Namespace

An XML *namespace* is a collection of element and attribute names as defined by the DTD of a particular markup language. In XML documents, you must explicitly identify the namespace so the client (in this case, the browser) knows that you intend the `q` element in your document to be a "quote" and not a "question" from some other (theoretical) XML language for exams.

The namespace is specified using the `xmlns` attribute in the `html` root element. The value is the location of an online documentation of that namespace. The namespace identifier for XHTML 1.0 and 1.1 is `xmlns="http://www.w3.org/1999/xhtml"`.

Language

Because this Web of ours is "World Wide," the HTML specifications take into account that documents are published in a variety of languages. For that reason, it is important to identify the language in which the document is written (as in `lang="en"`) and the language of the XML version (as in `xml:lang="en"`). These attributes are placed in the `html` root element along with the namespace identifier. The XHTML 1.0 Recommendation suggests you include both attributes in the interest of backward compatibility. See Chapter 6 for a complete list of two-letter language codes.

The Document Header

The header provides a place to include important information about the document to users, browsers, and search engines. It is also a common place to stow scripts and embedded style sheets. This section looks at the `head` element and the elements it contains.

head

```
<head>...</head>
```

Attributes

Internationalization attributes: `lang`, `xml:lang`, `dir`
`id="text"` *(XHTML only)*
`profile="URLS"`

Every `head` element must include a `title` element that provides a description of the document. The `head` element may also include any of the following elements in any order: `script`, `style`, `meta`, `link`, `object`, `isindex`, and `base`. The `head` element merely acts as a container of these elements and does not have any content of its own.

It is recommended that HTML documents (and XHTML documents without an XML declaration) also include a `meta` element that specifies the content type and character encoding for the document, although this element is not required. The `meta` element is discussed in the upcoming "Providing Meta Data section.

Titles

The most important (and only required) element within the header is the document `title`, which provides a description of the page's contents.

title

`<title>...</title>`
This element is required.

Attributes

Internationalization: `lang`, `xml:lang`, `dir`

Starting in HTML 4.01, the `title` element is required, which means that every HTML document must have a meaningful title in its header in order to be valid. The title is typically displayed in the top bar of the browser, outside the regular content window.

Titles should contain only ASCII characters (letters, numbers, and basic punctuation). Special characters (such as `&`) should be referred to by their character entities within the title, for example:

```
<title>The Adventures of Peto &amp;amp; Fleck</title>
```

The title is what is displayed in a user's bookmarks or favorites list. Descriptive titles are also a key tool for improving accessibility, as they are the first thing a person hears when using a screen reader. Search engines rely heavily on document titles as well. For these reasons, it's important to provide thoughtful and descriptive titles for all your documents and avoid vague titles, such as "Welcome" or "My Page." You may also want to keep the length of your titles in check so they are able to display in the browser's title area.

Other Header Elements

Other useful HTML elements also placed within `head` of the document include:

`base`
: This element establishes the document's base location, which serves as a reference for all pathnames and links in the document. For more information, see Chapter 11.

`isindex`
: *Deprecated.* This element was once used to add a simple search function to a page. It has been deprecated by HTML 4.01 in favor of form inputs.

`link`
: This element defines the relationship between the current document and another document. Although it can signify such relationships as index, next, and previous, it is most often used to link a document to an external style sheet (see Chapter 16).

`script`
: JavaScript and VBScript code may be added to the document within its header using this element. For examples of using the `script` element, see Chapter 27.

`style`
: One method for attaching a style sheet to an HTML document is to embed it in the head of the document with the `style` element. For more information, see Chapter 16.

`meta`
: The `meta` element is used to provide information about a document, such as keywords or descriptions to aid search engines. It is discussed in detail in the next section.

Providing Meta Data

The `meta` element has a wide variety of applications but is primarily used to include information about a document, such as the character encoding, creation date, author, or copyright information.

meta

```
<meta />
```

Attributes

Internationalization: `lang`, `xml:lang`, `dir`
`id="`*`text`*`"` *(XHTML only)*
`content="`*`text`*`"` *(Required)*
`http-equiv="`*`text`*`"`
`name="`*`text`*`"`
`scheme="`*`text`*`"`

The data included in a `meta` element is useful for servers, web browsers, and search engines but is invisible to the reader. It must always be placed within the `head` of the document.

A document may have any number of `meta` elements. There are two types of `meta` elements, using either the `name` or `http-equiv` attribute. In each case, the `content` attribute is necessary to provide a value (or values) for the named information or function. These examples show the syntax of both `meta` types:

```
<meta http-equiv="name" content="content" />
<meta name="name" content="content" />
```

http-equiv

Information provided by an `http-equiv` attribute is processed as though it had come from an HTTP response header. HTTP headers contain information the server passes to the browser just before it sends the HTML document, such as media type information and other values that affect the action of the browser. Therefore, the `http-equiv` attribute provides information that will, depending on the description, affect the way the browser handles your document.

An *HTTP header* is not the same as the header indicated by the head element within the HTML document. HTTP headers exist outside the HTML text document and are tacked on by the server. When a document is requested via an HTTP request (that is, via the Web), the HTTP header goes along for the ride to give the browser a heads-up on what kind of document to expect. Its contents are not displayed.

There is a large number of predefined http-equiv types available. This chapter introduces just a few of the most useful. For a complete listing, see the Dictionary of HTML META Tags at *vancouver-webpages.com/META*.

name

The name attribute is used to insert hidden information about the document that does not correspond to HTTP headers. For example:

```
<meta name="author" content="Jennifer Niederst Robbins" />
<meta name="copyright" content="2006, O'Reilly Media" />
```

You can make up your own names or use one of the names put forth by search engine and browser companies for standardized use. A few of the accepted and more useful meta names are discussed later in this section.

Identifying media type and character encoding

It is recommended (although not required) that the media type and character encoding be specified within (X)HTML documents as a way to keep that information with the document. (For more information on declaring character encodings, see Chapter 6.)

This is done using the meta element, as shown in this example:

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
```

The parts are broken down as follows:

Media type identification

The media type, very similar to MIME types used for sending email attachments, is another bit of information sent in the HTTP header. For HTML documents, the media type is always text/html. That one is easy. XHTML documents, on the other hand, are not as straightforward.

XHTML 1.0 documents may be served as either XML or HTML documents. Although XML is the proper method, due to lack of browser support for XML files, many authors choose to deliver XHTML 1.0 files with the text/html MIME type used for HTML documents. When XHTML documents are served in this manner, they may not be parsed as XML documents.

XHTML 1.0 files may also be served as XML using the MIME types application/xhtml+xml, application/xml, or text/xml. The W3C recommends that you use application/xhtml+xml only.

XHTML 1.1 documents are *not* permitted to use the text/html media type. This poses a problem because some browsers do not know what to do with

the non-text/html media types. This is another reason why XHTML 1.1 is still difficult to implement properly and why developers generally opt for the XHTML 1.0 standard.

For more information on the W3C's recommended media types, see *www.w3.org/TR/xhtml-media-types/*.

Character encoding

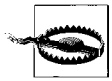
It is recommended, although not required, that the character encoding be specified for all documents. The character encoding describes the set of actual glyphs, or character shapes, that your document uses. Sets of characters are standardized: you can refer to them by identifying standard numbers. For example, in documents written in English, the most common character encoding is ISO-8859-1, which consists of all the characters in Western European languages. For more information on character encoding, see Chapter 6.

Character encoding should be set on the server as part of the HTTP header. It may also be set in the XML declaration for XHTML documents that use a character encoding other than the XML standard UTF-8 or UTF-16. For instances when it is necessary to override or guarantee the server setting (and if the XML declaration is not used), the character encoding may be provided with a meta element, as shown in the example.

For more information on the preferred methods of specifying character encoding, see Chapter 6 and the Web Standards Project article at *www.webstandards.org/learn/askw3c/dec2002.html*.

Using the meta element for client-pull

Client-pull refers to the ability of the browser (the client) to automatically request (pull) a new document from the server. The effect for the user is that the page displays, and after a period of time, automatically refreshes with new information or is replaced by an entirely new page. This technique can be used to automatically redirect readers to a new URL (for instance, if an old URL has been retired).



Be aware, however, that the W3C strongly discourages the use of this method for automatic forwarding in favor of server-side redirects for reasons of accessibility.

Client-pull uses the refresh attribute value, first introduced by Netscape. It tells the browser to wait a specified number of seconds (indicated by an integer in the content attribute) and then to load a new page. If no page is specified, the browser just reloads the current page. The following example instructs the browser to reload the page after 15 seconds (assume there's something fancy happening on the server side that puts updated information in the HTML document):

```
<meta http-equiv="refresh" content="15" />
```

To reload a different file, provide the URL for the document within the content attribute:

```
<meta http-equiv="refresh" content="1; url=http://doc2.html" />
```

Note that there is only a single set of quotation marks around the value for content. Although URLs usually require their own quotation marks, these are omitted within the context of the content attribute.

Other uses of http-equiv

Here are some other uses of the http-equiv attribute:

expires
: Indicates the date and time after which the document should be considered expired. Web robots may use this information to delete expired documents from a search engine index. The date and time format (as shown below) follows the date/time standard for HTTP headers because the http-equiv attribute is intended to mimic an HTTP header field.

```
<meta http-equiv="expires" content="Wed 12 Jun 2001 10:52:00 EST" />
```

content-language
: This may be used to identify the language in which the document is written. The browser can send a corresponding Accept-Language header, which causes the server to choose the document with the appropriate language.

  This example tells the browser that the document's natural language is French:

```
<meta http-equiv="content-language" content="fr" />
```

  The W3C now recommends using the lang and xml:lang attributes in the html element for language specification, but this method may be used for backward compatibility. For more information on internationalization and a listing of two-letter language codes, see Chapter 6.

meta names for search engines

Search engines introduced several meta names that aid their search engines in finding pages. Note that not all search engines use meta-data, but adding them to your document won't hurt. There is a blurry distinction between name and http-equiv in this case, so most of these meta names also work as http-equiv definitions.

description
: This provides a brief, plain-language description of the contents of your web page, which is particularly useful if your document contains little text, is a frameset, or has extensive scripts at the top of the HTML document. Search engines that recognize the description may display it in the search results page. Some search engines use only the first 20 words of descriptions, so get to the point quickly.

```
<meta name="description" content="Jennifer Robbins' resume
    and web design samples" />
```

keywords
: You can supplement the title and description of the document with a list of comma-separated keywords that would be useful in indexing your document. Note: Search engines have largely abandoned meta keywords in