



# WebEdit Help

Welcome to **Kenn Nesbitt's WebEdit(TM)**, the easiest and most powerful way to create pages for the World Wide Web!

This help file is divided into the following six sections:

## About WebEdit

General information about WebEdit and Nesbitt Software.

## Getting Started With WebEdit

Who WebEdit is designed for and how to create simple documents

## WebEdit HTML Reference

Information about HTML, tags, attributes, and the various versions of HTML

## WebEdit General Reference

Help on dialog boxes, menu selections, buttons, etc.

## Frequently Asked Questions

Answers to common questions

## Learning HTML

Finding out more about Hypertext Markup Language

# Introduction

## What is WebEdit?

WebEdit is a Windows-based text editor specifically designed to ease the editing of hypertext markup language (HTML) documents.

If you are not familiar with HTML, there are many places to learn the basics. For example, your local bookstore should carry quite a few HTML books. In addition, there are many tutorial and reference documents available on the web that can get you started quickly. Some of these web sites are listed in this Help file in the topic [What is HTML?](#)

WebEdit strives to be the best Windows-based HTML editor available. Specifically, we have tried to include support for every feature of every version of the HTML specification, from HTML version 1 through HTML 3.2, including optional features and special non-standard extensions supported by browsers such as Netscape and Internet Explorer.

Note: HTML 3.x is a moving target. However, as the standard evolves, we will release minor upgrades to WebEdit that support the current syntax. Additionally, you can add any elements you like with WebEdit's [User-Defined Tags](#) dialog.

Moreover, WebEdit tries to make all of these features available in a consistent, well-organized fashion, with a minimum number of keystrokes, allowing you to create HTML documents as rapidly as possible.

## [WebEdit features](#)

## Design Intentions

WebEdit is a professional tool, designed for those seeking to:

- accelerate the process of document creation
- support the most current HTML specification
- advance to an expert level in Web publishing

In addition, if you are new to the Web, you will find that WebEdit can help you learn HTML quickly and proficiently. By automatically generating much of your HTML for you, WebEdit can "show you how it's done."

## Creating an HTML Document with WebEdit

One obvious benefit of WebEdit for those just getting started is its ability to create starter documents with the click of a button. A good minimal HTML document looks like this:

```
<HEAD>
<TITLE>Page Title</TITLE>
</HEAD>

<BODY>
<H1>First Heading</H1>
Add body text here.
<HR>
Last Updated: Friday, June 16, 1996
</BODY>

</HTML>
```

Keep in mind that this help file includes an HTML reference listing each HTML

command along with a short description, so whenever you see a command that is not familiar (like most of the above example, probably) you can find out what it does by looking up the relevant section in the help. As a reminder, remember that you can cut & paste from the examples included with this help file.

### **The Add Minimal HTML Button**

To create the above document with WebEdit click on the Add Minimal HTML button (looks like a big '+'). Add some text to the body of the document and save the file. It will be saved with an .html extension. Without the .html extension your browser would present it exactly as it appears above; the .html extension tells the browser to follow the HTML commands embedded in the document. Using your browser, open the file to see the formatted document.

If you prefer, you can edit the default minimal HTML document by editing the file MINIMAL.HTML located in the WebEdit directory.

# Tag Index

## HTML 0

<A>	<u><a href="#">Anchor</a></u>
<ADDRESS>	<u><a href="#">Address</a></u>
<BASE>	<u><a href="#">Base</a></u>
<BLOCKQUOTE>	<u><a href="#">Block Quote</a></u>
<BODY>	<u><a href="#">Body</a></u>
 	<u><a href="#">Line Break</a></u>
<DD>	<u><a href="#">Definition Item</a></u>
<DIR>	<u><a href="#">Directory List</a></u>
<DL>	<u><a href="#">Definition List</a></u>
<DT>	<u><a href="#">Definition Title</a></u>
<H>	<u><a href="#">Headings</a></u>
<HEAD>	<u><a href="#">Head</a></u>
<HR>	<u><a href="#">Horizontal Rule</a></u>
<HTML>	<u><a href="#">HTML</a></u>
<IMG>	<u><a href="#">Image</a></u>
<ISINDEX>	<u><a href="#">Is Index</a></u>
<LI>	<u><a href="#">List Item</a></u>
<LINK>	<u><a href="#">Links</a></u>
<MENU>	<u><a href="#">Menu List</a></u>
<META>	<u><a href="#">Meta</a></u>
<NEXTID>	<u><a href="#">Next ID</a></u>
<OL>	<u><a href="#">Ordered List</a></u>
<P>	<u><a href="#">Paragraph</a></u>
<PRE>	<u><a href="#">Preformatted</a></u>
<TITLE>	<u><a href="#">Title</a></u>
<UL>	<u><a href="#">UnOrdered List</a></u>
<!-- -->	<u><a href="#">Comment</a></u>
<!DOCTYPE>	<u><a href="#">Document Type</a></u>

## HTML 1

<BOLD>	<u><a href="#">Bold</a></u>
<CITE>	<u><a href="#">Citation</a></u>
<CODE>	<u><a href="#">Code</a></u>
<EM>	<u><a href="#">Emphasis</a></u>
<I>	<u><a href="#">Italic</a></u>
<KBD>	<u><a href="#">Keyboard</a></u>
<SAMP>	<u><a href="#">Sample</a></u>
<STRONG>	<u><a href="#">Strong Emphasis</a></u>
<TT>	<u><a href="#">Typewriter Text</a></u>
<VAR>	<u><a href="#">Variable</a></u>

## HTML 2

<FORM>	<u><a href="#">Form</a></u>
<INPUT>	<u><a href="#">Input</a></u>
<OPTION>	<u><a href="#">Option</a></u>
<P>	<u><a href="#">Paragraph</a></u>

<SELECT>  
<TEXTAREA>

Select  
Text Area

## HTML 3

<ABBREV>  
<ACRONYM>  
<ARG>  
<BYLINE>  
<DFN>  
<LIT>  
<PERSON>  
<Q>  
<RANGE>  
<SPOT>  
<STYLE>  
<SUB>  
<SUP>  
<TAB>

Abbreviation  
Acronym  
Argument  
Byline  
Defining Instance  
Literal  
Person  
Inline Quote  
Range  
Spot  
Style  
Subscript  
Superscript  
Tab

## HTML 3.2

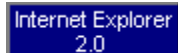
<APPLET>  
<BIG>  
<CAPTION>  
<CENTER>  
<DIV>  
<FRAME>  
<FRAMESET>  
<MAP>  
<NOFRAMES>  
<PARAM>  
<SMALL>  
<STRIKE>  
<TABLE>  
<TH>  
<TD>  
<TR>  
<U>

Applet  
Big  
Table Caption  
Center  
Division  
Frame  
Frame Set  
Client Side Mapping  
No Frames  
Param  
Small  
Strikethrough  
Table  
Table Header  
Table Data  
Table Row  
Underline



<BASEFONT>  
<BLINK>  
<FONT>  
<NOBR>  
<WBR>

Base Font  
Blink  
Font  
No Break  
Word Break



<BGSOUND>

Background Sound

<MARQUEE>

Marquee



<ALIAS>  
<OBJECT>  
<PARAM>

Alias  
Object  
Param

## **About WebEdit**

Webedit is your tool to create wonderful World Wide Web Pages. It allows you to directly manipulate all aspects of the HyperText Markup Language (HTML) on which Web pages are based. When used in combination with Web browsers like Netscape and Mosaic you'll be making professional (or just fun!) looking Web pages in no time!

WebEdit provides many features for both novice and experienced Web authors. For those just getting into the waters of Web page creation WebEdit offers easy to use buttons and menus to shield you from the many complexities of HTML. For experienced authors WebEdit will give you easy access to the most in depth aspects of HTML.

WebEdit grows with you. WebEdit allows you to create and save your own HTML tags, or add newly defined tags. WebEdit keeps you on the cutting edge!

[About the Authors](#)

[WebEdit Features](#)

[System Requirements](#)

[Ordering WebEdit](#)

[Upgrading WebEdit](#)

[Where to Find Us](#)

[License Agreement and Copyright Information](#)

## About the Authors



Kenn Nesbitt is President of Nesbitt Software Corporation, developers of WebEdit. Formerly with Microsoft Consulting Services, Kenn has specialized in applications for Microsoft Windows and the Internet. He is a Contributing Editor to *Internet Advisor* magazine and *Data Based Advisor* magazine. He has written more than 150 articles for computer magazines all over the world, presented at many database and application development conferences, and is co-author of the book *Power Shortcuts...Paradox for Windows* published by MIS Press.



Scot is our hotshot programmer who thinks he can't possibly be a hotshot programmer since he just graduated college. These days, most of the actual programming of WebEdit is done by Scot. He has recently finished up his BA/BS degree in Computer Science at Western Oregon State College. His hobbies include driving his Miata with the top down (hard to do in Oregon!), collecting old videogame systems, and programming in Delphi (C++? Yuck!). Before joining Nesbitt Software he served six years in the United States Marine Corps spending time in such sunny locales as California, Hawaii, and Saudi Arabia. Before that he met his wife Bobbi (10 years now!) who is far too lenient on letting him have his toys.... His latest faves are playing games on the Playstation, coding to Juliana Hatfield, Rush, Patty Smyth and Suzanne Vega, and getting his private pilots license. Now if he only could find enough free time to have a life!





Jay is our catch-all guy who seems to inherit all of the programming tasks that nobody else wants. In a past life he served four years with the United States Marine Corps and worked as a programmer for the Bureau of Land Management before joining the corporate world at Intel Corporation. After a short stint with technical support, Jay is now pursuing his first love: coding. In his free time, Jay enjoys playing basketball, thinking of ways to prolong learning OLE, and riding his mountain bike in Forest Park.

# WebEdit features

## WebEdit offers the following features:

- **Unlimited number and size of documents (32-bit versions only)** - WebEdit's rich interface lets you work on an unlimited number of HTML documents simultaneously. The size of each document is limited only by your computer's memory.
- **HTML Dialogs** - WebEdit has specially designed dialog boxes for defining anchors and links, inline images and figures, forms, tables, etc., including a URL Builder for rapid creation of Uniform Resource Locators. Additionally, WebEdit saves every URL you enter, letting you choose from a list rather than retyping the same URLs over and over.
- **Quick Previewer** - WebEdit's quick preview utility is a quick and easy way to preview your HTML documents. It supports all HTML tags up to and including HTML 2.0. It also supports HTML 3.0 Forms and Tables.
- **Browser Support** - Whatever Windows-based browser you use, you can link it into WebEdit to test your HTML documents at the click of a button. WebEdit allows you to instantly choose from any number of browsers.
- **Home Page Wizard** - Helps you create your first page in a flash!
- **Modules (Professional version only)** - WebEdit supports third party module plugins via an API. Purchase modules or write your own!.
  - HTML tag checker
  - WYSIWYG frame wizard
  - WYSIWYG form designer
  - FTP upload facility
  - Link validation wizard
  - Table of contents wizard
- **HTML 3.2, Netscape 3.0 and Internet Explorer 3.0 tags** - WebEdit supports all the latest tags keeping your Web pages on the cutting edge!
- **Document Structure Elements** - Document structure tags such as <HTML>, <HEAD>, <TITLE>, <BODY>, etc. are all supported, including HTML 3 and Netscape-specific attributes.
- **Block Formatting Elements** - WebEdit supports all HTML block formatting tags, such as <ADDRESS>, <BLOCKQUOTE>, and <PRE>, plus HTML 3 extensions such as <NOTE>.
- **Logical Font Formatting Elements** - Every logical formatting tag and attribute from HTML version 1 through HTML version 3.2 is included, from standard tags such as code and citation, to new proposed tags such as person, acronym, etc.
- **Physical Font Formatting Elements** - Blinking text, bold, italics, underlining, typewriter text, emphasis, strong emphasis, font sizing, etc. It's all in here.
- **List and Miscellaneous Elements** - Select a block of text and choose Numbered List or Un-numbered List, and WebEdit automatically inserts list-item tags on each

line. WebEdit also includes all standard insertion tags such as <P>, <HR>, <BR>, etc.

- **Special Characters** - WebEdit includes support for the entire ISO Latin character set, allowing you to easily insert extended ANSI characters into your documents. WebEdit also includes support for special characters such as "<", ">", "&", non-breaking spaces, and more.
- **Form Elements** - Create web forms quickly and easily with WebEdit's built-in Form support.
- **Table Elements** - HTML 3.2 defines a new syntax for displaying tabular information. Netscape now supports this syntax and even extends it.
- **WYSIWYG table builder (Standard and Professional versions only)** - To simplify the creation of HTML tables, WebEdit also includes a WYSIWYG table builder; you simply enter your data in a spreadsheet-style grid, and WebEdit writes the HTML for you.
- **Client-Side image mapping (Standard and Professional versions only)** - WebEdit's Map Builder allows you to visually divide images into different hotspots.
- **User-Defined Elements (Standard and Professional versions only)** - If there are any HTML tags or other text you enter regularly that are not already built-into WebEdit, you can add them to WebEdit's User-Defined Tags dialog box for easy insertion into your documents. You can even add filenames into the dialog box for larger "insertion macros". Selecting a filename in the User-Defined Tags dialog tells WebEdit to insert the contents of the file into the current document.
- **Non-standard tags and attributes** - WebEdit includes support for non-standard tags and attributes, such as those recognized by Netscape (font sizing, special image alignment, etc.) and HTML 3.2. These tags and attributes are ignored by most browsers, but are included in WebEdit in case you need them.
- **HTML Removal** - WebEdit lets you quickly and easily remove HTML tags from any document or portion of a document. Simply highlight the text from which to remove HTML tags, and click the Remove HTML Tags button.
- **Spell Checker (Standard and Professional versions only)** - Correct the spelling of your documents directly within WebEdit using our built-in spell checker.
- **Document Export (Standard and Professional versions only)** - WebEdit allows you to save your HTML documents in either UNIX or Macintosh format. No more strange characters when using different platforms!
- **Shortcut Keys** - We have provided shortcut keys (e.g., Ctrl-B for Boldface) for all of the most common tags, allowing you to enter HTML codes in your documents as quickly as possible.
- **Configurable Toolbars (Standard and Professional versions only)** - Configure WebEdit to work the way you do! Put whatever buttons you want, in any order, on the toolbar by dragging and dropping.
- **Project Support (Professional version only)** - Organize all of your HTML files into projects and then load all of those files with the click of a button!

- **Tooltips** - Every field on every dialog has popup tooltips that provide a brief explanation of the purpose or use of the field, reducing the amount of time you will spend looking things up.
- **Right-Click Menu** - Right-clicking on any document pops up a menu that allows you to close or save the file, or choose from a list of the most common HTML tags to insert in your document.

## How to Order WebEdit

If you are using the evaluation version of Kenn Nesbitt's WebEdit, you must purchase a license to continue using it beyond the 30-day evaluation period.

After your order has been processed, you will receive a software key to "unlock" the software and eliminate the "nag" screen. Registering will also give you the following benefits:

- Priority online technical support
- Automatic email notification of product updates
- Free minor updates
- Discounts on major upgrades

You may order WebEdit from any of the following distributors:

United States

Compuserve

Germany, Austria and Switzerland

Italy

Japan

Netherlands

Sweden

United Kingdom

If you are using the Standard version of WebEdit you should look at the advantages that upgrading to the Professional version gives you.

## WebEdit Professional Upgrade Information

As a WebEdit 2.0 user, you are probably already aware of WebEdit's many benefits, including its well-designed and easy-to-use interface, numerous web creation tools and great online support. What you may not yet be aware of are the benefits of moving up from WebEdit 2.0 to the commercially available WebEdit PRO.

WebEdit PRO has an enhanced feature set that goes well beyond the capabilities of WebEdit 2.0, including the following:

WebEdit PRO contains all the features of WebEdit 2.0 plus:

- Multi-file project support
- Syntax highlighting
- HTML tag syntax checker
- WYSIWYG frame wizard
- WYSIWYG form designer
- FTP upload facility
- Link validation wizard
- Table of contents wizard
- Third party add-in module support
- Sample source files and templates
- Images, graphics, sounds and multimedia files
- and more

Whether it's for business or just for fun, WebEdit includes the tools you need to create exciting and dynamic web sites.

We've priced WebEdit PRO to be affordable to everyone, so it's definitely an investment worth making. WebEdit PRO is just \$129.95 (SRP). The software only price when downloading from our website is just \$109.95 ( manual and disks are extra). In addition, volume licensing is available for all size organizations. Place your order now, or visit your local software retailer. If you wish to try it before you buy it, download a 30-day trial version of WebEdit PRO from our web site at: <http://www.nesbitt.com>

For more information visit our web site or call (800) 582-4022.

## System Requirements

WebEdit is a very efficient program - if your computer can run Windows it should run WebEdit just fine. WebEdit requires approximately 3 Mb of disk space.

WebEdit 2.0 comes in both 16 and 32 bit versions. The 16-bit version runs under Windows 3.1, Windows for Workgroups, and Windows 95. The 32-bit version runs under Windows 95 and Windows NT.

If you use WebEdit on any other Operating System please drop us a line and we'll update this list!

A list of known compatibility issues and other problems is posted at <http://www.nesbitt.com/bugs.html>.

## United States

### **Link sandiego.com, Inc.**

The English-language version of Kenn Nesbitt's WebEdit is distributed worldwide by Link sandiego.com, Inc. (formerly Data Transfer Group).

### **Link sandiego.com, Inc.**

2251 San Diego Avenue, Suite A-141  
San Diego, CA 92110  
USA

**Voice:** +1 (619) 220-8601

**Fax:** +1 (619) 220-8324

**Email:** [orderdesk@thegroup.net](mailto:orderdesk@thegroup.net)

**Web:** <http://www.nesbitt.com/>

For current versions and pricing, or to place an order, please see our online order form at:

**Online order form:** <http://www.nesbitt.com/order.html>



# CompuServe

## CompuServe

You may order Kenn Nesbitt's WebEdit online on CompuServe in the Shareware Registration forum (GO SWREG).

For current versions follow these instructions:

1. GO SWREG
2. Choose "Register Shareware"
3. Agree to the Registration Agreement
4. Choose your Geographic Region (e.g., United States)
5. Search for the Keyword "WebEdit"

If you have any questions or difficulty using SWREG to order WebEdit, please send email to 76100,57.

## Germany, Austria and Switzerland

### Michael Kalus Soft- und Hardwarehandel

Die Deutsche Version von Kenn Nesbitt's WebEdit wird von Michael Kalus Soft- und Hardwarehandel vertrieben. Für mehr Informationen wenden Sie sich bitte an:

Michael Kalus  
Soft- und Hardwarehandel  
Reinsburgstr. 44  
70178 Stuttgart  
GERMANY

**Voice:** 0711-618022 / International: +49-711-618022

**Fax:** 0711-618031 / International: +49-711-618031

**Email:** [webedit@aol.com](mailto:webedit@aol.com)

**CompuServe:** 100265,3065

**Preis:** DM 139,- DM inkl. MwSt. für die Deutsche Vollversion.  
DM 39,- DM inkl. MwSt. für das Update von einer Englischen WebEdit Version auf die Deutsche.  
DM 69,- DM inkl. MwSt. für Studenten, Schüler, Schulen und Bildungseinrichtungen (Nachweis erforderlich).  
Für Informationen über Netzwerklizenzen und Mengen Discount, rufen Sie bitte an oder schicken Sie eine E-Mail.

Michael Kalus Soft- und Hardwarehandel akzeptiert die VISA Karte, oder eine Überweisung auf folgendes Konto:

Michael Kalus  
Soft- und Hardwarehandel  
Kontonr.: 1267567  
BLZ: 600 501 01  
Bei: Landesgirokasse Stuttgart

Bitte schicken Sie uns eine E-Mail, wenn Sie die Überweisung gemacht haben, damit wir wissen wohin wir den Key schicken müssen.

### CompuServe

Sie können die Deutsche Version von WebEdit auch Online über CompuServe bestellen. Dazu gehen Sie bitte SWREG und wählen die folgenden IDs aus:

	Email:	100265,3065
	Deutsche Version:	US\$99.00 (Registration ID #9200)
	Deutsche Schulversion:	US\$49.00 (Registration ID #9201)
	Update auf die Deutsche Version:	US\$28.00 (Registration ID #9202)
	WebEdit 2.0 Pro Deutsch:	US\$120.00 (Registration ID #11548)
	WebEdit 2.0 Standard Deutsch:	US\$99.00 (Registration ID#11549)
	Update von 1.x Deutsch auf 2.0 Pro Deutsch:	US\$35.00 (Registration ID #11550)
	Update von 1.x Englisch auf 2.0 Pro Deutsch:	US\$64.00 (Registration ID#11551)

# Italy

## **Studio Sikorsky**

The English language version of WebEdit is available from Studio Sikorsky in Italy.

For more information, please contact:

### **Studio Sikorsky**

Attn. Eng. J. Sikorsky  
Via S. Antonio 2/B - Fossoli  
41010 Carpi (Mo)  
ITALY

**Tel:** +39,59,669287

**Fax:** +39,59,660838

**Email:** [webedit@studios.it](mailto:webedit@studios.it)

**Web:** <http://www.studios.it/webedit/>

# Japan

The Japanese version of Kenn Nesbitt's WebEdit is now available. For more information, please contact:

**Personal Data Factory**

Shimoueki-cho 451-3

Iseaki-shi

Gunma-ken 372

JAPAN

**Voice:** +81 0270-26-1513

**Fax:** +81 0270-26-1636

**Email:** [101153.21@compuserve.com](mailto:101153.21@compuserve.com)

## Netherlands

Kenn Nesbitt's Webedit kan in Nederland besteld worden via:

### **Solution Consultancy**

Postbus 127  
8252 AC Dronten  
The Netherlands

**Voice:** 0321-381121 / International: +31-321-381121  
**Fax:** 0321-380698 / International: +31-321-380698  
**Email:** [peterv@solcon.nl](mailto:peterv@solcon.nl)  
**CompuServe:** 100103,3500  
**Excalibur BBS:** 0321-381560 / Internet: @solcon.nl  
**WEB:** <http://www.solcon.nl>  
**Banknummer:** 40.78.98.581  
**Gironummer:** 7215282

### **Prices:**

### **Alleen software**

WebEdit 2.0 (Win 3.1,win95)		f	87,45	f	69,95*
WebEdit 2.0 Handleiding	f	14,00			
WebEdit 2.0 Disk Set (Incl. handleiding)	f	17,50			
WebEdit PRO	f	227,45	f	192,45*	
WebEdit PRO Handleiding	f	26,25			
WebEdit PRO Disk Set (Incl. handleiding)	f	35,00			

Bovengenoemde prijzen zijn excl. 17,5 % btw.

\*Alleen de software. De software kan worden gedownload van onze website <http://www.solcon.nl> of via ons Excalibur BBS.

Voor Informatie over netwerklicenties of kwantumkorting kunt u direct contact met ons opnemen.

Hoe kan ik WebEdit registreren?

1) Met uw creditcard:

Solution Consultancy accepteert VISA, AMERICAN EXPRESS, MASTERCARD en DINERSCLUB. Via onze website <http://www.solcon.nl> kunt u een bestelformulier uitprinten. Vul dit formulier in en zend het per fax naar: 0321-380698. U kunt dit bestelformulier ook telefonisch aanvragen 0321-381121. Wij zenden het formulier dan per fax of per post naar u toe. Na ontvangst van het bestelformulier ontvangt u van ons per omgaande uw unlock code voor WebEdit. On-line registratie, gebruik makend van uw creditcard, is ook mogelijk via ons Excalibur BBS: 0321-381560, internet: @solcon.nl.

2) Via incassomachtiging:

Print het incasso/bestelformulier uit dat u kunt vinden op onze website <http://www.solcon.nl> of ons Excalibur bbs: 0321-381560. Vul dit formulier in en zend het per fax naar: 0321-380698. U kunt het bestelformulier ook telefonisch aanvragen 0321-381121. Wij zenden het formulier dan per fax of per post naar u toe. Na ontvangst van het bestelformulier ontvangt u van ons per omgaande uw unlock code voor WebEdit.

Customers inside the European Union please add 17,5 % VAT to get your purchase price.

## United Kingdom

### **Grey Matter Ltd.**

The English-language version of Kenn Nesbitt's Webedit is distributed in the UK and Europe by:

### **Grey Matter Ltd.**

Prigg Meadow, Ashburton  
Devon TQ13 7DF  
ENGLAND

**Voice:** 01364 654100 / International +44 1364 654100

**Fax:** 01364 654200 / International +44 1364 654200

**Email:** [maildesk@greymatter.co.uk](mailto:maildesk@greymatter.co.uk)

For current pricing information please email [maildesk@greymatter.co.uk](mailto:maildesk@greymatter.co.uk), putting in the subject line "PLEASE send WebEdit pricing information"

E.G.

To: [maildesk@greymatter.co.uk](mailto:maildesk@greymatter.co.uk)

Subject: PLEASE send WebEdit pricing information

To find the best UK mirror sites for Nesbitt Software Corporation, please refer to:  
<http://www.nesbitt.com>

For UK-oriented "New to HTML" information, try: \_  
<http://www.demon.co.uk/dita/new2html.html>

# License Agreement and Copyright Information

## Copyright Notice

Kenn Nesbitt's WebEdit(TM) is Copyright © 1996, SmartDesk, Inc. All Rights Reserved. Portions Copyright © 1996, Nesbitt Software Corporation. WebEdit is a Trademark of Nesbitt Software Corporation. The Sentry Spelling-Checker Engine Copyright © 1993 Wintertree Software Inc.

## License Agreement

**GRANT.** Subject to the provisions contained herein, SmartDesk Inc., (herein "SmartDesk") hereby grants you a non-exclusive 30-day license to use its accompanying proprietary software ("Software"), described as Kenn Nesbitt's WebEdit, free of charge for the sole purpose of evaluating whether to purchase an ongoing license to the Software.

You may evaluate the software for not more than 30 days. At the end of 30 days you must purchase a license in order to continue using the Software. If you do not fit within the description above, a license fee is due to SmartDesk and no license is granted herein. If you are using a 'Lite' or evaluation version of the Software, you will not be entitled to technical support.

**SOFTWARE AND DOCUMENTATION.** SmartDesk shall furnish the Software to you electronically or on media in machine-readable object code form. If you receive your first copy of the Software electronically, and a second copy on media, the second copy may be used for backup and archive purposes only. This license does not grant you any right to any enhancement or update to the Software and Documentation. Enhancements and updates, if available, may be obtained by you at SmartDesk's then-current standard pricing, terms, and conditions.

**RESTRICTED USE.** You may not lend, rent, lease or otherwise transfer the Software. The Software is protected by the copyright laws of the United States and international copyright treaties. This license is valid for only one user on only one computer.

**TITLE.** Title, ownership rights, and intellectual property rights in and to the Software and Documentation shall remain in SmartDesk and/or its suppliers. This Agreement does not include the right to copy or sublicense the Software and is personal to you and therefore may not be assigned (by operation of law or otherwise) or transferred without the prior written consent of SmartDesk. You acknowledge that the Software in source code form remains a confidential trade secret of SmartDesk and/or its suppliers and therefore you agree not to attempt to decipher, decompile, disassemble or reverse engineer the Software or allow others to do so, except to the extent applicable laws specifically prohibit such restriction. You further agree not to modify or create derivative works of the Software.

**CONTENT.** Title, ownership rights, and intellectual property rights in and to the content accessed through the Software is the property of the applicable content owner and may be protected by applicable copyright or other law. This License gives you no rights to such content.

**DISCLAIMER OF WARRANTY.** Since the Software is provided free of charge, the Software is provided on an "AS IS" basis, without warranty of any kind, including without limitation the warranties of merchantability, fitness for a particular purpose and non-infringement. The entire risk as to the quality and performance of the Software is borne by you.



Should the Software prove defective, you and not SmartDesk assume the entire cost of any service and repair.

This disclaimer of warranty constitutes an essential part of the agreement. SOME STATES DO NOT ALLOW EXCLUSIONS OF AN IMPLIED WARRANTY, SO THIS DISCLAIMER MAY NOT APPLY TO YOU AND YOU MAY HAVE OTHER LEGAL RIGHTS THAT VARY FROM STATE TO STATE OR BY JURISDICTION.

LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT, OR OTHERWISE, SHALL SMARTDESK OR ITS SUPPLIERS OR RESELLERS BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES. IN NO EVENT WILL SMARTDESK BE LIABLE FOR ANY DAMAGES IN EXCESS OF SMARTDESK'S LIST PRICE FOR A LICENSE TO THE SOFTWARE, EVEN IF SmartDesk SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. FURTHERMORE, SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS LIMITATION AND EXCLUSION MAY NOT APPLY TO YOU.

**EXPORT CONTROLS.** You may not download or otherwise export or reexport the Software or any underlying information or technology except in full compliance with all United States and other applicable laws and regulations. In particular, but without limitation, none of the Software or underlying information or technology may be downloaded or otherwise exported or reexported (i) into (or to a national or resident of) Cuba, Haiti, Iraq, Libya, North Korea, Iran, Syria or any other country to which the U.S. has embargoed goods; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Deny Orders. By downloading or using the Software, you are agreeing to the foregoing and you are representing and warranting that you are not located in, under the control of, or a national or resident of any such country or on any such list.

**TERMINATION.** Either party may terminate this Agreement immediately in the event of default by the other party. Upon any termination of this Agreement, you shall immediately discontinue the use of the Software and shall within ten (10) days return to SmartDesk all copies of the Software and Documentation. You may also terminate this Agreement at any time by destroying the Software and Documentation and all copies thereof. Your obligations to pay accrued charges and fees shall survive any termination of this Agreement.

**MISCELLANEOUS.** This Agreement represents the complete and exclusive statement of the agreements concerning this license between the parties and supersedes all prior agreements and representations between them. It may be amended only by a writing executed by both parties. THE ACCEPTANCE OF ANY PURCHASE ORDER PLACED BY YOU IS EXPRESSLY MADE CONDITIONAL ON YOUR ASSENT TO THE TERMS SET FORTH HEREIN, AND SMARTDESK AGREES TO FURNISH THE SOFTWARE AND DOCUMENTATION ONLY UPON THESE TERMS AND NOT THOSE CONTAINED IN YOUR PURCHASE ORDER. If any provision of this Agreement is held to be unenforceable for any reason, such provision shall be reformed only to the extent necessary to make it enforceable, and such decision shall not affect the enforceability

(i) of such provision under other circumstances or (ii) of the remaining provisions hereof under all circumstances. Headings shall not be considered in interpreting this Agreement. This Agreement shall be governed by and construed under California law as such law applies to agreements between California residents entered into and to be performed entirely within California, except as governed by Federal law. This Agreement will not be governed by the United Nations Convention of Contracts for the International Sale of Goods, the application of which is hereby expressly excluded.

U.S. Government Restricted Rights. Use, duplication or disclosure by the Government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer-Restricted Rights clause at FAR 52.227-19 when applicable, or in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is SmartDesk, Inc., 26062 Merit Circle, Suite 106, Laguna Hills, CA 92653.

**Contact:**

SmartDesk, Inc.  
26062 Merit Circle  
Suite 106  
Laguna Hills, CA 92653  
Phone: 714-582-4020  
Fax: 714-348-0006

# Sweden

## **Pihlström IT Service**

Kenn Nesbitt's WebEdit Skandinavien distributör

Den svenska versionen av Kenn Nesbitts WebEdit distribueras i Sverige och Skandinavien av Pihlström IT Service:

## **Pihlström IT Service**

Bilradiogatan 45

421 34 Västra Frölunda

SWEDEN

**Voice:** 0705 477377 / Internationellt +46 705 477377

**Fax:** 031 472919 / Internationellt +46 31 472919

**Email:** [webedit@pits.se](mailto:webedit@pits.se)

För mer information om beställning, se svenska hemsidan för WebEdit, tryck  
<http://www.pits.se/Webedit>

## Where to Reach Us

The best way to reach us is online. The WebEdit home page is on the World Wide Web at: <http://www.nesbitt.com/>

Technical support is available to all registered users by sending email to [techsupt@nesbitt.com](mailto:techsupt@nesbitt.com).

WebEdit has grown so popular that we now have our own Mailing List! To subscribe, send mail to [majordomo@thegroup.net](mailto:majordomo@thegroup.net) with the words 'Subscribe WebEdit-List me@myaddress' in the body of the message.

In addition to the mailing list, you can get support for WebEdit on usenet in the group alt.html.editors.webedit, and on CompuServe in the Internet Publishing forum (GO INETPUB). If your usenet server doesn't have alt.html.editors.webedit, please ask your service provider to carry it.

### See Also

[How to Order WebEdit](#)  
[About the Authors](#)

# WebEdit Reference

## **HTML**

[Tag Index](#)

[Document Structure](#)

[Block Style Tags](#)

[Logical Font Style Tags](#)

[Physical Font Style Tags](#)

[Headings](#)

[Special Characters](#)

[Form Elements](#)

[List Elements](#)

[Miscellaneous Tags](#)

[Table Elements](#)

[Java Tags](#)

[Object Tags](#)

[Links / Anchors](#)

[Inline Images](#)

[Uniform Resource Locators \(URLs\)](#)

[User-Defined Tags](#)

[Learning HTML](#)

## **Miscellaneous Features**

[Spelling Checker](#)

[Quick Previewer](#)

# Document Structure Tags

## HTML Document Structure Tags

HTML provides methods for organizing and navigating within HTML documents using the following elements:

The Banner element is used for graphic elements that shouldn't be scrolled with the rest of the form.

The Base element is for establishing a location context for other URLs referenced.

The Body element determines the start and finish of the document's body.

The Division element is used to represent different types of containers.

The Document Type element describes to what level of HTML the document is written for.

The Frame element defines a single frame in a frameset.

The Frame Set element defines the main container for a frame.

The Head element determines the start and finish of the document's body.

The HTML element shows where the entire HTML document starts and finishes.

The Is Index element indicates that the document supports CGI script for searches.

The Meta element is a method for including extra information about your document.

The Next ID element indicates the next Web page after the current one. This allows for document chaining.

The No Frames element specifies alternative content that is viewable by non-frame-capable clients.

The Range element marks the range of the document.

The Spot element is used to insert IDs at arbitrary places.

The Style element allows for the inclusion of rendering information using a specified style notation.

The Title element specifies the title of the document.

In HTML documents, tags define the start and end of headings, paragraphs, lists, character highlighting and links etc. Most HTML elements are identified in a document as a start tag, which gives the element name and attributes, followed by the content, followed by the end tag. Start tags are delimited by < and >, while end tags are delimited by </ and >. For example:

```
<title>This is a Title</title>
<h1>This is a Heading</h1>
<P>This is a paragraph.
```

As can be seen from the <P>, not every tag has an end tag.

Every HTML document as a minimum must have a title. To identify the document as being HTML 3.0, it is recommended that documents start with the prologue:

```
<!doctype HTML public "-//W30//DTD W3 HTML 3.0//EN">
```

When absent, this prologue is implied by the MIME content type for HTML 3.0

together with the associated version parameter.

### **Document Structure**

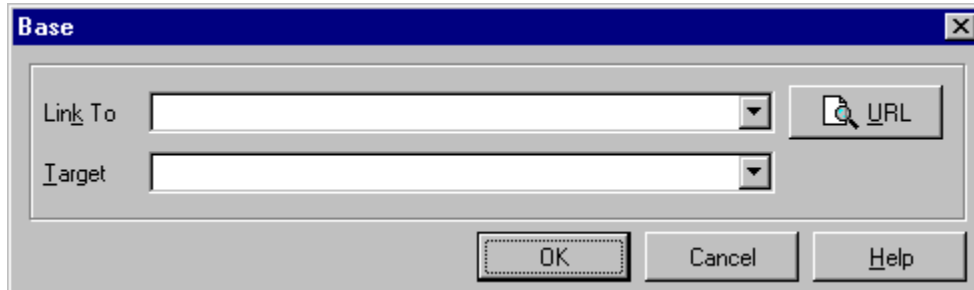
HTML 3.0 documents formally have the following structure:

```
<HTML>  
<HEAD> head elements ...  
<BODY> body elements ...  
</HTML>
```

In most cases, the HTML, HEAD and BODY tags can be safely omitted.

# Base

## HTML 0




### Description

The Base element allows the URL of the document itself to be recorded in situations in which the document may be read out of context. URLs within the document may be in a "partial" form relative to this base address. Where the base address is not specified, the reader will use the URL it used to access the document to resolve any relative URLs.

### Attributes

**HREF** - **HTML 0** A URL.

**TARGET** -  Links in any window can refer to another window by name using the TARGET attribute. When you click the link, the document you requested will appear in that named window. If the window is not already open, the browser will open and name a new window for you.

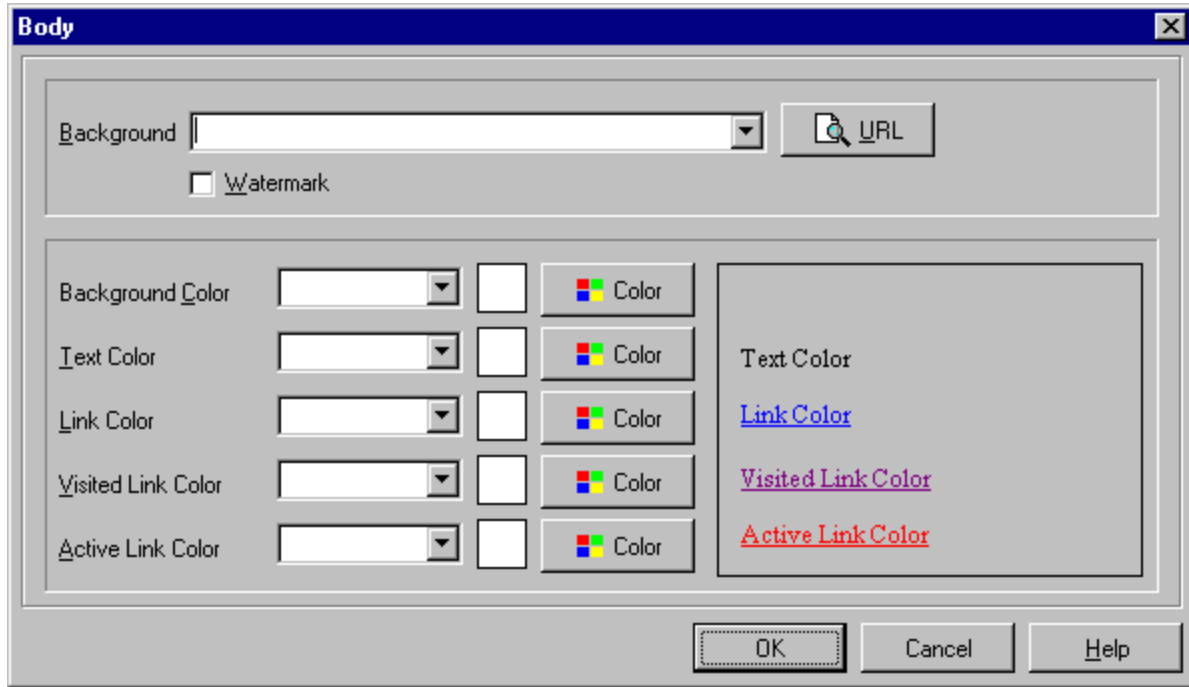
### Example

```
<BASE HREF="http://www.nesbitt.com/webedit/webedit.htm">
```



# Body

## HTML 0




### Description


The Body element contains all the information which is part of the document, as opposed information about the document which is in the Head. The elements within the Body element are in the order in which they should be presented to the reader. Place the <BODY> and </BODY> tags above and below the body of the text (not including the Head) of your HTML document.

### Example

```
<HTML>
<HEAD>
<TITLE>Page Title</TITLE>
</HEAD>
<BODY>
Add body text here.
</BODY>
</HTML>
```

### Attributes

**BACKGROUND** -  Specifies the image tile to appear in the document background.

**BGPROPERTIES** -  Specifies a watermark, which is a background picture that does not scroll.

```
<BODY BACKGROUND="linoleum.gif" BGPROPERTIES=FIXED>
The background on this page is fixed so that it does not
scroll.</BODY>
```

**BGCOLOR**= #rrggbb  or colorname

## HTML 0

. Sets the background color of the page. rrggbb is a hexadecimal number denoting a red-green-blue color value (the pound sign is optional). BGCOLOR and each of the other color attributes here can also be set to a colorname.


```
<BODY BGCOLOR=#ff0000>This page has a red background.</BODY>
```

or


```
<BODY BGCOLOR=RED>This page also has a red background.</BODY>
```

**LINK=#rrgbb or colorname.**  1.1 Sets the color of shortcuts that have not yet been visited.

```
<BODY LINK=#0000ff>This page has blue shortcuts</BODY>
```

**TEXT=#rrgbb or colorname.**  1.1 Sets the color of text on the page.

```
<BODY TEXT=FUCHSIA>This text is fuchsia.</BODY>
```

**ALINK=#rrgbb or colorname.**  1.1 Sets color of shortcuts that are active.

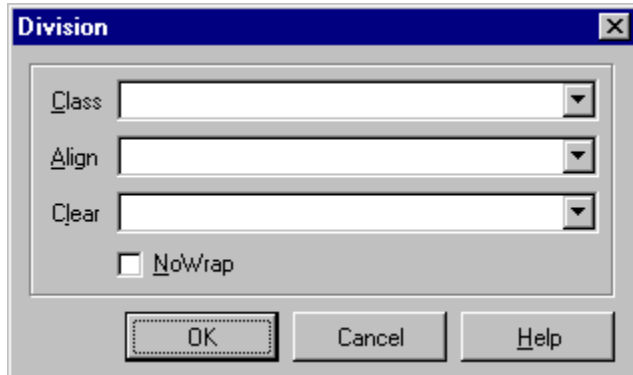
```
<BODY ALINK=#ff0000>Active shortcuts</BODY>
```

**VLINK=#rrgbb or colorname.**  1.1 Sets color of shortcuts that have already been visited.

```
<BODY VLINK=#00ff00>This page has green visited  
shortcuts</BODY>
```

# Division

## HTML 0



### Description

The DIV element is used with the CLASS attribute to represent different kinds of containers, e.g. chapter, section, abstract, or appendix. Use the DIV element together with header elements when you want to make the hierarchical structure of a document explicit. This is needed as header elements themselves only contain the text of the header, and do not imply any structural division of documents into sections. Header elements have the same content model as paragraphs, that is text and character level markup, such as character emphasis, inline images, form fields and math.

### Attributes

**ID** - [HTML 3.2](#) An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**LANG** - [HTML 3.2](#) This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - [HTML 3.2](#) This is a space separated list of SGML NAME tokens and is used to subclass tag names. For instance, <DIV CLASS=APPENDIX> defines a division that acts as an appendix. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

**ALIGN** - [HTML 3.2](#) The ALIGN attribute can be used to explicitly specify the horizontal alignment of paragraphs within a division:

**align=left** - Paragraphs are rendered flush left (the default).

**align=center** - Paragraphs are centered.

**align=right** - Paragraphs are rendered flush right.

**align=justify** - Text lines are justified where practical, otherwise this gives the same effect as the default align=left setting.

**NOWRAP** - [HTML 3.2](#) The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks in paragraphs using the BR element.

**CLEAR** - **HTML 0** This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start the division below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

**clear=left** - move down until left margin is clear

**clear=right** - move down until right margin is clear

**clear=all** - move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

**clear="40 en"** - move down until there is at least 40 en units free

**clear="100 pixels"** - move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

### Example

```
<DIV CLASS=Abstract>  
<P>TheChieftain product range is the white hot hope for the  
coming year. This report sets out how to position Chieftain  
against competing products.  
</DIV>
```

# Head

## HTML 0

### Description

The Head element contains all information about the document in general. It contains HTML elements that describe the document's title, usage and relationship with other documents. It does not contain any text which is part of the document - this is in the Body. Within the Head element, only certain elements are allowed.

### Attributes

None

### Example

```
<HTML>
<HEAD>
<TITLE>Page Title</TITLE>
</HEAD>
<BODY>
Add body text here.
</BODY>
</HTML>
```

# Document Type

## HTML 0



### Short Non-Technical Description

To formally identify the file as containing HTML elements, the beginning of the file should contain a line identifying the version of HTML being used - this is the DocType tag.

To just state only that the document is HTML:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
```

To state exactly what level of HTML it is:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 1.0//EN">  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN">  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML Level 0//EN">  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML Level 1//EN">  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML Level 2//EN">  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML Level 3//EN">
```

### Formal Variants of HTML 3.0

The HTML 3.0 document type definition includes two flags for controlling how prescriptive or how lax the language is.

### HTML Recommended

Certain features of the language are necessary for compatibility with widespread usage, but they may compromise the structural integrity of a document. The HTML.Recommended entity should be defined as INCLUDE in the DTD subset to enable a more prescriptive version of HTML 3 that eliminates the above features. For example:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2//EN"  
[ <!ENTITY % HTML.Recommended "INCLUDE"> ] >
```

In particular, this prevents text from appearing except within block elements.

### HTML Deprecated

By default, for backwards compatibility, the %HTML.Deprecated entity is defined as INCLUDE, enabling certain features which are now deprecated. These features can be eliminated by defining this entity as IGNORE in the DTD subset. For example:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2//EN" [  
<!ENTITY % HTML.Deprecated "IGNORE"> ] >
```

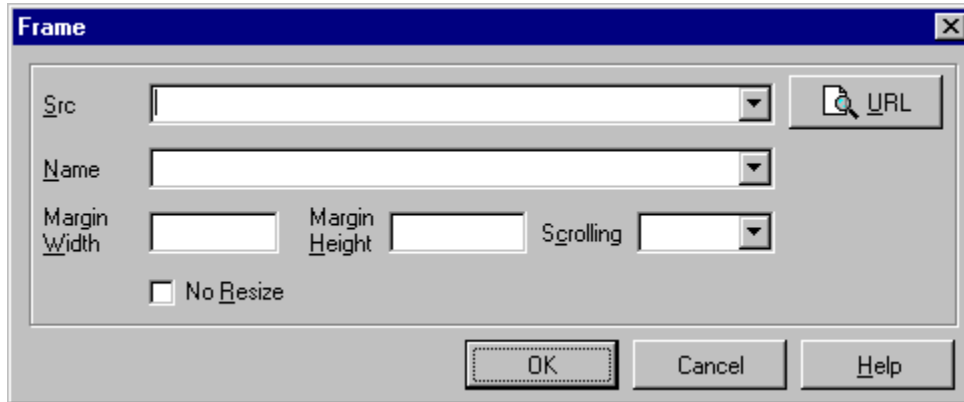
**Note:** defining %HTML.Recommended as INCLUDE automatically sets %HTML.Deprecated to IGNORE.

In the spirit of being liberal in what you accept and strict in what you generate, HTML user agents are recommended to accept syntax corresponding to the specification

with %HTML.Deprecated turned on, while HTML user agents generating HTML are recommended to generate documents that conform to the specification with %HTML. Recommended turned on.

# Frame

## HTML 0



### Description

This tag defines a single frame in a frameset.

### Attributes

**SRC** - **HTML 0** The SRC attribute takes as its value the URL of the document to be displayed in this particular frame. FRAMEs without SRC attributes are displayed as a blank space the size the frame would have been.

**NAME** - **HTML 0** The NAME attribute is used to assign a name to a frame so it can be targeted by links in other documents (These are usually from other frames in the same document.) The NAME attribute is optional; by default all windows are unnamed. Names must begin with an alphanumeric character. However, several reserved names have been defined, which start with an underscore.

These are currently:

- `_blank` Always load this link into a new, unnamed window.
  - `_self` Always load this link over yourself.
  - `_parent` Always load this link over your parent. (becomes self if you have no parent).
  - `_top` Always load this link at the top level. (becomes self if you are at the top).
- All other names starting with '\_' will be ignored.

**MARGINWIDTH** - **HTML 0** The MARGINWIDTH attribute is used when the document author wants some control of the margins for this frame. If specified, the value for MARGINWIDTH is in pixels. Margins can not be less than one-so that frame objects will not touch frame edges-and can not be specified so that there is no space for the document contents. The MARGINWIDTH attribute is optional; by default, all frames default to letting the browser decide on an appropriate margin width.

**MARGINHEIGHT** - **HTML 0** The MARGINHEIGHT attribute is just like



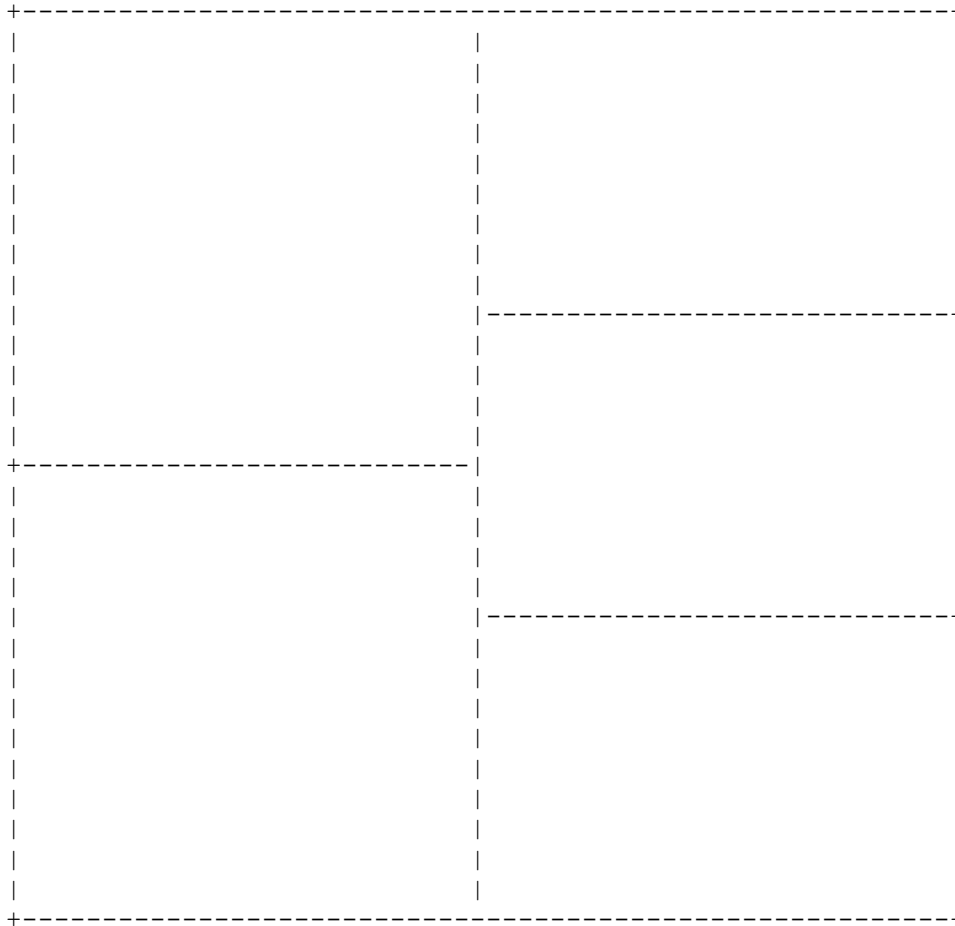
MARGINWIDTH above, except it controls the upper and lower margins instead of the left and right margins.

**SCROLLING** - **HTML 0** The SCROLLING attribute is used to describe if the frame should have a scrollbar or not. Yes results in scrollbars always being visible on that frame. No results in scrollbars never being visible. Auto instructs the browser to decide whether scrollbars are needed, and place them where necessary. The SCROLLING attribute is optional; the default value is auto.

**NORESIZE** - **HTML 0** The NORESIZE attribute has no value. It is a flag that indicates that the frame is not resizable by the user. Users typically resize frames by dragging a frame edge to a new position. Note that if any frame adjacent to an edge is not resizable, that entire edge will be restricted from moving. This will effect the resizability of other frames. The NORESIZE attribute is optional; by default all frames are resizable.

**Example** (Courtesy Netscape Communications)

This example compares Frame syntax and TABLE syntax.



## THE ABOVE LAYOUT USING TABLES

```
<TABLE WIDTH="100%" HEIGHT="100%" BORDER>
  <TR><TD ROWSPAN=2>CELL1</TD><TD>CELL2</TD></TR>
  <TR><TD ROWSPAN=2>CELL3</TD></TR>
  <TR><TD ROWSPAN=2>CELL4</TD></TR>
  <TR><TD>CELL5</TD></TR>
</TABLE>
```

## THE ABOVE LAYOUT USING FRAMES

```
<FRAMESET COLS="50%, 50%">
  <FRAMESET ROWS="50%, 50%">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
  </FRAMESET>
  <FRAMESET ROWS="33%, 33%, 33%">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
  </FRAMESET>
</FRAMESET>
```

## THE ABOVE LAYOUT USING NOFRAMES INFO

```
<FRAMESET COLS="50%, 50%">

<NOFRAMES>
<h1 align=center><blink>Frame ALERT!</blink></h1>
<p>
This document is designed to be viewed using <b>Netscape 2.0</b>'s
Frame features. If you are seeing this message, you are using
a frame <i>challenged</i> browser.
</p>
<p>
A <b>Frame-capable</b> browser can be gotten from
<a href=http://home.netscape.com/>Netscape Communications</a>.
</p>
</NOFRAMES>

<FRAMESET ROWS="50%, 50%">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
</FRAMESET>
<FRAMESET ROWS="33%, 33%, 33%">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
</FRAMESET>

</FRAMESET>
```



## Overview of Frames

Frames are a way of specifying multiple, independent, scrollable regions within a display window. Each frame can contain a separate HTML document. Users can scroll and resize frames, at the choice of the page creator. Each frame can also be given NAME values, so they can be targeted by links in other documents or other frames. These features provide a powerful new way of presenting documents.

Properties of frames:

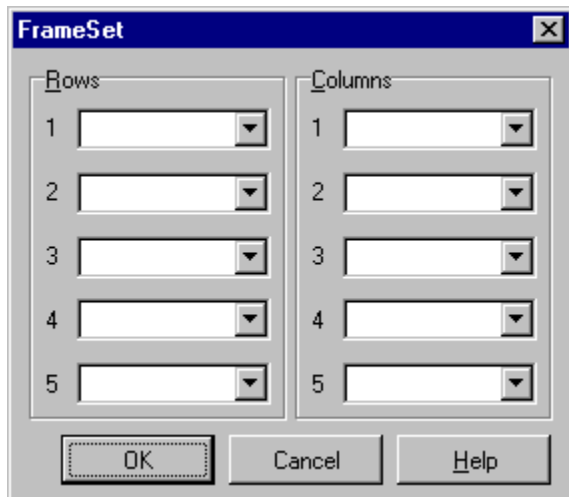
- A frame can be given an individual URL, so it can load information independent of the other frames on the page;
- A frame can be given a NAME, allowing it to be targeted by other URLs, and;
- A Frame can resize dynamically if the user changes the window's size. (Resizing can also be disabled, ensuring a constant frame size.)

These properties offer new possibilities:

- Elements that the user should always see, such as control bars, copyright notices, and title graphics can be placed in a static, individual frame. As the user navigates the site in "live" frames, the static frame's contents remain fixed, even though adjoining frames redraw.
- Table of contents are more functional. One frame can contain TOC links that, when clicked, display results in an adjoining frame.
- Frames side-by-side design allows queries to be posed and answered on the same page, with one frame holding the query form, and the other presenting the results.

# Frame Set

## HTML 0



### Description

This is the main container for a Frame. It has 2 attributes ROWS and COLS. A frame document has no BODY, and no tags that would normally be placed in the BODY can appear before the FRAMESET tag, or the FRAMESET will be ignored. The FRAMESET tag has a matching end tag, and within the FRAMESET you can only have other nested FRAMESET tags, FRAME tags, or the NOFRAMES tag.

The FRAMESET tag can be nested inside other FRAMESET tags. In this case the complete subframe is placed in the space that would be used for the corresponding frame if this had been a FRAME tag instead of a nested FRAMESET.

### Attributes

**ROWS** - **HTML 0** The ROWS attribute takes as its value a comma separated list of values. These values can be absolute pixel values, percentage values between 1 and 100, or relative scaling values. The number of rows is implicit in the number of elements in the list. Since the total height of all the rows must equal the height of the window, row heights might be normalized to achieve this. A missing ROWS attribute is interpreted as a single row arbitrarily sized to fit.

#### Syntax of value list.

value

A simple numeric value is assumed to be a fixed size in pixels. This is the most dangerous type of value to use since the size of the viewer's window can and does vary substantially. If fixed pixel values are used, it will almost certainly be necessary to mix them with one or more of the relative size values described below. Otherwise the client engine will likely override your specified pixel value to ensure that the total proportions of the frame are 100% of the width and height of the user's window.

value%

This is a simple percentage value between 1 and 100. If the total is greater than 100

all percentages are scaled down. If the total is less than 100, and relative-sized frames exist, extra space will be given to them. If there are no relative-sized frames, all percentages will be scaled up to match a total of 100%.

value\*

The value on this field is optional. A single '\*' character is a "relative-sized" frame and is interpreted as a request to give the frame all remaining space. If there exist multiple relative-sized frames, the remaining space is divided evenly among them. If there is a value in front of the '\*', that frame gets that much more relative space. "2\*," would give 2/3 of the space to the first frame, and 1/3 to the second.

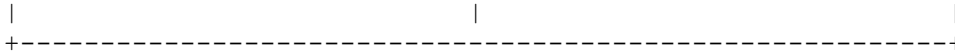
Example for 3 rows, the first and the last being smaller than the center row:  
<FRAMESET ROWS="20%,60%,20%">

Example for 3 rows, the first and the last being fixed height, with the remaining space assigned to the middle row:  
<FRAMESET ROWS="100\*,100">

**COLS** - **HTML 0** The COLS attribute takes as its value a comma separated list of values that is of the exact same syntax as the list described above for the ROWS attribute.

**Example** (Courtesy Netscape Communications)

This example compares Frame syntax and TABLE syntax.

## THE ABOVE LAYOUT USING TABLES

```
<TABLE WIDTH="100%" HEIGHT="100%" BORDER>
  <TR><TD ROWSPAN=2>CELL1</TD><TD>CELL2</TD></TR>
  <TR><TD ROWSPAN=2>CELL3</TD></TR>
  <TR><TD ROWSPAN=2>CELL4</TD></TR>
  <TR><TD>CELL5</TD></TR>
</TABLE>
```

## THE ABOVE LAYOUT USING FRAMES

```
<FRAMESET COLS="50%, 50%">
  <FRAMESET ROWS="50%, 50%">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
  </FRAMESET>
  <FRAMESET ROWS="33%, 33%, 33%">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
  </FRAMESET>
</FRAMESET>
```

## THE ABOVE LAYOUT USING NOFRAMES INFO

```
<FRAMESET COLS="50%, 50%">

<NOFRAMES>
<h1 align=center><blink>Frame ALERT!</blink></h1>
<p>
This document is designed to be viewed using <b>Netscape 2.0</b>'s
Frame features. If you are seeing this message, you are using
a frame <i>challenged</i> browser.
</p>
<p>
A <b>Frame-capable</b> browser can be gotten from
<a href=http://home.netscape.com/>Netscape Communications</a>.
</p>
</NOFRAMES>

<FRAMESET ROWS="50%, 50%">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
</FRAMESET>
<FRAMESET ROWS="33%, 33%, 33%">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
</FRAMESET>

</FRAMESET>
```





# HTML

## HTML 0

### Description

The HTML identifier defines the document as containing HTML elements. It contains only the Head and Body elements.

### Attributes

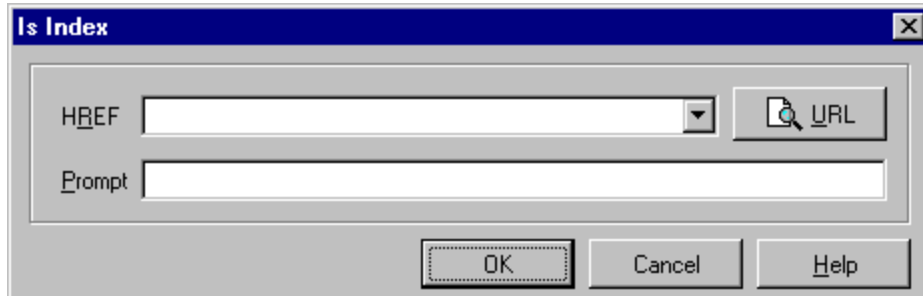
None

### Example

```
<HTML>  
<HEAD>  
<TITLE>Page Title</TITLE>  
</HEAD>  
<BODY>  
Add body text here.  
</BODY>  
</HTML>
```

# Is Index

## HTML 0



### Description

The Is Index Element informs the reader that the document is an index document. As well as reading it, the reader may use a keyword search. The node may be queried with a keyword search by suffixing the node address with a question mark, followed by a list of keywords separated by plus signs. Note that this tag is normally generated automatically by a server. If it is added by hand to an HTML document, then the client will assume that the server can handle a search on the document. Obviously the server must have this capability for it to work: simply adding <ISINDEX> in the document is not enough to make searches happen if the server does not have a search engine!

### Attributes

**ACTION** - **HTML 0** filename. Specifies the gateway program to which the string in the text box should be passed.

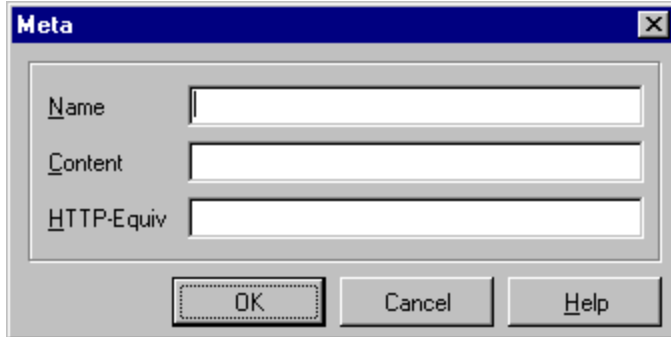
**PROMPT** - **HTML 0** "prompt text". Specifies a prompt to be used instead of the above.

### Example

```
<ISINDEX ACTION = "Search" PROMPT="Type in keywords here">
```

# Meta

## HTML 0



### Description

The META element is used within the HEAD element to embed document meta-information not defined by other HTML elements. Such information can be extracted by servers/clients for use in identifying, indexing and cataloging specialized document meta-information.

Although it is generally preferable to use named elements that have well defined semantics for each type of meta-information, such as title, this element is provided for situations where strict SGML parsing is necessary and the local DTD is not extensible.

In addition, HTTP servers can read the contents of the document head to generate response headers corresponding to any elements defining a value for the attribute HTTP-EQUIV. This provides document authors with a mechanism (not necessarily the preferred one) for identifying information that should be included in the response headers of an HTTP request.

### Attributes

**NAME** - **HTML 0** Used to name a property such as author, publication date etc. If absent, the name can be assumed to be the same as the value of HTTP-EQUIV.

**CONTENT** - **HTML 0** Used to supply a value for a named property.

**HTTP-EQUIV** - **HTML 0** This attribute binds the element to an HTTP response header. If the semantics of the HTTP response header named by this attribute is known, then the contents can be processed based on a well defined syntactic mapping, whether or not the DTD includes anything about it. HTTP header names are not case sensitive. If absent, the NAME attribute should be used to identify this meta-information and it should not be used within an HTTP response header.

### Examples:

If the document contains:

```
<META HTTP-EQUIV=Expires CONTENT="Tue, 04 Dec 1993 21:29:02 GMT">  
<META HTTP-EQUIV="Keywords" CONTENT="Nanotechnology, Biochemistry">  
<META HTTP-EQUIV="Reply-to" CONTENT="dsr@w3.org (Dave Raggett)">
```

The server will include the following response headers:

Expires: Tue, 04 Dec 1993 21:29:02 GMT  
Keywords: Nanotechnology, Biochemistry  
Reply-to: dsr@w3.org (Dave Raggett)

When the HTTP-EQUIV attribute is absent, the server should not generate an HTTP response header for this meta-information, e.g.

```
<META NAME="IndexType" CONTENT="Service">
```

Do not use the META element to define information that should be associated with an existing HTML element.

Example of an inappropriate use of the META element:

```
<META NAME="Title" CONTENT="The Etymology of Dunsel">
```

Do not name an HTTP-EQUIV attribute the same as a response header that should typically only be generated by the HTTP server. Some inappropriate names are "Server", "Date", and "Last-Modified". Whether a name is inappropriate depends on the particular server implementation. It is recommended that servers ignore any META elements that specify HTTP equivalents (case insensitively) to their own reserved response headers.

# Next ID

## HTML 0

### Description

The Next ID element takes a single attribute which is the number of the next document-wide numeric identifier to be allocated. When modifying a document, old anchor ids should not be reused, as there may be references stored elsewhere which point to them. This is read and generated by hypertext editors. Human writers of HTML usually use mnemonic alphabetical identifiers. Browser software may ignore this tag.

### Attributes

**N** - **HTML 0** number of the next document-wide numeric identifier to be allocated.

### Example

```
<NEXTID N=27>
```

# No Frames

## HTML 0

This tag is for content providers who want to create alternative content that is viewable by non-Frame-capable clients. A Frame-capable Internet client ignores all tags and data between start and end NOFRAMES tags.

### Example

This example compares Frame syntax and TABLE syntax.

CELL1	CELL2
CELL3	CELL4
CELL5	CELL6

### THE ABOVE LAYOUT USING TABLES

```
<TABLE WIDTH="100%" HEIGHT="100%" BORDER>  
  <TR><TD ROWSPAN=2>CELL1</TD><TD>CELL2</TD></TR>  
  <TR><TD ROWSPAN=2>CELL3</TD></TR>  
  <TR><TD ROWSPAN=2>CELL4</TD></TR>  
  <TR><TD>CELL5</TD></TR>  
</TABLE>
```

## THE ABOVE LAYOUT USING FRAMES

```
<FRAMESET COLS="50%,50%">
  <FRAMESET ROWS="50%,50%">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
  </FRAMESET>
  <FRAMESET ROWS="33%,33%,33%">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
    <FRAME SRC="cell.html">
  </FRAMESET>
</FRAMESET>
```

## THE ABOVE LAYOUT USING NOFRAMES INFO

```
<FRAMESET COLS="50%,50%">

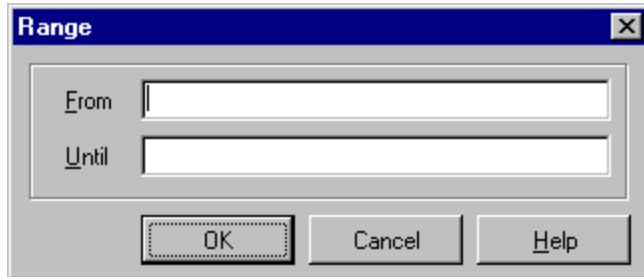
<NOFRAMES>
<h1 align=center><blink>Frame ALERT!</blink></h1>
<p>
This document is designed to be viewed using <b>Netscape 2.0</b>'s
Frame features. If you are seeing this message, you are using
a frame <i>challenged</i> browser.
</p>
<p>
A <b>Frame-capable</b> browser can be gotten from
<a href=http://home.netscape.com/>Netscape Communications</a>.
</p>
</NOFRAMES>

<FRAMESET ROWS="50%,50%">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
</FRAMESET>
<FRAMESET ROWS="33%,33%,33%">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
  <FRAME SRC="cell.html">
</FRAMESET>

</FRAMESET>
```

# Range

HTML 0



## Description

The RANGE element is used to mark a range of the document, for example for highlighting regions of the document matching some search criteria, or which are the subject of an annotation etc.

```
<RANGE CLASS=Search FROM=spot01 UNTIL=spot02>
```

The FROM and UNTIL attributes specify positions in the document using SGML identifiers. Most elements in the document body can define such identifiers using ID attributes. The SPOT element is useful in this regard, as it allows search software etc. to insert IDs at random places:

```
<SPOT ID=spot01> ... <SPOT ID=spot02>
```

## Attributes

**ID** - **HTML 0** An SGML identifier used to name the range element.

**CLASS** - **HTML 0** A character string used to subclass the range element.

**FROM** - **HTML 0** References an SGML identifier for an element in the document body. It identifies the start of the marked range.

**UNTIL** - **HTML 0** References an SGML identifier for an element in the document body. It identifies the end of the marked range.



# Spot

## HTML 0

### Description

The SPOT element is used to insert IDs at arbitrary places, e.g., for end points of a marked range (see [RANGE](#)).

### Attributes

**ID** - **HTML 0** An SGML identifier used to name the range element.

**CLASS** - **HTML 0** A character string used to subclass the range element.

**FROM** - **HTML 0** References an SGML identifier for an element in the document body. It identifies the start of the marked range.

**UNTIL** - **HTML 0** References an SGML identifier for an element in the document body. It identifies the end of the marked range.

### Example

```
<RANGE CLASS=Search FROM=spot01 UNTIL=spot02>
```

The FROM and UNTIL attributes specify positions in the document using [SGML](#) identifiers. Most elements in the document body can define such identifiers using ID attributes. The SPOT element is useful in this regard, as it allows search software etc. to insert IDs at random places:

```
<SPOT ID=spot01> ... <SPOT ID=spot02>
```

# Style

## HTML 0

### Description

The STYLE element provides a means for including rendering information using a specified style notation. Information in the STYLE element overrides client defaults and that of linked style sheets. It allows authors to specify overrides, while for the most part using a generic style sheet, and as such improves the effectiveness of caching schemes for linked style sheets. Stylistic rules will in general match tag names and attribute values for elements in the document body. Context sensitive rules may be used for such purposes as rendering drop down capitals for the initial letter in the first paragraph following a header.

### Attributes

**Notation** - **HTML 0** specifies an entity identifying an SGML notation in the HTML 3.0 DTD.

### Example

```
<style notation=dsssl-lite>  
  some dsssl-lite stuff ...  
</style>
```

# Title

## HTML 0

### Description

The Title element specifies the title of a document. The TITLE element should occur in the HEAD of the document. There may only be one title in any document. It should identify the content of the document in a fairly wide context. The title is not part of the text of the document, but is a property of the whole document. It may not contain anchors, paragraph marks, or highlighting. The title may be used to identify the node in a history list, to label the window displaying the node, etc. It is not normally displayed in the text of a document itself. Contrast titles with headings. The title should ideally be less than 64 characters in length. That is, many applications will display document titles in window titles, menus, etc. where there is only limited room. While there is no limit on the length of a title (as it may be automatically generated from other data), information providers are warned that it may be truncated if long. A title should be short, yet, descriptive. The important thing to remember about a title is that when someone, hopefully, adds your home page to their "hotlist", the title is saved and helps them remember what it was that was so hot.

### Attributes

None

### Examples

Appropriate titles:

```
<TITLE>Rivest and Neuman. 1989(b)</TITLE>  
<TITLE>A Recipe for Maple Syrup Flap-Jack</TITLE>  
<TITLE>Introduction -- AFS user's Guide</TITLE>
```

Inappropriate titles are only meaningful within context, for example,

```
<TITLE>Introduction</TITLE>
```

or too long,

```
<TITLE>Remarks on the Quantum-Gravity effects of "Bean Pole"  
diversification in Mononucleosis patients in Developing Countries  
under Economic Conditions Prevalent during the Second half of the  
Twentieth Century, and Related Papers: a Summary</TITLE>
```

# Banner

## Description

The BANNER element is used for corporate logos, navigation aids, disclaimers and other information which shouldn't be scrolled with the rest of the document. It provides an alternative to using the LINK element in the document head to reference an externally defined banner.

## Attributes

**ID** - An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**LANG** - This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - This is a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

## Example

```
<BANNER>July 4th, 1776</BANNER>
```

## Using LINK to include a Document Banner

The LINK element can be used with REL=Banner to reference another document to be used as banner for this document. This is typically used for corporate logos, navigation aids, and other information which shouldn't be scrolled with the rest of the document. For example:

```
<LINK REL=Banner HREF=banner.html>
```

The use of a LINK element in this way, allows a banner to be shared between several documents, with the benefit of being able to separately cache the banner. Rather than using a linked banner, you can also include the banner in the document itself, using the Banner element.

# Block Style Tags

## HTML Block Text Formats

HTML provides for the formatting of blocks of text with the following elements.

The Address element is for address information, signatures, authorship, etc., often at the top or bottom of a document.

The Blockquote element allows text quoted from another source to be rendered specially.

The Byline element is used to denote authorship.

The Centered element is an extension supported by NetScape 1.x for centering blocks of text.

The Literal element works like the Preformatted element, except that it is rendered in a proportional font.

The Paragraph element is used for normal text.

The Preformatted text element is used for displaying computer output or plain text files. It is usually rendered in a fixed-pitch font with carriage returns and spacing left intact.

# Address

## HTML 0

### Description

The Address element is for address information, signatures, authorship, etc., often at the top or bottom of a document. Typically, an address element is italic and/or right justified or indented. The address element implies a paragraph break. Paragraph marks within the address element do not cause extra white space to be inserted.

### Attributes

None

### Examples

```
<ADDRESS><A HREF="Author.html">A.N.Other</A></ADDRESS>
<ADDRESS>
Newsletter editor<p>
J.R. Brown<p>
JimquickPost News, Jumquick, CT 01234<p>
Tel (123) 456 7890
</ADDRESS>
```

# Blockquote

## HTML 0

### Description

The Blockquote element allows text quoted from another source to be rendered specially. A typical rendering might be a slight extra left and right indent, and/or italic font. Blockquote causes a paragraph break, and typically a line or so of white space will be allowed between it and any text before or after it. Single-font rendition may for example put a vertical line of ">" characters down the left margin to indicate quotation in the Internet mail style.

### Attributes

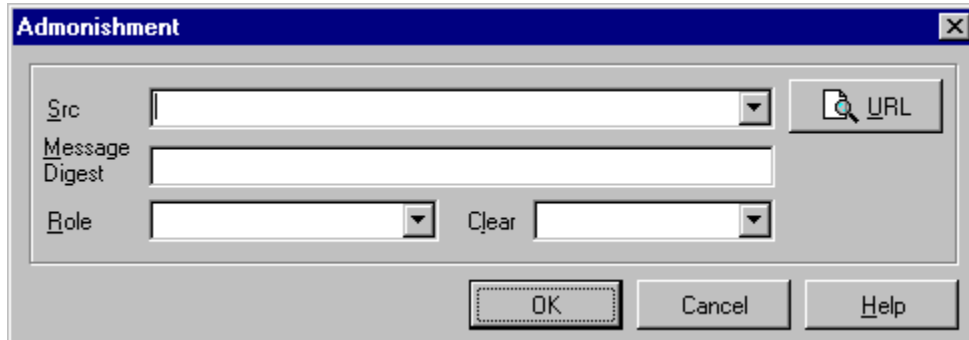
None

### Example

```
I think it ends  
<BLOCKQUOTE>Soft you now, the fair Ophelia. Nymph, in thy  
orisons, be all my sins remembered.  
</BLOCKQUOTE>  
but I am not sure.
```

# Admonishment

## HTML 0



The NOTE element is designed for use as admonishments such as notes, cautions or warnings, as commonly used in technical documentation. The CLASS attribute specifies the type of the element and is typically associated with different graphics such as a road traffic warning sign. The graphic can be customized with the SRC attribute.

### Attributes

**SRC** - **HTML 0** Specifies an image to appear preceding the note. The image is specified as a URI. This attribute may appear together with the MD attribute.

**MESSAGE DIGEST** - **HTML 0** Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQITDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

**ROLE** - **HTML 0** The role names NOTE, CAUTION and WARNING are recommended for standard admonishments. In the absence of the ROLE attribute, a NOTE element is typically rendered indented, without an accompanying graphic. Some browsers expect the CLASS attribute in place of the ROLE attribute - useage is the same.

**CLEAR** - **HTML 0** This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start the division below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

**clear=left** - move down until left margin is clear

**clear=right** - move down until right margin is clear

**clear=all** - move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

**clear="40 en"** - move down until there is at least 40 en units free

**clear="100 pixels"** - move down until there is at least 100 pixels free



The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

**Example**

```
<NOTE ROLE=WARNING>Please check with the local weather service before
starting your climb. The mountain weather is subject to rapid
deterioration. It is essential to carry a good map and compass.</NOTE>
```

# Byline

## HTML 0

### Description

The Byline element is used to denote authorship.

### Attributes

None

### Example

```
<BYLINE>by Mark Twain</BYLINE>
```

# Center

## HTML 0

### Description

The Center element is used for centering blocks of text.

### Attributes

None

### Note

This tag was previously a Netscape tag before becoming adopted in HTML 3.2  
The Center tag is shorthand for <DIV ALIGN=center>

### Example

```
<CENTER>  
This text should appear in the horizontal center of the page.  
</CENTER>
```

Here is alternate way of centering text in HTML 3:

```
<P ALIGN=center>  
This text should appear in the horizontal center of the page.  
</P>
```

# Literal

## HTML 0

### Description

The Literal element works like the Preformatted element, except that it is rendered in a proportional font. The ability to set tab stops in literal text makes it much easier to write filters that convert documents written on word processors into HTML 3. Tab stops can be set with the Tab element and apply for the scope of the Literal element.

### Attributes

None

### Example

```
<LIT>  
  <B>ROYAL TEA</B>
```

```
Today the Earl of Sandwich  
  is having lunch with me,  
and then His Highness, Earl Grey,  
  is visiting for tea.
```

```
At dinnertime, the good Sir Loin  
  will sit down at my table.  
And Lady Fingers, for dessert,  
  will come if she is able.
```

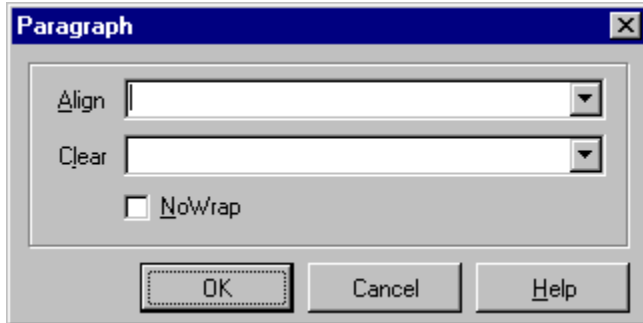
```
I think, of all the royalty  
  who joined me as I ate,  
the ones I liked the least of all  
  were Count Your-Crumbs and Baron Plate.
```

```
  Copyright © 1995, Kenn Nesbitt  
</LIT>
```

# Paragraph

HTML 0

HTML 0



## Description

The Paragraph element is used for normal text. Normal text is automatically wrapped by the browser at the current window margin and adapts to changes in window size. The text is generally shown in a proportional font. HTML level 0 has only a `<P>` tag, but no `</P>` tag. In HTML 2 and HTML 3 the P element acts as a container for the text between the start tag `<P>` and the end tag `</P>`. However, you don't need to give the end tag as it is implied by the context, e.g. the following `<P>` tag. If you wish, you may think of the `<P>` tag as a paragraph separator. This works since HTML formally doesn't require you to wrap text up as paragraphs.

## Status

Basic `<P>` tag is HTML 0.

Block Element `<P>...</P>` is HTML 2.

## Attributes

The following attributes are part of the current HTML 3 draft specification.

**ID** - **HTML 0** An identifier for the current paragraph.

**ALIGN** - **HTML 0** The alignment of the paragraph (e.g., left, right, justify, center or indent)

**NOWRAP** - **HTML 0** Can be set to OFF to prevent automatic word-wrapping of text.

## Example

```
<P>This is a regular paragraph</P>  
<P ALIGN=center>This paragraph is centered.</P>
```

The text that makes up your paragraph in HTML does not rely on carriage returns. Multiple spaces and carriage returns or reduced to a single space.

```
This will all appear  
as only one continuous sentence. <P>
```

Would do exactly what it says. Web browsers pay no attention to line breaks and start new paragraphs only if they reach a <P> tag. Just make sure that you do separate paragraphs with <P>.

# Preformatted

## HTML 0

### Description

Preformatted text between the start and end PRE tag is rendered using a fixed width font, in addition whitespace characters are treated literally. The spacing and line breaks are rendered directly, unlike other elements, for which repeated whitespace characters are collapsed to a single space character and line breaks introduced automatically.

Line breaks within the text are rendered as a move to the beginning of the next line. The exceptions are line breaks immediately following the starting PRE tag or immediately preceding the ending PRE tag, which should be ignored.

The <P> tag should be avoided, but for robustness, user agents are recommended to treat these tags as line breaks.

Anchor elements, and character highlighting elements may be used.

FORM elements may be included, and the fixed width font exploited to control layout (the TAB or TABLE elements give similar control for normal text though).

Block-like elements such as headers, lists, FIG and TABLES should be avoided.

The horizontal tab character (encoded in US ASCII and ISO 8859-1 as decimal 9) should be interpreted as the smallest nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8.

### Attributes

**WIDTH** - **HTML 0** specifies the width of the presentation.

**ID** - **HTML 0** An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**LANG** - **HTML 0** This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - **HTML 0** This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

**CLEAR** - **HTML 0** This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start the preformatted text below the figure rather than alongside it. The

CLEAR attribute allows you to move down unconditionally:

**clear=left**      move down until left margin is clear

**clear=right**     move down until right margin is clear

**clear=all**      move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

**clear="40 en"**    move down until there is at least 40 en units free

**clear="100 pixels"**    move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

### **Example**

A verse from Shelley (To a Skylark):

```
<PRE>
    Higher still and higher
      From the earth thou springest
    Like a cloud of fire;
      The blue deep thou wingest,
And singing still dost soar, and soaring ever singest.</PRE>
```

which is rendered as:

```
Higher still and higher
  From the earth thou springest
Like a cloud of fire;
  The blue deep thou wingest,
And singing still dost soar, and soaring ever singest.
```



# Logical Font Style Tags

## HTML Logical Font Style Text Formats

HTML provides for the formatting of blocks of text with the following elements.

The Abbrev element is used to markup abbreviations.

The Acronym element is used to markup acronyms.

The Argument element is used for logically denoting program function arguments.

The Cite element specifies a citation. Sections tagged with the CITE element are typically rendered in italics.

The Code element indicates an example of code; typically rendered in a mono-spaced font. Do not confuse with PRE.

The DFN element indicates the defining instance of a term.

The Person element is used for names of people to allow these to be extracted automatically by indexing programs.

The Q element is used for a short quotation. It is typically shown enclosed in quotation marks as appropriate to the language context. For English these would be matching double or single quotation marks, alternating for nested quotes. The language context is set by the LANG attribute.

The Samp (Sample) element indicates a sequence of literal characters.

The Var element indicates a variable name, and might typically be used in an instruction manual.

## Description

HTML distinguishes between two groups of character-formatting tags; Logical character-attribute tags and Physical character-attribute tags. It may help to think of Physical character-attribute tags as closely related to the direct formatting you could apply to text from a word processor, e.g., bold. The appearance of HTML text formatted with Physical character-attribute tags is more likely to remain constant from one browser to another. Logical character-attribute tags in HTML can be thought of as "styles" in a word processor, i.e., the appearance of the text formatted with a style in a word processor depends on how that style is defined in that word processor. Similarly, the appearance of HTML text formatted with Logical character-attribute tags depends upon that browsers interpretation of that Logical character-attribute tag.

## Examples

```
This text contains an <em>emphasized</em> word.  
<strong>Don't assume</strong> that it will be italic!  
It was made with the <code>EM</code> element. A cite is  
often italic and has no formally required structure:  
<cite>Moby Dick</cite> is a book title.
```

HTML source code giving examples of various HTML Logical Font Style Tags.

```
<HTML>  
<HEAD>  
<TITLE>Logical Font Style Tags</TITLE>  
</HEAD>  
<BODY>  
<H3>Logical Font Style Tags</H3>
```

```
<ABBREV>Abreviation</ABBREV><BR>
<ACRONYM>Acronym</ACRONYM><BR>
<ARG>Argument</ARG><BR>
<CITE>Citation</CITE><BR>
<CODE>Code</CODE><BR>
<DFN>Defining Instance</DFN><BR>
<Q>Inline Quote</Q><BR>
<PERSON>Person</PERSON><BR>
<SAMP>Sample</SAMP><BR>
<VAR>Variable</VAR><BR>
<HR>
</BODY>
</HTML>
```

# Abbreviation

HTML 0

## Description

The Abbreviation element is used for logically denoting abbreviations.

## Attributes

None

## Example

```
<ABBREV>SW 100th St.</ABBREV>
```

# Acronym

HTML 0

## Description

The Acronym element is used for logically denoting acronyms.

## Attributes

None

## Example

```
<ACRONYM>HTML</ACRONYM>
```

# Argument

## HTML 0

### Description

The Argument element is used for logically denoting program function arguments.

### Attributes

None

### Example

```
LPSTR lstrcpy(<ARG>lpszString1</ARG>, <ARG>lpszString2</ARG>);
```

# Citation

## HTML 0

### Description

The Citation element is used to logically denote citations and is typically rendered in italic.

### Attributes

None

### Example

```
<CITE>Hahn, Harley (1996),  
<U>The Internet Yellow Pages, Third Edition</U>,  
Osborne McGraw-Hill.</CITE>
```

# Code

## HTML 0

### Description

The Code element is used to logically denote an inline example of program code and is typically rendered in a fixed pitch font. This is a text formatting element and not a block formatting element. To format multiple lines of code, use the Preformatted element.

### Attributes

None

### Example

To copy a string, use the `<CODE>LPSTR lstrcpy(lpszString1, lpszString2);</CODE>` function.

# Defining Instance

## HTML 0

### Description

The Defining Instance element is used to logically denote the defining instance of a term.

### Attributes

None

### Example

Fear of computers is often called `<DFN>technophobia</DFN>`.



# Inline Quote

## HTML 0

### Description

The Inline Quote element is used in place of double quotation marks to denote a short inline quotation. For longer quotations, use the Block Quote element.

Since Inline Quotes is HTML 3, and doesn't work in Internet Explorer or Netscape we recommend that you use simple quotation marks, or the &quot; element.

### Example

```
Thus spake Master Rinzei: &quot;Truly excellent programs need no  
advertising; word of mouth is sufficient.&quot;
```

### Attributes

None

### Example

```
As Duke once said, <Q>Come get some!!</Q>
```

# Person

## HTML 0

### Description

The Person element is used to logically denote a proper name.

### Attributes

None

### Example

"Green Eggs and Ham" was written by `<PERSON>Dr. Suess</PERSON>`.

# Sample

## HTML 0

### Description

The Sample element is used to logically denote sample output or a sequence of literal characters.

### Attributes

None

### Example

The report will have a title such as `<SAMP>1996 Crop Report</SAMP>`.

# Variable

## HTML 0

### Description

The Variable element is used to logically denote a variable name, and is typically rendered in italic.

### Attributes

None

### Example

The variables `<VAR>X</VAR>` and `<VAR>Y</VAR>` represent a point on the grid.

# Physical Font Style Tags

## HTML Logical Font Style Text Formats

HTML provides for the formatting of blocks of text with the following elements.

The Big element changes the physical rendering of the contents of the element to a bigger font than normal text, if practical.

The Blink element specifies that the enclosed text should blink.

The B (Boldface) element specifies that the enclosed text should be displayed in a boldface. If this is not practical, an alternative mapping is allowed.

The EM element provides typographic emphasis, typically italics. While `<EM>` and `<I>` often give the same effect, use `<EM>` except where it is necessary in the text to refer to the formatting, as in "The italic parts are mandatory". This will help to ensure consistency between documents from various sources if (for example) the reader prefers to use color in place of italics for emphasis.

The I (Italic) element specifies that the enclosed text should be displayed, if practical, in an italic font (or slanted).

The KBD element indicates text typed (keyboarded) by the user. It might typically be used in an instruction manual.

The S (Strike through) element specifies that the enclosed text should be displayed with a horizontal line striking through the text. If this is not practical, an alternative mapping is allowed.

The Small element changes the physical rendering of the contents of the element to a smaller font than normal text, if practical.

The STRONG element provides strong typographic emphasis, typically bold.

The SUB (Subscript) element specifies that the enclosed text should be displayed as a subscript, and if practical, using a smaller font (compared with normal text). The ALIGN attribute for SUB is only meaningful within the MATH element.

The SUP (Superscript) element specifies that the enclosed text should be displayed as a superscript, and if practical, using a smaller font (compared with normal text). The ALIGN attribute for SUP is only applicable within the MATH element.

The TT (TeleType) element specifies that the enclosed text should be displayed, if practical, in a fixed-pitch typewriter font.

The U (Underline) element specifies that the enclosed text should be displayed, if practical, as underlined.

## Description

HTML distinguishes between two groups of character-formatting tags; Physical character-attribute tags and Logical character-attribute tags. It may help to think of Physical character-attribute tags as closely related to the direct formatting you could apply to text from a word processor, e.g., bold. The appearance of HTML text formatted with Physical character-attribute tags is more likely to remain constant from one browser to another. Physical character-attribute tags may be nested within one another, but browsers may not always be able to combine the different tags predictably. Logical character-attribute tags in HTML can be thought of as "styles" in a word processor, i.e., the appearance of the text formatted with a style in a word processor depends on how that style is defined in that word processor. Similarly, the appearance of HTML text formatted with Logical character-attribute tags depends upon that browser's interpretation of that Logical character-attribute tag.

## Examples

This text contains some ***bold italic*** text, some ~~struck through~~ text and some small print.

HTML source code giving examples of various HTML Physical Font Style Tags.

```
<H2>Physical Font Style Tags produced on Netscape 1.1</H2>
<BLINK>Blink</BLINK><BR>
<B>Bold</B><BR>
<EM>Emphasis</EM><BR>
<I>Italic</I><BR>
<KBD>Keyboard</KBD><BR>
<S>Strikethrough</S><BR>
<STRONG>Strong Emphasis</STRONG><BR>
<SUB>Subscript</SUB><BR>
<SUP>Superscript</SUP><BR>
<TT>Typewriter Text</TT><BR>
<U>Underlined</U><BR>
```

Blink actually does blink in Netscape, but Subscript, Superscript and Underlined are not supported.

# Blink

## HTML 0

### Description

The Blink `<BLINK>` element specifies that the enclosed text should blink. This is supported in Netscape 1.1 and has not been officially incorporated in the HTML 3.0 specification.

### Attributes

None

### Example

```
Click here to <BLINK><A HREF="newfile.zip">download new files</A></BLINK>.
```

# Big

## HTML 0

### Description

The Big <BIG> element changes the physical rendering of the contents of the element to a bigger font than normal text, if practical.

### Attributes

None

### Example

```
This is small<BIG>This is big</BIG>
```



# Bold

## HTML 0

### Description

The Bold `<B>` element specifies that the enclosed text should be displayed in a boldface. If this is not practical, an alternative mapping is allowed.

### Attributes

None

### Example

Click here to `<B><A HREF="newfile.zip">download new files</A></B>`.

# Emphasis

## HTML 0

### Description

The Emphasis `<EM>` element provides typographic emphasis, typically italics. While `<EM>` and `<I>` often give the same effect, use `<EM>` except where it is necessary in the text to refer to the formatting, as in "The italic parts are mandatory". This will help to ensure consistency between documents from various sources if (for example) the reader prefers to use color in place of italics for emphasis.

### Attributes

None

### Example

```
<EM>It is recommended that you register your shareware.</EM>
```

# Italic

## HTML 0

### Description

The Italic `<I>` element specifies that the enclosed text should be displayed, if practical, in an italic font (or slanted).

### Attributes

None

### Example

He found the *<I>dialectic</I>* of hermeneutics historicity redundant historicity while other aspects seemed mere exigencies.

# Keyboard

## HTML 0

### Description

The `<KBD>` element indicates text typed (keyboarded) by the user. It might typically be used in an instruction manual.

### Attributes

None

### Example

Select `<KBD>File</KBD>` and press `<KBD>Enter</KBD>`.

# Strikethrough

## HTML 0

### Description

The Strikethrough `<STRIKE>` element specifies that the enclosed text should be displayed with a horizontal line striking through the text. If this is not practical, an alternative mapping is allowed.

### Attributes

None

### Example

Some aspects of editing should `<S>never</S>` not be revealed.

# Small

## HTML 0

### Description

The Small `<SMALL>` element changes the physical rendering of the contents of the element to a smaller font than normal text, if practical.

### Attributes

None

### Example

```
This is normal <SMALL>This is small</SMALL>
```

# Strong Emphasis

## HTML 0

### Description

The Strong `<STRONG>` element provides strong typographic emphasis, typically bold.

### Attributes

None

### Example

```
<STRONG>It is strongly recommended that you register your  
software.</STRONG>
```

# Subscript

## HTML 0

### Description

The Subscript `<SUB>` element specifies that the enclosed text should be displayed as a subscript, and if practical, using a smaller font (compared with normal text). The `ALIGN` attribute for `SUB` is only meaningful within the `MATH` element.

### Attributes

None

### Example

When you get thirsty your best drink value is `H<SUB>2</SUB>O`.



# Superscript

## HTML 0

### Description

The Superscript<SUP> element specifies that the enclosed text should be displayed as a superscript, and if practical, using a smaller font (compared with normal text). The ALIGN attribute for SUP is only applicable within the MATH element.

### Attributes

None

### Example

Force = Mass X Velocity<SUP>2</SUP>, but E=MC<SUP>2</SUP>.

# Typewriter Text

## HTML 0

### Description

The Typewriter Text<TT> element specifies that the enclosed text should be displayed, if practical, in a fixed-pitch typewriter font.

### Attributes

None

### Example

The message read <TT>All questions must be submitted in writing.</TT>

# Underlined

## HTML 0

### Description

The Underlined `<U>` element specifies that the enclosed text should be displayed, if practical, as underlined. Not widely supported

### Attributes

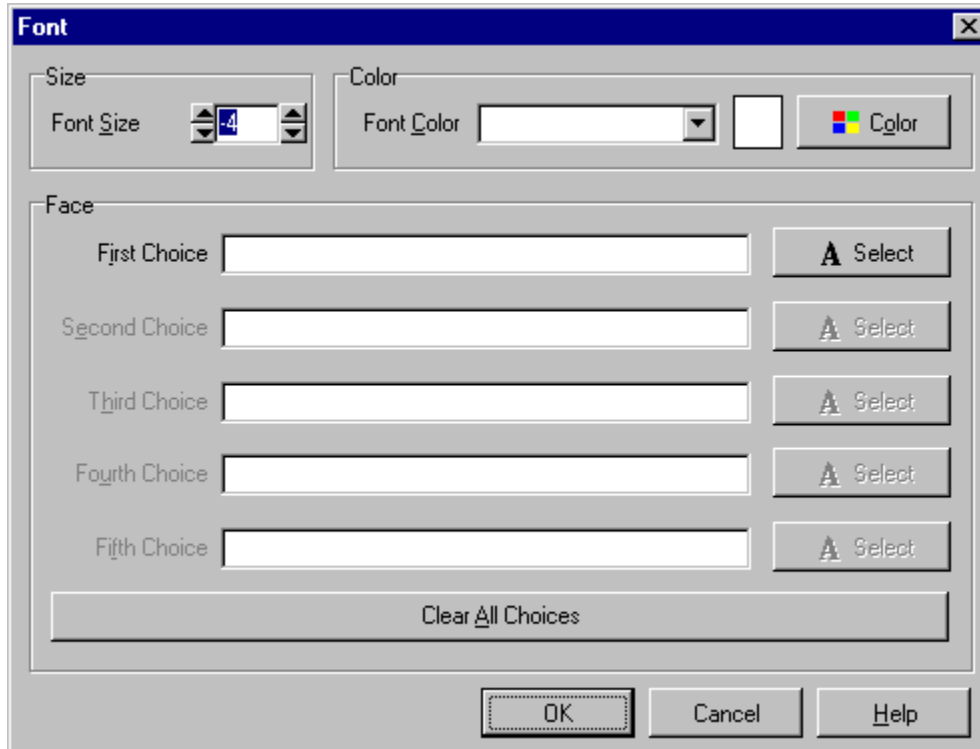
None

### Example

Why didn't `<U>somebody</U>` think of this years ago.

# Font

## HTML 0



### Description

The Font tag sets a font's size and typeface. Within the bounds of the Basefont tag, the Font Size tag may be used to override the Basefont.

### Attributes

**SIZE=*n* SIZE=+*n* or -*n*** **HTML 0** Specifies font size between 1 and 7 (7 is largest). A plus or minus before the number indicates a size relative to the current BASEFONT setting. Note: Relative font sizes are not cumulative. Putting two <FONT SIZE=+1> tags in a row does not result in the font size being increased by 2.

**FACE="name [,name2] [,name3]"** - **HTML 0** Sets the font. A list of font names can be specified. If the first font is available on the system, it will be used, otherwise the second will be tried, and so on. If none are available, a default font will be used.

**COLOR=#rrgbbb or COLOR=color name** - **HTML 0** Sets text color. rrgbbb is a hexadecimal number denoting a red-green-blue color value (the pound sign is optional). Can also be set to a colorname.

### Example.

```
<TD BGCOLOR="Black"></TD>
```

Cell color is black

```
<TD BGCOLOR="#FF0000"></TD>
```

Cell color is red

```
<BASEFONT SIZE=3> This sets the base font size to 3.
```

```
<FONT SIZE=+4> Now the font size is 7.
```

```
<FONT SIZE=-1> Now the font size is 2.
```

```
<FONT SIZE=+2>Now the font size is 5.
```

```
<FONT SIZE=1>Now the font size is 1.
```

**A second example to demonstrate when the tags are effective**

```
<BASEFONT SIZE=4>Some text at the Basefont size of 4.
```

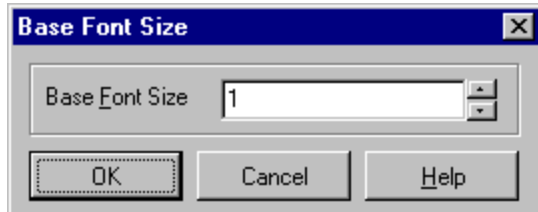
```
<FONT SIZE=7>Some text at the specified font size</FONT>
```

```
And some more text at the Basefont size.</BASEFONT>
```

```
<FONT FACE="Arial, Lucida Sans, Times Roman"> This text will be in  
either Arial, Lucida Sans, or Times Roman, depending on which fonts  
you have installed on your system.</FONT>
```

# Base Font

## HTML 0



### Description

The Basefont tag sets a font size over a range of text. Within the bounds of the Basefont tag, the Font Size tag may be used to override the Basefont.

### Attributes

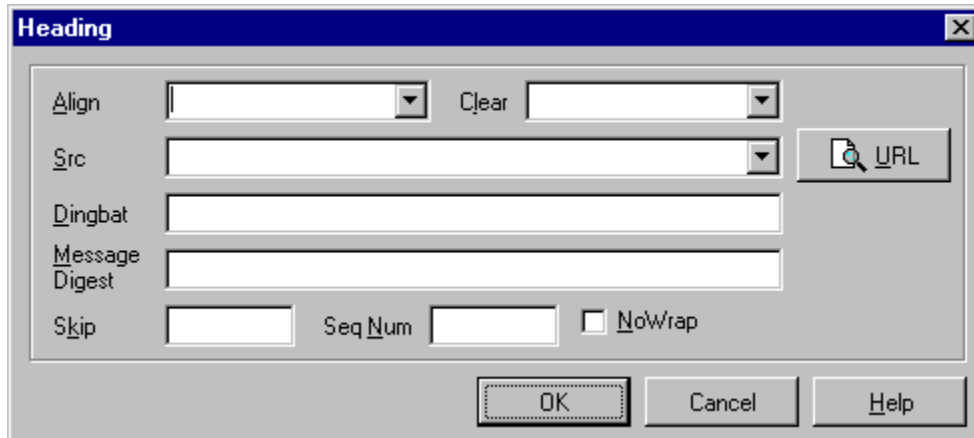
**SIZE=n** - **HTML 0** Specifies font size between 1 and 7 (7 is largest).

### Example

```
<BASEFONT SIZE=4>Some text at the Basefont size of 4.  
<FONT SIZE=7>Some text at the specified font size</FONT>  
And some more text at the Basefont size.
```

# Headings

## HTML 0



### Description

HTML uses six levels of headings, with heading one being the largest and heading six being the smallest. For example the coding for a level six heading would be `<H6>Heading</H6>`. A heading element implies all the font changes, paragraph breaks before and after, and any white space necessary to render the heading. The heading elements are H1, H2, H3, H4, H5, and H6 with H1 being the highest (or most important) level and H6 the least.

Use the DIV element together with header elements when you want to make the hierarchical structure of a document explicit. This is needed as header elements themselves only contain the text of the header, and do not imply any structural division of documents into sections. Header elements have the same content model as paragraphs, that is text and character level markup, such as character emphasis, inline images, form fields and math.

Headers play a related role to lists in structuring documents, and it is common to number headers or to include a graphic that acts like a bullet in lists. HTML 3.0 recognizes this with attributes that assist with numbering headers and allow authors to specify a custom graphic.

The numbering style is controlled by the style sheet, e.g.

The style sheet specifies whether headers are numbered, and which style is used to render the current sequence number, e.g. arabic, upper alpha, lower alpha, upper roman, lower roman or a numbering scheme appropriate to the current language.

Whether the parent numbering is inherited, e.g. "5.1.d" where 5 is the current sequence number for H1 headers, 1 is the number for H2 headers and 4 for H3 headers.

The seqnum and skip attributes can be used to override the default treatment of header sequence numbers, and provide for a continuity with numbered lists.

The dingbat or src attribute may be used to specify a bullet-like graphic to be placed adjacent to the header. The positioning of this graphic is controlled by the style sheet. The graphic is for decorative purposes only and silently ignored on non-graphical HTML user agents.

## Example

```
<HTML>
<HEAD>
<TITLE>Sample Headings</TITLE>
</HEAD>
<BODY>
<H1>Heading 1</H1>
<HR>
<H2>Heading 2</H2>
<HR>
<H3>Heading 3</H3>
<HR>
<H4>Heading 4</H4>
<HR>
<H5>Heading 5</H5>
<HR>
<H6>Heading 6</H6>
<HR>
</BODY>
</HTML>
```



# Miscellaneous Tags

## HTML Miscellaneous Tags

HTML provides the means for defining miscellaneous types of control over the appearance of HTML documents.

The Line Break element specifies that there should be a line break at that tag.

The No Break element specifies that there should be no line breaks for any text between the <NOBR>and</NOBR> tags.

The Word Break element specifies that there should be a break within the <NOBR>and</NOBR> tags.

The Horizontal Rule element specifies that there should be a horizontal line drawn across the screen.

The Comment element specifies that the text between the tags should not be displayed.

The Tab element specifies tab stop positioning.

The BGSound element specifies a background sound to play when the page is displayed.

The Marquee element specifies a scrolling line of text at the tag location.

The Marquee tag specifies that there be a scrolling line of text.

# Line Break

## HTML 0

### Description

The Line Break element and tab elements can be used when you need a little more control over how the browser renders the text. The `<BR>` element is used to force a line break. Remember that browsers are free to wrap lines at whitespace characters so as to ensure lines fit within the current window size. Use the **&nbsp;** entity for the non-breaking space character, when you want to make sure that a line isn't broken! Alternatively, use the NOWRAP attribute to disable word wrapping and the `<BR>` element to force line breaks where desired.

*Netscape includes two tags: `<NOBR>...</NOBR>`, and `<WBR>`. The former turns off word-wrapping between the start and end NOBR tag, while WBR is for the rare case when you want to specify where to break the line if needed. At some point HTML 3.0 may provide an equivalent mechanism to WBR, (either a tag or an entity), but it currently is not implemented.*

**Note:** Do not use empty paragraphs to add white space around headings, lists or other elements. White space is added by the rendering software.

### Example

```
This is the first line<BR>
and this is the second<BR>
and this is the third
```

# No Break

## HTML 0

### Description

*Netscape includes two tags: <NOBR>...</NOBR>, and <WBR>. The former turns off wordwrapping between the start and end NOBR tag, while WBR is for the rare case when you want to specify where to break the line if needed. At some point HTML 3.0 may provide an equivalent mechanism to WBR, (either a tag or an entity), but It currently is not implemented.*

The NOBR element stands for NO BReak. This means all the text between the start and end of the NOBR elements cannot have line breaks inserted between them. While NOBR is essential for those odd character sequences you really don't want broken, please be careful; long text strings inside of NOBR elements can look rather odd.

**Note:** Do not use empty paragraphs to add white space around headings, lists or other elements. White space is added by the rendering software.

# Word Break

## HTML 0

### Description

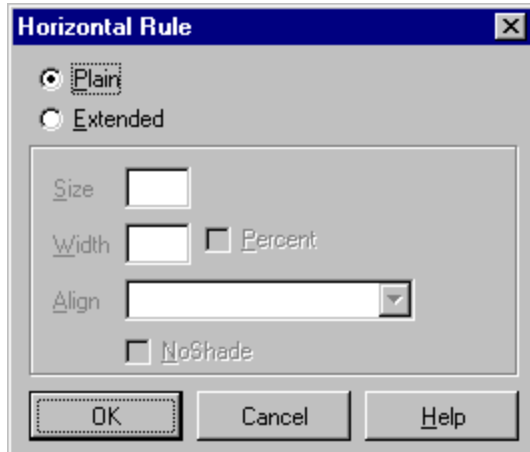
*Netscape includes two tags: <NOBR>...</NOBR>, and <WBR>. The former turns off wordwrapping between the start and end NOBR tag, while WBR is for the rare case when you want to specify where to break the line if needed. At some point HTML 3.0 may provide an equivalent mechanism to WBR, (either a tag or an entity), but It currently is not implemented.*

The WBR element stands for Word BReak. This is for the very rare case when you have a NOBR section and you know exactly where you want it to break. Also, any time you want to give the Netscape Navigator help by telling it where a word is allowed to be broken. The WBR element does not force a line break (BR does that) it simply lets the Netscape Navigator know where a line break is allowed to be inserted if needed.

**Note:** Do not use empty paragraphs to add white space around headings, lists or other elements. White space is added by the rendering software.

# Horizontal Rule

## HTML 0



### Description

The Horizontal Rule `<HR>` element is used for making lines that act as dividers between sections.

### Attributes

**SIZE** - **HTML 0** The SIZE tag lets the author give an indication of how thick they wish the horizontal rule to be. `<HR SIZE=number>`

**WIDTH** - **HTML 0** The default horizontal rule is always as wide as the page. With the WIDTH tag, the author can specify an exact width in pixels, or a relative width measured in percent of document width. `<HR WIDTH=number|percent>`

**ALIGN** - **HTML 0** Now that horizontal rules do not have to be the width of the page we need to allow the author to specify whether they should be pushed up against the left margin, the right margin, or centered in the page. `<HR ALIGN=left|right|center>`

**NOSHADE** - **HTML 0** Finally, for those times when you really want a solid bar, the NOSHADE tag lets you specify that you do not want any fancy shading of your horizontal rule. `<HR NOSHADE>`

# Comment

## HTML 0

### Description

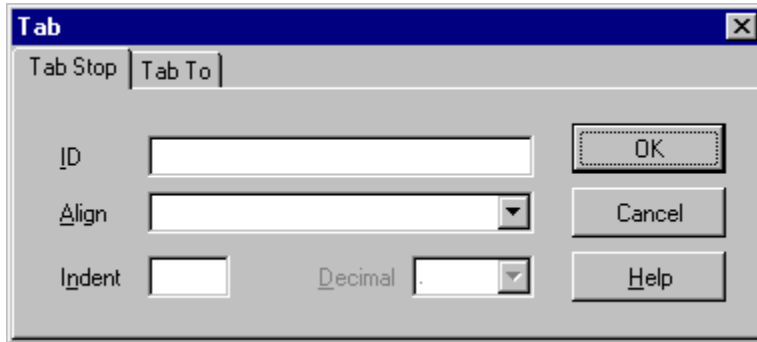
To include comments in an HTML document that will be ignored by the parser, surround them with `<!--` and `-->`. Comments cannot be nested.

### Example

```
<HEAD>  
<TITLE>Comment Usage</TITLE>  
<!-- Id: 06/18/95 11:03:35 AM -->  
</HEAD>
```

# Tabs

## HTML 0



The TAB element can be used when you want fine control over the horizontal positioning. The TAB element is used with the **<tab id=name>** attribute to define named tab stops. Subsequently, you can use the TAB element with the **<tab to=name>** attribute to move to the previously defined tab stop. This approach avoids the need to know the font metrics in advance. The TAB element, together with style sheets, allows conversion software to preserve layout information when importing documents created with conventional word processing software.

### Example

```
<p><b>noct<tab id=t1>ambulant</b> - walking at night<br>
<tab to=t1>(from Latin: <i>nox noctis</i> night + <i>ambulare</i> walk)
```

which is rendered as:

```
noctambulant - walking at night
      (from Latin: nox noctis night + ambulare walk)
```

The tab stop name (t1 in the example) should be unique within the current document and composed from an initial letter followed by letters, digits or hyphens.

Sometimes, you want to make the remainder of the line flush right while leaving the earlier words unmoved. This is possible with the *align* attribute. For example:

```
Left part of line<tab align=right>and right part of line.
```

which is rendered as:

```
Left part of line
```

```
and right part of line
```

### Attributes

**ID** - **HTML 0** An SGML identifier used to name a new tab stop at the current position. The scope of the tab stop is the rest of the document.



**INDENT** - **HTML 0** Specifies the number of en units before the tab stop. The en is a typographical unit equal to half the point size. It allows authors to control the leading indent before text, e.g. in poetry, one might use: `<TAB INDENT=6>` to indent six en units at the start of a line. The **INDENT** attribute is not meaningful when combined with the **TO** attribute.

**TO** - **HTML 0** Specifies a previously defined tab stop (see **ID** attribute).

**ALIGN** - **HTML 0** Lines are usually rendered according to the alignment option for the enclosing paragraph element. The **ALIGN** attribute can be used to explicitly specify the horizontal alignment:

**align=left**

Following text starts immediately after the designated tab stop (the default).

**align=center**

Following text up to next tab or line break is centered on the designated tab stop. If the **TO** attribute is missing, it centers the text between the current left and right margins.

**align=right**

Following text up to the next tab or line break is rendered flush right to the designated tab stop. If the **TO** attribute is missing, it renders the text flush right against the current right margin.

**align=decimal**

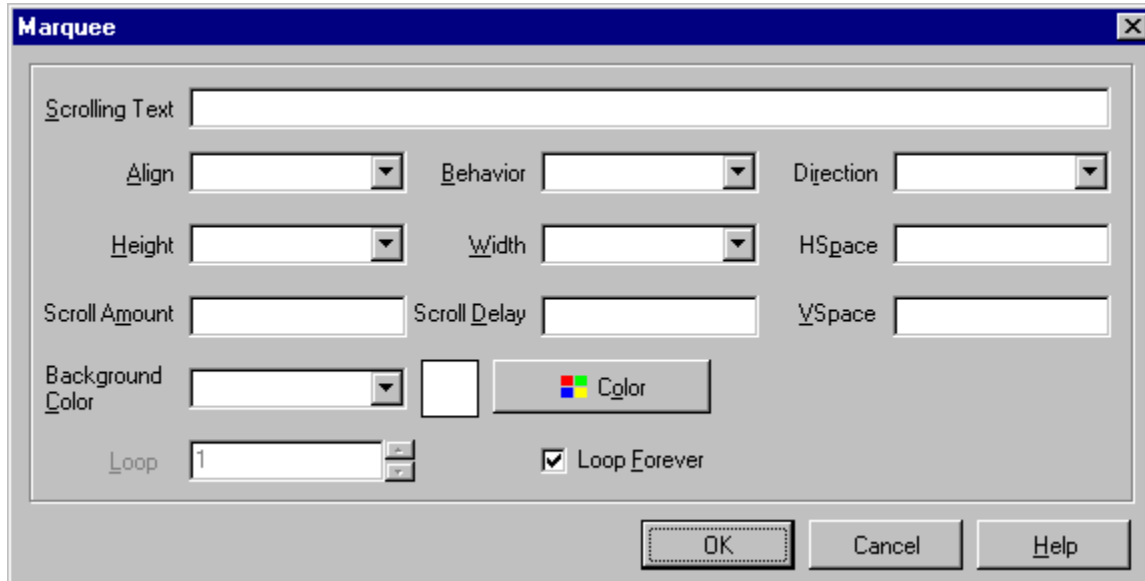
The following text is searched for the first occurrence of the character representing the decimal point. The text up to the next tab or line break is then aligned such that the decimal point starts at the designated tab stop. If the **TO** attribute is missing, the tab element is treated as a single space character.

**DP** - **HTML 0** This specifies the character to be used for the decimal point with the **ALIGN** attribute, e.g. `dp="."` (the default) or `dp=","`. The default may be altered by the language context, as set by the **LANG** attribute on enclosing elements.

**Note:** if the specified alignment and tab stop would cause text to overlap preceding text, then the tab element may be treated as a single space character.

# Marquee

## HTML 0



### Description

The MARQUEE tag allows you to create pages with scrolling text messages.

### Attributes

**ALIGN** - **HTML 0** Specifies how the text around the marquee should align with the top, bottom or middle of the marquee.

```
<MARQUEE ALIGN=TOP>The following words, "Hi there!", will be aligned with the top of this marquee.</MARQUEE> Hi there!
```

**BEHAVIOR** - **HTML 0** Specifies how the text should behave. SCROLL (the default) means start completely off one side, scroll all the way across and completely off, and then start again. SLIDE means start completely off one side, scroll in, and stop as soon as the text touches the other margin. ALTERNATE means bounce back and forth within the marquee.

```
<MARQUEE BEHAVIOR=SCROLL>This text will scroll all the way on and then all the way off.</MARQUEE>
<MARQUEE BEHAVIOR=SLIDE>This marquee will scroll in and "stick."</MARQUEE>
<MARQUEE BEHAVIOR=ALTERNATE>This text will bounce back and forth.</MARQUEE>
```

**BGCOLOR** - **HTML 0** Specifies a background color for the marquee, either as a RGB triple or using a "friendly" colorname

```
<MARQUEE BGCOLOR=#FF0000>This marquee has a red background!</MARQUEE>
```

**DIRECTION** - **HTML 0** Specifies which direction the text should scroll. The default is LEFT, which means scrolling to the left from the right.

```
<MARQUEE DIRECTION=RIGHT>This marquee will scroll from the left in a rightward direction.</MARQUEE>
```

**HEIGHT** - **HTML 0** Specifies the height of the marquee either in percentage of screen size or in pixels.

```
<MARQUEE HEIGHT=50% WIDTH=80%>This marquee is half the height of the screen and 80% of the width.</MARQUEE>
```

**WIDTH** - **HTML 0** Specifies the width of the marquee either in percentage of screen size or in pixels.

```
<MARQUEE HEIGHT=50% WIDTH=80%>This marquee is half the height of the screen and 80% of the width.</MARQUEE>
```

**HSPACE** - **HTML 0** Specifies the left and right margins for the outside of the marquee in pixels.

```
<MARQUEE HSPACE=10 VSPACE=10> This marquee will be separated from the surrounding text by a 10-pixel border.</MARQUEE>
```

**VSPACE** - **HTML 0** Specifies the top and bottom margins for the outside of the marquee in pixels.

```
<MARQUEE HSPACE=10 VSPACE=10>This marquee will be separated from the surrounding text by a 10-pixel border.</MARQUEE>
```

**LOOP** - **HTML 0** Specifies how many times a marquee will loop when activated. If n=-1, or if LOOP=INFINITE is specified, it will loop indefinitely.

```
<MARQUEE LOOP=5>This marquee will loop five times.</MARQUEE>
```

**SCROLLAMOUNT** - **HTML 0** Specifies the number of pixels between each successive draw of the marquee text.

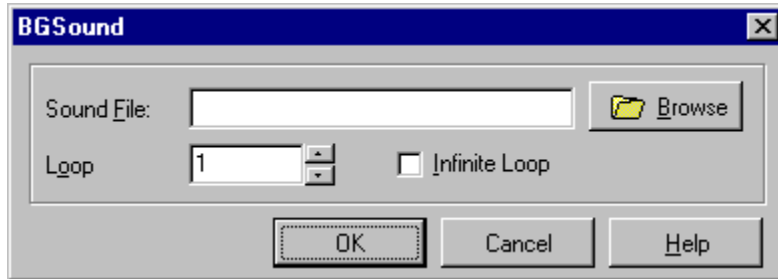
```
<MARQUEE SCROLLDELAY=5 SCROLLAMOUNT=2>This is a very slow marquee.</MARQUEE>
```

**SCROLLDELAY** - **HTML 0** Specifies the number of milliseconds between each successive draw of the marquee text.

```
<MARQUEE SCROLLDELAY=5 SCROLLAMOUNT=50>This is a very fast marquee.</MARQUEE>
```

# Background Sound

## HTML 0



### Description

The BGSOUND tag allows you to create pages with background sounds or "soundtracks." Sounds can either be samples (.wav or .au format) or MIDI (.mid format). Be careful with the size of the sound file you choose (large files take longer to download) - the user of your page might be already on to your next page before the sound loads!

### Attributes

**SRC** - **HTML 0** Specifies the address of a sound to be displayed.

```
<BGSOUND SRC="boing.wav">
```

The user will hear a boinging noise as soon as the page is loaded by the browser.

**LOOP** - **HTML 0** Specifies how many times a sound will loop when activated. If n=-1, or if LOOP=INFINITE is specified, it will loop indefinitely.

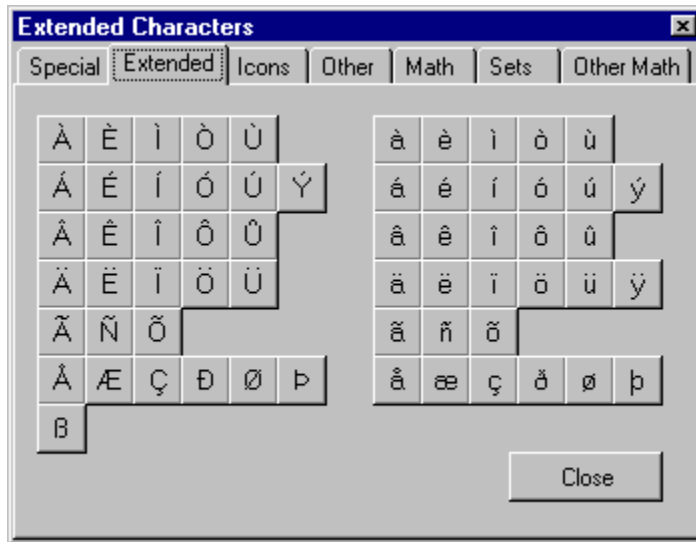
```
<BGSOUND SRC="boing.wav" LOOP=5><BR>
```

You will hear a boinging noise five times in a row.

```
<BGSOUND SRC="boing.wav" LOOP=INFINITE><BR>
```

You will hear boinging noises as long as the page is active.

# Character Format



Extended Characters

Math Symbols

Special Characters

# Special Characters

Certain characters may be interpreted by a Web browser as markup or may not be accessible from your keyboard. WebEdit makes it simple to use these characters by letting you select them from a dialog box. Simply click on the desired character and WebEdit inserts the character as they should be "escaped"; that is, represented by markup -- numeric character or entity references.

## Special Characters

Certain characters are taken to have special meaning within the context of an HTML document.

### Space

Interpreted as a word space in all contexts except <PRE>.

Interpreted as a no-break space within <PRE>.

The character entities &nbsp; and &nbsp; denote an en space and an em space respectively, where an en space is half the point size and an em space is equal to the point size of the current font. For fixed pitch fonts, the user agent can treat the en space as being equivalent to a single space character, and the em space as being equivalent to two space characters.

### Non-breaking Space (&nbsp;)

This should be treated in the same way as the space character (ASCII character code 32 decimal), except that the user agent should never break lines at this point. It is useful when you want to ensure that neighboring words always stay together and don't get split across lines.

### Hyphen

Interpreted as a hyphen glyph in all contexts.

Interpreted as a potential word space by hyphenation engine.

The character entities &dash; and &dash; denote dash marks with the same widths as the &nbsp; and &nbsp; entities respectively.

## Control Characters

Control characters are non-printable characters that are typically used for communication and device control, as format effectors, and as information separators.

In SGML applications, the use of control characters is limited in order to maximize the chance of successful interchange over heterogeneous networks and operating systems. In HTML, there are only three control characters which are used. The remaining 55 control characters are shunned and should not appear in an HTML document. The valid control characters and their interpretation are:

### Horizontal Tab (HT - 9 dec)

Interpreted as a word space in all contexts except <PRE>.

Within <PRE>, the tab should be interpreted to shift the horizontal column position to the next position which is a multiple of 8 on the same line; that is,  $col := (col+8) \bmod 8$ .

### Line Feed (LF - 10 dec)

Interpreted as a word space in all contexts except <PRE>.

Within <PRE>, the tab should be interpreted as a shift to the start of a new line; that is, col := 0; row := row+1

**Carriage Return (CR - 13 dec)**

Interpreted as a word space in all contexts except <PRE>.

Within <PRE>, the tab should be interpreted as a shift to the start of the line; that is, col := 0;

## Extended Characters

Certain characters may be interpreted by a Web browser as markup or may not be accessible from your keyboard. WebEdit makes it simple to use these characters by letting you select them from a dialog box. Simply click on the desired character and WebEdit inserts the character as they should be "escaped"; that is, represented by markup -- numeric character or entity references.

### Additions

**HTML 0**

&reg; -> Registered Trademark -> ®

**HTML 0**

&copy; -> Copyright -> ©

### Numeric Character References

Any printing character within the 8-bit character encoding of ISO 8859/1 (256 character positions) or the 7-bit character encoding of ISO 646 (128 character positions) may be represented within the text of an HTML document by a numeric character reference, e.g. &#233; is a small e with an acute accent. It is recommended that character entity references such as &eacute; are used in preference to numeric character references.



# Math Symbols

## HTML 0

Certain characters may be interpreted by a Web browser as markup or may not be accessible from your keyboard. WebEdit makes it simple to use these characters by letting you select them from a dialog box. Simply click on the desired character and WebEdit inserts the character as they should be "escaped"; that is, represented by markup -- numeric character or entity references.

### Description

The <MATH> element is used to include math expressions in the current line. HTML math is powerful enough to describe the range of math expressions you can create in common word processing packages, as well as being suitable for rendering to speech. When rendering to fixed pitch text-only media, simple text graphics can be used for math symbols such as the integration sign, while other symbols can be rendered using their entity names. The SGML SHORTREF capability is used to provide abbreviations for hidden brackets, subscripts and superscripts.

HTML math follows general practice in mathematical typesetting by rendering functions, numbers and other constants in an upright font, while variables are rendered in an italic font. You can set particular terms in a bold face, and for chemical formulae, you can force the use of an upright font. Limits for symbols like the integral and summation signs are placed directly above (below) the symbol or to the immediate right depending on the symbol.

Spacing between constants, variables and operators is determined automatically. Additional spacing can be inserted with entities such as &thinsp; &sp; and &quadsp;. White space in the markup is used only to delimit adjacent variables or constants. You don't need spaces before or after binary operators or other special symbols, as these are recognized by the HTML math tokeniser. White space can be useful, though, for increased legibility while authoring.

### Math Markup

The following elements are permitted within MATH elements:

**BOX** - Used for hidden brackets, stretchy delimiters, and placing one expression over another (e.g. numerators and denominators).

**SUB, SUP** - Subscripts and superscripts. Also used for limits.

**ABOVE** - Used to draw an arrow, line or symbol above an expression.

**BELOW** - Used to draw an arrow, line or symbol below an expression.

**VEC, BAR, DOT, DDOT, HAT, TILDE** - These are convenience tags for common accents as an alternative to using ABOVE.

**SQRT, ROOT** - For square roots and other roots of an expression.

**ARRAY** - For matrices and other kinds of arrays.

**TEXT** - Used to include a short piece of text within a math element, and often combined with SUB or SUP elements.

**B, T, BT** - These elements are used to override the default rendering. **B** renders the enclosed expression in a bold face. **T** designates a term to be rendered in an upright font, while **BT** designates a term to be rendered in a bold upright font. The class attribute can be used to describe the kind of term, e.g. vector, tensor, or matrix.

## HTML Math Entities

Functions

Operators

Continuation Dots

Greek Letters

Relations

Accents, arrows, and pointers

Delimiters

Other symbols

Spacing entities

**Note:** In practice, only a limited range of font sizes are suitable, as a result, deeply nested expressions like continued fractions can't use ever smaller fonts. This is simply handled by a parameter to the *ParseExpression* routine that sets the font size to be used for that expression. *ParseExpression* is called recursively for nested expressions and uses the next smaller font until it bottoms out with the smallest font available. The size parameter corresponds to an enumeration of the available font sizes.

## Attributes

**ID** - An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**Class** This is a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period.

For the MATH element, CLASS can be used to describe the kind of math expression involved. This can be used to alter the way formulae are rendered, and to support exporting the expression to symbolic math software. The class "chem" is useful for chemical formulae which use an upright font for variables rather than the default italic font.

## Examples

```
<math class=chem> Fe_2_^2+^Cr_2_O_4_</math>
```

which is rendered as 
$$\text{Fe}_2\text{O}_2 + \text{Cr}_2\text{O}_4$$

The integral from  $a$  to  $b$  of  $f(x)$  over  $1+x$

```
<MATH>&int;_a^b^{f(x)<over>1+x} dx</MATH>
```

which can be rendered on a fixed pitch text-only medium as:

```
b
/  f(x)
| ----- dx
/  1 + x
a
```

The example uses { and } as shortrefs for <BOX> and </BOX> respectively. This is used for invisible brackets, stretchy delimiters and integral signs, and placing one thing over another. The shortref characters "\_" and "^" are used for subscripts and superscripts respectively.

# Form Tags

## HTML Form Tags

HTML provides the means for passing information from HTML documents.

The Form element indicates the beginning and end of a form. HTML fill-out forms can be used for questionnaires, etc. The form is specified as part of an HTML document. Every form must be enclosed within a FORM element. There can be several forms in a single document, but the FORM element can't be nested.

The Input element is used to specify a simple input element inside a FORM. It is a standalone tag; it does not surround anything and there is no terminating tag.

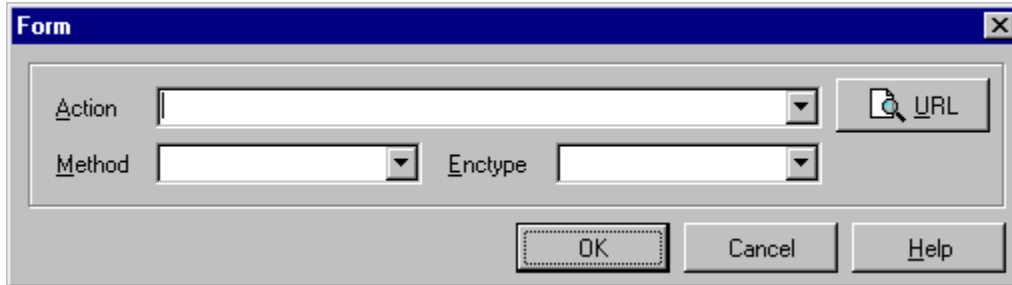
The Option element is used within SELECT tags to present a number of options for the user to select.

The Select element is used inside <FORM>...</FORM>, any number of SELECT tags are allowed, freely intermixed with other HTML elements (including INPUT and TEXTAREA elements) and text (but *not* additional forms).

The Textarea element can be used to place a multiline text entry field with optional default contents in a fill-out form.

# Forms

## HTML 0

A screenshot of a classic Windows-style dialog box titled "Form". The dialog has a blue title bar with a close button (X) on the right. The main area contains four input fields: "Action" (a text box with a dropdown arrow), "Method" (a dropdown menu), "Enctype" (a dropdown menu), and "URL" (a text box with a magnifying glass icon). At the bottom, there are three buttons: "OK" (with a dashed border), "Cancel", and "Help".

### Description

HTML fill-out forms can be used for questionnaires, hotel reservations, order forms, data entry and a wide variety of other applications. The form is specified as part of an HTML document. The user fills in the form and then *submits* it. The user agent then sends the form's contents as designated by the FORM element. Typically, this is to an HTTP server, but you can also email form contents for asynchronous processing.

**Note** You are not allowed to nest FORM elements!

Every form must be enclosed within a FORM element. There can be several forms in a single document, but the FORM element can't be nested. The browser is responsible for handling the input focus, i.e. which field will currently get keyboard input. Many platforms have existing conventions for forms, for example, using Tab and Shift-Tab to move the keyboard focus forwards and backwards between fields, and using the Enter key to submit the form.

This standard defines and requires support for the HTTP access protocol only. Under any protocol, the submitted contents of the form logically consist of a list of name/value pairs where the names are given by the NAME attributes of the various fields in the FORM. Each field will normally be given a distinct name. Several radio buttons can share the same name, as this is how you specify that they belong to the same control group - at any time, only one button in the group can be selected.

**Note** The contents list of name/value pairs excludes unselected radio buttons and checkboxes. In general, any field with a null value can be omitted from the contents list.

Forms are created by placing input fields within paragraphs, preformatted text, lists and tables. This gives considerable flexibility in designing the layout of forms.

HTML 3.0 supports the following kinds of fields:

- Simple text fields
- Multi-line text fields
- Radio buttons
- Checkboxes
- Range controls (sliders, or knobs)
- Single/multiple choice menus
- Scribble on image
- File widgets for attaching files to forms.

- Submit buttons for sending form contents
- Reset buttons for resetting fields to their initial values
- Hidden fields for book keeping information

It is expected that future revisions to HTML will add support for audio fields, multi-row entry of database tables, and extending multi-line text fields to support a range of other data types, in addition to plain text. Client-side scripts will provide the means to constrain field values and to add new field types.

### Attributes

**ACTION** - [HTML 2](#) The ACTION attribute is a URL specifying the location to which the contents of the form is submitted to elicit a response. If the ACTION is missing, the URL for the document itself is assumed. The way data is submitted varies with the access protocol of the URL, and with the values of the METHOD and ENCTYPE attributes.

**METHOD** - [HTML 2](#) This specifies variations in the protocol used to send the form contents. It is currently restricted to GET (the default) or POST. The attribute was introduced to inform user agents which HTTP methods the server supports.

**ENCTYPE** - [HTML 2](#) This attribute specifies the [MIME](#) content type to be used to encode the form contents. It defaults to the string: "application/x-www-form-urlencoded"

**SCRIPT** - This can be used to give a URI for a script. The scripting language and the interface with the user agent is not part of the HTML 3.0 specification.

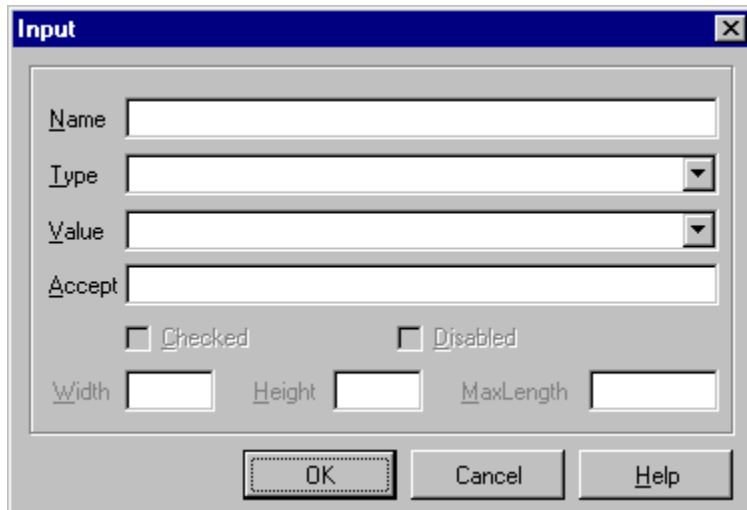
### Example

This fictitious example is a questionnaire. It uses the INPUT element for simple text fields, radio buttons, checkboxes, and the submit and reset buttons. The TEXTAREA field is used for a multi-line text entry field. The form fields are laid out with several paragraph elements and an unordered list. Notice the use of the NAME attribute to name each field:

```
<TITLE>Sample Questionnaire</TITLE>
<H1>Sample Questionnaire</H1>
<P>Please fill out this questionnaire:
<FORM METHOD=post ACTION="http://www.hal.com/sample">
<P>Your name: <input name="name" size="48">
<P><input name="male" type=radio> Male
<P><input name="female" type=radio>Female
  Number in family: <input name="family" type=int>
<P>Cities in which you maintain a residence:
<UL PLAIN>
<LI><input name="city" type=checkbox value="kent"> Kent
<LI><input name="city" type=checkbox value="miami"> Miami
<LI>Others <textarea name="other" cols=48 rows=4></textarea>
</UL>
<P>Nickname: <INPUT NAME="nickname" size ="42">
<P>Thank you for responding to this questionnaire.
<P><INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
</FORM>
```

# Input

## HTML 0



### Description

The INPUT element is used to specify a simple input element inside a FORM. It is a standalone tag; it does not surround anything and there is no terminating tag.

### Attributes

**TYPE** - **HTML 0** must be one of:

- "text" (text entry field; this is the default)
- "password" (text entry field; entered characters are represented as asterisks)
- "checkbox" (a single toggle button; on or off)
- "radio" (a single toggle button; on or off; other toggles with the same NAME are grouped into "one of many" behavior)
- "submit" (a pushbutton that causes the current form to be packaged up into a query URL and sent to a remote server)
- "reset" (a pushbutton that causes the various input elements in the form to be reset to their default values)

**NAME** - **HTML 0** the symbolic name (not a displayed name -- normal HTML within the form is used for that) for this input field. *This must be present for all types except "submit" and "reset", as it is used when putting together the query string that gets sent to the remote server when the filled-out form is submitted.*

**VALUE** - **HTML 0** for a text or password entry field, can be used to specify the default contents of the field. For a checkbox or a radio button, Value specifies the value of the button *when it is checked* (unchecked checkboxes are disregarded when submitting queries); can be used to specify the label for the pushbutton.

**CHECKED** - **HTML 0** (no value needed) specifies that this checkbox or radio button is checked by default; *this is only appropriate for checkboxes and radio buttons.*

**SIZE** - **HTML 0** the physical size of the input field in characters; *this is only appropriate for text entry fields and password entry fields.* If this is not present, the default is 20. Multiline text entry fields can be specified as SIZE=width,height; e.g., SIZE=60,12. **Note:** the SIZE attribute should not be used to specify multiline text entry fields now that the TEXTAREA tag is available.

**MAXLENGTH** - **HTML 0** the maximum number of characters that are accepted as input; *this is only appropriate for text entry fields and password entry fields* (and only for single-line text entry fields at that). If this is not present, the default will be unlimited. The text entry field is assumed to scroll appropriately if MAXLENGTH is greater than SIZE.

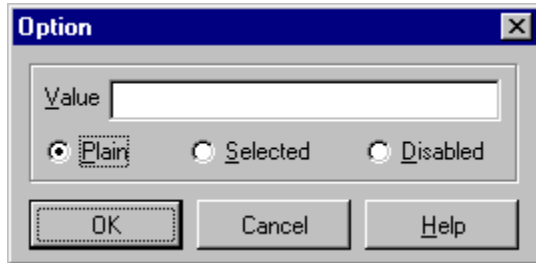
### Example

```
<TITLE>Sample Questionnaire</TITLE>
<H1>Sample Questionnaire</H1>
<P>Please fill out this questionnaire:
<FORM METHOD=post ACTION="http://www.hal.com/sample">
<P>Your name: <input name="name" size="48">
<P><input name="sex" type=radio value="male"> Male
<P><input name="sex" type=radio value="female"> Female
  Number in family: <input name="family" type=int>
<P>Cities in which you maintain a residence:
<UL PLAIN>
<LI><input name="city" type=checkbox value="kent"> Kent
<LI><input name="city" type=checkbox value="miami"> Miami
<LI>Others <textarea name="other" cols=48 rows=4></textarea>
</UL>
<P>Nickname: <INPUT NAME="nickname" size ="42">
<P>Thank you for responding to this questionnaire.
<P><INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
</FORM>
```



# Option

## HTML 0



### Description

OPTION tags are used within SELECT tags to present a number of options for the user to select.

### Attributes

**PLAIN** - **HTML 0** specifies that this option is not disabled or specified.

**SELECTED** - **HTML 0** specifies that this option is selected by default. If the SELECT allows multiple selections (via the MULTIPLE attribute), multiple options can be specified as SELECTED.

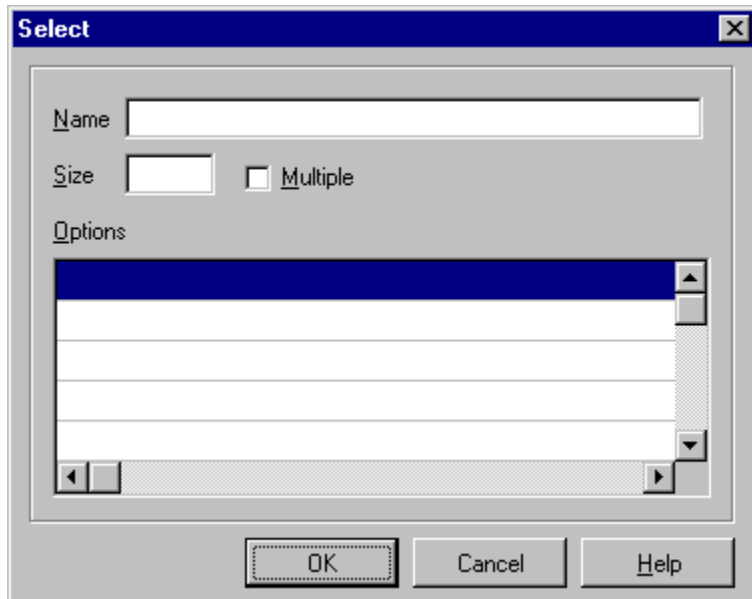
**DISABLED** - **HTML 0** specifies that this option is visible but not selectable. It should appear "greyed-out".

### Example

```
<SELECT NAME="FooBar">
<OPTION>First Option
<OPTION SELECTED>Second Option
<OPTION DISABLED>Third Option
</SELECT>
```

# Select

## HTML 0



### Description

Inside `<FORM>...</FORM>`, any number of `SELECT` tags are allowed, freely intermixed with other HTML elements (including `INPUT` and `TEXTAREA` elements) and text (but *not* additional forms).

Unlike `INPUT`, `SELECT` has both opening and closing tags. Inside `SELECT`, only a sequence of `OPTION` tags -- each followed by an arbitrary amount of plain text (*no HTML markup*) -- is allowed.

### Attributes

**NAME** - **HTML 0** the symbolic name for this `SELECT` element. This must be present, as it is used when putting together the query string for the submitted form.

**SIZE** - **HTML 0** if `SIZE` is 1 or if the `SIZE` attribute is missing, by default the `SELECT` will be

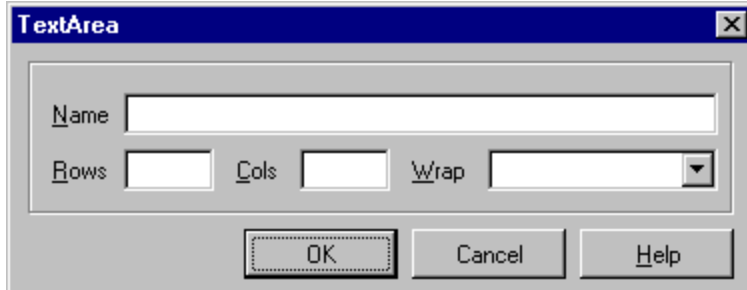
**MULTIPLE** - **HTML 0** if present (no value), specifies that the `SELECT` should allow multiple selections (n of many behavior). The presence of `MULTIPLE` forces the `SELECT` to be represented by a scrolled list, regardless of the value of `SIZE`.

### Example

```
<SELECT NAME="FooBar">
<OPTION>First Option
<OPTION SELECTED>Second Option
<OPTION DISABLED>Third Option
</SELECT>
```

# Text Area

## HTML 0



### Description

The TEXTAREA tag can be used to place a multiline text entry field with optional default contents in a fill-out form.

### Attributes

**NAME** - **HTML 0** the symbolic name of the text entry field.

**ROWS** - **HTML 0** the number of rows (vertical height in characters) of the text entry field.

**COLS** - **HTML 0** the number of columns (horizontal width in characters) of the text entry field.

**WRAP** - **HTML 0** Specifies how to handle word-wrapping in text input areas in forms.

`<TEXTAREA WRAP=OFF>` -- the default setting

Wrapping doesn't happen. Lines are sent exactly as typed.

`<TEXTAREA WRAP=VIRTUAL>`

The display word-wraps, but long lines are sent as one line without new-lines.

`<TEXTAREA WRAP=PHYSICAL>`

The display word-wraps, and the text is transmitted at all wrap points.

TEXTAREA fields automatically have scrollbars; any amount of text can be entered in them.

The TEXTAREA element *requires* both an opening and a closing tag. A TEXTAREA with no default contents looks like this:

```
<TEXTAREA NAME="foobar" ROWS=3 COLS=20></TEXTAREA>
```

A TEXTAREA with default contents looks like this

```
<TEXTAREA NAME="foobar2" ROWS=4 COLS=20>
Default contents go here
</TEXTAREA>
```

The default contents must be straight ASCII text. New lines are respected (so in the above example there will be a new line both before and after "Default contents go

here.").

# List Tags

## HTML List Tag Formats

HTML provides the means for defining different types of lists in HTML documents.

The Ordered List element specifies that the following list items should be ordered, i.e., numbered. WebEdit provides a dialog for selecting the type of numbering to be used.

The Unordered List element specifies that the following list items should be unordered, i.e., unnumbered, (bulleted). WebEdit provides a dialog for selecting the type of bulleting to be used.

The Menu List element specifies that the following list items should appear as simple bulleted menu items.

The Directory List element specifies that the following list items should appear as simple bulleted menu items.

The List Item element specifies each of the items included in a certain list.

The Definition List element specifies that the following items will include Definition Titles and Definition Items, i.e., words and their definitions.

The Definition Title element specifies that the following item is a Definition Title, i.e., a word to be defined.

The Definition Item element specifies that the following item is a Definition Item, i.e., the definition of a word (Definition Title).

# Ordered List

## HTML 0

### Description

An ordered list typically is a numbered list of items. Additions to HTML 0 give you the ability to control the sequence number - to continue where the previous list left off, or to start at a particular number. The numbering style is left to associated style sheets, e.g. whether nested lists contribute to a compound item number, e.g. "3.1.5", or whether numbers are rendered as arabic, upper or lower case roman numerals or using the numbering scheme appropriate to the language context.

The opening list tag must be `<OL>`. It is followed by an optional list header (`<LH>caption</LH>`) and then by the first list item (`<LI>`).

### Example:

```
<OL>
  <LH>Meeting Agenda</LH>
  <LI>Minutes of the last meeting
  <LI>Do we need yet more meetings?
  <LI>Any other business
</OL>
```

which could be rendered as:

```
Meeting Agenda
1. Minutes of the last meeting
2. Do we need yet more meetings?
3. Any other business
```

### Attributes

**ID** - **HTML 0** An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**LANG** - **HTML 0** This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - **HTML 0** This is a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

**CLEAR** - **HTML 0** This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want

to start an element like a header, paragraph or list below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

**clear=left** move down until left margin is clear

**clear=right** move down until right margin is clear

**clear=all** move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

**clear="40 en"** move down until there is at least 40 en units free

**clear="100 pixels"** move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

**CONTINUE** - **HTML 0** Don't restart the sequence number, i.e. continue where previous list left off, e.g. <OL CONTINUE>

**SEQNUM** - **HTML 0** Set the starting sequence number for the first item, e.g. <OL SEQNUM=23>

**COMPACT** - The presence of this attribute indicates the user agent should use reduced inter-item spacing. In practice, there are several ways to increase the compactness of lists: reduced vertical interitem spacing, smaller font size, or even to avoid line breaks between items. This is best handled through associated style sheets and the class attribute.

**TYPE** - **HTML 0** Specifies whether list items are marked with: capital letters (TYPE=A), small letters (TYPE=a), large roman numerals (TYPE=I), small roman numerals (TYPE=i), or the default numbers (TYPE=1).

**START** - **HTML 0** Specifies the starting count. START is always specified in the default numbers, and will be converted based on TYPE before display. Thus START=5 would display either an 'E', 'e', 'V', 'v', or '5' based on the TYPE tag.

# Un-ordered List

## HTML 0

### Description

An unordered list typically is a bulleted list of items. HTML 3.0 gives you the ability to customize the bullets, to do without bullets and to wrap list items horizontally or vertically for multicolumn lists.

The opening list tag must be `<UL>`. It is followed by an optional list header (`<LH>caption</LH>`) and then by the first list item (`<LI>`).

### Example

```
<UL>
  <LH>Table Fruit</LH>
  <LI>apples
  <LI>oranges
  <LI>bananas
</UL>
```

which could be rendered as:

```
Table Fruit
•   apples
•   oranges
•   bananas
```

**Note:** Some legacy documents may include headers or plain text before the first LI element. Implementors of HTML 3.0 user agents are advised to cater for this possibility in order to handle badly formed legacy documents.

### Attributes

**ID** - **HTML 0** An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**LANG** - **HTML 0** This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - **HTML 0** This is a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

**CLEAR** - **HTML 0** This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want



to start an element like a header, paragraph or list below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

**clear=left** move down until left margin is clear

**clear=right** move down until right margin is clear

**clear=all** move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

**clear="40 en"** move down until there is at least 40 en units free

**clear="100 pixels"** move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

**PLAIN** - **HTML 0** The presence of this attribute suppresses the display of bullets, e.g. <UL PLAIN>.

**SRC** - **HTML 0** Specifies an image for use as a bullet. The image is specified as a URI. This attribute may appear together with the MD attribute.

**MD** - **HTML 0** Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQITDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

**DINGBAT** - **HTML 0** Specifies an iconic image for use as a bullet. The icon is specified as an entity name.

**WRAP** - **HTML 0** The WRAP attribute is used for multicolumn lists. Use wrap=vert if you want to arrange the list items down the page before wrapping to the next column. Use wrap=horiz if you want to arrange the items across the page (less useful). The user agent is responsible for determining how many columns are appropriate.

**COMPACT** - **HTML 0** The presence of this attribute indicates the user agent should use reduced interitem spacing. In practice, there are several ways to increase the compactness of lists: reduced vertical interitem spacing, smaller font size, or even to avoid line breaks between items. This is best handled through associated style sheets and the class attribute.

**TYPE** - **HTML 0** Specifies the shape of the bullet.

# Menu List

## HTML 0

### Description

Both MENU and DIR consist of one or more LI elements, similar to UL. MENU lists are typically rendered without bullets in a more compact style than UL. You can get the same effect with <UL PLAIN>.

### Example

```
<MENU>Minimum Configuration  
<LI>Dual Pentium 150  
<LI>9 Gig Hard Drive  
<LI>8 Meg Video Card  
</MENU>
```

# Directory List

## HTML 0

### Description

DIR lists are used to present lists of items containing up to 20 characters each. Items in a DIR list are arranged in columns. You can get the same effect with `<UL PLAIN WRAP=HORIZ>`.

### Example

```
<DIR>List Of Items  
<LI>Apples  
<LI>Oranges  
<LI>Peaches</DIR>
```

# List Item

## HTML 0

### Description

The List Item element specifies each of the items included in any of the list types except the Definition List.

### Example

```
<DIR>List Of Items  
<LI>Peaches  
<LI>Pears  
<LI>Prunes</DIR>
```

# Definition List

## HTML 0

### Description

A definition list is a list of terms and corresponding definitions. Definition lists are typically formatted with the term on the left with the definition following on the right or on the next line. The definition text is typically indented with respect to the term.

An alternative format places the term left aligned in a wide margin and the definition on one or more lines to the right of the term. If the DT term does not fit in the DT column (one third of the display area), it may be extended across the page with the DD section moved to the next line, or it may be wrapped onto successive lines of the left hand column.

The opening list tag must be `<DL>`. It is followed by an optional list header (`<LH>caption</LH>`) and then by term names (`<DT>`) and definitions (`<DD>`).

### Example:

```
<DL>
  <LH>List Header</LH>
  <DT>Term 1<dd>This is the definition of the first term.
  <DT>Term 2<dd>This is the definition of the second term.
</DL>
```

which could be rendered as:

### List Header

Term 1    This is the definition of the first term.

Term 2    This is the definition of the second term.

The definition list element can take the `COMPACT` attribute, which suggests that a compact rendering be used, and is appropriate if the list elements are small and/or the entire list is large.

**Note:** Use the `NOTE` element when you want to have an indented note. The practice of using `<DD>` elements without corresponding `<DT>` elements is deprecated.

### Attributes

**ID** - **HTML 0** An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**LANG** - **HTML 0** This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select

language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - **HTML 0** This is a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names.

**CLEAR** - **HTML 0** This attribute is common to all block-like elements. When text flows around a figure or table in the margin, you sometimes want to start an element like a header, paragraph or list below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

**clear=left**                    move down until left margin is clear  
**clear=right**                move down until right margin is clear  
**clear=all**                    move down until both margins are clear

Alternatively, you can decide to place the element alongside the figure just so long as there is enough room. The minimum width needed is specified as:

**clear="40 en"**    move down until there is at least 40 en units free  
**clear="100 pixels"**    move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

**COMPACT** - **HTML 0** The presence of this attribute indicates the user agent should use reduced interitem spacing. The COMPACT attribute may also reduce the width of the left-hand (DT) column. In practice, there are several ways to increase the compactness of lists: reduced vertical interitem spacing, smaller font size, or even to avoid line breaks between items. This is best handled through associated style sheets and the class attribute. The opening list tag must be DL COMPACT. It must be immediately followed by the first term (DT).

### Example

```
<DL compact>
<DT>Term<DD>This is the first definition in compact format.
<DT>Term<DD>This is the second definition in compact format.

</DL>
```

# Definition Title

## HTML 0

### Description

The Definition Title <DT> element specifies that the following item is a Definition Title, i.e., a word to be defined.

### Example

```
<DL>
  <LH>List Header</LH>
  <DT>Term 1<dd>This is the definition of the first term.
  <DT>Term 2<dd>This is the definition of the second term.
</DL>
```

which could be rendered as:

### List Header

Term 1    This is the definition of the first term.

Term 2    This is the definition of the second term.

# Definition Item

## HTML 0

### Description

The Definition Item `<DD>` element specifies that the following item is a Definition Item, i.e., the definition of a word (Definition Title).

### Example

```
<DL>
  <LH>List Header</LH>
  <DT>Term 1<dd>This is the definition of the first term.
  <DT>Term 2<dd>This is the definition of the second term.
</DL>
```

which could be rendered as:

### List Header

Term 1    This is the definition of the first term.

Term 2    This is the definition of the second term.



# Table Specification

## HTML Table Specification Formats

HTML provides for the arranging of data into tables for clearer displays of information.

The Table element by default sizes a table automatically according to the cell contents and the current window size.

The Caption element is optional and used at the start of a table followed by one or more rows.

The Table Header element is used for naming columns and rows of data.

The Table Data element specifies information to be included in a certain cell.

The Table Row element determines the row count - any rows implied by cells spanning rows beyond this should be ignored.

## Description

The HTML table model was chosen for its simplicity and flexibility. By default the table is automatically sized according to the cell contents and the current window size. The COLSPEC attribute can be used when needed to exert control over column widths, either by setting explicit widths or by specifying relative widths. You can also specify the table width explicitly or as a fraction of the current margins (see WIDTH attribute).

Tables start with an optional caption followed by one or more rows. Each row is formed by one or more cells, which are differentiated into header and data cells. Cells can be merged across rows and columns, and include attributes assisting rendering to speech and Braille, or for exporting table data into databases. The model provides little direct support for control over appearance, for example border styles and margins, as these can be handled via subclassing and associated style sheets.

Tables can contain a wide range of content, such as headers, lists, paragraphs, forms, figures, preformatted text and even nested tables. When the table is flush left or right, subsequent elements will be flowed around the table if there is sufficient room. This behavior is disabled when the *noflow* attribute is given or the table align attribute is *center* (the default), or *justify*.

## Example

```
<TABLE BORDER>
  <CAPTION>A test table with merged cells</CAPTION>
  <TR><TH ROWSPAN=2><TH COLSPAN=2>Average
    <TH ROWSPAN=2>other<BR>category<TH>Misc
  <TR><TH>height<TH>weight
  <TR><TH ALIGN=LEFT>males<TD>1.9<TD>0.003
  <TR><TH ALIGN=LEFT ROWSPAN=2>females<TD>1.7<TD>0.002
</TABLE>
```

There are several points to note:

- By default, header cells are centered while data cells are flush left. This can be overridden by the ALIGN attribute for the cell; the COLSPEC attribute for the TABLE element; or the ALIGN attribute on the enclosing row's TR element (from the most specific to the least).
- Cells may be empty.
- Cells spanning rows contribute to the column count on each of the spanned rows, but only appear in the markup once (in the first row spanned).
- If the column count for the table is greater than the number of cells for a given

row (after including cells for spanned rows), the missing cells are treated as occurring on the right hand side of the table, and rendered as empty cells.

- The row count is determined by the TR elements - any rows implied by cells spanning rows beyond this should be ignored.
- The user agent should be able to recover from a missing <TR> tag prior to the first row as the TH and TC elements can only occur within the TR element.
- It is invalid to have cells overlap, see below for an example. In such cases, the rendering is implementation dependent.
- Table data can be lists, images, Forms, and other elements.
- TH element is typically rendered as bold text.
- TD element is typically rendered as regular weight text.
- Always use TR's as "holders" for TH's and TD's.
- The browser sets the number of columns in a table to be the greatest number of columns in all the row. Blank cells are used to fill any extra columns in the rows.

### An example of an invalid table:

```
<table border>
<tr><td rowspan=2>1<td>2<td>3<td>4<td>5
<tr><td rowspan=2>6
<tr><td colspan=2>7<td>8
</table>
```

which looks something like:

```
/-----| 1 | 2 | 3 | 4 | 5 |
| |-----|
| | 6 | | | | | The cells labeled 6 and 7 overlap!
|---|...|-----|
| 7 : | 8 | | | |
\-----/
```

Borderless tables are useful for layout purposes as well as their traditional role for tabular data, for instance with fill-out forms:

```
name: [John Smith]
card number: [4619 693523 20851]
expires: [03] / [97]
telephone: [212 873 2739]
```

This can be represented as a table with one row and two columns. The first column is right aligned, while the second is left aligned. This example could be marked up as:

```
<table>
<tr valign=baseline>
<td align=right>
name:<br>
card number:<br>
expires:<br>
telephone:
<td align=left>
<input name="name" size=18><br>
<input name="cardnum" size=18><br>
<input name="expires-month" size=2> /
<input name="expires-year" size=2><br>
<input name="phone" size=18><br>
</table>
```

The use of such techniques is one of the motivations for using nested tables, where borderless tables are used to layout cell contents for an enclosing table

**Hint:** You can achieve a similar effect to the above by using decimal alignment and using the DP attribute to set the alignment character to a convenient character, for example:

```
<table>
  <tr align=decimal dp=":">
  <td>
    name: <input name="name" size=18><br>
    card number: <input name="cardnum" size=18><br>
    expires: <input name="expires-month" size=2> /
    <input name="expires-year" size=2><br>
    telephone:<input name="phone" size=18><br>
  </td>
</table>
```

Each line in the table is then indented so that all the colons are positioned under one another.

### Attributes

**ID** - **HTML 0** An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**LANG** - **HTML 0** This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - **HTML 0** This is a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

**CLEAR** - **HTML 0** When there is a figure or another table in the margin, you sometimes want to start another table below the figure rather than alongside it. The CLEAR attribute allows you to move down unconditionally:

**clear=left**      move down until left margin is clear

**clear=right**    move down until right margin is clear

**clear=all**      move down until both margins are clear

Alternatively, you can decide to place the table alongside the figure just so long as there is enough room. The minimum width needed is specified as:

**clear="40 en"**      move down until there is at least 40 en units free

**clear="100 pixels"** move down until there is at least 100 pixels free

The style sheet (or browser defaults) may provide default minimum widths for each class of block-like elements.

**NOFLOW** - **HTML 0** The presence of this attribute disables text flow around the table. It avoids the need to use the CLEAR or NEEDS attributes on the following element.

**ALIGN** - **HTML 0** Specifies horizontal alignment of the table (*not* its contents):

**BLEEDLEFT** - **HTML 0** Flush left with the left (window) border.

**LEFT** - **HTML 0** Flush left with the left text margin.

**CENTER** - **HTML 0** The table is centered between the text margins and text flow around the table is disabled. This is the default setting for ALIGN.

**RIGHT** - **HTML 0** Flush right with the right text margin.

**BLEEDRIGHT** - **HTML 0** Flush right with the right (window) border

**JUSTIFY** - **HTML 0** When applicable the table should be sized to fill the space between the left and right text margins. Text flow around the table is disabled for align=justify.

**UNITS** - **HTML 0** Specifies the choice of units for the COLSPEC attribute:  
**units=en** - Specifies en units (a typographical unit equal to half the point size). This is the default setting and allows user agents to render the table a row at a time without waiting until all of the table's data has been received.

**units=relative** - Used to set the relative width of columns. The user agent sums the values to determine the proportional width of each column.

**units=pixels** - The least useful!

A design issue for user agents is how to handle cases where cell contents won't fit into the specified column widths. One approach is to clip the contents to the given column width, another is to resize the columns to fit the contents regardless of the COLSPEC attribute (its best to wait until all of the table's data has been processed before resizing).

**COLSPEC** - **HTML 0** The colspec attribute is a list of column widths and alignment specifications. The columns are listed from left to right with a capital letter followed by a number, e.g. COLSPEC="L20 C8 L40". The letter is L for left, C for center, R for right alignment of cell contents. J is for justification, when feasible, otherwise this is treated in the same way as L for left alignment. D is for decimal alignment, see DP attribute. Capital letters are required to avoid a particularly common error when a lower case L is confused with a one. Column entries are delimited by one or more space characters. The number specifies the width in en's, pixels or as a fractional value of the table width, as according to the associated units attribute. This approach is more compact than used with most SGML table models and chosen to simplify hand entry. The width attribute allows you to specify the width of the table in pixels, em units or as a percentage of the space between the current left and right margins.

**DP** - This specifies the character to be used for the decimal point with the COLSPEC attribute, e.g. dp="." (the default) or dp=", ". The default may be altered by the language context, as set by the LANG attribute on enclosing elements.

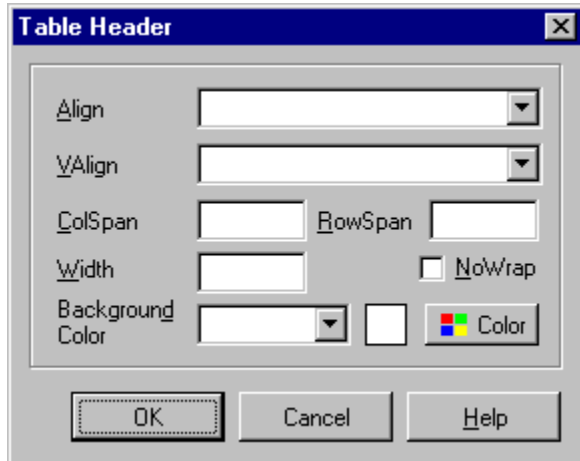
**WIDTH** - **HTML 0** This specifies the width of the table according to the UNITS attribute. If units=relative, the width is taken as a percentage of the width between the current left and right margins. The user agent should disregard this attribute if it would result in columns having less than their minimum widths.

**BORDER** - **HTML 0** This presence of this attribute instructs the user agent to render borders around tables. For instance: <TABLE BORDER>. The precise appearance, along with the size of margins around cells, can be controlled by associated style sheets, or via information in the STYLE element in the document head. Subclassing tables, rows and cells is particularly useful in this regard.

**NOWRAP** - The NOWRAP attribute is used when you don't want the browser to automatically wrap lines. You can then explicitly specify line breaks in paragraphs using the BR element.

# Table Header

## HTML 0



### Description

The Table Header element <TH> is used for naming columns and rows of data.

### Attributes

**ALIGN** - **HTML 0** The ALIGN attribute can be used to explicitly specify the horizontal alignment of paragraphs within a table row:

**align=left** - Paragraphs are rendered flush left (the default).

**align=center** - Paragraphs are centered.

**align=right** - Paragraphs are rendered flush right.

**align=justify** - Text lines are justified where practical, otherwise this gives the same effect as the default align=left setting.

**VALIGN** - **HTML 0** vertical alignment of material in a cell. Values include "top," "middle," "bottom," "baseline."

**COLSPAN** - **HTML 0** the number of columns the cell spans.

**ROWSPAN** - **HTML 0** the number of rows the cell spans.

**NO WRAP** - **HTML 0** prevents the browser from wrapping the contents of the cell.

**COLOR=#rrggbb** or **COLOR=color name** - **Internet Explorer 1.0** Sets cell color. rrggbb is a hexadecimal number denoting a red-green-blue color value (the pound sign is optional). Can also be set to a colorname.

Example:

```
<TD BGCOLOR="Black"></TD>
```

Cell color is black

```
<TD BGCOLOR="#FF0000"></TD>
```

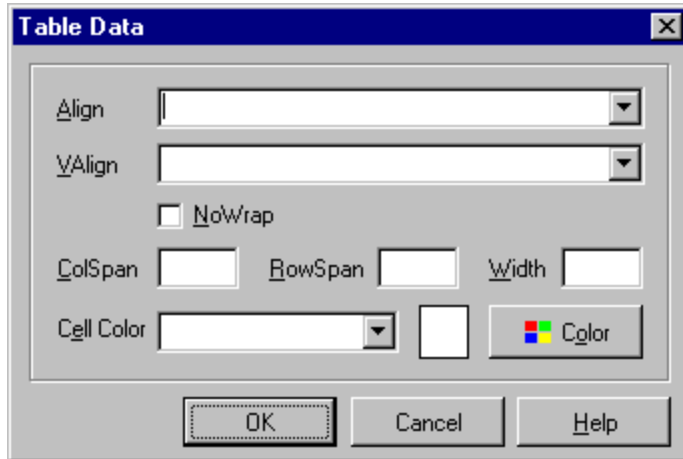
Cell color is red

**Example**

```
<TH>Widgets</TH>
```

# Table Data

## HTML 0



### Description

The Table Data element is for inserting a data point into a Table.

### Attributes

**ALIGN** - **HTML 0** The ALIGN attribute can be used to explicitly specify the horizontal alignment of paragraphs within a table row:

**align=left** - Paragraphs are rendered flush left (the default).

**align=center** - Paragraphs are centered.

**align=right** - Paragraphs are rendered flush right.

**align=justify** - Text lines are justified where practical, otherwise this gives the same effect as the default align=left setting.

**VALIGN** - **HTML 0** vertical alignment of material in a cell. Values include "top," "middle," "bottom," "baseline."

**COLSPAN** - **HTML 0** the number of columns the cell spans.

**ROWSPAN** - **HTML 0** the number of rows the cell spans.

**NO WRAP** - **HTML 0** prevents the browser from wrapping the contents of the cell.

**WIDTH** - **HTML 0** This specifies the width of the table according to the UNITS attribute. If units=relative, the width is taken as a percentage of the width between the current left and right margins. The user agent should disregard this attribute if it would result in columns having less than their minimum widths.

**COLOR=#rrggbb** or **COLOR=color name** - **HTML 0** Sets cell color. rrggbb is a hexadecimal number denoting a red-green-blue color value (the pound sign is optional). Can also be set



to a colorname.

**Example:**

```
<TD BGCOLOR="Black"></TD>
```

Cell color is black

```
<TD BGCOLOR="#FF0000"></TD>
```

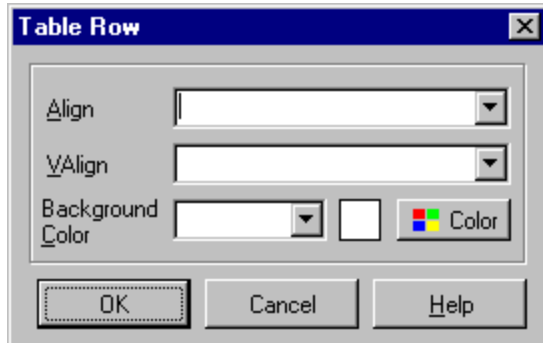
Cell color is red

### **Examples**

```
<TR><TH ALIGN=LEFT>males<TD>1.9<TD>0.003
```

# Table Row

## HTML 0



### Description

The Table Row element is for defining a row of data that will presumably be filled with data points.

### Attributes

**ALIGN** - **HTML 0** The ALIGN attribute can be used to explicitly specify the horizontal alignment of paragraphs within a table row:

**align=left** - Paragraphs are rendered flush left (the default).

**align=center** - Paragraphs are centered.

**align=right** - Paragraphs are rendered flush right.

**align=justify** - Text lines are justified where practical, otherwise this gives the same effect as the default align=left setting.

**VALIGN** - **HTML 0** vertical alignment of material in a cell. Values include "top," "middle," "bottom," "baseline."

**COLOR**=#rrggbb or COLOR=color name - **HTML 0** Sets cell color. rrggbb is a hexadecimal number denoting a red-green-blue color value (the pound sign is optional). Can also be set to a colorname.

Example:

```
<TD BGCOLOR="Black"></TD>
```

Cell color is black

```
<TD BGCOLOR="#FF0000"></TD>
```

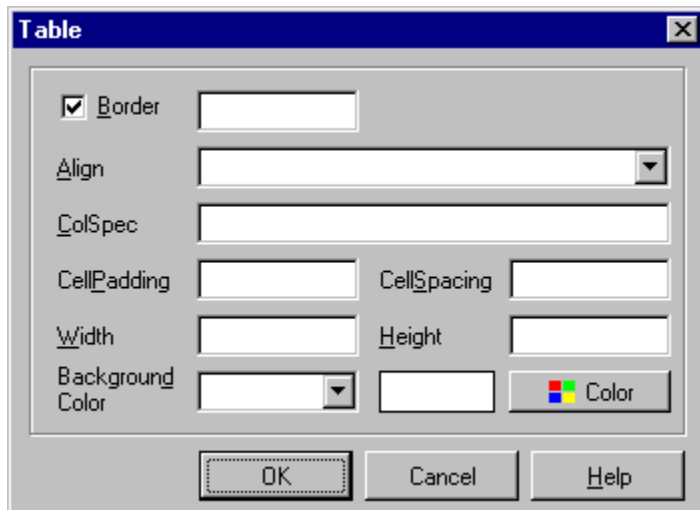
Cell color is red

### Example

```
<TR><TH ALIGN=LEFT>males<TD>1.9<TD>0.003
```

# Table

## HTML 0



### Description

The TABLE element defines a series of rows of table cell elements. The contents of the TABLE element contains a sequence of elements which describe various parts of the table. The order in the sequence is important and consists of: at most one CAPTION element, possibly some COL or possibly some COLGROUP elements, at most one THEAD element, at most one TFOOT element, and finally at least one TBODY element or at least one TR element.

### Attributes

**ALIGN** - **HTML 0** the horizontal alignment of the table on the screen (not the contents of the table). Possible values are:  
bleedleft: aligned at the left window border  
left: at the left text margin  
center: centered between text margins  
right: at the right text margin  
bleedright: aligned at the right window border  
justify: table should fill space between text margins

**BORDER** - **HTML 0** causes browser to render a border around the table; if missing, the table has no grid around it or its data.

**WIDTH** - **HTML 0** specifies how wide the table will be; if given as "NN%", the width is NN% of the width of the display.

**COLSPEC** - **HTML 0** specifies the alignment of items in the columns; for example, Colspec="Lnn Rnn Cnn" specifies that column contents of column 1 are to be aligned left, column 2 right, and column 3 centered. The "nn" specifies the column width in Units.

**CELLSPACING** - **HTML 0** defines spacing between cells.

**CELLPADDING** - **HTML 0** defines spacing within cells.

**COLOR**=#rrggbb or **COLOR**=color name - **HTML 0** Sets cell color. rrggbb is a hexadecimal number denoting a red-green-blue color value (the pound sign is optional). Can also be set to a colorname.

Example:

```
<TD BGCOLOR="Black"></TD>
```

Cell color is black

```
<TD BGCOLOR="#FF0000"></TD>
```

Cell color is red

**Example**

```
<TABLE Border>
  <CAPTION>January Standings</CAPTION>
  <TR><TH Rowspan="5"><TH Colspan="1">Totals</TR>
  <TR><TH>Wins<TH>Losses</TR>
  <TR><TH Align="left">Bears<TD>22<TD>55</TR>
  <TR><TH Align="left">Bengals<TD>84</TD><TD>8</TD></TR>
</TABLE>
```

# Caption

## HTML 0

### Description

The CAPTION element is used for putting a Title above the Table.

### Attributes

**ALIGN** - **HTML 0** Take values TOP or MIDDLE or BOTTOM, defining whether the top or middle or bottom of the graphic should be aligned with the baseline for the text line in which the Table element appears. With ALIGN=LEFT, the caption will float down and over to the current left margin, and subsequent text will wrap around the right hand side of the caption. Likewise for ALIGN=RIGHT, the caption aligns with the current right margin and, and text wraps around the left.

### Example

```
<CAPTION>A test table with merged cells</CAPTION>
```

# Anchors and Links

## HTML 0

The screenshot shows a dialog box titled "Anchor / Link" with a close button in the top right corner. The dialog contains the following fields and controls:

- HRef**: A text input field with a search icon and the label "URL" to its right.
- Target**: A dropdown menu.
- Name**: A text input field.
- Title**: A text input field.
- Message Digest**: A text input field.
- Rel**: A dropdown menu.
- Rev**: A dropdown menu.
- URN**: A dropdown menu.
- Shape**: A dropdown menu.

At the bottom of the dialog, there are four buttons: "Anchor" (with a downward arrow icon), "Link" (with a chain link icon), "Cancel", and "Help".

### Description

The A element brackets (or anchors) a piece of text (and/or image) which is identified as a hypertext link. The A element must have either an HREF attribute or a NAME attribute. The HREF attribute identifies a destination URL, and the bracketed text is rendered as a hypertext link to the URL. Browsers will display the contents of an A element with an HREF attribute in a special manner to indicate that if the contents are selected, the browser will execute that hypertext link. The NAME attribute identifies a destination tag, and the bracketed text is thereby identified as an available hypertext target within this document. Browsers do not display the contents of an A element with a NAME attribute in any special way. However, an A element with an HREF attribute can now be constructed by using the document URL suffixed with #name.

This will load the document, but will position the display starting at the location of this NAME tag. An A element with an HREF attribute can also be constructed to jump directly to this destination tag within the same document by a URL consisting solely of #name. The presence of REL=relation in document A with HREF to document/object B identifies a relationship that B has to A that A recognizes/authorizes/verifies. The presence of REV=relation of the identical relation in document B with HREF to document/object A identifies a desired/expected/claimed relationship that B has to A, but must be verified by checking with A.

The LINK element indicates a relationship between the document and some other object. A document may have any number of LINK elements. The LINK element is empty (does not have a closing tag), but takes the same attributes as the anchor element.

### Attributes

**ID** - **HTML 0** An SGML identifier used as the target for hypertext links or

for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document. This attribute supersedes the "NAME" attribute, see below. For example, the following paragraph is defined as an anchor named "charles":

```
<P ID="charles">The Charles river flows into Boston harbor,
and played an important role in opening up the hinterland
to early settlers...
```

Elsewhere, you can define a link to this paragraph, as follows:

```
<A HREF="#charles">Boston</A> is a historic city and
a thriving center of commerce and higher education.
```

The reader can select the link labeled "Boston" to see further information on the Boston area.

**LANG** - **HTML 0** This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - **HTML 0** This is a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

**HREF** - **HTML 0** The HREF attribute implies that the anchor acts as the start of a hypertext link. The destination is designated by the value of the HREF attribute, which is expressed in the Universal Resource Identifier (URI) notation.

**MD** - **HTML 0** Specifies a message digest or cryptographic checksum for the linked document designated by the HREF attribute. It is used when you want to be sure that a linked object is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQITDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

**NAME** - **HTML 0** This attribute is used to define a named anchor for use as the destination of hypertext links. For example, the following defines an anchor that can be used as the destination of a jump into a description of the Boston area.

```
The <A NAME="potomac">Potomac river</A> flows into Boston
harbor.
```

**Note:** the NAME attribute has been superseded by the ID attribute.

**SHAPE** - **HTML 0** This attribute is used within figures to define shaped hotzones for graphical hypertext links. Full details of how to use this feature will be given with the description of the figure element. The attribute value is a string taking one of the following forms:

**"default"** - Used to define a default link for the figure background.

**"circle x, y, r"** - Where x and y define the center and r specifies the radius.

**"rect x, y, w, h"** - Where x, y define the upper left corner and w, h define the width and height respectively

**"polygon x1, y1, x2, y2, ..."** - Given n pairs of x, y coordinates, the polygon is closed by a line linking the n'th point to the first. Intersecting polygons use the non-zero winding number rule to determine if a point lies inside the polygon.

If a pointer event occurs in a region where two or more shapes overlap, the distance from the point to the center of gravity of each of the overlapping shapes is computed and the closest one chosen. This feature is useful when you want lots of closely spaced hotzones, for example over points on a map, as it allows you to use simple shapes without worrying about overlaps.

**Note:** The x coordinate increases to the right, and the y coordinate increases downwards in the same way as IMG and image maps. If both numbers are integers, the coordinates are interpreted as pixel offsets from the upper left corner of the figure. Otherwise, the coordinates are interpreted as scaled values in the range 0.0 to 1.0 across the figure. Note the syntax is tolerant of repeated white space characters between tokens.

**TARGET** - **HTML 0** Links in any window can refer to another window by name using the TARGET attribute. When you click the link, the document you requested will appear in that named window. If the window is not already open, the browser will open and name a new window for you.

**TITLE** - **HTML 0** This is informational only and describes the object specified with the HREF attribute. It can be used for object types that don't possess titles, such as graphics, plain text and Gopher menus.

**REL** - **HTML 0** Used to describe the relationship of the linked object specified with the HREF attribute. The set of relationship names is not part of this specification, although "Path" and "Node" are reserved for future use with hypertext paths or guided tours. The REL attribute can be used to support search for links serving particular relationships.

**HTML 0** Extensions to the Rel attribute; Rel can be used to define a series of values for browser toolbar or other buttons:

- Rel = Home; defines the home page link relative to this document
- Rel = ToC; table of contents link
- Rel = Index; an index
- Rel = Glossary; the glossary of terms
- Rel = Copyright; the copyright statement
- Rel = Up; the parent document
- Rel = Next; the next document to visit in a "tour"



Rel = Previous; the previous document in a "tour"

Rel = Help; a link to a help document or service

Rel = Bookmark. Bookmarks are used to provide direct links to key entry points into an extended document. The TITLE attribute may be used to label the bookmark. Several bookmarks may be defined in each document, and provide a means for orienting users in extended documents.

Rel = StyleSheet; a stylesheet to control the rendering of the current document

## HTML 0

**REV** - This defines a reverse relationship. A link from document A to document B with `REV=relation` expresses the same relationship as a link from B to A with `REL=relation`. `REV=made` is sometimes used to identify the document author, either the author's email address with a *mailto* URI, or a link to the author's home page. Tables of contents can use anchors with `REV="ToC"` to allow software to insert page numbers when printing hypertext documents. The plain text version of this specification was generated in this way!

## HTML 0

**URN** - This indicates the Uniform Resource Name of the document.

### Examples

#### Anchors in other documents

```
<A HREF="WebEdit.zip">Download WebEdit Now</A>
```

the text **Download WebEdit Now** would appear in color and/or underlined indicating it is a link.

#### Anchors in parts of other documents

If you wanted to create a link from one document to another, specific section of a second document you need to create a named anchor in the second document. For example, to set up an anchor named "Support" in the second document you would type, Welcome `<A NAME = "Support">`to Tech Support. Then just create a link to it from document one, including the filename and the named anchor, separated by a hash mark (#).

For example,

```
This is your <A HREF ="doc2.htm#Support">link</A> to the second document.
```

#### Anchors in the same document

This is handled in the same manner as linking to part of another document except that there is no need to include the filename.

For example, to link to the Support anchor from the same file (second document), you could use:

```
This is your <A HREF="#Support">link</A> within the second document.
```

# URL's

URL's are used by the World Wide Web to specify the location of files on other servers. For example to create a link to a file on another server you could use:

```
<A HREF = "http://www.nesbitt.com/download.html">Download WebEdit  
Now</A>
```

A URL consists of several parts:

- a resource type,
- a server name,
- a port number,
- a file name, and
- an anchor

The resource type specifies whether the link points to a Web page, a gopher server, a file on an ftp server, an email address, etc. For example, in the above tag, "http://" is the resource type. This indicates that the file is a Web page. If the file were on an ftp server, the resource type would be "ftp://". Gopher resources are specified with "gopher://". Some browsers such as NetScape even support extended resource types like "mailto:" for sending email, etc.

A server name is the name of a host on the Internet. As an example, Microsoft has a web server at [www.microsoft.com](http://www.microsoft.com). You can usually connect to a remote host with nothing more than a resource type and a server name. Specifying <http://www.microsoft.com/> will provide a hotlink to Microsoft's Web home page.

Ordinarily, resources are on "well-known" ports. Web resources are on port 80, ftp files on port 79 and so on. Sometimes though, a resource will be on a non-standard port. In this case, the hotlink needs to include the port number as part of the server name, separated by a colon, as in this example:

```
http://www.lysator.liu.se:7500/
```

If you want your link to point at a specific resource on the server, you will need to include a file name. For example, WebEdit is located at <http://www.nesbitt.com/webedit.zip>. The [webedit.zip](http://www.nesbitt.com/webedit.zip) is the file name,

Sometimes, you will also need to specify a directory path as well, as in <http://www.geom.umn.edu/apps/gallery.html>. This indicates that the file [gallery.html](http://www.geom.umn.edu/apps/gallery.html) is in the subdirectory [/apps](http://www.geom.umn.edu/apps/).

Lastly, if the file is an HTML file, it can have bookmarks within the file, called anchors, allowing you to point not just to a file, but to a specific point within the file. Anchors are tagged onto the end of the filename, separated by a pound sign, as in [myfile.html#myanchor](#). In most cases, URLs consist of just three parts: a resource type, a server name and a file name.

# Java Tags

## **HTML Java Tags**

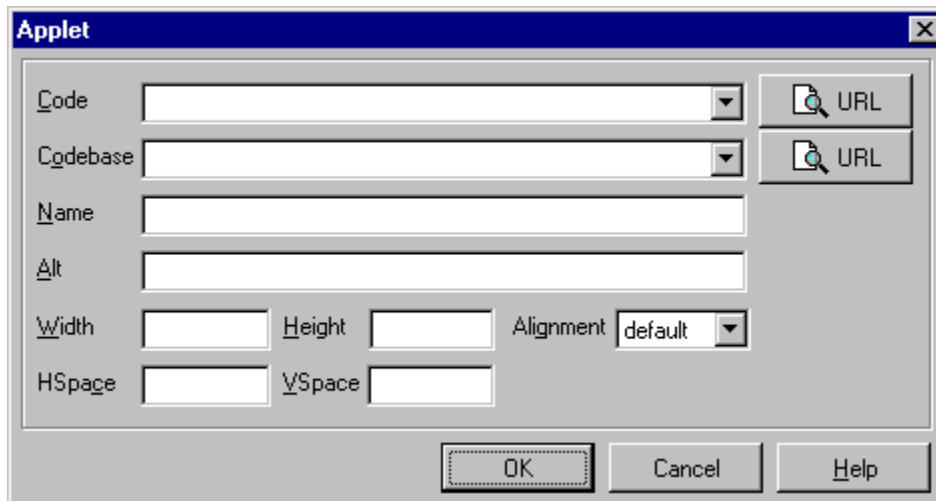
HTML provides these means for defining ways of interacting with Java.

The Applet element identifies and invokes a JAVA application.

The Param (Java) element is a mechanism to define general purpose parameters to be passed to APPLET applications.

# Applet

## HTML 0



### Description

The APPLET element replaced the APP element as the mechanism to identify and invoke a JAVA(tm) application. A browser that understands this element will ignore everything in the content of the APPLET element except the PARAM elements. Browsers that do not understand this element should ignore it and the PARAM elements and instead process the content of the element. Thus the content is the alternate HTML if the application is not invoked.

### Attributes

**CODE** - **HTML 0** the name of the file that contains the compiled Applet subclass. This name is relative to the base URL of the applet and cannot be an absolute URL.

**CODEBASE** - **HTML 0** specifies the base URL of the applet.

**ALT** - **HTML 0** specifies parsed character data to be displayed if the browser understands the APPLET tag but can't/won't run them. NAME specifies a name for the applet instance, which allows applets on the same page to communicate with each other.

**WIDTH** and **HEIGHT** - **HTML 0** give the initial width and height (in pixels) of the applet display area.

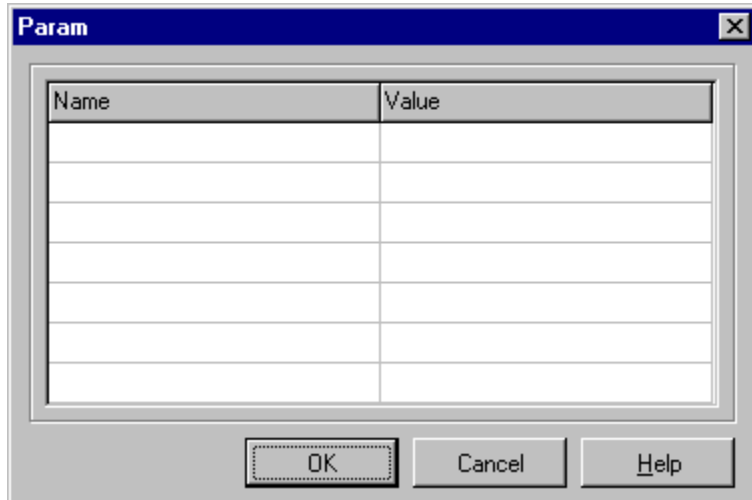
**ALIGN** - **HTML 0** specifies the display alignment.

**VSPACE** and **HSPACE** - **HTML 0** specify the reserved space around the applet (in pixels).



## Param (Java)

HTML 0



Name	Value

OK Cancel Help

### Description

The PARAM element is a mechanism to define general purpose parameters to be passed to APPLET applications.

### Attributes

**NAME** - HTML 0 the name of the parameter.

**VALUE** - HTML 0 value obtained by the applet with the getParameter() method.

# Object Tags

## HTML Object Tag Formats

This specification extends HTML to support the insertion of multimedia objects including Java applets, Microsoft Component Object Model (COM) objects (e.g. OLE Controls and OLE Document embeddings), and a wide range of other media plug-ins. The approach allows objects to be specified in a general manner and provides the ability to override the default implementation of objects.

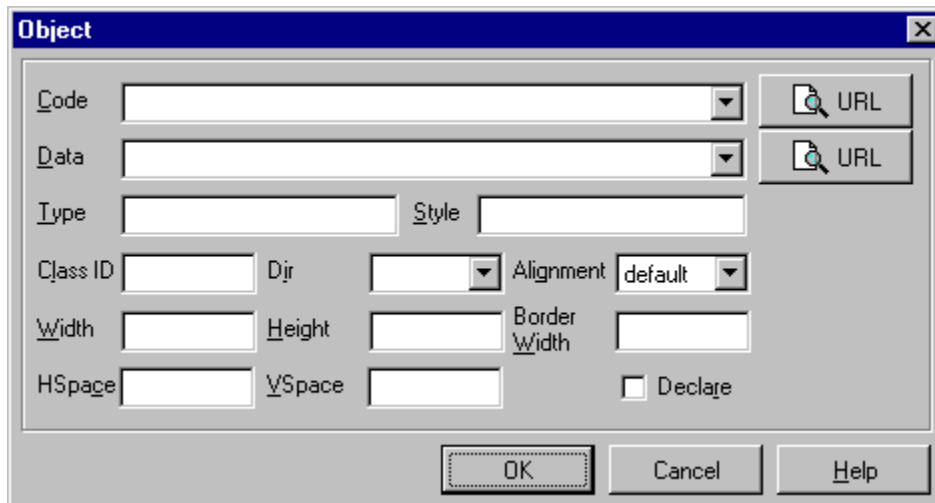
The Object element is used to insert an object into an HTML document.

The Alias element is used to define an object without inserting it into the document.

The Param element allows a list of named property values (used to initialize a OLE control, plug-in module or Java applet) to be represented as a sequence of PARAM elements.

# Object

## HTML 0





### Description

The OBJECT element is used to insert an object into an HTML document. It requires both start and end tags. The OBJECT element has the same content model as the HTML BODY element, except that one or more optional PARAM, or EVENT elements can be placed immediately after the OBJECT start tag and used to initialize the inserted object. The content of the OBJECT element is rendered if the object specified by the data, code or classid attributes can't be rendered (user agents may choose to display the content of the OBJECT element if displaying the actual element will take a long time to render). This provides for backwards compatibility with existing browsers, and allows authors to specify alternative media via nested OBJECT elements.

Note that this doesn't provide the same level of flexibility as would be provided by a richer description of resource variants. For instance when a resource is available in several media types and for each such type in English, Spanish, French and German.


### Attributes

**CODE** -  This specifies a URL referencing where to find the code which implements the object's behaviour. If this URL is insufficient to locate the intended object, when for instance, a file contains the implementations for several classes, the CLASSID may be used to supply a disambiguating class identifier.

**DATA** -  Specifies a URL referencing the object's data. This could be a GIF file or the pickled data representing an object's state. In many cases the media type or the data itself contains sufficient information to identify what code is needed to initialize the object. Note that an object's data can even be included inline for super efficient loading. This specification proposes a new URL scheme "data:". The rest of the URL is a base64 encoded character string that specifies the object's data as an opaque byte stream. On its own, this would be meaningless. If the DATA attribute appears without a CODE or CLASSID attribute, then a TYPE attribute may be sufficient to interpret the data. For instance a Microsoft COM object can be asked to write its state using the WriteClassStream procedure. This inserts the object's class id as the first 16 bytes of the stream. If the TYPE attribute indicates that the



data is in the COM persistent stream format, then the class id can be retrieved from the DATA attribute and used to find the code implementing the object's behaviour. The CLASSID attribute can be used to override the default implementation as implied by the DATA attribute. For example, you may have the pickled data for an Excel spread sheet but want to view it with the "SuperGraph" package. You would then use the DATA attribute to point to the Excel spreadsheet data, and the CLASSID or CODE attribute to point to the SuperGraph plugin. The CLASSID, CODE and DATA attributes specify URLs. Any fragment identifier included as part of these URLs should be passed to the object, either directly, or by callback.

**TYPE** -  This specifies an Internet Media Type (see RFC 1590) for the object's data. The attribute can be used to allow user agents to quickly skip media they don't support, and instead to render the contents of the OBJECT element. It is also useful when loading objects off local drives as it allows the media type to be specified explicitly rather than being derived from the file extension. The following grammar for media types is a superset of that for MIME because it does not restrict itself to the official IANA and x-token types.

```
media-type    = type "/" subtype *( ";" parameter )
type         = token
subtype      = token
```


where token is defined by:


```
token        = 1*<any (ASCII) CHAR except SPACE, CTLs, or tspecials>
tspecials   = <one of the set>  ( ) < > @ , ; \ " / [ ] ? =
```

Parameters may follow the type/subtype in the form of attribute/value pairs.

```
parameter   = attribute "=" value
attribute   = token
value       = token | quoted-string
```

The type, subtype, and parameter attribute names are case-insensitive. Parameter values may or may not be case-sensitive, depending on the semantics of the parameter name. White space characters must not be included between the type and subtype, nor between an attribute and its value. If a given media-type value has been registered by the IANA, any use of that value must be indicative of the registered data format. Although HTML allows the use of non-registered media types, such usage must not conflict with the IANA registry. Data providers are strongly encouraged to register their media types with IANA via the procedures outlined in RFC 1590. All media-type's registered by IANA must be preferred over extension tokens. However, HTML does not limit applications to the use of officially registered media types, nor does it encourage the use of an "x-" prefix for unofficial types outside of explicitly short experimental use between consenting applications.

**STYLE** -  The STYLE attribute allows you to include rendering information.

**CLASSID** -  This can be used to specify a class identifier for an object. This could be a DCE universally unique object identifier (uuid), Java class name, or another type of class name as appropriate to the object system, e.g. Corba. This allows effective use of caching, as the user agent can use simple string comparison to check whether two objects are the same independent of their location. The CLASSID attribute value takes the form of a URL scheme prefix separated by a colon from the character string defining the class identifier. The prefix is used to identify the object system for the class identifier, for example

classid="clsid:{663C8FEF-1EF9-11CF-A3DB-080036F12502}" gives the (clsid) uuid for a Microsoft COM object, using the CLSID name space, while classid="java:Animator.class" gives the class name for Java applet. CLASSID may be sufficient for the user agent to locate the code implementing the object. However, the CODE attribute can be used with CLASSID to provide a hint as to where to look for this code. The search mechanism will in general depend on the object system the identifier belongs to. Note that the value specified with CLASSID takes precedence over a class identifier derived from the object's data stream. When searching for the implementation of an object, the CLASSID attribute takes precedence over the DATA/TYPE attributes. A decision tree giving further details on this resolution procedure appears later on in this specification. In the absence of CLASSID a value for the class identifier may be derivable from the DATA attributes, for instance the Internet media type for the DATA may be sufficient, e.g. when the data is for a GIF encoded image.

**DIR** - **HTML 0** Human writing systems are grouped into scripts, which determine amongst other things, the direction the characters are written. Elements of the Latin script are nominally left to right, while those of the Arabic script are nominally right to left. These characters have what is called strong directionality. Other characters can be directionally neutral (spaces) or weak (punctuation). The DIR attribute specifies an encapsulation boundary which governs the interpretation of neutral and weakly directional characters. It does not override the directionality of strongly directional characters. The DIR attribute value is one of LTR for left to right, or RTL for right to left, e.g. DIR=RTL.

**ALIGN** - **HTML 0** This determines where to place the object. The ALIGN attribute allows objects to be placed as part of the current text line, or as a distinct unit, aligned to the left, center or right. The following values are chosen for their ease of implementation, and their independence of other graphics occurring earlier on the same line:

For ALIGN=TEXTTOP, the top of the object is vertically aligned with the top of the current font.

For ALIGN=MIDDLE, the middle of the object is vertically aligned with the baseline.

For ALIGN=TEXTMIDDLE, the middle of the object is vertically aligned with the position midway between the baseline and the x-height for the current font. The x-height is defined as the top of a lower case x in western writing systems. If the text font is an all-caps style then use the height of a capital X. For other writing systems, align the middle of the object with the middle of the text.

For ALIGN=BASELINE, the bottom of the object is vertically aligned with the baseline of the text line in which the object appears.

For ALIGN=TEXTBOTTOM, the bottom of the object is vertically aligned with the bottom of the current font.

The following alignment values allow the object to float rather than being treated as part of the current line:

For ALIGN=LEFT, the object is floated down and over to the current left margin.

Subsequent text is flowed past the right hand side of the visible area of the object.

For ALIGN=CENTER, the object is floated to after the end of the current line and centered between the left and right margins. Subsequent text starts at the beginning of the next line.

For ALIGN=RIGHT, the object is floated down and over to the current right margin.

Subsequent text is flowed past the left hand side of the visible area of the object.

**WIDTH** - **HTML 0** This gives the suggested width of a box enclosing the visible area of the object. The width is specified in standard units. User agents may use this value to scale an object to match the requested width if appropriate. Smooth scaling a small image to a larger size provides an effective solution to reducing the time needed to download an image, offering better subjective results when compared to color reduction.

**HEIGHT** - **HTML 0** This gives the suggested height of a box enclosing the visible area of the object. The height is specified in standard units. User agents may use this value to scale an object to match the requested height if appropriate.

**BORDER** - **HTML 0** This attribute applies to the border shown when the object forms part of a hypertext link, as specified by an enclosing anchor element. The attribute specifies the suggested width of this border around the visible area of the object. The width is specified in standard units. For **BORDER=0** no border should be shown. This is normally used when such a border would interfere with the visual affordances presented by the object itself. For instance, the object could render itself as a number of beveled buttons.

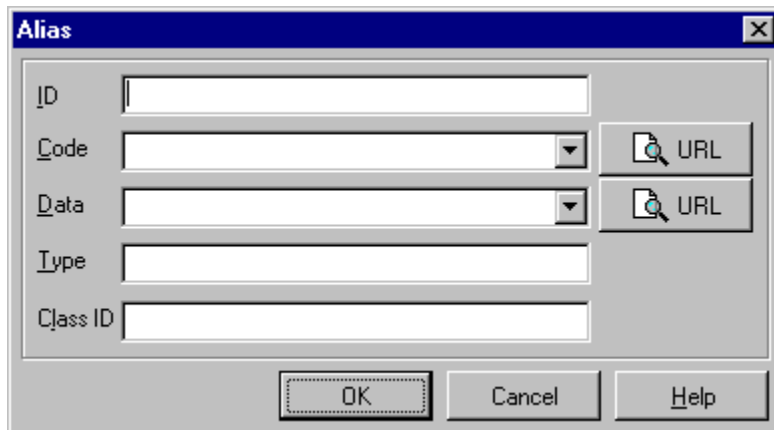
**HSPACE** - **HTML 0** The suggested width of the space to the left and right of the box enclosing the visible area of the object. The width is specified in standard units. This attribute is used to alter the separation of preceding and following text from the object.

**VSPACE** - **HTML 0** The suggested height of the space to the top and bottom of the box enclosing the visible area of the object. The height is specified in standard units.

**DECLARE** - **HTML 0** Used to indicate that the object is not to be instantiated, only declared.

# Alias

## HTML 0



The image shows a dialog box titled "Alias" with a close button (X) in the top right corner. The dialog contains several input fields and buttons:

- ID**: A text input field.
- Code**: A dropdown menu with a search icon and the text "URL" to its right.
- Data**: A dropdown menu with a search icon and the text "URL" to its right.
- Type**: A text input field.
- Class ID**: A text input field.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

### Description

The ALIAS element is used to define an object without inserting it into the document. It is used with the valueref attribute of the PARAM element to allow an object to be passed as parameter, when initializing an object associated with another OBJECT or ALIAS element. The attributes take exactly the same meaning as for the OBJECT element. The ALIAS element is a container and requires both start and end tags. It can be placed anywhere in the document HEAD or BODY. The contents are limited to PARAM and ALIAS elements, although it is anticipated that this may be extended to cover the same content model as OBJECT at some point in the future.

Note that the object isn't created until its needed by something that references it (i.e. late binding). Each such reference when bound creates a separate copy of the object. In other words, the alias element is treated as a declaration for making an object. When another object is being created and has an alias as a parameter, the declaration defined by the alias is then used to create a new object. Notice that a given object class may be used by different aliases to create multiple instances of that class with different data.

If the aliased object isn't supported, or fails to load, the user agent should try the contents of the ALIAS element, which is currently restricted to another ALIAS element. The TYPE attribute can be used to specify the Internet Media Type for the object as a hint for this situation.

Like the OBJECT, ALIAS elements are limited in scope to the document in which they appear. The objects created for these elements have a life time at least equal to that of the document. So if the user clicks on a link to move to another document, the objects are not destroyed and can be seen again when the user backtracks to the original document.

### Attributes

**ID** - **HTML 0** Used to define a document-wide identifier. This can be used for naming positions within documents as the destination of a hypertext link. It may also be used by style sheets for rendering an element in a unique style. An ID attribute value is an SGML NAME token. NAME tokens are formed by an initial letter followed by letters, digits, "-" and "." characters. The letters are restricted to A-Z and a-z. It may also be used by the

user agent or other objects in the document to find and communicate with objects on the document.

**CODE** - **HTML 0** This specifies a URL referencing where to find the code which implements the object's behaviour. If this URL is insufficient to locate the intended object, when for instance, a file contains the implementations for several classes, the CLASSID may be used to supply a disambiguating class identifier.

**DATA** - **HTML 0** Specifies a URL referencing the object's data. This could be a GIF file or the pickled data representing an object's state. In many cases the media type or the data itself contains sufficient information to identify what code is needed to initialize the object. Note that an object's data can even be included inline for super efficient loading. This specification proposes a new URL scheme "data:". The rest of the URL is a base64 encoded character string that specifies the object's data as an opaque byte stream. On its own, this would be meaningless. If the DATA attribute appears without a CODE or CLASSID attribute, then a TYPE attribute may be sufficient to interpret the data. For instance a Microsoft COM object can be asked to write its state using the WriteClassStream procedure. This inserts the object's class id as the first 16 bytes of the stream. If the TYPE attribute indicates that the data is in the COM persistent stream format, then the class id can be retrieved from the DATA attribute and used to find the code implementing the object's behaviour. The CLASSID attribute can be used to override the default implementation as implied by the DATA attribute. For example, you may have the pickled data for an Excel spread sheet but want to view it with the "SuperGraph" package. You would then use the DATA attribute to point to the Excel spreadsheet data, and the CLASSID or CODE attribute to point to the SuperGraph plugin. The CLASSID, CODE and DATA attributes specify URLs. Any fragment identifier included as part of these URLs should be passed to the object, either directly, or by callback.

**TYPE** - **HTML 0** This specifies an Internet Media Type (see RFC 1590) for the object's data. The attribute can be used to allow user agents to quickly skip media they don't support, and instead to render the contents of the OBJECT element. It is also useful when loading objects off local drives as it allows the media type to be specified explicitly rather than being derived from the file extension. The following grammar for media types is a superset of that for MIME because it does not restrict itself to the official IANA and x-token types.

media-type = type "/" subtype \*( ";" parameter )  
type = token  
subtype = token

where token is defined by:

token = 1\*<any (ASCII) CHAR except SPACE, CTLs, or tspecials>  
tspecials = <one of the set> ( ) < > @ , ; : \ " / [ ] ? =

Parameters may follow the type/subtype in the form of attribute/value pairs.

parameter = attribute "=" value  
attribute = token  
value = token | quoted-string

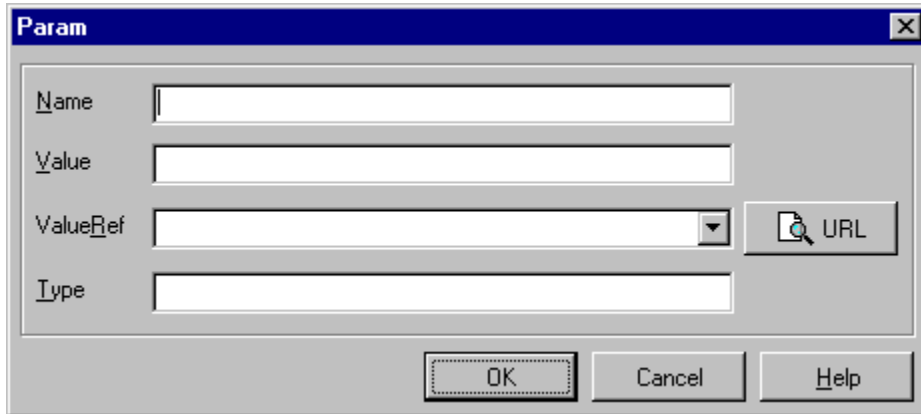
The type, subtype, and parameter attribute names are case-insensitive. Parameter values may or may not be case-sensitive, depending on the semantics of the parameter name.

White space characters must not be included between the type and subtype, nor between an attribute and its value. If a given media-type value has been registered by the IANA, any use of that value must be indicative of the registered data format. Although HTML allows the use of non-registered media types, such usage must not conflict with the IANA registry. Data providers are strongly encouraged to register their media types with IANA via the procedures outlined in RFC 1590. All media-type's registered by IANA must be preferred over extension tokens. However, HTML does not limit applications to the use of officially registered media types, nor does it encourage the use of an "x-" prefix for unofficial types outside of explicitly short experimental use between consenting applications.

**CLASSID** - **HTML 0** This can be used to specify a class identifier for an object. This could be a DCE universally unique object identifier (uuid), Java class name, or another type of class name as appropriate to the object system, e.g. Corba. This allows effective use of caching, as the user agent can use simple string comparison to check whether two objects are the same independent of their location. The CLASSID attribute value takes the form of a URL scheme prefix separated by a colon from the character string defining the class identifier. The prefix is used to identify the object system for the class identifier, for example classid="clsid:{663C8FEF-1EF9-11CF-A3DB-080036F12502}" gives the (clsid) uuid for a Microsoft COM object, using the CLSID name space, while classid="java:Animator.class" gives the class name for Java applet. CLASSID may be sufficient for the user agent to locate the code implementing the object. However, the CODE attribute can be used with CLASSID to provide a hint as to where to look for this code. The search mechanism will in general depend on the object system the identifier belongs to. Note that the value specified with CLASSID takes precedence over a class identifier derived from the object's data stream. When searching for the implementation of an object, the CLASSID attribute takes precedence over the DATA/TYPE attributes. A decision tree giving further details on this resolution procedure appears later on in this specification. In the absence of CLASSID a value for the class identifier may be derivable from the DATA attributes, for instance the Internet media type for the DATA may be sufficient, e.g. when the data is for a GIF encoded image.

# Param

## HTML 0



### Description

The PARAM element allows a list of named property values (used to initialize a OLE control, plug-in module or Java applet) to be represented as a sequence of PARAM elements. Note that PARAM is an empty element and should appear without an endtag.

### Attributes

**NAME** - **HTML 0** defines the property name. The case sensitivity of the name is dependent on the code implementing the object.

**VALUE** - **HTML 0** used to specify the property value. It is an opaque character string whose meaning is determined by the object based on the property name. Note that CDATA attribute values need characters such as & to be escaped using the standard SGML character entities, e.g. &amp; for "&". It is also essential to escape the > character to defend against incorrect handling by many existing browsers (use &gt;).

**VALUEREF** - **HTML 0** used when the property is itself an object. A distinct attribute is needed as in some cases the property type cannot be deduced from the property name. VALUEREF typically provides a URL based reference to an OBJECT DECLARE element that defines the object itself. VALUEREF can also be used to specify an object directly. For example valueref=foo.gif. Another possibility is to use inline data with the "data:" URL scheme. Both of these options save having to include an associated ALIAS element.

The VALUEREF attribute of PARAM allows properties to be "object valued"; the property is an object itself. For example, Windows provides a standard Font object that is exposed as a COM object. Any OLE Control can use the "font object type" as the type of the "Font" property. The following Visual Basic example illustrates how a programmer would manipulate such a property:

```
Set x = CreateObject("Button")
x.Text = "Press me"
x.BackColor = vbWhite
x.Font.Face = "Courier New"
```

```
x.Font.Bold = True
x.Font.Size = 12
```

The equivalent OBJECT tag would look like this:

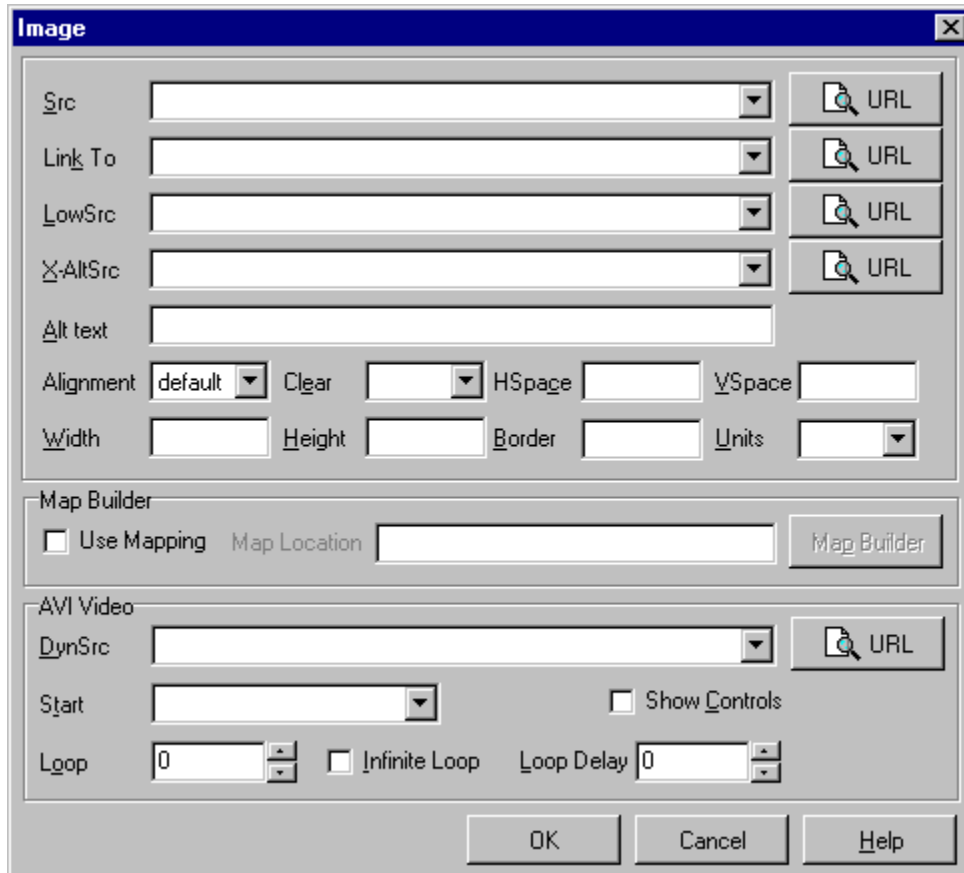
```
<object id="x" classid=#CLSID-CoolButton>
  <object declare id=Btn1Font classid=#CLSID-StdFont>
    <param name="Face" value="Courier New">
    <param name="Bold" value="True">
    <param name="Size" value=12>
  </object>
  <param name="Text" value="Press me">
  <param name="BackColor" value=0xFFFFFFFF >
  <param name="Font" valueref=Btn1Font>
</object>
```

**TYPE** - **HTML 0** required when the "data:" URL scheme is used. It specifies the media type for the data stream and allows the user agent to decode the stream, e.g. to pick out an embedded class identifier.



# Image

## HTML 0



### Description

The <IMG> tag is used to incorporate in-line graphics (typically icons or small graphics) into an HTML document. This element is NOT intended for embedding other HTML text.

### Additions

There have been two major additions to the functionality of the IMG tag. [Client-Side Image Mapping](#) and [AVI Video](#).

### Attributes

**ALIGN** - **HTML 0** Takes values TOP or MIDDLE or BOTTOM, defining whether the top or middle or bottom of the graphic should be aligned with the baseline for the text line in which the IMG element appears. With ALIGN=LEFT, the graphic will float down and over to the current left margin, and subsequent text will wrap around the right hand side of the graphic. Likewise for ALIGN=RIGHT, the graphic aligns with the current right margin and, and text wraps around the left. It is inappropriate to use this feature for larger graphics as these are best represented with the FIG element.

### Additions

**ALIGN=texttop** **HTML 0** does what many people thought top should do which is align itself with the top of the tallest text in the line (this is usually but not always the same as ALIGN=top).

**ALIGN=absmiddle** **HTML 0** does what middle should have done which is align the middle of the current line with the middle of the image.

**ALIGN=baseline** **HTML 0** aligns the bottom of the image with the baseline of the current line.

**ALIGN=bottom** **HTML 0** does just what it always did (which is identical to ALIGN=baseline but baseline is a better name).

**ALIGN=absbottom** **HTML 0** does what bottom should have done which is align the bottom of the image with the bottom of the current line.

**ALT (Alternate text)** - **HTML 0** Optional alternative text as an alternative to the graphics for display in text-only environments. The alt text can contain entities e.g. for accented characters or special symbols, but it can't contain markup. The latter is possible, however, with the FIG element. **ID** - An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**BORDER** - **HTML 0** This lets the document author control the thickness of the border around an image displayed. Warning: setting BORDER=0 on images that are also part of anchors may confuse your users as they are used to a colored border indicating an image is an anchor.

**CLASS** - **HTML 0** This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

**HEIGHT** - **HTML 0** Optional suggested height for the image. By default, this is given in pixels. If specified, the viewer of the document will not have to wait for the image to be loaded over the network and its size calculated.

**HSPACE** - **HTML 0** specifies horizontal margins for the image. Similar to BORDER, except the margins are not painted with color when the image is a hyperlink.

```
<IMG SRC="sample.gif" HSPACE=5 VSPACE=10>
```

This image has five pixels of space on its left and right, and 10 pixels of space above and below.

**LANG** - **HTML 0** This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**MD** - **HTML 0** Specifies a message digest or cryptographic checksum for the associated graphic specified by the SRC attribute. It is used when you want to be sure that the image is indeed the same one that the author intended, and hasn't been modified in any way. For instance, MD="md5:jV2OfH+nnXHU8bnkPAad/mSQITDZ", which specifies an MD5 checksum encoded as a base64 character string. The MD attribute is generally allowed for all elements which support URI based links.

**SRC** - **HTML 0** The SRC attribute specifies the URI for the image to be embedded. Its syntax is the same as that of the HREF attribute of the <A> tag. SRC is mandatory.

**TARGET** - **HTML 0** Links in any window can refer to another window by name using the TARGET attribute. When you click the link, the document you requested will appear in that named window. If the window is not already open, the browser will open and name a new window for you.

**UNITS** - **HTML 0** This optional attribute specifies the units for the width and height attributes. It is one of: units=pixels (the default) or units=en (half the point size).

**USEMAP** - See Client Side Image Mapping.

**VSPACE** - **HTML 0** specifies vertical margins for the image. Similar to BORDER, except the margins are not painted with color when the image is a hyperlink.

```
<IMG SRC="sample.gif" HSPACE=5 VSPACE=10>
```

This image has five pixels of space on its left and right, and 10 pixels of space above and below.

**WIDTH** - **HTML 0** Optional suggested width for the image. By default, this is given in pixels. If specified, the viewer of the document will not have to wait for the image to be loaded over the network and its size calculated.

**XALTSRC** - If the browser supports it, this will display an image that uses the Johnson-Grace ART image compression standard. For more information than what follows see <http://www.jgc.com>.

To add a graphic image to your Web pages, you use the IMG tag. Its basic format is

```
<IMG SRC="URL">
```

And here's an example that references a GIF file:

```
<IMG SRC="/images/daytona.gif">
```

In the relatively simple case where your entire audience is using ART-enabled Web browsers and you don't care about supporting "old-style" (non-ART-enabled) browsers, you can use a URL that directly references an ART image. Here's an example:

```
<IMG SRC="/images/daytona.art">
```

In this case, an ART-enabled browser would easily recognize the ART file format and do the right thing. But an old-style browser would not recognize the format and therefore would display no image at all, clearly a problem if you have a diverse audience and are unable to predict what vintage browsers they might be using. In fact, what you'd probably like to see happen for users of old-style browsers is that they would see a GIF or JPEG image rather than no image at all. To handle this situation, Johnson-Grace and the major browser companies have come up with alternative, actually a companion, to the SRC attribute.

### **The X-ALTSRC Attribute**

To allow newer browsers to access ART images while still allowing old-style browsers to access non-ART images, Johnson-Grace and many of the major browser companies have settled on a new attribute for the IMG tag. Here's its general form:

```
<IMG X-ALTSRC=" [mime/type;]URL">
```

When used in conjunction with the SRC attribute, this new attribute allows you to specify an alternate source file that can be used to display a graphic image. For ART-enabled browsers, this means that the browser would automatically select an ART image. Old-style browsers continue to work as before. Here's an example:

```
<IMG X-ALTSRC="/images/daytona.art" SRC="/images/daytona.gif">
```

In this case, ART-enabled browsers would read the entire IMG tag and use /images/daytona.art instead of /images/daytona.gif. Old-style browsers, who know nothing of X-ALTSRC, would use /images/daytona.gif. We call this "dual encoding", and it is the easiest and safest way to incorporate support for ART images into your HTML scripts.

The optional [mime/type;] portion of the attribute bears a little explaining. It lets you explicitly tell the browser the format of the file that follows, rather than having the browser determine the format based on the file extension, its default behavior. Here's a variation of the previous example that illustrates how this works:

```
<IMG X-ALTSRC="image/x-art;/images/daytona.v1"  
SRC="/images/daytona.gif">
```

In this case, the browser knows that the file format is ART based on the mime/type, and the file extension is ignored.

### **Examples**

Here are some more examples of how to use the new X-ALTSRC attribute.

Here's another example that illustrates how different browsers might behave:

```
<IMG X-ALTSRC="image/x-art;/images/picture.art"
```

```
SRC="/images/picture.xbm">
```

In this example, old-style browsers that know nothing of X-ALTSRC would use /images/picture.xbm, provided they recognize the XBM format. (In case you're wondering, XBM is an X-Windows graphics format.) ART-enabled browsers would use /images/picture.art, unless the XBM format is known to the browser and the user had somehow set the browser's preferences to indicate that XBM files were more desirable than ART files.

This next example illustrates how your knowledge of your target audience might affect your use of the SRC and X-ALTSRC attributes.

```
<IMG X-ALTSRC="image/x-art;triangle.art" X-ALTSRC="triangle.tif"  
SRC="triangle.gif" ALT="Warning:"> Be very careful.
```

Suppose all the members of your target audience were using a sophisticated ART-enabled browser capable of displaying a fallback image if another image could not be successfully displayed. In this case, depending on the preferences that the user had set, the browser might attempt to select the ART file, then TIFF, then GIF. If for some reason it had trouble displaying all of the images, it would display the text associated with the ALT attribute.

# Client-Side Image Mapping

## HTML 0

### Introduction

Image maps allow users to access documents by clicking on different areas in an image.

### Tutorial

There is a [tutorial](#) available for the Map Builder.

### Syntax

Adding a USEMAP attribute to an IMG element indicates that it is a client-side image map. The argument to USEMAP specifies which map to use with the image, in a format similar to the HREF attribute on anchors. If the argument to USEMAP starts with a '#', the map is assumed to be in the same document as the IMG tag.

The different regions of the image are described using a MAP element. The map describes each region in the image and indicates where it links to. The basic format for the MAP element is as follows:

```
<MAP NAME="name">
<AREA [SHAPE="shape"] COORDS="x,y,..." [HREF="reference"] [NOHREF]>
</MAP>
```

The name specifies the name of the map so that it can be referenced by an IMG element. The shape gives the shape of this area. Currently the only shape defined is "RECT", but the syntax is defined in such a way to allow other region types to be added. If the SHAPE tag is omitted, SHAPE="RECT" is assumed. The COORDS tag gives the coordinates of the shape, using image pixels as the units. For a rectangle, the coordinates are given as "left,top,right,bottom". The rectangular region defined includes the lower-right corner specified, i.e. to specify the the entire area of a 100x100 image, the coordinates would be "0,0,99,99".

The NOHREF tag indicates that clicks in this region should perform no action. An HREF tag specifies where a click in that area should lead. Note that a relative anchor specification will be expanded using the URL of the map description as a base, rather than using the URL of the document from which the map description is referenced. If a BASE tag is present in the document containing the map description, that URL will be used as the base.

An arbitrary number of AREA tags may be specified. If two areas intersect, the one which appears first in the map definition takes precedence in the overlapping region. For example, a button bar in a document might use a 160 pixel by 60 pixel image and appear like this:

```
<MAP NAME="buttonbar">
<AREA SHAPE="RECT" COORDS="10,10,49,49" HREF="about_us.html">
<AREA SHAPE="RECT" COORDS="60,10,99,49" HREF="products.html">
<AREA SHAPE="RECT" COORDS="110,10,149,49" HREF="index.html">
<AREA SHAPE="RECT" COORDS="0,0,159,59" NOHREF>
</MAP>
<IMG SRC="../../../images/tech/bar.gif" USEMAP="#buttonbar">
```

This example includes a region encompassing the entire image with a NOHREF tag, but

this is actually redundant. Any region of the image that is not defined by an AREA tag is assumed to be NOHREF.

**Examples:**

```
<IMG SRC="../../../images/picture1.gif" USEMAP="maps.html#map1">
```

This states that when picture1.gif is clicked on, the browser will look up the map file (map1) found inside *maps.html* to determine what action to take.

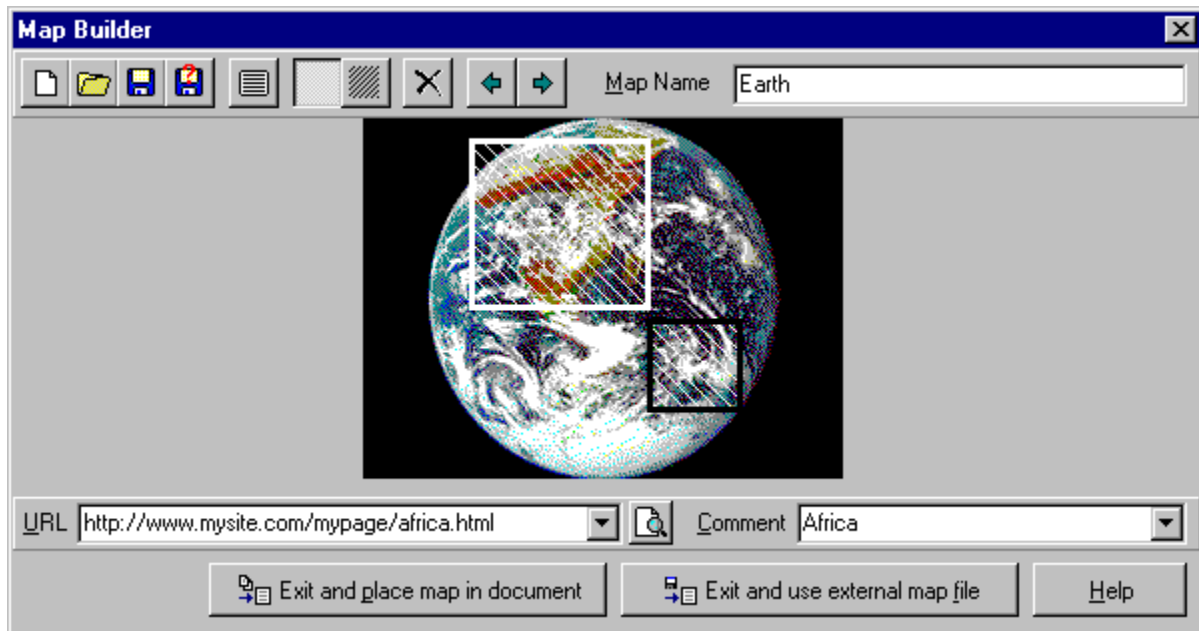
```
<IMG SRC="../../../images/picture1.gif" USEMAP="#map1">
```

This is very similar to above except that the map file (map1) is looked for in the current HTML document rather than an external file.

```
<A HREF="no_csim.html"> <IMG SRC="../../../images/picture3.gif" USEMAP="maps.html#map3"></A>
```

This allows for browsers that do not support client-side image mapping. In such a browser clicking on picture3.gif will access the no\_csim.html document.

# Map Builder Tutorial



After starting the Map Builder choose the new session button. Now select the image that will be used as the basis for the map. The picture will load and new buttons and text boxes will appear. Give your map a name by entering it in the Map Name box located in the upper right hand corner. Next, select an area with the mouse. Now we must assign a URL to be called when this area is selected. Do this by entering a URL in the lower left hand text box (the URL builder button, located to the right of the URL text box can be used to quickly build a URL). Optionally, a comment can be added describing the purpose of this area. More areas can be added - to move from area to area use the arrow buttons. The currently selected area will be outlined in white. If your image is composed of mostly light colors it might make the areas easier to see by selecting the black area fill button.

At any time you can view the current text of the map file by clicking on the view map tags button. When finished creating areas either of the two large buttons on the bottom of the form will return you to the image window. 'Exit and place map in document' will place the map text you have created into the html document you are currently working on. There is no limit to the number of maps withing a single document, but be sure that each map has a distictive name. When using this button you will also be prompted to save the map to a file - this is simply to back up your map file so that it can be reloaded at a later time, it does not affect the how the map file is used. The 'Exit and use external map file' button will place the map text into a seperate file; in this case the IMAGE tag in your document will reference this external file to read the map information.

Because client-side mapping is an HTML 3.0 feature be aware that not all browsers support it. WebEdit's quick preview, being only HTML 2.0 compatable, does not allow you to test your map; however, using the 'View Document with Browser' feature along with an HTML 3.0 compatable browser such as Netscape 2.0 will.



# AVI Video

## HTML 0

### Description

These attributes allow you to embed .avi (Audio Video Interleave) video clips in HTML pages.

### Attributes

**CONTROLS** - **HTML 0** If a video clip is present, a set of controls is displayed under the clip.

```
<IMG DYN SRC="fun.avi" CONTROLS><BR>
```

The above video has a set of transport controls under it.

**DYN SRC=URL** - **HTML 0** Specifies the address of a video clip or VRML world to be displayed in the window. Stands for Dynamic Source.

```
<IMG SRC="sample.gif" DYN SRC="test.avi">
```

If your browser supports inline video, you will see the movie Test.avi; otherwise you will see the picture Sample.gif.

**LOOP=n, LOOP=INFINITE** - **HTML 0** Specifies how many times a video clip will loop when activated. If n=-1, or if LOOP=INFINITE is specified, it will loop indefinitely.

```
<IMG SRC="preview1.gif" DYN SRC="movie.avi" LOOP=3><BR>
```

The above video will loop three times when activated.

```
<IMG SRC="preview1.gif" DYN SRC="movie.avi" LOOP=INFINITE><BR>
```

The above video will loop indefinitely until stopped.

**START= FILEOPEN and/or MOUSEOVER** - **HTML 0** For video clips: specifies when the file should start playing. FILEOPEN means start playing as soon as the file is done opening. This is the default. MOUSEOVER means start playing when the user moves the mouse cursor over the animation.

```
<IMG SRC="preview1.gif" DYN SRC="skiing.avi" START=FILEOPEN><BR>
```

The above video will start playing as soon as it is opened.

```
<IMG SRC="preview2.gif" DYN SRC="curling.avi" START=MOUSEOVER LOOP=5><BR>
```

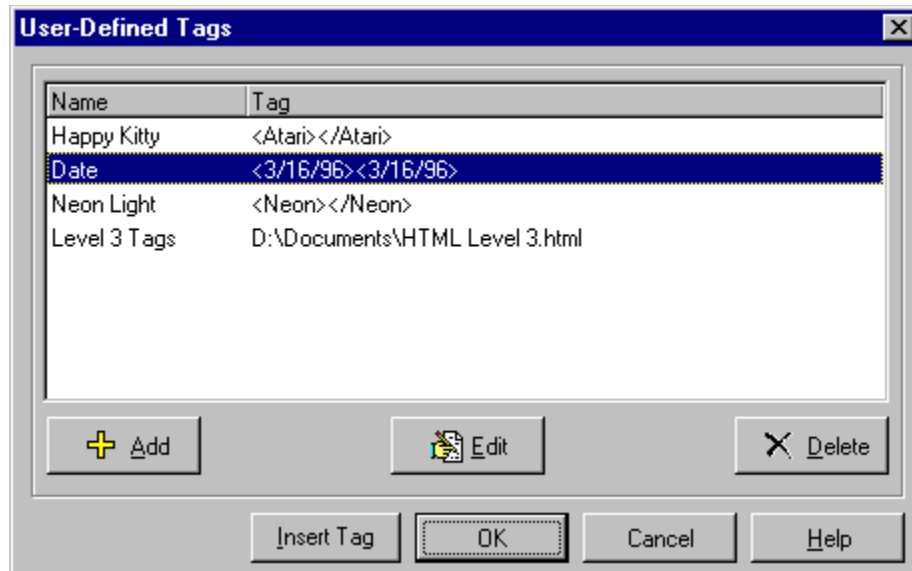
The above video will start playing when the user moves the mouse over it, and will loop five times before stopping. IExplore The user can specify both together.

```
<IMG SRC="preview3.gif" DYN SRC="windsurfing.avi"
```

```
START=MOUSEOVER, FILEOPEN><BR>
```

The above video will play once as soon as it opens and thereafter will play whenever the user moves the mouse over it.

## User-Defined Tags



### Description

User-Defined Tags are one way of ensuring that WebEdit allows you to stay absolutely current with the latest feature of HTML. Suppose, for example, that your favorite browser software develops a tag of their own that is not included with your version of WebEdit. Or you might frequently nest certain tags. With User-Defined Tags you can create your own. To create a User-Defined Tag select New... and type in the tag exactly as you want it, then select OK. Whenever you need to use this tag again just come back to User Defined Tags and select the tag, it will be inserted into the open HTML document at the cursor.

You can also specify the name of a text file, e.g., MYFILE.HTM. Choosing this tag and clicking OK causes the contents of the file MYFILE.HTM to be inserted into the active document and the current insertion point.

### Example

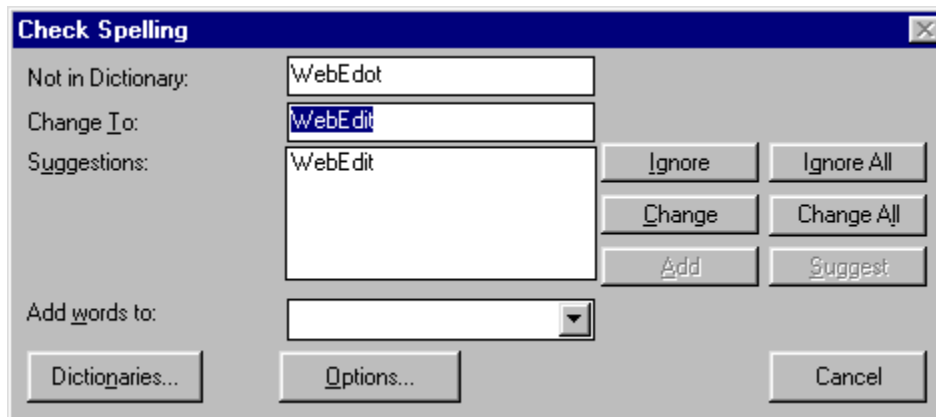
Some browser that you want to write HTML for might support a feature of their own called NEON. You could create the NEON User-Defined Tag.

```
<NEON></NEON>
```

## **Quick Previewer**

WebEdit's quick preview utility is a quick and easy way to preview your HTML documents. It supports all HTML tags up to and including HTML 2.0, plus Tables, Forms and Fonts from HTML 3.2. To view browser specific tags such as Netscape Navigator or Internet Explorer specific tags you can use WebEdit's 'View Document with Browser' button in combination with a browser that supports the features you need.

# Spelling Checker



To check the spelling of a document, select Tools|Spelling... or press the F7 key. The Check Spelling Dialog appears if a word requiring your attention is detected. You can use the dialog to specify whether the word should be ignored or replaced.

**Add:** Causes the reported word to be added to the dictionary selected in the Add Words To list. Use the add button if correctly spelled word you use often is reported as a misspelling (e.g., your family name). If the word is not used frequently, you may want to select the Ignore or Ignore All buttons instead.

**Add words to:** Indicates which user dictionary words will be added to when you select the Add button. The Add Words To list shows all ignore-type user dictionaries currently open. You can open or close other dictionaries via the Dictionaries dialog, which is accessible by selecting the Dictionaries button.

**Cancel:** Stops the current spell-checking operation.

**Change:** Causes the reported word to be replaced with the word in the Change To field. Only this occurrence of the reported word is replaced. If you want this and all following occurrences of the word replaced, select the Change All button.

**Change all:** Causes this and all following occurrences of the reported word to be replaced with the word in the Change To field. If you want only this occurrence of the word to be replaced, use the Change button. If the reported word is one you frequently misspell, you might consider adding it to a change-type or suggest-type dictionary via the Dictionaries dialog. You can display the Dictionaries dialog by selecting the Dictionaries button.

**Change to:** Contains a word which will replace a misspelled word when you select the Replace or Replace All buttons. You can enter a word in the Change to field by typing, or you can select one of the suggested replacements from the Suggestions list.

**Consider changing:** Contains a word which may be misspelled or otherwise incorrect, and is presented with a candidate replacement word. You can change the word by selecting the Change button, or skip it by selecting the Ignore button.

**Dictionaries:** Causes the Dictionaries dialog to be displayed. You can use the Dictionaries dialog to open or close user dictionaries, and to edit the contents of user dictionaries.

**Ignore:** Causes this occurrence of a misspelled word to be skipped. If the same misspelled word appears later, it will be reported.

**Ignore All:** Causes this and all further occurrences of a misspelled word to be skipped. You might use this button if the word reported as a misspelling is actually correctly spelled. If the word is one you use frequently, you may wish to ignore it permanently by selecting the Add button.

**Mixed case:** Indicates that a word containing an unusual combination of upper- and lower-case letters was detected. Examples of such words include "hapPy," "PrintScreen," and "DoT." Words containing mixed case are reported only if the "Report words with mixed case" option is enabled.

**Not in dictionary:** Indicates that a misspelled word was detected. The word is considered misspelled because it could not be located in any open dictionaries, or was located in an exclude-type dictionary.

**Options:** Causes the Options dialog to be displayed. You can use the Options dialog to set spelling-checker options.

**Suggest:** Causes a set of suggested replacements for misspelled words to be added to the Suggestions list. This button is enabled only if the Always Suggest option is disabled. Each time you select the Suggest button, a more intensive search for replacements is conducted. Once all possible suggestions are located, the Suggest button is disabled.

**Suggestions:** Contains a list of suggested replacements for the word reported as misspelled. If you have enabled the Always Suggest option, this list will be filled in automatically when a misspelled word is reported. Otherwise, the list is filled in when you select the Suggest button. You can use the words in the list to select a replacement for the misspelled word.

Dictionaries Dialog

The Dictionaries Dialog allows you to open and close user dictionaries, and to edit the contents of an open user dictionary. The contents of dictionaries are saved in disk files. You can open some or all of your user dictionary files at any time. Only open dictionaries are checked during a spell-checking operation.

**Add File:** Opens a user dictionary file. When you select the Add File button, a dialog appears which you can use to select the dictionary file to open. The set of open dictionary files is remembered, so once you add a dictionary file you don't need to add it again. If you need to create a new user dictionary, use the New button.

**Add Word:** Causes the word entered in the edit area of the Words list to be added to the currently selected dictionary. Note that if the dictionary type is "auto change" or "conditionally change," you must enter a word and a replacement, separated by a colon (e.g., "teh:the").

**Close:** Closes the Dictionaries dialog.

**Delete Word:** Causes the word appearing in the edit area of the Words list to be removed from the currently selected dictionary. If the dictionary type is "auto change" or "conditionally change," enter just the word without the colon or replacement word.

**Export:** Saves the contents of the currently selected dictionary to a text file. When you select the Export button, a dialog appears which you can use to select the name of the text file to which words in the dictionary will be exported. The words are written to the file one per line. If the dictionary type is "auto change" or "conditionally change," words and their replacements are written, separated by a colon.

**Files:** Contains the list of open dictionary files.

**Import:** Adds the words contained within a text file to the currently selected dictionary. When you select the Import button, a dialog appears which you can use to select the text file to be imported. Each word in the selected file is loaded into the dictionary. If the dictionary type is "auto change" or

"conditionally change," words in the file must be in *word:replacement* form (e.g., "teh:the").

**Language:** Displays the language (e.g., English or French) of the words in the currently selected dictionary.

**New:** Creates a new user dictionary file. When you select the New button, a dialog appears which you can use to specify attributes of the new dictionary. See the New Dictionary dialog for details.

**Remove File:** Closes the currently selected dictionary file. Closed dictionaries are not checked during a spelling check. Although the file is closed, it is not deleted. Closed dictionary files can be later reopened using the Add File button.

**Type:** Displays the type or purpose of the currently selected user dictionary. The dictionary type indicates what will happen if a word is located in the dictionary during a spelling check. For information about the specific types, see Dictionary Types.

**Words:** Contains the list of words in the currently selected user dictionary. If the dictionary type is "auto change" or "conditionally change," then the list contains words and their replacements, separated by colons.

Options Dialog

You can use the Options Dialog to specify various spelling-checker options. These options affect the way the spelling checker operates.

**Ignore Capitalized Words:** When enabled, any words beginning with a capital letter are ignored (i.e., are skipped over without being checked). You might enable this option if the text being checked contains many proper names.

**Ignore All-Caps Words:** When enabled, any words containing all capital letters are ignored (i.e., are skipped without being checked). You might enable this option if the text being checked contains many acronyms.

**Ignore Words with Numbers:** When enabled, any words containing embedded digits are ignored (i.e., are skipped without being checked). Examples of such words include "Win95" and "Q4." You might enable this option if the text being checked contains many code-words or other symbols containing digits.

**Ignore Words with Mixed Case:** When enabled, any words containing an unusual mixture of upper- and lower-case letters are ignored (i.e., are skipped without being checked). Examples of such words include "MicroHouse" and "CapsLock." You might enable this option if the text being checked contains many variable names or other symbols which use case changes to distinguish words.

**Report Words with Mixed Case:** When enabled, any words containing an unusual mixture of upper- and lower-case letters are reported via the Check-Spelling Dialog. Examples of such words include "The" and "Monday."

**Report Doubled Words:** When enabled, any word appearing twice in a row is reported via the Check-Spelling Dialog.

**Always Suggest:** When enabled, a list of suggested replacements is automatically displayed when a misspelled word is reported. If this option is disabled, a list of suggestions can be obtained by selecting the Suggest button in the Check-Spelling dialog.

**OK:** Closes the Options Dialog, and saves any changes made to the option settings.

**Cancel:** Closes the Options Dialog, and discards any changes made to the option settings.

## New Dictionary Dialog

You can use the New Dictionary Dialog to specify the attributes of a new user dictionary.

**Browse:** Displays a dialog which shows the names of other user dictionary files. You can use the dialog to view the names of existing dictionary files, and to enter the name of the new dictionary file.

**Cancel:** Closes the New Dictionary Dialog without creating a new dictionary.

**File Name:** Contains the name of the disk file used to hold the new dictionary's contents. You can enter a name here or use the Browse button to display a dialog showing the names of other dictionary files.

**Language:** Specifies the language (e.g., French, English) of the words the new user dictionary will contain. If the language you want to use is not listed, select "Other."

**OK:** Closes the New Dictionary Dialog and creates the new dictionary.

**Type:** Specifies the type or purpose of the new user dictionary. The type defines what happens when a word is located in the dictionary during a spelling check. For information on the different dictionary types, see Dictionary Types.

### Dictionary Types

User dictionaries can be in several different types. The type indicates what happens when a word is found in the dictionary during a spelling check.

**Auto Change:** Words found in an Auto Change dictionary are automatically replaced with other words. Typically, Auto Change dictionaries hold frequently misspelled words and their correct replacements. For example, an Auto Change dictionary might contain the misspelled word "recieve" and its replacement "receive." Each entry in an Auto Change dictionary contains a word and the replacement, separated by a colon (e.g., "recieve:receive").

**Conditional Change:** Words found in a Conditional Change dictionary are presented to you as candidates for replacement, and are replaced with other words if you request. Typically, Conditional Change dictionaries hold potentially misspelled words and their usual replacements. Each entry in a Conditional Change dictionary contains a word and the replacement, separated by a colon (e.g., "recieve:receive").

**Exclude:** Words found in an Exclude dictionary are always considered misspelled, even if they are defined in other dictionaries. Typically, Exclude dictionaries hold words you may use but don't want to appear in your writing. You might also use an Exclude dictionary to hold words you type by accident -- for example, you might enter "newt" in an Exclude dictionary if you occasionally type "newt" when you mean to type "neat."

**Ignore:** Words found in an Ignore dictionary are considered correctly spelled. Typically, Ignore dictionaries hold words you use but which don't appear in the other dictionaries. For example, you may add your family name or street name to an Ignore dictionary.

### Additional Lexicons

The WebEdit spelling checker supports optional lexicons in various languages. These additional lexicons can be downloaded from the WebEdit download page at:

<http://www.nesbitt.com/download.html>

If you plan to use multiple lexicons, you should also download the WebEdit lexicon manager. The lexicon manager makes it easy activate and deactivate installed lexicons. The lexicon manager is

also available from the WebEdit download page.

The Sentry Spelling-Checker Engine Copyright © 1993 Wintertree Software Inc.



# Learning HTML

[What is HTML?](#)

[Getting Started With HTML](#)

[HTML Elements](#)

[HTML 3.2 Document Structure](#)

[Glossary of Terms](#)

[Index of Tags](#)

# Getting Started With HTML

## Getting Connected

An Internet service provider can get you connected to the Internet. The package they provide you with may not include a browser but it will almost always include an ftp utility. With ftp you can login to various file servers and download the current shareware versions of popular browsers.

If you don't know where to begin, there are many good books that can help you get started. In particular, we recommend *The Internet Complete Reference* and *The Internet Yellow Pages*, both by Harley Hahn.

## Browse The Web

Although you can use WebEdit without any experience at writing HTML, you probably would not have a clear idea of the Web page you want to create. You would not try to publish a book without ever having looked through a book. Similarly, you should spend as many hours as possible browsing the Web and learning all the features of your browser. Spend your time wisely, with an eye towards style. Find pages with a look and feel that appeals to you and add them to your "bookmarks" (in Netscape), or "hotlist" (in Mosaic). Also, some browsers allow you to see the underlying HTML source when you are viewing a Web page - if you are using Netscape select 'Source' under the 'View' menu.

## WebEdit Home Page

Once you are browsing the Web you have the means to view Web page tutorials on the subject of HTML. There are many HTML primers and tutorials available on the World Wide Web. If you are new to HTML you will almost certainly find these documents indispensable. The WebEdit home page provides hotlinks to several of these. You can connect to the WebEdit home page at: <http://www.nesbitt.com/>

## Find a Good Example

One way to get started with HTML is to browse the Web until you find a Web page you like. Select Save As from your browsers file menu and save the file on your hard drive. You can now open this file with WebEdit and modify it. Make some simple changes to the text and save it. Select Open Local File from the File menu of your browser. If you see the changes you made then you have successfully modified an HTML document. At this point you may want to study over the document you just modified so that you can learn how to change the structure of the document as well as its content. One of the best ways to learn is by taking and modifying an existing Web page, notice what effect each of your changes has.

## View the page as you create it

WebEdit offers two ways for you to visually see your Web page. The first is WebEdit's built in Quick Previewer. The second is the Check Document button. This provides an ideal way to check the progress of your Web page. From the **Options** menu of WebEdit choose the item Browser. Change to the directory for your browser and select the executable file, e.g., netscape.exe. Once you have a browser configured you can check the appearance of your HTML document and test its functionality by clicking on the Check button. This will automatically load up your browser and pass it your Web page to display.

## Defining Your Home Page

You can have an HTML document load automatically by making it your default home page. To do this select Preferences from your browsers Options menu and enter the

path to your local file, e.g., **file:///c:/webedit/welcome.html**.

# HTML Elements

## Description

Tags indicate the beginning and end of titles, headings, paragraphs and links. HTML elements are identified as a start tag, giving the element name and attributes, then the content, then the end tag. Start tags are indicated with `<element>`, and end tags are indicated with `</element>`. For example:

```
<H1>Level One Heading</H1>
<P>A very short paragraph.
```

Some elements consist of only a start tag. A line break, for instance, is simply `<BR>`. Also some elements do not require end tags, e.g., P, LI, because the end tag is implied by the context. The content of an element consists of characters or other nested elements.

## Names

The element name immediately follows the tag open delimiter. An element name consist of a letter followed by up to 72 letters, digits, periods, or hyphens. Names are not case sensitive. For example, H1 is equivalent to h1. This limit of 72 characters is set by the NAMELEN parameter in the SGML declaration for HTML 3.0.

## Attributes

In a start tag, white space and attributes are allowed between the element name and the closing delimiter. An attribute typically consists of an attribute name, an equal sign, and a value (although some attributes may be just a value). White space is allowed around the equal sign.

The value of the attribute may be either:

- A string literal, delimited by single quotes or double quotes
- A name token (a sequence of letters, digits, periods, or hyphens)

In this example, A is the element name, HREF is the attribute name, and `http://host/dir/file.html` is the attribute value:

```
<A HREF="http://host/dir/file.html">
```

Some implementations consider any occurrence of the `>` character to signal the end of a tag. For compatibility with such implementations, when `>` appears in an attribute value, you may want to represent it with an entity or numeric character reference, such as:

```
<IMG SRC="eq1.ps" alt="a &#62; b">
```

To put quotes inside of quotes, you can use single quotes if the outer quotes are double or vice versa, as in:

```
<IMG SRC="image.ps" alt="First 'real' example">
```

Alternatively, you use the character representation `&quot;`; as in:

```
<IMG SRC="image.ps" alt="First &quot;real&quot;; example">
```

The length of an attribute value (after replacing entity and numeric character references) is limited to 1024 characters. This number is defined by the LITLEN

parameter in the SGML declaration for HTML 3.0.

**Note:** Some implementations allow any character except space or > in a name token. Attributes values must be quoted only if they don't satisfy the syntax for a name token.

Attributes with a declared value of NAME (e.g. ISMAP, COMPACT) may be written using a minimized syntax. The markup:

```
<UL COMPACT="compact">
```

can be written as:

```
<UL COMPACT>
```

**Note:** Unless you use the minimized syntax, some implementations won't understand.

### Special Characters

The characters between the tags represent text in the ISO-Latin-1 character set, which is a superset of ASCII. Because certain characters will be interpreted as markup, they should be represented by markup -- entity or numeric character references, for instance the character "&" must be represented by the entity &amp;

### Comments

To include comments in an HTML document that will be ignored by the parser, surround them with <!-- and -->. After the comment delimiter, all text up to the next occurrence of --> is ignored. Hence comments cannot be nested. White space is allowed between the closing -- and >, but not between the opening <! and --. For example:

```
<HEAD>
<TITLE>HTML Guide: Recommended Usage</TITLE>
<!-- Id: Text.html,v 1.6 1994/04/25 17:33:48 connolly Exp -->
</HEAD>
```

# HTML 3.2 Document Structure

## The Structure of HTML 3.2 Documents

Every HTML 3.2 document begins with the following `<!DOCTYPE>` declaration (to distinguish HTML 3.2 from other versions of HTML), followed by `HEAD` and `BODY` elements. The `TITLE` tags are required. All other tags are optional.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>A study of population dynamics</TITLE>
... other head elements
</HEAD>
<BODY>
... document body
</BODY>
</HTML>
```

Note that in practice, many documents don't contain a `<!DOCTYPE>` declaration. This makes it difficult for browsers, validation tools, and other software to determine the version of HTML used in the document.

### The **HEAD** element

This contains the document head, but you can always omit both the start and end tags for **HEAD**. The following elements belong to the document head:

- TITLE** defines the document title, and is always needed.
- ISINDEX** for simple keyword searches, see `PROMPT` attribute.
- BASE** defines absolute URL for resolving relative URLs.
- STYLE** reserved for future use with style sheets.
- SCRIPT** reserved for future use with scripting languages.
- META** used to supply meta info as name/value pairs.
- LINK** used to define relationships with other documents.

`TITLE`, `STYLE` and `SCRIPT` are containers and requires both start and end tags. The other elements are not containers so that end tags are forbidden. Note that conforming browsers won't render the contents of `STYLE` and `SCRIPT` elements.

### The **BODY** element

This contains the document body. Both start and end tags for `BODY` may be omitted. The key attributes are: `BACKGROUND`, `BGCOLOR`, `TEXT`, `LINK`, `VLINK` and `ALINK`. These can be used to set a repeating background image, plus background and foreground colors for normal text and hypertext links. Colors are given in RGB as hexadecimal numbers (e.g. `#C0FFC0`) or as one of 16 widely understood color names:

```
aqua, black, blue, fuchsia, gray, green, lime, maroon,
navy, olive, purple, red, silver, teal, white, and yellow
```

These colors were originally picked as being the standard 16 colors supported with the Windows VGA palette.

## Block and Text level elements

Most elements that can appear in the document body fall into one of two groups: block level elements which cause paragraph breaks, and text level elements which don't. Common block level elements include H1 to H6 (headers), P (paragraphs) LI (list items), and HR (horizontal rules). Common text level elements include EM, I, B and FONT (character emphasis), A (hypertext links), IMG and APPLET (embedded objects) and BR (line breaks). Note that block elements generally act as containers for text level and other block level elements (excluding headings and address elements), while text level elements can only contain other text level elements. The exact model depends on the element.

## Headings

H1, H2, H3, H4, H5 and H6 are used for document headings. You always need the start and end tags. H1 elements are more important than H2 elements and so on, so that H6 elements define the least important level of headings. More important headings are generally rendered in a larger font than less important ones. Use the ALIGN attribute to set the text alignment within a heading, e.g.

```
<H1 ALIGN=CENTER> ... centered heading ... </H1>
```

## ADDRESS

The ADDRESS element is used for information about the author of the document. It requires start and end tags.

## Block elements

### P paragraphs

The paragraph element requires a start tag, but the end tag can always be omitted. Use the ALIGN attribute to set the text alignment within a paragraph, e.g. <P ALIGN=RIGHT>

### UL unordered lists

These require start and end tags, and contain one or more LI elements representing individual list items.

### OL ordered (i.e. numbered) lists

These require start and end tags, and contain one or more LI elements representing individual list items.

### DL definition lists

These require start and end tags and contain DT elements that give the terms, and DD elements that give corresponding definitions.

### PRE preformatted text

Requires start and end tags. These elements are rendered with a monospaced font and preserve layout defined by whitespace and line break characters.

### DIV document divisions

Requires start and end tags. It groups related elements together and can be used with the ALIGN attribute to set the text alignment of the block elements it contains. ALIGN can be one of LEFT, CENTER or RIGHT.

**CENTER** text alignment

Requires start and end tags. It is used to center text lines enclosed by the CENTER element. See DIV for a more general solution.

**BLOCKQUOTE** quoted passage

Requires start and end tags. It is used to enclose extended quotations and is typically rendered with indented margins.

**FORM** fill-out forms

Requires start and end tags. This element is used to define a fill-out form for processing by HTTP servers. The attributes are ACTION, METHOD and ENCTYPE. Form elements can't be nested.

**ISINDEX** primitive HTML forms

Not a container, so the end tag is forbidden. This predates FORM and is used for simple kinds of forms which have a single text input field, implied by this element.

**HR** horizontal rules

Not a container, so the end tag is forbidden. attributes are ALIGN, NOSHADE, SIZE and WIDTH.

**TABLE** can be nested

Requires start and end tags. Each table starts with an optional CAPTION followed by one or more TR elements defining table rows. Each row has one or more cells defined by TH or TD elements. Attributes for TABLE elements are WIDTH, ALIGN, BORDER, CELSPACING and CELLPADDING.

## Lists

List items can contain block and text level items, although headings and address elements are excluded.

Unordered lists take the form:

```
<UL>
  <LI> ... first list item
  <LI> ... second list item
  ...
</UL>
```

The TYPE attribute can be used to set the bullet style on UL and LI elements.

Ordered (i.e. numbered) lists take the form:

```
<OL>
  <LI> ... first list item
  <LI> ... second list item
  ...
</OL>
```



The OL START attribute can be used to initialize the sequence number. You can reset it later on with the VALUE attribute on LI elements.

Definition lists take the form:

```
<DL>
  <DT> term name
  <DD> term definition
  ...
</DL>
```

DT elements can only act as containers for text level elements, while DD elements can hold block level elements as well, excluding headings and address elements.

## Tables

These take the general form:

```
<TABLE BORDER=3 CELLSPACING=2 CELLPADDING=2 WIDTH="80%">
<CAPTION ALIGN=bottom> ... table caption ... </CAPTION>
<TR><TD> first cell <TD> second cell
<TR> ...
...
</TABLE>
```

The attributes on TABLE are all optional. By default, the table is rendered without a surrounding border. The table is generally sized automatically to fit the contents, but you can also set the table width using the WIDTH attribute. BORDER, CELLSPACING and CELLPADDING provide further control over the table's appearance. The ALIGN attribute can be used to position the table to the LEFT, CENTER or RIGHT. The CAPTION element is used for captions. These are rendered at the top or bottom of the table depending on the optional ALIGN attribute.

Each table row is contained in a TR element, although the end tag can always be omitted. Table cells are defined by TD elements for data and TH elements for headers. Like TR, these are containers and can be given without trailing end tags. TH and TD support several attributes: ALIGN and VALIGN for aligning cell content, ROWSPAN and COLSPAN for cells which span more than one row or column. A cell can contain a wide variety of other block and text level elements including form fields and other tables.

## Text level elements

These don't cause paragraph breaks. Text level elements that define character styles can generally be nested. They can contain other text level elements but not block level elements.

### Font style elements

These all require start and end tags, e.g.

```
This has some <B>bold text</B>.
```

**TT** teletype or monospaced text

**I** italic text style  
**B** bold text style  
**U** underlined text style  
**STRIKE** strike-through text style  
**BIG** places text in a large font  
**SMALL** places text in a small font  
**SUB** places text in subscript style  
**SUP** places text in superscript style

## Phrase Elements

These all require start and end tags, e.g.

```
This has some <EM>emphasized text</EM>.
```

**EM** basic emphasis typically rendered in an italic font  
**STRONG** strong emphasis typically rendered in a bold font  
**DFN** defining instance of the enclosed term  
**CODE** used for extracts from program code  
**SAMP** used for sample output from programs, and scripts etc.  
**KBD** used for text to be typed by the user  
**VAR** used for variables or arguments to commands  
**CITE** used for citations or references to other sources

## Form fields

INPUT, SELECT and TEXTAREA

INPUT elements are not containers and so the end tag is forbidden. INPUT, SELECT and TEXTAREA are only allowed within FORM elements. INPUT can be used for a variety of form fields including single line text fields, password fields, checkboxes, radio buttons, submit and reset buttons, hidden fields, file upload, and image buttons. SELECT elements require start and end tags and contain one or more OPTION elements. SELECT elements are used for single or multi-selection menus. TEXTAREA elements require start and end tags, and are used to define multi-line text fields. The content of the element is used to initialize the field.

## Special Text level Elements

Anchor, IMG, APPLET, FONT, BR and MAP.

### The A (anchor) element

Used to define hypertext links and also to define named locations for use as targets for hypertext links, e.g.

```
The best <a href="ouzo.html">cat</a>.
```

The attributes are: NAME, HREF, REL, REV and TITLE. HREF is used to supply a URL identifying the linked document or image etc. NAME is used to associate a name with this part of a document for use with URLs that target a named section of a document. Anchors can't be nested.

## IMG

e.g. `<IMG SRC="canyon.gif" ALT="Grand Canyon">`

Used to insert images. This is an empty element and so the end tag is forbidden. The attributes are: SRC, ALT, ALIGN, WIDTH, HEIGHT, BORDER, HSPACE, VSPACE, USEMAP and ISMAP. Images can be positioned vertically relative to the current textline or floated to the left or right. See BR with the CLEAR attribute for control over textflow.

### **APPLET**

Requires start and end tags. This element is supported by all Java enabled browsers. It allows you to embed a Java applet into HTML documents, e.g. to include an animation. The contents of the element are used as a fallback if the applet can't be loaded. The attributes are: CODE, CODEBASE, NAME, ALT, ALIGN, WIDTH, HEIGHT, HSPACE and VSPACE. APPLET uses associated PARAM elements to pass parameters to the applet.

### **FONT**

Requires start and end tags. This allows you to change the font size and/or color for the enclosed text. The attributes are: SIZE, COLOR. Colors are given as RGB in hexadecimal notation or as one of 16 widely understood color names.

### **BR**

Used to force a line break. This is an empty element and so the end tag is forbidden. The CLEAR attribute can be used to move down past floating images on either margin, e.g. `<BR CLEAR=LEFT>`.

### **MAP**

Requires start and end tags. This allows you to define client-side image maps. MAP elements contain one or more AREA elements that specify hotzones on the associated image and bind these hotzones to URLs.

# Glossary of Terms

**Attribute** - A characteristic quality of an element, other than type or content.

**Browser** - A tool used to read HTML documents. Common browsers include Netscape and Mosaic.

**CGI** - Common Gateway Interface. The format and syntax for passing information from browsers to servers via forms or document based queries in HTML.

**Document** - For the purposes of this standard, an HTML instance.

**DTD** - Document Type Definitions (DTD). A written specification of the HTML language,

**Element** - A component of the hierarchical structure defined by the document type definition; it is identified in a document instance by descriptive markup, usually a start-tag and an end-tag.

**HTML** - HyperText Markup Language.

**HTTP** - A generic stateless object-oriented protocol, which may be used for many similar tasks by extending the commands, or "methods", used. For example, you might use HTTP for name servers and distributed object-oriented systems. With HTTP, the negotiation of data representation allows systems to be built independent of the development of new representations.

**Internet Explorer** - Microsoft's browser.

**Markup** - Text added to the data of a document to convey information about it. There are four different kinds of markup: descriptive markup (tags), references, markup declarations, and processing instructions.

**MIME** - Multipurpose Internet Mail Extensions.

**Netscape** - A browser from NetScape Communications.

**Representation** - The encoding of information for interchange. For example, HTML is a representation of hypertext.

**Rendering** - Formatting and presenting information to human readers.

**SGML** - Standard Generalized Markup Language.

**Tag** - Descriptive markup. There are two kinds of tags; start-tags and end-tags. e.g. <StartTag> </EndTag>

**URI** - Universal Resource Identifiers.

**URL** - Universal Resource Locators.

**W3 (WWW)** - The World-Wide Web.

# General Reference

## MENUS

### FILE

[Open Location](#)  
[Export](#)  
[Project Manager](#)

### INSERT

[\(All Tags\)](#)  
[User-Defined Tags](#)  
[Special Characters](#)  
[ID / Class / Language](#)  
[Anchor / Link](#)  
[Image](#)  
[Current Date/Time](#)

### TOOLS

[Spelling Checker](#)  
[Easy Images](#)  
[Easy Links](#)  
[Extended Chars -> Tags](#)  
[Tags -> Extended Chars](#)  
[Modules](#)  
[Tag Checker](#)

### OPTIONS

[Configure](#)

**File**

## **Open Location**

Open Location allows you to download a document from the Net into WebEdit. The details button will allow you to see all of the information going back and forth from the remote location.

# Export

Export allows you to save your documents in different formats so that they can be read correctly by other operating systems. While these other operating systems will read files saved by WebEdit they may have strange characters at the end of each line.

Export has two options:

## **Unix**

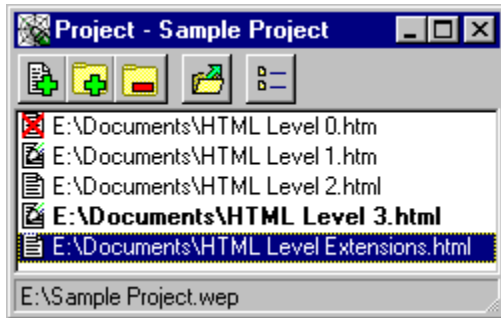
This replaces the MS-DOS standard Carriage Return/Line Feed markers at the end of every line with UNIX standard Line Feeds only.

## **Macintosh**

This replaces the MS-DOS standard Carriage Return/Line Feed markers at the end of every line with Macintosh standard Carriage Returns only.



# Project



WebEdit's project manager allows you to keep multiple files together in groups for easier, quicker access.





There are five buttons across the top of the Project Manager - from left to right they allow you to:

- Add the current document to the project
- Add a file to the project.
- Remove a document from the project.
- Open the selected documents.
- Display the full path and filename or just the filename.

You may use a combination of Shift and Control keys to select or deselect multiple files.

WebEdit can be set to close all open documents in WebEdit that belong to the project when the project is closed. See the [Configure](#) section.

The Project Manager can visually display information on each file.

 C:\Filename	A standard file.
 C:\Filename	The file is loaded.
 <b>C:\Filename</b>	The file is loaded and has been modified.
 C:\Filename	The file could not be found.

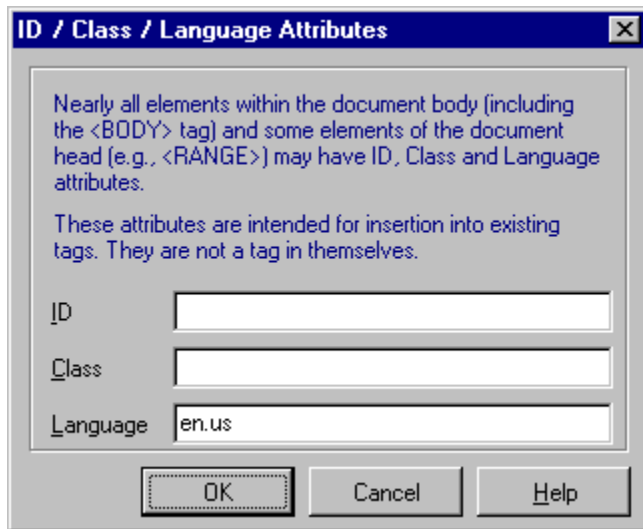
These extra abilities of the Project Manager slow it down somewhat - users with slower machines may wish to turn these functions off. See the [Configure](#) section.

**Insert**

## **Current Date/Time**

Inserts the current date and time at the cursor position

## ID / Class / Language



In HTML 3.2, most tags within the Body element can have ID, CLASS and LANG attributes.

**ID** - An SGML identifier used as the target for hypertext links or for naming particular elements in associated style sheets. Identifiers are NAME tokens and must be unique within the scope of the current document.

**LANG** - This is one of the ISO standard language abbreviations, e.g. "en.uk" for the variation of English spoken in the United Kingdom. It can be used by parsers to select language specific choices for quotation marks, ligatures and hyphenation rules etc. The language attribute is composed from the two letter language code from ISO 639, optionally followed by a period and a two letter country code from ISO 3166.

**CLASS** - This a space separated list of SGML NAME tokens and is used to subclass tag names. By convention, the class names are interpreted hierarchically, with the most general class on the left and the most

specific on the right, where classes are separated by a period. The CLASS attribute is most commonly used to attach a different style to some element, but it is recommended that where practical class names should be picked on the basis of the element's semantics, as this will permit other uses, such as restricting search through documents by matching on element class names. The conventions for choosing class names are outside the scope of this specification.

## Tools

## Extended Chars -> Tags

This will scan the HTML document changing all special characters into a form that browsers can process. The character output is the same as if the Special Characters dialog was used.

Examples:

The character 'ó' would be changed to '&oacute;'

The character 'ü' would be changed to '&uuml;'

## Tags -> Extended Chars

This will scan the HTML document changing all HTML character tags into special characters.

Examples:

The tag `&oacute;` would be changed to character 'ó'

The tag `&uuml;` would be changed to character 'ü'

## Modules

These are plug-in programs that extend the functionality of WebEdit. WebEdit ships with different Modules depending on the version of WebEdit you are using. All versions come with the Home Page Wizard. You can also [write your own Modules](#).



# Tag Checker

## Introduction

The tag checker does a basic validation of the HTML tags in the current document against several well-known standards. The tag checker does not validate attribute parameters, nor does it enforce a particular HTML markup style outside of checking for required tags.

## Supported Standards

The tag checker allows you to check the tags of your document against the HTML 2.0 standard as well as extensions supported by Netscape Navigator 1.1, Netscape Navigator 2.0, and Microsoft Internet Explorer 2.0.

## Usage

To check your tags, activate the tag checker by selecting 'Tag Checker' from the 'Tools' menu. Select a standard to compare the tags against by clicking on the appropriate radio button in the 'HTML Standard' box. Then, click on the 'Check' button to begin checking the tags.

## Output

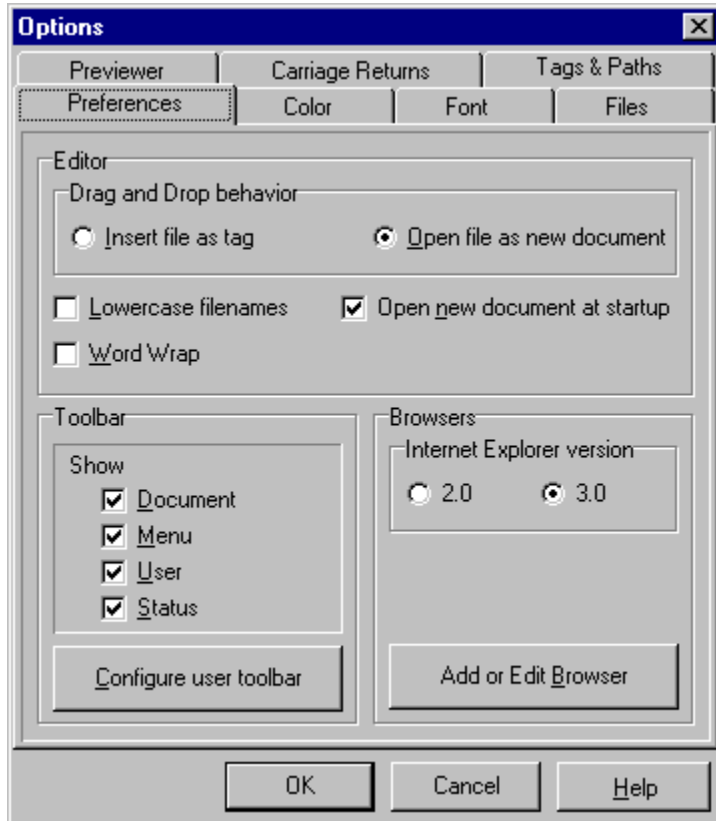
The tag checker will scan through the document and alert you of 'Warnings' and 'Errors'. Since most browsers will simply ignore tags and attributes that it doesn't understand, a Warning is printed if the tag checker finds a tag that does not comply with the selected standard. The tag checker will alert you of an error, however, when it finds something that is not proper HTML markup, such as the absence of a <TITLE> tag. Double Click on the message in the tag checker and the cursor in the WebEdit editor will go to the offending line.

## Notes

The tag checker will only check the document that was current when the tag checker was activated. To check a new document, you must first close the tag checker window, then re-open it as described above.

## Options

## Configure



This is where you can configure WebEdit to your personal preferences.

The Configuration box has eight pages

Preferences

Color

Font

Files

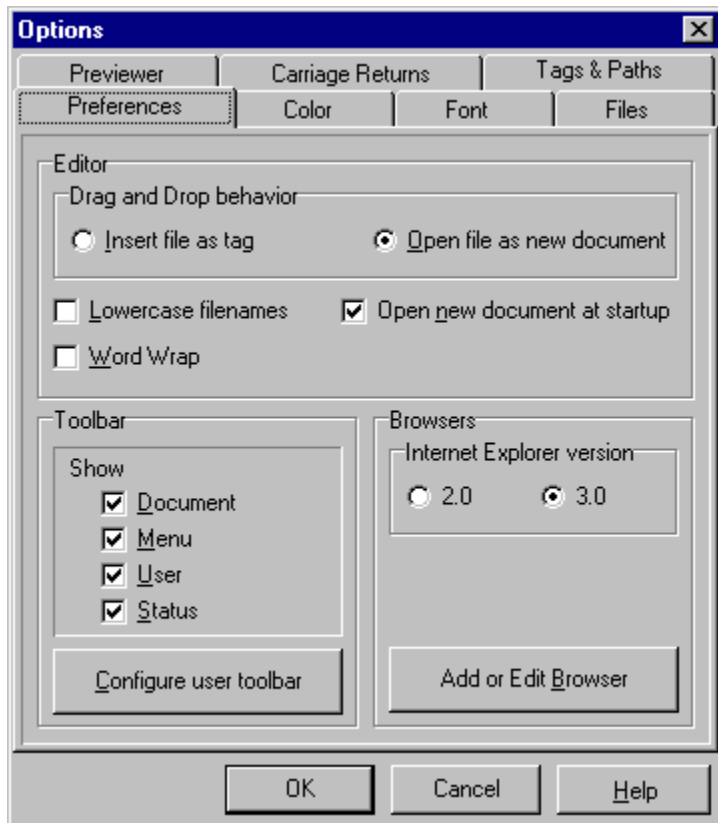
Previewer

Carriage Returns

Tags and Paths

Project Manager (Professional version)

# Preferences



## Editor

### Drag and Drop Behavior

This setting controls what WebEdit does in response to having a file or files dropped onto it.

There are two options:

#### Insert file as tag

A tag is inserted according to the type of file or files. A picture file will produce `<IMG SRC="PICTURE.JPG">`, any other file will produce `<A HREF="FILE.ZIP">`

#### Open file as new document

The file or files will be opened, ready for editing.

### Lowercase Filenames

Selects whether all filenames should hold their original case or be all lowercase. This is used primarily for compatibility reasons with Unix - some older versions of Unix require that filenames contain only lower case characters.

### Word Wrap

When selected this causes the window text to automatically wrap around to the next line when the edge of the window is reached. This is useful while editing in order to see all of the text without scrolling the window around.

**Open new document at startup**

Selects whether a new document should be opened up each time WebEdit is run.

**Syntax Highlighting**

This tells WebEdit to highlight HTML tags a different color from the normal text. This version of WebEdit only highlights tags when the document is first loaded.

**Toolbar****Show Toolbars**

Determines if the chosen toolbar should be visible.

**Configure Toolbar**

Opens up the User toolbar configuration box to allow the changing of buttons on the User toolbar.

See [Toolbars](#) for more information.

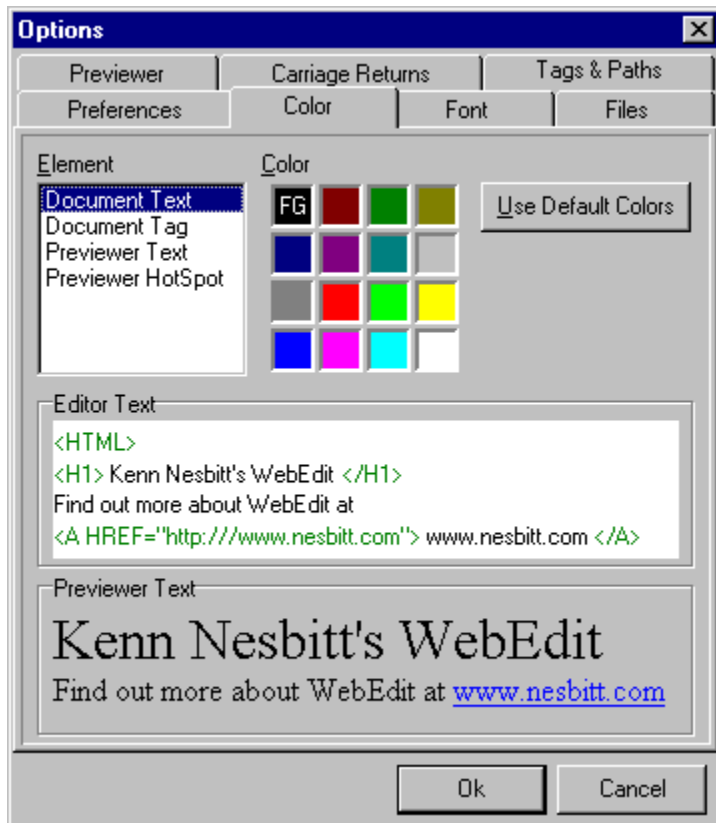
**Browsers****Internet Explorer version**

Microsoft's Internet Explorer accepts files from other programs using two different formats. Set this button according to which version of Internet Explorer you are using to check your documents with. If you are not using Internet Explorer this setting is not used.

**Add or Edit browser**

Brings up the [Browsers](#) screen to allow the addition or editing of third-party browsers.

## Color



This is where you can change some of the colors that WebEdit uses.

### Element

Select the text you whose color you wish to modify. Hint: you can also click on the text in the example boxes.

### Editor

Document Text: the standard text in your document

Document Tag: the HTML tags in your document. Make sure to turn on Syntax Highlighting to see the color. (Professional version only)

### Previewer

Previewer Text: the text in the quick previewer whose color has not been explicitly set using HTML tags.

Previewer Hotspot: the text designating a link.

### Color

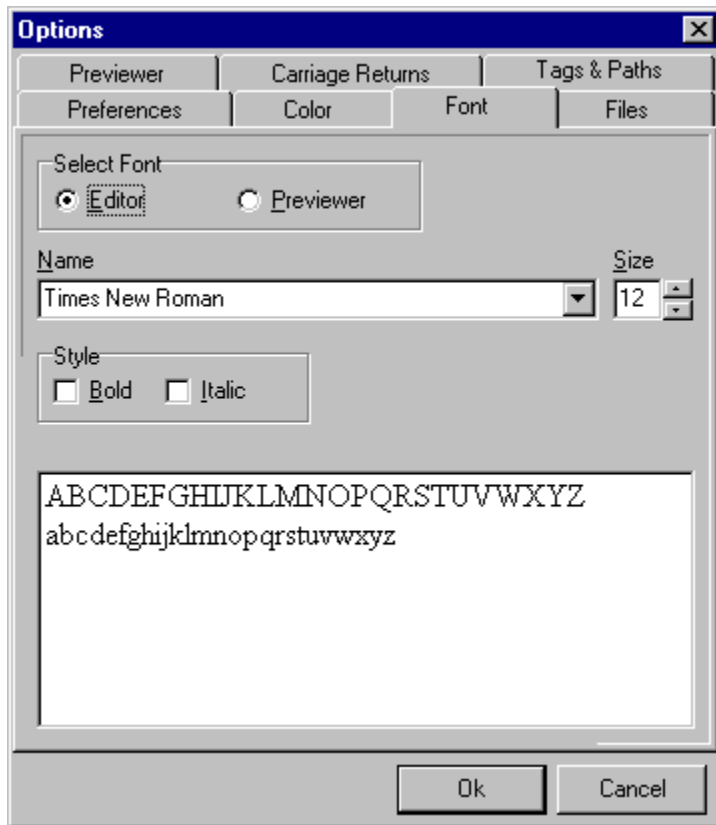
Use the left mouse button to select the foreground color or the right button to select the background color (if applicable).

### Use default colors

This will set the color back to a 'standard color'. Standard colors are black for the Document and Previewer text, green for Document tags, and blue for Previewer

Hotspots.

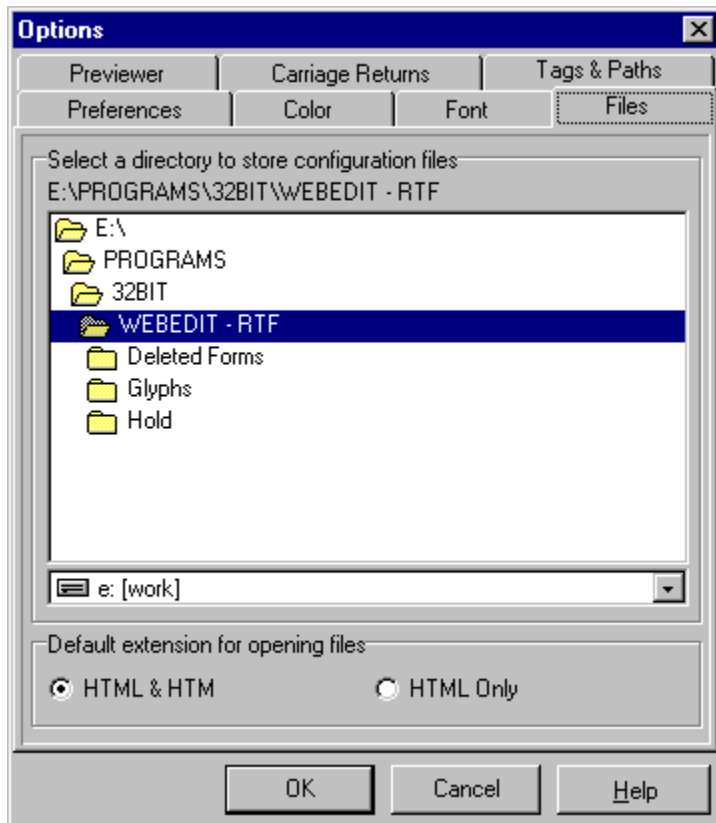
## Font



This selects the font that the HTML text is displayed in on **YOUR** system. This does **NOT** determine the font it will be seen in on the Web. Use the `<FONT>` tag to change the font as it is viewed in a browser. You can type in the sample box to see what a particular word or arrangement of letters looks like in the selected font.



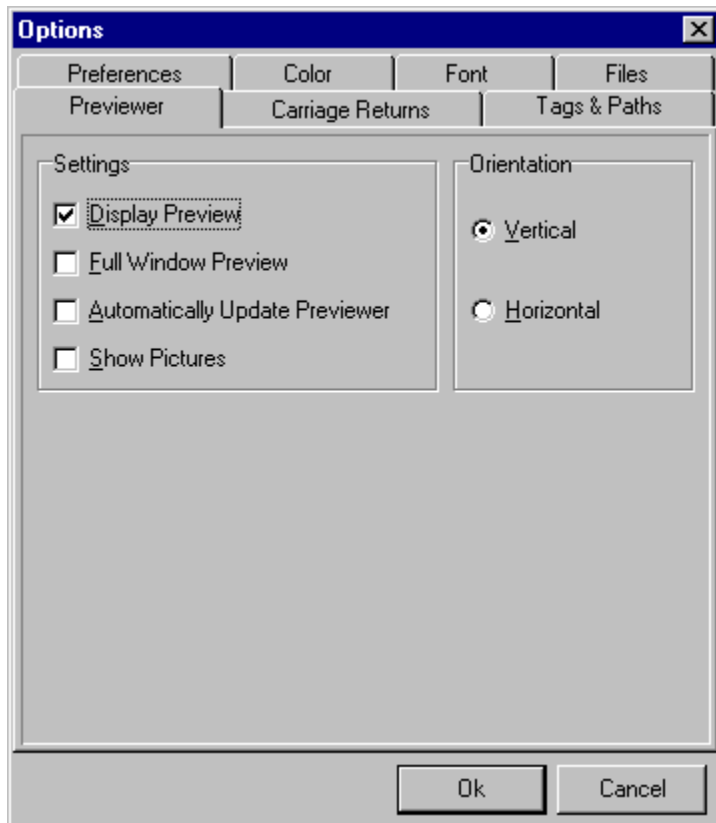
## Files



Determines the save path for all configuration files that WebEdit generates - an example being URL files. A URL file is generated the first time a dialog box containing a URL button is used. These files allow you to quickly and easily pull up URLs that you have used in the past, thus saving you much typing!

The Default extension for opening files lets you select between having WebEdit default to loading either '\*.HTML and \*.HTM' or just '\*.HTML' files. This adjustment is provided for users of certain versions of Windows NT who experience the problem of having the same file repeated twice in the File Open box.

# Previewer



## Settings

### Display Preview

Determines whether the quick viewer should be visible. A quicker way to accomplish this is to use the yellow 'lightning bolt' button on the Document Toolbar.

### Full window Preview

If set, this displays the quick previewer using the whole WebEdit screen.

### Automatically Update Previewer

This allows you to see in realtime any changes that you make to your document. Be aware that if you have large images on your page it may lag behind your typing.

### Show Pictures

If set, the quick previewer will display pictures specified by the HTML tags. If not set, the Previewer will display a placeholder.

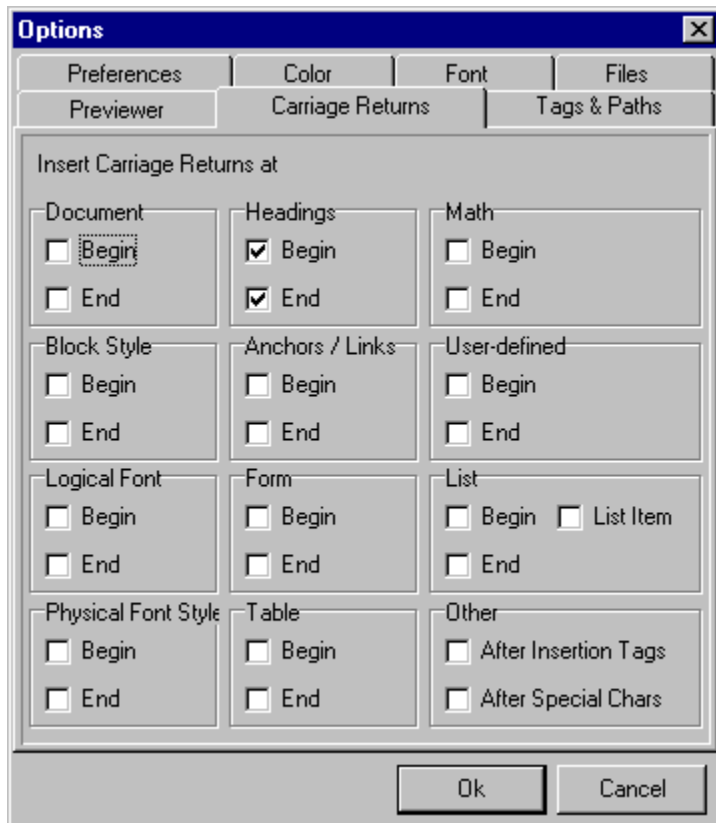
## Orientation

### Vertical & Horizontal

Vertical displays the quick previewer to the side of the document while Horizontal displays it to the bottom.

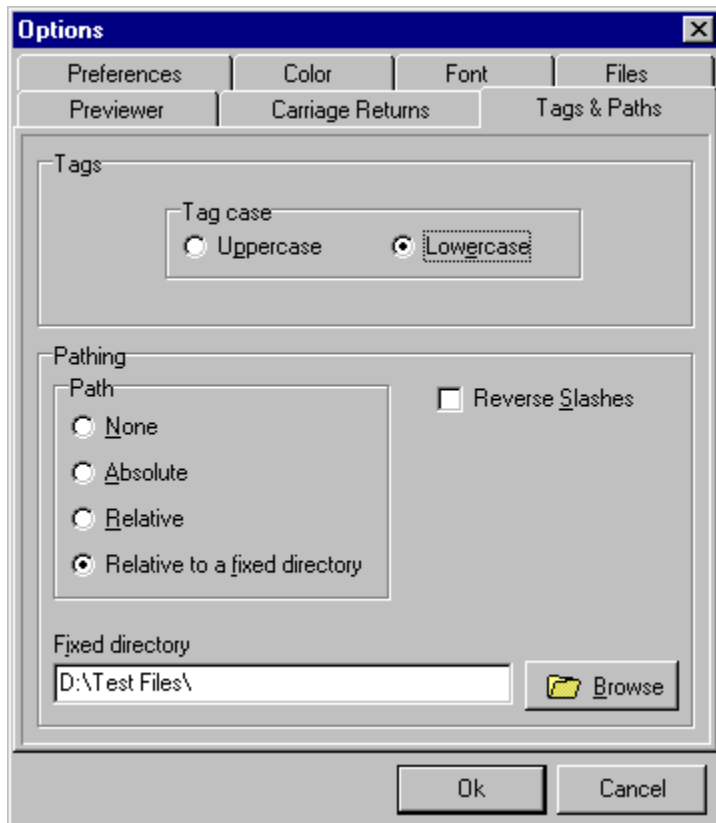


## Carriage Returns



This allows you to have WebEdit automatically place a Carriage Return and Line Feed either before or after the selected type of HTML tag.

# Tags and Paths



## Tags

### Tag Case

This selects between uppercase and lowercase tags. This only affects tags that WebEdit generates from that point on - it does not change any existing tags in the document.

## Pathing

### Path

This selection determines what type of pathing WebEdit should use when a file is selected for use in an HTML tag.

As an example let's say this is how a hard drive is laid out:

The current document in WebEdit is located at:

'C:\My HTML Documents\Home Page\Page1.html'

The file you are adding as a background is located at:

'C:\Pictures\Kitten.gif'

You have selected a fixed directory (see below) of:

'C:\My HTML Documents\Common Pictures\Construction.gif'

The filenames generated by each pathing option

None

No path is used.

'Kitten.gif'

Absolute

Only the path of the picture is used.

'C:\Pictures\Kitten.gif'

Relative

A path to the picture relative to the documents' path.

'..\..\Pictures\Kitten.gif'

Relative to a fixed directory

'..Common Pictures\Construction.gif'

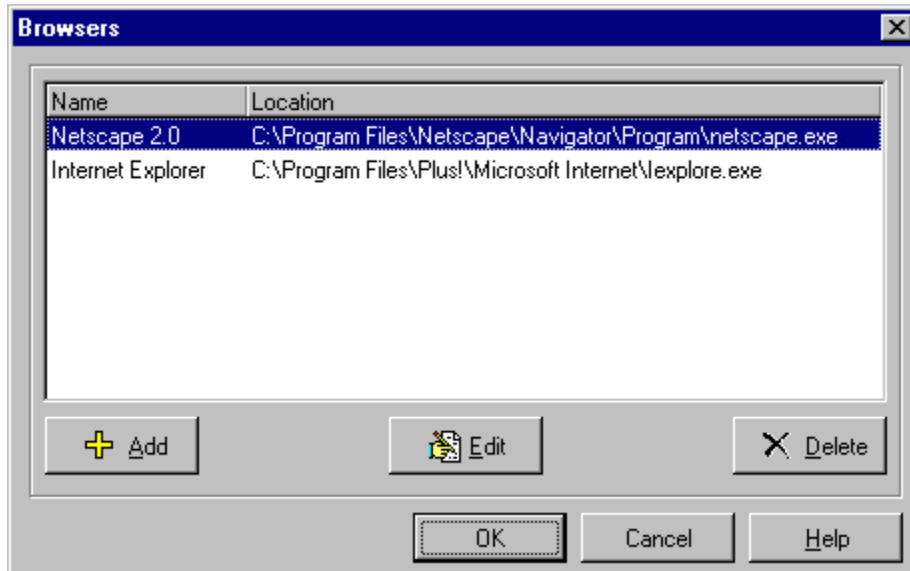
### **Fixed Directory**

This is where you select the fixed directory which is used with 'Relative to a fixed directory' pathing.

### **Reverse Slashes**

If selected, Unix '/' slashes are used, otherwise DOS slashes '\' are used.

## Browser



The browser feature allows you to preview your document using whichever browser you specify.

To add a browser, click the add button and enter the name of the browser and the location of its program file. If your browser requires any switches (ex. Mosaic takes a '-s' to operate in stand-alone mode) enter them after the filename.

You can have any number of browsers in the browser list but only the first ten will appear when the Check Document button is pressed.

# Toolbars

WebEdit has four toolbars:

## Document

This shows the operations common to all documents, such as File Open, Save, Find and Print.



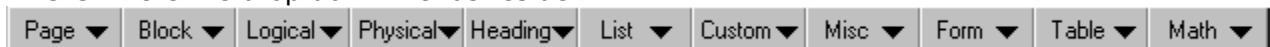
## User

This toolbar contains the user chosen buttons.



## Menu

This is where the drop down menus reside.



## Status

This reports on the status of the current document.



Cursor Position | Cursor Mode (Insert or Overwrite) | Has the file been modified? | Full path and filename | Last status message

## OPTIONS

### Show Toolbar

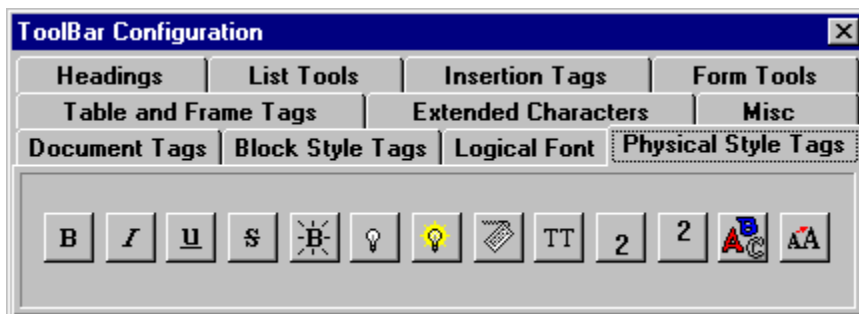
Each toolbar can be either visible or hidden determined by the presence of a checkmark next to the name of the toolbar.

### Alignment

Most toolbars can be moved to either the top, bottom, left or right side of the document. The alignment of each toolbar can be changed by right-clicking anywhere on the toolbar.

### Configure

This only applies to the user toolbar. To configure the toolbar right click on it and select Configure or go up to the main menu and select Options > Toolbars > Configure > User. Also, this may be chosen through the Options box. The Toolbar Configuration box will appear.



To add a button simply left click on it, and while holding down the left mouse button drag it



to the User toolbar.

To remove a button from the user toolbar simply left click on it, and while holding down the left mouse button drag it to the Toolbar Configuration box and let go of the button.

To re-arrange existing buttons of the user toolbar just left click on them and drag them to the new location.

# Project Manager

## HTML 0

### Close open documents on Project Manager shutdown.

When you close down Project Manager it will close down every file that belongs to the project, or not, depending on the setting of this check box.

### Extended Functionality

The Project Manager can visually display information on each file.

HTML 0	C:\Filename	A standard file.
HTML 0	C:\Filename	The file is loaded into WebEdit.
HTML 0	<b>C:\Filename</b>	The file is loaded into WebEdit and has been modified.
HTML 0	C:\Filename	The file could not be found.

These extra abilities of the Project Manager slow it down somewhat - users with slower machines may wish to turn these functions off.

## Frequently Asked Questions

[How can I get my Web pages onto the Web?](#)

[What is HTML?](#)

[How come some of my pictures don't show up?](#)

[How come some of the tags I use don't work?](#)

[How can I test my HTML page?](#)

[Where can I learn more about the Internet?](#)

[Can I write my own plug-in Modules to use with WebEdit?](#)

[Do you have any tips for using WebEdit?](#)

# What is HTML?

HTML, **hypertext markup language**, is a relatively standardized hypertext page description language, primarily used for creating hypertext pages for the World Wide Web (WWW).

Before you begin using WebEdit, you should have an understanding of HTML fundamentals. If you already know the basics, you will find that using WebEdit will help you learn HTML more thoroughly because it does much of the work for you, letting you choose HTML tags from menus and toolbars, and offering the attributes appropriate to each tag in dialog boxes.

To help you further we recommend the following documents:

## Introductory Documents

- A Beginner's Guide to HTML  
<http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>
- How to Write HTML Files  
<http://www.ucc.ie/info/net/html/doc.html>
- Introduction to HTML  
[http://melmac.harris-atd.com/about\\_html.html](http://melmac.harris-atd.com/about_html.html)

## Style Guides

- Composing Good HTML  
<http://www.willamette.edu/html-composition/strict-html.html>
- CERN's style guide for online hypertext  
<http://www.w3.org/hypertext/WWW/Provider/Style/Introduction.html>

## Reference Documents

- The HTML Quick Reference Guide  
[http://kuhttp.cc.ukans.edu/lynx\\_help/HTML\\_quick.html](http://kuhttp.cc.ukans.edu/lynx_help/HTML_quick.html)
- The Official HTML Specification  
<http://www.w3.org/pub/WWW/MarkUp>
- A Description of SGML  
<http://www.w3.org/pub/WWW/MarkUp/SGML/>
- Mosaic for X 2.0 Fill-Out Form Support  
<http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html>
- NetScape Extensions to HTML  
[http://home.netscape.com/assist/net\\_sites/html\\_extensions.html](http://home.netscape.com/assist/net_sites/html_extensions.html)
- Thalia Guide: The Background FAQ  
<http://www.sci.kun.nl/thalia/guide/color/faq.html>
- Mosaic Backgrounds  
<http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/Backgrnd/>
- An Instantaneous Introduction to CGI Scripts and HTML Forms  
<http://kufacts.cc.ukans.edu/info/forms/forms-intro.html>
- The Web Browser Torture Test  
  
<http://www.cnet.com/Content/Reviews/Compare/Browsers/Test/index.html>

## Browser Information

- Yahoo's list of Windows Web browsers

**[http://www.yahoo.com/Computers/Internet/World\\_Wide\\_Web/Browsers/](http://www.yahoo.com/Computers/Internet/World_Wide_Web/Browsers/)**

- Microsoft's Internet Explorer

**<http://www.microsoft.com/ie/msie.htm>**

- Netscape

**<http://home.netscape.com>**

- Mosaic

**<http://www.ncsa.uiuc.edu/SDG/Software/Mosaic>**

Here are some additional useful places to find the above documents and learn more about HTML:

**<http://www.gov.nb.ca/hotlist/htmldocs.htm>**

An amazing site with all the HTML documentation you could want!

**<http://www.physics.auburn.edu/html.html>**

Guides and examples

**<http://www.loc.gov/global/html.html>**

Library of Congress site

**<http://www.dai.ed.ac.uk/html/>**

**<http://www.peinet.pe.ca:2080/1/HTML>**

**<http://www.sbcc.cc.ca.us/web/htmlinfo.html>**

**<http://www-star.stanford.edu/instructions.html>**

**<http://www.wsu.edu:8000/isl.www.tech.res.html>**

**<http://www.utirc.utoronto.ca/HTMLdocs/NewHTML/bibliography.html>**

## **How can I get my Web pages onto the World Wide Web?**

In most cases you will need to contact your Internet Service Provider and find out their procedures for placing your pages on the Web.

## How come some of my pictures don't show up?

Chances are that your Web page cannot find your pictures.

WebEdit allows you to place links to your pictures in two different ways:

The first way is called a 'relative reference'. This is when your picture is located in the same directory as your Web page. If you include a link to a picture called "MYPIC.GIF" that is located in the same directory as your page, WebEdit will insert the HTML tag `<IMG SRC="mypic.gif">`

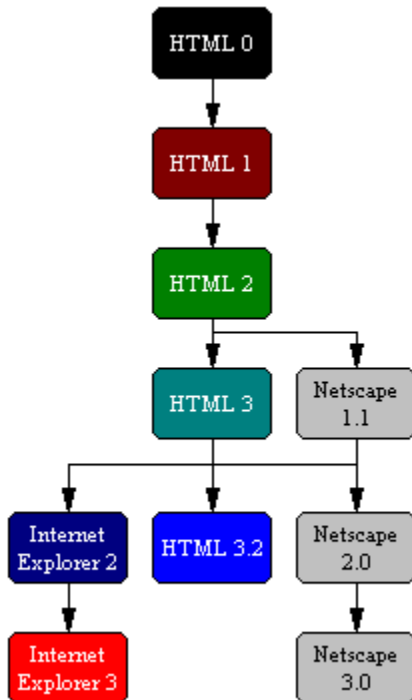
The second approach is called an 'absolute reference'. This is used when your picture is *not* located in the same directory as your page. If you wished to include the picture "MYPIC.GIF" located in the "C:\PICTURES\" directory, WebEdit would insert the HTML tag `<IMG SRC="file:///c:/pictures/mypic.gif">`

Both these methods should work just fine on the computer you are using. Difficulties arise when you place your page on the Web. Chances are that the computer your Web pages will reside on does not have the same directories as your own computer. In this case absolute references will not work properly - you will need to change them all to match the other computers' directory structure.

The simplest way to use pictures is to just keep your pictures in the same directory as your Web page that uses them.

## How come some of the tags I use don't work?

Tags belong to different levels of HTML. Different browsers support different levels of HTML. The trick is to make sure the browser you use supports that level tag. Usually a browser will support all of the tags previous to its' level of HTML.



For example, Netscape 2.0 supports most of HTML 0,1,2,3 and Netscape 1.1 in addition to the new Netscape 2.0 tags, but does not support the Internet Explorer tags..

When you find a tag you want to use. look at its HTML level. If the browser you use is farther down the list than the tag it should support that tag. An example being: Why doesn't the MARQUEE tag work in Netscape 2.0? Well by looking up the tag we see that it is Internet Explorer 2.0 level. Looking at the chart it states that the only two browsers that support the MARQUEE tag are Internet Explorer 2 and Internet Explorer 3.

There are always exceptions to rules - check your browser's documentation to see exactly what tags it supports.



## How can I test my HTML page?

There are several HTML validation services on the Web. These can not only help you spot problems in your documents, but they can also help you learn to write better HTML documents. Here are a few we have seen:

WebLint

**<http://www.unipress.com/cgi-bin/WWWeblint>**

WebTech's HTML Validation Service

**<http://www.webtechs.com/html-val-svc/>**

Gerald Oskoboiny's "Kindler, Gentler Validator

**<http://ugweb.cs.ualberta.ca/~gerald/validate/>**

## Where can I learn more about the Internet?

For those of you who would like to learn more about the Internet and the World Wide Web, including how it works and what resources are available, we strongly recommend the following books:

### **The Internet Complete Reference**

The Internet Complete Reference is one of the most comprehensive and fun-to-read books ever written about the Internet. This book provides thorough and clear explanations of the Net and its various resources, including Usenet, mail, the world wide web, gopher, telnet, wais,archie, etc.

Author: Harley Hahn  
Publisher: Osborne McGraw-Hill  
ISBN: 0-07-882138-X  
Price: US\$32.95

**Note:** Be sure to get the second edition.

### **The Internet Yellow Pages**

If there is a "roadmap" for the Internet, this is it. The Internet Yellow Pages, Second Edition is indispensable when it comes to finding and accessing what's on the Net. This book contains well over 5,000 entries. We strongly encourage everyone who uses the Internet, from beginner to advanced user, to pick up a copy of this book.

Author: Harley Hahn  
Publisher: Osborne McGraw-Hill  
ISBN: 0-07-882192-7  
Price: US\$29.95

**Note:** Be sure to get the third edition.

## **Can I write my own plug-In Modules for WebEdit?**

Yes you can! We have developed an API for programmers to follow that allows their programs to interface with WebEdit. If you write a Module to this API it should work on future versions of WebEdit. To download the latest version of this API go to the WebEdit Home Page <http://www.nesbitt.com/>.

## Do you have any tips for using WebEdit?

Well we have put many, many features into WebEdit - if you blink you might miss some of them!

Have you tried:

- Right-clicking on the toolbars and in dialog boxes like the Custom Tags and Project Manager?
- Alt - Right Arrow and Alt - Left Arrow?
- Shift - Enter and Control - Enter?
- Control - < or > (with more than one document open)?
- Highlighting text and selecting a list tag?
- [Configuring the user toolbar?](#)
- [Dragging and Dropping a file onto WebEdit?](#)



