# Accessibility

* Will adopt web accessibility design and will conform to the **World Wide Web Consortium's (W3C)** internationally recognized **Web Content Accessibility Guidelines 2.0** to the level **AA (Double-A conformance)** which means meeting both basic and advanced accessibility standards that can be used by most people with special needs
* Will follow the 4 basic principles
1. **Perceivable** - Information and user interface components will be presentable to users in ways they can perceive
2. **Operable** - User interface components and navigation will be operable
3. **Understandable** - Information and the operation of user interface will be understandable
4. **Robust** - Content will be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies
* client-centric design
* Use of **HTML5**, **Cascading Style Sheets (CSS)** and **standardized templates** that ensure all pages have a consistent format, layout and use of headings, thus enhancing accessibility
* compliant with **screen readers** - can easily read the content in a correct and easy-to-understand order
	+ Consistent document structure of a webpage- first listen to the headings of a webpage and jump directly to the most interested section
	+ Use of “**hidden links**” for screen readers easier navigation- skip over standard parts of the website that are basically the same for every page (main layout, logo, search buttons and language option buttons etc…)
	+ **Hidden text**- not visible on screen, but can be read out by screen readers (visually impaired, extra description for images, videos, sections)
	+ Use of **alternate text** (graphical elements such as photos, logos and banners)
	+ The wording for each link will be carefully chosen (ex. 'click here' or 'here' describe nothing)
* Suppressing the animations or videos on webpages (where needed)
* Suppressing pictures on webpages (where needed)
* use the keyboard exclusively to access all features and contents on the portal
* Adjustable font size
* **Summary** on every webpage for enhanced accessibility (ex ‘in this section you will find….’)

# Principle 1 – Perceivable

Text Alternatives, Time-based Media, Adaptable, Distinguishable

Information and user interface components must be presentable to users in ways they can perceive.

* **Ordering the content in a meaningful sequence**

The objective of this technique is to ensure that the order of content presented to assistive technologies allows the user to make sense of the content.

* **Positioning content based on structural markup**

The objective of this technique is to demonstrate how visual appearance may be enhanced via style sheets while still maintaining a meaningful presentation when style sheets are not applied

* **Making the DOM order match the visual order**

The objective of this technique is to ensure that the order of content in the source code is the same as the visual presentation of the content. The order of content in the source code can be changed by the author to any number of visual presentations with CSS. Each order may be meaningful in itself but may cause confusion for assistive technology users. This could be due to the user switching off the author-specified presentation, by accessing the content directly from the source code (such as with a screen reader), or by interacting with the content with a keyboard. If a blind user, who reads the page with a screen reader that follows the source order, is working with a sighted user who reads the page in visual order, they may be confused when they encounter information in different orders. A user with low vision who uses a screen magnifier in combination with a screen reader may be confused when the reading order appears to skip around on the screen. A keyboard user may have trouble predicting where focus will go next when the source order does not match the visual order.

**Note:** The tabindex attribute in HTML has two functions. One is to make an element focusable and the other is to assign the element a position in the focus order. A tabindex of 0 makes an element focusable, but adds it to the focus order in the order of source elements.

* **Using *aria-label* to provide labels for objects that can be read by assistive technology**

The aria-label attribute provides the text label for an object, such as a button. When a screen reader encounters the object, the aria-label text is read so that the user will know what it is.

<div role="navigation" aria-label="Primary"></div>

* **Using aria-labelledby to provide a name for user interface controls**The purpose of this technique is to provide names for user interface controls that can be read by assistive technology. WAI-ARIA provides a way to associate text with a section, drawing, form element, picture, and so on, using the aria-labelledby property.

Labelling a simple text field:

<input name="searchtxt" type="text" aria-labelledby="searchbtn">

<input name="searchbtn" id="searchbtn" type="submit" value="Search">

A label from multiple sources:

<label id="l1" for="f3">Notify me</label>

<select name="amt" id="f3" aria-labelledby="l1 f3 l2">

 <option value="1">1</option>

 <option value="2">2</option>

</select>

<span id="l2" tabindex="-1">days in advance</span>

Labelling a non-text-content:

<div role="img" aria-labelledby="star\_id">

<img src="fullstar.png" alt=""/>

</div>

<div id="star\_id">4 of 5</div>

* **Combining adjacent image and text links for the same resource**

A link contains an icon and text, and the site help refers to the icon. The img has a text alternative which is the name used for the icon in the site help, which describes clicking the home page icon.

<a href="home.html">

 <img src="house.gif" alt="home page icon">

 Go to the home page

</a>

* **Using alt attributes on img elements**

When an image contains words that are important to understanding the content, the alt text should include those words

<img src="newsletter.gif" alt="Free newsletter.” />

* **Using ARIA landmarks to identify regions of a page**

Landmarks help assistive technology (AT) users orient themselves to a page and help them navigate easily to various sections of a page. They are inserted into the page using the role attribute on an element that marks the section. The value of the attribute is the name of the landmark. These role values are:

* + **banner**: A region that contains the prime heading or internal title of a page.
	+ **complementary**: Any section of the document that supports the main content, yet is separate and meaningful on its own.
	+ **contentinfo**: A region that contains information about the parent document such as copyrights and links to privacy statements.
	+ **form**: A region of the document that represents a collection of form-associated elements
	+ **main**: Main content in a document
	+ **navigation**: A collection of links suitable for use when navigating the website.
	+ **search**: The search tool of a website.

<div id="header" role="banner">A banner image and introductory title</div>

<div id="sitelookup" role="search">....</div>

<div id="nav" role="navigation">...a list of links here ... </div>

<div id="content" role="main"> ... Main content ...</div>

<div id="rightsideadvert" role="complementary">....an advertisement here...</div>

<div id="footer" role="contentinfo">(c)Copyright</div>

* **Using role=heading to identify headings**

The purpose of this technique is to provide a way for Assistive Technologies (AT) to identify a piece of content as a heading. Applying role="heading" to an element causes an AT (like a screen reader) to treat it as though it were a heading.

If there is more than one heading on the page and the heading hierarchy is defined through the visual presentation, the aria-level attribute should be used to indicate the hierarchical level of the heading.

<div role="heading">Global News items</div>
...

<div role="heading" aria-level="7">Jonagold/div>

<p>Jonagold is a cross between the Golden Delicious and Jonathan varieties...</p>

* **Using grouping roles to identify related form controls**

The objective of this technique is to mark up a set of related controls within a form as a group.

<div role="group" aria-labelledby="ssn1">

 <span id="ssn1">Social Security#</span>

 <input size="3" type="text" aria-required="true" title="First 3 digits" />-

 <input size="2" type="text" aria-required="true" title="Next 2 digits" />

</div>

Related CSS:

div[role=radiogroup] {

 border: black thin solid;

}

* **Using the region role to identify a region of the page**

This technique demonstrates how to assign a generic region role to a section of a page so that user agents and assistive technologies may be able to programmatically identify it.

<div role="region" aria-labelledby="pollhead">

<h3 id="pollhead">This week's Poll</h3>

...

</div>

* **Using label elements to associate text labels with form controls**

The objective of this technique is to use the label element to explicitly associate a form control with a label.

<div role="region" aria-labelledby="pollhead">

<h3 id="pollhead">This week's Poll</h3>

...

</div>

* **Providing textual identification of items that otherwise rely only on sensory information to be understood**

The objective of this technique is to ensure that items within a Web page are referenced in the content not only by shape, size, sound or location, but also in ways that do not depend on that sensory perception.

* + A round button is provided on a form to submit the form and move onto the next step in a progression. The button is labeled with the text "go." The instructions state, "to submit the form press the round button labeled go ". This includes both shape and textual information to locate the button.
	+ Instructions for a Web page providing on-line training state, "Use the list of links to the right with the heading, 'Class Listing' to navigate to the desired on-line course." This description provides location as well as textual clues to help find the correct list of links.
* **Ensuring that information conveyed by color differences is also available in text**

The objective of this technique is to ensure that when color differences are used to convey information, such as required form fields, the information conveyed by the color differences are also conveyed explicitly in text.

* **Ensuring that additional visual cues are available when text color differences are used to convey information**

The intent of this technique is to provide a redundant visual cue for users who may not be able to discern a difference in text color. Color is commonly used to indicate the different status of words that are part of a paragraph or other block of text or where special characters or graphics cannot be used to indicate which words have special status. This technique describes a way to provide cues in addition to color so that users who may have difficulty perceiving color differences or have low vision can identify them.

* **Specifying the size of text containers using em units**

The objective of this technique is to specify the width and/or height of containers, that contain text or that will accept text input, in em units. This will allow user agents that support text resizing to resize the text containers in line with changes in text size settings.

The width and/or height of text containers that have been specified using other units risk text cropping because it falls outside the container boundaries when the text size has been increased.

#nav\_menu { width: 20em; height: 100em }

#nav\_list { font-size: 100%; }

* **Using percent for font sizes**

The objective of this technique is to specify text font size proportionally so that user agents can scale content effectively. If a font-size is specified for the body element, all other elements inherit that value, unless overridden by a more specific selector.

Related CSS:

strong {

font-size: 120%

}

* **Using named font sizes**

The objective of this technique is to specify a named font size that expresses the relative font size desired. These values provide hints so that the user agent can choose a font-size relative to the inherited font-size.

Related CSS:

strong {

font-size: larger

}

* **Using liquid layout**

The objective of this technique is to be able to present content without introducing horizontal scroll bars by using layout techniques that adapt to the available horizontal space. Liquid layouts define layout regions that both resize with text, and reflow as needed to display the region on the screen. Although the exact layout therefore varies, the relationship of elements and the reading order remains the same. This is an effective way to create designs that present well on different devices and for users with different font size preferences.

Related CSS:

strong {

font-size: larger

}

* **Providing controls on the Web page that allow users to incrementally change the size of all text on the page up to 200 percent**

The purpose of this technique is to provide a mechanism on the Web page to incrementally increase the size of text. Many people with low vision do not use magnifying software, and they may not be familiar with their browser's text size adjustments. This may be particularly true of older people who are learning about computers later in life and who may be experiencing age related vision loss. It may also be true of some people with cognitive disabilities who require increased font size.

This technique provides a mechanism that some users will find easier to use. The mechanism may include links or buttons that will switch the visual presentation to a different style sheet or use scripts to change the text size dynamically.

* + - The "increase text size" button has a big letter "T" with an upward arrow and the "decrease text size" button has a small letter "T" with a down arrow. There is alt text on each button.
		- A site has a number of style sheets with different text size. The user can choose any of the style sheets if their browser provides this functionality. Each page also includes the links "Increase text size" and "Decrease text size" that will change the style sheet currently applied to the appropriate alternate style sheet.
		- A site includes the text "Change text size:" followed by text links "Up" and "Down" on every Web page. The links trigger a Javascript that alters the value of the text-size property accordingly.
* **Ensuring that there is no loss of content or functionality when the text resizes and text containers do not change their width**

Some user agents support changing the size of text without changing other dimensions of the text container. Loss of content or functionality can occur when the text overflows the space that was allocated for it. However, the layout properties may provide a way to continue to display the content effectively. The block sizes may be defined wide enough that the text does not overflow when resized by 200%. Text may wrap when it no longer fits within the block, and the block may be tall enough that all the text continues to fit in the block. The block may provide scrollbars when the resized text no longer fits.

HTML and CSS are used to create a two-column layout for a page of text. Using the default value of the **white-space** property, **normal**, causes text to wrap. So as the size of the text is increased to 200%, the text reflows and the column of text grows longer. If the column is too long for the viewport, the user agent provides scrollbars so the user can scroll text into view because the author has specified the CSS rule **overflow:scroll** or **overflow:auto**.

# Principle 2 – Operable

Keyboard Accessible, Enough Time, Seizures, Navigable

User interface components and navigation must be operable.

* **Ensuring keyboard control for all functionality**

The objective of this technique is to provide keyboard operation for all the functionality of the page. When all functionality of content can be operated through a keyboard or keyboard interface, it can be operated by those with no vision as well as by those who must use alternate keyboards or input devices that act as keyboard emulators like speech input software or on-screen keyboards.

* A page with images used as links changes when the user hovers over the image with a mouse. To provide keyboard users with a similar experience, the image is also changed when a user tabs to it.
* A page that allows users to click and drag items in a list to reorder them also includes a series of controls that allows keyboard users to move items up, down or to the beginning and end of the list.
* The mobile version of a web site includes a menu button that is tapped to open a site menu, which is implemented as a floating overlay. To provide access to people using external keyboards or ability switches with their mobile device, the menu button and the site menu are both implemented such that they can be operated via the mobile device's keyboard interface.
* **Ensuring that users are not trapped in content**

The objective of this technique is to ensure that keyboard users do not become trapped in a subset of the content that can only be exited using a mouse or pointing device. A common example is content rendered by plug-ins. Plug-ins are user agents that render content inside the user agent host window and respond to all user actions that takes place while the plug-in has the focus. If the plug-in does not provide a keyboard mechanism to return focus to the parent window, users who must use the keyboard may become trapped in the plug-in content.

This problem can be avoided by using one of the following mechanisms to provide a way for users to escape the subset of the content:

* Ensuring that the keyboard function for advancing focus within content (commonly the tab key) exits the subset of the content after it reaches the final navigation location.
* Providing a keyboard function to move the focus out of the subset of the content. Be sure to document the feature in an accessible manner within the subset.
* If the technology used in the subset of the content natively provides a "move to parent" keyboard command, documenting that command before the user enters the plug-in so they know how to get out again.
* **Providing a checkbox on the first page of a multipart form that allows users to ask for longer session time limit or no session time limit**

The objective of this technique is to minimize the risk that users with disabilities will lose their work by providing a checkbox to request additional time to complete multipart forms. The checkbox can allow the user to request a specific amount of additional time (for example 15 minutes) or an indefinite extension. (Note that allowing an indefinite extension would be inappropriate if it jeopardized user privacy or network security.)

* **Adding a link at the top of each page that goes directly to the main content area**

The objective of this technique is to provide a mechanism to bypass blocks of material that are repeated on multiple Web pages by skipping directly to the main content of the Web page. The first interactive item in the Web page is a link to the beginning of the main content. Activating the link sets focus beyond the other content to the main content.

* A "Skip to main content" link
* **Adding a link at the beginning of a block of repeated content to go to the end of the block**

The objective of this technique is to provide a mechanism to bypass a block of material by skipping to the end of the block. The first link in the block or the link directly preceding the block moves focus to the content immediately after the block. Activating the link advances the keyboard focus past the block. When there are multiple blocks to be skipped, the user skips from block to block via these links.

* "Skip navigation links"
* A book index "Skip Links into Index"
* "Skip Section Links"
* **Adding links at the top of the page to each area of the content**The objective of this technique is to provide a mechanism to bypass blocks of material by providing a list of links to the different sections of the content. The links in this list, like a small table of contents at the beginning of the content, set focus to the different sections of the content.
* “The Web pages on a site all start with three links that navigate to the main content of that Web page, the search field, and the navigation bar.”
* **Providing heading elements at the beginning of each section of content**

Headings are designed to convey logical hierarchy. Skipping levels in the sequence of headings may create the impression that the structure of the document has not been properly thought through or that specific headings have been chosen for their visual rendering rather than their meaning. When headings are nested **hierarchically**, the most important information is given the highest logical level, and subsections are given subsequent logical levels.(i.e., h2 is a subsection of h1). Providing this type of structure will help users understand the overall organization of the content more easily.

Since headings indicate the start of important sections of content, it is possible for users with assistive technology to jump directly to the appropriate heading and begin reading the content. This significantly speeds interaction for users who would otherwise access the content slowly.

* **Creating Custom Dialogs in a Device Independent Way**

Site designers often want to create dialogs that do not use the pop-up windows supplied by the browser. This is typically accomplished by enclosing the dialog contents in a div and placing the div above the page content using z-order and absolute positioning in CSS.

To be accessible, these dialogs must follow a few simple rules.

* Trigger the script that launches the dialog from the onclick event of a link or button.
* Place the dialog div into the Document Object Model (DOM) immediately after the element that triggered it. The triggering element will maintain focus, and inserting the dialog content after that element will make the content inside the dialog next in the screen-reader reading order and next in the tab order. The dialog can still be absolutely positioned to be elsewhere on the page visually. This can be done either by creating the dialog in the HTML and hiding it with CSS, as in the example below, or by inserting it immediately after the triggering element with script.
* Ensure that the HTML inside the dialog div meets the same accessibility standard as other content.
* It is also nice, but not always necessary, to make the launching link toggle the dialog open and closed, and to close the dialog when the keyboard focus leaves it.
* **Identifying the purpose of a link using link text combined with the text of the enclosing sentence**

The objective of this technique is to identify the purpose of a link from the link and its sentence context. The sentence enclosing the link provides context for an otherwise unclear link. The description lets a user distinguish this link from links in the Web page that lead to other destinations and helps the user determine whether to follow the link. Note that simply providing the URI of the destination is generally not sufficiently descriptive.

* A Web page contains the sentence "To advertise on this page, click here."

**Note:** Although the link phrase 'click here' is not sufficient to understand the link, the information needed precedes the link in the same sentence.

* In the news summary containing the sentence "The Smallville Times reports that the School Board chose a 2007 school calendar that starts on August 27.", the words "reports that" are a link to an article in the Smallville Times about the School Board meeting.

**Note:** Although this example satisfies the Success Criterion, putting information needed to understand the link after the link in this way is awkward for those who are reading through the document with a screen reader.

* **Using CSS to hide a portion of the link text**

The objective of this technique is to supplement the link text by adding additional text that describes the unique function of the link and styling the additional text so that it is not rendered on the screen by user agents that support CSS.

This technique works by creating a CSS selector to target text that is to be hidden. This ensures the text does not display on screen but remains accessible to assistive technologies such as screen readers and braille displays. Note that the technique does not use **visibility:hidden** or **display:none** properties, since these can have the unintentional effect of hiding the text from assistive technology in addition to preventing on-screen display.

<p>Washington has announced plans to stimulate economic growth. <a href="#"> <span>Washington stimulates economic growth </span> Full Story</a></p>

Related CSS:

a span {

height: 1px;

width: 1px;

position: absolute;

overflow: hidden;

top: -10px;

}

* **Providing a Table of Contents**

The information in the document is usually organized hierarchically, and is intended to be read sequentially.

The table of contents serves two purposes:

* + It gives users an overview of the document's contents and organization.
	+ It allows readers to go directly to a specific section of an on-line document.

This section contains a **table of contents** that is a hierarchical list of links to the sections and subsections of the document. The hierarchy of the table of contents reflects the organization of the sections, and each item in the table of contents is a link that takes the user directly to that section.

* **Providing a Site Map**

A site map is a Web page that provides links to different sections of the site. To make the site map available within the site, at a minimum every page that is listed in the site map contains a link to the site map.

The site map serves several purposes:

* + It provides an overview of the entire site.
	+ It helps users understand what the site contains and how the content is organized.
	+ It offers an alternative to complex navigation bars that may be different at different parts of the site.

* **Providing a search function to help users find content**

Implementing a search function that will spell-check the terms, include different endings for the terms (stemming), and allow for the use of different terminology (synonyms) will further increase the accessibility of the search function.

Test procedure:

1. Check that the Web page contains a search form or a link to a search page
2. Type text into the search form that occurs in the set of Web pages
3. Activate the search
4. Check that the user is taken to a page that contains the search term
5. Check that the user is taken to a page that contains a list of links to pages containing the search term
* **Providing a breadcrumb trail**

It can be helpful to users to separate the items in the breadcrumb trailing with a visible separator. Examples of separators include ">", "|", "/", "::", and "→"

* Home :: Developer Center :: How To Center
* Home / Galleries / Antarctica / Penguins / Gentoo Penguin
* You are here: Acme Company → Electronics → Computers → Laptops

# Principle 3 – Understandable

Readable, Predictable, Input Assistance
Information and the operation of user interface must be understandable.

* **Using language attributes on the html element**

The objective of this technique is to identify the default language of a document by providing the **lang** and/or **xml:lang** attribute on the **html** element. Identifying the language of the document is important for a number of reasons:

* It allows braille translation software to substitute control codes for accented characters, and insert control codes necessary to prevent erroneous creation of Grade 2 braille contractions.
* Speech synthesizers that support multiple languages will be able to orient and adapt to the pronunciation and syntax that are specific to the language of the page, speaking the text in the appropriate accent with proper pronunciation.
* Marking the language can benefit future developments in technology, for example users who are unable to translate between languages themselves will be able to use machines to translate unfamiliar languages.
* Marking the language can also assist user agents in providing definitions using a dictionary.

<html lang="en">

<html lang="mk">

* **Describing what will happen before a change to a form control that causes a change of context to occur is made**

The objective of this technique is to provide information to users about what will happen when a change to a form control results in a change of context. Because changing the value of a form control does not typically result in a change of context, it is important that authors provide instructions that make the user aware of the behavior in advance. Where possible, it is a good idea to programmatically associate the instructions describing the change with the form control itself.

The following are some examples of how to provide the instruction in different situations.

* Provide instruction on the Web page with reading order that precedes the user interface element that causes change of context by change of setting.
* For a multi-step process where users must complete particular steps in order to reach the user interface element where changes of setting would cause a change of context, provide the instruction as part of the process prior to the step where they would encounter the change of context.
* **Presenting repeated components in the same relative order each time they appear**
The objective of this technique is to make content easier to use by making the placement of repeated components more predictable. This technique helps maintain consistent layout or presentation between Web pages by presenting components that are repeated in these Web units in the same relative order each time they appear. Other components can be inserted between them, but their relative order is not changed.
* **Providing a help link on every Web page**

The objective of this technique is to provide context sensitive help for users as they enter data in forms by providing at least one link to the help information on each Web page. The link targets a help page with information specific to that Web page. Another approach is to provide a help link for every interactive control. Positioning this link immediately before or after the control allows users to easily tab to it if they have problems in the control.

* **Providing text instructions at the beginning of a form or set of fields that describes the necessary input**

The objective of this technique is to help the user avoid input errors by informing them ahead of time about restrictions on the format of data that they must enter. Instructions on such restrictions are provided at the top of forms.

Enter requested information.
Dates should be entered in mm/dd/yyyy format.

* **Providing expected data format and example**

The objective of this technique is to help the user avoid input errors by informing them about restrictions on the format of data that they must enter. This can be done by describing characteristics of the format or providing a sample of the format the data should have.

<label for="date">Date (dd-mm-yyyy)</label>

<input type="text" name="date" id="date" />

* **Identifying a required field with the aria-required property**

The objective of this technique is to provide programmatic indication that a form field (which shown through presentation to be required) is mandatory for successful submission of a form.

The WAI-ARIA **aria-required** property indicates that user input is required before submission. The **aria-required** property can have values of "true" or "false"

<p>

 <label for="usrname">Login name: </label>

<input type="text" name="usrname" id="usrname" aria-
required="true"/>[\*]

</p>

<ul id="rg" role="radiogroup" aria-required="true" aria-
 labelledby="radio\_label">

 <li id="rb1" role="radio">Yes</li>

 <li id="rb2" role="radio">No</li>

 </ul>

Related CSS:

[aria-required=true] {

 border: red thin solid;

}

[data-required=true]:after {

 content: url('/iconStar.gif');

}

# Principle 4 – Robust

Compatible
Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

* **Validating Web pages**
The objective of this technique is to avoid ambiguities in Web pages that often result from code that does not validate against formal specifications.

Validation will usually eliminate ambiguities (and more) because an essential step in validation is to check for proper use of that technology's markup (in a markup language) or code (in other technologies). Validation does not necessarily check for full conformance with a specification but it is the best means for automatically checking content against its specification.

* **Ensuring that opening and closing tags are used according to specification**

The objective of this technique is to avoid key errors that are known to cause problems for assistive technologies when they are trying to parse content which involve having opening and closing tags that are not used according to specification.

* **Ensuring that id attributes are unique on a Web page**

The objective of this technique is to avoid key errors that are known to cause problems for assistive technologies when they are trying to parse content that has the same id attribute on different elements. These errors can be avoided by making sure the Web page does not have duplicate **id** values.

* **Ensuring that elements do not contain duplicate attributes**

The objective of this technique is to avoid key errors that are known to cause problems for assistive technologies when they are trying to parse content that has duplicate attributes on the same element.

# References, links, tips & tricks

* <http://sites.udel.edu/webpresence/accessibility/>
* <http://sites.udel.edu/webpresence/accessibility/best-practices/>
* <http://sites.udel.edu/webpresence/accessibility/resources/>
* <https://sites.udel.edu/webpresence/accessibility/web-accessibility-at-ud/>
* <http://diveintoaccessibility.org/?hg=0&nr=0>
* <http://www.456bereastreet.com/archive/201108/do_not_use_displaynone_to_visually_hide_content_intended_for_screen_readers/>
* <http://www.accessiq.org/create/content/how-to-create-an-accessible-infographic>
* <http://a11yproject.com/posts/skip-nav-links/>
* <https://www.w3.org/TR/UNDERSTANDING-WCAG20/navigation-mechanisms-skip.html>
* <https://www.w3.org/TR/2012/NOTE-WCAG20-TECHS-20120103/G1>
* <https://www.w3.org/TR/2012/NOTE-WCAG20-TECHS-20120103/H69>
* <https://www.w3.org/WAI/intro/aria>
* <https://www.w3.org/TR/html-aam-1.0/>
* <https://www.w3.org/TR/html-aam-1.0/#accessible-name-and-description-computation>
* <http://a11yproject.com/posts/aria-landmark-roles/>
* <https://github.com/paypal/bootstrap-accessibility-plugin>
* <https://paypal.github.io/bootstrap-accessibility-plugin/demo.html>
* <http://shakedwebdesign.com/dev/template/index5.html>
* <http://jbst.eu/bootstrap-a11y-theme/>
* <https://www.w3.org/WAI/WCAG20/quickref/?showtechniques=111%2C121%2C122>
* <https://blogs.msdn.microsoft.com/winuiautomation/2015/09/29/so-how-will-you-help-people-work-with-text-part-1-introduction/>
* <https://www.govt.nz/browse/leaving-nz/move-and-live-in-australia/moving-to-australia/>
	+ **benchmark** > jquery.simulate.js <https://github.com/jquery/jquery-simulate> for Chrome, Mozilla Safari
	+ **default IE**