

BIS

Digress-it Plug-in

Accessibility Audit & Usability Review

25th March 2010

Created by

nomensa
humanising technology

For

BIS | **Department for Business
Innovation & Skills**

Change log

Revision History

Date	Author	Version	Change reference and Summary
22/03/2010	Matt Lawson	0.1	Created document

Reviewers and Approvals

Date of approval	Name	Position	Version
23/03/2010	Léonie Watson	Director of Accessibility	0.2

Distribution

Date	Name	Organisation	Version
23/03/2010	Steph Gray, Joss Winn	BIS, University of Lincoln	1.0

Department for Business, Innovation and Skills

Digress-it Plug-in Accessibility Audit & Usability Report

Document Summary

This document contains the results and analysis of the accessibility audit conducted by Nomensa on the 22nd March 2010 for BIS. The document details the outcomes of the audit, including actionable recommendations. Nomensa have also carried out an informal usability review of the Digress-it Plug-in.

For any further information about any points made in this document please do not hesitate in contacting Nomensa:

Léonie Watson
Project Manager

Email: bis@nomensa.com
Tel: +44 (0)117 929 7333

Contents

Introduction.....	5
Web Content Accessibility Guidelines.....	5
Web Content Accessibility Guidelines 2.0	5
Methodology.....	6
Accessibility Audit Results.....	8
Text Alternatives - Guideline 1.1	8
Adaptable - Guideline 1.3.....	10
Distinguishable - Guideline 1.4	11
Keyboard Accessible - Guideline 2.1	14
Navigable - Guideline 2.4	15
Readable - Guideline 3.1	19
Input Assistance - Guideline 3.3.....	20
Compatible - Guideline 4.1	22
Usability Review	24
Quick wins.....	24
Appendix A	28
Appendix B	30

Introduction

Nomensa were commissioned to conduct an accessibility audit for the Digress-it plug-in website. This report covers the outcomes of the accessibility audit, provides an insight into the impact each issue has on the user experience and offers explanations and solutions for each problem.

Nomensa have also carried out a usability review of the Digress-it plug-in and providing some quick wins and recommendations on improving the usability of the plug-in.

Web Content Accessibility Guidelines

The Web Content Accessibility Guidelines are developed by the Web Accessibility Initiative (WAI). The WAI are a branch of the World Wide Web Consortium (W3C), the organisation that provides guidelines and specifications for many web technologies.

The mission statement for the W3C explains the organisation's aim:

"The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential."

The Web Content Accessibility Guidelines are recognised as the international standard for building accessible websites, and measuring web accessibility. They are part of a suite of guidelines that represent three different aspects of web accessibility:

- Authoring Tool Accessibility Guidelines (ATAG);
- Web Content Accessibility Guidelines (WCAG);
- User Agent Accessibility Guidelines (UAAG).

The Web Content Accessibility Guidelines are intended to cover accessibility issues relating to all kinds of disability. This includes visual, hearing, cognitive and physical conditions. The aim is to provide a unified approach to web accessibility, preventing a solution for one user group unintentionally becoming an obstacle for another.

Web Content Accessibility Guidelines 2.0

WCAG 2.0 was released in 2008, bringing web accessibility guidance up to date with modern web development techniques. In the time since WCAG 1.0 were released, the assistive technologies used by people with disabilities improved dramatically. New technologies and techniques were also introduced into web development. WCAG 2.0 reflects these changes, taking a wider range of web technologies into account.

WCAG 2.0 consists of four principles:

- Principle 1 Perceivable: Information and user interface components must be presentable to users in ways they can perceive;
- Principle 2 Operable: User interface components and navigation must be operable;
- Principle 3 Understandable: Information and the operation of the user interface must be understandable;
- Principle 4 Robust: Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

The principles are broken down into 12 guidelines, and from there into a number of checkpoints. There are 61 checkpoints in total, and each one has a number of success criteria assigned to it. WCAG 2.0 defines three levels of success criteria:

- Single-A, the most basic level of accessibility;
- Double-A, the intermediate level of accessibility;
- Triple-A, the highest level of success criteria.

WCAG 2.0 measures conformance using the same three levels that define the success criteria. This approach also carries through the conformance levels used in WCAG 1.0, although the requirements are slightly different:

- Single-A, achieved when all applicable Single-A success criteria are satisfied;
- Double-A, achieved when all Single-A and Double-A success criteria are satisfied;
- Triple-A, achieved when all Single-A, Double-A and Triple-A success criteria are satisfied.

Web accessibility is measured on a page by page basis. The guidelines are applied to each individual page within a website and each page is awarded the appropriate accessibility level. A website is only as strong as its weakest page, but overall conformance is usually based on the level achieved by the majority of pages.

Methodology

For the Digress-it Plug-in accessibility audit, a representative sample of three pages was selected. These pages encompassed each different kind of template or content variation found throughout the site.

The sample web pages were measured against all WCAG 2.0 checkpoints, using all Single-A and Double-A success criteria. For each success criteria relating to a given checkpoint, the page was given a pass (p), fail (f) or marked as not applicable (n/a). For a full list of the pages included in the audit please see Appendix A.

Format

Each issue that was identified during the audit is detailed in the Audit Results section of this report, presented in checkpoint order.

For each checkpoint a description is provided, which covers the reasons why that issue is important to web accessibility. A summary of the problem is also included covering where the problem was identified. Lastly, each checkpoint is accompanied by suggested solutions for resolving the problem, either now or as part of a longer term strategy. The checkpoints that the pages passed have not been detailed in this report.

Accessibility Audit Results

Overall the Digress-it plug-in did not perform well in terms of accessibility. Nine Single-A and four Double-A accessibility issues were identified which failed to meet the requirements set by the WCAG 2.0.

However, many of the Single-A issues, such as the lack of alternative text and skip links should be straight forward to address. Achieving Single-A accessibility should be part of a quick win strategy.

By far the biggest issue to address relates to the accessibility for keyboard users. Several elements such as the commenting icon and add/expand comment icon elements were not keyboard accessible which means it would be very difficult for a keyboard user to access the content and post their reply.

Although JavaScript support is given more flexibility under the WCAG 2.0 guidelines, Nomensa still recommends that progressive enhancement techniques are used to ensure that users with JavaScript disabled or unavailable are still able to perform actions on the site. Testing of the Digress-it plug-in found that when JavaScript was disabled, the user was unable to read or post comments on the site, preventing them from carrying out a core component of the Digress-it plug-in.

A list of issues, grouped together by success criteria, can be found in Appendix B.

Text Alternatives - Guideline 1.1

Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, Braille, speech, symbols or simpler language.

Non-text Content: checkpoint 1.1.1



Level-A: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose.

Description

Providing textual information for content that is not text is the most important accessibility issue. People with sensory disabilities have ways of accessing text even if they cannot see or hear, for example through visual, auditory or tactile means.

Text is the basic format of information, and is vital for providing equivalent information for people using access technologies. For example, a person who can't see or hear could use a Braille reader to read out the alternative text of an image in an HTML page.

Summary

All three pages included in the audit failed this checkpoint because alt attributes were missing or alternative methods could have been used to insert the decorative images.

Recommendations

Make sure all non-text content has appropriate text alternatives and ensure that content providers have the tools & knowledge to add text alternatives successfully.

There are many types of non-text content. Images, audio, video and animations are all types of non-text content and all need to have appropriate text alternatives.

Icons that appear in the top navigation links on all pages, see Figure 1, are for decorative purposes only and therefore would ideally be inserted as a background image with CSS. Alternatively, if they must be inline images they should use a null alt value (e.g.: alt="").

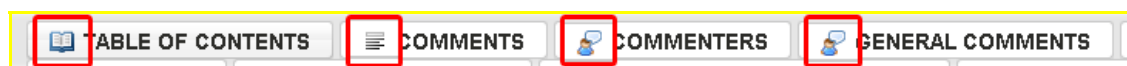


Figure 1: Decorative images which would ideally be inserted with CSS

Attribute text was also missing for the comment link images on the 'Comment test' page and 'Layout test' page, see Figure 2.

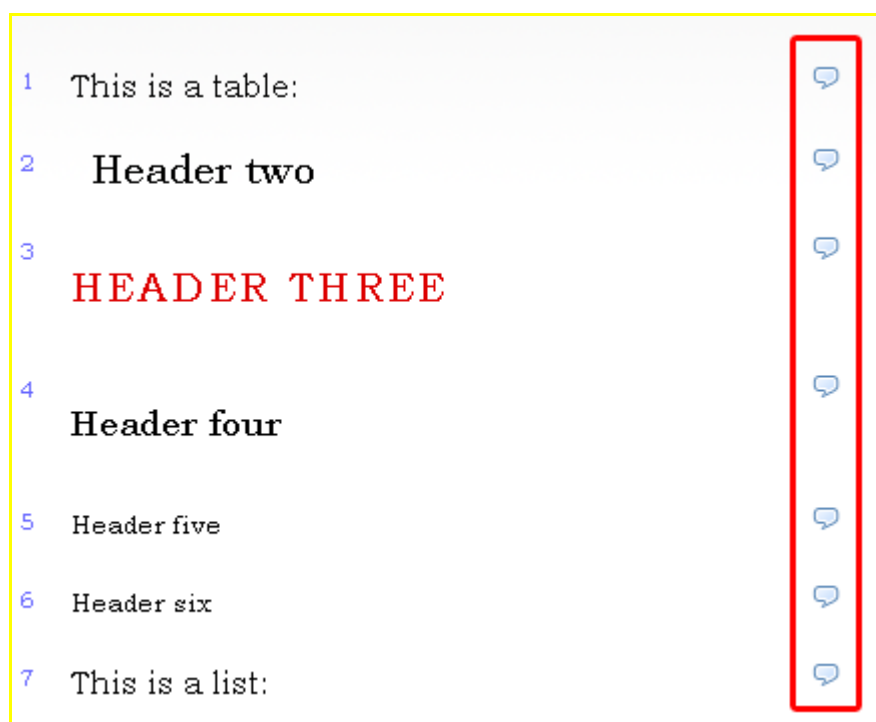


Figure 2: Comment icon links missing alternative text.

Adaptable - Guideline 1.3

Create content that can be presented in different ways without losing information or structure

Info and Relationships: checkpoint 1.3.1



Level-A: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.

Description

Using the correct methods to define structure for content allows access technologies to interpret the content separately from visual presentation. Access technologies use this structure to render the content of the document in a way most useful to its user. This is how access technologies make sense of pages.

For example in an HTML page, if a heading is created by increased font-size rather than using structural code, a sighted person would understand that it is a heading, but a non-sighted person would not.

Only if such structure is present can access technologies correctly interpret the purpose of each element on a page.

Summary

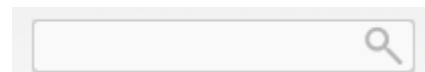
All of the pages failed this checkpoint because the structure and mark-up of elements on the pages could be improved to convey the same relationships as the visual presentation.

Recommendations

The method used to represent the visual display is generally decided at the template stage of web development, when a design is translated into HTML, Flash or other formats.

When considering what technologies to use for a site, one aspect to include is that Flash does not have certain structural elements such as headings, lists and quotes. Therefore, Flash is not a suitable technology for predominantly text based content.

An associated label was missing for the search input field on all of the pages tested.



The 'Comment test' page and 'Layout test page' also failed this checkpoint because:

- Fieldsets and legends should be used to wrap the "Leave a reply" input fields;
- Tabular mark-up (e.g.: <td>) should not be used to structure the comments box;

- The comments list should be marked-up as an ordered list ();
- The top navigation and comments area would benefit from having headings that define the content that follows. These additional headings can be hidden off screen if desired.

Distinguishable - Guideline 1.4

Make it easier for users to see and hear content including separating foreground from background.

Contrast (Minimum): checkpoint 1.4.3



Level-AA: The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following:

- **Large Text:** Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;
- **Incidental:** Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement;
- **Logotypes:** Text that is part of a logo or brand name has no minimum contrast requirement.

Description

For people who wear glasses, who are colour blind or who have a more serious visual impairment, reading information online can be difficult and tiring. Using colour combinations that provide poor contrast can make it almost impossible to read content at all.

Using colours that show up clearly against each other will make all the difference. This is particularly important for the colours of text used in images, as there is no opportunity to adjust the image to try and read it more easily.

There are clear guidelines and tools for testing this criterion, regardless of what technology is used.

Summary

All of the pages included in the audit failed this checkpoint because a small colour contrast issue was identified for links that appear on a gradient background, see Figure 3.

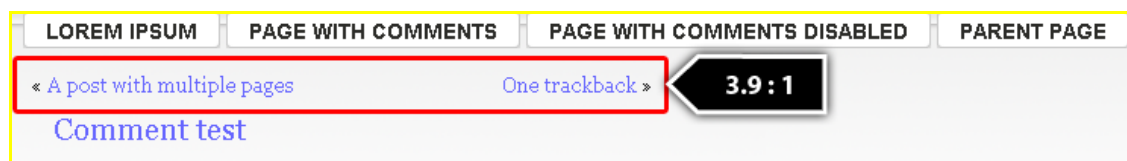


Figure 3: Colour contrast issue of links that appear on a gradient background

The links had a colour contrast ratio of 3.9:1 which failed to meet the minimum 4.5:1 requirement set by the WCAG 2.0 guidelines.

Recommendations

If the palette does not provide enough (or any) colour combinations with sufficient contrast, then there are three main possibilities:

- The colour palette is updated; **or**
- Techniques are used to get around the issue, such as a contrasting drop-shadow around text in images, or using white/black backgrounds rather than coloured ones; **or**
- A style switcher could be implemented to give the user choice about which colour scheme they use. For example, the default style might be the fully branded one, and a higher contrast version is made available by selecting a link. The higher contrast version is then applied throughout the site for that user.

For an example style-switcher please see <http://style-switcher.nomensa.com/>

Resize text: checkpoint 1.4.4



Level-AA: *Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality.*

Description

For many people, the act of increasing the size of text on a page can make the reading experience more comfortable. For people with mild or moderate visual impairments, it can become a necessity, and the website must support the ability to resize text.

Text in HTML pages can be resized in the browser by default, if implemented correctly.

For websites that predominantly use plug-ins such as Flash, there are several methods to support this aim, described in the Vendors Development Standards document.

Summary

All of the pages included in the audit failed this checkpoint because the site heading was getting cut-off at smaller browser windows sizes or when text size was increase, see Figure 4.



Figure 4: Site heading getting cut-off at smaller browser windows sizes

The 'Comment test' page and 'Layout test' page also had a browser bug in Firefox that caused the comment box to get cut-off at lower screen resolutions, see Figure 5.

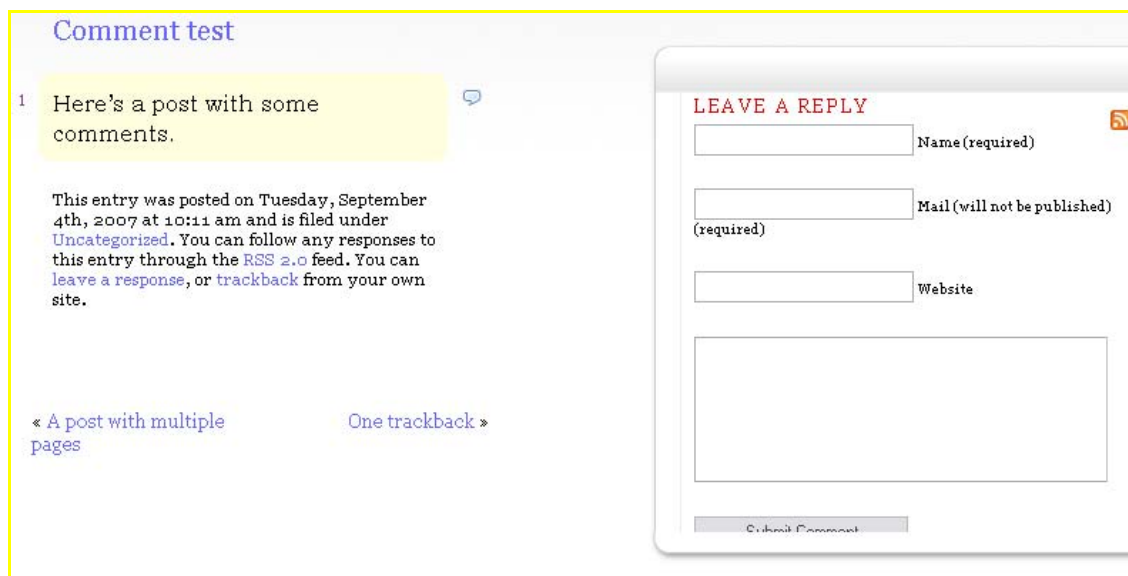


Figure 5: Comment box content getting cut-off at smaller browser window sizes

Recommendations

When defining elements such as font size, width or height, ensure that relative measurements are used, such as em or percent, rather than absolute measurements such as px or pt.

Resizing the browser window or font size should not result in the content becoming hidden or cut-off. It is important to check that the page reflows correctly to allow for different resolutions and font sizes.

Keyboard Accessible - Guideline 2.1

Make all functionality available from a keyboard.

Keyboard: checkpoint 2.1.1



Level-A: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.

Description

Not everyone uses a mouse to access their computer. People with physical disabilities that affect their ability to control a mouse, for example Parkinson's disease, will rely on the keyboard instead. Websites are often built on the assumption that everyone uses a mouse and in some cases, these websites are impossible to use if you cannot.

Actions on a website are sometimes referred to as events. Links or buttons that require a mouse event are often impossible to handle via the Enter key of the keyboard. As technology develops, this problem is becoming less prevalent, but even the smallest obstruction can be fatal in terms of accessibility.

Note: These checkpoints should not discourage mouse or other input *in addition* to keyboard input.

Summary

The 'Comment test' page and 'Layout test' page both failed this checkpoint because several areas on the page did not use semantic HTML elements as controls and/or links.

For example the comment quote icon and green add/expand comment icon used an image and/or span, see Figure 6.

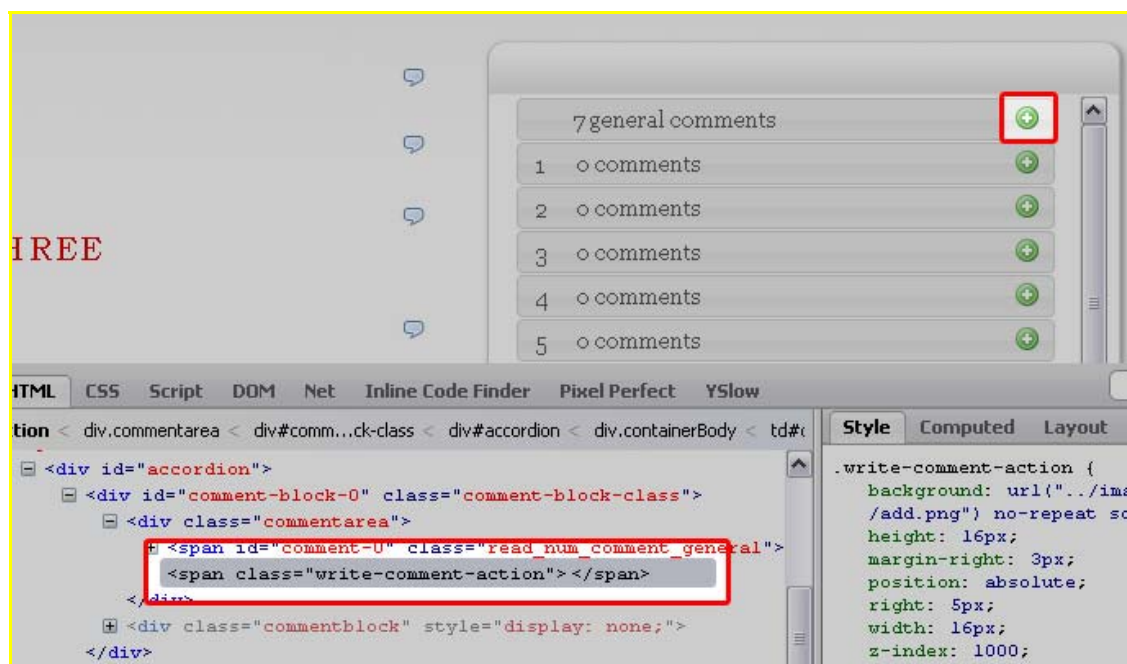


Figure 6: Use semantic elements for controls on the page to ensure keyboard accessibility

These elements are not suitable as they are not keyboard accessible by default and therefore cannot be tabbed to or selected using the keyboard only.

Recommendations

Where technologies such as Flash, Java or JavaScript are used on the page, steps should be taken to ensure that it is operable through the keyboard. In other words, it should not rely on mouse interaction in order to function.

For elements that require user input such as the comment quote and add/expand icon, an anchor element (`<a>`) would be better suited as it provides keyboard functionality without additional scripting.

Navigable - Guideline 2.4

Provide ways to help users navigate, find content and determine where they are.

Bypass Blocks: checkpoint 2.4.1



Level-A: A mechanism is available to bypass blocks of content that are repeated on

multiple Web pages.

Description

Skip links point to another part of the same page, instead of to a new web page altogether. Often used to provide “Back to top” links in long pages, skip links are so called because they allow a person visiting a website to skip quickly over large blocks of content. In accessibility terms, they are most commonly used to allow people to skip over the navigation of a page and move directly to the start of the main content area.

Skip links are particularly useful for people with some physical impairment, particularly those who rely on simple input devices to access a computer. Such devices can often mean moving from one element on a web page to the next, for example from one navigation link to the next, in order to reach a given part of a page. On pages where there are large numbers of navigation links, this can be a painstaking process. A skip link enables the person visiting the website to bypass all that effort by activating a single link.

Summary

All of the pages included in the audit failed this checkpoint as they did not provide skip links.

Recommendations

Where large blocks of content are included on the page, for example a list of navigation options, provide a skip link that allows the user to bypass the area and move directly to another point on the page.

For more information on skip links, see Nomensa’s article: Skip Links.

<http://www.nomensa.com/resources/articles/web-development-articles/skip-links.html>

Focus Order: checkpoint 2.4.3



Level-A: *If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.*

Description

Some input devices are linear, for example, a keyboard goes through items one at a time, whereas using a mouse allows access to any visible item in any order. For people using a screen reader, the output is also linear, as you can only hear one thing at a time.

The order of content and controls should therefore be logical, so that people using linear devices can understand and use the content.

Depending on the technology, the order might be determined in HTML by the order of the source code, or in Flash it must be set by the developer explicitly.

Summary

The 'Comment test' page and 'Layout test' page both failed this checkpoint because *tabindex* attributes had been given to the comment input fields.

Recommendations

Nomensa would recommend avoiding the use of *tabindex* where possible as it can confuse and disorientate users when navigating content on the site. Allowing the focus order to follow the natural source order with clear skip links and heading structure will be sufficient to allow users to navigate the content.

Headings and Labels: checkpoint 2.4.6



Level-AA: *Headings and labels describe topic or purpose.*

Description

Descriptive and concise headings and other page elements help people navigate the page easily and understand the content of the section.

In a similar way to page titles, it helps to put the keywords first, so rather than headings in the form "Section 1: Introduction", it would say simply "Introduction".

Summary

All of the pages failed this checkpoint as headings and labels could be added and/or improved. For more information please refer to checkpoint 1.3.1.

Recommendations

Form labels tend to be something added by developers, therefore should be checked at the implementation stage.

Clear and concise, descriptive headings help all users find and understand content. Headings help people understand the structure and relevance of the content presented, as well as allowing users to skip over content of no interest to them.

Headings must clearly describe the content which follows the heading at the first time of reading. It is often helpful to put keywords or important information at the beginning of headings as this aids users when scanning the content on the page. Headings should be considered content, therefore checked at the content creation stage.

Focus Visible: checkpoint 2.4.7



Level-AA: Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.

Description

People with motor impairments may not be able to use a mouse, but can see the screen, so it is important that people can work out where the focus of the keyboard is.

Most technologies have methods of highlighting keyboard focus built in, and these should not be removed, but can be enhanced.

Summary

All of the pages included in the audit were given a tentative pass. While the browser default outline was supported for the focus styling, see Figure 7, Nomensa would recommend improving this to provide focus styling that has higher visibility.



Figure 7: Focus styling using the default browser outline.

Recommendations

The site should not remove the default browser behaviour for showing the focus, and can optionally add styling to make the focus more obvious. This can include, but is not limited to, underlines, background/foreground colours and font style changes.

For more information about focus indicators please see Nomensa's article, Using Focus pseudo-class to improve your site's accessibility:

<http://www.nomensa.com/resources/articles/accessibility-articles/using-focus-pseudoclass-to-improve-your-sites-accessibility.html>

Readable - Guideline 3.1

Make text content readable and understandable.

Language of Page: checkpoint 3.1.1



Level-A: *The default human language of each Web page can be programmatically determined.*

Description

If natural language changes in a document are marked up correctly, speech synthesizers and Braille devices can automatically switch to the new language, making the document more accessible to multilingual users. The predominant natural language of a document's content should be identified in the code through the use of mark-up or HTTP headers.'

In addition to helping assistive technologies, natural language mark-up allows search engines to find key words and identify documents in a desired language. Natural language mark-up also improves readability of the Web for all people, including those with learning disabilities, cognitive disabilities, or people who are deaf.

When abbreviations and natural language changes are not identified, they may be indecipherable when machine-spoken or converted to Braille.

Summary

All of the pages included in the audit failed this checkpoint because the language of the pages was not defined in the HTML element.

Recommendations

The HTML element at the beginning of each page should use the HTML 4.0 "lang" attribute to help specify the main language of the text on the page, such as English, French, German, Japanese, etc.

```
<html lang="en">
```

Or for an XHTML doctype:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

This helps the computer or assistive device present information in a way that is appropriate to the language and also helps automatic translation software that translates text from one language into another.

Input Assistance - Guideline 3.3

Help users avoid and correct mistakes.

Error Identification: checkpoint 3.3.1



Level-A: *If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text.*

Description

People often make mistakes when filling in forms on websites. Websites should clearly identify where an error has occurred within the form and relay this information back to the user. Error messages should be clear, meaningful and help the user determine where the error has occurred, what is wrong with the particular form entry and how to correct the problem.

Summary

The 'Comment text' page and 'Layout test' page were given a tentative pass for this checkpoint because inline validation messages could be used to identify required fields when entering a comment.

Recommendations

One solution would be to use client-side validation for simple checks such as required fields. This could be handled with JavaScript with Wordpress' default validation acting as a backup for users without JavaScript.

Labels or Instructions: checkpoint 3.3.2



Level-A: *Labels or instructions are provided when content requires user input.*

Description

People who do not have a view of the web page in a graphical layout may not be able to determine which label applies to which form control, and therefore will not know what data to enter. Proper positioning and correct implementation of the label helps to avoid this.

Other aspects that help users are:

- Showing an example of the format expected, for example "Date (dd-mm-yyyy)";
- A short paragraph explaining the necessary input for of a set of fields;

- Grouping items visually and in the underlying structure.

The intent behind the checkpoint is not to overwhelm the person with lots of information, but to prevent common errors.

Summary

All of the pages failed this checkpoint because the search input was missing an associated label element.

The 'Comment test' page and 'Layout test' page also failed because the "Leave a reply" form elements should be grouped using `<fieldset>` and `<legend>` elements.

It is also worth highlighting that Nomensa recommends labels for text inputs appear before their associated input field, see Figure 8.

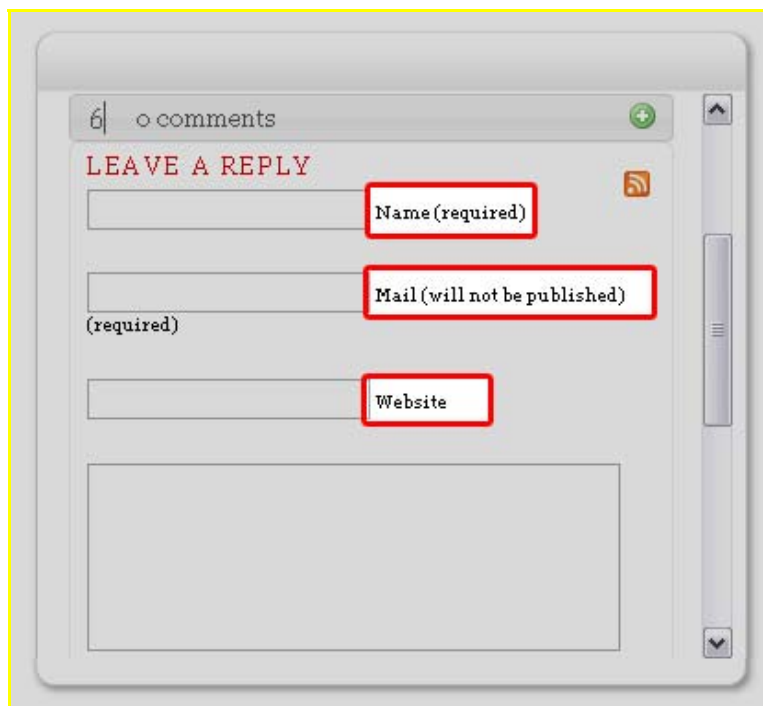


Figure 8: Labels appear after their associated input field

Recommendations

Ensure that form labels are provided for each form control and that they are correctly positioned in relation to the form field they correspond to. For label elements, Nomensa recommend:

- Labels for text inputs are before the input;
- Labels for text-areas appear before the text areas;

- Labels for radio-buttons and check boxes should appear after the radio-button or check-box;
- Labels for any group of form controls should appear before the group.

Compatible - Guideline 4.1

Maximize compatibility with current and future user agents, including assistive technologies.

Parsing: checkpoint 4.1.1



Level-A: *In content implemented using mark-up languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features.*

Description

Whether it is HTML or XHTML, it is important to use valid code so that past and future Internet devices can show the content in a way most suitable to their requirements. Using World Wide Web Consortium (W3C) technologies such as valid HTML and CSS code is also better for accessibility because:

- **W3C technologies include "built-in" accessibility features.**
- **W3C specifications undergo early review to ensure that accessibility issues are considered during the design phase.**
- **W3C specifications are developed in an open, industry consensus process.**

Summary

All of the pages included in the audit failed to pass the W3C Validation Service. Errors included invalid attributes, missing required attributes and incorrectly nested elements.

Recommendations

All code documents are validated using a suite of tools from the W3C. HTML code is tested with the W3C Markup Validation Service:

<http://validator.w3.org/>

CSS code is tested with the W3C CSS Validation Service:

<http://jigsaw.w3.org/css-validator/>

Ensure that all documents, whether HTML or CSS, pass through the validation tools outlined above. Where issues are identified within the results of a validation check, follow the guidance given on fixing each individual problem.

Usability Review

Digress-it-it provides functionality for collaborative commenting on articles, reports and other types of documents. The online tool allows articles to be reached by a wide audience and provides visible tracking of comments and feedback. The expert usability review of Digress-it-it involved a high-level review to identify the most critical usability issues. To help make quick and easy usability improvements a series of 'Quick win' recommendations have been identified.

In terms of usability, the key overarching concern is making the tool appear inviting and easy to use. This is particularly important as BIS are aiming to engage with the general public and therefore the tool needs to cater for a range of technical ability. It is also likely that this will be the first time users have encountered a tool of this kind and so it is crucial they are not put off because they feel intimidated or have to spend time learning to use it.

The version used for 'This is service design thinking' (<http://service.engagement.ac/>) gives an example of how the tool can be adapted to feel more friendly and inviting. (Please note, that the 'Method & tools' section of this site would be easier to use if the table of contents was hidden once a user has clicked into a section to review).

Quick wins

Introduce the purpose of the tool

On the home page, clearly and simply explain what the tool enables users to do and how it works. As it is likely that users are not familiar with this sort of tool a simple explanation and steps to get started will help break down the barriers to entry. Currently the tool requires a substantial proportion of time for a user to learn how it works and this may put off many users who do not have that time or lack confidence in their computer skills.

Use meaningful labels

The labels, such as navigation labels and page headings, could communicate more meaning to help users quickly understand how the tool works. For example 'table of contents' can be confusing as users may expect it to be a list of the top level IA sections rather than the documents that are available (see Figure 9). Using the label 'Policies for review' or 'Policies for consultation' instead would explain to users that these are the policies available and they are open for review.

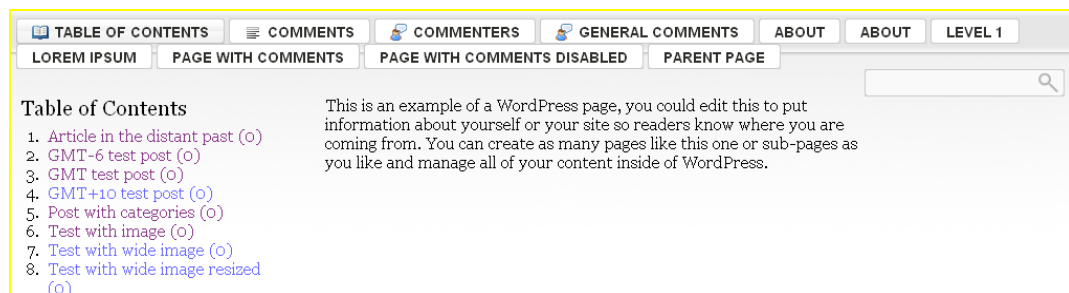


Figure 9: The phrase 'Table of Contents' may be confusing to users

Use a small number of sections

To make the tool as simple as possible use only the minimum number of top level sections required. The 'elearnmanifesto' (<http://elearnmanifesto.org/>) is a good example of this, as they have removed the top navigation and opted to use only the list of articles and commenting facility.

In the standard version of tool it is difficult to understand the difference between 'Table of contents' and 'Comments'. Although the 'Comments' page shows full comment threads these are hard to interpret without the context of the article or paragraph they refer to. Therefore, it would be better to use only the 'Table of contents' page and add in useful 'Comments' page functionality, such as the links to RSS feeds (see Figure 10).

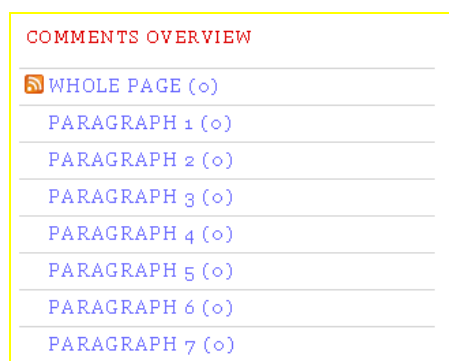


Figure 10: The 'elearnmanifesto' site includes the RSS feed links within the comment review/add panel

Use the comment icon alongside any functionality or sections that relate to comments

On the article pages a clickable 'comment' icon is used alongside each paragraph to allow users to jump to the comments on that section. Using this icon throughout

the site, alongside links to comments and on the comment review/add panel, will help group this functionality together and help users understand this connection (see Figure 11). It will also allow users to quickly identify where and when they can jump to comments throughout the site.

Where appropriate, the icon should also list the number of comments next to it, even when no comments have been made so users can understand this on first glance rather than clicking to find out.

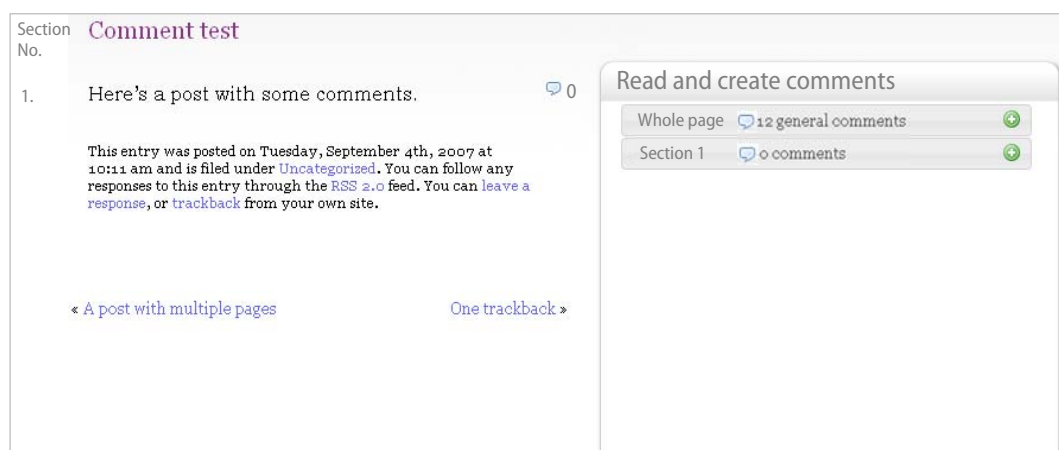


Figure 11: Mock-up showing where to use the comments icon on an article page

Communicate that numbers in brackets are numbers of comments

Although the number of comments per article is listed within brackets in each of the summary lists, for example 'Introduction (0)' or 'Test User (7)', it is likely users will not understand what this number means as it has no explanation. It may be particularly detrimental if users interpret the '0' to mean no documents available and therefore do not click the link. In this circumstance whole documents may not get reviewed.

The document lists are a great place to implement the recommendation above, inserting the comment icon alongside each number will help users realise this is numbers of comments. For example:

[1. Introduction](#) ( 0)

If an icon cannot be included within the hyperlink then the following layout would also work:

[1. Introduction](#) ( 1)

[2. Getting started](#) (💬 0)

The comment icon should also have tool tip text to say 'There is 1 comment for this article'.

💡 **Label the paragraph numbers**

The paragraph numbers are currently listed on the left hand side without an explanation of what the numbers mean. Give this column a title, for example 'Section number' to explain to users what these numbers are for.

💡 **Enhance comment review/add panel**

The comment review and add panel provides really useful functionality but may appear confusing to users. A few quick improvements can make this feature much easier to understand and learn (see Figure 11 to view a mock-up of these recommendations):

- Put a title on the widget, such as 'Read and create comments';
- Make the window larger so more content is visible at any one time, this will make it easier for users to get a fuller picture of the content and require less scrolling;
- Label the number that identifies the paragraph as 'Section', currently it is not clear what this number refers to and this is critical to users using the functionality correctly. For example 'Section 1 (💬 1comment)' (see Figure 11), this is similar to the style implemented on the 'elearnmanifesto' site;
- When users click a section, expand the section but show the first comment first, rather than scroll to the 'leave a reply' section. This will make it easier for users to understand their position within the widget rather than have the focus jumping around unexpectedly;
- Allow users to expand multiple sections, this will let users see more comments side by side if they wish. The current design, using '+' buttons also suggests this functionality, rather than its current accordion style behaviour.

💡 **Avoid using tag clouds**

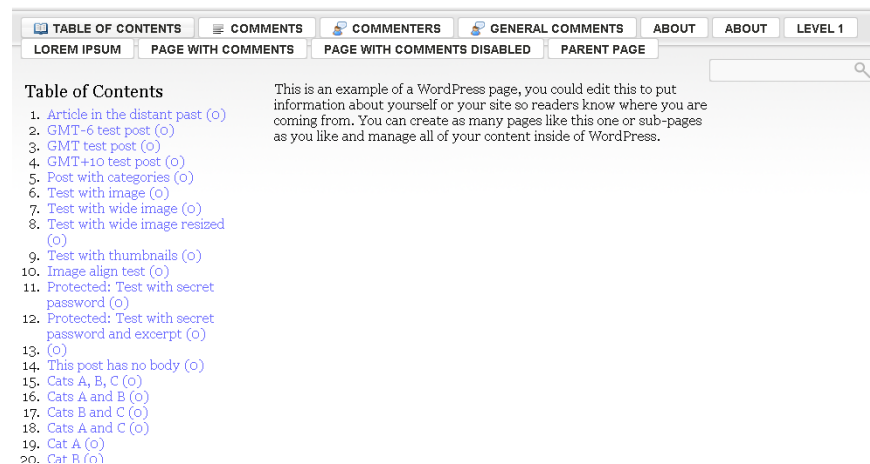
Research has shown that most users feel confused about what tag clouds do and how to use their functionality. We would recommend not including a tag cloud in order for the site to appear less confusing to users.

Appendix A

The full list of pages evaluated during the audit is given below:

Homepage

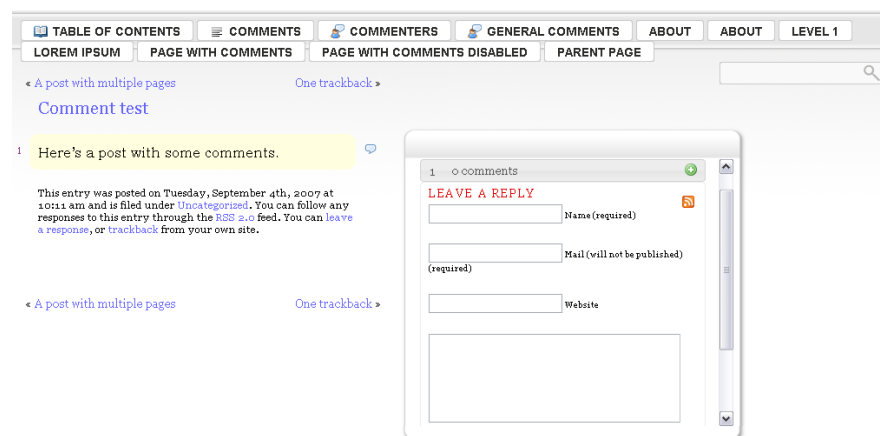
REVIEWING DIGRESS.IT FOR ACCESSIBILITY



<http://writetoreply.org/accessibility/>

Comment test page

REVIEWING DIGRESS.IT FOR ACCESSIBILITY



Powered by [digress.it](#) (version 2.3)

<http://writetoreply.org/accessibility/comment-test/>

© Nomensa Ltd

Bristol

London

E info@nomensa.com
www.nomensa.com

T +44 (0)117 929 7333
F +44 (0)117 929 7543

T +44 (0)207 872 5715
F +44 (0)207 753 2848

Page 28 of 30
25/3/10

Layout test page

REVIEWING DIGRESS.IT FOR ACCESSIBILITY

The screenshot shows a web page titled "Layout Test" with a navigation bar at the top containing links: TABLE OF CONTENTS, COMMENTS, COMMENTERS, GENERAL COMMENTS, ABOUT, ABOUT, and LEVEL 1. Below the navigation bar are tabs: LOREM IPSUM, PAGE WITH COMMENTS, PAGE WITH COMMENTS DISABLED, and PARENT PAGE. The main content area includes a search bar and a list of items. A comment box is open over the list of items, showing 7 general comments and a list of 10 items. The list of items is as follows:

- 1 This is a table:
- 2 Header two
- 3 **HEADER THREE**
- 4 Header four
- 5 Header five
- 6 Header six
- 7 This is a list:
- 8
 - o List item one
 - o List item one
 - o List item two
 - o List item three
 - o List item four
- 9 This is an ordered list:
- 10
 - 1. List item one

<http://writetoreply.org/accessibility/layout-test/3/>

Appendix B

Summary of issues grouped together by WCAG guideline:

Text Alternatives

- Non-text Content: Checkpoint 1.1.1 (A)

Adaptable

- Info and Relationships: Checkpoint 1.3.1 (A)

Distinguishable

- Contrast (Minimum): Checkpoint 1.4.3 (AA)
- Resize text: Checkpoint 1.4.4 (AA)

Keyboard Accessible

- Keyboard: Checkpoint 2.1.1 (A)

Navigable

- Bypass Blocks: Checkpoint 2.4.1 (A)
- Focus Order: Checkpoint 2.4.3 (A)
- Headings and Labels: Checkpoint 2.4.6 (AA)
- Focus Visible: Checkpoint 2.4.7 (AA)

Readable

- Language of Page: Checkpoint 3.1.1 (A)

Input Assistance

- Error Identification: Checkpoint 3.3.1 (A)
- Labels or Instructions: Checkpoint 3.3.2 (A)

Compatible

- Parsing: Checkpoint 4.1.1 (A)