add
accessible digital documents

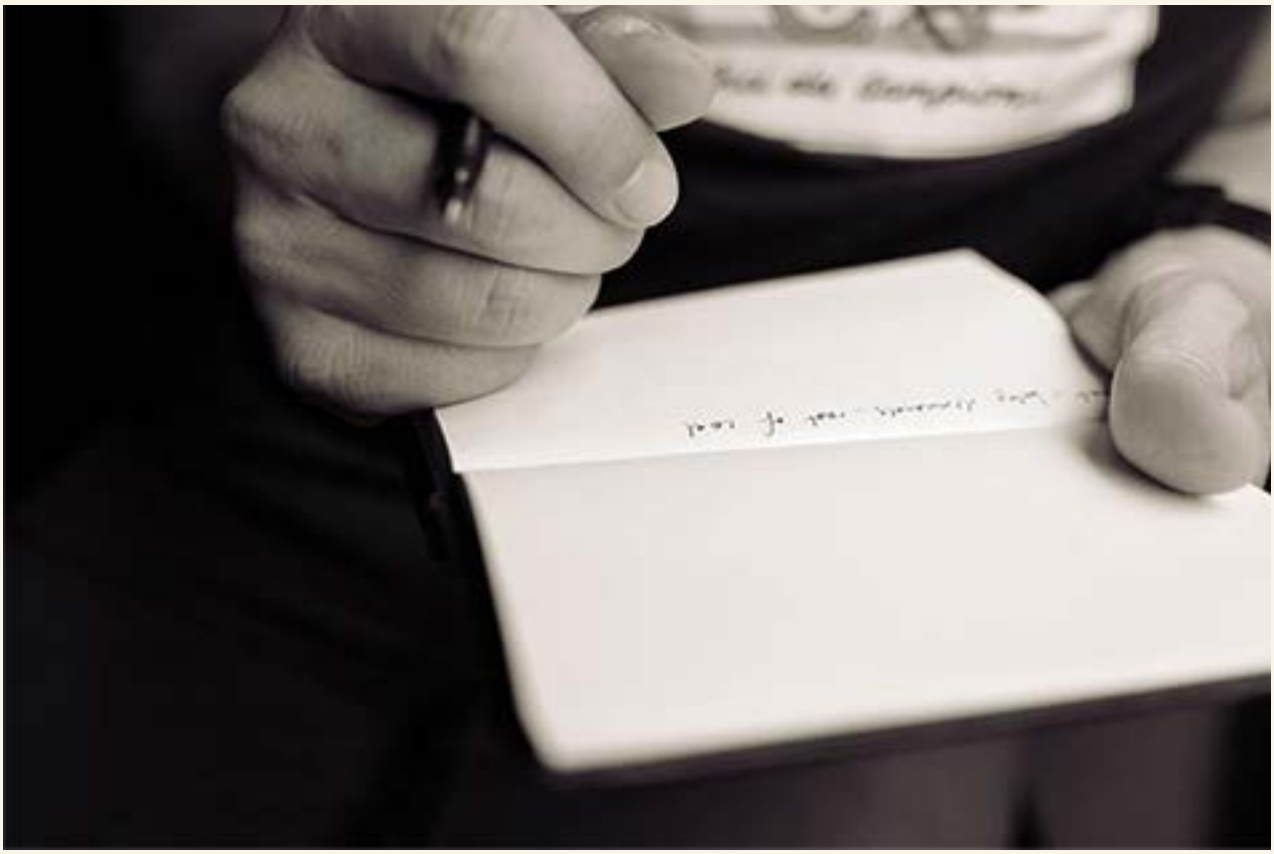
Selected indexed PDF accessibility articles

Contents

Accessible PDF footnotes and endnotes

26th June 2017 | by Ted Page

Footnotes and endnotes in PDFs, whether created in Microsoft Word or Adobe InDesign, are not very accessible out-of-the-box. They can be made more accessible by creating links from each note reference to its respective note and back again. However, as will be seen, this approach is not ideal, for a number of reasons. Fortunately, there is a better way, using some simple JavaScript to embed each note in a pop-up.



Defining the problem

From an accessibility point of view, the most serious problem with footnotes is reading order (so too with endnotes, but the following will focus on footnotes only—endnotes will be addressed later).

The problem is that a footnote can be separated from its reference (for example, a superscript "1") by several paragraphs of text. The assumption, of course, is that the reader can simply glance down a page from a footnote reference to the footnote itself and then back again.

Screen reader and screen magnifier users

However, for many screen reader users, glancing down the page and back will be impossible (obviously so, if you can't see the page at all). It may also be difficult for screen magnifier users, as the distance between reference and note can be considerable. The optimal position in the reading order for footnotes for such readers is likely to be immediately after the footnote reference, often in the middle of a paragraph.

Reflowed documents

By contrast, if you are viewing on a small screen a document that has, for example, been typeset in columns, in order to avoid either tiny font sizes or horizontal scrolling, you are likely to want to reflow the PDF into a single column (**`Cmd/Ctrl + 4`** toggles reflow on and off).

By convention, footnotes in reflowed content should appear in the reading order immediately after the paragraph or other content block from which they are referenced. Fortunately, as will be seen, it is possible to meet both of these reading order requirements simultaneously in a PDF.

User choice

A second problem is that, as footnotes are not part of the primary narrative, being forced to read them can be seriously distracting. Document authors (and reading systems) should give readers the option to read notes or else to skip over them entirely.

The problems of the simple linking approach

In Microsoft Word or InDesign it is possible to create links from references to footnotes and back again. However, it is fair to say that such links work better in some screen readers than others: in some, such links are virtually impossible to use. In addition, once the reader has reached the bottom of the page, the contents of any note located in the footer will be read out by screen readers, whether required or not.

Multiple references to a single footnote

A third problem that can only be solved using the scripting method below is that of multiple references to a single note. In the simple linking approach, footnotes will have to be duplicated in full each time they are referenced, leading to potentially serious redundancy problems (back links from a single footnote to multiple references are unworkable).

The scripting method

Step 1: fixing the reflow reading order

First we need to set the reading order for reflow view users. To do so, in the Acrobat Pro **Order Panel**, move the footnote so that it appears after the paragraph (or other content block) from which it is referenced. This will ensure the footnote appears in the right place in the reading order both in reflow view and for assistive technologies that don't read PDF tags.

Step 2: creating a JavaScript pop-up

Next, we need to create the pop-up. To do so:

- Click the **Link icon** to activate the link tool (alternatively, in Acrobat 11 select **Tools**, **Content Editing**, **Add or Edit Link**, or in Acrobat DC it's (**Tools**, **Edit PDF**, **Link**), **Add or edit links**, **Add/Edit Web or Document Link**)
- Drag a box around the note reference
- From the resulting **Create Link** dialogue box, select **Custom link**
- Click the **Next** button
- In the **Link Properties** dialogue box select **Actions**
- From the **Select Action** dropdown, select **Run a JavaScript**
- Click the **Add** button
- In the **JavaScript Editor** dialogue box, paste or type the following script: `app.alert({cMsg:"", cTitle:"", nIcon:"3"})`

Step 3: adding the content

- The footnote text should be placed inside the double quotes after `cMsg`
- A short description, such as "Footnote 1" or "Endnote 1", should be placed

inside the double quotes after **`cTitle`**—this will then appear in the title bar of the pop-up

- The number "3" in double quotes following **`nIcon`** generates an Information icon in the pop-up (except on Macs which just display an Acrobat icon)

Step 4: creating Link and Link-OBJR tags

When using the **Custom link** method, a **<Link>** tag and a **Link-OBJR** tag will have to be added manually for each link. To do so:

Creating a Link tag

- In the tag tree, right-click the tag containing the footnote reference and select **New Tag**
- In the **Type field** of the **New Tag** dialogue box, type "Link", or select Link from the dropdown
- Nest the tag containing the footnote text and the tag containing the footnote reference inside (as a child of) the newly created **<Link>** tag

Creating a Link-OBJR tag

- Select the **<Link>** tag again, right-click and select **Find**
- From the **Find** dropdown select **Unmarked links**
- Click the **Find** button
- If the correct link is highlighted in the document, click **Tag Element**. A Find completed alert box will appear

The **Link–OBJR** tag will be created and the link will be screen-reader-accessible.

Step 5: adding alt text

- Lastly, add alternate text of, for example, "Footnote 1" to the **<Link>** tag (**right-click** the <Link> tag, **Properties**, **Alternative Text**)

The result

Sighted readers will typically use footnotes in the usual way, although they could, of course, opt to trigger the pop-ups.

Readers who choose to reflow the document, or those using assistive technologies that don't take their reading order from a PDF's tags will find each footnote located immediately after the paragraph or other content block from which it is referenced.

A screen reader user, when encountering a footnote link, will hear something like: "Footnote 1, link". The reader can then choose to ignore the footnote or press enter or return to trigger the pop-up and hear the content, followed by a prompt to press the space bar to close the pop-up. Doing so will take you directly back to the reference, ready to continue reading from where you left off.

Similarly, a screen magnifier user can click the reference to trigger the pop-up, and hence avoid having to navigate to the bottom of the page and back.

Endnotes

Pop-ups for endnotes can be created in the same way as for footnotes, except that in the third bullet point in step 4 above, only the reference need be nested inside the **<Link>** tag.

This is because the endnotes don't have to be hidden in the same way as footnotes do: being located at the end of the document or at the end of each section of a document, the reader should have little difficulty skipping over them if required.

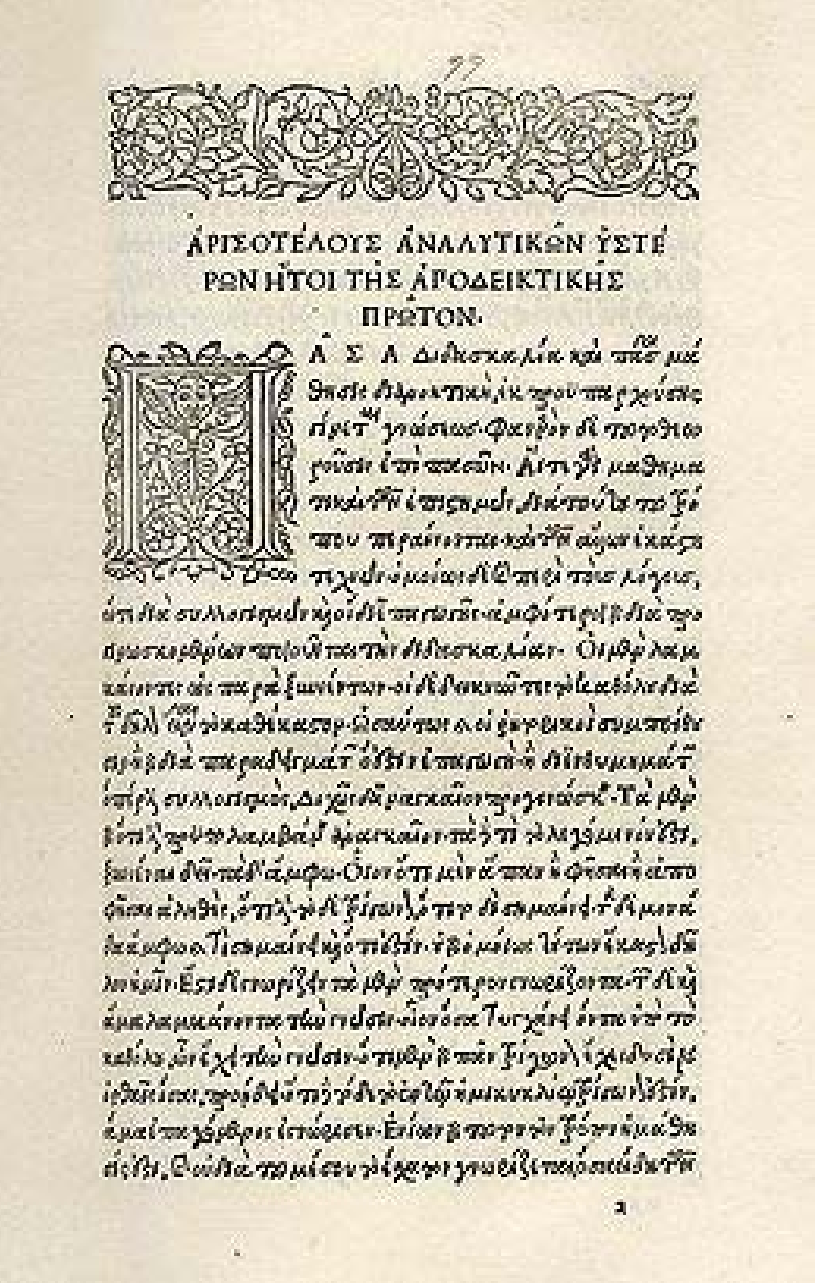
Conclusion

This method makes footnotes and endnotes significantly more accessible than any other method. It solves the reading order problem, makes reading notes entirely optional, and is by far the most efficient way of referencing the same note from multiple locations within a document.

The case against banning italics

7th January 2019 | by Ted Page

Why a blanket ban on italics for accessibility reasons is likely to do more harm than good



Aristotle printed by Aldus Manutius, 1495–98

For accessibility reasons, some organisations impose a blanket ban on the use of italics in online copy. It will be argued here that this is a mistake.

Although blocks of italic text can be difficult to read for many people, banning italics in all contexts can strip text of much of its semantic richness, making it less readable and therefore less accessible. For the time being, the judicious use of italics will yield the greatest overall benefits, but in the longer run, the solution is to develop reading systems that allow full text customisation.

The research to date

To date there has been only limited research into this area. One of the most influential studies has been Rello and Baeza-Yates, *Good Fonts For Dyslexia,* 2013 (PDF, 683KB). (Note: this is an untagged PDF which may be more or less inaccessible to some readers.) This study measured the impact of font type on people with dyslexia. It tested 48 subjects, each of whom was given 12 texts to read with 12 different fonts, 3 of which were italics. One of the study's conclusions was that, for people with dyslexia, "italic fonts decreased reading performance".

Only blocks of text were tested

In the present context, the most important point to note about the study is that the 12 texts used in the test were all 60 words in length. In other words, the study was conducted only on blocks of text. But as will be seen, text that is conventionally rendered in italics often comes in short phrases, or even in single words or numerals.

WCAG/WebAIM on the use of italics

Many respected and trusted accessibility resources advise against the use of *blocks* of italic text. For example, WCAG Understanding Guideline 3.1 includes an advisory technique for "avoiding chunks of italic text".

Similarly WebAIM advises as follows: "Do not use italics or bold on long sections of text", but at the same time "use various stylistic elements (italics, bold, color, brief animation, or differently-styled content) to highlight important content".

International standards

It should also be noted that many conventions for the use of italics are enshrined in international standards, including BS 5605 (citing and referencing published material), BS ISO 690 (bibliographic references and citations), and ISO 999 (content, organization and presentation of indexes).

What you lose when you ban italics

As detailed in a 2008 post in this blog (Italics), italics give meaning to content in many important ways. These include:

- emphasis (as opposed to bold for strong emphasis)
- works (book or document titles, acts of Parliament, legal cases, film, TV and radio programmes, paintings, compositions, ship and aircraft names…)
- foreign words or phrases (*tête-à-tête*, *pièce de résistance*, *faux*), or in biology for Latin binomials (*homo sapiens*)
- stylised text such as "*see*" and "*see also*" index cross-references, or index locators that refer to illustrations, maps or diagrams

No consensus amongst readers with dyslexia

Even amongst people with dyslexia there is no consensus that a blanket ban on italics would be helpful. For example, Martin Pitt, in a (May 2017) GitHub forum discussion stated:

> "Monotonous walls of text are no good for anybody. Rich text is fantastic … [it adds] legibility to paragraphs of copy by sprinkling different font weights (e.g. bold), italics and colour. … Thus practices, or dogma, that outlaw potential for variability in text are just a really bad idea."

The long-term solution: full text customisation

The ideal long-term solution to the problems that some readers may experience with italics is, of course, user choice, by way of full text customisation. But how realistic a proposition is this?

Olaf Drümmer's 2012 paper, *How feasible is text customisation for PDF?* offers an interesting overview with respect to PDFs, especially in the light of the then recently published PDF/UA standard.

However, I would very much take issue with the paper's contention that colour customisation and text reflow in PDF only work well for very simple documents. On the contrary, these important features are very much controllable by proficient document authors and I would contend that a PDF that is not optimised for both cannot truly call itself an accessible PDF.

For more details on why this is so, please see PDF accessibility and reading order on the topic of reflow. With respect to colour customisation please see PDF background colour bug returns in Acrobat/Reader DC which explains one of the principal problems, and The InDesign workaround: paste into which provides the solution. (Note: fancy background colours are best avoided in MS Word which is a great word processor, but a not-so-hot page layout programme.)

I will leave to others more knowledgeable in the field to comment on the feasibility of full text customisation in web browsers.

Conclusions

There is a general consensus that large blocks of italic text are best avoided. However, a blanket ban on the use of italics will undoubtedly strip some texts of much of their meaning. Document producers therefore need to be free to use their judgment, and bring their experience and knowledge to bear on the question of the appropriate use of italic text in online documentation.

Until such a time that all the major reading systems can offer good levels of text customisation, common sense, compromise, good judgment and suitable levels of document editing and typographic skills seem the way to go. By contrast, a blanket ban on italics is an over-reaction to a problem which will, in many cases, do more harm than good.

Accessible continuation sheets in PDF forms, made easy

3rd February 2020 | by Ted Page

How a little (beginner-level) JavaScript can save the day

Many different types of PDF form necessarily contain answer spaces that span two or more pages.

The problem

But there is potentially a serious accessibility problem with creating a multi-page answer space in a PDF form: on a Mac you will get a beep to tell you when you have filled up a text field and need to move on to the next page to continue. However, on a Windows machine, out of the box, there is no such warning.

As a result, having filled up the space available in a particular field, users who can't see the page could keep on typing, unaware that everything they subsequently write will be lost. If you are sitting an exam, or filling out a job application form, the consequences could be very serious, if not catastrophic.

The solution

The following, easy-to-apply script, will solve the problem.

```
if (event.fieldFull) app.alert("You have filled the given space with text. \nPlease press the space bar to close this message and then press the tab key to go to the next page to continue.", 1);
```

This script alerts users, by way of a screen reader accessible pop-up, that the field is full, and that they should press the space bar to close the pop-up, and then press the tab key to go to the next field.

Example form

An example of this can be seen in action in the "additional information" field in the sample PDF form available on our Accessible PDF forms training course web page.

(Note: you don't actually have to type the 1,611 characters that it would take to fill this field. Instead you can just press the Enter key 23 times.)

Remote continuation sheets

If the continuation sheet is not the next field in the form's tab order (it could, for example, be an additional information or notes page at the end of a document) you may also choose to add the following to the above script:

```
this.getField("fieldName").setFocus();
```

…where "fieldName" is the continuation field's name attribute value. You would, of course, also have to amend the message in the pop-up as appropriate.

WCAG compliance

Given that you have warned the user as to what is about to happen, this should be fine with respect to WCAG 2.1 SC 3.2.2 (change of context). However, if you are not comfortable with this, the pop-up message could be amended to advise the user how to navigate to the appropriate page, perhaps using the **`Ctrl`** + **`Shift`** + **`n`** page navigation command.

Adding the script in Acrobat Pro

To add this script to a form field in Acrobat Pro:

- Select **Tools**, **Prepare Form**
- Go to the field in question and double click to open the **Text Field Properties** dialogue box
- Select the **Format** tab
- From the **Select format category** dropdown select **Custom**
- In the **Custom Keystroke Script** section click the **Edit** button
- Type or paste the script into the **JavaScript Editor** dialogue box and click **OK**

When the user now runs out of space in this field, he or she will get the above pop-up advising of that fact, and on how to proceed.

Conclusion

This simple solution can be applied in seconds to any continuation sheet in a PDF form. By contrast, not alerting users that they have reached the character limit in a particular field, and hence that any further text they might type would be lost, could have potentially catastrophic consequences in some contexts, and would at least present a serious obstacle in others.

Creating accessible links in PDFs

11th April 2010 | by Ted Page

Summary

Creating accessible links in PDFs is a basic accessibility requirement. This article looks at a selection of techniques for tagging links correctly to ensure hat they are both keyboard-operable and usable with a screen reader. It also looks at, amongst other things, how to make URLs more intelligible for screen reader users.

Tagging

Creating links that are keyboard operable is a single A requirement under the Web Content Accessibility Guidelines (WCAG) 2.0., and is vital in ensuring the accessibility of PDFs.

Figure 1: screen reader accessible link tag structure

For a link to be accessible it must have a specific tag structure consisting of:

- a parent **<Link>** tag
- one or more child tag(s) containing the link text
- a **Link–OBJR** tag which must also be a child of the **<Link>** tag

The **Link–OBJR** tag enables assistive technologies to properly recognise the link and handle it correctly.

Word, InDesign and beyond

Provided that PDFs are tagged on creation, links originated in Microsoft Word documents will have the required tag structure and will present no problems (provided that they don't span two or more lines). However, links currently authored in InDesign will need fixing, as will those found in untagged PDFs.

InDesign originated links

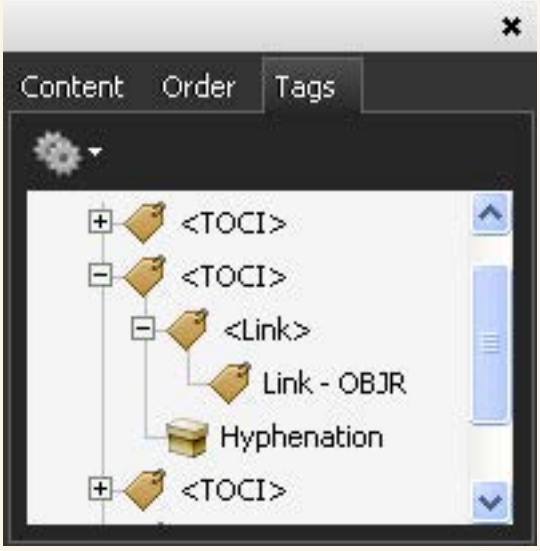


Figure 2: InDesign CS4 and CS5 link tag structure

As of the date of writing, hyperlinks created in InDesign (CS4), when converted to PDF, will produce inaccessible tag structures. An example can be seen in Figure 2 above. [Update (May 2011): note that this problem was resolved in InDesign version CS5.5 and later.]

The problem is that the tag containing the link text will be created at the wrong level in the tag tree hierarchy. The tag in question – in this case containing the word "Hyphenation" – will be created at the same level as the **<Link>** tag, rather than being a child of it.

Such links work with a mouse and appear normally in the tab order but don't work with screen readers.

However, the problem is easily solved. Just click and drag the link text tag ("Hyphenation") and drop it underneath (as child of) the **<Link>** tag, at the same level as the **Link-OBJR** tag. Figure 3 below shows the same tag tree corrected.

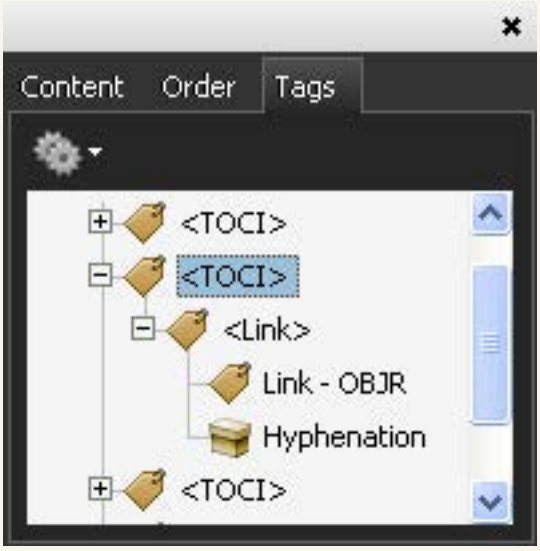


Figure 3: corrected InDesign CS4 or CS5 link tag structure

Creating accessible links in Acrobat

Creating an accessible link from scratch in Acrobat Professional is (usually) relatively straightforward. To do so, highlight the text in question, right-click and, from the context menu, select **Create Link**. Choose **Go to a page view**.

Creating link tags manually

In some PDFs the create link process outlined above won't work (in some Quark-generated documents, for example, or when links span across line breaks). In such a case it may be necessary to create **<Link>** and **Link–OBJR** tags manually.

Creating a Link tag

To create a **<Link>** tag, in the tags panel select **New Tag**, either by right-clicking or from the **Options** menu. In the **Type** input field, either type "**Link**" or select **Link** from the dropdown.

Create a Link-OBJR tag

To create a **Link-OBJR** tag, in the tag tree, either right-click or open the **Options** menu and select **Find**. From the **Find** dropdown select **Unmarked Links**. Click **Find** and then **Tag Element**. A **Link–OBJR** tag will be created.

Making links screen reader friendly

Because PDFs are often designed to be printed, link text frequently comes in the form of raw URLs which may not be particularly screen reader friendly. It can be helpful to screen reader users to provide texts alternatives for such links in order to make them easier to understand. (In terms of WCAG 2.0 compliance this is a triple A requirement but fixing it is arguably more beneficial than this would imply).

Again, the solution is simple. Highlight the URL and press **`Ctrl + C`** to copy it. In the tag tree, right-click the **<Link>** tag and select **Properties**. In the **Alternative Text** field of the **TouchUp Properties** dialogue box, paste the previously copied URL.

Then, for a URL such as www.ecdp.org, type "www.e c d p.org" (with spaces between the "e", "c", "d" and "p"), or for www.bbc.co.uk, type "www.bbc.co.u k" with a space between the "u" and the "k".

Following these simple techniques will ensure that your links are accessible to a wide range of readers.

Index

S

T

U

W