Kris’s WCAG 2.1 AA Guide

By Kris Rivenburgh

Table of Contents

Click a title to jump to any section

Version

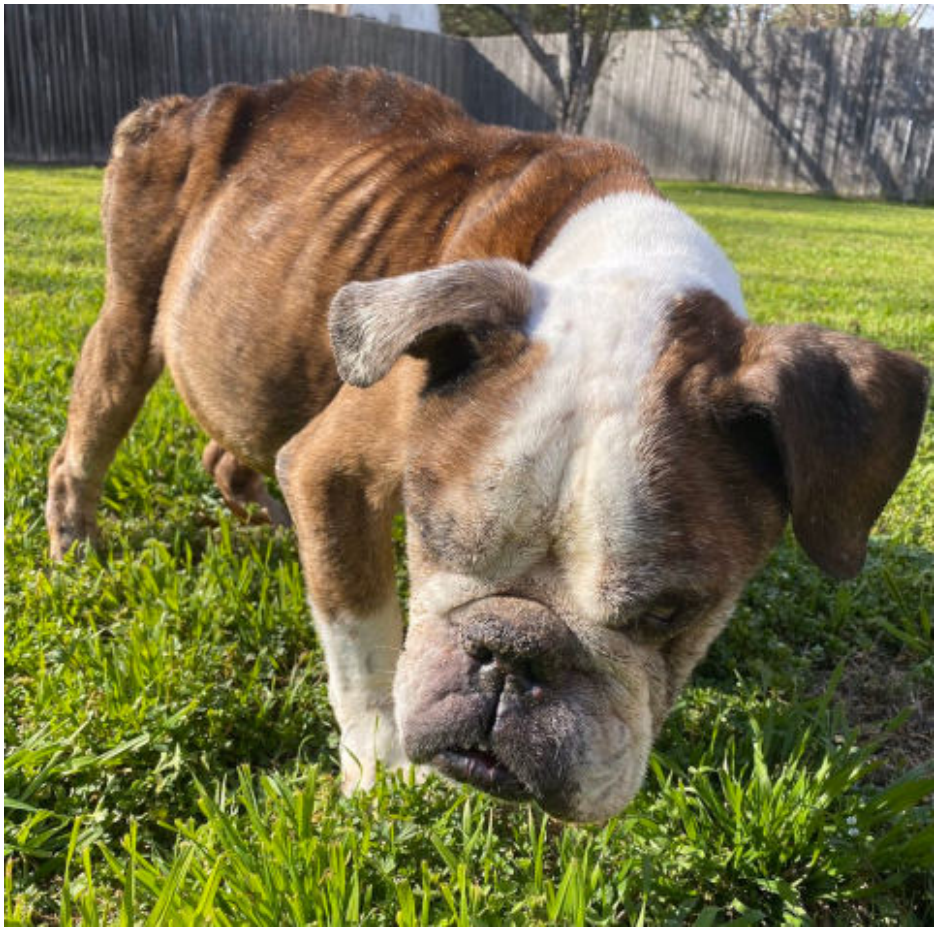
This document is Version 1.0 of Kris's WCAG 2.1 AA Guide and incorporates all of my previous writings.

The official publish date of this document is April 24, 2021.

Later versions of this document will be published (to make corrections, supplement and improve material, etc).

The latest, current version of this document can always be found at https://accessible.org and supersedes any previous versions. If a newer version is available, use of previous versions should be discontinued.

In Memory of Groucho



Groucho had been abandoned by his previous owners and we took him on as a hospice foster, hoping to give him the best life possible in whatever time he had left. Groucho only made it 36 hours but in his brief time he became part of the family and was a great dog.

My favorite memory of Groucho was sitting with him on the back porch while I read and he soaked up the sun on the outside dog bed right beside me.

I love the idea that Groucho will now be known by people all over the world. And I also love the idea that a big part of his legacy will be helping fellow dogs get adopted.

There are so many amazing senior dogs at your local shelter or rescue.

Look out for The Alma and Archer Foundation in 2023. This will be a different kind of dog rescue.

Guide Explainer

This guide includes my WCAG 2.0 AA guide first and my WCAG 2.1 AA guide second.

Although I've separated these out, WCAG 2.1 AA includes all of the success criteria (i.e., requirements) found in WCAG 2.0 AA with 12 new success criteria added in. Thus, conformance with WCAG 2.1 AA will result in conformance with WCAG 2.0 AA.

I'll explain more of the difference between 2.0 and 2.1 in a moment but first, what is WCAG?

WCAG stands for the Web Content Accessibility Guidelines which is a set of technical standards to make digital assets (e.g., websites, mobile apps, etc.) more accessible to persons with disabilities. WCAG is authored by the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C).

As of the date this guide was published, there are three versions of WCAG (1.0, 2.0, 2.1) and three conformance levels (A, AA, AAA). Another version, 2.2, is expected to be released in 2021.

Level AA conformance is what is required for compliance with laws across the world. However, the version for compliance may differ (as of the date of publication, some laws require a minimum of 2.0 and others 2.1).

Version 2.0 is best viewed as the classic version of WCAG. It was published in 2008 and provides a strong baseline for accessibility.

Version 2.1 is the current version as of April 2021. It was published in 2018 and includes key mobile considerations. WCAG 2.1 AA conformance is highly recommended. Although 2.0 AA conformance is a very strong start, 2.1 AA conformance is even better.

There are 38 success criteria in WCAG 2.0 AA.

There are 50 success criteria in WCAG 2.1 AA.

Think of success criteria as requirements or things you need to do to meet the different WCAG versions and conformance levels.

WCAG 2.1 AA includes all of 2.0 and adds in 12 additional success criteria so it consists of 50 success criteria in total.

Again, nothing in WCAG 2.0 AA has been undone, 2.1 simply adds to it.

In this guide, I have separated the 38 WCAG 2.0 AA success criteria and the 12 WCAG 2.1 AA success criteria into different sections.

To help you understand the bullet points of WCAG 2.1 AA, I've done a lot of wire yanking to make things easy and simple. I've:

- Distilled many words into a few
- Reworded technical language into non-technical words (including changing many of the WCAG success criterion titles into more plain English titles)
- Left a lot of details and more remote exceptions out

Think of this guide as a cheatsheet from someone who studied really hard for a WCAG test. It's not going to cover every last detail – and it may not be completely right – but you'll at least get a great idea of the main points.

The How to Do This Part

The formatting of my 2.0 and 2.1 guides is slightly different.

One of my favorite parts about my 2.0 guide is I linked to helpful resources that actually tell you how to implement the different success criteria.

I didn't include a resource section for many of the 2.1 success criteria because, for several of them, it's either you have a development background and understand what you need to do or you don't.

This means you will need a developer to help you implement all or some of the success criteria. However, understanding what 2.1 is calling for is still very important to you as a website owner.

Even though you may not know how to update the code, you can still evaluate your website or mobile app and have a better understanding of whether you meet different success criteria.

Introduction

This guide is great but imperfect and doesn't contain every last detail.

If you question something in the guide or feel you may qualify for an exception or a detail isn't outlined, reference the official WCAG source documentation – this is why I have linked to it at the end of every success criteria.

(If WCAG isn't helpful, try the three resources outlined in the educational material section: WebAIM.org, Rob Dodson's A11ycasts, and Mozilla Developer Network.)

Remember, this guide is only my interpretation of WCAG, it's not WCAG itself.

Even if my interpretation for a given success criterion is perfectly aligned, keep in mind that my interpretation is still only a condensed summary of the source and so there will always be something lost in the distilled version.

Context for WCAG

We all have a disability at some point in our life. Sometimes disabilities are temporary and other times disabilities last for an extended period of time.

Because disabilities can affect how we access the web and other digital items, we want to make sure we make our digital assets as accessible as possible.

When it comes to digital accessibility, the best place to start is WCAG.

In practical terms, WCAG is a bunch of steps we can take to improve the accessibility of our digital assets.

Sometimes we've already unknowingly completed a WCAG to-do and all we have to do is cross that item off checklist.

Other times we need to make an update before we can check off that item. This is all that's happening when we make our assets WCAG conformant.

The great thing about WCAG is it shortcuts the process of making our digital assets accessible. WCAG was published by a bunch of experts who took many different types of disabilities into account and thought of all the different ways to improve accessibility.

When you embrace WCAG conformance, you improve user experience for everyone, including those with disabilities.

If you ignore digital accessibility, you can literally exclude people.

This is not an exaggeration.

If you don't address accessibility, you can literally exclude people from accessing content and interacting with your website, mobile app, etc.

This is why WCAG conformance is so important and so helpful.

WCAG helps us improve our accessibility so people with all types and ranges of disabilities can access our content and engage with our digital offerings.

Here are some illustrations of how WCAG conformance helps.

Illustration 1:

If someone cannot visually see our app's content, we need to make sure they still have access to the content by making it programmatically available. When we do this, assistive technology such as a screen reader can effectively relay our app's content.

Illustration 2:

If someone experiences consistent hand tremors, we need to make sure our website is fully navigable by a keyboard and doesn't rely on a mouse. When we do this, our website remains navigable even if someone cannot effectively use a mouse.

Illustration 3:

If someone cannot hear the audio from our podcast, we need to make sure our podcast content is still available via a text transcript. When we do this, our podcast remains accessible even if someone has difficulty hearing.

While I've only provided three illustrations, you now have a much better idea of just how crucial it is for us to embrace WCAG and bring our digital assets into conformance.

Before I get to the guide, I will cover overlays and additional educational material because they're both important.

Overlays

Overlays are a detriment to accessibility and, as they exist, should never be used as a solution.

Think of overlays as widgets you can embed on your website that open up a menu of accessibility options that make adjustments on your website.

The problem isn't necessarily having these options on your website (although it can be). The problem is when you think an overlay can make your website accessible.

It cannot.

Overlays are marketed and sold as instant accessibility fixes / solutions by several vendors.

Overlays are also marketed as partial fixes to be used in conjunction with manual remediation.

The truth is overlays can't make your website WCAG conformant. At best, overlays amount to tack-on widgets that provide nominal benefit.

In other words, if it makes you feel good to have one and you accept the privacy and security risks that come with an overlay, go for it – but only after you actually make your website accessible / WCAG conformant.

Accessibility expert Karl Groves has put in tremendous time and energy in dispelling overlay myths and I highly recommend you read his websites, OverlayFactSheet.com and OverlaysDontWork.com, to find out more on why you should never rely upon an overlay to make your website accessible.

This dismissal is not a matter of competition within an industry. Rather, this is a matter of nobody wants web accessibility progress to be set back; any reputable entity will unequivocally recommend a competitor that offers legitimate services over an overlay.

Manual work must be done to make a website accessible. There are no instant fixes to website accessibility.

As you read this guide, you will quickly start to realize why it's asinine for any overlay vendor to claim to automatically make your website accessible / WCAG conformant.

Not only do overlays not legitimately make your website accessible or WCAG conformant, overlays are not a stopgap, partial solution, holdover, or anything of the sort.

Overlays are a waste of money, do not make your website accessible, and create separate and horrible experiences for users with disabilities.

Does your organization want to be known for requiring someone with a disability to go through a separate entrance only for that person to have a horrible experience anyway?

Educational Material

When researching and studying accessibility, I found myself spending time at three resources over and over again:

WebAIM.org

Rob Dodson's A11ycasts

Mozilla Developer Network (MDN)

WebAIM.org has the best collection of foundational, beginner friendly accessibility information.

A11ycasts provides excellent video tutorials and explanations of how to implement accessibility.

MDN does a great job of breaking down accessibility from the basics to advanced techniques for developers.

I highly, highly recommend WebAIM, A11ycasts, and MDN.

Two Things I Ask

1. Read the legal section below because it emphasizes that my presentation of WCAG's material will be incomplete and may be inaccurate or wrong and that you release me from all liability.

2. Do not copy, edit, sell, resell, white label, require a subscription for, etc. this guide in any way. It is my copyrighted work and it is to remain free in its original form.

Let's get through the legal stuff and then we'll dive straight into WCAG.

Legal Information

Attribution

I want to make three things very clear:

1) When written, this guide, myself, my company (Kris's WCAG 2.1 AA Guide, Kris Rivenburgh, and Accessible.org) are not affiliated or associated with W3C, WAI, WCAG, or any W3C initiatives in any way. I may participate in W3C activities at some point but this guide is completely independent and separate of the W3C.
2) WCAG is the source of any material web accessibility and technical information that I present in this guide. The W3C and WAI gets all credit for any material ideas, recommendations, guidelines, specifications, etc.
3) All I have done is condensed, rewritten, interpreted, and presented WCAG's material information in an informal, less technical manner. In brief instances, I have included short snippets from WCAG to provide full context and improve understanding and alignment.

Here is full attribution to W3C:

Disclaimers (Important)

By reading this document, you agree to release Author/Creator/Publisher, Kris Rivenburgh and Accessible.org, from any and all liability resulting from your use of this document.

This guide is an interpretation of WCAG 2.1 AA success criteria by one person. This will not fully convey all information found in WCAG 2.1 AA. Moreover, this guide may be inaccurate or misrepresentative of the information contained in WCAG 2.1 AA.

By continuing to read this guide, you understand that it will be incomplete and may be inaccurate. You agree not to hold the Author/Creator/Publisher of this guide (or any of his businesses/companies) liable for any reliance upon this document.

You further agree to base your ultimate web accessibility decision-making on the original WCAG 2.1 AA source document itself and not this guide.

The material in this guide is provided “as is” and without warranties of any kind express or implied. The Author/Creator/Publisher disclaims all warranties, express or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose. The Author/Creator/Publisher does not warrant or make any representations regarding the use or the results of the use of the materials in this guide in terms of their correctness, accuracy, reliability, or otherwise.

Under no circumstances, including but not limited to, negligence, shall the Author/Creator/Publisher be liable for any special or consequential damages that result from the use of, or the inability to use this guide. In no event shall the Author/Creator/Publisher’s total liability to you for all damages, losses, and causes of action exceed the amount paid by you, if any, for this guide. You agree to hold the Author/Creator/Publisher and any connected principals, agents, or entities free from any and all liability for all claims for all damages except for claims of gross negligence and intentional wrongdoing. Note that applicable law may preclude the exclusion of implied warranties or the limitation of damages so these may not apply to you.

You agree that any and all claims for gross negligence or intentional tort shall be settled solely by confidential binding arbitration per the American Arbitration Association’s commercial arbitration rules. All arbitration must occur in the municipality where the Author/Creator/Publisher’s principal place of business is located, Dallas, Texas at the time of publication. Your claim cannot be aggregated with third party claims. Arbitration fees and costs shall be split equally, and you are solely responsible for your own attorney’s fees.

Your use of this guide and engagement of any activities mentioned in this guide means that you are legally bound to these terms. You should always conduct your own due diligence engaging in activity that potentially has large financial risks or expenditures.

No links to external third-parties constitutes an endorsement or recommendation of any kind. Keep in mind that domains expire and pages are taken down so some links may no longer work (or may go to unknown websites) when you access this guide.

If you have any questions or reservations about the legal terms, conditions, and/or copyright of this guide, contact me at kris@accessible.org.

As a condition to use of this document, you must agree to all terms herein prior to use of this document. By continuing, you agree to all legal terms, disclaimers, copyright, and other conditions.

Copyright

For information, please contact the author by email at kris@accessible.org or by mail at PO Box 791691, San Antonio, Texas 78279.

Accessibility

PDF/UA and WCAG 2.0 AA standards were followed in the creation of this document.

If you have any difficulty accessing the contents of this document, contact me at kris@accessible.org and I will ensure that you have full access to this guide.

WCAG 2.0 AA

1.1.1 Text Alternatives for Non-text Content

All non-text content (images, buttons, and content that isn't relayed by text) needs to have a text alternative that conveys the same information found in the content.

Purely decorative, non-meaningful images (e.g., dividing line between nav bar and content) does not need alt text but does need an empty alt attribute.

What to do:

- add alt text and/or descriptions to meaningful images
- leave empty alt attributes for non-meaningful images
- programmatically label form fields and buttons

Resources:

- How to add alt text (YouTube – 3 min 26 secs)
- Alt text for complex images

Plain English Explanation:

This success criteria comes down to making sure all non-text content on your website has a text equivalent.

Alt text for images is fairly easy to understand and implement but writing accurate, concise alt text can be more difficult than you might think.

Only images that convey meaning or information should have alt text values. Purely decorative images should have alt attributes that are left empty.

For complicated images that convey large amounts of information (e.g., infographics), alt text should reference a full text description rather than attempt to all information.

Additionally, elements such as form fields and buttons need to have text equivalents that label those elements.

Understanding 1.1.1

1.2.1 Alternatives for Audio-Only and Video-Only Content

For all prerecorded video-only files (no dialogue) and/or audio-only files (no video), make a text transcript available. You may also record a synchronized audio track (or audio description) of what is happening inside the video as an alternative to the text transcript.

What to do:

- add a text transcript link or full text transcript near the audio or video file

An audio description will convey all of the important visual content found in the video. As we are only dealing with video-only files in this success criteria, we don't have to work in the description during pauses.

Resources:

- What is an audio description (YouTube – 2 min 12 secs)
- How to add an audio description to a video (YouTube – 8 min 0 seconds)
- How to create text transcripts for free (YouTube – 1 min 29 seconds)
- How to format a text transcript
- Best practices for text transcripts
- How to embed a text transcript using HTML (complicated – not for beginners)

Plain English Explanation:

Here we're focused in media files that are only audio or only video (as opposed to video with audio).

A prime example of an audio-only file is a podcast. For a podcast, you will provide a full text transcript.

For a video-only file in which there is no dialogue, you want to provide either a full text transcript or the option to hear an audio description of what is taking place in the video.

A text transcript can convey the speakers' names and roles and what they say. A full text description will convey other relevant information contained within the media.

For example, a door slamming is not part of the spoken dialogue, but, if it's for effect, it may be relevant for the content.

You can always copy and paste a full text transcript directly below a video but this is not a viable option for many websites. Another option is to create a descriptive link below the video (e.g., text transcript for video) and include the full text transcript on another page.

Understanding 1.2.1

1.2.2 Closed Captions for Video with Audio

For all prerecorded (non-live) video files, there must be synchronized and accurate closed captions.

What to do:

- Add closed captioning to all videos with audio

Note that closed captions are more than subtitles. Subtitles merely convey dialogue. Closed captions also include any important sounds in the video (e.g., door slamming, background music, etc.)

Here's a great distinction between the two from Matinee.co.uk: Subtitles assume the audience can hear the audio but needs the dialogue available as well (for instance, the dialogue may be in another language or difficult to understand). Closed captioning assumes the audience cannot hear the audio and needs a text description of what they would be hearing.

Resources:

- How to use YouTube's automatic subtitles to make closed captions

Plain English Explanation:

To satisfy this requirement, your videos need to have closed captions.

Closed captions require some amount of manual work but there are a handful of free or inexpensive tools out there to make the process free or relatively cheap.

What is really important is that your closed captions sync up with the video and that they accurately convey the spoken words.

Automatic subtitles – even YouTube's subtitles – aren't perfect so you do need to edit them in addition to adding in any background information for closed captions.

Understanding 1.2.2

1.2.3 Audio Description or Full Text Description of Video

For all prerecorded video files, there must be either an audio description or a full text description.

An audio description provides important information (e.g., actions, characters, scene changes) during pauses in the dialogue of a video.

A full text description provides all visual information contained in the video. Think of it as a text transcript plus all of the important context (e.g., scenery, expressions, laughter, etc.).

What to do:

- include an audio description with any video with meaningful information conveyed in the background

Note that if your video's sound conveys all relevant information, no audio description is needed.

Plain English Explanation:

If you're noticing some overlap between the success criteria, that's because there is.

You'll need to create an audio description version of your video because an audio description ends up being required by 1.2.5.

Understanding 1.2.3

1.2.4 Live Captions

For any live video broadcasts, real-time closed captions that identify speakers and include relevant sounds are provided.

What to do:

- if you host a live video broadcast, you must have real-time closed captions included

This success criterion specifically says it is not intended to require two-way calls through web apps to be captioned.

This success criterion seems to apply to more formal broadcasts and in, in my research, I did not find much elaboration on this requirement.

There are numerous companies that offer real-time closed captioning.

Plain English Explanation:

If you’re presenting a formal, live broadcast, you need to have the ability to show real-time closed captioning.

A good example of a formal broadcast is a webinar.

Understanding 1.2.4

1.2.5 Audio Description for Video

All prerecorded (non-live) video must have an audio description.

What to do:

- provide audio description in your video

Plain English Explanation:

In success criterion 1.2.3, an audio description was an option. 1.2.5 requires it.

Remember, an audio description is only required if meaningful information is conveyed visually but not audibly.

For example, if your video only consists of someone speaking into the camera, then no audio description is required because no important information is presented visually that is not presented in audio.

Understanding 1.2.5

1.3.1 Structure Semantically with HTML

Structure your website so that content is read by a screen reader in the same way it is presented visually to sighted users.

What to do:

- Use <h1> to <h6> headings to structure and organize content
- Use HTML5 semantic markup to structure content
- Make sure data cells are associated with header cells inside a table
- Programmatically label form fields
- Use HTML markup for ordered and unordered lists

Resources:

- How to use headings correctly
- How HTML5 semantics work
- How to make tables accessible
- How to label forms for beginners
- How to create lists correctly
- Introduction to ARIA (YouTube – 9 min 5 seconds)

Plain English Explanation:

If you're not a web developer, you probably don't understand what 1.3.1 is asking for.

Semantics, in this context, means you are giving things on your website meaning through code.

Here's a good illustration:

Let's say we want to make a subtopic to organize and break up content within an article. We go between two paragraphs and write a subtopic and then increase the font to size 28 and bold it like this:

Subtopic Goes Here

A sighted user will be able to instantly perceive that this text is a subtopic heading from the visual cues of increased font size and the bold.

A nonsighted user will have this text read by a screen reader without any of the visual cues; it will simply read as regular text.

With semantics, we code meaning into our website so that screen readers can convey what is visually presented.

In the example above, rather than simply making the font bigger and bolder, we'd probably want to wrap an <h2> or <h3> tag around the text to indicate a heading and the beginning of a new section. We can visually make it have the same appearance with CSS coding while retaining the semantic property.

That's really at the heart of what this success criterion is getting to; we want to add meaning to the different elements and components on our website so that meaning can be conveyed to screen reader users who are listening to content rather than seeing it.

We add this meaning through coding.

This is why it's crucial to use HTML5 whenever possible when coding our website. Sometimes HTML5 may not have an option available so we'll need to draw from what is called WAI-ARIA or, more commonly, ARIA.

ARIA standards for Accessible Rich Internet Applications and ARIA labels can add meaning for more dynamic, custom, or advanced applications where HTML5 is insufficient.

For our websites to meet this success criteria – which is an extremely important one – we need to markup and structure the contents and elements on our website properly.

This is certainly not beginner level stuff but, if you've ever examined HTML code before, you will be able to quickly learn how to use correct HTML for headings and lists. Even the landmark elements from HTML5 are fairly easy to understand.

Where you'll likely need the most help is with coding large tables and interactive elements such as forms. There are several resources above to help you gain a better understanding.

Understanding 1.3.1

1.3.2 Correct Reading Order Sequence

Structure your website so that your content is presented in a logical order.

What to do:

- Structure your website's content so that it reads in a logical order
- Test your website with a screen reader (or without CSS enabled) to make sure its reading order is presented correctly

Resources:

- Chrome/Firefox extension for disabling CSS

Plain English Explanation:

We need to make sure our website presents content in a sensical order to screen readers.

Where you might run into trouble is if your website has a unique layout (e.g., main content in center with a sidebar on the left and right) or multiple columns of content.

The best way to present content is in the order that most people would naturally read it – from top to bottom and left to right.

Basic reading order:

1. Header
2. Main Content
3. Sidebar
4. Footer

The browser extension in the resources section will enable you to view your website without CSS enabled.

CSS is a language that styles the appearance/design of websites.

The effect is you can compare your website as it shows normally (with styling) and without so you can see if there are any differences in the visual presentation vs. the order in which a screen reader will read the page.

There can be more than one correct reading order. The key is that your website programmatically reads in a logical, sensical order.

Understanding 1.3.2

1.3.3 Instructions Involve More Than One Sense

Write clear instructions that incorporate multiple senses. No instructions should rely solely on the ability to perceive shape, size, visual location, orientation, or sound.

What to do:

- Be deliberate and purposeful when writing instructions on your website
- Attempt to include a text reference (e.g., click on the green button labeled “subscribe now”) with every set of instructions
- Remember that some people will not:
    - Be able to tell what shape an object is
    - Be able to perceive where an object is positioned
    - Be able to hear an audible indicator

Resources:

- A different but good explanation of 1.3.3

Plain English Explanation:

The main point here is to incorporate multiple ways to perceive instructions. A best practice is to label things with text and include a reference to that text.

Here are some good and bad examples of sensory instructions.

Bad examples:

You can search for additional information using the search bar at the bottom.

Continue to the next page as soon as you hear a chime.

Good example:

Click on the green button labeled “next page” directly below this section.

Note that not all instructions will take the form of direct statements. WCAG uses the example of a data table and providing both a specific background color and icon shape to provide information.

Understanding 1.3.3

1.4.1 Color Not Only Way of Conveying Information

Color should not be used as the only visual way to convey information or make an indication.

What to do:

- Identify any ways your website conveys information by color and make sure there is another way to extract the same information

Plain English Explanation:

This success criterion attempts to eliminate any circumstances where color alone is used to convey information, make an indication or prompt the user, or distinguish between visual elements.

Reliance on color alone is very common.

Examples:

1. Toggle switches where green indicates activated or on and gray indicates off
2. Required fields are in red
3. Color coded graphs and charts
4. Blue links without additional marker (e.g., underline) embedded in a paragraph of black text

You do not have to eliminate color for information or as an indicator but you do have to account for the fact that some people may not be able to perceive the color and add an alternate means (such as text) where people can gather the same information or indication.

Understanding 1.4.1

1.4.2 Audio Control

Users must have a way to pause or stop any audio that automatically plays for more than three seconds.

What to do:

- Don’t have any audio that automatically plays on your website.
- If you do have audio that automatically plays, limit it to three seconds and provide for an easy way to adjust, mute, pause, or stop the video.

Plain English Explanation:

One of the fundamental principles of accessibility is to put the user in control of the page’s content: website owners provide the content and functionality and users choose what they want to access and engage with.

This success criteria drives that point home and is fairly simple to meet.

WCAG discourages the practice of having any automatic audio but provides a quick three second exception.

I highly recommend not having any automatic media.

Understanding 1.4.2

1.4.3 Color Contrast

Text and images of text should have a color contrast ratio of 4.5:1 against the background.

Exceptions:

- Large text (18 point or 14 point bold) only has a minimum 3:1 ratio
- Logos or brand names don't have a requirement

What to do:

- Make sure all text on your website meets a 4.5:1 color contrast ratio (or 3:1 ratio if its larger text)

Resources:

- WebAIM color contrast checker
- Colorzilla color selector Chrome and Firefox extension
- Gimp photo editing software
- How to use WebAIM color contrast checker (YouTube – 1 min 12 secs)

Plain English Explanation:

This is one of the easier success criteria to check for and fix.

The point is to make sure our text sufficiently stands out from the background. The closer text color is to background color, the harder the text is to read.

Here's an example:

It's much, much easier to read the white text on a black background vs. the dark gray text which shows why color contrast ratio is so important.

To figure out what colors your website is using, use a color selector tool to grab a color off your website. You will be able to get its hexadecimal code (e.g., #efefef). You can then enter this into the WebAIM checker and instantly know what your color contrast ratio is.

One way to do this is to install the Colorzilla browser extension and use its color selector.

Another way is to take a screenshot of a webpage and then paste it into a photo editing software like Gimp (free) and then use the color selector from there to find out the hexadecimal code.

If this all sounds confusing, the video in the resources section will help you out.

I highly recommend you meet a 4.5:1 ratio with all text regardless of what size it is.

Understanding 1.4.3

1.4.4 Text is Resizable to 200%

All text on your website can be resized up to 200% without any loss of content or functionality on your website.

What to do:

- Use min height in CSS so that containers increase when text size increases

Resources:

- Text resizing explained (YouTube – 2 min 14 secs)
- Text resize best practices

Plain English Explanation:

You should be able to increase text size on your page by up to 200% without the page display messing up (i.e., having text overlap over other text or elements).

Here's an example where I increased the text size beyond 200% on ADA.gov:

In the top right-hand corner, the "Search ADA.gov" search bar is hard to see and you can't read the text that says "More Search Options" and you can't even see the "go" button anymore.

Beyond accounting for minimum height with containers in CSS (remember, CSS is code that styles websites), I recommend creating a starting default size of at least 14 point so that users don’t have to zoom or resize text as much as they normally would.

The key here is that your text can be resized without causing any negative experience for someone using your website.

Understanding 1.4.4

1.4.5 Avoid Images of Text

Do not use images of text on your website except where absolutely necessary or for branding purposes (e.g., logo).

What to do:

- Make sure you only have absolutely necessary images of text.
- Avoid images that include text that conveys information if at all possible
- Graphs, screenshots, and diagrams which convey important information of more than just text are probably okay as images
- If you do have an image that includes text, add alt text and potentially a caption or description

Plain English Explanation:

Avoid images of text as much as possible. Logos are fine.

If you do have an image that has text inside it, account for that information with alternative text or a description of the image.

If the same presentation can be made using only text (and not an image), never use an image.

Images of text are acceptable when it would be impossible to replicate the affect otherwise.

Understanding 1.4.5

2.1.1 Keyboard Only

Your website must be fully accessible with only a keyboard.

What to do:

- Unplug your mouse and make sure you can do everything on your website that you would be able to do with the mouse plugged in
- Check that any shortcut keys used are not the same available through browsers or screen readers.

Plain English Explanation:

Everything on your website must be accessible without a mouse.

Elements to double check:

- Forms
- Submission buttons
- Media players
- Navigation menus and submenus
- Sidebar content
- Embedded content

Certain programs that require a keyboard (e.g., painting program) may fall under an exception.

Avoid keyboard shortcuts unless you make sure they don’t conflict with browsers or screen readers.

Understanding 2.1.1

2.1.2 No Keyboard Trap

Keyboard-only users must be able to navigate to and from all parts of a website. No mouse should be necessary to move away from any element of a website.

What to do:

- Unplug your mouse and make sure you can get navigate both forwards and backwards from every element and section of a website

Resources:

- Keyboard accessibility overview
- Example of keyboard trap (YouTube – 1 min 4 secs)
- How to use the tabindex attribute to improve usability for keyboard-only users

Plain English Explanation:

This success criterion is an extension of the previous one on keyboard accessibility. Here, we're not only concerned with full access but the ability to withdraw from all elements.

With no keyboard trap, it's especially important to test any pop-ups, forms, embeds, or widgets on your website to make sure that users can back out of them with ease.

Understanding 2.1.2

2.2.1 Adjustable Time Limit

For any time limit on a website, a user must have one of the following options:

- Ability to turn off time limit
- Ability to adjust time limit to at least 10x the default setting
- After a warning of time limit expiration, given at least 20 seconds to extend time limit at least 10x

Exceptions:

- The time limit is part of a real-time event (e.g., auction) and no alternative is possible
- The time limit is essential (e.g., buying tickets online) and extending it would invalidate the activity
- The time limit is longer than 20 hours

What to do:

- Avoid time limits wherever possible
- If a time limit is necessary, provide the ability to adjust or extend it unless you meet an exception above

Plain English Explanation:

This success criterion is self-explanatory. If you have a time limit on your website, definitely reference the WCAG link below.

Understanding 2.2.1

2.2.2 Pause, Stop, Hide Moving Content

Any moving, blinking, scrolling, or auto-updating content that starts automatically and lasts more than five seconds must provide the ability for a user to pause, stop, or hide it.

Auto-updating means content updates or disappears on a preset schedule.

Exceptions:

- Unless the content is essential

Examples:

- Stock ticker
- Sports scores
- Blinking advertisement
- Scrolling news updates
- Gifs

What to do:

- Limit or eliminate any content that moves or updates automatically

Plain English Explanation:

For any content that begins to move on its own, provide the ability to pause, stop, or hide it after five seconds.

Moving content can be extremely distracting so this success criterion severely limits it. WCAG allows for five seconds for automatic moving content to catch someone’s attention.

Again, you should be picking up on a theme: the user remains in control of the website's content. Keep this in mind when you remediate or build your digital asset for accessibility.

Understanding 2.2.2

2.3.1 Limited Flashing Content

Nothing on a webpage flashes more than three times in a second.

The terms “flashing” and “blinking” can mean the same thing. WCAG distinguishes flashing as content that can cause a seizure. Blinking refers to content that can be a distraction. The two can be the same if blinking occurs faster than three times per second.

What to do:

- Limit or eliminate any flashing content

Plain English Explanation:

2.3.1 is written to prevent seizures.

The point here is to severely restrict flashing content.

Although there are exceptions that provide for automatic content, I recommend making your website as static as possible by default, with options to activate helpful automatic content.

Understanding 2.3.1

2.4.1 Skip Navigation

Provide a link in the top left of your that allows users to skip the header of your website and go straight to the main content.

You may also see this referred to as a “skip to content” link or generally as a “skip link”.

Skip navigation links can be visible or hidden in the presentation of a website.

What to do:

- Add a skip navigation link to all pages of your website

Resources:

- Why a skip navigation link is helpful (YouTube – 0 min 50 secs)
- How to create skip links and hide them (YouTube – 4 min 58 secs)

Plain English Explanation:

Provide the ability to skip past your header (with menu, navigation, ads, etc.) and go directly to the main content of your website.

Understanding 2.4.1

2.4.2 Descriptive Page Titles

Each webpage has a unique and descriptive page title that lets the user know what the topic or purpose of the page is.

What to do:

- Make sure all pages of your website have a page title

Resources:

- How to write helpful page titles (YouTube – 2 min 10 secs)

Plain English Explanation:

It's not enough to have a title tag exist on each page of your website, the titles need to be descriptive enough for a user to identify what the topic or purpose of a page is.

Also, no two pages of your website should have the exact same title tag – a title tag may be close to identical to another page but there should always be some distinguishing component.

For example, two bank statement pages may be the exact same except one has "Oct" and the other has "Sep" in the title tag.

Understanding 2.4.2

2.4.3 Focus Order

When users tab through your website, the content appears in a logical order.

What to do:

- Ensure the links, form fields, and any other areas of engagement that receive focus on your website are tabbed through in a sensical order

Resources:

- Great guide to focus order

Plain English Explanation:

This success criterion requires that you're able to tab through a website in a logical, sensical manner.

For example, it wouldn't make sense to tab from a menu link in the header and immediately go to a search box in the footer, bypassing a registration form in the middle of the page.

What we're doing here is making sure the tab order goes in a logical sequence that flows in a sensical manner.

If you meet all of the success criteria thus far, you're probably in good shape but you'll definitely want to triple check and make sure your focus order is sensical.

Understanding 2.4.3

2.4.4 Descriptive Links

Write your link text so that users have an idea what the linked page is.

What to do:

- Use obvious link anchor text (anchor text are the words you link under)
- Avoid long URL and “click here” and “learn more” general type of links
- Try to use mostly keywords (vs. fluffy words like “click here”) when creating links
- Be especially cognizant of creating useful alt text when linking images

Resources:

- Descriptive vs. Non-descriptive links (YouTube – 3 min 46 secs)

Plain English Explanation:

Write your links so that they’re descriptive and easy to read. Also, try to keep your anchors concise.

This is actually one of the more common accessibility failures of websites, but the good news is it’s also one of the easiest to fix.

I used to link under “click here” quite frequently until I learned the importance of descriptive anchor text.

The video in the resources shows how a long URL link can be frustrating.

Understanding 2.4.4

2.4.5 Multiple Ways of Finding Pages

Make it so that there is more than one way to find a web page within your website.

What to do:

- Add a global search
- Add a sitemap
- Add a related pages section below content

These bullets aren't explicitly required but I've included them as examples to funnel you in the direction this success criterion is getting after.

Exception:

- If a page is part of a process (such as a review cart page before you checkout), you don't have to provide multiple ways to find it

Plain English Explanation:

Make it as easy as possible to find pages on your website. If you add a search bar and sitemap, you've very likely satisfied this success criterion.

Another idea is to incorporate what are referred to as breadcrumbs. Breadcrumbs are links that show you the path of where a particular page is within a website structure so you can easily know where you are.

Think back to the story of Hansel and Gretel and how Hansel dropped pieces of bread so they could find their way home if they got lost. Website breadcrumbs help you find your way within a website.

Here's an example from Section508.gov:

This trail of links makes finding your way within Section508.gov easy no matter what page you're on.

Breadcrumbs are not essential but they can be extremely helpful.

You can go another step further and add a related resources section below content on your website (e.g., links below a blog post to similar posts or related content).

Again, you're just trying to make content easier to find by making your website more navigable.

Understanding 2.4.5

2.4.6 Headings and Labels

Write clear, descriptive headings for content and/or labels for form fields and other interactive elements.

What to do:

- Use concise, clear language for headings within a page (e.g., <h1>, <h2>) and labels on interactive elements
- A single word can be sufficient if it relays the necessary information

Plain English Explanation:

This success criterion isn't requiring headings and labels, it's merely saying that when you use them, write clearly and descriptively so a user knows what content follows or what form fields are asking for.

For example, an effective label for a form that is asking for a first name is "First name".

As another example of sufficient descriptiveness, the <h1> heading for an article might be "How to Make a Website Accessible" and an <h2> heading might be "Provide Content Alternatives".

The key here is that headings and labels are clear and descriptive so a user knows what content to expect or what input a form field is asking for.

Understanding 2.4.6

2.4.7 Visible Focus

When an interactive element (link, button, form field, selectable element, etc.) receives focus, a visual indicator shows so a user can easily see what element they are currently on.

What to do:

- Style your website so that any interactive element shows visible focus when it receives focus

Resources:

- CSS code example for styling a visible focus

Plain English Explanation:

Make sure that when someone tabs through your website, any interactive element (such as a link or form field) that receives focus has a focus indicator of some kind.

An interactive element is one in which a user can engage with or take an action on (e.g., click on or input a value).

Typically, a focus indicator is a rectangle box around an interactive element.

I recommend you make focus very clear and conspicuous by providing a strong, contrasting color for a box border or potentially highlighting an input field such as a search box.

There are many styles you can use to provide visible focus.

Understanding 2.4.7

3.1.1 Default Language

The default language for each page of your website is set in the code.

What to do:

- Make sure your language is set inside your HTML code (the example below shows US English)

Resources:

- List of language codes

Plain English Explanation:

Whatever the primary language is for your website, make sure that is the default language set in your website's HTML code.

To check this right now, type

view-source:https://yourwebsite.com

into Chrome or Firefox.

Look at the very top for <html lang=

Understanding 3.1.1

3.1.2 Language Change

If the language of a page or on parts of a page is different from the default language of the website, the change needs to be indicated in the HTML code.

What to do:

- Use HTML markup to indicate a language change when you switch to words or phrases that are meant to be read in another language
- For any links to alternative versions of the website/page in another language, use the HTML markup to indicate a language change
- Note that some adopted words can be incorporated into content without calling for a language markup (e.g., the use of “siesta” to refer to a nap in an English blog)

Resources:

- How to set language of a paragraph
- How to set language of a link

Plain English Explanation:

If there is a language switch for a page, linked page, or paragraph, set the language for the switch in the HTML code.

Understanding 3.1.2

3.2.1 No Automatic Change on Focus

Nothing on a website should automatically change just because it receives focus.

What to do:

- Check your website to make sure nothing automatically happens just because an interactive element (such as a link, form field, selectable option, etc.) is tabbed through or on
- An automatic change can be a pop-up, switching focus to another element, or automatically taking action as if the user had initiated action (e.g., clicking a link, submitting a form, etc.) when all they had done was focus on an element

Plain English Explanation:

You always want to put the user in control of your website's content and functions. This success criterion speaks to that by requiring that no elements automatically change just because they receive focus.

For example, if you tab on the submit button after completing a form, the form should not automatically submit. The form should only submit once you initiate action.

Here we're making sure our website only responds once the user initiates reaction, never because they shifted focus onto an element.

Most websites should not have a problem with this.

Understanding 3.2.1

3.2.2 No Automatic Change on Input

Nothing on a website should automatically change just because a user inputs text, checks a box, or navigates down a drop-down box.

What to do:

- Check your website to make sure nothing automatically happens just because an interactive element receives input

Plain English Explanation:

This is almost the same as 3.2.1 but this time we're making sure that adding input doesn't create an automatic change. In 3.2.1 we were talking about focus.

For example, inputting text into a registration form should not result in the form being submitted.

As another example, checking a check box next to the shipping address you want to use for your order doesn't automatically complete the checkout.

Can you see that familiar theme shining through once again?

Put the user in control.

3.2.2 shouldn't be a problem for your website but it is definitely something that needs to be verified and accounted for.

Understanding 3.2.2

3.2.3 Consistent Navigation

Keep your navigation menu placement and order consistent throughout your website.

What to do:

- Make sure your header and footer menus stay consistent throughout your website
- Consistency involves placement and order of links, search bars, and skip navigation links

Plain English Explanation:

If your page contains a navigation menu (in the header, footer, or both), make sure it is consistent with the menu(s) on other pages of your website.

3.2.3 is getting after predictability. When it comes to website navigation, we don't want to switch things up on users.

We want users to be familiar with where the search bar is located on our website.

We want users to know the order of our navigation menus.

Understanding 3.2.3

3.2.4 Consistent Identification

Page components (e.g., links, buttons, icons) that have the same functionality should be identified (with labels, names, and text alternatives) consistently throughout a website.

What to do:

- Keep all of your identification consistent throughout your website
- Consistent usually means identical but not always

Examples:

- The Twitter share icons for your website should all have the same alt text throughout the website
- Your search bar should have the same label throughout your website (unless you have multiple search bars for different functions)
- The same check mark icon used for different functions on a website should have different alt text to match its different functions (e.g., “correct”, “finished”)

Plain English Explanation:

If you read through this success criterion too quickly, you might come away thinking that everything that looks identical needs to have the same identification.

This is incorrect.

Rather, here, we’re just trying to make sure we have consistency in our labels. Sometimes this might mean that things stay the exact same across our website. However, in other instances where the functionality is different, we merely want to maintain consistency.

For example, the same exact button within a series of pages might be labeled, “Go to Page 4” and “Go to Page 5”. Note this is consistent but not identical.

With 3.2.4, we need to maintain consistency within our website by identifying things with the same functionality or meaning in a consistent manner. If something has the same function throughout a website, it will probably have the same identification on all pages but it might not depending on the function.

Understanding 3.2.4

3.3.1 Input Errors

When any input error is automatically detected, alert the user, describe the error, and provide instructions on how to correct it.

What to do:

- Clearly and specifically identify any errors that can be detected automatically
- Provide precise instructions on how to fix that error, potentially including an example where applicable (e.g., correct date format)
- Make it easy to find where the error is by adding a visual indicator

Resources:

- Amazing video that explains input error best practices (YouTube – 22 min 10 secs) --- Developers, watch this video
- Example and explanation of code for error identification

Plain English Explanation:

When a user is filling out a form on your website, for any errors that are automatically detectable, make it easy for users to correct the error. You can do this by:

- Clearly identifying what and where the error is
- Providing clear, precise instructions on how to fix the error

Examples:

When someone enters in an incorrect email format (e.g., tom@gmailcom)

When someone enters a date not in the required format (e.g., entering 21 instead of 2021)

When someone fails to enter a value in a required field (e.g., no state on an address is selected)

Remediating to meet this success criterion is not a beginner's task. However, it's fairly easy to understand what 3.3.1 is asking for and check to make sure your asset's forms meet the requirements.

Many websites currently fail this success criterion and it's an extremely important one.

I highly, highly recommend watching the video in the resources section as 1) it thoroughly explains 3.3.1 and provides numerous examples and 2) it makes some improvements on even WCAG's recommendations.

Understanding 3.3.1

3.3.2 Labels and Instructions

For any element that requires user input, concise labels and/or instructions are provided.

What to do:

- Assign a visible label for every form field or area of user engagement/control
- Inside the code, make sure labels are associated with the appropriate form fields
- Do not overdo instructions, one word or a few can suffice
- Provide examples of expected input formats or structure data input so that the expected format must be entered
- Conspicuously mark required fields with an icon

Resources:

- Great tutorial on creating accessible forms (YouTube – 51 min 34 secs)

Plain English Explanation:

Every element that allows for user input should have clear, concise, descriptive labels and, where applicable, instructions.

A label could be “First Name”.

An instruction could be “All fields marked with an * are required.”

It’s very important that you don’t overdo your instructions and bog down the process of inputting data.

For example, rather than simply labeling a form field as “email”, you put:

“your email that you prefer to receive our newsletter at”

The purpose of this success criterion is to make it easy to understand what data input is being asked for.

Understanding 3.3.2

3.3.3 Error Suggestions

Make suggestions on how to fix errors on forms if an input error is automatically detected.

What to do:

- Provide clear, specific instructions on how to fix input errors.
- Make it easy to correct form fields.
- Preserve other form data so the entire form doesn't have to be re-filled out

Plain English Explanation:

This looks very familiar, right?

3.3.3 is an extension of 3.3.1. Whereas 3.3.1 requires notification of errors, 3.3.3 takes it a step further by requiring suggestions and instructions to help users correct those errors.

We've already addressed this in 3.3.1. The key takeaway with this success criterion is to be thorough in helping users fix input errors.

Understanding 3.3.3

3.3.4 Prevent Serious Errors

For any web pages that have important implications (legal commitments, rights waivers, spending money, scheduling reservations, etc.), make sure that at least one of the following options is available:

1. Submissions are reversible
2. Input is checked for errors and the user is provided an opportunity to correct errors
3. There is an option to review, confirm, and correct information before finalizing the submission

What to do:

- Provide ample opportunity for users to review and correct any potential errors, especially when it comes to stuff with big implications

Examples of big implications:

- Legal contracts
- Money being spent
- Reservations

Plain English Explanation:

We want to make sure everyone gets their desired result when making a submission, especially when it comes to things with big implications.

For example, imagine the difference one “0” can make it when it comes to buying a $25 stock. If instead of entering “100” shares to purchase, I enter “1000”, I just committed to spend $25,000 instead of $2,500.

Or imagine, if when booking a hotel, I select “September” as my month instead of “August”.

As we all know, it's extremely easy to make these types of errors. And this is why it's so crucial that we provide users with ample opportunity to review, correct, and/or reverse any submissions they make on our website.

I recommend you incorporate as many ways as possible to prevent errors:

1. Where feasible, allow users to reverse a submission after it is made. For example, you provide for a window of time for someone to completely cancel/undo/delete a submission or agreement.
2. Always automatically check for detectable errors and provide an opportunity to correct them.
3. Before any final submission is made, users are given an opportunity to review, confirm, and/or correct a potential submission.

If your website potentially falls under the serious implications/transactions category (think legal, financial, data, reservations, and anything else with a big impact), scrutinize your form submission process heavily.

Understanding 3.3.4

4.1.1 Use Good, Clean Code

Your website should follow best practices and standards for coding. Any notable errors in code should be fixed.

What to do:

- Make sure all elements have start (open) and end (close) tags
- Nest all HTML elements correctly
- Make sure elements do not contain duplicate attributes
- Make sure element IDs are unique
- Check for and eliminate broken links

Resources:

- Markup Validator (check for errors in your code – not all, in fact many errors won't be relevant for accessibility and some errors won't show)
- How to fix HTML with examples

Plain English Explanation:

This is a very broad and sweeping requirement. It ensures that browsers, screen readers, and other assistive technologies can analyze and make your website's code understandable ("parse" is the technical word) so that your website is rendered correctly.

Many browsers can go past errors and still render a website correctly but it's bad practice to rely on browsers and assistive technologies to interpret incorrect code.

What 4.1.1 boils down to is making sure your website's code is fundamentally sound by checking and testing.

Developers should look over and check your code.

Everyone should test your website, especially using screen readers, to make sure it renders correctly.

The markup validator above will help with catching some errors but it is not a substitute for a manual examination of your code.

The most important takeaway here is to make sure your website is free of any major coding errors that prevent assistive technologies like screen readers from accurately rendering your website.

Note: Assistive technologies are any technologies that help persons with disabilities access the web. Screen readers are the most common assistive technology but others include screen magnification software, text readers, voice to text or speech input software, and other devices that facilitate access, navigation, and input.

Understanding 4.1.1

4.1.2 Custom Components Are Accessible

For all user interface components (e.g., forms, links, scripts, controls, etc.), the name and role of those components is coded in. Also, any states, properties, and values set by the user can be programmatically updated so browsers and assistive technologies are aware of and reflect changes.

This success criterion applies specifically for custom components such as ones a developer has created or from third party vendors.

What to do:

- Make name, role, state, and value of all components determinable
- Use ARIA labels when HTML labels are not available
- Check any code that is not HTML to make sure it is accessible
- Any custom code (plugins, widgets, scripts) that is from a third party needs to be vetted for accessibility

A name can be a label or the assigned text to an element.

A role is whatever a component is stated to be; what the component is. For example, a component can be assigned a role of button, drop down list, progress bar, etc. and that role will be relayed to a screen reader.

Value is whatever the input is for a component. For example, value might be the slider value selected (e.g., 4 on a scale of 1-10) or the selection of yes or no. Not all components will have a value.

State is the current status of a component. For example, state might be whether a checkbox is checked or unchecked or whether a component is collapsed or expanded.

Resources:

- Understanding role, name, and value (YouTube – 1 min 2 secs)
- Why name, state, role, and value are important

- ARIA basics
- iFrame explanation and examples

Plain English Explanation:

When you deviate from HTML in your website's code, you need to make sure whatever custom or third-party code you've added remains accessible to assistive technologies.

This mostly comes down to the name, role, state, and/or value of page components being determinable from the code.

To maintain accessibility for custom or third-party components, you'll need to incorporate ARIA labels and possibly iFrames.

ARIA will take a while to understand but if your website has dynamic components, you'll need to learn how to incorporate it into your code. The ARIA basics link above provides a great introduction into ARIA.

With iFrames, you're embedding content to a page from external sources. Think of YouTube videos and advertising scripts you paste into your site.

Understanding 4.1.2

WCAG 2.1 AA

1.3.4 Orientation

WCAG Official Success Criterion:

Content does not restrict its view and operation to a single display orientation, such as portrait or landscape, unless a specific display orientation is essential.

Read the official WCAG page for 1.3.4

My Interpretation:

What to do:

- Style your website or mobile app so that it does not lock on or require a single display mode

Resources:

- Adjusting layout based on orientation

Plain English Explanation:

Unless absolutely necessary, make sure that your website and/or app and its content can display well in both portrait (vertical) and landscape (horizontal) mode on phones, tablets, etc.

The only reason you should lock a page display is if it's absolutely necessary.

For example, when you deposit a bank check, you need to snap the photo horizontally so a lock of landscape would be permissible.

Not only should your website generally meet this success criteria but you also need to check and ensure that video embeds, widgets, and any third-party integrations also display well properly no matter which way a mobile device is turned.

Understanding 1.3.4

1.3.5 Identify Input Purpose

WCAG Official Success Criterion:

The purpose of each input field collecting information about the user can be programmatically determined when:

- *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*

Read the official WCAG page for 1.3.5

My Interpretation:

What to do:

- Update your forms to provide for autocomplete
- Specify the type of the type of input (i.e., text)
- Add autocomplete to the code (e.g., autocomplete=”username”)

Resources:

- Sample code for simple form (scroll to bottom of page)
- W3 list of available autocomplete information

Plain English Explanation:

Enable the ability to have repetitive information (such as name, username, email, phone number, address, and email) autocompleted when users fill out forms.

This makes filling out forms much easier for anyone.

To meet this success criteria, add autocomplete to form fields asking for user information wherever possible.

The information will be autocompleted based on previous user input stored in the browser.

Understanding 1.3.5

1.4.10 Reflow

WCAG Official Success Criterion:

Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions for:

- *Vertical scrolling content at a width equivalent to 320 CSS pixels;*
- *Horizontal scrolling content at height equivalent to 256 CSS pixels.*

Except for parts of content which require two-dimensional layout for usage or meaning.

Read the official WCAG page for 1.4.10

My Interpretation:

What to do:

- Ensure your CSS enables your website content is able to zoomed up to 400% without causing 1) loss of info/function and/or 2) users to scroll to see it

Resources:

- Responsive Web Design Basics
- CSS Media Query (YouTube video 6 minutes)
- ResizeMyBrowser.com

Plain English Explanation:

The two goals here are 1) to make it so all functionality is possible and all information is available when a website is zoomed-in and 2) to allow people to increase the size of content without having to scroll to see it.

For some content this the no-scrolling isn't possible. For example, if you have a map or a table of data on your website, it might not fit neatly within a single column without making you scroll.

Some objects are just large and can't be effectively resized to meet this success criteria. For those objects, there is an exception.

This is a fairly straightforward success criteria that helps users who are using mobile devices or who have low vision and need to increase the size of the content.

What it allows for is smooth scrolling in one direction (the direction can either be vertical or horizontal) without the need to scroll in a second direction to view content that is off the screen.

Imagine if you were looking at Instagram on your phone but to see the comments for a picture, you had to scroll to the side of the picture instead of beneath it.

That's getting at what reflow is all about: We want people to be able to be able to effectively access content without having to scroll back and forth.

Again, when I say effectively it means that no functionality is lost (e.g., I can no longer add a product to my shopping cart) and no content is lost (e.g., the text overlaps and now makes an article unreadable).

This applies to any device including desktop. Some users with low vision may need to zoom to 400% and, even if they do, they should still be able to effectively scroll your website in one direction without having to scroll another direction.

If your website is already responsive, then odds are you meet this success criterion. However, don't forget about any embeds, widgets, scripts, or third-party integrations that may not reflow.

One way to get a good gauge on whether you meet this success criterion is to open up a browser on a 1280 pixel screen and zoom to 400%. If your website still functions properly and there is no loss of content, then you're very likely in good shape.

I’ve included a link to ResizeMyBrowser.com in the resource section. If you go to ResizeMyBrowser.com, you can easily set the dimensions of your browser and then all you need to do is zoom in from there.

Understanding 1.4.10

1.4.11 Non-text contrast

WCAG Official Success Criterion:

The visual presentation of the following have a contrast ratio of at least 3:1 against adjacent color(s):

User Interface Components

Visual information required to identify user interface components and states, except for inactive components where the appearance of the component is determined by the user agent and not modified by the author;

Graphical Objects

Parts of the graphics required to understand the content, except when a particular presentation of graphics is essential to the information being conveyed.

Read the official WCAG page for 1.4.11

My Interpretation:

What to do:

- Check that all meaningful non-text content on your website has sufficient contrast with the background.

Resources:

- Accessibility Color Contrast Checker

Plain English Explanation:

This success criterion is just an extension of the text color contrast ratio criterion from 2.0 AA.

We want to make sure that important non-text content that conveys information is easier to see by insuring it has a proper color contrast with its background. This will help those with low vision.

For an example of non-text content, think of a download icon. That icon conveys information so it needs sufficient contrast.

The WCAG link below is particularly helpful with details and providing examples of what non-text content needs to be addressed and what constitutes a pass or fail.

Understanding 1.4.11

1.4.12 Text spacing

WCAG Official Success Criterion:

In content implemented using markup languages that support the following text style properties, no loss of content or functionality occurs by settings all of the following and by changing no other style property:

- *Line height (line spacing) to at least 1.5 times the font size;*
- *Spacing following paragraphs to at least 2 times the font size;*
- *Letter spacing (tracking) to at least 0.12 times the font size;*
- *Word spacing to at least 0.16 times the font size.*

Read the official WCAG page for 1.4.12

My Interpretation:

What to do:

- Make sure your text spacing is able to be adjusted

Resources:

- Steve Faulkner's Text Spacing Bookmarklet

Plain English Explanation:

Text spacing must be adjustable without creating any problems in reading or using the website.

This makes sense because text is easier to read when there is more space and we want people to have the freedom to increase spacing based on what suits them best.

The best way to test if you meet this success criteria is by using Steve Faulkner's text spacing bookmarklet above.

The way you use this is you add that bookmarklet to your browser bookmarks and then you go to the website you want to test and then you open Steve Faulkner's bookmark.

That's going to adjust your website's spacing. If everything still looks good (no loss of content or function) once the spacing has increased, you're in good shape.

If not, you'll want to adjust your CSS so that your website is fully responsive.

Understanding 1.4.12

1.4.13 Content on Hover or Focus

WCAG Official Success Criterion:

Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true:

Dismissible

A mechanism is available to dismiss the additional content without moving pointer hover or keyboard focus, unless the additional content communicates an input error or does not obscure or replace other content;

Hoverable

If pointer hover can trigger the additional content, then the pointer can be moved over the additional content without the additional content disappearing;

Persistent

The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

Exception: The visual presentation of the additional content is controlled by the user agent and is not modified by the author.

Read the official WCAG page for 1.4.13

My Interpretation:

What to do:

- Make it so any additional content (e.g., pop-ups, submenus) can be dismissed or remain visible if the user desires

Resources:

- Deque.com homepage

Plain English Explanation:

If a user on your website does something that causes more content to show (e.g., a pop-up, submenu, instructions, etc.), then you have to make the following three provisions:

1) A user can get rid of the additional content without moving their pointer or tabbing onto something else (e.g., by hitting the ESC key)
2) The additional content will not disappear if the user moves their pointer over it
3) The additional content remains visible until pointer hover or focus is removed, the user dismisses it, or the content is no longer valid/applicable

This is a tough monkey to banana.

A website and its content should remain fully accessible when additional content is triggered.

You know when you move your mouse over something and more information pops up as a result?

We're trying to make it so that additional information is accessible and doesn't cause 1) the user any inconvenience and 2) anything else to be inaccessible.

But how do you actually code this in?

I've scoured Google for a resource on specifically how to implement this success criteria and was unable to find anything.

Deque.com, a good accessibility company, has submenus and they follow this success criterion so you can certainly look to their homepage for an example (I've linked to Deque in the resources section above).

Understanding WCAG 1.4.13

2.1.4 Character Key Shortcuts

WCAG Official Success Criterion:

If a keyboard shortcut is implemented in content using only letter (including upper- and lower-case letters), punctuation, number, or symbol characters, then at least one of the following is true:

Turn off

A mechanism is available to turn the shortcut off;

Remap

A mechanism is available to remap the shortcut to include one or more non-printable keyboard keys (e.g., Ctrl, Alt);

Active only on focus

The keyboard shortcut for a user interface component is only active when that component has focus.

Read the official WCAG page for 2.1.4

My Interpretation:

What to do:

- If you have a keyboard shortcut, make sure to modify it to satisfy the three requirements above

Plain English Explanation:

Think of this success criteria in the terms of someone who uses voice commands. If you use voice command and your website or app has a shortcut that is triggered by the command, then someone might get an unintended action.

Also, someone may accidentally hit a key which, again, can generate an unintended action.

We're trying to prevent these accidental keyboard triggers by providing a way to turn off keyboard shortcuts entirely, including a second key in the action, or limiting the shortcut to only when a specific component is activated.

Here is this success criteria one more time:

If you have a keyboard shortcut on your website, then do any or all of the following:

1. provide a way to turn it off,
2. provide a way to change the shortcut to include Ctrl or Alt, and/or
3. make the keyboard shortcut only available when a specific component is activated

Understanding 2.1.4

2.5.1 Pointer gestures

WCAG Official Success Criterion:

All functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture, unless a multipoint or path-based gesture is essential.

Read the official WCAG page for 2.5.1

My Interpretation:

What to do:

- Provide simple alternatives to potentially complex finger motions on touch screens

Plain English Explanation:

I immediately thought of a dating app when reading this success criterion.

With dating apps like Tinder and Bumble, you can swipe right if you want to match with someone and left if you don't.

But for some people, swiping might be difficult so you want to provide a way for them to indicate yes or no without requiring a swipe. The way to do this is by providing two buttons, one indicating yes and another indicating no that only need to be tapped or clicked to create the desired effect without the swipe.

That's the essence of 2.5.1: provide a simpler alternative to potentially complex finger motions on screens.

Not everyone has the ability to readily swipe, drag, or pinch a screen so we want to account for that and provide an alternative way of accomplishing the same thing.

So, for anything on your website or app that calls for a user to use two or more fingers, to move their finger or (mouse) pointer in a specific direction, or anything along those lines, make an easier alternative available such as single-click (or tap), double-click (or tap), or click and hold.

For example, think of how we can pinch to zoom screens. All you'd have to do to make an alternative here is to provide a way to tap to incrementally increase browser size.

Note: You don't have to get rid of the more complex finger movements, you just need to make an easier alternative available.

Understanding 2.5.1

2.5.2 Pointer cancellation

WCAG Official Success Criterion:

For functionality that can be operated using a single pointer, at least one of the following is true:

No Down-Event

The down-event of the pointer is not used to execute any part of the function;

Abort or Undo

Completion of the function is on the up-event, and a mechanism is available to abort the function before completion or to undo the function after completion;

Up Reversal

The up-event reverses any outcome of the preceding down-event;

Essential

Completing the function on the down-event is essential.

Read the official WCAG page for 2.5.2

My Interpretation:

What to do:

- Provide a way to cancel pointer input

Plain English Explanation:

The first thing you should know is down-event means when you click down on a mouse or press down/touch with a finger.

The second thing you should know is up-event means when you release the mouse or lift up your finger.

If you click or press down and that causes or triggers an activation, that's a down event.

And if you release the clicker or your finger and it causes or triggers an activation, that's an up event.

This should clear up a lot.

The easiest way to satisfy this success criterion is simply to make it so that the release of a finger or mouse button (and not the initial click or touch) causes activation or execution of the action/event.

You can also add a cancel and/or abort option which means the user can move their mouse or finger to somewhere else and release with no effect. Another option is to actually provide the ability to undo the action or confirm they want to proceed with the action. This could be as simple as a confirmation pop-up.

The up reversal part of this means that when the down event triggers a behavior, there is an up event that reverses it when the up event concludes. So if you hold a button down until a video plays but the video disappears as soon as you let go, this counts because the user can cancel the original trigger by letting go.

Think of this as the no harm-no foul part of this success criterion. If you can activate something by clicking down but you can let go to cancel it, you've still provided the user a means of control.

The only way a down action should be available is if it's essential. WCAG gives the example of a piano requiring down key activation which makes sense.

Understanding 2.5.2

2.5.3 Label in Name

WCAG Official Success Criterion:

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

Read the official WCAG page for 2.5.3

My Interpretation:

What to do:

- Make sure all visual text and programmatic labels are aligned

Plain English Explanation:

Think about all of the buttons, form fields, and interactive elements like selectors and checkboxes where you have where the programmatic label (the label that is inside the code) may be slightly different from what visually shows on the website.

Programmatic labels include HTML labels, ARIA labels and alt text.

For example, let's say your button visually says "buy" but the programmatic label you have associated with it is "order".

You'd want to change this so that the visual label and programmatic label are aligned.

If there is a slight difference in length, start the label and text off with the same few words.

For example, visually a user might see "buy desktop" but programmatically the label might be "buy desktop computer". As long as there are no other labels that start with "buy desktop", you're in good shape.

It's actually somewhat common that websites have a few of these mismatches.

As another example, a password field may be labeled as login.

This success criterion benefits users who use text-to-speech and voice commands. If they make a command based on what is presented visually and there is nothing that corresponds in the code, then they can’t make the desired action.

This is fairly easy to fix. Just run through your website and make sure that all interactive components with labels have their labels match what is visually presented.

Understanding 2.5.3

2.5.4 Motion Actuation

WCAG Official Success Criterion:

Functionality that can be operated by device motion or user motion can also be operated by user interface components and responding to the motion can be disabled to prevent accidental actuation, except when:

Support Interface

> *The motion is used to operate functionality through an accessibility supported interface;*

Essential

> *The motion is essential for the function and doing so would invalidate the activity.*

Read the official WCAG page for 2.5.4

My Interpretation:

What to do:

- If you use functions that are activated by motion, provide a simpler, alternative means of action. Also, give users the option to turn off the motion activation.

Plain English Explanation:

If your website or app has functions that are initiated/activated/triggered by moving a phone/tablet (e.g., shake to do undo, tilt to move) or by the user making gestures to a camera or sensor, these same responses can be made an alternative way (e.g., through a keyboard, pressing a button, voice command, and/or etc.)

Additionally, users should have the option to turn off the moving or gesture activation.

For example, imagine you're taking an Uber from the airport and every time the driver hits a pothole, your phone shakes and the app you're using undoes your previous action. You'd want the ability to turn off that shake feature and make it so only when you press or say undo, etc., your previous action is undone.

The key here is to make it so your website or app is fully functional without the use of device movement or gestures. Of course, your asset can still have these features but you'll want to provide alternatives.

This success criterion provides an exception to those movements or gestures which are essential.

Understanding 2.5.4

4.1.3 Status Messages

WCAG Official Success Criterion:

In content implemented using markup languages, status messages can be programmatically determined through role or properties such that they can be presented to the user by assistive technologies without receiving focus.

Read the official WCAG page for 4.1.3

My Interpretation:

What to do:

- When a status message appears, it should be coded with role or properties so that people using assistive technologies (e.g., screen reader) are alerted without losing focus

Plain English Explanation:

Think of someone using a screen reader while shopping on a website and they decide to remove an item from the shopping cart. A notification visibly shows at the top of the website that says, “Item removed” but the shopper is not alerted to the status change via their screen reader.

Implementing this success criterion changes that.

With 4.1.3, a user now will receive those nice notifications/status messages because they will be coded in through either role or properties.

As a result, users will be notified without having to change their focus; They will be notified of updates/receive status messages without having to take any proactive measure.

Read 4.1.3 to see what actually constitutes a status message.

Basically, first, a status message is anything that isn't directly brought to the user's attention.

Second, a status message is information that tells a user whether something was successfully completed or what happened, whether a page is loading or busy, how far along they are in a process, or alerts them to errors on a page.

Status alerts and notifications are subtle but can be necessary and play an important when we use websites or apps as they guide us through processes and help us reaffirm the actions we're making.

It's important to distinguish status messages from more significant alerts that do warrant a change in focus. For example, if someone makes a critical error and a dialog pops up or they are taken to a confirmation page to confirm their reservation before completing checkout.

Understanding 4.1.3

Great Job

Just making it through this guide means you deserve some kind of award.

Yes, there is a lot to do (you need to account for each success criterion across your entire digital asset) but at least now you're more aware of what you need to do.

Thanks for reading my guide. Kris.

Products

I have two accessibility products for sale.

First, my ADA Compliance Course. This course is specifically designed to quickly help anyone understand how to 1) lower litigation risk and 2) understand, organize, and prioritize WCAG success criteria.

Second, my Accessibility Statement template. This template was written with best practices as extracted from Department of Justice (DOJ) private enforcement actions.

The DOJ regulates and enforces Title II and Title III of the ADA.

Here are my channels and websites:

https://www.linkedin.com/in/krisrivenburgh/

https://medium.com/@krisrivenburgh

https://krisrivenburgh.com

https://adabook.com

https://accessible.org

The ADA Compliance Course is my personal course on how to best prevent litigation against your website or mobile app.

In this course, you will learn:

- What the best practices for ADA website compliance are (as interpreted from DOJ private enforcement actions and litigation activity)
- What each WCAG success criteria is asking for (along with extended explanations, examples, illustrations, and curated resources)
- How to prioritize accessibility issues for risk mitigation

- What roles should be responsible for success criteria (this can help with designating or outsourcing work)

The ADA Compliance Course contains the exact steps I recommend to immediately reduce your risk and follow best practices for ADA website compliance.

I'm an attorney who researches digital accessibility and related law and litigation.

You can learn more about the ADA Compliance Course at Accessible.org.

If you have any questions, you are welcome to contact me at kris@accessible.org.

Thank you very much.