**15 rules for making accessible links**

Links are the soul of the web — they are what makes it special.

From ancient times printed texts have relied on far clunkier devices — things like footnotes, appendices and [marginalia](http://en.wikipedia.org/wiki/Marginalia) — to add related content and extra context to the body text.

But, if the web has had one great trick, it is its ability to stitch together disparate content in a non-linear, but still meaningful way.

That is why there is a huge onus on us — the humble web developer — to make sure all those links work for **users of all abilities**!

This is, unfortunately, a little more complicated than just avoiding the use of phrases like “click here”.

Let’s look at what it takes.

So, what makes good link text?

While the current W3C’s Web Content Accessibility Guidelines – WCAG2 – emphasise providing overall context for a link, little emphasis is placed on making the text of a link itself understandable to users with disabilities.

I believe this is the wrong approach.

Screen reading applications offer only limited ways to interpret a page. One common method is to generate a list of links (without context) to determine the content of the page. Screen reader users also often scan a page by simply tabbing from link to link (without reading the text in-between).

When confronted with a bunch of “Click here to download the annual report” and “More on boating”, such techniques are useless.

Link text also becomes a serious issue once you start talking about mobile and tablet sites.

Two of the most well-known sets of mobile accessibility guidelines — the [W3C Mobile Best Practices](http://www.w3.org/TR/2008/REC-mobile-bp-20080729/) and the [BBC’s Mobile Accessibility Guidelines](http://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile_access.shtml) — suggest clearly identifying and describing the target of the link.

In my work auditing web sites, I identify all the “click here” links and flag them as accessibility failures. Sure, “click here” links can be activated by the keyboard, but it implies that you need a mouse to activate the link.

And that’s a failing.

Sure, it’s a little more difficult to argue that “here” and “more” links are inaccessible *strictly* under WCAG2. I still flag them as errors, but it’s almost impossible to find a success criterion that they fail.

So, here is my 15 point checklist to keep in mind when linking text on the web.

Rule 1 : Don’t use the word “link” in your links

This is an easy one.

Screen readers tell the user when they encounter a link, so you don’t need to use the words “*link*” or “*links to*” or “*goes to*” in your link text.

Rule 2: Don’t capitalize links



Photo credit: ~dgies

There are two problems with capitalized text in links.

Firstly, some screen readers read capitalized text letter-by-letter. And this occasionally even occurs when the HTML is in sentence-case and the CSS forces the capitalization.

The second problem is that capital letters are harder to read (for everyone, but especially people with reading disabilities).

Capitalized text has no difference in shape: all the words are just one big rectangle. Sentence capitals have differentiated shapes based on the letters used.

See the following as an example:

*CAPITALIZED TEXT IS DIFFICULT TO READ. IT’S ALL ONE BIG RECTANGLE.*

*Capitalized text is difficult to read. It’s all one big rectangle.*

The general rule when it comes to reading is: if it’s more difficult to the general public then it will be almost impossible for a person with a cognitive disability to read.

And who wants to shout at their audience?

Rule 3: Avoid ASCII characters



Photo credit: jessicahtam

Under [Success Criterion 1.1.1](http://www.w3.org/TR/2013/NOTE-UNDERSTANDING-WCAG20-20130905/text-equiv-all.html): Non-text content is the requirement to provide text alternatives for ASCII characters.

The specific requirement is [H86: Providing text alternatives for ASCII art, emoticons, and leetspeak](http://www.w3.org/TR/2013/NOTE-WCAG20-TECHS-20130905/H86), which means you need to provide text alternatives for things like smiley faces.

They recommend using the TITLE attribute; even though it’s not well supported, or the ABBR or ACRONYM elements.

However if you absolutely *must* use emoticons, then at least mark them up with some ARIA:

<span class="sr-only">Smiley face</span>

<span aria-hidden="true">:-)</span>

However I’ve always believed this technique should go one step further, and include most punctuation as well.

Let’s look at em-dashes as an example.

The following link text may be clear to visual users, but it’s a different story for screen reader users:

[*16 – 17 years*](http://www.sitepoint.com/15-rules-making-accessible-links/)*,*[*17 – 21 years*](http://www.sitepoint.com/15-rules-making-accessible-links/)

Some screen readers will read the above link as “*link one six one seven years end link*“.

It is much clearer to replace the em-dash with the word “to”. In that case, screen readers will read it as “link sixteen to seventeen years end link”.

Much clearer, right?

Rule 4: Avoid using URLs as link text

When we see “www.londontoylibrary.co.uk”, we see the words ‘london’, ‘toy’ and ‘library’, but a screen reader is going to read the URL letter-by-letter. “*Double-U, Double-U, Double-U, Dot, El-Oh-En-Dee..*”

As you can imagine, this becomes unintelligible after the first 4-5 letters.

So, always use meaningful text as links, such as “London Toy Library”. If you really need to provide the URL in text — for example if you expect people to print out the content — then append the URL to the link text in the print style sheet with something like:

@media print{

 a:after{content:" (" attr(href) ") ";font-size:0.8em;font-weight:normal;}

}

Rule 5: Keep link text concise

It’s a really good idea to restrict the length of your link text to a maximum of 100 characters.

Screen readers have a huge array of functions that allow users to skip to the next word, next sentence, next paragraph or next heading, but they have to read the entirety of a link.

So, you can imagine that if you have turned an entire paragraph into a single link (which we’re seeing more of with the advent of block-level links in HTML5), the entire link is read out by the screen reader.

You can imagine how annoying this could become.

Rule 6: Restrict the number of text links on a page

This is important because users see links as a form of navigation: they know they are not on the right page so they are looking for links that will take them to where they want to go.

If there are a lot of links on a page, it makes it that much harder to navigate a site.

And of course, screen reader users can pull out all the links in a page, so if there are hundreds of links then reading through them all is a nightmare.

Ok, so how many links are too many? That’s the ‘How long is a piece of string’ question, and depends on the type of site that you have.

Just bear in mind the users that are navigating from link to link when you’re constructing your pages.

**What about areas where additional link text or contextual information is essential?**

There are three key areas where additional link text or contextual information is a must:

* linking to downloads
* links opening in a new window
* pagination / alphabetized links.

We’ll cover them separately.

Rule 7: Don’t link directly to downloads

In the usability testing I’ve witnessed, one thing is universal: people hate downloads.

It is even more annoying when they open a download without expecting it.

But what is a mere annoyance for general users becomes a very serious problem for people with disabilities. It is essential you properly indicate when a link activates a download – and make sure this information is **in the link text**.

If you use a ‘PDF’ or ‘Word’ icon instead of the text, always make sure they have useful ALT attributes!

One thing I have seen recently, is the automatic addition of a download icon via CSS. Remember that this is essentially invisible to screen readers — they don’t read CSS images — so never embed critical information in CSS.

I’ve always recommended that you add some more detail than just the type of download, for example the size of the download too.

So, that’s all fine if you have one document. But what if you have a whole bunch?

WCAG2 now allows something like the following (with “Disability Services Annual Report” coded as a heading):

***Disability Services Annual Report***[*2013 PDF*](http://www.sitepoint.com/15-rules-making-accessible-links/)*,*[*2013 Word*](http://www.sitepoint.com/15-rules-making-accessible-links/)[*2012 PDF*](http://www.sitepoint.com/15-rules-making-accessible-links/)*,*[*2012 Word*](http://www.sitepoint.com/15-rules-making-accessible-links/)[*2011 PDF*](http://www.sitepoint.com/15-rules-making-accessible-links/)*,*[*2011 Word*](http://www.sitepoint.com/15-rules-making-accessible-links/)

Or even something like this (where the first column and first row are coded table headers using <th>):

| **Title** | **PDF** | **Word** |
| --- | --- | --- |
| 2013 Disability Services Annual Report | PDF download  | Microsoft Word Download  |
| 2012 Disability Services Annual Report | PDF download  | Microsoft Word Download  |
| 2011 Disability Services Annual Report | PDF download  | Microsoft Word Download  |

Or even a combination of the two (headings and table headers):

**Disability Services Annual Report**

| **Title** | **PDF** | **Word** |
| --- | --- | --- |
| 2013 | PDF download  | Microsoft Word Download  |
| 2012 | PDF download  | Microsoft Word Download  |
| 2011 | PDF download  | Microsoft Word Download  |

Much cleaner, more semantic, AND more accessible, I think.

Rule 8: Always alert the user when opening new windows

It is important to warn people with disabilities that a new window has been opened — especially people with cognitive disabilities who may not notice.

The best way to indicate that a link opens in a new window is to add text to the link, such as “(opens in new window)”.

However, you can also add an icon with an appropriate ALT attribute, but you must explain what the icon means somewhere on the page, like [FRIENDS Life](http://www.friendslife.co.uk/common/layouts/subSectionLayout.jhtml?pageId=online/SitePageSimple%3Aaccessibility_checklist) does (although note their accessibility information is woefully out-of-date):



If you absolutely ***must*** use a character to indicate a link, I’ve started using[Font Awesome’s](http://fortawesome.github.io/Font-Awesome/) “[External link](http://fortawesome.github.io/Font-Awesome/icon/external-link/)” icon for opening in new windows.

The HTML is:

<a href="http://www.google.com/" target="\_blank">Google

 <span class="sr-only">Opens in new window</span>

 <i aria-hidden="true" class="fa fa-edit fa-external-link"></i>

</a>

And the CSS is:

.sr-only {

 position: absolute;

 width: 1px;

 height: 1px;

 padding: 0;

 margin: -1px;

 overflow: hidden;

 clip: rect(0, 0, 0, 0);

 border: 0;

}

It renders like this :



Always include the icon *after* the link text. Otherwise screen reader users who pull out the links of each page, will get a whole list starting with “*opens in new window*” instead of meaningful link text.

Rule 9: Be aware of pagination and alphabetized links

For a visual user, a list of numbered links at the bottom of the search results means “move to the next page”, but in my user testing, I find over and over again that people with disabilities do not know what these links mean.

To make these links accessible, just add some contextual information before the list of links, such as “Go to search page: 1 2 .. etc”.

If it’s an alphabetized list of links, then a heading explaining the links is best, such as:

**Author’s surnames starting with:**
[A](http://www.sitepoint.com/15-rules-making-accessible-links/) | [B](http://www.sitepoint.com/15-rules-making-accessible-links/) | [C](http://www.sitepoint.com/15-rules-making-accessible-links/) | [D](http://www.sitepoint.com/15-rules-making-accessible-links/) | [E](http://www.sitepoint.com/15-rules-making-accessible-links/) | [F](http://www.sitepoint.com/15-rules-making-accessible-links/) | [G](http://www.sitepoint.com/15-rules-making-accessible-links/) | [H](http://www.sitepoint.com/15-rules-making-accessible-links/) | [I](http://www.sitepoint.com/15-rules-making-accessible-links/) | [J](http://www.sitepoint.com/15-rules-making-accessible-links/) | [K](http://www.sitepoint.com/15-rules-making-accessible-links/) | [L](http://www.sitepoint.com/15-rules-making-accessible-links/) | [M](http://www.sitepoint.com/15-rules-making-accessible-links/) | [N](http://www.sitepoint.com/15-rules-making-accessible-links/) | [O](http://www.sitepoint.com/15-rules-making-accessible-links/) | [P](http://www.sitepoint.com/15-rules-making-accessible-links/) | [Q](http://www.sitepoint.com/15-rules-making-accessible-links/) | [R](http://www.sitepoint.com/15-rules-making-accessible-links/) | [S](http://www.sitepoint.com/15-rules-making-accessible-links/) | [T](http://www.sitepoint.com/15-rules-making-accessible-links/) | [U](http://www.sitepoint.com/15-rules-making-accessible-links/) | [V](http://www.sitepoint.com/15-rules-making-accessible-links/) | [W](http://www.sitepoint.com/15-rules-making-accessible-links/) | [X](http://www.sitepoint.com/15-rules-making-accessible-links/) | [Y](http://www.sitepoint.com/15-rules-making-accessible-links/) | [Z](http://www.sitepoint.com/15-rules-making-accessible-links/)

User agents and assistive technologies now are adept at rendering adjacent links, so there’s no need to put vertical bars, like in the example above, between links any more.

It is still important that there is space between adjacent links.

Android devices zoom in on an area when the user touch encompasses more than one link, but Apple and Windows devices do not.

I recommend using the CSS border property around links to make them look more like buttons. This makes the entire space within the border clickable, and can be achieved by simply adding:

border: 1px solid black;

I also see color being used in pagination or alphabetized links to indicate the current link, such as:



This violates [Success Criterion 1.4.1 Use of Color:](http://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-without-color.html) “*Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element*“.

The best thing to do is make the current page number plain text and the other numbers links.

And make sure those links are underlined! This way the links are visually distinct (without relying solely on color) and programmatically distinct as well.

The other problem with pagination and alphabetized links is that they generally present a very small target area for the user to select.

This is no problem for keyboard users, however people using other input devices, such as a mouse, joystick or touch screen can find one letter links very difficult to properly select.

It’s a good idea to make single character links a larger text size than the surrounding content.

It’s also wise to use plenty of CSS padding around the link to increase the clickable area, such as:

padding-left: 1em;

padding-right: 1em;

So if you put these techniques together you get:

span.pag a {

 padding: 1em;

 border: 1px solid black;

 margin: 0.5em;

 background: #ffffc3;

 color: black;

}

span.current-page {

 padding: 1em;

 border: 1px solid black;

 margin: 0.5em;

 background: black;

 color: white;

}

And the HTML for the actual paginated links is:

<span class="pag">

 <a href="http://www.one.com">1</a>

 <a href="http://www.two.com">2</a>

 <a href="http://www.three.com">3</a>

 <span class="current-page">4</span>

 <a href="http://www.five.com">5</a>

 <a href="http://www.six.com">6</a>

</span>

And you end up with paginated links that look like (for a working example see [The Accessible Wizard of Oz](http://wizardofoz.accessibilityoz.com.au/accessible.html)):



Of course, feel free to fancy it up a bit with rounded corners and some color!

Rule 10: Be mindful when using anchor links



Photo: El Bibliomata

It’s important to be very careful using anchor links.

With really long pages, I’ve seen many users scan the page and then select anchor links, assuming it will take them to a different page. Often they don’t realize they are just moving up and down the page they have already scanned.

Remember: if something is an issue for the general user — and we know this is — it will be a serious problem for a person with a disability.

Screen readers do usually identify whether a link is an anchor by adding the phrase “in- page link”, so they at least have some feedback when activating an anchor link. But visually-challenged users, people who use magnifiers and people with cognitive disabilities may not realize they are activating an anchor link at all.

I recommend using a standard preceding phrase to indicate anchor links such as “In this page:”, “Jump down to:” or “This page contains the following content:”.

Rule 11: The case for underlining links

People expect links to be underlined.

When they see underlined text they expect it to be a link (which is why you should never underline text in the online world unless you are representing a link).

WCAG2 does recommend that you underline your inline text links, but also allows developers to meet the accessibility criterion if they use a contrast ratio of 3:1 with surrounding text and providing additional visual cues on focus for links or controls where color alone is used to identify them.

This requires that your text links contrast sufficiently with surrounding text (the W3C has a [list of link colors that contrast appropriately with black text and a white background](http://www.w3.org/TR/2013/NOTE-WCAG20-TECHS-20130905/working-examples/G183/link-contrast.html)) and there is an additional visual cue when the link receives mouse or keyboard focus.

This visual cue can be an underline (go on, make those links underlined!), bold, italic or increase in font size or it can be the addition of a glyph or image. It can be implemented through CSS as this only needs to be a visual indicator.

But remember to add a:focus to a:hover!

Rule 12: Design with keyboard-only users in mind

The extra information conveyed in the TITLE attribute of a link HREF, ([see H33: Supplementing link text with the title attribute](http://www.w3.org/TR/2008/WD-WCAG20-TECHS-20081103/H33.html), is not available to keyboard users.

In addition, this tooltip cannot be resized to suit the user, [failing Level AA Success Criterion 1.4.4: Resize text.](http://www.w3.org/TR/2008/REC-WCAG20-20081211/#visual-audio-contrast-scale) Neither can the colors of the text or background color be changed.

Another serious keyboard issue can be a lack of a keyboard focus indicator, which can often make a site almost impossible to use by keyboard users.

The only way for a keyboard user to know which link, navigation item or field has focus is if that item has a keyboard focus indicator. As you can see from the next screenshot, the item with the keyboard focus is a different color to the surrounding content and outlined with a dotted border.



Indicating keyboard focus.

I always recommend that if content changes on mouse hover, the same should occur on keyboard focus. In almost all cases you need to add “a:focus” wherever you have “a:hover” to achieve this.

**Scripted links**

Using scripting events, via JavaScript, to emulate links is also banned under the WCAG2.

All links should be created using A HREF or AREA. The use of ONCLICK on random elements, such as SPAN, IMG and DIV is not permitted.

**Negative TABINDEX values**

Negative TABINDEX values, introduced with HTML5, very clearly violate [Success Criterion 2.1.1: Keyboard: All functionality of the content is operable through a keyboard interface](http://www.w3.org/TR/UNDERSTANDING-WCAG20/keyboard-operation-keyboard-operable.html).

Rule 12: Be mindful when using images as links

There are special requirements for images that are links. The ALT attribute acts as the link text. As mentioned earlier, you don’t need to add the word “link”, and you also don’t need to add the word “graphic” or “image”, as screen readers identify images to their users as well.

You do need to be careful when creating ALT attributes for image links, because the ALT attribute has two requirements: it must describe the image and it must tell the user what activating the link will do.

For an image button then an ALT attribute like “Search”, “Find” or “Submit” is great, but avoid using the word “Go” as I’ve found users don’t understand that terminology (“Go where?” they ask).

If you do have a linked image next to a text link that goes to the same page, then in most cases that image needs an empty ALT attribute.

This is a requirement under [Technique H2: Combining adjacent image and text links for the same resource](http://www.w3.org/TR/2013/NOTE-WCAG20-TECHS-20130905/H2).

It is also a requirement that one link should encompass the link text and the image. Happily, this can be done now as HTML5 allows A as a block-level element.

There is one exception to this rule, and that is when the image conveys *additional important information*not provided in the text link.

Here’s what **NOT** to do with linked images



The code for this image/link combination is so woeful, I just had to include it here:

<div class="list-preview-item-wide">

<a onclick="(new Image()).src='/rg/list-user-wide/list-image/images/b.gif?link=%2Flist%2Fgm7xsjP4vD8%2F';" href="/list/gm7xsjP4vD8/"><img alt="image of title" title="image of title" src="http://ia.media-imdb.com/images/M/MV5BMTIxMjc4NTA2Nl5BMl5BanBnXkFtZTYwNTU2MzU5.\_V1.\_SX86\_CR0,0,86,86\_.jpg" class="loadlate" width="86" height="86"/>

<noscript><img height="86"

 width="86"

 alt="image of title" title="image of title"

 src="http://ia.media-imdb.com/images/M/MV5BMTIxMjc4NTA2Nl5BMl5BanBnXkFtZTYwNTU2MzU5.\_V1.\_SX86\_CR0,0,86,86\_.jpg" class="" /></noscript>

</a></div>

 <div class="list\_name"><b><a onclick="(new Image()).src='/rg/list-user-wide/list-title/images/b.gif?link=%2Flist%2Fgm7xsjP4vD8%2F';" href="/list/gm7xsjP4vD8/">Top TV Series of the Past 20 Years</a></b></div>

This code contains a great swath of meaningless inline JavaScript that could easily be replaced with HTML. It also has a mind-bogglingly useless ALT attribute that reads “image of title”.

Just to be sure, they added “image of title” to the TITLE attribute too!

In the previous screenshot, the image and the link text “Top TV series of the Past 20 years” go to the same place, so they should all be one link.

However the image of the Sopranos TV series tells us some important information: that the Sopranos is in the top 20. Therefore the image needs to have an ALT attribute like “The Sopranos”.

The correct code would be (and I just had to get rid of that JavaScript too):

<a href="/list/gm7xsjP4vD8/">

<span class="list-preview-item-wide">

<img alt="The Sopranos" src="http://ia.media-imdb.com/images/M/MV5BMTIxMjc4NTA2Nl5BMl5BanBnXkFtZTYwNTU2MzU5.\_V1.\_SX86\_CR0,0,86,86\_.jpg" class="loadlate" width="86" height="86"/>

</span>

<span class="list\_name"><b>Top TV Series of the Past 20 Years</b></span></a>

Rule 13: Eliminate broken or empty links

Most people use a tool to find the broken links on their site, so I don’t often come across those.

But in almost every site that I test, I find empty links like this:

<a href='someplace.html'></a>



Photo: roeyahram

I can only guess that they are inserted by some Content Management Systems glitch.

Whatever the origin, they are inaccessible and annoying to screen reader users.

All they’ll hear is “*link end link*” and find themselves wondering if they are missing something — or perhaps they’ll assume the link is an image with an empty ALT attribute.

Either way, very irritating.

Rule 14: Make your links consistent

Consistency is your friend. It lets your users orient themselves quickly.

Everything on your site should be consistent: your headings, field labels, buttons etc. You links are no exception to this rule.

This is especially important if you have decided, despite my urging, not to underline your links.

Your users will browse your site and determine how you display your inline link text – so make sure it is consistent! You must never use one color to indicate links on one page and then another color on another page.

Rule 15: Test your color contrast

You need to ensure that the color of your link text contrasts sufficiently with the background color. This is required whether your links are underlined or not.

If you change the color of the link on mouse and keyboard focus, you need to make sure that new color also meets color contrast requirements.

There are a number of online tools that make testing contrast both painless and scientific.

These include:

* [Check My Colors](http://www.checkmycolours.com/) (checkmycolours.com)
* [Luminosity Colour Contrast Ratio Analyser](http://juicystudio.com/services/luminositycontrastratio.php) (juicystudio.com)

You made it!

As you now know if you got this far, there are a range of (usually) simple ways you can make your site easier to access for people with disabilities.

There are also even more ways to make the experience a nightmare.

Choose wisely.

I think that should cover things off — but if you’re looking for even more detail:

1) I have an even more [detailed discussion on accessible links](http://www.accessibilityoz.com.au/2014/02/links-and-accessibility/) on my own blog.

2) Also try these Accessibility fact sheets:

* [Manager Factsheet on links](http://wiki.accessibilityoz.com.au/doku.php/checklists/content/manager#link_text1)
* [Developer Factsheet on links](http://wiki.accessibilityoz.com.au/doku.php/checklists/content/developer#link_text1)