

Web Accessibility Training



Jon Whiting & John Northup

WebAIM.org

Resources online at:

webaim.org/presentations/2022/palci

Outline

Day 1

- Overview of WCAG 2
- Brief overview of law
- WAVE
- User experience
 - Deaf & hard of hearing
 - Low vision
 - Colorblind
 - Blind (& other screen reader users)

Day 2

- User experience, contd.
 - Photosensitive epilepsy
 - Motor disabilities
 - Cognitive and learning disabilities
- Main track: Advanced principles
 - Rules of ARIA use
 - Advanced forms
 - JavaScript
 - ARIA roles, states, and properties
- *Breakout option: Word & Acrobat Pro*

Is your web content accessible?

WCAG 2

- Web Content Accessibility Guidelines
- Versions 1.0 (1999), 2.0 (2008), and 2.1 (2018)
- WCAG 2.2 in the works
- Principles based
 - Perceivable
 - Operable
 - Understandable
 - Robust

Principle 1 – Perceivable

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

Guideline 1.1 – Text Alternatives

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.

➤ [Show techniques and failures for 1.1](#)

1.1.1 Non-text Content — Level A

All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. ➤ [Show full description](#)

 Understanding 1.1.1

➤ [Show techniques and failures for 1.1.1](#)

 SHARE |  BACK TO TOP

Guideline 1.2 – Time-based Media

Provide alternatives for time-based media.

1.2.1 Audio-only and Video-only (Prerecorded) — Level A

For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled as such: ➤ [Show full description](#)

 Understanding 1.2.1

www.w3.org/WAI/WCAG21/quickref/

Principle 1: Perceivable

Web content is made available to the senses - sight, hearing, and/or touch

Guideline 1.1 Text Alternatives

Provide text alternatives for any non-text content

Success Criteria	Recommendations
1.1.1 Non-text Content (Level A)	<ul style="list-style-type: none"> <input type="checkbox"/> All images, form image buttons, and image map hot spots have appropriate, equivalent alternative text. <input type="checkbox"/> Images that do not convey content, are decorative, or contain content that is already conveyed in text are given null alt text (alt="") or implemented as CSS backgrounds. All linked images have descriptive alternative text. <input type="checkbox"/> Equivalent alternatives to complex images are provided in context or on a separate linked page. <input type="checkbox"/> Form buttons have a descriptive value. <input type="checkbox"/> Form inputs have associated text labels. <input type="checkbox"/> Embedded multimedia is identified via accessible text. <input type="checkbox"/> Frames and iframes are appropriately titled.

Guideline 1.2 Time-based Media

Provide alternatives for time-based media

NOTE: If the audio or video is designated as an alternative to web content (e.g., an audio or sign language version of a web page, for example), then the web content itself serves as the alternative.

Success Criteria	Recommendations
1.2.1 Prerecorded Audio-only and Video-only (Level A)	<ul style="list-style-type: none"> <input type="checkbox"/> A descriptive text transcript that includes relevant auditory content is provided for non-live audio-only (audio podcasts, MP3 files, etc.). <input type="checkbox"/> A descriptive text transcript or audio description is provided for non-live video-only (e.g., video that has no audio track), unless the video is decorative.
1.2.2 Captions (Prerecorded)	<ul style="list-style-type: none"> <input type="checkbox"/> Synchronized captions are provided for non-live video (YouTube videos, etc.).

Legal requirements

- Section 508
 - US federal agencies
 - WCAG 2.0
- Americans with Disabilities Act
 - 3 Important sections – employment, state/local government, public & commercial facilities
 - No technical accessibility standard
 - Complaints reference WCAG 2.0 or 2.1
- EN 301 549
 - EU public sector
 - WCAG 2.1
- Canada
 - By province (e.g., AODA)
 - Most laws public & private-sector
 - Primarily WCAG 2.0
- Other laws reference (or do not exceed) WCAG 2

VPAT – Voluntary Product Accessibility Template

Criteria	Supporting Feature	Remarks and Explanations
2.4.2 Page Titled: Web pages have titles that describe topic or purpose.	Supported	
2.4.3 Focus Order: If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.	Supported With Exceptions	On invoking 'Generate a description for me' button in the Alt Text pane, keyboard focus falls on 'Close' button instead of moving to Description edit field.
2.4.4 Link Purpose (In Context): The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general.	Supported	
2.4.5 Multiple Ways: More than one way is available to locate a Web page	Supported	

[Templates Available for 508, WCAG, EU, and all-in-one](#)



WAVE

web accessibility evaluation tool

wave.webaim.org

User experience

Auditory Disabilities



PULL THE LEVER, KRONK.

Captioning Methods

- Type, then sync
- Stenographer or “Shadow Speaker”
- AI voice recognition





Is officially tomatoes it's prisons rations?
He look for it.

Other multimedia considerations

Audio Description

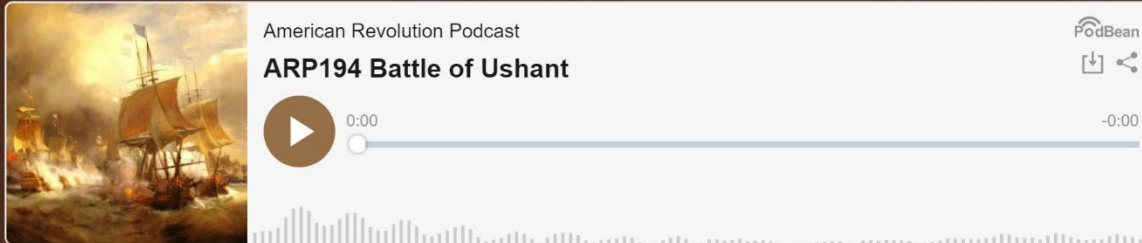
Typically a narrator that describes visual content.
Sometimes called “Descriptive Video Service” (DVS).

Avoid expensive audio descriptions by ensuring
important visual content is presented audibly.



Descriptive Transcript

ARP194 Battle of Ushant



After Britain and France went to war in the spring of 1778, America became a sideshow to the main event. Britain and France had been traditional enemies for centuries. Part of it was the whole Catholic-Protestant rift that had divided Europe. Part of it was conflicting claims over each other's countries. King George III still held the title of King of France, a claim that dated back more than 400 years. Although the British Channel kept the two kingdoms separated, there was a continuing rivalry between the two countries that simply would not end.

In the prior decades most of the fighting had been fought over colonies around the world. Britain and France traded colonies in wars back and forth. North America was only one pawn in that larger game of chess.

In the hundred years prior to this war, Britain and France had faced off in at least five major wars, totaling 39 years of fighting. These were a continuation of centuries more fighting between the two kingdoms.

In the Seven Years War, the British Navy had dominated the French at sea. That was a big reason why France lost North America. In the intervening years, France focused on rebuilding her navy to compete with the British. France, which had three times the population of Britain, thought that



Battle of Ushant

- Called “Alternative for Time-based Media” in WCAG
- Text-based presentation of important audio and visual information
 - Does not have to be text-only
- Not synchronized

Deaf-blind

- Content accessible to blind users will generally be accessible to users who are deaf-blind
 - With the exception of media
- Descriptive transcripts provide media accessibility



WCAG Requirements

Level A

- Captions
- Transcript for audio-only or video-only content
- Audio description OR transcript (if needed)

Level AA

- Audio description (if needed)
- Live audio – Captions

Level AAA

- Descriptive transcript
- Sign language
- Live audio – Transcript

Our recommendation

Level A

- Captions
- Transcript for audio-only or video-only content
- Audio description OR transcript (if needed)

Level AA

- Audio description (if needed)
- Live audio – Captions
- **Descriptive transcript**

Level AAA

- Sign language
- Live audio – Transcript

What are additional benefits of captions and transcripts?

Visual Disabilities

Low Vision

Browser Zoom

- Browser controls
 - Larger: Ctrl/command & +
 - Smaller: Ctrl/command & -
 - Reset : Ctrl/command & 0
- Zoomed content will trigger responsive breakpoints
 - Responsive design supports users with low vision who zoom page content

Quick Reference: Testing Web Content for Accessibility



Test with WAVE

- Run a report at wave.webaim.org. For very complex or non-public pages, use the WAVE Chrome or Firefox extension (wave.webaim.org/extension).
 - Watch the overview video on the WAVE homepage.
- Error icons flag known issues. Other icons identify potential problems or features. Click an icon to highlight the corresponding element and learn more using the **Reference** panel.
- Use the **Details** panel to review page issues. Click an icon to find it in the page. Uncheck icons to hide them.
- Turn off **Styles** to simplify the page view and to check the reading and navigation order.
- Click the **Code** button at the bottom of the page to see the page code with WAVE icons.

Images

- Ensure alternative text (shown in green) conveys the equivalent **content** and/or **function** of the image.
- Look for ways to replace images of text with true text.
- Content conveyed via CSS images must have a text alternative.

Headings and Regions/Landmarks

- Use the **Structure** panel in the sidebar to review.
- The main heading should usually be an `<h1>`.
- Ensure the headings and regions/landmarks reflect the page structure.
- Look for skipped heading levels (e.g., `<h2>` to `<h4>`).

Contrast

- Select the **Contrast** panel.

Check keyboard accessibility

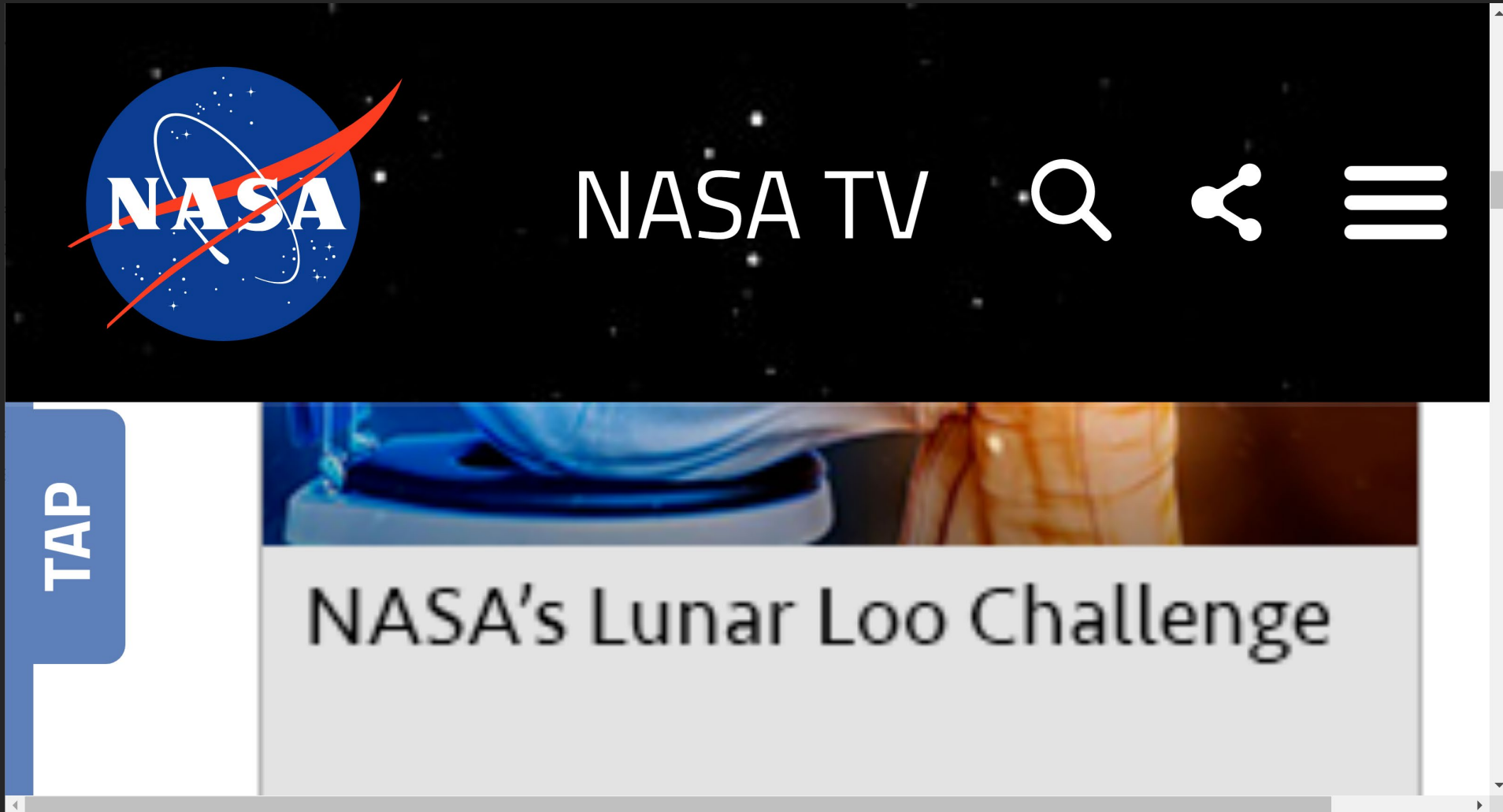
- The first time testing with Mac, press **Control + F7** to enable full keyboard accessibility.
 - In Safari, select **Preferences > Advanced > Accessibility > Press Tab to highlight each item...**
- Navigate the page using only the keyboard:
 - **Tab**: Navigate interactive elements (links, form controls, etc.)
 - **Shift + Tab**: Navigate backwards.
 - **Enter**: Activate links or buttons, submit most forms.
 - **Spacebar**: Activate checkboxes and buttons, expand a select menu, or scroll the window.
 - **Arrow keys**: Navigate radio buttons, select/dropdown menus, sliders, tab panels, tree menus, etc.
- Look for mouse-only interaction (e.g., rollover menus).
- Confirm every focusable element has a keyboard focus indicator/outline with at least 3:1 contrast.
- Ensure any “skip” links work correctly and are visible to sighted keyboard users.
- Make sure the navigation order is logical and intuitive.
- Test dialog and pop-ups. Can you navigate and close the dialog? Does focus return to a logical place?
 - Modal dialogs must maintain focus until dismissed.
 - Non-modal dialogs must close when focus is lost.
 - **Esc** should also close all dialogs and menus.

Test content scaling

- In **Chrome**, press **Ctrl/cmd** and:

webaim.org/resources/evalquickref

Browser Zoom on nasa.gov



WCAG Zoom Requirements

1.4.4 Resize text (Level AA)

- “Except for captions and images of text, text can be resized without assistive technology up to 200% without loss of content or functionality.”
- Zoom to 200% is supported.

1.4.10 Reflow (Level AA - WCAG 2.1)

- “Content can be presented without loss of information or functionality...” and without horizontal scrolling (except when necessary) at 400% zoom with viewport width at 1280 pixels.

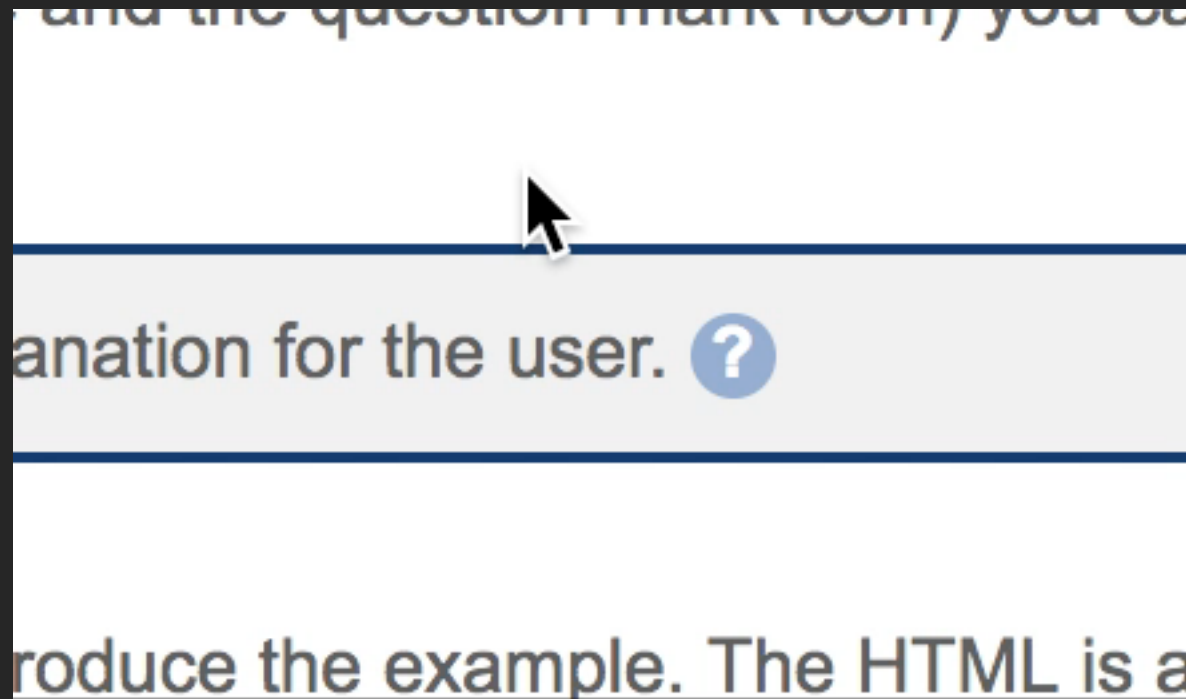
Images of Text



Evaluate page zoom

- Ctrl/command & +, -, 0
- Zoom to 200%
- Resize to 1280px and Zoom to 400%
 - Any horizontal scrolling?
- Test the mobile menu

Mouse Hover interactions



WCAG 2.1 – Content on Hover or Focus (Level AA)

Content that appears on hover and focus must be:

- “Dismissible...without moving pointer hover or keyboard focus”: Esc key dismisses
- “Hoverable”: Doesn’t disappear when moving the pointer to the new content
- “Persistent”: Visible until you move mouse away or dismiss it

Or use **mouse click** and **Enter/Spacebar** to trigger interactions.

Provide Sufficient Contrast

[WebAIM contrast article](#)

WCAG Contrast Formula

$$(L1 + 0.05) / (L2 + 0.05)$$

where

$$L = 0.2126 * R + 0.7152 * G + 0.0722 * B$$

where

R, G, and B =

$$\begin{aligned} &R_{sRGB} \leq 0.03928 \text{ then } R = R_{sRGB} / 12.92 \\ &\text{else } R = ((R_{sRGB} + 0.055) / 1.055) ^ 2.4 \end{aligned}$$

where

$$R_{sRGB} \leq R_{8bit} / 255$$

21:1

21:1

Level AA Contrast Examples

4.5:1

Gray (#767676) on White

Purple (#CC21CC) on White

Red (#E30000) on Yellow (#FFFF00)

3:1 – “Large” text

≥18pt (24px)

BOLD: ≥14pt (18.67px)

“Images of text ”

50 most played
songs by genre



WebAIM Contrast Checker

webaim.org/resources/contrastchecker/



I

I

AM

AM

TEXT

TEXT



I

I

AM

AM

TEXT

TEXT

Exceptions

“Pure Decoration”



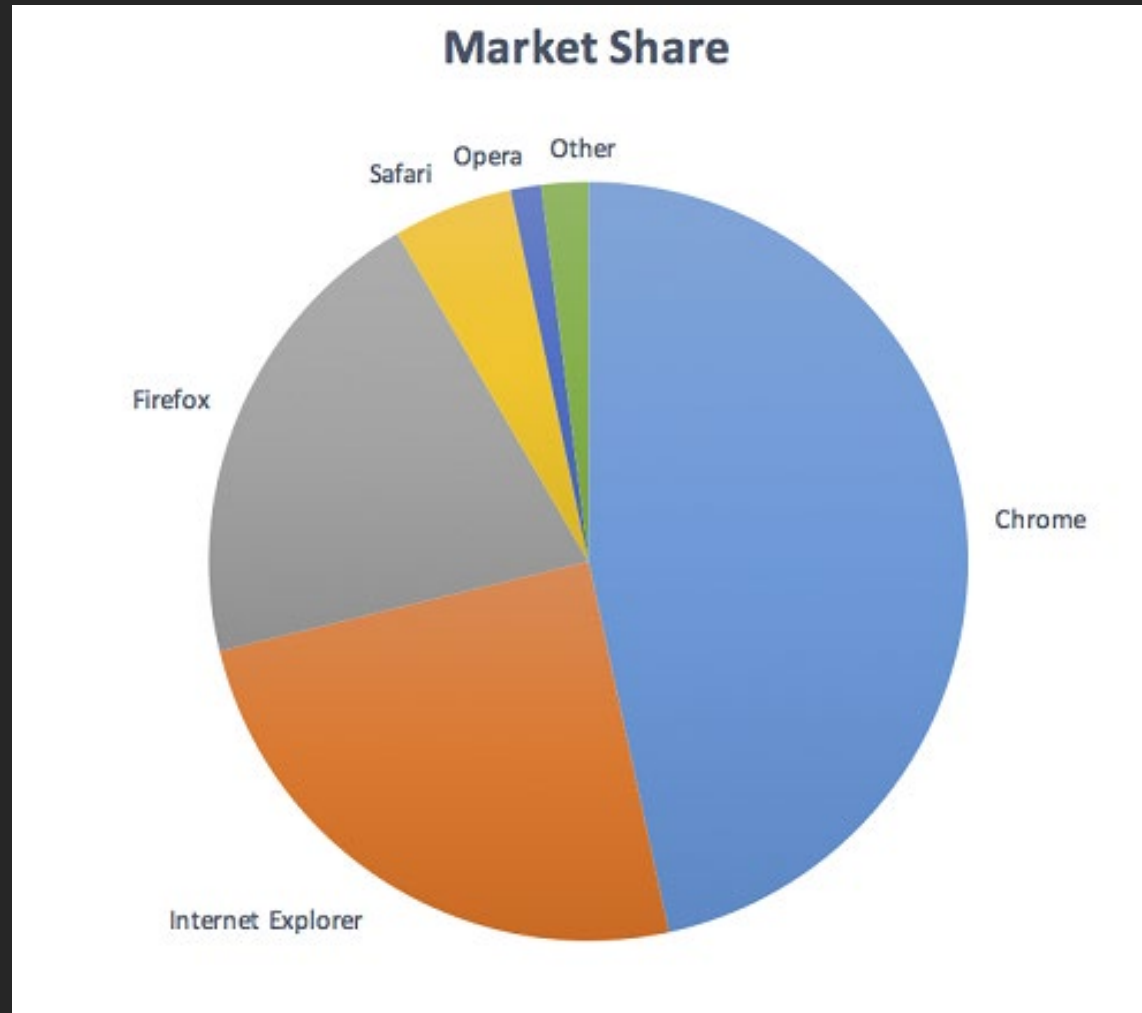
"Inactive"

“Logotypes”

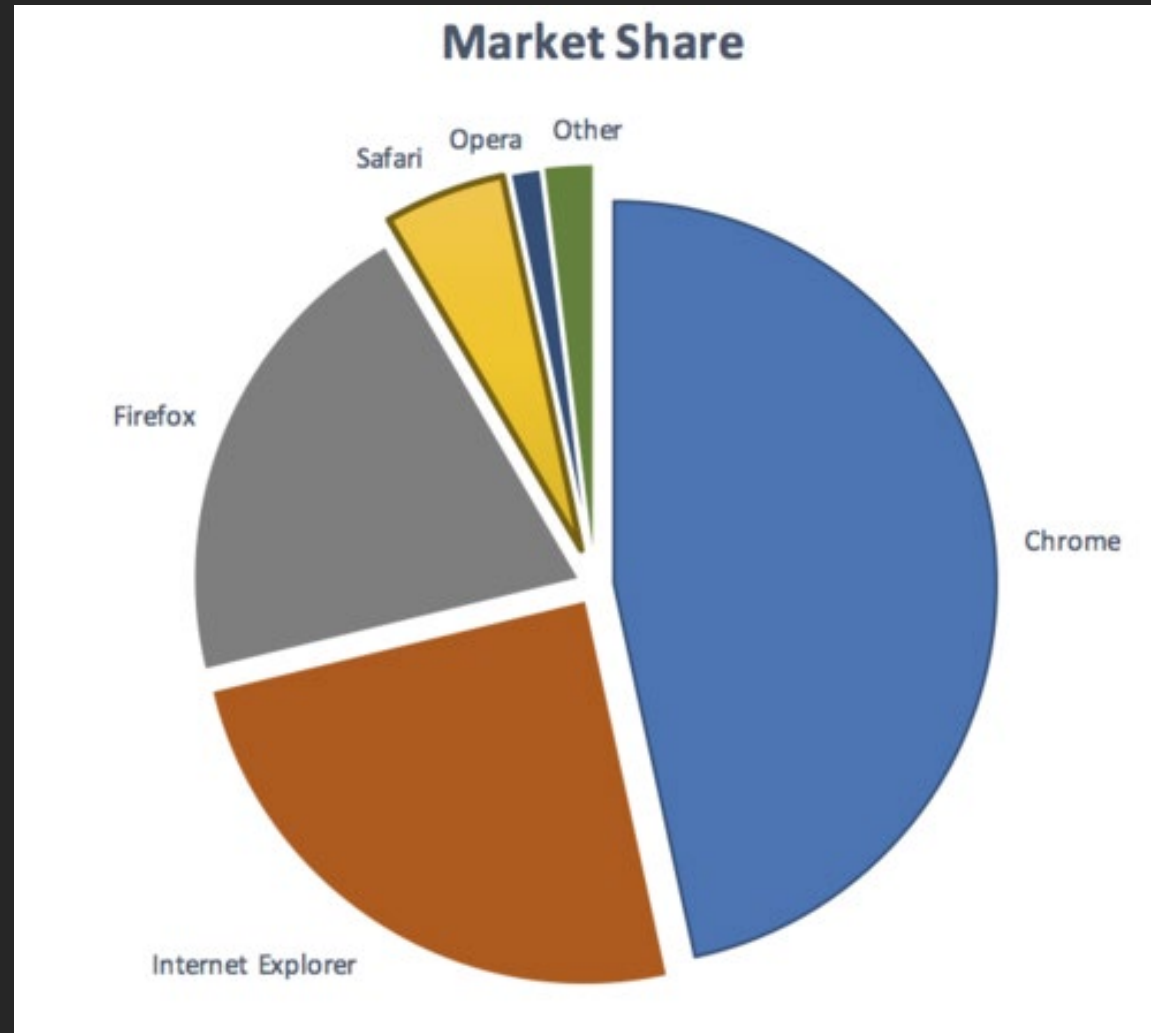
50 most played
songs by genre



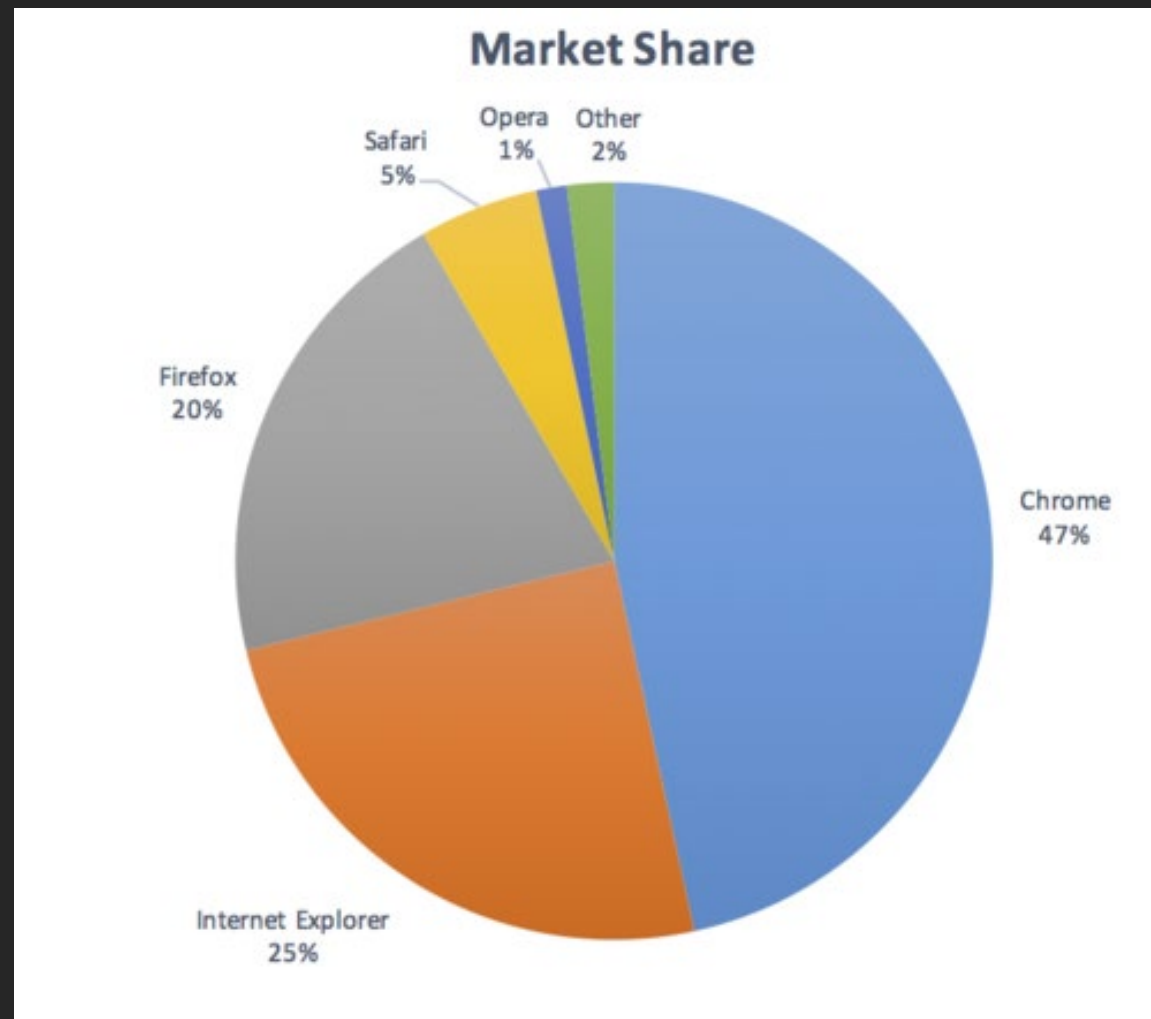
WCAG 2.1 – 3:1 Contrast of “Graphical Objects”



“Against Adjacent color(s)”



“...required to understand”



“...required to understand”



3:1 Contrast of UI “Boundaries”

☐

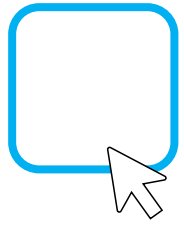
I have read and agree to the [Terms of Use](#)

WCAG 2.1 – 3:1 contrast of keyboard focus indicators



Follow your interests.

You must test contrast of different states

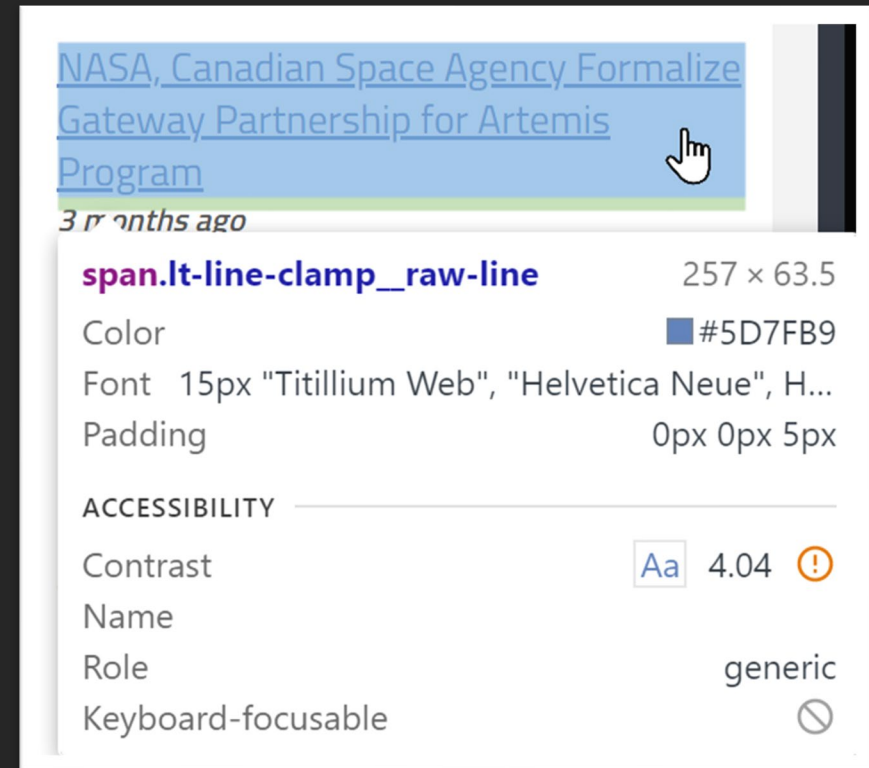
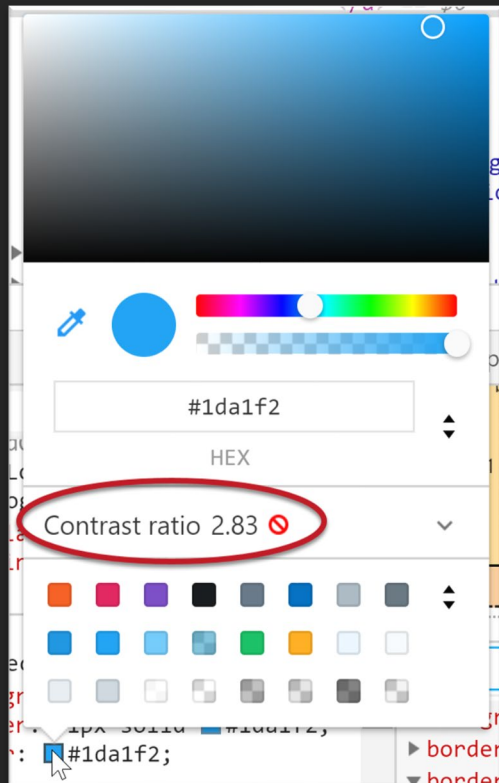


I have read and agree to the [Terms of Use](#)



I have read and agree to the [Terms of Use](#)

Chrome Developer Tools



Evaluating contrast with Chrome DevTools

Color Blindness



WCAG 1.4.1 – Use of Color

The green mushrooms listed here are okay to eat. The red mushrooms are poisonous.

- Amanita
- Chanterelle
- Porcini
- Shiitake
- Tylopilus

Icons (with alternative text)

- Amanita 
- Chanterelle
- Porcini
- Shiitake
- Tylopilus 

Separate lists

Edible

- Chanterelle
- Porcini
- Shiitake

Poisonous

- Amanita
- Tylopilus

Bold or Italicized text?

- *Amanita*
- Chanterelle
- Porcini
- Shiitake
- *Tylopilus*

1. Ensure sufficient contrast
2. Do not rely on color alone

Requirements for non-underlined links

- A 3:1 contrast ratio between link text and non-link text
- Link must present a non-color cue (typically underline) on mouse hover AND keyboard focus
 - Not possible in most electronic documents (e.g., Word and PowerPoint)

You must agree to the **Terms of Use**

You must agree to the Terms of Use

You must agree to the **Terms of Use**

webaim.org/resources/linkcontrastchecker

You must agree to the Terms of Use

You must agree to the [Terms of Use](#)

and link becomes underlined on hover/focus.

You must agree to the **Terms of Use**

You must agree to the **Terms of Use**

You **must** agree to the **Terms of Use**

You must agree to the [Terms of Use](#)

Evaluate contrast

1. WAVE contrast tab – Automatically-detected text
2. WebAIM contrast checker or ColorZilla - Images
3. Chrome DevTools - States, non-solid backgrounds

Screen Reader Users

- Not all screen reader users are completely blind
- Most have low vision
 - 45% of respondents to our low-vision survey use a screen reader
- Some users have a reading or cognitive disability
- Focus on structure and semantics



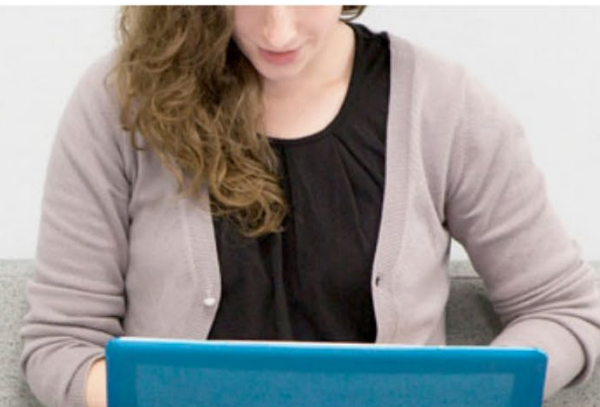
Search:



[Introduction to Web Accessibility](#)



[WebAIM Training](#)



We have web accessibility in mind

Expanding the potential of the web for people with disabilities by empowering individuals and organizations to create accessible content.

[Accessibility Training](#)



Whether here in Utah or on-site at your organization, WebAIM can provide web and document training to fit your needs.

[Accessible Site Certification](#)



As a respected third party accessibility expert, WebAIM can evaluate and certify your site to established web accessibility guidelines.

[StrategicA11y](#)



The Strategic Web Accessibility Workshop helps participants optimize their organization's accessibility.

[Evaluation and Reporting](#)



We can provide reports to help you know how accessible your site is and how to make it better.

Community

[WebAIM Blog](#)

[Newsletter](#)

[E-mail Discussion List](#)

[Twitter](#)

h2 Main Navigation

- [Services](#)
- [Articles](#)
- [Resources](#)
- [Projects](#)
- [Community](#)

Search:

[Introduction to Web Accessibility](#)

[WebAIM Training](#)

h1 We have web accessibility in mind

Expanding the potential of the web for people with disabilities by empowering individuals and organizations to create accessible content.

h2 [Accessibility Training](#)

Whether here in Utah or on-site at your organization, WebAIM can provide web and document training to fit your needs.

h2 [Accessible Site Certification](#)

As a respected third party accessibility expert, WebAIM can evaluate and certify your site to established web accessibility guidelines.

h2 [Technical Assistance](#)

Need assistance implementing accessibility? WebAIM's expert staff can provide the assistance you need.

h2 [Evaluation and Reporting](#)

We can provide reports to help you know how accessible your site is and how to make it better.

h2 Community

- [WebAIM Blog](#)
- [Newsletter](#)
- [E-mail Discussion List](#)
- [Twitter](#)

Navigation

Search

Header

Main

Structure & Semantics

Recommended screen readers for testing

Desktop

- Windows
 - [JAWS](#) – 54%, \$1000+
 - [NVDA](#) – 31%, Free
- [VoiceOver for Mac](#) – 7%

Mobile

- [VoiceOver for iOS](#) – 72%
- [Talkback for Android](#) – 25%

Use WebAIM tutorials to learn commands and practice common tasks

Screen reader testing

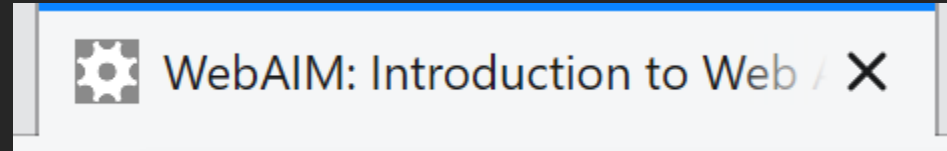
- Don't try to be an “expert”
 - 10 shortcuts to get started, 20 is about all you'll ever need
- Screen readers read things differently and can misbehave
- Verbosity settings impact the amount and types of content that are read
- Focus on document structure, navigation, forms, and dynamic content

Screen Reader Demo

Headings

- The most common method for finding information on a page
 - 69% of WebAIM screen reader user survey respondents
- Headings should describe a section of content
 - Should never be empty
- Usually one `<h1>` per page
- Do not skip heading levels (e.g., `<h2>` to `<h4>`)
 - Can skip backward (e.g., `<h4>` to `<h2>`)

Page Title



```
<title>WebAIM: Introduction to  
Web Accessibility</title>
```

- Often the first thing read
- Should usually match, or be similar to, the first-level heading
- Must be descriptive
- Should be succinct

HTML Regions

- `<header>`
- `<nav>`
- `<main>`
- `<aside>`
- `<footer>`

Screen reader users can navigate by region

ARIA

Accessible Rich Internet Applications

ARIA expands the vocabulary of HTML

Rule #1 of ARIA Use

“If you can use a native HTML element or attribute...then do so.”

[Notes on Using ARIA in HTML](#)

Rule #1, paraphrased

If you can use HTML, don't use ARIA

ARIA Landmarks

- `<header>` - `role="banner"`
- `<nav>` - `role="navigation"`
- `<main>` - `role="main"`
- `<aside>` - `role="complementary"`
- `<footer>` - `role="contentinfo"`
- `??? – role="search"`

Regions/Landmarks

- We recommend HTML regions (Rule #1)
 - `<header role="banner">` is unnecessary
- “Region” / “Landmark” are generally synonymous
- All content in the page should be within a region
- More doesn’t always mean better

Lists

- `` for numbers, sequence, or hierarchy
- `` for parallel or equal importance
 - Can you reorder the items in the list?
- `<dl>` for “name/value pairs”
 - E.g., FAQ or glossary

EASY SHORTBREAD



Ingredients

- 1 cup butter, softened
- 1/2 cup sugar
- 2 1/2 cups flour

Directions

1. Preheat the oven to 300 F.
2. Cream butter and sugar.
3. Gradually stir flour into the creamed mixture until blended.
4. Pat the dough in the bottom of an ungreased 9x13 baking pan.
5. Bake at 300 F for 30 to 40 minutes, until just lightly browned.
6. Remove from oven and pierce all over with a fork.

Baking Terms

Softened butter

Butter left at room temperature for approximately 20 minutes.

Cream butter and sugar

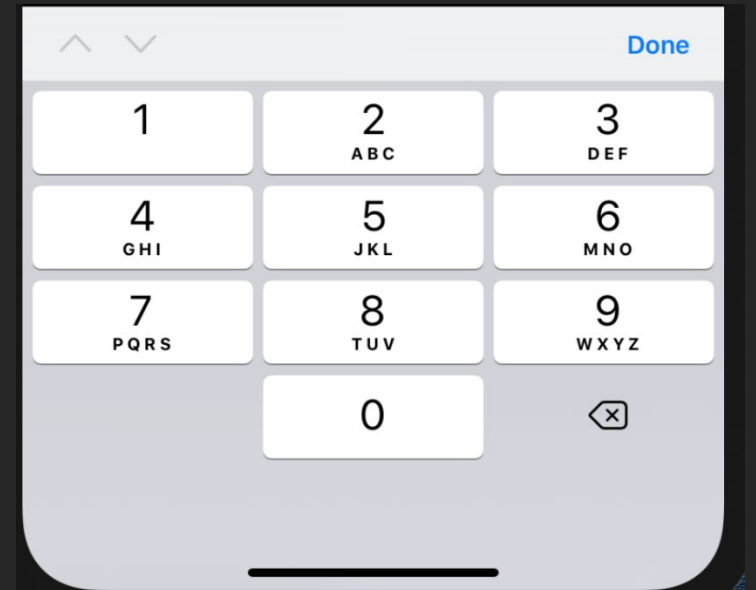
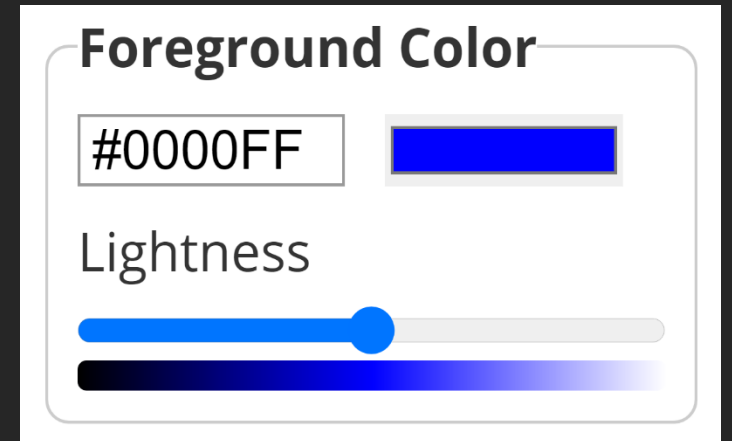
Mix butter and sugar together until it is light and fluffy.



Forms

HTML Input Types

- Use the correct input type
 - `checkbox` (multiselect)
 - `radio` (single select)
- Native inputs have better accessibility than custom inputs
 - E.g., `color` and `range`
- Mobile devices may present custom keyboards
 - e.g., phone keypad for `type="tel"`



Form Labels

First Name:

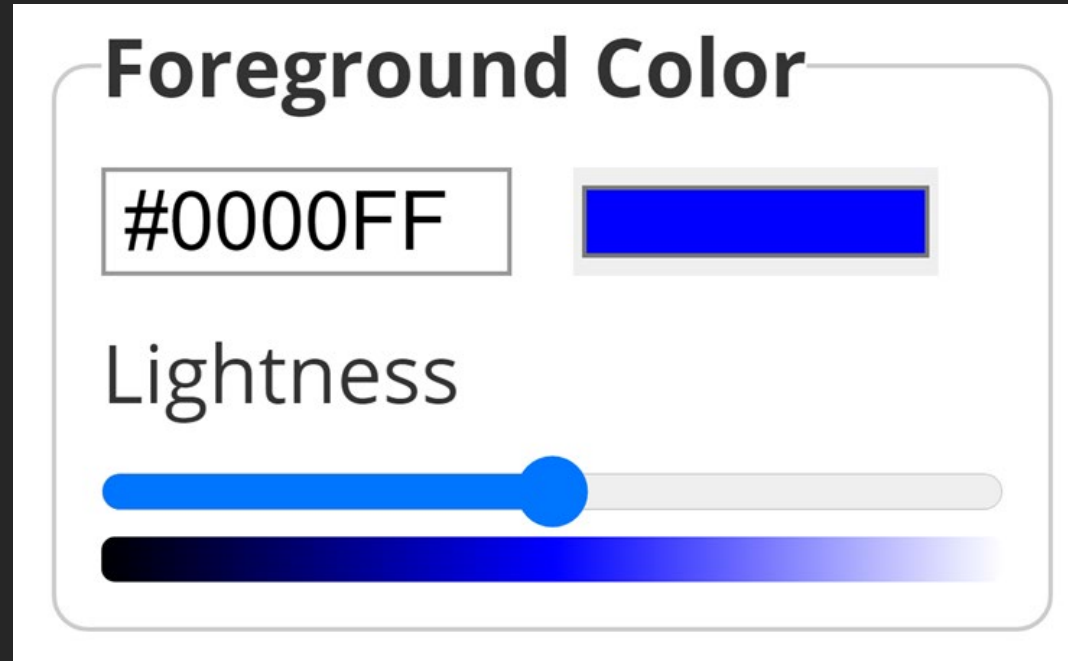
```
<label for="fname">First Name:</label>  
<input type="text" id="fname"...>
```

Text boxes, text areas, select menus, checkboxes,
radio buttons, and HTML5 input types.

Implicit Form Labels

```
<label>First Name:  
<input type="text"></label>
```

<fieldset> and <legend>



The image shows a web form titled "Foreground Color" enclosed in a rounded rectangle. Inside the form, there is a text input field containing the hex color code "#0000FF" and a color picker button showing a blue square. Below these, the word "Lightness" is displayed above a horizontal slider. The slider has a blue track and a blue knob positioned at approximately the 50% mark. Below the slider is a color gradient bar transitioning from black on the left to white on the right.

```
<fieldset>
  <legend>Foreground Color</legend>
  ...
</fieldset>
```

Evaluate forms and labels

Tables

Data tables

Class Schedule

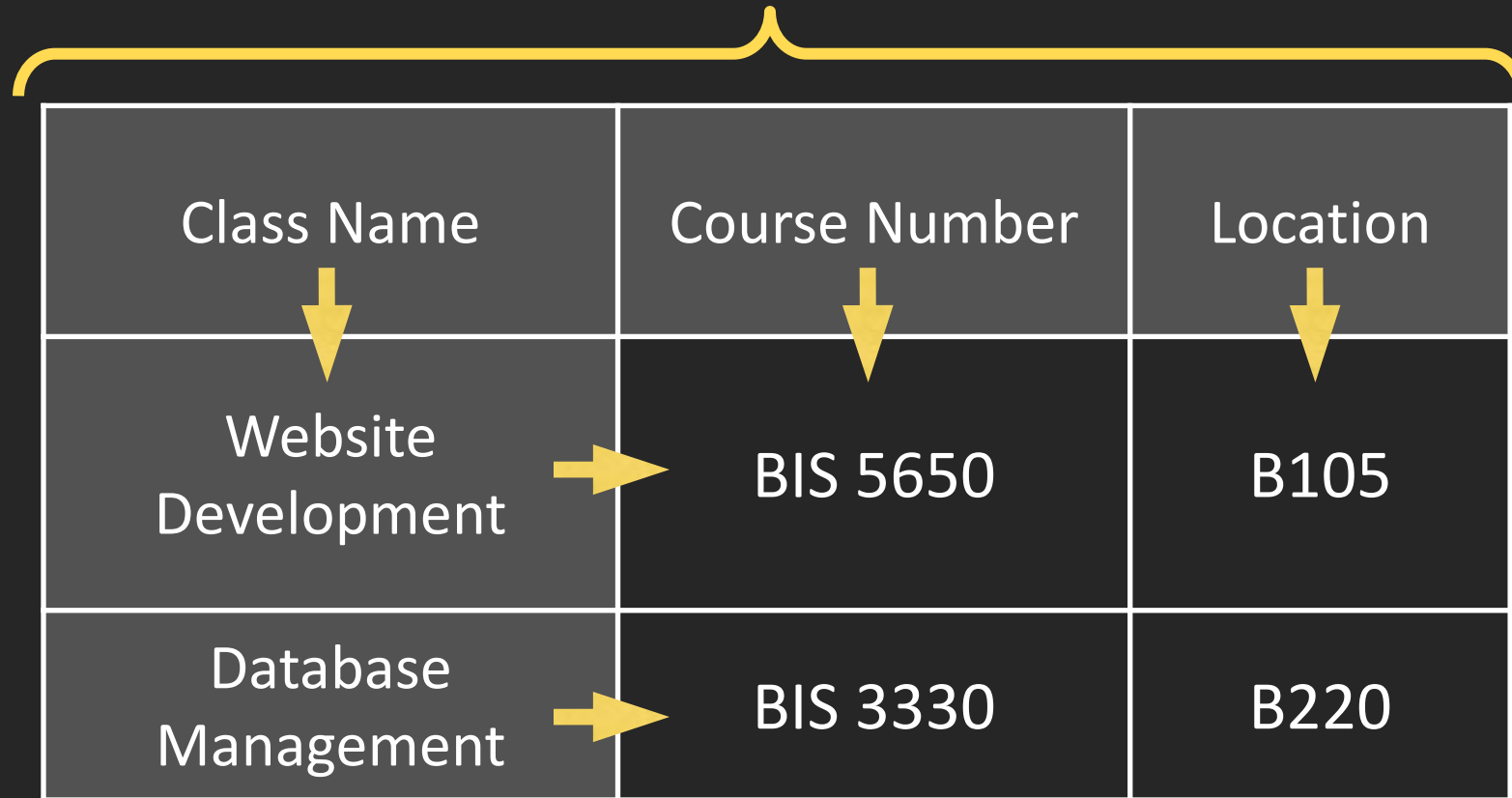
Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220

Data tables

```
<table>
<caption>Class Schedule</caption>
<tr>
  <th scope="col">Class Name</th>
  <th scope="col">Course Number</th>
  <th scope="col">Location</th>
</tr>
<tr>
  <th scope="row">Advanced Website Development</th>
  <td>BIS 5650</td>
  <td>B105</td>
</tr>
...
```


Structure for screen readers

Class Schedule



Class Name	Course Number	Location
Website Development	BIS 5650	B105
Database Management	BIS 3330	B220

[Table example](#)

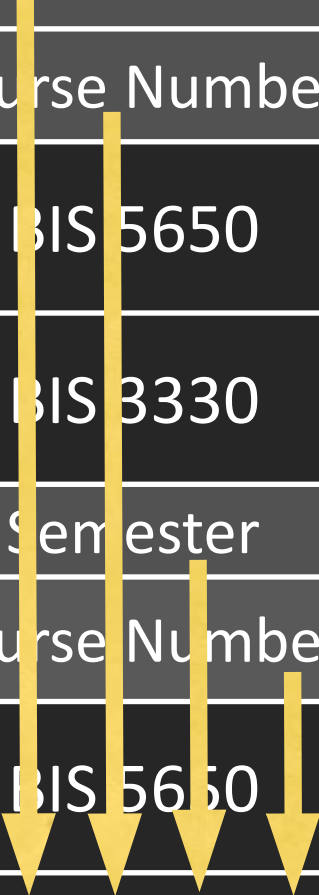
Fall Semester		
Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220

Fall Semester

Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220

Fall Semester		
Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220
Winter Semester		
Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220

Fall Semester		
Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220
Winter Semester		
Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220



Fall Semester

Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220

Winter Semester

Class Name	Course Number	Location
Advanced Website Development	BIS 5650	B105
Database Management	BIS 3330	B220

Language

“Most people today can hardly conceive of life without the internet. Some have argued that no other single invention has been more revolutionary since Gutenberg's printing press in the 1400s. Now, at the click of a mouse, the world can be “at your fingertips”—that is, if you can use a mouse... and see the screen... and hear the audio—in other words, if you don't have a disability of any kind.”



Language of Page (Level A)

```
<html lang="en">
```


Language of Parts (Level AA)

`<option>Japanese</option>`

vs.

`<option lang="ja">日本語</option>`

[WebAIM language article](#)

Alternative Text

Alternative Text

- Read by screen readers
- Alternative to images if they are disabled or not supported
- Used by search engines

What is **equivalent** alternative text?

CONTENT and **FUNCTION**

VERY RARELY Description

Alternative Text

- Must
 - Be equivalent
- Should
 - be succinct
- Should not
 - be redundant
 - start with "image of ..." or "graphic of ..."





alt="WebAIM - Web
Accessibility in Mind"

Text alternatives can be presented two ways

- In the **alt attribute** of the img element.
``
- In the **context** or surroundings of the image itself.



```
<a href="minivans.html">  
    <br>  
    Minivans  
</a>
```



```
<a href="minivans.html">  
    <br>  
    Minivans  
</a>
```

Empty vs. Missing alt attribute

- Empty alt (`alt=""`) definitively states:
 - The image is decorative, or
 - Alternative text is nearby
- Missing alt is ambiguous
 - Screen readers may or may not treat it the same as `alt=""`
- Content tools (e.g., WordPress and Word) should have a way to identify decorative images

Images that are the only thing
within a link must **ALWAYS**
have alternative text

...and image map hot-spots and image buttons too.

Image Link



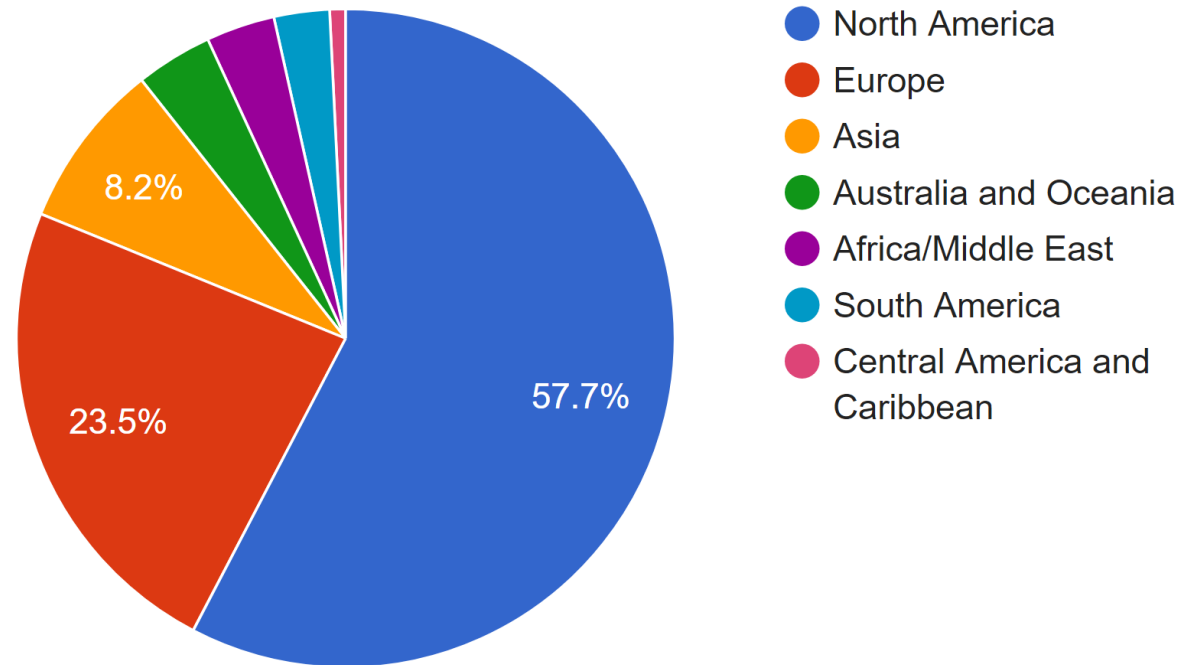
Text Link

```
<a href="minivans.html">  
</a>  
<br>  
<a href="minivans.html">Minivans</a>
```



alt="students studying
under a tree"???

Complex Images



Complex Images

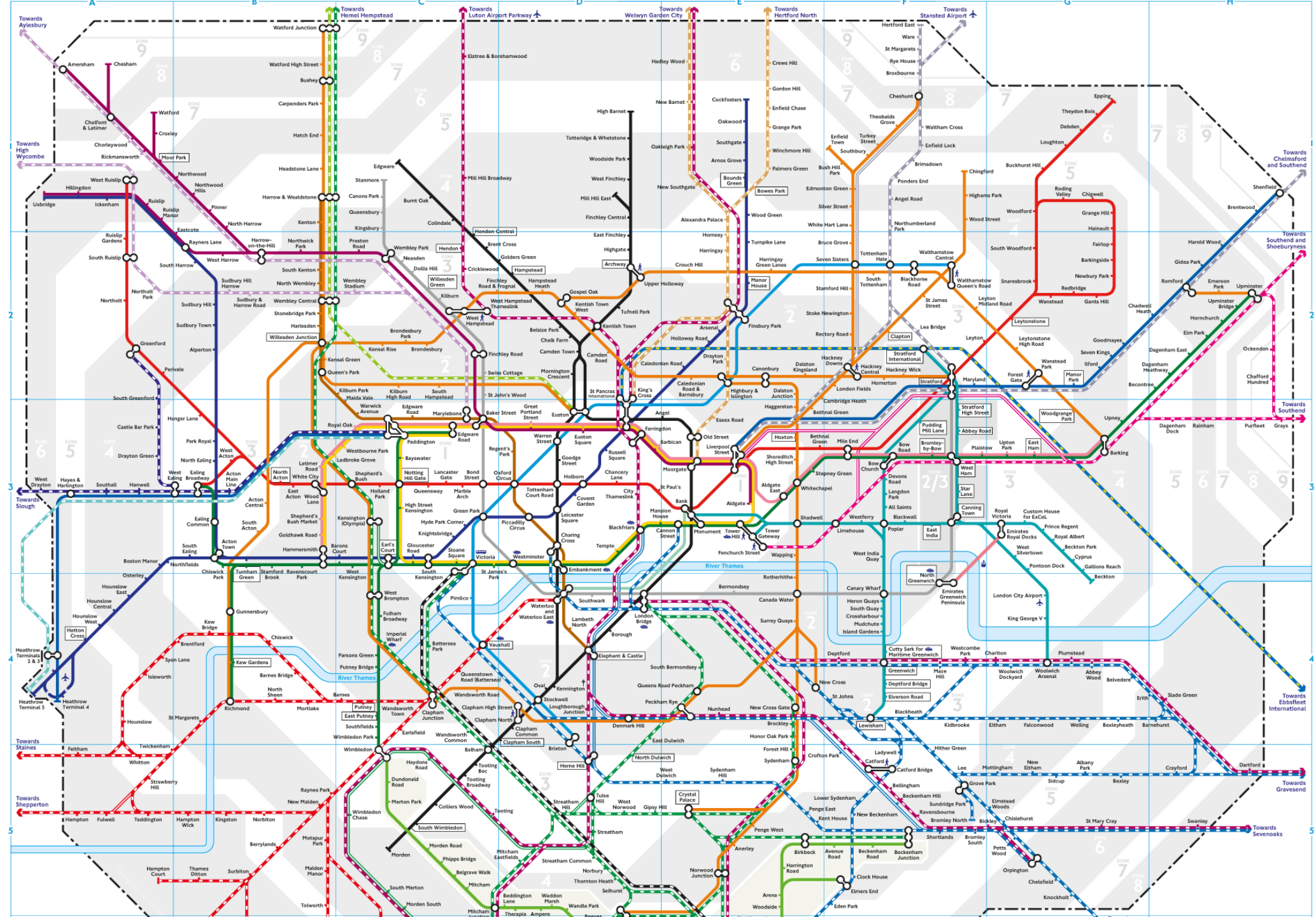
- Give the image succinct alternative text and...
 - Provide the description in context *OR*
 - Provide a link to a page that contains the longer description
- Avoid `longdesc`—support is poor, and it has been dropped from HTML.

London's Rail & Tube services

Key to lines and symbols


- Bakerloo
 - Central
 - Circle
 - District
 - Hammersmith & City
 - Jubilee
 - Metropolitan
 - Northern
 - Piccadilly
 - Victoria
 - Waterloo & City
 - DLR
 - London Overground
 - London Trams
 - TfL Rail
 - Emirates Air Line cable car
- limited service
- Chiltern Railways
 - c2c
 - Gatwick Express
 - Great Northern
 - Great Western Railway
 - Greater Anglia
 - Heathrow Express
 - Southern
 - Southeastern
 - Southeastern high speed
 - South Western Railway
 - Thameslink
 - West Midlands Trains
- peak hours only
- peak hours or limited service
- peak hours only
- peak hours only

- London Trams fare zone
- Stratford
- Station in both fare zones
- Interchange stations
- Street level transfer between stations
- Airport
- Riverboat services
- Victoria Coach Station



[Plan a journey](#)

Plan a journey

  Leaving: now [change time](#) >[Hide preferences](#) <[Plan my journey](#)[My Journeys](#)[Recents](#)Heathrow Airport Terminal 4 to
Piccadilly Circus, London, UK > [Add favourites](#)[Turn off / clear](#)[Public transport](#)[Cycling](#)[Walking](#)

Travel by

[select all](#) | [deselect all](#)

Bus



National Rail



London Overground



River Bus



Emirates Air Line



Tube



DLR



TfL Rail



Tram



Coach



Show me



The fastest routes



Routes with fewest changes



Routes with least walking



Access options

No accessibility requirement



Use escalators, not stairs



Use stairs, not escalators



Step-free to platform only



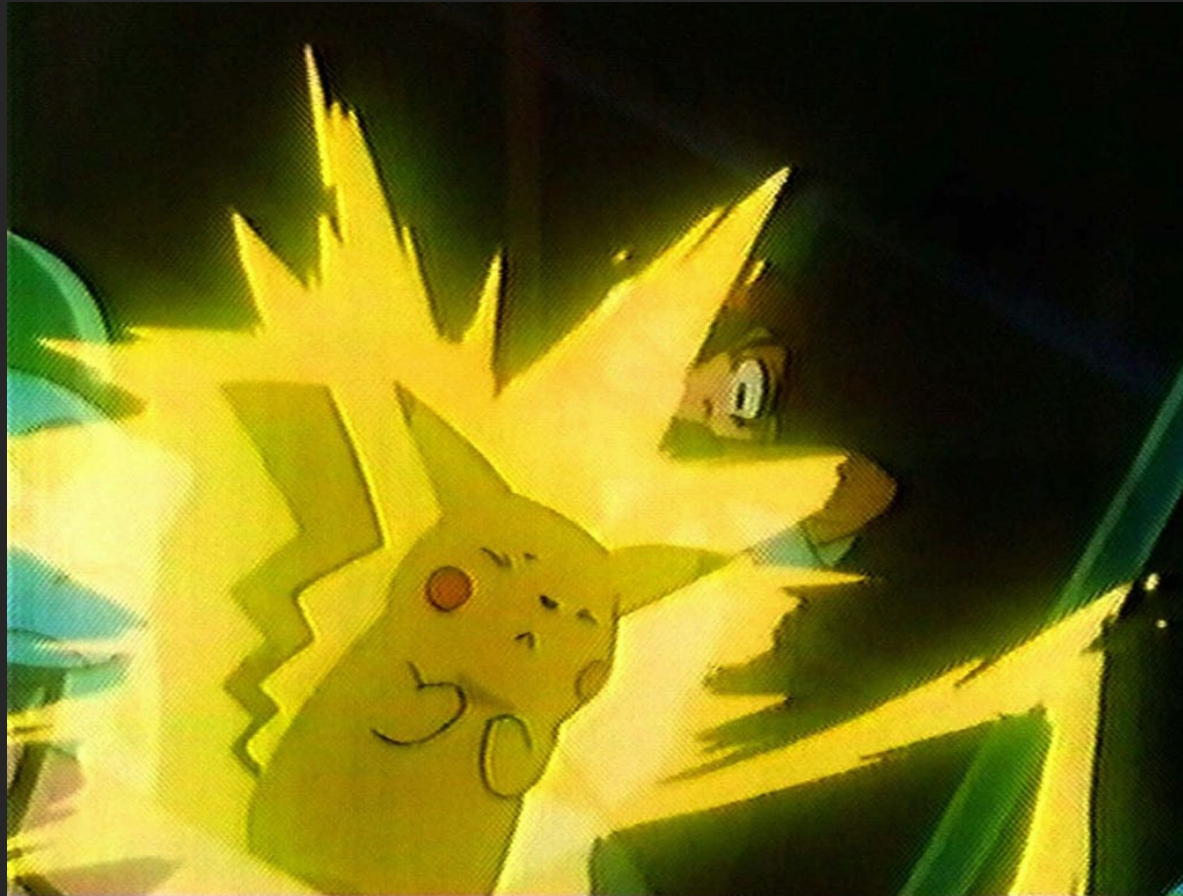
Full step-free access



Evaluate alternative text

- Use WAVE
- Search for images that do not have WAVE icons

Photosensitive Epilepsy



Caution with flashing/strobing content

- 3 times per second or greater
- Size, brightness, and red threshold
- Annoying rule

Cognitive/Learning Disabilities

- Largest disability group. Larger than all the others put together
- Because users' needs vary greatly, we will focus on general recommendations
- Most of the earlier principles can make content more understandable
 - Captions, contrast, color, headings, lists, simple tables, etc.

Make Content “Understandable”

- Be consistent
- Use plain language
 - “readable to users with a lower secondary reading level” (WCAG AAA)
 - Write for your audience
- Be careful with movement and other distractions

There must be a mechanism for the user to
pause, stop, or hide...

- Moving, blinking, or scrolling information that:
 1. starts automatically,
 2. lasts more than five seconds,
 3. *and* is presented in parallel with other content.
- Common failures:
 - Carousels
 - Auto-playing media
 - Animating ads

Legible Text

- Small text negatively impacts readability
 - WCAG has no text size requirements
- Avoid long line lengths
 - Consider line height/spacing
- Choose legible fonts
 - Web fonts and embedded fonts are OK

Typefaces and Fonts

C vs O

C vs O

e vs o

e vs o

l vs l vs l

I vs l vs 1

Each new typeface **INTRODUCES**
additional cognitive overhead.

WCAG 2.1 – 1.4.12 Text Spacing (Level AA)

“No loss of content or functionality occurs” when the *user* increases spacing between:


- Paragraphs: $2 \times$ font size
- Lines: $1.5 \times$ font size
- Words: 16% font size
- Letters: 12% font size

Evaluate text spacing

Add text spacing bookmarklet

Avoid CSS height

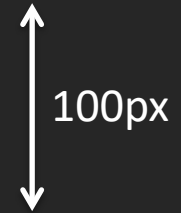
I am some text in a div
that has a pixel height



100px

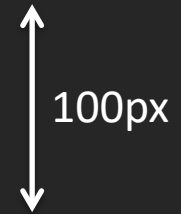
Avoid CSS height

I am some text in a
div that has a pixel
height



Use CSS `min-height`

I am some text in a
div that has a pixel
`min-height`



Motor disabilities

- Content and functionality should be accessible to mouse users and keyboard users
- Lack of fine motor control
- Repetition and fatigue
- Control over timing or moving elements

Standard keystrokes

- Navigate links, form controls, etc.: Tab, Shift + Tab
- Link: Enter
- Button: Enter or Spacebar
- Checkbox: Spacebar
- Navigate options in one “tab stop”: ↑/↓ or ←/→
 - E.g., radio buttons, slider, select menu
- Close: Esc
- Spacebar: Scrolls the page unless focused on a control
- Other widgets: use standard patterns

Example – `<select>` menu

Which is your favorite city?

New York ▾

Amsterdam

Buenos Aires

Delhi

Hong Kong

London

Los Angeles

Moscow

Mumbai

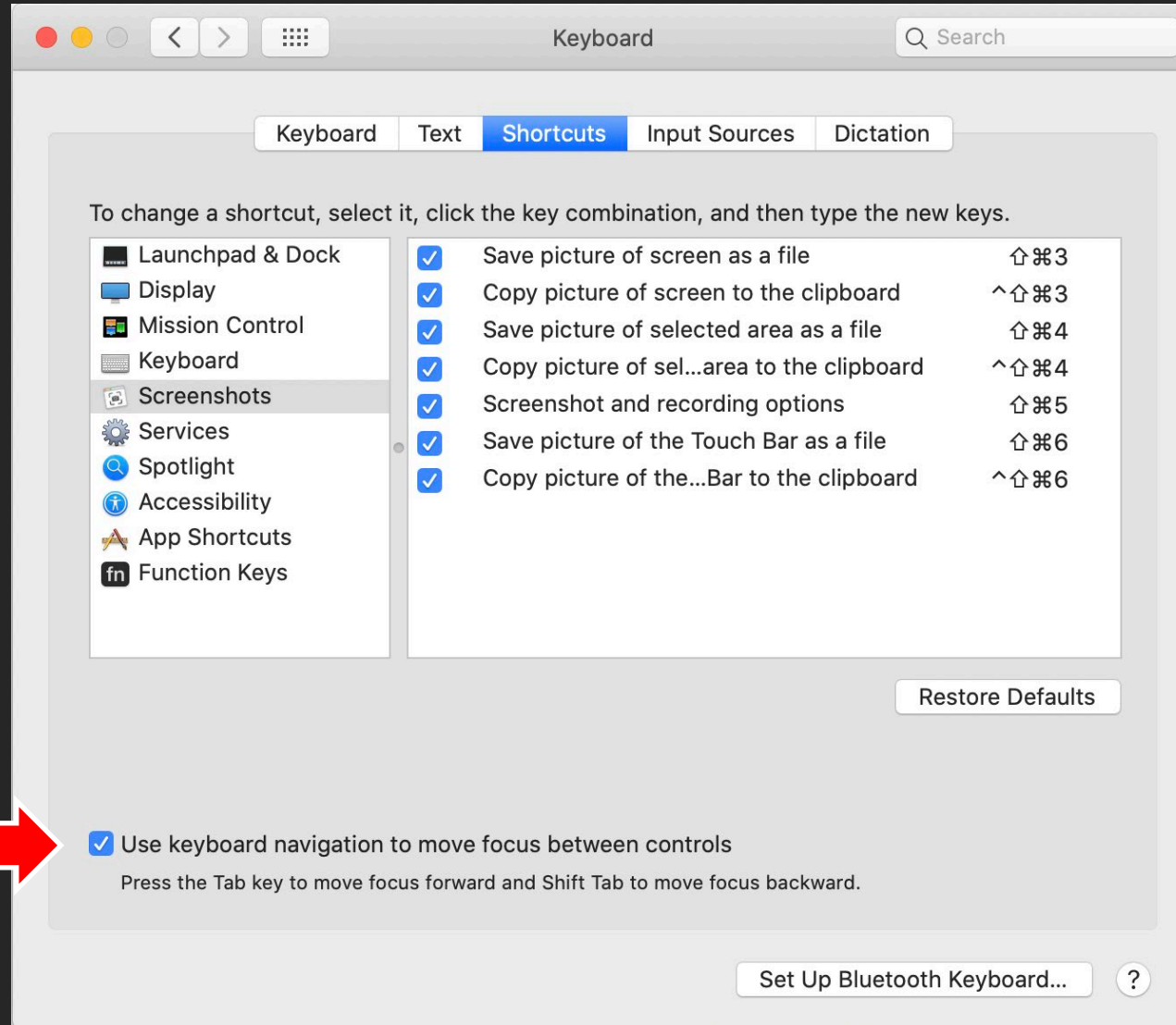
New York

São Paulo

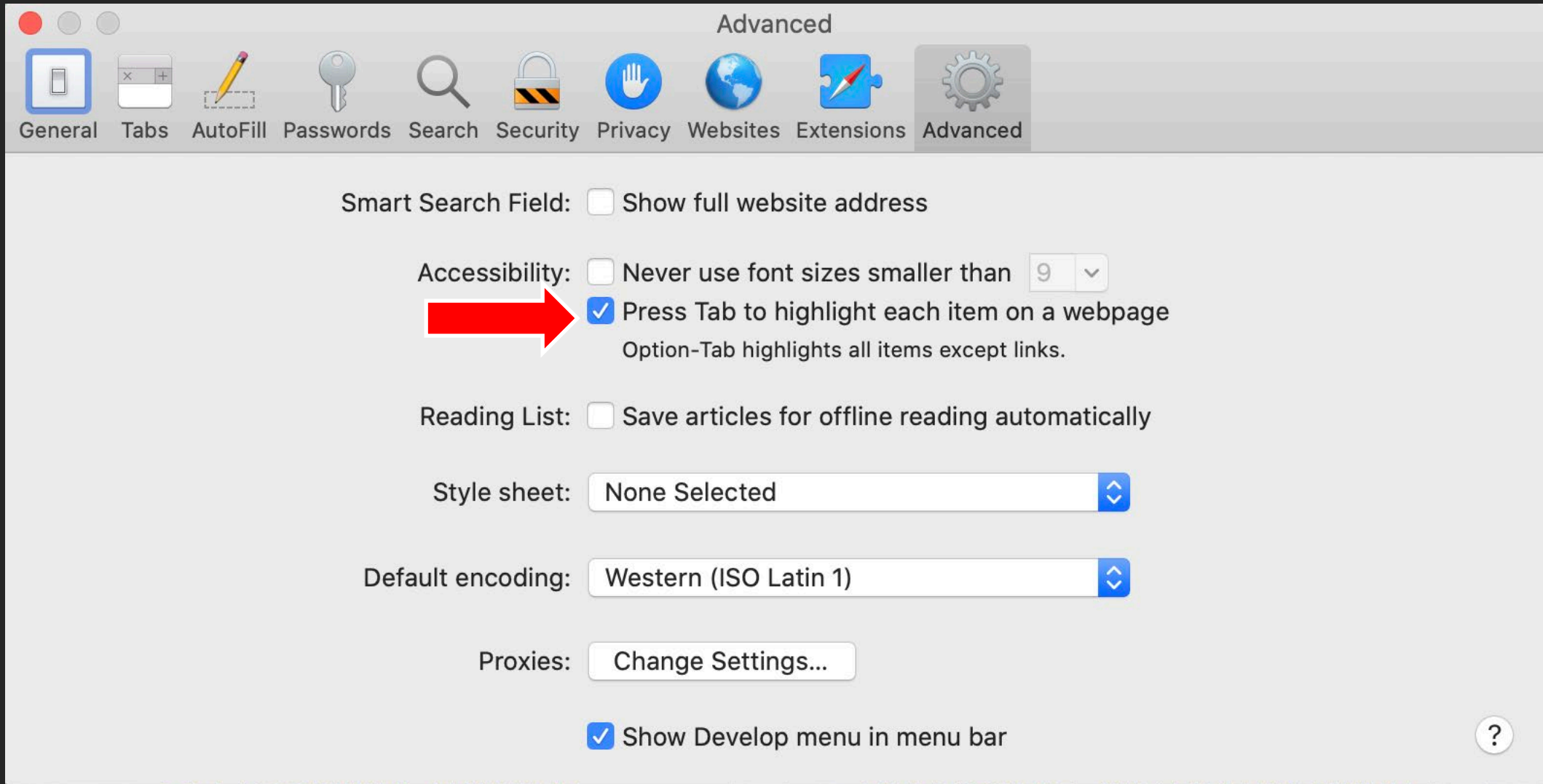
Tokyo

- Tab: Navigate in/out
- Space: Expand
- Arrow keys or Letters: Choose an option
- Enter: Select and collapse
- Esc: Collapse
- Tab: Move to the next element

Some tweaking required on Mac



Safari Keyboard Settings



Disable or remap single key shortcuts

**Keyboard
shortcuts:**

[Learn more](#)



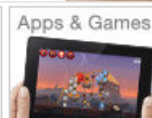
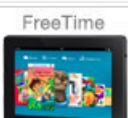
Keyboard shortcuts off



Keyboard shortcuts on

kindle fire HD

Our most affordable Kindle Fire, now in HD



[See 1 customer image](#)

[Share your own customer images](#)

learn more 17 of 17

★★★★★ (9,193 customer reviews)

\$139.00

Prime
Exclusive digital benefits with Amazon Prime

Storage Size: 8 GB

8 GB

16 GB

Offer Type: **With Special Offers**

With Special Offers

Without Special Offers

Special offers and sponsored screensavers display on the Kindle Fire lock screen. [Learn more](#)

In Stock.

Ships from and sold by **Amazon Digital Services**.
Important information about using outside the U.S.

Want it tomorrow, Feb. 19? Order within **7 hrs 50 mins** and choose **One-Day Shipping** at checkout. [Details](#)

We want you to know...

[Learn more](#) about design decisions and feature changes to help make an informed purchase

Our most affordable Kindle Fire—now with a stunning HD display, faster processor, and longer battery life

- Experience movies, TV, and games, and more on a stunning HD display (216 ppi / 1280x800)
- Fast 1.5GHz dual-core processor—apps launch quickly, games and videos play smoothly
- Create profiles and set time limits for children with Kindle FreeTime. Easy-to-use parental controls let everyone enjoy, worry-free
- Ultra-fast web browsing over built-in Wi-Fi, plus updated e-mail and calendar support for Gmail, Outlook, and more
- Instant access to over 100,000 apps and games in the Amazon Appstore, including a new paid app for free every day
- **Prime** In addition to Free Two-Day Shipping, Prime members can stream tens of thousands of Prime Instant Video titles at no additional cost, over half of which can be downloaded to the latest generation of Kindle Fire tablets. [Learn more](#)

Meaningful link text

- [Click here](#)
- [Click here](#) to log in
- [Click here to log in](#)
- [Log in](#)

Do not remove the keyboard focus indicators from links

Avoid

```
a {outline:0;}
```

or

```
a {outline:none;}
```

WCAG AA Failure

Hiding Content

- Hide from everyone:
 - `display:none`
 - `visibility:hidden`
 - HTML `hidden` attribute
- Ensure hidden elements are not navigable
 - Hide them with `display:none` or ensure they become visible when they receive focus
- Avoid 0 pixels, same color as background, etc.

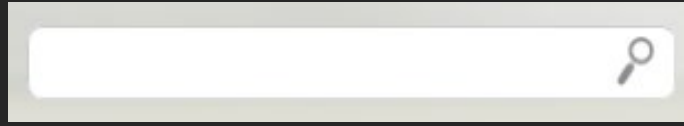
Visually Hiding Content

Position off-screen left with CSS for screen readers

```
.sr-only {  
    position: absolute;  
    left: -10000px;  
    top: auto;  
    width: 1px;  
    height: 1px;  
    overflow: hidden;  
}
```

Can also use CSS clip and/or clip-path

Hidden <label>



```
<label class="sr-only" for="s">  
  Search Terms</label>  
<input type="text" id="s">
```

“Skip” links

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

...

```
<main id="maincontent">...
```

WCAG 2.4.1 (Level A): A mechanism is available to bypass blocks of content that are repeated on multiple Web pages.

Hidden “Skip” links

```
a#skip {  
    position:absolute;  
    left:-10000px;  
    top:auto;  
    width:1px;  
    height:1px;  
    overflow:hidden;  
}
```

```
a#skip:focus {  
    position:static;  
    width:auto;  
    height:auto;  
}
```

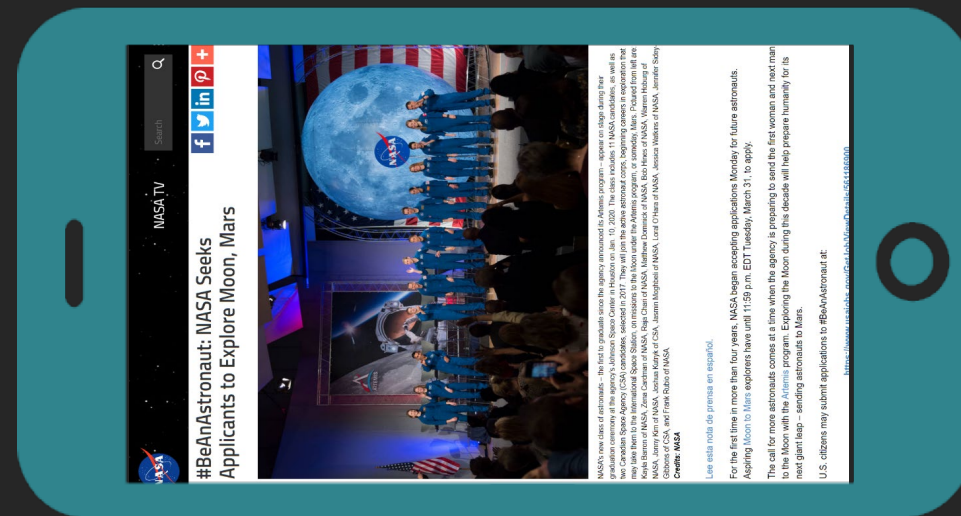
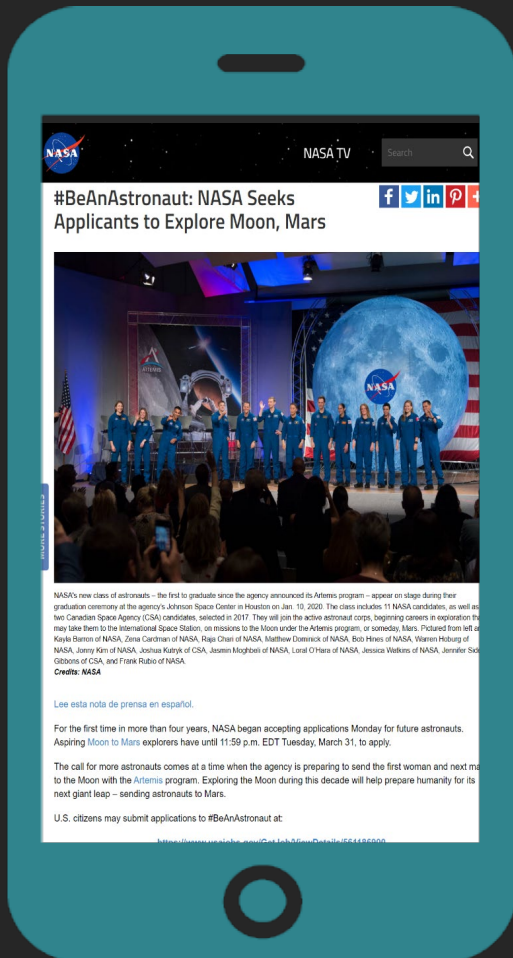
Don't use `display:none` on the link



!= Touch Only

WCAG 2.1 - Mobile

1.3.4 (AA) – Don't lock orientation



New to 2.1 – Guideline 2.5 Input Modalities

- 2.5.1 (A) – Don't rely on multipoint or path-based gestures (e.g., swipe to navigate)
 - WCAG 2.2 proposes “dragging” functionality have single click/touch alternative
- 2.5.2 (A) – Pointer functionality can be canceled (e.g., no down-event to trigger)
- 2.5.3 (A) – “Label in Name”
- 2.5.4 (A) – Don't rely on device motion (e.g., shake, tilt, or pan)
- 2.5.5 (AAA) – $\geq 44 \times 44$ -pixel clickable targets
 - WCAG 2.2 proposes 24×24 px. target or “target offset” (AA)
- 2.5.6 (AAA) – Don't restrict input type (e.g., touch-only)

Evaluate keyboard accessibility

- Navigate the site using only the keyboard (Tab, Shift + Tab, Enter, Space, arrow keys, Esc). Is all functionality available?
- Is navigation order logical?
- Is a visible keyboard focus indicator/outline present?
- How about responsive layouts?

WCAG 2 “Label” and “Name”

WCAG requires a “label” (2.4.6, 3.3.2) and “name” (1.1.1)

- The **Label** is visually presented
- The **Name** is presented to assistive technology
 - Also called “accessible name”
 - May be visually hidden

What is the WCAG “Label”?



What is the WCAG “Name”?



```
<a href="page2.php"...>  
  <img alt="Next"...>  
</a>
```

3 principles of accessible names

1. All interactive elements must have an accessible name.
2. An element can only have one accessible name.
3. To be sure of the name, you must test with a screen reader or inspect the HTML.

“Label” and “Name” are often the same

Code	Label	Name
<code>Log in</code>	<u>Log in</u>	“Log in”
<code><label>First Name: <input type="text"> </label></code>	First Name: <input type="text"/>	“First Name:”
<code><button>Activate </button></code>	Activate	“Activate”

2.5.3 – Label in Name (Level A)

“For user interface components with labels that include text or images of text, the name contains the text presented.”

“User interface components”




UI component



NOT a UI component

“Labels that **include text** or images of text”

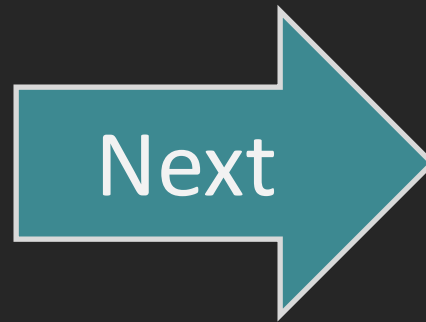
A horizontal search bar with a light gray background. On the left, the text "Search:" is displayed in a dark gray font. To the right of the text is a white rectangular input field with a thin gray border. At the far right end of the input field is a small, faint magnifying glass icon.

Text in label

No text in label

“The name **contains** the text presented”

Label not in Name – Fails 2.5.3



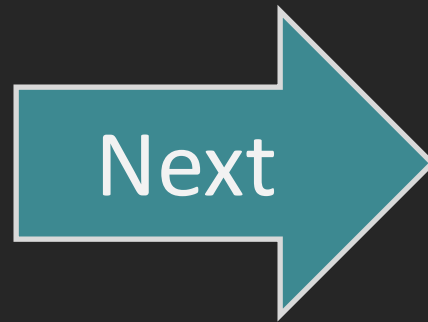
```
<a href="page2.php"...>  
    <img alt="Continue"...>  
</a>
```

Label *in* Name – Passes 2.5.3



```
<a href="page2.php"...>  
    <img alt="Next Page"...>  
</a>
```

Better solution



```
<a href="page2.php"...>  
  <img alt="Next"...>  
</a>
```

Rule #1 of ARIA Use

If you can use HTML, then do so.

Rule #2

Do not change native semantics,
unless you really have to.

```
<div role="navigation">  
  <ul>...</ul>  
</div>
```

instead of

```
<ul role="navigation">
```


ARIA does not change functionality; it only changes the roles/properties presented to screen reader users

Rule #3

All interactive ARIA controls must be usable with the keyboard.

Design Patterns

[ARIA Authoring Practices Guide \(APG\)](#)

Rule #4

Do not use

`role="presentation"` or
`aria-hidden="true"` on visible,
focusable elements.

Rule #5

All interactive elements must have an
accessible name

What if you have more than one candidate for the accessible name?

1. If the control has an `aria-label` or an `aria-labelledby` attribute the `accessible name` is the text content of the element(s) using the algorithm defined in [Accessible Name and Description: Computation](#).
2. Otherwise use the associated `label` element(s) `accessible name`(s) - if concatenate by DOM order, delimited by spaces.
3. If the `accessible name` is still empty, then: use the control's `title` attribute.
4. Otherwise use the control's `placeholder` attribute.
5. If none of the above yield a usable text string there is no `accessible name`.

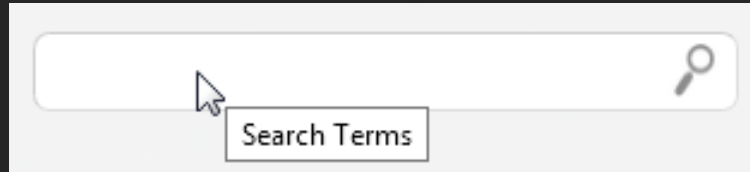
Accessible Name Computation

Form <label>

First Name:

```
<label for="fname">First Name:</label>  
<input type="text" id="fname"...>
```

title attribute (if no <label>)

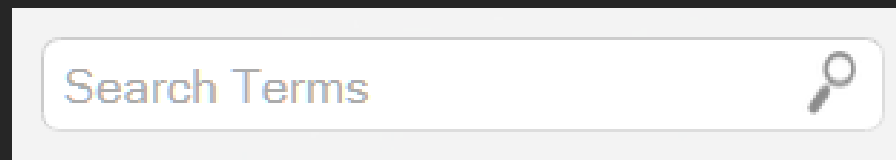


```
<input title="Search Terms"...>
```


title attribute

- Advisory information only
- Should be read for:
 - Form controls missing labels
 - Frames and iframes (iframe title is optional)
 - `<abbr>/<acronym>...usually`
 - Typically best to avoid `<abbr>` and `<acronym>`
- May or may not be read in other places (links, images, etc.)
- Not accessible to keyboard users, touch screen users, etc.

placeholder attribute (if no title or <label>)

A screenshot of a search input field. It is a light gray rectangular box with rounded corners. Inside, there is a white rounded rectangle containing the text 'Search Terms' in a light blue font. To the right of the text is a small, dark gray magnifying glass icon.

```
<input placeholder="Search Terms"...>
```

Placeholder is read by a screen reader (accessible name),
but is not a suitable “label”

name@example.com



password

confirm password

first name

last name

birthday

< Add debit or credit card

First name

Last name



Card number

Expires

CSC

MM/YY

3 digits



Billing address

[Add](#)

< Add debit or credit card

Smith

John



Card number



Expires

1/24




Enter a valid expiration
date

CSC

3 digits



Add

`<label>` 
First Name:

```
<label for="fname">First name:</label>
```

```
<input type="text" id="fname">
```

1:1 relationship between `<label>` and a form control.

First Name:

 `aria-labelledby`

```
<span id="fnamelabel">First name:</span>
```

```
<input type="text"  
aria-labelledby="fnamelabel">
```




```
<label for="fname" id="fnamelabel">  
First name:</label>
```

```
<input type="text" id="fname"  
aria-labelledby="fnamelabel">
```

Unnecessary markup (Rule #1 of ARIA),
but label will only be read once.

One label for multiple controls

`id="name1label"`



Name	Age	Weight
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>

```
<input type="text" name="name1"  
aria-labelledby="name1label">
```

Multiple labels for multiple controls

`id="officenum"`

`id="cyndi"`

Name	Office Number	Phone
Cyndi	<input type="text"/>	<input type="text"/>
Jared	<input type="text"/>	<input type="text"/>
Jonathan	<input type="text"/>	<input type="text"/>

```
<input type="text" name="office1"
aria-labelledby="cyndi officenum">
```

Form values as labels for other inputs

Name	Office Number	Phone
<input type="text" value="Cyndi"/>	<input type="text"/>	<input type="text"/>
<input type="text" value="Jared"/>	<input type="text"/>	<input type="text"/>
<input type="text" value="Jonathan"/>	<input type="text"/>	<input type="text"/>

```
<input type="text" name="office1"  
aria-labelledby="name1 officenum">
```

aria-label



```
<input aria-label="Search Terms"...>
```

ARIA *labels* override default accessible *names*

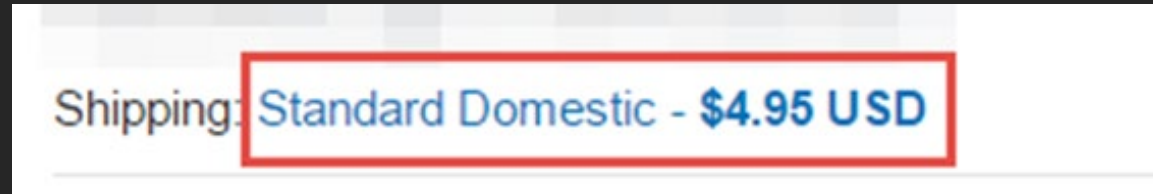
WCAG Label → First Name:

Accessible Name

```
<label for="fname">First Name:</label>  
<input type="text" id="fname"...  
aria-label="Your Name">
```

The diagram illustrates the difference between WCAG labels and accessible names. A green arrow points from 'WCAG Label' to the text 'First Name:' in the UI. Another green arrow points from 'Accessible Name' to the aria-label="Your Name" attribute in the HTML code. The HTML code shows a label for an input field, with the accessible name being overridden by the aria-label attribute.

...and not just in forms



```
<a... aria-label="Change shipping  
method">Standard Domestic - $4.95 USD</a>
```

Also applies to alternative text, button text, etc.

 cannot have an accessible name

[Download the Employment Application](#) 

```
<a href="app.pdf">Download the Employment Application  
<span class="PDFlink" aria-label="PDF"></span>  
</a>
```

Neither can <div>, <p>, <i>, etc.

Three possible solutions

Download the Employment Application

1. `Download the Employment Application(PDF)`
2. `Download the Employment Application`
3. `Download the Employment Application`

CSS generated content



```
<i class="pdf"></i>
```

```
.pdf::after {  
  font-family: FontAwesome;  
  content: "\f1c1";  
}
```

CSS generated content will be read in modern screen readers that understand the generated content.

CSS generated content



```
<i class="pdf" role="img" aria-label="PDF"></i>
```

or

```
<button aria-label="PDF"><i class="pdf"></i></button>
```

Scalable Vector Graphics (SVG)

1. ``
2. `<svg role="img" aria-labelledby="title">
 <title id="title">Sales increased 10%
 from 2010 to 2015</title>
 ...</svg>`
3. `<svg role="img" aria-label="Sales increased 10%
 from 2010 to 2015">
 ...</svg>`

Use `role="presentation"` and `aria-hidden="true"` if the SVG is decorative (or `alt=""` for ``).

aria-describedby

Username:

Must be 8-15 characters

```
<label for="user">Username:</label>
```

```
<input type="text" id="user"
```

```
aria-describedby="usernameetails">
```

```
<p id="usernameetails">Must be 8-15 characters</p>
```

Evaluate accessible name and description with Chrome DevTools

1. Use “Inspect Mode Tooltip” (top-left corner) to inspect the form field.
2. Open accessibility tab
3. Review “Accessibility Tree” and “ARIA Attributes”
4. Using “Computed Properties”, find the accessible name and description

ARIA and HTML labels and descriptions

- Are read as a stream of text. Users cannot easily navigate or explore the referenced content
 - Semantics (lists, links, etc.) are removed from referenced content.
- aria-label is not automatically translated (e.g., Google Translate)

Username:

Must:

- *be at least 14 characters in length*
- *contain at least 4 vowels*
- *contain one medieval rune of alien origin*

Username:

Must meet [these requirements](#)

Hidden ARIA and HTML labels and descriptions

- `<label>` hidden with CSS `display:none` or `hidden` **will not** be read by screen readers.
- Elements hidden with `display:none` or `hidden`, but referenced by `aria-labelledby` or `aria-describedby` **WILL** be read.

```
<label for="user">Username:</label>
```

```
<input type="text" id="user"  
aria-describedby="usernameerror">
```

```
<p id="usernameerror"  
style="display:none">This username is already  
taken. Please try again.</p>
```


aria-required

Username:*

```
<label for="username">Username:</label>*  
<input type="text" id="username"  
aria-required="true">
```

A screen reader indicates the field is required...
and that's all.

HTML required

Username: *

 Please fill out this field.

```
<label for="username">Username:</label>*  
<input type="text" id="username" required>
```

Semantics are the same, but validation messaging is provided

Pay with debit or credit card

We don't share your financial details with the merchant.

United States



Card number

Expires

MM



YY



CSC

3 digits



First name

Last name

Email



Mobile



Phone number

Billing address

Street address

Apt., ste., bldg., etc. (optional)

City

State



ZIP

aria-invalid

Password:

```
<label for="password">Password:</label>  
<input type="password" id="password"  
aria-invalid="true"...>
```

A screen reader indicates the field is invalid or broken...
and that's all.

Use ARIA attributes to control styling:

```
[aria-invalid=true] {  
  border: 2px solid red;  
  background-image: url("error.png");  
}
```

Form Validation and Error Recovery

- Avoid
 - Collecting unnecessary information
 - Forcing users to provide formatting unless necessary
- Identify
 - Let the users know there is an error
 - Direct users to errors
 - Describe the error
- Suggest
 - How to fix it
- Prevent
 - Important changes (e.g., Financial) are reversible/checked/confirmed
- SC 4.1.3 (WCAG 2.1) requires accessible “status messages”

Form Validation Types

- Alert, then focus
- Errors at the top
- Inline errors

Alert, then focus

First Name:

Last Name:

Email Address:

webaim.org says

First Name is required.

OK

Errors at the top

There was a problem with the form.

- [Please enter Your Name.](#)
- [Please enter Your E-mail Address.](#)

Inline Errors

Username

spiderman

Username not available

Password

.....

Must have more than 8 characters
Must have at least one number
Must have upper & lowercase letters

Use links/buttons appropriately

- Links open a new page or jump to another location within the existing page
- Buttons submit form data or perform an in-page function

Ensure Interactive Elements are Links or Form Controls

Tabindex of 1 or greater

- `tabindex="1+"` defines an explicit tab order
- Never use positive tabindex values!
- If the default tab order is not logical, fix your source code order.

TabIndex of 0 or -1

- `tabindex="0"` allows things besides links and form controls to receive keyboard focus.
- `tabindex="-1"` allows things besides links and form controls to receive programmatic focus (by scripting, links, etc.)

```
<div onclick="submitForm()">  
  Search</div>
```

```
<div onclick="submitForm()" "  
  tabindex="0">  
  Search</div>
```

Check for Enter (13) and Space (32) key presses

```
if (event.keyCode==13 || event.keyCode==32)
{
    doStuff();
}
```

```
function allyClick(event) {
    if(event.type === 'click') {
        return true;
    }
    else if(event.type === 'keypress') {
        var code = event.charCode || event.keyCode;
        if(code === 32) {
            event.preventDefault(); // don't scroll the page
        }
        if((code === 32) || (code === 13)) {
            return true;
        }
    }
    else {
        return false;
    }
}

$('#fake-button').on('click keypress', function(event) {
    if(allyClick(event) === true) {
        // do magic javascript stuff
    }
});
```


Just Use A Button

```
<div onclick="submitForm()" "  
  tabindex="0">  
  Search</div>
```

```
<a href="#" onclick="submitForm()">  
  Search</a>
```

```
<a href="#" onclick="submitForm()"
    role="button">
    Search</a>
```

Just Use A Button

```
<button onclick="submitForm()">  
  Search</button>
```

`tabindex="-1"`

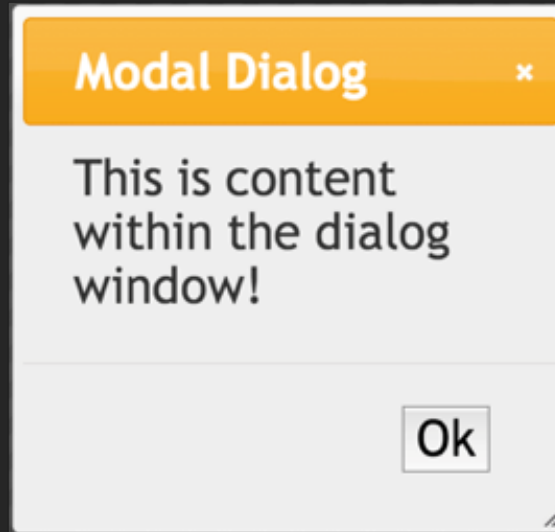
- Allows non-focusable elements to receive programmatic focus (by scripting, links, etc.)
- Necessary for focusing dialog boxes, error messages, etc.
- Also used to manage keyboard focus in some widgets (e.g., a tree menu or tab panel)
- **WARNING:** This removes the element from the default tab order.

`role="dialog"`
w/ `aria-labelledby`

`tabindex="-1"` then set focus
to dialog (or focus a control or
element inside the dialog)

Maintains keyboard
focus if modal

Button with
`aria-haspopup="dialog"`



Returns
focus when
dismissed

Closes with
ESC key

Dialog focus management

- Currently requires scripted focus detection and re-focus, or `tabindex="-1"` and `aria-hidden`

- Future:

```
<div inert>  
  page content  
</div>  
<dialog modal>  
  dialog content  
</dialog>
```


ARIA Roles

- Avoid duplicating default roles:
`<button role="button">`
- ARIA roles override HTML native roles, but do NOT change functionality.
- Be very careful! You can destroy accessibility by adding one attribute.



`<input type="checkbox" role="radio">`

ARIA states and properties

- Elements can have more than one `aria-` attribute.
- ARIA states and properties should not override HTML properties if there's a conflict.
- Be very careful!



```
<input type="checkbox" aria-checked="false">
```

Differentiating regions/landmarks

```
<div role="navigation" aria-label="Main navigation">  
  <nav aria-label="Page navigation">
```

Generic regions/landmarks

```
<div role="region" aria-labelledby="filterheading">  
  <h2 id="filterheading">Filters</h2>  
  ...  
</div>
```

Generic regions must have an accessible name.

More ARIA

```
<button aria-expanded="false">Details</button>  
webaim.org/presentations/2021/examples/disclosure.htm
```

```
<button aria-pressed="true">Toggle Highlights</button>  
webaim.org/presentations/2021/examples/ariapressed.htm
```

```
<a aria-current="page | step | location | date | time | true">
```

Windows screen reader modes

- Reading / Virtual Cursor / Document
- Forms / Application

The current mode determines whether the screen reader or the browser handles most keyboard commands

Some ARIA roles (tree, slider, grid, tabpanel, menu, etc.) trigger forms/application mode.

You must ensure the user is aware and that the proper keyboard interactions are implemented.

Test with AND without a Windows screen reader.

Navigation menus are not application menus!

Navigation tabs (links) are not application tabs!

Data tables are not grids!

etc.

These trigger application mode and thus disable standard keyboard navigation.

Live Regions / Alerts

`aria-live=assertive` - read now

`aria-live=polite` - read at a pause

`aria-live=off` - read when the user encounters it

`aria-busy`

`aria-atomic` - read the entire region (true) or only what has changed (false)

`aria-relevant` - If `aria-atomic=false`, read additions, removals, text, or all

`aria-controls`

Special live regions: `alert` (important), `status` (not important), `timer` (always changing), `marquee` (same as `aria-live="polite"`), and `log` (updates added to the bottom)

Live Regions / Alerts

- Element must be present in the DOM when the page loads, then updated dynamically.
- Some highly dynamic content updates simply cannot be made accessible using ARIA
- Often best to have one messaging element, rather than multiple live regions
- Give users control over content updates

Single Page Applications

- Ensure good document structure
 - Use structural elements (`<main>` or `role="region"`)
- Update page titles to reflect content/state
- Test keyboard navigation
- Ensure only visible elements are navigable
- Set `focus()` when necessary
 - `$('#main').attr("tabIndex", -1).focus();`
- Use live regions for messaging, if necessary

ARIA

With great power comes great responsibility!

Thank You!

webaim.org

- E-mail discussion list
- Monthly newsletter
- Tutorials, articles, and resources
- Blog

