W3C® Internationalization

The byte-order mark (BOM) in HTML

intended audience: XHTML/HTML coders (using editors or scripting), script developers (PHP, JSP, etc.), CSS coders, Web project managers, and anyone who needs to better understand what the BOM is, and how it affects HTML.

Updated 2013-01-31 17:18

About this article

This article has been reviewed by the W3C Internationalization Working Group and has gone through public review to make it as accurate as possible.

If you have comments, please let us know.

QUESTION

What is the byte-order mark, and what do I need to know about it when creating HTML?

ANSWER

What is a byte-order mark?

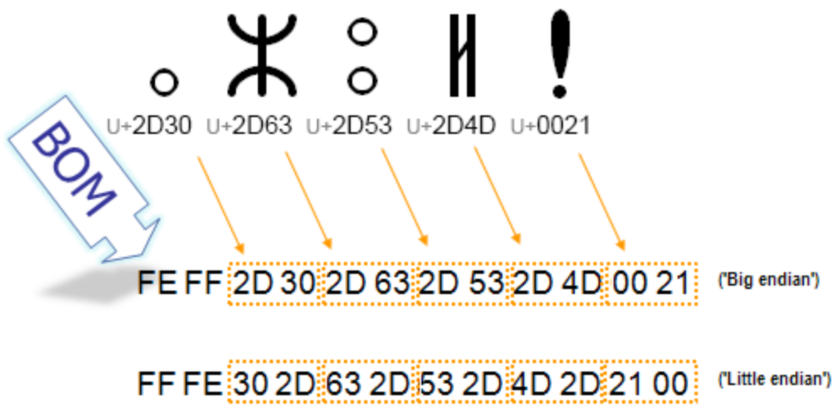
At the beginning of a page that uses a Unicode character encoding you may find some bytes that represent the Unicode code point U+FEFF BYTE ORDER MARK (abbreviated as **BOM**).

The BOM, when correctly used, is invisible.

The name BYTE ORDER MARK is an alias for the original character name ZERO WIDTH NO-BREAK SPACE (ZWNBSP). With the introduction of U+2060 WORD JOINER, there's no longer a need to ever use U+FEFF for its ZWNSP effect, so from that point on, and with the availability of a formal alias, the name ZERO WIDTH NO-BREAK SPACE is no longer helpful, and we will use the alias here.

Before UTF-8 was introduced in early 1993, the expected way for transferring Unicode text was using 16-bit code units using an encoding called UCS-2 which was later extended to UTF-16. 16-bit code units can be expressed as bytes in two ways: the most significant byte first ('*big-endian*') or the least significant byte first ('*little-endian*'). To communicate which byte order was in use, U+FEFF (the byte-order mark) was used at the start of the stream as a magic number that is not logically part of the text the stream represents.

The picture below shows the bytes used in a sequence of two-byte characters. Each 2-digit hexadecimal number represents a byte in the stream of text. You can see that the order of the two bytes that represent a single character is reversed for big endian vs. little endian storage. The byte-order mark indicates which order is used, so that applications can immediately decode the content.



U+2D30
U+2D63
U+2D53
U+2D4D
U+0021
BOM
FE FF 2D 30 2D 63 2D 53 2D 4D 00 21
('Big endian')
FF FE 30 2D 63 2D 53 2D 4D 2D 21 00
('Little endian')

In the UTF-8 encoding, the presence of the BOM is not essential because, unlike the UTF-16 encodings, there is no alternative sequence of bytes in a character. However, the BOM may still occur in UTF-8 encoded text, either as a by-product of an encoding conversion or because it was added by an editor to flag the content as UTF-8. In this situation, the BOM is often called a **UTF-8 signature**.

What do I need to know about the BOM?

Most of the time you will not have to worry about the byte-order mark in UTF-8. You will find that some editors (such as Notepad on Windows) will always add a BOM when you save a file with the UTF-8 encoding, others will offer you a choice.

In HTML5 browsers are required to recognize the UTF-8 BOM and use it to detect the encoding of the page, and recent versions of major browsers handle the BOM as expected when used for UTF-8 encoded pages.

The UTF-8 BOM offers reliable encoding detection, since it is extremely short and stable, works in XML and HTML, and works whether your page is read over the network or not (unlike HTTP declarations). However, bear in mind that it is always a good idea to declare the encoding of your page using the meta element, in addition to the BOM, so that the encoding is apparent to people looking at the source text.

Also there are a number of situations where the BOM, particularly because it is invisible, may cause a problem. See the section below for more information about those.

If you use a UTF-16 encoding for your page (and we strongly recommend that you don't), there are some additional considerations.

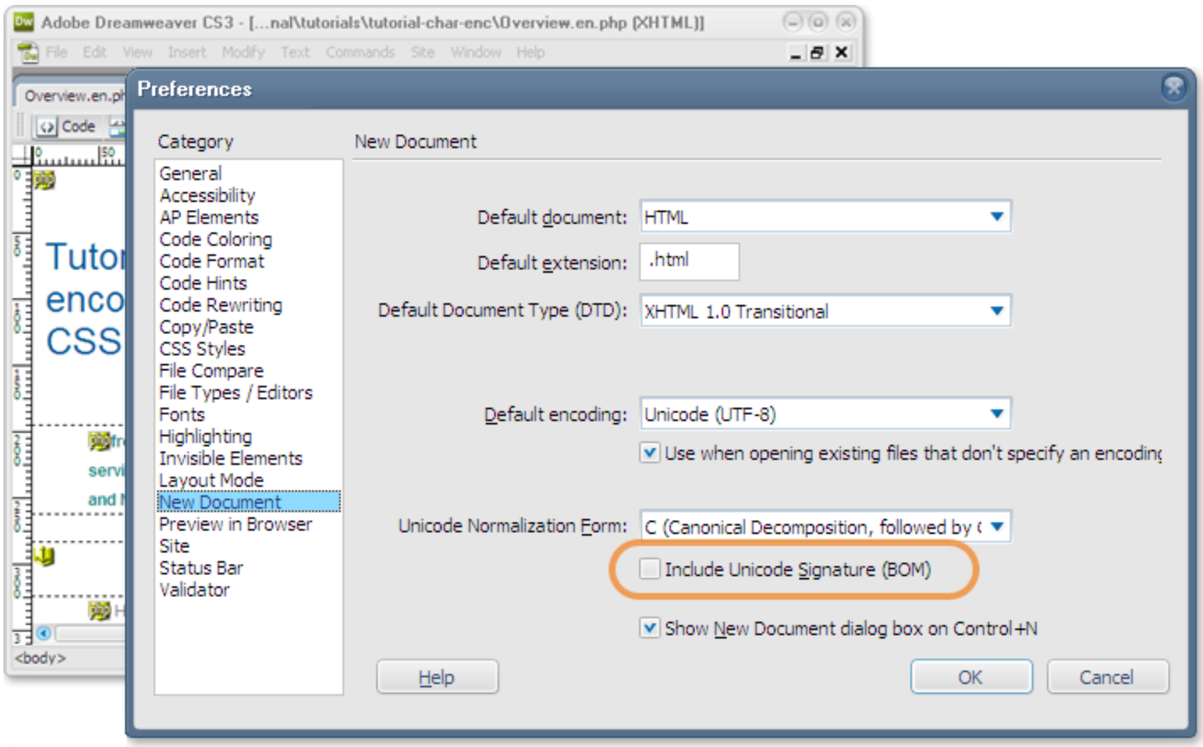
Detecting the BOM

You can find out whether a page contains a BOM at the start or further down in the content by using the W3C Internationalization Checker. A BOM at the start of the page will be reported in the Information panel. A BOM that is included in the page lower down (typically due to content being added to the page from an external source) will be reported in the Detailed Report section.

You can try looking for a UTF-8 signature in your content in your editor, but if your editor handles the BOM correctly you probably won't be able to see it. With a binary editor capable of displaying the hexadecimal byte values in the file, the UTF-8 signature displays as EF BB BF.

If your editor or browser applies the wrong character encoding to a UTF-8 encoded file with a BOM, you are likely to see a sequence of bytes at the start of the file. These are the bytes that compose BOM represented as the characters those bytes represent in that encoding. With the Latin 1 (ISO 8859-1) character encoding, the signature displays as characters ï»¿.

Alternatively, your editor may tell you in a status bar or a menu what encoding your file is in, including information about the presence or not of the UTF-8 signature. For example, if you use Save As in Dreamweaver and your file has a BOM at the start you will see a check mark in the box labeled 'Include Unicode Signature (BOM)'. You can also specify in your preferences (see illustration) whether new documents should use a BOM by default.


Preferences

New Document

Default document: HTML

Default extension: .html

Default Document Type (DTD): XHTML 1.0 Transitional

Default encoding: Unicode (UTF-8)

[✓] Use when opening existing files that don't specify an encoding

Unicode Normalization Form: C (Canonical Decomposition, followed by C

[ ] Include Unicode Signature (BOM)

[✓] Show New Document dialog box on Control+N

Potential issues with the UTF-8 BOM

What follows are some situations where the byte-order mark has been known to cause problems.

In general, these issues are fading away as people adopt newer versions of browsers and editing tools. It is worth knowing about them if your user base still uses older technology. However, this is not solely about legacy issues.

PHP includes

At the time this article was written, if you include some external file in a page using PHP and that file starts with a BOM, it may create blank lines.

This is because the BOM is not stripped before inclusion into the page, and acts like a character occupying a line of text. See an example. In the example, a blank line containing the BOM appears above the first item of included text.

You should ensure that the included files do not start with a BOM.

You may also find that the BOM causes problems for an ordinary PHP page. When sending custom HTTP headers the code to set the header must be called before output begins. A BOM at the start of the file causes the page to begin output before the header command is interpreted, and may lead to error messages and other problems in the displayed page.

Processing with program code

You need to be careful to take the BOM into account in scripts or program code that automatically process files that start with a BOM. For example, when pattern matching at the start of a file that begins with a BOM you need additional code to test for the presence of the BOM and ignore it if found.

The UTF-8 encoding without a BOM has the property that a document which contains only characters from the US-ASCII range is encoded byte-for-byte the same way as the same document encoded using the US-ASCII encoding. Such a document can be processed and understood when encoded either as UTF-8 or as US-ASCII. Adding a BOM inserts additional non-ASCII bytes, so this is no longer true. If you have processes or scripts that assume that the content is comprised of US-ASCII characters only, you will need to avoid the BOM.

HTTP precedence

Changes introduced with HTML5 mean that the byte-order mark overrides any encoding declaration in the HTTP header when detecting the encoding of an HTML page. This can be very useful when the author of the page cannot control the character encoding setting of the server, or is unaware of its effect, and the server is declaring pages to be in an encoding other than UTF-8. If the BOM has a higher precedence than the HTTP headers, the page should be correctly identified as UTF-8.

At the time of writing, not all browsers do this, so you should not rely on all readers of your page benefitting from this just yet.

> Previous versions of Internet Explorer gave the BOM precedence over HTTP, but IE10 and IE11 give a higher precedence to HTTP. It is hoped that the next version of Internet Explorer will revert to the previous behaviour, which will then be in line with the other major browsers.

In browsers where the HTTP header still overrides the byte-order mark and the server is declaring pages to have a non-Unicode character encoding, you are likely to find unexpected characters at the start of the page (such as ï»¿ in a page labelled in HTTP as ISO 8859-1) as well as problems displaying non-ASCII characters on the page.

Other issues

If you use applications or scripts in the back end of your site you should check that they are also able to recognize and handle the BOM.

We strongly recommend that you don't change the encoding of a UTF-8 file from a Unicode encoding to a non-Unicode encoding, but if, for some exceptional reason, you do you must ensure that the BOM is removed. If you don't, either the browser will continue to treat your content as UTF-8, or you will see strange characters at the beginning of the page.

Removing the BOM

If you need to remove the BOM, check whether your editor allows you to specify whether a UTF-8 signature is added or kept while you save the file. Such an editor provides a way of removing the signature by simply reading the file in then saving it out again. For example, in editors such as Notepad++ on Windows and TextWrangler on the Mac, it is possible to select the encoding from a list while using the Save As function. The list has options to save as UTF-8 with or without the BOM. Just choose the option without the BOM and save.

One of the benefits of using a script is that you can remove the signature quickly, and from multiple files. In fact the script could be run automatically as part of your process. If you use Perl, you could use a simple script created by Martin Dürst.

Note: You should check the process impact of removing the signature. It may be that some part of your content development process relies on the use of the signature to indicate that a file is in UTF-8. Bear in mind also that pages with a high proportion of Latin characters may look correct superficially but that occasional characters outside the ASCII range (U+0000 to U+007F) may be incorrectly encoded.

ADDITIONAL INFORMATION

Here are some additional notes for those who are encoding their HTML pages using UTF-16. Note that, for HTML it's recommended that you use UTF-8 and that you avoid UTF-16. So for most people this section will be academic.

According to RFC 2718 and the Unicode Standard, if you declare the character encoding of your page using HTTP as either "UTF-16LE" or "UTF-16BE" then you should not use a byte-order mark at the beginning of the page. Only if the page is labelled in HTTP using IANA charset name "UTF-16" is a byte-order mark appropriate.

! Note that this is solely about the *labeling* of the content. Of course, the actual sequence of bytes is the same, whether you label content as UTF-16 and add a BOM, or whether you label it as UTF-16LE or UTF-16BE.

The HTML5 specification currently disallows the use of any other, text-based in-document encoding declaration for pages using the UTF-16 encoding. In effect, this means that the BOM is, itself, the declaration that you have to add.

The byte-order mark is also used for text labeled as UTF-32, and should not be used for text labeled as UTF-32BE or UTF-32LE. The use of UTF-32 for HTML content, however, is strongly discouraged and some implementations have removed support for it, so we haven't even mentioned it until now.

FURTHER READING

- Getting started? *Introducing Character Sets and Encodings* (/International/getting-started/characters)
- Tutorial, *Handling character encodings in HTML and CSS* (/International/tutorials/tutorial-char-enc/)
- Related links, *Authoring HTML & CSS*

Characters (/International/techniques/authoring-html#charset)

Declaring the character encoding for HTML (/International/techniques/authoring-html#indoc)

Links in this document:

By: Richard Ishida, W3C.

Thanks also to the following people whose contribution or feedback was included: Albert Lunde, Asmus Freytag, Björn Höhrmann, Henri Sivonen, John Cowan, Leif Halvard Silli, Norbert Lindenberg

Content first published 2010-08-10. Last substantive update 2013-01-31 17:18 GMT. This version 2016-04-20 11:10 GMT

For the history of document changes, see the news feed for substantive changes, and the Github commit list for all changes since Jan 2016.