THE MINISTRY OF SCIENCE AND TECHNOLOGY

THE CORPORATION FOR FINANCING AND PROMOTING TECHNOLOGY

GUIDELINE

HTML Coding Convention

Code	09ee-HD/PM/HDCV/FSOFT
Issue/revision	2/0
Effective date	10/04/2007

TABLE OF CONTENT

1 INTRODUCTION

1.1 Purpose of the Guidelines

This is not a tutorial on HTML. The purpose of this document is to provide **good practices with Do's and Don'ts** for software developers of our FPT-Software, who **design and write HTML 4.01 pages** (site designers and page authors).

Good practice does not mean that the knowledge described should always be applied uniformly on all projects; the project team is responsible for determining what is appropriate for their projects.

1.2 Scope of the Guidelines

Some of the **expected benefits** include:

- Providing **consistency** throughout the entire site;
- **Reducing effort** in maintenance tasks, when the content of pages needs to be updated, or given a new layout;
- Generating pages that **look acceptable** to users regardless of the browser they are using;
- Making the web pages **search-engine-friendly**.

This guidelines will focus only on the HTML language which was defined by the **Official W3C HTML 4.01 Specification**.

At the time of writing this guidelines, W3C recommends that user agents and authors (and in particular, authoring tools) produce HTML 4.01 documents rather than HTML 4.0 documents. HTML 4.01 is a revision of HTML 4.0 that corrects errors and makes some changes since the previous revision. HTML 4 extends HTML with mechanisms for style sheets, scripting, frames, embedding objects, improved support for right to left and mixed direction text, richer tables, and enhancements to forms, offering improved accessibility for people with disabilities.

What you **will NOT find in this guidelines** is:

- This guidelines does not try to cover all requirements and recommendations of the HTML 4.01 specification.
- This guidelines does not try to explain how to produce the next generation of HTML, "The Extensible HyperText Markup Language" [XHTML].
- This guidelines does not try to explain how to make Web content accessible to people with disabilities.
- This guidelines does not try to explain how to design documents that take advantage of the characteristics of the media where the document is to be rendered (e.g., graphical displays,

television screens, handheld devices, speech-based browsers, braille-based tactile devices, etc.).

1.3 Terms and Abbreviations

Typical abbreviations used in the document are listed in the below table.

Abbreviations	Description
HTML	Hyper Text Markup Language
W3C	the World Wide Web Consortium
XHTML	EXtensible HyperText Markup Language

1.4 References

Ord#	Source Information
01	Adding a touch of style (http://www.w3.org/MarkUp/Guide/Style)
02	Getting started with HTML (http://www.w3.org/MarkUp/Guide/)
03	HTML 4.01 Specification (http://www.w3.org/TR/html4/)
04	More advanced features (http://www.w3.org/MarkUp/Guide/Advanced.html)
05	Style guide for online hypertext (http://www.htmlhelp.com/design/style/style.html)
06	Web Content Accessibility Guidelines 1.0 (http://www.w3.org/TR/WCAG10/)

2 WEB CONTENT ACCESSIBILITY

2.1 The Importance of Providing Text Equivalents of Non-Text Content

- Use "**alt**" for the **IMG**, **INPUT**, and **APPLET** elements, or provide a text equivalent in the content of the **OBJECT** and APPLET elements.
- For **image maps**, either use the "alt" attribute with **AREA**, or use the MAP element with **A** elements (and other text) as content.

2.2 Create Tables That Transform Gracefully

- For data tables, identify row and column headers. For example, in HTML, use **TD** to identify data cells and **TH** to identify headers.
- For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells. For example, in HTML, use **THEAD**, **TFOOT**, and **TBODY** to group rows, **COL** and **COLGROUP** to group columns, and the "axis", "scope", and "headers" attributes, to describe more complex relationships among data.
- Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent. **Once user agents support style sheet positioning, tables should not be used for layout**.

This example illustrates the order and structure of table heads, feet, and bodies.

```
<TABLE>
<THEAD>
     <TR> ...header information...
</THEAD>
<TFOOT>
     <TR> ...footer information...
</TFOOT>
<TBODY>
     <TR> ...first row of block one data...
     <TR> ...second row of block one data...
</TBODY>
<TBODY>
     <TR> ...first row of block two data...
     <TR> ...second row of block two data...
     <TR> ...third row of block two data...
</TBODY>
</TABLE>
```

2.3 Ensure That Pages Featuring New Technologies Transform Gracefully

- Organize documents so they **may be read without style sheets**. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document.
- Ensure that pages are **usable when scripts, applets, or other programmatic objects are turned off or not supported**. If this is not possible, provide equivalent information on an alternative accessible page. For example, ensure that links that trigger scripts work when scripts are turned off or not supported (e.g., do not use "javascript:" as the link target). If it is not possible to make the page usable without scripts, provide a text equivalent with the NOSCRIPT element, or use a server-side script instead of a client-side script.

2.4 Provide Context and Orientation Information

- Title each frame to facilitate frame identification and navigation. For example, in **HTML use the "title" attribute on FRAME elements**.
- Divide large blocks of information into more manageable groups where natural and appropriate. For example, in HTML, use **OPTGROUP** to group **OPTION** elements inside a **SELECT**; group form controls with **FIELDSET** and **LEGEND**; use nested lists where appropriate; use headings to structure documents, etc.
- Associate labels explicitly with their controls. For example, in HTML use **LABEL** and its "**for**" attribute.

2.5 HTML File Naming Convention

Many Web professionals recommend that you use only lower-case letters and no space in file names.

- Less chance of coding errors because you don't have to remember what to capitalize and when.
- Visitors have an easier time if they're trying to type the URL directly into their browsers instead of clicking on a link.

An all lower-case format reduces errors, but also makes file names harder to read. As a compromise, designers use the underscore symbol (_) or a hyphen (-) to separate individual words in a file name.

3 HTML LANGUAGE - REQUIREMENTS AND RECOMMENDATIONS

3.1 Specifying HTML Version Information

HTML 4.01 specifies three DTDs, so authors must include one of the following document type declarations in their documents. The DTDs vary in the elements they support.

The **HTML 4.01 Strict DTD** includes all elements and attributes that have not been deprecated or do not appear in frameset documents. For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
```

The **HTML 4.01 Transitional DTD** includes everything in the strict DTD plus deprecated elements and attributes (most of which concern visual presentation). For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
```

The **HTML 4.01 Frameset DTD** includes everything in the transitional DTD plus frames as well. For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML4.01 Frameset//EN"
        "http://www.w3.org/TR/html4/frameset.dtd">
```

3.2 Specifying a HTML Title

Every HTML document must have a TITLE element in the HEAD section.

Authors should use the TITLE element to identify the contents of a document. Since users often consult documents out of context, authors should provide context-rich titles.

```
<title>My first HTML document</title>
```

3.3 Specifying the Character Encoding

Use the "**charset**" parameter of the "**Content-Type**" header field of the **HTTP protocol**.

For example, to specify that the character encoding of the current document is "UTF-8", a document should include the following META declaration:

```
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

META declarations should appear as early as possible in the HEAD element.

3.4 Specify Meta Data for Document Properties Identification

This example refers to a hypothetical profile that defines useful properties for document indexing. The properties defined by this profile -- including "author", "copyright", "keywords", and "date" -- have their values set by subsequent META declarations.

```
<HEAD profile="http://www.acme.com/profiles/core">
 <TITLE>How to complete Memorandum cover sheets</TITLE>
 <META name="author" content="John Doe">
 <META name="copyright" content="&copy; 1997 Acme Corp.">
 <META name="keywords" content="corporate,guidelines,cataloging">
 <META name="date" content="1994-11-06T08:49:37+00:00">
</HEAD>
```

3.5 Specify Meta Data for Search Engines

A common use for META is to specify keywords that a search engine may use to improve the quality of search results. When several META elements provide language-dependent information about a document, search engines may filter on the lang attribute to display search results using the language preferences of the user. For example,

```
<-- For speakers of US English -->
<META name="keywords" lang="en-us"
         content="vacation, Greece, sunshine">
<-- For speakers of British English -->
<META name="keywords" lang="en"
         content="holiday, Greece, sunshine">
```

3.6 Supply a Title for Each Link

The title attribute may be set for both **A** and **LINK** to add information about the nature of a link. This information may be spoken by a user agent, rendered **as a tool tip**, cause a change in cursor image, etc.

```
<BODY>
...some text...
<P>You'll find a lot more in <A href="chapter2.html"
       title="Go to chapter two">chapter two</A>.
<A href="./chapter2.html"
       title="Get chapter two.">chapter two</A>.
See also this <A href="../images/forest.gif"
       title="GIF image of enchanted forest">map of
the enchanted forest.</A>
</BODY>
```

3.7 Don’t use HTML Elements and Attributes to Specify Color

The use of HTML elements and attributes for specifying color is deprecated. You are encouraged to **use style sheets instead**.

Colors specified with the BODY and FONT elements and bgcolor on tables look different on different platforms (e.g., workstations, Macs, Windows, and LCD panels vs. CRTs), so you shouldn't rely entirely on a specific effect.

3.8 Don't Use Color Names, Do Use Hexadecimal Color Values Instead Of

Values like "#FF9999" represent colors as hexadecimal numbers for red, green and blue. The first two characters following the # give the number for red, the next two for green and the last two for blue.

The standard set of color names is: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. Most browsers accept a wider set of color names but use of these is not recommended.

3.9 Avoid Using Large Image Files

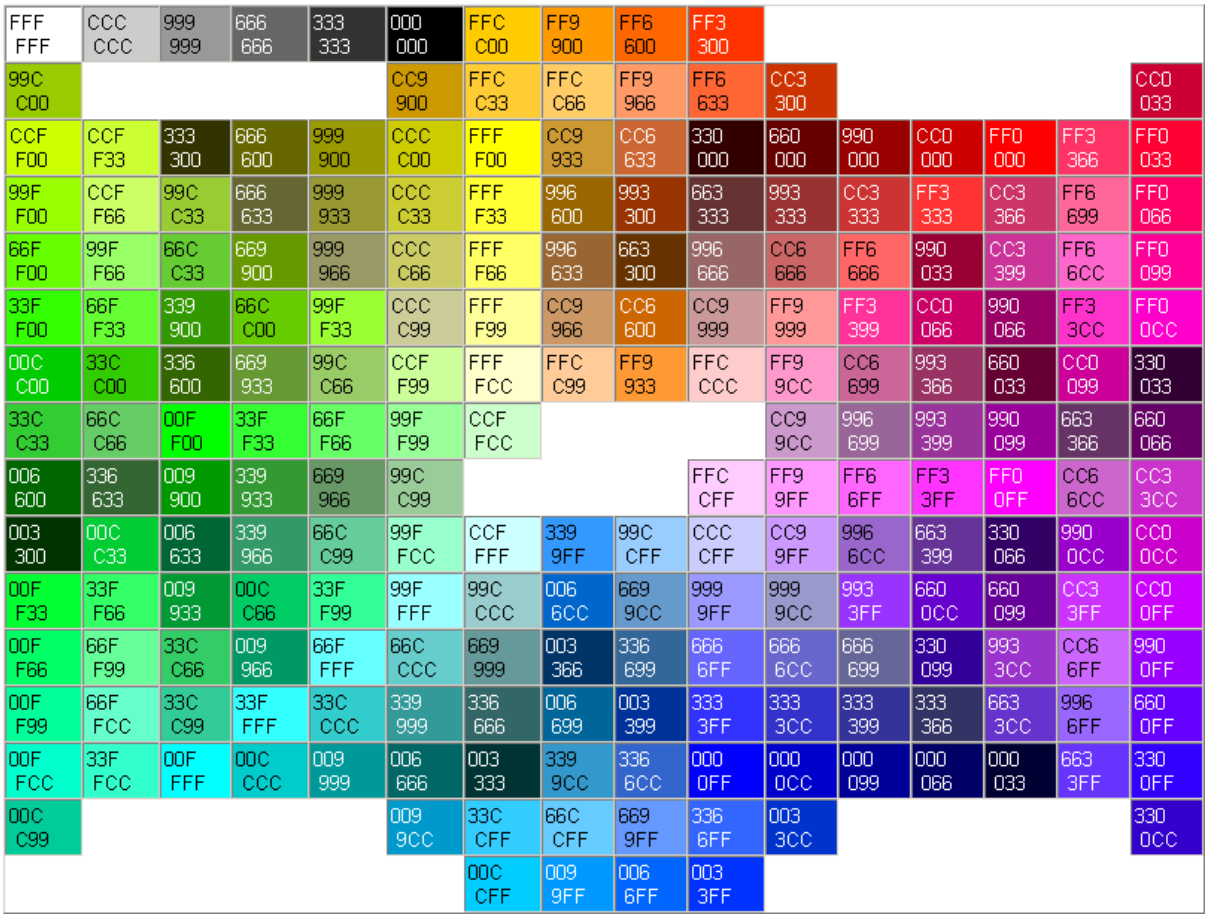
To avoid long delays while the image is downloaded over the network, you should avoid using large image files.

Most browsers understand GIF and JPEG image formats, newer browsers also understand the PNG image format.

Generally speaking, JPEG is best for photographs and other smoothly varying images, while GIF and PNG are good for graphics art involving flat areas of color, lines and text. All three formats support options for progressive rendering where a crude version of the image is sent first and progressively refined.

3.10 Use The Browser Safe Colors

New computers support thousands or millions of colors, but many older color systems can only show up to 256 colors at a time. To cope with this, these browsers make do with colors from a fixed palette. Below is a table of the browser safe colors and their hex values.



3.11 Notes On Length Values

HTML specifies three types of length values for attributes:

- **Pixels**: The value (%Pixels; in the DTD) is an integer that represents the number of pixels of the canvas (screen, paper). Thus, the value "50" means fifty pixels.
- **Length**: The value (%Length; in the DTD) may be either a %Pixel; or a percentage of the available horizontal or vertical space. Thus, the value "50%" means half of the available space.
- **MultiLength**: The value (%MultiLength; in the DTD) may be a %Length; or a relative length. A relative length has the form "i*", where "i" is an integer. When allotting space among elements competing for that space, user agents allot pixel and percentage lengths first, then divide up remaining available space among relative lengths. Each relative length receives a portion of the available space that is proportional to the integer preceding the "*". The value "*" is equivalent to "1*". Thus, if 60 pixels of space are available after the user agent allots pixel and percentage space, and the competing relative lengths are 1*, 2*, and 3*, the 1* will be alloted 10 pixels, the 2* will be alloted 20 pixels, and the 3* will be alloted 30 pixels.

3.12 Using External Style Sheets

Authors may separate style sheets from HTML documents. This offers several benefits:

- Authors and Web site managers may share style sheets across a number of documents (and sites).
- Authors may change the style sheet without requiring modifications to the document.
- User agents may load style sheets selectively (based on media descriptions).

Specify that the style sheet is persistent, preferred, or alternate:

- To make a style sheet persistent, set the rel attribute to "stylesheet" and don't set the title attribute.
- To make a style sheet preferred, set the rel attribute to "stylesheet" and name the style sheet with the title attribute.
- To specify an alternate style sheet, set the rel attribute to "alternate stylesheet" and name the style sheet with the title attribute.

In this example, we first specify a persistent style sheet located in the file mystyle.css:

```
<LINK href="mystyle.css" rel="stylesheet" type="text/css">
```

Setting the title attribute makes this the author's preferred style sheet:

```
<LINK href="mystyle.css" title="compact" rel="stylesheet" type="text/css">
```

Adding the keyword "alternate" to the rel attribute makes it an alternate style sheet:

```
<LINK href="mystyle.css" title="Medium" rel="alternate stylesheet"
type="text/css">
```

In the following example, we specify two alternate style sheets named "compact". If the user selects the "compact" style, the user agent must apply both external style sheets, as well as the persistent "common.css" style sheet. If the user selects the "big print" style, only the alternate style sheet "bigprint.css" and the persistent "common.css" will be applied.

```
<LINK rel="alternate stylesheet" title="compact" href="small-base.css"
type="text/css">
<LINK rel="alternate stylesheet" title="compact" href="small-extras.css"
type="text/css">
<LINK rel="alternate stylesheet" title="big print" href="bigprint.css"
type="text/css">
<LINK rel="stylesheet" href="common.css" type="text/css">
```

3.13 Text Alignment – Don't Use HTML Attribute, Use CSS Instead Of

It is possible to align block elements (tables, images, objects, paragraphs, etc.) on the canvas with the align attribute. Although this attribute may be set for many HTML elements, its range of possible values sometimes differs from element to element. Here we only discuss the meaning of the align attribute for text.

DEPRECATED EXAMPLE:

This example centers a heading on the canvas.

```
<H1 align="center"> How to Carve Wood </H1>
```

Using CSS, for example, you could achieve the same effect as follows:

```
<HEAD>
 <TITLE>How to Carve Wood</TITLE>
 <STYLE type="text/css">
  H1 { text-align: center}
 </STYLE>
<BODY>
 <H1> How to Carve Wood </H1>
```

Note that this would center all H1 declarations. You could reduce the scope of the style by setting the class attribute on the element:

```
<HEAD>
 <TITLE>How to Carve Wood</TITLE>
 <STYLE type="text/css">
  H1.wood {text-align: center}
 </STYLE>
<BODY>
 <H1 class="wood"> How to Carve Wood </H1>
```

DEPRECATED EXAMPLE:

Similarly, to right align a paragraph on the canvas with HTML's align attribute you could have:

```
<P align="right">...Lots of paragraph text...
```

which, with CSS, would be:

```
<HEAD>
 <TITLE>How to Carve Wood</TITLE>
 <STYLE type="text/css">
  P.mypar {text-align: right}
 </STYLE>
<BODY>
 <P class="mypar">...Lots of paragraph text...
```

DEPRECATED EXAMPLE:

To right align a series of paragraphs, group them with the DIV element:

```
<DIV align="right">
 <P>...text in first paragraph...
 <P>...text in second paragraph...
 <P>...text in third paragraph...
</DIV>
```

With CSS, the text-align property is inherited from the parent element, you can therefore use:

```
<HEAD>
 <TITLE>How to Carve Wood</TITLE>
 <STYLE type="text/css">
  DIV.mypars {text-align: right}
 </STYLE>
<BODY>
 <DIV class="mypars">
  <P>...text in first paragraph...
  <P>...text in second paragraph...
  <P>...text in third paragraph...
 </DIV>
```

To center the entire document with CSS:

```
<HEAD>
 <TITLE>How to Carve Wood</TITLE>
 <STYLE type="text/css">
  BODY {text-align: center}
 </STYLE>
<BODY>
 ...the body is centered...
</BODY>
```

3.14 Don't Use FONT and BASEFONT (Font Modifier Elements)

FONT and BASEFONT are deprecated.

DEPRECATED EXAMPLE:

```
<P><font size=1>size=1</font>
<font size=2>size=2</font>
<font size=7>size=7</font>
```

3.15 Frame Target Names

Except for the reserved names listed below, frame target names (%FrameTarget; in the DTD) must begin with an alphabetic character (a-zA-Z). The following target names are reserved and have special meanings.

- **_blank**: The user agent should load the designated document in a new, unnamed window.
- **_self**: The user agent should load the document in the same frame as the element that refers to this target.
- **_parent**: The user agent should load the document into the immediate FRAMESET parent of the current frame. This value is equivalent to _self if the current frame has no parent.
- **_top**: The user agent should load the document into the full, original window (thus canceling all other frames). This value is equivalent to _self if the current frame has no parent.

3.16 Associating Label with HTML Form Controls

Some form controls automatically have labels associated with them (press buttons) while most do not (text fields, checkboxes and radio buttons, and menus). For those controls that have implicit labels, user agents should use the value of the value attribute as the label string. The LABEL element is used to specify labels for controls that do not have implicit labels.

This example creates a table that is used to align two text input controls and their associated labels. Each label is associated explicitly with one text input:

```
<FORM action="..." method="post">
<TABLE>
  <TR>
    <TD><LABEL for="fname">First Name</LABEL>
    <TD><INPUT type="text" name="firstname" id="fname">
  <TR>
    <TD><LABEL for="lname">Last Name</LABEL>
    <TD><INPUT type="text" name="lastname" id="lname">
</TABLE>
</FORM>
```

3.17 Adding Structure to Forms: the FIELDSET and LEGEND Elements

The FIELDSET element allows authors to group thematically related controls and labels. Grouping controls makes it easier for users to understand their purpose while simultaneously facilitating tabbing navigation for visual user agents and speech navigation for speech-oriented user agents. The proper use of this element makes documents more accessible.

The LEGEND element allows authors to assign a caption to a FIELDSET. The legend improves accessibility when the FIELDSET is rendered non-visually.

In this example, we create a form that one might fill out at the doctor's office. It is divided into three sections: personal information, medical history, and current medication. Each section contains controls for inputting the appropriate information.

```
<FORM action="..." method="post">
 <P>
 <FIELDSET>
  <LEGEND>Personal Information</LEGEND>
  Last Name: <INPUT name="personal_lastname" type="text" tabindex="1">
  First Name: <INPUT name="personal_firstname" type="text" tabindex="2">
  Address: <INPUT name="personal_address" type="text" tabindex="3">
  ...more personal information...
 </FIELDSET>
 <FIELDSET>
  <LEGEND>Medical History</LEGEND>
  <INPUT name="history_illness"
         type="checkbox"
```

```
        value="Smallpox" tabindex="20"> Smallpox
  <INPUT name="history_illness"
        type="checkbox"
        value="Mumps" tabindex="21"> Mumps
  <INPUT name="history_illness"
        type="checkbox"
        value="Dizziness" tabindex="22"> Dizziness
  <INPUT name="history_illness"
        type="checkbox"
        value="Sneezing" tabindex="23"> Sneezing
  ...more medical history...
 </FIELDSET>
 <FIELDSET>
  <LEGEND>Current Medication</LEGEND>
  Are you currently taking any medication?
  <INPUT name="medication_now"
        type="radio"
        value="Yes" tabindex="35">Yes
  <INPUT name="medication_now"
        type="radio"
        value="No" tabindex="35">No
  If you are currently taking medication, please indicate
  it in the space below:
  <TEXTAREA name="current_medication"
          rows="20" cols="50"
          tabindex="40">
  </TEXTAREA>
 </FIELDSET>
</FORM>
```

3.18 Tabbing Navigation

The tabbing order defines the order in which elements will receive focus when navigated by the user via the keyboard. The tabbing order may include elements nested within other elements.

In this example, the tabbing order will be the BUTTON, the INPUT elements in order (note that "field1" and the button share the same tabindex, but "field1" appears later in the character stream), and finally the link created by the A element.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<TITLE>A document with FORM</TITLE>
</HEAD>
<BODY>
...some text...
```

```
<P>Go to the
<A tabindex="10" href="http://www.w3.org/">W3C Web site.</A>
...some more...
<BUTTON type="button" name="get-database"
           tabindex="1" onclick="get-database">
Get the current database.
</BUTTON>
...some more...
<FORM action="..." method="post">
<P>
<INPUT tabindex="1" type="text" name="field1">
<INPUT tabindex="2" type="text" name="field2">
<INPUT tabindex="3" type="submit" name="submit">
</P>
</FORM>
</BODY>
</HTML>
```

3.19 Hiding Style And Script Data From User Agents

Some style sheet languages support syntax intended to allow authors to hide the content of STYLE elements from non-conforming user agents. This example illustrates for CSS how to comment out the content of STYLE elements to ensure that older, non-conforming user agents will not render them as text.

```
<STYLE type="text/css">
<!--
   H1 { color: red }
   P  { color: blue}
   -->
</STYLE>
```

Some scripting engines, including those for languages JavaScript, VBScript, and Tcl allow the script statements to be enclosed in an SGML comment.

Another solution to the problem is to keep scripts in external documents and refer to them with the src attribute.

Commenting scripts in JavaScript

The JavaScript engine allows the string "<!--" to occur at the start of a SCRIPT element, and ignores further characters until the end of the line. JavaScript interprets "//" as starting a comment extending to the end of the current line. This is needed to hide the string "-->" from the JavaScript parser.

```
<SCRIPT type="text/javascript">
<!--  to hide script contents from old browsers
  function square(i) {
```

```
    document.write("The call passed ", i ," to the function.","<BR>")
    return i * i
  }
  document.write("The function returned ",square(5),".")
// end hiding contents from old browsers  -->
</SCRIPT>
```

Commenting scripts in VBScript

In VBScript, a single quote character causes the rest of the current line to be treated as a comment. It can therefore be used to hide the string "-->" from VBScript, for instance:

```
<SCRIPT type="text/vbscript">
  <!--
    Sub foo()
     ...
    End Sub
  ' -->
 </SCRIPT>
```

Commenting scripts in TCL

In Tcl, the "#" character comments out the rest of the line:

```
<SCRIPT type="text/tcl">
<!--  to hide script contents from old browsers
  proc square {i} {
    document write "The call passed $i to the function.<BR>"
    return [expr $i * $i]
  }
  document write "The function returned [square 5]."
# end hiding contents from old browsers  -->
</SCRIPT>
```

3.20 Designing Documents For User Agents That Don't Support Scripting

The NOSCRIPT element allows authors to provide alternate content when a script is not executed.

In the following example, a user agent that executes the SCRIPT will include some dynamically created data in the document. If the user agent doesn't support scripts, the user may still retrieve the data through a link.

```
<SCRIPT type="text/tcl">
 ...some Tcl script to insert data...
</SCRIPT>
<NOSCRIPT>
 <P>Access the <A href="http://someplace.com/data">data.</A>
</NOSCRIPT>
```

3.21 Indenting Text For A Better Layout

Indenting the content of elements **makes the markup easier to read**.

Avoid indenting the content of TITLE, P and LI elements:

3.22 Validate Your HTML Documents

To ensure that a document can be successfully read by any browser, make sure it adheres to the syntax rules for HTML. You don't need to do this by hand, that's what computers are for. Several free tools exist to check HTML for syntax errors to W3C Recommendations and other standards.

W3C Markup Validation Service
http://validator.w3.org/

W3C CSS Validation Service
http://jigsaw.w3.org/css-validator/

3.23 Test the Documents

Even when the document passes validation, it will not necessarily work as you intended on all platforms. Several programs are available tho check documents for stylistic problems. Even a syntactically valid document can be hard to read because of things like forgotten alternative texts for images, unwanted whitespace, deeply nested lists or non-hierarchical use of headers.

4 HOW-TO

4.1 How to Solve the Empty Cells Problem?

One quirk (see the below illustration) is the way browsers deal with empty cells like

```
<td></td>
```

Year	Sales
2000	$18M
2001	$25M
2002	$36M
2003	

To prevent this, include a non-breaking space or a 1x1 pixel image:

```
<td> </td>
```

4.2 How to use entities for special characters?

For copyright notices, or trademarks it is customary to include the appropriate signs:

Symbol	Entity	Example
Copyright sign	©	Copyright © 1999 W3C
Registered trademark	®	FPT-SOFTWARE ®
Trademark	™	FPT-ELEAD™

Note HTML 4.0 defines ™ for the trademark sign but this is not yet as widely supported as ™

There are a number of other entities you may find useful:

Symbol	Entity	Example
Less than	<	<
Greater than	>	>
Ampersand	&	&
nonbreaking space		
em dash	—	—
quotation mark	"	"

4.3 How to Introduce Non-breaking Spaces?

Browsers automatically wrap text to fit within the margins. Line breaks can be introduced wherever space characters appear in the markup. Sometimes you will want to prevent the browser from wrapping text between two particular words. For instance between two words in

a brand name such as "Coca Cola". The trick is to use the ** entity (or)** in place of the space character, for example:

```
Sweetened carbonated beverages, such as Coca Cola,
```

have attained world-wide popularity.

5 APPENDIX

5.1 Some Notes on HTML Structure

- HTML includes **element types** that represent paragraphs, hypertext links, lists, tables, images, etc.
- **Element** names are always **case-insensitive**.
- **Attribute** names are always **case-insensitive**.
- Each element type declaration generally describes three parts: a **start tag**, **content**, and an **end tag**.
- **Elements are not tags**. Some people refer to elements as tags (e.g., "the P tag"). Remember that the element is one thing, and the tag (be it start or end tag) is another. For instance, the HEAD element is always present, even though both start and end HEAD tags may be missing in the markup.
- Elements may have associated properties, called attributes, which may have values (by default, or set by authors or scripts). Attribute/value pairs appear before the final ">" of an element's start tag.
- By default, SGML requires that all attribute values be delimited using **either double quotation marks** (ASCII decimal 34) **or single quotation marks** (ASCII decimal 39). Single quote marks can be included within the attribute value when the value is delimited by double quote marks, and vice versa. Authors may also use numeric character references to represent double quotes (") and single quotes (').

5.2 HTML Comments Syntax

```
<!-- this is a comment -->
<!-- and so is this one,
    which occupies more than one line -->
```

White space is not permitted between the markup declaration open delimiter("<!") and the comment open delimiter ("--"), but is permitted between the comment close delimiter ("--") and the markup declaration close delimiter (">"). A common error is to include a string of hyphens ("---") within a comment. Authors should avoid putting two or more adjacent hyphens inside comments.

Approver	Reviewer	Creator
Nguyen Lam Phuong	**Dang Quang Dinh**	**Dang Sy Hai**