WCAG 2.0 Reference Document

Section 1 – Perceivable

1.1 Text Alternatives – WCAG states that target sites should "Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language[1]".

1.1.1 Non-text Content (Level A) – With the exception of the instances indicated below, any non-text content that is displayed to the user has a text alternative that fulfills the same function. The exceptions are as follows: controls/input, time-based media, tests, sensory experiences, CAPTCHA, non-text content that is decorative. Some examples of non-text content could include data charts, audio recordings of speech, animations that illustrate a concept, video footage, photographs, linked thumbnails, or an image map[2].

Situation A: If a short description can serve the same purpose and present the same information as the non-text content:

Technique G94: Providing Short Text Alternative for Non-Text Content That Serves the Same Purpose and Presents the Same Information as The Non-Text Content[3]:

Examples

- A search button uses an image of a magnifying glass. The text alternative is "search" and not "magnifying glass".
- A picture shows how a knot is tied including arrows showing how the ropes go to make the knot. The text alternative describes how to tie the knot, not what the picture looks like.
- A picture shows what a toy looks like from the front. The text alternative describes a front view of the toy.
- An animation shows how to change a tire. A short text alternative describes what the animation is about. A long text alternative describes how to change a tire.
- A logo of the TechTron company appears next to each product in a list that is made by that and has a short text alternative that reads, "TechTron."
- A chart showing sales for October has an short text alternative of "October sales chart". It also has a long description that provides all of the information on the chart.
- A heading contains a picture of the words, "The History of War" in stylized text. The alt text for the picture is "The History of War".
- An image of a series of books on a shelf contains interactive areas that provide the navigation means to a Web page about the particular book. The text alternative "The books available to buy in this section. Select a book for more details about that book." describes the picture and the interactive nature.

Testing Procedure

1. Remove, hide, or mask the non-text content
2. Replace it with the text alternative
3. Check that nothing is lost (the purpose of the non-text content is met by the text alternative)
4. If the non-text content contains words that are important to understanding the content, the words are included in the text alternative

Situation B: If a short description cannot serve the same purpose and present the same information as the non-text content (e.g., a chart or diagram):

[1] https://www.w3.org/WAI/WCAG21/quickref/#non-text-content
[2] https://www.w3.org/WAI/WCAG21/Understanding/non-text-content.html
[3] https://www.w3.org/WAI/WCAG21/Techniques/general/G94.html

Technique G95: Providing Short Text Alternatives That Provide a Brief Description of The Non-Text Content[4]:

Example

- A chart showing sales for October has a short text alternative of "October sales chart". It also has a long description that provides all of the information on the chart.

Testing Procedure

1. Check for the presence of a short text alternative that provides a brief description of the non-text content.

Situation C: If non-text content is a control or accepts user input:

Technique G82: Providing A Text Alternative That Identifies the Purpose of The Non-Text Content[5]:

Examples

- An eye-hand coordination development applet has the following text alternative "Applet that uses the mouse and moving targets to develop eye-hand coordination"
- A camera applet that has a round disk where you push on the edges to control a remote camera and a slider in the middle for zooming has the following text alternative "Control for aiming and zooming remote video camera".

Testing Procedures

1. Remove, hide, or mask the non-text content
2. Replace it with the text alternative
3. Check that the purpose of the non-text content is clear - even if function is lost.

Situation D: If non-text content is time-based media (including live video-only and live audio-only); a test or exercise that would be invalid if presented in text; or primarily intended to create a specific sensory experience:

Acceptable Technique Solution:

1. Providing a descriptive label[6]

Situation E: If non-text content is a CAPTCHA:

Technique G143: Providing a text alternative that describes the purpose of the CAPTCHA[7] **AND** Technique G144: Ensuring that the Web Page contains another CAPTCHA serving the same purpose using a different modality[8]

Examples

For G143

- A CAPTCHA test asks the user to type in text that is displayed in an obscured image. The text alternative is "Type the word in the image".
- A CAPTCHA test asks the user to type in text that is played in an audio file. The text alternative is "Type the letters spoken in the audio".

For G144

- A Web page that includes a CAPTCHA test that must be completed successfully before the user can advance to the next step in a process. The page includes both a visual task, such as typing words displayed in a image, and an

[4] https://www.w3.org/WAI/WCAG21/Techniques/general/G95.html
[5] https://www.w3.org/WAI/WCAG21/Techniques/general/G82.html
[6] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=122%2C131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412%2C244#non-text-content
[7] https://www.w3.org/WAI/WCAG21/Techniques/general/G143.html
[8] https://www.w3.org/WAI/WCAG21/Techniques/general/G144.html

audio task, such as typing letters spoken in an audio file. A user with hearing disabilities who cannot pass the audio CAPTCHA may be able to pass the visual CAPTCHA.

- A blog comment form includes a visual CAPTCHA that must be completed before a user can submit comments. In addition to the visual CAPTCHA, it includes a CAPTCHA with a form field that asks, "What is two plus seven?" with a text entry field that allows users to enter the correct answer.

Testing Procedures

For G143

1. Remove, hide, or mask the CAPTCHA.
2. Replace it with the text alternative.
3. Check that the text alternative describes the purpose of the CAPTCHA.

For G144

For each CAPTCHA in a Web page:

1. Check that the Web page contains another CAPTCHA for the same purpose but using a different modality.

1.2 Time-based Media[9] – In order to reach at least AA conformance, WCAG simply required that the target site provided alternatives for any time-based media. This included pre-recorded audio and video, pre-recorded captions, audio descriptions or similar media alternatives, live-captions, and pre-recorded audio descriptions.

1.2.1 Audio-only and Video-only (Pre-Recorded) (Level A)[10] – The purpose of this requirement is to make time-based media information available through text-based alternatives since text could be displayed in any sensory modality (for instance, visual, auditory, or tactile) to suit the user's demands. Some examples of time-based media could include an audio recording of a speech, an audio recording of a press conference, an animation illustrating a concept, or a video file.

Situation A: If the content is prerecorded audio-only:

Technique G158: Providing an Alternative for Time-Based Media for Audio-only[11]

Examples

- A podcast includes a description of new features in a recent software release. It involves two speakers informally discussing the new and updated features and describing how they are used. One of the speakers works from a list of questions that is used to outline the discussion prior to recording. After the recording is complete, the outline is then edited and supplemented to match the dialogue etc. The resulting transcript is then made available on the speakers Web site along with the audio-only file. The text alternative that identifies the audio only content reads, "Episode 42: Zap Version 12 (text transcript follows)" and the link to the transcript is provided immediately following the audio-only content.

Testing Procedure

1. View the audio-only content while referring to the alternative for time-based media.
2. Check that the dialogue in the transcript matches the dialogue and information presented in the audio-only presentation.
3. If the audio includes multiple voices, check that the transcript identifies who is speaking for all dialogue.
4. Check that at least one of the following is true:
   a. The transcript itself can be programmatically determined from the text alternative for the audio-only content
   b. The transcript is referred to from the programmatically determined text alternative for the audio-only content

[9] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#time-based-media

[10] https://www.w3.org/WAI/WCAG21/Understanding/audio-only-and-video-only-prerecorded.html

[11] https://www.w3.org/WAI/WCAG21/Techniques/general/G158.html

5. If the alternate version(s) are on a separate page, check for the availability of link(s) to allow the user to get to the other versions.

Technique SL17: Providing Static Alternative Content for Silverlight Media Playing in a MediaElement

Examples

- This example has a UI definition in XAML and interaction logic in C#. In this case the MediaElement has no visual representation itself and is 0x0 size because it plays audio only. As a simple placeholder, this example displays the text "Library of Congress Audio" to represent the media element as something visible in the UI. In addition to Play/Stop controls, this interface includes a Display Transcript button. Activating the button displays static text that represents the transcript of the audio.

Testing Procedure

1. Using a browser that supports Silverlight, open an HTML page that references a Silverlight application through an object tag. That application has audio-only media content and is expected to supply a text alternative, or has media that is expected to be replaced entirely with a transcript or similar text alternative.
2. Check for a control that indicates that activating it will supply static alternative content for the media. Activate the control.
3. Verify that the media control is replaced with alternate content, and that assistive technologies represent the change to the user interface

Situation B: If the content is prerecorded video-only:

Technique G159: Providing an Alternative for Time-Based media for Video-only Content[12]

Examples

- An animation shows how to assemble a woodworking project. There is no audio, but the animation includes a series of numbers to represent each step in the process as well as arrows and picture-in-picture highlights illustrating how the assembly is completed. It also includes short outtake animations illustrating what will happen if assembly is done incorrectly. A text alternative that identifies the video-only content reads, "Breadbox assembly video (text description follows)," and the text description of the video includes a full text description of each step in the video.

Testing Procedure

1. View the video-only content while referring to the alternative for time-based media.
2. Check that the information in the transcript includes the same information that is in the video-only presentation.
3. If the video includes multiple people or characters, check that the transcript identifies which person or character is associated with each action described.
4. Check that at least one of the following is true:
   a. The transcript itself can be programmatically determined from the text alternative for the video-only content
   b. The transcript is referred to from the programmatically determined text alternative for the video-only content
5. If the alternate version(s) are on a separate page, check for the availability of link(s) to allow the user to get to the other versions.

Technique G166: Providing Audio that Describes the Important Video Content and Describing it as Such[13]

Example

- A Web page has a link to a video-only presentation of a spaceship landing on Mars. The link to the video is a picture of a spaceship. Near the video is a link to an audio file of a person describing the video. This would look something like the following code example in HTML.

[12] https://www.w3.org/WAI/WCAG21/Techniques/general/G159
[13] https://www.w3.org/WAI/WCAG21/Techniques/general/G166

```
<a href="../video/marslanding.mp4"><img src="../images/spaceship.jpg"
                alt="Mars landing, video-only" width="193" height="255"/></a>
                <br />
                <a href="Mars_landing_audio.mp3">Audio description of "Mars Landing"</a>
```

Testing Procedure
For a Web page that contains video-only content:
1. Check that there is link to an audio alternative which describes the contents of the video immediately before or after the video-only content.

Technique SL17: Providing Static Alternative Content for Silverlight Media Playing in a MediaElement

Example
{See Above}

Testing Procedure
{See Above}

1.2.2 Captions (Pre-Recorded) (Level A)[14] – All prerecorded audio information in synced media must have included captions unless the media is expressly marked as a text alternative and is otherwise not captioned. Some examples might have included a captioned video recording, captions that describe non-verbal occurrences within a video, or a captioned description of a complex object depicted in a media file.

Technique G93: Providing Open (always visible) Captions[15]

Example
- In order to ensure that everyone can view their online movies, even if users do not know how to turn on captions in their media player, a library association puts the captions directly into the video.
- A news organization provides open captions on all of its material.

Testing Procedure
1. Watch the synchronized media with closed captioning turned off.
2. Check that captions (of all dialogue and important sounds) are visible.

Technique G87: Providing Closed Captions[16]

Example 1
- In order to ensure that users who are deaf can use their interactive educational materials, the college provides captions and instructions for turning on captions for all of their audio interactive educational programs.

Example 2
- The online movies at a media outlet all include captions and are provided in a format that allows embedding of closed captions.

Example 3
- Special caption files including synchronization information are provided for an existing movie. Players are available that can play the captions in a separate window on screen, synchronized with the movie window.

Example 4

[14] https://www.w3.org/WAI/WCAG21/Understanding/captions-prerecorded.html
[15] https://www.w3.org/WAI/WCAG21/Techniques/general/G93.html
[16] https://www.w3.org/WAI/WCAG21/Techniques/general/G87.html

- A video of a local news event has captions provided that can be played over the video or in a separate window depending on the player used.

Testing Procedure

1. Turn on the closed caption feature of the media player
2. View the synchronized media content
3. Check that captions (of all dialogue and important sounds) are visible

Differing Methods to Meet G87 –

Technique SM11: Providing Captions Through Synchronized Text Streams in SMIL 1.0[17]

Example 1: SMIL 1.0 caption sample for Quickime player

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns:qt="http://www.apple.com/quicktime/resources/smilextensions"
  xmlns="https://www.w3.org/TR/REC-smil" qt:time-slider="true">
  <head>
    <layout>
      <root-layout width="320" height="300" background-color="black"/>
      <region top="0" width="320" height="240" left="0" background-color="black"
      id="videoregion"/>
      <region top="240" width="320" height="60" left="0" background-color="black"
      id="textregion"/>
    </layout>
  </head>
  <body>
    <par>
      <video dur="0:01:20.00" region="videoregion" src="salesdemo.mov"
      alt="Sales Demo"/>
      <textstream dur="0:01:20.00" region="textregion" src="salesdemo_cc.txt"
      alt="Sales Demo Captions"/>
    </par>
  </body>
</smil>
```

Example 2: SMIL 1.0 caption sample for RealMedia player

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns="https://www.w3.org/TR/REC-smil">
  <head>
    <layout>
      <root-layout background-color="black" height="310" width="330"/>
      <region id="video" background-color="black" top="5" left="5"
      height="240" width="320"/>
      <region id="captions" background-color="black" top="250"
      height="60" left="5" width="320"/>
    </layout>
  </head>
  <body>
    <par>
      <video src="salesdemo.mpg" region="video" title="Sales Demo"
      alt="Sales Demo"/>
      <textstream src="salesdemo_cc.rt" region="captions"
      system-captions="on" title="captions"
      alt="Sales Demo Captions"/>
    </par>
  </body>
</smil>
```

The example shows a <par> segment containing a <video> and a <code><![CDATA[<textstream> tag. The system-captions attribute indicates that the textstream should be displayed when the user's player setting for captions

[17] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM11.html

indicates the preference for captions to be displayed. The <layout> section defines the regions used for the video and the captions.

Example 3: SMIL 1.0 caption sample with internal text streams

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns="https://www.w3.org/TR/REC-smil">
  <head>
    <layout>
      <root-layout background-color="black" height="310" width="330"/>
      <region id="video" background-color="black" top="5" left="5"
      height="240" width="320"/>
      <region id="captions" background-color="black" top="250"
      height="60" left="5" width="320"/>
    </layout>
  </head>
  <body>
    <par>
      <video src="salesdemo.mpg" region="video" title="Sales Demo"
      alt="Sales Demo"/>
      <text src="data:,This%20is%20inline%20text." region="captions" begin="0s"
      dur="3" alt="Sales Demo Captions">
        <param name="charset" value="iso-8859-1"/>
        <param name="fontFace" value="System"/>
        <param name="fontColor" value="yellow"/>
        <param name="backgroundColor" value="blue"/>
      </text>
    </par>
  </body>
</smil>
```

This example shows a <text> element that includes synchronized text streams within the SMIL file.

Testing Procedure

1. Enabled caption preference in player, if present
2. Play file with captions
3. Check whether captions are displayed

Technique SM12: Providing Captions Through Synchronized Text Streams in SMIL 2.0[18]

Example 1: SMIL 2.0 caption sample for RealMedia player

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns="https://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout backgroundColor="black" height="310" width="330"/>
      <region id="video" backgroundColor="black" top="5" left="5"
      height="240" width="320"/>
      <region id="captions" backgroundColor="black" top="250"
      height="60" left="5" width="320"/>
    </layout>
  </head>
  <body>
    <par>
      <video src="salesdemo.mpg" region="video" title="Sales Demo"
      alt="Sales Demo"/>
      <textstream src="salesdemo_cc.rt" region="captions" systemCaptions="on"
      title="captions" alt="Sales Demo Captions"/>
    </par>
  </body>
</smil>
```

[18] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM12.html

The example shows a <par> segment containing a <video> and a <textstream> tag. The systemCaptions attribute indicates that the textstream should be displayed when the user's player setting for captions indicates the preference for captions to be displayed. The <layout> section defines the regions used for the video and the captions.

Example 2: SMIL 2.0 caption sample with internal text streams for RealMedia player

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns="https://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout backgroundColor="black" height="310" width="330"/>
      <region id="video" backgroundColor="black" top="5" left="5"
      height="240" width="320"/>
      <region id="captions" backgroundColor="black" top="250"
      height="60" left="5" width="320"/>
    </layout>
  </head>
  <body>
    <par>
      <video src="salesdemo.mpg" region="video" title="Sales Demo"
      alt="Sales Demo"/>
      <text src="data:,This%20is%20inline%20text." region="captions"
      begin="0s" dur="3">
        <param name="charset" value="iso-8859-1"/>
        <param name="fontFace" value="System"/>
        <param name="fontColor" value="yellow"/>
        <param name="backgroundColor" value="blue"/>
      </text>
      <text src="data:,This%20is%20a%20second%20text."
      region="captions" begin="3s" dur="3">
        <param name="charset" value="iso-8859-1"/>
        <param name="fontFace" value="System"/>
        <param name="fontColor" value="yellow"/>
        <param name="backgroundColor" value="blue"/>
      </text>
    </par>
  </body>
</smil>
```

This example shows a `<text>` element that includes synchronized text streams within the SMIL file.

Testing Procedure:

1. Enabled caption preference in player, if present
2. Play file with captions
3. Check whether captions are displayed

Technique H95: Using the Track Element to Provide Captions[19]

Example 1: Captions in one language

A `video` element for a video in the English language with an English caption track. The captions are provided in the WebVTT format.

```
<video poster="myvideo.png" controls>
       <source src="myvideo.mp4" srclang="en" type="video/mp4">
       <track src="myvideo_en.vtt" kind="captions" srclang="en" label="English">
                </video>
```

Example 2: Captions in multiple languages

A `video` element for a video in the English language with an English caption track. The captions are provided in the WebVTT format.

```
<video poster="myvideo.png" controls>
```

[19] https://www.w3.org/WAI/WCAG21/Techniques/html/H95.html

```
<source src="myvideo.mp4" srclang="en" type="video/mp4">
<source src="myvideo.webm" srclang="fr" type="video/webm">
<track src="myvideo_en.vtt" kind="captions" srclang="en" label="English">
<track src="myvideo_fr.ttml" kind="captions" srclang="fr" label="French">
        </video>
```

Testing Procedure

For each video element used to play a video:

1. Check that the video contains a track element of kind captions in the language of the video.

Technique SL16: Providing Script-Embedded Text Captions for Mediaelement Content[20]

Example: MediaElement handles MarkerReached, displays marker text in existing TextBox

This example has a UI definition in XAML and interaction logic in C#. The following is the basic UI in XAML. This example is deliberately simple and does not include `AutomationProperties` for identification or user instructions. The most relevant part of this example is that the Silverlight author declares a handler for the event MarkerReached. This event fires potentially hundreds of times, once for each caption in the stream. Each time the event fires, the event handler runs and adds the text to the dedicated `TextBox` in the user interface.

```
<UserControl x:Class="MediaTimelineMarkers.MainPage"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
   <StackPanel x:Name="LayoutRoot" Background="White">
       <MediaElement MarkerReached="OnMarkerReached"
       HorizontalAlignment="Left"
       Source="/spacetime.wmv"
       Width="300" Height="200" />
       <ScrollViewer>
           <TextBox Name="captionText" Height="40"
           IsReadOnly="true" AcceptsReturn="true"/>
       </ScrollViewer>
   </StackPanel>
 </UserControl>

private void OnMarkerReached(object sender, TimelineMarkerRoutedEventArgs e)
{
   captionText.Focus();
   captionText.SelectedText = e.Marker.Text.ToString() + "\n";
}
```

Testing Procedure

1. Using a browser that supports Silverlight, open an HTML page that references a Silverlight application through an object tag. The application plays media that is expected to have text captioning.
2. Check that a text area in the user interface shows captions for the media.

1.2.3 Audio Description or Media Alternative (Pre-Recorded) (Level A) – This Success Criterion aims to make visual information in a synchronized media presentation accessible to those who are blind or visually impaired. There are two approaches highlighted by WCAG-EM for addressing audio descriptions or media alternatives on a web page within a target site.

Technique G69: Providing an Alternative for Time Based Media[21]

Technique G58: Placing a link to the alternative for time-based media immediately next to the non-text content[22]

Example 1: An .MOV Document in an HTML Document

Code on a page called "Olympic_Sports.htm"

[20] https://www.w3.org/WAI/WCAG21/Techniques/silverlight/SL16.html
[21] https://www.w3.org/WAI/WCAG21/Techniques/general/G69.html
[22] https://www.w3.org/WAI/WCAG21/Techniques/general/G58.html

```
 <a name="Olympic_Wrestling"></a>
  <p><a href="http://www.example.com/movies/olympic_wrestling.mov">Olympic Wrestling
movie</a>,
  <a href="http://www.example.com/transcripts/olympic_wrestling_transcript.htm">Olympic
  Wrestling collated Transcript</a></p>
```

Example 2: The link back to the .MOV Document in an HTML Document

Code on the page olympic_wrestling_transcript.htm

```
  <p>Sports announcer 1: This is a great battle tonight between England's "Will Johnson" and
  "Theodore Derringo" from Argentina</p>
  <p>Scenery: There is a mat set out in the middle of the stadium with 500 people in the
  stands...</p>
  <p> ...more dialogue...<p>
  <p> ...more scenery...</p>
  <p> ...etc...</p>
  <p>Sports announcer 2: And that is all for tonight, thank you for joining us tonight where
  Will Johnson is the new Gold Medalist.
  <a href="../movies/Olympic_Sports.htm#Olympic_Wrestling>Return to Movie page</a> </p>
```

Testing Procedure

1. Check for the presence of a link immediately before or after the non-text content.
2. Check that it is a valid link that points directly to the collated document of this particular synchronized media.
3. Check for the availability of a link or back function to get the user back to the original location of the synchronized media content.

Technique SL17: Providing Static Alternative Content for Silverlight Media Playing in a MediaElement[23]

Example

{See Above}

Testing Procedure

{See Above}

Linking to the alternative for time-based media using one of the following techniques:

Technique H53: Using the body of the object element[24]

Example 1: An object includes a long description that describes it

```
<object classid="http://www.example.com/analogclock.py">
  <p>Here is some text that describes the object and its operation.</p>
</object>
```

Example 2: An object includes non-text content with a text alternative

```
<object classid="http://www.example.com/animatedlogo.py">
  <img src="staticlogo.gif" alt="Company Name" />
</object>
```

Example 3: The image object has content that provides a brief description of the function of the image

```
<object data="companylogo.gif" type="image/gif">
  <p>Company Name</p>
</object>
```

Example 4

This example takes advantage of the fact the object elements may be nested to provide for alternative representations of information.

```
<object classid="java:Press.class" width="500" height="500">
  <object data="Pressure.mpeg" type="video/mpeg">
```

[23] https://www.w3.org/WAI/WCAG21/Techniques/silverlight/SL17.html
[24] https://www.w3.org/WAI/WCAG21/Techniques/html/H53.html

```
    <object data="Pressure.gif" type="image/gif">
      As temperature increases, the molecules in the balloon...
    </object>
  </object>
</object>
```

Testing Procedure

1. Check that the body of each object element contains a text alternative for the object.

Technique G78: Providing A Second, User-Selectable, Audio Track That Includes Audio Descriptions[25]

Examples

- A travelogue of the northeast has additional audio description added during the gaps in the dialogue to let listeners who are blind know what the person is talking about at any point in time.
- A video shows a woodpecker carving a nest in a tree. A button within the content allows users to turn the audio description track on or off.
- A lecture has audio description added whenever the instructor says things like "and *this* is the one that is most important." The audio descriptions lets listeners who can not see the video know what "this" is.
- A movie file has two audio tracks, one of which includes audio description. Users can choose either one when listening to the movie by selecting the appropriate track in their media player.

Testing Procedure

1. Check that the ability exists to turn on the audio track that includes audio descriptions. For example, by using a control within the content itself or by selecting a control or preference in the media player or operating system.
2. Listen to the synchronized media
3. Check to see if gaps in dialogue are used to convey important information regarding visual content

Technique G78: Providing a second, user-selectable, audio track that includes audio descriptions **AND** Technique SL1: Accessing Alternate Audio Tracks in Silverlight Media[26]

Examples for G78

{See Above}

Testing Procedure for G78

{See Above}

Examples for SL1: Changing AudioStreamIndex

This example has a UI definition in XAML and interaction logic in C#. In addition to the typical Play/Pause/Stop controls, this interface includes a Play Full-Description Audio button. Activating the button invokes a function that swaps the audio channels and plays an alternative synchronized audio channel that contains a composite full-description audio track.

The following is the basic UI in XAML. This example is deliberately simple and does not include AutomationProperties. Audio streams are identified by an index in a collection.

```
<Grid x:Name="LayoutRoot" Background="White">
     <Grid.ColumnDefinitions>
         <ColumnDefinition Width="*" />
         <ColumnDefinition Width="*" />
         <ColumnDefinition Width="*"/>
     </Grid.ColumnDefinitions>
     <Grid.RowDefinitions>
         <RowDefinition Height="*" />
         <RowDefinition Height="Auto" />
         <RowDefinition Height="20" />
     </Grid.RowDefinitions>
     <MediaElement x:Name="media" Source="/combined.wmv"
```

[25] https://www.w3.org/WAI/WCAG21/Techniques/general/G78.html
[26] https://www.w3.org/WAI/WCAG21/Techniques/silverlight/SL1.html

```
            Width="300" Height="300"
            Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="3"
            AutoPlay="false"
        />
        <Button Click="StopMedia"
     Grid.Column="0" Grid.Row="1" Content="Stop" />
        <Button Click="PauseMedia"
     Grid.Column="1" Grid.Row="1" Content="Pause" />
        <Button Click="PlayMedia"
     Grid.Column="2" Grid.Row="1" Content="Play" />
        <Button Name="AltAudioBtn" Grid.Row="2" HorizontalAlignment="Left" Grid.ColumnSpan="2"
        Click="AltAudioBtn_Click">Play Full-Description Audio</Button>
    </Grid>
The following is the C# logic.
        private void AltAudioBtn_Click(object sender, RoutedEventArgs e)
        {
            if (media.AudioStreamCount > 1)
            {
                if (media.AudioStreamIndex == 1)
                {
                    media.AudioStreamIndex = 0;
                    (sender as Button).Content = "Play full-description audio";
                } else {
                    media.AudioStreamIndex = 1;
                   (sender as Button).Content = "Play default audio";
                }
            }
            else
            {
                (sender as Control).IsEnabled = false;
            }
        }
        private void StopMedia(object sender, RoutedEventArgs e)
        {
            media.Stop();
        }
        private void PauseMedia(object sender, RoutedEventArgs e)
        {
            media.Pause();
        }
        private void PlayMedia(object sender, RoutedEventArgs e)
        {
            media.Play();
        }
```

Testing Procedure for SL1

1. Open the HTML page for a Silverlight application, where that application plays media and the media is expected to support an alternate audio track for the video.
2. Verify that the application user interface presents a control that enables the user to cause the media to play with an alternate audio track.
3. Activate that control. Verify that the audio portion of the media player output as played through the computer's audio system is now playing the alternate audio track

Technique G173: Providing a version of a movie with audio descriptions[27]

Technique SM6: Providing audio description in SMIL 1.0[28]

Example 1: SMIL 1.0 audio description sample for QuickTime player

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns:qt="http://www.apple.com/quicktime/resources/smilextensions"
  xmlns="https://www.w3.org/TR/REC-smil" qt:time-slider="true">
  <head>
    <layout>
      <root-layout background-color="black" height="266" width="320"/>
```

[27] https://www.w3.org/WAI/WCAG21/Techniques/general/G173.html
[28] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM6.html

```
      <region id="videoregion" background-color="black" top="26" left="0"
      height="144" width="320"/>
    </layout>
  </head>
  <body>
    <par>
      <video dur="0:01:20.00" region="videoregion" src="salesdemo.mov"
      alt="Sales Demo"/>
      <audio dur="0:01:20.00" src="salesdemo_ad.mp3"
      alt="Sales Demo Audio Description"/>
    </par>
  </body>
</smil>
```

Example 2: SMIL 1.0 audio description sample for RealTime player

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns="https://www.w3.org/TR/REC-smil">
  <head>
    <layout>
      <root-layout background-color="black" height="266" width="320"/>
      <region id="videoregion" background-color="black" top="26" left="0"
      height="144" width="320"/>
    </layout>
  </head>
  <body>
    <par>
      <video src="salesdemo.mov" region="videoregion" title="Sales Demo"
      alt="Sales Demo"/>
      <audio src="salesdemo_ad.mp3" title="audio description"
      alt="Sales Demo Audio Description"/>
    </par>
  </body>
</smil>
```

Testing Procedure

1. Find method for turning on audio description from content/player (unless it is always played by default)
2. Play file with audio description
3. Check whether audio description is played

Technique SM7: Providing audio description in SMIL 2.0[29]

Example 1: SMIL 2.0 audio description sample for RealMedia player

```
<smil xmlns="https://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout backgroundColor="black" height="266" width="320"/>
      <region id="video" backgroundColor="black" top="26" left="0"
      height="144" width="320"/>
    </layout>
  </head>
  <body>
    <par>
      <video src="salesdemo.mpg" region="video" title="Sales Demo"
      alt="Sales Demo"/>
      <audio src="salesdemo_ad.mp3" begin="33.71s" title="audio description"
      alt="Sales Demo Audio Description"/>
    </par>
  </body>
</smil>
```

The example shows a `<par>` segment containing an `<audio>` and a `<video>` tag. The audio stream is not started immediately.

[29] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM7.html

Testing Procedure

1. Find method for turning on audio description from content/player (unless it is always played by default)
2. Play file with audio description
3. Check whether audio description is played

Technique SL1: Accessing Alternate Audio Tracks in Silverlight Media

Examples

{See Above}

Testing Procedure

{See Above}

Technique G8: Providing a movie with extended audio descriptions[30]

Example 1

An alternate version of an online video of a family escaping from a burning building: there is a continuous dialogue between the husband and wife about where the children are. Meanwhile, in the background, a wall caves in. This is important information in the story because it will block their exit from that part of the building. The video track halts (same frame is repeated) while a narrator gives the details about the wall falling and the video continues.

Example 2

A training film has narrative that runs almost continuously throughout. An alternate version is available for people who have difficulty viewing the video portion. The alternate version freezes the video and provides audio description of key information.

Testing Procedure

1. Open the version of the movie that includes extended audio descriptions.
2. Check that the video halts for extended audio description when there is not enough space to include necessary narration between the natural dialogue.
3. Check that the necessary information is in the audio description.
4. If the alternate version(s) are on a separate page, check for the availability of link(s) to allow the user to get to the other versions.

Technique G203: Using a static text alternative to describe a talking head video[31]

Example 1: A video of a CEO speaking to shareholders

A CEO is speaking to shareholders from their office. The video has a title page at the beginning of the video giving the date. When the speaker begins, there is a strip of text at the bottom of the video saying "Jane Doe, President of XYZ Cooperation". At the end of the video are title credits that say "produced by the Honest TV Productions Ltd." As an alternative, there is a paragraph below the video which is associated with the video file using aria-describedby which says: "July 22, 2011, Jane Doe, President of XYZ cooperation, speaking from her office. Video produced by the Honest TV Productions Ltd."

Testing Procedure

1. Check that there is no important time-based information in the video track
2. Check that the programmatically associated description of the media contains any context of the content that is not contained in the audio track (e.g. speaker identification, credits, context)

1.2.4 Captions (Live) (Level AA) – This Success Criterion's goal is to make real-time presentations accessible to those who are hard of hearing or deaf. The information made available through the audio track has to be included in the captions. Along with the dialogue, captions must also list the speaker's name,

[30] https://www.w3.org/WAI/WCAG21/Techniques/general/G8.html
[31] https://www.w3.org/WAI/WCAG21/Techniques/general/G203.html

sound effects, and other important audio[32]. There are three, two-part techniques that are considered acceptable by WCAG-EM for addressing live captions.

Technique G9: Creating captions for live synchronized media[33] **AND** Technique G93: Providing open (always visible) captions[34]

Examples for G9:
Example 1
A television studio uses a real-time captioning service to create captions for its evening news online.
Example 2
A user watches an online seminar on their mobile device, including captioning provided through the use of Communication Access Real-time Translation (CART). The captions provided also benefit in-person participants who need captioning and can view the information on their own device.

Testing Procedure for G9:
1. Check that a procedure and policy are in place to ensure that captions are delivered in real-time.

Examples for G93:
{See Above}

Testing Procedure for G93:
{See Above}

Technique G9: Creating captions for live synchronized media **AND** Technique G87: Providing closed captions[35]

Examples for G9:
{See Above}

Testing Procedure for G9:
{See Above}

Examples for G87:
{See Above}
Testing Procedure for G87:
{See Above}

Technique G9: Creating captions for live synchronized media

Technique SM11: Providing captions through synchronized text streams in SMIL 1.0[36]

Examples
{See Above}

Testing Procedure
{See Above}

Technique SM12: Providing captions through synchronized text streams in SMIL 2.0[37] **AND** Technique G87: Providing closed captions

[32] https://www.w3.org/WAI/WCAG21/Understanding/captions-live.html
[33] https://www.w3.org/WAI/WCAG21/Techniques/general/G9.html
[34] https://www.w3.org/WAI/WCAG21/Techniques/general/G93.html
[35] https://www.w3.org/WAI/WCAG21/Techniques/general/G87.html
[36] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM11.html
[37] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM12.html

Examples for SM12:
{See Above}
Testing Procedure for SM12:
{See Above}

Examples for G87:
{See Above}
Testing Procedure for G87:
{See Above}

Technique SM11: Providing captions through synchronized text streams in SMIL 1.0[38]

Examples
{See Above}

Testing Procedure
{See Above}

Technique SM12: Providing captions through synchronized text streams in SMIL 2.0

Examples
{See Above}

Testing Procedure
{See Above}

1.2.5 Audio Description (Pre-Recorded) (Level AA)[39] – This Success Criterion aimed to make visual information in a synchronized media presentation accessible to those who are blind or visually impaired. When the video portion is unavailable, the audio description should add the necessary information to the presentation's audio section. When there are pauses in the dialogue, audio description should fill them with details regarding crucial acts, characters, scene changes, and on-screen text that are not spoken or discussed in the main soundtrack.

Technique G78: Providing A Second, User-Selectable, Audio Track That Includes Audio Descriptions

Examples
{See Above}

Testing Procedure
{See Above}

Technique G78: Providing A Second, User-Selectable, Audio Track That Includes Audio Descriptions **AND**
Technique SL1: Accessing Alternate Audio Tracks in Silverlight Media

Examples for G78:
{See Above}

Testing Procedure for G78:
{See Above}

[38] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM11.html
[39] https://www.w3.org/WAI/WCAG21/Understanding/audio-description-prerecorded.html

Examples for SL1:
{See Above}

Testing Procedure for SL1:
{See Above}

Technique G173: Providing a version of a movie with audio descriptions

Technique SM6: Providing audio description in SMIL 1.0

Examples
{See Above}

Testing Procedure
{See Above}

Technique SM7: Providing audio description in SMIL 2.0

Examples
{See Above}

Testing Procedure
{See Above}

Technique G8: Providing a movie with extended audio descriptions[40]

Technique SM1: Adding extended audio description in SMIL 1.0[41]

Example 1: SMIL 1.0 Video with audio descriptions that pause the main media in 4 locations to allow extended audio description

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns:qt="http://www.apple.com/quicktime/resources/smilextensions"
xmlns="https://www.w3.org/TR/REC-smil" qt:time-slider="true">
  <head>
    <layout>
      <root-layout background-color="black" height="266" width="320"/>
      <region id="videoregion" background-color="black" top="26" left="0"
      height="144" width="320"/>
    </layout>
  </head>
  <body>
  <par>
   <seq>
     <par>
       <video src="video.rm" region="videoregion" clip-begin="0s" clip-end="5.4"
       dur="8.7" fill="freeze" alt="videoalt"/>
       <audio src="no1.wav" begin="5.4" alt="audio alt"/>
     </par>
     <par>
       <video src="video.rm" region="videoregion" clip-begin="5.4" clip-end="24.1"
       dur="20.3" fill="freeze" alt="videoalt"/>
       <audio src="no2.wav" begin="18.7" alt="audio alt"/>
     </par>
     <par>
       <video src="video.rm" region="videoregion" clip-begin="24.1" clip-end="29.6"
       dur="7.7" fill="freeze" alt="videoalt"/>
       <audio src="no3.wav" begin="5.5" alt="audio alt"/>
     </par>
     <par>
       <video src="video.rm" region="videoregion" clip-begin="29.6" clip-end="34.5"
```

[40] https://www.w3.org/WAI/WCAG21/Techniques/general/G8.html
[41] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM1.html

```
        dur="5.7" fill="freeze" alt="videoalt"/>
        <audio src="no4.wav" begin="4.9" alt="audio alt"/>
      </par>
      <par>
        <video src="video.rm" region="videoregion" clip-begin="77.4" alt="video alt"/>
      </par>
    </seq>
  </par>
</body>
</smil>
```

Testing Procedure

1. Play file with extended audio descriptions
2. Play file with audio description
3. Check whether video freezes in places and plays extended audio description

Technique SM2: Adding extended audio description in SMIL 2.0[42]

Example 1: Video with extended audio description.

```
<smil xmlns="https://www.w3.org/2001/SMIL20/Language">
<head>
<layout>
<root-layout backgroundColor="black" height="266" width="320"/>
<region id="video" backgroundColor="black" top="26" left="0"
height="144" width="320"/>
</layout>
</head>
<body>
<excl>
<priorityClass peers="pause">
<video src="movie.rm" region="video" title="video" alt="video" />
<audio src="desc1.rm" begin="12.85s" alt="Description 1" />
<audio src="desc2.rm" begin="33.71s" alt="Description 2" />
<audio src="desc3.rm" begin="42.65s" alt="Description 3" />
<audio src="desc4.rm" begin="59.80s" alt="Description 4" />
</priorityClass>
</excl>
</body>
     </smil>
```

Testing Procedure

1. Play file with extended audio description
2. Check whether video freezes in places and plays extended audio description

Technique G203: Using a static text alternative to describe a talking head video

Examples

{See Above}

Testing Procedure

{See Above}

1.3 Adaptable[43] – This section requires that the pages within the target site have the capacity to display content that may be shown in a variety of ways (for instance, a simplified layout) without

[42] https://www.w3.org/WAI/WCAG21/Techniques/smil/SM2.html

[43] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#adaptable

losing any of its structure or information. There are six criteria points that had to be addressed in order to reach level AA conformance.

1.3.1 Info and Relationships (Level A)[44] – This Success Criterion's goal is to make sure that when the presentation format changes, information and connections that are implied by visual or auditory formatting are still available. For instance, the presentation format could be altered when a screen reader reads the information or when a user style sheet takes the place of the author's style sheet. The target website might either possess the technology that would provide the semantic structure to make information and relationships conveyed through presentation programmatically determinable through one of eleven possible features or it does not and would have some type of workaround would have to be built in.

Situation A: The technology provides semantic structure to make information and relationships conveyed through presentation programmatically determinable:

Technique ARIA11: Using ARIA landmarks to identify regions of a page[45]

Example 1: Simple landmarks
The following example shows how landmarks might be added to an HTML4 or XHTML 1.0 document:

```
<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"> ... Ottawa is the capital of Canada ...</div>
<div id="rightsideadvert" role="complementary">....an advertisement here...</div>
<div id="footer" role="contentinfo">(c)The Freedom Company, 123 Freedom Way, Helpville, USA</div>
```

Example 2: Multiple landmarks of the same type and aria-labelledby
The following example shows a best practice of how landmarks might be added to an HTML4 or XHTML 1.0 document in situations where there are more than two of the same type of landmark on the same page. For instance, if a navigation role is used multiple times on a Web page, each instance may have a unique label specified using aria-labelledby:

```
<div id="leftnav" role="navigation" aria-labelledby="leftnavheading">
<h2 id="leftnavheading">Institutional Links</h2>
<ul><li>...a list of links here ...</li> </ul></div>
<div id="rightnav" role="navigation" aria-labelledby="rightnavheading">
<h2 id="rightnavheading">Related topics</h2>
<ul><li>...a list of links here ...</li></ul></div>
```

Example 3: Multiple landmarks of the same type and aria-label
The following example shows a best practice of how landmarks might be added to an HTML4 or XHTML 1.0 document in situations where there are more than two of the same type of landmark on the same page, and there is no existing text on the page that can be referenced as the label.

```
<div id="leftnav" role="navigation" aria-label="Primary">
<ul><li>...a list of links here ...</li></ul> </div>
<div id="rightnav" role="navigation" aria-label="Secondary">
<ul><li>...a list of links here ...</li> </ul></div>
```

Example 4: Search form
The following example shows a search form with a "search" landmark. The search role typically goes on the form element or a div surrounding the form.

```
<form role="search">
<label for="s6">search</label><input id="s6" type="text" size="20">
...
</form>
```

[44] https://www.w3.org/WAI/WCAG21/Understanding/info-and-relationships.html
[45] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA11.html

Testing Procedure

1. Examine each element with a landmark role.
2. Examine whether the landmark role attribute is applied to the section of the page that corresponds with that role. (i.e., the "navigation" role is applied to a navigation section, the "main" role is applied to where the main content begins.)

Technique ARIA12: Using role=heading to identify headings[46]

Example 1: Simple headings

This example demonstrates how to implement simple headings using role="heading" when retrofitting a legacy site where scripts depend on the existing element hierarchy or the level is unknown. For example, web content which is syndicated from various sources may be constructed without knowledge of what the final presentation will be.

```
<div role="heading">Global News items</div>
... a list of global news with editorial comment....

<div role="heading">Local News items</div>
... a list of local news, with editorial comment ...
```

Example 2: Additional heading levels

This example demonstrates how to implement a level 7 heading using `role="heading"` and the `aria-level` attribute. Since HTML only supports headings through level 6, there is no native element to provide these semantics.

```
...
<h5>Fruit Trees</h5>
...
<h6>Apples</h6>
<p>Apples grow on trees in areas known as orchards...</p>
...
<div role="heading" aria-level="7">Jonagold/div>
<p>Jonagold is a cross between the Golden Delicious and Jonathan varieties...</p>
```

Testing Procedure

1. Examine each element with the attribute `role="heading".`
2. Determine whether the content of the element is appropriate as a heading.
3. If the element has an `aria-level` attribute, determine whether the value is the appropriate hierarchical level.

Technique ARIA13: Using aria-labelledby to name regions and landmarks[47]

Example 1: Identify a landmark with on-page text

Below is an example of aria-labelledby used on a complementary Landmark. The region of the document to which the heading pertains could be marked with the aria-labelledby property containing the value of the id for the header.

```
<div role="complementary" aria-labelledby="hdr1">
  <h1 id="hdr1">
    Top News Stories
    ...
  </h1>
</div>
```

Example 2: Identification for Application landmarks

The following code snippet for application landmarks with static prose. If you have a regional landmark of type application and static descriptive text is available, then on the application landmark, include an aria-describedby reference to associate the application and the static text as shown here:

```
<div role="application" aria-labelledby="p123" aria-describedby="info">
```

[46] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA12.html
[47] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA13.html

```
  <h1 id="p123">Calendar<h1>
  <p id="info">
    This calendar shows the game schedule for the Boston Red Sox.
  </p>
<div role="grid">
  ...
</div>
```

Testing Procedure

1. Examine each element with attribute role=region or with a landmark role, where an aria-labelledby attribute is also present.
2. Check that the value of the aria-labelledby attribute is the id of an element on the page.
3. Check that the text of the element with that id accurately labels the section of the page.

Technique ARIA16: Using aria-labelledby to provide a name for user interface controls[48]

Example 1: Labelling a simple text field

The following is an example of aria-labelledby used on a simple text field to provide a label in a situation where there is no text available for a dedicated label but there is other text on the page that can be used to accurately label the control.

```
<input name="searchtxt" type="text" aria-labelledby="searchbtn">
<input name="searchbtn" id="searchbtn" type="submit" value="Search">
```

Example 2: Labelling a slider

Below is an example of `aria-labelledby` used to provide a label for a slider control. In this case the label text is selected from within a longer adjacent text string. Please note that this example is simplified to show only the labeling relationship; authors implementing custom controls also need to ensure that controls meet other success criteria.

```
<p>Please select the <span id="mysldr-lbl">number of days for your trip</span></p>
<div id="mysldr" role="slider" aria-labelledby="mysldr-lbl"></div>
```

Example 3: A label from multiple sources

The following example of `aria-labelledby` with multiple references uses the `label` element. For additional detail on concatenating multiple sources of information into a label with `aria-labelledby`, please view the technique Using ARIA labelledby to concatenate a label from several text nodes.

```
<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>
```

Testing Procedure

For each user interface control element where an aria-labelledby attribute is present:

1. Check that the value of the aria-labelledby attribute is the id of an element or a space separated list of ids on the web page.
2. Check that the text of the referenced element or elements accurately labels the user interface control.

Technique ARIA17: Using grouping roles to identify related form controls[49]

Example 1: Social Security Number

Social security number fields which are 9 digits long and broken up into 3 segments can be grouped using `role="group"`.

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

[48] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA16.html
[49] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA17.html

```
   <span id="ssn1">Social Security#</span>
   <span style="color: #D90D0D;"> * </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" />-
   <input size="4" type="text" aria-required="true" title="Last 4 digits" />
</div>
```

Example 2: Identifying radio groups
This example demonstrates use `role=radiogroup`. Note also that the radio buttons are custom controls with `role=radio`. (But the script to make the span actually work like radio buttons is not included in this example.) One may optionally employ CSS to place a border around groups of such fields to visually reinforce the group relationship. The CSS properties are available below the form.

```
<h3>Set Alerts for your Account</h3>
  <div role="radiogroup" aria-labelledby="alert1">
    <p id="alert1">Send an alert when balance exceeds $ 3,000</p>
    <div>
      <span role="radio" aria-labelledby="a1r1" name="a1radio"></span>
      <span id="a1r1">Yes</span>
    </div>
    <div>
      <span role="radio" aria-labelledby="a1r2" name="a1radio"></span>
      <span id="a1r2">No</span>
    </div>
  </div>
  <div role="radiogroup" aria-labelledby="alert2">
    <p id="alert2">Send an alert when a charge exceeds $ 250</p>
    <div>
      <span role="radio" aria-labelledby="a2r1" name="a2radio"></span>
      <span id="a2r1">Yes</span>
    </div>
    <div>
      <span role="radio" aria-labelledby="a2r2" name="a2radio"></span>
      <span id="a2r2">No</span>
    </div>
  </div>
  <p><input type="submit" value="Continue" id="continue_btn" name="continue_btn" /></p>
```

Related CSS Style Definition to place a border around the group of fields :

```
div[role=radiogroup] {
  border: black thin solid;
}
```

Testing Procedure
For groups of related controls where the individual labels for each control do not provide a sufficient description, and an additional group level description is needed:
1. Check that the group of logically related input or select elements are contained within an element with `role=group`.
2. Check that this group has an accessible name defined using `aria-label` or `aria-labelledby`.

Technique ARIA20: Using the region role to identify a region of the page[50]

Example 1: Region on a news website
A section on the home page of a news website that contains a poll that changes every week is marked up with `role="region"`. The h3 text above the form is referenced as the region's name using `aria-labelledby`.

```
<div role="region" aria-labelledby="pollhead">
<h3 id="pollhead">This week's Poll</h3>
<form method="post" action="#">
  <fieldset>
    <legend>Do you believe the tax code needs to be overhauled?</legend>
    <input type="radio" id="r1" name="poll" />
```

[50] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA20.html

```
    <label for="r1">No, it's fine the way it is</label>
    <input type="radio" id="r2" name="poll" />
    <label for="r2">Yes, the wealthy need to pay more</label>
    <input type="radio" id="r3" name="poll" />
    <label for="r3">Yes, we need to close corporate loopholes</label>
    <input type="radio" id="r4" name="poll" />
    <label for="r4">Changes should be made across the board</label>
  </fieldset>
</form>
<a href="results.php">See Poll Results</a>
</div>
```

Example 2: Identifying a region on a banking site

A user can expand links on a bank website after logging in to see details of term deposit accounts. The details are within a span marked up with region role. The heading for the region has role=heading and is included in the `aria-labelledby` that names the region.

```
<ol>
       <li><a    id="l1"    href="#"    aria-expanded="false"    title="Show    details"    aria-
controls="block1" >John Henry's Account</a><img src="images/panel_expand.gif" alt="" />
                <div  id="block1"  class="nowHidden"  tabindex="-1"  aria-labelledby="l1  cd1"
role="region"><span id="cd1" role="heading" aria-level="3">Certificate of  Deposit:</span>
                <table>
                       <tr><th scope="row">Account:</th> <td>25163522</td></tr>
                       <tr><th scope="row">Start date:</th> <td>February 1, 2014</td></tr>
                       <tr><th scope="row">Maturity date:</th><td>February 1, 2016</td></tr>
                       <tr><th scope="row">Deposit Amount:</th> <td>$ 3,000.00</td></tr>
                       <tr><th scope="row">Maturity Amount:</th> <td>$ 3,072.43</td></tr>
                </table>
                </div>
        </li>
 </ol>
```

Example 3: Identifying a portlet with a generic region

This example shows how a generic region landmark might be added to a weather portlet. There is no existing text on the page that can be referenced as the label, so it is labelled with `aria-label`.

```
<div role="region" aria-label="weather portlet">
       ...
</div>
```

Testing Procedure

For each section marked up with `role="region"`:

1. Examine the content and ensure that it is important enough to have an independent landmark
2. Ensure that a standard landmark role is not appropriate for this content
3. Check that the region has a programmatically determined name

Technique G115: Using semantic elements to markup structure[51] **AND** Technique H49: Using semantic markup to mark emphasized or special text[52]

Examples for G115:

Example 1

A paragraph contains a hyperlink to another page. The hyperlink is marked up using the `<a>` element.

```
       <p> Do you want to try our new tool yourself? A free
               demonstration version is available in our
               <a href="download.html">download section </a></p>
```

Example 2

[51] https://www.w3.org/WAI/WCAG21/Techniques/general/G115.html
[52] https://www.w3.org/WAI/WCAG21/Techniques/html/H49.html

A page about the history of marriage uses a quotation from Jane Austen's novel, Pride and Prejudice, as an example. The reference to the book is marked up using the cite element and the quotation itself is marked up using the blockquote element.

```
<p>Marriage is considered a logical step for a bachelor,
        as can be seen in the first chapter of the novel
        <cite>Pride and Prejudice</cite>:</p>
        <blockquote>
        <p>It is a truth universally acknowledged, that a single man in
        possession of a good fortune, must be in want of a wife.</p>
        <p>However little known the feelings or views of such a man may
        be on his first entering a neighbourhood, this truth is so well
        fixed in the minds of the surrounding families, that he is considered
        the rightful property of some one or other of their daughters.</p>
        </blockquote>
```

Example 3

A car manual explains how to start the engine. The instructions include a warning to make sure the gear is in neutral. The author feels the warning is so important that it should be emphasized so the warning is marked up using the strong element.

```
<h1>How to start the engine</h1>
  <p>Before starting the engine, <strong>make sure the gear
  is in neutral</strong>. Next, turn the key in the ignition.
  The engine should start.</p>
```

Example 4

This example shows how to use the em and strong elements to emphasize text.

```
<p>What she <em>really</em> meant to say is,
  "This is not ok, it is <strong>excellent</strong>!"</p>
```

Example 5: Using highlighting and background color to visually and semantically identify important information.

```
<style type="text/css">
  .vocab {
  background-color:cyan;
  font-style:normal;
  }
  </style>
  ...
  <p>New vocabulary words are emphasized and highlighted
  with a cyan background</p>
  <p>The <em class="vocab">scathing</em> review of the play
  seemed a bit too harsh... </p>
```

Testing Procedure for G115:

1. Check if there are parts of the content that have a semantic function.
2. For each part that has a semantic function, if corresponding semantic markup exists in the technology, check that the content has been marked up using that semantic markup.

Examples for H49:

Example 1: Using the em and strong elements to emphasize text

The `em` and `strong` elements are designed to indicate structural emphasis that may be rendered in a variety of ways (font style changes, speech inflection changes, etc.).

```
... What she <em>really</em> meant to say is, &quot;This is not OK,
it is <strong>excellent</strong>&quot;! ...
```

Example 2: Using the blockquote element to mark up long quotations from another source

This example also demonstrates the use of the cite element to specify a reference.

```
<p>The following is an excerpt from the <cite>The Story Of My Life</cite>
   by Helen Keller:</p>
<blockquote>
  <p>Even in the days before my teacher came, I used to feel along the square stiff
     boxwood hedges, and, guided by the sense of smell, would find the first violets
     and lilies. There, too, after a fit of temper, I went to find comfort and to hide
     my hot face in the cool leaves and grass.</p>
</blockquote>
```

Example 3: Using the q element to mark up a shorter quotation from another source

Quotation marks aren't manually added to the quote because they are added by the user agent.

```
<p>Helen Keller said, <q>Self-pity is our worst enemy and if we yield to it,
   we can never do anything good in the world</q>.</p>
```

Example 4: Using the sup and sub elements to markup superscripts and subscripts
The `sup` and `sub` elements must be used only to markup typographical conventions with specific meanings, not for typographical presentation for presentation's sake.

```
<p>Beth won 1<sup>st</sup> place in the 9<sup>th</sup> grade science competition.</p>
<p>The chemical notation for water is H<sub>2</sub>O.</p>
```

Example 5: Using the code element to mark up code
This example shows use of the code element to provide visual emphasis for a CSS rule:

```
<code>
#trial {
  background-image: url(30daytrial.jpg);
  background-position: left top;
  background-repeat: no-repeat;
  padding-top: 68px;
}
</code>
```

Testing Procedure for H49:
1. Examine the content for information that is conveyed through variations in presentation of text.
2. Check that appropriate semantic markup (such as `em`, `strong`, `cite`, `blockquote`, `sub`, and `sup`) have been used to mark the text that conveys information through variations in text.

Technique G117: Using text to convey information that is conveyed by variations in presentation of text[53]

Example 1: Indicating new content with boldface and a text indicator
The following example shows a list of accessibility standards. WCAG 2.0 is new, so is indicated in bold face. To avoid conveying information solely by presentation, the word "(new)" is included after it as well.

```
<h2>Web Accessibility Guidelines</h2>
  <ul>
  <li><strong>WCAG 2.0 (New)</strong></li>
  <li>WCAG 1.0</li>
  <li>Section 508</li>
  <li>JIS X 8341-3</li>
  ...
  </ul>
```

Example 2: Font variations and explicit statements.
An on-line document has gone through multiple drafts. Insertions are underlined and deletions are struck through. At the end of the draft a "change history" lists all changes made to each draft.

[53] https://www.w3.org/WAI/WCAG21/Techniques/general/G117.html

Example 3: Providing an alternate way to know which words in the text have been identified by using a different font.
An on-line test requires students to write a short summary of a longer document. The summary must contain certain words from the original document. When a sentence in the original document contains a word or phrase that must be used in the summary, the word or phrase is shown in a different font than the rest of the sentence. A separate section also lists all the words and phrases that must be used in the summary.

Testing Procedures
1. Find items where variations in presentation of text are used to convey information.
2. For those items, check to determine if information conveyed visually is also stated explicitly in text.

Technique G140: Separating information and structure from presentation to enable different presentations[54]

Example 1: HTML with CSS
An HTML document uses the structural features of HTML, such as paragraphs, lists, headings, etc., and avoids presentational features such as font changes, layout hints, etc. CSS is used to format the document based on its structural properties. Well-crafted "class" attributes in the HTML extend the semantics of the structural markup if needed to allow more flexible formatting with CSS. Assistive technologies can substitute or extend the CSS to modify presentation, or ignore the CSS and interact directly with the structural encoding.

Example 2: Tagged PDF
A PDF document consists mostly of the content embedded with formatting information. Information about the structure is provided in a separate section of the document using XML-like tags; this is called "tagged PDF". The information in these tags can be used by assistive technologies to perform meaningful structure transformations (e.g., generating a list of sections) or to support interaction with content based on structural characteristics (e.g., jumping to the start of forms).

Testing Procedures
1. Examine the encoding of a document.
2. Check that structural information and functionality are explicitly provided and is logically separated from presentational information.

Technique ARIA24: Semantically identifying a font icon with `role="img"`[55]

Example 1: Star Icon Font used as an indicator (not interactive)
In this example a star icon is used to indicate a favorite. It is not interactive and does not disappear if the user overrides the font family via CSS.

```
Author CSS
/* default class for fonts-face with icons */
.icon { font-family: 'IconFontRoleImg' !important; }

/* specific class for icon */
.icon-star-bg:before { content: "\e982"; }

HTML
- Instead of... -
<p>
   <span class="icon icon-star-bg"></span>
</p>

- Do... -

<p>
   <span class="icon icon-star-bg" role="img" aria-label="Favorite"></span>
</p>
```

[54] https://www.w3.org/WAI/WCAG21/Techniques/general/G140.html
[55] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA24.html

User CSS

```
*:not([role="img"]) { font-family: Verdana, sans-serif !important; }
```

Example 2: Two colored / stacked star Icon Font used as an indicator

In this example a two colored star icon is created by stacking two fonts with different colors on top of each other. This way it's possible to mimic only half the star is filled. It is not interactive and does not disappear if the user overrides the font family via CSS.

Author CSS

```
/* default class for fonts-face with icons */
.icon { font-family: 'IconFontRoleImg' !important; }

/* specific classes for icons */
.icon-star-bg:before    {content: "\e982"; }
.icon-star-half:before  {content: "\e983"; }
```

HTML

```
- Instead of... -
<span class="icon-stacked">
   <span class="icon icon-star-bg grey"></span>
   <span class="icon icon-star-half yellow"></span>
</span>

- Do... -
<span class="icon-stacked" role="img" aria-label="Favorite star half filled">
   <span class="icon icon-star-bg grey" role="img" aria-hidden="true"></span>
   <span class="icon icon-star-half yellow" role="img" aria-hidden="true"></span>
</span>
```

User CSS

```
*:not([role="img"]) { font-family: Verdana, sans-serif !important; }
```

Example 3: Email Icon Font in a link WITHOUT visible text

In this example an email icon is in a link with no visible text. It does not disappear if a user overrides font family. The icon font is identified by assistive technology as a "link image" and the name "Email" (keyboard or mouse).

Author CSS

```
/* default class for fonts-face with icons */
.icon { font-family: 'IconFontRoleImg' !important; }

/* specific class for icon */
.icon-email:before { content: "\e93e"; }
```

HTML

```
- Instead of... -
<a href="email.html">
   <span class="icon icon-email"></span>
</a>

- Do... -
<a href="email.html">
   <span class="icon icon-email" role="img" aria-label="Email"></span>
</a>
```

User CSS

```
*:not([role="img"]) { font-family: Verdana, sans-serif !important; }
```

Example 4: Multiple Icon Fonts as part of another sematic element WITH visible text

This example already has a visible text label in the link to be used as an accessible name, the mail and chevron font icons must stay visible when the font family is changed. This can be done by ensuring the icons are contained in their own element and the attribute aria-hidden="true" is used so the font icons will be ignored by assistive technologies.

Author CSS

```
/* default class for fonts-face with icons */
.icon { font-family: 'IconFontRoleImg' !important; }
```

```
/* specific class for icon */
- See style declarations in HTML examples -

HTML
- Instead of... -
<style>
.icon-double-link:before   { content: "\e93e"; }
.icon-double-link:after    { content: "\e993"; }
</style>

<a href="email.html" class="icon-double-link">
   Email
</a>

- Do... -
<style>
.icon-email:before   { content: "\e93e"; }
.icon-chevron:before {  content: "\e993";  }

.icon-double-link .icon-chevron  { float: right; margin-left: 1.5rem; }
</style>

<a href="email.html" class="icon-double-link">
   <span class="icon icon-email" role="img" aria-hidden="true"></span>
   <span class="icon icon-chevron" role="img" aria-hidden="true"></span>
   Email
</a>

User CSS
*:not([role="img"]) { font-family: Verdana, sans-serif !important; }
```

Testing Procedure
For each font icon check that:
1. The element providing the font icon has `role="img"`.

Situation B: The technology in use does NOT provide the semantic structure to make the information and relationships conveyed through presentation programmatically determinable:

Technique G117: Using text to convey information that is conveyed by variations in presentation of text

Examples
{See Above}

Testing Procedure
{See Above}

Making information and relationships conveyed through presentation programmatically determinable or available in text using the following techniques:

Technique T1: Using standard text formatting conventions for paragraphs[56]

Examples
Two paragraphs. Each starts and ends with a blank line.

```
This is the first sentence in this
paragraph. Paragraphs may be long
or short.

    In this paragraph the first line is
indented. Indented and non-indented
sentences are allowed. White space within
```

[56] https://www.w3.org/WAI/WCAG21/Techniques/text/T1.html

```
the paragraph lines is ignored in
defining paragraphs. Only completely blank
lines are significant.
```

Testing Procedures

For each paragraph:

1. Check that the paragraph is preceded by exactly one blank line, or that the paragraph is the first content in the Web page
2. Check that the paragraph is followed by at least one blank line, or that the paragraph is the last content in the Web page.
3. Check that no paragraph contains any blank lines

Technique T2: Using standard text formatting conventions for lists[57]

Example 1: Unordered list

```
- unordered list item
- unordered list item
- unordered list item
```

Example 2: Numeric ordered list

```
1. Ordered list item
2. Ordered list item
3. Ordered list item
```

Example 3: Roman numeral ordered list

```
i.   Ordered list item
ii.  Ordered list item
iii. Ordered list item
iv.  Ordered list item
```

Example 4: Alphabetic ordered list

```
A) Ordered list item
B) Ordered list item
C) Ordered list item
```

Testing Procedure

For each list in the text content

1. Check that each list item is a paragraph that starts with a label
2. Check that the list contains no lines that are not list items
3. Check that all list items in a list use the same style label
4. Check that the labels in ordered lists are in sequential order
5. Check that the labels in each unordered list are the same

Technique T3: Using standard text formatting conventions for headings[58]

Example

A paragraph is followed by two blank lines, then a heading, then one blank line, then another paragraph:

```
...this is the end of paragraph 1.

The Text of the Heading

This is the beginning of paragraph 2.
```

Testing Procedure

For each heading in the content:

[57] https://www.w3.org/WAI/WCAG21/Techniques/text/T2.html
[58] https://www.w3.org/WAI/WCAG21/Techniques/text/T3.html

1. Check that each heading is preceded by two blank lines
2. Check that each heading is followed by a blank line
3. Check that no heading contains any blank lines

1.3.2 Meaningful Sequence (Level A)[59] – This Success Criterion's goal is to make sure that when the presentation format changes, information and connections that are implied by visual or auditory formatting are still available. For instance, the presentation format can be altered when a screen reader reads the information or when a user style sheet takes the place of the author's style sheet. It's important to note that some platforms might not offer a way to programmatically determine certain sorts of data and relationships. In that situation, a text description of the data and connections is required. Asterisks (*) are used to indicate that a field is necessary. When the page is linearized, the text description should be close to the information it is describing, such as in the parent element or in the adjacent element.

Technique G57: Ordering the content in a meaningful sequence[60]

Examples

A Web page from a museum exhibition contains a navigation bar containing a long list of links. The page also contains an image of one of the pictures from the exhibition, a heading for the picture, and a detailed description of the picture. The links in the navigation bar form a meaningful sequence. The heading, image, and text of the description also form a meaningful sequence. CSS is used to position the elements on the page.

Markup:

```
  <h1>My Museum Page</h1>
  <ul id="nav">
    <li><a href="#">Link 1</a></li>
    ...
    <li><a href="#">Link 10</a></li>
  </ul>
  <div id="description">
    <h2>Mona Lisa</h2>
    <p>
    <img src="img.png" alt="Mona Lisa">
    </p>
    <p>...detailed description of the picture...</p>
  </div>
```

CSS:

```
  ul#nav {
    float: left;
    width: 9em;
    list-style-type: none;
    margin: 0;
    padding: 0.5em;
    color: #fff;
    background-color: #063;
  }
  ul#nav a {
   display: block;
    width: 100%;
    text-decoration: none;
    color: #fff;
    background-color: #063;
  }
  div#description {
    margin-left: 11em;
  }
```

Testing Procedure

[59] https://www.w3.org/WAI/WCAG21/Understanding/info-and-relationships.html
[60] https://www.w3.org/WAI/WCAG21/Techniques/general/G57.html

1. Linearize content using a standard approach for the technology (e.g., removing layout styles or running a linearization tool)
2. Check to see if the order of content yields the same meaning as the original

Marking sequences in the content as meaningful **AND** Technique G57: Ordering the content in a meaningful sequence:

Technique H34: Using a Unicode right-to-left mark (RLM) or left-to-right mark (LRM) to mix text direction inline[61]

Example
This example shows an Arabic phrase in the middle of an English sentence. The exclamation point is part of the Arabic phrase and should appear on its left. Because it is between an Arabic and Latin character and the overall paragraph direction is LTR, the bidirectional algorithm positions the exclamation mark to the right of the Arabic phrase.

The title is "مفتاح معايير الويب!" in Arabic.

Visually-ordered ASCII version (RTL text in uppercase, LTR in lower):
the title is "HCTIWS SDRADNATS BEW!" in arabic.
Inserting a Unicode right-to-left mark in the code immediately after the exclamation mark positions it correctly when you view the displayed text (see below). You can use a character escape or the (invisible) control character to insert the right-to-left mark.

The title is "مفتاح معايير الويب!" in Arabic.

Visually-ordered ASCII version:
the title is "!HCTIWS SDRADNATS BEW" in arabic.

Testing Procedure
1. Examine the source for places where text changes direction.
2. When text changes direction, check whether neutral characters such as spaces or punctuation occur adjacent to text that is rendered in the non-default direction.
3. When check 2 is true and the HTML bidirectional algorithm would produce the wrong placement of the neutral characters, check whether the neutral characters are followed by Unicode right-to-left or left-to-right marks that cause neutral characters to be placed as part of the preceding characters.

Technique H56: Using the `dir` attribute on an inline element to resolve problems with nested directional runs[62]

Example: Defining the text direction of a nested, mixed-direction phrase, in Hebrew and English, to be right-to-left
Because the whole quote is in Hebrew, and therefore runs right to left, the text "W3C" and the comma should appear to the left of (i.e., after) the Hebrew text, like this:

The title is "פעילות הבינאום, W3C" in Hebrew.

Visually-ordered ASCII version (RTL text in uppercase, LTR in lower):
the title is "w3c ,YTIVITCA NOITAZILANOITANRETNI" in Hebrew.
The Unicode bidirection algorithm alone is insufficient to achieve the right result, and leaves the text "W3C" on the right side of the quote:

The title is "פעילות הבינאום, W3C" in Hebrew.

Visually-ordered ASCII version:
the title is "YTIVITCA NOITAZILANOITANRETNI, w3c" in hebrew.

[61] https://www.w3.org/WAI/WCAG21/Techniques/html/H34.html
[62] https://www.w3.org/WAI/WCAG21/Techniques/html/H56.html

The following markup will produce the expected result:

```
<p>The title says "<span lang="he" dir="rtl">פעילות הבינאום, W3C</span>" in Hebrew.</p>
```

Testing Procedure

1. Examine the text direction of text in the document
2. If the text direction is right-to-left, check that for the ancestor element that has a dir attribute, the attribute has the value `"rtl"`
3. If the text direction is left-to-right, check that there is no ancestor element with a dir attribute, or that for the ancestor element that has a dir attribute, the attribute has the value `"ltr"`

Technique C6: Positioning content based on structural markup[63]

Example

In this example structural markup (definition lists) have been applied to the content. CSS has been used to style the content into columnar form. Each class absolutely positions the content into columns and the margins have been set to 0 to override the default behavior of user agents to display HTML definition lists with the DD element indented.

Here is the content to be displayed:

```
 <div class="box">
  <dl>
    <dt class="menu1">Products</dt>
    <dd class="item1">Telephones</dd>
    <dd class="item2">Computers</dd>
    <dd class="item3">Portable MP3 Players</dd>
    <dt class="menu2">Locations</dt>
    <dd class="item4">Idaho</dd>
    <dd class="item5">Wisconsin</dd>
    </dt>
  </dl>
 </div>
```

Here is the CSS which positions and styles the above elements:

```
 .item1 {
   left: 0;
   margin: 0;
   position: absolute;
   top: 7em;
 }
 .item2 {
   left: 0;
   margin: 0;
   position: absolute;
   top: 8em;
 }
 .item3 {
   left: 0;
   margin: 0;
   position: absolute;
   top: 9em;
 }
 .item4 {
   left: 14em;
   margin: 0;
   position: absolute;
   top: 7em;
 }
 .item5 {
   left: 14em;
   margin: 0;
   position: absolute;
   top: 8em;
 }
```

[63] https://www.w3.org/WAI/WCAG21/Techniques/css/C6.html

```
.menu1 {
  background-color: #FFFFFF;
  color: #FF0000;
  font-family: sans-serif;
  font-size: 120%;
  left: 0;
  margin: 0;
  position: absolute;
  top: 3em;
}
.menu2 {
  background-color: #FFFFFF;
  color: #FF0000;
  font-family: sans-serif;
  font-size: 120%;
  left: 10em;
  margin: 0;
  position: absolute;
  top: 3em;
}
#box {
  left: 5em;
  position: absolute;
  top: 5em;
}
```

When style sheets are applied, the data are displayed in two columns of "Products" and "Locations." When the style sheets are not applied, the text appears in a definition list which maintains the structure and reading order.

Testing Procedure
For content which uses CSS for positioning
1. Remove the style information from the document or turn off use of style sheets in the user agent.
2. Check that the structural relations and the meaning of the content are preserved.

Technique C8: Using CSS letter-spacing to control spacing within a word[64]

Example: Separating characters in a word
The following CSS would add the equivalent of a space between each character in a level-2 heading:

```
h2 {   letter-spacing: 1em; }
So for the markup:
<h2>Museum</h2>
the rendered result might look something like:
M u s e u m
```

Testing Procedure
For each word that appears to have non-standard spacing between characters:
1. Check whether the CSS letter-spacing property is used to control spacing.

Technique C27: Making the DOM order match the visual order[65]

Example
- An online newspaper has placed a navigation bar visually in the top left corner of the page directly below its initial logo. In the source code, the navigation elements appear after the elements encoding the logo.

Testing Procedure
1. Visually examine the order of the content in the Web page as it is presented to the end user.
2. Examine the elements in the DOM using a tool that allows you to see the DOM.
3. Ensure that the order of the content in the source code sections match the visual presentation of the content in the Web page. (e.g., for an English language page, the order is from top to bottom and from left to right.) "

[64] https://www.w3.org/WAI/WCAG21/Techniques/css/C8.html
[65] https://www.w3.org/WAI/WCAG21/Techniques/css/C27.html

1.3.3 Sensory Characteristics (Level A)[66] – The purpose of this criterion is to make it possible for a user agent to deliver content in a different way while still maintaining the necessary reading sequence for comprehension. According to the guidelines, it is crucial that at least one logical sequence of the content could be determined automatically. The goal is to identify that when assistive technology reads the material out of sequence, when other style sheets are used, or when other formatting modifications are made, content that does not match this Success Criterion could cause users to become confused or disoriented.

Note: Other techniques may also be sufficient if they meet the success criterion.

Technique G96: Providing textual identification of items that otherwise rely only on sensory information to be understood[67]

Example 1
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.

Example 2
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.

Example 3
The following layout places a button in the lower right corner and indicates it by position. An indication of the text label clarifies which button to use for users who access a linearized version in which the position is not meaningful.

```
<table>
  <tbody>
  <tr>
  <td colspan="2">Push the lower right [Preview] button.</td>
  <td>
  <span style="background: ButtonFace; color: ButtonText; border:
  medium outset ButtonShadow;
  width: 5em; display: block; font-weight: bold; text-align: center;">
  Print</span>
  </td>
  </tr>
  <tr>
  <td>
  <span style="background: ButtonFace; color: ButtonText; border:
  medium outset ButtonShadow;
  width: 5em; display: block; font-weight: bold; text-align: center;">
  Cancel</span>
  </td>
  <td>
  <span style="background: ButtonFace; color: ButtonText; border:
  medium outset ButtonShadow;
  width: 5em; display: block; font-weight: bold; text-align: center;">
  OK</span>
  </td>
  <td>
  <span style="background: ButtonFace; color: ButtonText; border:
  medium outset ButtonShadow;
  width: 5em; display: block; font-weight: bold; text-align: center;">
  Preview</span>
  </td>
  </tr>
```

[66] https://www.w3.org/WAI/WCAG21/Understanding/meaningful-sequence.html
[67] https://www.w3.org/WAI/WCAG21/Techniques/general/G96.html

```
                </tbody>
                </table>
```

Testing Procedure
Find all references in the Web page that mention the shape, size, or position of an object. For each such item:
1. Check that the reference contains additional information that allows the item to be located and identified without any knowledge of its shape, size, or relative position.

1.3.4 Orientation (Level AA)[68] – This Success Criterion's objectives are to ensure that anyone who wants to use the content may do so, even if they are unable to grasp concepts like size, shape, or orientation. Some material depended on information about the position or shape of objects that are not provided by the content's structure (for example, "round button" or "button to the right"). Because of how their assistive technologies are designed, some disabled users might be unable to discern shape or position. Additional information must be provided to clarify instructions that rely on this kind of information in order to meet this success criterion.

Note: Other techniques may also be sufficient if they meet the success criterion.

Using CSS to set the orientation to allow both landscape and portrait.
Use of show/hide controls to allow access to content in different orientations.

1.3.5 Identify Input Purpose (Level AA)[69] – The purpose of this criteria point is to make sure that content displays in the user's desired orientation (landscape or portrait). Some websites and applications restrict the screen's display orientation automatically and expect users to respond by turning their devices to conform. However, this can lead to issues. Some users have their devices fixedly mounted (e.g. on the arm of a power wheelchair). Websites and software should therefore support both orientations by not limiting the orientation. This criterion, which is concentrated on limits of orientation, does not encompass changes in content or functionality caused by changes in display size.

The input field serves a purpose identified in the Input Purposes for User Interface Components section; and the content is implemented using technologies with support for identifying the expected meaning for form input data.

1.3.6 Identify Purpose (Level AA)[70] – In order for user agents to extract and convey this purpose to users using various modalities, it must be possible to programmatically establish the purpose of a form input gathering user information. Filling out forms is made simpler, especially for those with cognitive limitations, by the ability to programmatically indicate the specific type of data required in a certain field. Users may benefit from having fields that collect specific types of information be rendered in an unambiguous, consistent, and possibly customized way for different modalities - either through defaults in their user agent, or with the help of assistive technologies. Appropriate visible labels and instruction can help users understand the purpose of form input fields.

Technique ARIA11: Using ARIA landmarks to identify regions of a page

Examples
{See Above}
Testing Procedures
{See Above}

[68] https://www.w3.org/WAI/WCAG21/Understanding/sensory-characteristics.html
[69] https://www.w3.org/WAI/WCAG21/Understanding/orientation.html
[70] https://www.w3.org/WAI/WCAG21/Understanding/identify-input-purpose.html

Using microdata to markup user interface components (future link)

1.4 Distinguishable[71] – Under WCAG, this section requires that the target site create content that is simpler for users to see and hear, distinguishing foreground from background. There are nine criteria points. The first two grant level A conformance while the remaining seven are for level AA.

1.4.1 Use of Color (Level A)[72] – This Success Criterion's goal is to make sure that all visually impaired users can access information that is communicated through color disparities – through the use of color when each color has a specific meaning. Users with color blindness might not be able to see the color if the information is presented by color differences in an image (or other non-text format). In this instance, giving the color conveys information through another visual medium guarantees that users who are color-blind can still understand the content.

Situation A: If the color of particular words, backgrounds, or other content is used to indicate information:

Technique G14: Ensuring that information conveyed by color differences is also available in text[73]

Example 1: A color-coded schedule
The schedule for sessions at a technology conference is organized into three tracks. Sessions for Track 1 are displayed over a blue background. Sessions in Track 2 are displayed over a yellow background. Sessions in Track 3 are displayed on a green background. After the name of each session is a code identifying the track in text: T1 for Track 1, T2 for Track 2, and T3 for Track 3.

Example 2: A color-coded schedule with icons
The schedule for sessions at a technology conference is organized into three tracks. Next to the title of each session is an icon consisting of a colored circle with a number in the middle showing what track it belongs to: blue circles with the number 1 represent track 1, yellow circles with the number 2 represent Track 2, and green circles with the number 3 represent Track 3. Each icon is associated with a text alternative reading "Track 1," "Track 2," or "Track 3," as appropriate.

Example 3: A form with required fields
A form contains several required fields. The labels for the required fields are displayed in red. In addition, at the end of each label is an asterisk character, *. The instructions for completing the form indicate that "all required fields are displayed in red and marked with an asterisk *", followed by an example.
Note: Asterisks may not be read by all screen readers (in all reading modes) and may be difficult for users with low vision because they are rendered in a smaller size than default text. It is important for authors to include the text indicating that asterisk is used and to consider increasing the size of the asterisk that is presented.

Example 4: A form with a green submit button
An on-line loan application explains that green buttons advance in the process and red buttons cancel the process. A form contains a green button containing the text *Go*. The instructions say, "Press the button labeled *Go* to submit your results and proceed to the next step."

Testing Procedure
For each item where a color difference is used to convey information:
1. Check that the information conveyed is also available in text and that the text is not conditional content.

[71] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#distinguishable
[72] https://www.w3.org/WAI/WCAG21/Understanding/use-of-color.html
[73] https://www.w3.org/WAI/WCAG21/Techniques/general/G14.html

Technique G205: Including a text cue for colored form control labels[74]

Example: Required fields in an HTML form
The instructions for an online form say, "Required fields are shown in red and marked with (required)." The cue "(required)" is included within the label element.

```
<label for="lastname" class="required">Last name (required): </label>
<input id="lastname" type="text" size="25" value=""/>
<style type="text/css">
  .required {
    color:red;
  }
</style>
```

Testing Procedure
For any content where color differences are used to convey information:
1. Check that the same information is available through text or character cues.

Technique G182: Ensuring that additional visual cues are available when text color differences are used to convey information[75]

Examples
- The default formatting for links on a page includes presenting them both in a different color than the other text on the page underlining them to make the links identifiable even without color vision.
- An article comparing the use of similar elements in different markup languages uses colored text to identify the elements from each language. Elements from the first markup language are identified using BLUE, bolded text. Elements from the second are presented as RED, italicized text.
- A news site lists links to the articles appearing on its site. Additional information such as the section the article appears in, the time the article is posted, a related location or an indication that it is accompanied by live video appears in some cases. The links to the articles are in a different color than the additional information but the links are not underlined, and each link is presented in a larger font than the rest of the information so that users who have problems distinguishing between colors can identify the links more easily.
- Short news items sometimes have sentences that are also links to more information. Those sentences are printed in color and use a sans-serif font face while the rest of the paragraph is in black Times-Roman.

Testing Procedure
1. Locate all instances where the color of text is used to convey information.
2. Check that any text where color is used to convey information is also styled or uses a font that makes it visually distinct from other text around it.

Technique G183: Using a contrast ratio of 3:1 with surrounding text and providing additional visual cues on hover for links or controls where color alone is used to identify them[76]

Example 1: Colors that would provide 3:1 contrast with black words and 4.5:1 contrast with a white background
Refer to Links with a 3:1 contrast ratio with surrounding text

Example 2
The hypertext links in a document are medium-light blue (#3366CC) and the regular text is black (#000000). Because the blue text is light enough, it has a contrast of 3.9:1 with the surrounding text and can be identified as being different than the surrounding text by people with all types of color blindness, including those individuals who cannot see color at all.

Testing Procedure

[74] https://www.w3.org/WAI/WCAG21/Techniques/general/G205.html
[75] https://www.w3.org/WAI/WCAG21/Techniques/general/G182.html
[76] https://www.w3.org/WAI/WCAG21/Techniques/general/G183.html

For each instance where color is used to convey information about text:
1. Check that the relative luminance of the color of the text differs from the relative luminance of the surrounding text by a contrast ratio of at least 3:1.
2. Check that hovering over the link causes a visual enhancement (such as an underline, font change, etc.)

Situation B: If color is used within an image to convey information:

Technique G111: Using color and pattern[77]

Example 1
A real estate site provides a bar chart of average housing prices in several regions of the United States. The bar for each region is displayed with a different solid color and a different pattern. The legend uses the same colors and patterns to identify each bar.

Example 2
An on-line map of a transportation system displays each route in a different color. The stops on each route are marked with a distinctive icon such as a diamond, square, or circle to help differentiate each route.

Example 3
A flow chart describes a set of iterative steps to complete a process. It uses dashed, arrowed lines with a green background to point to the next step in the process when the specified condition passes. It uses dotted arrowed lines with a red background to point to the next step in the process when the specified condition fails.

Example 4
The content includes an interactive game. The game pieces for the 4 players are distinguished from one another using both color and pattern.

Testing Procedure
For each image within the Web page that use color differences to convey information:
1. Check that all information that is conveyed using color is also conveyed using patterns that do not rely on color.

Technique G14: Ensuring that information conveyed by color differences is also available in text

Examples
{See Above}
Testing Procedure
{See Above}

1.4.2 Audio Control (Level A)[78] – On the target site, if other audio is playing at the same time as the voice output, people who use screen reading software might have trouble hearing it. When the screen reader's voice output is software-based (as most are) and is controlled by the same volume control as the sound, this problem is made worse. The guidelines make it clear that it is crucial that the user be able to turn off the background noise because of this. Reducing the volume to zero is part of having control over the volume.

Technique G60: Playing a sound that turns off automatically within three seconds[79]

Examples
- Example 1: A Web page opens with a trumpet fanfare and then goes silent
- Example 2: A homepage opens with the chairman saying "Binfor, where quality is our business." then going silent.

[77] https://www.w3.org/WAI/WCAG21/Techniques/general/G111.html
[78] https://www.w3.org/WAI/WCAG21/Understanding/audio-control.html
[79] https://www.w3.org/WAI/WCAG21/Techniques/general/G60.html

- Example 3: A Web page opens with instructions on how to get started: "To begin, press the enter key."
- Example 4: A Web page opens with a warning and then goes silent.

Testing Procedures
1. Load the Web page
2. Check that all sound that plays automatically stops in 3 seconds or less

Technique G170: Providing a control near the beginning of the Web page that turns off sounds that play automatically[80]

Example 1
A Web page contains a time-based media presentation that includes an audio track as well as an animated video describing how to repair a lawnmower engine. The page contains 2 buttons that say "Pause" and "Stop", which give the user control over when and if the time-based media plays.

Example 2
A Web page contains an embedded short film. The page contains a button that says, "Pause the movie", which allows the user to pause the film.

Example 3
A Web page contains a presentation that includes video and audio. The page contains a button that says "Turn off multimedia", which allows the user to stop any video and audio from playing.

Testing Procedure
1. Load a Web page.
2. Check for music or sounds that start automatically.
3. Check that a control that allows the user to turn off the sounds is provided near the beginning of the page.

Technique G171: Playing sounds only on user request[81]

Example 1
A Web page from a grey whale conservation society has a looping background sound of grey whales singing. There are also sounds of water splashing. The sounds do not start automatically. Instead, the Web content provides a link at the top of the page to allow the user to start the sounds manually. The button says, "Turn sounds on." After pressing the "turn sounds on" button, the sounds are heard. The user is then presented with an option to "turn sounds off."

Example 2
A link is provided to a sound file that includes the sounds of the grey whales. The link text says, "Hear the song of the grey whale (mp3)."

Testing Procedure
1. Load a Web page that is known to contain sounds that play for 3 seconds or longer.
2. Check that no sounds play automatically.
3. Check that there is a way for a user to start sounds manually.

1.4.3 Contrast (Minimum) (Level AA)[82] – Hue and saturation have little to no impact on readability, as measured by reading performance, in those without color impairments (Knoblauch et al., 1991). Luminance contrast can be slightly impacted by color deficits. So that those with color vision impairments will also have sufficient contrast between the text and the backdrop, the recommendation's contrast calculation does not take color into account. Excluded is ornamental text that provides no information. For instance, it won't need to meet this requirement if random words are used to form a background and the words can be swapped out or rearranged without changing their meaning.

[80] https://www.w3.org/WAI/WCAG21/Techniques/general/G170.html
[81] https://www.w3.org/WAI/WCAG21/Techniques/general/G171.html
[82] https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html

Situation A: text is less than 18-point if not bold and less than 14-point if bold

Technique G18: Ensuring that a contrast ratio of at least 4.5:1 exists between text (and images of text) and background behind the text[83]

Examples

- A black background is chosen so that light colored letters that match the company logo can be used.
- Text is placed over a picture of the college campus. Since a wide variety of colors and shades appear in the picture, the area behind the text is fogged white so that the picture is very faint, and the maximum darkness is still light enough to maintain a 4.5:1 contrast ratio with the black text written over the picture.

  See also the contrast samples in related resources.

Testing Procedure

(1) Measure the relative luminance of each letter (unless they are all uniform) using the formula:
   - L = 0.2126 * **R** + 0.7152 * **G** + 0.0722 * **B** where **R**, **G** and **B** are defined as:
     - if R sRGB <= 0.04045 then **R** = R sRGB /12.92 else **R** = ((R sRGB +0.055)/1.055) ^ 2.4
     - if G sRGB <= 0.04045 then **G** = G sRGB /12.92 else **G** = ((G sRGB +0.055)/1.055) ^ 2.4
     - if B sRGB <= 0.04045 then **B** = B sRGB /12.92 else **B** = ((B sRGB +0.055)/1.055) ^ 2.4

   and R sRGB, G sRGB, and B sRGB are defined as:
     - R sRGB = R 8bit /255
     - G sRGB = G 8bit /255
     - B sRGB = B 8bit /255

(2) Measure the relative luminance of the background pixels immediately next to the letter using same formula.

(3) Calculate the contrast ratio using the following formula.
   - (L1 + 0.05) / (L2 + 0.05), where
     - L1 is the relative luminance of the lighter of the foreground or background colors, and
     - L2 is the relative luminance of the darker of the foreground or background colors.

(4) Check that the contrast ratio is equal to or greater than 4.5:1

Technique G148: Not specifying background color, not specifying text color, and not using technology features that change those defaults[84]

Examples

The author specifies neither text color nor background and does not use CSS. As a result, the user can set their browser defaults to provide the colors and contrasts that work well for them.

Testing Procedure

1. Look in all places that text color can be specified
2. Check that text color is not specified
3. Look in all areas that background color or image used as a background can be specified
4. Check that no background color or image used as a background is specified

Technique G174: Providing a control with a sufficient contrast ratio that allows users to switch to a presentation that uses sufficient contrast[85]

Examples

- A page with some headlines that do not meet the 3:1 contrast requirements has a high contrast (5:1) link at the top of the page that takes the user to a new version of the page with minimum 4.5:1 contrast on all text and images of text.

[83] https://www.w3.org/WAI/WCAG21/Techniques/general/G18.html
[84] https://www.w3.org/WAI/WCAG21/Techniques/general/G148.html
[85] https://www.w3.org/WAI/WCAG21/Techniques/general/G174.html

- A page uses shaded backgrounds for effect but results in text to background contrast of 4:1. A control at the top of the page says, "high contrast". Clicking on it causes different styles to be used and dropping the background colors to achieve 7:1 contrast.

Testing Procedure

1. Check that a link or control exists on the original page that provides access to the alternate version.
2. Check that the link or control on the original page conforms to all success criteria for the conformance level being tested.
3. Check that the alternate version meets the contrast and all other success criteria for the conformance level being tested.

Situation B: text is at least 18-point if not bold and at least 14-point if bold

Technique G145: Ensuring that a contrast ratio of at least 3:1 exists between text (and images of text) and background behind the text[86]

Examples

- A black background is chosen so that light colored letters that match the company's logo can be used.

  Larger-scale text is placed over a picture of the college campus. Since a wide variety of colors and darknesses appear in the picture, the area behind the text is fogged white so that the picture is very faint, and the maximum darkness is still light enough to maintain a 3:1 contrast ratio with the black text written over the picture.

Testing Procedure

(1) Measure the relative luminance of each letter (unless they are all uniform) using the formula:
- L = 0.2126 * **R** + 0.7152 * **G** + 0.0722 * **B** where **R**, **G** and **B** are defined as:
  - if R sRGB <= 0.04045 then **R** = R sRGB /12.92 else **R** = ((R sRGB +0.055)/1.055) ^ 2.4
  - if G sRGB <= 0.04045 then **G** = G sRGB /12.92 else **G** = ((G sRGB +0.055)/1.055) ^ 2.4
  - if B sRGB <= 0.04045 then **B** = B sRGB /12.92 else **B** = ((B sRGB +0.055)/1.055) ^ 2.4

and R sRGB, G sRGB, and B sRGB are defined as:
- R sRGB = R 8bit /255
- G sRGB = G 8bit /255
- B sRGB = B 8bit /255

(2) Measure the relative luminance of the background pixels immediately next to the letter using same formula.
(3) Calculate the contrast ratio using the following formula.
- (L1 + 0.05) / (L2 + 0.05), where
  - L1 is the relative luminance of the lighter of the foreground or background colors, and
  - L2 is the relative luminance of the darker of the foreground or background colors.

(4) Check that the contrast ratio is equal to or greater than 3:1

Technique G148: Not specifying background color, not specifying text color, and not using technology features that change those defaults

Examples
{See Above}
Testing Procedure
{See Above}

Technique G174: Providing a control with a sufficient contrast ratio that allows users to switch to a presentation that uses sufficient contrast

[86] https://www.w3.org/WAI/WCAG21/Techniques/general/G145.html

Examples
{See Above}
Testing Procedure
{See Above}

1.4.4 Resize Text (Level AA)[87] – The goal of this Success Criterion is to make sure that visually rendered text, including text-based controls, can be successfully scaled so that people with mild visual impairments can read it without the aid of assistive technology like a screen magnifier. Text-based controls are text characters that have been displayed so they can be seen, as opposed to text characters that are still in data form like ASCII. All of the web page's content may benefit from scaling for users, but text is the most important.

Technique G142: Using a technology that has commonly available user agents that support zoom[88]

Examples

- Internet Explorer 7 and Opera 9 provide a zoom function that scales HTML/CSS page content uniformly.
- To allow users to resize text, Adobe Reader provides a magnification tool that scales PDF pages uniformly.

Testing Procedures

1. Display content in a user agent
2. Zoom content to 200%
3. Check whether all content and functionality is available

Ensuring that text containers resize when the text resizes AND using measurements that are relative to other measurements in the content by using one or more of the following techniques:

Technique C28: Specifying the size of text containers using `em` units[89]

Example 1: Em units for sizes for layout container containing text
In this example, `a div` element, with `id` value of "nav_menu", is used to position the navigation menu along the left-hand side of the main content area of the Web page. The navigation menu consists of a list of text links, with id value of "nav_list." The text size for the navigation links and the width of the container are specified in em units.

```
#nav_menu { width: 20em; height: 100em }

#nav_list { font-size: 100%; }
```

Example 2: Em units for text-based form controls
In this example, `input` elements that contain text or accept text input by the user have been given the class name "form1." CSS rules are used to define the font size in percent units and width for these elements in `em` units. This will allow the text within the form control to resize in response to changes in text size settings without being cropped (because the width of the form control itself also resizes according to the font size).

```
input.form1 { font-size: 100%; width: 15em; }
```

Example 3: Em units in dropdown boxes
In this example, select elements have been given the class name "pick." CSS rules are used to define the font size in percent units and width in em units. This will allow the text within the form control to resize in response to changes in text size settings without being cropped.

```
input.pick { font-size: 100%; width: 10em; }
```

Example 4: Em units for non-text-based form controls

[87] https://www.w3.org/WAI/WCAG21/Understanding/resize-text.html
[88] https://www.w3.org/WAI/WCAG21/Techniques/general/G142.html
[89] https://www.w3.org/WAI/WCAG21/Techniques/css/C28.html

In this example, `input` elements that define checkboxes or radio buttons have been given the class name "choose." CSS rules are used to define the width and height for these elements in `em` units. This will allow the form control to resize in response to changes in text size settings.

```
input.choose { width: 1.2em; height: 1.2em; }
```

Testing Procedures

- Identify containers that contain text or allow text input.
- Check the container's width and/or height are specified in `em` units.

Techniques for relative measurements:

Technique C12: Using percent for font sizes[90]

Example: Percent font sizes in CSS

This example defines the font size for the strong element so that its text will always be larger than the surrounding text, in whatever context it is used. Assuming that headings and paragraphs use different font sizes, the emphasized words in this example will each be larger than their surrounding text.

```
strong {font-size: 120%}

...

<h1>Letting the <strong>user</strong> control text size</h1>
<p>Since only the user can know what size text works for him,
it is <strong>very</strong> important to let him configure the text size.
…
```

Testing Procedures

1. Check that the value of the CSS property that defines the font size is a percentage.

Technique C13: Using named font sizes[91]

Example 1: Named font sizes in CSS

This example selects a larger font size for strong elements so that their text will always be larger than the surrounding text, in whatever context they are used. Assuming that headings and paragraphs use different font sizes, the emphasized words in this example will each be larger than their surrounding text.

strong {font-size: larger}

```
...

<h1>Letting the <strong>user</strong> control text size</h1>
<p>Since only the user can know what size text works for him,
it is <strong>very</strong> important to let him configure the text size.
…
```

Testing Procedures

1. Check that the value of the CSS property that defines the font size is one of xx-small, xx-small, x-small, small, medium, large, x-large, xx-large, xsmaller, or larger.

Technique C14: Using `em` units for font sizes[92]

Example 1: Em font sizes in CSS

This example defines the font size for strong element so that its text will always be larger than the surrounding text, in whatever context it is used. Assuming that headings and paragraphs use different font sizes, the strong words in this example will each be larger than their surrounding text.

[90] https://www.w3.org/WAI/WCAG21/Techniques/css/C12.html
[91] https://www.w3.org/WAI/WCAG21/Techniques/css/C13.html
[92] https://www.w3.org/WAI/WCAG21/Techniques/css/C14.html

strong {font-size: 1.6em}

```
...
<h1>Letting the <strong>user</strong> control text size</h1>
<p>Since only the user can know what size text works for him,
it is <strong>very</strong> important to let him configure the text size.  </p>
…
```

Testing Procedures

1. Check that the value of the CSS property that defines the font size is expressed in em units.

Techniques for text container resizing:

Technique SCR34: Calculating size and position in a way that scales with text size[93]

Examples

The Javascript function:

```
function calculatePosition(objElement, strOffset)
{
    var iOffset = 0;

    if (objElement.offsetParent)
    {
        do
        {
            iOffset += objElement[strOffset];
            objElement = objElement.offsetParent;
        } while (objElement);
    }

    return iOffset;
}
```

The following example illustrates using the function above by aligning an object beneath a reference object, the same distance from the left:

```
// Get a reference object
var objReference = document.getElementById('refobject');
// Get the object to be aligned
var objAlign = document.getElementById('lineup');

objAlign.style.position = 'absolute';
objAlign.style.left = calculatePosition(objReference, 'offsetLeft') + 'px';
objAlign.style.top      =      calculatePosition(objReference,      'offsetTop')      +
objReference.offsetHeight + 'px';
```

Testing Procedures

1. Open a page that is designed to adjust container sizes as text size changes.
2. Increase the text size up to 200% using the browser's text size adjustment (not the zoom feature).
3. Examine the text to ensure the text container size is adjusted to accommodate the size of the text.
4. Ensure that no text is "clipped" or has disappeared as a result of the increase in text size.

Technique G146: Using liquid layout

Example: Simple liquid layout in HTML and CSS

The following fairly simple example uses HTML and CSS to create a liquid layout. The three columns adjust their size as text size is adjusted. When the total horizontal width exceeds the available width of the columns, the last column wraps to be positioned below, rather than beside, the previous column. The font size can be increased without either clipping or introducing horizontal scrolling until the longest word no longer fits in a column. This particular example uses percent sizes for the columns and defines them as floating regions using the "float" property.

[93] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR34.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"https://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Example of Basic Liquid Layout</title>
<style type="text/css">
.column
{
border-left: 1px solid green;
padding-left:1%;
float: left;
width: 32%;
}
#footer
{
border-top: 1px solid green;
clear: both;
}
</style>
</head>
<body>
<h1>WCAG Example</h1>
<h2>Text in Three Columns</h2>
<div title="column one" class="column">
<h3>Block 1</h3>
<p> 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.
</p>
</div>
<div title="column two" class="column">
<h3>Block 2</h3>
<p> This is a very simple example of a page layout that adapts as the
text size changes.
</p>
</div>
<div title="column three" class="column">
<h3>Block 3</h3>
<p> For techniques that support more complex page layouts, see the
Resources listed below.
</p>
</div>
<p id="footer">Footer text</p>
</body>
</html>
```

Testing Procedure

1. Display content in a user agent.
2. Increase text size to 200%.
3. Check whether all content and functionality is available with no horizontal scrolling.

Technique G178: Providing controls on the Web page that allow users to incrementally change the size of all text on the page up to 200 percent[94]

Examples

- A newspaper article has two buttons near the top of the page. 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.

[94] https://www.w3.org/WAI/WCAG21/Techniques/general/G178.html

- 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.
- A site includes a link on every page that reads "Change text size." The resulting page includes a series of links that includes options representing the available sizes. The links read, "Smallest font size," "Small font size," "Default font size," "Large font size," etc. Instructions preceding the list direct users to choose a link to change to the desired font size.

Testing Procedure

1. Set the viewport size to 1024px by 768px or larger.
2. Increase the text size and check to see if the text size increased.
3. Check that the text size can be increased to 200% of the original size.
4. Check that after increasing the text size to 200% of the original size, there is no loss of content or functionality (e.g. no parts of the text are clipped, boxes do not overlap, controls are not obscured or separated from their labels, etc.).
5. Decrease the text size to its default value and check to see if it in fact returned to the default size.

Technique G179: Ensuring that there is no loss of content or functionality when the text resizes, and text containers do not change their width[95]

Example 1: A multi-column page layout

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.

Example 2

A newspaper layout with blocks of text in columns. The blocks have a fixed width, but no height set. When the text is resized in the browser, the text wraps and makes the blocks taller.

Testing Procedure

1. Increase text size to 200%.
2. Check whether all content and functionality is available.

1.4.5 Images of Text (Level AA)[96] – The information should be presented as text rather than a picture if the author can do so without sacrificing the material's visual impact. If the author is unable to format the text to achieve the desired result for any reason, the effect won't be reliably displayed on user agents that are widely used or if using a technology to meet this criterion would conflict with meeting other criterion such as 1.4.4, then an image of the text may be used instead. This includes situations, such as type samples, logotypes, branding, etc., where a certain text presentation is necessary to the information being communicated. Images of text may also be used to ensure that the text is anti-aliased on all user agents or to utilize a specific typeface that is either not widely used or that the author does not have permission to share.

Technique C22: Using CSS to control visual presentation of text[97]

Example 1: Using CSS font-family to control the font family for text

The XHTML component:

```
<p>The Javascript method to convert a string to uppercase is <code>toUpperCase()</code>.</p>
```

The CSS component:

```
code { font-family:"Courier New", Courier, monospace }
```

[95] https://www.w3.org/WAI/WCAG21/Techniques/general/G179.html
[96] https://www.w3.org/WAI/WCAG21/Understanding/images-of-text.html
[97] https://www.w3.org/WAI/WCAG21/Techniques/css/C22.html

Example 2: Using CSS text-align to control the placement (alignment) of text
The XHTML component:
```
<p class="right">This text should be to the right of the viewport.</p>
```

The CSS component:
```
.right { text-align: right; }
```

Example 3: Using CSS font-size to control the size of text
The XHTML component:
```
<p>09 <strong class="largersize">March</strong> 2008</p>
```
The CSS component:
```
strong.largersize { font-size: 1.5em; }
```

Example 4: Using CSS color to control the color of text
The style used in this example is not used to convey information, structure or relationships.

The XHTML component:
```
<p>09 <em class="highlight">March</em> 2008</p>
```

The CSS component:
```
.highlight{ color: red; }
```

Example 5: Using CSS font-style to italicize text
The style used in this example is not used to convey information, structure or relationships.

The XHTML component:
```
<p>The article is available in the <a href="http://www.example.com"
class="featuredsite">Endocrinology
Blog</a>.</p>
```

The CSS component:
```
.featuredsite{ font-style:italic; }
```

Example 6: Using CSS font-weight to control the font weight of the text
The style used in this example is not used to convey information, structure or relationships.

The XHTML component:
```
<p>This deal is available <span class="highlight">now!</span></p>
```

The CSS component:
```
.highlight { font-weight:bold; color:#990000; }
```

Example 7: Using CSS text-transform to control the case of text
The style used in this example is not used to convey information, structure or relationships.

The XHTML component:
```
<p>09 <span class="caps">March</span> 2008</p>
```

The CSS component:
```
.caps { text-transform:uppercase; }
```

Example 8: Using CSS line-height to control spacing between lines of text
The CSS line-height property is used to display the line height for the paragraph at twice the height of the font.

The XHTML component:
```
<p>Concern for man and his fate must always form the<br />
chief interest of all technical endeavors. <br />
Never forget this in the  midst of your diagrams and equations. </p>
```

The CSS component:

```
p { line-height:2em; }
```

The CSS line-height property is used to display the line height for the text at less than the height of the font. The second line of text is positioned after the first line of text and visually appears as though the text is part of the first line but dropped a little.

The XHTML component:

```
<h1 class="overlap"><span class="upper">News</span><br />
<span class="byline">today</span></h1>
```

The CSS component:

```
.overlap { line-height:0.2em;  }
.upper { text-transform:uppercase; }
.byline { color:red; font-style:italic; font-weight:bold; padding-left:3em; }
```

Example 9: Using CSS letter-spacing to space text

The CSS letter-spacing property is used to display the letters farther apart in the heading.

The XHTML component:

```
<h1 class="overlap"><span class="upper">News</span><br />
<span class="byline">today</span></h1>
```

The CSS component:

```
.overlap { line-height:0.2em;  }
.upper { text-transform:uppercase; }
.byline { color:red; font-style:italic; font-weight:bold; padding-left:3em; letter-spacing:-0.1em; }
```

The CSS letter-spacing property is used to display the letters closer together in the second line of text.

The XHTML component:

```
<h1 class="upper2">News</h1>
The CSS component:
.upper2 { text-transform:uppercase; letter-spacing:1em; }
```

Example 10: Using CSS background-image to layer text and images

The CSS font-style property is used to display the textual component of a banner and background-image property is used to display a picture behind the text.

The XHTML component:

```
<div id="banner"><span id="bannerstyle1">Welcome</span>
<span id="bannerstyle2">to your local city council</span></div>
```

The CSS component:

```
#banner {
  color:white;
  background-image:url(banner-bg.gif);
  background-repeat:no-repeat;
  background-color:#003399;
  width:29em;
}

#bannerstyle1 {
  text-transform:uppercase;
  font-weight:bold;
  font-size:2.5em;
}

#bannerstyle2 {
  font-style:italic;
  font-weight:bold;
  letter-spacing:-0.1em;
  font-size:1.5em;
}
```

Example 11: Using CSS first-line to control the presentation of the first line of text
The CSS ::first-line pseudo-element is used to display the first line of text in a larger, red font.

The XHTML component:
```
<p class="startline">Once upon a time...<br />
...in a land far, far away...  </p>
```

The CSS component:
```
.startline::first-line { font-size:2em; color:#990000; }
```

Example 12: Using CSS first-letter to control the presentation of the first letter of text
The CSS ::first-letter pseudo-element is used to display the first letter in a larger font size, red and vertically aligned in the middle.

The XHTML component:
```
<p class="startletter">Once upon a time...</p>
```

The CSS component:
```
.startletter::first-letter { font-size:2em; color:#990000; vertical-align:middle; }
```

Testing Procedure
1. Check whether CSS properties are used to control the visual presentation of text

Technique C30: Using CSS to replace text with images of text and providing user interface controls to switch[98]

Examples
A design studio site uses a style switcher to allow users to view two presentations of their home page. For the default version, the heading text is replaced with images of text. A control on the page allows users to switch to a version that presents the headings as text.

The CSS component:
```
...
<div id="Header">
  <h1><span>Pufferfish Design Studio</span></h1>
  <h2><span>Surprising Identity and Design Solutions</span></h2>
  </div>
...
```
The CSS for the presentation that includes images of text follows. Note that the CSS uses positioning to place the contents of the heading elements offscreen so that the text remains available to screen reader users.
```
...
#Header h1 {
        background-image: url(pufferfish-logo.png);
        height: 195px;
        width: 290px;
        background-repeat: no-repeat;
        margin-top: 0;
        position: absolute;
        }
#Header h1 span {
        position: absolute;
         left: -999em;
        }
#Header h2 {
        background-image: url(beauty.png);
        background-repeat: no-repeat;
        height: 234px;
        width: 33px;
        margin-left: 8px;
        position: absolute;
```

[98] https://www.w3.org/WAI/WCAG21/Techniques/css/C30.html

```
        margin-top: 250px;
        }
#Header h2 span {
        position: absolute;
         left: -999em;
        }
```

The CSS for the presentation that does not include images of text.

```
...
#Header h1 {
        font: normal 200%/100% Garamond, "Times New Roman", serif;
        margin-bottom: 0;
        color: #000099;
        background: #ffffff;
        }

#Header h2 {
        font: normal 160%/100% Garamond, "Times New Roman", serif;
        margin-bottom: 0;
        color: #336600;
        background: #ffffff;
        }
```

Testing Procedure

1. Check that the Web page includes a control that allows users to select an alternate presentation.
2. Check that when the control is activated the resulting page includes text (programmatically determined text) wherever images of text had been used.

Technique G140: Separating information and structure from presentation to enable different presentations

Examples
{See Above}

Testing Procedure
{See Above}

1.4.10 Reflow (Level AA)[99] – This Success Criterion is meant to assist those who have impaired eyesight and need to magnify text so they can read it in a single column. Content reflows, or is presented in one column, so that scrolling in more than one direction is not required, when the browser zoom is used to scale content to 400%. Enlarged text with reflow makes reading possible for those with low vision. It is essential. Character perception is made possible through enlargement. Reflow makes tracking possible. Tracking involves moving between the ends of one line and the beginning of the next line of text. The guidelines state that it is crucial to avoid having to scroll in the direction of reading to disclose lines that are obscured by the viewport because doing so considerably increases the amount of effort needed to read. Additionally, it's crucial that content is not obscured off-screen. A vertically scrolling website shouldn't, for instance, have material that is hidden to the side when zoomed into.

Technique C32: Using media queries and grid CSS to reflow columns[100]

Example: Grid layout in HTML and CSS - Medium complexity
The following medium complexity example uses HTML and CSS to create a grid layout. The layout regions adjust their size as the viewport is adjusted. When the total viewport width matches the width defined via media queries, columns wrap to be positioned below, rather than beside each other or vice versa.

[99] https://www.w3.org/WAI/WCAG21/Understanding/reflow.html
[100] https://www.w3.org/WAI/WCAG21/Techniques/css/C32.html

The zoom level can be increased to 400% without requiring scrolling in more than one direction. This particular example uses fr units as a fraction of the free space of the grid container for the grid items by using the "grid-template-columns" property and are laid out in source order.

```
<!DOCTYPE html>
  <html lang="en">
    <head>
      <meta charset="UTF-8">
      <title>CSS: Using media queries and grid CSS to reflow columns</title>
      <style>

      /* Reflow Styling */
      header[role="banner"]       { grid-area: header; }
      main[role="main"]           { grid-area: main; }
      aside[role="complementary"] { grid-area: aside; }
      footer[role="contentinfo"]  { grid-area: footer; }

      .grid,
      .subgrid {
        display: grid;
        grid-template-columns: minmax(0, 1fr);
      }

      .grid {
        grid-template-areas:
        'header'
        'main'
        'aside'
        'footer';
        width: 100%;
      }

      .subgrid {
        width: calc(100% + 2rem);
        margin: 0 -1rem;
      }

      .grid-item,
      .subgrid-item {
        padding: 1rem;
      }

      @media all and (min-width: 576px) {
        .subgrid {
          grid-template-columns: minmax(0, 1fr) minmax(0, 1fr);
          margin-bottom: 1rem;
        }
        .subgrid-item {
          padding-bottom: 0.25rem;
        }
      }

      @media all and (min-width: 992px) {
        .grid {
          grid-template-areas:
            'header header header'
            'main main aside'
            'footer footer footer';
          grid-template-columns: minmax(0, 1fr) minmax(0, 1fr) minmax(0, 1fr);
        }
      }

      </style>

    </head>

    <body class="grid">
```

```
        <header role="banner" class="grid-item">
          ...
        </header>

        <main role="main" class="grid-item">
          ...
          ...
          <div class="subgrid">
            <div class="subgrid-item">
              ...
            </div>
            <div class="subgrid-item">
              ...
            </div>
          </div>
        </main>

        <aside role="complementary" class="grid-item">
          ...
        </aside>

        <footer role="contentinfo" class="grid-item">
          ...
        </footer>

      </body>
    </html>
```

Testing Procedure

1. Display the web page in a user agent capable of 400% zoom and set the viewport dimensions (in CSS pixels) to 1280 wide and 1024 high.
2. Zoom in by 400%.
3. For content read horizontally, check that all content and functionality is available without horizontal scrolling.
4. For content read vertically, check that all content and functionality is available without vertical scrolling.

NOTE: If the browser is not capable of zooming to 400%, you can reduce the width or height of the browser proportionally. For example, at 300% zoom the viewport should be sized to 960px wide.

Technique C31: Using CSS Flexbox to reflow content[101]

Examples: Medium complex flexbox layout in HTML and CSS

The following medium complex example uses HTML and CSS to create a flexbox layout. The layout regions adjust their size as the viewport is adjusted. When the total viewport width matches the width defined via media queries, columns wrap to be positioned below, rather than beside each other or vice versa.

The zoom level can be increased to 400% without requiring scrolling in more than one direction. This particular example uses percent sizes for the flex items by using the "flex-basis" property and are laid out in source order.

```
  <!DOCTYPE html>
  <html lang="en">
    <head>
    <meta charset="UTF-8">
    <title>Using CSS Flexbox for Reflow</title>
    <style>

    /* Reflow Styling */

    .row {
      width: 100%;
      display: flex;
      flex-flow: row wrap;
    }

    .row-nested {
```

[101] https://www.w3.org/WAI/WCAG21/Techniques/css/C31.html

```
      width: calc(100% + 2rem);
      margin: 0 -1rem 1rem -1rem;
    }

    .col {
      padding: 1rem;
      flex: 0 1 100%;
    }

    @media all and (min-width: 576px) {
      .col-panel {
        flex: 0 1 50%;
        padding-bottom: 0.25rem;
      }
    }

    @media all and (min-width: 992px) {
      main[role="main"] {
        flex: 0 1 66.333333%;
      }
      aside[role="complementary"] {
        flex: 0 1 33.333333%;
        margin-top: 0;
      }
    }

    </style>

    </head>

    <body class="row">

      <header role="banner" class="col">
        ...
      </header>

      <main role="main" class="col">
        ...
        ...
        <div class="row row-nested">
          <div class="col col-panel">
            ...
          </div>
          <div class="col col-panel">
            ...
          </div>
        </div>
      </main>

      <aside role="complementary" class="col">
        ...
      </aside>

      <footer role="contentinfo" class="col">
        ...
      </footer>

    </body>
  </html>
```

Testing Procedure

1. Display the web page in a user agent capable of 400% zoom and set the viewport dimensions (in CSS pixels) to 1280 wide and 1024 high.
2. Zoom in by 400%.
3. For content read horizontally, check that all content and functionality is available without horizontal scrolling.
4. For content read vertically, check that all content and functionality is available without vertical scrolling.

NOTE: If the browser is not capable of zooming to 400%, you can reduce the width of the browser proportionally. For example, at 300% zoom the viewport should be sized to 960px wide.

Technique C33: Allowing for Reflow with Long URLs and Strings of Text[102]

Examples: Breaking long URLs

Using the following CSS will cause long URLs to break at appropriate places (hyphens, spaces, etc.) and within words without causing reflow.

List of CSS declarations used and why they are used:

- **overflow-wrap: break-word**: Allows words to be broken and wrapped within words.
- **word-wrap: break-word**: Allows words to be broken and wrapped within. (Microsoft only)

```
a {overflow-wrap: break-word;}
```

IE and Edge only support this declaration when used with the * (wildcard) selector

```
* { word-wrap: break-word;}
```

Testing Procedure

1. Display the web page in a user agent capable of 400% zoom and set the viewport dimensions (in CSS pixels) to 1280 wide and 1024 high.
2. Zoom in by 400%.
3. For content read horizontally, check that all content and functionality is available without horizontal scrolling.
4. For content read vertically, check that all content and functionality is available without vertical scrolling.

NOTE: If the browser is not capable of zooming to 400%, you can reduce the width of the browser proportionally. For example, at 300% zoom the viewport should be sized to 960px wide.

Technique C38: Using CSS `width, max-width and flexbox to fit labels` and `inputs`[103]

Examples: Fitting labels, inputs and flexbox layout with HTML and CSS.

The following example uses HTML and CSS to fit labels and inputs within various width containers, including the viewport. The layout regions adjust their size as the viewport is adjusted. The labels and inputs subsequently adjust their size to fit within the layout region containers.

The zoom level can be increased to 400% without requiring horizontal scrolling. This particular example uses a percent size for the width and max-width for the labels and inputs. The max-width is applied in order to fix elements spilling out of the grid in a cross-browser way, as replaced elements such as the select have intrinsic sizing.

```
<style>

/* Fitting Inputs Styling */

.form-group {
  display: flex;
  flex-flow: row wrap;
  margin: 0 -1rem 1rem -1rem;
}

[class*="form-col"] {
  flex: 0 1 100%;
  padding: 0 1rem;
}

@media (min-width: 576px) {
  .form-col-4 {
    flex: 0 0 33.33333%;
    max-width: 33.33333%;
  }

  .form-col-8 {
    flex: 0 0 66.66667%;
```

[102] https://www.w3.org/WAI/WCAG21/Techniques/css/C33.html
[103] https://www.w3.org/WAI/WCAG21/Techniques/css/C38.html

```
    max-width: 66.66667%;
  }

  .offset-form-col-4 {
    margin-left: 33.33333%;
  }

}

input {
  display: block;
  width: 100%;
}

label,
select {
  display: block;
  width: 100%;
  max-width: 100%;
}

</style>

<div class="form-group">
  <div class="form-col-4">
    <label for="fname">First Name</label>
  </div>
  <div class="form-col-8">
    <input type="text" id="fname" autocomplete="given-name">
  </div>
</div>

<div class="form-group">
  <div class="form-col-4">
    <label for="lname">Last Name</label>
  </div>
  <div class="form-col-8">
    <input type="text" id="lname" autocomplete="family-name">
  </div>
</div>

<div class="form-group">
  <div class="form-col-4">
   <label for="favorite-fruit">Favorite fruit</label>
  </div>
  <div id="favorite-fruit" class="form-col-8">
    <select>
      <option>Banana</option>
      <option>Pineapple</option>
      <option>Strawberry</option>
    </select>
  </div>
</div>

<div class="form-group">
  <div class="offset-form-col-4 form-col-8">
    <button>Submit</button>
  </div>
</div>
```

Testing Procedure

1. Display the web page in a user agent capable of 400% zoom and set the viewport dimensions (in CSS pixels) to 1280 wide and 1024 high.
2. Zoom in by 400%.
3. For vertically scrolling content, all labels and inputs fit in their available space without horizontal scrolling.

NOTE: If the browser is not capable of zooming to 400%, you can reduce the width of the browser proportionally. For example, at 300% zoom the viewport should be sized to 960px wide.

Technique SCR34: Calculating size and position in a way that scales with text size

Examples
{See Above}
Testing Procedure
{See Above}

Technique G206: Providing options within the content to switch to a layout that does not require the user to scroll horizontally to read a line of text[104]

Example 1
A real estate company has an online annual report that has an identical layout to that of their print version, and as such, requires horizontal scrolling to read a line of text. A control is on the page that switches the stylesheet and provides a layout that does not require horizontal scrolling.

Example 2
A financial spreadsheet is online. It includes text explaining changes in the housing market in January. Off-screen to the right, there is a column with an explanation of changes to the market in September. The user can horizontally scroll to the September area and read each line of text without any further scrolling when the window size is maximized.

Testing Procedure
1. Open the content that requires horizontal scrolling on a full screen window.
2. Check that there is an option within the content to switch to a layout that does not require the user to scroll horizontally to read a line of text.
3. Activate the option.
4. Check to make sure that horizontal scrolling is not required to read any line of text.

1.4.11 Non-Text Contrast (Level AA)[105] – According to WGAC-EM, low contrast controls are more challenging to see and may go entirely unnoticed by those who have vision impairments. Like this, if a graphic is required to comprehend the information or functioning of a webpage, it should be readable without the use of contrast-enhancing assistive technology for individuals with low vision or other disabilities. Although it is not a requirement for this success criterion that controls have a visual boundary showing the hit region, if the control's visual indicator is the sole method to recognize it, it must have enough contrast. If there is no obvious sign of the hit region and text (or an icon) within a button or placeholder text inside a text entry is visible, the success criterion is met. There is no additional contrast requirement beyond the text contrast specified in 1.4.3 Contrast (Minimum) if a button with text additionally has a colored border because the border does not serve as the lone indicator. Notably, it is advised to mark the limits of controls for those with cognitive disorders to facilitate control detection and, as a result, activity completion.

User Interface Component contrast:

Technique G195: Using an author-supplied, visible focus indicator[106]

Example 1: Links
A Web page has a dark background color and light text and links. When focus lands on a link, the link is outlined with a bright yellow line, 3 pixels wide.

Example 2: Form Elements

[104] https://www.w3.org/WAI/WCAG21/Techniques/general/G206.html
[105] https://www.w3.org/WAI/WCAG21/Understanding/non-text-contrast.html
[106] https://www.w3.org/WAI/WCAG21/Techniques/general/G195.html

A Web page includes a form inside a table. The borders of both the table and the form elements are thin, black lines. When focus lands on a form element, the element is outlined with a 5 pixel red line that is partially transparent. The red is equivalent to a hex color of #FF3838, providing a 3.6:1 contrast ratio with the white background.

Example 3: Menus
A Web page includes an interactive menu with sub-menus. A user can move focus in the menu using the arrow keys. As focus moves, the currently focused menu item changes its background to a different color, which has a 3:1 contrast ratio with the surrounding items and a 4.5:1 contrast ratio with its own text.

Testing Procedures
For each user interface component on the page that should receive keyboard focus:
1. Navigate to the component and check that it has a visible focus indicator.
2. Check that the focus indicator area is at least the size of a 1 CSS px border around the component.
3. If the focus indicator area is not at least equal to the area of a 1 CSS pixel border, check that it has an area of at least 4 CSS pixels along the shortest side of the component.
4. Check that the change of contrast of the indicator between focused and unfocused states has a ratio of 3:1 or more for the minimum focus indicator area.
5. If the focus indicator does not have 3:1 contrast ratio with its adjacent colors, check that it is at least 2px thick.

Graphics with sufficient contrast:

Technique G207: Ensuring that a contrast ratio of 3:1 is provided for icons[107]

Example 1: Solid icon color against the background
A solid icon such as a telephone symbol uses orange on a white background. The color orange (#E3660E) is tested against the white background (#FFFFFF) and it has a contrast ratio of 3.4:1.

Example 2: Solid icon color against a custom background
A solid icon such as a telephone symbol used within an orange background. The orange and white colors are the same as in example 1, in this case the contrast against the white background is not relevant, the white icon within the orange background is what provides the information in the icon and as a result needs to meet the contrast requirement.

Example 3: Solid icon with a gradient background
A solid icon such as a telephone symbol using a dark blue icon on a white-to-blue gradient background. The first test of the icon should be against the darkest (least contrasting) background that is adjacent to the icon color. If that is at least 3:1, it passes the success criterion.

Example 4: Solid icon with gradient background overlapping in contrast
A solid icon on a gradient background can overlap in contrast if the graphic is still understandable where it does not have contrast against all of the background. If you find the part of the gradient where it does not meet a 3:1 ratio with the graphic and treat that part as if it is removed, does the icon still convey the appropriate meaning?
A method of visualizing this is to remove the non-contrasting area and check that you can still understand the icon. If so, it is sufficient. The images below shows an icon on a gradient background, and a second version where it removes the area of the icon that does not meet the 3:1 contrast ratio. It is still recognizable as a phone icon, so passes the success criterion.

Testing Procedures
For each graphical object required for understanding use a color contrast tool to:
1. Determine the foreground color of the graphical object.
2. Determine the adjacent background color. If the background color is a gradient or pattern, identify the color with the least contrast to the foreground color.
3. Check that the contrast ratio is equal to or greater than 3:1.
4. If part of the background area does not meet 3:1 with the foreground, assume that parts of the icon adjacent to the area or areas are not visible.
5. Check that the icon is still recognizable without any area of insufficient contrast.

[107] https://www.w3.org/WAI/WCAG21/Techniques/general/G207.html

Technique G209: Provide sufficient contrast at the boundaries between adjoining colors[108]

Testing Procedures
For each graphical object required for understanding use a color contrast tool to:
1. Measure the contrast ratio of each color compared to the adjacent color(s) or border (if present).
2. Check that the contrast ratio is at least 3:1 for each adjacent color or border (if present).

Text in or over graphics:

Technique G18: Ensuring that a contrast ratio of at least 4.5:1 exists between text (and images of text) and background behind the text

Examples
{See Above}
Testing Procedure
{See Above}

Technique G145: Ensuring that a contrast ratio of at least 3:1 exists between text (and images of text) and background behind the text

Examples
{See Above}
Testing Procedure
{See Above}

Technique G174: Providing a control with a sufficient contrast ratio that allows users to switch to a presentation that uses sufficient contrast

Examples
{See Above}
Testing Procedure
{See Above}

1.4.12 Text Spacing (Level AA)[109] – This Success Criteria is concerned with how well information can adjust to wider spaces between lines, words, letters, and paragraphs. Any of these in combination may help a user read text efficiently. The possibility that users can overrule the author's spacing preferences is also considerably increased by making sure that the material adapts effectively when users do so. For instance, to read content clearly, a user might need to switch to a broader font family than the author has chosen.

Technique C36: Allowing for text spacing override[110]

Examples: A paragraph expands vertically within container

```
/* CSS: No height property is set.*/

<!-- HTML -->
 <div class="card">
  <img src="image.png" alt="proper alt text">
  <h3>Heading</h3>
  <p class="lede">Long lede paragraph…</p>
 </div>
```

[108] https://www.w3.org/WAI/WCAG21/Techniques/general/G209.html
[109] https://www.w3.org/WAI/WCAG21/Understanding/text-spacing.html
[110] https://www.w3.org/WAI/WCAG21/Techniques/css/C36.html

None of the paragraphs on this page have a height specified, so all are effectively using this technique.

Testing Procedure
For elements which contain text that is intended to wrap:
1. Set zoom level to 100%.
2. Use a tool or another mechanism to apply the text spacing metrics (line height, and paragraph, letter, and word spacing), such as the Text Spacing Bookmarklet or a user-style browser plugin.
3. Check that all content and functionality is available e.g., text in containers is not truncated and does not overlap other content.

Technique C35: Allowing for text spacing without wrapping[111]

Example 1: A box sized with space to allow for expansion
The containers are sized to a value greater than the default width of the text.

```
/* Links are less than 8ex wide, so 10ex width of each li allows for expanded letter and word width*/
 nav li { width: 10em; }

<!-- HTML -->
 <nav>
  <ul>
   <li><a href="/">Home</a></li>
   <li><a href="/contact/">Contact</a></li>
  <ul>
 </nav>
```

If the navigation element used fix-width containers of the same size, the width would need to allow for text 20% larger than the longest word.

Example 2: A box which expands with the text size

```
/* CSS containers are given a display of inline-block. No negative margins set. */
 nav li { display: inline-block; }

<!-- HTML -->
 <nav>
  <ul>
   <li><a href="/">Home</a></li>
   <li><a href="/contact/">Contact</a></li>
  <ul>
 </nav>
```

In the case of variable-width text containers for each item, the parent item may need to allow for wrapping of the items.

Testing Procedure
For elements which contain text that is not intended to wrap:
1. Set zoom level to 100%.
2. Use a tool or another mechanism to apply the text spacing metrics (line height, and paragraph, letter, and word spacing), such as the Text Spacing Bookmarklet or a user-style browser plugin.
3. Check that all content and functionality is available e.g., text in containers is not truncated and does not overlap other content.

1.4.13 Content on Hover or Focus (Level AA)[112] – This requirement is there to make sure that the extra material won't make it difficult to see or use the page's original content. The amount of the page that can be seen in the viewport when it is magnified can be considerably diminished. Frequently, mouse users will pan the magnified viewport to show another area of the screen. It is challenging for a user to pan without re-triggering the new content because practically the entire area of the page displayed in this constrained viewport may trigger it. There is a fix by using the keyboard to remove the extra text.

[111] https://www.w3.org/WAI/WCAG21/Techniques/css/C35.html
[112] https://www.w3.org/WAI/WCAG21/Understanding/content-on-hover-or-focus.html

- @@ ARIA: Using role="tooltip"
- @@ CSS: Using hover and focus pseudo classes

Section 2 – Operable

2.1 Keyboard Accessible[113] – This required that all functionality within the target site is available from a keyboard. There are three criteria points in this section to reach level A conformance. For this section, there is not an option for level AA conformance – only A or AAA.

2.1.1 Keyboard (Level A)[114] – The purpose of this Success Criterion is to ensure that information could be accessed and used with a keyboard or keyboard interface whenever and wherever it is feasible to do so (so an alternate keyboard can be used). When content can be controlled by a keyboard or an alternative keyboard, it can be utilized by people who require the use of alternative keyboards or input devices that emulate keyboards, as well as by people who are blind and, as a result, are unable to control devices such as mice that require eye-hand coordination. Examples of keyboard emulators include speech input software, sip-and-puff, on-screen keyboards, scanning software, and a variety of assistive devices and alternative keyboards. Other types of keyboards include alternative keyboards. People who have trouble seeing clearly may have a difficult time following the movement of a pointer, and these individuals may be unable to use software unless it can be controlled via the keyboard. Certain situations qualify as "specific timings for individual keystrokes," such as those in which a user is required to quickly repeat or carry out a series of keystrokes, or in which a user must hold down a key for an extended period of time before the keystroke is recognized.

Technique G202: Ensuring keyboard control for all functionality[115]

Examples

- 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.

Testing Procedure

1. Identify all functionality on the content.
2. Check that all functionality can be accessed using only the keyboard or keyboard interface.

Ensuring keyboard control by using one of the following techniques.

Technique H91: Using HTML form controls and links[116]

Example 1: Links

[113] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#keyboard-accessible

[114] https://www.w3.org/WAI/WCAG21/Understanding/keyboard.html

[115] https://www.w3.org/WAI/WCAG21/Techniques/general/G202.html

[116] https://www.w3.org/WAI/WCAG21/Techniques/html/H91.html

User agents provide mechanisms to navigate to and select links. In each of the following examples, the role is "link" from the <a href>. Note that <a name> does not provide a role of "link". The value is the URI in the 'href' attribute.

Example 1a
In example 1a, the name is the text inside the link, in this case "Example Site".
```
<a href="www.example.com">Example Site</a>
```

Example 1b
In example 1b of an image inside a link, the 'alt' attribute for the image provides the name. Some tools for viewing APIs, such as Microsoft Inspect Objects, will not surface this, but AT does.
```
<a href="www.example.com"><img src="example_logo.gif" alt="Example"></a>
```
Example 1c
In example 1c, some assistive technology will not automatically insert a space character when concatenating the image's alt text and the text of the link. If the text should not be concatenated without a space, it is safest to insert a space between the image and the adjacent word so that words will not run together.
```
<a href="www.example.com"><img src="example_logo.gif" alt="Example"> Text</a>
```

Example 2: Buttons
There are several ways to create a button in HTML, and they all map to the "push button" role.

Example 2a
In example 2a, the text is contained in the button element, in this case "save", as the name. There is no value.
```
<button>Save</button>
```

Example 2b
Example 2b uses the 'value' attribute, in this case "Save", "Submit", or "Reset" as the name.
```
<input type="button" value="Save" />
<input type="submit" value="Submit" />
<input type="reset" value="Reset" />
```

Example 2c
Example 2c uses the 'alt' attribute, in this case "save", as the name.
```
<input type="image" src="save.gif" alt="save" />
```

Example 2d
In example 2d, there is no 'alt' attribute so the 'title' attribute, in this case "save", is used as the name.
```
<input type="image" src="save.gif" title="save" />
```

Example 2e
Example 2e clarifies how the user agent determines the name if the author specifies both the 'alt' and 'title' attributes of the input element. In this case, the user agent uses the 'alt' attribute ("Save") and ignores the 'title' attribute.
```
<input type="image" src="save.gif" alt="save" title="save the file" />
```

Example 3

Example 3a
In example 3a, the input field has a role of "editable text". The label element is associated to the input element via the 'for' attribute which references the 'id' attribute of the input> element. The name comes from the label element, in this case, "Type of fruit". Its value comes from its value attribute, in this case "bananas".
```
<label for="text_1">Type of fruit</label>
<input id="text_1" type="text" value="bananas">
```

Example 3b
In example 3b, the input field has the same role as example 3a, but the value is the empty string and the field gets its name from the 'title' attribute.
```
<input id="text_1" type="text" title="Type of fruit">
```

Example 4: Checkbox
Example 4 has a role of "checkbox", from the 'type' attribute of the input element. The label element is associated with the input element via the 'for' attribute which refers to the 'id' attribute of the input element. The name comes from the label element, in this case "cheese". Its state can be "checked" or "unchecked" and comes from the 'checked' attribute. The state can be changed by the user's interaction with the control.

```
<label for="cb_1">Cheese</label>
<input id="cb_1" type="checkbox" checked="checked">
```

Example 5: Radio Buttons
Example 5 has a role of "radio button" from the 'type' attribute on the input element. Its name comes from the label element. The state can be "checked" or "unchecked" and comes from the 'checked' attribute. The state can be changed by the user.

```
<input type="radio" name="color" id="r1" checked="checked"/><label for="r1">Red</label>
<input type="radio" name="color" id="r2" /><label for="r2">Blue</label>
<input type="radio" name="color" id="r3" /><label for="r3">Green</label>
```

Example 6

Example 6a
Example 6a has a role of "list box" from the select element. Its name is "Numbers" from the label element. Forgetting to give a name to the select is a common error. The value is the option element that has the 'selected' attribute present (with a value of "selected" in XHTML). In this case, the default value is "Two".

```
<label for="s1">Numbers</label>
<select id="s1" size="1">
 <option>One</option>
 <option selected="selected">Two</option>
 <option>Three</option>
</select>
```

Example 6b
Example 6b has the same name, role, and value as the above, but sets the name with the 'title' attribute on the select element. This technique can be used when a visible label is not desirable.

```
<select id="s1" title="Numbers" size="1">
 <option>One</option>
 <option selected="selected">Two</option>
 <option>Three</option>
</select>
```

Example 7: Textarea

Example 7a
Example 7a has a role of "editable text" from the textarea element. The name is "Type your speech here" from the label element. The value is the content inside the textarea element, in this case "Four score and seven years ago".

```
<label for="ta_1">Type your speech here</label>
<textarea id="ta_1" >Four score and seven years ago</textarea>
```

Example 7b
Example 7b has the same role, the name is set using the 'title' attribute, and the value is the empty string.

```
<textarea id="ta_1" title="Type your speech here" >Four score and seven years ago</textarea>
```

Example 8
Radio Fieldset
The radio fieldset in example 8 has a role of "grouping". The name comes from the legend element.

```
<fieldset>
  <legend>Choose a Color:</legend>
     <input id="red" type="radio" name="color" value="red" /><label for="red">Red</label><br
/>
     <input      id="blue"      type="radio"      name="color"      value="blue"      /><label
for="blue">Blue</label><br />
     <input     id="green"     type="radio"     name="color"     value="green"     /><label
for="green">Green</label>
```

```
</fieldset>
```

Technique G90: Providing keyboard-triggered event handlers[117]

Examples

- **Example 1: A drag and drop feature** A photo application includes a "drag" and "drop" feature to allow users to re-order photographs in an on-line album for presentation as a slide show. It also includes a feature that allows users to select a photo and 'cut' and 'paste' the items into the list at the appropriate point using only the keyboard.
- **Example 2: A reorder feature** A Web application that allows users to create surveys by dragging questions into position includes a list of the questions followed by a text field that allows users to re-order questions as needed by entering the desired question number.

Testing Procedure

1. check that all functionality can be accessed using only the keyboard

Technique SCR20: Using both keyboard and other device-specific functions[118]

Example 1

In this example of an image link, the image is changed when the user positions the pointer over the image. To provide keyboard users with a similar experience, the image is also changed when the user tabs to it.

```
<a href="menu.php" onmouseover="swapImageOn('menu')" onfocus="swapImageOn('menu')"
onmouseout="swapImageOff('menu')" onblur="swapImageOff('menu')">
  <img id="menu" src="menu_off.gif" alt="Menu" />
</a>
```

Example 2

This example shows a custom link control where both the mouse and the keyboard can be used to activate the function. The mouse onclick event is duplicated by an appropriate keyboard onkeypress event. The tabindex attribute ensures that the keyboard will have a tab stop on the span element. Note that in this example, the nextPage() function should check that the key pressed is Enter, otherwise it will respond to all keyboard actions while the span has focus, which is not the desired behavior.

```
<span onclick="nextPage();" onkeypress="nextPage();" role="link" tabindex="0">
  <img alt="Go to next page" src="arrow.gif">
</span>
```

Testing Procedure

1. Find all interactive functionality
2. Check that all interactive functionality can be accessed using the keyboard alone

Technique SCR35: Making actions keyboard accessible by using the onclick event of anchors and buttons[119]

Example 1

Link that runs a script and has no fallback for non-scripted browsers. This approach should only be used when script is relied upon as an Accessibility Supported Technology.
Even though we do not want to navigate from this link, we must use the href attribute on the a element in order to make this a true link and get the proper eventing. In this case, we're using "#" as the link target, but you could use anything. This link will never be navigated.
The "return false;" at the end of the doStuff() event handling function tells the browser not to navigate to the URI. Without it, the page would refresh after the script ran.

```
<script>
function doStuff()
```

[117] https://www.w3.org/WAI/WCAG21/Techniques/general/G90.html
[118] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR20.html
[119] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR35.html

```
 {
  //do stuff
    return false;
  }
</script>
<a href="#" onclick="return doStuff();">do stuff</a>
```

Example 2
Link that runs script, but navigates to another page when script is not available. This approach can be used to create sites that don't rely on script, if and only if the navigation target provides the same functionality as the script. This example is identical to the example 1, except that its href is now set to a real page, dostuff.htm. Dostuff.htm must provide the same functionality as the script.The "return false;" at the end of the doStuff() event handling function tells the browser not to navigate to the URI. Without it, the browser would navigate to dostuff.htm after the script ran.

```
<script>
function doStuff()
 {
  //do stuff
  return false;
 }
</script>
<a href="dostuff.htm" onclick="return doStuff();">do stuff</a>
```

A working example of this code is available. Refer to Creating Action Links using JavaScript.

Example 3
Button that runs a script and falls back to a form post for users without script. This approach can be used by sites that do not rely on script, if and only if the form post provides the same functionality as the script. The onsubmit="return false;" prevents the form from submitting.

```
<script>
  function doStuff()
 {
     //do stuff
 }
</script>
<form action="doStuff.aspx" onsubmit="return false;">
 <input type="submit" value="Do Stuff" onclick="doStuff();" />
</form>
```

A working example of this code is available. Refer to Creating Action Buttons using JavaScript.

Example 4
Button that runs a script, implemented with input `type="image".` Note that an alt attribute must be added to the `input` to provide a text equivalent for the image. This approach should only be used when script is relied upon.

```
<script>
  function doStuff()
  {
     //do stuff
   return false;
  }
</script>
<input  type="image"  src="stuff.gif"  alt="Do stuff"  onclick="return doStuff();" />
```

Example 5
Button that runs a script, implemented with input `type="submit"`, `input type="reset"` or `input type="button"`. This approach should only be used when script is relied upon.

```
<input type="submit" onclick="return doStuff();" value="Do Stuff" />
```

Example 6

Button that runs a script, implemented with button…/button. This is valuable when you want more control over the look of your button. In this particular example, the button contains both an icon and some text. This approach should only be used when script is relied upon.

```
<button onclick="return doStuff();">
 <img src="stuff.gif" alt="stuff icon">
 Do Stuff
</button>
```

Testing Procedure

1. In a user agent that supports Scripting
   - Click on the control with the mouse.
   - Check that the scripting action executes properly.
   - If the control is an anchor element, check that the URI in the `href` attribute of the anchor element is not invoked.
   - Check that it is possible to navigate to and give focus to the control via the keyboard.
   - Set keyboard focus to the control.
   - Check that pressing ENTER invokes the scripting action.
   - If the control is an anchor element, check that the URI in the href attribute of the anchor element is not invoked.
2. In a user agent that does not support Scripting
   - Click on the control with the mouse.
   - If the control is an anchor element, check that the URI in the href attribute of the anchor element is invoked.
   - Check that it is possible to navigate to and give focus to the control via the keyboard.
   - Set keyboard focus to the control.
   - If the control is an anchor element, check that pressing ENTER invokes the URI of the anchor element's `href` attribute.

Technique SCR2: Using redundant keyboard and mouse event handlers[120]

Examples

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Changing Image Source in a Device Independent Manner</title>
  <script>
    /* This function will change the image src of an image element.
     * param imgId - the id of the image object to change
     * param isOver - true when mouse is over or object has focus,
      false when mouse move out or focus is lost
    */
    function updateImage(imgId, isOver) {
    var theImage = document.getElementById(imgId);
    if (theImage != null) {
      if (isOver) {
        theImage.setAttribute("src","yellowplus.gif");
      }
      else {
        theImage.setAttribute("src","greyplus.gif");
      }
    }
  }
  </script>
 </head>
 <body>
 <p>Mouse over or tab to the links below and see the image change.</p>
 <a href="https://www.w3.org/WAI/"
```

[120] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR2.html

```
    onmouseover="updateImage('wai', true);" onfocus="updateImage('wai', true);"
    onmouseout="updateImage('wai',false);" onblur="updateImage('wai',false);">
    <img alt="" id="wai" src="greyplus.gif">
    W3C Web Accessibility Initiative</a> &amp;
 <a href="https://www.w3.org/International/" onmouseover="updateImage('i18n', true);"
    onfocus="updateImage('i18n',true);" onmouseout="updateImage('i18n',false);"
    onblur="updateImage('i18n',false);">
    <img alt="" id="i18n" src="greyplus.gif">
    W3C Internationalization</a>
 </body>
 </html>
```

Testing Procedure

Load the Web page and test the events using a mouse and via the keyboard.

1. Check that the "standard" image is displayed as expected when the Web page is loaded.
2. Using the Mouse

1. Move the mouse over the element containing the event handlers (in this example it is an anchor element). Check that the image changes to the expected image.
2. Move the mouse off of the element. Check that the image changes back to the "standard" image.

3. Using the Keyboard

1. Use the keyboard to set focus to the element containing the event handlers. Check that the image changes to the expected image.
2. Use the keyboard to remove focus from the element (generally by moving focus to another element). Check that the image changes to the "standard" image.

4. Verify that the layout of other elements on the page is not affected when the image is changed.

2.1.2 No Keyboard Trap (Level A)[121] – The purpose of this Success Criteria is to eliminate the possibility of keyboard focus becoming "stuck" within specific sections of content on a web page. This is a common problem that arises if multiple file formats are combined on a single page in a manner that is produced by plug-ins or other embedded programs. The functionality of the website may, on occasion, restrict the user's focus to a certain section of the material; however, as long as the user is aware of how to "untrap" the focus, this limitation should not affect the user's experience.

Technique G21: Ensuring that users are not trapped in content[122]

Examples

- Once a user tabs into an applet, further tabs are handled by the applet preventing the person from tabbing out. However, the applet is designed so that it returns keyboard focus back to the parent window when the person finishes tabbing through the tab sequence in the applet.
- A page that includes content that is not accessibility-supported contains instructions about how to move focus back to the accessibility-supported content via the keyboard. The instructions precede the non accessibility-supported content.
- The help information available from the content that is not accessibility supported documents how to move focus back to the accessibility-supported content via the keyboard, and the help information can be accessed via the keyboard.
- The help information available for the Web page documents how to move focus from the content that is not accessibility supported to the accessibility-supported content via the keyboard, and the help information can be accessed via the keyboard.

Testing Procedure

1. Tab through content from start to finish.
2. Check to see that keyboard focus is not trapped in any of the content.
3. If keyboard focus appears to be trapped in any of the content, check that help information is available explaining how to exit the content and can be accessed via the keyboard.

[121] https://www.w3.org/WAI/WCAG21/Understanding/no-keyboard-trap.html
[122] https://www.w3.org/WAI/WCAG21/Techniques/general/G21.html

2.1.4 Character Key Shortcuts (Level A)[123] – The purpose of this Success Criterion is to reduce the chances of inadvertently using a keyboard shortcut by making the shortcuts less obvious. Users of speech input, who enter words as strings of letters, and users of keyboards who frequently touch keys by accident find character key shortcuts inefficient and bothersome. Character key shortcuts, on the other hand, are helpful for users of keyboards who touch keys frequently by accident. In order to solve this issue, authors need to provide users the opportunity to turn off or customize shortcuts that solely use character keys.

Sufficient Techniques:

- Users have a way to turn off single-key shortcuts.
- A mechanism is provided to allow users to change character-key shortcuts. The remapping mechanism includes non-printing characters. The alternative shortcuts could be longer strings of up to 25 characters that would directly serve as native speech commands for any speech engine.

2.2 Enough Time[124] – The purpose of this section is to ensure that users interacting with the target site had enough time to read and use the content. There are two criteria points for this section in order to obtain level A conformance. For this section, there is not an option for level AA conformance – only A or AAA.

2.2.1 Timing Adjustable (Level A)[125] – The purpose of this Success Criterion is to make certain that users with disabilities are provided with sufficient time to interact with material on the website whenever it is feasible to do so. Reading text or performing duties such as filling out online forms may take longer for people with disabilities such as blindness, low vision, dexterity difficulties, and cognitive limitations. These people may require additional time. If certain Web functions are time-sensitive, it will be difficult for some users to complete the task that is necessary before the timer reaches its maximum. Because of this, it's possible that they won't be able to use the service. People with impairments will have a better chance of completing functions successfully if these functions are designed so that they are not time dependent. It is helpful for users who require more time than expected to effectively accomplish activities to have the ability to disable time limitations, alter the length of time restrictions, or request more time before a time limit occurs. These choices are presented in the order in which the user will find them to be of the greatest assistance to them. Turning off time restrictions is preferable to adjusting the length of time limits, which in turn is preferable to asking additional lead time before a time limit kicks in.

Situation A: If there are session time limits:

Technique G133: 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[126]

Example 1: A checkbox for requesting a specific extension
A Web page contains the first part of a five-part form. Immediately following the general instructions for completing the form is a checkbox with the label, "Allow an additional 15 minutes to complete each part of this form."

Example 2: Requesting an indefinite extension
A Web page contains the first part of a three-part form. Each part of the form includes more than 10 items. Some items require users to follow links for additional information. Immediately following the general instructions for

[123] https://www.w3.org/WAI/WCAG21/Understanding/character-key-shortcuts.html
[124] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#enough-time
[125] https://www.w3.org/WAI/WCAG21/Understanding/timing-adjustable.html
[126] https://www.w3.org/WAI/WCAG21/Techniques/general/G133.html

completing the form is a checkbox with the label, "Allow as much time as I need to complete this form. I understand that I must close (quit) the Web browser if I choose to stop before completing the last part of the form."

Testing Procedure
If the Web page contains the first part of a multipart form:
1. Check that the Web page includes a checkbox to request additional time to complete the form.
2. Check that if the checkbox is checked, additional time is provided to complete the form.

Technique G198: Providing a way for the user to turn the time limit off[127]

Examples
- A page has a listing of news headlines that automatically update every minute. At the top of the page is a link that turns off the updating.

Testing Procedure
1. Check that there is a mechanism to turn off any time limits near the top of the page.
2. Verify that the time limit for the page is long enough that a user can easily navigate to the mechanism even if they are 10 times slower than most users.

Situation B: If a time limit is controlled by a script on the page:

Technique G198: Providing a way for the user to turn the time limit off

Examples
{See Above}
Testing Procedure
{See Above}

Technique G180: Providing the user with a means to set the time limit to 10 times the default time limit[128]

Examples
- An airline has an online ticket purchasing application. By default, the application has a 1 minute time limit for each step of the purchase process. At the beginning of the session, a Web page includes information that says, "We expect that each step in the purchasing process will take users one minute to complete. Would you like to adjust the time limit?" followed by several radio buttons "1 minute, 2 minutes, 5 minutes, 10 minutes."
- A Web based email application automatically logs users out when there has been no activity for 30 minutes. The application includes a preference that allows users to adjust the amount of time to any value.

Testing Procedure
1. Check to see if there is a mechanism to set the time limit to 10 times the default time limit.
2. Change the time limit to a new value that is 10 times the default time limit.
3. Perform an action that has a time limit.
4. Wait until the default time limit has passed.
5. Check that the time limit does not expire until the limit specified in step 2 has passed.

Technique SCR16: Providing a script that warns the user a time limit is about to expire[129] AND Technique SCR1: Allowing the user to extend the default time limit[130]

Examples for SCR16:

[127] https://www.w3.org/WAI/WCAG21/Techniques/general/G198.html
[128] https://www.w3.org/WAI/WCAG21/Techniques/general/G180.html
[129] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR16.html
[130] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR1.html

A page of stock market quotes uses script to refresh the page every five minutes in order to ensure the latest statistics remain available. 20 seconds before the five-minute period expires, a confirm dialog appears asking if the user needs more time before the page refreshes. This allows the user to be aware of the impending refresh and to avoid it if desired.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Stock Market Quotes</title>
  <script>
    function timeControl() {
      // set timer for 4 min 40 sec, then ask user to confirm.
      setTimeout('userCheck()', 280000);
    }
    function userCheck() {
      // set page refresh for 20 sec
      var id=setTimeout('pageReload()', 20000);
      // If user selects "OK" the timer is reset
      // else the page will refresh from the server.
      if (confirm("This page is set to refresh in 20 seconds.
        Would you like more time?"))
        {
          clearTimeout(id);
          timeControl();
        }
    }
    function pageReload() {
      window.location.reload(true);
    }
    timeControl();
  </script>
</head>
<body>
  <h1>Stock Market Quotes</h1>
  ...
</body>
</html>
```

Testing Procedure SCR16:
On a Web page that has a time limit controlled by a script:
1. Load the page and start a timer that is 20 seconds less than the time limit.
2. When the timer expires, check that a confirmation dialog is displayed warning of the impending time limit.

Examples SCR1:
- A Web page contains current stock market statistics and is set to refresh periodically. When the user is warned prior to refreshing the first time, the user is provided with an option to extend the time period between refreshes.
- In an online chess game, each player is given a time limit for completing each move. When the player is warned that time is almost up for this move, the user is provided with an option to increase the time.

Testing Procedure SCR1:
1. On a Web page that uses scripts to enforce a time limit, wait until the time limit has expired.
2. Determine if an option is provided to extend the time limit.

Situation C: If there are time limits on reading:

Technique G4: Allowing the content to be paused and restarted from where it is paused[131]

Examples

[131] https://www.w3.org/WAI/WCAG21/Techniques/general/G4.html

- A site contains a scrolling news banner at the top of the page. Users who need more time to read it can press the Escape key to pause the scrolling. Pressing Escape again restarts it.
- A Web page contains a link labeled "How to tie a shoe" which links to an animation. Text immediately preceding the link informs the user that pressing the spacebar will pause the animation and restart it again.

Testing Procedure
On a page with moving or scrolling content,
1. Use the mechanism provided in the Web page or by the user agent to pause the moving or scrolling content.
2. Check that the moving or scrolling has stopped and does not restart by itself.
3. Use the mechanism provided to restart the moving content.
4. Check that the movement or scrolling has resumed from the point where it is stopped.

Technique G198: Providing a way for the user to turn the time limit off

Examples
{See Above}
Testing Procedure
{See Above}

Technique SCR33: Using script to scroll content, and providing a mechanism to pause it[132]

Examples
In this example CSS and Javascript are used to visually present some text in a scrolling format. A link is included to pause the scrolling movement.
This implementation will display the full text and omit the link when Javascript or CSS are unsupported or inactive.
The following code is an amended version of webSemantic's Accessible Scroller (as at July 2008).

The XHTML component:
```
...
<div id="scroller">
<p id="tag">This text will scroll and a Pause/Scroll link will be present
when Javascript and CSS are supported and active.</p>
</div>
...
```

The CSS component:
```
...
body {font:1em verdana,sans-serif; color:#000; margin:0}

/* position:relative and overflow:hidden are required */
#scroller { position:relative; overflow:hidden; width:15em; border:1px solid #008080; }

/* add formatting for the scrolling text */
#tag { margin:2px 0; }

/* #testP must also contain all text-sizing properties of #tag  */
#testP { visibility:hidden; position:absolute; white-space:nowrap; }

/* used as a page top marker and to limit width */
#top { width:350px; margin:auto; }
...
```

The JavaScript component:
```
var speed=50        // speed of scroller
var step=3          // smoothness of movement
var StartActionText= "Scroll"  // Text for start link
var StopActionText = "Pause"   // Text for stop link

var x, scroll, divW, sText=""
```

[132] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR33.html

```
function onclickIE(idAttr,handler,call){
  if ((document.all)&&(document.getElementById)){idAttr[handler]="Javascript:"+call}
}

function addLink(id,call,txt){
  var e=document.createElement('a')
  e.setAttribute('href',call)
  var linktext=document.createTextNode(txt)
  e.appendChild(linktext)
  document.getElementById(id).appendChild(e)
}

function getElementStyle() {
    var elem = document.getElementById('scroller');
    if (elem.currentStyle) {
        return elem.currentStyle.overflow;
    } else if (window.getComputedStyle) {
        var compStyle = window.getComputedStyle(elem, '');
        return compStyle.getPropertyValue("overflow");
    }
    return "";
}

function addControls(){
// test for CSS support first
// test for the overlow property value set in style element or external file
if (getElementStyle()=="hidden") {
  var f=document.createElement('div');
  f.setAttribute('id','controls');
  document.getElementById('scroller').parentNode.appendChild(f);
  addLink('controls','Javascript:clickAction(0)',StopActionText);
  onclickIE(document.getElementById('controls').childNodes[0],"href",'clickAction(0)');
  document.getElementById('controls').style.display='block';
  }
}

function stopScroller(){clearTimeout(scroll)}

function setAction(callvalue,txt){
  var c=document.getElementById('controls')
  c.childNodes[0].setAttribute('href','Javascript:clickAction('+callvalue+')')
  onclickIE(document.getElementById('controls').childNodes[0],"href",'clickAction

('+callvalue+')')
  c.childNodes[0].firstChild.nodeValue=txt
}

function clickAction(no){
  switch(no) {
    case 0:
      stopScroller();
      setAction(1,StartActionText);
      break;
    case 1:
      startScroller();
      setAction(0,StopActionText);
  }
}

function startScroller(){
  document.getElementById('tag').style.whiteSpace='nowrap'
  var p=document.createElement('p')
  p.id='testP'
  p.style.fontSize='25%' //fix for mozilla. multiply by 4 before using
  x-=step
  if (document.getElementById('tag').className) p.className=document.getElementById

('tag').className
  p.appendChild(document.createTextNode(sText))
```

```
  document.body.appendChild(p)
  pw=p.offsetWidth
  document.body.removeChild(p)
  if (x<(pw*4)*-1){x=divW}
  document.getElementById('tag').style.left=x+'px'
  scroll=setTimeout('startScroller()',speed)
}

function initScroller(){
  if (document.getElementById && document.createElement && document.body.appendChild) {
    addControls();
    divW=document.getElementById('scroller').offsetWidth;
    x=divW;
    document.getElementById('tag').style.position='relative';
    document.getElementById('tag').style.left=divW+'px';
    var ss=document.getElementById('tag').childNodes;
    for (i=0;i<ss.length;i++) {sText+=ss[i].nodeValue+" "};
    scroll=setTimeout('startScroller()',speed);
  }
}

function addLoadEvent(func) {
  if (!document.getElementById | !document.getElementsByTagName) return
  var oldonload = window.onload
  if (typeof window.onload != 'function') {
    window.onload = func;
  } else {
    window.onload = function() {
      oldonload()
      func()
    }
  }
}
addLoadEvent(initScroller)
```

Testing Procedure

1. Check that a mechanism is provided to pause the scrolling content.
2. Use the pause mechanism to pause the scrolling content.
3. Check that the scrolling has stopped and does not restart by itself.
4. Check that a mechanism is provided to restart the paused content.
5. Use the restart mechanism provided to restart the scrolling content.
6. Check that the scrolling has resumed from the point where it is stopped.

Technique SCR36: Providing a mechanism to allow users to display moving, scrolling, or auto-updating text in a static window or area[133]

Expanding Scrolling Text in Place

A large block of text is scrolled through a small marquee area of the page. A button lets the user stop the scrolling and display the entire block of text.

This code example requires that both CSS and JavaScript be turned on and available.

The CSS component:

```
#scrollContainer {
        visibility: visible;
        overflow: hidden;
        top: 50px; left: 10px;
        background-color: darkblue;
      }
      .scrolling {
        position: absolute;
```

[133] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR36.html

```
        width: 200px;
        height: 50px;
      }
      .notscrolling {
        width: 500px;
        margin:10px;
      }
      #scrollingText {
        top: 0px;
        color: white;
      }
      .scrolling #scrollingText {
        position: absolute;
      }
      </a>

The script and HTML content:
<script type="text/javascript">

      var tid;
      function init() {
        var st = document.getElementById('scrollingText');
        st.style.top = '0px';
        initScrolling();
      }
      function initScrolling () {
        tid = setInterval('scrollText()', 300);
      }
      function scrollText () {
        var st = document.getElementById('scrollingText');
        if (parseInt(st.style.top) > (st.offsetHeight*(-1) + 8)) {
          st.style.top = (parseInt(st.style.top) - 5) + 'px';
        } else {
          var sc = document.getElementById('scrollContainer');
          st.style.top = parseInt(sc.offsetHeight) + 8 + 'px';
        }
      }
      function toggle() {
        var scr = document.getElementById('scrollContainer');
        if (scr.className == 'scrolling') {
          scr.className = 'notscrolling';
          clearInterval(tid);
           document.getElementById('scrollButton').value="Shrink";
        } else {
          scr.className = 'scrolling';
          initScrolling();
          document.getElementById('scrollButton').value="Expand";
        }
      }
  <input type="button" id="scrollButton" value="Expand" onclick="toggle()" />
  <div id="scrollContainer" class="scrolling">
    <div id="scrollingText" class="on">
    .... Text to be scrolled ...
    </div>
  </div>
...
```

Testing Procedure
{None Listed}

2.2.2 Pause, Stop, Hide (Level A)[134] – When visitors are interacting with the target site, this Success Criterion's main objective is to ensure that they do not become sidetracked in any way. Content is said to be "moving, blinking, and scrolling" when it gives the impression of being in motion just by its

[134] https://www.w3.org/WAI/WCAG21/Understanding/pause-stop-hide.html

appearance. A few common examples include moving visuals, synchronized media presentations, animations, real-time games, and scrolling financial tickers. The term "auto-updating" refers to content that alters or disappears at regular intervals according to a schedule that the user specifies. Examples of time-based content include audio, automatically updated weather reports, news, automatically progressing presentations and messaging, and stock price updates. The requirements for content that is moving, blinking, or scrolling and the requirements for content that is automatically updating are equivalent, with the exception that authors may give users a way to regulate the frequency of updates when content is auto-updating, and there is no five-second exception for auto-updating because it is ineffective to auto-update for a short period of time before ceasing to do so.

Technique G5: Allowing users to complete an activity without any time limit

Examples

- An interactive exam for a course provides all questions on one Web page. Users can take as much time as they need to complete it.
- In an interactive game, users can take as much time as they like on their turn instead of having to complete their move within a limited amount of time.
- In an online auction, each bidder can submit only one bid rather than submitting multiple competitive bids based on timing. The bidding is open for a full day, providing enough time for anyone to complete the simple bid form. Once bidding is closed, the best bid wins.

Testing Procedure

1. Determine if any timed interactions are present (client and/or server side).

2.3 Seizures and Physical Reactions[135] – This section works to ensure that the target site do not design its content in a manner that would be known to cause seizures or any other physical reactions. There is one criterion point for this section to be level A. For this section, there is not an option for level AA conformance – only A or AAA.

2.3.1 Three Flashes or Below Threshold (Level A)[136] – Users should be able to access the entirety of a website without having their photosensitive seizures triggered in order for this Success Criterion to be considered met. People who have photosensitive seizure disorders are more likely to have seizures if they are exposed to information that flashes at certain frequencies for a longer period of time than a few times. Because people are significantly more sensitive to intense red flashing than they are to other colors, a separate test has been developed specifically for it. Rules from the broadcasting sector served as the foundation for these guidelines, which are later adapted for desktop displays, on which content is viewed more closely (using a larger angle of vision). Flashing might be caused by the display itself, the computer that is rendering the image, or the material that is being created. The author does not have any influence over the first two points. It is possible to address issues by using the layout of the computer and display as well as the processing capability of the computer. This criterion is developed to ensure that flickering that is more severe than the flash threshold is not caused by the actual material being displayed on the screen. A video clip, an animated graphic, or even close-up shots of a series of strobe lights or explosions could be used as part of the material, for instance.

Technique G19: Ensuring that no component of the content flashes more than three times in any 1-second period[137]

[135] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#seizures-and-physical-reactions

[136] https://www.w3.org/WAI/WCAG21/Understanding/three-flashes-or-below-threshold.html

[137] https://www.w3.org/WAI/WCAG21/Techniques/general/G19.html

Examples

- Content has lightning flashes. Content is designed so that lightning only flashes two or three times without a pause in flashing.

Testing Procedure

1. Check that there are no more than three flashes during any 1-second period.
2. If there are three flashes, check that the Light/Dark status at the end of the 1-second period is the same as at the start.

Technique G176: Keeping the flashing area small enough[138]

Examples

- An author creates an animation that will be displayed on a screen in the entrance lounge at a company. Using the size and resolution of the display and the closest distance that a person can stand when viewing the display, they calculate the size of 25% of the 10 degree of central vision in pixels (using the formula above). This would be the *small safe area*. They then are careful to never flash any area larger than the *small safe area*.

Testing Procedure

1. The *small safe area* is calculated.
2. Check that only one area of the screen is flashing at any time.
3. Check that the flashing content would fit into a contiguous container whose area is less than the *small safe area*.

Technique G15: Using a tool to ensure that content does not violate the general flash threshold or red flash threshold[139]

Examples

- An animation of a thunderstorm shows six flashes of lightning. The flashes are so fast and large that the general flash threshold is violated when tested with a flash analysis tool. The animation is modified to create a short pause after each pair of lightning flashes. After the changes are made, the animation does not violate the general flash threshold.

Testing Procedure

Check to see to see that content does not violate the general flash and/or red flash threshold

1. use a tool to determine that neither the General Flash nor Red Flash threshold are exceeded

2.4 Navigable[140] – Under WCAG, this section ensures that the target site provides ways to assist individuals navigate the site, find content, and determine their location on the page. In order to reach level AA conformance, seven criteria points have to be addressed.

2.4.1 Bypass Blocks (Level A)[141] – This Success Criterion aims to provide readers with a more direct path to the page's primary information by rewarding them for reading the content in the order in which it is presented. The content that is displayed on web pages and within applications constantly change between screens and pages. The heading graphics, navigation links, and advertising frames that are listed below are just a few instances of recurring blocks of content that can be found on websites. For the purposes of this phrase, single words, sentences, or links do not count as blocks because they are considered to be relatively short recurring portions.

- Creating links to skip blocks of repeated material using one of the following techniques:

[138] https://www.w3.org/WAI/WCAG21/Techniques/general/G176.html
[139] https://www.w3.org/WAI/WCAG21/Techniques/general/G15.html
[140] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#navigable
[141] https://www.w3.org/WAI/WCAG21/Understanding/bypass-blocks.html

Technique G1: Adding a link at the top of each page that goes directly to the main content area[142]

Example 1: An online newspaper
An on-line newspaper contains many sections of information: a search function, a corporate banner, sidebars, minor stories, how to contact the newspaper, etc. The lead story is located in the middle of the page. The first link that the user reaches when tabbing through the page is titled "Skip to Lead Story". Activating the link moves visual focus to the story. Pressing tab again takes the user to the first link in the main story.

Example 2: A "Skip to main content" link
A Web page includes a variety of navigation techniques on each page: a bread crumb trail, a search tool, a site map, and a list of related resources. The first link on the page is titled "Skip to Main Content". A user activates the link to skip over the navigation tools.

Testing Procedure
1. Check that a link is the first focusable control on the Web page.
2. Check that the description of the link communicates that it links to the main content.
3. Check that the link is either always visible or visible when it has keyboard focus.
4. Check that activating the link moves the focus to the main content.
5. Check that after activating the link, the keyboard focus has moved to the main content.

Technique G123: Adding a link at the beginning of a block of repeated content to go to the end of the block[143]

Example 1: Skip navigation links
The pages on an organization's Web site include a navigation bar or main menu containing links to major sections of the site, the site map, information about the organization, and how to contact the organization. The first link in this area is titled "Skip Navigation Links". A user activates the link to skip over these links.

Example 2: A book index
A book contains an index that is divided into a set of pages. In the content at the beginning of each page of the index are links for each letter of the alphabet, linking into the index where the entries start with that letter. The first link in the set is titled "Skip Links into Index". A user activates this link to skip over the links.

Example 3: Several sets of links
All the pages on a Web site include a section containing links to the site map, information about the organization, and how to contact the organization. All the pages in each section of the site also contain a set of links to its subsections. The first link in the first block is titled "Skip Navigation Links" and skips over the first set of links. The first link in the second block is titled "Skip Section Links" and skips over the subsection links.

Example 4: HTML page with several blocks of navigation links
This example demonstrates both the use of Heading elements at the beginning of each section (H69) and links that skip to the end of each section. This allows people to skip blocks of repeated content using keyboard navigation or using heading navigation, depending on the capabilities of their user agents. Note that some sections of the content are wrapped in a div element to work around a limitation of Internet Explorer (see the User Agents Notes for Creating HTML links to skip blocks of content (future link)).

```
<p><a href="#content">Content title</a></p>
  <h2>Main Navigation</h2>
  <ul>
  <li><a href="#subnav">Sub Navigation</a></li>
  <li><a href="/a/">Link A</a></li>
  <li><a href="/b/">Link B</a></li>
  <li><a href="/c/">Link C</a></li>
  ...
  <li><a href="/j/">Link J</a></li>
  </ul>
  <div class="iekbfix">
```

[142] https://www.w3.org/WAI/WCAG21/Techniques/general/G1.html
[143] https://www.w3.org/WAI/WCAG21/Techniques/general/G123.html

```
<h2 id="subnav">Sub Navigation</h2>
<ul>
<li><a href="#ultranav">Ultra Sub Navigation</a></li>
<li><a href="/suba/">Sub A</a></li>
<li><a href="/subb/">Sub B</a></li>
<li><a href="/subc/">Sub C</a></li>
...
<li><a href="/subj/">Sub J</a></li>
</ul>
</div>
<div class="iekbfix">
<h2 id="ultranav">Ultra Sub Navigation</h2>
<ul>
<li><a href="#content">Content title</a></li>
<li><a href="/ultraa/">Ultra A</a></li>
<li><a href="/ultrab/">Ultra B</a></li>
<li><a href="/ultrac/">Ultra C</a></li>
...
<li><a href="/ultraj/">Ultra J</a></li>
</ul>
</div>
<div>
<h2 id="content">Content title</h2>
<p>Now that I have your attention...</p>
</div>
```

And the CSS

```
div.iekbfix {
  width: 100%;
  }
```

Testing Procedure

1. Check that a link is the last focusable control before the block of repeated content or the first link in the block.
2. Check that the description of the link communicates that it skips the block.
3. Check that the link is either always visible or visible when it has keyboard focus.
4. Check that after activating the link, the keyboard focus has moved to the content immediately after the block.

Technique G124: Adding links at the top of the page to each area of the content[144]

Examples

- 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.

Testing Procedure

For each link in the set of links provided for this purpose:

1. Check that the only controls in the Web page that precede the link are other links in the set.
2. Check that the description of each link communicates that it links to some section of the content.
3. Check that the link is either always visible or visible when it has keyboard focus.
4. Check that activating the link moves the focus to that section of the content.

- Grouping blocks of repeated material in a way that can be skipped, using one of the following techniques:

Technique ARIA11: Using ARIA landmarks to identify regions of a page

Examples

{See Above}

Testing Procedures

{See Above}

[144] https://www.w3.org/WAI/WCAG21/Techniques/general/G124.html

Technique H69: Providing heading elements at the beginning of each section of content[145]

Example 1: Organizing the sections of a search page
This example marks up each section heading using h2 elements.

```
<h1>Search Technical Periodicals</h1>
 <h2>Search</h2>
 <form action="/search" role="search">
  <label for="searchInput">Enter search topic:</label>
  <input id="searchInput" type="text">
  <input type="submit" value="Search">
 </form>
 <section>
   <h2>Available Periodicals</h2>
   <ul>
     <li><a href="https://example.com/pcoder">Professional Coder</a></li>
     <li><a href="https://example.com/algo">Algorithms</a></li>
     <li><a href="https://example.com/jse">Journal of Software Engineering</a></li>
   </ul>
 </section>
 <section>
   <h2>Search Results</h2>
   ... search results are returned in this section ...
 </section>
```

Example 2: Headings show the overall organization of the content
In this example, heading markup is used to make the navigation and main content sections perceivable.

```
<main>
  <h1>Why Monday Monkey Lives For The Weekend</h1>
  ... text, images, and other material making up the main content ...
</main>
<nav aria-labelledby="nav-heading">
  <h2 id="nav-heading">More About Monday Monkey, Ltd.</h2>
  <ul>
    <li><a href="/about">About us</a></li>
    <li><a href="/contact">Contact us</a></li>
    ...
  </ul>
</nav>
```

Example 3: Headings show the organization of material within the main content

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Cooking techniques</title>
  </head>
  <body>
    <main>
      <h1>Cooking techniques</h1>
      <p>... some text here ...</p>
      <h2>Cooking with oil</h2>
      <p>... text of the section ...</p>
      <h2>Cooking with butter</h2>
      <p>... text of the section ...</p>
    </main>
  </body>
</html>
```

Testing Procedure

[145] https://www.w3.org/WAI/WCAG21/Techniques/html/H69.html

1. Check that the content is divided into separate sections.
2. Check that each section on the page starts with a heading.

Technique H70: Using frame elements to group blocks of repeated material[146]

Examples

The following example shows the use of two frames to organize content. The source of the first frame is the Web page, navigation.html, which contains the HTML for the navigation. This frame has a title attribute which identifies it as a navigation bar. The second frame contains the main content of the site as indicated by the source parameter of main.html and the title attribute, "Main News Content" which identifies its function.

```
<frameset cols="20%, *">
  <frame src="navigation.html" name="navbar" title="Navigation Bar" />
  <frame src="main.html" name="maincontent" title="Main News Content" />
  <noframes>
    <p>View <a href="noframe.html">no frame version</a>.</p>
  </noframes>
</frameset>
```

Testing Procedure

If the Web page uses frames to organize content:

1. Check if repeated blocks of content are organized into separate frames.
2. Check that the frames with repeated content appear in the same location within each frameset.

Technique SCR28: Using an expandable and collapsible menu to bypass block of content[147]

Example 1

The navigation links at top of a Web page are all entries in a menu implemented using HTML, CSS, and Javascript. When the navigation bar is expanded, the navigation links are available to the user. When the navigation bar is collapsed, the links are not available.

```
...

  <script type="text/javascript">
  function toggle(id){
    var n = document.getElementById(id);
    n.style.display =  (n.style.display != 'none' ? 'none' : '' );
  }
  </script>

...

  <a href="#" onclick="toggle('navbar')">Toggle Navigation Bar</a>

  <ul id="navbar">
  <li><a href="http://target1.html">Link 1</a></li>
  <li><a href="http://target2.html">Link 2</a></li>
  <li><a href="http://target3.html">Link 3</a></li>
  <li><a href="http://target4.html">Link 4</a></li>
  </ul>

...
```

Example 2

The table of contents for a set of Web pages is repeated near the beginning of each Web page. A button at the beginning of the table of contents lets the user remove or restore it on the page.

```
...

   <script type="text/javascript">
  function toggle(id){
    var n = document.getElementById(id);
    n.style.display =  (n.style.display != 'none' ? 'none' : '' );
  }
```

[146] https://www.w3.org/WAI/WCAG21/Techniques/html/H70.html
[147] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR28.html

```
  </script>

  ...

  <button onclick="return toggle('toc');">Toggle Table of Contents</button>
  <div id="toc">
    ...
  </div>

...
```

Testing Procedure

1. Check that some user interface control allows the repeated content to be expanded or collapsed.
2. Check that when the content is expanded, it is included in the programmatically determined content at a logical place in the reading order.
3. Check that when the content is collapsed, it is not part of the programmatically determined content.

2.4.2 Page Titled (Level A)[148] – The goal of this Success Criterion is to make it easier for site users to identify the content they are looking for and to locate themselves within that content by requiring that every web page have a descriptive title. Consumers are provided with the location information they require by titles, which do not require visitors to read or comprehend the content of the page. When titles are presented in site maps or lists of search results, users are able to locate the content that they require in a more expedient manner. User agents ensure that the page title is easily accessible to the user at all times so that they are able to correctly identify the page. For instance, the title of the page can be displayed by a user agent either in the bar at the top of the window or as the name of the tab that contains the page itself. When a page is a document or a web application, the function of the page can be determined simply by looking at the name of the document or application. Note that in addition to the usage of the document's or web application's name, alternative terms may also be used to identify the objective or subject of the page. Using the document's or web application's name is not required.

Technique G88: Providing descriptive titles for Web pages[149]

Example 1: A title that lists the most important identifying information first
A Web page is published by a group within a larger organization. The title of the Web page first identifies the topic of the page, then shows the group name followed by the name of the parent organization.

```
<title>Working with us: The Small Group: The Big Organization</title>
```

Example 2: A synchronized media presentation with a descriptive title
A synchronized media presentation about the 2004 South Asian tsunami is titled "The Tsunami of 2004."

Example 3: A Web page with a descriptive title in three parts
A Web page provides guidelines and suggestions for creating closed captions. The Web page is part of a "sub-site" within a larger site. The title is separated into three parts by dashes. The first part of the title identifies the organization. The second part identifies the sub-site to which the Web page belongs. The third part identifies the Web page itself. (For a working example, see WGBH – Media Access Group – Captioning FAQ.)

Example 4: A newspaper Web page
A Web site that only permits viewing of the current edition titles its Web page "National News, Front Page". A Web site that permits editions from different dates to be viewed titles its Web page, "National News, Front Page, Oct 17, 2005".

Testing Procedure

[148] https://www.w3.org/WAI/WCAG21/Understanding/page-titled.html
[149] https://www.w3.org/WAI/WCAG21/Techniques/general/G88.html

1. Check that the Web page has a title
2. Check that the title is relevant to the content of the Web page.
3. Check that the Web page can be identified using the title.

Technique H25: Providing a title using the title element[150]

Examples

This example defines a document's title.

```
<!doctype html>
<html lang="en">
  <head>
    <title>The World Wide Web Consortium</title>
  </head>
  <body>
      ...
  </body>
</html>
```

Testing Procedure

1. Examine the source code of the HTML document and check that a non-empty `title` element appears in the `head` section.
2. Check that the `title` element describes the document.

2.4.3 Focus Order (Level A)[151] – The purpose of this Success Criterion is to ensure that, when users navigate sequentially through the material, they will meet information in a format that is both keyboard-operable and consistent with the way the content is meant to be interpreted. Confusion is reduced as a result of this since it makes it possible for customers to form a unified mental picture of the material. It is feasible to arrange the content in a variety of ways that represent the logical relationships found within it. Moving through the table components one at a time, for example, and doing so one row or one column at a time both represent the logical linkages that are present in the material, as does doing so one row or one column at a time. Either of the two possible sequences could fulfill this success criterion.

Technique G59: Placing the interactive elements in an order that follows sequences and relationships within the content[152]

Examples

- A form contains two text input fields that are to be filled in sequentially. The first text input field is placed first in the content, the second input field is placed second.
- A form contains two, side-by-side sections of information. One section contains information about an applicant; the other section contains information about the applicant's spouse. All the interactive elements in the applicant section receive focus before any of the elements in the spouse section. The elements in each section receive focus in the reading order of that section.

Testing Procedure

1. Determine the order of interactive elements in the content.
2. Determine the logical order of interactive elements.
3. Check that the order of the interactive elements in the content is the same as the logical order.

- Giving focus to elements in an order that follows sequences and relationships within the content using one of the following techniques:

[150] https://www.w3.org/WAI/WCAG21/Techniques/html/H25.html
[151] https://www.w3.org/WAI/WCAG21/Understanding/focus-order.html
[152] https://www.w3.org/WAI/WCAG21/Techniques/general/G59.html

Technique H4: Creating a logical tab order through links, form controls, and objects[153]

Example 1
A genealogical search form searches for marriage records. The search form includes several input fields for the bride and the groom. The form is marked up using a data table that includes the fields of the groom in the first column and the fields of the bride in the second column. The order in the content is row by row but the author feels it is more logical to navigate the form column by column. This way, all the groom's criteria can be filled in before moving on to the bride's criteria. The tabindex attributes of the input fields are used to specify a tab order that navigates column by column.

```
<form action="#" method="post">
 <table summary="the first column contains the search criteria
  of the groom, the second column the search criteria of
  of the bride">
 <caption>Search for marriage records</caption>
 <tr>
   <th>Search criteria</th>
   <th>Groom</th>
   <th>Bride</th>
 </tr>
 <tr>
  <th>First name</th>
  <td><input type="text" size="30" value="" name="groomfirst"
      title="First name of the groom" tabindex="1"></td>
  <td><input type="text" size="30" value="" name="bridefirst"
       title="First name of the bride" tabindex="4"></td>
 </tr>
 <tr>
  <th>Last name</th>
  <td><input type="text" size="30" value="" name="groomlast"
      title="Last name of the groom" tabindex="2"></td>
  <td><input type="text" size="30" value="" name="bridelast"
      title="Last name of the bride" tabindex="5"></td>
 </tr>
 <tr>
  <th>Place of birth</th>
  <td><input type="text" size="30" value="" name="groombirth"
      title="Place of birth of the groom" tabindex="3"></td>
  <td><input type="text" size="30" value="" name="bridebirth"
      title="Place of birth of the bride" tabindex="6"></td>
 </tr>
</table>
</form>
```

Example 2
A Web page contains a search field in the upper right corner. The field is given `tabindex="1"` so that it will occur first in the tab order, even though it is not first in the content order.

Example 3
`Tabindex` values need not be sequential nor must they begin with any particular value. The values do not have to be unique. Elements that have identical `tabindex` values are navigated in the order they appear in the character stream.

In sections of the content where the tab order follows the content order, it can be less error prone to give all elements the same tabindex value rather than specifying a different number for each element. Then it is easy to rearrange those elements or add new elements and maintain a logical tab order.

```
 <a href="xxx" tabindex = "1">First link in list</a>
<a href="xxx" tabindex = "1">Second link in list</a>
<a href="xxx" tabindex = "1">Link that is added long
  after the original list is created</a>
```

[153] https://www.w3.org/WAI/WCAG21/Techniques/html/H4.html

```
<a href="xxx" tabindex = "1">Third link in list</a>
  ...
<a href="xxx" tabindex = "1">Twentieth link in list</a>
```

Testing Procedure

1. Check if `tabindex` is used
2. If `tabindex` is used, check that the tab order specified by the `tabindex` attributes follows relationships in the content.

Technique C27: Making the DOM order match the visual order

Examples

{See Above}

Testing Procedure

{See Above}

- Changing a Web page dynamically using one of the following techniques:

Technique SCR26: Inserting dynamic content into the Document Object Model immediately following its trigger element[154]

Examples

This example creates a menu when a link is clicked and inserts it after the link. The onclick event of the link is used to call the ShowHide script, passing in an ID for the new menu as a parameter.

```
<a href="#" onclick="ShowHide('foo',this)">Toggle</a>
```

The ShowHide script creates a div containing the new menu, and inserts a link into it. The last line is the core of the script. It finds the parent of the element that triggered the script, and appends the div it created as a new child to it. This causes the new div to be in the DOM after the link. When the user hits tab, the focus will go to the first focusable item in the menu, the link we created.

```
function ShowHide(id,src)
{
	var el = document.getElementById(id);
	if (!el)
	{
		el = document.createElement("div");
		el.id = id;
		var link = document.createElement("a");
		link.href = "javascript:void(0)";
		link.appendChild(document.createTextNode("Content"));
		el.appendChild(link);
		src.parentElement.appendChild(el);
	}
	else
	{
		el.style.display = ('none' == el.style.display ? 'block' : 'none');
	}
}
```

CSS is used to make the div and link look like a menu.

Testing Procedure

1. Find all areas of the page that trigger dialogs that are not pop-up windows.
2. Check that the dialogs are triggered from the click event of a button or a link.
3. Using a tool that allows you to inspect the DOM generated by script, check that the dialog is next in the DOM.

[154] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR26.html

Technique SCR37: Creating Custom Dialogs in a Device Independent Way[155]

Example: An options button that opens a dialog

The HTML for this example includes a triggering Element, in this case a button, and a div that acts as the frame for the dialog.

The triggering element is a button and the script is triggered from the onclick event. This sends the appropriate events to the operating system so that assistive technology is aware of the change in the DOM.

In this example, the Submit and Reset buttons inside the dialog simply hide the div.

```
...
<button onclick="TogglePopup(event,true)"
        name="pop0001">Options</button>

<div class="popover" id="pop0001">
  <h3>Edit Sort Information</h3>
  <form action="default.htm" onsubmit="this.parentNode.style.display='none'; return false;"
onreset="this.parentNode.style.display='none'; return false;">
    <fieldset>
      <legend>Sort Order</legend>
      <input        type="radio"        name="order"        id="order_alpha"        /><label
for="order_alpha">Alphabetical</label>
      <input    type="radio"    name="order"    id="order_default"    checked="true"    /><label
for="order_default">Default</label>
    </fieldset>
<div class="buttons">
  <input type="submit" value="OK" />
  <input type="reset" value="Cancel" />
</div>
</form>

</div>
...
The div, heading and form elements are styled with CSS to look like a dialog.
...
a { color:blue; }
a.clickPopup img { border:none; width:0; }

div.popover  {  position:absolute;  display:none;  border:1px  outset;  background-color:beige;
font-size:80%; background-color:#eeeeee; color:black; }
div.popover h3 { margin:0; padding:0.1em 0.5em; background-color:navy; color:white; }
#pop0001 { width:20em; }
#pop0001 form { margin:0; padding:0.5em; }
#pop0001 fieldset { margin-bottom:0.3em; padding-bottom:0.5em; }
#pop0001 input, #pop0001 label { vertical-align:middle; }
#pop0001 div.buttons { text-align:right; }
#pop0001 div.buttons input { width:6em; }
...
The script toggles the display of the popup div, showing it and hiding it.
...
function TogglePopup(evt,show)
{
        HarmonizeEvent(evt);
        var src = evt.target;
        if ("click" == evt.type)
        {
                evt.returnValue = false;
        }
        var popID = src.getAttribute("name");
        if (popID)
        {
                var popup = document.getElementById(popID);
                if (popup)
                {
```

[155] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR37.html

```
                        if (true == show)
                        {
                                popup.style.display = "block";
                        }
                        else if (false == show)
                        {
                                popup.style.display = "none";
                        }
                        else
                        {
                                popup.style.display = "block" == popup.style.display ? "none" :
"block";
                        }
                        if ("block" == popup.style.display)
                        {
                                //window.alert(document.documentElement.scrollHeight);
                                popup.style.top  =  ((document.documentElement.offsetHeight  -
popup.offsetHeight) / 2 ) + 'px';
                                popup.style.left  =  ((document.documentElement.offsetWidth  -
popup.offsetWidth) / 2) + 'px';
                        }
                }
        }
}

function SubmitForm(elem)
{
        elem.parentNode.style.display='none';
        return false;
}

function ResetForm(elem)
{
        elem.parentNode.style.display='none';
        return false;
}
...
```

Testing Procedure

1. Find all areas of the page that trigger dialogs that are not pop-up windows.
2. Check that the dialogs can be opened by tabbing to the area and hitting enter.
3. Check that, once opened, the dialog is next in the tab order.
4. Check that the dialogs are triggered from the click event of a button or a link.
5. Using a tool that allows you to inspect the DOM generated by script, check that the dialog is next in the DOM.

Technique SCR27: Reordering page sections using the Document Object Model[156]

Examples

This example does up and down reordering. This approach can also be used for two-dimensional reordering by adding left and right options.

The components in this example are list items in an unordered list. Unordered lists are a very good semantic model for sets of similar items, like these components. The menu approach can also be used for other types of groupings. The modules are list items, and each module, in addition to content in div elements, contains a menu represented as a nested list.

```
<ul id="swapper">
    <li id="black">
        <div class="module">
            <div class="module_header">
```

[156] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR27.html

```
                <!-- menu link -->
                <a href="#" onclick="ToggleMenu(event);">menu</a>
                <!-- menu -->
                <ul class="menu">
                    <li><a href="#" onclick="OnMenuClick(event)"
                        onkeypress="OnMenuKeypress(event);">up</a></li>
                    <li><a href="#" onclick="OnMenuClick(event)"
                        onkeypress="OnMenuKeypress(event);">down</a></li>
                </ul>
            </div>
            <div class="module_body">
                Text in the black module
            </div>
        </div>
    </li>
    ...
</ul>
```

Since we've covered the showing and hiding of menus in the simple tree samples, we'll focus here just on the code that swaps the modules. Once we harmonize the events and cancel the default link action, we go to work. First, we set a bunch of local variables for the elements with which we'll be working: the menu, the module to be reordered, the menuLink. Then, after checking the reorder direction, we try to grab the node to swap. If we find one, we then call swapNode() to swap our two modules, and PositionElement() to move the absolutely-positioned menu along with the module, and then set focus back on the menu item which launched the whole thing.

```
function MoveNode(evt,dir)
{
    HarmonizeEvent(evt);
    evt.preventDefault();

    var src = evt.target;
    var menu = src.parentNode.parentNode;
    var module = menu.parentNode.parentNode.parentNode;
    var menuLink = module.getElementsByTagName("a")[0];
    var swap = null;

    switch(dir)
    {
        case 'up':
        {
            swap = module.previousSibling;
            while (swap && swap.nodeType != 1)
            {
                swap = swap.previousSibling;
            }
            break;
        }
        case 'down':
        {
            swap = module.nextSibling;
            while (swap && swap.nodeType != 1)
            {
                swap = swap.nextSibling;
            }
            break;
        }
    }
    if (swap && swap.tagName == node.tagName)
    {
        module.swapNode(swap);
        PositionElement(menu,menuLink,false,true);
    }
    src.focus();
}
```

The CSS for the node swap is not much different than that of our previous tree samples, with some size and color adjustment for the modules and the small menu.

```
ul#swapper { margin:0px; padding:0px; list-item-style:none; }
ul#swapper li { padding:0; margin:1em; list-style:none; height:5em; width:15em;
    border:1px solid black; }
ul#swapper li a { color:white; text-decoration:none; font-size:90%; }

ul#swapper li div.module_header { text-align:right; padding:0 0.2em; }
ul#swapper li div.module_body { padding:0.2em; }

ul#swapper ul.menu { padding:0; margin:0; list-style:none; background-color:#eeeeee;
    height:auto; position:absolute; text-align:left; border:1px solid gray; display:none; }
ul#swapper ul.menu li { height:auto; border:none; margin:0; text-align:left;
    font-weight:normal; width:5em; }
ul#swapper ul.menu li a { text-decoration:none; color:black; padding:0 0.1em;
    display:block; width:100%; }
```

Testing Procedure

1. Find all components in the Web Unit which can be reordered via drag and drop.
2. Check that there is also a mechanism to reorder them using menus build of lists of links.
3. Check that the menus are contained within the reorderable items in the DOM.
4. Check that scripts for reordering are triggered only from the onclick event of links.
5. Check that items are reordered in the DOM, not only visually.

2.4.4 Link Purpose (In Context) (Level A)[157] – This Success Criterion's mission is to simplify the process by which users can identify the purpose of each link and make a decision regarding whether or not to follow that link. When it is possible, provide text in the link that explains the function of the link without the need for any more explanation. Users of the website can have access to a list of the links on the page thanks to the usage of assistive technology. Users who want to select a link from this list will benefit from link wording that is as relevant as is possible to be as specific as possible. Those who wish to tab from link to link will also benefit from link content that contains useful information. Visitors can more easily determine which links to follow without having to resort to more laborious page-reading approaches if the connections have meaningful context.

Technique G91: Providing link text that describes the purpose of a link[158]

Example 1: Describing the purpose of a link in HTML in the text content of the <a> element

```
<a href="routes.html">
  Current routes at Boulders Climbing Gym
  </a>
```

Testing Procedure

For each link in the content that uses this technique:

1. Check that text of the link describes the purpose of the link

Technique H30: Providing link text that describes the purpose of a link for anchor elements[159]

Example 1

Describing the purpose of a link in HTML in the text content of the a element.

```
<a href="routes.html">
  Current routes at Boulders Climbing Gym
</a>
```

[157] https://www.w3.org/WAI/WCAG21/Understanding/link-purpose-in-context.html
[158] https://www.w3.org/WAI/WCAG21/Techniques/general/G91.html
[159] https://www.w3.org/WAI/WCAG21/Techniques/html/H30.html

Example 2
Using the alt attribute for the img element to describe the purpose of a graphical link.

```
<a href="routes.html">
  <img src="topo.gif" alt="Current routes at Boulders Climbing Gym">
</a>
```

Example 3
Using an empty alt attribute when the anchor (a) element contains text that describes the purpose of the link in addition to the img element. Note that the link text will appear on the page next to the image.

```
<a href="routes.html">
  <img src="topo.gif" alt="">
  Current routes at Boulders Climbing Gym
</a>
```

Example 4
A site allows users to provide feedback on products by clicking on the "Feedback" link in a product detail page. Other users are able to provide a response to any feedback. When a response to the user's feedback is available, the feedback link displays an icon, with "response received" as its text alternative, after the "Feedback" text. The site's help document describes this icon as a speech bubble containing quotation marks and includes the icon, with its text alternative, as an example. The same text alternative is used in the product detail pages (when a response is available) to help ensure that users following the documentation can identify the image correctly as documented.

```
<a href="prod_123_feedback.htm">
  Feedback
  <img src="response.gif" width="15" height="15" alt="response received">
</a>
```

Example 5
A link contains text and an icon, and the icon provides additional information about the target.

```
<a href="WMFP.pdf">
  Woodend Music Festival Program
  <img src="pdficon.gif" alt="PDF format">
</a>
```

Example 6
The "MyCorp" company's annual report is made available on the corporate website as a PDF file, and the annual corporate budget is made available as an Excel file on the web site.

Many users prefer to know the file type when opening a file that results in opening a new application to view the file, so it is often regarded as useful to include this additional information. However, this is not required for compliance with this Success Criterion.

```
<ul>
  <li>
    <a href="2009mycorp_report.pdf">MyCorp 2009 Annual Report (PDF)</a>
  </li>
  <li>
    <a href="2009mycorp_budget.xls">MyCorp 2009 Annual Budget (Excel)</a>
  </li>
</ul>
```

Testing Procedure
For each link in the content that uses this technique:
1. Check that text or a text alternative for non-text content is included in the `<a>` element.
2. If an img element is the only content of the `<a>` element, check that its text alternative describes the purpose of the link.
3. If the `<a>` element contains one or more img element(s) and the text alternative of the `img` element(s) is empty, check that the text of the link describes the purpose of the link.

4. If the `<a>` element only contains text, check that the text describes the purpose of the link.

Technique H24: Providing text alternatives for the area elements of image maps[160]

Examples

This example uses the alt attribute of the area element to provide text that describes the purpose of the image map areas.

```
<img src="welcome.gif" usemap="#map1"
    alt="Areas in the library. Select an area for
    more information on that area.">
<map id="map1" name="map1">
  <area shape="rect" coords="0,0,30,30"
    href="reference.html" alt="Reference">
  <area shape="rect" coords="34,34,100,100"
    href="media.html" alt="Audio visual lab">
</map>
```

Testing Procedure

For each `area` element in an image map:

1. Check that the `area` element has an `alt` attribute.
2. Check that the text alternative specified by the `alt` attribute serves the same purpose as the part of image map image referenced by the `area` element of the image map.

- Allowing the user to choose short or long link text using one of the techniques below:

Technique G189: Providing a control near the beginning of the Web page that changes the link text[161]

Example 1: Providing a Link to another Version

A Web page lists books for download in different formats. Alternate versions of the Web page use just the book format as the link text or the book title and format type.

Version with short link text:

```
...
  <h1>Books for download</h1>
  <p><a href="books-full-links.html" >Full link Version</a></p>
  <ul>
  <li>The History of the Web:
  <a href="history.docx" class="hist">Word</a>,
  <a href="history.pdf" class="hist">PDF</a>,
  <a href="history.html" class="hist">HTML</a>
  </li>
  ...
  </ul>
```

Version with full link text:

```
...
  <h1>Books for download</h1>
  <p><a href="books-short-links.html" >Short link Version</a></p>
  <ul>
  <li>The History of the Web:
  <a href="history.docx" class="hist">The History of the Web(Word)</a>,
  <a href="history.pdf" class="hist">The History of the Web(PDF)>/a>,
  <a href="history.html" class="hist">The History of the Web(HTML)</a>
  </li>
  ...
  </ul>
```

Testing Procedure

1. Check that there is a control near the beginning of the Web page to change link text.

[160] https://www.w3.org/WAI/WCAG21/Techniques/html/H24.html
[161] https://www.w3.org/WAI/WCAG21/Techniques/general/G189.html

2. Activate the control.
3. Check that all links in the resulting Web page have link text that describes their purpose.

Technique SCR30: Using scripts to change the link text[162]

Examples

This example uses Javascript to add contextual information directly to the text of a link. The link class is used to determine which additional text to add. When the "Expand Links" link is activated, each link on the page is tested to see whether additional text should be added.

```
...
<script type="text/javascript">
var expanded = false;
var linkContext = {
	"hist":" version of The History of the Web",
	"cook":" version of Cooking for Nerds"
};

function doExpand() {
	var links = document.links;

	for (var i=0; i<links.length; i++) {
		var link = links[i];
		var cn = link.className;
		if (linkContext[cn]) {
			span = link.appendChild(document.createElement("span"));
			span.setAttribute("class", "linkexpansion");
			span.appendChild(document.createTextNode(linkContext[cn]));
		}
	}
	objUpdate = document.getElementById('expand');
	if (objUpdate)
	{
		objUpdate.childNodes[0].nodeValue = "Collapse links";
	}
	expanded = true;
}

function doCollapse() {
	objUpdate = document.getElementById('expand');
	var spans = document.getElementsByTagName("span");
	var span;

	// go backwards through the set as removing from the front changes indices
	// and messes up the process
	for (i = spans.length - 1; i >= 0; i--) {
		span = spans[i];
		if (span.getAttribute("class") == "linkexpansion")
			span.parentNode.removeChild(span);
	}
	if (objUpdate)
	{
		objUpdate.childNodes[0].nodeValue = "Expand links";
	}
	expanded = false;
}

function toggle() {
	if (expanded) doCollapse();
	else doExpand();
}
</script>

...
```

[162] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR30.html

```
<h1>Books for download</h1>
<p><button id="expand" onclick="toggle();">Expand Links</button></p>
<ul>
       <li>The History of the Web: <a href="history.docx" class="hist">Word</a>, <a
href="history.pdf" class="hist">PDF</a>, <a href="history.html" class="hist">HTML</a> </li>

       <li>Cooking for Nerds: <a href="history.docx" class="cook">Word</a>, <a
href="history.pdf" class="cook">PDF</a>, <a href="history.html" class="cook">HTML</a> </li>
</ul>

...
```

Testing Procedure

1. Check that there is a link near the beginning of the page to expand links
2. Check that the link identified in step 1 can be identified from link text alone
3. Find any links on the page that cannot be identified from link text alone
4. Activate the control identified in step 1
5. Check that the purpose of the links identified in step 3 can now be identified from link text alone

Technique G53: Identifying the purpose of a link using link text combined with the text of the enclosing sentence[163]

Example 1

A Web page contains the sentence "To advertise on this page, click here."
Although the link phrase 'click here' is not sufficient to understand the link, the information needed precedes the link in the same sentence.

Example 2

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.

Testing Procedure

For each link in the content that uses this technique:

1. Check that the link is part of a sentence
2. Check that text of the link combined with the text of its enclosing sentence describes the purpose of the link

- Providing a supplemental description of the purpose of a link using one of the following techniques:

Technique H33: Supplementing link text with the title attribute[164]

Example 1: Clarifying the purpose of a link

```
<a href="http://example.com/WORLD/africa/kenya.elephants.ap/index.html"
   title="Read more about failed elephant evacuation">
   Evacuation Crumbles Under Jumbo load
</a>
```

Testing Procedure

Examine the source code for anchor elements.

1. For each anchor element that has a `title` attribute, check that the `title` attribute together with the link text describes the purpose of the link.

Technique C7: Using CSS to hide a portion of the link text[165]

Examples

The following examples use the CSS selector and rule set below:

[163] https://www.w3.org/WAI/WCAG21/Techniques/general/G53.html
[164] https://www.w3.org/WAI/WCAG21/Techniques/html/H33.html
[165] https://www.w3.org/WAI/WCAG21/Techniques/css/C7.html

```
.visually-hidden {
   clip-path: inset(100%);
   clip: rect(1px, 1px, 1px, 1px);
   height: 1px;
   overflow: hidden;
   position: absolute;
   white-space: nowrap;
   width: 1px;
}
```

Example 1
This example describes a news site that has a series of short synopsis of stories followed by a link that says "full story". Hidden link text describes the purpose of the link.

```
<p>Ishington has announced plans to stimulate economic growth.
  <a href="#"><span class="visually-hidden">Ishington stimulates economic growth </span>
  Full Story</a></p>
```

Example 2
This example describes a resource that has electronic books in different formats. The title of each book is followed by links that say "HTML" and "PDF." Hidden text describes the purpose of each link.

```
<dl>
<dt>Winnie the Pooh </dt>
   <dd><a href="winnie_the_pooh.html">
      <span class="visually-hidden">Winnie the Pooh </span>HTML</a></dd>
   <dd><a href="winnie_the_pooh.pdf">
         <span class="visually-hidden">Winnie the Pooh </span>PDF</a></dd>
<dt>War and Peace</dt>
    <dd><a href="war_and_peace.html">
      <span class="visually-hidden">War and Peace </span>HTML</a></dd>
    <dd><a href="war_and_peace.pdf">
      <span class="visually-hidden">War and Peace </span>PDF</a></dd>
</dl>
```

Testing Procedure
For each anchor element using this technique:
1. Check that an element has been defined that confines its display to a pixel and hides the text
2. Check that the element of that class is included in the content of the anchor
3. Check that the combined content of the anchor describes the purpose of the link

- Identifying the purpose of a link using link text combined with programmatically determined link context using one of the following techniques:

Technique ARIA7: Using aria-labelledby for link purpose[166]

Example 1: Providing additional information for links
This example will mean that the link text as shown on screen is then used as the start of the accessible name for the link. Screen readers will announce this as: "Read more ...Storms hit east coast" and will display that same text in the links list which is very useful for screen reader users who may browse by links.

```
<h2 id="headline">Storms hit east coast</h2>

<p>Torrential rain and gale force winds have struck the east coast,
   causing flooding in many coastal towns.
<a id="p123" href="news.html" aria-labelledby="p123 headline">Read more...</a></p>
```

Example 2: Concatenating link text from multiple sources
There may be cases where an author will place a tag around a section of code that will be referenced.

[166] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA7.html

```
<p>Download <span id="report-title">2012 Sales Report</span>:
<a aria-labelledby="report-title pdf" href="#" id="pdf">PDF</a> |
<a aria-labelledby="report-title doc" href="#" id="doc">Word</a> |
<a aria-labelledby="report-title ppt" href="#" id="ppt">Powerpoint</a></p>
```

Testing Procedure

For each link that has an `aria-labelledby` attribute:

1. Check that each ID in the value of the `aria-labelledby` attribute matches an ID of a text element used as part of the link purpose.
2. Check that the combined value of the text referenced by the one or more ID's in `the aria-labelledby` attribute properly describes the purpose of the link element.

Technique ARIA8: Using aria-label for link purpose[167]

Example 1: Describing the purpose of a link in HTML using aria-label.

In some situations, designers may choose to lessen the visual appearance of links on a page by using shorter, repeated link text such as "read more". These situations provide a good use case for aria-label in that the simpler, non-descriptive "read more" text on the page can be replaced with a more descriptive label of the link. The words 'read more' are repeated in the aria-label (which replaces the original anchor text of "[Read more...]") to allow consistent communication between users.

```
<h4>Neighborhood News</h4>
 <p>Seminole tax hike:  Seminole city managers are proposing a 75% increase in
 property taxes for the coming fiscal year.
 <a href="taxhike.html" aria-label="Read more about Seminole tax hike">[Read more...]</a>
 </p>

 <p>Baby Mayor:  Seminole voters elect the city's youngest mayor ever by voting in 3 year
 old Willy "Dusty" Williams in yesterday's mayoral election.
 <a  href="babymayor.html"  aria-label="Read  more  about  Seminole's  new  baby  mayor">[Read
more...]</a>
 </p>
```

Testing Procedure

For link elements that use `aria-label`:

1. Check that the value of the `aria-label` attribute properly describes the purpose of the link element.

Technique H77: Identifying the purpose of a link using link text combined with its enclosing list item[168]

Example 1

```
<ul>
  <li>
    Check out the video report for last year's
    <a href="festival.htm">National Folk Festival</a>.
  </li>
  <li>
    <a href="listen.htm">Listen to the instruments</a>
  </li>
  <li>
    Guitar Man: George Golden talks about
    <a href="mkguitars.htm">making guitars</a>.
  </li>
</ul>
```

Example 2: A list of video games for download

```
<ul>
  <li>
```

[167] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA8.html
[168] https://www.w3.org/WAI/WCAG21/Techniques/html/H77.html

```
    <a href="tomb_raider.htm">Tomb Raider: Legend</a>
    <a href="tomb_raider_images.htm">See Images</a>
    <a href="tomb_raider.mpeg">(Download Demo)</a>
  </li>
  <li>
    <a href="fear_extraction.htm">F.E.A.R. Extraction Point</a>
    <a href="fear_extraction_images.htm">See Images</a>
    <a href="fear_extraction.mpeg">(Download Demo)</a>
  </li>
  <li>
    <a href="call_of_duty.htm">Call of Duty 2</a>
    <a href="call_of_duty_images.htm">See Images</a>
    <a href="call_of_duty.mpeg">(Download Demo)</a>
  </li>
  <li>
    <a href="Warhammer 40K.htm">Warhammer 40K</a>
    <a href="warhammer_40k_images.htm">See Images</a>
    <a href="Warhammer_40k.mpeg">(Download Demo)</a>
  </li>
</ul>
```

Testing Procedure

For each link in the content that uses this technique:

1. Check that the link is part of a list item.
2. Check that text of the link combined with the text of its enclosing list item describes the purpose of the link.

Technique H78: Identifying the purpose of a link using link text combined with its enclosing paragraph[169]

Example 1: Announcements content on a folk festival Web page

```
<h3>The final 15</h3>
<p>
  Coming soon to a town near you: the final 15 in the
  National Folk Festival lineup.
  <a href="final-15.html">Read more</a>
</p>

<h3>Folk artists get awards</h3>
<p>
  Performers from the upcoming National Folk Festival receive
  National Heritage Fellowships.
  <a href="national-heritage.html">Read more</a>
</p>
```

Testing Procedure

For each link in the content that uses this technique:

1. Check that the link is part of a paragraph.
2. Check that text of the link combined with the text of its enclosing paragraph describes the purpose of the link.

Technique H79: Identifying the purpose of a link in a data table using the link text combined with its enclosing table cell and associated table header cells[170]

Example 1: A table of rental car choices

```
<table>
  <caption>Available rental cars with cost per day</caption>
  <tr>
    <th>Type of car</th>
    <th>Alamo</th>
    <th>Budget</th>
    <th>National</th>
```

[169] https://www.w3.org/WAI/WCAG21/Techniques/html/H78.html
[170] https://www.w3.org/WAI/WCAG21/Techniques/html/H79.html

```
    <th>Avis</th>
    <th>Hertz</th>
  </tr>
  <tr>
    <th scope="row">Economy</th>
    <td><a href="econ-ala.html">$67</a></td>
    <td><a href="econ-bud.html">$68</a></td>
    <td><a href="econ-nat.html">$72</a></td>
    <td><a href="econ-av.html">$74</a></td>
    <td><a href="econ-hz.html">$74</a></td>
  </tr>
  <tr>
    <th scope="row">Compact</th>
    <td><a href="comp-ala.html">$68</a></td>
    <td><a href="comp-bud.html">$69</a></td>
    <td><a href="comp-nat.html">$74</a></td>
    <td><a href="comp-av.html">$76</a></td>
    <td><a href="comp-hz.html">$76</a></td>
  </tr>
  <tr>
    <th scope="row">Mid-sized</th>
    <td><a href="mid-ala.html">$79</a></td>
    <td><a href="mid-bud.html">$80</a></td>
    <td><a href="mid-nat.html">$83</a></td>
    <td><a href="mid-av.html">$85</a></td>
    <td><a href="mid-hz.html">$85</a></td>
  </tr>
  <tr>
    <th scope="row">Full-sized</th>
    <td><a href="full-ala.html">$82</a></td>
    <td><a href="full-bud.html">$83</a></td>
    <td><a href="full-nat.html">$89</a></td>
    <td><a href="full-av.html">$91</a></td>
    <td><a href="full-hz.html">$91</a></td>
  </tr>
</table>
```

Testing Procedure

For each link in the content that uses this technique:

1. Check that the link is in a table cell.
2. Check that text of the link combined with the text of the associated table header cells describes the purpose of the link.

Technique H81: Identifying the purpose of a link in a nested list using link text combined with the parent list item under which the list is nested[171]

Example 1: A document provided in three formats

```
<ul>
  <li>Annual Report 2021
    <ul>
      <li>
        <a href="ar-2021.html"><abbr title="HyperText Markup Language">HTML</abbr></a>
      </li>
      <li>
        <a href="ar-2021.pdf"><abbr title="Portable Document Format">PDF</abbr></a>
      </li>
      <li>
        <a href="ar-2021.rtf"><abbr title="Rich Text Format">RTF</abbr></a>
      </li>
    </ul>
  </li>
  <li>Annual Report 2022
    <ul>
```

[171] https://www.w3.org/WAI/WCAG21/Techniques/html/H81.html

```
      <li>
        <a href="ar-2022.html">HTML</a>
      </li>
      <li>
        <a href="ar-2022.pdf">PDF</a>
      </li>
      <li>
        <a href="ar-2022.rtf">RTF</a>
      </li>
    </ul>
  </li>
</ul>
```

Example 2: Blocks of information about hotels
The information for each hotel consists of the hotel name, a description and a series of links to a map, photos, directions, guest reviews and a booking form.

```
<nav>
  <ul>
    <li><a href="royal-palm-hotel.html">Royal Palm Hotel</a>
      <ul>
        <li><a href="royal-palm-hotel-map.html">Map</a></li>
        <li><a href="royal-palm-hotel-photos.html">Photos</a></li>
        <li><a href="royal-palm-hotel-directions.html">Directions</a></li>
        <li><a href="royal-palm-hotel-reviews.html">Guest reviews</a></li>
        <li><a href="royal-palm-hotel-book.html">Book now</a></li>
      </ul>
    </li>
    <li><a href="hotel-three-rivers.html">Hotel Three Rivers</a>
      <ul>
        <li><a href="hotel-three-rivers-map.html">Map</a></li>
        <li><a href="hotel-three-rivers-photos.html">Photos</a></li>
        <li><a href="hotel-three-rivers-directions.html">Directions</a></li>
        <li><a href="hotel-three-rivers-reviews.html">Guest reviews</a></li>
        <li><a href="hotel-three-rivers-book.html">Book now</a></li>
      </ul>
    </li>
  </ul>
</nav>
```

Testing Procedure
For each nested link in the content that uses this technique:
1. Find the link's parent `ul` or `ol` element.
2. Check that this list element (`ul, ol`) is a descendant of `an li` element.
3. Check that the text of the link combined with the text of that parent `li` element describes the purpose of the link.

2.4.5 Multiple Ways (Level AA)[172] – It is the purpose of this Success Criterion to make it possible for users to locate content in a manner that is tailored specifically to meet their needs. Users of one approach could find that utilizing it is easier or more intuitive than utilizing another method. Even the smallest websites should provide some type of assistance to users in navigating the site. For a website that consists of three or four pages and in which each page is connected from the home page, it may be adequate to simply offer links from and to the main page. This is because the links on the home page can also function as a site map for the website.

- Using two or more of the following techniques

Technique G125: Providing links to navigate to related Web pages[173]

[172] https://www.w3.org/WAI/WCAG21/Understanding/multiple-ways.html
[173] https://www.w3.org/WAI/WCAG21/Techniques/general/G125.html

Examples
The Web Content Accessibility Guidelines 2.0 contains links to definitions of terms used in guidelines and Success Criteria, links to documents explaining how to meet different Success Criteria, a table of contents for each section containing links to different subsections of that section, and a Comparison of WCAG 1.0 checkpoints to WCAG 2.0. As users browse the document, they can follow these links to find related information.

Testing Procedure
For each link in the Web page:
1. Check whether the link leads to related information.

Technique G64: Providing a Table of Contents[174]

Example 1
The Web Content Accessibility Guidelines 2.0 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.

Example 2
The table of contents for Accessing PDF Documents with Assistive Technology: A Screen Reader User's Guide begins on the second page.

Testing Procedure
1. Check that a table of contents or a link to a table of contents exists in the document.
2. Check that the values and order of the entries in the table of contents correspond to the names and order of the sections of the document.
3. Check that the entries in the table of contents link to the correct sections of the document.

Technique G63: Providing a site map[175]

Example 1
The Web Accessibility Initiative provides a WAI site map that lists different sections of its Web site. The site map shows the different sections of the Web site, and shows some of the substructure within those sections.

Example 2
The site map for an on-line magazine lists all the sections of the magazine and the subsections in each section. It also include links for Help, How to Contact Us, Privacy Policy, Employment Opportunities, How to Subscribe, and the home page for the magazine.

Testing Procedure
1. Check that the site contains a site map.
2. Check that the links in the site map leads to the corresponding sections of the site.
3. For each link in the site map, check that the target page contains a link to the site map.
4. For each page in the site, check that the page can be reached by following some set of links that start at the site map.

Technique G161: Providing a search function to help users find content[176]

Example 1: A Shopping Site
A shopping site organizes its products into different categories, such as women's clothes, men's clothes, and children's clothes. These have subcategories, such as tops, pants, shoes, and accessories. Each page also contains a search form. Users can type the product number or product description into the search field and go directly to that product, rather than needing to navigate the product categories to find it.

[174] https://www.w3.org/WAI/WCAG21/Techniques/general/G64.html
[175] https://www.w3.org/WAI/WCAG21/Techniques/general/G63.html
[176] https://www.w3.org/WAI/WCAG21/Techniques/general/G161.html

Example 2: A Help Center
A Help Center contains thousands of pages of Help information about a company's products. A search form allows users to search just the Help Center pages to find articles that contain the search terms.

Testing 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

Technique G126: Providing a list of links to all other Web pages[177]

Example 1
A family Web site contains home pages for all the members of the family. Each page contains a list of links to the home pages of the other family members.

Example 2
An electonic book is broken into separate Web pages for each chapter. Each Web page starts with a small table of contents that contains links to all the chapters in the book.

Testing Procedure
1. Check that each Web page contains a list of links to the other Web pages in the site
2. Check that the links in the list lead to the corresponding Web pages.
3. Check that the list contains a link for every Web page in the site.

Technique G185: Linking to all of the pages on the site from the home page[178]

Examples
- A small commercial Web site for a consultant contains a home page, a Contacts page for contacting the consultant, a page describing the consultant's background, and a page with examples of the consultant's work. Each page contains a navigation bar that links to all the other pages in the site.

Testing Procedure
1. Check that the home page contains links to all other pages in the Web site.
2. Check that all other pages in the Web site contain links to the home page.

2.4.6 Headings and Labels (Level AA)[179] – The purpose of this Success Criterion is to simplify the process by which individuals may comprehend the content that can be found on websites and how it is organized on those websites. When users encounter headings that are both clear and comprehensive, it is much simpler for them to find the information they require and to comprehend the relationships that exist between the various parts of the text. By assigning descriptive labels to individual content elements, users will have an easier time recognizing those elements. There is no necessity for labels or titles that are very lengthy. A single letter, a single word, or even both may be adequate if the information provides a helpful hint for discovering and accessing material.

Technique G130: Providing descriptive headings[180]

[177] https://www.w3.org/WAI/WCAG21/Techniques/general/G126.html
[178] https://www.w3.org/WAI/WCAG21/Techniques/general/G185.html
[179] https://www.w3.org/WAI/WCAG21/Understanding/headings-and-labels.html
[180] https://www.w3.org/WAI/WCAG21/Techniques/general/G130.html

Example 1
An HTML page that describes the range of tasks for disaster preparation may have the following headings:

```
<h1>Disaster preparation</h1>
  <h2>Flood preparation</h2>
  <h2>Fire preparation</h2>
```

Note that the level 2 headings have the distinguishing information at the beginning (ie, instead of "Preparation for floods", "Preparation for fires", etc).

Example 2
A short article about the history of a town that explains about the founding and expansion of the town and then goes into some depth about the current situation. The title of the Web page is "History of Ourtown". The first section is called "The founding of Ourtown". The second section is called "Expansion of Ourtown". The third section is called "Ourtown today" which has the following subsections: "People in Ourtown", "Organizations in Ourtown" and "Buildings in Ourtown".

Testing Procedures
1. Determine if the Web page contains headings.
2. Check that each heading identifies its section of the content.

Technique G131: Providing descriptive labels[181]

Example 1: Online maps with controls for zooming in and out
A Web application presents maps of a city. Users can “zoom in" to view part of the map in greater detail and can “zoom out" to make it show a larger part of the city. The controls can be operated using either a mouse or a keyboard. The controls are labeled “Zoom in (Ctrl + Shift + L)" And “Zoom out (Ctrl + Shift + R)."

Example 2: A form asking the name of the user
A form asks the name of the user. It consists of two input fields to ask for the first and last name. The first field is labeled "First name", the second is labeled "Last name".

Example 3: A form with required fields
A purchasing form includes several fields that are required. In addition to identifying the field, the label for each required field includes the word “required" in parentheses.

Testing Procedures
For each interface component with a label:
1. Identify the purpose of the interface component.
2. Check that each label makes the component's purpose clear.

2.4.7 Focus Visible (Level AA)[182] – The purpose of this Success Criterion is to make it clear which component has the keyboard emphasis at any given time. It is necessary for a person to be able to identify, from among a number of different elements, which one constitutes the keyboard emphasis. Since the visual design only shows one keyboard actionable item, the success condition would be met if there is just one keyboard actionable control displayed on the screen. The term "mode of operation" takes into account the possibility that some user agents may either not display a focus indication at all times or will only display it when the keyboard is being used. User agents have the ability to optimize the amount of time that the focus indicator is displayed by doing things like showing it just when a keyboard is being used. According to the authors, the focal point must to be obvious in at least one of the modes of operation. This success criterion can be applied because there is typically just one way to carry out the task at hand. It is not possible for the focus indicator to be transitory while the keyboard is displayed as

[181] https://www.w3.org/WAI/WCAG21/Techniques/general/G131.html
[182] https://www.w3.org/WAI/WCAG21/Understanding/focus-visible.html

having the focus. Take note that the focus indication on the keyboard can be configured in a variety of different ways.

Technique G149: Using user interface components that are highlighted by the user agent when they receive focus[183]

Examples

- When html text input fields receive focus, browsers display a blinking vertical bar at the insertion point in the text field.
- When html links receive focus, they are surrounded by a dotted focus highlight rectangle.

Testing Procedure

For each focusable component in the Web page:

1. Set focus to the control
2. Check whether the user agent has highlighted the control in some way

Technique C15: Using CSS to change the presentation of a user interface component when it receives focus[184]

Example 1: Link elements

In this example, mouse and keyboard focus indicators have been applied to the link elements. CSS has been used to apply a background color when the link elements receive focus.

Here is the content to be displayed:

```
<ul id="mainnav">
  <li class="page_item">Home</li>
  <li class="page_item"><a href="/services">Services</a></li>
  <li class="page_item"><a href="/projects">Projects</a></li>
  <li class="page_item"><a href="/demos">Demos</a></li>
  <li class="page_item"><a href="/about-us">About us</a></li>
  <li class="page_item"><a href="/contact-us">Contact us</a></li>
  <li class="page_item"><a href="/links">Links</a></li>
</ul>
```

Here is the CSS that changes the background color for the above elements when they receive mouse or keyboard focus:

```
#mainnav a:hover, #mainnav a:active, #mainnav a:focus {
  background-color: #DCFFFF;
  color:#000066;
}
```

Example 2: Highlighting elements that receive focus

In this example, the :focus pseudo-class is used to change the style applied to input fields when they receive focus by changing the background color.

```
<html>
  <head>
    <style type="text/css">
      input.text:focus {
        background-color: #7FFF00;
        color: #000;
      }
      input[type=checkbox]:focus + label, input[type=radio]:focus + label {
        background-color: #1E2EB8;
        color: #FFF;
      }
    </style>
  </head>
```

[183] https://www.w3.org/WAI/WCAG21/Techniques/general/G149.html
[184] https://www.w3.org/WAI/WCAG21/Techniques/css/C15.html

```
  <body>
    <form method="post" action="form.php">
      <p><label for="fname">Name: </label>
        <input class="text" type="text" name="fname" id="fname" />
      </p>
      <p>
        <input type="radio" name="sex" value="male" id="sm" /> <label for="sm">Male</label><br
/>
        <input type="radio" name="sex" value="female" id="sf" /> <label for="sf">Female</label>
      <p>
    </form>
  </body>
</html>
```

Testing Procedure
For each element able to attain focus:
1. Using a mouse, hover over the element.
2. Check that the background or border changes color.
3. Move the mouse away from the object before attempting keyboard focus.
4. Using a keyboard, tab to the element.
5. Check that the background or border changes color.
6. Check that the background or border changes in color are removed when the element loses focus.

Technique G165: Using the default focus indicator for the platform so that high visibility default focus indicators will carry over[185]

Example 1
The default focus indicator on Microsoft Windows is a one-pixel, black dotted line around the focused element. On a page with a dark background, this can be very difficult to see. The creator of the page uses the default, and the user customizes it in Windows to make it a bright color.

Example 2
In HTML, form elements and links can be focused by default. In addition, any element with a `tabindex` attribute >= 0 can take focus. Both types of focused elements use the system focus indicator and will pick up platform changes in the focus indicator style.

Testing Procedure
1. Use the features of your platform to customize the appearance of the focus indicator
2. Tab through the page, noting the path of the focus
3. Check that the focus indicator for each control is visible

Technique G195: Using an author-supplied, visible focus indicator

Example
{See Above}

Testing Procedure
{See Above}

Technique SCR31: Using script to change the background color or border of the element with focus[186]

Examples
In this example, when the link receives focus, its background turns yellow. When it loses focus, the yellow is removed. Note that if the link had a background color to begin with, you would use that color rather than "" in the script.

```
...
<script>
```

[185] https://www.w3.org/WAI/WCAG21/Techniques/general/G165.html
[186] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR31.html

```
 function toggleFocus(el)
 {
  el.style.backgroundColor =  el.style.backgroundColor=="yellow" ? "inherit" : "yellow";
 }
</script>

...

<a href="example.html" onfocus="toggleFocus(this)" onblur="toggleFocus(this)">focus me</a>
...
```

Testing Procedure

1. Tab to each element in the page
2. Check that the focus indicator is visible

2.5 Input Modalities[187] – This section works to make it easier for individuals to operate the target site's functionality through various inputs beyond the keyboard. There are four criteria points to reach level A conformance. For this section, there is not an option for level AA conformance – only A or AAA.

2.5.1 Pointer Gestures (Level A)[188] – This Success Criterion is developed to ensure that the material could be manipulated by utilizing a variety of pointing devices, cognitive abilities, and assistive technologies. Some people have difficulty producing exact motions, thus they might use an input device that is developed specifically for them, such as a head pointer, an eye-gaze system, or a speech-controlled mouse emulator. Alternatively, they might just use a computer mouse. Some pointing approaches lack the capability of performing multipoint or path-based gestures, and others lack the accuracy necessary to do so. Path-based gestures require an engagement in which the endpoints themselves are not the only significant elements. A gesture is considered to be path-based if its meaning shifts even after passing through an intermediary location, which is usually located in close proximity to the commencement of the gesture. The user initiates the process by establishing a pointer as the starting point, then navigates to at least one other site before finally letting go of the pointer (end point). Even though the complete path is not specified, the gesture can be characterized as having requirements for a certain path based on the intermediate location.

• GXXX: Do not rely on path-based gestures
• GXXX: Do not rely on multipoint gestures
• GXXX: Provide controls that do not require complex gestures and perform the same function as complex gestures
• GXXX: Single-point activation for spatial positioning and manipulation

2.5.2 Pointer Cancellation (Level A)[189] – Making it easier for users to avoid unintended or inaccurate pointer input is the aim of this Success Criterion. The unintentional initiation of touch or mouse activities that result in undesired results can happen to people with a variety of disabilities. The methods for permitting users to cancel pointer operations are explained in each of the subsections that follow, and they generally match to this Success Criterion's bullet points.

Technique G210: Ensuring that drag-and-drop actions can be cancelled[190]

[187] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#input-modalities

[188] https://www.w3.org/WAI/WCAG21/Understanding/pointer-gestures.html

[189] https://www.w3.org/WAI/WCAG21/Understanding/pointer-cancellation.html

[190] https://www.w3.org/WAI/WCAG21/Techniques/general/G210.html

Examples

- A site shows a file directory. Files can be picked up and moved over a trash can icon. When the picked-up file is released outside this target, it reverts to the old position.
- A site shows a file directory. Files can be picked up and moved over a trash can icon to delete it. When the picked-up file is released over the trash can, a modal dialog asks the user to confirm or cancel the delete action.
- A kanban implementation of a project planning site shows different columns for phases of an activity. Users can pick up and move icons representing planning items to another column. When an item has been picked up and now follows the pointer, it can be moved outside the drop targets (columns) and dropped there to cancel the action. The item will then jump back to the old position.
- A kanban implementation of an issue tracking system shows columns that indicate different phases in handling issues (such as new / processed / done / closed). There is no screen space outside the kanban columns. Users can pick up and move icons representing issues between columns. When an item has been dropped in another column, the action can be reversed by dragging the icon back to the original column where it will return to its original position (defined by sorting preferences).

Testing Procedure

For content that is draggable, check whether the drag-and-drop action can be reversed by:

1. releasing the picked-up item outside a drop target
2. dragging the picked-up item back to its old position
3. a confirmation dialog or an undo control appears after the item has been dropped

2.5.3 Label in Name (Level A)[191] – The purpose of this Success Criterion is to ensure that the words used to visually identify a component are the same terms that are used to identify it programmatically. This is accomplished by comparing the two sets of words. Because of this, it is possible for those with impairments to interact with the components by depending on labeling that is easily readable. The bulk of the controls provide a text label that is easy to read. The accessible name, which is often referred to as the programmatic name, is given to the controls that are present. Users typically have a far more positive experience all around when the visible label of a control's words and characters match or contain the accessible name of the control.

Technique G208: Including the text of the visible label as part of the accessible name[192]

Example 1: Link text matches the beginning of the accessible name

A link contains visible text and hidden link text. Both together make up the link's accessible name. The visible text comes first. The idea is to make the link more descriptive for users of assistive technologies.

```
<p>Go to <a href="code-of-conduct.html">Code of conduct <span class="hidden_accessibly"> of ACME
Corporation</span></a><p>
```

Example 2: Generic link text concatenated with heading

A generic link is combined with the heading of the paragraph to give context. It is a variation on the first example, this time using aria-labelledby. The advantage of this implementation is that it uses existing visible text on the page, and so is more likely to be properly translated during any localization transformations.

```
<h4 id="poor">Insufficient Link Names Invade Community</h4>
<p>Citizens are reeling from the growing invasion of useless "read more" links appearing in
their online resources. <a href="poor.html" aria-labelledby="generic poor"><span
id="generic">More...</span></a>
```

Example 3: Link text included in aria-label

Where two strings cannot be grammatically or seamlessly combined using aria-labelledby, aria-label can be used to make a new name which includes the visible label.

[191] https://www.w3.org/WAI/WCAG21/Understanding/label-in-name.html
[192] https://www.w3.org/WAI/WCAG21/Techniques/general/G208.html

```
...end of news story. <a href="poor.html" aria-label="Read more about Insufficient link
names">Read more</a>
```

Example 4: The visible button text matches the beginning of the accessible name
The visible text inside a button element matches the beginning of accessible name, which also includes hidden text. The idea of the hidden text is to make the button more descriptive for users of assistive technologies.

```
<button>Send <span class="accessibly-hidden"> Mail</span></button>
```

Testing Procedure
For all controls with a visible label, check that:
1. The accessible name of the control contains the same letters in the same order as the visible label.

2.5.4 Motion Actuation (Level A)[193] – This Success Criterion's goal is to make sure that functions that are activated by moving a device—for example, shaking or tilting—or by gesturing toward it—so that sensors like a camera may pick up and interpret the gesturing—can also be controlled by more traditional user interface elements. Sensors that can be used as inputs are frequently found in devices, such as the accelerometer and gyroscope sensors on a phone or tablet. The user may be able to control things with these sensors by simply rotating the gadget or moving it in specific directions. In other cases, web content can employ the camera or other sensors to read user motions and activate features. For instance, shaking the device may send a "Undo" instruction, or a light hand wave could be used to forward or reverse a series of pages. Because the device is mounted on a stationary object (like a wheelchair), certain disabled users with disabilities are unable to operate these device sensors (either completely or accurately). As a result, any functionality provided by motion must also be provided by another mechanism.

Technique G213: Provide conventional controls and an application setting for motion activated input[194]

Example 1: Shake to undo
After text is entered in a field, shaking a device shows a dialog offering users to undo the entry. Supporting use of the backspace key and/or providing a clear button next to the text field offers the same functionality.
Shake to undo can be turned off in a settings page.

Example 2: Motion Activated Slider
A slider can be adjusted by tipping the device to the left and right. There are also buttons to achieve the same functionality, and a tick-box that prevents the motion from having an effect.

Testing Procedure
For each input that performs a function in response to a motion sensor:
1. Check that there is a mechanism to perform the same function that does not rely on a sensor.
2. Check that there is a user setting which disables the motion detection.

GXXX: Supporting system level features which allow the user to disable motion actuation

Section 3 – Understandable

[193] https://www.w3.org/WAI/WCAG21/Understanding/motion-actuation.html
[194] https://www.w3.org/WAI/WCAG21/Techniques/general/G213.html

3.1 Readable[195] – This purpose of this section's guidelines is to make sure that the target site's text content is readable and understandable. There are two criteria points for level AA conformance.

3.1.1 Language of Page (Level A)[196] – The purpose of this Success Criterion is to ensure that web page content providers include the necessary data for user agents to correctly display text and other linguistic content on the web pages they create. Once the language of the website has been identified, both assistive technology and regular user agents will be able to render text with a higher degree of accuracy. The criteria for correct pronunciation can be loaded for those using screen readers. Visual browsers are capable of accurately displaying both characters and scripts. Media players will display the captions in the correct format. As a direct consequence of this change, content will be more accessible to users with various kinds of impairments.

Technique H57: Using the language attribute on the `HTML` element[197]
Example 1: Defining the content of an HTML document to be in French

```
<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>document écrit en français</title>
</head>
<body>
  ... document écrit en français ...
</body>
</html>
```

Testing Procedure
Examine the `html` element of the document.
1. Check that the `html` element has a `lang` attribute.
2. Check that the value of the lang attribute conforms to BCP 47: Tags for the Identification of Languages or its successor and reflects the primary language used by the Web page.

3.1.2 Language of Parts (Level AA)[198] – The purpose of this Success Criterion is to provide assurance that user agents are able to correctly provide content that has been written in more than one language. This makes it possible for user agents and assistive technologies to display content in a manner that conforms to the presentation and pronunciation rules of the language in question. This holds true for graphical browsers, voice browsers, braille displays, and screen readers equally. If the language of each section of text can be identified, conventional user agents, as well as assistive technologies, will be able to produce more accurate renderings of the content. Screen readers are able to make use of the pronunciation norms of the language used in the text. Visual browsers make it possible to display characters and scripts within their appropriate environments. When text is displayed in a language that uses a different alphabet, or when moving between languages that read from left to right and languages that read from right to left, this is an extremely important consideration. Users with disabilities who are fluent in all of the languages that are used on the website will have an easier time understanding the content once each component of the page has been correctly displayed.

[195] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#readable
[196] https://www.w3.org/WAI/WCAG21/Understanding/language-of-page.html
[197] https://www.w3.org/WAI/WCAG21/Techniques/html/H57.html
[198] https://www.w3.org/WAI/WCAG21/Understanding/language-of-parts.html

Technique H58: Using language attributes to identify changes in the human language[199]

Example 1: The use of the lang attribute to define a quote written in German

```
<blockquote lang="de">
  <p>
    Da dachte der Herr daran, ihn aus dem Futter zu schaffen,
    aber der Esel merkte, daß kein guter Wind wehte, lief fort
    und machte sich auf den Weg nach Bremen: dort, meinte er,
    könnte er ja Stadtmusikant werden.
  </p>
</blockquote>
```

Testing Procedure

For each element in the document:

1. Check that the human language of the content of the element is the same as the inherited language for the element as specified in HTML - The lang and xml:lang attributes.

For each `lang` attribute in the document:

1. Check that the value of the lang attribute conforms to BCP 47: Tags for the Identification of Languages or its successor.
2. Check that the language code matches the language of the content it applies to.

3.2 Predictable[200] – This section worked to make the target site's web pages appear and operate in ways that are predictable. In order to reach level AA conformance, there are four criteria points that had to be addressed.

3.2.1 On Focus (Level A)[201] – As readers navigated through a document, the purpose of this Success Criterion is to ensure that the functionality remained consistent and predictable. When a component is given the attention, that component should not be given the ability to change the context in any way. When one component gets focus, the context around it can vary in a number of ways, including but not limited to: forms submitting themselves when that component gets focus; new windows opening; and focus moving to a different component when that component gets focus. It is possible to change the focus to a different control by using either the mouse or the keyboard (e.g., clicking on a text field). If this behavior is not supported by scripting, moving the mouse over a control will not cause it to switch the attention to that control. Keep in mind that activating some control types (such as a button) by clicking on them can also trigger a change in the context of the current screen.

Technique G107: Using "activate" rather than "focus" as a trigger for changes of context[202]

Examples

- A page pops up a new window only when the user clicks (or uses spacebar) on a button rather than using onfocus to pop up a new window.
- A submit button is used to move on to the next data entry screen rather than having the next screen appear automatically when the user tabbed onto a 'done' button.

Testing Procedure

1. Using a keyboard, cycle focus through all content

[199] https://www.w3.org/WAI/WCAG21/Techniques/html/H58.html
[200] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#predictable
[201] https://www.w3.org/WAI/WCAG21/Understanding/on-focus.html
[202] https://www.w3.org/WAI/WCAG21/Techniques/general/G107.html

2. Check that no changes of context occur when any component receives focus.

** NOTE: A change of content is not always a change of context. This success criterion is automatically met if changes in content are not also changes of context.**

3.2.2 On Input (Level A)[203] – The purpose of this Success Criterion is to ensure that the results of selecting a form control or entering data are consistent and easy to anticipate. Any part of the user interface that can have its settings modified also has a corresponding update to the control that will remain in place even after the user has ceased using the component. Therefore, activating it requires clicking a link or a button, but selecting a checkbox, entering text into a text field, or altering the item that is chosen in a list control do not modify its settings. Users who are unable to rapidly identify the shift or who are easily distracted by changes may become confused as a result of changes in context. It can be utilized if and only if it is clear that the activity of the user will cause a discernible shift in the context of the conversation.

Technique G80: Providing a submit button to initiate a change of context[204]

Examples

- Example 1: A submit button is used for each form that causes a change in context.

Testing Procedure

1. Find all forms in the content
2. For each form, check that it has a submit button

Technique H32: Providing submit buttons[205]

Example 1: A basic example of a form with a submit button

```
<form action="/subscribe" method="post">
 <p>Enter your email address to subscribe to our mailing list.</p>
 <label for="address">Your email address:</label>
 <input autocomplete="email" id="address" name="address" type="text">
 <input type="submit" value="Subscribe">
</form>
```

Testing Procedure

1. Find all forms in the content.
2. For each form, check that it has a submit button (`<input type="submit">`, `<input type="image">`, or `<button type="submit">`).

Technique H84: Using a button with a select element to perform an action[206]

Example 1: A calendar

A Web page lets the user choose a quarter of any year and display the calendar for those months. After the user has set the quarter and year, they display the calendar by pressing the "Show" button. This example relies on client-side scripting to implement the action.

```
<label for="quarter">Quarter:</label>
<select id="quarter" name="quarter">
  <option value="1">Quarter 1 (January – March)</option>
  <option value="2">Quarter 2 (April – June)</option>
  <option value="3">Quarter 3 (July – September)</option>
```

[203] https://www.w3.org/WAI/WCAG21/Understanding/on-input.html
[204] https://www.w3.org/WAI/WCAG21/Techniques/general/G80.html
[205] https://www.w3.org/WAI/WCAG21/Techniques/html/H32.html
[206] https://www.w3.org/WAI/WCAG21/Techniques/html/H84.html

```
  <option value="4">Quarter 4 (October – December)</option>
</select>
<label for="year">Year:</label>
<input name="year" type="text" id="year">
<button name="show" type="button">Show</button>
```

Example 2: Choosing an action
A select element contains a list of possible actions. The action is not performed until the user presses the "Update" button.

```
<form action="/action" method="post">
  <label for="action">Options:</label>
  <select name="action" id="action">
    <option value="add">Add</option>
    <option value="remove">Remove</option>
    <option value="cancel">Cancel</option>
    <option value="order">Order</option>
  </select>
  <button name="submit" type="submit">Update</button>
</form>
```

Testing Procedure
For each `select` element/button element combination:
1. Check that focus (including keyboard focus) on an option in the `select` element does not result in any actions
2. Check that selecting the button performs the action associated with the current `select` value

Technique G13: Describing what will happen before a change to a form control that causes a change of context to occur is made[207]

Examples
- A series of radio buttons at the top of a page include options for German, French and Spanish. Instructions precede the buttons that instruct the user that the language will be changed upon selecting an option.
- A 50-question online survey displays one question at a time. Instructions appear at the beginning of the survey that explain that users will be taken to the next question of the survey upon selecting an answer to each question.

Testing Procedure
- Locate content where changing the setting of a form control results in a change of context
- Check to see that an explanation of what will happen when the control is changed is available prior to the controls activation

Technique SCR19: Using an onchange event on a select element without causing a change of context[208]

Examples
This example contains two select elements. When an item is selected in the first select, the choices in the other select are updated appropriately. The first select element contains a list of continents. The second select element will contain a partial list of countries located in the selected continent. There is an onchange event associated with the continent select. When the continent selection changes, the items in the country select are modified using JavaScript via the Document Object Model (DOM). All of the data required, the list of countries and continents, is included within the Web page.

Overview of the code below
- countryLists array variable which contains the list of countries for each continent in the trigger select element.
- countryChange() function which is called by the onchange event of the continent select element.
- The XHTML code to create the select elements in the body of the Web page.

[207] https://www.w3.org/WAI/WCAG21/Techniques/general/G13.html
[208] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR19.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "https://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="content-type" content="text/xhtml; charset=utf-8" />
    <title>Dynamic Select Statements</title>
<script type="text/javascript">
 //<![CDATA[
 // array of possible countries in the same order as they appear in the country selection list
 var countryLists = new Array(4)
 countryLists["empty"] = ["Select a Country"];
 countryLists["North America"] = ["Canada", "United States", "Mexico"];
 countryLists["South America"] = ["Brazil", "Argentina", "Chile", "Ecuador"];
 countryLists["Asia"] = ["Russia", "China", "Japan"];
 countryLists["Europe"]= ["Britain", "France", "Spain", "Germany"];
 /* CountryChange() is called from the onchange event of a select element.
 * param selectObj - the select object which fired the on change event.
 */
 function countryChange(selectObj) {
 // get the index of the selected option
 var idx = selectObj.selectedIndex;
 // get the value of the selected option
 var which = selectObj.options[idx].value;
 // use the selected option value to retrieve the list of items from the countryLists array
 cList = countryLists[which];
 // get the country select element via its known id
 var cSelect = document.getElementById("country");
 // remove the current options from the country select
 var len=cSelect.options.length;
 while (cSelect.options.length > 0) {
 cSelect.remove(0);
 }
 var newOption;
 // create new options
 for (var i=0; i<cList.length; i++) {
 newOption = document.createElement("option");
 newOption.value = cList[i];  // assumes option string and value are the same
 newOption.text=cList[i];
 // add the new option
 try {
 cSelect.add(newOption);  // this will fail in DOM browsers but is needed for IE
 }
 catch (e) {
 cSelect.appendChild(newOption);
 }
 }
 }
//]]>
</script>
</head>
<body>
  <noscript>This page requires JavaScript be available and enabled to function
properly</noscript>
  <h1>Dynamic Select Statements</h1>
  <label for="continent">Select Continent</label>
  <select id="continent" onchange="countryChange(this);">
    <option value="empty">Select a Continent</option>
    <option value="North America">North America</option>
    <option value="South America">South America</option>
    <option value="Asia">Asia</option>
    <option value="Europe">Europe</option>
  </select>
  <br/>
  <label for="country">Select a country</label>
  <select id="country">
    <option value="0">Select a country</option>
  </select>
```

```
</body>
 </html>
```

Testing Procedure

1. Navigate to the trigger select element (in this example, the one to select continents) and change the value of the select.
2. Navigate to the select element that is updated by the trigger (in this example, the one to select countries).
3. Check that the matching option values are displayed in the other select element.
4. Navigate to the trigger select element, navigate through the options but do not change the value.
5. Check that the matching option values are still displayed in the associated element.

It is recommended that the select elements are tested with an assistive technology to verify that the changes to the associated element are recognized.

3.2.3 Consistent Navigation (Level AA)[209] – This Success Criterion's objective is to encourage the adoption of a consistent presentation and layout for users who engage with repeated material across a collection of Web sites and have a recurring requirement to locate a certain piece of functionality or piece of information. People with restricted eyesight often enlarge a small portion of the screen at a time in order to recognize repetitious information as rapidly as possible. This allows them to make use of visual cues and page boundaries. It is essential to provide the content in the same order each time for users who rely on their spatial memories or on visual hints in the design to discover repeated content. These users can find it easier to navigate the interface.

Technique G61: Presenting repeated components in the same relative order each time they appear[210]

Examples

- A Web site has a logo, a title, a search form and a navigation bar at the top of each page; these appear in the same relative order on each page where they are repeated. On one page the search form is missing but the other items are still in the same order.
- A Web site has a left-hand navigation menu with links to the major sections of the site. When the user follows a link to another section of the site, the links to the major sections appear in the same relative order in the next page. Sometime links are dropped and other links are added, but the other links always stay in the same relative order. For example, on a Web site of a company that sells products and offers training, when a user moves from the section on products to the section on training, the links to individual products are removed from the navigation list, while links to training offerings are added.

Testing Procedure

1. List components that are repeated on each Web page in a set of Web pages (for example, on each page in a Web site).
2. For each component, check that it appears in the same relative order with regard to other repeated components on each Web page where it appears.
3. For each navigational component, check that the links or programmatic references are always in the same relative order.

3.2.4 Consistent Identification (Level AA)[211] – This Success Criterion is developed to ensure the reliable detection of functional features that appear in a number of different web pages throughout a website's collection. When surfing a website, people who use screen readers will frequently rely on their familiarity with various elements that may exist on a variety of different web pages. If identical functions on separate web pages have different names or labels (or, more generally, if they have a different

[209] https://www.w3.org/WAI/WCAG21/Understanding/consistent-navigation.html
[210] https://www.w3.org/WAI/WCAG21/Techniques/general/G61.html
[211] https://www.w3.org/WAI/WCAG21/Understanding/consistent-identification.html

accessible name), then using the website will be a lot more difficult. It is possible that persons with cognitive limitations will find it confusing, which will increase the amount of mental effort they have to do. Therefore, labeling everything in the same manner will be advantageous.

Technique G197: Using labels, names, and text alternatives consistently for content that has the same functionality[212]

Examples

- A Web page has a form field at the top of the page labeled "Search". On the bottom of the page is another form field which provides the same function. It is also labeled "Search."
- A picture of a question mark is used to steer users to sections of the page that provide additional information. Each time the picture of the question mark appears it has the same text alternative "more information."
- A link to the Contact Us page of a Web site has the link text "Contact". At the bottom of the page there is a link that also goes to the Contact Us page. It also has the link text "Contact".

Testing Procedure

1. Check that each component is associated with text that identifies it (i.e., label, name, or text alternative).
2. Check that this associated text is identical for each user interface component with the same function.

***Note 1:* Text alternatives that are "consistent" are not always "identical." For instance, you may have an graphical arrow at the bottom of a Web page that links to the next Web page. The text alternative may say "Go to page 4." Naturally, it would not be appropriate to repeat this exact text alternative on the next Web page. It would be more appropriate to say "Go to page 5". Although these text alternatives would not be identical, they would be consistent, and therefore would satisfy this Success Criterion. **

***Note 2:* A single non-text-content-item may be used to serve different functions. In such cases, different text alternatives are necessary and should be used. Examples can be commonly found with the use of icons such as check marks, cross marks, and traffic signs. Their functions can be different depending on the context of the Web page. A check mark icon may function as "approved", "completed", or "included", to name a few, depending on the situation. Using "check mark" as text alternative across all Web pages does not help users understand the function of the icon. Different text alternatives can be used when the same non-text content serves multiple functions. **

3.3 Input Assistance[213] – Under WCAG, this section's purpose is to assist users avoid and correct mistakes when entering data into the target site. There are four criteria points for level AA conformance.

3.3.1 Error Identification (Level A)[214] – This Success Criterion is developed to ensure that users are made aware of problems and are able to determine what went wrong with the system. The error message should include as much specific information as is reasonably possible. In the event that a user is unable to successfully submit a form, redisplaying the form and highlighting any fields in which they made a mistake are not always enough to convince the user that an error has been made. Users of screen readers, for example, won't become aware of the error until they come across one of the indicators. They might quit up on the form altogether and come to the conclusion that the page is corrupted before they see the error notice. According to the WCAG standard, information that is submitted by the user that is not permitted constitutes a "input error." This includes information that the user omits that the web page requires, as well as information that the user provides but that does not follow the needed data structure or permitted values. Also included in this category is information that the user provides but that does not comply with the needed data structure.

[212] https://www.w3.org/WAI/WCAG21/Techniques/general/G197.html

[213] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#input-assistance

[214] https://www.w3.org/WAI/WCAG21/Understanding/error-identification.html

Situation A: If a form contains fields for which information from the user is mandatory.

Technique G83: Providing text descriptions to identify required fields that were not completed[215]

Examples

- A user attempts to submit a form but has neglected to provide input or select a choice in one or more mandatory fields. Using client-side validation, the omission is detected and an alert dialog appears informing the user that mandatory fields have not been completed. The labels of the fields with this problem are changed to identify the problem field, and links to the problem fields are inserted in the document after the submit button so the user can move to them after dismissing the alert.
- A user attempts to submit a form but has neglected to provide input or select a choice in one or more mandatory fields. Using server-side validation, the omission is detected and the form is re-displayed with a text description at the top informing which mandatory fields were omitted. Each omitted mandatory field is also identified using a text label so that the user does not have to return to the list at the top of the form to find the omitted fields.
- A user is completing a form that contains mandatory fields. The labels of the fields indicate whether or not they are mandatory. The user tabs to a mandatory field, and tabs out of the field without entering any data or selecting a choice. A client-side script modifies the label of the field to indicate that leaving it blank was an error.

** NOTE: Some screen readers may not notice and announce the change to the label so screen reader users may be unaware of the error. **

Testing Procedure

1. Fill out a form, deliberately leaving one or more required (mandatory) fields blank, and submit it.
2. Check that a text description is provided identifying the mandatory field(s) that was not completed.
3. Check that other data previously entered by the user is re-displayed, unless the data is in a security related field where it would be inappropriate to retain the data for re-display (e.g. password).

Technique ARIA21: Using Aria-Invalid to Indicate an Error Field[216]

Example 1: Using aria-invalid on required fields

The aria-invalid attribute is used on required fields that have no input. A message above the form conveys that form submission has failed due to this.

A portion of the jQuery code and the HTML form markup follow:

```
<code>
<script>
...
...
               if ($('#first').val() === '') {
                      $('#first').attr("aria-invalid", "true");
$("label[for='first']").addClass('failed');
               }
               if ($('#last').val() === '') {
                      $('#last').attr("aria-invalid", "true");
$("label[for='last']").addClass('failed');
               }
               if ($('#email').val() === '') {
                      $('#email').attr("aria-invalid", "true");
$("label[for='email']").addClass('failed');
               }
...
...
</script>
<style type="text/css">
label.failed {
       border: red thin solid;
}
</style>
<form name="signup" id="signup" method="post" action="#">
```

[215] https://www.w3.org/WAI/WCAG21/Techniques/general/G83.html
[216] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA21.html

```
 <p>
    <label for="first">First Name (required)</label><br>
    <input type="text" name="first" id="first">
  </p>
  <p>
    <label for="last">Last Name (required)</label><br>
    <input type="text" name="last" id="last">
  </p>
  <p>
    <label for="email">Email (required)</label><br>
    <input type="text" name="email" id="email">
  </p>
  <p>
    <input type="submit" name="button" id="button" value="Submit">
  </p>
</form>
</code>
```

Example 2: Identifying errors in data format

Aria-invalid and aria-describedby are used together to indicate an error when the personal identification number (PIN), email address, or start date are not in the expected format. The error message is associated with the field using aria-describedby, and aria-invalid makes it easier to programmatically find fields with errors.
Below is the rendered HTML code for the email address field in Example 1: When an invalid email address is entered by the user such as "samexample.com" (instead of sam@example.com), the HTML code is:

```
<div class="control">
<p><label for="email">Email address: [*]</label>
<input   type="text"   name="email"   id="email"   class="error"   aria-invalid="true"   aria-
describedBy="err_1" /></p>
<span class="errtext" id="err_1">Error: Incorrect data</span></div>
```

And when no data is entered in the email field, the HTML code is:

```
<div class="control">
<p><label for="email">Email address: [*]</label>
<input   type="text"   name="email"   id="email"   class="error"   aria-invalid="true"   aria-
describedBy="err_2" /></p>
<span class="errtext" id="err_2">
 Error: Input data missing</span>
</div>
jQuery code: jQuery is used to add aria-invalid or aria-describedby attributes as well as the
class attribute and append the error text. This is the code that inserts aria-invalid and
class="error" but does not associate the error text "incorrect data" with the control
programmatically:
$(errFld).attr("aria-invalid", "true").attr("class", "error");
// Suffix error text:
$(errFld).parent().append('<span class="errtext">Error: Incorrect data</span>');

CSS Code:
input.error {
   border: red thin solid;}
span.errtext {
       margin-bottom: 1em;   padding: .25em 1.4em .25em .25em;
       border: red thin solid;       background-color: #EEEEFF;
       background-image:url('images/iconError.gif');
       background-repeat:no-repeat;  background-position:right;
}
```

Testing Procedure

For each form control that relies on `aria-invalid` to convey a validation failure:

1. Check that `aria-invalid` is not set to true when a validation failure does not exist.
2. Check that `aria-invalid` is set to true when a validation failure does exist.
3. Check that the programmatically associated labels / programmatically associated instructional text for the field provide enough information to understand the error.

Technique SCR18: Providing client-side validation and alert[217]

Example 1: Checking a single control with an event handler
The following script will check that a valid date has been entered in the form control.

```
<label for="date">Date:</label>
<input type="text" name="date" id="date"
onchange="if(isNaN(Date.parse(this.value)))
alert('This control is not a valid date.
Please re-enter the value.');" />
```

Example 2: Checking multiple controls when the user submits the form
The following sample shows multiple controls in a form. The form element uses the onsubmit attribute which creates an event handler to execute the validation script when the user attempts to submit the form. If the validation is successful, the event returns true and the form submission proceeds; if the validation finds errors, it displays an error message and returns false to cancel the submit attempt so the user can fix the problems.
This example demonstrates an alert for simplicity. A more helpful notification to the user would be to highlight the controls with problems and add information to the page about the nature of the errors and how to navigate to the controls that require data fixes.

Although this example uses an onsubmit attribute on the form element for brevity, normal practice is to create a submit event listener when the page is loaded.

Script code:

```
function validate() {
	// initialize error message
	var msg = "";

	//validate name
	var pattern = /^[a-zA-Z\s]+$/;
	var el = document.getElementById("name");
	if (!pattern.test(el.value))  msg += "Name can only have letters and spaces. ";

	// validate number
	var pattern = /^[\d\-+\.\s]+$/;
	var el = document.getElementById("tel");
	if  (!pattern.test(el.value))    msg  +=  "Telephone  number  can  only  have  digits  and
separators. ";

	if (msg != "") {
		alert(msg);
		return false;
	} else return true;
}
```

Form code:

```
<form action="multiple-controls.html" onsubmit="return validate()">
	<p>
		<label for="name">Name: </label>
		<input type="text" name="name" id="name" />
	</p>
	<p>
		<label for="tel">Telephone number: </label>
		<input type="text" name="tel" id="tel" />
	</p>
	<p>
		<input type="submit" />
	</p>
</form>
```

[217] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR18.html

Testing Procedure
For form fields that require specific input:
1. enter invalid data
2. determine if an alert describing the error is provided.

Situation B: If information provided by the user is required to be in a specific data format or of certain values.

Technique ARIA18 : Using `aria-alertdialog` to Identify Errors[218]

Example 1: Alert dialog
This example shows how a notification using role="alertdialog" can be used to notify someone they have entered invalid information.

```
<div role="alertdialog" aria-labelledby="alertHeading" aria-describedby="alertText">
<h1 id="alertHeading">Error</h1>
<div id="alertText">Employee's Birth Date is after their hire date. Please verify the birth date
and hire date.</div>
<button>Save and Continue</button>
<button>Return to page and correct error</button>
</div>
```

Testing Procedure
1. Trigger the error that causes the alertdialog to appear.
2. Determine that the alertdialog contains at least one focusable control, and the focus moves to that control when the alertdialog opens.
3. Determine that the tab order is constrained within the alertdialog while it is open, and when the dialog is dismissed, the focus moves back to the position it had before the dialog opene, if possible.
4. Examine the element with `role="alertdialog"` applied.
5. Determine that either the `aria-label` or `aria-labelledby` attribute has been correctly used to give the alertdialog an accessible name.
6. Determine that the contents of the alertdialog identifies the input error.
7. Determine whether contents of the alertdialog suggests how to fix the error.

Technique ARIA19: Using ARIA role=alert or Live Regions to Identify Errors[219]

Example 1: Injecting error messages into a container with role=alert already present in the DOM
The following example uses role=alert which is equivalent to using aria-live=assertive.
In the example there is an empty error message container element with aria-atomic=true and an aria-live property or alert role present in the DOM on page load. The error container must be present in the DOM on page load for the error message to be spoken by most screen readers. aria-atomic=true is necessary to make Voiceover on iOS read the error messages after more than one invalid submission.

jQuery is used to test if the inputs are empty on submit and inject error messages into the live region containers if so. Each time a new submit is attempted the previous error messages are removed from the container and new error messages injected.

```
<script src="http://code.jquery.com/jquery.js"></script>
<script>
$(document).ready(function(e) {
	$('#signup').submit(function() {
		$('#errors').html('');
		if ($('#first').val() === '') {
			$('#errors').append('<p>Please enter your first name.</p>');
		}
		if ($('#last').val() === '') {
```

[218] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA18.html
[219] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA19.html

```
                        $('#errors').append('<p>Please enter your last name.</p>');
                }
                if ($('#email').val() === '') {
                        $('#errors').append('<p>Please enter your email address.</p>');
                }
                return false;
        });
});
</script>

<form name="signup" id="signup" method="post" action="">
  <p id="errors" role="alert" aria-atomic="true"></p>
  <p>
    <label for="first">First Name (required)</label><br>
    <input type="text" name="first" id="first">
  </p>
  <p>
    <label for="last">Last Name (required)</label><br>
    <input type="text" name="last" id="last">
  </p>
  <p>
    <label for="email">Email (required)</label><br>
    <input type="text" name="email" id="email">
  </p>
  <p>
    <input type="submit" name="button" id="button" value="Submit">
  </p>
</form>
```

Testing Procedure

1. Determine that an empty error container `role=alert` or `aria-live=assertive` attribute is present in the DOM at page load.
2. Trigger the error that causes the content in the live region to appear or update.
3. Determine that the error message was injected into the already present error container.

Technique ARIA21: Using Aria-Invalid to Indicate an Error Field[220]

Example 1: Using aria-invalid on required fields

The aria-invalid attribute is used on required fields that have no input. A message above the form conveys that form submission has failed due to this.

A portion of the jQuery code and the HTML form markup follow:

```
<code>
<script>
...
...
                if ($('#first').val() === '') {
                        $('#first').attr("aria-invalid", "true");
$("label[for='first']").addClass('failed');
                }
                if ($('#last').val() === '') {
                        $('#last').attr("aria-invalid", "true");
$("label[for='last']").addClass('failed');
                }
                if ($('#email').val() === '') {
                        $('#email').attr("aria-invalid", "true");
$("label[for='email']").addClass('failed');
                }
...
...
</script>
```

[220] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA21.html

```
<style type="text/css">
label.failed {
        border: red thin solid;
}
</style>
<form name="signup" id="signup" method="post" action="#">
 <p>
    <label for="first">First Name (required)</label><br>
    <input type="text" name="first" id="first">
  </p>
  <p>
    <label for="last">Last Name (required)</label><br>
    <input type="text" name="last" id="last">
  </p>
  <p>
    <label for="email">Email (required)</label><br>
    <input type="text" name="email" id="email">
  </p>
  <p>
    <input type="submit" name="button" id="button" value="Submit">
  </p>
</form>
</code>
```

Example 2: Identifying errors in data format

Aria-invalid and aria-describedby are used together to indicate an error when the personal identification number (PIN), email address, or start date are not in the expected format. The error message is associated with the field using aria-describedby, and aria-invalid makes it easier to programmatically find fields with errors.

Below is the rendered HTML code for the email address field in Example 1: When an invalid email address is entered by the user such as "samexample.com" (instead of sam@example.com), the HTML code is:

```
<div class="control">
<p><label for="email">Email address: [*]</label>
<input type="text" name="email" id="email" class="error" aria-invalid="true" aria-
describedBy="err_1" /></p>
<span class="errtext" id="err_1">Error: Incorrect data</span></div>
```

And when no data is entered in the email field, the HTML code is:

```
<div class="control">
<p><label for="email">Email address: [*]</label>
<input type="text" name="email" id="email" class="error" aria-invalid="true" aria-
describedBy="err_2" /></p>
<span class="errtext" id="err_2">
 Error: Input data missing</span>
</div>
```

jQuery code: jQuery is used to add aria-invalid or aria-describedby attributes as well as the class attribute and append the error text. This is the code that inserts aria-invalid and class="error" but does not associate the error text "incorrect data" with the control programmatically:

```
$(errFld).attr("aria-invalid", "true").attr("class", "error");
// Suffix error text:
$(errFld).parent().append('<span class="errtext">Error: Incorrect data</span>');
```

CSS Code:

```
input.error {
   border: red thin solid;}
span.errtext {
        margin-bottom: 1em;    padding: .25em 1.4em .25em .25em;
        border: red thin solid;       background-color: #EEEEFF;
        background-image:url('images/iconError.gif');
        background-repeat:no-repeat;  background-position:right;
```

```
}
```

Testing Procedure

For each form control that relies on `aria-invalid` to convey a validation failure:

1. Check that `aria-invalid` is not set to true when a validation failure does not exist.
2. Check that `aria-invalid` is set to true when a validation failure does exist.
3. Check that the programmatically associated labels / programmatically associated instructional text for the field provide enough information to understand the error.

Technique G84: Providing a text description when the user provides information that is not in the list of allowed values[221]

Examples

- The user inputs invalid data on a form field. Before the user submits the form, an alert dialog appears that describes the nature of the error so the user can fix it.
- The user inputs invalid data on a form field and submits the form. The server returns the form, with the user's data still present, and indicates clearly in text at the top of the page that there were input errors. The text describes the nature of the error(s) and clearly indicates which field had the problem so the user can easily navigate to it to fix the problem.

Testing Procedure

1. Enter invalid data in a form field.
2. Check that information is provided in text about the problem.

Technique G85: Providing a text description when user input falls outside the required format or values[222]

Examples

- The user inputs invalid data on a form field. When the user exits the field, an alert dialog appears that describes the nature of the error so the user can fix it.
- The user inputs invalid data on a form field and submits the form. The server returns the form, with the user's data still present, and indicates clearly in text at the top of the page that there were input errors. The text describes the nature of the error(s) and clearly indicates which field had the problem so the user can easily navigate to it to fix the problem.
- The user inputs invalid data on a form field and attempts to submit the form. Client-side scripting detects the error, cancels the submit, and modifies the document to provide a text description after the submit button describing the error, with links to the field(s) with the error. The script also modifies the labels of the fields with the problems to highlight them.

Testing Procedure

1. Fill out a form, deliberately enter user input that falls outside the required format or values
2. Check that a text description is provided that identifies the field in error and provides some information about the nature of the invalid entry and how to fix it.
3. Check that other data previously entered by the user is re-displayed, unless the data is in a security related field where it would be inappropriate to retain the data for re-display (e.g. password).

Technique SCR18: Providing client-side validation and alert

Example

{See Above}

Testing Procedure

[221] https://www.w3.org/WAI/WCAG21/Techniques/general/G84.html
[222] https://www.w3.org/WAI/WCAG21/Techniques/general/G85.html

{See Above}

Technique SCR32: Providing client-side validation and adding error text via the DOM[223]

Examples
This example validates required fields as well as fields where a specific format is required. When an error is identified, the script inserts a list of error messages into the DOM and moves focus to them.

HTML and Javascript code

Here is the HTML for the example form:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "https://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <title>Form Validation</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <link href="css/validate.css" rel="stylesheet" type="text/css"/>
        <script type="text/javascript" src="scripts/validate.js"/>
    </head>
    <body>

        <h1>Form Validation</h1>

        <p>The following form is validated before being submitted if scripting is available,
            otherwise the form is validated on the server. All fields are required, except those
            marked optional. If errors are found in the submission, the form is cancelled and
            a list of errors is displayed at the top of the form.</p>

        <p> Please enter your details below. </p>

        <h2>Validating Form</h2>

        <form id="personalform" method="post" action="index.php">
            <div class="validationerrors"/>
            <fieldset>
                <legend>Personal Details</legend>
                <p>
                    <label for="forename">Please enter your forename</label>
                    <input type="text" size="20" name="forename" id="forename" class="string"
                        value=""/>
                </p>
                <p>
                    <label for="age">Please enter your age</label>
                    <input type="text" size="20" name="age" id="age" class="number" value=""/>
                </p>
                <p>
                    <label for="email">Please enter your email address</label>
                    <input   type="text"   size="20"   name="email"   id="email"   class="email"
value=""/>
                </p>
            </fieldset>
            <p>
                <input type="submit" name="signup" value="Sign up"/>
            </p>
        </form>
        <h2>Second Form</h2>
        <form id="secondform" method="post" action="index.php#focuspoint">
            <div class="validationerrors"/>
            <fieldset>
                <legend>Second Form Details</legend>
                <p>
                    <label for="suggestion">Enter a suggestion</label>
                    <input type="text" size="20" name="suggestion" id="suggestion"
```

[223] https://www.w3.org/WAI/WCAG21/Techniques/client-side-script/SCR32.html

```
                    class="string" value=""/>
            </p>
            <p>
                <label for="optemail">Please enter your email address (optional)</label>
                <input type="text" size="20" name="optemail" id="optemail"
                    class="optional email" value=""/>
            </p>
            <p>
                <label for="rating">Please rate this suggestion</label>
                <input type="text" size="20" name="rating" id="rating"
                  class="number" value=""/>
            </p>
            <p>
                <label for="jibberish">Enter some jibberish (optional)</label>
                <input type="text" size="20" name="jibberish" id="jibberish" value=""/>
            </p>

        </fieldset>
        <p>
            <input type="submit" name="submit" value="Add Suggestion"/>
        </p>
    </form>
  </body>
</html>
```

Here is the Javascript which performs the validation and inserts the error messages:

```
window.onload = initialise;

function initialise()
{
   var objForms = document.getElementsByTagName('form');
   var iCounter;

   // Attach an event handler for each form
   for (iCounter=0; iCounter<objForms.length; iCounter++)
   {
      objForms[iCounter].onsubmit = function(){return validateForm(this);};
   }
}


// Event handler for the form
function validateForm(objForm)
{
   var arClass = [];
   var iErrors = 0;
   var objField = objForm.getElementsByTagName('input');
   var objLabel = objForm.getElementsByTagName('label');
   var objList = document.createElement('ol');
   var objError, objExisting, objNew, objTitle, objParagraph, objAnchor, objPosition;
   var strLinkID, iFieldCounter, iClassCounter, iCounter;

   // Get the id or name of the form, to make a unique
   // fragment identifier
   if (objForm.id)
   {
      strLinkID = objForm.id + 'ErrorID';
   }
   else
   {
      strLinkID = objForm.name + 'ErrorID';
   }

   // Iterate through input form controls, looking for validation classes
   for (iFieldCounter=0; iFieldCounter<objField.length; iFieldCounter++)
   {
      // Get the class for the field, and look for the appropriate class
```

```
      arClass = objField[iFieldCounter].className.split(' ');
      for (iClassCounter=0; iClassCounter<arClass.length; iClassCounter++)
      {
         switch (arClass[iClassCounter])
         {
            case 'string':
               if (!isString(objField[iFieldCounter].value, arClass))
               {
                  if (iErrors === 0)
                  {
                     logError(objField[iFieldCounter], objLabel, objList, strLinkID);
                  }
                  else
                  {
                     logError(objField[iFieldCounter], objLabel, objList, '');
                  }
                  iErrors++;
               }
               break;
            case 'number':
               if (!isNumber(objField[iFieldCounter].value, arClass))
               {
                  if (iErrors === 0)
                  {
                     logError(objField[iFieldCounter], objLabel, objList, strLinkID);
                  }
                  else
                  {
                     logError(objField[iFieldCounter], objLabel, objList, '');
                  }
                  iErrors++;
               }
               break;

            case 'email' :
               if (!isEmail(objField[iFieldCounter].value, arClass))
               {
                  if (iErrors === 0)
                  {
                     logError(objField[iFieldCounter], objLabel, objList, strLinkID);
                  }
                  else
                  {
                     logError(objField[iFieldCounter], objLabel, objList, '');
                  }
                  iErrors++;
               }
               break;
         }
      }
   }

   if (iErrors > 0)
   {
      // If not valid, display error messages
      objError = objForm.getElementsByTagName('div');

      // Look for existing errors
      for (iCounter=0; iCounter<objError.length; iCounter++)
      {
         if (objError[iCounter].className == 'validationerrors')
         {
            objExisting = objError[iCounter];
         }
      }

      objNew = document.createElement('div');
      objTitle = document.createElement('h2');
      objParagraph = document.createElement('p');
```

```
      objAnchor = document.createElement('a');

      if (iErrors == 1)
      {
         objAnchor.appendChild(document.createTextNode('1 Error in Submission'));
      }
      else
      {
         objAnchor.appendChild(document.createTextNode(iErrors + ' Errors in Submission'));
      }
      objAnchor.href = '#' + strLinkID;
      objAnchor.className = 'submissionerror';

      objTitle.appendChild(objAnchor);
      objParagraph.appendChild(document.createTextNode('Please review the following'));
      objNew.className = 'validationerrors';

      objNew.appendChild(objTitle);
      objNew.appendChild(objParagraph);
      objNew.appendChild(objList);

      // If there were existing error, replace them with the new lot,
      // otherwise add the new errors to the start of the form
      if (objExisting)
      {
         objExisting.parentNode.replaceChild(objNew, objExisting);
      }
      else
      {
         objPosition = objForm.firstChild;
         objForm.insertBefore(objNew, objPosition);
      }

      // Allow for latency
      setTimeout(function() { objAnchor.focus(); }, 50);

      // Don't submit the form
      objForm.submitAllowed = false;
      return false;
   }

   // Submit the form
   return true;
}

// Function to add a link in a list item that points to problematic field control
function addError(objList, strError, strID, strErrorID)
{
   var objListItem = document.createElement('li');
   var objAnchor = document.createElement('a');

   // Fragment identifier to the form control
   objAnchor.href='#' + strID;

   // Make this the target for the error heading
   if (strErrorID.length > 0)
   {
      objAnchor.id = strErrorID;
   }

   // Use the label prompt for the error message
   objAnchor.appendChild(document.createTextNode(strError));
   // Add keyboard and mouse events to set focus to the form control
   objAnchor.onclick = function(event){return focusFormField(this, event);};
   objAnchor.onkeypress = function(event){return focusFormField(this, event);};
   objListItem.appendChild(objAnchor);
   objList.appendChild(objListItem);
}
```

```
function focusFormField(objAnchor, objEvent)
{
   var strFormField, objForm;

   // Allow keyboard navigation over links
   if (objEvent && objEvent.type == 'keypress')
   {
      if (objEvent.keyCode != 13 && objEvent.keyCode != 32)
      {
         return true;
      }
   }

   // set focus to the form control
   strFormField = objAnchor.href.match(/[^#]\w*$/);
   objForm = getForm(strFormField);
   objForm[strFormField].focus();
   return false;
}

// Function to return the form element from a given form field name
function getForm(strField)
{
   var objElement = document.getElementById(strField);

   // Find the appropriate form
   do
   {
      objElement = objElement.parentNode;
   } while (!objElement.tagName.match(/form/i) && objElement.parentNode);

   return objElement;
}

// Function to log the error in a list
function logError(objField, objLabel, objList, strErrorID)
{
   var iCounter, strError;

   // Search the label for the error prompt
   for (iCounter=0; iCounter<objLabel.length; iCounter++)
   {
      if (objLabel[iCounter].htmlFor == objField.id)
      {
         strError = objLabel[iCounter].firstChild.nodeValue;
      }
   }

   addError(objList, strError, objField.id, strErrorID);
}

// Validation routines - add as required

function isString(strValue, arClass)
{
   var bValid = (typeof strValue == 'string' && strValue.replace(/^\s*|\s*$/g, '')
     !== '' && isNaN(strValue));

   return checkOptional(bValid, strValue, arClass);
}

function isEmail(strValue, arClass)
{
   var objRE = /^[\w-\.\']{1,}\@([\da-zA-Z\-]{1,}\.){1,}[\da-zA-Z\-]{2,}$/;
   var bValid = objRE.test(strValue);

   return checkOptional(bValid, strValue, arClass);
}
```

```
function isNumber(strValue, arClass)
{
   var bValid = (!isNaN(strValue) && strValue.replace(/^\s*|\s*$/g, '') !== '');

   return checkOptional(bValid, strValue, arClass);
}

function checkOptional(bValid, strValue, arClass)
{
   var bOptional = false;
   var iCounter;

   // Check if optional
   for (iCounter=0; iCounter<arClass.length; iCounter++)
   {
      if (arClass[iCounter] == 'optional')
      {
         bOptional = true;
      }
   }

   if (bOptional && strValue.replace(/^\s*|\s*$/g, '') === '')
   {
      return true;
   }

   return bValid;
   }
```

Testing Procedure

Create error messages using anchor tags and appropriate scripting via the technique above.

1. Load the page.
2. Enter a valid value in the field(s) associated with an error message and verify that no error messages are displayed.
3. Enter an invalid value in the field(s) associated with an error message and verify that the correct error message for the field is displayed.
4. Verify that the error messages receive focus.
5. Enter a valid value in the field(s) associated with the displayed error message and verify that the error message is removed.
6. Repeat for all fields with associated error messages created via anchor tags

** Note: It is recommended that you also run the above procedure using an assistive technology. **

3.3.2 Labels or Instructions (Level A)[224] – This Success Criterion requires content providers to give instructions or labels that identify the controls in a form. This is done so that users would have a better understanding of what data should be entered into the form. Each option in controls such as radio buttons, checkboxes, combo boxes, and any other control that presents users with a selection must have a suitable label for them to be able to comprehend what it is that they are selecting. Data formats for data entry fields may also be given in instructions or labels, particularly if they differ from common forms or if there are unique requirements for accurate input. This is particularly important in situations where there are specific requirements for correct input. If the instructions are extensive and detailed, the writers of the material may choose to only make them accessible to users when the particular control is in the focus of their attention.

Technique G131: Providing descriptive labels

[224] https://www.w3.org/WAI/WCAG21/Understanding/labels-or-instructions.html

Example
{See Above}

Testing Procedure
{See Above}

Technique ARIA1: Using the `aria-describedby` property to provide a descriptive label for user interface controls[225]

Example 1: Using `aria-describedby` to associate instructions with form fields
Sample form field using `aria-describedby` to associate instructions with form fields while there is a form label.

```
<form>
  <label for="fname">First name</label>
  <input aria-describedby="int2" autocomplete="given-name" id="fname" type="text">
  <p id="int2">Your first name is sometimes called your "given name".</p>
</form>
```

Example 2: Using `aria-describedby` property to provide more detailed information about the button

```
<div>
  <span id="fontDesc">Select the font faces and sizes to be used on this page</span>
  <button aria-describedby="fontDesc" id="fontB" type="button">Fonts</button>
</div>
<div>
  <span id="colorDesc">Select the colors to be used on this page</span>
  <button aria-describedby="colorDesc" id="colorB" type="button">Colors</button>
</div>
<div>
  <span id="customDesc">Customize the layout and styles used on this page</span>
  <button aria-describedby="customDesc" id="customB" type="button">Customize</button>
</div>
```

Testing Procedure
1. Check that there is a user interface control having an `aria-describedby` attribute that references one or more elements via unique `id`.
2. Check that the referenced element or elements provide additional information about the user interface control.

Technique ARIA9: Using `aria-labelledby` to concatenate a label from several text nodes[226]

Example 1: A time-out input field with concatenated label
A text input allows users to extend the default time before a time-out occurs.

The string "Extend time-out to" is contained in a native label element and is associated with the input with the input by id="timeout-duration" . This label is associated with this input using the for/id association only on user agents that don't support ARIA. On user agents that support ARIA, the for/id association is ignored and the label for the input is provided only by aria-labelledby, per the accessible name and description calculation in the HTML to Platform Accessibility APIs Implementation Guide.

The aria-labelledby attribute on the text input references three elements: (1) the span containing the native label, (2) the text input containing the default text '20' (recall that this input is not labelled with the for/id associated label text), and (3) the string 'minutes' contained in a span. These elements should be concatenated to provide the full label for the text input

[225] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA1.html
[226] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA9.html

The use of tabindex="-1" on the span element is not meant to support focusing by scripts - here, it merely serves to ensure that some browsers (IE9, IE10) will include the span element in the accessibility tree, thus making it available for reference by aria-labelledby. For more details see Accessible HTML Elements

```
<form>
<p><span  id="timeout-label"  tabindex="-1"><label  for="timeout-duration">Extend  time-out
to</label></span>
<input type="text" size="3" id="timeout-duration" value="20"
    aria-labelledby="timeout-label timeout-duration timeout-unit">
<span id="timeout-unit" tabindex="-1"> minutes</span></p>
</form>
```

Working example, Time-out input field with concatenated label, adapted from Easy ARIA tip #2: aria-labelledby and aria-describedby, an example put together by Marco Zehe.

Example 2: A simple data table with text inputs
A simple data table containing text inputs. The input labels are concatenated through aria-labelledby referencing the respective column and row headers.

```
<table>
	<tr>
		<td></td>
		<th id="tpayer">Taxpayer</th>
		<th id="sp">Spouse</th>
	</tr>

	<tr>
		<th id="gross">W2 Gross</th>
		<td><input type="text" size="20" aria-labelledby="tpayer gross" /></td>
		<td><input type="text" size="20" aria-labelledby="sp gross" /></td>
	</tr>

	<tr>
		<th id="div">Dividends</th>
		<td><input type="text" size="20" aria-labelledby="tpayer div" /></td>
		<td><input type="text" size="20" aria-labelledby="sp div" /></td>
	</tr>
</table>
```

Working example, Using aria-labelledby for simple table with text inputs, based on an example by Jim Thatcher.

Example 3: A conference workshop booking table
A conference workshop booking table with two parallel tracks allows users to select the workshop they want to attend. When tabbing through the checkbox inputs in the table, the track (1 or 2), the title, and the speaker of the workshop followed by the adjacent checkbox label "Attend" are provided as concatenated label for the checkboxes via aria-labelledby.

Some browser / screen reader combinations (e.g. Mozilla Firefox and NVDA) will in addition speak the relevant table cell headers.

```
<h1>Dinosaur Conference workshops timetable Thursday, 14.  & Friday, 15. March 2013</h1>
<table>
<caption>Dinosaur Conference workshop booking table</caption>
<tbody><tr>
	<td rowspan="2"></td>
	<th colspan="2" scope="colgroup">Thursday</th>
	<th colspan="2" scope="colgroup">Friday</th>
</tr>

<tr>
	<th scope="col" id="am1">9 to 12 AM</th>
	<th scope="col" id="pm1">2 to 5 PM</th>
	<th scope="col" id="am2">9 to 12 AM</th>
	<th scope="col" id="pm2">2 to 5 PM</th>
```

```
</tr>

<tr>
        <th id="track1" scope="row">track 1</th>
        <td>
                <h2 id="title-TM1">The Paleozoic era </h2>
                <p>2 places left</p>
                <p><input type="checkbox" id="TM1" aria-labelledby="title-TM1 track1 am1 TM1-
att">
                 <label id="TM1-att" for="TM1">Attend</label></p>
        </td>

        <td>
                <h2 id="title-TA1">The Mesozoic era overview</h2>
                <p>2 places left</p>
                <p><input type="checkbox" id="TA1" aria-labelledby="title-TA1 track1 am2 TA1-
att">
                 <label id="TA1-att" for="TA1">Attend</label></p>
        </td>

        <td>
                <h2 id="title-FM1">The Triassic period, rise of the dinosaurs</h2>
                <p>1 place left</p>
                <p><input type="checkbox" id="FM1" aria-labelledby="title-FM1 track1 pm1 FM1-
att">
                 <label id="FM1-att" for="FM1">Attend</label></p>

        </td>

        <td>
                <h2 id="title-FA1">The Jurassic period</h2>
                <p>11 places left</p>
                <p><input type="checkbox" id="FA1" aria-labelledby="title-FA1 track1 pm2 FA1-
att">
                 <label id="FA1-att" for="FA1">Attend</label></p>
        </td>
</tr>


<tr>
        <th id="track2" scope="row">track 2</th>
        <td>
                <h2 id="title-TM2">The Cretaceous period</h2>
                <p>18 places left</p>
                <p><input type="checkbox" id="TM2" aria-labelledby="title-TM2 track2 am1 TM2-
att">
                 <label id="TM2-att" for="TM2">Attend</label></p>
        </td>

        <td>
                <h2 id="title-TA2">The end of the dinosaurs</h2>
                <p>2 places left</p>
                <p><input type="checkbox" id="TA2" aria-labelledby="title-TA2 track2 am2 TA2-
att">
                 <label id="TA2-att" for="TA2">Attend</label></p>
        </td>

        <td>
                <h2 id="title-FM2">First discoveries of dinosaurs</h2>
                <p>2 places left</p>
                <p><input type="checkbox" id="FM2" aria-labelledby="title-FM2 track2 pm1 FM2-
att">
                 <label id="FM2-att" for="FM2">Attend</label></p>
        </td>

        <td>
                <h2 id="title-FA2">Emerging scholarship</h2>
                <p>19 places left</p>
```

```
                <p><input type="checkbox" id="FA2" aria-labelledby="title-FA2 track2 pm2 FA2-
att">
                 <label id="FA2-att" for="FA2">Attend</label></p>
        </td>
</tr>
</tbody>
</table>
```

Testing Procedure

For inputs that use `aria-labelledby`:

1. Check that ids referenced in `aria-labelledby` are unique and match the ids of the text nodes that together provide the label
2. Check that the concatenated content of elements referenced by `aria-labelledby` is descriptive for the purpose or function of the element labeled

Technique ARIA17: Using grouping roles to identify related form controls

Example
{See Above}

Testing Procedure
{See Above}

Technique G89: Providing expected data format and example[227]

Examples
The following HTML form control for a date indicates in the label that the date must be in day-month-year format, not month-day-year as many users in the United States may assume.

```
<label for="date">Date (dd-mm-yyyy)</label>
  <input type="text" name="date" id="date" />
```

Testing Procedure
1. Identify form controls that will only accept user input data in a given format.
2. Determine if each of the form controls identified in 1 provides information about the expected format.

Technique G184: Providing text instructions at the beginning of a form or set of fields that describes the necessary input[228]

Example 1
A business networking site allows users to post descriptions of jobs they have held. The form to gather the information includes fields for the company name, job title, from and to dates, and job description. At the top of the form are the following instructions:

- Enter requested information about the position you wish to add to your profile. Dates should be entered in mm/dd/yyyy format."

Example 2
A corporate directory allows users to customize information such as telephone number and job responsibilities by editing their profile. At the top of the form are the following instructions:

- You can modify the information in any field. When you select Finish, your changes will be saved and you will have the opportunity to publish your profile. Should you decide that you don't want to keep your changes, select the Cancel button.

[227] https://www.w3.org/WAI/WCAG21/Techniques/general/G89.html
[228] https://www.w3.org/WAI/WCAG21/Techniques/general/G184.html

- You cannot edit the information that is displayed as system text in your profile (i.e., not contained in a field). This information has been obtained from an corporate human resources information. If you find something is incorrect or out of date that you cannot edit, select the help icon next to the information to find out how to correct it.
- Phone numbers may contain numbers and dashes (-) only.
- Required fields are marked with an asterisk (*) and must be filled in to complete the form.

Testing Procedure
1. Identify form controls that will only accept user input data in a given format.
2. Determine if instructions are provided at the top of the form about the expected format of each of the form controls identified in 1.

Technique G162: Positioning labels to maximize predictability of relationships[229]

Testing Procedure
For each form field on the Web page:
1. Check that the form field has a visible label.
2. If the form field is a checkbox or radio button, check that the label is immediately after the field.
3. If the form field is not a checkbox or radio button, check that the label is immediately before the field.

Technique G83: Providing text descriptions to identify required fields that were not completed

Example
{See Above}

Testing Procedure
{See Above}

Technique H90: Indicating required form controls using label or legend[230]

Example 1: Using text to indicate required state
The text field in the example below has the explicit label of "First name (required):".
The label element's for attribute matches the id attribute of the input element and the label text indicates that the control is required.

```
<label for="firstname">First name (required):</label>
<input type="text" name="firstname" id="firstname" />
```

Some authors abbreviate "required" to "req." but there is anecdotal evidence that suggests that this abbreviation is confusing.

Example 2: Using an asterisk to indicate required state
The text field in the example below has an explicit label that includes an asterisk to indicate the control is required. It is important that the asterisk meaning is defined at the start of the form. In this example, the asterisk is contained within a abbr element to allow for the asterisk character to be styled so that it is larger than the default asterisk character, since the asterisk character can be difficult to see for those with impaired vision.

CSS:
```
.req {font-size: 150%}
```

HTML:
```
<p> Required fields are marked with an asterisk (<abbr class="req"
title="required">*</abbr>).</p>
<form action="http://www.test.com" method="post">
<label for="firstname">First name <abbr class="req" title="required">*</abbr>:</label>
```

[229] https://www.w3.org/WAI/WCAG21/Techniques/general/G162.html
[230] https://www.w3.org/WAI/WCAG21/Techniques/html/H90.html

```
<input type="text" name="firstname" id="firstname" />
```

Example 3: Using an image to indicate required state
The text field in the example below has an explicit label that includes an image to indicate the control is required. It is important that the image meaning is defined at the start of the form.

```
<p><img src="req_img.gif" alt="Required Control" /> indicates that the form control is required</p>
<form action="http://www.test.com" method="post">
<label for="firstname">First name <img src="req_img.gif" alt="Required Control" />:</label>
<input type="text" name="firstname" id="firstname" />
...
```

Example 4: Indicating required state for groups of radio buttons or check box controls
Radio buttons and checkboxes are treated differently than other interactive controls since individual radio buttons and checkboxes are not required but indicates that a response for the group is required. The methods used in examples 1-3 apply to radio buttons and checkboxes, but the indication of the required state should be placed in the legend element instead of the label element.

```
<fieldset>
<legend>I am interested in the following (Required):</legend>
<input type="checkbox" id="photo" name="interests" value="ph">
<label for="photo">Photography</label></br>
<input type="checkbox" id="watercol" name="interests" checked="checked" value="wa">
<label for="watercol">Watercolor</label></br>
<input type="checkbox" id="acrylic" name="interests" checked="checked" value="ac">
<label for="acrylic">Acrylic</label>
…
</fieldset>
```

Testing Procedure

Technique H44: Using label elements to associate text labels with form controls[231]

Example 1: A text input field
The text field in this example has the explicit label of "First name:". The label element's for attribute matches the id attribute of the input element.

```
<label for="firstname">First name:</label>
<input id="firstname" name="firstname" type="text">
```

Example 2: A checkbox

```
<input checked id="markuplang" name="computerskills" type="checkbox">
<label for="markuplang">HTML</label>
```

Example 3: A group of radio buttons
A small, related group of radio buttons with a clear description and labels for each individual element.

```
<h1>Doughnut Selection</h1>
<form action="/buy-doughnuts" method="post">
  <fieldset>
    <legend>Pick the doughnut you would like</legend>
    <input id="dn-choc" name="flavor" type="radio" value="chocolate">
    <label for="dn-choc">Chocolate</label>
    <input id="dn-cream" name="flavor" type="radio" value="cream">
    <label for="dn-cream">Cream Filled</label>
    <input id="dn-raspberry" name="flavor" type="radio" value="raspberry">
    <label for="dn-raspberry">Raspberry Filled</label>
  </fieldset>
  <input type="submit" value="Purchase Your Doughnut">
```

[231] https://www.w3.org/WAI/WCAG21/Techniques/html/H44.html

```
</form>
```

Testing Procedure

For all `input` elements of type `text`, `file` or `password`, for all `textarea` elements, and for all select elements in the Web page:

1. Check that there is a `label` element that identifies the purpose of the control before the `input`, `textarea`, or `select` element
2. Check that the `for` attribute of the `label` element matches the id of the `input`, `textarea`, or `select` element.
3. Check that the `label` element is visible.

For all input elements of type checkbox or radio in the Web page:

1. Check that there is a `label` element that identifies the purpose of the control after the `input` element.
2. Check that the `for` attribute of the label element matches the `id` of the `input` element.
3. Check that the `label` element is visible.

Technique H71: Providing a description for groups of form controls using `fieldset` and `legend elements`[232]

Example 1: A multiple choice test

This example shows a test item with one question and five possible answers. Each answer is represented by a radio button (input type="radio"). The radio buttons are contained within a fieldset. The test question is tagged with the legend element. Each field has the same name attribute, indicating these radio buttons are related, and should be grouped as shown. Also note that while the name attributes have the same value, the id attributes' values must be unique.

```
<fieldset>
  <legend>The play <cite>Hamlet</cite> was written by:</legend>
  <div>
    <input checked="checked" id="shakesp" name="hamlet" type="radio" value="a">
    <label for="shakesp">William Shakespeare</label>
  </div>
  <div>
    <input id="austen" name="hamlet" type="radio" value="b">
    <label for="austen">Jane Austen</label>
  </div>
  <div>
    <input id="gbshaw" name="hamlet" type="radio" value="c">
    <label for="gbshaw">George Bernard Shaw</label>
  </div>
  <div>
    <input id="woolf" name="hamlet" type="radio" value="d">
    <label for="woolf">Virginia Woolf</label>
  </div>
  <div>
    <input id="dickens" name="hamlet" type="radio" value="e">
    <label for="dickens">Charles Dickens</label>
  </div>
</fieldset>
```

Example 2: A set of checkboxes

The User Profile page for a Web site allows users to indicate their interests by selecting multiple checkboxes. Each checkbox (input type="checkbox") has a label. The checkboxes are contained within a fieldset, and the legend element contains the prompt for the entire group of checkboxes.

```
<fieldset>
  <legend>I am interested in the following (check all that apply):</legend>
  <div>
    <input id="photo" name="interests" type="checkbox" value="ph">
    <label for="photo">Photography</label>
  </div>
  <div>
    <input checked="checked" id="watercol" name="interests" type="checkbox" value="wa">
    <label for="watercol">Watercolor</label>
```

[232] https://www.w3.org/WAI/WCAG21/Techniques/html/H71.html

```
  </div>
  <div>
    <input checked="checked" id="acrylic" name="interests" type="checkbox" value="ac">
    <label for="acrylic">Acrylic</label>
  </div>
</fieldset>
```

Example 3: Logically related controls
In this example, form fields for residential and postal addresses are distinguished by the value of the legend in each fieldset grouping.

```
<form action="/adduser" method="post">
  <fieldset>
    <legend>Your Residential Address</legend>
    <div>
      <label for="raddress">Address:</label>
      <input autocomplete="street-address" id="raddress" name="raddress" type="text">
    </div>
    <div>
      <label for="rzip">Postal/Zip Code:</label>
      <input autocomplete="postal-code" id="rzip" name="rzip" type="text">
    </div>
  </fieldset>
  <fieldset>
    <legend>Your Postal Address</legend>
    <div>
      <label for="paddress">Address:</label>
      <input autocomplete="street-address" id="paddress" name="paddress" type="text">
    </div>
    <div>
      <label for="pzip">Postal/Zip Code:</label>
      <input autocomplete="postal-code" id="pzip" name="pzip" type="text">
    </div>
  </fieldset>
</form>
```

Testing Procedure
For groups of related controls where the individual labels for each control do not provide a sufficient description, and an additional group level description is needed:
1. Check that the group of logically related `input` or `select` elements are contained within `fieldset` elements.
2. Check that each `fieldset` has a legend element that includes a description for that group.

Technique G167: Using an adjacent button to label the purpose of a field[233]

Example 1: A search function
A Web page contains a text field where the user can enter search terms and a button labeled "Search" for performing the search. The button is positioned right after the text field so that it is clear to the user that the text field is where to enter the search term.

Example 2: Picking a form
A user in the United States must fill in a form. Since the laws and requirements are different in different states within the United States, the user must select the version of a form for their state of residence. A dropdown list allows the user to pick a state. The adjacent button is labeled "Get Form for State." Pressing the button takes the user to the Web page containing the form for the selected state.

Testing Procedure
For a field and a button using this technique:
1. Check that the field and button are adjacent to one another in the programmatically determined reading sequence.
2. Check that the field and button are visually rendered adjacent to one another.

[233] https://www.w3.org/WAI/WCAG21/Techniques/general/G167.html

** *Note:* The techniques at the end of the above list should be considered "last resort" and only used when the other techniques cannot be applied to the page. The earlier techniques are preferred because they increase accessibility to a wider user group.**

3.3.3 Error Suggestion (Level AA)[234] – This Success Criterion is created with the intention of ensuring that users are provided with relevant advice for correcting an input error whenever it is feasible to do so. The term "input error" is defined by the (WCAG) as "information submitted by the user that is not accepted" by the system. Information that is necessary but not provided by the user is an example of the type of information that is not accepted. Another type of information that is not accepted is information that is provided by the user but does not adhere to the required data format or authorized values.

Situation A: If a mandatory field contains no information:

Technique G83: Providing text descriptions to identify required fields that were not completed

Example
{See Above}

Testing Procedure
{See Above}

Technique ARIA2: Identifying a required field with the aria-required property[235]

Example 1: The required property is indicated by an asterisk placed in the label element

```
<form action="#" method="post">
  <p>Note: * denotes a required field</p>
  <div>
    <label for="usrname">Login name *:</label>
    <input aria-required="true" autocomplete="username" id="usrname" type="text">
  </div>
  <div>
    <label for="pwd">Password *:</label>
    <input aria-required="true" autocomplete="current-password" id="pwd" type="password">
  </div>
  <div>
    <input type="submit" value="Login">
  </div>
</form>
```

Example 2: The required property is indicated by the word "required" placed next to the label element

```
<form action="#" method="post">
  <div>
    <label for="fname">First name:</label> <span>(required)</span>
    <input aria-required="true" autocomplete="given-name" id="fname" type="text">
  </div>
  <div>
    <label for="mname">Middle name:</label> <span>(required)</span>
    <input autocomplete="additional-name" id="mname" type="text">
  </div>
  <div>
    <label for="lname">Last name:</label> <span>(required)</span>
    <input aria-required="true" autocomplete="family-name" id="lname" type="text">
  </div>
  <div>
    <label for="email">Email address:</label> <span>(required)</span>
    <input aria-required="true" autocomplete="email" id="email" type="text">
```

[234] https://www.w3.org/WAI/WCAG21/Understanding/error-suggestion.html
[235] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA2.html

```
  </div>
  <div>
    <label for="zip_post">Zip / Postal code:</label> <span>(required)</span>
    <input aria-required="true" autocomplete="postal-code" id="zip_post" type="text">
  </div>
  <div>
    <input type="submit" value="Next Step">
  </div>
</form>
```

Example 3: Required fields are indicated by a red border around the fields and a star icon rendered via CSS using content::after

This example also uses custom radio buttons with role=radio but the script to make the span actually work like radio buttons is not included in this example. The CSS properties are available below the form.

```
<form action="#" method="post">
  <label data-required="true" for="acctnum">Account Number</label>
  <input aria-required="true" id="acctnum" type="text">

  <p data-required="true" id="radio_label">
    Please send an alert when balance exceeds $3,000.
  </p>

  <ul aria-required="true" aria-labelledby="radio_label" role="radiogroup">
    <li aria-checked="false" id="rb1" role="radio" tabindex="0">Yes</li>
    <li aria-checked="false" id="rb2" role="radio" tabindex="-1">No</li>
  </ul>
</form>
```

Related CSS style definition for this example:

```
[aria-required=true] {
  border: red thin solid;
}
[data-required=true]::after {
  content: url('/iconStar.gif');
}
```

Testing Procedure

For each control which is shown via presentation to be required.

1. Check whether the aria-required attribute is present:
2. Check whether the value of the aria-required attribute is the correct required state of the user interface component.

Situation B: If information for a field is required to be in a specific data format:

Technique ARIA18: Using `aria-alertdialog` to Identify Errors[236]

Example 1: Alert dialog

This example shows how a notification using `role="alertdialog"` can be used to notify someone they have entered invalid information.

```
<div role="alertdialog" aria-labelledby="alertHeading" aria-describedby="alertText">
<h1 id="alertHeading">Error</h1>
<div id="alertText">Employee's Birth Date is after their hire date. Please verify the birth date
and hire date.</div>
<button>Save and Continue</button>
<button>Return to page and correct error</button>
</div>
```

Working example: Alert dialog.

[236] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA18.html

Testing Procedure

1. Trigger the error that causes the alertdialog to appear.
2. Determine that the alertdialog contains at least one focusable control, and the focus moves to that control when the alertdialog opens.
3. Determine that the tab order is constrained within the alertdialog while it is open, and when the dialog is dismissed, the focus moves back to the position it had before the dialog opene, if possible.
4. Examine the element with `role="alertdialog"` applied.
5. Determine that either the `aria-label` or `aria-labelledby` attribute has been correctly used to give the alertdialog an accessible name.
6. Determine that the contents of the alertdialog identifies the input error.
7. Determine whether contents of the alertdialog suggests how to fix the error.

Technique G85: Providing a text description when user input falls outside the required format or values

Example

{See Above}

Testing Procedure

{See Above}

Technique G177: Providing suggested correction text[237]

Examples

- A form field requires the user to input a length of time that could range from days to years. The user enters the number "6". The server returns the form as the user had submitted it and also includes a suggested text next to the form field: "Error detected. Did you mean: 6 days, 6 weeks, 6 months or 6 years?"
- The user enters an incorrectly spelled city name. The server returns the form as the user had submitted it and also includes a message at the top of the form informing the user of the error and a link to a list of city names that the user may have meant, as determined by comparing their original input to a database of city names.
- A bus route trip planner allows users to enter their origin an destination, allowing users to enter street addresses, intersections and city landmarks. When a user enters "Kohl," they are prompted with a list of search results with similar matches that reads, "Your search for 'Kohl' returned the following". A select box follows the prompt lists, "Kohl Center," "Kohl's Dept. Store-East" and "Kohl's Dept. Store-West" as options the user can choose from.
- A search runs a spell check on input and provides a link of alternatives if a spelling error is detected. When the user clicks on the link, the search is automatically resubmitted with the correct spelling.

Testing Procedure

1. Identify form fields where correct text could be inferred from incorrect text.
2. Fill out the form, deliberately filling in the identified form fields with incorrect text.
3. Check that the user is presented with suggestions for the correct text.
4. Check that the suggestions are provided next to the form field or a link to the suggestions is provided close to the form field.

Technique SCR18: Providing client-side validation and alert

Example

{See Above}

Testing Procedure

{See Above}

Technique SCR32: Providing client-side validation and adding error text via the DOM

[237] https://www.w3.org/WAI/WCAG21/Techniques/general/G177.html

Example {See Above} ***Testing Procedure*** {See Above}

Situation C: Information provided by the user is required to be one of a limited set of values:

Technique ARIA18: Using aria-alertdialog to Identify Errors ***Example*** {See Above} ***Testing Procedure*** {See Above}
Technique G84: Providing a text description when the user provides information that is not in the list of allowed values ***Example*** {See Above} ***Testing Procedure*** {See Above}
Technique G177: Providing suggested correction text ***Example*** {See Above} ***Testing Procedure*** {See Above}
Technique SCR18: Providing client-side validation and alert ***Example*** {See Above} ***Testing Procedure*** {See Above}
Technique SCR32: Providing client-side validation and adding error text via the DOM ***Example*** {See Above} ***Testing Procedure*** {See Above}

3.3.4 Error Prevention (Legal, Financial, Data) (Level AA)[238] – This Success Criterion is developed to aid users who have impairments in avoiding undesirable outcomes that may result from making mistakes while carrying out tasks that cannot be undone. The placement of an order to acquire shares through a brokerage account or the purchase of non-refundable plane tickets are two examples of the types of

[238] https://www.w3.org/WAI/WCAG21/Understanding/error-prevention-legal-financial-data.html

financial transactions that might have significant ramifications. In the event that a user inputs the date of the flight in the incorrect format, the end result may be a ticket that cannot be modified in any way. If the client is not careful when determining the number of stock shares that they want to purchase, they can wind up purchasing more stock than they had planned. Both of these mistakes have the potential to be quite costly due to the fact that they involve transactions that take place immediately and cannot be altered once they have taken place.

Situation A: If an application causes a legal transaction to occur, such as making a purchase or submitting an income tax return:

Technique G164: Providing a stated time within which an online request (or transaction) may be amended or canceled by the user after making the request[239]

Example 1: Online shopping
An online shopping Web site lets users cancel purchases up to 24 hours after they have been made. The Web site explains their policy, and includes a summary of the policy on the purchase receipt emailed to the user. After 24 hours, the purchase will be shipped to the user and can no longer be canceled.

Example 2: Custom orders
A Web site sells custom sports jackets that are made to order. The customer chooses the fabric and provides body measurements for the tailor. The Web site gives customers up to three days to change or cancel an order. Once the material has been cut to the customer's specifications, it is no longer possible to change or cancel the order. The company policy is described on its Web site.

Testing Procedure
1. Check that the Web page describes the time period to cancel or change an order.
2. Check that the Web page describes the process for canceling or changing an order.

Technique G98: Providing the ability for the user to review and correct answers before submitting[240]

Examples
- An online banking application provides multiple steps to complete a transfer of funds between accounts as follows:

1. Select "transfer from" account
2. Select "transfer to" account
3. Enter transfer amount

A summary of the transaction is provided showing the from and to accounts and the transfer amount. The user can select a button to either complete the transaction or cancel it.

- A testing application provides multiple pages of questions. At any time, the user can choose to return to previously completed sections to review and change answers. A final page is displayed providing buttons to either submit the test answers or review answers.

Testing Procedure
In a testing application or one that causes financial or legal transactions to occur and that also collects data from users in multiple steps:
1. Check that the user is prompted to review and confirm data.
2. If user data are collected in multiple steps, the user is allowed to return to previous steps to review and change data.
3. Determine if a summary of all data input by the user is provided before the transaction is committed and a method is provided to correct errors if necessary.

Technique G155: Providing a checkbox in addition to a submit button[241]

[239] https://www.w3.org/WAI/WCAG21/Techniques/general/G164.html
[240] https://www.w3.org/WAI/WCAG21/Techniques/general/G98.html
[241] https://www.w3.org/WAI/WCAG21/Techniques/general/G155.html

Examples

- An online bank service allows users to transfer money between accounts in different currencies. Because exchange rates are constantly in flux, the money cannot be re-exchanged at the same rate if the user discovers an error in their input after the transaction has been carried out. In addition to the "account from", "account to", and "amount" fields, there is a checkbox with a label "I have checked that the amount I wish to transfer is correct". If this checkbox is not selected when the user submits the form, the transaction is not carried out and the user is notified. If the checkbox is selected, the (irreversible) transaction is carried out.
- An online payment system stores user bank account information in order to process payments. There is an elaborate procedure for users to enter new accounts and verify that they are the owner. There is the facility to delete old accounts, but if an account is accidentally deleted, it would be difficult to reinstate it, and the transaction history with that account would be lost. Therefore, on pages that allow users to delete accounts, there is a checkbox with the label "I confirm that I wish to delete this account." If this checkbox is not selected when the user submits the form, the account is not deleted and the user is given an error message. Only if the checkbox is selected is the account deleted.
- A checkout form on a shopping site includes a form that collects order, shipping and billing information. After submitting the form, the user is taken to a page where the information they have submitted is summarized for review. Below the summary, a checkbox with the label "I have reviewed and confirmed that this data is correct" is shown. The user must mark the checkbox and activate a "complete order" button in order to complete the transaction.

Testing Procedure

For user input pages that cause irreversible transactions to occur:

1. Check that a checkbox indicating user confirmation of the input or action is provided in addition to the submit button.
2. Check that if the checkbox is not selected when the form is submitted, the input is rejected and the user is prompted to review their entry, select the checkbox, and resubmit.

Situation B: If an action causes information to be deleted:

Technique G99: Providing the ability to recover deleted information[242]

Examples

- A Web application allows users to set up folders and store data within them. Each folder and data item is accompanied by a checkbox to mark it for action, and two buttons, one to move and one to delete. If the user selects the delete button by mistake, large amounts of data could be lost. The application presents the data as deleted to the user right away but schedules it for actual deletion in one week. During the week, the user may go into a "deleted items" folder and request any folder or data item awaiting actual deletion to be restored.

Testing Procedure

1. Identify functionality that allows deleting content
2. Delete content and attempt to recover it.
3. Check if deleted information can be recovered.

Technique G168: Requesting confirmation to continue with selected action

Example 1: Airline travel

An online travel Web site lets users create travel itineraries that reserve seats with different airlines. Users may look up, amend and cancel their current itineraries. If the user needs to cancel their travel plans, they find the itinerary on the Web page and delete it from their list of current itineraries. This action results in the cancellation of their seat reservations and is not reversible. The user is informed that the selected action will cancel their current seat reservations and that it may not be possible to make a comparable booking on the same flights once this action has been taken. The user is asked to confirm or cancel the deletion of the itinerary.

Example 2: Webmail

[242] https://www.w3.org/WAI/WCAG21/Techniques/general/G99.html

A Webmail application stores a user's email on a server, so that it can be accessed from anywhere on the web. When a user deletes an email message, it is moved to a trash folder from which it can be retrieved if it was deleted by accident. There is an "empty trash" command for deleting the messages in the trash folder from the server. Once the trash folder has been emptied, the messages can no longer be retrieved. Before emptying the trash folder, the user is asked to confirm or cancel deletion of the email in the trash folder.

Example 3: An online test
A form is used to collect answers for a test. When the 'submit' or 'reset' button is selected the user is presented with a web page that informs them of their choice and asks for confirmation to continue. Example 1: "You have selected to reset the form. This will delete all previously entered data and will not submit any answers. Would you like to reset the form? [yes button] [no button]" Example 2: "You have selected to submit the form. This will submit entered data as your final answers and can not be changed. Would you like to submit the form? [yes button] [no button]"

Example 4: Trading stocks
A brokerage site allows users to buy and sell stocks and other securities. If the user makes a transaction during trading hours, a dialog is presented informing the user that the transaction is immediate and irreversible, and has buttons that say "continue" and "cancel."

Testing Procedure
1. Initiate the action that can not be reversed or changed.
2. Check that a request to confirm the selected action is presented.
3. Check that the action can be confirmed and canceled.

Technique G155: Providing a checkbox in addition to a submit button

Example
{See Above}

Testing Procedure
{See Above}

Situation C: If the Web page includes a testing application:

Technique G98: Providing the ability for the user to review and correct answers before submitting

Example
{See Above}

Testing Procedure
{See Above}

Technique G168: Requesting confirmation to continue with selected action

Example
{See Above}

Testing Procedure
{See Above}

Section 4 – Robust

4.1 Compatible[243] – This final section of the manual audit under WCAG works to ensure that the target site maximized the compatibility with current and future user agents, including any assistive technologies that might someday be used. There are three criteria points that had to be addressed in order to meet level AA conformance.

4.1.1 Parsing (Level A)[244] – In order to guarantee that user agents, including assistive technology, can correctly comprehend and process material, this success criterion is created. Different user agents may render the content differently or be altogether unable to parse it if it cannot be converted into a data structure. Some user agents render poorly coded material using "repair techniques." Authors cannot assume that content will be accurately parsed into a data structure or that it will be rendered correctly by specialized user agents, including assistive technologies, unless the content is created in accordance with the rules specified in the formal grammar for that technology. This is due to the fact that repair techniques differ between user agents. The failure to provide properly nested start/end tags and flaws in element and attribute syntax in markup languages result in issues that hinder user agents from reliably interpreting the content. The Success Criteria demands that the material be able to be parsed using only the formal grammatical rules.

Technique G134: Validating Web pages

Example 1: Validating HTML
HTML pages include a document type declaration (sometimes referred to as `!DOCTYPE` statement) and are valid according to the HTML version specified by the document type declaration. The developer can use off-line or online validators (see Resources section) to check the validity of the HTML pages.

Example 2: Validating XML
SVG, SMIL and other XML-based documents reference a Document Type Definition (DTD) or other type of XML schema. The developer can use online or off-line validators (including validation tools built into editors) to check the validity of the XML documents.

Example 3: Batch validation with Ant
The `xmlvalidate` task of Apache Ant can be used for batch validation of XML files. The following Apache Ant target is a simple example for the validation of files with the extension .xml in the directory dev\\Web (relative to the Ant build file).

```
<target name="validate-xml">
  <xmlvalidate lenient="no">
    <fileset dir="dev/web" includes="*.xml" />
  </xmlvalidate>
</target>
```

Testing Procedure
For HTML, SGML-based and XML-based technologies:
1. Load each page or document into a validating parser.
2. Check that no validation errors are found.

For other technologies:
1. Follow the validation procedure defined for the technology in use, if any exists.

Technique G192: Fully conforming to specifications

Examples

[243] https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=131%2C132%2C133%2C134%2C141%2C142%2C143%2C144%2C145%2C1410%2C1411%2C1412#compatible

[244] https://www.w3.org/WAI/WCAG21/Understanding/parsing.html

- A page is created with care to make sure that all technologies are used according to specification. It is run through a validator and all identified errors are corrected. Specification requirements that can not be identified by validation are also checked and any failures are corrected.

Testing Procedure
1. Check that all technologies are used according to specification

** NOTE: While validators can be great tools for catching errors, they usually cannot catch all cases where content fails to fully conform to a specification. **

Technique H88: Using HTML according to spec

Testing Procedure
For each HTML or XHTML page:
1. Check that the page uses only elements, attributes, and attribute values that are defined in the relevant specification.
2. Check that elements, attributes, and values are used in the manner prescribed by the relevant specification.
3. Check that the page can be parsed correctly, according to the rules of the relevant specification.

NOTE: Check #1 and #3 are most easily checked with page validation tools. Check #2 can be checked with the assistance of heuristic evaluation tools though manual judgment is usually required.

Ensuring that Web pages can be parsed by using one of the following techniques:

Technique H74: Ensuring that opening and closing tags are used according to specification[245] AND Technique H93: Ensuring that `id` attributes are unique on a Webpage[246] AND Technique H94: Ensuring that elements do not contain duplicate attributes[247]

Examples for H74:
HTML pages include a document type declaration (sometimes referred to as `!DOCTYPE` declaration). A developer can use offline or online validators (see the Resources section) to check that all id attribute values are unique and that opening and closing tags are used according to the specification.

Testing Procedure for H74:
1. Check that there are closing tags for all elements with required closing tags.
2. Check that there are no closing tags for all elements where closing tags are forbidden.
3. Check that opening and closing tags for all elements are correctly nested.

Examples for H93:
HTML pages include a document type declaration (sometimes referred to as `!DOCTYPE` statement). The developer can use offline or online validators (see Resources below) to check that id attributes values are only used once on a page. The W3C validador, for example, will report ID "X already defined" when it encounters the second use of an id value.

Testing Procedure for H93:
1. Check that all id attribute values are unique on the web page.

Examples for H94:
HTML pages include a document type declaration (sometimes referred to as `!DOCTYPE` statement). The developer can use offline or online validators (see Resources below) to check that attributes are only used once on an element. The W3C validador, for example, will report "duplicate specification of attribute X" when it encounters the second definition of the same attribute on an element.

Testing Procedure for H94:
1. Check that no attribute occurs more than once on any element

[245] https://www.w3.org/WAI/WCAG21/Techniques/html/H74.html
[246] https://www.w3.org/WAI/WCAG21/Techniques/html/H93.html
[247] https://www.w3.org/WAI/WCAG21/Techniques/html/H94.html

Technique H75: Ensuring that Web pages are well-formed[248]

Example 1
XML files include a document type declaration, a xsi:schemaLocation attribute or other type of reference to a schema. The developer can use off-line or online validators, an XML editor or an IDE with XML support (see Resources below) to check well-formedness.

Example 2
When XML files do not include a document type declaration, a xsi:schemaLocation attribute or a processing instruction referencing a schema even though there is a schema for them, the relevant schema is specified by a command line instruction, a user dialog or a configuration file, and the XML files are checked against the schema.

Example 3
When XML files do not include a document type declaration, a xsi:schemaLocation attribute or a processing instruction referencing a schema even though there is a schema for them, the namespace is dereferenced to retrieve a schema document or resource directory (Resource Directory Description Language: RDDL), and the XML files are checked against the schema.

Example 4
When a Website generates XML dynamically instead of serving only static documents, a developer can use XMLUnit, XML Test Suite or a similar framework to test the generated XML code.

Testing Procedure
1. Load each file into a validating XML parser.
2. Check that there are no well-formedness errors.

4.1.2 Name, Role, Value (Level A)[249] – This Success Criterion is developed with the intention of guaranteeing that Assistive Technologies (AT) are able to acquire data about user interface controls in the content, activate (or set), and monitor the state of these controls. When the standard rules for accessible technology are applied, this operation becomes straightforward. If the user interface elements are implemented in line with the specification, then it will be determined that this clause's requirements have been satisfied. However, if custom controls are developed or interface elements are programmed (in code or script) to perform a different role and/or function than is typical, then additional precautions need to be taken to ensure that the controls permit assistive technologies to control them and provide vital information to them. This is because the controls will need to be accessible to the assistive technologies in order for them to work properly.

Situation A: If using a standard user interface component in a markup language (e.g., HTML):

Technique ARIA14: Using aria-label to provide an invisible label where a visible label cannot be used

Example 1: A close button (X) in a pop-up box
On a page, a link displays a pop-up box (a div) with additional information. The 'close' element is implemented as a button containing merely the letter 'x'. The property aria-label="close" is used to provide an accessible name to the button.

```
<div id="box">
   This is a pop-up box.
   <button  aria-label="Close"  onclick="document.getElementById('box').style.display='none';"
class="close-button">X</button>
</div>
```

Working example: Close button example.

[248] https://www.w3.org/WAI/WCAG21/Techniques/html/H75.html
[249] https://www.w3.org/WAI/WCAG21/Understanding/name-role-value.html

Example 2: A phone number with multiple fields

```
<div role="group" aria-labelledby="groupLabel">
  <span id="groupLabel>Work Phone</span>
  +<input type="number" aria-label="country code">
  <input type="number" aria-label="area code">
  <input type="number" aria-label="subscriber number">
</div>
```

Testing Procedure
For elements that use `aria-label`:
1. Check that the value of the `aria-label` attribute properly describes the purpose of an element where user input is required

Technique ARIA16: Using `aria-labelledby` to provide a name for user interface controls

Example
{See Above}

Testing Procedure
{See Above}

Technique G108: Using markup features to expose the name and role, allow user-settable properties to be directly set, and provide notification of changes[250]

Examples
Example 1: A Web page written in HTML or XHTML uses standard form controls, and identifies the form control using the title attribute. The user agent makes information about these controls, including the name, available to assistive technology through the DOM and through a platform-specific Accessibility API.

Testing Procedure
1. Visually inspect the markup or use a tool.
2. Check that proper markup is used such that the name and role, for each user interface component can be determined.
3. Check that proper markup is used such that the user interface components that accept user input can all be operated from AT.

Technique H91: Using HTML form controls and links

Example
{See Above}

Testing Procedure
{See Above}

Technique H44: Using label elements to associate text labels with form controls

Example
{See Above}

Testing Procedure
{See Above}

Technique H64: Using the `title` attribute of the `iframe` element[251]

[250] https://www.w3.org/WAI/WCAG21/Techniques/general/G108.html
[251] https://www.w3.org/WAI/WCAG21/Techniques/html/H64.html

Example 1: Using the `title` attribute with an `iframe` to describe the `iframe's` content

```
<!doctype html>
<html lang="en">
<head>
   <title>A document using an iframe</title>
</head>
...
<iframe src="banner-ad.html" name="ad-iframe" title="Advertisement"></iframe>
...
</html>
```

Testing Procedure

1. Check each `iframe` element in the HTML source code for the presence of a `title` attribute.
2. Check that the `title` attribute contains text that describes the `iframe's` content.

Technique H65: Using the title attribute to identify form controls when the label element cannot be used

Example 1: A pulldown menu that limits the scope of a search

A search form uses a pulldown menu to limit the scope of the search. The pulldown menu is immediately adjacent to the text field used to enter the search term. The relationship between the search field and the pulldown menu is clear to users who can see the visual design, which does not have room for a separate visible label.
The title attribute is used to identify the select menu. The title attribute can be spoken by screen readers or displayed as a tool tip for people using screen magnifiers. There must be visual information on the page which allows sighted users to understand the purpose of the form controls and user agents.

```
<label for="searchTerm">Search for:</label>
<input id="searchTerm" type="text" size="30" value="" name="searchTerm">
<select title="Search in" id="scope">
…
</select>
```

Example 2: Input fields for a phone number

A Web page contains controls for entering a phone number in the United States, with three fields for area code, exchange, and last four digits.

```
<fieldset><legend>Phone number</legend>
<input id="areaCode" name="areaCode" title="Area Code"
type="text" size="3" value="" >
<input id="exchange" name="exchange" title="First three digits of phone number"
type="text" size="3" value="" >
<input id="lastDigits" name="lastDigits" title="Last four digits of phone number"
type="text" size="4" value="" >
</fieldset>
```

Example 3: A Search Function

A Web page contains a text field where the user can enter search terms and a button labeled "Search" for performing the search. The title attribute is used to identify the form control and the button is positioned right after the text field so that it is clear to the user that the text field is where the search term should be entered.

```
<input type="text" title="Type search term here"/> <input type="submit" value="Search"/>
```

Example 4: A data table of form controls

A data table of form controls needs to associate each control with the column and row headers for that cell. Without a title (or off-screen label) it is difficult for non-visual users to pause and interrogate for corresponding row or column header values using their assistive technology while tabbing through the form.

For example, a survey form has four column headers in first row: Question, Agree, Undecided, Disagree. Each following row contains a question and a radio button in each cell corresponding to answer choice in the three columns. The title attribute for every radio button contains the information necessary to identify the control.

Testing Procedure
For all form controls that are not associated with a label element:

1. Check that the control has a title attribute
2. Check that the purpose of the form control is clear to users who can see the control.
3. Check that the title attribute identifies the purpose of the control and that it matches the apparent visual purpose.

Technique H88: Using HTML according to spec

Example
{See Above}

Testing Procedure
{See Above}

Situation B: If using script or code to re-purpose a standard user interface component in a markup language:

- Exposing the names and roles, allowing user-settable properties to be directly set, and providing notification of changes using one of the following techniques:

Technique ARIA16: Using `aria-labelledby` to provide a name for user interface controls

Example 1: Labelling a simple text field
The following is an example of aria-labelledby used on a simple text field to provide a label in a situation where there is no text available for a dedicated label but there is other text on the page that can be used to accurately label the control.

```
<input name="searchtxt" type="text" aria-labelledby="searchbtn">
<input name="searchbtn" id="searchbtn" type="submit" value="Search">
```

Example 2: Labelling a slider
Below is an example of aria-labelledby used to provide a label for a slider control. In this case the label text is selected from within a longer adjacent text string. Please note that this example is simplified to show only the labeling relationship; authors implementing custom controls also need to ensure that controls meet other success criteria.

```
<p>Please select the <span id="mysldr-lbl">number of days for your trip</span></p>
<div id="mysldr" role="slider" aria-labelledby="mysldr-lbl"></div>
```

Example 3: A label from multiple sources
The following example of aria-labelledby with multiple references uses the label element. For additional detail on concatenating multiple sources of information into a label with aria-labelledby, please view the technique Using ARIA labelledby to concatenate a label from several text nodes.

```
<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>
```

Note: The use of the label element is included for a number of reasons. If the user clicks on the text of the label element, the corresponding form field will receive focus, which makes the clicking target larger for people with dexterity problems. Also the label element will always be exposed via the accessibility API. A span could have been used (but if so, it should receive a tabindex="-1" so that it will be exposed via the accessibility API in all versions of Internet Explorer). However, a span would lose the advantage of the larger clickable region.

Testing Procedure

For each user interface control element where an `aria-labelledby` attribute is present:

1. Check that the value of the `aria-labelledby` attribute is the id of an element or a space separated list of `ids` on the web page.
2. Check that the text of the referenced element or elements accurately labels the user interface control.

Situation C: If using a standard user interface component in a programming technology:

Technique G135: Using the accessibility API features of a technology to expose names and roles, to allow user-settable properties to be directly set, and to provide notification of changes[252]

Examples

- A Web page uses java to create an applet. Java swing objects (e.g., pushbutton) are used because they have accessibility properties built in that can be accessed from assistive technology written in Java and, with the Java Access Bridge, those written in other languages that use the Accessibility API of the operating system. The author fills in the values for the components and the result is accessible to AT.

Testing Procedure

1. Render content using an accessible User Agent
2. Use an Accessibility Tool designed for the Accessibility API of the User agent to evaluate each user interface component
3. Check that name and role for each user interface component are found by the tool.

Situation D: If creating your own user interface component in a programming language:

Technique G10: Creating components using a technology that supports the accessibility API features of the platforms on which the user agents will be run to expose the names and roles, allow user-settable properties to be directly set, and provide notification of changes[253]

Examples

- A Web page uses java to create an applet. A group of authors wants to create an entirely new type of interface component so they cannot use existing Java objects. They use Java swing classes to create their component because the Java swing classes already have provisions for connecting to different accessibility APIs. Using the Java swing classes they are able to create an interface component that exposes its name and role, is able to be set by AT and alerts AT to any updates.
- A Web page uses an original ActiveX control that is written in the C++ programming language. The control is written to explicitly support the Microsoft Active Accessibility (MSAA) API to expose information about accept commands. The control then interacts directly with assistive technology running the user agent on systems that support MSAA.

Testing Procedure

1. Render content using an accessible User Agent.
2. Use an Accessibility Tool designed for the Accessibility API of the User agent to evaluate each user interface component.
3. Check that name and role for each user interface component is found by the tool.
4. Change the values on the component.
5. Check that the Accessibility tool is alerted.
6. Check that the component works with assistive technologies.

Technique ARIA4: Using a WAI-ARIA role to expose the role of a user interface component[254]

Example 1: A simple toolbar

[252] https://www.w3.org/WAI/WCAG21/Techniques/general/G135.html
[253] https://www.w3.org/WAI/WCAG21/Techniques/general/G10.html
[254] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA4.html

A toolbar containing three buttons. The div element has a role of "toolbar", and the img elements have "button" roles:

```
<div role="toolbar"
      tabindex="0"
      id="customToolbar"
      onkeydown="return optionKeyEvent(event);"
      onkeypress="return optionKeyEvent(event);"
      onclick="return optionClickEvent(event);"
      onblur="hideFocus()"
      onfocus="showFocus()"
      >
      <img src="img/btn1.gif"
           role="button"
           tabindex="-1"
           alt="Home"
           id="b1"
           title="Home">
      <img src="img/btn2.gif"
           role="button"
           tabindex="-1"
           alt="Refresh"
           id="b2"
           title="Refresh">
     <img src="img/btn3.gif"
           role="button"
           tabindex="-1"
           alt="Help"
           id="b3"
           title="Help">
 </div>
```

The Authoring Practices Toolbar Pattern provides a working example of a toolbar.

Example 2: A Tree Widget
A basic tree widget. Note the use of the roles "tree", "treeitem", and "group" to identify the tree and its structure.

Here is a simplified excerpt from the code:

```
<ul role="tree" tabindex="0">
  <li role="treeitem">Birds</li>
  <li role="treeitem">Cats
    <ul role="group">
      <li role="treeitem">Siamese</li>
      <li role="treeitem">Tabby</li>
    </ul>
  </li>
  <li role="treeitem">Dogs
    <ul role="group">
      <li role="treeitem">Small Breeds
        <ul role="group">
          <li role="treeitem">Chihuahua</li>
          <li role="treeitem">Italian Greyhound</li>
          <li role="treeitem">Japanese Chin</li>
        </ul>
      </li>
      <li role="treeitem">Medium Breeds
        <ul role="group">
          <li role="treeitem">Beagle</li>
          <li role="treeitem">Cocker Spaniel</li>
          <li role="treeitem">Pit Bull</li>
        </ul>
      </li>
      <li role="treeitem">Large Breeds
        <ul role="group">
          <li role="treeitem">Afghan</li>
          <li role="treeitem">Great Dane</li>
```

```
          <li role="treeitem">Mastiff</li>
        </ul>
      </li>
    </ul>
  </li>
</ul>
```

The Authoring Practices Tree View Pattern provides a working example of a tree.

Testing Procedure

Technique ARIA5: Using WAI-ARIA state and property attributes to expose the state of a user interface component[255]

Example 1: A toggle button
A widget with role button acts as a toggle `button` when it implements the attribute `aria-pressed`. When `aria-pressed` is true, the button is in a "pressed" state. When `aria-pressed` is false, it is not pressed. If the attribute is not present, the button is a simple command button.
The following snippet from The Open Ajax Accessibility Examples, Example 38, shows WAI-ARIA mark-up for a toggle button that selects bold text:

```
<li id="bold1"
    class="toggleButton"
    role="button"
    tabindex="0"
    aria-pressed="false"
    aria-labelledby="bold_label"
    aria-controls="text1">
    <img        src="http://www.oaa-accessibility.org/media/examples/images/button-bold.png"
alt="bold text" align="middle">
</li>
```

The `li` element has a role of "`button`" and an "`aria-pressed`" attribute. The following excerpt from the Javascript for this example updates the value of the "`aria-pressed`" attribute:

```
                              /**
   * togglePressed() toggles the aria-pressed atribute between true or false
   *
   * @param ( id object) button to be operated on
   *
   * @return N/A
   */
  function togglePressed(id) {

    // reverse the aria-pressed state
    if ($(id).attr('aria-pressed') == 'true') {
      $(id).attr('aria-pressed', 'false');
    }
    else {
      $(id).attr('aria-pressed', 'true');
    }
  }
```

This button is available as part of the working example of Example 38 - Toolbar using inline images for visual state, on the OpenAjax Alliance site.

Example 2: A slider
A widget with role slider lets a user select a value from within a given range. The slider represents the current value and the range of possible values via the size of the slider and the position of the handle. These properties of the slider are represented by the attributes aria-valuemin, aria-valuemax, and aria-valuenow.

[255] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA5.html

The following snippet from The Open Ajax Accessibility Examples, Example 32, shows WAI-ARIA mark-up for a slider created in Javascript. Note that the javascript sets the attributes `aria-valuemin`, `aria-valuemax`, and `aria-valuenow`:

```
 var handle = '<img id="' + id + '" class="' + (this.vert == true ? 'v':'h') +'sliderHandle"
' +
    'src="http://www.oaa-accessibility.org/media/examples/images/slider_' + (this.vert ==
true ? 'v':'h') + '.png" ' + 'role="slider" ' +
    'aria-valuemin="' + this.min +
    '" aria-valuemax="' + this.max +
    '" aria-valuenow="' + (val == undefined ? this.min : val) +
           '" aria-labelledby="' + label +
           '" aria-controls="' + controls + '" tabindex="0"></img>';
```

The following excerpt from the Javascript for this example updates the value of the `"aria-valuenow"` attribute when the value of the slider handle is changed:

```
slider.prototype.positionHandle = function($handle, val) {
    ...
   // Set the aria-valuenow position of the handle
  $handle.attr('aria-valuenow', val);
   ...
  }
```

This slider is available as part of the working example of Example 32 - Slider, on the OpenAjax Alliance site.

Testing Procedure
The WAI-ARIA specification, Section 5.3, Categorization of Roles defines the required and inherited states and properties for each role.

For a user interface component using the WAI-ARIA role attribute:
1. Check that the required states and properties for the role are present.
2. Check that no WAI-ARIA states or properties that are neither required, supported, nor inherited are present.
3. Check that the state and property values are updated to reflect the current state when the user interface component changes state.

Technique ARIA16: Using aria-labelledby to provide a name for user interface controls

Example
{See Above}

Testing Procedure
{See Above}

4.1.3 Status Messages (Level AA)[256] – The purpose of this Success Criterion is to notify users of substantial content changes that are not now the center of attention while doing so in a way that do not unnecessarily interfere with their work. Beneficiaries targeted for this effort are people who use assistive technology and have vision impairments that prevent them from using screen reader capabilities. Depending on the choices of the user, assistive technology for persons who have cognitive disabilities may be able to discover a different way to signal (or even delay or suppress) status updates. This is another advantage of using such technology. The adjustments to the content that are centered on status signals are where this Success Criterion places its primary emphasis. The term "status message" is given its own definition in WCAG. The user is provided with information regarding the success or outcomes of an activity, the waiting status of an application, the progression of a process, or the presence of problems;

[256] https://www.w3.org/WAI/WCAG21/Understanding/status-messages.html

the message is not supplied by means of a change in context. These are the two most important considerations that go into deciding whether or not anything may be classified as a status message.

Situation A: If a status message advises on the success or results of an action, or the state of an application:

Technique ARIA22: Using `role=status` to present status messages[257]

Example 1: Including a search results message
After a user presses a Search button, the page content is updated to include the results of the search, which are displayed in a section below the Search button. The change to content also includes the message "5 results returned" near the top of this new content. This text is given an appropriate role for a status message. A screen reader will announce "Five results returned".

```
<div role="status">5 results returned.</div>
```

Working example: role=status on search results

Example 2: Updating the shopping cart status
After a user presses an Add to Shopping Cart button, content near the Shopping Cart icon updates to read "1 items". The container for this text (in this case a <p>) is marked with the role of status. Because it adds visual context, the shopping cart image -- with succinct and accurate ALT text -- is also placed in the container. Due to the default aria-atomic value, a screen reader will announce "Shopping cart, six items".

```
<p role="status" >
        <img src="shopping-cart.png" alt="Shopping Cart">
        <br>
        <span id="cart">0</span> items
</p>
```

Working example: role=status on a shopping cart

Testing Procedure
For each status message:
1. Check that the container destined to hold the status message has a role attribute with a value of `status` *before* the status message occurs.
2. Check that when the status message is triggered, it is inside the container.
3. Check that elements or attributes that provide information equivalent to the visual experience for the status message (such as a shopping cart image with proper ALT text) also reside in the container.

Technique G199: Providing success feedback when data is submitted successfully[258]

Examples
- A user logs into a system and gets a response indicating that: "You have successfully logged in," so they do not need to navigate through the screen to find an indicator that they are logged in, such as finding their user name, or perhaps the login link replaced with a logout link. Finding these cues can be time consuming.
- A user fills in a quiz or test and submits it. The response informs them that the test was successfully submitted, so that they don't need to navigate through data, such as a list of submitted tests, to confirm that the test is listed there.
- A visitor creates an account on a Web site. After submission of the form, feedback suggests that "Registration was successfully submitted ...," If they are automatically logged in after registration, the response also says "...and you have been logged in." If confirmation is required, the feedback includes a message such as "...an email has been sent to you to which you must reply to confirm your registration."
- A user submits a form with information directed at support staff. The feedback indicates that the "The message was successfully sent, and you should receive a reply within the next 48 hours."

[257] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA22.html
[258] https://www.w3.org/WAI/WCAG21/Techniques/general/G199.html

Testing Procedure
1. Fill in form fields with no errors.
2. Submit the form.
3. Check that a feedback message on the screen confirms that the submission was successful.

Situation B: If a status message conveys a suggestion, or a warning on the existence of an error:

Technique ARIA19: Using ARIA `role=alert` or Live Regions to Identify Errors

Example
{See Above}

Testing Procedure
{See Above}

Technique G83: Providing text descriptions to identify required fields that were not completed

Example
{See Above}

Testing Procedure
{See Above}

Technique G84: Providing a text description when the user provides information that is not in the list of allowed values

Example
{See Above}

Testing Procedure
{See Above}

Technique G85: Providing a text description when user input falls outside the required format or values

Example
{See Above}

Testing Procedure
{See Above}

Technique G177: Providing suggested correction text

Example
{See Above}

Testing Procedure
{See Above}

Technique G194: Providing spell checking and suggestions for text input[259]

Examples
- A search engine has a form field for search terms. When the form is submitted, a server-side application checks the spelling. If the spelling doesn't match any words for that language, it sends back a page with a text

[259] https://www.w3.org/WAI/WCAG21/Techniques/general/G194.html

message at the top saying "Did you mean ..." with a link to the suggested word. If the user clicks on the link the suggested term is entered into the form field and is resubmitted.

- An airline has a on online ticket purchasing application. When a user types the name of a city into the form field a dropdown menu shows the closest match to the city in the top of the menu and other suggestions below.

Testing Procedure

1. Check that there is a form field on the page.
2. Enter a misspelled word.
3. Check that a suggested spelling is presented.
4. Check that a mechanism is available to enter the suggested word into the form.

Situation C: If a status message conveys information on the progress of a process:

Technique ARIA23 : Using `role=log` to identify sequential information updates[260]

Example 1: Updating the contents of a chat conversation

Comments that users type into a chat input field are appended to the end of the chat history region. The region is marked with role of log so that new additions are announced by ATs. When each new chat message appears, a screen reader should announce its content (depending on AT/browser compatibility).

```
<div id="chatRegion" role="log" aria-labelledby="chatHeading">
        <h4 id="chatHeading">Chat History</h4>
        <ul id="conversation">
                <li>The latest chat message</li>
        </ul>
</div>
```

Working example: chatlog.html

Example 2: Updating the log of a server

An application log records time-stamped activities. The log is exposed in the app as a view, with the region marked with the role of log so that the new additions are announced by the ATs. (The default value for the aria-relevant attribute is "additions", so the removal of the old top entries due to log size limitations will not be announced.) When each new log entry is added, a screen reader announces it.

```
<div id="activityLog" role="log">
        <h4 id="logHeading">Recent activity</h4>
        <ul id="logentries"">
                <li>08:03 UserX logged off</li>
        </ul>
</div>
```

Working example: serverlog.html

Testing Procedure

On a page that contains sequentially updating information:

1. Check that the container for the information is given a role of `log`.

Using `role="progressbar"` (future link)

Technique ARIA22: Using `role=status` to present status messages AND Technique G193: Providing help by an assistant in the Web page[261]

[260] https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA23.html
[261] https://www.w3.org/WAI/WCAG21/Techniques/general/G193.html

Examples for ARIA22:
{See Above}

Testing Procedure for ARIA22:
{See Above}

Examples for G193:

- The home page of an online banking application has an embedded avatar named Vanna. She gives new online banking clients a tour of the features provided in the application. The assistant can be started and stopped and paused. The client can rewind and fast forward through the material. A text alternative of the information is available from a link next to the avatar.
- A volunteer site has a welcoming page for new volunteers. In it there is an application form. On the right side of the page there an interactive multimedia file with an avatar that explains all the features and sections of the application form.

Testing Procedure for G193:

1. Check that there is an assistant in the Web page.
2. Check that the assistant provides information to help understand the content of the page.