University of Cambridge Computing Service

Accessibility Guidelines

Helen Sargan

Minor updates June 2011

1. Introduction to accessibility

The University of Cambridge is committed to making accessible for all users information and resources that are available via the web. We favour the principles of usability and universal design to ensure that all materials accessed via the web are usable and effective for everyone. By following these principles and incorporating techniques to ensure information and resources are accessible with assistive technologies, separate resources for disabled users should not be required.

Background

First in 2001 and again in 2005 the Government passed legislation that requires public bodies to promote equality of opportunity for disabled people (see http://www.cam.ac.uk/site/accessibility/ for further clarification). The Equality Act 2010 has superseded these laws but the guidance has yet to be published on how it might affect these guidelines.

In the 1990’s the World Wide Web Consortium (W3C) formed a Web Accessibility Initiative (WAI) to give guidelines and supporting information about how to accomplish accessibility as a whole (considering browsers and authoring tools as well as web content), the guidelines about content was published in 1999 as the Web Content Accessibility Guidelines (WCAG) 1.0 (see http://www.w3.org/TR/WCAG10/). These guidelines are extensive and suggested a number of measurable parameters, which gave rise to a number of tools that could be used to test a web page and give an accessibility ‘result’ and therefore a mark of success. While a good many of the parameters measured could be useful for those with disabilities (such as whether there was an alt text attribute for an image), many features that could not be tested for (such as colours used together and their contrast) were left out of the ‘result’, making it only a partial measure of success. Without a full understanding of what elements of a page needed to be assessed and why, the guidelines couldn’t be fully useful. So almost immediately the WAI starting working on a new set of guidelines that would give a better representation of whether web content was accessible – the Web Content Accessibility Guidelines (WCAG) 2.0 (see http://www.w3.org/TR/2008/REC-WCAG20-20081211/) were published at the end of 2008. The following appears at the start of these guidelines:

“Web Content Accessibility Guidelines (WCAG) 2.0 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these. Following these guidelines will also often make your Web content more usable to users in general.”

These laudable aims must be set in the context of the legislation that states we must “take reasonable measure” to make the information available to our users, not that you must make available the information in every way for every possible user group (which would be a near impossible task). We don’t have to avoid video or colour on pages because some people can’t get the full benefit – we just need to be aware of potential difficulties and work around them.

What are these guidelines for and who needs to read them?

The University Accessibility statement (http://www.cam.ac.uk/site/accessibility/) states that:

- All **new** web pages should be written to at least conformance level 2 standard (AA) of Web Content Accessibility Guidelines (WCAG) 2.0, available at http://www.w3.org/TR/WCAG20/.
- All **existing** pages should meet at least conformance level 1 standard (A) of Web Content Accessibility Guidelines (WCAG) 1.0 (http://www.w3.org/TR/WCAG10)/.
- **Most pages** should meet conformance level 1 standard (A) of the newer guidelines by 1 June 2011. A development plan should be in hand to make all pages conformant to at least A level within as short a time as possible.

These guidelines have been written to help you through the process of producing web pages that conform to these standards.

You need to read these guidelines if you are creating a website for a University department or institution or if you are taking over or adding information to an existing website.

- Those who are creating a website need to design the templates and/or the content management system so they produce pages with accessible content.
- Those who are editing or adding content to templates need to do so in a way that doesn't pose accessibility problems for users.

How to improve accessibility

A few actions can help a lot. These guidelines and checklist summarise how best to concentrate your efforts to achieve accessible web content. To be most time-effective this ought to be part of a workflow that includes the following:

- For **producing and checking xhtml/html** create a work environment with a set of appropriate tools (further details in 'Evaluating your pages')-
    - an **up-to-date html editor** to help ensure you are creating valid code
    - a **set of web browsers** that reflect what your expected audience are using (the most recent two versions of MS Internet Explorer, Firefox, and Safari)
    - a **set of tools** to check and test your code
- For ensuring consistency between pages, you should use **a set of templates** that incorporate appropriate accessibility measures and have been code tested. The University department templates are there for use if you represent a department or institution, or a part of. If you can't or don't want to use them, designing a set of templates should be the first step to producing content.
- If you are **producing or providing content that is not xhtml/html**, such as slide presentations (PowerPoint, etc), pdfs, audio or video, or Flash, ensure you are working with up-to-date applications (since built-in accessibility aids have been added to many in recent releases) and have explored the ways of making these types of content inclusive. Create or utilise accessible templates. If you are using Javascript, write accessible Javascript that is usable by assistive technology software or make sure your pages are usable without the Javascript.

Offer help

Bearing in mind that we must we must "take reasonable measure", ensure that on your website it is clear that users who are unable to access information can ask for that information in another form, and that you will do your best to make this happen.

Guidelines for producing accessible (inclusive) web pages

The four principles used in WCAG2.0 are that information should be:

- **Perceivable** - Information and user interface components must be presentable to users in ways they can perceive.

This means that users must be able to perceive the information being presented (it can't be invisible to all of their senses)

- **Operable** - User interface components and navigation must be operable.

This means that users must be able to operate the interface (the interface cannot require interaction that a user cannot perform)

- **Understandable** - Information and the operation of user interface must be understandable.

This means that users must be able to understand the information as well as the operation of the user interface (the content or operation cannot be beyond their understanding)

- **Robust** - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

This means that users must be able to access the content as technologies advance (as technologies and user agents evolve, the content should remain accessible)

If you are familiar with WCAG1.0 you can see that this approach is different, but after interpretation of the WCAG2.0 guidelines and addition of guidelines for good usability, the steps to accomplishing inclusive content break down into the following:

1 **Structure**: Ensure your pages are properly structured xhtml/html and that text works effectively both for communicating the information and being easily visible and adjustable.

2 **Layout**: Make sure the templates for your pages include relevant inclusive navigation aids

3 **Validation**: Ensure your templates and your pages are valid xhtml/html, and that your stylesheets are valid css. Any Javascript should also be validated.

4 **Style**: Use stylesheets (CSS) rather than formatting mark-up. Ensure that the page makes sense if the stylesheet is not used. Use a multipurpose stylesheet, with additional 'special use' stylesheets so they are easy to maintain. Ensure you have an adequate print stylesheet.

5 **Colour**: Pay attention to colours used (the contexts are described below)

6 **Images**: Be cautious about the number of images you use. Always optimise graphics and use descriptive alt tags unless they are purely decorative, in which case use a null alt tag (alt="")

7 **Non-xhtml/html content**: Moving images, back-end databases, javascript, java may all reduce or block access to the information for some

8 User experience: Test for user experience against a number of different graphical browsers (and different versions of the same browser) on different platforms, not forgetting smartphones and test either with lynx, the most widely used non-graphical browser, or with a speaking browser or screen reader. Make your pages dynamic. If fixed width is required, use a 'standard' width of, say 800 pixels, set with the stylesheet.

9 Don't use **old technologies** such as frames or image maps

1 Structure

Properly structured xhtml/html

Good xhtml/html content should be properly structured with a title, metadata, headings, paragraphs, lists, etc. This structuring will make adding styling easier and it will add to the meaning of the content, particularly for those who use tools to listen to the web pages and also for search engines. Using a recent xhtml/html editor may ensure that your code is correct and can be checked within the editor.

In the future html5 will start to become more common but it will take several years (at least) before all the common browsers and assistive technologies in use support it, so don't remove current accessibility solutions in a hurry. See http://webaim.org/ for further information.

Effective text

Users become less willing to read down when pages get long. Various measures will help all readers get more from a page:

- using an active voice to give the text energy
- putting a summary at the top of the page

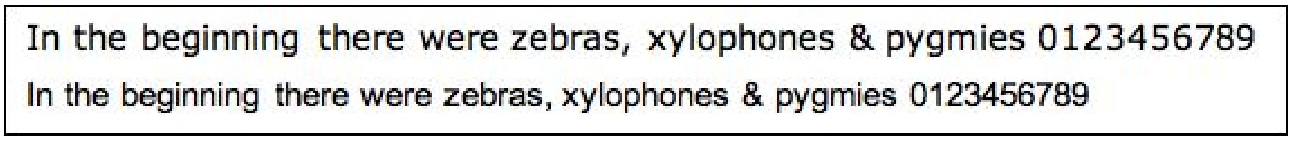
- breaking information into chunks
- making lists of points rather than using a narrative style
- to enhance readability, make sentences 20 words or less, and paragraphs 6 sentences or less.

Although the text as a whole should be written clearly, pay particular attention that the title of the page and the links within the text are in clear and concise language.

Adjustable text

You cannot hope to make a web page for which the size of the type suits everyone. The aim is for pages that are comfortable to read in standard browser setting by people with normal eyesight, but on which the type can be scaled to a larger (or smaller) size by the user and the page still works as a whole. To allow for this user-based resizing, the font sizes in your style sheets should be relative (sized in ems or percentages rather than pixels). This is all taken care of in the University stylesheets, but for working with your own, see further information in the article ‘How to Size Text in CSS’ at A List Apart (http://www.alistapart.com/articles/howtosizetextincss).

One of the clearest and most readable typefaces to use for continuous text on web pages is Verdana. It is the screen-adapted version of Arial – the adaptations are visible in the capital I and the number one, the chunkier type and more defined inter-letter spacing (see below for comparisons of Verdana [top] and Arial [bottom], both 12pt):



In the beginning there were zebras, xylophones & pygmies 0123456789

In the beginning there were zebras, xylophones & pygmies 0123456789

2 Layout

Order of elements on the page

The layout of your pages should reflect the logical order of the information on the page, so that people using screen readers hear the information in a sensible order. The best way to make sure your templates work effectively is to draw out a diagram of the parts of the page relative to each other and satisfy yourself that the ordering of them is what you need. The most common problem areas are the placing of tabbed navigation and page headings relative to the left hand navigation of the content and the text content of the page itself, and the use of tables or positioning.

- WAVE is a very effective tool for looking at this problem – put your url into the page at http://wave.webaim.org/ and look at the results under the tab ‘Structure/Order’.
- In the Firefox web developer toolbar, go to **Information > Display div order** to see similar information. You can also use **Miscellaneous > Linearize page** to see the logical order of the information in the page.

Appearance of the page

For design purposes space on a page is vital however too much space between elements can be problematic for those using a screen magnifier. Make sure the final layout is cohesive and doesn’t have space that such a user could get lost in (all common operating systems have a screen magnifier built in that you can test with).

Navigation

Consistent navigation at several levels is essential for a website. The levels of navigation available should include some of the following

- global navigation that is included on every page, usually associated with a search box (essential)
- Tabbed navigation to take users to well-defined sections of a site
- Breadcrumb trail to cue users about where they are within the site and allow them to step upwards (essential)
- Left-hand navigation for finding your way between sub-sections or pages in a section, which may include sub-navigation (essential)
- ‘Contents list’ navigation at the top of a long page

Within each of these navigation elements, there needs to be some sort of flag to show users where they are – an 'active' state that changes the appearance of the element. Use of colours in navigation needs to be approached with some care (see section on colour for the issues involved and how to tackle them). Any scripted navigation must be usable by keyboard as well as by mouse.

'Invisible' navigation

Extra elements can be added to a page that are spoken but are styled in such a way that they aren't seen. Some of these elements are used and others maybe not much but once they are provided in your template they are no extra work to maintain. They include:

- 'Skip' links, very usefully allow users to bypass the global information at the top of a file and get to the content of the page (see http://www.rnib.org.uk/professionals/webaccessibility/designbuild/navigation/Pages/hidden_navigation.aspx)
- Access keys – use numbers as some letter-key combinations are reserved for operating systems (see http://www.rnib.org.uk/professionals/webaccessibility/designbuild/navigation/Pages/access_tabs.aspx for information). Because of this access keys present some problems and help pages will be needed to explain what access keys are available.

3 Validation

If you have used a good xhtml/html editor to write your code you might already trust that it is valid, but you still ought to validate it. The dtd in the head of the file, which might look like this:

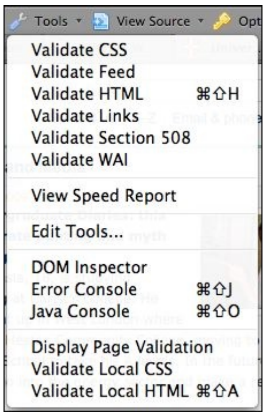
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

tells a browser what type of page you have created, and the browser processes and renders the page accordingly. The validator checks the code against the dtd and tells you if you have any errors, which you then ought to fix, as they could present traps for browsers and cause your pages to be rendered wrongly.

Stylesheets can be validated in a similar way. Javascript (and other code) should be validated before use.

Validation tools

One of the easiest ways to get quick access to validators is to use Firefox and install into it the web developers toolbar (http://chrispederick.com/work/web-developer/). This gives access to many useful tools, mentioned elsewhere in this information, but for validation you will find what you need under the 'Tools' menu:



The options at the top of the menu will directly validate the page currently being viewed in the browser, while at the bottom of the menu there are options to check pages that are being viewed locally (not available through a url). The validators being used are already configured but you can change them if you wish to use a different service.

Another handy extension in Firefox, HTML Validator (http://users.skynet.be/mgueury/mozilla/), adds a little icon to the bottom right of your window, which is a green tick if the page is valid, a warning sign if there is a possible problem, and a red cross if there is an error. This is a really quick glance way of seeing if there is a problem with a page, although it shouldn't be entirely trusted.

4 Using stylesheets

For accessibility purposes (and in order to be up to date) you should use web templates that are controlled by stylesheets, with no formatting in the code of the pages. If you have never used stylesheets before, a good introduction and set of tutorials can be found at HTMLdog (http://htmldog.com). Providing the web pages are composed of properly structured xhtml/html and the layout allows the information to sequence in the correct order, the information on them should be usable with the style sheets turned off (which you can view via the Firefox web developers toolbar).

As noted above, stylesheets can and should be validated and this can conveniently be done via the Firefox web developers toolbar.

A main stylesheet, with additional 'special use' stylesheets will be the easiest to manage and maintain. If you want to use the University templates and add extra styles then we advise you add an extra stylesheet for them. Ensure you have an adequate print stylesheet and then you may be able to minimize the number of pdfs in use. Make good use of notation with stylesheets. This is especially useful for styles added for particular purposes and for noting changes. Loading long stylesheets is an overhead in the time taken to render a web page, so remove from stylesheets styles that become redundant.

5 Colour and web pages

There are a number of different issues with colour that need to be addressed – if you do this for your template as whole you will only have to do it once:

- There needs to be adequate contrast between type and background (this contrast is given a value for the different levels of accessibility). The level of contrast can be measured by using appropriate tools, either via the web accessibility toolbar (for Internet Explorer only – see http://www.wat-c.org/tools/index.html) or via the Firefox Colour Contrast Analyser (https://addons.mozilla.org/en-US/firefox/addon/7313) or WCAG Contrast checker (https://addons.mozilla.org/en-US/firefox/addon/7391) extensions. I find the last one of these the easiest to use.
- Since you have no way of knowing whether users can see the page in colour, you must not use colour as the sole means of distinguishing between information – for instance the right answers in one colour and the wrong answers in another – the difference must also be distinct without the use of colour.
- You must select your colours keeping in mind that 8% of northern European men are red/green colourblind (see http://www.visibone.com/colorblind/ for red/green colour blind view of web-safe colour palette). http://www.vischeck.com has on it a tool that simulates colourblind vision – it doesn't work on all web pages but allows you to upload images as well.
- When you add styling for any link colours, make sure you style all of them otherwise a user-selected value might intervene and cause problems with legibility.
- For third world users in particular, using the web-safe colour set (216 colours) is important to ensure cross-browser and cross-platform consistency.
- Alternate-row background tints can be very useful for helping eye tracking in tables or lists.

The Firefox web developer toolbar **Information > View color information** will show you the colours in use on a page.

6 Images

As with colours there are a number of issues concerned with the use of images.

- Graphics on a web page will make loading the page take longer, so it is wise to keep the number low. You can get an indication of the size of your images from the Firefox web developer toolbar **Images > Display Image File Sizes** and how long a page takes to load **Information > View document size** or via many other tools
- Make sure the images you use are optimised for use on the web – they should be at 72dpi and are at a size suitable for their use on the page. Don't use a bigger file than you need and resize it in the browser. When possible do add height and width measurements in the browser to allow for faster loading of the page.
- Use descriptive alt tags unless the image is purely decorative, in which case use a null alt tag (alt=""). For a functional image, such as a button, use an alt tag that describes what the button is for unless it is part of a link that already does that, in which case the alt text would duplicate it. Some image

links duplicate nearby text links, so look carefully at the page with images disabled to check that only one set of links appear and that they are clear.

- Avoid using a background image on a page as it can make text impossible to read. If you have to use a background image, also use a complimentary background colour so the text is legible if the image doesn't load.
- Avoid using moving images (by default) for the following reasons:
    - they can be very distracting and movement may stop the user from reading the page properly
    - moving images such as flash presentations will not be viewable by users without the plugin and may be version dependent – use them to add interest but also make the information available in an accessible form (see next section on non-xhtml/html content).
- Don't provide words within images - if you are providing text, do so in xhtml/html. The words within images are not indexed in search engines and are not readable to those who cannot see the image, unless you make provide the words at alt text or a long description. Words within images are difficult to magnify so are awkward for those who use screen magnifiers.

7a Non-xhtml/html content: Multimedia

PDFs, Flash, PowerPoint and video and audio players may all reduce or block access to the information for some, so you need to be aware of what you need to do to make the information accessible. A first step is to always make sure there is a link to update or acquire a media player. WebAim has a series of articles about how to add accessibility to multimedia – see http://www.webaim.org/articles/.

In the last 10 years Adobe has worked very hard to make Flash and Acrobat produce accessible content, but to do so you do need to actively engage with the process when producing the content files.

If you are producing **Flash** (or are commissioning it), the pages at http://www.adobe.com/accessibility/products/flash/ and http://www.rnib.org.uk/professionals/webaccessibility/designbuild/multimedia/Pages/flash.aspx give good guidelines on how to proceed. If the Flash presentation provides meaningful content (for instance for teaching and learning purposes) then that content needs to be made available from a link next to the Flash presentation.

Most people put **PDFs** on the web. Wherever possible the information should also be provided in an alternative form (xhtml/html, Word or text)

- PDFs are often produced from a scanned document, which produces images of the pages so cannot be used by blind or partially sighted people. Acrobat Pro can perform Optical Character Recognition (OCR) to produce text from the image and embed that in the PDF file.
- At a very basic level most disabled users can access from a PDF a converted text document but the content can often be chaotic – there is often no structure so headings and text are indistinguishable; forms can be particularly difficult to use. Use Acrobat Pro with an existing PDF or during production of a PDF to add accessibility features for forms and buttons and to add structure to text.

There are a number of resources at http://www.adobe.com/accessibility/products/acrobat/training.html that will help you produce more accessible PDFs.

PowerPoint (and other presentation software): Slide presentations can be problematic for screen readers but much of the problems can be resolved by properly structuring your slides with headings, text and lists. Using a provided template will help a lot. The only sure-fire way to ensure that people using screen readers can access the information is to make available an xhtml/html version (converted from the PowerPoint file) or use another method for creating the slides. If slides are converted to PDFs or Flash (as with some slide upload services like Slideshare), the access problems associated with those formats apply. Providing a transcript of the talk or a very thorough handout may be a better alternative than converting slides. Useful help at http://www.rnib.org.uk/professionals/accessibleinformation/electronicdocuments/fileformats/Pages/powerpoint.aspx and http://www.webaim.org/techniques/powerpoint/.

Video and audio

Providing audio and video content should be thought of as a visual aid, and not the primary means of presenting content. It can enhance web pages for many users but because the media presents content primarily for one sense, it can cause difficulties for the blind, partially sighted, deaf and deaf-blind, for whom the content ought to be available as a text transcript. For a video the transcript may need to be descriptive so that action without dialogue can be understood. Captioning should not be thought of as the same as a transcript – often it can be a précis.

When providing audio and video, there should be controls (for stopping and starting) and links to alternative media players (for those with incompatible browsers).

Webcasts are not accessible for all but will provide benefits for some and alternatives can be added after the event. Any potential users need to be aware of issues they may present before the event.

7b Non-xhtml/html content: ~Scripts and forms

- **xhtml/html forms:** Long forms can be arduous to use when using a speaking browser or accessing via a keyboard. The RNIB gives detailed information about what to look out for (see http://www.rnib.org.uk/professionals/webaccessibility/designbuild/scriptsforms/Pages/forms.aspx). For all forms, making them usable will help with accessibility, and tips for usability of forms may be found at http://www.usability.gov/lessons/form.html

- **Javascript and AJAX:** Javascript can add valuable interactivity to pages, but it can cause problems for those using speaking browsers or magnifiers, or those who are not using a mouse. Further details and techniques are available at http://www.rnib.org.uk/professionals/webaccessibility/designbuild/scriptsforms/Pages/javascript.aspx, http://www.webaim.org/techniques/javascript/ and http://www.webaim.org/techniques/ajax/

  - Don't use Javascript when using xhtml/html would work instead
  - Try not to add text through the Javascript, but if you do ensure it is available via <noscript> tags.
  - Ensure the scripts work if the user is only able to use a keyboard (use the ONKEYPRESS event handler as well as the ONCLICK one)
  - If you are using Javascript to enhance navigation (usually by opening dynamic drop-down menus), make sure the navigation is adequate when the enhancement is not available (the target page contains all the navigation the user needs). For decorative effects the lack of accessibility shouldn't matter.
  - If you are providing essential information on a page that changes on the fly, make sure that information is also available in a static form and is kept up to date. Screen reading software cannot cope with content changing under its feet.

- **Java applets:** The main accessibility problems with java applets are that they need to be available to those who don't use a mouse, so use the ONKEYPRESS event handler as well as the ONCLICK one. Descriptive text alternatives should be included for Java Applets and other objects. These should be inserted between the opening and closing <OBJECT></object> tags in the web page. Then, if the browser does not support the applet or object, the alternative content will be presented in its place.

8 User experience and usability testing

Test for user experience against a number of different graphical browsers (and different versions of the same browser) on different platforms, not forgetting smartphones, and test either with lynx, the most widely used non-graphical browser, or with a speaking browser or screen reader. The first preference should be to make your pages size dynamically. If fixed width is required, use a 'standard' width of, say 800 pixels, set with the stylesheet.

Find out how fit-for-purpose your design/architecture is by watching some users work through a set of tasks. Identify your most important user groups (three or four at most) and recruit five individuals or pairs (useful because they talk to each other, with individuals you'll have to ask them to talk to you). They must be individuals who are not already familiar with the pages. The Computing Service can run the user testing for you and let you have the results – our software records the screen usage and audio records the session – contact helen.sargan@ucs.cam.ac.uk.

9 Old technologies

Don't use old technologies such as frames or image maps – they have been superseded for a purpose and have intrinsic accessibility problems.

Evaluating your pages

In setting up your work environment it is sensible to work mainly through an html editor and a specific (up to date) browser, keeping a range of other browsers for checking. By using a specific browser you can install useful tools into it and make your testing process as streamlined as possible.

To create valid code you need to check it via a validator, such as http://validator.w3.org/. There are several tools that allow you access direct from a browser:

- For **Firefox**, the web developer toolbar adds the largest range of tools – find it at http://chrispederick.com/work/web-developer/
- Again for **Firefox**, **HTML Validator** (http://users.skynet.be/mgueury/mozilla/), adds a little icon to the bottom right of your window, which is a really quick indicator of if your page has problems.
- For **MS Internet Explorer**, web developer tools are built into IE8/9. http://www.visionaustralia.org.au/ais/toolbar/ is a good accessibility toolbar
- For **Safari**, choose **Safari > Preferences** and in the General panel (the first you come to) check the box by 'Show Develop menu in menu bar'. This will give you a whole extra menu of functionality

Assembling a set of test browsers

It is useful to look in your logs (easiest with Google Analytics) and see what browsers and versions of browser your audience are using. For a typical measurement, in May 2011 (May 2010) on http://www.cam.ac.uk the breakdown is:

- Firefox 26.96% (29.62%)
- Internet Explorer 8 24.95% (23.23%)
- Safari 15.84% (versions too difficult to untangle) (12.04%)
- Chrome 15.64% (8.43%)
- Internet Explorer 7 9.20% (18.43%)
- Internet Explorer 9 2.70%
- Internet Explorer 6 2.37% (6.31%)

So for pages on this server the test browsers ought to be Firefox (preferably on more than one platform, but the differences are slight), MSIE 8, Safari and Chrome plus making sure MSIE 7 gives usable results. That's probably a good representation of what browser testing ought to be done.

Assembling a set of tools

If you choose to use Firefox as your authoring browser, there are a large number of tools that come with a Firefox extension – many of these are also available via a website of their own and/or for download as a application, and most will also let you upload files or paste in sections of code:

The first five are all accessibility checkers – you should look at all of them and choose which you think suits you best

- **TotalValidator** can be used on the web, installed as an application or as a Firefox extension (see http://www.totalvalidator.com/). It will validate xhtml.html, against different levels of the W3C Web Accessibility Guidelines (1 and 2), check for broken links and give you snapshots of your pages in a range of different browsers and platforms. By default it will do a minimal validation for code and accessibility - to select the options you want to check against, you need to go to follow 'Select validation options' - see http://www.totalvalidator.com/validator/ValidatorForm. The xhtml/html validation is terrifying thorough – choose to believe the w3c validator rather than this.
- **WAVE** is useful for looking at the organization of your pages – see http://wave.webaim.org/ - it's available as an application or as a Firefox extension but it is still set up for checking against version 1 of the Web Accessibility Guidelines.

- **EvalAccess** (http://sipt07.si.ehu.es/evalaccess2/) is useful as it allows you to assess up to 15 pages at a time via the web – it is still set up for checking against version 1 of the Web Accessibility Guidelines.
- **Web accessibility checker** (http://achecker.ca/checker/index.php) gives very well formatted results and allows concurrent xhtml/html validation. Checks against different levels of the W3C Web Accessibility Guidelines (1 and 2).
- **SortSite** is a very good testing tool (http://www.powermapper.com/products/sortsite/index.htm) – it is commercial but cheap (about £60) and only for Windows but can be used for a 30-day trial period or an online test. It will test all pages it finds on the site.

- **Firefox WCAG Contrast checker** (https://addons.mozilla.org/en-US/firefox/addon/7391) extension, which reports on all the colour usages on a page
- For checking CSS there are tools in the **Firefox web developers toolbar**, but for working with inherited styles and working with Javascript the **Firebug** extension is the best choice (see https://addons.mozilla.org/en-US/firefox/addon/1843). This is also built in to Opera (v10/11).

Checklist

The following checklist for A and AA compliance is derived from that produced by WebAIM (see http://www.webaim.org/standards/wcag/checklist for complete version, including level AAA issues). It is intended as a straightforward way to understand what is necessary and is not for verifying conformance. The success criteria point to http://www.w3.org/TR/WCAG20/.

Perceivable

Web content is made available to the senses - sight, hearing, and/or touch

Guideline 1.1
Text Alternatives: Provide text alternatives for any non-text content

Success Criteria	WebAIM's Recommendations
1.1.1 Non-text Content (Level A)	• All images, form image buttons, and image map hot spots have appropriate, equivalent alternative text. • Images that do not convey content, are decorative, or with content that is already conveyed in text are given null alt text (alt="") or implemented as CSS backgrounds. All linked images have descriptive alternative text. • Equivalent alternatives to complex images are provided in context or on a separate (linked and/or referenced via longdesc) page. • Form buttons have a descriptive value. • Form inputs have associated text labels or, if labels cannot be used, a descriptive title attribute. • Embedded multimedia is identified via accessible text. • Frames are appropriately titled.

Guideline 1.2
Time-based Media: Provide alternatives for time-based media

NOTE: If the audio or video is designated as an alternative to web content (e.g., an audio or sign language version of a web page, for example), then the web content itself serves as the alternative.

Success Criteria	WebAIM's Recommendations
1.2.1 Prerecorded Audio-only and Video-only (Level A)	• A descriptive text transcript (including all relevant visual and auditory clues and indicators) is provided for non-live, web-based audio (audio podcasts, MP3 files, etc.). • A text or audio description is provided for non-live, web-based video-only (e.g., video that has no audio track).
1.2.2 Captions (Prerecorded) (Level A)	• Synchronized captions are provided for non-live, web-based video (YouTube videos, etc.)
1.2.3 Audio Description or Media Alternative (Prerecorded) (Level A)	• A descriptive text transcript OR audio description audio track is provided for non-live, web-based video
1.2.4 Captions (Live) (Level AA)	• Synchronized captions are provided for all live multimedia that contains audio (audio-only broadcasts, web casts, video conferences, Flash animations, etc.)
1.2.5 Audio Description (Prerecorded) (Level AA)	• Audio descriptions are provided for all video content NOTE: Only required if the video conveys content visually that is not available in the default audio track.

Guideline 1.3

Adaptable: Create content that can be presented in different ways (for example simpler layout) without losing information or structure

Success Criteria	WebAIM's Recommendations
1.3.1 Info and Relationships (Level A)	• Semantic markup is used to designate headings (<h1>), lists (<ul>, <ol>, and <dl>), emphasized or special text (<strong>, <code>, <abbr>, <blockquote>, for example), etc. Semantic markup is used appropriately. • Tables are used for tabular data. Where necessary, data cells are associated with their headers. Data table captions and summaries are used where appropriate. • Text labels are associated with form input elements. Related form elements are grouped with fieldset/legend.
1.3.2 Meaningful Sequence (Level A)	• The reading and navigation order (determined by code order) is logical and intuitive.
1.3.3 Sensory Characteristics (Level A)	• Instructions do not rely upon shape, size, or visual location (e.g., "Click the square icon to continue" or "Instructions are in the right-hand column"). • Instructions do not rely upon sound (e.g., "A beeping sound indicates you may continue.").

Guideline 1.4

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

Success Criteria	WebAIM's Recommendations
1.4.1 Use of Color (Level A)	• Color is not used as the sole method of conveying content or distinguishing visual elements. • Color alone is not used to distinguish links from surrounding text unless the luminance contrast between the link and the surrounding text is at least 3:1 and an additional differentiation (e.g., it becomes underlined) is provided when the link is hovered over or receives focus.
1.4.2 Audio Control (Level A)	• A mechanism is provided to stop, pause, mute, or adjust volume for audio that automatically plays on a page for more than 3 seconds.
1.4.3 Contrast (Minimum) (Level AA)	• Text and images of text have a contrast ratio of at least 4.5:1. • Large text (over 18 point or 14 point bold) has a contrast ratio of at least 3:1
1.4.4 Resize text (Level AA)	• The page is readable and functional when the text size is doubled.
1.4.5 Images of Text (Level AA)	• If the same visual presentation can be made using text alone, an image is not used to present that text.

Operable

Interface forms, controls, and navigation are operable

Guideline 2.1
Keyboard Accessible: Make all functionality available from a keyboard

Success Criteria	WebAIM's Recommendations
2.1.1 Keyboard (Level A)	• All page functionality is available using the keyboard, unless the functionality cannot be accomplished in any known way using a keyboard (e.g., free hand drawing). • Page-specified shortcut keys and accesskeys (accesskey should typically be avoided) do not conflict with existing browser and screen reader shortcuts.
2.1.2 No Keyboard Trap (Level A)	• Keyboard focus is never locked or trapped at one particular page element. The user can navigate to and from all navigable page elements using only a keyboard.

Guideline 2.2
Enough Time: Provide users enough time to read and use content

Success Criteria	WebAIM's Recommendations
2.2.1 Timing Adjustable (Level A)	• If a page or application has a time limit, the user is given options to turn off, adjust, or extend that time limit. This is not a requirement for real-time events (e.g., an auction), where the time limit is absolutely required, or if the time limit is longer than 20 hours.
2.2.2 Pause, Stop, Hide (Level A)	• Automatically moving, blinking, or scrolling content that lasts longer than 3 seconds can be paused, stopped, or hidden by the user. Moving, blinking, or scrolling can be used to draw attention to or highlight content as long as it lasts less than 3 seconds. • Automatically updating content (e.g., automatically redirecting or refreshing a page, a news ticker, AJAX updated field, a notification alert, etc.) can be paused, stopped, or hidden by the user or the user can manually control the timing of the updates.

Guideline 2.3
Seizures: Do not design content in a way that is known to cause seizures

Success Criteria	WebAIM's Recommendations
2.3.1 Three Flashes or Below Threshold (Level A)	• No page content flashes more than 3 times per second unless that flashing content is sufficiently small and the flashes are of low contrast and do not contain too much red. (See general flash and red flash thresholds)

Guideline 2.4
Navigable: Provide ways to help users navigate, find content, and determine where they are

Success Criteria	WebAIM's Recommendations
2.4.1 Bypass Blocks (Level A)	• A link is provided to skip navigation and other page elements that are repeated across web pages. • If a page has a proper heading structure, this may be considered a sufficient technique instead of a "Skip to main content" link. Note that navigating by headings is not yet supported in all browsers. • If a page uses frames and the frames are appropriately titled, this is a sufficient

	technique for bypassing individual frames.
2.4.2 Page Titled (Level A)	• The web page has a descriptive and informative page title.
2.4.3 Focus Order (Level A)	• The navigation order of links, form elements, etc. is logical and intuitive.
2.4.4 Link Purpose (In Context) (Level A)	• The purpose of each link (or form image button or image map hotspot) can be determined from the link text alone, or from the link text and it's context (e.g., surrounding paragraph, list item, table cell, or table headers). • Links (or form image buttons) with the same text that go to different locations are readily distinguishable.
2.4.5 Multiple Ways (Level AA)	• Multiple ways are available to find other web pages on the site - at least two of: a list of related pages, table of contents, site map, site search, or list of all available web pages.
2.4.6 Headings and Labels (Level AA)	• Page headings and labels for form and interactive controls are informative. Avoid duplicating heading (e.g., "More Details") or label text (e.g., "First Name") unless the structure provides adequate differentiation between them.
2.4.7 Focus Visible (Level AA)	• It is visually apparent which page element has the current keyboard focus (i.e., as you tab through the page, you can see where you are).

Understandable

Content and interface are understandable

Guideline 3.1
Readable: Make text content readable and understandable

Success Criteria	WebAIM's Recommendations
3.1.1 Language of Page (Level A)	• The language of the page is identified using the HTML lang attribute (<html lang="en">, for example).
3.1.2 Language of Parts (Level AA)	• When appropriate, the language of sections of content that are a different language are identified, for example, by using the lang attribute (<blockquote lang="es")>

Guideline 3.2
Predictable: Make Web pages appear and operate in predictable ways

Success Criteria	WebAIM's Recommendations
3.2.1 On Focus (Level A)	• When a page element receives focus, it does not result in a substantial change to the page, the spawning of a pop-up window, an additional change of keyboard focus, or any other change that could confuse or disorient the user.
3.2.2 On Input (Level A)	• When a user inputs information or interacts with a control, it does not result in a substantial change to the page, the spawning of a pop-up window, an additional change of keyboard focus, or any other change that could confuse or disorient the user unless the user is informed of the change ahead of time.

3.2.3 Consistent Navigation (Level AA)	• Navigation links that are repeated on web pages do not change order when navigating through the site.
3.2.4 Consistent Identification (Level AA)	• Elements that have the same functionality across multiple web pages are consistently identified. For example, a search box at the top of the site should always be labeled the same way.

Guideline 3.3
Input Assistance: Help users avoid and correct mistakes

Success Criteria	WebAIM's Recommendations
3.3.1 Error Identification (Level A)	• Required form elements or form elements that require a specific format, value, or length provide this information within the element's label (or if a label is not provided, within the element's title attribute). • If utilized, form validation cues and errors (client-side or server-side) alert users to errors in an efficient, intuitive, and accessible manner. The error is clearly identified, quick access to the problematic element is provided, and user is allowed to easily fix the error and resubmit the form.
3.3.2 Labels or Instructions (Level A)	• Sufficient labels, cues, and instructions for required interactive elements are provided via instructions, examples, properly positioned form labels, and/or fieldsets/legends.
3.3.3 Error Suggestion (Level AA)	• If an input error is detected (via client-side or server-side validation), provide suggestions for fixing the input in a timely and accessible manner.
3.3.4 Error Prevention (Legal, Financial, Data) (Level AA)	• If the user can change or delete legal, financial, or test data, the changes/deletions are reversible, verified, or confirmed.

Robust

Content can be used reliably by a wide variety of user agents, including assistive technologies

Guideline 4.1
Compatible: Maximize compatibility with current and future user agents, including assistive technologies

Success Criteria	WebAIM's Recommendations
4.1.1 Parsing (Level A)	• Significant HTML/XHTML validation/parsing errors are avoided. Check at http://validator.w3.org/
4.1.2 Name, Role, Value (Level A)	• Markup is used in a way that facilitates accessibility. This includes following the HTML/XHTML specifications and using forms, form labels, frame titles, etc. appropriately.