1. Background

Author Steve Griffiths (steve.griffiths@guidedogs.org.uk)

Date: August 2022

We are aware that people have different ways of accessing digital services, and we want to ensure that our services are usable by as wide a variety of people as possible. We try to consider all areas of usability but the three we concentrate on are:

- Using different screen colours
- Using magnification software and
- Using a screen reader

These are not independent - a given person may use one, two or all three options at different times or in combination.

Modern operating systems contain many features that may be useful to people with non-standard access needs. These include:

- Windows - Start menu, Settings, Ease of Access (Windows 10) or Accessibility (Windows 11). Note that high contrast themes are not honoured by some browsers.
- iOS - Settings, Accessibility. The screen reader, VoiceOver, changes all gestures - make sure you know how to turn it off before you turn it on!
- Android - Settings, Accessibility. As for iOS, make sure you know how to turn the screen reader, TalkBack, off before you turn it on! Some companies that use an Android version or fork have developed their own accessibility tools, e.g. Amazon Fire's VoiceView. These tools are not covered by this document.
- macOS - Apple menu, System Preferences, Accessibility.
- Chrome OS - Time, Settings, Advanced, Manage Accessibility Settings.

For Windows, third-party manufacturers have created apps with greater functionality than the built-in features, and these are used by many people. Examples are ZoomText screen magnification, and NVDA and JAWS screen readers.

2. Checklist for Web Accessibility

(Figures in brackets are for WCAG guidelines where applicable)

2.1 Essential

- Use semantic HTML where possible because screen readers gather much of their information about a page from it.
- On a computer, the web page must be navigable with the keyboard so that screen reader users can navigate and work it. Use structural elements such as regions, headings, tables and lists to improve navigation. (2.1, 2.4)
- There must be good contrast between foreground and background colours, including hover states, and when inverse colours are used. (1.4)
- Colour must not be used as the sole way to convey information. (1.4.1)
- Screens must be scalable to 200% without the need to scroll horizontally. (1.4)
- All links, and controls such as buttons, edits, combo (drop-down) boxes etc. must be sensibly and visibly labelled, and work in standard ways. Labels such as "click here" or "read more" are to be avoided. (2.4.9, 3.3.2)
- Labels must be programmatically associated with their controls. (2.4.9, 3.3.2)
- The keyboard focus must be visible at all times when tabbing between elements, because not all keyboard users are screen reader users. (2.4.7)
- Information on the page must not dynamically update when (for example) a value in a combo box is changed. (3.2.2)
- When new information appears on the page (menu, pop-up, error message etc) this must be relayed to a screen reader and magnification user. (2.4.3)
- Images must contain alt text or a caption. For decorative images, alt text should be blank. For informative or functional images, the alt text or caption should convey the information or function, not describe the image. (1.1)
- Many CAPTCHAS are inaccessible - refrain from using them where possible. (1.1.1)
- Use clear signposting and explicit call-to-actions to help the user know what is going to happen when they do something. For instance, a button that triggers a pop-up should include this information ("has pop-up").

- If form fields are mandatory, this should be stated within the label.
- Validation error messages must be made explicit and take focus for a screen magnifier and a screen reader. (3.3)
- Animations and videos must not start automatically. The controls to pause/play and stop animations and videos must be keyboard accessible. (2.3.3)
- Check the language attributes for text - a screen reader may switch languages automatically when it detects a language tag. (3.1.1)

2.2 Desirable

- A consistent look to pages, including header/footer structure, is helpful (3.2.3, 3.2.4).
- There should only be one <H1> per page, preferably at the start of the main content. Other headings should follow a logical and consistent structure, for instance don't jump from <H1> to <H3>.
- A "skip to content" link at the top of the page is useful; it should only be visible when it gains keyboard focus.
- Forms should be constructed so that fields follow in a logical order.
- Hover actions that provoke action at a distance should be avoided. They are hard to make work with screen readers or screen magnifiers.
- Styles should be used so that if a user wishes to change the style, the change is made throughout the site.
- Tables should only be used to display tabular information; neither style-based columns nor layout tables should be used.
- Keep as much as possible to one column, left-aligned and unjustified for the benefit of magnification users.
- Pages should be scalable to 400% without the need to scroll horizontally.
- Avoid large blank areas as it can disorientate magnification users.
- Text should always be text, not a graphic of text.
- If images are used, they should include no overlaid text unless it is also included in the alt tag.
- Meaningful page titles should be used.
- Pages should be useable with commonly used user preferences i.e. Windows High Contrast Black theme and iOS larger text.

- Documentation within or separate from the system should be accessible

3. Keyboards

Some assistive computer users find it difficult or impossible to be productive with the mouse. Windows and MacOS have keyboard functionality and shortcuts built in.

A screen reader or magnifier may have its own modifier key to make it easier to include keyboard shortcuts that won't conflict with an operating system or application. This is often the Insert key by default but can be changed to the Caps Lock key. An increasing number of compact keyboards do not include an Insert key, or put it in an inconvenient place.

Some assistive apps also make use of the keys on the number pad to the right of a standard external keyboard.

For these reasons, many users of assistive apps on laptops still prefer to connect a standard external keyboard.

4. Using a screen reader

A screen reader outputs text and underlying information as synthetic speech or to a Braille display. It does more than "text-to-speech" because it can give status also, e.g. whether text is tagged as a heading, a checkbox is checked, or a link has been recently visited.

A screen reader outputs information one bit at a time, giving a linear experience. Changes to a screen in multiple places, or in unexpected places, that are not conveyed to the screen reader may be missed.

Learning to use a screen reader is not straightforward. On a computer it assumes only keyboard use, because if you can't see the screen, you can't effectively use a mouse. On a touch screen, switching on a screen reader alters the way gestures work.

On a computer, the operating system and application contain keyboard shortcuts, and the screen reader will have additional keyboard shortcuts that are useful or essential to know. For using a browser, common keystrokes are:

- **Tab** - moves to the next interactive element (link or control).
- **Shift + Tab** - moves to the previous interactive element
- **Space** - toggles the status of a checkbox.
- **Enter** - activates a button or link.

Additionally, some useful Windows screen reader keystrokes include:

- **Arrow keys** - moves the smallest amount in any direction (add **Ctrl** to move by larger amounts).
- **H** - moves to the next heading.
- **B** - moves to the next button.
- **Insert + F7** - opens a list of links on the current page.
- **Ctrl** - mutes the screen reader.
- Insert + 1 - toggles keyboard help on and off. While on, press a key to hear its function in the current context.

Use of single letter keystrokes is possible because a screen reader has two modes in a browser - navigation and text-entry, with the former being the default. To switch to text-entry on an edit area, you may have to press Enter; to revert to navigation mode you can press Escape.

On a touch-screen device, standard gestures become:

- **Single tap** - identifies an item.
- **Double tap** - activates a link or control
- **Swipe right** - moves to the next item.
- **Swipe left** - moves to the previous item.

5. Forms and form fields

Forms must be linear, logical and clear. Form fields must be labelled clearly, uniquely and programmatically. Errors must be indicated to all users.

Use standard HTML-tagged field elements where possible. Check boxes, radio buttons, buttons, links and drop-downs are all examples of elements that have been in use for many years and work well with assistive technology. Screen readers scrape information from the HTML code of a page and present it to the user. Also, there are standard ways a keyboard interacts with controls, and people will expect these to work.

Where possible, all the above should apply to external screens such as postcode lookups or payment screens, that are part of a user journey.

6. Alt text

Screen readers can read text but not images, and someone using a screen reader will need to be told what information an image conveys. This is done through alt tags added to the image, or a caption.

Informative images convey useful or supporting information that isn't included elsewhere on the page. This includes functional images like logos, and images that attempt to convey an emotion. For these images there should

be an alt text of one sentence, up to approximately 20 words. The alt text should convey the information held in the image, not describe it. For instance, "no entry sign" rather than "red circle with white bar through the centre". What counts as important may change depending on the context.

The reason for keeping alt text short is that often a screen reader reads it all in one go, rather than allowing you to choose to read it a word or a sentence at a time, as with most text. If the alt text is complex, it may have to be read multiple times; if it's also long, that may make rereading it tedious.

A screen reader will preface the alt text with "graphic" or "image", so you don't need to include that.

Some images are purely decorative; they do not add anything to the page. For these images, the conventional view is that there should not be any text within the tags so that a screen reader doesn't spend time reading non-important information. However, some screen reader users want to know if an image is used, even if it imparts no information. Therefore, if in doubt about whether an image is decorative or informative, assume the latter and add alt text.

For complex images such as maps or charts, where more than a sentence is needed to convey the information, the alt text should include a short description and a link to a longer description. There is an HTML tag for this latter case called <longdesc>, but its worth is debated and it's probably best not to used it.

7. Testing

Testing can be done automatically or manually. Automatic testing allows whole websites to be tested swiftly but cannot provide the interpretation and nuance so often required when applying guidelines to real world situations. Manual testing can provide page-by-page analysis but is time-consuming and requires knowledge of a variety of assistive tools.

It's relatively easy to test screen colours, zooming and magnification, because they change the way things look, but not how they work. The necessary tools are also built into the operating system.

- On a computer, you should be able to use the pages without using a mouse:
  - Can you Tab to all the interactive elements on the page in a logical manner?

- Is there a visible indication of the keyboard focus as you Tab?

- Using the keyboard or a mouse, can you zoom the browser to 200% and still use the page? Use Ctrl/Cmd Plus and Ctrl/Cmd Minus to zoom.
- When you invert the colours of the page, can you still see and use everything? Common issues are buttons losing their outline or icon, or radio button/check box selection statuses not being distinct.

Learning to use a screen reader is more complicated because it changes how you interact with your device. On a computer you can only use the keyboard, and on a touch device gestures work differently. For PCs, most screen reader users use a third-party application. Sighted people using a screen reader to test a page must learn to rely only on the screen reader for the information they need - unplugging the mouse and turning off the monitor can help. Examples of things to check are:

- Can you get to everything visible on the screen?
- Is there structure to a page? E.g. to help you navigate past repeating information such as a header? Are elements that look like buttons or headings coded as buttons or headings? On forms, are elements labelled, and mandatory fields indicated? Are tables navigable by row or column?
- Are you told when new information appears on the screen, for instance when you submit a form successfully, if an error occurs, or if a pop-up appears?

WebAIM have created an introduction to using NVDA to evaluate web accessibility.

7.1 What we test with in Guide Dogs

We test with at least the following assistive technology and browsers:

- Windows 10: JAWS/Chrome, NVDA/Chrome, NVDA/Firefox, Narrator/Edge, keyboard only and zoom up to 200% on each browser
- iOS: VoiceOver/Safari
- Android: TalkBack/Chrome
- All platforms: built in magnifier and invert colour option

8. Browsers and inverted colours

Inverse colours are an important option liked by many people. There are different ways to invert colours in a browser, but the Windows High

Contrast Black (HCB) theme has been around for many years and is well-used, and for that reason it is probably the best option for testing.

You can turn on HCB from Start > Settings > Ease Of Access > High contrast; turn on high contrast and then ensure "Choose a theme" is set to High Contrast Black. The option has the shortcut Alt + LeftShift + PrintScreen. Firefox users may need to go to Settings > Colours, check "Use system colours" and choose "Only with High Contrast themes" for "override the colours specified..." before Firefox will honour HCB.

Note that HCB does more than simply invert a browser's colours; it also changes the look of some Windows elements.

Other methods of changing colours within a browser include browser settings or dark mode (don't apply to all elements of a browser screen), Windows 10 colour filters, or the use of high contrast extensions such as Dark Reader, Midnight Lizard or Google's High Contrast. These all work slightly differently and therefore are worth considering but if time is tight, stick to HCB!

9. Resources

9.1 Third part assistive technologies

NVDA - free! This includes an open-source synthesiser that many people find too robotic. Windows 10 includes some good alternative voices.

JAWS costs, but it's possible to download a demo which can be run repeatedly for 40-minutes per boot session. The EULA does not specifically prohibit using 40-minute mode for testing, but a 90-day Limited Timed License is available and intended for the purpose of testing.

The JAWS support pages include Surf's Up, an interactive guide to using a browser with JAWS, although much of the information is useful for any screen reader.

ZoomText and SuperNova both cost but have a 30-day demo.

9.2 Guidelines and testing tools

Web Content Accessibility Guidelines **(WCAG)**

These are the bedrock for most accessibility work. They are large and complex, and not often updated[1]. It's worth reading at least a checklist of WCAG such as those by Luke McGrath or WebAIM, but an in-depth

[1] v1 1999, v2 2008, v2.1 2018. V2.2 scheduled for September 2022 release, v3 is also being worked on

understanding is only for specialists. Websites that are serious about accessibility will aim to conform to at least the AA level of WCAG 2.1.

Guide Dogs go beyond WCAG AA, for example by explicitly preferring semantic HTML, wanting links to be identifiable from the link text alone (2.4.9, WCAG AAA), aiming to apply good colour contrast not just text but icons that can be actioned, and supporting sections 11.7 (user preferences) and 12 (documentation) of EN 301 549.

The Paciello Group

A world leader in all things related to accessible websites, they were acquired by the umbrella company that owns JAWS in 2017. Their colour contrast analyser is a very useful tool (Chromium browsers now include similar information in Inspect Element Mode, Ctrl + Shift + C).

Webaim

Have some good information on how to code controls and have created a widely used automated accessibility testing tool called WAVE. They also publish the Screen Reader User Survey, which is the only one of its kind.

Deque

Creators of axe automated accessibility tools.

10. Examples of common issues

Form errors not indicated to a screen reader user. How to provide accessible form error identification gives an overview of how to correct this. Here are some examples of good practice:

- On the Guide Dogs volunteering page, search on a postcode, check "Apply" for one of the Fundraiser roles, choose "No" for "have you applied within the last 12 months", then choose "Next: Apply for chosen roles". The next page contains a form; leave it blank and choose "Next: Address". Focus jumps to the first field with an error and a screen reader reads the field and error details. You need a screen reader on to fully appreciate it!
- Paciello contact form - leave it blank and choose Submit. Again, you'll need a screen reader on to appreciate the result.
- Cantina have a video demonstration which doesn’t require you to have a screen reader turned on!
- Salesforce have a good example when you need to create a new password, but to get to this you need a Salesforce account. The form puts help text

next to the fields e.g. "too weak", "good" or "match", and (if you type slowly) a screen reader reads these out.

There are other ways information can be added to a page, e.g. with an address search or through the use of accordion buttons. Labelling an accordion correctly means it carries the status of whether it is expanded or contracted. When new information is added to a page, the new information should take focus. Ideally the new information will be added below the current keyboard focus so as to not interrupt the flow of the user journey.

Mismatch between visual and coded information. Designers often want to make a link stand out as a call to action, and do this by coding it as a button. This can be confusing to low vision users because buttons usually perform an action (e.g. add to cart) while links take you to another location or page (and you can then press the Back button to return to where you were). Also, keyboard users may be told by a sighted colleague to "press that button" and use a keystroke to move to a button and not find it.

Radio button labels. Radio buttons have two labels; the overall label (fieldset) for the control e.g. what drink do you prefer? and the label for the particular option e.g. tea, juice, water. Both labels need to be associated with the radio button, otherwise a screen reader user hears the options but doesn't know what the question is.

Journey steps/progress bar. These often appear as numbers at the top of the screen; labelling them as numbers doesn't give the full story - usually a sighted user can see a total number of steps with one highlighted. So a label like "step 2 of 4" is more useful; or add that info into the main heading on the page: "step 2 - contact details".

Here are some more links to explanations of some other issues and examples of accessible coding:

- Skip to main content
- Form controls
- Date pickers - Axess Lab date picker, Mozilla date picker, date picker with disabled weekends
- Colour contrast
- Two views on alt text - one from Scribely, the other from Axess Lab
- Creating links - front load for links list

- Pop-up dialogue - has a Live Example near the bottom. Pop-ups need to take focus and be navigable.
- Pop-up menu, approach one. If you have access to Jira, each ticket has an Actions menu at the top right which also works pretty well.
- Buttons
- Radio buttons with HTML, radio buttons with CSS and radio buttons with aria
- Accordions
- Required form fields
- Autocomplete edit
- HTML email
- Responsive table
- There are occasional times when you want to bring keyboard focus to something that is not interactive. You can use Tabindex=0 for this.
- Megamenus - these must be carefully built-in order to make sense and be operable with the keyboard. An example is the International Association of Accessibility Professionals - or look at the Guide Dogs shop!

Guide Dogs

11. And finally…

The links in this guide take you to specific pages but have been chosen also to introduce you to some useful jumping-off points for further exploration. Here are some more that haven't yet been mentioned.

- Most guides are aimed at designers or developer or content creators. The "Must-Have Accessibility Handbook" has sections aimed at all three. (It also, unfortunately, has a good example of a dropdown list that is coded as a button and so doesn't work in the expected way with a keyboard!)
- Webaim, Mozilla and the Accessibility Developer Guide have lots of explanations and examples of accessible coding.
- Accessibly labelling interactive elements. Up until now I've deliberately not mentioned ARIA (accessible rich internet applications), which is an attempt to plug the accessibility shortfalls in HTML, because it's often seen as a quick fix but ends up as a disaster. The first rule of ARIA is often given as "don't use ARIA" but if the information in this guide doesn't give you want you need, you may want to look at a beginner's guide to ARIA or Deque's Top 5 rules of ARIA to get you started on ARIA.
- One good use of ARIA is where a link is used, but is formatted like a button. If it's not possible to recode it as a button, you can give it an ARIA role of button (and include JavaScript to make it respond to pressing Space).
- CAPTCHAs aren't as widely used as they used to be, but they still present serious accessibility issues which this article outlines.
- A web for everyone: designing accessible user experiences by Sarah Horton and Whitney Queensberry (2014)
- Accessibility for everyone by Laura Kalbag (2017)
- Mismatch: how inclusion shapes design by Kat Holmes (2018)
- Inclusive components by Heydon Pickering (2018)

End of document