Evaluating Features for Promoting Accessible Content in Content Management Systems

HANNES WESTBERG
Abstract

As the web continues to evolve, so does our need for achieving an accessible web for people with disabilities. Content management systems (CMSs) have well observed accessibility problems with generated content, and in recent years, several features have been proposed in order to minimize or eliminate these problems. This study investigated CMSs in current use to find common accessibility problems and evaluated a set of features proposed by Acosta, T. et al. in 2018, targeting these problems.

The study initially found a general lack of information, guidance and technical support provided by CMSs to editors promoting the generation of accessible content. The results indicate that even editors highly aware of accessibility may not be able to create accessible content due to the limitations of their systems. The study also received positive feedback towards the evaluated features from professionals, indicating that the features are of practical value and may help the editor by minimizing or eliminating common accessibility problems in content generated through CMSs.
Sammanfattning

Webben fortsätter att utvecklas, och det gör också vårt behov av att göra webben tillgänglig för personer med funktionshinder. Innehållshanteringssystem (CMS) har flera kända tillgänglighetsproblem med dess genererande innehåll och under de senaste åren så har ett antal tillgänglighetsfunktioner föreslagits för att minimera eller eliminera dessa problem. Den här studien undersökte CMS som används idag för att hitta vanliga tillgänglighetsproblem och evaluerade en samling av föreslagna funktioner av Acosta, T. et al. som riktade sig mot dessa problem.

Studien fann i början en generell brist på information, vägledning och tekniskt stöd från CMS till redaktörer som främjar skapandet av tillgängligt innehåll. Resultaten visar att även redaktörer som är medvetna om tillgänglighet inte alltid har möjligheten att skapa tillgängligt innehåll på grund av begränsningarna i deras system. Studien fick också positiv återkoppling av de utvärderade funktionerna från yrkesverksamma inom området, vilket indikerar att funktionerna har ett praktiskt värde och kan hjälpa redaktören genom att minimera eller eliminera vanliga tillgänglighetsproblem i innehåll som skapats via CMS.
Evaluating Features for Promoting Accessible Content in Content Management Systems

Hannes Westberg
KTH, Royal Institute of Technology
Stockholm, Sweden
hannesve@kth.se

ABSTRACT
As the web continues to evolve, so does our need for achieving an accessible web for people with disabilities. Content management systems (CMSs) have well observed accessibility problems with generated content, and in recent years, several features have been proposed in order to minimize or eliminate these problems. This study investigated CMSs in current use to find common accessibility problems and evaluated a set of features proposed by Acosta, T. et al. in 2018, targeting these problems.

The study initially found a general lack of information, guidance and technical support provided by CMSs to editors promoting the generation of accessible content. The results indicate that even editors highly aware of accessibility may not be able to create accessible content due to the limitations of their systems. The study also received positive feedback towards the evaluated features from professionals, indicating that the features are of practical value and may help the editor by minimizing or eliminating common accessibility problems in content generated through CMSs.

Author Keywords
HCI; Accessibility; Accessible Content; Web Accessibility; Web Accessibility Evaluation; Accessibility Features; Content Management System;

INTRODUCTION
As the content available on the internet expands and the number of applications of the web continue to increase, so does the importance to keep improving the user experience on the internet. One area of user experience that has been in focus in recent decades is accessibility. The definitions of web accessibility can vary, but the following definition has been stated as the most preferred (WAI) [12]: “Web accessibility means that people with disabilities can use the Web. More specifically, Web accessibility means that people with disabilities can perceive, understand, navigate and interact with the Web, and that they can contribute to the Web.” Following this definition of accessibility on the web helps to build an inclusive community with users with special needs. However, even though a broad number of web accessibility standards exist today, many challenges remain for developers and designers to deliver accessible content [8, 10].

Content management systems (CMSs) have been in use since the early turn of the century and is currently implemented in an extensive number of websites mainly for their features to easily create, organize and administer web content without any deeper technical expertise or coding skills [5, 6]. Producing web content is today considered quite simple, with most editors delivering visual interfaces with previews of the final look of the content produced. As the process of content generation no longer requires a developer, users with low technical expertise or awareness of accessibility may alter the content of the page with a risk of a reduction of the accessibility [11].

In the past years, several studies have investigated how to promote the generation of accessible content using different methodologies. In 2018, a study by Acosta, T. et al. presented a method for accessibility assessment of online content editors consisting of 63 accessibility features that should be met by images, headings and tables in CMSs [1]. The implementation of these features might help in enabling, informing and promoting the editor to produce accessible content.

In this study, seven professional editors were interviewed and their respective currently worked on CMS was investigated in order to find common accessibility problems with respect to images and headings. Based on the result of the investigation, a set of the proposed features by Acosta, T. et al. was selected to be evaluated. Following the selection of the features, they were developed and implemented into an Umbraco CMS solution. The features were then evaluated through user tests with professional editors and web developers with working experience with accessibility. The following question was explored:

*How can a selected set of the proposed features of assessing the accessibility for images and headings be incorporated into a CMS in order to minimize potential accessibility problems with editor generated content?*

The result of this thesis work highlights current accessibility problems with content generated through CMSs and provides useful information on how these features can be applied in a commonly used CMS in order to minimize or eliminate common accessibility problems. It further contributes to the goal to achieve an accessible web by providing information and increasing awareness of accessibility inside CMSs, mainly supporting lay users.
BACKGROUND
The field of web development is rapidly evolving as new frameworks, systems and working methods become the new standard. The premillennial web was mostly characterized by manual approaches to maintaining its content, but since the early turn of the century, dynamic content systems have become in widespread use for its simplified maintenance functionalities [5]. The web may have benefited immensely from these systems, but the increasing use of dynamic content is also linked to several accessibility problems notoriously known for dynamic websites [9]. Web accessibility has since been a heavily researched area (see e.g. 1, 2, 3, 4, 8, 9, 10, 11, 13, 16) with the purpose of promoting the use of the internet for people with disabilities.

Accessibility Standards and Guidelines
Several web accessibility standards that guides editors and developers of producing accessible content have been proposed in recent years. Commonly referenced are the Web Content Accessibility Guidelines 2.0 (WCAG henceforth) and the Authoring Tool Accessibility Guidelines 2.0 (ATAG henceforth) [3, 4]. WCAG is primarily intended for content developers and categorizes websites based on its conformance level with recommended accessibility guidelines. While ATAG focuses mainly on the development of authoring tools that promotes the creation of accessible content. However, these guidelines are not complied by most developers due to lack of knowledge or ignorance on the subject [10].

A significant number of web users need a high accessibility in order to use the web successfully. Over 300 million people around the world are visually impaired and needs assistive technologies, such as screen readers that interprets and reads out the content on the web [9]. This group often confronts difficulties accessing the web. Even important emergency related information from county and municipal-level government sites, as seen in a study from 2014 [13]. The vast number of this user group highlights the importance of achieving an accessible web.

Content Management Systems
Content management systems provide several advantages to the maintenance of a website. One of these are enabling lay users without deeper technical expertise or coding skills to administrate and generate web content [5, 6]. Examples of commonly used systems now in use are WordPress, Joomla, Drupal, Umbraco and EPiServer. The typical content management team today consists of several roles: the editors, site planners, developers, administrators and stakeholders [7]. This study will focus on two of these roles, namely the developers and the editors. The developers are responsible for the development, installation, configuration, implementation and the CMS templating during the project. While the editors are responsible for creating, editing and managing the content within the CMS.

Accessibility within Content Management Systems
Both the editor and the developer have important tasks to do in order to ensure the possibility of creating accessible content and enforcing its practice. The developer needs to ensure that the developed CMS supports the production of accessible content from a technical standpoint. The editor must be aware of how to produce accessible content using the functionalities provided by the developer. If an accessibility error is made, it is often hard to pinpoint who is responsible due to the relationship between these two roles.

Many CMSs now enable people to completely build their own site using website templates, and thus no developer needs to be involved in the implementation process. These websites rely heavily on the built-in base functionality of the system, highlighting the importance of good base support for accessibility. In a study from 2011, six CMSs were selected and evaluated based on their default installation with respect to ATAG [16]. The results indicate a major lack in accessibility features as none of the CMSs achieved the basic “A” level of compliance regarding ATAG fulfillment. However, the study concluded that the CMSs could be configured so they can be accessible if necessary modifications to internal CMS classes and templates were made. The study also highlighted a possible remedy by including an accessibility plugin to the CMSs. But those solutions require deep knowledge regarding each CMS.

Web content production editors are today often based on the What You See Is What You Get (WYSIWYG) philosophy, with a visual editor previewing the final look of the content as you create it. This enables non-technical editors to efficiently customize parts of or even make considerable changes to the appearance and the structure of their website [11, 2]. Users with no technical expertise or knowledge of accessibility risks impacting the accessibility of the page when altering the layout or the content within a website. In 2012, a methodology was presented that allows identifying and solving accessibility problems with websites utilizing CMSs [2]. The proposed methodology was based on necessary steps to perform in order to provide a proper CMS environment for the management of accessible web pages. The methodology involved an iterative analysis of WCAG and ATAG [3, 4].

In 2015 a prototype WYSIWYG content editor was developed for evaluating the accessibility of images, headings, tables and language [11]. The content editor proposed to help lay users that produce web content to make web accessibility concepts more understandable with the incorporation of the WCAG 2.0 guidelines into the editor. While the accessibility warnings inside the editor were easy to understand and apply, they proved difficult for lay users to perceive.

Features Promoting Accessible Content in CMSs
In 2018, a study by Acosta, T. et al. presented a method consisting of 63 accessibility features that should be met by images, headings and tables in CMSs, contributing to the
creation of accessible content [1]. The set of features partly derived from previous research and was built upon during the study, using a deep analysis of WCAG 2.0 and ATAG 2.0 [3, 4]. The total of 63 features is split up by 21 features for images, 15 features for headings, 18 features for tables and 9 general features for assessing the accessibility of content in the CMS. The type of features presented ranges from enabling technical support in the CMS for accessible information, to include pedagogical information about accessibility to the editors.

The first 10 features for images manage the alternative text attribute. The alternative text attribute provides essential information for assistive technologies, such as screen readers, to describe the information provided by an image to users not able to acquire the information visually. The alternative text should properly convey the meaning of the image, and not necessarily describe its visual characteristics [14]. The alternative text attribute should always be included in images; however, the text can be empty. An image with an empty alternative text is called a decorative image and assistive technologies will simply ignore the image. Besides increasing the accessibility for images, the alternative text metadata also provide information to search engines, increasing the visibility of the content on the page on these platforms [17]. The features 11 - 16 for images manage the long description attribute, used to describe more complex images by providing an URL to a more informative page. The features 17 - 21 for images manage the title attribute, which often provides the user with additional information when hovering over an image.

The features for the headings vary in its targeted problems. One of the problems addressed in the features for headings is incorrect nesting orders. The nesting order for headings is important for users with assistive technologies to properly navigate through the content available on the page. The orders can vary from the most important heading order (H1), down to the least important (H6). If a heading has a lower order than the previous heading, it indicates a subsection of the previous heading. A heading order should never be skipped, e.g. go directly from order 1 to 3, to avoid navigational problems on the website using assistive technologies [15].

This study will attempt to evaluate a selected set of the proposed accessibility features within the areas of images and headings, integrated into a CMS. It will further try to evaluate if the selected set of features could be practically considered effective in minimizing or eliminating common accessibility problems related to their respective areas. Due to time and scope limitations of the project, only a specific set of the proposed features will be evaluated, focusing on the features considered most relevant from an initial selection study. The accessibility features for tables will not be investigated due to relative low occurrences on the web.

METHOD

In order to answer the research question, the project was split into three parts: the initial study, implementation and evaluation.

The focus of the initial study was to identify common accessibility problems and provide information on what features to prioritize in the selection for the evaluation. In the study, seven companies of differing business areas participated, with all utilizing a CMS for the generation and management of their website content. The key aspects investigated in the initial study were:

1. Investigating the knowledge of accessibility by the editors using their CMS
2. Acquiring an accessibility status of content generated by their CMS
3. Investigating the technical support for producing accessible content within the CMS
4. Investigating the pedagogical information within the CMS relevant to generating accessible content

This project was performed in collaboration with the web agency Winston. The need to investigate the technical aspects of the CMS limited the possible companies participating in the study to Winston’s customers, where they provided the possibility to access their systems. The sample of differing companies aimed at providing insight in diverse CMSs in current use to find common accessibility problems across them. In order to acquire the information listed above, the study included a smaller structured interview with one editor per company (participants P1-P7), and an investigation of the corresponding websites (websites W1-W7). Where P1 was an editor to the website W1, and so on. The participants were aged between 36 to 57 years and had between 5 to 21 years of working experience in the web. The websites used either EPiServer, Umbraco or WordPress as the CMS. For each website, ten pages was manually selected semi-randomly for evaluation in order to identify potential accessibility problems. The pages were respectively selected to act as a representative sample of the accessibility of the website content. To further clarify where the accessibility problems were created, each CMS was also analyzed based on the features by Acosta, T. et al [1]. From the results of the initial study, the set of features were selected for the evaluation (see next section).

A CMS solution was developed to implement the selected features, with the relevant functionalities needed to generate web content containing headings and images. The CMS was built in ASP.NET using Umbraco and enabled the editor to generate content in a Rich Text Editor interface.

The evaluation aimed at providing information how the selected features could be integrated into a CMS and if they can promote accessible content. The implementation of the proposed features was done in cooperation with Winston, and thus the participants of the evaluation study were limited to the employees of the company. In the evaluation, five
internal testers (tester T1-T5) within Winston with experiences working with accessibility were given specific tasks to complete in a think-aloud evaluation demonstrating the functionalities of the implemented features. Some of the tasks were clearly intended to generate content containing accessibility problems. The testers were aged between 26 to 52 years and had 2.5 to 20 years of working experience in the web. A concluding interview was conducted to obtain professional feedback on the features evaluated.

Selection of Proposed Features
The result of the initial study (presented in the result section) indicated a high frequency of problems with alternative texts for images and incorrect nesting of the heading order. And so, the features addressing these problems were prioritized in the selection. The study also found a low or no frequency of the long description and the title attribute for images. Thus, the features addressing problems with these attributes were not prioritized.

Based on the reasoning above, the following set of features were selected from Acosta, T. et al. to be integrated and evaluated: For images, features 1 - 10, and for headings, features 1 - 8. The current version of Umbraco (7.12.3) already comes with the features 1, 2, 10 for the images and features 1, 2, 3, 7 for the headings. Thus, the focus of the implementation was to include the features 3 - 9 for the images and the features 4 - 6 & 8 for the headings.

Features for images
1. Allows inserting alternative text from the user interface, i.e. <img> alt attribute
2. Allows editing of alternative text from the user interface
3. The name of the input field for the alternative text represents its function
4. Provides help about how to use the alternative text to improve accessibility
5. Warns if alternative text has not been provided
6. Gives an indication that the lack of alternative text produces a decorative image, it is not accessible for assistive technology users, and the attribute Alt = "" is included in the HTML code
7. Warns about the possibility that alternative text does not provide the same content of information transmitted by the image
8. Does not generate alternative text automatically
9. Gives indication after the insertion that the alternative text should properly describe the image
10. Keeps the alternative text associated with an image when it was added to the media library and allows editing it.

Features for headings
1. Allows assigning the heading from the user interface
2. Allow to edit the heading from the user interface
3. Allows changing the level of heading from the user interface
4. Provides help about how to use the headings to improve accessibility, and that the heading should describe the subject or purpose being treated in the content
5. Verifying it’s a correct nesting of headings
6. Displays an error message in case of failures nesting of heading
7. Warns if a heading is not provided
8. Use heading for the structure of the page in the template

Implementation of the Missing Selected Features
The features selected were implemented as two custom functions integrated into the Umbraco CMS.

Alternative text property
The features for the images were integrated into a custom property field that managed the alternative text for inserted images into articles. The field was displayed whenever the user added an image to an article inside the CMS (please see figure 1).

Alternative Text
The alternative text should properly describe the image so that the content can be accessed with assistive technology users. However, the possibility exists that alternative text does not provide the same content of information transmitted by the image.

Figure 1 - displays the editor view for the custom function managing the alternative text

When the user first adds an image to the article a warning is initially displayed, stating that no alternative text has been provided. The user then has a choice to either start typing an alternative text in a text field or click on a checkbox stating that it’s a decorative image. Clicking the checkbox disables the text field. Then, depending on the action of the user, additional information is presented to provide useful feedback on the current status of the image.

If an alternative text is provided, the following information is displayed: “The alternative text should properly describe the image so that the content can be accessed by assistive technology users. However, the possibility exists that the alternative text does not provide the same content of information transmitted by the image” If the checkbox is checked, the following information is displayed: “This is a decorative image, and is not accessible for assistive technology users” At the bottom of the property field is a link to an instruction on how to use the alternative text provided by WCAG [14]. The function was constructed so no matter the user input, the image would always result in an accessible image for screen readers (please see figure 2).
This first described function integrated features 1 - 10 for images.

![Diagram showing function states of alternative text](image1)

Figure 2 - displays the function states of the alternative text function

Accessibility Helper

The second developed function integrated the features for the headings into a plugin for the Rich Text Editor (RTE) inside the CMS. The plugin was called Accessibility Helper which provided feedback to the user if the headings were correctly nested inside the RTE. The plugin was activated by pressing a dedicated button in the toolbar of the RTE (please see figure 3).

When activated, the plugin searched for all heading instances in the text and evaluated if the heading nesting was correct. Since the CMS automatically generated the page title as the main heading (H1), the plugin evaluated the contents of the RTE to be nested using headings of order 2 and lower.

![Accessibility Helper button in the RTE](image2)

Figure 3 - displays the Accessibility Helper button in the RTE

Depending on the result of the heading search, different information was displayed in a popup window (please see figure 4).

![Accessibility Helper popup](image3)

**The text does not have a correct nesting of headings.**

Please adjust the headings to never skip a heading order, and follow the structure H2 - H3 - H4

**The following headings have the wrong order:**

H4 - Second subheading for the article

For more information, please visit: [WAI Heading Tutorials](#)

![Search result with developed plugin](image4)

Figure 4 - displays the result of a search with the developed plugin

If the first subheading in the RTE wasn’t a heading of order 2 (H2), the plugin would notify the editor to change it to order 2. Since the main heading always was included on the title of the page, the first subheading in the RTE required to be of order 2. If the nesting was incorrect, i.e. a heading of order 2 was directly followed by a heading of order 4 (H4), the plugin would let the user know and indicate which headings that were incorrectly nested. The text in the popup would say “The text does not have a correct nesting of headings. Please adjust the headings to never skip a heading order and to follow the structure H2 - H3 - H4 (see figure 4)”. Otherwise, if the nesting were correct, the plugin also stated that they were correct. At the bottom of the popup
window the user could find a link to additional information about how to use headings, provided by WAI [15]. The developed plugin implemented the features 1 - 8 for headings.

RESULTS

The Initial Study

Interviews with the editors

All the participants had heard of the term accessibility and accessible content previously. Three participants (P1, P2, P6) said to have extensive experience working with accessibility while the rest (P3, P4, P5, P7) only said to be aware of the area. When asked to what extent their focus is on the creation of accessible content today, three participants (P2, P3, P6) answered a high focus. The remaining participants (P1, P4, P5, P7) answered low focus or no focus at all. P1 expressed having a high focus on accessibility in a previous job “I experienced it sometimes to be quite limiting actually”.

When asked to what extent their current CMS informs them about accessible content, all participants answered none. P2 expressed: “I don’t think the CMS informs me at all. It would be great to get information about - if there is an H1, an H2 [...] if an image is missing an alternative text [...] I don’t think we get a reminder at all, if it’s accessible. Spelling correction might be the only one”.

Concerning to what extent their current CMS solution guides them in creating accessible content, all the participants once again answered not at all. Both P4 and P7 highlighted that it was not a requirement during the development of the site, and that was the reason for the lacking pedagogical features. P7 expressed: “No, it’s the same here. It could be improved more in that regard. But it’s not anything we prioritized when we created the CMS, not any of them”. P4 expressed: “We didn’t exactly demand it. We didn’t have that as a parameter when we designed the site so even if it existed, it’s not sure that we would have enabled the feature [...] But it’s not anything the CMS does automatically”.

When asked about to what extent their CMS technically supports the creation of accessible content from the editor’s point of view, P1 answered: “I don’t see that the CMS limits us in any way. [...] I think it’s okay, it enables possibilities”.

Five participants (P2, P3, P5 - P7) answered low support. P3 expressed: “If we talk about accessibility in general, then the images should be in the correct sizes. I think the system should react to that and make large images smaller [...] There is much to be desired in such functionality”.

P4 had trouble answering due to low awareness.

Six of the participants (P1-P5, P7) were not aware of any specific functionalities that facilitated the creation of accessible content within their CMS. P6 were aware of several functionalities and answered: “Small things like how we handle the headings, texts and excerpts. There are separated sections and enough visual contrasts. Yes, I’m quite aware, I think. [...] There is support for me as an editor”.

Three of the participants (P3, P5, P6) answered that their CMS system ask them to add accessible information into inserted images, such as an alternative text. P1, P2 and P7 had possibilities of adding additional information into an image but was not required to do so. P2 expressed: “It does not ask, but I have the possibility to add additional information. It’s not a mandatory field, that would probably be better. [...] So I guess that the image name will be used as an alternative text, and that is not always correct, or that it provides an informative description.” P4 hadn’t noticed any way of providing additional information when attaching an image to a content.

All participants were aware of subheadings (heading tags H2-H6) and used them in their work. Four participants (P1, P2, P3, P7) explained a more thorough method for how they used subheadings in their text. The main reason for using subheadings was to improve the visual experience of the page and to structurally divide content into more easily read sections. P1 expressed: “It is a conscious decision to make the text easy to see. And then it also helps screen readers and other tools, because it’s tagged with H2, H3 and so on.” P4, P5 and P6 were not aware of any special methods for how they used the subheadings.

Website accessibility status

Out of the seven websites tested and the ten respective pages on each website, a total of 400 images were identified. Of the 400 images, 143 of them lacked an alternative text attribute, rendering these difficult for screen readers to interpret. 104 of them had the attribute but no alternative text and 20 had improper alternative texts (e.g. the file name of the image). Only 32 utilized the title attribute. Of the total 400 images, 95 were decorative images injected with the background-image CSS attribute and thus lacked the possibility to add an alternative text. In total, 199 images (104 + 95) were considered decorative, and 38 images were considered informative with an alternative text.

Out of these 70 pages, 63 missing and skipped headings were identified due to improper nesting, averaging almost one incorrect heading nesting per page.

CMS accessibility status

Out of the seven CMSs investigated, three of them (W1, W2, W5) allowed inserting and editing alternative texts to images (features 1 - 2 for images). Three of them (W3, W6, W7) did not support these features. W4 had some instances on the page were an alternative image could be inserted and edited, but not all. All websites except W2 had no instances of automatically generated alternative texts (feature 8). Only W2 had the alternative text stored associated with an image (feature 10). No CMS supported the pedagogical features 3 - 7 & 9 for alternative texts. No CMS had support for inserting and editing the long description or the title attribute (features 11 - 21).
All the websites had support for assigning and editing headings in the user interface (feature 1-2 for headings). All websites except W6 had the possibility of changing the level of heading from the user interface (feature 3). All the websites warned when a page heading was not provided (feature 7). All websites except W1 used headings for the structure of the page (feature 8). W1 had some instances of headings not used for the structure of the page. Only W1 had the possibility of adding headings not used for the structure of the page (feature 9).

None of the websites supported the pedagogical features 3-6, the warning features 10-12 & 15 and the possibility of inserting and editing the title attribute of a heading (feature 13-14).

**Evaluation of the Selected Features**

**Think aloud evaluation**

Several comments were received when the testers were given tasks aimed at exploring the functionality of the alternative text property. Both T2 and T4 (see method for explanation) highlighted that the interface lacked initial information about the difference between decorative and informative images. T4 even had problems understanding the difference between the two after reading the informative texts provided by the system. T4 also needed some guidance to notice the text written out by the system, as it were not obvious to them that the text was changed. However, the rest of the testers noticed the change, with T2 expressing a positive reaction to the text, as the change of information displayed directed their focus to the text. T5 thought the text “The alternative text should properly describe the image” were problematic, as some editors may not know what the word “properly” may relate to in terms of accessibility. They cited the classical example of the warning triangle symbol. Should the alternative text say: “A black frame with a yellow fill and a black exclamation mark”, or just simply “Warning”? T5 pointed out that to properly describe the image in some cases, the user should convey the meaning of the image and not the figurative content.

When the users tested the second accessibility helper function, some additional comments were received. T2 saw a problem with editors not finding the button to run the accessibility helper function. They stated that the editor needed to be informed about the functionality prior to the production of content. T3 wanted the results of the heading nesting search displayed in the popup to be more visual than just the informative text, with color being an example to better improve the experience. T5 saw a problem of providing the editor with a sense of false security with the collection of accessibility functionalities provided. By stating that the features help improve the accessibility of content produced using the system, the editor may ignore or forget about other important accessibility areas.

**Interview with the testers**

All the testers (T1-T5) were positive towards the handling of the images in the alternative text property. T1 and T2 both stated that the editor often resorts to shortcuts and need guidance to produce the content wanted. T2 said: “It’s not always that people read the help texts and links so I think that you must complement with education, onboardings and walkthroughs. To show the system and get the people to use it” T3 highlighted that the problem still remains if the alternative text actually represents the information transmitted by the image. But stated that the function worked as well as it could be now. T5 reflected on the difficulties of providing the appropriate amount of information to the user, saying: “It’s hard to find a balance, how detailed you have to be. I think in an overview it currently works well [...] Or maybe urge the editor to read more. To press on further that your responsibility as an editor is to know this. The functions are only a support and not a solution”

When asked how they considered the accessibility for images changed from the original system, all the testers considered the accessibility to improve. T2 answered: “In the original system the user had to have that knowledge and in the new one you get the knowledge from the system. That’s ‘Knowledge in the world’ instead of ‘Knowledge in the head’” T4 reflected that the use of decorative images might decrease as awareness increases about the importance of the information transmitted by the images. T5 expressed that for editors unaware of accessibility standards would quite effectively get notified and considered that a huge positive change.

For the second function, all the testers were positive towards the handling of the headings in the accessibility helper function. All the testers also perceived the accessibility for headings when using the accessibility helper function to improve in contrast to the original system. Three of the testers (T1, T2, T5) expressed that the function limits the problem with editors using headings as styling instead of the structure of the page. T1 also continued to include cases where the function not only limit problems due to ignorance, but also mistakes due to human error. Additionally, T1 said: “I also know from other CMSs that editors copy text from other sources, e.g. word documents where styling is included in the copy without thinking about it. So, this is a good step to secure that” T3 expressed that the function would be a great support when editing larger articles. T3 said: “Even if you think about it, it’s very easy to make mistakes somewhere. Or if you don’t scroll up and check if you used an H2 or an H3 before. And you really appreciate those quick ways of getting feedback.” T2 thought the warning texts provided in the popup was self-explanatory and informative. They stated that it was a pedagogical improvement since you previously had to know about these problems by your own.

T3 and T4 highlighted the importance of educating the editor to find and use the function regularly, since it is currently a
manual evaluation of the content in the RTE. T1 found it problematic that the possibility exists that lay users may ignore to use the functionality, but stated it was good that the editor still could save content containing errors. T4 and T3 both recommended the evaluation to be automatic instead of manual. T1 and T4 also both wanted to include markers in the content editor on where the headings were nested wrong. T4 recommended the evaluation to run every time a heading order was changed, in order to find errors instantly to indicate to the editor that the current action was wrong.

T5 expressed that the evaluation logic must be able to be interchangeable on a website to website basis as the subheadings won’t necessarily follow the same structure on different websites. For example, one content editor may only provide the headings H3-H5, if the first two headings occur outside the scope of the editable content in the RTE. T5 also provided some insight in why it is currently very difficult for developers to build accessible websites, even if the content produced by the editor might be made accessible using these features. Modern websites often include modular components to be reused on different pages and in different layouts. Thus, it becomes a difficult, near impossible task to in advance know what heading that should be used so the nesting should be correct on all the instances where the component might be used. This is an example of a problem that can’t easily be fixed using these features and requires unique validation for every website using dynamic components.

In the concluding question, all the testers considered the functions to be effective as an attempt to minimize or eliminate common accessibility problems. T2 said: “I think a lot depends on educating the editor, and to make people aware of the importance of working with inclusive design. To design for all makes it better for all”

DISCUSSION
This study evaluated how a selection of 18 features from the proposed features by Acosta, T. et al [1] can be incorporated into a CMS with the purpose of minimizing or eliminating potential accessibility problems. Additionally, this study investigated what common accessibility problems may exist in the content generated by CMSs and has presented an example of how the features may be implemented in a commonly used CMS. As the web continues to evolve in the direction of enabling lay users to produce web content, so does the need for an informative CMS to support the editor to produce accessible content.

The overall result indicates that editors working with the production of web content lack information about accessibility and technical support for producing accessible content in their CMS. Furthermore, the developed functions with the set of incorporated features proposed by Acosta, T et al [1] may help the editor in producing accessible content. The functions included pedagogical elements, such as a reactive text field, that may help the editor by providing information about the current accessibility status of images and headings. In addition, recommending a solution if a potential accessibility problem is found.

Interestingly, all the participants in the initial study pointed out a lack of information and guidance about accessibility in their CMS. The experienced lack of information occurred regardless of user knowledge of accessibility, noted that the participants stated to have varying knowledge about accessibility. Most participants perceived a lack of technical support for producing accessible content in their current CMS. This indicates that editors with already high awareness of accessibility still may not find support in their system to make their content satisfactory accessible. The participants’ perceptions of the functionality and the information provided by their CMS proved to be in coherence with the results from the investigation of their CMS. Thus, no accessibility functionality on their respective CMS was found that the participants were not aware of.

The results from the think aloud evaluation revealed an initial lack of information about the difference between decorative and informative images. Multiple testers had comments regarding the information provided about the decorative images. This highlights the need for better descriptive text about the differences between the two. Consequently, a new feature could be appropriate to add to the existing 21 for images: “Provides initial information about the difference between decorative and informative images”

The overall findings from the evaluation process indicates that it is not necessarily enough to only include the features inside the CMS, but also important to enforce their use. As several testers mentioned, the user may need complementary education about how to use the system prior to its use. T2 recommended this to be executed with the use of a walkthrough or an onboarding process but did not specify if the introduction should be in person by the developers implementing the CMS, or within the CMS itself. Ideally, the system would inform the editor without the need of personal guidance by a developer.

The alternative text and accessibility helper functions differ in an important part of the validation process. The first function for adding alternative texts to images includes an active validation for ensuring that an added image is either decorative or informative. Due to the active validation, no image could be added without the alternative text attribute and thus eliminate common problems with missing accessibility information for images. The problem persist that the alternative text may not represent the information in a proper way. However, the mentioned problem is outside the limitations of the proposed features and is currently hard to validate without manual evaluation. Most testers observed the change of information provided in the CMS depending on if the editor had written an alternative text or assigned the image as decorative. The reactive text can be described as a useful approach of providing the editor with relevant
information, promoting awareness and guidance towards producing accessible content.

The second function for validating the nesting of headings in the RTE can be considered a passive validation, as it requires a push of a button in order to run. Editors without proper knowledge about the functionality may not use it due to not finding the button or not knowing about it. The validation will then not run and the problems with incorrect nesting of headings will persist. Several testers recommended the validation for the nesting of headings to run automatically, ensuring that lay users still will be able to obtain the information about potential errors in their texts. With adjustments to the validation logic and visual presentation of the information about the nesting of headings, the function could become an active aid for the editor.

In the initial study, P1 stated that they had perceived accessibility to sometimes be limiting in a previous job. Considering such a statement, it is interesting to discuss if the developed functions are of support or hindrance to the effective production of content by the editor. With the first function, if an editor does not care for accessibility, they can choose to make all images decorative with the click of a checkbox. The resulting content produced will still be considered accessible as the images still include an alternative text attribute. The extra click on the checkbox will not impact the effectiveness of producing content in any noteworthy manner. As follows, only editors striving to improve the accessibility of their images can choose to do so by adding alternative text on informative images. In the case of the second function, the validation logic is currently passive and won’t affect the production effectiveness of editors if some editors chose not to use it. However, as T1 mentioned, even if the validation logic were active it is important to keep the ability to save content containing errors. The purpose of the features is not to actively prevent content containing accessibility errors to be produced, but to inform and guide the editor into producing accessible content.

The consensus by the testers was that the functions were the most effective when used by lay users with little to no experience working with accessible content. As the primary goal of the functions is to guide and inform the user about these areas, users with no prior experience working with accessibility will be most benefited from the information provided in the functions. However, even editors with high experience with accessibility will be aided if they make mistakes due to human error.

One important point that was brought up during some of the interviews with the editors was that the accessibility was not in focus during the development of their CMS. Accessibility features are in some cases rather extensive which translates into longer and more costly projects. Thus, even if experienced developers are working with a CMS, restrictions of time or resources may limit the development of these features. Due to the stated reasons, accessibility often is given lower priority in development projects [10], and it is of high importance that the CMS already has accessibility features included in its base functionality.

Currently, a significant part of the accessibility features is required to be implemented on an installation to installation basis. And in some cases, there is no way to generalize a solution to include the features for all generated content on separate websites. Differing website layouts means variant implementations of the features and thus a developer does need to be involved in the implementation of the features on websites with custom designs. However, the base functionality of the CMS may still support the developer by including features that is not altered for different websites. The Accessibility Helper function is an example of a reusable implementation of the selected features for headings. The validation logic for nesting of headings does not care for the design of the page, only that the nesting is correct. The inclusion of similar functionality for the base functionality of the CMS will benefit all implementations of websites.

When using websites templates with CMSs, there is no reason not to include all the accessibility features already implemented in the template. Since the design of the page is not generally changed when using templates, there is no need for custom implementations of the features. By including the features in the templates, all future installations will include them as well. There is a vast number of people in need of accessible content on the web [9], by including the features with the templates there is a chance of improving a substantial part of the content generated. It cannot be avoided that lay users manages the content on CMS websites, but the features might help lay users to become aware of accessibility. Additionally, through the guidance of the features lay users might start to generate accessible content, greatly contributing to achieving an accessible web.

The explored research question in this study asks how a selected set of features can be incorporated into a CMS in order to minimize or eliminate potential accessibility problems with editor generated content. As the concluding question in the evaluation study, all testers considered the functions to fulfill their objective effectively, which may indicate that the functions are a successful step forward to promote accessible content online. An active validation is arguably superior to a passive one in the implementation of the proposed features, as it provides most help to lay users. And the addition of a new feature for initially describing the difference between decorative and informative images would help editors become aware of the difference.

**Limitations of the Method**

It should be noted that the participants in the initial study were all customers of Winston and may not be representative of editors in general, and not lay users. Furthermore, the CMS investigated in the initial study were all implemented by Winston and may thus also not be representative of CMSs in general. However, the issues identified in the study have
been documented to exist across the web development landscape [10]. Also, one could argue that the initial study investigated the core functionality of the CMSs (Umbraco, EPiServer and WordPress) with respect to accessibility features, which Winston have not changed.

An alternative approach to answer the research question could have been to have the editors in the initial study participate in the evaluation study, as it would have provided an improved standardization of the answers. Due to time restrictions and experience differences in CMS platforms used by the editors, this was more difficult to achieve.

**Future Research**

As some testers in the evaluation study recommended the functions to evaluate automatically, one future study could investigate the effect on content produced by editors with automatic evaluation. To further evaluate the features proposed by Acosta, T. et al [1], a prototype with all the incorporated functionalities could be developed and tested.

An interesting future study may investigate the effect on produced content from a CMS with the incorporated features, if the improved awareness of accessibility translates into more accessible content by the editors. A future study may also study the effect of using a CMS with the incorporated features by different user groups of varying knowledge about accessibility.

**CONCLUSION**

This study evaluated 18 features proposed by Acosta, T. et al [1] for improving the accessibility of images and headings produced using a CMS. The study found a general lack of information, guidance and technical support provided by CMSs to editors promoting accessible content. The results show that the functions incorporating the proposed features could be considered effective to minimize or eliminate common accessibility problems with images and headings. This study contributes by evaluating how the selected features proposed by Acosta, T. et al can be integrated to a CMS. The results show that the features could be implemented into practical use with positive effects.

The developed functions included pedagogical elements which provided information and technical support to the editor by ensuring all added images in a content editor are either decorative or informative. Additionally, by verifying if the nesting of heading orders is correct. This study further proposes an additional feature to complement the list provided by Acosta, T. et al. to provide initial information about the difference between decorative and informative images. The findings from this study further contribute to the goal to achieve an accessible web, which helps to build an inclusive web community for all users using assistive technologies.

**ACKNOWLEDGEMENTS**

I would like to thank Mårten Boman for letting me into the team to conduct my thesis on Winston and for the support during the process. I would also like to thank Tony Hietapakka for the input and technical support with the implementation process of this thesis. Additionally, I would like to thank the whole team at Winston for their help and allowing me to conduct multiple interviews with them. Lastly, I would like to thank my supervisor Anders Lundström at KTH with Sharon Ghebremikael and Thujeepan Varatharajah for the valuable feedback during the process.

**REFERENCES**


