Accessibility in e-Learning

What You Need to Know

Prepared on behalf of the
Council of Ontario Universities by:

Greg Gay, Inclusive Design Research Centre
OCAD University, Toronto, ON

JULY 2014

Contents

Introduction

This paper looks at what you need to know as an educator about the accessibility of online courses, course materials, and other web-based learning activities and tools as part of teaching and learning at postsecondary institutions in Ontario. It introduces accessibility not simply as a requirement of the *Accessibility for Ontarians with Disabilities Act* (AODA) to accommodate people with disabilities, but as a best practice for creating and delivering e-learning opportunities that are accessible to everyone. It provides basic principles that capture the essence of accessibility – principles that you as an educator can apply going forward as part of the e-learning development process and the creation of online courses.

The paper provides some examples of how people with different learning needs can learn online. It introduces some of the common assistive technologies (ATs) that people with disabilities might use to access courses on the web, and describes some of the barriers that they could encounter and that might prevent them from participating fully in e-learning. It also provides practical strategies to create accessible learning experiences, as well as strategies for identifying potential barriers that may limit access to online learning activities and materials. Understanding what accessibility means in the context of e-learning is necessary for any instructor, content author, or e-learning specialist who creates or leads activities on the web.

Defining Accessibility and Inclusion

Traditional Definitions

Traditional definitions of disability focus on a medical description, “typically a physical or mental condition that limits a person’s movement, senses, or activities.” [Oxford Dictionary] When talking about accessibility however, the traditional definition of disability limits what one might consider when developing content that is accessible to everyone. Other considerations come into play when we talk about accessible learning, including how learners process different types of information. For instance, the common visual, verbal, kinesthetic distinction in learning styles [1] should also be considered when designing content that is accessible to everyone.

Disability vs. Mismatch

Everyone has a disability at one time or another that affects her/his ability to consume information or learn from it. Take for instance watching television in a noisy environment; one might be considered hearing impaired, regardless of how well one might normally hear. In such a case, captions would allow a person to watch the television, providing the viewer with an option of reading the captions as an alternative to listening to the audio. These types of accommodations are often referred to as “curb cuts.” Curb cuts were initially created to allow people in wheelchairs to navigate between street and sidewalk, but they also accommodate a person pushing a baby carriage, a person on a bicycle, or an elderly person walking who might find stepping up a curb stressful. In these cases, the accommodations will benefit a wider range of people than just those with disabilities. This paper proposes that it would be advantageous to

view accessibility as curb cuts – in that it provides accommodations for persons with various disabilities but that, in its application, it also has the potential to serve a larger population.

Inaccessibility is essentially a mismatch between the needs of an individual within an environment and the format in which that information or content is provided. Accessibility in e-learning can be thought of in terms of matching content or activities to individuals. For learners who are blind, text or audio alternatives for graphic materials can provide a match depending on the abilities of each learner. For learners who are deaf, captions or transcripts of audio material would be used to provide the match. For visual learners, graphics and other visual presentations of information would provide the match. For kinesthetic learners, hands-on or step-by-step presentation of information would be used. When developing content, consider providing multiple representations of the same information. Processing information through multiple modalities has long been associated with better learning outcomes [2], so adaptations of e-learning content to accommodate people with disabilities can provide the "multimodal curb cut."

What Is Web Accessibility?

It was not until the late 1990s that web accessibility – in essence, access to online information for people with disabilities – became a concern for web developers. About this time, the World Wide Web Consortium, known as the W3C, introduced the Web Accessibility Initiative (WAI), which focused on making the web and web technology accessible to people with disabilities. Today, this is still somewhat specialized knowledge, and not all web developers have mastered it. For the general public, web accessibility may not be a widely known concept but, with the AODA and emerging accessibility laws worldwide, there is a growing awareness around the need for equal access to the web and web-based learning opportunities.

How People with Disabilities Use the Web and the Barriers They Face

Vision Loss

Vision loss encompasses an array of conditions ranging from simple nearsightedness or farsightedness, to age-related difficulties, to a variety of colour-blind conditions, to significant loss of sight and complete loss of sight, with a continuum of conditions in between. How people with vision impairments use the web is just as varied. Those who are blind face the most significant barriers in accessing web content, given the web's obvious visual nature. Any meaningful visual element in web content must have an alternate text or audio equivalent to accommodate this group of learners.

A **screen reader** is used to access the web for those with total or significant vision loss. It extracts the text from web content, as well as features of the computer's operating system, and reads it back to the person using text-to-speech technology. Screen readers also have a collection of tools that allow a person who is blind to list the headings or links on a page, navigate through tabular information in a meaningful way, and successfully complete forms. Barriers are created when correct heading elements have not been used to structure content,

when link text is not meaningful on its own (e.g., “click here” provides no useful information), and when forms are not properly labelled. Some common screen readers include JAWS, Window Eyes, and NVDA for Windows systems; VoiceOver for Mac systems and iOS mobile devices; Orca for Linux systems; and Talkback for Android mobile devices.

Another significant issue for users of a screen reader is **keyboard accessibility**. Most learners who are blind will use a keyboard to access web content, not a mouse. Web developers often overlook this fact, creating features in websites or applications that operate with a mouse click and that do not include an equivalent keyboard alternative. For a quick test of keyboard accessibility, press the Tab key repeatedly and follow the cursor’s focus through the web content to be sure that all elements receive focus. While navigating with the Tab key, if you press the Enter key (or sometimes the space bar) when features receive focus, this action will reveal whether these elements are also keyboard operable.

An area that could create barriers for those who are blind is multimedia content. Making this accessible requires not only that the video player be keyboard accessible, but also that all meaningful actions in the video that are not understood by listening to the audio track are augmented by **audio description** (or described video). Audio description is created by recording short, spoken descriptions of the meaningful actions in a video, and weaving these descriptions into a separate audio track in the video file.

For those with significant vision loss but enough residual vision to see magnified content, a **screen magnification** program may be used to access web content or their computer’s operating system. These users could also just rely on a **browser’s zoom feature** to increase the text size to make the text more readable. Barriers can occur when content has been created in a way that does not allow for resizing. Web content sizes can be defined in absolute measures using points (pt) or pixels (px), or in **relative measures** using “em” or percent (%). Potential barriers can result when content is created using absolute measures, or a mix of absolute and relative measures. Content may not resize, or only parts of it resize. For example, if you use relative measures to create text size but use absolute measures to create the containers in which the text is presented, when the text size is increased, the container stays the same width, sometimes creating a column of words that can be difficult to read.

Another common issue for those who view magnified content is presenting **text in images**. In many cases when images are magnified, the text in the images becomes pixelated, losing its sharp edges and making it more difficult to read. Wherever possible, you should use actual text, to ensure that it remains readable regardless of the zoom factor.

From a curb cuts perspective, developing content using relative measures makes the content more adaptable to various screen sizes – for example, content created for a computer screen that is viewed on a mobile device – than if the same content had been created using absolute measures. Likewise, ensuring that visual content has meaningful text alternatives can help make visual information indexable by search engines, and therefore make image searches easier for everyone.

Hearing Impairment

People with hearing loss will typically experience fewer barriers in accessing web content than people with vision impairments, though multimedia and audio content will obviously present barriers if alternatives such as **captions or transcripts** are not provided. Many tools are available for creating captions, including Capscribe, as well as web-based tools such as Amara (formerly Universal Subtitles), and the captioning tools built into services such as YouTube. YouTube also has an automated caption service that uses speech recognition technology to produce text from a video's audio track. Automated captioning, however, is still an emerging technology and ideally should *not* be used to create captions for your videos. Test this for yourself by turning on YouTube's automated captioning for, say, an instructional video to get an idea of the errors that can occur.

Consider using captions, as a curb cut, to help all listeners in a noisy environment to read what would otherwise be an inaudible audio track. Captions can also make video indexable by search engines, making it possible to search for specific words in a video and potentially jump directly to that point in the video where the words are spoken. Captions can also be used by people whose first language may not be the language presented in the video, and read along while listening to the video. Figure 1, which follows immediately, shows the transcript displayed along with a YouTube video when captions are available.

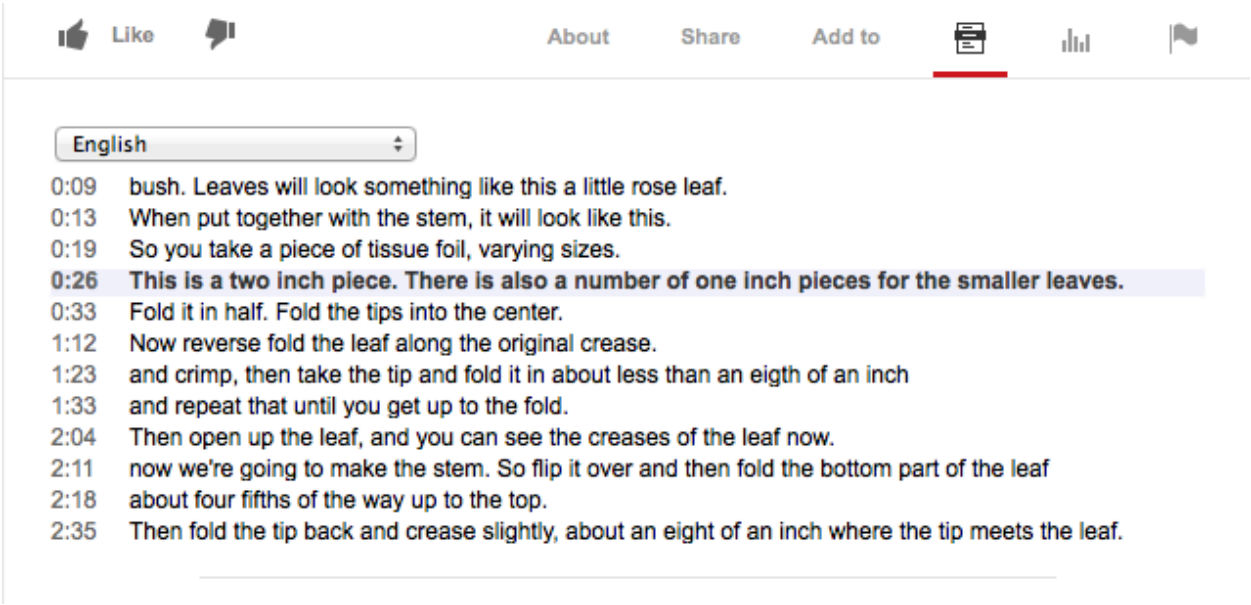

Figure 1: The YouTube transcript feature, which is a compilation of captions from a video, makes the video content indexable, and allows viewers to read while they watch the video. (The image unavailable in French.)

Motor Impairments

People with significant motor impairment generally face barriers associated with using a mouse or aspects of a keyboard to access web content. This group will often rely on **various keyboard technologies** to access web content, including a “large key” keyboard, an onscreen keyboard, or a scanning keyboard that is operated with a single switch or head mouse.

Switch access is similar to a mouse click. A switch can be a button device (similar to those shown to the left) that can be activated by a small body movement or by leaning on the device. Such a switch might be used by a person with limited mobility to operate an onscreen scanning keyboard.

Sip and puff switches operate in a similar fashion. However, instead of using movement to activate the switch, a person would use her/his mouth to either sip on the straw to mimic a mouse click, or blow into the straw to mimic a double click. A switch might be used along with a **head mouse**, which uses head movements to control the position of a mouse pointer on the screen.

There are now a variety of commercial and open-source, onscreen scanning keyboards that can be used with switch access devices. Typically, depending on your configuration settings, when you access the keyboard, it will begin to slowly scan down line by line. As shown in Figure 2, when the line with the target letter (in this case, “f”) is reached, the person might sip on the switch, or push a button switch, to start the scan moving right letter by letter, and when the scan reaches the target letter, sip again to type that letter. In the top row of the keyboard, a word prediction feature lists potential words that can be selected during the scan, or the scan can continue line by line and another letter could be selected to update the word prediction list. When the target word appears in the list, it would be selected during the scan of the first line in the keyboard, which would then enter the word into a web form.

Figure 2: Gnome onscreen keyboard (GOK)

People with limited mobility but with full use of sight and speech could use **speech recognition software** to navigate web content. Along with a variety of standard commands, a person might say the words associated with a link, along with a command such as “click” to activate a link. The same might be true for operating buttons, speaking the word on the button and announcing the word “click.” They might also bring focus into a form field on a web page, and begin speaking words to have them transcribed into the form.

The barriers that people with mobility impairments typically encounter are similar to those encountered by people who are blind – for instance, buttons created that only use images of text and that provide no text alternative. Speaking the text of this kind of button has no effect. Form fields that are not properly labelled may also present barriers, particularly for those who having difficulty positioning a mouse pointer over a tiny form element such as a radio button or checkbox. To format a proper label, use the HTML <label> element; it makes the label itself clickable to activate a form element, providing a larger target area on which the user can click.

Technologies created for people with mobility impairments have resulted in a number of curb cuts. Onscreen keyboards are now commonly found in kiosks, bank machines, ticket dispensers, and of course all smartphones and tablet devices. Speech recognition, in addition to captioning YouTube videos, is found in various hands-free devices, and can be used for transcribing, or simply for enabling multi-tasking computer users to work hands-free.

Cognitive Impairment and Learning Differences

Barriers for people with cognitive or learning impairments generally revolve around consistency, predictability, complexity, and memory. A wide range of cognitive impairments can be made more manageable in terms of accessibility if they are grouped based on functional abilities [3]: the ability to understand or comprehend; the ability to adapt or problem-solve; the ability to recall or recognize; and the ability to attend to the task at hand.

A variety of strategies can be used to make content understandable to the broadest possible audience. While a typical web user would likely deduce from clicking a “sign-up” link and landing on a page called “registration” that s/he is on the right page, a person with a cognitive impairment may not draw the association between the two words of similar meaning. The user could become confused or frustrated, which may ultimately prevent her/him from filling out the registration form. This could be easily avoided by simply using words in the link or link text that matches the title and heading on the page the link leads to.

The reading level of web content may also create barriers for those who have difficulty with comprehension. When writing for the web, especially for sites geared to the public, use the simplest language possible, generally at a grade 9 or 10 level. Avoid using sarcasm, idioms, metaphors, and other non-standard forms of writing that run the risk of being misunderstood. As a curb cut, the use of simple language can improve a website’s readability – and therefore its understanding – for people whose first language is not the language in which the content is

being presented. If your website's audience is more defined and advanced, use language that is appropriate for your audience, but still be mindful of simplifying words wherever possible.

While reading levels and complexity make up the majority of potential barriers, other areas of concern for learners with cognitive impairments are math comprehension and visual comprehension. The most effective strategy in both instances is providing multiple representations of web content.

While many strategies can help to make web content more accessible or usable, there will be times when an author may be required to use specific, advanced language (for example, in a legal document) that may not be understood by a person with a cognitive impairment. In this case, the use of simpler language would not be appropriate.

Standards and Specifications

Standards play an important role in the development of accessible e-learning. The primary accessibility standards are the W3C's Web Content Accessibility Guidelines (WCAG 2.0), its Authoring Tool Accessibility Guidelines (ATAG 2.0), and its Accessible Rich Internet Application (ARIA 1.0) specification.

The web accessibility requirements of the AODA mirror those of the WCAG 2.0, with the exception of its guidelines that refer to captioning for live multimedia content and providing audio descriptions for video (Guidelines 1.2.4 and 1.2.5). For a more details on accessibility standards relevant to e-learning, see Accessibility in e-Learning: Standards and Specifications.

Creating Accessible Content

Authoring Tools

A wide range of authoring tools are available for creating e-learning content, and many of these allow authors to add accessibility into the content they create. The W3C's ATAG specification provides guidance for developers on creating authoring tools that can create accessible content and that can be used by people with disabilities. Information about an authoring tool's accessibility features can usually be found in its documentation, and often online either on the developer's website or on websites that offer secondary forms of documentation.

Software that might be considered an authoring tool would include:

- HTML editors used to create web pages.
- Content manage systems (CMS) used to create and manage websites.
- Learning content management systems (LCMS) used for developing and storing e-learning content.
- Learning management systems (LMS) used to manage online courses.
- Blogs, Wikis, and Forums used for web-based documentation and communication.
- Multimedia production editors used to create video content.

- Flash editors used to create Adobe Flash-based content.
- Math editors used to create mathematical expressions.

To assess a tool's ability to create accessible e-learning content, see the Checklist of Checkpoints for Authoring Tool Accessibility Guidelines 2.0 [8][9]. Some of the main features to look for include:

Level A (must haves):

- The ability to add text alternatives for images, video, and audio.
- The ability to create document structure through headings, lists, and table headers.
- The ability to generate valid HTML markup.
- The ability to preserve HTML and accessibility features in existing content edited with the tool, or to inform the author when unrecognized markup is encountered, providing an option to preserve or replace existing content with markup generated by the tool.

Level AA (should haves):

- Provides adequate documentation on accessibility features.
- Makes the most accessible authoring options the default.
- Prompts content authors for accessibility information when it has not been provided.
- Offers accessibility checking and repair functions.

Multimedia Production

Multimedia content, more specifically video-based content, deserves special mention when it comes to producing or authoring video content that will be accessible to everyone because it presents a number of potential barriers when alternatives are not provided.

All video that includes meaningful sounds or spoken words requires **captioning** to reproduce those sounds or speech in a readable text form, accommodating those who cannot hear the audio track of a video. Captions are an AODA Level A requirement; as such, they must be part of the production of your video for e-learning content. There are a variety of tools for generating standard caption tracks, including the free, web-based Amara Caption Editor [10] that creates standard caption files, which can be added to a video track.

In addition to captions, the video should have **audio description,** as previously mentioned, to describe such things as meaningful actions, activities, and context that one may not be able to glean by merely listening to its audio track. Audio description and **extended audio description** are provided to help learners who are blind comprehend the meaningful visual content in a video. Audio descriptions are added into a track in the video when there are gaps in the speech output. Extended audio descriptions are used in video where there are few or no gaps in the speech; in the latter case, the video is paused momentarily when the audio description plays.

As you might imagine, producing a collection of short audio descriptions and weaving them into a video can require significant effort. While authors are encouraged to include audio description

with their video content, this is one of the exceptions in the AODA requirements. Though a Level AA requirement in WCAG, the AODA has removed it as a requirement for private organizations, understanding the potential burden it may create for video producers. Requiring it has the potential of discouraging the use of video in learning materials for fear of violating this particular WCAG requirement, counter to the intent of the AODA. That said, however, there are tools available now, such as CapScribe [11], that allow authors to embed descriptions in a text caption track, and have them read aloud using text-to-speech technology, removing the need to produce and integrate audio snippets typically used to create audio description.

Mathematical Notations

Mathematical notation is another form of e-learning content that poses accessibility challenge. There are two primary technologies for producing math notation: LaTeX [13] or MathML[14].

LaTeX, first developed in 1985, is a document typesetting system for producing a range of text, numeric, and symbol format manipulations. It is broadly supported across a wide range of authoring tools. When LaTeX markup is passed into a Latex processor, it generates math notation (or other text) as an image, which does not by default include a text alternative. Some systems will use the original markup as a text alternative for the generated image that can be read by assistive technologies: however, understanding what is read still requires knowledge of LaTeX on the part of the listener, and even then it can be difficult to comprehend.

MathML, on the other hand, is a relatively new technology – an emerging W3C standard [15] – that defines math notation using an XML-based markup language. XML, which is similar to HTML, is used to format and describe data. MathML is intended to support accessibility needs. Various projects have produced MathML authoring tools and readers that create math notation though WYSIWYG (what you see is what you get) type editor and convert the generated XML markup to output speech or Braille in an understandable form. MathML is becoming more broadly supported through various applications [16] that can be used to create and interpret MathML markup. It is currently supported in Firefox and Chrome, and reads well using the ChromeVox screen reader, though reads less effectively using the JAWS screen reader.

Document Accessibility

In addition to web content, office documents such as MS Word, PowerPoint, Excel and Adobe PDF must be created in an accessible fashion. Fortunately, many of the principles that apply to developing accessible web content also apply to creating accessible office documents. This includes the use of text alternatives for images in documents; structural elements such as headings, lists, and table headers that organize content; and high-contrast colours.

For a comprehensive online resource that describes the features available in various document authoring tools, refer to the Accessible Digital Office Documents (ADOD) Project [17].

Accessible Online Assessments

There are several accessibility considerations to weigh when developing online tests. Perhaps the most critical is that of **timing**. Learners with certain disabilities often require more time to complete an assessment, due to taking longer to read or process information, or perhaps having to navigate test items with assistive technology. The time allowed for typical test-takers might be extended 1.5 to 2 times for these learners, depending on the effect of her/his disability on completing the test. Generally the institution's disability services office will assess the student to determine an equitable time extension that will give the student enough time to complete the test but, at the same time, not give her/him an advantage over other students.

The accessibility of **visual elements** in tests can also be a challenge for authors of tests. It can be difficult to create an equivalent text or audio version of a test item that measures the same understanding or skill as the visual version of an item (for example, identifying a painter, given a painting), while at the same time maintaining the validity and reliability of an assessment. Standards such as the **IMS Accessible Portable Item Protocol** (APIP) [18] can help authors to better understand the complexities of developing accessible tests, as well as provide the tools they need to create assessments that work for everyone.

APIP is an AccessForAll extension of the IMS Question Test Interoperability (QTI) [19] specification (see Accessibility in e-Learning: Standards and Specifications). Designed for test-system developers and test- item authors, QTI produces tests and test questions that can be packaged and transferred between systems. The AccessForAll extension provides a means for developing tests that can be packaged along with their accessibility information.

Testing Web Content Accessibility

Content authors and e-learning instructors can use a number of strategies to ensure that their e-learning materials will be accessible to all learners.

Automated Accessibility Testing Tools

Although there are many automated accessibility testing tools that work well as a starting point for assessing the accessibility of your web content, none of them will identify all potential accessibility problems. Issues associated with meaning or understanding still fall to the sound judgement of a human being. For example, only a human being can determine if an image's text alternative accurately captures the meaning in the image for a person who is blind. Automated checkers have not advanced (and may not for some time) to the point where they can identify and address all potential accessibility problems.

The **AChecker Web Accessibility Checker** [20], developed at the Inclusive Design Research Centre at OCAD University in Toronto, evaluates the accessibility of e-learning and other web content. Designed for content authors and e-learning instructors, you can enter the web address (URL) of your online course, upload a file or paste HTML content into the checker to have its

accessibility evaluated against a range of international accessibility standards. The checker will generate a listing of known, likely, and potential accessibility issues, as well as providing detailed descriptions of potential strategies that can be used to remedy these issues.

Manual Accessibility Tests

In addition to automated accessibility testing, content authors can use various manual tests to assess the accessibility of their content:

1. Use your system's **Select All** command (e.g., ctrl + a) and see what is and is not selected. Items that are selected will be accessible, while items that are not selected may not be.

2. Place your mouse cursor in the location field of your web browser, then **press the Tab key repeatedly** until it enters the content into the browser's window, and continue to follow the cursor's path from there. Can you easily see the cursor as it moves through the content on the page? If not, the content may be difficult to use for some people. Do all the functional elements on the page such as links, buttons, and form fields receive focus while you move through the page? If not, the items that did not receive focus may not be accessible.

3. In your browser's settings, **turn images off** and view the content. Are you still able to understand the page's content through the text alternatives that display in place of the images, or do you see blank spaces where the images used to be? If the latter is the case, then the content of these images will not be accessible to learners who are blind.

Screen Reader Testing

Another tool that content authors and e-learning instructors should have in their accessibility toolkit is a screen reader. A screen reader enables a person who is blind to access content on the web. One of the most popular screen readers is **JAWS** (Job Access for Windows) developed by Freedom Scientific [21]. A licence for JAWS falls in the $1,000 range, typically more than most content developers want to spend. However, you can download a demo version of JAWS from the developer's website that will run for 40 minutes at a time, after which you would need to reboot your computer if you wanted to run it again for 40 minutes, and so on.

There are also several free, Open Source screen readers that provide much of the functionality found in JAWS. **NVDA** [22] is a free Open Source screen reader for windows. **ChromeVox** [23] is another; it easily plugs into the Chrome web browser and works well for testing access to web content presented in Chrome across all computer platforms. For ease of use, ChromeVox is highly recommended as an accessibility testing tool, although it is missing a few features found in JAWS and NVDA, such as the ability to read table headers when navigating through tabular content. For details on installing and configuring the ChromeVox screen reader, see Accessibility in e-Learning: ChromeVox Screen Reader Setup.

User-Testing

If you have a test audience, such as first-year psychology students, or students who use assistive technologies to access the web, it is a good idea to have them review your content or assessments to help identify access issues. When user-testing, particularly with assistive technology users, it is important to assess their level of skill using both their AT and web technology to be sure problems they might encounter are not due to inexperience.

User-testing consists of a collection of common tasks that might be undertaken in a web-based course, typically offered in an LMS. For example, a student might be asked to:

- Read messages in a discussion forum and post a reply.
- Open the assignment dropbox and submit a file.
- Take a short quiz and retrieve the score once complete.
- Using the system's search feature find information on a particular topic.

Social Networking and Social Media

Facebook

Instructors and students often make use of Facebook as a learning tool to supplement in-class or other online activities. How accessible is Facebook to people with disabilities? Facebook has a team of developers and a Facebook site dedicated to Facebook accessibility [24].

Some accessibility features you will find on Facebook include:

- ARIA landmarks that provide within-page navigation.
- Effective page structuring with heading markup.
- Keyboard shortcuts for various tools.
- Sizing that uses relative measures for improved zooming capabilities.
- Well-documented accessibility features.

Google+ etc.

Google has become a leader in promoting access for people with disabilities, and Google+ follows suit. Google offers range of documentation that describes accessibility features across its full collection of products [25].

While most video-conferencing systems have limited access for people using assistive technologies, Google Hangouts are generally accessible to users who are both blind and deaf, with just a few minor access issues, such as unlabelled form elements in the settings area. Some accessibility features on Google+ include:

- ARIA landmarks in Google Hangouts (though not for the rest of the Google+ interface).
- Effective page structuring with heading markup.
- Keyboard shortcuts for navigating Google Hangouts.
- Well-documented accessibility features.

Twitter

Like Google and Facebook, Twitter has also extended effort into ensuring access for people with disabilities over the past year [27]. Although there are still some aspects of Twitter that will challenge AT users, the user interface (UI) and service itself is relatively accessible. There are, however, many third-party, add-on tools that embed Twitter feeds into various web applications such as content managements systems and blogs, not all of which will be accessible.

Some accessibility features on Twitter include:

- ARIA landmarks that provide within-page navigation.
- Keyboard shortcuts for navigating through timelines (in Twitter, press "?" to list all shortcuts).
- Documentation through the Twitter Settings and Help menus.

YouTube

There are still access issues with the default Flash-based YouTube player, but there is also a more accessible HTML5 version available upon request, which users of screen reader should request and use. Although there is always room for improvement, YouTube's player provides a various features – including a caption editor and a tool for adding text annotations – that generally allow video producers to create accessible video content. Another handy feature when captions are added to a YouTube video is the Transcript feature (described in this paper's Hearing Impairments section), which presents captions as a scrollable list that allows the user to follow line-by-line as each is highlighted when the matching caption appears in the video player.

Other accessibility features on YouTube include:

- Keyboard shortcut for the video player (though not for video embedded external to YouTube).

On the downside, it is difficult to find YouTube accessibility documentation. While researching this paper, we found a number of external sites that document YouTube keyboard shortcuts [28][29], but not documentation originating from YouTube itself. Screen reader users may prefer to use the "Accessible Interface to YouTube" [30], which reduces much of the complexity of the YouTube interface; however, it does make use of the Flash-based YouTube player, employing a number of shortcut keys to operate the player through external scripting.

LMS Accessibility

The primary tool for creating and delivering e-learning is usually a learning management system (LMS). Some common examples include Blackboard, Desire2Learn, ATutor, and Moodle. E-learning content will only be as accessible as the LMS in which it is presented.

Not all systems, however, are equally accessible. When selecting an LMS, it is important to understand the common issues that prevent some students with disabilities from participating

fully. Seek out and evaluate an LMS's accessibility documentation to better understand its features. Look for third-party accessibility reviews if you can find them, and be wary of the statements of the developers or vendors of a particular LMS as your sole source for information. Use your experiences and that of colleagues, the tips and tools contained here, and a review of the e-learning environment you are choosing or currently using to help determine a system's accessibility to all your potential learners, authors, and instructors. This section highlights some areas for your consideration when assessing the accessibility of your online courses or LMS.

Keyboard Access

For many learners, keyboard access is the only way they can navigate web content. For example, a person who is blind will not likely ever use a mouse because s/he cannot see the location of the mouse pointer on the screen. Some quick manual checks (such as those described in this paper's Testing Web Content Accessibility section) can be used to determine whether all functional elements in your web-based courses are keyboard operable. If you find elements you cannot access with a keyboard, or are unable to operate, look for alternate ways to access these features; if none are available, you should avoid use of these features.

Within-Page Navigation

The students who rely on a keyboard to access web content will also need ways to navigate within pages of the LMS. Strategies that can be used to provide within-page navigation include using HTML heading markup (to represent topic structure and not to produce large bold font); adding bypass links to skip over repetitive navigation elements or menus; and using ARIA landmarks to identify key navigation points in the LMS's interface.

Building on the instructions, stated earlier, for setting up the ChromeVox screen reader with the Chrome web browser, you can list the headings on a page by pressing the assigned ChromeVox modifier key + l + h (e.g., alt + l + h), then use the arrow keys to move between headings. Similarly, you can press the modifier key + l + semi colon (alt + l + ;) to list the ARIA landmarks on the page (e.g., set roles for regions such as banner, navigation, and search).

Some LMSs use bypass links to provide within-page navigation. They are normally found in the top left corner of a page, and are often not visible but accessible when using a screen reader. When followed, these links lead to key navigation points in the interface of the LMS, and reposition the page so the user can continue navigating from that point. If you place your mouse cursor in the location bar of your browser and press the Tab key repeatedly until the cursor enters the content of the browser window, you can look at the status bar at the bottom of the browser to check if the bypass links are present. You will see something such as "#content" attached to the end of the web address (URL) that appears in the status bar. Notice the hash mark "#" that indicates a link to a location within the current page. In some cases, bypass links will become visible when they receive keyboard focus, depending on how they were created. If your LMS does not provide at least one of these strategies to allow navigation within pages, it will be difficult to use for some students using assistive technologies.

Logical Tab Order

For keyboard users, the Tab key is once again the main way they will move through web content, pressing the key repeatedly to move through links, buttons, and form elements. A common problem occurs when a particular link or button is clicked and a new window or perhaps a dialog box opens that is not immediately keyboard accessible. Frequently after opening a dialog box, the user has to press the Tab key repeatedly to reach the end of the content underneath the box before reaching the content of the dialog itself. When the dialog box is opened, the cursor should be focused in the box; when the box is closed, the cursor should return to the location from which the dialog box was originally opened.

Another common problem occurs when the cursor's path through the interface is not the usual left to right and top to bottom. For those who may have magnified the screen and are navigating by keyboard, the cursor can often move out of view. In this instance, when a system follows a logical tab order, the user can usually predict the cursor's location; however, when the system does not follow a logical tab order, the user will have experience difficulty locating the cursor.

Visible Focus

Many older learners or learners with poor vision will need to magnify the screen multiple times to increase the size of text in order to be able to read. As a result, some elements may be "pushed" off the visible screen. If these learners are using a keyboard to navigate and the focus leaves the visible area, they will often scroll to bring hidden areas back into view and search for the cursor's location on the screen; if they cannot easily find the cursor, navigating web content increases in difficulty. When navigating by keyboard, the process of following the cursor's location through elements on a web page should always be made as easy as possible.

The standard focus indicator is a dashed border that appears when an element receives keyboard focus; while it is an acceptable focus indicator, it can be difficult to follow, even for learners with good vision. To accommodate learners with poor vision, you should use a more enhanced focus indicator that, for example, would apply a background color or perhaps a more prominent border when elements receive focus.

Labelled Forms with Instructions

When using a screen reader to navigate through forms, the text nearest to the form element is usually read if form elements have not been explicitly labelled using the HTML Label element. This, however, is not always sufficient. For example, if forms are setup in a table, with text in one cell and input fields in another, a screen reader will often announce "edit text" or something similar, and not indicate the expected input described in the nearby text – unable to make the association between the input field and the visually apparent, though separated label in an adjacent table cell. Using the HTML Label element helps to ensure that form elements are properly described, regardless of where their text labels might be located.

Another accessibility feature of the Label element is that it makes the label clickable. For some people with motor impairments, positioning a mouse pointer over a tiny form element can be difficult. Using a Label element provides a larger target area for the user to click on to activate these form elements.

You can test for the presence of the Label element by clicking on the text next to a form field; if the form field becomes active, the Label element is present.

In some cases, it would be helpful to provide text instructions in addition to a Label that describes the expected input in more detail. A login name and password field in a registration form is an example where additional information might be associated with form elements. In the form's input element, using the HTML "title" attribute, you might add title="password must be at least eight characters made up of numbers, letters, and special characters." Although this information is often found near a password field, it can go unnoticed by AT users. Explicitly associating the information with the form input element using the title attribute ensures that all users have access to the same information.

You can test for the existence of title text with most web browsers by holding a mouse pointer over a form element and see if a small tooltip appears.

Accessible Feedback

Feedback and error messages have been a challenge for developers to create accessibly and a challenge for AT users to access. But with ARIA, feedback can be made accessible by simply adding a role="alert" attribute to the HTML element containing the feedback. The ARIA "alert" role is a type of live region that, when updated, announces changes to assistive technologies. In addition to presenting accessible error feedback, it also provides accessible success feedback, letting the user know that s/he has successfully completed an action.

Without this feedback, AT users might not know whether errors have occurred or, conversely, whether they have successfully completed a form. One past strategy to make feedback more accessible was to locate it consistently at the top of the main content area, immediately following the location of the #content bypass link (see this paper's *Within-Page Navigation* section). Assistive technology users could then predict where the feedback would appear and jump directly to it when a page loads; however, this strategy does not work when feedback is dynamically injected into a web page without reloading the page.

You can test for the existence of ARIA-enabled feedback using Chrome and ChromeVox. In your LMS, generate an error message by, for example, filling out a form in the system and leaving out a required field. If the feedback message is ARIA-enabled, ChromeVox will announce the error automatically.

Personal Preferences

With the introduction of AccessForAll, there is now a standardized way to define personal preferences that allow learners to modify the environment and the content to their specific needs. (For more information on AccesForAll, see Accessibility in e-Learning: Standards and Specifications.) Typical personalizations might include selecting a high-contrast display for a person with limited vision, replacing visual elements with text alternatives for a person who is blind, or adding captions to video for a person with a hearing impairment. Most LMSs have some personal preference settings, though AccessForAll support is still fairly new and not yet broadly supported.

When evaluating an LMS, evaluate its preference settings. There should be a range of settings to allow learners to personalize the environment to suit their needs, including:

- Font types and font sizes.
- Font colour and background colour.
- Navigation elements such as breadcrumbs, sequence links, and table of contents.
- Topic numbering to organize content numerically.
- Form focus for pages where the main content is a form.
- Ability to turn context sensitive help on and off.
- Choice of themes.
- Preferred content type (see AccessForAll).

Accessible Authoring

The ATAG guidelines (see Accessibility in e-Learning: Standards and Specifications) provide a standard way to implement authoring tools that will produce accessible content. The authoring tool itself must first be accessible, conforming with the WCAG guidelines. It must provide a way to add accessibility features, such as the ability to affix text alternatives when inserting images into content. The tool should also alert authors if they are creating content that may not be accessible. For example, if an image is saved without a text alternative, the authoring tool should prompt the author to include one.

In a typical LMS, content can be created through a number of features, such as a content editor for creating learning materials, a forum for communication, a news feature for posting announcements, and assessment tools for producing tests and test questions. All these tools, and others, must be able to produce accessible content.

To assess the authoring tools in an LMS, first determine if all the elements of the editor are keyboard accessible. Place the cursor's focus on an element in the editor, and then press the Tab key and follow the cursor's path; it should position itself in the toolbars across the top. Press the Tab key repeatedly to be sure that all the buttons receive focus, and press the Enter key to be sure that each button is operable (see Figure 3 for a typical editor interface).

If you are using ChromeVox, you can listen to its output as you navigate through the buttons to determine whether the text associated with each is meaningful. It is also helpful for AT users if they have the option to choose between an advanced editor with all features enabled, and a basic editor with just the main editor buttons turned on.

You should be able to navigate easily between the toolbar buttons at the top, the editor window below the buttons, and the Path bar at the bottom of the editor. The Path bar shows where the cursor is located within the HTML elements in the content of the editor window, so you should be able to select any element to reposition the cursor into the editor window where that particular element appears.

Also look for the existence of a field that allows the author to include a text alternative when adding an image. If this field exists, it should prompt the author to attach text if the field is left empty when saving an image.

If the editor provides the ability to author math equations, ask yourself this: are these equations generated using MathML or, if they are generated using LaTeX, is there an ability to include a text description? You can test the accessibility of the equations generated with an LMS math editor using ChromeVox. After creating and saving a math equation, use the ChromeVox modifier key + down arrow to move focus to the equation, and then listen to the output it generates.

Another thing to watch for is whether the cursor becomes trapped inside the editor, which would prevent the author who is only using a keyboard from accessing the content that follows the editor.

The editing tool should also provide a way to check the accessibility of the content being authored. The AChecker Web Accessibility Checker (described in this paper's Testing Web Content Accessibility section) is a web-based tool that web developers can add to an editor to assess the accessibility of the content they are creating.

Figure 3 is an example of the TinyMCE WYSIWYG (what you see is what you get) HTML editor that provides all these features, allowing AT users to author their own content and check its accessibility with AChecker.

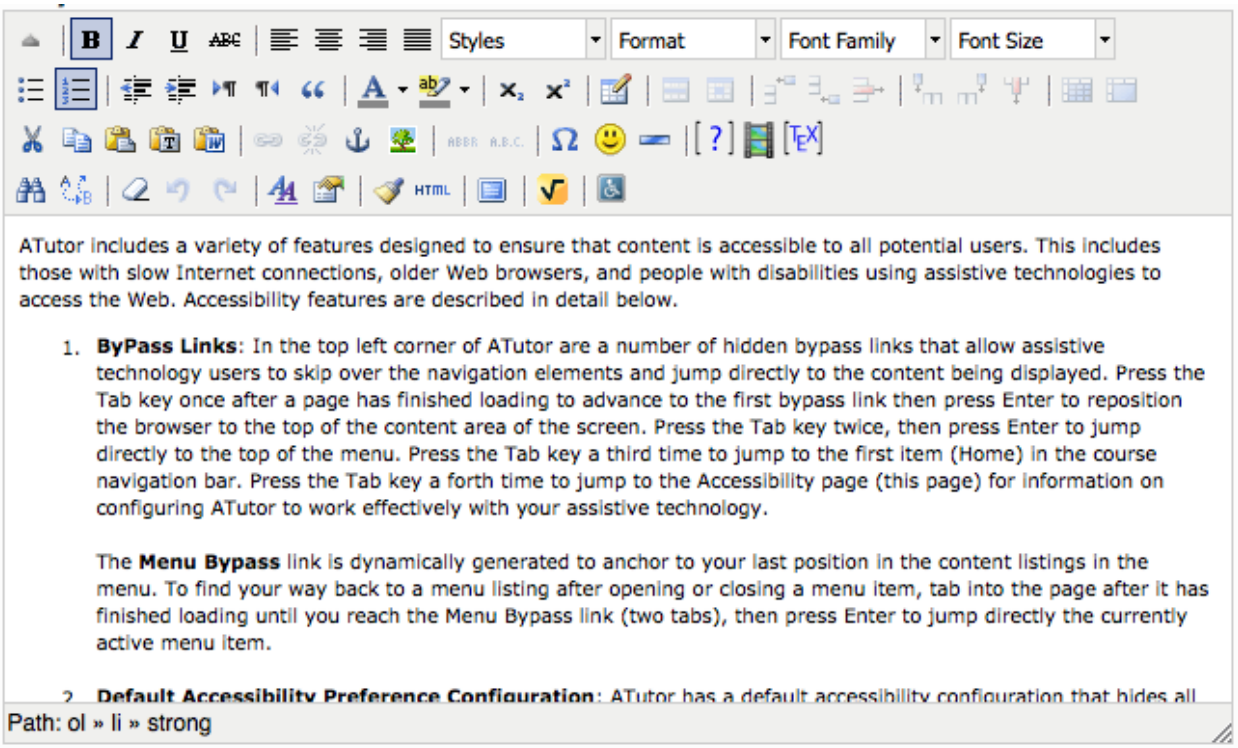

Figure 3: The TinyMCE WYSIWYG content editor, with tool button bars across the top, editor window in the centre, and element Path navigation bar at the bottom. The AChecker add-on tests the accessibility of the content (last button to the right, bottom row). Users can navigate between the buttons, editor window, and Path using the q, z, and x shortcut keys, respectively. (This image is unavailable in French.)

Timing

Learners with certain disabilities may take longer to complete tasks, due to requiring more time to read or process information, or perhaps having to navigate test items with assistive technology. Consequently, time limits set for typical learners may not be adequate for these learners. One example of a time limit is a session timeout, which is present to log out a student after a relatively short period of inactivity. For security purposes, this action is valuable; for accessibility, however, it can be problematic.

To test an LMS's timeout function, log into the system and let it sit idle. Before the timeout occurs, the LMS should generate a warning, perhaps in a pop-up dialog box that allows the learner to extend the current session and then resets the timeout period. If the learner does not acknowledge the warning within a few minutes, only then should the student be timed out, and her/his session ended.

Features such as timed tests can also pose a challenge for some learners with disabilities, who may need more time to complete tests than typical learners. Review the test authoring tools in an LMS and, in particular, look for settings that will allow you to extend time where needed.

Your institution’s disability services office will determine the actual time extension needed to accommodate individual students based on their disability.

Accessibility Statement

There are many ways to add accessibility enhancements into a learning environment, and learners with disabilities should be made aware of these features. An LMS should provide an easily accessible list of not only the system’s accessibility features but also the elements that may present barriers. An LMS can have some inaccessible elements if no current technology exists that could make these elements accessible, if alternatives are provided, and if these issues are acknowledged and described up front.

MOOC Accessibility

Massive open online courses, or MOOCs, are neither accessible nor inaccessible on their own. Rather, the accessibility of a MOOC is determined by its LMS or delivery system and the accessibility that the author has built into its e-learning content.

The Future of Accessibility

GPII/Cloud4All

The Global Public Inclusive Infrastructure (GPII) [31] is an ambitious international project that will see the broad implementation of personal preference matching with adaptable technologies that reconfigure themselves to an individual’s needs. For example, a person with poor vision may define a personal preference for high-contrast, large font. This preference, and others, would be stored in a central networked location, or perhaps on a smart card or a memory stick. When this person logs into an institution's LMS, it would recognize her/his need for high-contrast, large font, and the system would reconfigure itself. That person might later access a bank machine by swiping a bank card, and the system would once again recognize her/his need for high-contrast, large font, and the screen would reconfigure itself. If s/he were buying train tickets at a kiosk using a credit card, the same thing would happen again, and so on.

GPII collaborators include the Canadian, U.S. and European Union governments, Trace Centre in Wisconsin, OCAD University in Toronto, Andrew W. Mellon Foundation, William and Flora Hewlett Foundation, IBM, Microsoft, Mozilla, Serotek, and a number of individual contributors.

Conclusion

Accessible e-learning ensures that all potential students and instructors have a barrier-free opportunity to participate in web-based learning activities. While accommodating the needs of people with disabilities in teaching and learning environments is a requirement of the AODA, it also serves as a best practice for creating and delivering e-learning opportunities that are accessible to everyone.

The AODA adopts the W3C WCAG 2.0 accessibility requirements as a means to standardize the creation of accessible web content and, in so doing, e-learning experiences for everyone. Ensuring that e-learning opportunities meet the AODA requirements involves two basic principles: ensuring that content is provided in multiple formats in order to be accessible by **multiple senses** (such as text alternatives for images, and captions for a video's audio track), and ensuring that functionality is not tied to any one single input device (in other words, operates with both a **mouse and a keyboard**). The application of these two principles will address the vast majority of potential barriers that might arise in e-learning.

If you have not already done so, undertake an accessibility review of your current or future LMS, to ensure that your institution's system supports the efforts of your educators, content authors, and students in their e-learning activities. As a reminder, you can use the Tab key to test keyboard accessibility; turn images off in your browser to see if there are text alternatives; and/or use a screen reader such as ChromeVox to navigate through your LMS to identify if there are any barriers. If you find barriers, bring them to the attention of your institution's IT or e-learning staff, or the LMS developer directly.

As instructor, content author, or e-learning specialist, increasing your understanding of accessibility and its requirements will help you to create, lead, and deliver accessible learning activities on the web that are inclusive to everyone.

Checklist of Best Practices

If you are an educator creating e-learning content, you are responsible for making content perceivable and understandable. If you are an IT professional or web developer, you are largely responsible for making features operate with both a mouse and keyboard. If content or features in your institution's e-learning systems are not perceivable, operable, or understandable, try to fix them if you are able, or notify IT staff or the web developer to have them corrected.

1. Produce content that is **perceivable** through multiple senses:
    a. Use text descriptions for graphic content.
    b. Provide a transcript for audio content.
    c. Include captions and, where feasible, audio description for video content.
    d. Consider providing video, audio, or graphics that express equivalent meaning for text content.
    e. Use colours that contrast well.
    f. Use actual text rather than images with text embedded in them.
    g. Be sure you can follow the cursor easily when navigating with the Tab key.
    h. Do not use colour on its own to represent meaning.

2. Ensure that everything **operable** with a mouse is operable with a keyboard:
    a. Press the Tab key repeatedly to be sure anything that is clickable with a mouse can receive keyboard focus.
    b. Use your browser's "Select All" function and notice what does not get selected.
    c. If you have a rotating banner, make sure it can be stopped and rotated manually.
    d. Be sure video players and photo galleries work with both the mouse and keyboard.
    e. If a popup window or box opens, keyboard focus should be placed in the popup

3. Make content **understandable** to a broad audience:
    a. Write at a grade 9 or 10 reading level when your audience is the general public.
    b. Match link text, to the title and main heading of the page to which the link leads (for example, don't use different words such as "log in" for the link text and then "registration" for the page title or main heading).
    c. Use link text that describes the destination or function of a link (not links such as "click here" or "learn more" that provide no useful information).
    d. Use the same link text for two links on a page if they both lead to the same place.
    e. Make sure that all pages have unique meaningful page titles (see your browser's title bar above).
    f. Expand abbreviations and acronyms wherever they occur.
    g. Use regular-case text, and avoid using text that is all capitalized.
    h. Avoid using text that is justified.
    i. Click form labels to be sure that they activate the associated form input field.
    j. Ensure your LMS provides adequate error or success feedback after users who complete an action.
    k. Be consistent in the way you organize content across pages in your course

References

1. Learning Styles (VAK) http://en.wikipedia.org/wiki/Learning_styles#Neil_Fleming.27s_VAK.2FVARK_model

2. Multimodal Learning and the Quality of Intelligent Behaviour (Google Books), From Intelligence: Reconceptualization and Measurement (Chapter 5) http://books.google.ca/books?hl=en&lr=&id=KPm56N0T-_kC&oi=fnd&pg=PA57&dq=multimodal+learning+research+paper&ots=zaENeYZJDX&sig=yvfik5eFo4aW2i4711gkBElMYtk#v=onepage&q&f=false

3. Cognitive Disability (WebAIM) http://webaim.org/papers/cognitive/

4. W3C Web Content Accessibility Guidelines 2.0 http://www.w3.org/TR/WCAG20/

5. W3C Authoring Tool Accessibility Guidelines 2.0 http://www.w3.org/TR/ATAG20/

6. IMS Global Learning Consortium Accessibility (AccessForAll) http://www.imsglobal.org/accessibility/

7. Accessibility for Ontarians with Disabilities Act (AODA) http://www.e-laws.gov.on.ca/html/source/regs/english/2011/elaws_src_regs_r11191_e.htm

8. Checklist of Checkpoints for Authoring Tool Accessibility Guidelines 2.0 http://www.w3.org/TR/2004/WD-ATAG20-20041122/full-checklist.html

9. Checklist of Checkpoints for Authoring Tool Accessibility Guidelines 1.0 http://www.w3.org/TR/ATAG10/atag10-chktable.html

10. Amara Caption/Subtitle Editor http://www.amara.org

11. Capscribe http://www.inclusivemedia.ca/services/capscribe.shtml

12. ATutor Learning Management System http://www.atutor.ca

13. LaTeX – A document preparation system http://www.latex-project.org/

14. MathML http://www.w3.org/Math/

15. Mathematical Markup Language (MathML) Version 3 http://www.w3.org/TR/MathML3/

16. W3C MathML software list http://www.w3.org/Math/Software/

17. Accessible Digital Office Documents (ADOD) Project http://adod.idrc.ocad.ca/

18. IMS Accessible Portable Item Protocol http://www.imsglobal.org/apip/

19. IMS Question Test Interoperability Specification http://www.imsglobal.org/question/

20. AChecker Web Accessibility Checker http://achecker.ca/checker/index.php

21. JAWS Screen Reader (Freedom Scientific) http://www.freedomscientific.com/downloads/jaws/jaws-downloads.asp

22. NVDA Open Source Screen Reader for Windows http://www.nvaccess.org/

23. ChromeVox screen reader extension for the Chrome web browser http://www.chromevox.com/

24. Accessibility for People with Disabilities (Facebook) http://www.facebook.com/help/141636465971794

25. Using Google products: How to use accessibility features http://www.google.ca/accessibility/products/

26. Accessible Rich Internet Applications (ARIA 1.0) http://www.w3.org/TR/wai-aria/

27. Improving accessibility of twitter.com https://blog.twitter.com/2013/improving-accessibility-of-twittercom

28. Google YouTube Keyboard Shortcuts – Windows https://sites.google.com/a/umich.edu/going-google/accessibility/google-keyboard-shortcuts---YouTube

29. The Comprehensive Guideline to YouTube Player Keyboard Shortcuts http://www.makeuseof.com/tag/comprehensive-guide-YouTube-player-keyboard-shortcuts/

30. Accessible Interface to YouTube http://tube.majestyc.net/

31. Global Public Inclusive Infrastructure (GPII) http://gpii.net/

Appendix: Accessibility in e-Learning Companion Documents

Standards and Specifications

Accessible Content Authoring Practices

Easy-to-Use Resources

ChromeVox Screen Reader Setup

Turning Images Off in Web Browsers

Additional Resources and References