XHTML - QUICK GUIDE

XHTML - INTRODUCTION

XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage. It is the next step in the evolution of the internet. The XHTML 1.0 is the first document type in the XHTML family.

XHTML is almost identical to HTML 4.01 with only few differences. This is a cleaner and stricter version of HTML 4.01. If you already know HTML, then you need to give little attention to learn this latest version of HTML.

XHTML was developed by World Wide Web Consortium *W3C* to help web developers make the transition from HTML to XML. By migrating to XHTML today, web developers can enter the XML world with all of its benefits, while still remaining confident in the backward and future compatibility of the content.

Why Use XHTML?

Developers who migrate their content to XHTML 1.0 get the following benefits −

- XHTML documents are XML conforming as they are readily viewed, edited, and validated with standard XML tools.
- XHTML documents can be written to operate better than they did before in existing browsers as well as in new browsers.
- XHTML documents can utilize applications such as scripts and applets that rely upon either the HTML Document Object Model or the XML Document Object Model.
- XHTML gives you a more consistent, well-structured format so that your webpages can be easily parsed and processed by present and future web browsers.
- You can easily maintain, edit, convert and format your document in the long run.
- Since XHTML is an official standard of the W3C, your website becomes more compatible with many browsers and it is rendered more accurately.
- XHTML combines strength of HTML and XML. Also, XHTML pages can be rendered by all XML enabled browsers.
- XHTML defines quality standard for your webpages and if you follow that, then your web pages are counted as quality web pages. The W3C certifies those pages with their quality stamp.

Web developers and web browser designers are constantly discovering new ways to express their ideas through new markup languages. In XML, it is relatively easy to introduce new elements or additional element attributes. The XHTML family is designed to accommodate these extensions through XHTML modules and techniques for developing new XHTML-conforming modules. These modules permit the combination of existing and new features at the time of developing content and designing new user agents.

Basic Understanding

Before we proceed further, let us have a quick view on what are HTML, XML, and SGML.

What is HTML?

This is **S**tandard **G**eneralized **M**arkup **L**anguage *SGML* application conforming to International Standard ISO 8879. HTML is widely regarded as the standard publishing language of the World Wide Web.

What is SGML?

This is a language for describing markup languages, particularly those used in electronic document exchange, document management, and document publishing. HTML is an example of a language defined in SGML.

What is XML?

XML stands for E**X**tensible **M**arkup **L**anguage. XML is a markup language much like HTML and it was designed to describe data. XML tags are not predefined. You must define your own tags according to your needs.

XHTML - SYNTAX

XHTML syntax is very similar to HTML syntax and almost all the valid HTML elements are valid in XHTML as well. But when you write an XHTML document, you need to pay a bit extra attention to make your HTML document compliant to XHTML.

Here are the important points to remember while writing a new XHTML document or converting existing HTML document into XHTML document −

- Write a DOCTYPE declaration at the start of the XHTML document.
- Write all XHTML tags and attributes in lower case only.
- Close all XHTML tags properly.
- Nest all the tags properly.
- Quote all the attribute values.
- Forbid Attribute minimization.
- Replace the **name** attribute with the **id** attribute.
- Deprecate the **language** attribute of the script tag.

Here is the detail explanation of the above XHTML rules −

DOCTYPE Declaration

All XHTML documents must have a DOCTYPE declaration at the start. There are three types of DOCTYPE declarations, which are discussed in detail in XHTML Doctypes chapter. Here is an example of using DOCTYPE −

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Case Sensitivity

XHTML is case sensitive markup language. All the XHTML tags and attributes need to be written in lower case only.

```
<!-- This is invalid in XHTML -->
<A Href="/xhtml/xhtml_tutorial.html">XHTML Tutorial</A>

<!-- Correct XHTML way of writing this is as follows -->
<a href="/xhtml/xhtml_tutorial.html">XHTML Tutorial</a>
```

In the example, **Href** and anchor tag **A** are not in lower case, so it is incorrect.

Closing the Tags

Each and every XHTML tag should have an equivalent closing tag, even empty elements should also have closing tags. Here is an example showing valid and invalid ways of using tags −

```
<!-- This is invalid in XHTML -->
<p>This paragraph is not written according to XHTML syntax.

<!-- This is also invalid in XHTML -->
<img src="/images/xhtml.gif" >
```

The following syntax shows the correct way of writing above tags in XHTML. Difference is that, here we have closed both the tags properly.

```
<!-- This is valid in XHTML -->
<p>This paragraph is not written according to XHTML syntax.</p>

<!-- This is also valid now -->
<img src="/images/xhtml.gif" />
```

Attribute Quotes

All the values of XHTML attributes must be quoted. Otherwise, your XHTML document is assumed as an invalid document. Here is the example showing syntax −

```
<!-- This is invalid in XHTML -->
<img src="/images/xhtml.gif" width=250 height=50 />

<!-- Correct XHTML way of writing this is as follows -->
<img src="/images/xhtml.gif" width="250" height="50" />
```

Attribute Minimization

XHTML does not allow attribute minimization. It means you need to explicitly state the attribute and its value. The following example shows the difference −

```
<!-- This is invalid in XHTML -->
<option selected>

<!-- Correct XHTML way of writing this is as follows -->
<option selected="selected">
```

Here is a list of the minimized attributes in HTML and the way you need to write them in XHTML −

HTML Style	XHTML Style
compact	compact="compact"
checked	checked="checked"
declare	declare="declare"
readonly	readonly="readonly"
disabled	disabled="disabled"
selected	selected="selected"
defer	defer="defer"
ismap	ismap="ismap"
nohref	nohref="nohref"
noshade	noshade="noshade"
nowrap	nowrap="nowrap"
multiple	multiple="multiple"

noresize	noresize="noresize"

The *id* Attribute

The id attribute replaces the name attribute. Instead of using name = "name", XHTML prefers to use id = "id". The following example shows how −

```
<!-- This is invalid in XHTML -->
<img src="/images/xhtml.gif" name="xhtml_logo" />

<!-- Correct XHTML way of writing this is as follows -->
<img src="/images/xhtml.gif"  />
```

The *language* Attribute

The language attribute of the script tag is deprecated. The following example shows this difference −

```
<!-- This is invalid in XHTML -->

<script language="JavaScript" type="text/JavaScript">
   document.write("Hello XHTML!");
</script>

<!-- Correct XHTML way of writing this is as follows -->

<script type="text/JavaScript">
   document.write("Hello XHTML!");
</script>
```

Nested Tags

You must nest all the XHTML tags properly. Otherwise your document is assumed as an incorrect XHTML document. The following example shows the syntax −

```
<!-- This is invalid in XHTML -->
<b><i> This text is bold and italic</b></i>

<!-- Correct XHTML way of writing this is as follows -->
<b><i> This text is bold and italic</i></b>
```

Element Prohibitions

The following elements are not allowed to have any other element inside them. This prohibition applies to all depths of nesting. Means, it includes all the descending elements.

Element	Prohibition
<a>	Must not contain other <a> elements.
<pre>	Must not contain the <img>, <object>, <big>, <small>, <sub>, or <sup> elements.
<button>	Must not contain the <input>, <select>, <textarea>, <label>, <button>, <form>, <fieldset>, <iframe> or <isindex> elements.
<label>	Must not contain other <label> elements.
<form>	Must not contain other <form> elements.

A Minimal XHTML Document

The following example shows you a minimum content of an XHTML 1.0 document −

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/TR/xhtml1" xml:lang="en" lang="en">
   <head>
      <title>Every document must have a title</title>
   </head>

   <body>
      ...your content goes here...
   </body>
</html>
```

HTML VERSUS XHTML

Due to the fact that XHTML is an XML application, certain practices that were perfectly legal in SGML-based HTML 4 must be changed. You already have seen XHTML syntax in previous chapter, so differences between XHTML and HTML are very obvious. Following is the comparison between XHTML and HTML.

XHTML Documents Must be Well-Formed

Well-formedness is a new concept introduced by XML. Essentially, this means all the elements must have closing tags and you must nest them properly.

CORRECT: Nested Elements

```
<p>Here is an emphasized <em>paragraph</em>.</p>
```

INCORRECT: Overlapping Elements

```
<p>Here is an emphasized <em>paragraph.</p></em>
```

Elements and Attributes Must be in Lower Case

XHTML documents must use lower case for all HTML elements and attribute names. This difference is necessary because XHTML document is assumed to be an XML document and XML is case-sensitive. For example, <li> and <LI> are different tags.

End Tags are Required for all Elements

In HTML, certain elements are permitted to omit the end tag. But XML does not allow end tags to be omitted.

CORRECT: Terminated Elements

```
<p>Here is a paragraph.</p><p>here is another paragraph.</p>
<br><hr/>
```

INCORRECT: Unterminated Elements

```
<p>Here is a paragraph.<p>here is another paragraph.
<br><hr>
```

Attribute Values Must Always be Quoted

All attribute values including numeric values, must be quoted.

CORRECT: Quoted Attribute Values

```
<td rowspan="3">
```

INCORRECT: Unquoted Attribute Values

```
<td rowspan=3>
```

Attribute Minimization

XML does not support attribute minimization. Attribute-value pairs must be written in full. Attribute names such as compact and checked cannot occur in elements without their value being specified.

CORRECT: Non Minimized Attributes

```
<dl compact="compact">
```

INCORRECT: Minimized Attributes

```
<dl compact>
```

Whitespace Handling in Attribute Values

When a browser processes attributes, it does the following −

- Strips leading and trailing whitespace.
- Maps sequences of one or more white space characters *includinglinebreaks* to a single inter-word space.

Script and Style Elements

In XHTML, the script and style elements should not have “<” and “&” characters directly, if they exist; then they are treated as the start of markup. The entities such as “<” and “&” are recognized as entity references by the XML processor for displaying “<” and “&” characters respectively.

Wrapping the content of the script or style element within a CDATA marked section avoids the expansion of these entities.

```
<script type="text/JavaScript">
   <![CDATA[
      ... unescaped VB or Java Script here... ...
   ]]>
</script>
```

An alternative is to use external script and style documents.

The Elements with *id* and *name* Attributes

XHTML recommends the replacement of *name* attribute with *id* attribute. Note that in XHTML 1.0, the *name* attribute of these elements is formally deprecated, and it will be removed in a subsequent versions of XHTML.

Attributes with Pre-defined Value Sets

HTML and XHTML both have some attributes that have pre-defined and limited sets of values. For example, ***type*** attribute of the ***input*** element. In HTML and XML, these are called **enumerated attributes**. Under HTML 4, the interpretation of these values was case-insensitive, so a value of **TEXT** was equivalent to a value of **text**.

Under XHTML, the interpretation of these values is case-sensitive so all of these values are defined in lower-case.

Entity References as Hex Values

HTML and XML both permit references to characters by using hexadecimal value. In HTML these references could be made using either **&#Xnn;** or **&#xnn;** and they are valid but in XHTML documents, you must use the lower-case version only such as **&#xnn;**.

The <html> Element is a Must

All XHTML elements must be nested within the <html> root element. All other elements can have sub elements which must be in pairs and correctly nested within their parent element. The basic document structure is −

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

<html>
   <head> ... </head>
   <body> ... </body>
</html>
```

XHTML - DOCTYPES

The XHTML standard defines three Document Type Definitions *DTDs*. The most commonly used and easy one is the XHTML Transitional document.

XHTML 1.0 document type definitions correspond to three DTDs −

- Strict
- Transitional
- Frameset

There are few XHTML elements and attributes, which are available in one DTD but not available in another DTD. Therefore, while writing your XHTML document, you must select your XHTML elements or attributes carefully. However, XHTML validator helps you to identify valid and invalid elements and attributes.

Please check XHTML Validations for more detail on this.

XHTML 1.0 Strict

If you are planning to use Cascading Style Sheet *CSS* strictly and avoiding to write most of the XHTML attributes, then it is recommended to use this DTD. A document conforming to this DTD is of the best quality.

If you want to use XHTML 1.0 Strict DTD then you need to include the following line at the top of your XHTML document.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional

If you are planning to use many XHTML attributes as well as few Cascading Style Sheet properties, then you should adopt this DTD and you should write your XHTML document accordingly.

If you want to use XHTML 1.0 Transitional DTD, then you need to include the following line at the top of your XHTML document.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset

You can use this when you want to use HTML Frames to partition the browser window into two or

more frames.

If you want to use XHTML 1.0 Frameset DTD, then you need to include following line at the top of your XHTML document.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Note − No matter what DTD you are using to write your XHTML document; if it is a valid XHTML document, then your document is considered as a good quality document.

XHTML - ATTRIBUTES

There are a few XHTML/HTML attributes which are standard and associated to all the XHTML/HTML tags. These attributes are listed here with brief description −

Core Attributes

Not valid in base, head, html, meta, param, script, style, and title elements.

Attribute	Value	Description
class	class_rule or style_rule	The class of the element.
Id	id_name	A unique id for the element.
style	style_definition	An inline style definition.
Title	tooltip_text	A text to display in a mouse tip.

Language Attributes

The ***lang*** attribute indicates the language being used for the enclosed content. The language is identified using the ISO standard language abbreviations, such as **fr** for French, **en** for English, and so on. More codes and their formats are described at www.ietf.org.

Not valid in base, br, frame, frameset, hr, iframe, param, and script elements.

Attribute	Value	Description
dir	ltr | rtl	Sets the text direction.
lang	language_code	Sets the language code.

Microsoft Proprietary Attributes

Microsoft introduced a number of new proprietary attributes with the Internet Explorer 4 and higher versions.

Attribute	Value	Description
accesskey	character	Sets a keyboard shortcut to access an element.
language	string	This attribute specifies the scripting language to be used with an associated script bound to the element, typically through an event handler attribute. Possible values might include JavaScript, jScript, VBS, and VBScript.
tabindex	number	Sets the tab order of an element.

contenteditable	boolean	Allows users to edit content rendered in Internet Explorer 5.5 or greater. Possible values are true or false.
disabled	boolean	Elements with the disabled attribute set may appear faded and will not respond to user input. Possible values are true or false.
hidefocus	on or off	This proprietary attribute, introduced with Internet Explorer 5.5, hides focus on an element's content. Focus must be applied to the element using the tabindex attribute.
unselectable	on or off	Used to prevent content displayed in Internet Explorer 5.5 from being selected.

XHTML - EVENTS

When users visit a website, they do things such as click on text, images and hyperlinks, hover-over things, etc. These are examples of what JavaScript calls events.

We can write our event handlers in JavaScript or VBScript and can specify these event handlers as a value of event tag attribute. The XHTML 1.0 has a similar set of events which is available in HTML 4.01 specification.

The <body> and <frameset> Level Events

There are only two attributes which can be used to trigger any JavaScript or VBScript code, when any event occurs at document level.

Attribute	Value	Description
onload	Script	Script runs when a XHTML document loads.
onunload	Script	Script runs when a XHTML document unloads.

Note − Here, the script refers to any function or piece of code of VBScript or JavaScript.

The <form> Level Events

There are following six attributes which can be used to trigger any JavaScript or VBScript code when any event occurs at form level.

Attribute	Value	Description
onchange	Script	Script executes when the element changes.
onsubmit	Script	Script executes when the form is submitted.
onreset	Script	Script executes when the form is reset.
onselect	Script	Script executes when the element is selected.
onblur	Script	Script executes when the element loses focus.
onfocus	Script	Script runs when the element gets focus.

Keyboard Events

The following three events are generated by keyboard. These events are not valid in base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, and title elements.

Attribute	Value	Description
onkeydown	Script	Script executes on key press.
onkeypress	Script	Script executes on key press and release.
onkeyup	Script	Script executes key release.

Other Events

The following seven events are generated by mouse when it comes in contact with any HTML tag. These events are not valid in base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, and title elements.

Attribute	Value	Description
onclick	Script	Script executes on a mouse click.
ondblclick	Script	Script executes on a mouse double-click.
onmousedown	Script	Script executes when mouse button is pressed.
onmousemove	Script	Script executes when mouse pointer moves.
onmouseout	Script	Script executes when mouse pointer moves out of an element.
onmouseover	Script	Script executes when mouse pointer moves over an element.
onmouseup	Script	Script executes when mouse button is released.

XHTML - VERSION 1.1

The W3C has helped move the internet content-development community from the days of malformed, non-standard mark-up into the well-formed, valid world of XML. In XHTML 1.0, this move was moderated by the goal of providing easy migration of existing HTML 4 *orearlier* based content to XHTML and XML.

The W3C has removed support for deprecated elements and attributes from the XHTML family. These elements and attributes had largely presentation-oriented functionality that is better handled via style sheets or client-specific default behavior.

Now the W3C's HTML Working Group has defined an initial document type based solely upon modules which are XHTML 1.1. This document type is designed to be portable to a broad collection of client devices, and applicable to the majority of internet content.

Document Conformance

The XHTML 1.1 provides a definition of strictly conforming XHTML documents which MUST meet all the following criteria −

- The document MUST conform to the constraints expressed in XHTML 1.1 Document Type Definition.
- The root element of the document MUST be <html>.
- The root element of the document MUST designate the XHTML namespace using the *xmlns* attribute.
- The root element MAY also contain a schema location attribute as defined in the XML Schema.

There MUST be a DOCTYPE declaration in the document prior to the root element. If it is present,

the public identifier included in the DOCTYPE declaration MUST refer the DTD found in XHTML 1.1 Document Type Definition.

Here is an example of an XHTML 1.1 document −

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
   xsi:schemaLocation="http://www.w3.org/MarkUp/SCHEMA/xhtml11.xsd" xml:lang="en">

   <head>
      <title>This is the document title</title>
   </head>

   <body>
      <p>Moved to <a href="http://example.org/">example.org</a>.</p>
   </body>

</html>
```

Note − In this example, the XML declaration is included. An XML declaration such as the one above is not required in all XML documents. XHTML document authors are strongly encouraged to use XML declarations in all their documents. Such a declaration is required when the character encoding of the document is other than the default UTF-8 or UTF-16.

XHTML 1.1 Modules

The XHTML 1.1 document type is made up of the following XHTML modules.

Structure Module − The Structure Module defines the major structural elements for XHTML. These elements effectively act as the basis for the content model of many XHTML family document types. The elements and attributes included in this module are − body, head, html, and title.

Text Module − This module defines all of the basic text container elements, attributes, and their content model − abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, and var.

Hypertext Module − The Hypertext Module provides the element that is used to define hypertext links to other resources. This module supports element a.

List Module − As its name suggests, the List Module provides list-oriented elements. Specifically, the List Module supports the following elements and attributes − dl, dt, dd, ol, ul, and li.

Object Module − The Object Module provides elements for general-purpose object inclusion. Specifically, the Object Module supports − object and param.

Presentation Module − This module defines elements, attributes, and a minimal content model for simple presentation-related markup − b, big, hr, i, small, sub, sup, and tt.

Edit Module − This module defines elements and attributes for use in editing-related markup − del and ins.

Bidirectional Text Module − The Bi-directional Text module defines an element that can be used to declare the bi-directional rules for the element's content − bdo.

Forms Module − It provides all the form features found in HTML 4.0. Specifically, it supports − button, fieldset, form, input, label, legend, select, optgroup, option, and textarea.

Table Module − It supports the following elements, attributes, and content model − caption, col, colgroup, table, tbody, td, tfoot, th, thead, and tr.

Image Module − It provides basic image embedding and may be used in some implementations

of client side image maps independently. It supports the element − img.

Client-side Image Map Module − It provides elements for client side image maps − area and map.

Server-side Image Map Module − It provides support for image-selection and transmission of selection coordinates. The Server-side Image Map Module supports − attribute ismap on img.

Intrinsic Events Module − It supports all the events discussed in XHTML Events.

Meta information Module − The Meta information Module defines an element that describes information within the declarative portion of a document. It includes element meta.

Scripting Module − It defines the elements used to contain information pertaining to executable scripts or the lack of support for executable scripts. Elements and attributes included in this module are − noscript and script.

Style Sheet Module − It defines an element to be used when declaring internal style sheets. The element and attribute defined by this module is − style.

Style Attribute Module *Deprecated* − It defines the style attribute.

Link Module − It defines an element that can be used to define links to external resources. It supports *link* element.

Base Module − It defines an element that can be used to define a base URI against which relative URIs in the document are resolved. The element and attribute included in this module is − *base*.

Ruby Annotation Module − XHTML also uses the Ruby Annotation module as defined in RUBY and supports − ruby, rbc, rtc, rb, rt, and rp.

Changes from XHTML 1.0 Strict

This section describes the differences between XHTML 1.1 and XHTML 1.0 Strict. XHTML 1.1 represents a departure from both HTML 4 and XHTML 1.0.

- The most significant is the removal of features that were deprecated.
- The changes can be summarized as follows −
- On every element, the lang attribute has been removed in favor of the *xml:lang* attribute.
- On the <a> and <map> elements, the name attribute has been removed in favor of the id attribute.
- The *ruby* collection of elements has been added.

XHTML - TIPS & TRICKS

This chapter lists out various tips and tricks which you should be aware of while writing an XHTML document. These tips and tricks can help you create effective documents.

Tips for Designing XHTML Document

Here are some basic guidelines for designing XHTML documents −

Design for Serving and Engaging Your Audience

When you think of satisfying what your audience wants, you need to design effective and catchy documents to serve the purpose. Your document should be easy for finding required information and giving a familiar environment.

For example, Academicians or medical practitioners are comfortable with journal-like document with long sentences, complex diagrams, specific terminologies, etc., whereas the document accessed by school-going children must be simple and informative.

Reuse Your Document

Reuse your previously created successful documents instead of starting from scratch each time you bag a new project.

Inside the XHTML Document

Here are some tips regarding elements inside the XHTML document −

The XML Declaration

An XML declaration is not required in all XHTML documents but XHTML document authors are strongly encouraged to use XML declarations in all their documents. Such a declaration is required when the character encoding of the document is other than the default UTF-8 or UTF-16.

Empty Elements

They include a space before the trailing / and > of empty elements. For example,
, <hr />, and .

Embedded Style Sheets and Scripts

Use external style sheets if your style sheet uses “<”, “&”, “]]>”, or “—”.

Use external scripts if your script uses “<”, “&”, or “]]>”, or “—”.

Line Breaks within Attribute Values

Avoid line breaks and multiple whitespace characters within attribute values. These are handled inconsistently by different browsers.

Isindex Element

Do not include more than one *isindex* element in the document head. The *isindex* element is deprecated in favor of the input element.

The *lang* and *xml:lang* Attributes

Use both the *lang* and *xml:lang* attributes while specifying the language of an element. The value of the *xml:lang* attribute takes precedence.

Element Identifiers

XHTML 1.0 has deprecated the name attributes of a, *applet, form, frame, iframe, img,* and *map* elements. They will be removed from XHTML in subsequent versions. Therefore, start using *id* element for element identification.

Using Ampersands in Attribute Values

The ampersand character "&" should be presented as an entity reference &.

Example

```
<!-- This is invalid in XHTML -->
http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user.

<!-- Correct XHTML way of writing this is as follows -->
http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user
```

Whitespace Characters in HTML and XML

Some characters that are legal in HTML documents are illegal in XML document. For example, in

HTML, the form-feed character U+000C is treated as white space, in XHTML, due to XML's definition of characters, it is illegal.

Named Character Reference *&Apos;*

The named character reference ' the apostrophe, U+0027 was introduced in XML 1.0 but does not appear in HTML. Web developers should therefore use ' instead of ' to work as expected in HTML 4 Web Browsers.

XHTML - VALIDATIONS

Every XHTML document is validated against a Document Type Definition. Before validating an XHTML file properly, a correct DTD must be added as the first or second line of the file.

Once you are ready to validate your XHTML document, you can use W3C Validator to validate your document. This tool is very handy and helps you to fix the problems with your document. This tool does not require any expertise to perform validation.

The following statement in the text box shows you details. You need to give complete URL of the page, which you want to validate and then click ***Validate Page*** button.

Input your page address in the box below −

This validator checks the markup validity of web documents with various formats especially in HTML, XHTML, SMIL, MathML, etc.

There are other tools to perform different other validations.

- **RSS/Atom feeds Validator**
- **CSS stylesheets Validator**
- **Find Broken Links**
- **Other validators and tools**

XHTML - SUMMARY

We assume you have understood all the concepts related to XHTML. Therefore, you should be able to write your HTML document into a well-formed XHTML document and get a cleaner version of your website.

Converting HTML to XHTML

You can convert your existing HTML website into XHTML website.

Let us go through some important steps. To convert your existing document, you must first decide which DTD you are going to adhere to, and include document type definition at the top of the document.

- Make sure you have all other required elements. These include a root element <html> that indicates an XML namespace, a <head> element, a <title> element contained within the <head> element, and a <body> element.
- Convert all element keywords and attribute names to lowercase.
- Ensure that all attributes are in a name="value" format.
- Make sure that all container elements have closing tags.

- Place a forward slash inside all standalone elements. For example, rewrite all
 elements as
.
- Designate client-side script code and style sheet code as CDATA sections.

XHTML Upcoming Versions

Still XHTML is being improved and its next version XHTML 1.1 has been drafted. We have discussed this in detail in XHTML Version 1.1 chapter.

XHTML Tags, Characters, and Entities

XHTML tags, characters, and entities are same as HTML, so if you already know HTML then you do not need to put extra effort to learn these subjects, especially for XHTML. We have listed out all HTML stuff along with XHTML tutorial also, because they are applicable to XHTML as well.

What is Next?

We have listed out various resources for XHTML and HTML so if you are interested and you have time in hand, then we recommend you to go through these resources to enhance your understanding on XHTML. Otherwise this tutorial must have given you enough knowledge to write your web pages using XHTML.

Your feedback on this tutorial is welcome at contact@tutorialspoint.com.