# Beware of XHTML

Advancing web developers have probably heard about XHTML, the eXtensible HyperText Markup Language developed in 1999 to supercede HTML. Most people use XHTML simply because they believe they are staying on top of technology. But there is a lot more to it than you may realize, and if you're using it on your website, even if it validates, you are probably using it incorrectly.

## What is XHTML?

XHTML is a markup language meant to eventually replace HTML on the Web. XHTML 1.0 differs from HTML 4.01 only in the format that underlies the language. HTML is written in SGML, the Standard Generalized Markup Language, while XHTML is written in XML, the eXtensible Markup Language. XML has several improvements over SGML, including the ability to combine multiple markup languages in a single document, and forced user agent validation that makes it obvious when you have an error in your document.

As far as the element set, XHTML 1.0 is the same language as HTML 4.01. The only added benefit of XHTML is that it is an XML format and shares the benefits XML has over SGML.

## Content type is everything

When your website sends a document to the visitor's browser, it adds on a special content type header that lets the browser know what kind of document it's dealing with. For example, a PNG image has the content type image/png and a CSS file has the content type text/css. HTML documents have the content type text/html. Web servers typically send this content type whenever the file extension is .html, and server-side scripting languages like PHP also typically send documents as text/html by default.

XHTML does not have the same content type as HTML. The proper content type for XHTML is application/xhtml+xml. Most web servers currently don't have this content type reserved for any file extension, so you would need to modify the server configuration files or use a server-side scripting language to send the header manually.

When a web browser sees the text/html content type, regardless of what the doctype says, it automatically assumes that it's dealing with plain old HTML. Therefore, rather than using the XML parsing engine, it treats the document like tag soup, expecting HTML content. Because HTML 4.01 and XHTML 1.0 are so similar, the browser can still understand the page fairly well. It considers things like the self-closing portion of a tag (as in <br />) as a simple HTML error and strips it out, usually ending up with the HTML equivalent of what the author intended.

However, when the document is treated like HTML, you get none of the benefits XHTML offers. The browser won't understand other XML formats like MathML and SVG that are included in the document, and it won't do the automatic validation that XML parsers do. In order for the document to be treated properly, the server would need to send the application/xhtml+xml content type.

The problems go deeper. Comment markers are sometimes handled differently between SGML and XML, and when you enclose the contents of a script or style element with comments, it will cause those sections to be ignored when the document is treated like XML. Furthermore, the CSS and DOM specifications have special provisions for HTML that don't apply to XHTML when it's treated as XML, so your page may look and behave in unexpected ways.

The following are some examples of differing behavior between XHTML treated as HTML and XHTML treated as XML. The anticipated results are based on the way Internet Explorer, Firefox, and Opera treat XHTML served as HTML. Some other browsers are known to behave differently. Also note that Internet Explorer doesn't recognize the application/xhtml+xml content type (see below for an explanation), so it will not be able to view the examples in the second column.

*Examples table removed, files attached to the article in the archives.*

## HTML compatibility guidelines

When the XHTML 1.0 specification was first written, there were provisions that allowed an XHTML document to be sent as text/html as long as certain compatibility guidelines were followed. The idea was to ease migration to the new format without breaking old user agents. However, these provisions are now viewed by many as a mistake. The whole point of XHTML is to be an XML format, yet due to the allowance of XHTML documents to be sent as text/html, most so-called XHTML documents on the Web now would break if they were treated like XML. Aware of the problem, the first revision of the XHTML specification had these provisions removed. In XHTML 1.1 and onward, it is now incorrect to send XHTML documents as text/html under any circumstances. XHTML should be sent as application/xhtml+xml or one of the more elaborate XHTML content types.

## Internet Explorer incompatibility

Internet Explorer does not support XHTML. Like other web browsers, when a document is sent as text/html, it treats the document as if it was a poorly constructed HTML document. However, when the document is sent as application/xhtml+xml, Internet Explorer won't recognize it as a webpage; instead, it will simply present the user with a download dialog. The Internet Explorer development team has announced that they will not be able to add XHTML support in the upcoming IE7 either.

Although all other major web browsers, including Firefox, Opera, Safari, and Konqueror, support XHTML, the lack of support in Internet Explorer as well as major search engines and web applications makes use of it very discouraged.

## Null End Tags (NET)

In XHTML, all elements are required to be closed, either by an end tag or by adding a slash to the start tag to make it self-closing. Since giving empty elements like img or br an end tag would confuse browsers treating the page like HTML, self-closing tags tend to be promoted. However, XML self-closing tags directly conflict with a little-known SGML feature: Null End Tags.

A Null End Tag is a special shorthand form of a tag that allows you to save a few characters in the document. Instead of writing <title>My page</title>, you could simply write <title/My page/ to accomplish the same thing. Due to the rules of Null End Tags, a single slash in an empty element's start tag would close the tag right then and there, meaning <br/ is a complete and valid tag. As a result, if you have <br/> or <br />, a browser supporting Null End Tags would see that as a br element immediately followed by a simple > character. Therefore, an XHTML page treated as HTML could be littered with unwanted > characters.

This problem is often overlooked because most popular browsers today are lacking support for Null End Tags, as well as some other SGML shorthand features. However, there are still many smaller user agents that properly support Null End Tags. One of the more well-known user agents that support it is the W3C validator. If you send it a page that uses XHTML self-closing tags, but force it to parse the page as HTML/SGML like most user agents do for text/html pages, you can see the results in the parse tree: immediately after each br element, there is an unwanted > character that will be displayed on the page itself. An HTML doctype was used in this case just so the validator would use the SGML parser, although an SGML parser should give the same results even with an XHTML doctype.

In summary, although the effects don't show in most popular web browsers, a user agent that more fully supports SGML would see unwanted > characters all over XHTML pages that are sent with the text/html content type, and such user agents do exist and frequently run into this issue.

## Conclusion

XHTML is a very good thing, and I certainly hope to see it gain widespread acceptance in the future. However, it simply isn't widely supported in its proper form. XHTML is an XML format, and to force a web browser to treat it like HTML is going against the whole purpose of XHTML and also inevitably causes other complications. Assuming you don't want to dramatically limit access to your information, XHTML can only be used incorrectly, be interpretted as invalid markup by most user agents, cause unwanted results in others, and offer no added benefit over HTML. HTML 4.01 Strict is still what most user agents and search engines are most accustomed to, and there's absolutely

nothing wrong with using it if you don't need the added benefits of XML.

---