Accessibility checklist for designers and developers

This checklist has been created to help achieve a minimum of WGAC 2.1 AA compliance for accessibility. The checklist also goes further to try and ensure we are supporting those visiting our sites using assistive technologies or have disability that may make viewing a website more challenging.

1. Colour and contrast

- Test colour contrast all text, features and content on the site. See DCC's colour palette for guidance on colour combinations.
- Ensure any text that overlaps images has sufficient contrast or has a background colour applied to it to ensure it is readable.
- Test all page designs for different types of colour blindness

See accessibility tools document for useful testing tools.

2. Page structure

2.1. Language attribute

- Declaring a language attribute on the HTML element enables a screen reader to read out the text with correct pronunciation.

  <html lang="en_GB">

- Specify a language with the lang attribute on the html element.

2.2 Landmarks

There a 7 landmarks that should be considered - If the site is using HTML5 (required for any new sites) then the HTML5 tags below should be used. If this is not possible then the equivalent ARIA "roles" should be used on the containing tag, for example

HTML5 : <header>
Non HTML5 : .

HTML 5	ARIA Role
<header>	role="banner"
<nav>	role="navigation"
<main>	role="main"
<footer>	role="contentinfo”
<aside>	role="complementary"
<section>	role="region" *
<article>	role="article" *
none	role="search"
<form>	role="form"

* The region and article roles are not ARIA landmarks but should still be used where appropriate

How should these landmarks be used:

1. **Header role=”banner”** A region of the page that is site focused. Typically your global page header.

2. **Nav role=”navigation”** Contains links to navigate to different pages of a website, screens of an application, or a sections within a single document.

3. **Main role=”main”** Wraps the focal content of document. Use only once.

4. **Article role=”article”** Represents an independent item of content such as a blog post or news item. There may be many articles in a single document. articles are not considered landmarks, but screen readers may still surface articles when navigating by regions or landmarks in a document.

5. **Aside role=”complementary”** Supporting section related to the main content even when separated (for example, the sidebar).

6. **Section role=”region”** A section is a thematic grouping of content, typically with a heading

7. **Footer role=”contentinfo”** Contains information about the document (meta info, copyright, company info, etc).

8. **role=”search”** Add a `search` role to your primary search.

Document outline

- Use semantic headings and structure. For example, ensure that heading tags are appropriately nested and that every page has a single h1 tag that describes the page’s content.

  While new HTML5 standards do suggest that multiple H1 tags can be used for articles and sections, screen reader software and W3C guidelines haven’t quite caught up with that as a standard.

- Look at header, footer and sidebar.
- Ensure all sections have a header even if that header is only available for screen readers or those viewing the site without css.

3. Page features

3.1 Navigation

- Esure aria-haspopup="true" and aria-expanded="true/false” are used when a navigation item has a dropdown sub menu.

3.2 Call to action / listed items

- Ensure a consistent approach is used. For example: banners should be consistently presented so users can quickly understand their layout and what elements are links, content blocks or media.This is especially important for screen reader users.

4. Page content

4.1 Text

- Ensure all copy on a website has appropriate text spacing, line-height and padding.
- Ensure full width text is not used unless the font size is sufficient to only have 90 characters on one line
- Users can resize text up to 200% without the layout or page breaking

4.2 Links

- Ensure links are recognisable - underlined is preferred.
- Ensure links have :focus state and are styled correctly. For example: Solid focus colour for buttons / banners, border top and bottom for text links
- Check link content is useful and describes what a user will get if clicking on the link. If something descriptive can't be used in the link text then ensure a descriptive "aria-label" has been added. For example: aria-label="telephone number 01392 383000"
- Ensure links tab in a logical order (use tab indexing if needed although good HTML structure should mean that this happens correctly without needing to define a tab index)
- Provide a 'Skip to main content' link that goes to the main content landmark. For example: <main> or
- Ensure the site doesn't have any empty links. For example: a link with just an icon in will appear as empty. Some supporting text should also be available but can be hidden using text-intent or another similar alternative. Do not use display:none
- Check for redundant links

4.3 Buttons

- Ensure all buttons have a descriptive title or aria-label for users with screen readers

4.4 Images

- Use alt text that describes what's in an image. For example "group photo of outside outside smiling" ??????
- Check header, footer, sidebar and fix any issues with images that are not in library.
- If the alt attribute is empty, ensure the image is purely decorative.
- Avoid using "Image of" or "Picture of" as the screen reader will notify the user that it's an image. Also avoid using all caps as some screen readers will read each letter, i.e. W-A-R-N-I-N-G.

4.5 Media (Video / Audio)

- Ensure all video content includes subtitles
- Where reasonable, transcripts should be created to support audio and video content
- All video controls should be accessible by keyboard

4.6 Tables

- Tables have proper headers and column attributes

4.7 iFrames

- When using iframes, it's important that all content contained in them is accessible.
- Ensure a title=” or name=” attribute has been specified that describes the content of the iframe

4.8 Target size

- Any link/button or interface item that requires a click or press should be at least 44px by 44px. For example a button to close a modal.

5. Forms

- Logical layout - Tab order of the form follows a logical pattern.
- Associated label for all form controls (eg input, select etc.)
  (e.g. <label for="name">Name:</label><input id="name" type="text">)
- Make sure placeholder attributes are not being used in place of label tags. An exception to this may be where a smaller form is being used. Smaller forms can have labels hidden using css as long as screen readers are able to read the label.
- Group related form elements with fieldset and describe the group with legend
  Important for input type="radio" and input type="checkbox"
- Identify Input Purpose - See https://www.w3.org/TR/WCAG21/#input-purposes for input purposes to be applied
- Where ever possible use auto fill form fields to help visitors with physical/deteratory limitations
- Ensure where ‘name=’ is used, it’s value is the same as the text that is presented on a label, button or other interface component.
- Error messaging should be helpful and aim to guide a user to a correct approach

See our form design patterns for more detail on presentation and errors.

6. Page functionality

6.1 Screen orientation

- Do not fix orientation, it becomes an issue for people who cannot turn their screen

6.2 Hover or popup content

- **Dismissable**

Open and close buttons should be in the same place - accessible with moving the mouse or keyboard cursor

- **Hoverable**

Ensure additional content that appears on hover doesn't disappear when your mouse moves from the button or trigger to the content being displayed

- **Persistent**

The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

6.3 Timeouts

- Ensure that if a timeout exists, users are made aware or are giving a prompt if it's about to expire. This is a AAA requirement though but helpful nonetheless.

6.4 Flashing

- Flashing is generally a bad idea. It can cause all sorts of issues, from seizures to motion sickness. If you absolutely must have a flashing element there are a few things to consider.
    - Look for elements that contain flashing
        - Scrolling or blinking text
        - Scrolling or blinking page elements
        - Videos that contain flickering or flashing
        - GIFs that contain flickering or flashing
    - Check if you can determine the frequency of "flashing."
    - Check that the rate of flashing is less than 3 Hz (3 times a second), or scroll delay is set to >= 400.

6.5 Javascript

- Avoid using inline javascript, opt for linking to javascript in and include document
- Provide alternatives for users who do not have JavaScript enabled and for environments where JavaScript is unavailable.

6.6 Input Modalities

- **Pointer gestures**
  Create alternative ways to navigate functionality where gestures are used - for example - pinch and zoom should have alternative buttons to do the same functions (Mapping)
- **Pointer cancellation**
  Do not use the click down event to run any function

7. Browser support

- Ensure that the website fully supports colour and text size changes that have been set in the browser.

8. Hidden content

Hiding content is very useful for accessibility. We can hide things visually and only display it to screen reader users, we can hide content from screen reader users and only show it visually, or we can hide content from both. Below are some examples of best practice

Technique	**Visually hidden**	**Screen reader hidden**	**More info**
CSS: `.sr-only { border: 0; clip: rect(0 0 0 0); height: 1px; margin: -1px; overflow: hidden; padding: 0; position: absolute; width: 1px; }`	Yes	No	There are multiple ways to accomplish this with CSS. This is the current way we are recommending it.

CSS:
`{`
`display:none;`
`}`
	Yes	Yes	

HTML5 attribute:
hidden
	Yes	Yes	In supported browsers, this is the same as
			`{`
			`display:none;`
			`}`

aria attribute:
aria-hidden='false'
	No	No	This is overwritten by other techniques. i.e. Using
			`{`
			`display:none;`
			`}`
			
			will cause the element not to be read or seen.

aria attribute: **aria-hidden='true'**	No	Yes	

The Process

This checklist has been created as a guide for designers and developers to use when creating websites. The checklist is the result of a period of research, review and testing conducted by DCC's digital communications team. This is a 'live' document that will be regularly reviewed and updated as more planned testing and user engagement is completed.

To date the following research and activity has been completed:

Review of a number of websites using the following setups or environments

- Modern browsers
- Keyboard only navigation
- Screen reader (tested using NVDA and Chrome Vox)
- CSS disabled

- Javascript disabled

It is our aim to ensure disabilities are support within what's reasonable for each specific site and the people who are using it. So far we have paid specific attention to:

- Diversity of abilities
- Auditory
- Cognitive, learning, and neurological
- Physical
- Visual

Tools

- Funkify
- Colour contrast checker
- Colour contrast Analyzer
- WebAim Wave tool
- Landmarks

The Landmark browser extension can be found here:

- Google Chrome
- Firefox

Is this useful?
https://webaccess.berkeley.edu/resources/tips/web-accessibility#accessible-ARIA

Credits
https://accessibility.18f.gov/
GDS Design manual
A11y Web Accessibility Checklist
WCAG 2.1 - What's new article