find it  main

T E C H N O T E 1088

Strategies for Producing Browser-Based Technical Documentation

By Tom Maremaa
Apple Developer Technology Services (DTS)
maremaa@apple.com

CONTENTS

For better or worse, Web browsers have become one of the preferred ways of viewing technical documentation. Some 30,000 pages of *Inside Macintosh* were recently converted to HTML for online viewing and retrieval -- a monumental and important achievement. Yet Web browsers and technical documents make strange bedfellows indeed. Viewing documents in a browser (especially if those documents contain complex technical data, extensive illustrations, block diagrams, tables and cross-references) may be less than satisfactory -- the equivalent of reading programmer manuals in a fishbowl.

The problem may be that many technical documents originated in other forms, such as books, articles, or dissertations, and when converted to straight HTML -- without concern for how the material will be "read" in a Web browser -- show their true roots. They beg to be printed or read in book format, which may frustrate those engineers and programmers who are reading them through their Web browsers.

This Note attempts to provide a few good strategies for resolving some of the issues around producing and viewing Web-based technical documentation. The Note may be useful for engineers, technical writers and content producers who must wrestle with issues of producing documents such as ReadMe files, Release Notes, technical articles, and other forms of technical communication that land on the Web.

The Omnipresent Trend: Web Bloat

There is no doubt that the number of technical documents produced and published on the Web over the last several years has grown exponentially. With powerful new search engines indexing every known document ever produced, the Web is a tremendous resource for developers and programmers who need to get information on the latest Macintosh system updates, for example, or answers to frequently-asked technical questions, such as those on Apple's Technical Q&A site. This has accelerated the trend for many companies to throw every imaginable piece of technical information on the Web -- often without regard for how that information is to be understood or used by the readers who visit that company's Website. Indeed, Web bloat is here and shows no signs of abating. This may make it more difficult, in fact, to find necessary documents to aid in your development efforts, but that's another issue, tangential to this Note.

Factors to Consider

As more and more technical documentation is being converted to HTML, it is important to consider several factors:

1. How you expect the readers who come to your Website to follow the line of reasoning presented in each document, and what you want them to take away from the experience of reading and viewing these documents online. (Of course, you can assume that your readers may download the Acrobat versions of your documents, and then print them out and read them in paper form. But in truth, how you present your technical documents in HTML may dictate whether your Web readers follow through by downloading the Acrobat versions and printing them out.)

2. Finding ways of increasing "reader interactivity" with your technical documents by adding hypertext links, buttons and cross-references, as well as embedding binhexed, downloadable files within the body of each document. For example, one of the most popular Notes I've produced is Technote 1031, "History & Peregrinations: The Dogcow Goes QuickTime VR," which contains a number of embedded, downloadable files that are designed to illustrate the main points of the piece.

Developing a Few Strategies for HTML Technical Publishing

There are several strategies you can develop to make sure that your technical documents, when converted to HTML, are structured for optimal browsing.

Strategy #1: Produce the Originals in HTML

If your technical documents are produced originally in HTML, then reviewed and distributed as HTML documents before getting posted on your Website, you'll be in a much better position to assess if they really work for Web viewing and understanding. This may belabor the obvious, but you would be surprised how conditioned we are to think in terms of "paper first" and HTML only as an afterthought.

The idea is to be able to determine the response to a technical document by seeing if that document works structurally to get across its main points. By writing a document with your word processor and then printing that document for review and allowing reviewers to comment and mark up the document in pen or pencil, you are in effect avoiding all consideration of how your reader will view or respond to the content in a Web browser. It's a little like writing a screenplay and handing it out to moviegoers at the theatre and asking them to imagine the results onscreen.

Producing documents originally in HTML may seem like an impossible chore, and until recently it was because you had to write the document and then insert all the requisite HTML tags so your browser could display it. But that's changed dramatically over the last year.

Because of the quality of HTML authoring and editing tools available, you can ask your technical writers, engineers and programmers to produce documents initially in HTML. This won't be a big stretch for them, if they're not doing it already. I won't list all the tools -- Claris Home Page, Adobe PageMill, BBEdit, and FrontPage are just a few -- that offer technical writers and engineers the power and flexibility to generate HTML documents, often without having to write even a single line of HTML code.

If desktop publishing was the wave of the 1980s, propelling forward a new generation of graphic artists, illustrators, printers and electronic publishers, certainly HTML publishing is the wave of the 1990s. Picture an environment where all of us are familiar with the basics of HTML and work with one of the HTML authoring and editing tools available.

All indications are that HTML is becoming the lingua franca of the known universe, as more and more documents are written, exchanged, published, and read in Web browers. If Marshall McLuhan changed the world in one sentence -- The Medium is the Message -- I think the world has changed us by making us dance to the tune of HTML everywhere.

Strategy #2: Build an HTML Template That Everybody Can Use

Producing HTML technical documents at the outset not only makes more sense from the point of view of presenting content to your readers -- it just saves more production time in the fast and frenzied world of Web publishing.

For this process to work, you need to build a template in HTML for your technical documents. If you build and distribute such a template, you can then ask your technical writers and engineers to pour the contents of their drafts into the template for peer review.

With Claris Home Page, for example, you can use the Strikethrough feature and color text to indicate changes and markups made by your reviewers electronically, and then when you are satisfied with the results move to publish the documents on your Website.

We've developed a template for Technotes, which you can download by clicking here.

The template is designed specifically for the content of Technotes. It's a distillation of what works best for those who read and rely on Technotes for information from Apple.

Because Technotes work with a Thesis-Antithesis-Synthesis structure, the first few paragraphs of a Note are important for setting up the main thesis. The thesis is not an introductory paragraph, as in the traditional essay structure, but rather a formulation of a particular theory or problem, which is then "proved" or "disproved" with code snippets, diagrams or examples that follow. Each section of the Note must contribute to the issues raised at the beginning. The thesis is then resolved in a summary section at the end of the Note.

With this content structure in mind, we designed an HTML format that would work most effectively for the reader's understanding of the issues presented in a Note. The idea here was a simple marriage of form and content with HTML as the minister of ceremonies.

Note:
Delivering your documents in HTML does not necessarily mean you have to give up using your tried-and-true text editor. You can still produce the initial drafts of your documents in a text editor of your choosing. From that point on, you can copy and paste the content into an HTML template that you've customized, such as the one we've created for Technotes, and then distribute that HTML document for review among your peers.

Countering Resistance to the Strategy

In proposing these strategies, I can hear the voices of resistance clamoring with arguments and counterarguments.

Resistance #1

Aren't we giving up the paper/printed versions of our documents by moving right to HTML?

The answer is yes and no. Anyone can still print out an HTML document and read it to their heart's content. In fact, they can print it out in a font of their choosing and seven different font sizes.

Resistance #2

How can you assume that everybody who reads, reviews, or comments on a technical document has a Web browser?

I heard this one about a year ago and agreed with the answer implicit in the question. A year later, well...

Resistance #3

What about the PDF or Acrobat delivery of my documents?

The answer is that it's not difficult at all to produce a PDF or Acrobat version of your document from HTML. It just takes a little tweaking of that document, then printing it from your Web browser to a file for Acrobat distilling and conversion to PDF.

Using Tables in your HTML Template: A Sidebar

If you download this Technote template, you'll see that it is set up with a series of tables for each section and each code snippet. The code snippets are in a gray table color, notes are in yellow. (Code snippets can be placed in a table and specified as preformatted text.) Tables, where you specify the width you want the information in them displayed, are the way to go.

If you open the template in Claris Home Page 2.0, for example, you'll see that the tables have a thin dotted line demarcating their boundaries. This is very useful for capturing and viewing technical information; it also helps in terms of reviewing the content.

What's cool is that you can embed tables within tables within tables and live happily with the results.

Web browsers are here to stay as one of the preferred ways of viewing, navigating and retrieving technical documentation. With that in mind, if you begin to produce your documents originally in HTML, you'll have a better idea of how they'll be read and understood in a browser-based viewing environment. This Note provides you with a few strategies you can implement to make this happen.

Further References

- Technote 1031: History & Peregrinations: The Dogcow Goes QuickTime VR
- Technote 1047: On Creating Web-Friendly Documentation

Acknowledgments

Thanks to Otto Schlosser, Vinnie Moscaritolo, Jeanne Woodward, and John Gorham.

Send feedback to maremaa@apple.com
Updated: 20-December-96