

HTML

a reference chapter in [Software Engineering for Internet Applications](#); revised May 2003

Hypertext Markup Language, or HTML, is the language used to specify how a browser should display a Web page. HTML is a *markup* language, as opposed to a programming language, meaning that it contains codes that say how a page should be formatted, but does not contain procedural code.

Let's take a look at a simple example:

Code Example	Typical Rendering
<pre><p> Don't look at your instruments and adjust the flight controls to, for example, keep the altimeter steady. The instruments have a tendency to lag behind reality and therefore you're overcorrecting and oscillating. </p></pre>	Don't look at your instruments and adjust the flight controls to, for example, keep the altimeter steady. The instruments have a tendency to lag behind reality and therefore you're overcorrecting and oscillating.

HTML consists of tags, such as `<p>`, interspersed with plain text. The `<p>` tag begins a paragraph; `</p>` ends the paragraph. Similarly, `` starts text emboldening and `` ends it.

Basics

In HTML, almost every opening tag has a closing tag, as in the example above. There are a few exceptions, which we will encounter shortly, but the overwhelming majority of tags must be closed.

Some tags have *attributes*, such as the `face` attribute of the `` tag. Example:

```
<font face=arial>
```

If an attribute value contains a space, it is necessary to enclose it in quotation marks:

```
<font face="arial narrow">
```

Logical Markup

HTML has two kinds of markup: logical markup and physical markup. Physical markup, such as the bold (``) tag specifies how the browser is supposed to render text. In contrast, logical markup, or *semantic tags*, specifies something about the *meaning* of what is being marked up; the browser is free to choose a rendering that is sensible for the user's hardware, e.g., italics might be a good choice on a desktop PC, but reverse video might work better on a low-resolution mobile phone.

Here are a few examples of semantic tags:

Tag	Code Example	Typical Rendering
Emphasis <code></code>	You can fly all day <code>in mid-air</code> without using the airplane's rudder.	You can fly all day <i>in mid-air</i> without using the airplane's rudder.
Strong <code></code>	On short final, press relatively hard on <code>both</code> rudder pedals.	On short final, press relatively hard on both rudder pedals.
Code <code><code></code>	Alaska and Hawaii's airports are identified starting with a <code><code>PA</code></code> for "Pacific".	Alaska and Hawaii's airports are identified starting with a PA for "Pacific".
Headline Level 1 <code><h1></code>	<code><h1>Flight Plan</h1></code>	Flight Plan
Headline Level 2 <code><h2></code>	<code><h2>Flight Plan</h2></code>	Flight Plan
Headline Level 3 <code><h3></code>	<code><h3>Flight Plan</h3></code>	Flight Plan
Headline Level 4 <code><h4></code>	<code><h4>Flight Plan</h4></code>	Flight Plan
Headline Level 5 <code><h5></code>	<code><h5>Flight Plan</h5></code>	Flight Plan

Headline Level 6 <h6>	<h6>Flight Plan</h6>	Flight Plan
-----------------------------	----------------------	-------------

Physical Markup

Here are some common physical markup tags and attributes:

Tag	Code Example	Typical Rendering
Bold 	Use the flight controls to keep the nose of the airplane at a constant attitude relative to the horizon.	Use the flight controls to keep the nose of the airplane at a constant attitude relative to the horizon.
Italics <i>	Have you read <i>Stick and Rudder</i>?	Have you read <i>Stick and Rudder</i> ?
Underline <u>	Flying in the clouds on a summer afternoon, you run the risk of entering an <u>embedded thunderstorm</u>.	Flying in the clouds on a summer afternoon, you run the risk of entering an <u>embedded thunderstorm</u> .
Note: Generally it's best to avoid the <u> tag; underlining should be reserved for hyperlinks.		
Superscript <sup>	Avogadro's number is approximately equal to 6.022 x 10²³	Avogadro's number is approximately equal to 6.022 x 10 ²³
Subscript <sub>	log_ex	log _e x
Font Size 	I want a huge house, a big dog, and a small waist.	I want a huge house, a big dog, and a small waist.
Font Color 	An airplane's navigation lights are green on the right wing and red on the left.	An airplane's navigation lights are green on the right wing and red on the left.
Note: A table of colors and their hexadecimal equivalents is available from http://falco.elte.hu/COMP/HTML/colors.html		
Font Face 	The NASA Aviation Safety Program is the only source of innovation.	The NASA Aviation Safety Program is the only source of innovation.
Typewriter Text <tt>	The terminal forecast called for winds <tt>02015G25KT</tt>, which means from the northeast at 15 knots, gusting to 25 knots.	The terminal forecast called for winds 02015G25KT, which means from the northeast at 15 knots, gusting to 25 knots.
Preformatted Text <pre>	Winds aloft for Buffalo, Boston, and Nantucket, at 3000, 6000, and 9000': <pre> 3000 6000 9000 BUF 0517 0215+01 3306-01 BOS 2218 2325+08 2321+03 ACK 2118 2012+08 1917+03 </pre>	Winds aloft for Buffalo, Boston, and Nantucket, at 3000, 6000, and 9000': 3000 6000 9000 BUF 0517 0215+01 3306-01 BOS 2218 2325+08 2321+03 ACK 2118 2012+08 1917+03
Blockquote <blockquote>	Aviation safety quote: <blockquote> All life is the management of risk, not its elimination. — Walter Wriston, former Chairman of Citibank </blockquote>	Aviation safety quote: All life is the management of risk, not its elimination. — Walter Wriston, former Chairman of Citibank

It's generally considered more tasteful to use logical markup instead of physical markup. It has become especially important now that there is such a wide variety of devices on which to browse Web sites, e.g., mobile phones and handheld devices. A phone might ignore tags, but it will probably try to make headlines (<h1>) stand out.

Hyperlinks

Hyperlinks, often just called *links*, allow the user to jump to a new page or a new location within the same page. Hyperlinks are generally represented by blue, underlined text. Although it is possible to change how hyperlinks appear to the user, we recommended against it; users expect a consistent user interface for Web pages.

An *absolute link* is a hyperlink that specifies the full URL of the destination. Example:

`aviationweather.gov`

Relative links are hyperlinks to documents in relation to the location of the current document. You do not need to specify the server name in the URL. Example:

`Glossary`

embedded in a file in the directory `/seia/` will take a user to the `glossary` file in the same directory. If you're reading this book online, try it out right here: [Glossary](#).

You can make a Web page open up in a new browser window by specifying a target window:

`Glossary`

If there is no browser window named `glossary_window`, a new window will pop up. However, you should use this feature sparingly because the appearance of new windows can be confusing to users. Furthermore, a number of users have pop-up ad blockers installed; these ad blockers will also prevent legitimate windows from popping up. If you're reading this book online, try it out right here: [Glossary](#).

You can also link to specific locations within a document so that your user doesn't have to scroll down to find a particular item on the page. To accomplish this, first you have to mark the location in the document to which you need to link. For example,

`DNS`

Then you can link to that location within the file with:

see the `glossary entry for DNS`

If you're reading this book online, try it out right here: see the [glossary entry for DNS](#). Note that if you want to link to another location within the same file you can omit the file name, e.g., `DNS`.

You will often see a question mark followed by form variables at the end of a URL; this is called the *query string*. For example,

`rec.aviation.student newsgroup`

The variables in this query string are `hl` (headline language?) and `group`. Most Web programming [APIs](#) provide convenient facilities for reading the values of query string variables. If you're reading this online, [try out the link above with its French-language headers](#).

Breaks

All whitespace is treated equally in HTML, meaning that spaces, tabs, and linebreaks are all rendered as single spaces. To force a newline to occur, you need to use a tag.

Here are some common breaks:

Tag	Code Example	Typical Rendering
Paragraph <code><p></code>	<code><p> "I'll be seeing you," he said. </p> <p> Then he walked away. </p></code>	"I'll be seeing you," he said. Then he walked away.
Line Break <code>
</code>	<code>Carson's Plumbing
 123 Main St.
 Seattle, WA 98101</code>	Carson's Plumbing 123 Main St. Seattle, WA 98101
Horizontal Rule <code><hr></code>	<code>And they lived happily ever after. <hr> The End</code>	And they lived happily ever after. <hr/> The End

Notice that `
` and `<hr>` have no closing tags. Additionally, the `</p>` tag is optional; the browser assumes that, when it encounters a new `<p>` tag, the old paragraph has ended.

Lists

The most common types of lists are ordered lists, in which the browser places a number before each list item, and unordered lists, which appear as a series of bulleted items. You can also create definition lists, useful for online dictionaries or glossaries.

Tag	Code Example	Typical Rendering
Ordered List <code></code>	<code>Alaska summer survival gear: rations for each occupant one axe or hatchet</code>	Alaska summer survival gear: 1. rations for each occupant 2. one axe or hatchet

	<pre>one first aid kit Common training airplanes: <ol type=A> Cessna 172 Diamond DA20 Piper Tomahawk Class B VFR Weather Minimums: <ol type=i> 3 statute miles visibility clear of clouds </pre>	<pre>3. one first aid kit Common training airplanes: A. Cessna 172 B. Diamond DA20 C. Piper Tomahawk Class B VFR Weather Minimums: i. 3 statute miles visibility ii. clear of clouds</pre>
Unordered List 	<pre>Checklist for Mexican Flying: proof of airplane ownership proof of liability insurance pilot's license and medical seldom asked-for documents: radio station license radio operator's license border-crossing flight plan </pre>	<pre>Checklist for Mexican Flying: • proof of airplane ownership • proof of liability insurance • pilot's license and medical • seldom asked-for documents: ◦ radio station license ◦ radio operator's license • border-crossing flight plan</pre>
Definition List <dl>	<pre><dl> <dt>IFR <dd>Instrument Flight Rules <dt>VFR <dd>Visual Flight Rules <dt>VOR <dd>Very High Frequency Omni Ranging radio navigation beacon </dl></pre>	<pre>IFR Instrument Flight Rules VFR Visual Flight Rules VOR Very High Frequency Omni Ranging radio navigation beacon</pre>

Images

Images are stored as separate files, not part of the HTML page. An image can be included in a page as follows:





```

```

This tag instructs the user's browser to make a new request, possibly to a different server than the one from which the HTML document was obtained, for the image.

There are many optional attributes for images. The most important are the width and height attributes; by telling the browser the size of the image, it can render the entire Web page, leaving space for the image, before it has downloaded the image file itself.

Attribute	Code Example	Typical Rendering
Dimensions width/ height	<pre></pre>	
Border border	<pre></pre>	
Alignment align	<pre> Canine-American</pre>	Canine-American

		
Alignment align	<code> Canine-American</code>	 Canine-American
Horizontal Space (on both sides) hspace	<code> Canine-American</code>	 Canine-American
Vertical Space (top and bottom) vspace	<code></code>	

Tables

Here are the tags used when creating HTML tables:

<code><table></code> , <code></table></code>	start and end a table
<code><tr></code> , <code></tr></code>	table row
<code><td></code> , <code></td></code>	table cell
<code><th></code> , <code></th></code>	table heading; like a table cell except that the text is bold and centered

Many of these tags can have attributes, e.g. to specify alignment, borders, cell spacing and padding, and background colors. Examples:

Code Example	Typical Rendering												
<pre><table border=2 cellspacing=5 cellpadding=5> <tr> <th>Year</th> <th>Revenue</th> <th>Expenditures</th> <th>Profits</th> </tr> <tr> <td>1999</td> <td>\$58,295</td> <td>\$73,688</td> <td>\$(15,393)</td> </tr> <tr> <td>2000</td> <td>\$902,995</td> <td>\$145,400</td> <td>\$757,595</td> </tr> </table></pre>	<table border="1"> <thead> <tr> <th>Year</th> <th>Revenue</th> <th>Expenditures</th> <th>Profits</th> </tr> </thead> <tbody> <tr> <td>1999</td> <td>\$58,295</td> <td>\$73,688</td> <td>\$(15,393)</td> </tr> <tr> <td>2000</td> <td>\$902,995</td> <td>\$145,400</td> <td>\$757,595</td> </tr> </tbody> </table>	Year	Revenue	Expenditures	Profits	1999	\$58,295	\$73,688	\$(15,393)	2000	\$902,995	\$145,400	\$757,595
Year	Revenue	Expenditures	Profits										
1999	\$58,295	\$73,688	\$(15,393)										
2000	\$902,995	\$145,400	\$757,595										
<pre><!-- reduce the cellspacing and right-align the text in the cells --> <table border=2 cellspacing=2 cellpadding=5> <tr> <th>Year</th> <th>Revenue</th> <th>Expenditures</th> <th>Profits</th> </tr> <tr></pre>	<table border="1"> <thead> <tr> <th>Year</th> <th>Revenue</th> <th>Expenditures</th> <th>Profits</th> </tr> </thead> <tbody> <tr> <td>1999</td> <td>\$58,295</td> <td>\$73,688</td> <td>\$(15,393)</td> </tr> <tr> <td>2000</td> <td>\$902,995</td> <td>\$145,400</td> <td>\$757,595</td> </tr> </tbody> </table>	Year	Revenue	Expenditures	Profits	1999	\$58,295	\$73,688	\$(15,393)	2000	\$902,995	\$145,400	\$757,595
Year	Revenue	Expenditures	Profits										
1999	\$58,295	\$73,688	\$(15,393)										
2000	\$902,995	\$145,400	\$757,595										

<pre> <td>1999</td> <td align=right> \$58,295</td> <td align=right> \$73,688</td> <td align=right> \$(15,393)</td> </tr> <tr> <td>2000</td> <td align=right> \$902,995</td> <td align=right> \$145,400</td> <td align=right> \$757,595</td> </tr> </table> </pre>													
<pre> <!-- remove the border --> <table border=0 cellspacing=2 cellpadding=5> <tr> <th>Year</th> <th>Revenue</th> <th>Expenditures</th> <th>Profits</th> </tr> <tr> <td>1999</td> <td>\$58,295</td> <td>\$73,688</td> <td>\$(15,393)</td> </tr> <tr> <td>2000</td> <td>\$902,995</td> <td>\$145,400</td> <td>\$757,595</td> </tr> </table> </pre>	<table border="1"> <thead> <tr> <th>Year</th> <th>Revenue</th> <th>Expenditures</th> <th>Profits</th> </tr> </thead> <tbody> <tr> <td>1999</td> <td>\$58,295</td> <td>\$73,688</td> <td>\$(15,393)</td> </tr> <tr> <td>2000</td> <td>\$902,995</td> <td>\$145,400</td> <td>\$757,595</td> </tr> </tbody> </table>	Year	Revenue	Expenditures	Profits	1999	\$58,295	\$73,688	\$(15,393)	2000	\$902,995	\$145,400	\$757,595
Year	Revenue	Expenditures	Profits										
1999	\$58,295	\$73,688	\$(15,393)										
2000	\$902,995	\$145,400	\$757,595										
<pre> <!-- shade every other row --> <table border=0 cellspacing=2 cellpadding=5> <tr bgcolor="#cecece"> <th>Year</th> <th>Revenue</th> <th>Expenditures</th> <th>Profits</th> </tr> <tr bgcolor=white> <td>1999</td> <td>\$58,295</td> <td>\$73,688</td> <td>\$(15,393)</td> </tr> <tr bgcolor="#cecece"> <td>2000</td> <td>\$902,995</td> <td>\$145,400</td> <td>\$757,595</td> </tr> </table> </pre>	<table border="1"> <thead> <tr> <th>Year</th> <th>Revenue</th> <th>Expenditures</th> <th>Profits</th> </tr> </thead> <tbody> <tr> <td>1999</td> <td>\$58,295</td> <td>\$73,688</td> <td>\$(15,393)</td> </tr> <tr> <td>2000</td> <td>\$902,995</td> <td>\$145,400</td> <td>\$757,595</td> </tr> </tbody> </table>	Year	Revenue	Expenditures	Profits	1999	\$58,295	\$73,688	\$(15,393)	2000	\$902,995	\$145,400	\$757,595
Year	Revenue	Expenditures	Profits										
1999	\$58,295	\$73,688	\$(15,393)										
2000	\$902,995	\$145,400	\$757,595										

Forms

To collect data from users, use the form tag:

```
<form method=POST action=/register/new>
```

The action is the URL to which the form is submitted, which may correspond to a computer program in the server file system, e.g., a Java Server Page, a PHP or Perl script, etc.

The form's method can be either GET or POST. The only difference is that, with method=GET, the variables that the user submits are presented in the query string of the following page's URL. This is useful if you want the user to be able to bookmark the resulting page. However, if the user is expected to enter long strings of data, method=POST is more appropriate because some old browsers only handle query strings containing fewer than 256 characters (newer browsers can handle a few thousand). Note further that if you use the GET method, the form variable values will appear in the server access log and could create a security or privacy risk.

Code Example

```
<form method=POST action=/survey/demographic>
<input type=hidden name=user_id value=2205>
Age: <input type=text size=2><br>
Sex: <input type=radio name=sex value=m>male
     <input type=radio name=sex value=f>female<br>
What are you interested in (check all that apply)?
<input type=checkbox name=interest value="aerobatics">Aerobatics
<input type=checkbox name=interest value="helicopters">Helicopters
<input type=checkbox name=interest value="IFR">IFR
<input type=checkbox name=interest value="seaplanes">Seaplanes
<br>
Where do you live?
<select name=continent_live>
  <option value=north_america>North America
  <option value=south_america>South America
  <option value=africa>Africa
  <option value=europe>Europe
  <option value=asia>Asia
  <option value=australia>Australia
</select>
<br>
Which continents have you visited?<br>
<select multiple size=3 name=continent_visited>
  <option value=north_america>North America
  <option value=south_america>South America
  <option value=africa>Africa
  <option value=europe>Europe
  <option value=asia>Asia
  <option value=australia>Australia
</select>
<br>
Describe your favorite airplane trip:<br>
<textarea name=favorite_trip_story rows=5 cols=50></textarea>
<p>
<input type=submit value="Continue">
</form>
```

Typical Rendering

Age:

Sex: male female

What are you interested in (check all that apply)? Aerobatics Helicopters IFR Seaplanes

Where do you live?

Which continents have you visited?

- North America
- South America
- Africa
- Europe

Describe your favorite airplane trip:

Special Characters

A wide variety of non-alphanumeric characters can be specified in HTML. Here is a small sampling:

Entity	Code Example	Typical Rendering
n, tilde ñ	piñata	piñata
e, acute accent é	café	café
inverted question mark ¿	¿Qué pasa?	¿Qué pasa?
non-breaking space 	a b	a b
greater-than >	4 > 3	4 > 3

less-than <	5 < 6	5 < 6
copyright ©	© 2004	© 2004
pound sterling £	£50	£50

A more complete special character reference can be found at http://webmonkey.wired.com/webmonkey/reference/special_characters/.

HTML Document Structure

Up to this point, we have looked at individual tags within an HTML document. But what is the overall structure of an HTML document?

HTML documents are broken into two main sections: the *head* and the *body*. The head contains information pertaining to the entire document (most importantly, the document's title). The body contains the content of the page that appears within the browser window. Here is a basic HTML document:

```
<html>
<head>
  <title>This is the Title</title>
</head>
<body>
  ... This is the content of the page. ...
</body>
</html>
```

News pages often include instructions that the browser refetch the page. Here's a tag, located in the head, from news.google.com:

```
<meta HTTP-EQUIV="refresh" CONTENT="900">
```

If you load this page into a browser and step back from the computer, you should notice it updating itself every 900 seconds (15 minutes).

Also within the head, you can specify keywords and a description of the page. These tags were originally intended to help search engines index pages, but now they are often ignored due to abuse such as page authors using incorrect keywords to get more hits.

```
<META NAME="description" CONTENT="An owner's review of the Diamond Star DA40">
<META NAME="keywords" CONTENT="Diamond Star DA40 review Cirrus SR20 SR22">
```

You can modify properties of the Web page by using <body> tag attributes. For example:

```
<body bgcolor=white text=black link=blue vlink=purple alink=red>
```

However, you should use this sparingly; users are accustomed to the standard text colors and may become frustrated if they can't tell what's a link and what isn't.

Cascading Style Sheets

Ever since the development of the Web there has been a tension between people who focus on content and those who are more interested in presentation. The content people want to get relevant information on every page, possibly marking up a phrase with the H3 tag to say "this is a headline". The presentation folks say things like "move this two pixels to the right", "stick this in 18-point Helvetica Bold and make it red", and "stick this in 14-point Times Italic". They use tricks such as blank images for spacing and tags such as font and color.

Here are some of the problems with filling up site content and scripts with tags like font and color:

- Older browsers will ignore them; the latest and greatest tags tend to have been introduced with the latest and greatest browsers; H3 and EM, however, are understood by browsers going back to the early 1990s.
- Newer browsers will ignore them; mobile phones, palmtops, and hiptops often have very basic browsers that understand only the basic tags.
- When your service hires a new graphic designer, the programmers will have to edit 10,000 HTML documents and thousands of scripts.

A site-wide cascading style sheet addresses all of these issues. Here's part of the cascading style sheet for the online version of this book (<http://philip.greenspun.com/seia/style-sheet.css>):

```
body {margin-left: 10% ; margin-right: 10%}

P { margin-top: 0.3em; text-indent : 2em }

P.stb { margin-top: 12pt }
P.mtb { margin-top: 24pt; text-indent : 0in}
```



```

P.ltb { margin-top: 36pt; text-indent : 0in}

p.marginnote { background: #E0E0E0;
               text-indent: 0in ; padding-left: 5%; padding-right: 5%; padding-top: 3pt;
               font-size: 75%}
p.bodynote { background-color: #E0E0E0 }
...

```

Each line of the style sheet gives formatting instructions for one HTML element and/or a subclass of an HTML element. The body tag is augmented so that all of the pages will have extra left and right whitespace margins. The next directive, for the P tag, tells browsers not to separate paragraphs with a full blank line but rather to indent the first line of a new paragraph by "2em" and add only a smidgen of blank vertical space ("margin-top: 0.3em"). Now paragraphs will be pushed together like those in a printed book or magazine. Books and magazines do sometimes use whitespace, however, mostly to show thematic breaks in the text. We therefore define three classes of thematic breaks and tell browsers how to render them. The first, "stb" (for "small thematic break") will insert 12 points of white space. A paragraph of class "stb" will inherit the 2em first-line indent of the regular P element. For medium and large thematic breaks, more whitespace is specified, as well as an override for the first-line indent.

How does one use a style sheet? Park it somewhere on the server in a file with the extension ".css". This extension will tell the Web server program to MIME-type it "text/css". Inside each document that uses the cascading style sheet, put the following link element inside the document head:

```
<LINK REL=STYLESHEET HREF="/seia/style-sheet.css" TYPE="text/css">
```

The first time the user's browser sees a page that references this style sheet, it will come back and request "http://philip.greenspun.com/seia/style-sheet.css" before rendering any of the page. Note that this will slow down page viewing a bit, although if all of my pages refer to the same site-wide style sheet, users' browsers should be smart enough to cache it. If you read ten chapters from this book online, for example, the browser should request the common style sheet only once.

Okay, now the browser knows where to get the style sheet and that a small thematic break should be rendered with an extra bit of whitespace. How do we tell the browser that a particular paragraph is "of class stb"? Instead of "<P>", we use

```
<P CLASS="stb">
```

An excellent CSS reference can be found at <http://www.w3schools.com/css/default.asp>.

Frames

Frames consist of independent windows within a single Web page. Usually each window can be scrolled separately. Often, when you click a link, only one frame is updated with a new URL; the rest of the page content stays the same.

Frames sounded like a good idea at the time (mid-1990s), but have proven to be painful for both users and developers for the following reasons:

- *Frames waste screen space.* Often frames have their own scrollbars, which take up valuable space within the browser window. Furthermore, if you are only interested in one frame and you scroll down within that frame, the other frames remain in place, leaving less space for the content you want.
- *Frames make it difficult to bookmark pages.* When the user follows links that only update one frame, the URL of the page does not change. Suppose Joe User visits a travel site, follows five links within frames to get to a page about a tour of Mexico's Copper Canyon, and then bookmarks that page; the bookmark will point to the front page of the travel site, not the Copper Canyon page.
- *Frames make it difficult to share pages.* Suppose Joe User wants to see if his friend is interested in going on the Copper Canyon tour. While looking at the tour advertisement, he cuts and pastes the URL from the browser's Address field into an email message. Joe's friend clicks on the URL and gets the travel site home page, not the interior page about the Copper Canyon tour.
- *Frames make it difficult to report errors.* Consider a frame-using site with 200 scripts. A user isn't happy with the way a page works. You ask her "What's the URL of the broken script?" She looks in her browser's Address field and gives you the URL of the site's front page.
- *Frames make scrolling more difficult.* Experienced users know that you don't have to use a mouse to scroll through a Web page; you can use the space bar or arrow keys. However, if the page uses frames, the user must first click on the frame in which they wish to scroll.
- *Frames break the Reload button.* In our hypothetical travel site, if Joe User pushes Reload when looking at the Copper Canyon page, the browser will often show the travel site home page, because the URL has not been updated.
- *Frames break the Back button.* In some browsers, frames break the Back button; if a user visits a frame-based site and clicks 100 times on interior links, a click on the

Back button may take the user back 101 steps rather than 1.

- *Search engines send users to subpages.* Suppose your site uses two frames: one for navigation and one for content. Since each frame is defined by a separate HTML document at a separate URL, a public search engine such as Google is most likely to send the user to the HTML document containing content only. That user will never see the navigation frame and therefore won't be able to find the other parts of your site.

HTML Considered Harmful?

Vanilla HTML imposes limits on how you can display and collect information. Users can't drag and drop objects. There are no sliders, no paintbrushes, no real-time direct manipulation of screen objects. You can get around these limitations with a Java applet, Flash, or code targeted at another browser plugin, but it might not be a good idea.

Part of the genius of HTML and the Web is that all sites using HTML markup and forms work the same. A user who has learned to use amazon.com can apply his or her experience to using Google. Users visit a Web site because they are looking for unique content and services, not a unique interface.

Administration pages may constitute an exception to the "custom interface is bad" rule. Suppose you hire and train customer service agents who will be using the administration pages on a daily basis. If a Macintosh/Windows-like drag-and-drop interface saves them a lot of time (and you money), it is perfectly reasonable to write custom code that will run in their browsers. You may have to spend fifteen minutes training each agent, an unacceptably long time for a casual user, but the long-run productivity dividends make it worthwhile.

The Future

In the practical world, HTML is king. In the conference rooms of standards committees, however, it has been superseded by Extensible Hypertext Markup Language (XHTML). Should you wish to keep up with events in this area, visit <http://www.w3.org/MarkUp/>.

More

- visit your favorite Web page and use the browser command "View Source"
- HTML tag reference: http://www.w3schools.com/html/html_reference.asp (Web) and [*HTML & XHTML: The Definitive Guide*](#) by Musciano and Kennedy (O'Reilly, 2002; print)
- Colors and their hexadecimal equivalents: <http://falco.elte.hu/COMP/HTML/colors.html>
- Special characters: http://webmonkey.wired.com/webmonkey/reference/special_characters/
- Cascading Style Sheets: <http://www.w3schools.com/css/default.asp>

Return to [Table of Contents](#)

eve@eveandersson.com, philg@mit.edu, aegrumet@mit.edu