Adding more: HTML stage II

HTML has other structures apart from paragraphs and section heads. In this part, we're going to look at lists, links and visual effects.

Lists

HTML defines three main kinds of list: *ordered*, *unordered* and *discursive*.

Ordered Lists

This is an ordered list:

1. with a *list item* 1,
2. an item 2,
3. an item 3 and so on

The numbering is automatic, so if you delete or insert items, they get renumbered automatically by the client (browser) when the file is next displayed. An ordered list like this is written as:

```
<ol>
<li>with an item 1,</li>
<li>an item 2,</li>
<li>an item 3 and so on</li>
</ol>
```

Notice that the whole ordered list is enclosed in `<ol>`...`</ol>` tags and that each list item within the list goes in `<li>`...`</li>` tags. You can have a list within a list, but the numbering/lettering scheme is up to the browser (client): you can't affect how it numbers or letters the items.

Unordered lists

An unordered list has bullets instead of numbers: you can use this for lists where the sequence of items is not important:

- 2lb (4 cups) sugar
- 8oz (8 squares) dark chocolate
- 1/2pt (8 fl.oz) evaporated milk
- 2oz (1 stick) butter

(although the end result might be...) In this case, the tags to use to surround the whole list are `<ul>`...`</ul>`:

```
<ul>
<li>2lb (4 cups) sugar</li>
<li>8oz (8 squares) dark chocolate</li>
<li>1/2pt (8 fl.oz) evaporated milk</li>
<li>2oz (1 stick) butter</li>
</ul>
```

but you still use `<li>`...`</li>` tags for the list items. This makes it easy to change the kind of list later: just swap `<ul>`s for `<ol>`s.

A list can go inside a paragraph, or between paragraphs. In theory you can also have a paragraph inside a list (*ie* between two list items):

```
<ul>
<li>Here's the first item</li>
<p>And a second paragraph belonging to the same item, so there's no
bullet or number on it.</p>
<li>Then comes another item</li>
</ul>
```

produces

- Here's the first item

And a second paragraph belonging to the same item, so there's no bullet or number on it.

- Then comes another item

but there is no guarantee that every browser will format this sensibly. In cases where you want extra paragraphs, use discursive lists

Ordered and unordered lists can have an *attribute* of `compact` which most browsers interpret as meaning the list items will be short, so the space between items is reduced. Attributes go inside the angle brackets after the tag name: `<ul compact>` for example, would produce

- First item
- second
- third

Discursive lists

Discursive lists are designed to handle cases where each list item needs some paragraphs of explanation. The whole list is surrounded with `<dl>`...`</dl>` tags, and each item is given in `<dt>`...`</dt>` tags (for *discursive topic*). Each `<dt>`...`</dt>` can then be followed by one or more paragraphs, each surrounded with `<dd>`...`</dd>` (*discursive discussion*) tags, so

Managerial grades
: Senior and junior managers are expected to dress in conventional blue suits during office hours, especially while meeting clients.
: Any departure from this norm will be taken very seriously.

Salaried staff below managerial level
: Employees must dress neatly but need not wear suits. Unusual or extravagant clothing is inappropriate to the position of these employees within the company.
: When meeting clients, suits *must* be worn.

Programmers and HTML hacks
: Plebs like these can wear anything they want, your ever–loving and paternalistic company couldn't care less,
: In any case, we depend on them so much to keep our systems running that we'll make any exception they insist on.

can be obtained with

```
<dl>
<dt>Managerial grades</dt>
<dd>Senior and junior managers are expected to dress in
conventional blue suits during office hours, especially
while meeting clients.</dd>
<dd>Any departure from this norm will be taken very
seriously.</dd>
<dt>Salaried staff below managerial level</dt>
<dd>Employees must dress neatly but need not wear suits.
Unusual or extravagant clothing is inappropriate to the
position of these employees within the company.</dd>
<dd>When meeting clients, suits <em>must</em> be worn.</dd>
<dt>Programmers and HTML hacks</dt>
<dd>Plebs like these can wear anything they want, your
ever–loving and paternalistic company couldn't care less,</dd>
<dd>In any case, we depend on them so much to keep our systems
running that we'll make any exception they insist on.</dd>
</dl>
```

The original HTML also defined `<menu>`...`</menu>` *and* `<dir>`...`</dir>` *lists, but these will be dropped in the frozen version as they have the same effect as* `<ul>`. *They may be resurrected in HTML+ with slightly different interpretations.*

Links

Links are the slick bit about W3's hypertext. Click on a highlighted word or phrase and it retrieves a file related to the word or phrase. BUT...you as the author have to give it the right information about what file to retrieve, but it can be almost anything which is available from almost any server on the Internet. Not just another W3 hypertext file, but information available through

Gopher, WAIS, Usenet `news`, anonymous `ftp`, `telnet`, `finger` and other more specific services.

The hypertext link tag is called *anchor* (abbreviated to `<a...>`) because it can act both as a jumping−off point and as a target or another link to aim at. The file or service specification you give in quotes is called the *Uniform Resource Locator* or URL for short.

To point at another file

surround the text you want to highlight with the `<a href="...">` tag in the following form:

```
<a href="scheme://host/dir/subdir/filename">text</a>
```

where the URL in quotes is made up of:

- *scheme* is one of:
  - `http` − for W3 hypertext files (filenames end in `.html`)
  - `file` − for non−HTML files (plain text, for example)
  - `gopher` − for the Gopher directory−browser
  - `news` − for Usenet `news`
  - `telnet` − for login to services on remote machines
  - `ftp` − for file retrieval using anonymous `ftp` from public archives

  Note that the scheme name is followed by a colon and double slash;
- *host* is a regular Internet hostname, usually in the form *machine.site.net* like `info.cern.ch` (but you can give a numeric 'dotted−quad' IP address if you wish, like `143.239.1.8`). If the service you want to refer to runs on a port other than 80 (http), you must follow the hostname with a colon and the port number (like `lambda.parc.xerox.com:8888`);
- *dir/subdir/filename* is the full file specification for the file you want to point at. If you omit the file specification, the server at *host* should display their top−level entry−point file;

for example

```
<a href="http://info.cern.ch/hypertext/WWW/TheProject.html">WorldWideWeb</a>
<a href="gopher://sunic.sunet.se/">Gopher</a>
<a href="telnet://lamda.parc.xerox.com:8888">Telnet</a>
```

There are a few exceptions and additions:

- `news` just needs the colon and the name of the newsgroup, and it will use the reader's locally−defined server to perform the retrieval, for example `news:comp.text.sgml`
- With Gopher, WAIS and HTTP server scripts, you can follow the filename directly with a question mark, and this will be taken as the keyword to search for, for example `"http://curia.ucc.ie/htbin/acronym?ibm"` will run the acronym search service looking for 'ibm';
- With HTTP files, you can follow the filename with a # sign and the name of an anchor tag in that document that you want to jump straight to (rather than displaying the start of the file), for example `"http://curia.ucc.ie/info/net/htmlmore.html#links"` takes you to the beginning of this section, like this;

Server scripts are a feature of some HTTP servers, and allow a URL to point at a program to be executed, rather than a file to be retrieved. The script or program must produce only valid HTML output, so that the results it generates can be displayed in the browser screen. Fuller details of server scripts are in the documentation for the relevant servers, *eg* the NCSA's.

There are some other attributes defined in HTML for the anchor tag which are mostly unused: `rel`, `rev`, `urn`, `title`, `methods`.

To use an anchor tag as a *destination*

Use the form of the tag:

```
<a name="label">
```

where *label* is a name that you make up, which you will use after the # sign in a cross−reference from somewhere else (see last list item in the previous subsection).

You can combine both in the same tag:

```
<a name="intro" href="http://abc.edu/menu.html#widgets">...</a>
```

which makes that text a link *to* the tag labelled ‘widgets’ in the menu file on abc.edu and also labels the same text as ‘intro’ if you want to refer to it *from* elsewhere.

Visual effects

We’ve seen a lot of stuff about structure: here’s how to add changes to visual appearance like different type styles. Bear in mind that these may only be obvious if you have a graphical client (browser) like Mosaic, Cello or Emacs which can use different type styles. Plain−terminals cannot, so browsers like Lynx and WWW simulate the changes by using underscores or asterisks around the text.

Emphasis

There are two main kinds of emphasis:

- emphasizing a word or phrase is done with the ... tag like this to get *this*;
- stronger emphasis can be done with the ... tag like this to get **this**.
- *The original HTML also defined* <u> *for underlining, but this is not widely used.*

Exactly how the emphases are displayed is a function of the browser program. It may be italics and bold, or it may be colours or a different typeface: what’s important is the degree of emphasis. If you absolutely positively *have* to get italics or bold, you can use <i>...</i> to get *italics* and ... to get **bold**.

Fixed−width type

There are several ways of showing fixed−width (typewriter) type, useful if your files refer to computer documentation, or if you need aligned columns (the original HTML has no specification for tables: HTML+ may introduce this). Don’t forget that users on plain terminals only get typewriter−style letters anyway, so they won’t be able to see any difference.

- *Fragments of computer code, email addresses and machine names* in the text of a paragraph are done with <code>...</code> to get fixed−width characters like ‘cd /; rm −rf *’, but if you want to include the less−than and greater−than characters (as in these examples), you have to use the character entities < and > − how to actually display the ‘<’ itself is left as an exercise to the reader :−)
- *Longer portions of code and examples of usage* are done with the <xmp>...</xmp> tags to get fixed−width characters in the same way, but on a line to themselves (remember that linebreaks normally get swallowed up as if they were spaces). This lets you quote

  ```
  cd /; rm −rf *
  ```

  without having to start a new paragraph. With <xmp> you can include the less−than and greater−than characters in this tag without having to use < and > like

  ```
  \ifnum#1>\prevarg\&\else\message{Invalid data}\fi
  ```

- *Small−scale examples of preformatted fixed−width text* (like a program example, or data in columns) can be handled with the <pre>...</pre> tags. Here, you can have many lines and the linebreaks and blank lines *will* be honored:

  ```
  Results of the 2.30 at Fairyhouse

  1st       WorldWideWeb         2/1 on fav
  2nd       Gopher’s Delight     100/1
  3rd       Acrobat              500/1

  7 ran, Word Perfect (6) fell at the first fence.
  ```

  This also allows font changes like italics and bold to be honored within the preformatted display, so you do need to use `<` and `>` if you want less−thans and greater−thans;

- *Finally, large−scale listings or whole files* can be displayed with `<listing>...</listing>`:

```
<!-- inc href="http://curia.ucc.ie/info/net/demo.txt"-->
```

  You can either include the listing text verbatim inside the `<listing>...</listing>` tags, or use HTML's mechanism for including external files, which is explained in Part III. Angle brackets are honored as well as linebreaks and all spacing, so the file can contain any characters you wish;
- For keyboard examples you can use `<kbd>...</kbd>` like `this`, and for individual keys there is `<key>...</key>` to produce this (these may not be implemented in all browsers).

If none of these suits your purpose, single words and short phrases can use `<tt>...</tt>` to get `teletype` letters.

Assorted effects

There are a few new tags, undefined in HTML, which are useful and will likely be included in HTML+. These are empty tags − they don't have matching end−tags:

If you find you have to force a line−break at a specific point in your text, there is an undeclared tag `<br>` to do this.
There was one just after the word 'this' just now. It will likely be defined properly in HTML+.

You can get a horizontal rule

---

across the screen by using `<hr>`

A non−breaking space is useful to prevent a browser making a linebreak at a particular point: insert`<nbsp>`between two words, leaving no other space.

The original HTML also defines the following four tags:

- `<samp>...</samp>` to identify sample text like `this` (it may look like typewriter text);
- `<var>...</var>` for variants like *this*;
- `<dfn>...</dfn>` to identify definitions the first time they occur; and
- `<cite>...</cite>` to deal with short citations like *Goldfarb C, 'The SGML Handbook', OUP 1990, 0−19−853737−9*.

Not all of these necessarily cause a visual change in all browsers: some are for identifying important elements in your text which you might need to find or refer to in other ways.

And now . . .

By now you should be able to enhance your simple document with lists, links and changes in appearance. In the next part we'll look at the remaining features of HTML like block text, some things not yet implemented like mathematics and tables, and then go on to the inclusion of files and images and a new feature of HTML+, form−filling.