Web Author, (Non CMS) Readiness Checklist

General Web Page Composition

1.1 - Use valid, standard web programming code.

For web pages, indicate the programming language you are using by starting your code with a standard document type declaration, such as:

```xml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Ref: WCAG 1.0 3.2, 11.1, 11.2; WCAG 2.0 4.1.1

Microsoft Web Expressions and FrontPage will automatically generate this content once a new page is created.

1.2 – Uses appropriate markup to convey document structure.

How:

Identify headings, paragraphs, lists, quotations, etc., using the appropriate markup instead of relying solely on formatting. For example, use `<h1>` tags to identify the top-level heading rather than simply making its text large and bold. Do not misuse structural markup for formatting effects, such as using `<blockquote>` to indent a paragraph. Use Cascading Style Sheets (CSS) for formatting.

Ref: WCAG 1.0 3.5, 3.6, 3.7, 5.4; WCAG 2.0 1.3.1

Microsoft’s Web Expressions and Frontpage have different options for designating headings. The image at left, (Figure 1) is the Frontpage drop down text formatting option.

Figure 1: FrontPage Text Formatting Options

The image at the right, (Figure 2), is the Microsoft Web Expressions text formatting or style application pane. Text should be selected and the appropriate style or class designation applied. Text should have a specific style or class designation applied as opposed to simply applying a text size, style, color etc...

Figure 2 Microsoft Web Expressions Text Formatting

1.3 - Provides meaningful page titles.

How:

Every web page should have a title. The title should indicate both the name of the site and the topic of the page and should be unique within the site whenever possible.

Note: Titles should usually be 60 or fewer characters in length.

Ref: WCAG 1.0 13.2; WCAG 2.0 2.4.2
Microsoft’s Web Expressions users can designate the title from the save as dialogue box shown below.

**1.4 - Use headings to introduce sections and sub-sections, and use them in the correct order.**

**How:**

Start the main content area of each page with an `<h1>` heading that indicates the topic of the page as identified in the page title. If the page is long enough to be divided into sections, group each section in a `<div>` and begin each `<div>` with a heading of the appropriate level:

- `<h2>` for main sections,
- `<h3>` for sub-sections,
- `<h4>` for sub-sub sections, and so on.

Use Cascading Style Sheets (CSS) to control the size and appearance of headings.

Note: Headings (and other elements) may be hidden visually but revealed to assistive technologies using a style such as:

```
.invisible { position: absolute; left: -10000px; width: 1px; height: 1px; overflow: hidden; }
```

**Ref:**

WCAG 1.0 3.5, 12.3; WCAG 2.0 1.3.1, 2.4.6

Web Expressions and Frontpage users should apply heading styles to sections and subsections rather than simply altering the font size.
1.5 - Use lists to identify series of related items, including navigation menus.

How:

Use ordered lists (<ol>) when the sequence of items is important; use unordered lists (<ul>) when the order does not matter. Use Cascading Style Sheets (CSS) to control the appearance of numbers and bullets, but do not use bullets on ordered lists or number-styles on unordered lists.

Note: Navigation menus should be preceded by headings of the appropriate level (usually <h2>).

Ref: WCAG 1.0 3.6; WCAG 2.0 1.3.1

2. Text

2.1 - Use text to display text, unless formatting that cannot be achieved with CSS is required.

How:

Whenever possible, use actual text instead of images of text. Cascading Style Sheets (CSS) can be used to achieve specific sizes, colors, or effects. Text that requires exact formatting, such as logos and/or other branding elements, are appropriate exceptions.

Ref: WCAG 1.0 3.1; WCAG 2.0 1.4.5

2.2 - Use relative sizes for fonts.

How:

Set font sizes using relative measurements or avoid setting font sizes altogether.

Ref: WCAG 1.0 3.4; WCAG 2.0 1.4.4

When formatting text use size designations such as medium, large, X–large, rather than a point size, allows the user to view the text at a size determined by their browser settings.
2.3 - Identify the language of text.

How:

Use the lang attribute on the <html> element to identify the primary language of each document, for example, <html lang="en">, for English. Use the lang attribute on <span> or other elements to identify words or phrases in other languages. For example, a Spanish phrase within an English document could be coded as: <span lang="es">se habla español</span>.

Ref: WCAG 1.0 4.1, 4.3; WCAG 2.0 3.1.1, 3.1.2

Microsoft Web Expressions will automatically add this content at the top of the document. To confirm that it is present, view the html code at top of the page. It should appear as shown below.

2.4 - Use images instead of "ASCII art."

How:

Use images with appropriate alternate text instead of ASCII art.

Ref: WCAG 1.0 1.1; WCAG 2.0 1.1.1; 508 a

All images should have an Alternative text description. Long text descriptions are necessary for explaining content such as charts and graphs.
The accessibility properties dialogue box should automatically appear when an image is inserted into the page.

3. Colors

3.1 - Do not convey information with color alone.

How: Whenever color is used as an indicator, use a non-color-based indicator as well. For example, required form fields could be identified with an icon (an image with appropriate alternate text) in the label, as well as with color.

Ref: WCAG 1.0 2.1; WCAG 2.0 1.4.1; 508 c

3.2 - Use contrasting foreground and background colors.

How: For text, use dark colors on light backgrounds, or vice versa. Avoid combinations of red and green as well as busy background images. Text must have a contrast ratio of at least 3:1.

Ref: WCAG 1.0 2.2; WCAG 2.0 1.4.3

4. Images

4.1 - Provide appropriate "alternate text" for all images.

How: ALL images must have appropriate alternate text. As a rule of thumb, consider what you might say if you were reading the web page to someone over the telephone. You do not need to include the words "image of" or "graphic of." Specifically:

- For images that contain words or letters - use alternate text that includes the same words or letters.
- For image links - use alternate text that identifies the link's destination or function. You do not need to include the words "link to."
- For images that are invisible, purely decorative, do not convey meaning, or are completely redundant with text that is already on screen - use alt="" (empty string) to indicate that the image can be safely ignored by a screen reader.
- For CAPTCHA images, use alt text to indicate that the image is being used for CAPTCHA and provide an alternative test in a non-visual medium (e.g., an audio file).

Ref: WCAG 1.0 1.1; WCAG 2.0 1.1.1; 508 a

All images should have an Alternative text description. Long text descriptions are necessary for explaining content such as charts and graphs.
4.2 - Provide full descriptions for graphs, diagrams, and other meaningful images.

How:
Present a full description of a meaningful image either on the page on which the image appears or through a link immediately preceding or following the image. Use alternate text to provide a concise name for the image. For example, the alternate text of a graph should state its title and the full description should summarize its trends and/or present a table of its data.

Note: The longdesc attribute of the <img> element can also be used to provide a link to a full description. Because longdesc it is not yet supported by most web browsers, it should not be used as the only method of providing a full description.

Ref: WCAG 1.0 1.1; WCAG 2.0 1.1.1; 508 a

5. Image Maps

5.1 - Provide alternate text for each area in client-side image maps.

How:
Use alternate text that indicates the function or destination of the link for each area of a client-side image map. The image itself should have alternate text that indicates the overall function of the image map.

Ref: WCAG 1.0 1.1; WCAG 2.0 1.1.1; 508 a

Image maps should be avoided whenever possible.

5.2 - Use client-side image maps instead of server-side image maps unless areas cannot be defined with available shapes.

How:
Whenever possible, use client-side image maps instead of server-side image maps. If server-side image maps must be used, provide an accessible alternative that includes the same content and functionality. In cases where it is impossible to create an equivalent accessible version, such as with some geographical imaging and mapping systems, exceptions may be necessary.

Ref: WCAG 1.0 1.2, 9.1; 508 e, f

Image maps should be avoided whenever possible.

6. Sounds

6.1 - Do not convey information with sound alone.

How:
Whenever significant information is provided by sound, include a visual indicator that provides the same information as well.

Ref: WCAG 1.0 1.1; WCAG 2.0 1.1.1, 1.2.1; 508 a

Text transcripts should accompany any audio files posted to a website.

6.2 - Do not automatically play audio.

How:
Do not automatically play audio for more than 3 seconds. Provide a means for users to start audio playback when they desire (e.g., a "play" button).

Ref: WCAG 2.0 1.4.2

To disable auto playback on an audio file within a Web Expressions web page, left click on the audio content and then right click to access the properties.

__6.3 - Provide text transcripts for audio containing speech when it is provided to the public and/or required to be viewed by employees.

How:
Provide a link to an HTML or text transcript of any audio presented on a web site. Transcripts should be posted at the same time the audio is made available.
Communication Access Realtime Translation (CART) providers can transcribe live events.

Ref: WCAG 1.0 1.1; WCAG 2.0 1.1.1, 1.2.1; 508 a

7. Multimedia

__7.1 - Provide synchronized captions for all multimedia that contains essential auditory information when it is provided to the public and/or required to be viewed by employees.

How:
Whenever possible, captions should be implemented using Synchronized Multimedia Integration Language (SMIL) If multimedia contains essential audio but no essential video, a text transcript may be provided instead of captions.

Ref: WCAG 1.0 1.4; WCAG 2.0 1.2.2, 1.2.4; 508 b
Captions for multimedia content are not created with Web Expressions or Frontpage, but within a video editing software, such as Final Cut Pro or Camtasia. The CETLA staff are happy to assist with these projects.

7.2 - Provide audio descriptions for all multimedia that contains essential visual information when it is provided to the public and/or required to be viewed by employees.

How:
Carefully consider whether audio descriptions are necessary to present the significant information of a multimedia recording. Many speech-intensive events, such as speeches, lectures, or conferences, do not contain essential video and, therefore, do not need audio description. When necessary, audio descriptions are usually best implemented by a professional "audio describer."

Ref: WCAG 1.0 1.3; WCAG 2.0 1.2.3, 1.2.5; 508 b

8. Animation

8.1 - Provide a means of pausing any moving, blinking, scrolling, or auto-updating information.

How:
Avoid animation and movement unless it provides significant additional information. If animation is used, provide a means of pausing the animation.

Ref: WCAG 1.0 7.1, 7.2, 7.3; WCAG 2.0 2.2.2

Animation should be avoided, however any video content used should have the auto play option disabled and controls enabled as shown in the screen capture of the media properties below.
8.2 - Do not include content that flashes faster than 3 times per second.

How:

Do not include content that flashes faster than 3 times per second.

Ref: WCAG 1.0 7.1; WCAG 2.0 2.3.1; 508 j

9. Links

9.1 - Ensure that links are understandable out of context.

How:

Use link text that is clear and unambiguous. Link text should usually match the title of the page to which the link points. Ensure that links that point to the same URL use the same link text, and that links that point to different URLs use different link text. If title attributes are used, repeat the text of the link as the beginning of the title, followed by the additional information.

Note: Portions of links may be hidden visually but revealed to assistive technologies using a style such as:

```
.invisible { position: absolute; left: -10000px; width: 1px; height: 1px; overflow: hidden; }
```

Ref: WCAG 1.0 13.1; WCAG 2.0 2.4.4

9.2 - Provide a means of skipping past repetitive navigation links.

How:

Provide a link at the beginning of navigation lists that points to a target at the beginning of the main content area of the page. This link must be visible to screen reader and keyboard users, but can be hidden from other users, e.g.:

```
<style type="text/css">
  a.skipnavigation { position: absolute; left: -10000px; }
  a.skipnavigation:focus, a.skipnavigation:active { position: static;
```
9.3 - Avoid using small links.

How: Make sure that images used for links are reasonably large, preferably 16 pixels by 16 pixels or larger. Use standard or enlarged font sizes for text links, and avoid using text links that are shorter than 4 characters in length. Avoid placing small links close together.

9.4 - Ensure that same-page links move keyboard focus as well as screen focus.

How: To ensure that same-page links work correctly in Internet Explorer 5, 6 & 7, set tabindex="-1" on same-page link targets.

10. Forms

10.1 - Provide labels or titles for all form fields.

How: Use a <label> element whenever possible to identify each form field's label. Set the label element's for attribute to match the corresponding field's id, e.g.:

```html
<label for="fieldId">Label Text:</label>  
<input id="fieldId" name="fieldId" />
```

Add a title attribute to any form field that cannot be associated with a <label>. Because screen readers typically recognize either the <label> or the title (not both), the title must provide the full label, and a <label> should not be associated with the field.

For buttons (<input type="submit|reset|button">), use the value attribute to provide a label; for image buttons (<input type="image">), use the alt attribute for the label.

Ref: WCAG 1.0 12.4; WCAG 2.0 1.3.1, 2.4.6, 3.3.2; 508 n

10.2 - Provide legends for groups of form fields.

How: If possible, group related fields within a <fieldset> element and provide the group name or "question" in the fieldset's <legend>. If a fieldset cannot be used, apply a title attribute to each field in the group including the group name and the field's label. Because screen readers typically recognize either the <label> or the title (not both), the <label> elements should not be associated with their fields when using this technique. Also note that groups of radio buttons (and sometimes checkboxes) can
usually be replaced by a single drop-down list or list box (<select>) with the "question" as the <label> and the "answers" as <option> elements.

Ref: WCAG 1.0 12.3; WCAG 2.0 1.3.1, 2.4.6, 3.3.2; 508 n

___10.3 - Ensure that form fields are in a logical tab order.

How:
Make sure that fields appear in logical order in the HTML code or use the tabindex attribute on each field to set the appropriate order.

Note: Any element with a positive tabindex will come before all elements without tabindex in the tab order. As a result, tabindex must usually be applied to all or none of the focusable elements on a page.

Ref: WCAG 1.0 9.4; WCAG 2.0 2.4.3; 508 n

___10.4 - Avoid placing non-focusable text between form fields.

How:
Instructions should be given within field labels if possible. If instructions are too long to fit within labels, they should be provided in an instructions section before the beginning of the form. If neither of these approaches works, consider using a technique to make the text elements focusable.

Note that disabled form fields (disabled="disabled") are non-focusable; fields may be made read-only (readonly="readonly") to keep them focusable.

Ref: 508 n

___10.5 - Ensure that text in form fields can be enlarged.

How:
To ensure that text in form fields can be easily resized in Internet Explorer 5, 6 & 7, include a style rule such as the following:

input, select, textarea, button { font-size: 100%; }

Ref: WCAG 2.0 1.4.4

11. Tables

___11.1 - Identify a header cell for each column and row in simple data tables.

How:
Use <th> elements with scope="col" (for column headers) or scope="row" (for row headers) to identify column and row header cells. The cell that contains the most uniquely identifying information for the column or row should be the header; usually, this should be the first cell of each column and row. Do not include unnecessary columns or rows for formatting.

Ref: WCAG 1.0 5.1; WCAG 2.0 1.3.1; 508 g

___11.2 - Identify relationships in complex data tables using id and headers attributes.

How:
Whenever possible, simplify complex tables by re-arranging or dividing them into separate tables. If a table cannot be simplified, use a unique id attribute on each header cell, for example: <th id="header1">, and a headers attributes on each data
cell to list the id attributes of header cells that it relates to, for example: <td headers="header1 header2 header3">. Do not include unnecessary columns, rows, colspans, or rowspans for formatting.

See W3C's "Tables in HTML Documents" (http://www.w3.org/TR/html401/struct/tables.html#h-11.4.1) for more information on how to markup complex tables.

Ref: WCAG 1.0 5.2; WCAG 2.0 1.3.1; 508 h

11.3 - Provide summary attributes for data tables.

How:
Provide the name of the table in the summary attribute of the <table> tag. A brief description of the content and structure of the table may also be included in the summary. (No summary should be used if a table is used for layout.)

Ref: WCAG 1.0 5.5; WCAG 2.0 1.3.1

12. Frames

12.1 - Provide concise, unique, and understandable titles for frames.

How:
Give each frame and iframe a concise, unique, understandable title attribute that indicates the frame's function and is unique within the frameset. Do not include the word "frame." Set the <title> element of each page contained within a frame to match its title attribute or to identify the current content of that frame.

Ref: WCAG 1.0 12.1; WCAG 2.0 4.1.2; 508 i

12.2 - Avoid using hidden, empty, or non-essential frames.

How:
Use frames sparingly. If a frame is not necessary for page content, eliminate it. Minimize the nesting of frames. Recognize that information in hidden frames may be read by screen readers.

13. Scripts

13.1 - Ensure that scripted functions are usable with assistive technologies.

How:
Whenever scripts are used, it is the responsibility of the page developer to thoroughly test using assistive technologies to ensure that all information and functionality is accessible. Testing should confirm that content is usable with system large fonts and high contrast display settings, that interface elements are focusable and operable using the keyboard, that tab and reading order are appropriate, and that all content can be identified and operated with leading assistive technology tools. If there is any doubt, err on the safe side by ensuring that the essential elements of the page do not rely on scripts.

Scripting features that are purely decorative and do not present any significant information or functionality do not need to be made accessible.

Ref: WCAG 1.0 6.2, 6.3, 6.5, 8.1; WCAG 2.0 1.3.1, 4.1.2; 508 l
13.2 - Ensure that significant interactions can be performed with both

How:
Whenever using a mouse-only event (e.g., `onmouseover`, `onmouseout`) to trigger a significant script action, use the corresponding keyboard event (e.g., `onfocus`, `onblur`). Ensure that scripts are attached to elements that can receive keyboard focus, such as links and form fields.

Also, make sure that keyboard events do not unintentionally trigger script actions. For example, when a keyboard user arrows through the options in a drop-down list (`<select>`), the `onchange` event fires once for each option; if a script attached to this event reloads the page (or causes some other significant change), it may be impossible for a keyboard user to operate the control.

Ref: WCAG 1.0 6.4, 9.2, 9.3; WCAG 2.0 2.1.1; 508 1194.21 a

13.3 - Avoid changing focus unexpectedly.

How:
In most cases, changes to focus, location, or the current window should be initiated by a user activating (clicking) a link or button. If changes are initiated by other actions, make sure that there is an obvious cause-and-effect relationship between the action and the change. If changes are not likely to be expected, it may be necessary to provide explicit instructions to users before the changes occur.

Ref: WCAG 1.0 8.1; WCAG 2.0 3.2.1, 3.2.2; 508 1

13.4 - Avoid changing content unexpectedly.

How:
In most cases, content should only be changed or added after the current point of focus in the tab/reading order. For example, if entering a value in a field causes text to appear, the text should appear after that field in the tab/reading order. It is also important to ensure that changes happen quickly enough that they are complete before the user reaches the changed element. For example, if selecting an item from a list box changes the value in a subsequent field, the change must be complete before the focus moves to the subsequent field. If content is changed above the user’s focus, or if changes occur slowly enough that a user may move past the changed element before the change is complete, it may be necessary to provide an alert box or other instruction to direct the user to the changed element.

Ref: WCAG 1.0 8.1; 508 1

14. Embedded Objects

14.1 - Use accessible embedded objects whenever possible.

How:
Check with the manufacturer and/or developer to determine if and how the technology can be made accessible.

Ref: WCAG 1.0 8.1; WCAG 2.0 4.1.2; 508 m

14.2 - If an inaccessible embedded object must be used, provide an accessible alternative that includes the same content and functionality.

How:
Wherever an inaccessible object is embedded, also provide or link to an accessible version. Make sure that the information and functionality is completely equivalent and up-to-date. Be sure to consider whether the inaccessible version is actually necessary. In cases where it is impossible to create an equivalent accessible version, such as with some geographical imaging and mapping systems, exceptions may be necessary.

Ref: WCAG 1.0 6.2, 11.4; WCAG 2.0 1.1.1; 508

15. Downloadable Documents

15.1 - Provide natively accessible downloadable documents whenever possible.

How:
Check with the manufacturer and/or publisher of the format of a downloadable document to determine if and how it can be made accessible. If accessibility techniques exist, ensure that the downloadable document fully implements these techniques and meets the functional performance criteria defined in these standards.

Note: See Adobe's Accessibility Resource Center (http://www.adobe.com/accessibility) for information about PDF accessibility.

15.2 - If a downloadable document cannot be made natively accessible, provide an accessible alternative that includes the same content and functionality.

How:
Wherever a link is provided to an inaccessible document, also provide a link to an accessible version, preferably in HTML format. Make sure that the information and functionality is completely equivalent and up-to-date. Be sure to consider whether the inaccessible version is actually necessary. If an accessible alternative cannot be provided electronically, provide information on how to obtain an alternate format (e.g., large print, Braille, audio recording) in a timely manner.

Ref: WCAG 1.0 11.1, 11.4; WCAG 2.0 1.1.1; 508

16. Timing

16.1 - Notify users of time limits and provide a means to extend time if possible.

How:
Provide a clear explanation of any time limits. If possible, offer the user a way to extend or remove the limits. Avoid using time limits unnecessarily.

Ref: 508 p; WCAG 2.0 2.2.1

16.2 - Do not automatically refresh the current page.

How:
Do not use http-equiv="refresh". If necessary, provide a link or control to allow the user to refresh a page at his or her discretion.

Ref: WCAG 1.0 7.4, 7.5; WCAG 2.0 2.2.1

17. Page Layout

17.1 - When using tables for layout, ensure that reading order is logical.
How:
Avoid using tables for layout whenever possible. If tables are used for layout, check the reading order by following the order in which the table cells appear in the code. If necessary, restructure the table to achieve an appropriate reading order.

Ref: WCAG 1.0 5.3; WCAG 2.0 1.3.2

17.2 - When using style sheets for layout, ensure that reading order is logical.

How:
Check the reading order by following the order in which elements appear in the HTML code. Adjust the reading order by rearranging the order in which elements are defined in the code.

Ref: WCAG 1.0 6.1; WCAG 2.0 1.3.2; 508 d

17.3 - Avoid horizontal scrolling.

How:
Design pages so that they can resize to fit the width of the user's browser. Use relative widths and check for horizontal scrolling using a screen resolution of 800 by 600 pixels. If scrolling cannot be avoided, place the least important content in the rightmost part of the page.

Ref: WCAG 2.0 1.4.8

18. Alternate Accessible Versions

18.1 - Use separate accessible versions only as a last resort.

How:
Follow these standards to develop a single site that is universally accessible and efficient to maintain.

Ref: WCAG 1.0 11.4; 508 k