

# HTTP - HTML

---

## Network Effect Issues

Recall discussion of network effects and public standards, e.g. TCP/IP  
The HTTP/HTML standards dominate their niche in this same way  
There are other niches -- video standards, instant messaging, online shopping  
identify -- where the same sorts of battles have yet to play out. Vendors would  
prefer to "own" such standards if they can make that happen, or a public  
standard may come to dominate.

## HTTP

Tim Berners-Lee working at CERN -- a document distribution protocol.  
Hypertext Transfer Protocol.

"Hypertext" was an well established notion of having links to a document  
embedded in other documents. See the "Xanadu" project -- [www.xanadu.net](http://www.xanadu.net) --  
(1960) an example of being so ambitious, that the more ordinary  
implementation (HTTP) just passes you by. Network effect then makes the  
widespread (HTTP) dominate, regardless of technical merit.

From the original HTTP justification draft  
(<http://www.w3.org/Protocols/WhyHTTP.html>) ...

Why a new protocol?

Existing protocols cover a number of different tasks.

- \* Mail protocols allow the transfer of transient messages from a single author to a small number of recipients, at the request of the author.
- \* File transfer protocols allow the transfer of data at the request of either the sender or receiver, but allow little processing of the data at the responding side.
- \* News protocols allow the broadcast of transient data to a wide audience.
- \* Search and Retrieve protocols allow index searches to be made, and allow document access. Few exist: Z39.50 is one and could be extended for our needs.

The protocol we need for information access (HTTP) must provide

- \* A subset of the file transfer functionality
- \* The ability to request an index search
- \* Automatic format negotiation.
- \* The ability to refer the client to another server

---

Tim Berners-Lee (1991)

See...

<http://www.w3.org/Protocols/>

<http://www.w3.org/Protocols/rfc2616/rfc2616.txt>

## --HTTP Overview...

### HTTP Server

Listens on Port 80  
Gets HTTP requests, sends back responses

### Doc Root on Server

Usually, the HTTP server is looking at tree of documents in the file system...  
Suppose `/users/nick/WWW/` is the root of the HTTP document tree in the file system

There are subdirectories `"/users/nick/WWW/a/"` `"/users/nick/WWW/b/"` and `"/users/nick/WWW/c/"`, and each of those contains an "x.html" file.  
Incoming requests looking like `/a/x.html` and `/b/x.html` map to the analogous file in the file system.

### URL

`http://www.stanford.edu/class/cs193i/index.html`

Protocol

`http`

Host

`www.stanford.edu`

Path

`/class/cs193i/index.html`

### Client

Has Url --> makes request

### Request

HTTP: request from client to server

The client looks at the URL, contacts that host, and sends a short request that includes the path

### Response

HTTP: response from server back to client

Using doc root, maps the path into the local file system...

`/class/cs193i/index.html --> /usr/class/cs193i/WWW/index.html`

`/a/x.html --> /users/nick/a/x.html`

The server reads that file and sends it back to the client as the HTTP response

### Server Special URL Mapping

More complex mapping cases -- the server can choose to interpret things like `~nick` in special ways.

Usernames -- the server may have a rule by which it maps `~username` or some other pattern into a directory in the file system...

`/~nick/a/about.html --> /users/nick/WWW/a/about.html`

/class/cs193i/about.html --> /class/cs193i/WWW/about.html

### Directories

For url that identifies a directory (a url ending in a /) -- what HTML should the server return?

Usually, the directory can be mapped to some default file in that directory, such as index.html or default.html...

/class/cs193i/ --> /class/cs193i/WWW/index.html

### Directory listing

Some servers can be set so that for the directory case, they generate, on the fly, a directory listing HTML and send that back.

### Server control

HTTP does not specify how these mappings happen. Whoever sets up the server can program in the mappings however they like. HTTP servers, such as apache, are very configurable.

Certain mappings, such as directory->index.html, are defacto standards.

## Stateless -- Connection not kept open

By default, the HTTP server closes the connection after the response is sent. So the client/server protocol is not an extended dialog. It's more "one shot" with a single request/single response/close.

This will make it harder to do some things on the client side, but it keeps the HTTP server simpler.

## HTML on the Client

Most often the HTTP response is HTML

The HTML is designed to be **portable** -- it should be possible for many different types of client to display it.. (Most web designers produce only semi-portable HTML, but that's another lecture).

The browser displays the HTML as best it can. The HTML includes URL links that, when clicked, lead to further HTTP transactions.

## HTML

Hypertext Markup Language

### Portable Content

Represent textual and image content -- not exact appearance. Better than plain ASCII, not as rich in appearance as Postscript or PDF.

Encode text and image "page" content in a way that can be displayed on a wide variety of platforms

Page is made of text content, some layout information, images, and URL links

### Unsolved

That may sound like a modest goal, but it hadn't really been solved up to that time (this is part of the "Markets vs. Standards" theme)

### Different Platforms

Different OSes, different window/screen sizes, different installed fonts that render at different sizes.

Users who have particular preferences on what font to use.

# References

## HTML Primer

<http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>

## W3C

World Wide Web Consortium [W3C] -- the non-profit which tries to maintain a portable definition of HTML and other Internet related standards...

<http://www.w3.org/MarkUp/>

## Setting up a page in leland space

<http://www.stanford.edu/leland/howto.shtml>

## View Source

To learn how a page is written, use the "View Source" option in your browser to look at the HTML source code. This is a quick way to learn.

# Basic Document Structure

## Text

An HTML file is just a text file. The HTTP server sends it over to the client for display in the client browser.

## Structure vs. Appearance

**HTML defines the basic structure of a document, but the exact appearance is determined by the browser dynamically.**

## Text and tags.

Tags enclosed in <>'s. Some tags occur by themselves e.g. <br>, others are paired to enclose some text <h1>...</h1>. Upper and lower case tags act the same.

The HTML itself is plain text — no bold, no fonts, no rulers, etc. Just a sequence of characters. You can edit the HTML text in a word-processor or text editor. Just be sure to save it as "plain text" and not some other format.

## Whitespace

"Whitespace" (spaces, tabs, end-of-lines, ...) is largely ignored.

# What Defines Appearance?

Appearance in the browser is determined by..

What fonts and sizes the user has chosen in their browser preferences.

The current width of their browser window.

## Pour

The HTML "pours into" this structure set by the user.

Try changing the width of your window — the content defined by the HTML should rearrange to present itself in the window.

Properly written HTML will look reasonable on many different platforms -- different browsers, and even on things like Palm Pilots (essentially just a very narrow window).

See <http://cslibrary.stanford.edu/104/> for an example of a page with many elements that "pours" into the browser page -- try varying the window width and font sizing to see how it adjusts.

## Bad HTML (a pet peeve of mine)

Smart HTML authors understand that they control the structure, but the browser controls the appearance. HTML authors who do not "get" this basic

feature, are doomed to (a) be frustrated, (b) spend 10x too long producing their documents, and (c) their documents will only look right on the platform/browser combination the author happened to use.

## Basic Tags

`<html>...</html>`

Encloses the entire HTML source.

`<head>...</head>`

Encloses the "header" section of an HTML document. The header section most often just contains the document title, but it can contain other meta information about the document. Such meta information may be better developed in the future. Currently, the Alta Vista meta content tags which aid the indexing of the document go in the header section. See the Alta Vista help page.

`<meta>` tags -- used inside the head section to specify information about the document (these are from the page <http://cslibrary.stanford.edu/104/>)

`<meta name="description" content="Stanford CS Education Library: a fun 3 minute video that explains the basics features of pointers.">`

`<meta name="keywords" content="pointer, pointee, reference, pointer fun, basic pointer, pointer introduction, video, animation, animated">`

`<title>...</title>`

The title phrase for this page. Polite to always have one. Shows up as the title of the browser window, although that's a little to subtle to be useful. Also used by search engines, bookmarks, etc. as a one phrase name for this page. I frequently have use the same phrase in the `<title>` and a top of page `<h1>`.

`<body>...</body>`

The body is basically all of the document but the header.

`<h1>Large Header</h1>`

`<h2>Smaller Header</h2>`

`<p>`

Marks the beginning of a block of text which should stick together as single paragraphs. In the browser, each paragraph will be set off from its surroundings by some vertical whitespace.

*All of the text* after the `<p>` until another paragraph or header marker will be gathered up into a single paragraph. Blank lines etc. do not terminate the paragraph. The `</p>` is not required, but it may be someday.

`<ul>..</ul>`

Unordered list. Each item (below) shows up with a bullet. There's an ordered list `<ol>..</ol>` variant where the items are numbered.

`<li>`

Each item in the list begins with an `<li>`. The items themselves behave pretty much like little paragraphs. There's no `</li>` tag, the beginning of one list marks the end of the previous.

## Miscellaneous Adornment Tags

`<br>`

Force a line break (aka a "return") in the appearance of the text. Does not introduce extra vertical whitespace...Use `<p>` if you want a line break and vertical whitespace.

`<hr>`

Horizontal Rule. Draws a horizontal line. Good cheap way to provide visual separation of logically separate elements.

## Physical Styles

Indicate character appearance. These have won out over logical styles mostly

`<b>...</b>`

Bold.

`<i>...</i>`

Italic.

`<blink>...</blink>`

Regarded as quite-vulgar. One problem is: how do you print it on paper? I heard that Marc Andreesson admitted to being drunk when he added this feature to Mosaic.

## Logical Styles

Define the logical role of the text. Also causes the text to have a distinguishing appearance in the browser. The exact appearance is up to the browser.

`<em>...</em>`

Emphasis. Makes the text stand out in some way. (Often italic)

`<strong>...</strong>`

Strong emphasis. Make the text stand out more so. (Often bold)

`<cite>...</cite>`

Encloses the name of something, book, movie, etc. Someday, this information may be used to build more intelligent cross-references of the web.

`<address>...</address>`

A postal or email address or reference. Typically shows up on its own line.

# Special Characters

&lt; -- "<"

&gt; == ">"

&amp; -- "&"

&nbsp; -- a space

## Example

```
<html>
<head>
<title> HTML Basic </title>
</head>

<body bgcolor=white>

<!--
  This is what a comment looks like
-->

<h1> HTML Basic </h1>

<h2> Slightly Smaller Header </h2>

<p> Here's some body text. HTML runs all this text together,
basically ignoring spaces, tabs, and blank lines until it gets to
a tag which

tells it to
be something other
than a

body text paragraph.

<p> Another body text paragraph. Each of these is typically separated
from whatever precedes it by a little vertical whitespace.

<h3>&lt;br&gt;</h3>

<p>Use the &lt;br&gt;<br>
tag to force a linebreak. Try not to use this -- it tends to produce less portable
results
since it stops accounting for the window width in use.

<h3>&lt;hr&gt;</h3>

Use &lt;hr&gt;<hr>
To introduce a horizontal line to divide sections.
```

```
<h2>Lists!</h2>
```

```
<p>
And now some exciting foods...
<ul>
<li>Bannana -- a yellow fruit</li>
<li>Apple -- a red fruit
<li>Potato -- a starchy tuber
</ul>
```

```
<h2>Physical Character Styles</h2>
<ul>
<li><b>Bold</b>
<li><i>Italic</i>
<li><tt>Fixed Width</tt>
</ul>
```

```
<h2>Logical Character Styles</h2>
<ul>
<li><cite>Cite</cite>
<li><address>Italic</address>
</ul>
```

```
<h2>Tables</h2>
```

```
<table border=1>
<tr>
  <th>Fruit</th>
  <th>Feature</th>
</tr>

<tr>
  <td>Bannana</td><td>Squishy yellowness</td>
</tr>

<tr>
  <td>Apricot</td><td>Dried out orangey-ness</td>
</tr>

<tr>
  <td>Mystery Fruit with a long name</td>
  <td>Mysteriousness -- note how this
  left aligns with the earlier features. Tables are the portable way
  to get things to line up.</td>
</tr>
</td>

</table>

</body>
</html>
```

The screenshot shows the Netscape browser window titled "Netscape: HTML Basic". The address bar shows the file path: `file:///NULL/Classes/193i%2099-3/HTML%20demo/basic.html`. The browser toolbar includes buttons for Back, Forward, Reload, Home, Search, Netscape, Images, Print, Security, Shop, and Stop. Below the toolbar are navigation links for WebMail, Contact, People, Yellow Pages, Download, and Find Sites.

# HTML Basic

## Slightly Smaller Header

Here's some body text. HTML runs all this text together, basically ignoring spaces, tabs, and blank lines until it gets to a tag which tells it to be something other than a body text paragraph.

Another body text paragraph. Each of these is typically separated from whatever preceeds it by a little vertical whitespace.

**<br>**

Use the `<br>` tag to force a linebreak. Try not to use this -- it tends to produce less portable results since it stops accounting for the window width in use.

**<hr>**

Use `<hr>`

---

To introduce a horizontal line to divide sections.

## Lists!

And now some exciting foods...

- Bannana -- a yellow fruit
- Apple -- a red fruit
- Potato -- a starchy tuber

## Physical Character Styles

- **Bold**
- *Italic*
- Fixed Width

## Logical Character Styles

- *Cite*
- *Italic*

## Tables

Fruit	Feature
Bannana	Squishy yellowness
Apricot	Dried out orangey-ness
Mystery Fruit with a long	Mysteriousness -- note how this left aligns with the earlier features. Tables are

# URLs

## Url

Uniform Resource Locator -- specify the location where a document may be retrieved There's research into URN Uniform Resource Name which identifies a document reliably no matter where it is stored. URNs are not currently really used, but it's a clear future direction.

<http://www.stanford.edu/class/cs193i/>

Scheme -- http

Host -- www.stanford.edu

"Path" -- /class/cs193i/

Later, we'll divide the path in to sub-parts

`<a href = "url to link to">underlined anchor text</a>`

There can be other foo="yadda yadda" bindings in the `<a>` tag -- don't assume the href is the only one. Also, the quotes may be omitted if the url contains only ordinary characters

The anchor tag can make any element in the document (words in paragraphs, header, lists, pictures ...) a "hyperlink" to another document. In the browser, the link will show up with a distinctive appearance -- typically blue underlining.

E.G.

`<a href = "http://www.stanford.edu/class/cs193i/">CS193i</a>`

Shows up as an underlined CS193i which links to the class page when clicked.

Blue Underline

IMHO, it's stupid to override that default appearance, since even the most naive user has learned to associate the blue underline with the "link" idea. If pages use different appearances for linking, then everything begins to look potentially clickable which is huge a step backwards in usability for the web.

Usability: clickable things should look clickable

Style

Don't put too much text in the HREF— it's confusing to look at once the line breaks. Just anchor around the key phrases.

Do name it what it is: check out the [document archive](#), or surf over to our horrific [Summer Vacation Photos](#).

Some people think it's bad style to use phrases like [click here](#). Some use phrasing like "see our large complicated and difficult to explain with just a few words site [here](#)." I prefer to anchor the description itself if at all possible.

% Encoding

Whitespace and most other non [a-zA-Z0-9] characters are encoded in the URL as a '%' followed by the character's ASCII code in hex e.g. a space character within URL is encoded as "%20" since hex 20 = 32 decimal which is the ASCII code for a space.

W3C addressing

<http://www.w3.org/Addressing/>

## Page relative URL lookup

Suppose you are within the CS193i page who's full URL is...

<http://www.stanford.edu/class/cs193i/index.html>

The URL of an enclosing page is sometimes known as the "base" URL.

URLs within the page are evaluated relative to its base URL. That means that if a URL is incomplete, the system will assume any missing components are the same as the base URL.

Effectively, this just means that URLs within a page are assumed to point to things which are "near" the referring page.

Gives you the latitude to use short, relative URLs within a page.

## Relative Example

Suppose we are writing HTML which is at

<http://www.stanford.edu/class/cs193i/index.html>

The URL of the document where the `<a href = >` is embedded is the "base" URL.

Relative

`<a href = "binky.html">Binky!</a>`

The short, relative reference uses the protocol, host, and front part of the path of the base URL, so expands to the full URL...

<http://www.stanford.edu/class/cs193i/binky.html>

foo.html

So if you have a document called foo.html in the same directory as your home page, you can refer to it just as "foo.html" in your home page and count on the relative lookup to find it.

Extra For Experts

Relative addressing allows you to develop a group of interlinked documents in one place using relative references to refer to each other. Later you can move all the documents somewhere else, and it all continues to work even though the document's full URLs are all different.

You can use `../foo.html` to go up a directory -- this is a "fringe" use of HTML that few authors use (or understand).

## Root Relative

A relative url beginning with a '/' uses the protocol and host, but no part of the original path. So it essentially goes up to the document root of the same server.

`/images/jeffsad.jpeg`

e.g. `<a href = "/images/jeffsad.jpeg">` refers to an images directory in the document root of the same server as the base URL.

The full URL would be <http://www.stanford.edu/images/jeffsad.jpeg>

## Mailto URL Scheme

A URL which when clicked, opens a window to edit a message to be sent to the given address.

`<a href="mailto:nick@cs.stanford.edu">send email</a>`

## Name Tags

`<a name = spot>` marks a particular location in a document.

A URL can refer to that place in the document as `<a href = document.html#spot>` or with a relative reference within the same document just `<a href = #spot>`.

The `<a>` tag may contain both an `href=` and a `name=`.