Accessibility: WCAG2 at a Glance

- Search
- Accessibility
- I18N
- Typography

Perceivable

- Provide text alternatives for non-text content.
- Provide captions and alternatives for audio and video content.
- Make content adaptable; and make it **available** to assistive technologies.
- Use sufficient contrast to make things easy to see and hear.

Operable

- Make all functionality keyboard accessible.
- Give users enough time to read and use content.
- Do not use content that causes seizures.
- Help users navigate and find content.

Understandable

- Make text readable and understandable.
- Make content appear and operate in predictable ways.
- Help users avoid and correct mistakes.

Robust

- Maximize compatibility with current and future technologies.

See WCAG Overview.

Internationalization Quicktips

Use Unicode wherever possible for content, databases, etc. Always declare the encoding of content.

The character encoding you choose determines how bytes are mapped to characters in your text.

Normally character encodings limit you to a particular script or set of languages. Unicode allows you to deal simply with almost all scripts and languages in use around the world. In this way Unicode simplifies the handling of content in multiple languages, whether within a single page or across one or more sites. Unicode is particularly useful when used in forms, scripts and

databases, where you often need to support multiple languages. Unicode also makes it very straightforward to add new languages to your content.

Unless you appropriately declare which character encoding you are using your users may be unable to read your content. This is because incorrect assumptions may be made by the application interpreting your text about how the bytes map to characters.

Give me more background

> Character encodings for beginners explains some of the basic concepts about character encodings, and why you should care. Introducing Character sets and Encodings gives an gentle introduction to various aspects of the topic.

So, how do I do this?

> HTML & CSS authors • Spec developers• Server setup

Use characters rather than escapes (e.g. á á or á) whenever you can.

Escapes such as Numeric Character References (NCRs), and entities are ways of representing any Unicode character in markup using only ASCII characters. For example, you can represent the character á in X/HTML as á or á or á.

Such escapes are useful for clearly representing ambiguous or invisible characters, and to prevent problems with syntax characters such as ampersands and angle brackets. They may also be useful on occasion to represent characters not supported by your character encoding or unavailable from your keyboard. Otherwise you should always use characters rather than escapes.

Give me more background

> Using character entities and NCRs provides additional information about the use of escapes in markup languages. In particular, note that entities (such as á) should be used with caution.

So, how do I do this?

> HTML & CSS authors • Spec developers • SVG authors

Declare the language of documents and indicate internal language changes.

Information about the (human) language of content is already important for accessibility, styling, searching, editing, and other reasons. As more and more content is tagged and tagged correctly, applications that can detect language information will become more and more useful and pervasive.

When declaring language, you may need to express information about a specific range of content in a different way from metadata about the document as a whole. It is important to understand this distinction.

Give me more background

Language on the Web gives an gentle introduction to various aspects of the topic.

So, how do I do this?

HTML & CSS authors • SVG authors • XML authors • Schema developers • Server setup

Use style sheets for presentational information. Restrict markup to semantics.

It is an important principle of Web design to keep the way content is styled or presented separate from the actual text itself. This makes it simple to apply alternative styling for the same text, for example in order to display the same content on both a conventional browser and a small hand-held device.

This principle is particularly useful for localization, since different scripts have different typographic needs. For example, due to the complexity of Japanese characters, it may be preferable to show emphasis in Japanese X/HTML pages in other ways than bolding or italicisation. It is much easier to apply such changes if the presentation is described using CSS, and markup is much cleaner and more manageable if text is correctly and unambiguously labelled as 'emphasised' rather than just 'bold'.

It can save considerable time and effort during localization to work with CSS files rather than have to change the markup, because any needed changes can be made in a single location for all pages, and the translator can focus on the content rather than the presentation.

Give me more background

Read the talks slides from the 2007 @media conference presentation "Designing for International Users: Practical Tips".

Check for translatability and inappropriate cultural bias in images, animations & examples.

If you want your content to really communicate with people, you need to speak their language, not only through the text, but also through local imagery, color, objects and preoccupations. It is easy to overlook the culture-specific nature of symbolism, behaviour, concepts, body language, humor, etc. You should get feedback on the suitability and relevance of your images, video-clips, and examples from in-country users.

You should also take care when incorporating text in graphics when content is translated. Text on complex backgrounds or in restricted spaces can cause considerable trouble for the translator. You should provide graphics to the localization group that have text on a separate layer, and you should bear in mind that text in languages such as English and Chinese will almost certainly expand in translation.

Give me more background

Read the talks slides from the 2007 @media conference presentation "Designing for International Users: Practical Tips".

Use an appropriate encoding on both form and server. Support local formats of names/addresses, times/dates, etc.

The encoding used for an HTML page that contains a form should support all the characters needed to enter data into that form. This is particularly important if users are likely to enter information in multiple languages.

Databases and scripts that receive data from forms on pages in multiple languages must also be able to support the characters for all those languages simultaneously.

The simplest way to enable this is to use Unicode for both pages containing forms and all back-end processing and storage. In such a scenario the user can fill in data in whatever language and script they need to.

You should also try to avoid making assumptions that things such as the user's name and address will follow the same formatting rules as your own. Ask yourself how much detail you really need to break out into separate fields for things such as addresses. Bear in mind that in some cultures there are no street names, in others the house number follows the street name, some people need more than one line for the part of the address that precedes the town or city name, etc. In fact in some places an address runs top down from the general to the specific, which implies a very different layout strategy. Be very careful about building into validation routines incorrect assumptions about area codes or telephone number lengths. Recognize that careful labelling is required for how to enter numeric dates, since there are different conventions for ordering of day, month and year.

If you are gathering information from people in more than one country, it is important to develop a strategy for addressing the different formats people will expect to be able to use. Not only is this important for the design of the forms you create, but it also has an impact on how you will store such information in databases.

So, how do I do this?

HTML & CSS authors • HTML & CSS authors • Spec developers

Use simple, concise text. Use care when composing sentences from multiple strings.

Simple, concise text is easier to translate. It is also easier for people to read if the text they are reading is not in their first language.

You should take considerable care when composing messages from multiple substrings, or when inserting variable text into strings. For example, suppose your site uses JSP scripting, and you decide to compose certain messages on the fly. You may create messages by concatenating separate substrings, such as 'Only' or 'Don't', ' return results in ', and 'any format' or 'HTML'. Because the order of text in sentences of other languages can be very different, translating this may present major difficulties.

Similarly, it is important to avoid fixing the positions of variables in text such as "Page 1 of 10". The syntax of other languages may require the numbers to be reversed to make sense. If you use PHP, this would mean using a formatting string such as "Page %1\$d of %2\$d.", rather than the more simple "Page %d of %d.". The latter is untranslatable in some languages.

So, how do I do this?

HTML & CSS authors

On each page include clearly visible navigation to localized pages or sites, using the target language.

Where you have versions of a page or site in a different language, or for a different country or region, you should provide a way for the user to view the version they prefer. This should be available from any page on your site where an alternative exists.

When providing links to pages in other languages, use the name of the target language *in the native language and script*. Don't assume that the user can read English. For example, in a link to a French page, 'French' would be written 'français'. This also applies if you are guiding the user to a country- or region-specific page or site, eg. 'Germany' would be 'Deutschland'.

So, how do I do this?

HTML & CSS authors

For XHTML, add `dir="rtl"` to the `html` tag for right-to-left text. Only re-use it to change the base direction.

Text in languages such as Arabic, Hebrew, Persian and Urdu is read from right to left. This reading order typically leads to right-aligned text and mirror-imaging of things like page and table layout. You can set the default alignment and ordering of page content to right to left by simply including dir="rtl" in the html tag.

The direction set in the html tag sets a base direction for the document which cascades down through all the elements on the page. It is not necessary to repeat the attribute on lower level elements unless you want to explicitly change the directional flow.

Embedded text in, for example, Latin script still runs left to right within the overall right to left flow. So do numbers. If you are working with right to left languages, you should become familiar with the basics of the Unicode bidirectional algorithm. This algorithm takes care of much of this bidirectional text without the need for intervention from the author. There are some circumstances, however, where markup or Unicode control characters are needed to ensure the correct effect.

Give me more background

Creating (X)HTML Pages in Arabic & Hebrew provides a gentle introduction to the basics of handling right-to-left text in HTML. The principles are similar for other markup languages.

What you need to know about the bidi algorithm and inline markup provides a gentle introduction to the basics of handling inline bidirectional text.

So, how do I do this?

HTML & CSS authors • SVG authors • XML authors • Schema developers

Validate! Use techniques, tutorials, and articles at
http://www.w3.org/International/

English Typography

Use the proper English characters instead of their misused equivalents.

Quotes

“ (`&#8220;`) opening quote (instead of ")

” (`&#8221;`) closing quote (instead of ")

Apostrophe

’ (`&#8217;`) apostrophe (instead of ')

Dashes and Hyphens

– (`&#8211;` or `&ndash;`) en dash, used for ranges, e.g. “13–15 November” (instead of -)

— (`&#8212;` or `&mdash;`) em dash, used for change of thought, e.g. “Star Wars is—as everyone knows—amazing.” (instead of -, or --)

Ellipsis

… (`&#8230;` or `&hellip;`) horizontal ellipsis, used to indicate an omission or a pause (instead of ...)

W3C Cheatsheet

About the W3C Cheat Sheet