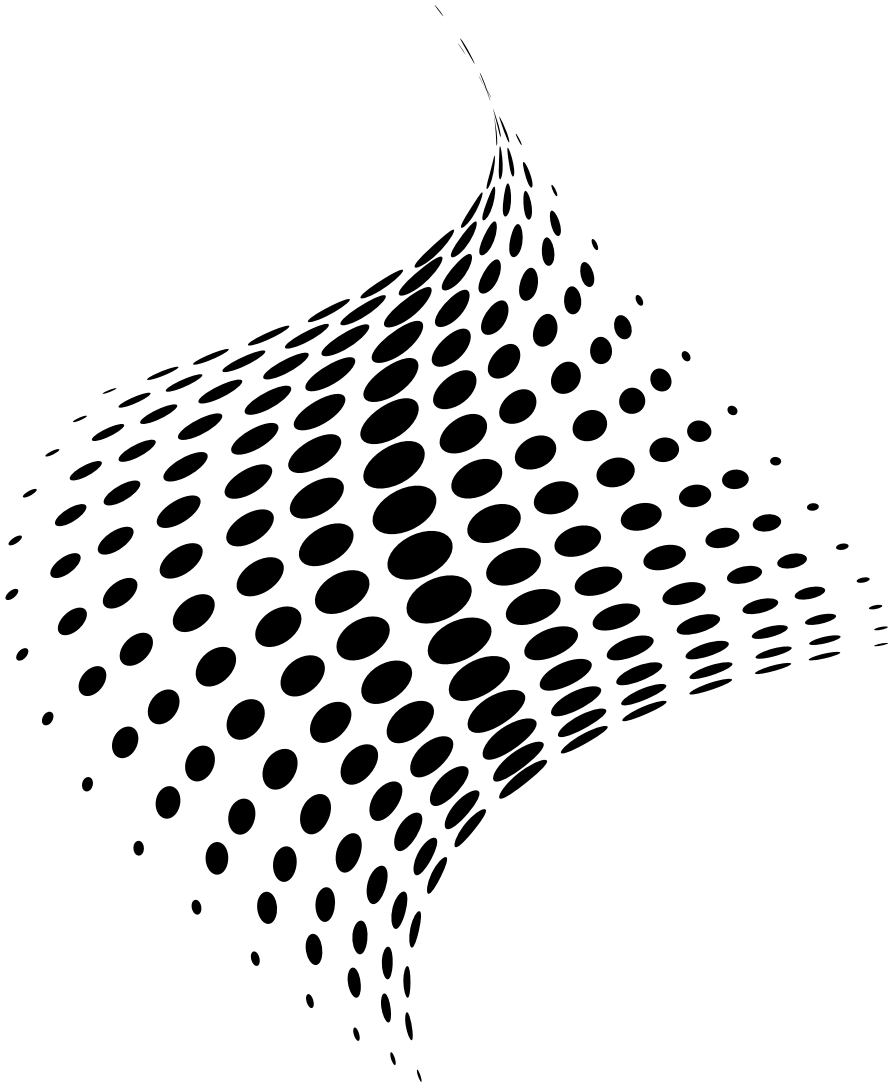
4A's Digital Horizons Series

Accessibility

The Why and How of Inclusive Design

Alison Walden
Director of Technology, SapientRazorfish

Mike Quattrin
Senior Experience Developer, SapientRazorfish



4A's SAPIENT RAZORFISH

4A's Digital Horizons Series

Volume I, No. 3

A publication of the 4A's (American Association of Advertising Agencies), the 4A's Digital Horizons Series explores the current and near future landscape of digital technology and innovation in the advertising industry.

Contributing authors represent digital thought leaders and practitioners from member agencies and associated industries.

For more information about the series, contact Chick Foxgrover - cfoxgrover@aaaa.org

4A's
1065 Avenue of the Americas (5 Bryant Park)
New York, NY 10018
+1 212.682.2500
www.aaaa.org

4A'S BULLETIN NO. 7927

Accessibility

The Why and How of Inclusive Design

By Alison Walden and Mike Quattrin

Digital experiences are pervasive and growing in importance. A combination of legal requirements, best practices and demographic realities make it important for agencies to understand the field of inclusive design.

Inclusive design addresses a variety of legal requirements, implements best practices—from initiation to maintenance—and makes experiences more accessible to our clients' audiences, increasing the scope and reach of who can engage with brands through digital channels.

Creating inclusive experiences that meet accessibility guidelines is not only the right thing to do from a human perspective, it also makes good business sense as we make experiences available to a wider audience, and from a client perspective by helping to protect our clients from potential lawsuits. Read on for a detailed analysis of how to approach accessibility compliance and overcome common challenges faced by disciplines creating accessible experiences.

In this paper we'll offer a usable definition of inclusive and accessible design as well as a case for implementing inclusive design for clients; an overview of current government rules and regulations and how they impact digital experience design; and a top-level view of implementing inclusive design, followed by a role-by-role, detailed description of the process as well as an addendum dealing with specifics and best practices for design and QA. Any agency wishing to get started with inclusive design will find the necessary tools in these pages.

INTRODUCTION

Defining Inclusive/Accessible

We define "inclusive design" to mean experiences that are usable and accessible by the most diverse set of individuals with a variety of skill levels, challenges, and backgrounds. We define "accessible" to mean that everyone, regardless of disability or special needs, can access content.

The Internet for People with Disabilities

From the late 1990s onward, the internet gradually became part of our lives, until in 2011 it was estimated that 2.1 billion people regularly used "the web".[1] With the growth in internet usage, physical modes of content have been steadily replaced by electronic modes of content, and commerce at physical locations has been replaced by electronic commerce storefronts. This shift had the potential to remove many of the barriers to communication that existed for people with disabilities in the physical world, but only if websites were designed and developed accessibly.

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.[2]

The web is recognized as "an increasingly important resource in many aspects of life: education, employment, government, commerce, health care, recreation, and more."[3] It is critical that internet content be designed and developed in an accessible way so that everyone can use the web, regardless of disabilities. If content creators from across disciplines work together, the potential of the web to provide "unprecedented access to information and interaction for many people with disabilities"[4] can be realized.

Many courts and lawmaking bodies, both in the U.S. and internationally, have recognized the increasing importance of the internet in everyday life by establishing standards for electronic and information technology to ensure that digital experiences are accessible to all individuals regardless of disabilities.

As digital experiences become more pervasive, the need for inclusive design practices becomes more critical. The ubiquity of mobile computers—smartphones—means that generational cohorts who are not "digital natives" still consume digital experiences at ever increasing rates. As these populations grow older, the toll of the human condition prevents people from seeing that light weight, light grey font on a white background. So it prevents customers from using the client's product. Inclusive design is a good practice because it delivers the most value to the widest audience possible.

ACCESSIBILITY MYTHS [5]

- **Accessibility can be taken care of by developers, and nobody else needs to concern themselves with it.** Inclusive design is a critical part of making an experience accessible.
- **Creating separate accessible versions of a website is a good idea.** Users who self-identify as needing an accessible website experience "fear of missing out." They often assume that the accessible version of the website is not updated as frequently, or is missing some interesting aspect provided to the "regular" website. We don't want to make people feel marginalized.
- **Accessible design is ugly.** Accessible design is as ugly as a designer makes it. Contrast rules are just one more constraint upon many when designing for the web. The idea is to think of these as beautiful constraints, and to find a way to make accessible designs beautiful.
- **Accessible sites take much more effort to build.** Once teams are accustomed to thinking in terms of inclusive design, the lead time to make an experience accessible decreases. Consider the lead time teams used to need to make a responsive experience. But now that teams know how to do that, it takes considerably less time than it did originally. Also, it takes much longer to fix accessibility issues on a site later than to build in accessibility thinking and inclusive design from the outset.
- **You can implement an accessible site and call it a day.** You must maintain it or the experience will gradually deteriorate and become inaccessible.

THE CASE FOR INCLUSIVE DESIGN

The Myth of the Minority User

A full 16% of the U.S. population had some kind of disability in 2011. It may sound like a minority of users, but the numbers are significant. The number of blind users in the U.S.[6] almost matches the entire Canadian population that is using the internet[7]. This is significant! Along with the importance of providing basic access to content, tapping into an audience of this size makes sense from a business perspective.

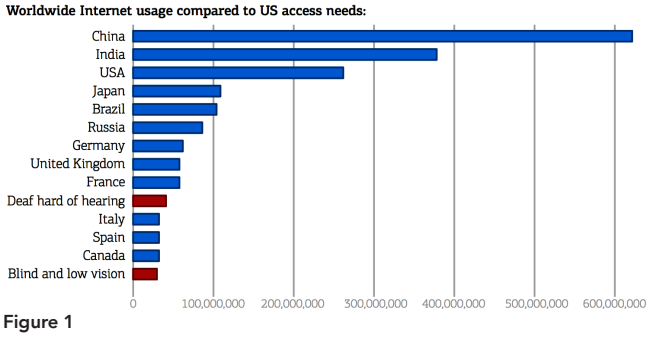


Figure 1

Figure 1 shows worldwide internet usage by select countries from 2015, compared to the number of people using the internet in the U.S. with accessibility needs.

Let's focus on vision problems in the U.S. Of the 10 million people with vision issues, some people are completely blind, some have low vision, and some are color blind. How does this affect them?

Many non-sighted people use screen readers to access the internet through a web browser. A screen reader reads the content on the page from top to bottom in a linear fashion. Compare that to how a sighted person can experience a page. A sighted user benefits from seeing images, scanning the page layout and noticing columns, and being able to recognize sections of the page like navigation, and custom controls. There is a detailed analysis later on the screen reader experience.

Figure 2

People who use screen magnifiers don't have a good experience with text embedded in images. As text embedded in images becomes magnified, it becomes pixelated and more difficult to read (**Figure 2**). This doesn't happen with "live" text that is not embedded in images.

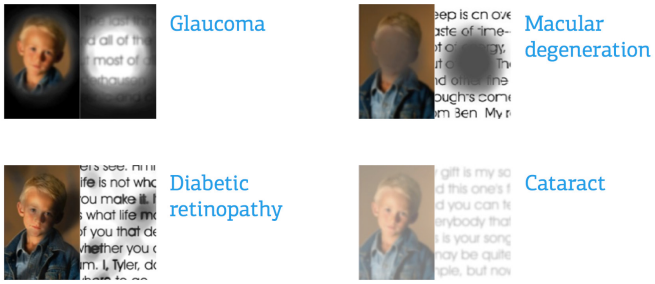


Figure 3

Low vision impacts users' experience online in various ways (**Figure 3**). People with glaucoma experience text as though they are looking through a straw. Macular degeneration, when blood vessels at the back of the eye leak fluid, obscures portions of the user's focus area. Diabetic retinopathy, the leaking of retinal blood vessels, also obscures text and makes it more difficult to read. People with cataracts will experience lower contrast in web experiences. Color blind users also experience a lack of contrast depending on the color combination used.

Inclusive Design for All

What about your average user? With the advent of smartphones, we are all subject to a degree of new challenges when trying to navigate online experiences. Glare on a screen can cut down on anyone's ability to view content.

In today's omnichannel landscape, it is more important than ever that we design and develop robust experiences that will behave as expected across a variety of form factors. Experiences that are not designed and developed properly will fail in unexpected ways, which will result in frustration for all users. Designing and developing experiences with accessibility principles in mind can provide a sense of security that experiences will be more robust across any form factor.

People surfing the internet have varying degrees of ability and dexterity. We have all ages of people interacting with online content, with degrees of vision loss, who would not even self-identify as having vision issues. When we design for accessibility, nobody is left out. "Design for people who are young, old, power users, casual users, and those who just enjoy a quality experience."[8] Websites designed with accessibility and inclusivity in mind create a better experience for all users, not just those with disabilities.

Who Benefits from Inclusive Design?

In short, everyone, including agency and client, benefits. Not only is inclusive design the right thing to do; but by building in accessibility from the beginning, the agency saves time and expense for the client, now and in the future; by offering expertise in this area, the agency ensures the widest possible audience for the client's offerings. And, it's the right thing to do.

U.S. Laws Pertaining to Accessibility Compliance

A variety of laws and rules exist within the United States mandating levels of accessibility compliance for Federal and State government websites, as well as private companies. They include but are not limited to:

- **Section 508**—an amendment to the United States Workforce

Rehabilitation Act of 1973, which applies to Federal government agencies.

- **The American Disabilities Act (ADA)**—a comprehensive civil rights law prohibiting discrimination on the basis of disability, drafted in 1990, that includes Titles II and III.
  - **Title II** applies to State and local government entities.
  - **Title III** applies to places of public accommodation (private entities whose operations affect commerce and that fall into one of 12 categories listed in the ADA, such as restaurants, movie theaters, schools, day care facilities, recreational facilities, and doctors´ offices).

Status: In 2010, an Advance Notice of Proposed Rulemaking was published by the U.S. Department of Justice (DOJ) seeking comments on what standards should be adopted with respect to compliance by public accommodations under Title III of the ADA: (1) standards then-applicable to federal agencies under Section 508, or (2) the Web Content Accessibility Guidelines (WCAG) 2.0 Level AA. These DOJ rules have been delayed several times, and will now not be issued until at least 2018. In the intervening years, WCAG 2.0 level AA has, in fact, become the new standard for accessibility under Section 508 for federal agency websites. The U.S. Architectural and Transportation Barriers Compliance Board (also known as the Access Board), the federal agency responsible for ensuring accessibility of government agency websites, adopted the WCAG standards through a January 2017 rulemaking.[9] In addition, at least one federal agency has applied the WCAG 2.0 Level AA standard to private industry; notably, the Department of Transportation has imposed the standard on airline websites and airport kiosks.[10]

Lawsuits alleging violations of the ADA are now not uncommon for companies that readily qualify as public accommodations, such as those in the retail, hospitality, and financial services industries. One review found that there had been at least 751 such suits between January 2015 and August 15, 2017, and that such suits were being filed at an ever-accelerating pace.[11] While some of these suits have been filed by public interest groups on behalf of the disabled community, many more derive from plaintiffs' law firms, likely drawn to the ADA's award of legal fees. In the absence of clear guidance, ensuring compliance with WCAG 2.0 Level AA is the best way to minimize the risk of ADA non-compliance at this time.

About WCAG

The Web Content Accessibility Guidelines (WCAG 2.0) are a series of standards that define how to make content accessible for people using digital devices. Agencies are best served by following WCAG 2.0 guidelines in order to create digital experiences that conform with existing, and future, laws and regulations.

The guidelines have levels of conformance that are broken into three levels of increasing constraints: A, AA and AAA, which are testable by people and tools.

DOJ regulations that will broadly impact public accommodations transacting business through websites and mobile apps are currently stuck in the rulemaking process, but there are many examples in which the DOJ has already acted to enforce the WCAG 2.0 Level AA standard. For example, the DOJ entered into a Consent Decree with H&R Block in 2014, in which H&R Block was required to comply with the standard to ensure accessibility for all of its websites and mobile apps. Other countries have adopted, or are adopting, legal standards that follow the WCAG 2.0 requirements. Following these standards should help our clients avoid litigation as well as deliver more inclusive experiences for a brand audience.[12]

Conforming to WCAG 2.0 guidelines requires understanding in each discipline and adherence to best practices. With that understanding, it's possible to deliver high-quality, inclusive experiences to everyone.

WCAG Principles

The goal of the WCAG guidelines is to help creative and technical teams to create experiences in which the content is perceivable, the interactions are operable, the content and functionality are understandable, and the underlying code is robust. Hence, the WCAG guidelines are organized along four principles: Perceivable; Operable; Understandable; Robust (**Figure 4,** opposite page). In plain language, we can think of the guidelines as organized as in **Figure 5,** opposite page.

This paper will not go through each of the guidelines one by one. Enough has been written to describe the guidelines, and this information is thoroughly covered by the W3C[13] and other organizations like WebAIM.

In the following sections, we'll outline the real-world challenges that each discipline will face when trying to adhere to the guidelines, and how to overcome these challenges.

WCAG MAPPING

Historically, creating accessible web experiences was considered the realm of the developer, but we recognize today that it is critical for experiences to be designed up front in an inclusive way. Having an inclusive mindset is the most critical factor in designing experiences that everyone will enjoy.

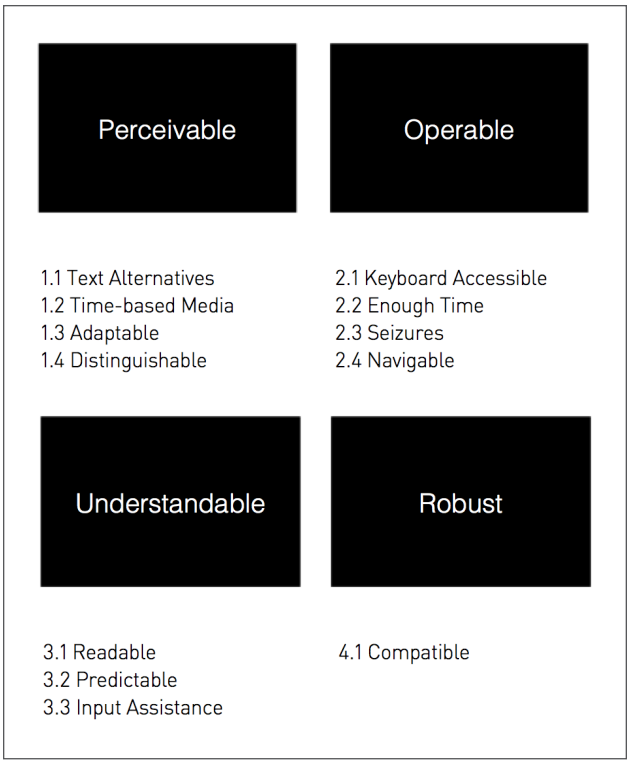


Figure 4

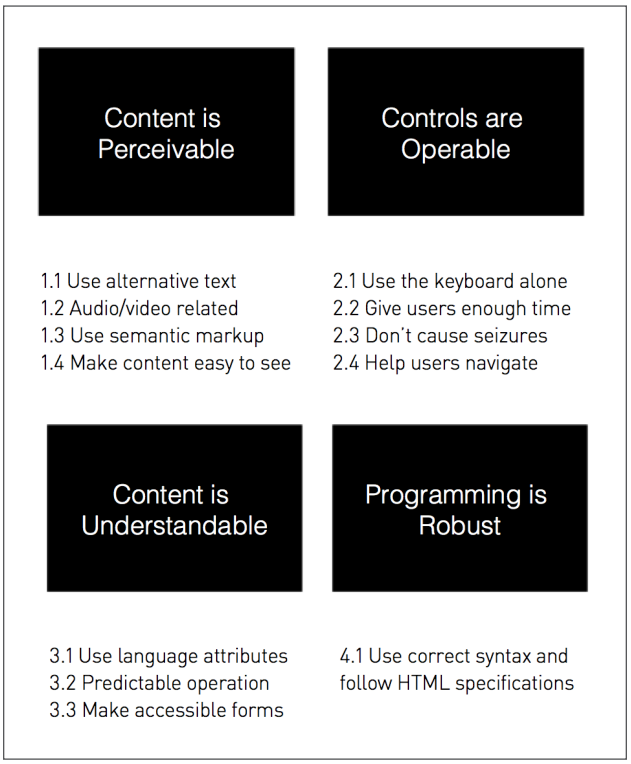


Figure 5

Starting at the Beginning

It's a common misconception that web accessibility can be ignored until development is almost complete, and then we can ask the developer to step in and fix all of the accessibility issues.

In reality, accessible digital experiences start in the design phase with the experience designer. The experience designer must take into consideration the linear keyboard user's experience.[14] Visual designers must design the experience in an accessible way. Copywriters also play a role in creating an accessible experience.

The Keyboard User's Experience

Keyboard navigation is used as a benchmark of accessibility compliance, as many assistive devices are based on the type of navigation that can be done with a keyboard. In general, navigating a website by keyboard differs in one critical way from navigating with a mouse pointer: With a mouse pointer, the user can click anywhere on the site and experience the content in the order they choose; whereas by keyboard, a user is forced to navigate the website in a linear way.

Consider a typical e-commerce product grid. An online shopper using their keyboard to navigate the page would have a long journey to tab through the page to finally get to the product grid. They would be taken through each element (e.g., link or form element) in a linear order, from the top left corner of the screen to the bottom right corner of the screen. In a typical page layout, tab focus would move through elements like the logo (top left), followed by utility links in the top right (e.g., "Choose a country" or "Log in"). Then followed by category navigation (often in a horizontal bar across the top of the screen), any links in the promotional banner, the left navigation, the product filters, and each item in the product grid.

The experience designer must consider this linear experience and offer way-finding cues that would assist a non-sighted keyboard user to orient themselves to the content, and provide helpful methods to navigate that content via the keyboard. For example, accessibility guidelines require that a "skip navigation" link be placed before the global navigation on the site. This link can be hidden and only appear on keyboard focus. This is an in-page link that allows a keyboard user to skip past the navigation menu links and land directly in the content area. You can see that this would be helpful for someone navigating with a keyboard.

The specifications for the user experience, which include consideration for the linear experience, must be documented in wireframes and visual design style guides. If accessibility is considered at these stages and documented for the developer, about 75% of the accessibility issues we typically see would not exist, and experiences would be designed in a way so that they made sense for keyboard users.

WCAG Level Compliance: A vs AA

WCAG level A compliance is the most basic level of compliance possible. Features included in level A are referred to as the "Must have" features. However, legal entities across the globe are insisting on level AA compliance, referred to as the "Should have" features. Here are some of the extra accessibility features needed to adhere to AA over A, and some of their impacts. (Note: This list doesn't include all of the level A criteria, just the additional criteria required for AA, over and above A.)

Minimum contrast requirement for text on a background There are contrast ratio rules that must be followed for AA that do not exist for A. I use the following web-based contrast checker, which provides pass/fail criteria based on font size: http://webaim.org/resources/contrastchecker/

Time-based media impacts "If A level means that you provide captions for prerecorded audio content in synchronized media (unless there's a clearly labeled media alternative for text), then AA means that you provide captions for all live audio content in synchronized media and that there's an audio description available for all prerecorded video content in synchronized media." [15]

Slightly longer front-end development time because developers must build interfaces in which text size can be doubled through the browser with the experience still remaining legible. This requires an additional unit test for front-end developers and for the quality assurance (QA) team. Sometimes it is not apparent how the text should appear as it grows and needs design input. This can only be tested in Firefox browser by zooming the text alone.

Possible production considerations if the client is relying on text embedded in images for any of their campaign banners: "If the same visual presentation can be made using text alone, an image is not used to present that text." (Web Content Accessibility Guidelines (WCAG) 2.0—1.4.5)

A visible focus indicator is needed for AA. (As you tab through the site, this shows you which element has focus.) This benefits from a designer's input and should have client sign-off on the look and feel of the focus indicator. (Web Content Accessibility Guidelines (WCAG) 2.0—2.4.7)

Different languages within a page need to be marked up as such with a "lang" attribute. This is a simple task easily performed by the web developer.

In summary, it takes slightly more care and effort both from a design and developer perspective to adhere to level AA as opposed to level A. Use this information to help inform your clients of the difference. Remember that even the W3C does not recommend trying to adhere to all of level AAA.[16] Last but not least, be aware that the trend is to support level AA from the onset, and this is what we recommend.

TEAM ROLES

Project Manager

Because the project manager runs the entire project, he or she is in a great position to ensure that the mechanisms that enable accessibility compliance are in place. The first and probably most important task is to lead the entire team through choosing the correct WCAG level for the project and clarifying the requirements each team member is responsible for. Many of the decisions made by design/development teams will be in collaboration with the client.

Designer

As mentioned above, accessible experiences start with design. Experience designers must consider the keyboard user's needs and account for these constraints in order to create a logical path through the experience. Visual designers must ensure that content is perceivable to all users (meets preset contrast rules).

Developer

Implementing the designer's direction for accessibility may depend on the developer's familiarity with and use of building and validating tools (see next section), and always on their awareness of the predetermined accessibility goals. A good rule of thumb for the developer is to always check all of their work with the keyboard alone to make sure it functions properly.

Copywriter

Copywriters have a critical role in designing an experience inclusively. Often they are required to think, and write, in an "alternative" way—one that bypasses generally accepted web terminology in favor of even clearer directions.

QA

The QA team, through involvement and awareness of WCAG levels and project specs from the beginning of the project, as well as by choice of methodology and periodic testing, can deliver an effective and valid, accessible site efficiently and on time.

PROCESS AND TASKS

Critical Steps

Think of these three steps as Beginning, Middle, and Final "Must-Dos" for the team:

Establish the level of compliance. Typically you should try to adhere to WCAG level AA; however, some organizations prefer a

mix of level A and AA guidelines. Level AAA compliance has not been recommended as a general policy for entire sites, because the W3C has noted that it may not be possible to satisfy all AAA requirements for some content.

Design, develop, and write to meet the guidelines. Each team creating content must understand the rules for meeting accessibility requirements and understand how to measure compliance. Ideally there should be an accessibility owner for each domain who can act as a decision point when there are questions.

Test at each project phase. Testing for accessibility compliance at each project phase before moving to the next phase enables the team to more efficiently fix issues and allows for more flexibility in the experience. Resulting defects should either be fixed or accepted before moving on to the next phase.

Common Inaccessible Design Paradigms

Avoid incorporating the following design paradigms into new experiences, because they explicitly break WCAG 2.0 guidelines:

Do not use placeholder text instead of labels for form fields. Studies show that not including a visible label that stays visible puts too much of a cognitive load on individuals to remember what information they need to provide.[17] This is covered in the WCAG level A guidelines. Guideline 1.1.1 states that form inputs must have associated text labels. Where visible labels are incompatible with the design, developers can add hidden labels that will be read by the screen reader. However, visible labels are best.

Do not design experiences that include motion that lasts longer than five seconds without including a way to stop the motion, e.g., animated carousels. WCAG level A guideline 2.2.2 states that automatically moving content that lasts longer than five seconds can be paused, stopped, or hidden by the user. Typically this is solved by not having interactions autoplay. In the case of carousels, they should not be animated. Instead, they should be controllable by the user via the keyboard or mouse.

You must explicitly warn users about changes in context, like content that opens in a new window. WCAG level A guideline 3.2.2 states that when a user interacts with a control, it must not result in a substantial change to the page that could confuse or disorient the user. Typically this issue is solved by including an icon next to a link to indicate that it opens in a new window.

Avoid overlaying text on background colors or imagery that do not provide enough contrast. WCAG level AA guideline 1.4.3 states that text must meet a specific contrast ratio. Typically this is solved through careful design. A client's brand colors must be carefully assessed in order to determine accessible color schemes for text on a background color. When text overlays images, the images should have a translucent overlay that contrasts with the text in the foreground, and text should be of large enough size.

Design Elements That Must Be Documented

During the design phase of a project, make sure to provide documentation to developers in the following five areas:

Heading Structure

Remember the last time you wrote an essay? You probably had a series of sections in your essay that each had a heading. Most likely you had a table of contents that listed all your headings. If someone read your table of contents, they would have had a good idea of what your essay was about.

This is what headings are for on a website. They should describe the page they are on. Assistive devices like screen readers allow users to navigate web pages by headings. This is one way that a non-sighted person can "scan" a webpage, by choosing to hear all the headings on the page. Listening to the headings allows them to assess if the page is useful for them. If the headings are not developed and written in a clear and proper way, they will not be helpful to this audience.

The experience designer or copywriter should decide the heading structure. The wireframe or content matrix should include an annotation that describes the heading structure. A developer should not decide the heading structure by herself (in absence of documentation).

Hidden Way-Finding Cues

Screen reader users benefit from additional way-finding cues to help them navigate a webpage such as "Bypass block links". This allows screen reader users to skip over repeated blocks of content. Without them, screen reader users would have to tab through repetitive blocks of content from page to page. They are most often used to skip over navigation menus. Other candidates for bypassing include filter menus or carousels with many panels.

Bypass block links can be hidden by default, but should appear on keyboard focus. They should allow the user to skip past the content block to the content immediately after. It's important to test these with your keyboard once they are implemented. Make sure that you can tab once into the content area, and tab again into the next interactive element in the content area.

Focus Order Information

Focus order is an important concept for keyboard accessibility. It refers to the order that elements on the page receive keyboard focus. The usual focus order for the Western world is from top to bottom, left to right, the same way we read.

For keyboard users, a common way to access a website is to move through the content by pressing the tab key. The tab key moves

the focus state to links or form elements. Screen reader users can have their device begin reading the page at any point.

Usually experiences benefit from implementing a default focus order. There are instances where the focus order should be changed. This is best determined by an experience designer.

The following flow illustrates a situation where the default tabbing order should be overridden:

- User clicks on a link to log in to a website, and the log-in link leads to a different page.
- By default, the first focused area on the page would probably be in the top left-most link or form field. But in this instance, the user is definitely there to log in. It makes sense here to put the focus state on the first field in the sign-in form.

Again, this cannot be decided by a developer in absence of documentation. It must be annotated in the wireframe. The user experience designer should annotate the non- standard focus order to completion, indicating where natural focus order should take over.

Visible Focus State Design

Visible focus state is the visual indicator that an element has focus. It is common for designers to create a hover state for mouse users. Usually the focus state should match the hover state. Try it out on any website: Press your tab key now and try to see what the current focused element is on the site. If implemented accessibly, you will see that the navigation links along the top of the site get underlines, or change color, or have an outline. This is so keyboard users can see which element has focus. If you can't tell which element has focus, you will have a taste of the frustration experienced regularly by keyboard users trying to navigate an experience.

For developers to handle the focus state in their implementations, it must be defined in the style guide. Each browser has its own default focus state. Either allow this to be used, or have the designer create a new one that matches your client's branding.

Many clients do not understand the value of the visible focus state. When it is described in the style guide, the client has an opportunity to see it up front and ask questions about it. Note: Some designers or clients have a concern that the visible focus state looks ugly for non-keyboard users. Be aware that it can be developed in such a way that it only appears for keyboard users who need it, not for mouse users.

Clear Link Labels

For users who navigate with a screen reader, many of them will only hear link labels. They will not have any context on surrounding information. That's why it is important to make sure the link itself (or the form field label) is meaningful.

Do this: "Learn more about our services"
Don't do this: "Learn more"

Do this: "Edit my account settings"
Don't do this: "Edit"

Sometimes the context is clear to a sighted user based on surrounding content. In this case, indicate in the wireframe that the developer should hide the additional content from sighted users. This way, a sighted user who can gain context from the surrounding content will see "Learn more." A non-sighted user will hear, "Learn more about our services."

Usable Forms and Error Messaging

I often ask designers if they plan to enter their form design into the coveted Form Design Awards. Usually they stare back at me blankly. Sometimes they perk up and ask for more information about these awards.

There are no Form Design Awards. Isn't it great? We can put our aesthetic goals aside and focus on making the form usable. Trust me: forms are meant to be filled out. They don't need to be flashy. It's okay if they aren't ultra clean and sleek. However, you may find that an accessible form looks great. Here are some form design best practices that also make the form more accessible:

- Place the label above the form field, not beside it.
- Put a "required field" indicator inside the field's label. It's also helpful to put "(optional)" within any optional field's labels.
- Do not replace proper form labels with placeholder text. Users of all ages and abilities complain about this. Placeholder text disappears when a user clicks into the field and begins to type. It can be hard to remember what content the field needed (e.g., email address vs username).
- If fields do not have a visible label (e.g., search fields), provide the annotation for a hidden label in the wireframes to ensure the developer creates it in the code. Indicate the label text. All form fields must have labels.

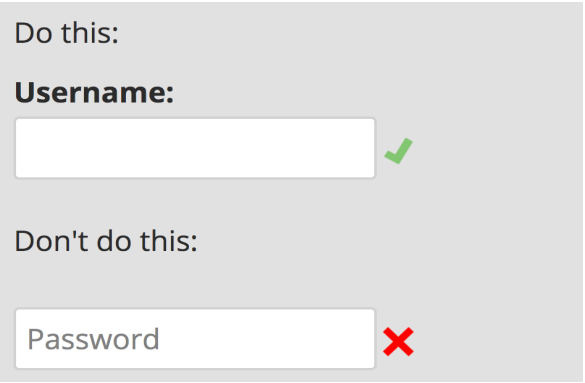


Figure 6

Screen readers automatically read out form field labels when the field has focus. The form error messaging flow should work in the following way:

1. User fills out a field the wrong way.
2. User attempts to submit form.
3. The first form field that has an error message should automatically gain keyboard focus.
4. The form field error message should be programmatically appended to the field's label.

In this way, the screen reader will read out the form field followed by the error message text. Example: "Email address. Email address is a required field." The user can then fix this issue, then tab through the rest of the form and hear any other error messages along the way. It is helpful to describe this behavior in the wireframes, and to design error messages to naturally occur close to the field's label.

Other Design Issues

Audio and video files need captions and transcripts. When designing or reskinning audio and video playing components, be sure to provide a design for transcript links and a button to toggle captions on and off. More details about accessible video and audio will be found in the **Addendum.**

Some popular components are more difficult to design in an accessible way due to their complexity. But because they are frequently used, they should be discussed, specifically carousel or slider components, calendars, and modal windows/lightboxes. You'll find detailed coverage in the **Addendum.**

Development Issues

Here are the top five most costly development issues, meaning that they take the most time and effort to fix later if not done properly in the first place:

Text Alternatives

Guideline 1.1: Text Alternatives: Provide text alternatives for any non-text content. All non-text content (like images or video) needs a text alternative that describes the non-text content. Images on a website need alternative text described in an alt attribute. It looks like this in the code:

**

Some general rules for creating alt attributes:

- All images need alt attributes. They should be provided to the developer. If none are provided, the alt attribute is alt="".
- Decorative images need empty alt attributes.
- All other imagery needs a descriptive alt attribute (should be provided to the developer).
- For the most part, don't bother using title attributes.
- You can't place an alt attribute on a CSS background image. Images should be marked up with an <img> tag.

If you do not use an alt attribute on an image, the assistive device will read the path to the image. This is annoying for non-sighted users, and doesn't help them understand the image content.

This guideline is easy to test with automated tools like the WebAIM Chrome browser extension. Install this in your Chrome browser. Test each page to see if there are alt attributes missing on your site's imagery by clicking on the button in the toolbar. Note that alt attributes should be written with care by a copywriter.

Semantic Content

Guideline 1.3: Create content that can be presented in different ways without losing information or structure.

- Use semantic markup to designate headings (<h1>), lists (<ul>, <ol>, and <dl>), emphasized or special text (, <code>, <abbr>, <blockquote>, for example), etc.
- Use tables for tabular data. Where necessary, associate data cells with their headers using scope attributes (e.g., *<th scope="row">Heading</th>*). All data tables must have captions. Often the heading for the table can be used as the caption. If no caption or heading is provided, please ask the user experience designer what the heading should be. Read more about accessible data tables on the WebAIM site: http://webaim.org/techniques/tables/data.
- Note for the integration team: Table captions, IDs, and headers should be exposed to a CMS authoring tool.

DEVELOPMENT TIPS

Keep heading CSS styles separate from heading structure in html. For example, create separate heading level CSS classes, e.g., \ **.primaryHeading**, **.secondaryHeading**, etc. with properties defined in CSS.

In this way, you can apply any style to any heading level, and place headings on the page in the expected order.

Heading Structure

Assistive devices have the ability to navigate web pages by structure. A user visiting the site using such a device can navigate the site in various ways. They may choose to navigate through a list of the headings to understand what content is available on a page.

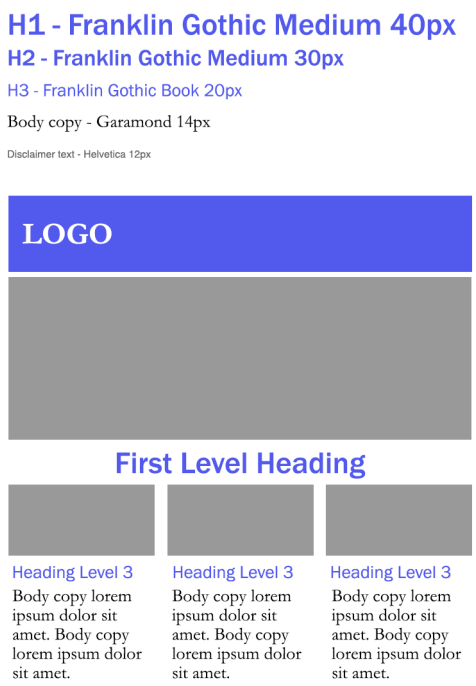


Figure 7

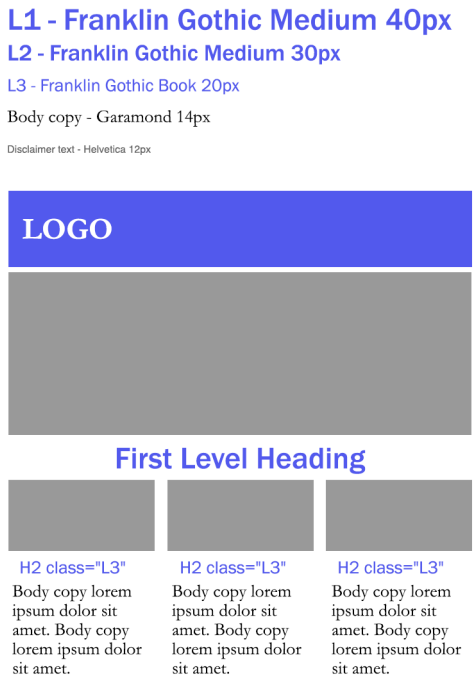


Figure 8

Elements styled to look like headings will not appear as headings for these users. Heading tags must be used for users to benefit from this feature of their assistive device.

It can be tempting to assign styles to heading levels based on the design style guide. When you do this, sometimes you will need to skip heading levels on a page to meet the design requirements. This breaks accessibility guidelines. See **Figure 7**, which illustrates a style guide where the styles are tied to the heading level, then the page design requires that headings appear out of order or missing levels. A better strategy would be to apply heading styles as a class to heading elements. In this way, proper heading structure can be observed and can still match the design.

This is demonstrated in **Figure 8**, where the developer applies styles to a class, and can then create a page design with heading styles in the correct order while still observing proper heading levels.

> **Do this:** *<h1 class="L1">Level 1 heading</h1>*
> **Don't do this:** * Level 1 heading *

Keyboard Operability

Guideline 2.1: Make all functionality available from a keyboard.This is another one that is easy to test during development. Developers are already tasked with testing their work across various browsers and devices. Developers should simply tab through all functionality with their keyboard as they are developing.

The importance of trying functionality with the keyboard alone as part of regular unit testing cannot be overstated. I encourage developers reading this to try navigating one of the pages you've built in the past with your keyboard to see if you are able to achieve the site's objectives. Try navigating common keyboard problem areas like modal windows or flyout menus. Items that show or hide content when the user clicks on them with the mouse pointer need also to work when the user tabs through with their keyboard. Content that is hidden because it is inactive (e.g., contents of a hidden tab) should not be read out to screen readers, or be focusable in its hidden state. These are complex challenges that are rarely addressed unless it is done purposefully.

Accessible Forms

So much depends on form design. Forms are not meant to be fancy. A form's main purpose is to function as intended—to capture and submit information. As developers, you should be able to contribute feedback to a design. Encourage designers to create simple forms that observe the following best practices:

- The full form label is included with the field (hint: do not rely on placeholder text)
- The form label is ideally above the form field. This way it will fit easily into a narrow area (e.g., a mobile device).

- Required fields are indicated with an asterisk.
- Browser default controls are the easiest to ensure accessibility compliance.

Developers need to ensure that form fields are programmatically associated with their labels. This is a simple exercise that is too often missed. Test it by tabbing through the fields of your form with a screen reader enabled. Screen readers will not read out form field names if the fields are not associated with their labels. The impact will be that the user will not know what information they should enter in the field.

Do this: *<label for="name">Name:</label>*
<input id="name" type="text" name="textfield">

Accessible Form Error Handling

3.2.2 On Input: When the user submits a form without filling out all fields, they will get an error. Fixing this issue requires javascript functionality that can be time-consuming to add later. There is also a design dependency that can be difficult to amend if the design was not done in an accessible way.

When a user fills out a form the wrong way and attempts to submit it, errors preventing form submission are commonly bypassed by the screen reader. Often the error will appear in red text, but non-sighted users will not see it. The user gets no feedback about what they did wrong so cannot fix the issue. It slams the door in the face of a user of assistive devices.

This can be easily fixed during the design phase. The error message for each field should appear next to or immediately below that field's label in the design. In the code, the error message should be placed within the label itself. When an error is generated, focus should be programmatically placed on the first field that has an error. The screen reader will automatically read out the field's label followed by the error message.

Using this method, the user's experience will improve. Let's say the user forgot to provide their email address and tries to submit the form. The keyboard focus will immediately be set on the email field, and the error message will be appended to the label. The user will hear "Email address. Please provide your email address." The email address field will have focus, the user will be able to immediately start typing their email address and can tab through the rest of the form and click "submit" again. See an accessible form demo: https://www.w3.org/WAI/demos/bad/Overview.html

Copywriting

Copywriters have a critical role in designing an experience inclusively. Their contributions include:

- Writing consistent, properly structured copy.
- Crafting clear labels and instructions for interactions and wayfinding cues that don't rely on visual characteristics.
- Providing descriptive, appropriate text alternatives for non-text content.

Structuring Copy

One of the ways a non-sighted user can navigate an experience is by hearing a list of the page's headings. This dialogue shown below is from the Voiceover screen reader, which can be set up to list the headings on the page.

As noted previously, headings are intended to describe the content outline of a page. A screen reader is able to determine the headings on the page because of how a developer tags them within the HTML code. For example, in **Figure 9**, a developer would have created the following heading:

<h1>Women</h1>

The "h" stands for "heading," and the number indicates the heading level, where a lower number represents a higher heading level. An <h1> is the highest level heading on a page. It is of critical importance that a copywriter or content strategist determine the heading structure for a page, and not the developer writing the code. A developer is not the best person to determine the heading structure. Specifications on how copy should be structured should be provided to the developer before development begins. This can be done as part of the content matrix or the wireframe.

Figure 9

Link Purpose in Context

We're all familiar with the paradigm in which multiple links in an experience have common link labels, although they point to different places. For example, "Learn more" or "Shop now."

It's confusing to have multiple links with the same link label that point to different places. There are three ways to deal with this:

- More descriptive link labels can be written. However, often the design team will prefer that an experience have the same look and feel for a repeated link label or button.

- In many cases, the component will include a title, description, and "Learn more" link. When this is the case, another option is to remove the common link label altogether and make the title the link instead. This is often done for articles.

- Finally, a copywriter can work with a developer to devise a strategy where the title text is appended to the common link label programmatically. In this case, a sighted user would see the common link label, but a non-sighted user would hear the common link label plus the title together for more context.

Clear Instructions

Instructions must be written clearly with a non-sighted user in mind. The order of the instructions matters, and they must not rely on color or other visual elements to convey content. For example, the instruction "Click on the green button to continue" would not be helpful to an individual who cannot perceive the color green.

Labels must be provided for all form elements and instructions on which fields are required, or specific formatting requirements should be communicated within the field's <label> tag. This can be done in multiple ways:

1. The label can include an asterisk. Historically, an asterisk has been used to indicate that a form field is required. It is read by the screen reader as "star," but users understand what this symbol means.

2. The label can include the word "required."

3. Fields that are not required can include the word "optional." Avoid using this strategy on its own. Required fields must always be indicated as well.

Copywriters are encouraged to try to use the experience after it is developed with their keyboard and screen reader to see if it makes sense. Typically only the desktop or mobile versions of an experience are tested, but this allows a lot of room for error if nobody is testing with the keyboard.

Form error messaging is one area that is commonly overlooked. All form fields should have carefully crafted, descriptive labels. Error messages should be as descriptive as possible and still be concise.

Way-Finding Cues

There are portions of online experiences that must be written although a sighted user will never see them. They will only be heard by someone using a screen reader. These include:

- **Bypass block links:** In-page links that allow keyboard users to skip over large blocks of content. A typical example of this is a "skip navigation" link, although they can be placed anywhere a keyboard user may prefer the option to skip over content.

- **Table captions:** Using the <caption> element will allow a screen reader user to scan all of the tables on a page by name. Typically whatever visible name is used for the table can be reused for the <caption> element.

- **Page titles:** It is helpful (and required) to provide a descriptive page title for every page. It is best for these to be written in a consistent format. Try listening to your page title with the screen reader (it is announced as the page loads). Some elements commonly used in titles, like the pipe character, may be read orally in a way you don't expect.

Non-Text Content

All prerecorded audio and video requires a descriptive text transcript and a text or audio description. The audio description is intended to provide information about actions, characters, scene changes, and other important contextual elements that are not spoken in the main soundtrack.[18] If all of the information in the video is already provided from the audio track, no audio description is required.

Other Alternate Text

Text alternatives must be supplied for all non-text content. This includes time-based media as outlined above, as well as images, form image buttons, image maps, etc. Even decorative images like spacer graphics, rounded corners, or other images that do not convey content must have alternate text. Alternate text is provided in the HTML code via an alt attribute. Please see the Alternative Text Decision Tree, **Figure 10,** opposite page. A copywriter should ideally decide what the content for that alt attribute should be, based on the following rules:

- Decorative images which do not convey content should have an empty alt attribute. This includes the above examples.

- Images that have a caption, or a description immediately next to them in the body of text, take an empty alt attribute.

- Images that have text embedded within them should have an alt attribute that matches the text embedded in the graphic.

- Images that are content should be described adequately.

- Images that are links should have an alt attribute that describes the link purpose.

Quality Assurance

QA teams need not double their work in order to test and validate an accessible-compliant site. Planning from the start, with informed choices about what to test, when to test, and how to test, will improve workflow and results. An argument for categorizing

WCAG guidelines by discipline, along with testing guidelines and recommendations, will be found in the **Addendum**. In brief, the most efficient accessibility QA performed at the end-of-development phase will focus on code-related defects alone.

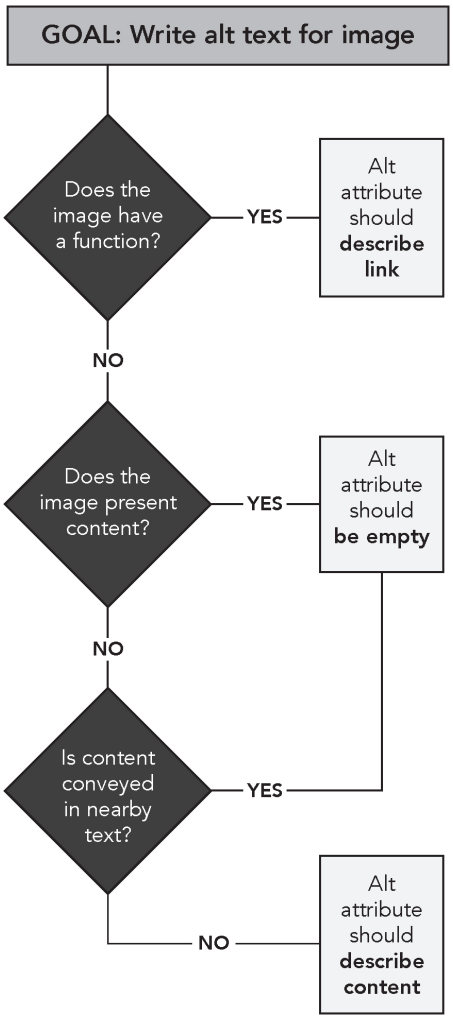


Figure 10

Project Management

A project manager can help by asking these questions at various project phases:

Is the client unsure what we mean about accessibility compliance? The project manager should have an idea of the client's expectations and understanding about accessibility and arrange for any necessary conversations to occur between the accessibility lead and the client.

Is accessibility compliance accounted for in the estimate?

- If the team hasn't designed an accessible site yet, they may need training at the beginning of a project. Do this first! Give each discipline at least half a day to ramp up.
- Internal reviews for accessibility should be added to the experience design and visual design phases. As designers become more familiar with accessibility principles, these reviews will have less feedback.
- The development phase should account for the extra unit testing (by keyboard) of work. I have added 10% of core effort to a project for developers who are new to accessibility, but this number decreases drastically as developers become familiar, and becomes insignificant for seasoned developers.
- Is an accessibility lead (or leads) accounted for in the project estimate?
- Have tools been discussed and selected?
- Has a level of compliance been agreed upon?

Are accessibility reviews happening during each project phase? The project manager should ensure that the required reviews are, in fact, taking place.

When the project manager looks at the site, they should try accessing the site's functionality with the keyboard.

Governance and Maintenance

Accessibility compliance is a journey, not a project. The following measures should be observed in order to ensure that accessible experiences remain accessible:

Review all site updates for accessibility compliance in every phase (wireframe, design, development, and production code).

Name accessibility owners for each domain.

Be aware of introducing content issues (e.g., "Learn more" links, images with no alt attribute).

Conduct periodic (quarterly) high-level accessibility audits on the entire site:

- "Happy path"—path to register, do transactions, etc.
- Form completion and error handling
- Data table reading
- Heading structure
- Keyboard navigation (especially navigation)

Ideally, have visually challenged users audit the site as part of the regular quality assurance team.

CONTENTS

CAROUSEL OR SLIDER COMPONENTS

Some people call them carousels; some call them sliders. A carousel or slider is a content display paradigm in which a list of related content items are presented in panels as a horizontal slide show. The slide show is usually navigated by "next" and "previous" arrow keys. Usually each item has a graphic, some text, and an associated link. Carousels can have one or multiple panels in the viewable area. **Figure A** shows a carousel that has three content panels, side arrows to navigate, and dot indicators below to show pagination.

Figure A

The carousel paradigm for displaying content exists for one reason: because groups fight for that coveted front and center position on a website home page. A carousel is one way to guarantee that each group's content has an opportunity to have a turn in that spot. Studies show that users rarely interact with them (Runyon, 2013). However, they are also one of the most popular paradigms web designers use to display content. The groups I mentioned above like them a lot, for the reason I mentioned above. So, if we are going to persist in using carousels, let's all learn to design them in an accessible way. We can consider three general personas who could use them:

- Sighted users who use a mouse-pointing device
- Non-sighted users who use a screen reader and keyboard to navigate
- Sighted users who use a keyboard to navigate

There are a variety of other users we could consider, but let's start with these three to create a baseline for accessibility. Let's look at **Figure A** again, showing a carousel with three panels in the visible area. Assume there are two sets of three panels, so six panels in all.

For Sighted Users Who Use a Mouse Pointer

Sighted users perceive carousel content as a series of panels usually navigated via arrow buttons. The main carousel accessibility consideration for sighted users is to not make it auto-play. If you must make the carousel auto-play, then you need to provide a pause or stop button.

Employing navigation dots below the carousel is another great visual cue for sighted users. The number of dots represents the number of panel sets available to click through. An active state on the dot tells the user which panel they are currently viewing. The dots should be clickable for a person using a mouse, enabling them to skip to the last panel.

"Previous" and "next" arrows should have disabled states. The "previous" arrow should be disabled when the carousel is showing the first panel. The "next" arrow should be disabled when the carousel is showing the last panel. Avoid having carousels that "wrap around," allowing the user to cycle through the carousel repeatedly, as these are difficult to make accessible for keyboard users.

For People Who Use a Keyboard to Navigate

Consider that non-sighted people do not need to know that you have collected content items for display in a carousel format. For a non-sighted person, the important interaction is that they can tab through each item. As they tab, relevant information should be included in the focused area so that they have context on each piece of content.

As a person tabs through the site, they should naturally tab into panel 1 of the carousel (without needing to know it is a carousel panel). Ideally, the panel 1 link should wrap around all panel text. In that case, the screen reader will read all text. In this example: Panel 1 title. Here is a panel description that lets someone know what this panel is about. Panel 1 call to action. Note that for a non-sighted person, hearing only the link label may not be enough to give them context.

A sighted person navigating by keyboard will need a visible focus indicator. This is a box that appears around content that has keyboard focus. Designers can work with developers to control the appearance of the visible focus indicator.

If the person presses "return" or "enter" on their keyboard while the panel has focus, they will be navigated to the link on that panel. If they press the tab key, focus will be moved to the next panel. When focus is on the last panel in the set, tabbing again should bring the next set of panels into view. The action should occur as though the person had clicked on the "next" arrow.

Once the person has tabbed through all the panel sets, focus should move to the next interactive element that follows the carousel. Note that for a keyboard user, the carousel cannot wrap. We do not want to trap the keyboard user in the carousel. They need to be able to tab past it.

Arrow Buttons Don't Work for Keyboard Users

Making the arrow keys work with the keyboard is a common mistake. The problem is that for a keyboard user, if the focus is on the arrow keys, it is not on the panel content.

A sighted keyboard user could potentially use the "next" arrow to cycle through the panels. But consider how the keyboard focus order would work in that case:

1. Focus on panel 1 content.
2. Focus on panel 2 content.
3. Focus on panel 3 content.
4. Focus on "next" arrow.
5. Pressing "return" or "enter" moves the carousel to reveal the next set of panels.
6. The user would need to "back-tab" off of the "next" arrow back onto the panel they are interested in if they want to follow the link. This is awkward. Note that this functionality would not be accessible for a non-sighted user. The non-sighted user would hear "next," the panel-set would move, and they would have no idea they needed to back-tab to get, say, the 4th panel content into focus.

Bypassing the Carousel

The sighted person is not forced to use the carousel at all. They can click around the page wherever they want. Yet keyboard users are forced to navigate through the carousel in its entirety.

It might not seem like a big deal for a simple carousel like this one. But what about a larger carousel with lots of panels? Luckily there is a best practice to allow screen readers to bypass large blocks of content. In the case of a large carousel, a hidden "skip" link could appear on keyboard focus. For example, consider a product recommendations carousel. The link before the carousel could read "Skip product recommendation carousel."

DESIGNING CALENDARS

Date-picking functionality is notoriously inaccessible on most websites. With this functionality, collaboration between the technology and design teams is most critical. It may make sense to sit down with the developer and look at different calendar plug-ins to see if you can find an accessible one that meets your needs. However, at the time of writing, I have not seen an out-of-the box calendar plug-in that is accessible. The main considerations for an accessible calendar are the following:

Users are able to tab through the calendar dates with their keyboard. They should be able to tab into the calendar and back out of the calendar. They should be notified that they are tabbing into a calendar (for example, with a "choose date from calendar" link).

Dates are read out in their entirety by the screen reader on focus. For example: The screen reader should NOT phonetically say "Tue Jun 5," it should say "Tuesday, June 5." Note that it is fine for a sighted person to see common abbreviations like "Tue. Jun. 5"; however, they are not easy to understand when read phonetically by the screen reader, so the developer should include a full text version as hidden content.

Users are able to select a date by hitting the enter key (equivalent to a mouse click). This should populate the date field with the selected date. If the date is incorrect, it should be easy for the user to return to the calendar and correct it.

There should be a simple text field where a user can type the date without having to use the interactive calendar in cases where calendar plug-ins don't function as described above. In these cases, it is very important that the date format be flexible, or be described in the field's label (e.g. "Date, format dd/mm/yyyy").

As with all keyboard functionality, these specific directions should be outlined clearly in the same specifications document that outlines mouse click behavior, hover behavior, or tapping behavior. In this way, a developer reading the specifications will be equipped to understand the accessibility requirements and develop the functionality accordingly.

MODAL WINDOWS OR LIGHTBOXES

There are three main considerations when creating accessible modal windows or lightboxes:

- Keyboard users should be able to open the modal window content by accessing a link, which may or may not include the notification that the content will open in a modal window.
- The modal must have a close button. Once the modal window opens, keyboard focus should be on the close button. If the close button is an icon of an "X," ensure that the actual text label of the link reads "close" or "dismiss". Make sure that the link text to open the modal and the text to close the modal give a non-sighted user context to what is happening: "Find out more about our services" on the link to open, and "Dismiss services information" on the link to close.
- Keyboard focus should be restricted within the modal window while it remains open. To close the modal window, the user can access the close button, or use the escape key.

NOTES ON AUDIO AND VIDEO

Audio Description Requirement

Many people are surprised to learn about the audio description requirement. An audio description includes a description of everything that is going on in the video that is not spoken. For example, it can describe the position of the speaker, the scenery in view, or body language that is not conveyed by the transcript. These descriptions allow a blind individual to better understand the context of the video.

Accessible Video Requirements

- Provide controls (e.g., stop, pause, play, and volume control) for the video and ensure that such controls are operable with a keyboard
- Follow contrast rules both visually and audibly (e.g., for text on a background within multimedia as well as between foreground and background noise)
- Obey rules designed to prevent seizures (no more than three flashes per second)
- Include synchronized captions
- Provide a descriptive text transcript or an audio description synchronized to the video via a link
- Provide a text or audio description when the video has no audio track

Notes on Using YouTube for Videos

YouTube allows for the creation of accessible videos; however, there are a number of obstacles that we have identified (as of the date of this White Paper):

- We recommend embedding YouTube videos on your own website, rather than directing users to the YouTube site in general. YouTube's website may not be entirely accessible.
- When embedding videos on your own website, we recommend using the HTML5 version (as opposed to the Flash version) by having your developer add "?html5=1" to the embed URL provided by YouTube. This worked well for most browsers, but still had issues in Internet Explorer and Firefox. As a result, we suggest: 1) adding disclaimer text for those browsers near the video, indicating that full keyboard accessibility is available on Chrome as well as Android and iOS devices; 2) telling Firefox users that they should add the YouTube ALL HTML5 add-on to their browser; and 3) telling Internet Explorer users that Flash must be disabled in their browser.
- YouTube can automatically create captions, but there are errors inherent in this process. It will be necessary to have a human go through and validate YouTube's suggested captions using YouTube's caption editor tool. YouTube's automatically generated captions frequently include spelling mistakes, words that should be shifted to a different caption as well as throw away words, such as "um," which can be removed.

Captioning in general is an art to be mastered — perfecting the timing, describing the speaker, and remembering to include ambient sounds like music can be a challenge.

GUIDELINES BY DISCIPLINE

The WCAG guidelines are already categorized by level and according to the four principles. We have seen above that it is also helpful to categorize them by discipline. In this way, we can narrow the focus area of the disciplines and make it less overwhelming to understand and meet the guidelines that pertain to each. It also makes accessibility testing easier when we test accessibility compliance by discipline at each project phase, leaving only code-level defects at the end for the quality assurance phase.

Figure B

Figure C

Take the example of creating and testing an experience that contains video content. It is fairly common in a wireframe to see a placeholder for video content like **Figure B**, in which a video plays when a user clicks on it. The keyboard functionality for the video player is often not described in the wireframe, and the video plugin itself may not be selected up front, or may be thrust upon the creative team due to client partnerships with video vendors.

If accessibility aspects of the wireframe are not reviewed during the wireframe phase, this video player wireframe element will go through the design phase, and finally to the development phase, and may contain embedded accessibility issues that a developer will not be able to fix on her own.

Consider **Figure C**. Imagine a quality assurance team finally listing all of the accessibility issues with this video player at the end of the project. The issues could include:

1.2.1 Prerecorded video only; no link to a transcript All videos need a transcript. A transcript would be accessed by a link, but there is no link present in the experience. How should the link be styled, and where should the link be placed? A developer alone is unable to fix this issue, because these are questions for a designer. However, at the end of a project, all of the designers may have moved on to other projects.

2.1.1 Keyboard The wireframe did not outline how the video player should work with a keyboard, so the developer might have used a video player plugin that doesn't work with the keyboard. Video plugins are all different, and the experience designer should discuss plugin options with the developer to be able to detail how the video player will work. Some video player plugins are not accessible, and may be the solution that our clients want to use. In this case, the experience designer should alert the client that their preferred video player is not accessible so that the issue can be dealt with (or accepted) up front. A developer cannot "just fix" this issue, either. An experience designer would need to help select a plugin up front, or to help adjust an existing plugin's functionality to make it work in a usable way. In the situation above, the experience designer may have left the project by the time quality assurance phase is in full swing. It is inefficient to call them back at the end; the issue should be dealt with during the experience phase itself.

1.4.3 Contrast minimum An interesting point about video content is that it needs to adhere to the same contrast requirements as the overall web page. This means that captions or other text in the video need to have an appropriate contrast against the background to make them legible. When we develop videos for our clients (or when our clients provide us with video content), we should take this requirement into account in a timely manner, and not wait until the quality assurance phase to fix this issue.

The crux of this issue is that there are guidelines that pertain to certain disciplines, and these guidelines should be reviewed and approved during that discipline's project phase for maximum efficiency. If all of the experience design guidelines are reviewed in wireframe phase, all of the design guidelines are reviewed in the design phase, and all of the copywriting is reviewed separately, then the quality assurance team will be testing only that the design matches specifications, and it does not have to do accessibility-specific testing of the design.

QA: AUTOMATED AND MANUAL TESTING

There is no single tool available to quality assurance teams that covers all aspects of accessibility testing. A certain amount of manual testing is always required. Compare automated and manual testing with this example, using **Guideline 1.1.1, Non-Text Content**. This is the guideline that stipulates that all non-text content (for example, an image) needs to have a text alternative.

GUIDELINE 1.1.1., NON-TEXT CONTENT	
Automated Testing	**Manual Testing**
Finds issues that do not require human judgment to be detected Tool detects the presence or absence of an alt attribute within an **<img>** tag in the code. If tool indicates a missing **alt** attribute, it generates an error.	Finds issues that require human judgment to be detected or resolved If tool detects **alt** attribute, it will not generate an error, but there may still be issues with alt attribute content. Human judgment resolves further questions, such as: • Should the alt attribute be null or populated? • What constitutes a descriptive alt attribute?

Page-Level Automated Testing

There are a number of page-level automated testing tools available. These include but are not limited to the WAVE toolbar (by

WebAIM), AChecker (by ATutor), aXe (by Deque). Typically these tools offer the following options for validation:

- Direct input: Copy and paste the page code into a text field.
- URL: Place the URL of the page to be tested into a text field.
- Inline with a toolbar: Requires downloading an extension that places a toolbar within your browser.

Other tools that help validate an experience's accessibility at a page level include the W3C Validator and developer toolbars. These allow an auditor to validate code syntax (W3C Validator), turn off images, and highlight headings among other things (developer toolbar). Page-level automated testing works well for smaller audits, and for use during development.

Site-Wide Automated Testing

Site-wide automated tools are able, as the name suggests, to audit an entire site, instead of just one page at a time. As such, they are best for larger audits, and they are good tools for our clients to invest in if they have a large experience to audit and maintain. Site-wide audit tools always have an associated licensing fee to set up, so usually require persuading our clients to adopt them. There are a variety of considerations when selecting a site-wide audit tool.

User-friendly: All tools are different, and some may be easier to use than others.

High-quality results: Tools generate reports. The look and feel as well as the sorting and filtering features of these reports make some better than others, depending on what our clients need.

Can test the DOM: This feature relates to a tool's ability to test the actual markup on the page, even markup that was generated dynamically through client-side scripting. Tools should indicate in their feature section that they can test the DOM, which stands for Document Object Model.

Can spider the site: This is really a critical feature of a site-wide tool. Starting at a single URL, the tool should be able to read through the links on that page and be able to define a site map of all of the pages on the site, and ensure that all of them are tested and none are left out.

Integration with QA tools: Some site-wide audit tools can integrate with quality assurance tools like JIRA. Depending which tool your client uses, this may be considered a benefit.

Manual test guidance: Some tools, while conducting automated tests, may be programmed to alert auditors of features present on the site that typically require additional manual testing. For example, if the tool sees a <form> tag, it may provide as part of the report the guidance that the auditor should manually try to use a form on a given page to supplement the automated test. This can be helpful.

Accessibility level supported: Some tools are only configured to test level A criteria, while others are configured to test AA or even some AAA criteria. This is an important thing to know up front.

Manual Testing

Testing an experience manually is the best way to assess whether or not it is accessible, and it is easier than you think. The main manual test is simply attempting to use the experience with your keyboard alone, without ever touching your mouse.

This is necessary because using the keyboard mimics the functionality that many assistive devices are limited to. For example, some people who are completely paralyzed may use a device that requires them to blow a puff of air into a straw in order to change focus from one element to another—mimicking the functionality of the keyboard's tab key. So when we test a site with our keyboard, we are actually helping to ensure it will work in a variety of assistive devices.

Note that the WCAG guidelines are device-agnostic, meaning that there is no particular device (other than some kind of computer) in which the site needs to work. The only criteria for functioning is that it must work with the keyboard alone (guideline 2.1.1).

The Screen Reader Experience

The common use case we consider when testing an experience for accessibility compliance is the screen reader user. The screen reader user uses a standard web browser with an application that reads the web browser contents to the user. Users often navigate the experience with their keyboard tab or arrow keys. It is helpful to compare how a screen reader user accesses an experience compared to a mouse user.

A mouse user is probably sighted, since they are able to track their mouse cursor with their eyes. Their eyesight allows them to scan a page quickly, immediately getting visual context on what that page is for. For example, sighted users can look at all of the components of the view in **Figure D** and understand them without any additional narrative. They can get a lot of context by just scanning. Scanning is digital. Our eyes can jump from any point of this view to any other point in any order.

But when we design interfaces for keyboard-only users, we have to consider that their experience is analog. They are forced through

an experience from the top of the page to the bottom in a linear way. Guiding a keyboard user through the experience in **Figure E** from the beginning to the end in a way that makes sense is an experience designer's challenge.

For non-sighted keyboard users, the experience is often like the view in **Figure F**—obscured by poorly labeled interactive elements and headings that are marked up the wrong way. These are the issues that we want to uncover through our manual testing efforts.

Figure D

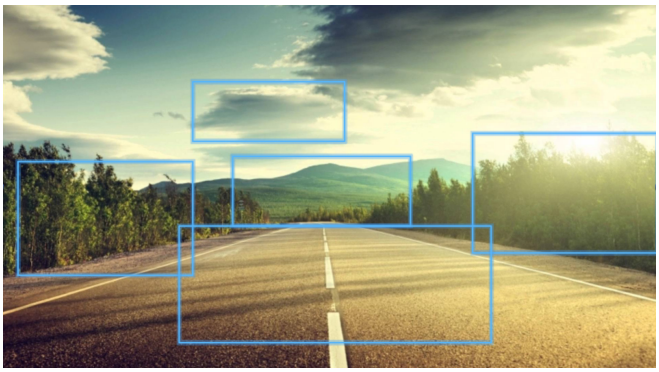


Figure E

Figure F

Testing with the Keyboard Alone

To test with the keyboard alone, navigate to a web page. Use your tab key to move you through the site. Using your tab key will allow you to jump between interactive elements, such as links or form fields. Things to note:

Can you tell which element on the page has focus? The element that is in focus (the element your cursor is on) should have some kind of focus indicator, like a box around it, or an underline.

Can you access all of the interactive elements on the page? For example, the page may have a set of tabs on it, that if you clicked with your mouse, would reveal additional content on the page. You should be able to access the content on the tabs with your keyboard alone. Make sure you test all of the interactions on the page with your keyboard.

Can you access the global navigation of the website? The global navigation is one of the most important components on a website, and also one of the most complex. Make sure that elements such as flyout menus are accessible with the keyboard alone.

Can you submit forms on the website? Make sure that you can submit the forms with your keyboard alone. Purposely leave information out of the form and try to submit it. Is helpful error messaging information provided?

Pay special attention to carousels and calendars, they are some of the most difficult components to make accessible.

Change to a different breakpoint and do the same tests.

Testing with Keyboard and Screen Reader

Try to do all of the same tests with your screen reader turned on, *not* looking at the interface when you test it. This is critical! Many times when we test with the keyboard and a screen reader, we say that the experience is accessible, but we are actually compensating for the bad experience with our sight. For example, we may "back tab" (use tab key plus shift key) to access content that a non-sighted user would not know was there.

Be mindful and employ good faith. Do not cheat by looking, really try to put yourself in the shoes of someone who is non-sighted, and ask yourself at every step if the experience actually is intuitive, or if you are accessing the experience the way you are because you already know how it works. Be diligent! Uncovering an accessibility issue is part of making the experience much better for everyone, not just non-sighted people or others who would self-identify as needing an accessible experience.

Change to a different breakpoint and do the same tests.

Testing with a High Contrast Theme

Both MacOS and Windows offer a high contrast mode to their users. High contrast mode is a tool used by visually impaired users to increase the readability of content on their computer. This tool is exceptionally useful for the web as modern websites are using thinner fonts, minimalist color palettes, and more subtle designs that can be difficult to read. The user will generally only enable this mode if the contrast of a website makes it difficult to read. A good website will offer a quality design so that users will not need to resort to this mode.

On MacOS, users are given an option to invert their screen color. This can improve readability of text, isn't as bright for people who suffer from light sensitivity, and can help people who suffer from color blindness discern between colors better. The user can open System Preferences > Accessibility. The user has a number of options to choose from; the main option would be to click the checkbox "invert colors". This is system functionality and will apply to all browsers.

On Windows, the user can apply different themes which will increase contrast of light and dark, hide images, and use bright colored text against dark backgrounds to help make text bolder and easier to read. The user can enable high contrast mode by going to Control Panel > Ease of Access > Optimize Visual Display, and then clicking the "Choose a high contrast theme" under the "high contrast" heading. I recommend the "high contrast #1" theme. Once enabled, the user will be in high contrast mode when they open up Internet Explorer or Firefox. If using Chrome on Windows, they will be asked if they wish to install a high contrast plugin.

Testing a website for proper contrast is a simple, low effort task that can make a huge difference to users without perfect sight. This is especially important for populations with aging demographics.

Manually Testing by Guideline

The table on **page 24** contains manual tests that should be done to assess if an experience meets the guidelines (this list includes general keyboard navigation tests outlined above).

Accessibility Testing Methodology

Selecting Pages to Test

Given the number of manual tests above, it should be clear that manually testing an experience takes time. For large-scale experiences, it would be a time-consuming and potentially prohibitive task to audit every single page. Luckily, most large-scale experiences are based on reusable templates and components. For the case of templated experiences, it is unnecessary to test each and every page. The minimum items that should be tested include:

All representative pages on the site. A representative page set will include examples of every kind of template and every kind of component used on the site. The idea is that testing this list of pages will give you confidence that all of the interactions on the site are accessible.

A list of all user flows for which the site was designed. For example, for an eCommerce experience, typical user flows include but are not limited to the path to registration, the path to login (including "forgot your password"), and the path to purchase. The user flows should provide URLs for each page of the journey. Testing each user flow gives reassurance that the tasks the site is designed to support are possible to do for a user using a keyboard or assistive device.

A list of "sanity check" pages. This will involve some redundant testing of templates or components already covered in the representative pages list. The number of pages tested will depend on the size of the site overall. These pages can be selected by popularity (in analytics tracking), or because the client feels they are important. The purpose of testing these pages is as a sanity check (hence the name), but also because some accessibility issues can be created through content alone. Meaning, that issues that did not arise on the representative page list could arise in the sanity check pages if inaccessible content was authored on those pages.

Selecting Testing Tools

Screen reader software is developed for specific browsers. It is not the case that you can use any screen reader with any browser. Consider the following criteria when selecting a testing tool:

Experience of auditor: If the person available to audit the site is an expert in the Safari/Voiceover browser/screen reader combination, it may make sense to do the audit using that tool. Consider the time it would add to learn a completely new tool to do an audit, and if this is worth it for your client.

Experience of client: If the client is most familiar with a particular screen reader, they may ask that the audit be done with that screen reader so they can more easily retest the issues.

Screen reader of choice for users: Ideally, if we could know what screen reader the majority of users employ, we could test in that screen reader. However, there is no way to tell through analytics which screen reader is being used (or if a screen reader is being used at all). The only way to find this out is for the client to conduct a screen reader survey. However, many clients are wary of doing this for it draws attention to their web experience and how accessible it is (or is not). Typically this information remains unavailable

to auditors. The most popular assistive devices reported in an independent survey conducted by WebAIM in 2015 are (in order): JAWS (63.7%), Window-Eyes (20.7%), NVDA (43%), and VoiceOver (30.7%). Usage of JAWS is declining rapidly; usage of Window-Eyes and VoiceOver is increasing. However, the problem remains that the users surveyed may not be your client's users.

Operating system and browser of choice for users: A way to infer which screen reader might be used is to look at the analytics reports for operating systems and browsers. For example, if a client's website is visited most often by people using the Safari browser on a Mac, then a good choice of screen reader to test the site is VoiceOver. If, however, most people visit on Firefox on a Windows system, a better choice might be NVDA.

It may not matter! The WCAG guidelines are browser/device independent. That means, from a compliance perspective, that the only requirement is to MEET the guidelines, NOT to work in a specific screen reader. This is because, if an experience meets all of the requirements in the guidelines but still doesn't work in a screen reader, it is possible that it is a screen reader-specific issue, not an issue with the experience. Our clients cannot be held responsible for screen reader-specific issues.

BROWSER/SCREEN READER COMBINATIONS	
Firefox	NVAccess (NVDA)
Safari	VoiceOver
Internet Explorer	Jaws
	Window-Eyes

In the majority of cases, if an experience meets all of the WCAG guideline criteria in one screen reader/browser combination, passing all of the manual and automated tests, it will work very well in any other screen reader/browser combination. It is not necessary and can make our clients less profitable if we tested, for example, all of the possible screen reader/browser combinations. At this point in our accessibility journey, it makes more sense to mindfully invest in testing in one screen reader/browser combination and making sure that it gets done properly, rather than overwhelming clients with a substantially higher cost to test in multiple screen reader/browser combinations.

Mobile Device Considerations

For responsive sites, it can be easier and faster to conduct the majority of accessibility testing from a desktop browser resized to various breakpoints. In rare cases, a website could have the same markup for the desktop, tablet, and mobile (or other) breakpoints. In this case, testing at a single breakpoint would be appropriate.

More typically, interaction and therefore markup variations will occur between breakpoints. For example, the global navigation menu on the desktop breakpoint will become a collapsed hamburger menu for tablet and mobile breakpoints. In this case, each breakpoint that has different functionality (markup) must be tested separately. Again, it is possible to do this on the mobile breakpoint of a desktop computer.

For mobile sites that are redirected to mobile phones only, testing must be done using a mobile device. For iOS devices, access Settings > General > Accessibility > VoiceOver > On. This will turn VoiceOver on. Then navigate to the web page in question (you will have to tap on applications twice to open them with VoiceOver on), and navigate through the site by swiping right. Swiping right will tab you through each node of the site, similar to using the arrow keys in the browser version of Safari and VoiceOver.

Prioritizing Defects

Typically a project will assign defect priority levels from 1-4 (priority 1 is the highest, priority 4 is the lowest) by definition. For example, a priority 1 defect will prevent a user from accessing content. Similarly, we can define priority levels to categorize accessibility related defects:

Problem Type	What It Means	Priority
Slammed doors (critical)	Barriers that stop someone from using an app or feature successfully—or at all. (e.g., functionality doesn't work with keyboard)	1
Frustrating (serious)	Problems that slow someone down, or force them into workarounds (e.g., having to backspace 10 times to get into a content area, discovering this after trial and error)	2
Annoying (moderate)	Things that make the experience less pleasant maybe even enough to leave (e.g., having a million "Learn More" links to tab through)	3
Noisy (minor)	Minor issues that damage credibility but are unlikely to cause problems (e.g., redundant screen reader content)	4

TABLE: MANUAL TESTING BY GUIDELINE

WCAG Guideline	Manual Test Description
1.1.1 All non-text content that is presented to the user has a text alternative that serves the equivalent purpose. (Level A)	CSS background images should not be used for images that convey important information. This is because you can't put alternate text on a CSS background image. Right-click on an image to view source, and if there is no tag, then it is probably a CSS background image. If tool detects alt attribute, it will not generate an error, but there may still be issues with alt attribute content.
1.3.1 Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)	Validate that changes in text presentation are not used to convey information without using appropriate markup (e.g., text styled as a heading but not marked up with heading tags), and that layout tables do not use any attributes associated with data tables (e.g., IDs and headers for columns/rows, or e.g., Use of <th> tag, "summary" attribute, etc. for a layout table).
1.3.2 When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)	Validate that an HTML layout table makes sense when linearized, i.e., when the table is read from top to bottom, left to right.
1.3.3 Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components, such as shape, size, visual location, orientation, or sound. (Level A)	Validate that a graphical symbol alone is not used to convey information (e.g., an empty shopping cart icon vs a shopping cart icon with an item in it, with no text alternative to indicate the cart's status).
2.1.1 All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)	Validate that all critical user flows are achievable through keyboard alone. Validate that all remaining content is accessible through keyboard alone.
2.1.2 If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)	Validate that there are no keyboard traps (interactions you can get into with your keyboard, but can't get out of). These would be most likely to occur in overlays and other interactive elements on the page, such as video players or image galleries.

TABLE: MANUAL TESTING BY GUIDELINE (CONT.)

WCAG Guideline	Manual Test Description
2.4.1 A mechanism is available to bypass blocks of content that are repeated on multiple web pages. (Level A)	Validate that a link is provided to skip navigation and other page elements that are repeated across web pages OR that a page has proper heading structure. Make sure it actually works to use the "skip navigation" link by tabbing to it, and hitting "Enter" with your keyboard to follow the link. Ensure that the content area moves into focus, then tab again and ensure that the tab focus continues to be in the content area. Note: A developer can provide the skip navigation link as a hidden anchor link, but it's helpful for keyboard users not using a screen reader if the link becomes visible on focus, or is always visible.
2.4.3 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. (Level A)	Validate that navigation order of links, form elements, etc., is logical and intuitive. Do this by tabbing through all elements. If the focus area does not move in a sensible way from in the reading order from top to bottom, there is an issue.
2.4.4 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. (Level A)	Validate that the purpose of each link (or form image button or image map hotspot) can be determined from the link text alone, OR from the link text and its context, by doing the following: (1) Locate content needed to understand how link text describes the purpose of the link, and (2) Check whether the content is contained in the same sentence, paragraph, list item, or table cell, or in the preceding heading. Validate that links (or form image buttons) with the same text labels that go to different locations are readily distinguishable (e.g., not in the same context in the markup). Note: If the design makes it difficult to include the link text IN CONTEXT in the markup, then the content should be updated.
3.2.1 When any component receives focus, it does not initiate a change of context. (Level A)	Validate that when any page element receives focus, it does not result in a substantial change to the page, the spawning of a pop-up window, an additional change of keyboard focus, or any other change that could confuse or disorient the user. (e.g., opening a new window when the page is loaded, or using a script to remove focus when focus is received).

TABLE: MANUAL TESTING BY GUIDELINE (CONT.)

WCAG Guideline	Manual Test Description
3.2.2 Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A)	Validate that required form elements or form elements that require a specific format, value, or length provide this information within the element's label. The information should be present between the <label> tags in the markup. Validate that form validation errors are presented in an efficient, intuitive, and accessible manner (e.g., form errors should receive keyboard focus and be read out by the screen reader).
1.4.4 Except for captions and images of text, text can be resized without assistive technology up to 200% without loss of content or functionality. (Level AA)	Validate that text can be resized up to 200% and remain legible. Test this in Firefox: View > Zoom > Zoom text only, then ctrl+ on your keyboard to increase the font size on your screen.
2.4.7 Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible. (Level AA)	Validate that it is visually apparent which page element has the current keyboard focus (e.g., as you tab through the page, you can see where you are). Note: This may need an associated design.
3.1.2 The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text. (Level AA)	Validate that the language of page content that is in a different language is identified using the lang attribute (e.g.,<blockquote lang="es">).

NOTES

[1] Ross, Nick. 80 Days that Changed Our Lives. "Advent of the Internet," 2012.

[2] WebAIM. "Introduction to Web Accessibility," 2016.

[3] Henry, Shawn Lawton. W3C. "Introduction to Web Accessibility," 2005.

[4] Interactive Accessibility. "Accessibility Statistics," 2008.

[5] http://www.uiaccess.com/myth-textonly.html

[6] https://en.wikipedia.org/wiki/List_of_countries_by_number_of_Internet_users

[7] http://www.afb.org/info/blindness-statistics/adults/facts-and-figures/235

[8] Hausler, Jesse. Salesforce UX. "7 Things Every Designer Needs to Know about Accessibility," 2015.

[9] https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-ict-refresh/overview-of-the-final-rule

[10] https://www.transportation.gov/sites/dot.dev/files/docs/Kiosk-website-FR-final%20rule.pdf

[11] https://www.adatitleiii.com/2017/08/website-accessibility-lawsuit-filings-still-going-strong/

[12] https://www.ada.gov/hrb-cd.htm

[13] W3C. "Web Content Accessibility Guidelines," 2008.

[14] WebAIM. "Designing for Screen Reader Compatibility," 2017.

[15] Menard, Jay. The Digital Echidna Blog. "AODA and You: Alphabet Soup—WCAG 2.0 A or AA?" 2013.

[16] https://www.w3.org/TR/UNDERSTANDING-WCAG20/conformance.html

[17] Sherwin, Katie. Nielsen Norman Group. "Placeholders in Form Fields are Harmful," 2014.

[18] https://www.w3.org/TR/UNDERSTANDING-WCAG20/media-equiv-audio-desc-only.html

REFERENCES

Enders, Jessica. SitePoint. "The Definitive Guide to Form Label Positioning," 2014.

Runyon, Erik. WeedyGarden. "Carousel Interaction Stats - June 2013 Update," 2013.

W3C. "Using ARIA," 2017.

Walden, Alison. Accessib.li. "If you must use a carousel, make it accessible," 2016.

Walden, Alison. Accessib.li. "How to make your wireframes more accessible in five easy steps," 2016.

Walden, Alison. Accessib.li. "12 manual tests for accessibility compliance you should do now," 2016.

Walden, Alison. Accessib.li. "Project implications of WCAG 2.0 level AA support (compared to single A)," 2016.

Walden, Alison. Accessib.li. "The Top 5 Most Costly Accessibility Issues," 2016.

Walden, Alison. Accessib.li. "Creating accessible experiences starts with experience design," 2016.

WebAIM. "WebAIM's WCAG 2.0 Checklist for html documents," 2013.

WebAIM. "Creating Accessible Forms," 2013.

WebAIM. "Creating Accessible Tables," 2017.

WebDev-il, "What is Web Accessibility? How to make web pages accessible."