Karnataka ReddyJana Sangha (R)
VEMANA INSTITUTE OF TECHNOLOGY
Approved by AICTE-New Delhi, Affiliated to VTU-Belagavi, Recognized by Govt. of Karnataka
#1,Mahayogi Vemana Road, 3rd Block, Koramangala, Bengaluru - 34.
Accredited by NBA
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INTRODUCTION TO WEB PROGRAMMING -2ND SEM

MODULE 1

Module-1: Traditional HTML and XHTML: First Look at HTML and XHTML, Hello HTML and XHTML World, HTML and XHTML: Version History, HTML and XHTML DTDs: The Specifications Up Close, (X)HTML Document Structure, Browsers and (X)HTML, The Rules of (X)HTML, Major Themes of (X)HTML, The Future of Markup—Two Paths?

TextBook1: Chapter 1

Books (Title of the Book/Name of the author/Name of the publisher/Edition and Year)

TextBook-1: HTML & CSS: The Complete Reference Thomas A. Powell, Fifth Edition, Tata McGraw Hill.

TextBook-2: WEB PROGRAMMING with HTML5, CSS and JavaScript, John Dean, Jones & Bartlett Learning, First Edition.

Module-1: Traditional HTML and XHTML

- First Look at HTML and XHTML,
- Hello HTML and XHTML World,
- HTML and XHTML: Version, History,
- HTML and XHTML DTDs: The Specifications Up Close,
- (X)HTML Document Structure,
- Browsers and (X)HTML,
- The Rules of (X)HTML,
- Major Themes of (X)HTML,
- The Future of Markup—Two Paths?

XHTML:

XHTML -Extensible Hyper Text Markup Language. It can be considered as part of the XML markup language this is because of XHTML features for both XML and HTML.

XHTML is extended from XML and HTML. XHTML can be considered as a better version of XHTML.

It is a next step to evolution of internet. The XHTML was developed by World Wide Web Consortium(W3C). It helps web developers to make the transition from HTML to XML.

Using XHTML, developers can enter the XML world with all its features of it, and they can still be confident about the backward and future compatibility of the content.

- XHTML is a stricter, more XML-based version of HTML.
- XHTML is HTML defined as an XML application.

XHTML is supported by all major browsers.

Why XHTML?

XML is a markup language where all documents must be marked up correctly (be "well-formed").

XHTML was developed to make HTML more extensible and flexible to work with other data formats (such as XML). In addition, browsers ignore errors in HTML pages and try to display the website even if it has some errors in the markup. So, XHTML comes with much stricter error handling.

Advantages of XHTML:

- Strict syntax: XHTML has a stricter syntax compared to HTML5, which means that it is more structured and easier to read.
- Standardization: XHTML follows the syntax rules of XML, which is a standardized markup language. This makes it easier to create interoperable web pages that work well with different web browsers and devices.
- Better for parsing: XHTML is easier to parse and process than HTML5, which makes it a better choice for developers who want to create web pages that can be easily processed by other software tools.

Disadvantages of XHTML:

- More difficult to code: XHTML has a stricter syntax compared to HTML5, which can make it more difficult to code.
- Not backward compatible: XHTML is not backward compatible with older versions of HTML,

which means that some older web browsers may not be able to display XHTML documents properly.

- Requires more bandwidth: XHTML documents tend to require more bandwidth compared to HTML5 documents due to their stricter syntax and increased number of tags.

HTML 5:

HTML is the Hypertext Markup Language which is the most widely used language on the internet. HTML is used to create web pages and link them from one to another. Under traditional HTML, the closetag for some elements is optional because their closure can be inferred.

For example, a ** ** tag cannot enclose another ** **

** **This is a paragraph.

** **This is also a paragraph.

Such shortened notations that depend on inference may be technically correct under the specification, but stylistically they are not encouraged. It is always preferable to be precise, so use markup like this instead:

** **This is a paragraph. ** **

There are markup elements, called *empty elements*, which do not enclose any content, thus need no close tags at all, or in the case of XHTML use a self-close identification scheme.

For example, to insert a line break, use a single **
** tag, which represents the empty **br** element, because it doesn't enclose any content and thus has no corresponding close tag:

However, in XML markup variants, particularly XHTML, an unclosed tag is not allowed, so you need to close the tag.

 </br> or, more commonly, use a self-identification of closure like so:
.

Advantages of HTML5:

- Easier to code: HTML5 has a more relaxed syntax compared to XHTML, which makes it easier to code.
- Backward compatibility: HTML5 is designed to be backward compatible with older versions of HTML, which means that it can be used with older web browsers.
- Multimedia support: HTML5 includes support for multimedia elements such as video and audio,

which makes it easier to create web pages that include multimedia content.

Disadvantages of HTML5:

- Non-standardization: HTML5 is not a standardized markup language, which means that different web browsers may interpret it differently.
- Security issues: HTML5 includes new features such as geolocation and offline storage, which can create security vulnerabilities if not implemented properly.
- Incompatibility with older web browsers: Some older web browsers may not be able to display HTML5 documents properly, which can create compatibility issues for developers.

Similarities between the HTML AND XHML:

- **Basic Structure:**
  Both XHTML and HTML5 have the same basic structure, consisting of a head and a body. The head contains information about the document, such as the title and any scripts or stylesheets. The body contains the content that is displayed on the web page.
- **Semantics:**
  Both XHTML and HTML5 use semantic markup, which means that the tags used to structure the content are meaningful and descriptive. This helps search engines and screen readers to understand the content and improve accessibility.
- **Browser Support:**
  Both XHTML and HTML5 are supported by all modern web browsers. This means that web developers can choose either language and be confident that their pages will be displayed correctly in most browsers.
- **Separation of Content and Presentation:**
  Both XHTML and HTML5 encourage the separation of content and presentation, which means that the content should be structured using semantic markup, while the presentation should be handled separately using CSS (Cascading Style Sheets).
- **Accessibility:**
  Both XHTML and HTML5 prioritize accessibility and provide tools and techniques for creating accessible web pages. This includes using semantic markup, providing alternative text for images, and using ARIA (Accessible Rich Internet Applications) attributes to improve accessibility for

screen readers and other assistive technologies.

Difference between HTML and XHTML :

S.No.	HTML	XHTML
1.	HTML stands for Hypertext Markup Language.	XHTML stands for Extensible Hypertext Markup Language.
2.	It was developed by Tim Berners-Lee.	It was developed by W3C i.e World Wide Web Consortium.
3.	It was developed in 1991.	It was released in 2000.
4.	It is extended from SGML.	It is extended from XML and HTML.
5.	The format is a document file format.	The format is a markup language.
6.	All tags and attributes are not necessarily to be in lower or upper case.	In this, every tag and attribute should be in lower case.
7.	Doctype is not necessary to write at the top.	Doctype is very necessary to write at the top of the file.
8.	It is not necessary to close the tags in the order they are opened.	It is necessary to close the tags in the order they are opened.
9.	While using the attributes it is not necessary to mention quotes. For e.g. <Geeks>.	While using the attributes it is mandatory to mention quotes. For e.g. <Geeks="GFG">.

First Look at HTML

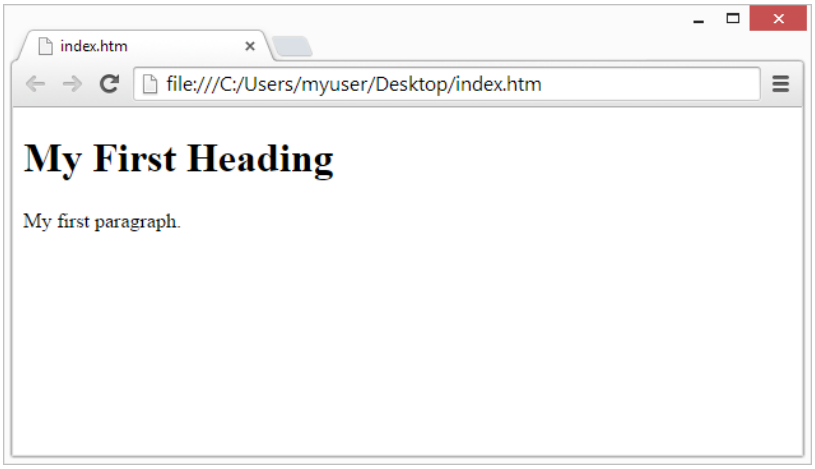
Example HTML Program:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph. </p>
</body>
</html>
```

- The <!DOCTYPE html> declaration defines that this document is an HTML document.
- The <html> element is the root element of an HTML page.

The <head> element contains meta-information about the HTML page.

- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The <h1> element defines a large heading.
- The <p> element defines a paragraph.



What is an HTML Element?

An HTML element is defined by a start tag, some content, and an end tag:

<tag name> Content goes here... </tag name>

The HTML **element** is everything from the start tag to the end tag:

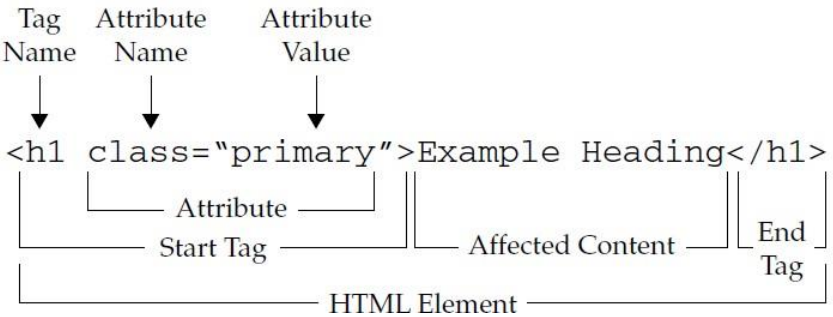
<h1>My First Heading</h1>

My first paragraph.

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	
 	None	*none*

Overview of HTML syntax

A graphical overview of the HTML markup syntax shown so far is presented



Hello HTML and XHTML World

HTML5 example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Hello HTML 4 World</title>
<! -- Simple hello world in HTML 4.01 strict example -->
</head>
<body>
<h1>Welcome to the World of HTML</h1>
<hr>
<p>HTML <em>really</em> isn't so hard! </p>
<p>Soon you will &hearts; using HTML. </p>
<p>You can put lots of text here if you want.
We could go on and on with fake text for you
to read, but let's get back to the book. </p>
</body>
</html>
```

XHTML EXAMPLE:

There are no major changes in XHTML documents**.** Here is the XHTML document,

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Hello XHTML World</title>
<! -- Simple hello world in XHTML 1.0 strict example -->
</head>
<body>
<h1>Welcome to the World of XHTML</h1>
<hr />
<p>XHTML <em>really</em> isn't so hard either! </p>
<p>Soon you will &hearts; using XHTML too. </p>
<p>There are some differences between XHTML
and HTML but with some precise markup you'll
see such differences are easily addressed. </p>
</body>
</html>
```

The most common elements used in (X)HTML documents are,

- The <!DOCTYPE> statement, which indicates the version of HTML or XHTML being used in the document. The first example uses the strict 4.01 specification, the second uses a reducedform for HTML5 the meaning of which will be explained a bit later, and the final example uses the XHTML 1.0 strict specification.

- The **<html>**, **<head>**, and **<body>** tag pairs are used to specify the general structure of the document. The required inclusion of the **xmlns** attribute in the **<html>** tag is a small difference required by XHTML.

- The <meta> tag used in the examples indicates the MIME type of the document and the character set in use. Notice that in the XHTML example, the element has a trailing slash to indicate that it is an empty element.

- The <title> and </title> tag pair specifies the title of the document, which generally appears in the title bar of the Web browser.

- A comment is specified by <!---->, allowing page authors to provide notes for future reference.

- The **<h1>** and **</h1>** header tag pair indicates a headline specifying some important information.

- The **<hr>** tag, which has a self-identifying end tag (**<hr />**) under XHTML, inserts a horizontal rule, or bar, across the screen.

- The ** ** and ** ** paragraph tag pair indicates a paragraph of text.

- A special character is inserted using a named entity (**♥**), which in this case inserts a heart dingbat character into the text.

- The **** and **** tag pair surrounds a small piece of text to emphasize which a browser typically renders in italics.

Viewing Markup Locally

Using a simple text editor, type in either one of the previous examples and save it with a

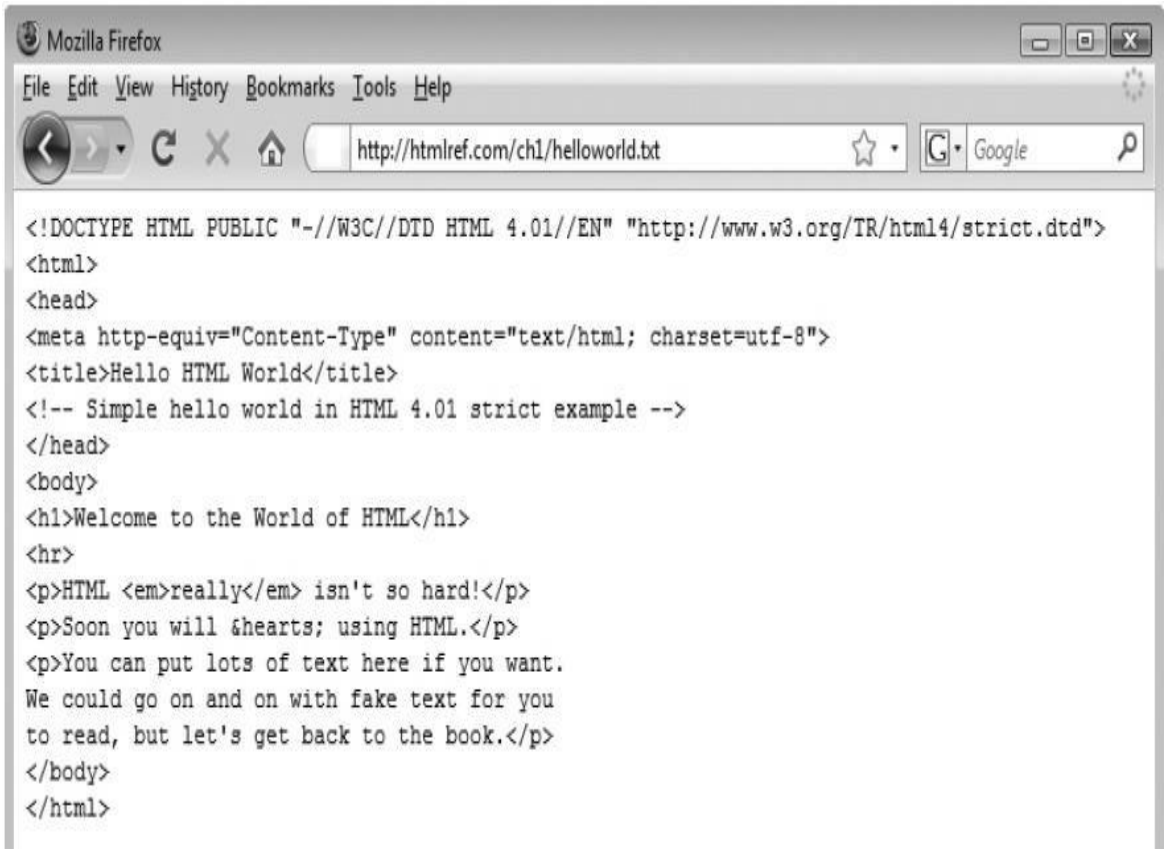
filename such as helloworld.html or helloworld.htm.

you can choose which file extension to use, .htm or .html, but whichever you pick for development, aim to be consistent.

After you save the example file on your local file system, open it in your Web browser by opening the File menu and choosing Open, Open Page, or Open File, depending on yourbrowser:

Once your browser reads the file, it should render a page like the one shown below,

If for some reason you didn't save your file with the appropriate extension, the browser shouldn't attempt to interpret the HTML markup. For example, notice here what happens when you try to open the content with a .txt extension:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://htmlref.com/ch1/helloworld.txt
Google
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Hello HTML World</title>
<!-- Simple hello world in HTML 4.01 strict example -->
</head>
<body>
<h1>Welcome to the World of HTML</h1>
<hr>
HTML really isn't so hard!
Soon you will ♥ using HTML.
You can put lots of text here if you want.
We could go on and on with fake text for you
to read, but let's get back to the book.
</body>
</html>

If you want to make a change to the document, you could update the markup, save the file, go back to the browser, and click the Reload or Refresh button. Sometimes the browser will still reload the page from its cache; if a page does not update correctly on reload, hold down the SHIFT key while clicking the Reload button, and the browser should refresh the page.

Viewing Markup with a Web Server:

There are many options for employing a Web server.

You may decide to run your own local development Web server on your desktop system or use some hosted server instead.

In either case, you need to get the files somewhere under the Web server's document root folder so that they can be served out.

Very often this directory has a common name like inetpub, htdocs, site, or www, but it really could be just about anything, so make sure you check the server you end.

HTML and XHTML: Version, History

Since its initial introduction in late 1991, HTML (and later its XML-based cousin, XHTML) has undergone many changes.

HTML or XHTML Version	Description
HTML 2.0	Classic HTML dialect supported by browsers such as Mosaic. This form of HTML supports core HTML elements and features such as tables and forms, but does not consider any of the browser innovations of advanced features such as style sheets, scripting, or frames.
HTML 3.0	The proposed replacement for HTML 2.0 that was never widely adopted, most likely due to the heavy use of browser-specific markup.
HTML 3.2	An HTML finalized by the W3C in early 1997 that standardized most of the HTML features introduced in browsers such as Netscape 3. This version of HTML supports many presentation-focused elements such as `font`, as well as early support for some scripting features.
HTML 4.0 Transitional	The 4.0 transitional form finalized by the W3C in December of 1997 preserves most of the presentational elements of HTML 3.2. It provides a basis of transition to Cascading Style Sheets (CSS) as well as a base set of elements and attributes for multiple-language support, accessibility, and scripting.
HTML 4.0 Strict	The strict version of HTML 4.0 removes most of the presentation elements from the HTML specification, such as `font`, in favor of using CSS for page formatting.
4.0 Frameset	The frameset specification provides a rigorous syntax for framed documents that was lacking in previous versions of HTML.

HTML 4.01 Transitional/ Strict/Frameset	A minor update to the 4.0 standard that corrects some of the errors in the original specification.
HTML5	Addressing the lack of acceptance of the XML reformulation of HTML by the mass of Web page authors, the emerging HTML5 standard originally started by the WHATWG[3] group and later rolled into a W3C effort aimed to rekindle the acceptance of traditional HTML and extend it to address Web application development, multimedia, and the ambiguities found in browser parsers. Since 2005, features now part of this HTML specification have begun to appear in Web browsers, muddying the future of XHTML in Web browsers.
XHTML 1.0 Transitional	A reformulation of HTML as an XML application. The transitional form preserves many of the basic presentation features of HTML 4.0 transitional but applies the strict syntax rules of XML to HTML.
XHTML 1.0 Strict	A reformulation of HTML 4.0 Strict using XML. This language is rule enforcing and leaves all presentation duties to technologies like CSS.
XHTML 1.1	A restructuring of XHTML 1.0 that modularizes the language for easy extension and reduction. It is not commonly used at the time of this writing and offers minor gains over strict XHTML 1.0.

HTML or XHTML Version	Description
XHTML 2.0	A new implementation of XHTML that will not provide backward compatibility with XHTML 1.0 and traditional HTML. XHTML 2 will remove all presentational tags and will introduce a variety of new tags and ideas to the language.
XHTML Basic 1.0	A variation of XHTML based upon the modularization of XHTML (1.1) designed to work with less-powerful Web clients such as mobile phones.
XHTML Basic 1.1	An improvement to the XHTML Basic specification that adds more features, some fairly specific to the constrained interfaces found in mobile devices.

HTML and XHTML DTDs -The Specifications Up Close

All (X)HTML documents should follow a formal structure defined by the World Wide Web Consortium (W3C; www.w3.org), which is the primary organization thatdefines Web standards.

Traditionally, the W3C defined HTML as an application of the Standard Generalized Markup Language (SGML).

SGML is a technology used to define markup languages by specifying the allowed document structure in the form of a document type definition (DTD)

A snippet of the HTML 4.01 DTD defining the **P** element, which indicates a paragraph, is shown here:

```
<!--==================== Paragraphs =================================-->
<!ELEMENT P - O (%inline;)*              -- paragraph -->
<!ATTLIST P
  %attrs;                                -- %coreattrs, %i18n, %events --
 >
```

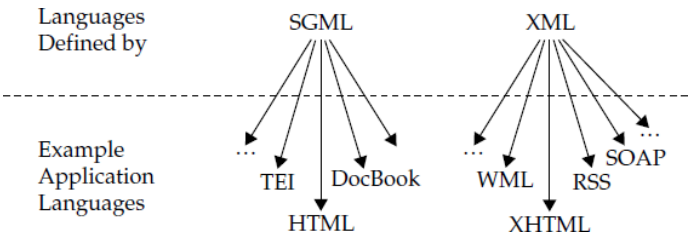
As another example, look at the HTML 4.01 DTD's definition of the **HR** element:

```
<!--==================== Horizontal Rule ================================-
<!ELEMENT HR - O EMPTY -- horizontal rule -->
<!ATTLIST HR
  %attrs;                                -- %coreattrs, %i18n, %events --
  >
```

DTD indicates the syntax that can be used for the various elements of a language such as HTML.

XML and SGML can be used to write arbitrary markup languages, not just HTML and XHTML. These would be called applications or, maybe more appropriately, application languages. Numerous markup languages have been defined with SGML and XML, and you could even define your own if you like.

The relationship between the various markup technologies is shown here,



The DTD defined in XML for the XHTML language is quite like the DTD for traditional HTML.

Document Type Statements and Language Versions

(X)HTML documents should begin with a <!DOCTYPE> declaration.

This statement identifies the type of markup that is supposedly used in a document.

For example,

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

indicates that we are using the transitional variation of HTML 4.01 that starts with a root element html.

In other words, an <html> tag will serve as the ultimate parent of all the content and elements within this document.

A <!DOCTYPE> declaration might get a bit more specific and specify the URI (Uniform Resource Identifier) of the DTD being used as shown here:

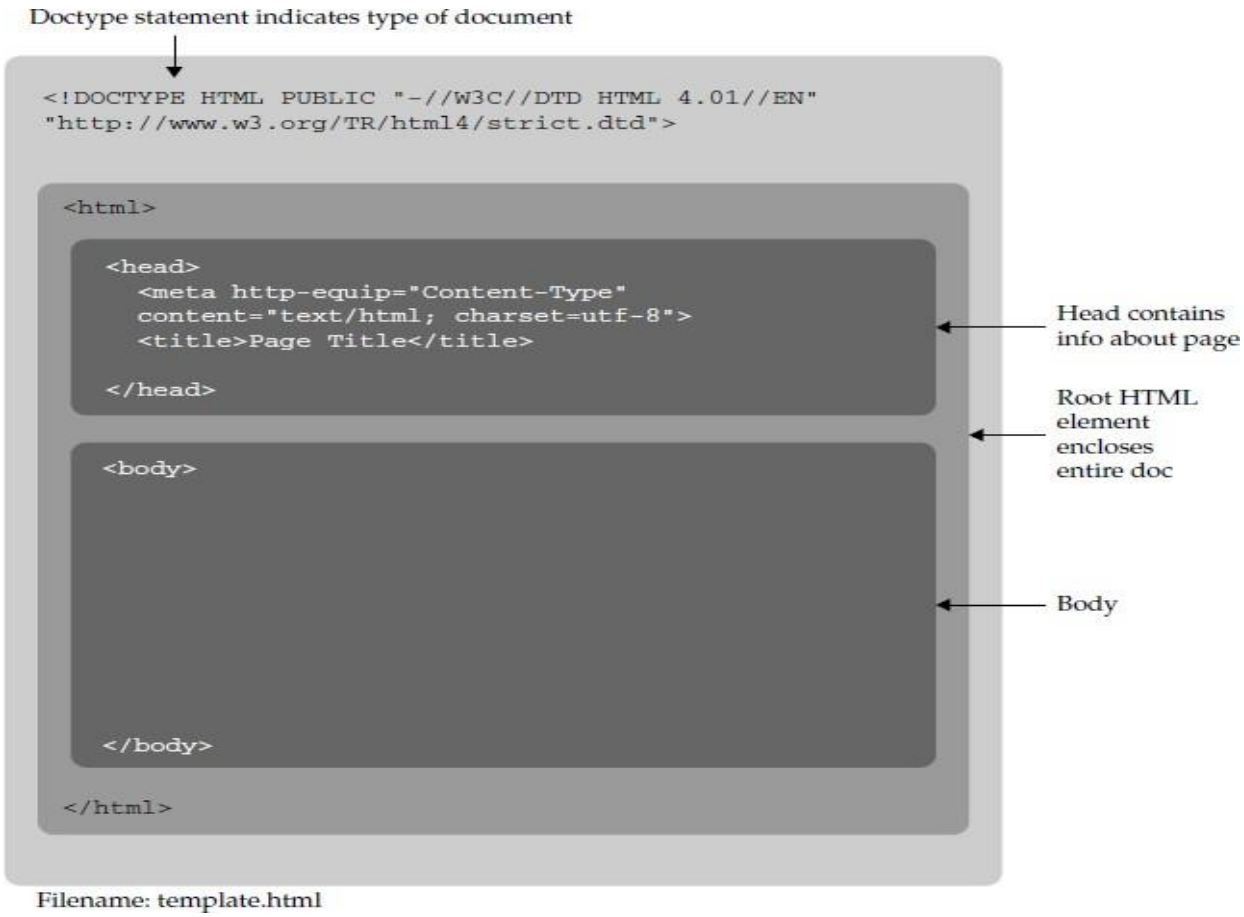
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

HTML or XHTML Version	!DOCTYPE Declaration
HTML 2.0	<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
HTML 3.2	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
HTML 4.0 Transitional	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
HTML 4.0 Frameset	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
HTML 4.0 Strict	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/html4/strict.dtd">
HTML 4.01 Transitional	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
HTML 4.01 Frameset	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
HTML 4.01 Strict	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
HTML5	<!DOCTYPE html>
XHTML 1.0 Transitional	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
XHTML 1.0 Strict	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
XHTML 1.0 Frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
XHTML 2.0	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN" "http://www.w3.org/MarkUp/DTD/xhtml2.dtd">
XHTML Basic 1.0	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN" "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
XHTML Basic 1.1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN" "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">

(X)HTML Document Structure

The DTDs define the allowed syntax for documents written in that version of (X)HTML. The core structure of these documents is fairly similar. Given the HTML 4.01 DTD, a basic document template can be derived from the specification, as shown below:



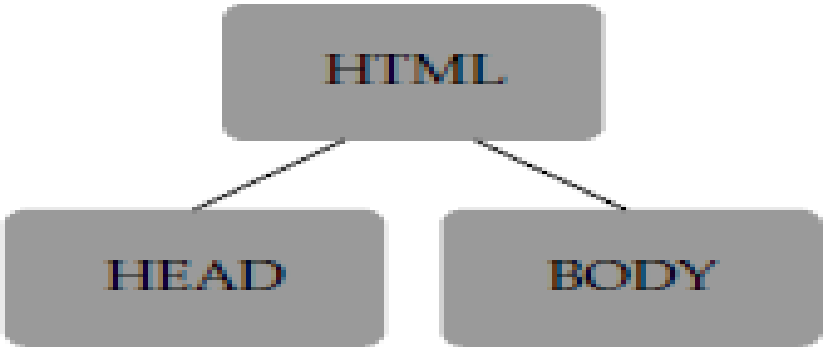
Alternatively, in either HTML or XHTML (but not in HTML5), we can replace the **<body>** tag with a **<frameset>** tag, which encloses potentially numerous **<frame>** tags corresponding to individual portions of the browser window, termed *frames*. Each frame in turn would reference another HTML/XHTML document containing either a standard document, complete with **<html>**, **<head>**, and **<body>** tags, or perhaps yet another framed document. The **<frameset>** tag also should include a **noframes** element that provides version

of the page for browsers that do not support frames.

Within this element, a **<body>** tag should be found for browsers that do not support frames. A visual representation of this idea is shown here:

Doctype statement indicates type of document.

Head contains info about the page.Root HTML element encloses entire document.



<!DOCTYPE> indicator, which, as previously mentioned, shows the version of HTML being used.
Within a root html element, the basic structure of a document reveals two elements: the head
and the body.
The head element contains information and tags describing the document, such as its title, while the body element houses the document itself, with associated markup required to specify its structure.

The structure of an XHTML document is pretty much the same except for a different <!DOCTYPE> indicator and an xmlns (XML name space) attribute added to the html tag so that it is possible to intermix
XML more easily into the XHTML document.
E ach of the document structuring markup elements are explained below,

The Document Head

The information in the **head** element of an (X)HTML document is very important because it is used to describe or augment the content of the document.
The element acts like the front matter or cover page of a document.
In many cases, the information contained within the **head** element is information about the page that is useful for visual styling, defining interactivity, setting the page title, and providing other useful informationthat describes or controls the document.

The title Element

A single **title** element is required in the **head** element and is used to set the text that most browsers display in their title bar.

The value within a **title** is also used in a browser's history system, recorded when the page is bookmarked,and consulted by search engine robots to help determine page meaning.

In short, it is important to have a syntactically correct, descriptive, and appropriate page title.

Thus, given

<title>Simple HTML Title Example**</title>**

<meta>: Specifying Content Type, Character Set, and More

A **<meta>** tag has several uses. For example, it can be used to specify values that are equivalent toHTTP response headers.

For example, if you want to make sure that your MIME type and character set for an English-based HTML document is set, you could use,

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

For standard HTML, the MIME type is always text/html. However, when XHTML is in play, confusion and browser problems ensue.

When a `title` is not specified, most browsers display the URL path or filename instead:

<meta>: Specifying Content Type, Character Set, and More

A <meta> tag has several uses. For example, it can be used to specify values that are equivalent toHTTP response headers.

For example, if you want to make sure that your MIME type and character set for an English-based HTML document is set, you could use,

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

For standard HTML, theMIME type is always text/html. However, when XHTML is in play, confusion and

browser problems ensue.

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" >

<title>Page title here</title>

</head>

Other Elements in the head:

In addition to the title and meta elements, under the HTML 4.01 and XHTML 1.0 strict DTDs, the elements allowed within the head element include base, link, object, script, and style. Comments are also allowed.

<base>

A <base> tag specifies an absolute URL address that is used to provide server and directory information for partially specified URL addresses, called relative links, used within the document:

<base href="http://htmlref.com/basexeample" >

<link>

A **<link>** tag specifies a special relationship between the current document and another document. Most commonly, it is used to specify a style sheet used by the document.

<link rel="stylesheet" media="screen" href="global.css" type="text/css" >

<object>

An **<object>** tag allows programs and other binary objects to be directly embedded in a Web page.

Here,for example, a nonvisible Flash object is being referenced for some use:

<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

width="0" height="0" id="Hidden Flash" >

<param name="movie" value="flashlib.swf" />

</object>

<script>

A **<script>** tag allows scripting language code to be either directly embedded within,

<script type="text/javascript">

alert("Hi from JavaScript!");

/* more code below */

</script>

or, more appropriately, linked to from a Web page:

<script type="text/javascript" href="ajaxtcr.js"></script>

<style>

A **<style>** tag is used to enclose document-wide style specifications, typically in Cascading Style Sheet (CSS) format, relating to fonts, colors, positioning, and other aspects of content presentation:

<style type="text/css" media="screen">

h1 {font-size: xx-large; color: red; font-style: italic;}

/* all h1 elements render as big, red and italic */

</style>

Comments

Finally, comments are often found in the head of a document. Following SGML syntax, a comment starts with <!-- and ends with --> and may encompass many lines:

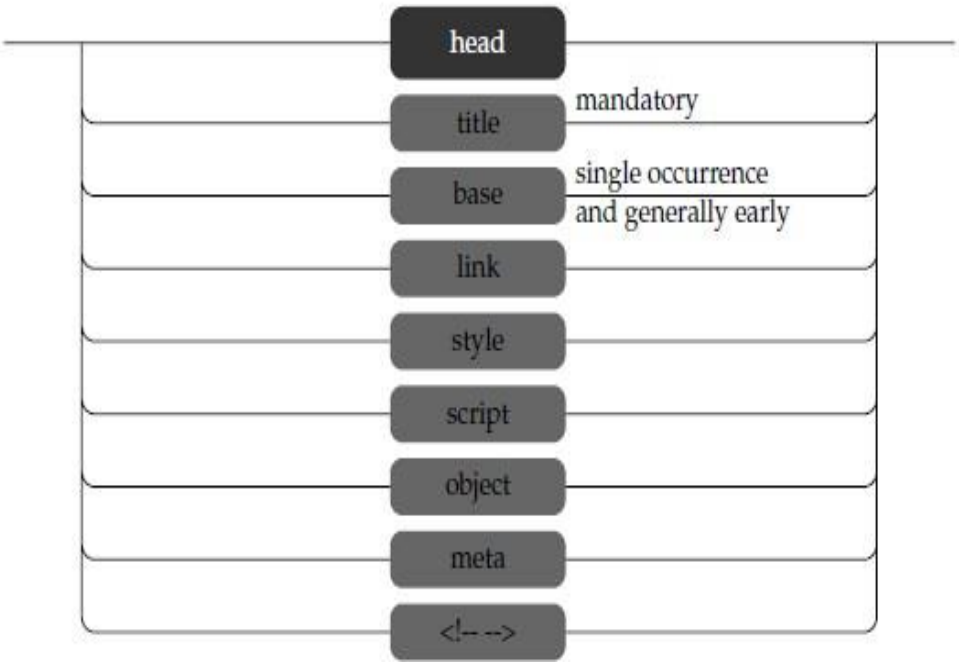
<!-- Hi I am a comment -->

<!-- Author: Thomas A. Powell

Book: HTML: The Complete Reference

Edition:5-->

The complete syntax of the markup allowed in the head element under strict (X)HTML is shown here:



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

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

<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>Sample Head Element</title>
<!-- Some example meta tags -->
<meta name="keywords" content="Fake, Head Example, HTML Ref" />
<meta name="description" content="A simple head example that shows a number
of the elements presented in action." />

<meta name="author" content="Thomas A. Powell" />


<!-- Set a global URI stem for all references -->
<base href="http://htmlref.com/baseexample" />

<!-- Linked and document specific styles -->
<link rel="stylesheet" href="screen.css" media="screen" />
<link rel="stylesheet" href="printer.css" media="print" />

<style type="text/css">
<!--
h1 {font-size: xx-large; color: red; font-style: italic;}
-->
</style>
<!-- Embedded and linked scripts -->
<script type="text/javascript">
```

```
<!--
var globalDebug = true;
//-->
</script>
<script src="ajaxtcr.js" type="text/javascript"></script>
<script src="effects.js" type="text/javascript"></script>
</head>
<body>
<p>Some body content here.</p>
</body>
</html>
```

The Document Body

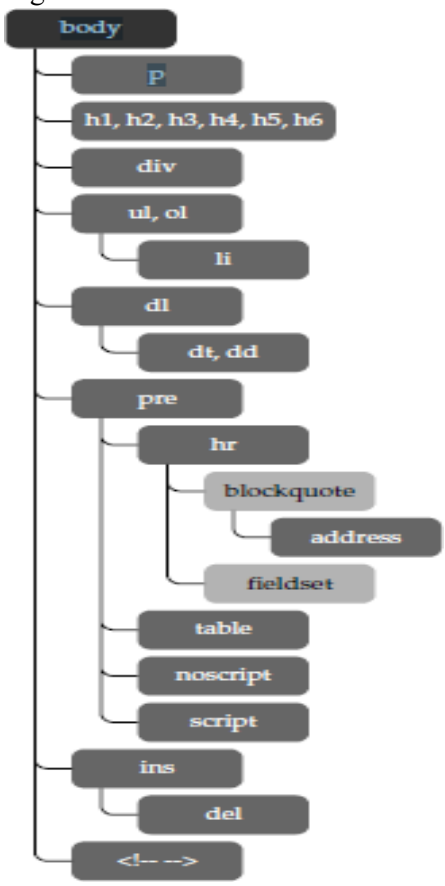
After the head section, the body of a document is delimited by **<body>** and **</body>**.
Within the body of a Web document is a variety of types of elements.

Block-level elements define structural content blocks such as paragraphs (**p)** or headings (**h1-h6**).

Block-level elements generally introduce line breaks visually. Special forms of blocks, such as unordered lists (**ul**), can be used to create lists of information.
Within nonempty blocks, *inline elements* are found.

There are numerous inline elements, such as bold (**b**), italic (**i**), strong (**strong**), emphasis (**em**), and numerous others. These types of elements do not introduce any returns.

Regulation :2022 scheme

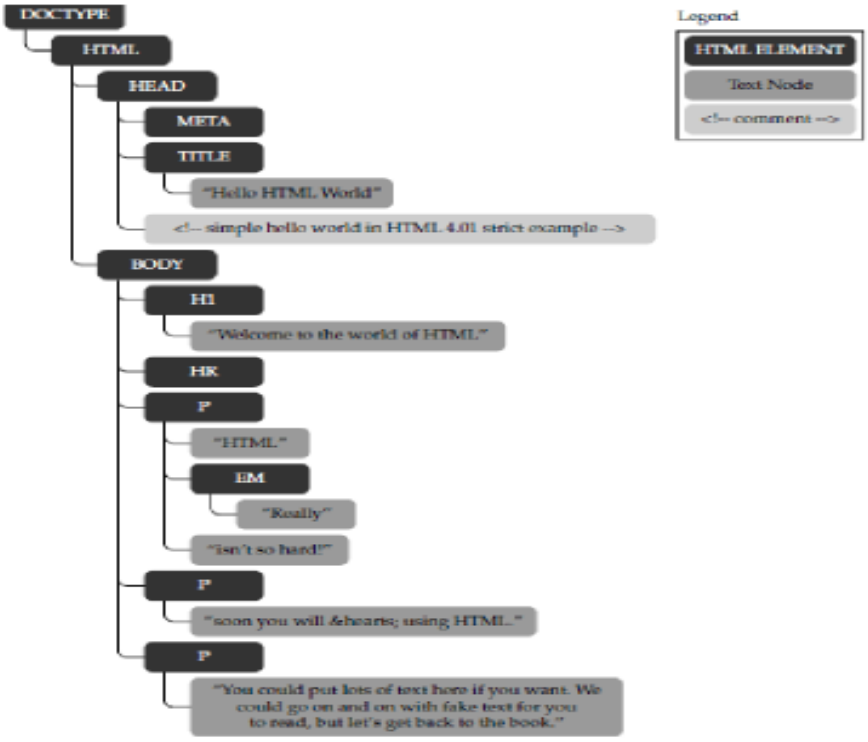



```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
<title>Hello HTML World</title>
<!-- Simple hello world in HTML 4.01 strict example
-->
</head>
<body>
<h1>Welcome to the World of HTML</h1>
 <hr>
<p>HTML <em>really</em> isn't so hard!</p>
<p>Soon you will &hearts; using HTML.</p>
```

```
<p>You can put lots of text here if you want.
We could go on and on with fake text for you
to read, but let's get back to the book.</p>

</body>

</html>
```



Browsers and (X)HTML

Browsers are quite permissive in what they will render. For Example,

```
<TITLE>Hello HTML World</title>
<!-- Simple hello malformed world -- example -->
</head>
<body>
<h1>Welcome to the World of HTML</H1>
<hr />
<p>HTML <eM>really</Em> isn't so hard!
<P>Soon you will &hearts; using HTML.
<p>You can put lots of text here if you want.
We could go on and on with fake text for you
to read, <foo>but</foo> let's get back to the book.
</html>
```

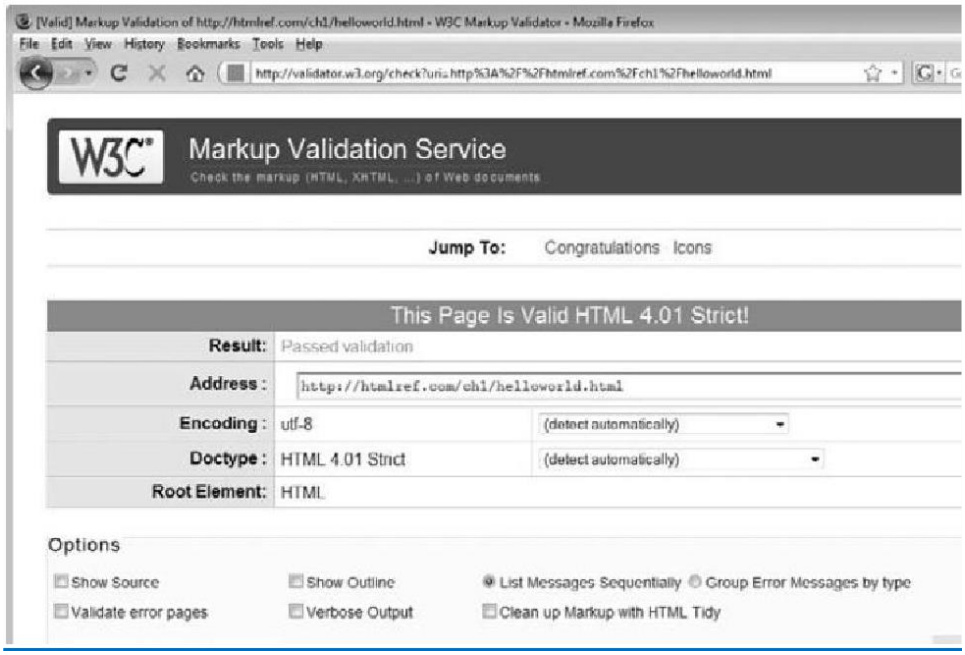
Validation

DTD defines the actual elements, attributes, and element relationships that are valid in documents.

If you take a document written in (X)HTML and then check whether it conforms to the rules specified by the DTD used.

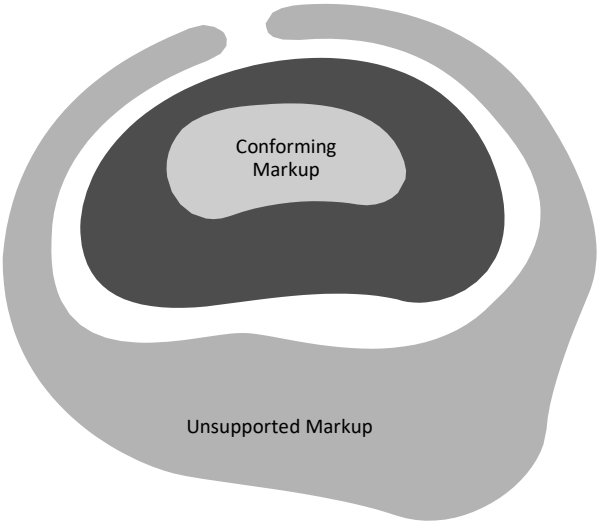
This process of checking whether a document conforms to the rules of the DTD is called validation.

The <!DOCTYPE> declaration allows validation software to identify the HTML DTD being followed in a document and verify that the document is syntactically correct—in other words, that all tags used are part of a particular specification and are being used correctly.



Web developers should aim to start with a baseline of valid markup before trying to address various browser quirks and bugs. Given that so many Web pages on the Web arepoorly coded, some developers opt to add a “quality” badge to a page to show or even prove standards conformance:

Whether users care about such things is debatable, but the aim for correctness is appropriate. Contrast this to the typical effort of testing a page by viewing it in various browsers to see what happens. The thought is, if it looks right, then it is right. However, this does not acknowledge that the set of supported or renderable pages a browser may handle is a superset of those which are conforming to a particular specification:

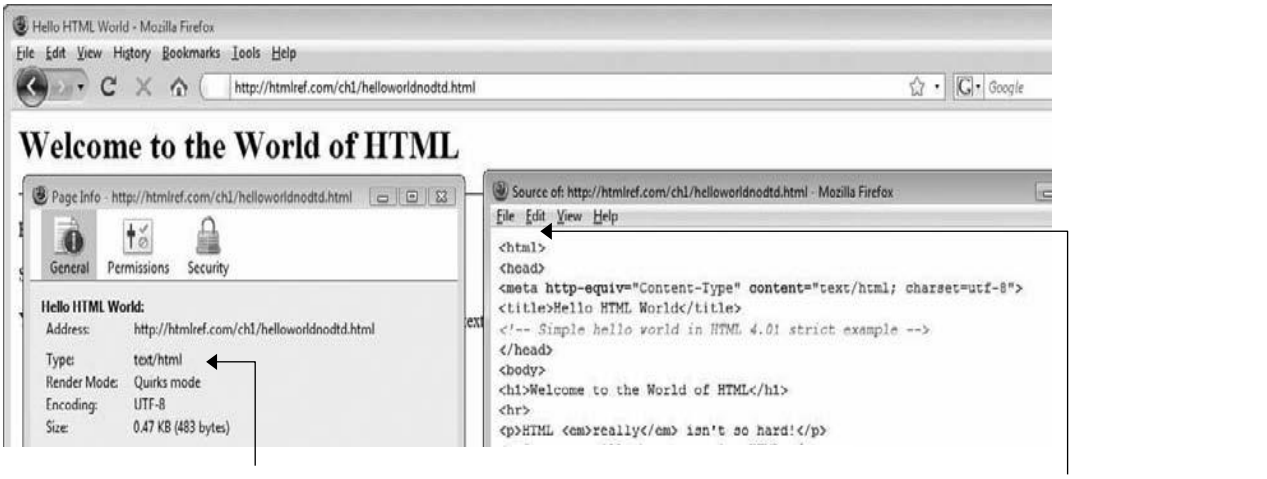


It is an unfortunate reality that browsers support a multitude of incorrect things and that developers often use a popular browser as an acceptance engine based upon some page rendering for better or worse. Such an approach to markup testing might seem reasonable in the short term, but it will ultimately lead to significant developer frustration, particularly as other technologies are added, such as CSS and JavaScript, and newer browsers are introduced. Unfortunately, given the browsers' current method of allowing garbage yet preferring standards, there is little reason for some developers to care until such a price is realized.

The Doctype Switch and Browser Rendering Modes

Modern Web browsers generally have two rendering modes: quirks mode and standards compliance mode. As their names suggest, quirks mode is more permissive and standards compliance mode is stricter. The browser typically chooses in which mode to parse a document by inspecting the `<!DOCTYPE>` statement if there is one. This process typically is dubbed the "doctype switch." When a browser sees a known standards-focused doctype indicator, it switches into a standards compliant parse:

However, if the **<!DOCTYPE>** statement is missing, references a very old version like 3.2, or is unknown, the browser will enter quirks mode. Browsers may provide an indication of the rendering mode via an entry in page info:



In other cases, you may need to use a tool to determine the parse mode:


Hello HTML World - Internet Explorer provided by Dell
http://htmlref.com/ch1/helloworldnodtd.html
File Edit View Favorites Tools Help
Favorites
Hello HTML World
DebugBar v5.0.2
DOM HTTP(S) Script HTMLCheck Info
Information Value
Main page URL http://htmlref.com/ch1/helloworldnodtd.html
Page title Hello HTML World
Main page Size 483
inline images (<im... 0
Size inline images (<... 0
Download time 0
Rendering Page Time 0
Page charset utf-8
Mime Type Firefox Document
Protocol HyperText Transfer Protocol
Display mode Quirks mode
Welcome to the World of HTML
HTML really isn't so hard!
Soon you will ♥ using HTML.
You can put lots of text here if you want. We could go on and on with fake text for you to read, but let's get back to the book.

The Rules of (X)HTML

These markup examples are all equivalent under traditional HTML:

- HTML Is Not
  Case
  Sensitive

Eg:

<B>Go boldly

Go boldly</B>

HTML5 reverts to case-insensitive markup and thus we may see a return to uppercase tags bystandards aware developers.

- Attribute Values May Be
  Case Sensitive

Eg:

<img src="test.gif"> and

<img src="TEST.GIF"> do not necessarily reference the

same image.Under traditional HTML,

these are equivalent because the <img> tag and the src attribute are not case sensitive.

- (X)HTML Is Sensitive to a Single Whitespace Character

Any white space between characters displays as a single space. This includes all tabs, linebreaks, and carriage returns. Consider this markup:

T e s t o f s p a c e s

T e s t o f s p a c e s

Te s t o f s p a c e s

As shown here, all the spaces, tabs, and returns are collapsed to a single element:However, it is possible to force the whitespace issue.

- HTML Follows a Content Model

All forms of markup support a content model that specifies that certain elements aresupposed to occur only within other elements.

For Eg:

What a simple way to break the content model!

The <ul> tag is only supposed to contain <li> tags. The tag is not appropriate in this context.

- Elements Should Have Close Tags Unless Empty

Under traditional HTML, some elements have optional close tags. For example, both paragraphs here are allowed, although the second one is better:

** **This isn't closed.

** **This is** **

- Unused Elements May Minimize

Sometimes tags may not appear to have any effect in a document. Consider, for example, the tag, which specifies a paragraph. As a block tag, it induces a return by default, butwhen used repeatedly, like so,

- Elements Should Nest

Under traditional HTML as well as under HTML5, simple attribute values do not need to bequoted.

****is not since tags nest****

- Attributes Should Be Quoted

<img src=robot.gif height=10 width=10 alt=robot>

<img src="robot.gif" height="10" width="10"

alt="robot" />

- Entities Should Be Used for Special Characters

instead of <, use **&lt**

Instead of >, use **&gt**

Yen symbol (¥), you would use **&yen**

- Browsers Ignore Unknown Attributes and Elements

<bogus>this text will display on screen**</bogus>**

** **will also render fine.** **

Major Themes of (X)HTML

Logical and Physical Markup

Physical markup refers to using a markup language such as (X)HTML to make pages look a particularway;
logical markup refers to using (X)HTML to specify the structure or meaning of content while using anothertechnology, such as CSS,to designate the look of the page.
Physical markup is obvious; if you want to highlight something that is important to the reader, you mightembolden it by enclosing it within a **** tag:
****This is important!****

This simple approach fits with the WYSIWYG (*what you see is what you get*) world of programs such asMicrosoft Word.

Logical markup is a little less obvious; to indicate the importance of the phrase, it should be enclosed in thelogical **strong** element:
****This is important.****

However the semantic meaning of **strong** provides a bit more flexibility and is preferred. Remember, the **** tag is used to say that something is important content, not to indicate how it looks. If a CSS rule were defined to say that important items should be big, red, anditalic.

```
<style="text/css">
strong
{
font-size:
xx-large;
color: red;
font-style: italic;
}
</style>
```

confusion would not necessarily ensue, because we shouldn't have a predisposed view of what **strong**

means visually.

Standards vs. Practice

Without standards, the modern world wouldn't work well. For example, imagine a world of construction inwhich every nut and bolt might be a slightly different size.

Standards provide needed consistency.

The Web needs standards, but standards must acknowledge what people do. Declaring that Web developers really need to validate, use logical markup, and separate the look from the structure of the document is great but it doesn't get them to do so.

Standards are especially pointless if they are never widely implemented.

Web standards and development practices provide an interesting study of the difference between whattheorists say and what people want and do.

HTML5 seems a step in the right direction.

Myths and Misconceptions About HTML and XHTML

The complete misunderstandings about HTML and XHTML are enormous.

- Misconception: WYSIWYG Works on the Web

(X)HTML isn't a specific, screen- or printer-precise formatting language like PostScript.

Interestingly, even the concept of a visual WYSIWG editor propagates this myth of HTML as a page layoutlanguage.

Other technologies, such as CSS, are far better than HTML for handling presentation issues and their usereturns HTML to its structural roots.

- Misconception: HTML Is a Programming Language

Many people think that making HTML pages is like programming.

However, HTML is unlike programming in that it does not specify logic. It specifies the structure of adocument.

The introduction of scripting languages such as JavaScript into Web documents and the confusing termsDynamic HTML (DHTML) and Ajax (Asynchronous JavaScript and XML) tacked on may lead many tooverestimate or underestimate the role of markup in the mix.

However, markup is an important foundation for scripting and should be treated with the same syntacticalprecision that script is given.

- Misconception: XHTML Is the Only Future

most documents are not authored in XHTML. Poor developer education, the more stringent syntax requirements, and ultimately the lack of obvious tangible benefit may have kept many from adopting theXML variant of HTML.

- Misconception: XHTML Is Dead

In fact, you can write XML-style markup in HTML, which most developers dub XHTML 5. For precision, XHTML is the way to go, particularly when used in an environment that includes otherforms of XML documents.

XHTML's future is bright for those who build well-formed, valid markup documents.

Myth: Traditional HTML Is Going Away

HTML is the foundation of the Web; with literally billions of pages in existence, not every document is going to be upgraded anytime soon. The "legacy" Web will continue for years, and traditional non standardized HTML will always be lurking around underneath even the most advanced Web page years from now. Beating the standards drum might speed things up a bit, but the fact is, there's a long way to go before we are rid of messed-up markup.

HTML5 clearly acknowledges this point by documenting how browsers should act considering malformed markup.

Having taught HTML for years and having seen how both HTML editors and people build Web pages, I think it is very unlikely that strictly conforming markup will be the norm anytime soon. Although (X)HTML has had rules for years, people have not really bothered tofollow them; from their perspective, there has been little penalty for failing to follow the rules, and there is no obvious benefit to studying the language rigorously. Quite often, people learn markup simply through imitation by viewing the source of existing pages, which are not necessarily written correctly, and going from there. Like learning a spoken language, (X)HTML's loosely enforced rules have allowed many document authors to get going quickly. Its biggest flaw is in some sense its biggest asset and has allowed millions of people to get involved with Web page authoring. Rigor and structure are coming, but it will take time, tools, and education.

Myth: Someday Standards Will Alleviate All Our *Problems*

Standards are important. Standards should help. Standards likely won't fix everything. From varying interpretations of standards, proprietary additions, and plain old bugs, there is likely never going to be a day where Web development, even at the level of (X)HTML markup, doesn't have its quirks and oddities. The forces of the market so far have proven this sentiment to be, at the very least, wishful thinking. Over a decade after first being considered during the writing of this book's first edition, the wait for some standards

nirvana continues.

Myth: Hand-Coding of HTML Will Continue Indefinitely

Although some people will continue to craft pages in a manner like mechanical typesetting, as Web editors improve and produce standard markup perfectly, the need to hand-tweak HTML documents will diminish. Hopefully, designers will realize that knowledge of the "invisible pixel" trick or the CSS Box Model Hack is not a bankable resume item and instead focus on development of their talents along with a firm standards-based understanding of markup, CSS, and JavaScript.

Myth: (X)HTML Is the Most Important Technology Needed to Create Web Pages

Whereas (X)HTML is the basis for Web pages, you need to know a lot more than markup to build useful Web pages (unless the page is very simple). However, don't underestimate markup because it can become a bit of a challenge itself. Based on the simple examples presented in this chapter, you might surmise that mastering Web page creation is merely a matter of learning the multitude of markup tags, such as `<h1>`, `<p>`, `<em>`, and so on, that specify the structure of Web documents to browsers. While this certainly is an important first step, it would be like believing you could master the art of writing by simply understanding the various commands available in Microsoft Word. There is a tremendous amount to know in the field of Web design and development, including information architecture, visual design, client- and server-side programming, marketing and search engines, Web servers and delivery, and much, much more.

The Future of Markup—Two Paths?

XHTML: Web Page Markup XML Style

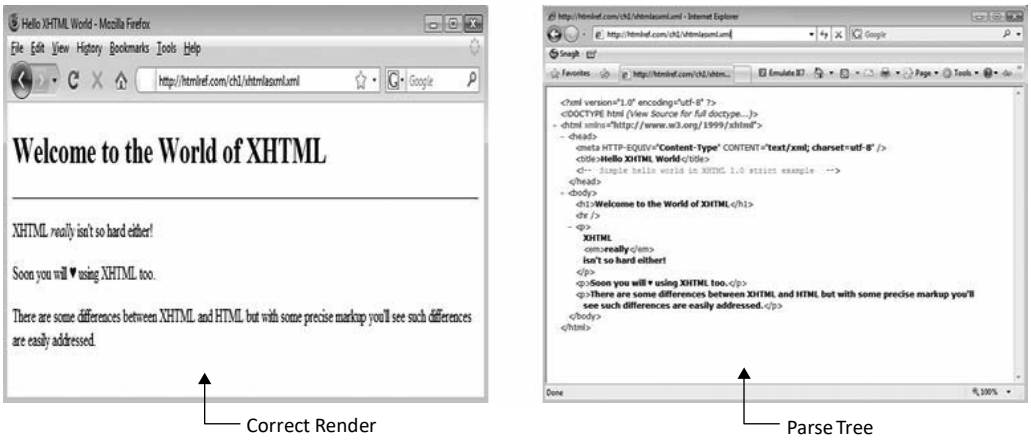
A new version of HTML called XHTML became a W3C recommendation in January 2000. XHTML, as discussed earlier in the chapter, is a reformulation of HTML using XML that attempts to change the direction and use of HTML to the way it ought to be. So what does that mean? In short, rules now matter. As you know, you can feed a browser just about anything and it will render. XHTML would aim to end that. Now if you make a mistake, it should matter.

Theoretically, a strictly XHTML-conforming browser shouldn't render a page at all if it doesn't conform to the standard, though this is highly unlikely to happen because browsers resort to a backward-compatibility quirks mode to display such documents. The question is, could you enforce the strict sense of XML using XHTML? The short answer is, maybe not ideally.

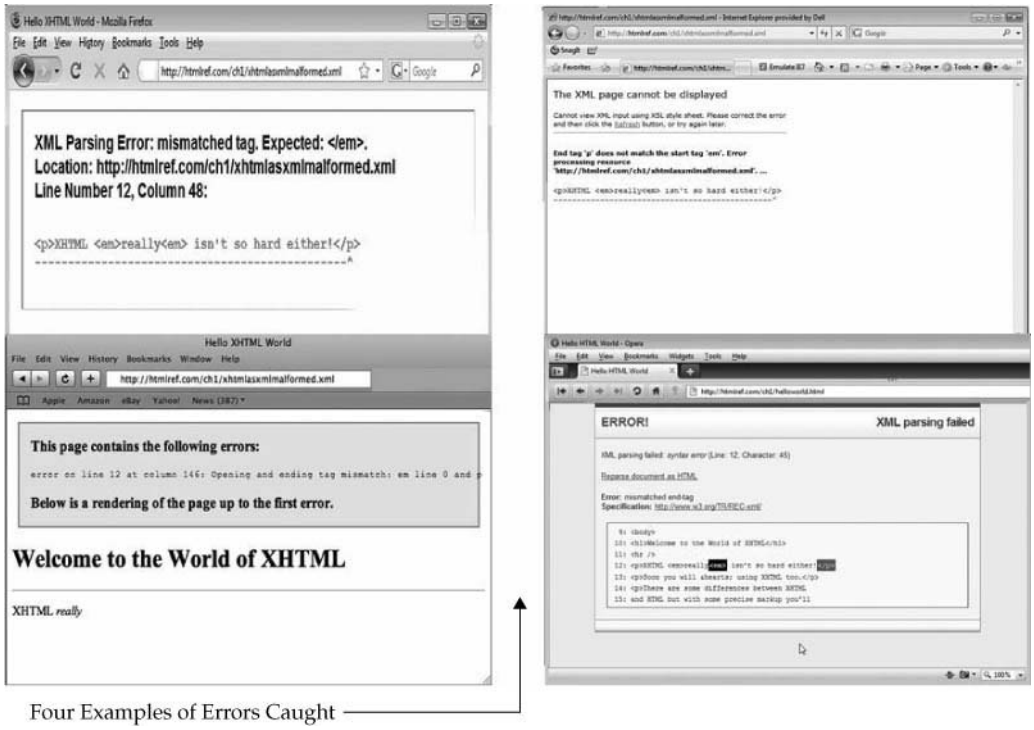
To demonstrate, let's reformulate the xhtmlhelloworld.html example slightly by adding an XML directive and forcing the MIME type to be XML. We'll then try to change the file extension to .xmlto ensure that the server gets the browser to really treat the file as XML data.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/xml; charset=utf-8" />
<title>Hello XHTML World</title>
<!-- Simple hello world in XHTML 1.0 strict example -->
</head>
<body>
<h1>Welcome to the World of XHTML</h1>
<hr />
<p>XHTML <em>really</em> isn't so hard either!</p>
<p>Soon you will &hearts; using XHTML too.</p>
<p>There are some differences between XHTML and HTML but with some precise markup you'll see such differences are easily addressed.</p>
</body>
</html>
```

Interestingly, most browsers, save Internet Explorer, will not have a problem with this. Internet Explorer will treat the apparent XML acting as HTML as normal HTML markup, but if we force the issue, it will parse it as XML and then render an XML tree rather than a default rendering:



To get the benefit of using XML, we need to explore if syntax checking is really enforced. Turns out that works if the browser believes markup to be XML, but not if the browser gets the slightest idea that we mean for content to be HTML. See for yourself when you try the examples that follow. You should note it properly fails when it assumes XML and not when it suspects HTML.



HTML5: Back to the Future

Starting in 2004, a group of well-known organizations and individuals got together to form a standards body called the Web Hypertext Application Technology Working Group, or WHATWG (www.whatwg.org), whose goal was to produce a new version of HTML. The exact reasons and motivations for this effort seem to vary depending on who you talk to—slow uptake of XHTML, frustration with the lack of movement by the Web standards body, need for innovation, or any one of many other reasons—but, whatever the case, the aim was to create a new, rich future for Web applications that include HTML as a foundation element. Aspects of the emerging specification such as the **canvas** element have already shown up in browsers like Safari and Firefox, so by 2008, the efforts of this group were rolled into the W3C and drafts began to emerge. Whether this makes HTML5 become official or likely to be fully adopted is obviously somewhat at the mercy of the browser vendors and the market, but clearly another very likely path for the future of markup goes through HTML5. Already we see Google adopting it in various places, so its future looks bright.

HTML5 is meant to represent a new version of HTML along the HTML 4 path. The emerging specification also suggests that it will be a replacement for XHTML, yet it ends up supporting most of the syntax that end users use, particularly self-identifying empty elements (for example, **
**). It also reverses some of the trends, such as case sensitivity, that have entered markup circles, so it would seem that the HTML styles of the past will be fine in the future. In most ways, HTML5 doesn't present much of a difference, as you saw earlier in the chapter's introductory example, shown again here:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Hello HTML World</title>
<!-- Simple hello world in HTML5 example -->
</head>
<body>
<h1>Welcome to the Future World of HTML5</h1>
<hr>
```

```
<p>HTML5 <em>really</em> isn't so hard!</p>

<p>Soon you will &hearts; using HTML.</p>

<p>You can put lots of text here if you want.
We could go on and on with fake text for you
to read, but let's get back to the book.</p>

</body>
<html>
```

All that is different in this example is that the **<!DOCTYPE>** statement is much simpler. In fact, the specific idea of using SGML and performing validation does not apply to HTML5. However, the syntax checking benefits of validation lives on and is now being called conformance checking and for all intents and purposes is the same. Interestingly, because of the statement in its shortened form, browsers will correctly enter into a standards compliance mode when they encounter an HTML5 document: