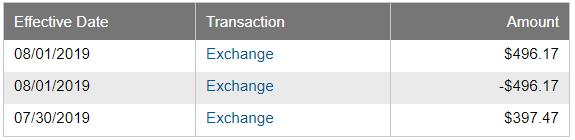
If <li> is used, it must be wrapped with either <ul> or <ol>

# Tables

There are two common types of tables, simple tables and complex tables.

## Simple Table

A simple table has a header row for each column. Here’s an example of a simple table.



On simple tables, the first row should identify the headers. Use scope=“col” to indicate which column the header is associated with. Add a <caption> below the table. If the caption is not be shown visually, use CSS to hide it off-screen.

<table id="recent-activity-table" class="data-table mb30">

<caption>recent activity</caption>

<tbody>

<tr>

<th scope="col">Effective Date</th>

<th scope="col">Transaction</th>

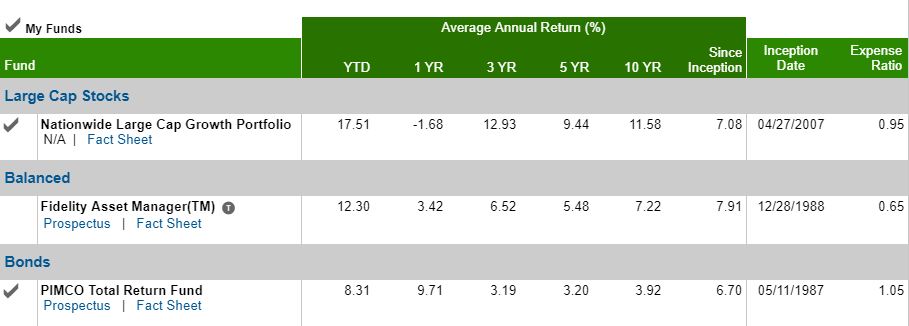
<th scope="col">Amount</th>

</tr>

<tr>

## Complex Table

A complex table typically has multiple header rows and may scan multiple data cells. Here’s an example of a complex table.



Complex tables require a unique ID on each of the headers.

Each data cell needs a headers attribute to associated which header should be associated with the data cell.

For instance, if a screen reader user was in the 3 YR column for the Fidelity Asset Manager fund, there are four headers that would need to be announced.

* Average Annual Return (%). (row 1, column 2)
* 3 YR (row 2, column 4)
* Balanced (row 5, column 1)
* Fidelity Asset Manager (row 6, column 1)

<td class="numeric nowrap" headers="r1c2 r2c4 r5c1 r6c1">6.52</td>

# Language

Since screen readers can read in multiple languages, the language of every webpage needs specified so that the screen reader knows how to properly enunciate the text.

## Language of the page

The lang attribute belongs on the <html> attribute.

<html lang=“en”>

The language attribute of “en” will read the contents of the page in English.

<html lang=“es”>

The language attribute of “es” will read the contents of the page in Spanish.

## Language of the part

If a portion of the page should be read in a different language, add the lang attribute on the element.

<a href="…" lang="es">Español</a>

# Title

Every webpage should have a unique title. The title is the first content that will be read by screen readers when the page loads.

The <title> attribute is added into the <head> of the document.

It is a best practice to put unique information first.

<title>How do Annuities Work? – Nationwide</title>

<title>What to Know Before Buying a House - Nationwide</title>

# Bypass Blocks

Bypass blocks allow screen reader users and keyboard-only users to skip to the important content quickly. There are two types of bypass blocks: skip links and landmark regions.

## Skip Links

Skip links are in-page anchor links that are used by keyboard-only users to quickly navigate to the main content on the page. The skip link should be the first link on the page, immediately following the <body>.

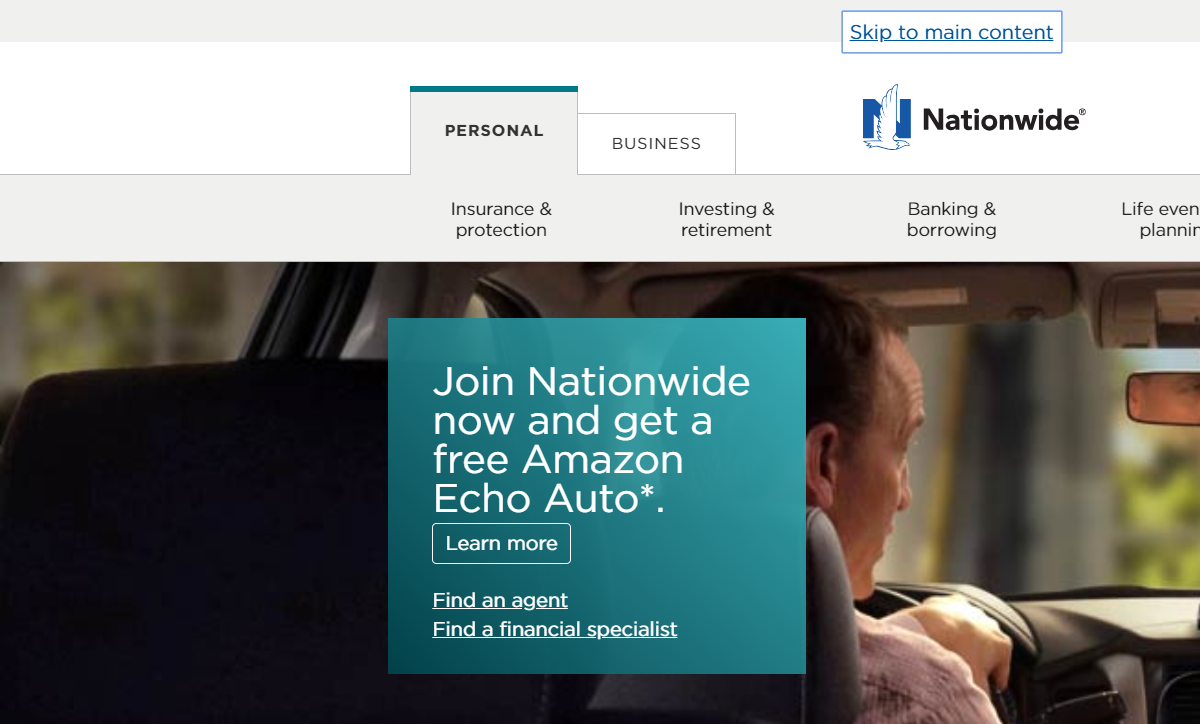
<a href="#main-content">Skip to main content</a>

When the skip link is clicked, focus will move to the location of the link specified in the anchor link.

<main id="main-content">

The skip link can be styled in a variety of ways and can be hidden off-screen until it receives focus.

Here is an example of a skip link on the Nationwide home page.



## Landmark regions

Landmark regions go a step further than skip links. Instead of one link to the main content on the page, landmark regions can be used to navigate to the header, navigation, main content, complementary content, forms, search and the footer.

Another difference is that landmark regions are used by screen reader users, but not keyboard-only users.

Landmark regions can be added to the webpage using HTML5 elements, ARIA landmark roles, or both. The recommendation is to use both to ensure wider support among various browser/screen reader combinations.

|  |  |
| --- | --- |
| HTML 5 Elements | ARIA Landmark Roles |
| <header> | role=“banner” |
| <nav> | role=“navigation” |
| <main> | role=“main” |
| <footer> | role=“contentinfo” |
| <aside> | role=“complementary” |
| <form> | role=“form” |
| N/A | role=“search” |

# ARIA

Websites have become more dynamic over time, and plain HTML can no longer be relied upon to provide full accessibility. ARIA (Accessible Rich Internet Applications) was created to improve accessibility of websites by providing extra information to assistive technologies, such as screen readers, via attributes which could be added to HTML.

Although ARIA has the potential to improve accessibility, it also has the potential to make the accessibility experience much worse. For instance, adding role=“navigation” on an unordered list will override the <ul> and cause screen readers to read it as <nav>. That’s why the first rule of ARIA is to only use it when semantic HTML will not suffice.

ARIA falls into three categories – roles, states and properties.

* ARIA roles define the component’s purpose in the interface.
* ARIA states define the current condition of the component.
* ARIA properties define characteristics of the component.

Although there is a long list of ARIA roles, states and properties, only the most commonly used ARIA elements will be covered. For more information on ARIA, click the ARIA roles, states and properties link in the resources section.

# ARIA Roles

ARIA roles are pre-defined elements that provide additional information to screen readers. All ARIA roles begin with role= followed by a pre-defined value.

## Landmark roles

Landmark roles identify large content areas and are used by screen readers for navigation. A best practice to have all the content with a web page within a landmark role. Landmark roles can be navigated to by screen reader users via shortcut keys.

Landmark regions can be added to the webpage using HTML5 elements, ARIA landmark roles, or both. The recommendation is to use both to ensure wider support among various browser/screen reader combinations.

|  |  |
| --- | --- |
| HTML 5 Elements | ARIA Landmark Roles |
| <header> | role=“banner” |
| <nav> | role=“navigation” |
| <main> | role=“main” |
| <footer> | role=“contentinfo” |
| <aside> | role=“complementary” |
| <form> | role=“form” |
| N/A | role=“search” |

## Alert

The alert role is used to provide time-sensitive information to screen reader users. For instance, if an error occurs while completing a form, the user needs to be informed right away so that they can correct the error.

The alert should not be used for information that changes often on a website, such as updating stock prices. This would become disruptive for screen reader users and create a bad user experience.

The alert role can be added to an existing element. When content within the element changes, screen reader users will be alerted to the change.

<p id="errors" role="alert"></p>

When an error is injected into the paragraph element, the screen reader will immediately read the information.

<p id="errors" role="alert">Please enter your first name.</p>

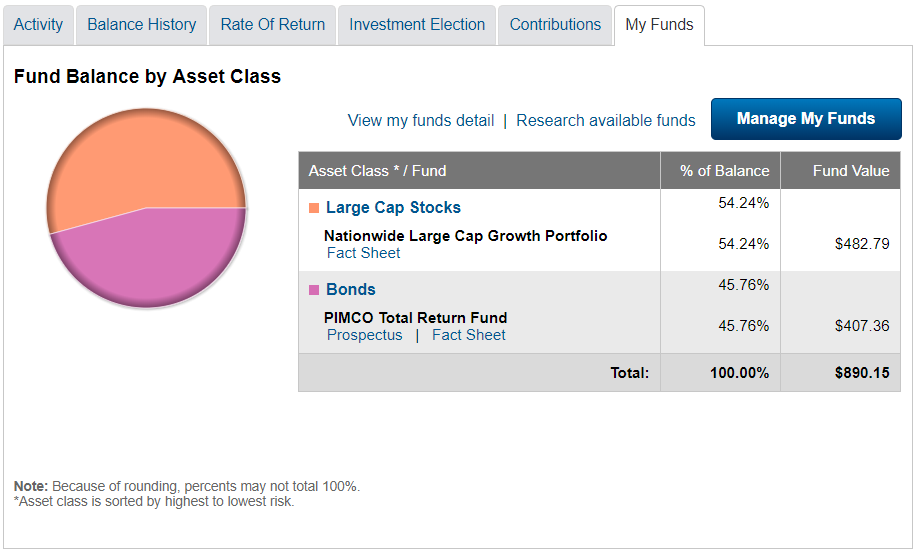
## Dialog

The dialog role is used to inform screen reader users that the focus has moved from the parent window into a dialog box.

In addition to adding role="dialog", there are a few additional steps that are necessary to ensure dialogs work correctly. The user should be trapped inside the dialog until an action is performed (submit, cancel or close, for instance). The user should not be able to tab back into the parent window. This will cause confusion and may make it difficult to navigate back into the dialog window. Also, once the dialog is closed, focus should move back to the element that launched the dialog. For instance, if a button was clicked to open the dialog, focus should move back to the button when the dialog is closed.

## Tab

Creating accessible tabs requires three roles – tab, tablist and tabpanel. The tab role is a role that needs specified on each tab. In addition to the tab role, the parent role of tablist is required. The tablist role is a list of all tab elements. The tabpanel role goes on each container that is associated with each tab.



Each of the tabs (Activity, Balance History, Rate or Return, Investment Election, Contributions, My Funds) need role="tab".

<li role="tab" aria-selected="false">

<a href="#activityOverview">Activity</a>

</li>

The outer element for each of the tabs needs role="tablist".

<ul class="tabs\_\_navigation group" role="tablist">

Each tabpanel needs role="tabpanel".

<div id="activityOverview" role="tabpanel" class="tabs\_\_content-container">

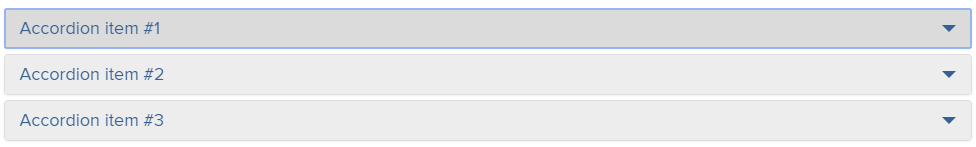
# ARIA States

ARIA states define the current condition of the component. Since the state is dynamic, JavaScript is required to toggle between the correct state, typically either true or false.

## ARIA Expanded

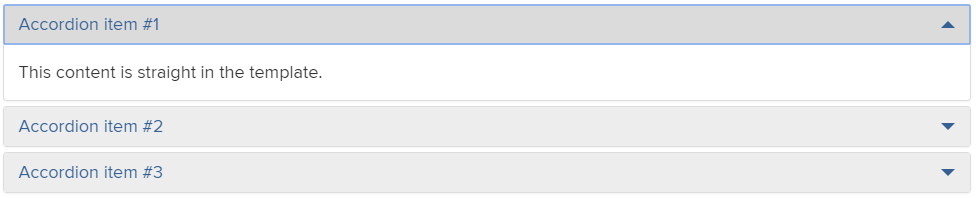
The aria-expanded state indicates whether an expandable/collapsible group of elements is currently expanded or collapsed. Accordions are an example of content that can be expanded or collapsed. Menus with submenus is another example. Any time a link or button can be clicked to expose more information to the user, aria-expanded is needed to inform screen reader users of the hidden content.

The aria-expanded state is added to the triggering element.



<a role="button" href="" aria-expanded="false">Accordion item #1</a>

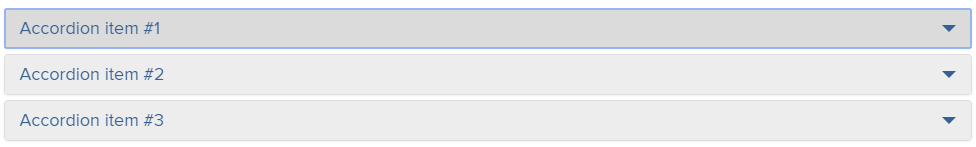
When the accordion is expanded, the aria-expanded state toggles to "true".



<a role="button" href="" aria-expanded="true">Accordion item #1</a>

## ARIA Hidden

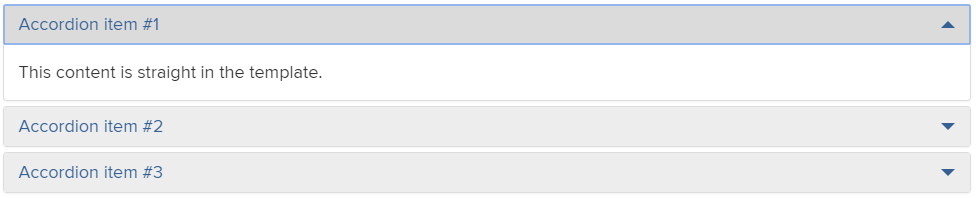
The aria-hidden state informs screen readers whether the information should be exposed and read. If a component has aria-hidden="true", the screen reader will not read the content. This is typically used in conjunction with aria-expanded. For instance, on accordions, when the accordion is in the collapsed state (aria-expanded="false"), the content within the accordions should be hidden from screen readers (aria-hidden="true").



<div aria-hidden="true">

This content is straight in the template.

</div>



When the accordion is expanded, the aria-hidden state toggles to "false".

<div aria-hidden="false">

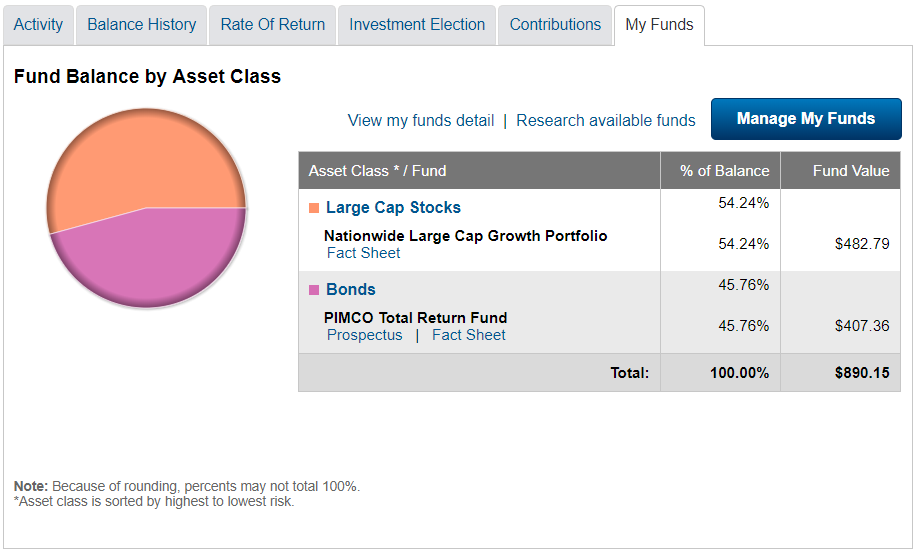
This content is straight in the template.

</div>

## ARIA Selected

The aria-selected state informs screen readers whether the focused widget is currently selected. For instance, when a tab is selected, the value is aria-selected="true" and the screen reader announces “selected”. If the tab is not selected, the value is aria-selected="false".

The Activity tab is currently not selected, so the state is set to aria-selected="false".

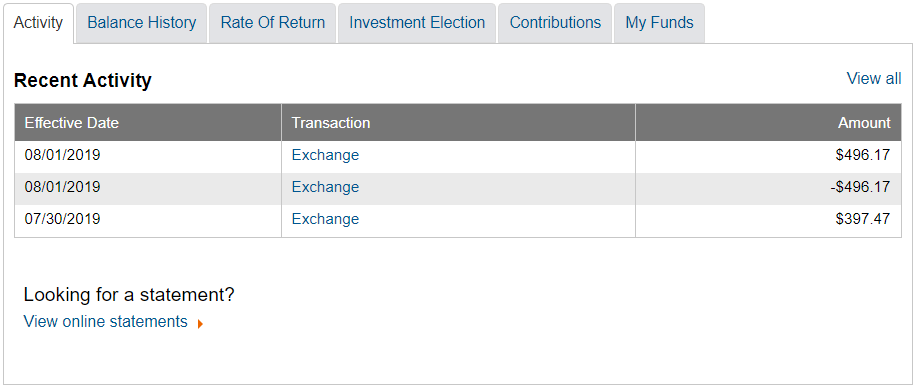


<li role="tab" aria-selected="false">

<a href="#activityOverview">Activity</a>

</li>

When the Activity tab is selected, the state is toggled to aria-selected="true".



<li role="tab" aria-selected="true">

<a href="#activityOverview">Activity</a>

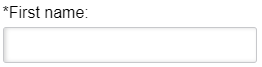
</li>

# ARIA Properties

ARIA properties define the characteristics of the component. Unlike ARIA states, the properties are static values that provide additional information to screen readers.

## ARIA Required

The aria-required property tells whether a form input is required or not. If the input is required, adding aria-required="true" will cause the screen reader to announce that the field is required. If the field is not required, there is no need to add aria-required="false", because the screen reader will only announce fields that are required.



<label for="firstName">\*First name:</label>

<input type=“text" aria-required="true" name="firstName" id="firstName" value="">

The screen reader will read “first name, edit, required”.

Required fields on radio buttons and checkboxes are identified on the fieldset element rather than the input.



<fieldset aria-required="true">

<legend>Gender:</legend>

<input type="radio" value="M" name="gender" id="gender-m" class="text-field-dd">

<label for="gender-m" class="radio-label">Male</label>

<input type="radio" value="F" name="gender" id="gender-f" class="text-field-dd">

<label for="gender-f" class="radio-label">Female</label>

</fieldset>

The screen reader will read “gender, grouping, required, male radio button not checked, 1 of 2”.

## ARIA Label

The aria-label property defines a label for the element. The aria-label property is used when a visible label is not available on the screen.

search input with no visible label

Since there is no visible label on the screen for the input, aria-label can be used as a replacement.

<input type="text" placeholder="Search For Your Company" name="company”

aria-label="search for your company">

The screen reader will read “search for your company, edit”.

It’s important to remember that all input fields need a label. Placeholder text is not a valid substitution for a label.

The aria-label property can also be used when links do not have discernible text.

link with no discernible text

Since the home icon is a background image and background images don’t have alternative text, the screen reader will read “link” but won’t know where the link will take the user.

<a href="/">

<span class="nw-breadcrumbs--home"></span>

</a>

Adding an aria-label with the appropriate link text will resolve the issue.

<a href="/" aria-label="Nationwide home">

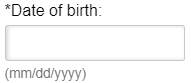
<span class="nw-breadcrumbs--home"></span>

</a>

The screen reader will read “link, Nationwide home”.

## ARIA Describedby

The aria-describedby property provides supplemental information to screen readers. The aria-describedby property is used on interactive elements and can find supplemental information via an ID association within the webpage. An example of supplemental information is hint text or error messages.



The association is created by adding an aria-describedby property on the input and associating it with the ID on the hint text.

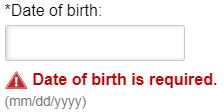
<label for="dateOfBirth">\*Date of birth:</label>

<input type="text" aria-required="true" aria-describedby="dateOfBirth-hint" name="dateOfBirth" id="dateOfBirth" value="">

<span class="hint" id="dateOfBirth-hint">(mm/dd/yyyy)</span>

The screen reader will read “date of birth, edit, required, mm slash dd slash yyyy, blank”.

Error message need associated with inputs as well, using the aria-describedby property. More than one association can be made. If there is hint text and an error message, include both within the aria-describedby property.



<label for="dateOfBirth">\*Date of birth:</label>

<input type="text" aria-required="true" aria-describedby="dateOfBirth-error dateOfBirth-hint" name="dateOfBirth" id="dateOfBirth" value="">

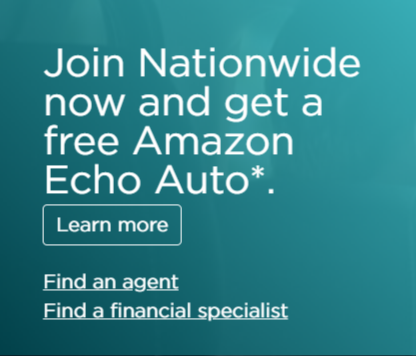
<span class="field-error" id="dateOfBirth-error">Date of birth is required.</span>

<span class="hint" id="dateOfBirth-hint">(mm/dd/yyyy)</span>

The screen reader will read “date of birth, edit, required, date of birth is required, mm slash dd slash yyyy, blank”.

The aria-describedby property can also be used to provide supplemental information on links and buttons.

The Learn more button has no meaning for screen reader users when read out of context.



The aria-describedby property can be associated with the header to provide better context for screen reader users.

<h1 class="nw-heading-mdlg" id="heading">Join Nationwide now and get a free Amazon Echo Auto\*.</h1>

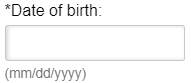
<a href="https://www.nationwide.com/echoauto" target="\_blank" aria-describedby="heading" role="button">Learn more</a>

When the Learn more button receives focus, the screen reader will read “Learn more, button, Join Nationwide now and get a free Amazon Echo Auto”.

## ARIA Labelledby

The aria-describedby and aria-labelledby properties work very similarly, but they do have a distinct difference: the aria-describedby property provides supplemental information, while the aria-labelledby property replaces information.

Here is an example of an input field with supplemental information.



<label for="dateOfBirth">\*Date of birth:</label>

<input type="text" aria-required="true" **aria-describedby="dateOfBirth-hint"** name="dateOfBirth" id="dateOfBirth" value="">

<span class="hint" id="dateOfBirth-hint">(mm/dd/yyyy)</span>

If the aria-describedby property is used on the input, the screen reader will read “date of birth, edit, required, mm slash dd slash yyyy, blank”.

If the aria-labelledby property is used on the input rather than the aria-describedby property, the aria-labelledby will override the existing label (Date of birth) and the label will become the hint text. The screen reader will read “mm slash dd slash yyyy, edit, required, blank”. The screen reader user will no longer know the purpose of the input field.

There are times when aria-labelledby is useful. Inputs with multiple labels are one example.



Since only one label can be associated with an input, aria-labelledby is used. Adding an aria-labelledby on the input and associating it with unique IDs causes the association.

<th id="tpayer">Taxpayer</th>

<th id="gross">W2 Gross</th>

<input size="20" aria-labelledby="tpayer gross" type="text"></td>

The screen reader will read “taxpayer, w2 gross, input, blank”.

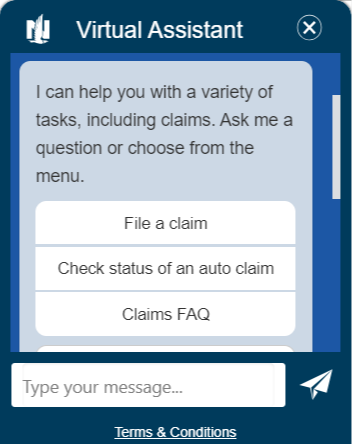
## ARIA Live

Like the alert role, the aria-live property informs screen readers when content has changed on the webpage.

The aria-live property should not be used for information that changes often on a website, such as updating stock prices. This would become disruptive for screen reader users and create a bad user experience.

The aria-live property can be added to an existing element. When content within the element changes, screen reader users will be alerted to the change.

The Virtual Assistant on the Nationwide Claims page uses aria-live to inform screen reader users when a new message appears.



<div class="wc-message-groups" role="log" tabindex="0" aria-live="assertive">

The aria-live property is on the outermost <div>, so that any changes occurring within the window is immediately announced by the screen reader.

The aria-live property can be either assertive or polite. If the aria-live property is set to assertive, the screen reader will immediately read the new content within the live region. If the aria-live property is set to polite, the screen reader will not read the new content within the live region until finishing the current sentence or paragraph.

# Comply Troubleshooting

Often, the biggest challenge to remediating accessibility issues is understanding what is wrong with the underlying code.

Comply, Nationwide’s enterprise tool, crawls through websites and provides a dashboard of issues. Some of the issues are easy to understand; others are not.

The following is a description of the issues, what it means, and how to remediate them.

## Ensures ARIA attributes are allowed for an element’s role

This means that an ARIA attribute was incorrectly used on an element that doesn’t support it. For instance, the aria-selected state informs screen reader users that a selectable attribute, such as a tab, is either selected or not selected. However, using the aria-selected state on a non-selectable attribute, such as a link, will cause this error. Links aren’t selected, they are clicked.

This is an example of invalid code that will cause this error.

<a href="/business/" aria-selected="false">See what we offer for businesses.</a>

The issue can be resolved by removing the aria-selected attribute.

## Ensures elements with an ARIA role that require child roles contain them

This means that an ARIA role was used without including the child role. For instance, using role=“feed” without using the child role=“article” will cause this error.

This issue can be fixed by either removing the parent role or adding the child role.

## Ensures elements with an ARIA role that require parent roles are contained by them

This means that an ARIA role was used without including the parent role. For instance, using role=“tab” without using the parent role=“tablist” will cause this error.

This is an example of invalid code that will cause this error.

<ul class="tabs\_\_navigation group">

<li role="tab" aria-selected="false">

<a href="#activityOverview">Activity</a>

</li>

The issue can be resolved by adding role="tablist" on the unordered list.

<ul class="tabs\_\_navigation group" role="tablist" >

<li role="tab" aria-selected="false">

<a href="#activityOverview">Activity</a>

</li>

## Ensures all elements with a role attribute use a valid value

This means that an invalid role attribute was used. For instance, creating a role is not valid.

This is an example of invalid code that will cause this error.

<img alt="Nationwide logo" role="Metlife logo" src="…">

The issue can be resolved by removing the invalid role.

This error will also occur if the role is misspelled or if it is capitalized.

This is an example of invalid code that will cause this error.

<div aria-label="Video track slider" tabindex="0" aria-valuemax="30.464" aria-valuemin="0" aria-valuenow="0" aria-orientation="horizontal" role="Slider">

Since role="Slider" is capitalized, the role is ignored by assistive technology because it is considered invalid.

The issue can be resolved by changing the role to role="slider">.

## Ensures all ARIA attributes have valid values

This means that the value is invalid. For instance, aria-required="maybe" is not a valid value. It must be either true or false.

This error will also occur if an attribute is used and the corresponding element is not available. For instance, using aria-describedby on an input field without a matching ID will cause the aria-describedby value to be invalid.

This is an example of invalid code that will cause this error.

<input type="text" value="" aria-describedby="firstNameError" aria-required="true">

If there is not an ID on the page that matches firstNameError, this will cause the aria-describedby value to be invalid.

This can be fixed by either adding an ID that matches the aria-describedby value or removing the aria-describedby value until an error message is displayed on the page that matches the aria-describedby value.

## Ensures the contrast between foreground and background colors meet WCAG 2 AA contrast ratio thresholds

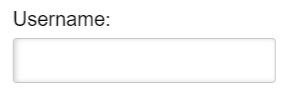
This means that the color contrast between the foreground and background colors is not at least 4.5:1 for normal text or 3:1 for large text.

This error can be fixed by either changing the foreground color, background color, or both, until the minimum contrast ratio is met.

## Ensures every form element has a label

This means that an input field, select box, radio button or checkbox is not associated with a label.

If a visible label exists, add a for attribute on the label that matches the ID on the input.



<label for="username">Username:</label>

<input type="text" class="text-field-lrg" id="username" name="username" value="">

If a visible label does not exist, add an aria-label on the input that explains the purpose of the input field.

search input with no visible label

<input type="text" placeholder="Search For Your Company" name="company”

aria-label="search for your company">

## Ensures each page has at least one mechanism for a user to bypass navigation and jump straight to the content

This means that no ARIA landmark roles or HTML5 equivalent elements exist on the webpage.

This can be fixed by adding the following in the appropriate locations within the webpage. Only one main landmark should be used and all content on the page should reside within the landmark roles. It is a best practice to use both the HTML 5 elements and the ARIA landmark roles to ensure the highest screen reader/browser support.

|  |  |
| --- | --- |
| HTML 5 Elements | ARIA Landmark Roles |
| <header> | role=“banner” |
| <nav> | role=“navigation” |
| <main> | role=“main” |
| <footer> | role=“contentinfo” |
| <aside> | role=“complementary” |
| <form> | role=“form” |
| N/A | role=“search” |

## Ensures every HTML document has a lang attribute

This means that the HTML document is missing the lang attribute, which is needed so that screen readers know how which language to read the content.

Every HTML document on the page needs a lang attribute. If an iframe is injected on the page, the iframe’s HTML document needs the lang attribute specified as well.

This can be fixed by adding the appropriate language to the HTML document.

<html lang="en">

## Ensures buttons have discernible text

This means that the button has no text to explain the purpose of the button.

magnifying glass button with no discernible text

Since the magnifying glass button is an icon font, the screen reader will read “button” but won’t know the button’s purpose.

<button class="btn" type="button" id="ebsearch">

<i class="glyphicon glyphicon-search"></i>

</button>

This can be fixed by adding an aria-label to the button.

<button class="btn" type="button" id="ebsearch" **aria-label="search"**>

<i class="glyphicon glyphicon-search"></i>

</button>

The screen reader will read “button, search”.

## Ensures links have discernible text

This means that the link has no text to explain the purpose of the link.

If the link text is empty, screen readers will announce a link, but won’t be able to discern where the link will take the user. This typically happens when background images or icon fonts are used.

link with no discernible text

Since the home icon is a background image and background images don’t have alternative text, the screen reader will read “link” but won’t know where the link will take the user.

<a href="/">

<span class="nw-breadcrumbs--home"></span>

</a>

This can be fixed by adding an aria-label with the appropriate link text.

<a href="/" aria-label="Nationwide home">

<span class="nw-breadcrumbs--home"></span>

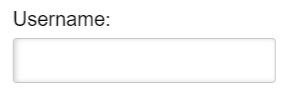
</a>

The screen reader will read “link, Nationwide home”.

## Ensures every ID attribute is unique

This means that an ID was used more than once on a webpage. Oftentimes, multiple IDs have no impact to screen reader users. It’s a technical nonconformance, meaning semantically IDs should always be unique, but most duplicate ID have no impact. The exception is duplicate IDs on input fields.

This is an example of invalid code that will cause this error.





<label for="username" >Username:

<input type="text" class="text-field-lrg" name="username" value="">

</label>

<label for="username">Password:</label>

<input type="password" class="text-field-lrg" placeholder="(case sensitive)" name="password" id="username"></span>

In this case, the username and password both have the same for attribute and the same ID. This typically occurs when copying/pasting code. Since the ID on the input is not unique, when focus is set to username input field, the screen reader will read “username, password, case sensitive, edit, blank”.

This can be fixed by changing the for attribute on the password label and the ID on the password input to a unique ID.

## Ensures every ID attribute value of active elements is unique

Like the above, this is usually a technical nonconformance with no impact to screen reader users. Screen reader users are only impacted when duplicate IDs occur on form inputs.

However, it is best to research all issues to ensure there is no user impact.

## Ensures every ID attribute value used in ARIA and in labels is unique

Again, this is usually a technical nonconformance with no impact to screen reader users. Screen reader users are only impacted when duplicate IDs occur on form inputs.

However, it is best to research all issues to ensure there is no user impact.

## Ensures <meta name="viewport"> does not disable text scaling and zooming

This means that the meta tag has been coded in a way that keeps users from pinching and zooming on tablets and smartphones.

Every meta tag on the page needs to be coded in a way that scaling and zooming is not disabled. If an iframe is injected on the page, the iframe’s meta tag needs to be coded correctly as well.

This can be fixed by ensuring the meta tag does not include the following:

* maximum-scale
* user-scalable="no"

This is considered a valid meta tag.

<meta name="viewport" content="width=device-width, initial-scale=1">

## Ensures that lists are structured correctly

This means that either list items have content that should not be included inside a list item, a <ul> or <ol> was used without corresponding list items or list items were used without a corresponding <ul> or <ol>.

List items should follow this order:

<ul>

<li></li>

<li></li>

</ul>

If a paragraph tag needs included, it should be within the list item.

<ul>

<li>

<p></p>

</li>

<li></li>

</ul>

If list items are used, either a <ul> or <ol> must be included.

This is not valid:

<li></li>

<li></li>

This is also not valid:

<ul>

<p></p>

</ul>

## Ensures <li> elements are used semantically

This means an attribute is being overridden by an ARIA role, state or property.

This is not valid because the ARIA role overrides the <ul>.

<ul role=“navigation”>

<li></li>

<li></li>

</ul>

This issue can be fixed by moving the role so that it is wrapped around the <ul>:

<nav role=“navigation”>

<ul>

<li></li>

<li></li>

</ul>

## Ensures each HTML document contains a non-empty <title> element

This means that each webpage needs a title. The title is the first content that will be read by screen readers when the page loads.

The <title> attribute is added into the <head> of the document.

## Ensures <iframe> and <frame> elements contain a non-empty title attribute

This means that an iframe or frame was included on the page, but the title attribute was not included within the iframe/frame.

The <title> attribute is added into the <head> of the document within the iframe/frame.

## Ensures <img> elements have alternate text or a role of none or presentation

This means that every image on the page needs an alt attribute. If the image is for decorative purposes, add empty alt text (alt="" ).

## Ensures <input type="image"> elements have alternate text

This means that an input with a type of image is being treated like <img>. Since all images must have alt text, alt text needs included.

This can be fixed by including alt text.

<input type="image" src="" alt="Submit">

# Resources

[Web Content Accessibility Guidelines 2.1](https://www.w3.org/TR/WCAG21/)

[Web Accessibility Tutorials](https://www.w3.org/WAI/tutorials/)

[ARIA Best Practices](https://www.w3.org/TR/wai-aria-practices/)

[ARIA roles, states and properties](https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/ARIA_Techniques)

[NVDA screen reader](https://www.nvaccess.org/download/)

[NVDA shortcuts](https://dequeuniversity.com/screenreaders/nvda-keyboard-shortcuts)

[Nest github repository](https://pages.github.nwie.net/Nationwide/nw-dev-portal/content/cross-cutting/accessibility/index.html)

[Color contrast checker](https://webaim.org/resources/contrastchecker/) website

[Colour contrast analyser](https://developer.paciellogroup.com/resources/contrastanalyser/) desktop tool

[Accessibility bookmarklets](https://accessibility-bookmarklets.org/)

[Nationwide Accessibility Community of Practice](https://www.yammer.com/nationwide.com/#/threads/inGroup?type=in_group&feedId=6083543&view=all)

[Accessibility Sharepoint Repository](https://onyourside.sharepoint.com/sites/accessibility/SitePages/Accessibility-SharePoint.aspx)

[Comply](https://ada.nwie.net)

[Axe-core](https://github.com/dequelabs/axe-core)

[Axe plug-in](https://www.deque.com/axe/)