Html – Polyglot Markup Rules

By John Bentley

Introduction

What is polyglot markup?

This document is based on the following standard

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) www.w3.org/TR/html-polyglot/

Essentially that entails producing markup which is HTML (5) valid and is XML valid, XHTML (5) in other words, with a small number of additional restrictions. That results in a markup that can be switched between being served as either of the MIME types "application/xhtml+xml" or "text/html", while rendering identically in the browser.

Mime Types and file extensions

To change the mime type with which the pages are served change one's file extension from ".xhtml" to ".html". The former will serve the page as the mime type "application/xhtml+xml" (which will signal to browsers to render it as XHTML), the later "text/html" (which will signal to browsers to render it as HTML).

Most web server's are configured, by default, to behave this way. That is, to send those respective mime types in the HTTP header as a function of the web page's file extension. But your web server's configuration would need to be verified.

Why chose polyglot markup?

Choosing to use polyglot markup helps if one's predilection for using the more strict XHTML (XML) syntax is meet with resistance from the HTML (non-XML) Syntax crowd, the more popular choice, as one move's one's web pages into that social environment. One need only changes one's file extension from .xhtml to .html and the pages are going to render as before.

Why Choose the XHTML (XML) syntax over HTML (non-xml) syntax?

But, if one had the freedom to choose, why would one choose to use XHTML (XML) syntax over HTML (non-xml) syntax?

- When you serve XHTML as "application/xhtml+xml" a browser will firstly check that it is valid XML. If there's any error that entails it is not valid XML the browser will throw an error message pointing to the problem. The helps ensure that many basic markup errors are caught early … before having to run the file through an external validator, like https://validator.w3.org/unicorn/ (although using an external validator is still good to do, after the basis errors have been caught).
- XML tools are available to use both in the creation of, and the consumption of, the page. On the consumption of the page, for example, you could issue an HTTP request from one's favoured programming language and parse the incoming data as XML.

  Even if one doesn't plan to use XML tools to create or consume web pages leaving that option open gives you an option that you wouldn't otherwise have. You never know when life might be made easier if your web pages are already XML valid. This is evidenced by the example given for pandoc "epub3 requires XHTML5" (https://github.com/jgm/pandoc/issues/5146#issuecomment-447069909).

Basics

XML declaration and XML Processing Instruction

Polyglot: the XML declaration and XML processing instructions are forbidden.

```
// OK
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

// Forbidden
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<?xml-stylesheet href="/style.css" type="text/css" title="default stylesheet" ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) *http://www.w3.org/TR/2014/CR-html-polyglot-20140717/#PI-and-xml*

Doctype and No-Quirks mode (formerly 'standards' mode)

For XHTML Syntax, to ensure no-quirks mode (formerly 'standards' mode):

- Give your file an extension ".xhtml"; and
- Verify your server thereby serves the page with a mime type of application/xhtml+xml;
- You don't need to worry about the presence of a doctype or XML declaration, for the sake of trigger no-quirks (formerly 'standards') mode. But other parts of the convention mandate the presence of the Doctype and the absence of the XML declaration.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

Serving a web page as application/xhtml+xml is what triggers no-quirks mode (formerly 'standards' mode), the presence or absence of a DOCTYPE does not effect this (but should be included anyway for conformance with the HTML 5 spec) and it doesn't matter if there is an XML declaration, or any other line occurring before a DOCTYPE ... in terms of effecting no-quirks mode.

(Mozilla Developer Network, 2014. Quirks Mode and Standards Mode) *https://developer.mozilla.org/en-US/docs/Quirks_Mode_and_Standards_Mode*

For HTML Syntax, to ensure no-quirks mode (formerly 'standards' mode):

- Give your file an extension of ".html" and verify your server thereby serves the page with a mime type of text/html.
- Include <!DOCTYPE html>.
- Ensure the doctype declaration is on the first line.

```
<!DOCTYPE html>
<html>
...
```

(Mozilla Developer Network, 2014. Quirks Mode and Standards Mode) *https://developer.mozilla.org/en-US/docs/Quirks_Mode_and_Standards_Mode*

"A document is always set to one of three modes: no-quirks mode, the default; quirks mode, used typically for legacy documents; and limited-quirks mode. Unless stated otherwise, a document must be in no-quirks mode."

'No-quirks mode was originally known as "standards mode" and limited-quirks mode was once known as "almost standards mode". They have been renamed because their details are now defined by standards. (And because Ian Hickson vetoed their original names on the basis that they are nonsensical.)'

(W3C, 2015. W3C DOM4, W3C Last Call Working Draft 28 April 2015) *http://www.w3.org/TR/dom/#concept-document-no-quirks*

We always want to have a web file be served in no-quirks mode (formerly 'standards' mode):

- To check the mode served of a page you can do any of the following (by serving a web page into a browser):
- In Firefox (38.0.5 or greater) then Context Menu > View Page Info > General [Tab] > Render Mode ...
- Use Browser extension http://chrispederick.com/work/web-developer/ toolbar and observe the left most "Standards Compliance Mode" tick.
- In any browser use JavaScript to examine the document.compatMode.

" Returns the string "CSS1Compat" if document is in no-quirks mode or limited-quirks mode, and "BackCompat", if document is in quirks mode." (See http://www.w3.org/TR/dom/#concept-document-quirks)

- Use C:\Users\John\Documents\Sda\Code\web\Libraries\Html5Library\quirksModeCheck.js & see quirksModeCheckDemo.xhtml which implements an easy way to use the JavaScript document.compatMode in your webpage.

Mode dependencies:

Mime Type Served	**Mode**	**When**
application/xhtml+xml	No-quirks (formerly 'standards')	Always
	Quirks or almost quirks	Never
text/html	No-quirks	If a DOCTYPE is the first line of the page
	Quirks or almost quirks	Maybe if a DOCTYPE is not on the first line of the page; or if DOCTYPE is absent.

(Mozilla Developer Network, 2014. Quirks Mode and Standards Mode) *https://developer.mozilla.org/en-US/docs/Quirks_Mode_and_Standards_Mode*

Namespaces

Polyglot: the following are the only three default element namespaces permitted:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<math xmlns="http://www.w3.org/1998/Math/MathML">
<svg xmlns="http://www.w3.org/2000/svg">
```

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) *https://www.w3.org/TR/html-polyglot/#element-level-namespaces*

Polyglot: you must have the default xhtml namespace on the root html element.

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

"Polyglot markup declares the default namespaces on the root HTML element, html,"
https://www.w3.org/TR/html-polyglot/#element-level-namespaces

This is also reflected in the Polyglot "Minimal HTML document". https://www.w3.org/TR/html-polyglot/#minimal-polyglot-html-document

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) https://www.w3.org/TR/html-polyglot/#element-level-namespaces

> [Polyglot explicitly requires the declaration of] the default namespaces on the root HTML element, html, the root SVG element, svg, and the root MathML element math, and on any HTML elements used as children of SVG or MathML elements.

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) https://www.w3.org/TR/html-polyglot/#element-level-namespaces

Polyglot: the only explicitly declared attribute namespaces is XLink as follows.

```
xmlns:xlink="http://www.w3.org/1999/xlink"

// Must be declared before using any of the xlink attributes
xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type
```

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) http://www.w3.org/TR/html-polyglot/#attribute-level-namespaces

Polyglot: the implicitly declared attribute namespace allows you to use xml: attributes as follows.

```
// Allowed
xml:lang, and xml:id

// Disallowed in HTML5, allowed in SVG and MathML
xml:base, xml:space,
```

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) http://www.w3.org/TR/html-polyglot/#attribute-level-namespaces
(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) http://www.w3.org/TR/html-polyglot/#disallowed-attributes

Language Declaration

Polyglot Rules

Polyglot: If you specify the language you must use lang and xml:lang with identical values.

```
<p lang="en-au" xml:lang="en-au">Lorem</p>

<p lang="en" xml:lang="en">Lorem</p>
```

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) http://www.w3.org/TR/html-polyglot/#language-attributes

Polyglot: You SHOULD put a language specification in the root html element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-gb" xml:lang="en-gb">

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

> The root element SHOULD always specify the language, (W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation).

In the HTML spec there's no explicit exhortation to include a language tag (or language tags in the case of XHTML). That is, there's nothing like the phrase found above in the Polyglot spec. However, implicitly the HTML spec regards the inclusion of a language tag (or language tags in the case of XHTML) as something you should do to avoid problems:

> It provides an explicit indication to user agents about the language of content in order to enable language specific behavior. For example, use of an appropriate language dictionary; selection of an appropriate font or glyphs for characters shared between different languages; or in the case of screen readers and similar assistive technologies with voice output, pronunciation of content using the correct voice / language library.
>
> Incorrect **or absent lang attributes** can produce unexpected results in other circumstances, as they are also used to determine quotation marks for q elements, styling such as hyphenation, case conversion, line-breaking, and spell-checking in some editors, etc. [Emphasis added].

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation)
http://www.w3.org/TR/html-polyglot/#language-attributes
(W3C HTML5, 2017. HTML 5.2 W3C Recommendation, 14 December 2017) , "The lang and xml:lang attributes",
https://www.w3.org/TR/html/dom.html#the-lang-and-xmllang-attributes

Inheritance Rule

"The lang attribute (in no namespace) specifies the primary language for the element's contents and for any of the element's attributes that contain text. ... If these attributes are omitted from an element, then the language of this element is the same as the language of its parent element, if any."

```
  <p lang="en-au" xml:lang="en-au">
    I said, <q>Streuth you must be feeling <q>pretty ordinary</q> this morning.</q> She
said, <q lang="fr" xml:lang="fr">bonne nuit.</q>
  </p>

// Language of "Streuth you must be feeling ..." is en-au
// Language of "bonne nuit" is fr.
```

(W3C HTML5, 2017. HTML 5.2 W3C Recommendation, 14 December 2017)
http://www.w3.org/TR/html5/dom.html#the-lang-and-xml:lang-attributes

Setting the language attribute to an empty string indicates the language is unknown.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="" xml:lang="">
```

(W3C HTML5, 2017. HTML 5.2 W3C Recommendation, 14 December 2017)
http://www.w3.org/TR/html5/dom.html#the-lang-and-xml:lang-attributes
[default.html5. Polyglot Markup. #237. Comment 2017-01-27 14:59](https://github.com/jgm/pandoc-templates/issues/237#issuecomment-275120456)

Language tag standards

HTML5: "Tags for Identifying Languages, Request for Comments 5646" is the current "Best Current Practice" (BCP 47) and referenced from the (W3C HTML5, 2017. HTML 5.2 W3C Recommendation, 14 December 2017) HTML5 spec.

"BCP stands for 'Best Current Practice', and is a persistent name for a series of RFCs whose numbers change as they are updated." (W3C Language tags in HTML and XML, 2009. Language tags in HTML and XML) http://www.w3.org/International/articles/language-tags/

(Internet Engineering Task Force, Tags for Identifying Languages, 2009. Tags for Identifying Languages, Request for Comments 5646, Best Current Practice 47) https://tools.ietf.org/html/rfc5646 https://tools.ietf.org/html/bcp47

Language Tag Syntax (omitting details after variant). Note we speak of the "language tag", which references the whole thing, and a "language subtag", which is just the first part (and perhaps the only part) of a "language tag".

```
'A language tag is composed from a sequence of one or more "subtags" each of which refines
or narrows the range of language identified by the overall tag'.

 langtag       = language
                 ["-" script]
                 ["-" region]
                 *("-" variant)
                 *("-" extension)
                 ["-" privateuse]

 language      = 2*3ALPHA            ; shortest ISO 639 code
                 ["-" extlang]       ; sometimes followed by
                                     ; extended language subtags
               / 4ALPHA              ; or reserved for future use
               / 5*8ALPHA            ; or registered language subtag

 extlang       = 3ALPHA              ; selected ISO 639 codes
                 *2("-" 3ALPHA)      ; permanently reserved

 script        = 4ALPHA              ; ISO 15924 code

 region        = 2ALPHA              ; ISO 3166-1 code
               / 3DIGIT              ; UN M.49 code

 variant       = 5*8alphanum         ; registered variants
               / (DIGIT 3alphanum)
```

For understanding the above Augmented Backus-Naur Form see ...
(Internet Engineering Task Force (IETF), 2008. Augmented BNF for Syntax Specifications: ABNF, Request for Comments: 5234, STD: 68) https://tools.ietf.org/html/rfc5234

(Internet Engineering Task Force, Tags for Identifying Languages, 2009. Tags for Identifying Languages, Request for Comments 5646, Best Current Practice 47) https://tools.ietf.org/html/rfc5646 https://tools.ietf.org/html/bcp47

Subtags

For the primary language subtag the registry selects the (ISO 639-1) two-character code where available above the three-character code (assigned by ISO 639-2, ISO 639-3, or ISO 639-5) .

```
en    English (language)
de    German (language)
fr    French (language)
```

"When languages have both an ISO 639-1 two-character code and a three- character code (assigned by ISO 639-2, ISO 639-3, or ISO 639-5), only the ISO 639-1 two-character code is defined in the IANA registry. ... users are directed in Internet applications to employ the alpha-3 code when an alpha-2 code for that language is not available."

(Internet Engineering Task Force, Tags for Identifying Languages, 2009. Tags for Identifying Languages, Request for Comments 5646, Best Current Practice 47) https://tools.ietf.org/html/rfc5646 https://tools.ietf.org/html/bcp47 http://www.loc.gov/standards/iso639-2/php/code_list.php (Non authoritative list of ISO 639-1 and ISO 639-2 language codes)

Region subtags are:

- Two-character alpha codes derived from ISO3166-1 "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes"; or

```
// ISO3166-1
AU      Australia
FR      France
```

(International Organization for Standardization (ISO), 2015. Country Codes - ISO 3166) *http://www.iso.org/iso/country_codes.htm*
(International Organization for Standardization (ISO), 2015. Country Codes - ISO 3166) *https://www.iso.org/obp/ui/#search/code/ ("Online Browsing Platform of Country Codes)*

- Three-character digit codes derived from UN M.49 "UN Standard Country or Area Codes for Statistical Use"

```
// UN M.49
001     World
021     North America
053     Australia and New Zealand
150     Europe
419     Latin America and the Caribbean
```

(UN Standard Country or Area Codes for Statistical Use is offline at this time, use (Internet Assigned Numbers Authority (IANA), 2015. Language Subtag Registrary) *http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry* *http://r12a.github.io/apps/subtags/ (unofficial practical lookup))*

Examples

Language tag examples

Language Tag	**Interpretation**
Language Subtag	
en	English (language)
de	German (language)
fr	French (language)
ja	Japanese (language)
Language-Region Subtags	
en-US	English (language), as used in United States (region)
en-GB	English (language), as used in United Kingdom (region)
en-AU	English (language), as used in Australia (region)
Language-Script Subtags	
zh-Hant	Chinese (language), written using Traditional Chinese (script)
zh-Hans	Chines (language), written using the Simplified Chinese (script)
sr-Cyrl	Serbian (language), written using the Cyrillic (script)

sr-Latn	Serbian (language), written using the Latin (script)
LanguageWithExtlang-Script-Region Subtags	
zh-cmn-Hans-CN	Chinese (language), Mandarin (extlang), Simplified script (script), as used in China (region)
Language-Script-Region Subtags	
cmn-Hans-CN	Mandarin Chinese (language), Simplified Chinese (script), as used in China (region).
LanguageWithExtlang-Region	
zh-yue-HK	Chinese (language), Cantonese (extlang), as used in Hong Kong (region)
Language-Region	
yue-HK	Cantonese Chines (language), as found in Hong Kong (region).

Derived from (Internet Engineering Task Force, Tags for Identifying Languages, 2009. Tags for Identifying Languages, Request for Comments 5646, Best Current Practice 47) https://tools.ietf.org/html/rfc5646 https://tools.ietf.org/html/bcp47#appendix-A

The Language Subtag Registry

"The Language Subtag Registry maintained by IANA ("the registry") is the source for valid subtags: other standards referenced in this section provide the source material for that registry."

"you used to find subtags by consulting the lists of codes in various ISO standards, but now you can find all subtags in one place." (W3C Language tags in HTML and XML, 2009. Language tags in HTML and XML) http://www.w3.org/International/articles/language-tags/

The Language Subtag Registry maintained by IANA ("the registry")	**(Internet Assigned Numbers Authority (IANA), 2015. Language Subtag Registrary)**	http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry
Unofficial Language Subtag Registry Practical lookup	(Richard Ishida, "r12a >> apps >> Language subtag lookup") As mentioned in (W3C Language tags in HTML and XML, 2009. Language tags in HTML and XML) http://www.w3.org/International/articles/language-tags/	http://r12a.github.io/apps/subtags/

(Internet Engineering Task Force, Tags for Identifying Languages, 2009. Tags for Identifying Languages, Request for Comments 5646, Best Current Practice 47) https://tools.ietf.org/html/rfc5646 https://tools.ietf.org/html/bcp47 (W3C Language tags in HTML and XML, 2009. Language tags in HTML and XML) http://www.w3.org/International/articles/language-tags/

Language Tag Case

Language tag case outside of this convention:

> ... case distinctions do not carry meaning in language tags ... The format of subtags in the registry is RECOMMENDED as the form to use in language tags. This format generally corresponds to the common conventions for the various ISO standards from which the subtags are derived.
>
> These conventions include:

- [ISO639-1] recommends that language codes be written in lowercase ('mn' Mongolian).
- [ISO15924] recommends that script codes use lowercase with the initial letter capitalized ('Cyrl' Cyrillic).
- [ISO3166-1] recommends that country codes be capitalized ('MN' Mongolia).

```
Language-Script-Region-Variant:

      hy-Latn-IT-arevela (Eastern Armenian written in Latin script, as
      used in Italy)

Language-Region:

      de-DE (German for Germany)
      en-US (English as used in the United States)
```

(Internet Engineering Task Force, Tags for Identifying Languages, 2009. Tags for Identifying Languages, Request for Comments 5646, Best Current Practice 47) https://tools.ietf.org/html/rfc5646 https://tools.ietf.org/html/bcp47

JLB: Language tag case within this convention: use all lower case.

```
Language-Script-Region-Variant:

      hy-latn-it-arevela (Eastern Armenian written in Latin script, as
      used in Italy)

Language-Region:

      de-de (German for Germany)
      en-us (English as used in the United States)
```

Lower case better conforms with the polyglot standard of generally using lower case for attribute values. See Case

(The Internet Engineering Task Force, 2009) https://tools.ietf.org/html/rfc5646 https://tools.ietf.org/html/bcp47 *(Source of the examples)*

File Extension, Mime Type, and HTML V XHTML Syntax

(JLB) Unless there is an overriding reason to do so:

- Use a file extension ".xhtml"; and
- (Thereby) serve your web pages with a mime type of "application/xhtml+xml"; and
- (Thereby) use the XHTML Syntax.

Markup errors will be better caught by enforcing the tighter coding standard (e.g. to have elements be closed or empty; and not rendering a page if it has minor errors).

IE 8 is the only browser that doesn't support application/xhtml+xml and, at the date of writing (2015-06-17), only comprises 2.19% of the browser share. So fuck IE 8.

http://caniuse.com/#feat=xhtml

To set the mime type for your web file use the following file extensions:

- ".html" for text/html;
- ".xhtml" for application/xhtml+xml;
- ... And verify your web server will serve those mime types, for those file extensions.

A web server usually determines the mime type of web file according to file extension used.

```
// For example in Apache 2.2. the mime type served is determined by
// "C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf\mime.types"
// With the default entries
// Mime Type                    File Extension.
...
text/html                 html htm
...
application/xhtml+xml     xhtml xht
...
application/xml           xml xsl
```

To check the Mime Type that is served, you can do any of the following (by serving a web page into a browser):

- In Firefox (38.0.5 or greater) then Context Menu > View Page Info > General [Tab] > Type ...
- In IE 10. F12 > Network [Tab] > Press Play > F5 [To reload] > Click on main web page ... [observe result in "Type" column"]
- In Firefox or Chrome. F12 > Network > Ctrl + F5 [Fetch page from server, ignore local cache] > Click on main web page > Observe Response Header, Content-Type field.
- JLB: Specify the Mime Type (aka "content type") that you expect the page to be served with, according to the file extension chosen, in comments: for the benefit of other coders. Do not use the legacy <meta http-equiv="Content-Type" ... />.

```
<!-- Do this -->
  <!--
    File Extension and MimeType.
    This file has an extension .xhtml in order to be served with a Mime Type (Content Type):
    application/xhtml+xml.  Under HTML 5 the only permitted value for content="",
    when <meta http-equiv="content-type" ...>, is "text/html". Therefore do not delete this
    comment to, instead, express mime type with
    <meta http-equiv="content-type" content="application/xhtml+xml" />
  -->

<!-- Not this -->
<meta http-equiv="content-type" content="application/xhtml+xml; charset=utf-8"/>

<!-- Nor this -->
<meta http-equiv="content-type" content="application/xhtml+xml" />
```

If you try to use the meta http-equiv way of expressing the Mime Type (Content Type) then the content string MUST start with the string literal "text/html" only. That's in the HTML 5 spec. https://validator.w3.org/nu/ will enforce that rule in the spec.

In addition, avoiding meta http-equiv="content-type" avoids the conflation of expressing the Mime Type and Character Encoding at the same time.

See (W3C HTML5, 2017. HTML 5.2 W3C Recommendation, 14 December 2017)
https://www.w3.org/TR/html5/document-metadata.html#pragma-directives , encoding declaration state

A browser processes a page by different syntax rules, given the page's mime type:

Mime Type	Syntax	Features
text/html	HTML Syntax	• Minor syntax errors ignored. • Namespaces not supported.
XML Mime Type: * "text/xml" * "application/xml" * [Any subtype ending with the four characters "+xml"], e.g. "**application/xhtml+xml**".	XHTML Syntax	• Minor syntax errors prevent document from being rendered fully. • Namespaces supported.

(W3C HTML5, 2017. HTML 5.2 W3C Recommendation, 14 December 2017)
http://www.w3.org/TR/html5/introduction.html#html-vs-xhtml

Character Encoding and References

Document encoding

See also …

(Bentley, n.d.), Web Code Convention - Character Sets and Encoding Concepts

Polyglot: save files as UTF-8 **without** a Byte Order Mark (BOM).

The Polyglot spec mentions that for "within the document" declarations of character encoding (signalling UTF-8) a combination of the following methods may be used:

- HTML encoding declaration (i.e. '<meta charset="utf-8" />') ; and
- By saving with file with a Byte Order Mark (BOM).

So saving with a BOM is permissible under the Polyglot spec.

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation)
http://www.w3.org/TR/html-polyglot/#character-encoding

However, saving with a BOM can cause problems. So we'll do without it and, instead, rely on: the HTML encoding declaration (i.e. '<meta charset="utf-8" />'); and the Server's HTTP Content-Type header.

(W3C 2013, "The Byte-Order Mark (BOM) in HTML", https://www.w3.org/International/questions/qa-byte-order-mark#problems) "Potential issues with the UTF-8 BOM", https://www.w3.org/International/questions/qa-byte-order-mark#problems

Polyglot: within the document head it is recommended you specify the character encoding (which must be UTF-8). JLB: Use <meta charset="utf-8"/>, do not use <meta http-equiv="content-type" ... />.

```
<!-- Do This -->
<meta charset="utf-8" />

<!-- Not this -->
<meta http-equiv="content-type" content="application/xhtml+xml; charset=utf-8"/>

<!-- Nor this -->
<meta http-equiv="content-type" content="application/xhtml+xml" />
```

jlb: The Polyglot spec specifies that you can express the charset, which must be utf-8 either with:
* <meta charset="UTF-8"/>; or
* <meta http-equiv="content-type" content="text/html; charset=utf-8"/>

However, if you intend on serving the page as application/xhtml+xml then the meta http-equiv="content-type" way of expressing the *Character Encoding* forbids beginning the content string with anything other than "text/html", under the HTML 5 spec. That is,

<meta http-equiv="content-type" content="application/xhtml+xml; charset=utf-8"/>

is illegal under the HTML 5 spec (and https://validator.w3.org/nu/ will correctly enforce this).

" The W3C Internationalization (i18n) Group recommends that one always include a visible encoding declaration in an HTML document, because it helps developers, testers, or translation production managers to check the encoding of a document visually." (W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation)

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) http://www.w3.org/TR/html-polyglot/#character-encoding
(W3C HTML5, 2017. HTML 5.2 W3C Recommendation, 14 December 2017) http://www.w3.org/TR/html5/document-metadata.html#other-metadata-names , encoding declaration state

Polyglot: When serving the web page you should:

- (jlb mandatory choice) For application/xhtml+xml served pages: don't worry about adding the charset to the the MIME/HTTP Content-Type header. UTF-8 will be the default encoding.

```
Content-type: application/xhtml+xml
```

- For text/html served pages: add the charset "charset=utf-8" to the MIME/HTTP Content-Type header.

```
Content-type: text/html; charset=utf-8
```

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) http://www.w3.org/TR/html-polyglot/#character-encoding

Literals V Character references: named or numbered.

A character can be expressed:

```
// Literally
&

// Or as a character reference.

  // Named character reference
  &amp;

  // Numeric character reference
  &#x00026;  // Hexadecimal
  &#38;      // Decimal
```

You must use character references for &, <, and >. When you do so, use a named character reference:

```
&amp;          ' &
&lt;           ' <
&gt;           ' >
```

... otherwise XML doesn't validate.

Single or double quotes can be literal.

```
'      // OK
"      // OK
```

Try to use literals rather than character references.

```
In UTF-8 use '€'      not '&x20AC;'
```

You only need to use character references for characters that are not part of the character encoding.

(Castro, 2003) p340

But if need to use a non literal character reference use one of the five named character references ...

```
&amp;          ' &
&lt;           ' <
&gt;           ' >
&quot;         ' "
&apos;         ' '
```

... or, for characters outside those five, use a numeric character reference in hexadecimal (not decimal).

```
// For
“                              // Validates in XHTML

// Do this
&#x0201C;                      // Validates in XHTML

// Not this
&ldquo; &OpenCurlyDoubleQuote;  // Chokes
&#8220;                        // Validates in XHTML, but discouraged.
```

> Polyglot markup uses only the following named entity references:
>
> amp lt gt apos quot
>
> For entities beyond the previous list, polyglot markup uses character references. For example, polyglot markup uses   instead of . Note that polyglot markup may use decimal values for escape characters (such as   in the previous example); however, the Character Model for the World Wide Web recommends that content SHOULD use the hexadecimal form of character escapes rather than the decimal form when both are available. [CHARMOD]

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation)
https://www.w3.org/TR/html-polyglot/#named-entity-references
(Bentley, n.d.) 2019-08-31

URL characters and encoding

Urls can be comprised of unreserved characters [A-Z, a-z, 0-9,-_.~] or reserved characters [!*'();:@&=+$,/?#[ ]].

https://en.wikipedia.org/wiki/Percent-encoding

You'll need to "percent-encode" a reserved character if it has a special meaning in a certain context.

https://en.wikipedia.org/wiki/Percent-encoding

Percent-encoding entails using a percentage, %, followed by the asci number in hexadecimal. E.g. '&' is encoded as '%26'.

Reserved characters after percent-encoding																	
!	#	$	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

https://en.wikipedia.org/wiki/Percent-encoding

A space is encoded as %20. [But are spaces allowed at all?).

Encode URL tools:

- Atom. Package "url-encode". cmd-u cmd-e to URL encode selected text. cmd-u cmd-d to URL decode selected text.
- PSPad

General Syntax Rules

Case

Polyglot: For element names, attribute names, and attribute values generally use lower case. There are exceptions for:

- SVG element names and attribute names.
- The MathML element name "definitionURL".
- Custom attribute values (e.g. for form name or id attribute values).

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) https://www.w3.org/TR/html-polyglot/#case-sensitivity

JLB: For attribute values holding form field names (in the name or id attribute) then, essentially, we use PascalCase.

```
<div>
  <label for="BirthDate">Birth Date</label>
  <input type="text" name="BirthDate" id="BirthDate" />
</dive>

// Do this
GivenName
TrackID
TVProgramFromAbc

// Not this
txtGivenName
Given_Name
given_name
Track_id
TVProgramFromABC
```

For details see (Bentley, 2018. Web Code Convention - Naming and Style - General, file://Atlas/Users/John/Documents/Sda/Info/Web/System/Conventions/Web Code Convention - Naming and Style - General.docx) file://Atlas/Users/John/Documents/Sda/Info/Web/System/Conventions/Web Code Convention - Naming and Style - General.docx , "Markup (xml or xhtml) form field names"

Make safe raw text in <script> and <style>

In HTML 5 Polyglot you need to make text in <script> and <style> safe. Do this either by:

1. Referencing an external resource:

   ```
   // Stylesheet external resource
   <link type="text/css" rel="stylesheet" href="../CssLibrary/simple.css" />

   // Script external resource
   <script src="quirksModeCheck.js"></script>
   ```

   (W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) http://www.w3.org/TR/html-polyglot/#safe-text-content

2. For inline text:
   a. Surround it with a CDATA section, commented out using the commenting syntax of the <script> or <style>. That is, use bookended /* ... */ comments.
   b. There is no spaces between the CDATA construct and the bookend comments.

c. Before the start CDATA section `<![CDATA[` and after the end CDATA section `]]>` there is only one node, a comment node (no other whitespace, including newlines)

```
' Do this
<style type="text/css">/*<![CDATA[*/
   h1 { color: blue; }
/*]]>*/</style>

<script>/*<![CDATA[*/
  function main () {
    document.getElementById("output").innerHTML = "Overwrite with new stuff";
  }
/*]]>*/</script>

' Not this
' Newlines
<script>
/*<![CDATA[*/
  foo
/*]]>*/
</script>

' Not this
' Spaces between CDATA construct and the bookedn comments
<script>/* <![CDATA[ */
  foo
/* ]]> */</script>
```

This is a simplified subset of all tha Polyglot allows.

You can also use "inline safe content" but those differences are hard to track.

> Before the CDATA section there can only be content that creates one node - preferably only one line of code - which may consist of whitespace, an XML comment, or a construct of the scripting/styling language (usually a comment of the scripting/styling language).
> After the CDATA section there can only be content that creates one node - preferably only one line of code - which may consist of whitespace, an XML comment, or a construct of the scripting/styling language (usually a comment of the scripting/styling language).

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) http://www.w3.org/TR/html-polyglot/#safe-CDATA-content

Comments inside style and script. Just use the language comment style, no longer any need to hide language comments in xml comments.

```
** Do this **
<style>/*<![CDATA[*/
  /* A comment */
   h1 { color: blue; }
/*]]>*/</style>

<script>/*<![CDATA[*/

  // This is nice
  function main () {
    document.getElementById("output").innerHTML = "Overwrite with new stuff";
  }
/*]]>*/</script>

** Not this (no longer needed)
<style>/*<![CDATA[*/
  <!--/* A comment */-->
   h1 { color: blue; }
/*]]>*/</style>

<script>/*<![CDATA[*/

  <!--// This is nice-->
```

```
  function main () {
    document.getElementById("output").innerHTML = "Overwrite with new stuff";
  }
/*]]*/</script>
```

Tested OK in:

- Firefox. 43.0.3
- Chrome 47.0...
- IE 11.20...

(Bentley, n.d.)

Self-closing tags and Void elements

> All elements listed as **void** in the HTML specification or in an extension spec, MUST in polyglot markup have the syntactic form of an XML empty-element tag (<foo/>) [Emphasis added]

Jlb: I've verified that
 will choke in firefox, when the document is served with a application/xhtml+xml mime type.

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation)
https://www.w3.org/TR/html-polyglot/#empty-elements
(Bentley, n.d.) 2016-03-21 07:33

A self-closing tag (a XML empty-element tag):

1. Is of the form `<xxx />`.
2. Represents a start tag (never an end tag).
3. Applies always to Void Elements: `area`, `base`, **`br`**, `col`, `embed`, `hr`, **`img`**, **`input`**, `keygen`, **`link`**, `menuitem`, **`meta`**, `param`, `source`, `track`, **`wbr`**.
4. Applies to Foreign Elements (MathML and SVG elements) when they have no content.
5. Does not apply to Raw Text Elements (`<script>`, `<style>`), nor Escapable Raw Text Elements (`<textarea>`, `<title>`). Instead these elements must have a start and end tag.

"Void elements only have a start tag". (W3C, 2016. HTML 5.1, W3C Working Draft, https://www.w3.org/TR/html51/)

"Foreign elements must either have a start tag and an end tag, or a start tag that is marked as self-closing". (W3C, 2016. HTML 5.1, W3C Working Draft, https://www.w3.org/TR/html51/)

"if the element is one of the void elements, or if the element is a foreign element, then there may be a single U+002F SOLIDUS character (/). This character has no effect on void elements, but on foreign elements it marks the start tag as self-closing." (W3C, 2016. HTML 5.1, W3C Working Draft, https://www.w3.org/TR/html51/)"

(W3C, 2016. HTML 5.1, W3C Working Draft, https://www.w3.org/TR/html51/)
https://www.w3.org/TR/html51/syntax.html#writing-html-documents-elements

JLB: A self-closing tag must have a space before the forward slash "/".

HTML merely permits a space ...

"After the attributes, or after the tag name if there are no attributes, there **may** be one or more space characters."

... and modern browsers probably don't choke if there is no space. But we'll insert a space for any lurking backwards compatibility reasons.

(W3C, 2016. HTML 5.1, W3C Working Draft, https://www.w3.org/TR/html51/)

Attribute Datatypes

Attribute quotes

> Polyglot markup surrounds all attribute values with quotations marks.

```
<link rel='stylesheet' href='simple.css' />
```

Attribute values in XML require quotes.

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation) (Bentley, 2015. XML Quick Reference)

> Polyglot markup surrounds attribute values with either single quotation marks or with double quotation marks.

```
<link rel='stylesheet' href='simple.css' />

<link rel="stylesheet" href="simple.css" />
```

(W3C Polyglot, 2015. Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation, p.Section 4.8 Attributes)

Valueless attributes

JLB: If an attribute exists it must have a value (this can be an empty string, but this is discouraged).

```
<!-- Do this -->
<input type='checkbox' checked='checked' />

<!-- Discouraged but permitted -->
<input type='checkbox' checked='' />

<!-- Not this -->
<input type='checkbox' checked />
```

A well-formed XML document requires an attribute value (an empty string counts as a value), if the attribute exists. Avoid using an empty string because it's not very semantic.

(Bentley, 2015. XML Quick Reference)

Booleans

Represent boolean attributes:

- True: repeat the attribute name (or use an empty string). Don't use 'true'.
- False: omit the attribute entirely. Don't use 'false'.

```
<!-- True -->
<input type='checkbox' checked='checked' />
<input type='checkbox' checked=''/> <!-- discouraged by JLB -->
```

```
<!-- False -->
<input type='checkbox' checked='checked' />

<!-- Invalid... -->
<input type='checkbox' checked='true' />
<input type='checkbox' checked='false' />
```

(W3C HTML5, 2017. HTML 5.2 W3C Recommendation, 14 December 2017)
https://www.w3.org/TR/html/infrastructure.html#boolean-attributes

Abbreviations

JLB
jlb

Stipulations of a John Bentley convention (above what Polyglot markup or the HTML spec provides).

References

Bentley, J., (2015-Jun-09). *XML Quick Reference*. [Electronic Source].

Bentley, J., (2018). *Web Code Convention - Naming and Style - General*. [Electronic Source]
Available at: file://Atlas/Users/John/Documents/Sda/Info/Web/System/Conventions/Web Code Convention - Naming and Style - General.docx.

Bentley, J., n.d. *Experimentally Verified.*

Bentley, J., n.d. *Web Code Convention - Character Sets and Encoding Concepts*. Available at:
file://Atlas/Users/John/Documents/Sda/Info/Web/System/Conventions/Web Code Convention - Character Sets and Encoding Concepts.docx.

Castro, E., 2003. *HTML for the world wide web with XHTML and CSS*. 5th ed. Peachpit Press.

International Organization for Standardization (ISO), (2015-Jun-21). *Country Codes - ISO 3166*. [Online]
Available at: http://www.iso.org/iso/country_codes.htm
[Accessed 205-Jun-21].

Internet Assigned Numbers Authority (IANA), (2015-Jun-08). *Language Subtag Registrary*. [Online]
Available at: http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry.

Internet Engineering Task Force (IETF), (2008-Jan). *Augmented BNF for Syntax Specifications: ABNF, Request for Comments: 5234, STD: 68*. [Online]
Available at: https://tools.ietf.org/html/rfc5234.

Internet Engineering Task Force, Tags for Identifying Languages, (2009-Sep). *Tags for Identifying Languages, Request for Comments 5646, Best Current Practice 47*. [Online]
Available at: https://tools.ietf.org/html/rfc5646
[Accessed 2015-Jun-20].

Mozilla Developer Network, (2014-Dec-18). *Quirks Mode and Standards Mode*. [Online]
Available at: https://developer.mozilla.org/en-US/docs/Quirks_Mode_and_Standards_Mode
[Accessed 2015-Jun-19].

W3C HTML5, (2017-Dec-15). *HTML 5.2 W3C Recommendation, 14 December 2017*. [Online]
Available at: https://www.w3.org/TR/html/.

W3C Language tags in HTML and XML, (2009-Sep-09). *Language tags in HTML and XML*. [Online]
Available at: http://www.w3.org/International/articles/language-tags/
[Accessed 2015-Jun-21].

W3C Polyglot, (2015-Sep-29). *Polyglot Markup: A robust profile of the HTML5 vocabulary, W3C Candidate Recommendation*. [Online]
Available at: http://www.w3.org/TR/html-polyglot/
[Accessed 2016-Mar-17].

W3C, (2015-Apr-28). *W3C DOM4, W3C Last Call Working Draft 28 April 2015*. [Online]
Available at: http://www.w3.org/TR/dom/.

W3C, (2016-Mar-10). *HTML 5.1, W3C Working Draft*. [Electronic Source]
Available at: https://www.w3.org/TR/html51/.

Document Licence