Ramping it up!
An action-based guide to creating accessible websites

Introduction

Web designers and developers need to know about accessibility. But web development is often a distributed team activity, and the multi-faceted nature of making web content accessible can be a challenging task. There are official guidelines – the Web Content Accessibility Guidelines (WCAG) – but many find these guidelines difficult to understand (Lazar 2004). In addition, focusing in on the WCAG success criteria shifts the mindset from designing for a user's needs to compliance with rules.

This work – written as a guide – reframes accessibility around the design choices that web team members make during a web project's life cycle, with an eye to the WCAG's four fundamental principles: perceivability, operability, understandability and robustness (POUR). Each level of the *web accessibility ramp* (shown in Figure 1) represents content types or design decisions that are encountered in the development life cycle, with progress up the ramp requiring more awareness, skills and resources.

The multi-step *action-based guide* begins with text and a solid structure (Step 1) and ends with audio, video and other other formats (Step 6). Progressing from Steps 1 to 6 (up the ramp in Figure 1) requires increasing amounts of awareness, skills and resources.

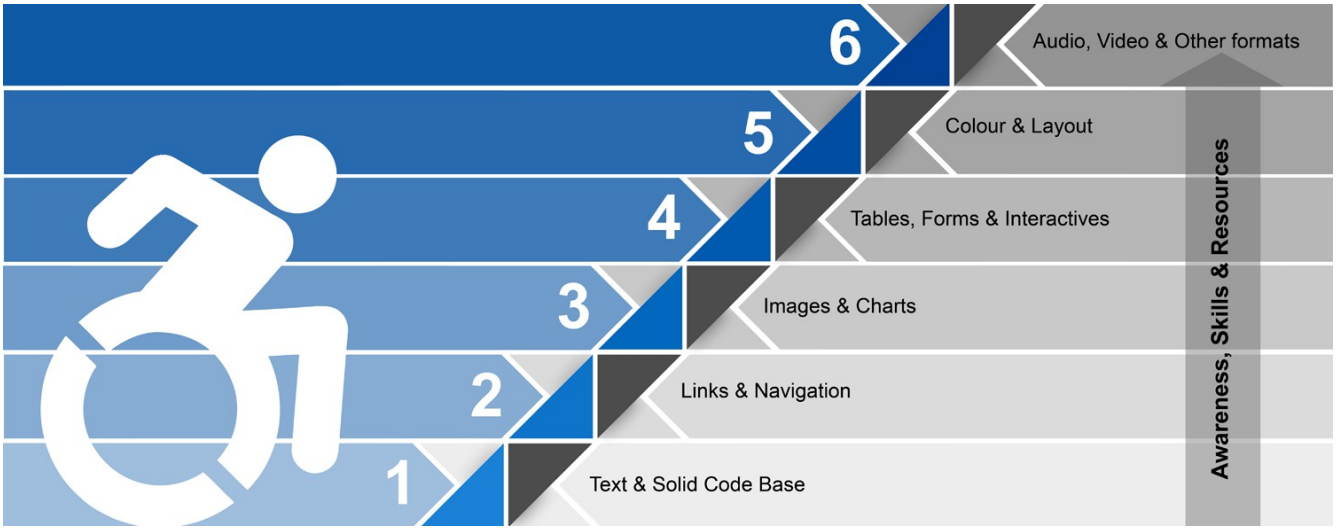

Figure 1 Each content type & design decision is a potential barrier (stair). The numbers associate each stair with techniques for building in the *virtual ramp*.

1. Text & Solid Structure

Electronic text is accessible by nature; without anything extra it can be interpreted and read by all devices. The beauty of electronic text is that it can be rendered visually, in sound, or, in tactile format such as braille (WCAG 2008). It can also be translated, simplified or transformed into an image-based or icon-based format. There are, however, a few critical techniques that one must employ to ensure that all aspects of text-based web content can be perceived, made operable and understandable to all devices and by all users. The more meaning we embed in the code the more options we have in our modal translations.

1.1 Code meaning into the content

HTML's built-in semantic structure (Clark 2003; WebAim.org 2013; Horton & Quesenbery 2014) is our main tool for making text fully and truly accessible. HTML is a structural markup language that gives us the ability to code meaning or semantics directly into the content (Horton & Quesenbery 2014). This semantic structure is invisible to most users, but not to machines such as screen readers and search engines. When the most meaningful HTML element is used for a particular piece of content, assistive technologies (AT) can interpret and convey that meaning to the user. Using HTML correctly and in the most meaningful way also improves a designer's ability to style that content more efficiently (this is discussed in section 5). Search engines also rank content based on their level of importance. Using a meaningful heading hierarchy with meaningful heading content will help search engines index content properly, so users can find it (Zeldman, 2007). Thus, several direct benefits result when designers and developers take the time to use semantic HTML markup in their designs. The first and most important benefit is a better and more usable experience for all users. Less obvious to the user are lower maintenance costs that result from a more consistent and efficient code base. More satisfied users and lower maintenance costs will result in a higher return on investment.

1.2 Keep the reading level simple

Web authors have the difficult job of composing content that can be understood by as many people as possible. A good rule of thumb is to keep the content as simple but appropriate to the audience. For example, a government web site has the full range of the population's reading level to consider when composing content, while a university professor writing content for a third year physics course has a much narrower target to meet. In both cases, the level is determined by the context.

1.3 Define the natural language

Defining the document's language and marking changes is important if content needs to

translated and it may also help assistive technologies like screen readers and text-to-speech software pronounce the content more accurately. Developers should use the *lang* attribute to define a page's natural language on all page templates and ensure content writer's know how to mark language changes when they add content to the site. The *lang* attribute can be placed on any HTML element.

```
Code sample:
<html lang=”en”>....
<p>Hello world can be expressed in french as <em
lang=”fr”>Bonjour tout le monde.</em></p>
```

1.4 Ensure a valid code base

Semantic structure together with a valid code base creates a solid foundation for a web site. There are three required elements that are hidden to non-developers, but are required on every web page for a complete valid (correct syntax) code base. Checking the validity of the code is a good practice. Error free code is more robust and likely to display consistently across browsers and devices including assistive technology (AT). These three essential bits of code are:

1. the DOCTYPE
2. the Character encoding (character set)
3. the HTML page TITLE element

Oddly enough, one or all of these items are often missing from web page templates. The doctype and the character set are required in order to ensure that a browser (or AT) can properly interpret the code. A concise and well-written unique page title is the first element that a screen reader reads out, telling the user that they have made it to the intended destination. Search engine algorithms weight the content in the page title heavily. A well-written page title welcomes the users that hear it read aloud, but also helps everyone find site more easily.

Summary for Text & Solid Structure

The perceivability, operability, understandability and robustness (POUR) of a website is profoundly affected by the content structure, the language setting, the reading level and the basic code base.

2. Navigation, Links & Landmarks

2.1 Use meaningful link text

Links are the stepping stones of the internet: they take users places. To ensure that users can easily determine that a link is really where they want to go, the text of the link needs to be visible and consist of unique, meaningful words that describe the destination or function of that link. Users of AT can scan through all the links on a page at once, so the link text needs to be clear even when accessed out of context. When a link wraps an image, the alternative text of that image must contain the destination or function of the link. The DRY-principle (Don't Repeat Yourself)(Wikipedia) is a good practice when it come to links. Repeating the same link several times on the same page can be frustrating for users who select a link with different text and then find it takes them to a place they have already visited. If duplicate links are required, it is important that the link text be consistent for each duplicate link. This practice creates a better user experience for all users.

2.2 Leave context changes to the user

A link is a change in context. Changes in context should be left in the control of the user. Specifically this means that one should not use the target attribute to force a new tab or window to open when a user selects a link. While this is unfortunately common practice, it is neither an inclusive practice nor a good practice. For some users (especially users of screen readers) this practice can be disorientating. Forcing a new tab or window starts a new browser history, so the user immediately loses access to their browsing history and the built-in *back button* of the browser won't take them back to where they came from. Furthermore, the target attribute has been deprecated, meaning it is no longer supported in the latest versions of the HTML specification. Using the *target* attribute will make one's code invalid and thus less robust (see section 1.4 Valid code base).

```
The following code sample is no longer valid in HTML and
presents problems to non-visual users.

<a href="http://www.myfriendswebsite.com" target="_blank">my
friend's site</a>
```

2.3 Skip navigation links & page landmarks

Skip navigation links were adopted as an accessible technique to provide direct access to a page's main content by allowing users of screen readers to by-pass long lists of navigation (usually the first content on the page). Designers often choose to hide these links from visual users. However, in reality, visual people who prefer to navigate using the keyboard also benefit from having access to these links in order to quickly get to the heart of the page's content.

With the advent of the ARIA specification and HTML5, page landmarks can now be coded right into the HTML via tags and the *roles* (Horton & Quesenbury 2014). Landmark tags and roles work exactly the same as in the built environment – they are "helpful way finders" (Horton & Quesenbury 2014). Users of screen readers can navigate from landmark to landmark greatly enhancing the navigation of the inner sections of the page.

2.4 Design consistent navigation

Well-planned, consistent navigation benefits all users. Consistency improves usability and prevents users from losing their way. Design teams spend a lot of time designing the site's navigation. Side navigation, drop-down menus, mega menus and bread-crumbing systems and even page links. Navigation is important. For navigation to be accessible it must be accessible via the keyboard which means all active controls can be reached, or "*focused*," using the keyboard (Sloan & Horton 2014).

Summary for navigation, links & landmarks:

Links and navigational elements are essential to a good user experience for everyone. Basic awareness training for content developers will ensure the creation of accessible links over the full lifespan of the project. A site's navigation and flow requires a hefty amount of design time anyway, adding accessibility goals to this process at the beginning adds little to the overall design budget; however, attempting to fix inaccessible drop-down menus at the end of a project can be extremely costly. Links and navigation are central to the usability of all websites; therefore, ensuring they are perceivable, operable and understandable to all users is a must.

3. Images & Charts

3.1 Text alternatives

Providing meaningful text alternatives for all content images is a requirement of the WCAG. Doing so makes the image content perceivable to those who cannot see. The goal of text alternatives is to maintain the meaning of the document whether or not the user can see the image. There are several ways to provide text alternatives for images.

1. In the code: The HTML *alt* attribute is a required attribute for all images. A succinct description fits well in the *alt* attribute. The content of the *alt* attribute is read aloud by screen readers, and browsers display this content on the screen if for some reason the browser cannot load the actual image. Keeping it succinct and clear at the same time can be a challenge.
2. In a visible element: In some cases a description of the image is beneficial for all

users. A web author or designer may choose to place the description of an image nearby in a nicely styled caption. HTML5 has a new tag, *figcaption*, specifically for this purpose.

3. Another possibility is that the image be fully described in the body of the nearby content, so no more supplementary description is needed to maintain the full meaning of the page.

In cases 2 or 3 above, and when an image is purely decorative, the *alt* attribute can be left intentionally blank to prevent a screen reader from reading out unnecessary, or duplicate content. In all cases, writing good alternative text and descriptions for images is not an easy task. Good image descriptions depend on context (WebAIM.org) and once the appropriate text is written by a web author, the designer should make sure it looks good, and the developer should make sure the description ends up in the right place in the code. Accessible web development is everyone's responsibility, and requires a team effort. Because of this, accessibility goals need to be communicated and made clear to everyone.

3.2 Complex images

For charts or complex images, one likely needs to link to supplementary content on a separate page. To do this, the link to the description must be accessible, and an accessible link back from the description page must be provided. Transcripts to video can be handled in a similar way. Linking to long descriptions can also be achieved programmatically using the *longdesc* attribute; however, this technique makes the long description content available only to certain users – users of screen readers.

Summary for Images & Charts

Images are not text-based content. To ensure the meaning of the images are perceivable to all users, we must provide an appropriate description of the image. This is an easy accessibility requirement to meet, but it is not always easy to get it right (WebAim.org). Due to the distributed nature of web design and development, image descriptions can easily fall through the cracks; a problem that is easily fixed through better work flow and communication. Scrambling to write descriptions for hundreds of images right before a launch is a costly scenario that is easily avoided by adding “accessible thinking” to the design process.

4. Tables, Forms & Interactives

As we make our way up the steps of potential barriers, our web content types become more complex, and require more skill and resources to make them accessible. Just by their function and purpose, tables and forms are complex structures. Good design can first

reduce complexity by keeping forms and tables simple. When designing a table for data, one should avoid using complex data relationships (i.e. avoid spanning columns and rows). For forms, one should ask only for the data that is needed (nothing more), clearly communicate errors and how to fix them and clearly communicate an events successful completion.

4.1 Use HTML's rich semantic markup for tables

A rich set of **semantic table markup** is available (*caption*, *thead, tfoot, tbody, th*, *td*); however, these elements are rarely used to their fullest potential. The *caption* element is of particular importance as a well-written caption offers a description of the data presented in the table. A concise descriptive summary is likely useful for all users, and for some users improves access to the content. A user with a learning disability may have trouble interpreting the data in the table and the *caption* can facilitate their ability to understand the data. For screen reader users, reading through table data is a laborious and time-consuming task, so a well-written *caption* will help the user make an informed choice on whether to embark on the full reading of the table data cell by cell or to skip over it as the *caption* provided what the user needed.

A heading element (*h2* through *h6*) is often used to introduce a data table; the *caption* element has two direct benefits over a heading:

1. the *caption* has an explicit relationship to the *table* element in the code. A heading element can be close in proximity to the table thus establishing an implicit relationship, but it cannot be programmatically connected to the table. Explicit relationships are more clear to users of AT.
2. the *caption* is unique from all other headings and thus it's presentation can be controlled separately by the design code offering the designer more flexibility over the presentation of the element.

Explicit relationships within the table are also very useful for the user and the designer for the same reasons. Relationships can be made clear in the code which is accessible to AT and similar elements such as the *tr*, *th* and *td* can be more easily targeted for visual enhancements. Both improve readability for all users. There are 3 main table sectioning elements that are rarely used by programmers, and that are very useful to designers and assistive technology. They are the *thead, tfoot* and *tbody*. The elements are placed in the code in that order, so non-visual users hear the summary data (often found in the *tfoot*) read before the data. This seems wrong to the visual programmer, but makes perfect sense to the non-visual user. The *th* elements act as labels for columns and rows and finally the *td* elements hold the actual data. Directional relationships can be further refined and defined by using the *scope* attribute on the *th* elements.

One last note on HTML tables, we use tables for tabular data only–never for content layout! Using a table for content layout does not create a meaningful document hierarchy making content difficult to access for some users. It also over complicates the code making the website less efficient and more difficult to maintain. One must use design thinking to solve layout problems and HTML tables to describe the relationships of tabular data.

4.2 Accessible forms & interactives

HTML forms and interactive games and applications are very dynamic structures. A user enters data and makes choices of one kind or another. When ready, the user submits the information. Many things can go wrong, causing frustration and resulting in a bad user experience. Before even getting to the code, the first task is to clearly define data requirements and goals. Simplification, intuitive flow, and a clear communication of errors and success are essential elements of a good user experience. Once these items are taken care of by designers, the developers need to employ the rich set of semantic form mark-up (and there are a lot of different form elements) to create and organize accessible forms. Using the built-in active form elements that are part of HTML will also help with keyboard accessibility. Like links, form elements are active HTML elements and receive focus by default. Keyboard accessibility benefits a large and diverse group of people who don't use a pointing device like a mouse (Sloan & Horton 2014).

The most basic form relationship, the explicit *label-control* relationship which programatically connects the form control with its label, is often left undefined, or worse, left out all together. This relationship is a must for creating an accessible form. The *label-control* relationship is strengthened further by visual proximity in the design, but visual proximity alone is not enough. Developers must explicitly code in that *label-control* relationship. It is easy to do when you just make it part of your regular work-flow.

Some form elements are used for creating logical groupings. Clear organization will help users understand the form better. Using grouping elements effectively takes practice. And one last point is that visuals need to match what is in the code, for example, a submit button's alternative text must match the visible submit button text for a user of voice activation software to be able to activate it. Often during the design process the visual text and the actual text can get out of synch.

The overarching goal of a form is successful completion. It is paramount to clearly define errors and guide the user on how and where to fix them. The user must be given the power to choose more time to prevent any unwanted time-outs. The ARIA specification is employed to define roles, states and behaviours that HTML cannot define on its own. For example, ARIA can define that a form field is required whereas HTML before HTML5 couldn't do that. There are many improvements for forms in HTML5 (Pilgrim). Finally, with the rich semantic form code in place, the designer has more more control over the visual

presentation of the form, to make it look awesome.

> Label and control connected via *for* and *id* attributes:
> `<label for="user-name">User Name</label><input id="user-name">`
> `<label for="password">Password</label><input id="password">`
> `<button>login/button>`

Summary for Tables, Forms & Interactives

As website designers, builders or owners, we cannot predict how a user will access our site. She may use voice activation software, a screen reader or zoom text. He may have a bad shoulder or a busy hand and just prefer to use the keyboard instead of a mouse to tab through the page, form or application. Understanding the complexities of how to build accessible tables and forms can mean the difference between a user getting essential information or not getting that information. With forms, it can be the difference between the success or failure of a critical interaction such as a sale. Both situations can effect the bottom line.

It is important to note that data tables and web forms have both been the result of landmark legal cases in web accessibility (Maguire v. SOCOG 1999 & Jodhan v. Canada 2011) that were filed and won under human rights legislation.

5. Colour & Layout

The most important feature or ability we have as web designers and developers, in my opinion, is the ability to decouple the structure from the presentation. This ability gives us a lot of control over the content and a lot flexibility in its presentation. Up until now we have discussed how to better use the full features of HTML to describe content structure. We now turn our attention to the Cascading Style Sheet standard (CSS) which is used to control all aspects of how that structure will look. The more meaningful and consistent our structure is, the easier it will be to efficiently control its presentation in diverse and even unique ways. Using these two standards together effectively gives us the most control and flexibility over our designs. Choices about design – about colour and layout – are placed high the ramp as these are fundamental choices that have impact across the project and implementation requires a high level of skill in CSS and HTML. Design decisions with this kind of impact, of course, are best made at the beginning of the project.

5.1 Choose colours wisely & test for contrast

So what design choices about presentation factor into creating accessible web pages? Due to many types of visual impairments, including colour blindness, as well as just differences in

colour perception, users see colour quite differently. Because of this, web designers have to think very carefully and critically about colour. Colour choices are critical decisions as they are often tied to many other aspects of a brand or product. It is not easy to change one colour, let alone a palette of colours, at the end of a project. Working the following golden rules about colour into the design process will help designers make accessible colour choices at the beginning of the cycle.

1. Test colours for sufficient contrast ratios.

The WCAG criteria for sufficient contrast ratios between an element's foreground and background have been in place since the standard was created. These requirements are not difficult to meet and do not hinder visual aesthetics when ratios are part of the colour decision making process. Making the decision to do accessible design means choosing colours that are not easily confused and testing for contrast. Testing all the possible foreground and background combinations that occur in the design (body text, link states, callout boxes, buttons, etc.) will increase the perceivability and robustness of the design.

2. Don't rely on colour alone for meaning.

Since colour perception is so diverse, and colour is often used to communicate meaning, it is important that an element's colour is not the only way of determining the element's function or purpose. For example, headings are often coloured, and designers also use size, weight or font-family to communicate the heading's position in the document hierarchy. Another example is the styling of links. Links are active elements that have to look distinct from the surrounding text. Colour is often used to make links stand out. A second formatting difference such as underlining, font-style or the addition of a small icon are all potential solutions for creating links that can be seen by everyone. Making accessible colour choices not only helps users with vision impairments, but also helps users in less than ideal lighting situations.

Cynthia Brewer's work (Brewer 2002-2013) in map accessibility is a great tool for choosing colour-blind-safe palettes. Styles for link states, focus in particular, should be explicitly defined. These states are: *link, visited, hover, focus, active*. A visible focus style is essential for helping users – especially keyboard users – see where they are on the page.

5.2 Layout

Using CSS and HTML together properly forms the foundation of accessible and flexible web design. With a solid HTML structure in place, designers can use CSS to completely change the page's design based on the needs of the user. The HTML literally stays the same, but is presented differently based on the user's context. This kind of flexibility in web design affords maximum inclusion. An essential part of making our designs flexible is keeping the

code base clean and the HTML and CSS decoupled. This means keeping presentational elements out of the HTML and introducing extra styling hooks such as CSS class names and id's carefully and judiciously.

A highly skilled CSS/HTML designer/developer improves flexibility by optimizing and reducing the code. This results in a website of much higher quality – less code loads faster, is easier to maintain and will last longer as technologies change – making the site more robust.

With the distributed nature of web design and development, it's often challenging to keep the code clean over the long term. Modern sites are often part of a content management system (CMS) that allows web authors who may have limited knowledge of HTML to add content to the site. The CMS's HTML editing tool often allows users to introduce presentational HTML elements and inline CSS styles. This can have an effect on accessibility of the content. Awareness training and team style guides go a long way in creating and maintaining good coding and writing practices. Training for web content authors is an important part of maintaining accessible content and a flexible design over the lifespan of the site.

Summary for Colour & Layout

Using HTML and CSS together with accessible design decisions about colour and layout have a big impact on the accessibility and flexibility of the web site. Often when we solve access issues for users on the margins there is a benefit for all users. For example, making sure buttons and menu links have a wide selectable area not only helps people with hand tremors, but also users commuting on a bumpy bus. A user with a learning disability is well-served by a simplified layout much like all users of mobile phones and tablets. Making essential accessible design choices around colour layout at the beginning of the project requires infinitely fewer resources than retrofitting a developed site. And developing team resources that support accessibility will improve the longevity of the site which should increase the return on investment.

6. Audio/Video & Other Formats

At the top of the ramp we find types of content that generally fall outside the responsibility of the web designer/developer. These types of content are not necessarily difficult to make accessible, but doing so often requires special processes and resources. For example, a third – party captioning and transcription service may be required to make video and audio content accessible. Making other formats such as PDF and word processing documents accessible is outside the normal skill set of the members on the web team and may need to be outsourced. The cost of making these formats accessible needs to be seen as a natural

cost of production rather than as a prohibitive cost.

6.1 Audio & Video

Like images, audio and video (multimedia content) are not text-based. To make multimedia perceivable, alternatives must be provided. These alternatives can take several forms such as transcripts, captions and audio/video description.

1. Transcripts provide the dialogue in an asynchronous, but nevertheless, in a perceivable format. This is the preferred format for audio content from podcasts and interviews. Transcripts may be considered an acceptable alternative for videos that consist of only a talking head.
2. Captions for video are the preferred choice. Closed captioning is usually the preferred implementation method and the user is provided an accessible control to turn captions on and off. High quality captions include the description of meaningful sound effects and follow recognized standards for readability.
3. Audio description (or video description) is the primary technique for making visual media accessible to the blind and visually impaired. A second sound track that takes advantage of natural pauses in the dialogue is used to describe the actions happening in the visuals. This type of accessibility is highly specialized.

On the web, however, access to the media player controls is often the first barrier when it comes to audio and video content. User control is paramount in all aspects of design and especially important with multimedia content. Because multimedia content is multi-sensory, auto-playing content can be highly disruptive to screen reader users and rather annoying to all users. Thus, no media should play automatically (WCAG Samurai 2008). The WCAG is actually more lenient on this issue (specifically see Success Criterion 1.4.2), however, the WCAG's own help guidelines actively “discourage the practice of starting sounds automatically”, as it can interfere with a screen reader user's ability to find the control that stops the sound from playing. Thus, in order to make the media operable, the controls that start and stop the media must be keyboard accessible.

6.2 Other Formats

HTML content is the native content type for web pages. Thus, it is the natural choice for putting content on the web. Other types of document types, however, do exist, and are important and found on the web. The portable document format is the most common non-HTML document type found on the web (PDF Techniques for WCAG 2.0). PDF documents, like HTML content need to be accessible. Luckily, the basic concepts for making PDFs accessible are very similar to those for making HTML content accessible. PDF documents need to have a natural language defined, meaningful document structure, descriptions for

images, a logical content order and form controls must have appropriate descriptions. The tools, however, to make PDFs accessible are, however, quite different and often require skills that a web designer/developer does not have. So, the first design question for PDF (or another non-HTML document type) is: Is there a reason not to use HTML? If there is, indeed a reason to use PDF, then the steps to make that document accessible need to be learned and taken. The PDF standard has a content tag nomenclature akin to HTML, however, these *tags* are not as apparent nor as easy to adjust and change as HTML tags. Accessible PDFs must be tagged.

Summary for Audio/Video & Other Formats

Audio, video and other formats have been placed at the top of the staircase mostly because making these types of media accessible are often a real cost item. Special in-house human resources have to be allocated or a decision to out source this type of content has to be approved and made. Speech recognition software and auto captioning tools are getting better and better; however, formatting for grammar and readability are still a long way off from automatic. As the users of the web want more video driven content, vigilance is required to ensure this content is made accessible.

Conclusion

Creating websites that meet the needs of people with disabilities overlaps with people with situational limitations such as being in a loud environment, having limited use of hands, using a mobile device, using old hardware or having limited access to bandwidth (Henry et al 2014). Just as access ramps and curb cuts benefit more than just people in wheelchairs, the *virtual ramp* or *electronic curb cut (*Jacobs 1999) benefits more than just people using assistive technology. Making accessibility a priority by integrating "accessible design thinking" into the design process helps us build better websites that are better for everyone.

An extensive long term study (Hanson& Richards 2013) that examined the progress on web accessibility found low compliance rates with the Web Content Accessibility Guidelines (WCAG), but at the same time, uncovered a growing adherence to *some* criteria. A follow-up study (Richards et al 2012) found that at least part of the improved accessibility was "not attributed to a focus on accessibility per se", but a side effect of other improvements: improvements in browser capabilities, concerns over page rankings and attention towards cross-browser and cross-device compatibility.

We can also invert our perspective. Cross-device compatibility and better search engine rankings (SEO) can also be viewed as a beneficial side effect of accessible design. Building tools that help designers and developers integrate accessibility into the design process, can result in better cross-device compatibility and SEO *as a side effect*. To this end, I feel that we

need tools that specifically address the barriers to understanding and implementing the WCAG. This guide is hoped to be such a tool. It's focus is practical – to enable people to find practical solutions to make accessible design choices in order to ensure that web content is POUR: perceivable, operable, understandable and robust.

References

(2013). Semantic Structure. *WebAim. Retreived from: http://webaim.org/techniques/semanticstructure/*

Don't Repeat Yourself. *Wikipedia*. Retrieved from: http://en.wikipedia.org/wiki/Don%27t_repeat_yourself

Brewer, C., Harrower, M. & Pennsylvania State University. (2002-2013). *Color Brewer 2.0*. Retrieved from: http://colorbrewer2.org/

Clark, J. (2003). *Building accessible websites*. Indianapolis: New Riders.

Clark, J. (2005). Facts and opinions about PDF accessibility. *A List Apart*. Retrieved from: http://alistapart.com/article/pdf_accessibility

Hanson, V. & Richards, J. (2013). Progress on website accessibility? *ACM Transactions on the Web, 7*(1). Doi:10.1145/2435215.2435217

Henry, S., Abou-Zahra, S., & Brewer, J. (2014). The role of accessibility in a universal web. Paper presented at the *W4A – 11th Web for All 2014 Conference*. DOI:10.1145/2596695.2596719

Horton, S. & Quesenbery, W. (2014). *A web for everyone: Designing accessible user experiences*. Brooklyn: Rosenfeld Media.

Horton, S. (2015). Interview in Effecting Cultural Change Seminar, OCAD University, Toronto, ON.

Jacobs, S. (1999). Section 255 of the Telecommunications Act of 1996: Fueling the Creation of New Electronic Curbcuts. Retrieved from: http://www.accessiblesociety.org/topics/technology/eleccurbcut.htm

Jodhan v. Canada (2011). http://recueil.cmf.gc.ca/eng/2011/2010fc1197.html

Lazar, J., Dudley-Sponaugle, A. & Greenidge, K. (2004). Improving web accessibility: as study of web master perceptions. *Computers in Human Behavior*, 20(2). pp 269-288. Doi: 10.1016/j.chb.2003.10.018

Maguire v. SOCOG (1999). Retrieved from: https://www.humanrights.gov.au/bruce-lindsay-maguire-v-sydney-organising-committee-olympic-games

Pilgrim, M. A form of madness. In *Dive into HTML5. Retrieved from:*

http://diveintohtml5.info/forms.html

Richards, J., Montague, K. & Hanson, V. (2012). Web Accessibility as a Side Effect, *ASSETS '12 Proceedings of the 14th International ACM SIGACCESS Conference on Computers & Accessibility*. pp 79-86.

Sloan, D. & Horton, S. (2014). Why Keyboard Usability Is More Important Than You Think. *User Testing.* Retrieved from: http://www.usertesting.com/blog/2014/10/08/why-keyboard-usability-is-more-important-than-you-think/

Zeldman, J. (2007). *Designing with web standards.* Berkely: New Riders.

HTML5: http://www.w3.org/TR/html5/

PDF Techniques for WCAG 2.0. http://www.w3.org/TR/2014/NOTE-WCAG20-TECHS-20140408/pdf.html

W3C: http://www.w3.org/

WAI: http://www.w3.org/WAI/

WAI-ARIA: http://www.w3.org/WAI/intro/aria.php

WCAG 2.0: http://www.w3.org/WAI/WCAG20/glance/Overview.html

WCAG 1.0: http://www.w3.org/TR/WAI-WEBCONTENT/

WCAG Samurai: http://www.wcagsamurai.org/

Tools & Resources

- Brewer Palette: http://colorbrewer2.org/
- Paciello Group. Contrast Analyser Tool: http://www.paciellogroup.com/resources/contrastanalyser/
- Verou, L. Contrast Ratio Tool: http://leaverou.github.io/contrast-ratio/
- WebAIM Contrast Checker Tool: http://webaim.org/resources/contrastchecker/
- Wave Accessibility Evaluation Tool: http://wave.webaim.org/
- WebAIM: http://webaim.org/
- The Docs at WebPlatform.org