Active Web Pages

Chris Lilley
http://info.mcc.ac.uk/CGU/staff/lilley/lilley.html

Manchester and North HPC Training & Education Centre
The University of Manchester

Abstract

It is easy to start producing simple, static HTML pages, but these may offer little advantage over their Gopher predecessors. The term 'Active Web Pages' is used here to indicate content that is dynamic, responsive, sensitive to reader input and accessible to a large audience – including low bandwidth and print disabled readers – while providing high quality material as added value for those who can use it. The present work gives an overview of how to become a provider of Active Web Pages. The tutorial notes should be consulted for more detailed information.

1 Portable document design

Portable documents are ones which are usable by the majority of target users, and which will continue to be usable over the projected lifetime of the information. Documents which can only be used on a particular platform, or with a particular browser, are of limited use unless you know that your entire audience have that browser or that platform (for example, a company wide weblet isolated behind a firewall). With an unrestricted world wide web, you can *never* make that assumption.

1.1 Text may replace images

Not all users can display images. They may be using a text terminal; they may be paying by the minute for their networking – one of the major ways to save networking costs is to not use images – or they may be visually disabled.

In most cases, you will not want to omit all images from your pages. Rather, you need to ensure that the pages are still *usable* without the images. The images should add extra, optional value for those who are able to use them. (If they do not, then they should of course be deleted!). Preserve the highest quality for those who are able to make use of it, but do not penalise those who cannot.

HTML 2.0 specifies an attribute of the IMG tag, ALT, which is a text string to be displayed in place of the image.

```
<IMG SRC="http://site/path/prothrombin.gif"
ALT="prothrombin molecule">
```

Note that this is a simple text string, it *cannot* contain HTML markup. This is because the IMG tag is not a container.A suitable text string to replace an image depends on what the image is, and it's function on the page. If it is merely decorative, or forms part of a corporate identity, replace it with nothing. This is not the same as having no ALT text at all – it means writing ATL="".

Beware adding explanatory text that adds nothing. In all cases, when you have written a page, test it out by using a linemode browser. Does the combination of alt text and HTML text flow together? If not, change it. Do any of your images contain text? Is it essential to understand the meaning?

1.2 Intelligent chunking

Low-bandwidth users can be helped by sending them nothing they do not need to read. Consider a document with three chapters, each with ten sections, each with five short paragraphs. In a paper document, all this would be in one chunk – the document. As an online document, there are various ways in which it could be organised ranging from one large file to 150 (3 * 10 * 5) tiny ones.

Documents converted from wordprocessors are often all one big file. You have to download the whole lot even if you only want to read one section. It is better to break the document into smaller pieces, often called 'chunks'.

A sensible chunking strategy would be to have each section in a separate file, linked to a table of contents for each chapter, and an overall hyper-index for the whole document. Given appropriate navigational aids such as these, there is no ease-of-use penalty for a fine chunk size, and the benefits in download time can be considerable. Consulting the document might mean downloading, say, the index, chapter one section 4, and chapter 3 section 8.

On the other hand, an *extremely small* chunk size - such as a paragraph - would increase the delay when reading the document. Typically a user would need to read several consecutive paragraphs

1.3 Use of graphics

There is no way to know, when selecting a link to an HTML file, how many inline images it has and how big they are. Older non-threaded browsers have to download the HTML, then all the images, before displaying anything.

Inline images can theoretically be of any size. There is no way to limit the size of inline image that you are prepared to wait for. (Actually the HTTP standard does have a parameter, mxbytes, which could be used to do this, but no current browser uses it). It is therefore helpful to readers with lower bandwidth connections to exercise moderation with inline images. No hard and fast rules can be given, but consider replacing large inline images with a smaller inline thumbnail image which is the anchor for a link to the large image. This gives the reader the option of loading the large image or not. It also gives a freer choice of image format for the large image. Often, the large image is a true colour JPEG file and the inline image is a GIF. Provided the browser uses an external program to view external images (not all do), having an image in a separate window while scrolling the relevant text can be handy.

All current browsers that display inline images can handle GIF87, and the vast majority can handle GIF89 (which allows a colour to be declared transparent, permitting apparently non-rectangular inline images; GIF87-only clients will display a colour instead of transparent, but will not fail). Most browsers can display XBM inline, but there is no good reason to use this format, GIF is often smaller. Increasingly, browsers can display JPEG JFIF inline (Mac users note: this is not the same thing as a PICT JPEG; both formats are colloquially known as 'a JPEG file' on their respective platforms). Future browsers will probably be able to display PNG inline. Developed specifically as an inline image format for the Web, PNG supports either indexed or truecolour images, has internal compression, and good support for transparency. The most portable choice for an inline image is currently GIF89, perhaps with white as a transparent background colour. This is because (in March 95) the major browser not to support GIF89 is NCSA Mosaic for Mac, and Mac browsers generally have whiter backgrounds!

External images can theoretically be in any format, although selecting something obscure such as Amiga IFF is likely to limit the audience who can view it. In practice, JPEG JFIF is a suitable choice for photographic material and 'soft' computer graphics such as radiosity and Monte-Carlo rendering. Baseline TIFF 5.0 is a good choice for external true colour images for which JPEG JFIF would be inappropriate, such as sharp computer graphics or material where the numerical comparison of images is required.

1.4 Warn the reader

To avoid surprises such as a half hour delay to download a document, it is helpful to warn the reader where links lead to very large files. One way is by placing the file sizes by big links.

> *The report is available as a text file (8k), in HTML, or in PostScript (176k)*

Compression schemes can reduce the size of files and thus the time they take to transfer over the network. Some formats have internal compression schemes for some or all of the data. GIF and JPEG image file formats include internal compression; TIFF images may optionally use internal compression.

1.5 Browser independence

A common beginners mistake is to design for the browser they are using, and assume it will look the same on everyone else's screen. This can make the document browser specific; an undesirable condition varying in severity from a slightly sub-optimal layout to the complete inability of any other browser to display the document.

There are a wide variety of user interface styles for browsers. A mouse-driven point and click system with multiple overlapping windows is common but not universal. Link selection need not be with a mouse – it may be by typing in a number, using the cursor keys or function keys, circling the link with a pen, using an audio-typist style foot pedal. So do not say '*click*', say '*select*' which covers all selection methods.

Incidentally, saying ‘click here’ is doubly bad; not only does it assume mouse selection, it also gives a poor clue about the nature of the link. Link text should be meaningful; a good way to skim an HTML document is to just read the link text. A page full of hilighted ‘*here*’ ‘*here*’ ‘*here*’ is much less accessible than a page with ‘*technical report*’ ‘*abstract*’ and ‘*introduction*’.

Many browsers support experimental features which may be revised, dropped, or folded into the standard at a later date. For example ViolaWWW supports inline PostScript documents and embedded client-side programs. While it is interesting to experiment with these sort of features, they should not be used for documents which are intended to have a general readership.

1.6 Standard HTML

It is wise to ensure that documents are written in standard HTML 2.0. This can be checked with tools such as *html-check*, which validate a document against the HTML DTD. All you have to do is declare at the start of the document:

```
<!DOCTYPE HTML PUBLIC  "-//IETF//DTD HTML Strict Level 2//EN">
```

This particular declaration will not only check your document is legal, it will also warn about deprecated features – old or experimental parts of HTML that are still legal for that level of HTML, but may not be legal in future levels. There is no guarantee that a deprecated feature will be supported in a later revision of the standard. Thus, as soon as a feature is known to be deprecated, for maximum portability you should not use it. In general this is not a problem, as some of these features are pretty obscure!

When dealing with non-standard HTML, some browsers will ignore entities that they do not understand; others will ignore the tags but print the textual part of the entity. Yet other browsers may print unknown tags, display rubbish on the screen, omit vital information, or simply display an error message. So you cannot be sure that using non-standard HTML will not break any browsers.

Many of the extensions provided by Netscape are also available in HTML Level 3, although they may be achieved in different ways. Also, Netscape is starting to support the HTML Level 3 entities too. For example, this will work in any HTML 3.0 browser (and also works with Netscape):

```
<H1 align="center">A centred heading</H1>
```

And this alternative method will only work with Netscape:

```
<CENTER><H1>A centred heading</H1></CENTER>
```

If you want to experiment with HTML Level 3 features such as equations and tables you should declare your document thus

```
<!DOCTYPE HTML PUBLIC  "-//IETF//DTD HTML Level 3//EN">
```

In addition, to fully warn older browsers that ignore DOCTYPE declarations, it is recommended that HTML Level 3 files are served with a MIME type of `text/x-html3` rather than `text/html`.

1.7 Large print

The partially sighted are often able to read text on a computer screen, provided it is large enough. They may need to use a pointing device to indicate where in the text they are, and to initiate sideways scrolling which is frequently required. So far so good. Many GUI browsers provide customisable or selectable text sizing. Imagine making the text of your browser as big as it will go, and then some - eight or more times bigger than you would normally use. Clearer now? What has happened to the graphics?

The problem is that many browsers assume a one to one mapping between image pixels and output device pixels. This can give problems with the extremes of device resolution – a graphic that looked fine on a postcard sized, 512x384 Mac Classic can look tiny on a 21’ screen, 1280x1024 workstation. Some graphics formats, such as TIFF, include the intended display size in device-independent, real-world units, which would (if notice were taken of the information) at least allow the same file to be shown at the same size on all browsers. This would still not help the print disabled, unless both text and graphics could be resized together.

Both the `FIG` and `IMG` entities in HTML 3.0 take optional x and y sizes. Currently there are two units of measurement supported: pixels and ems. An em is a typesetting unit (roughly equivalent to the width of a lowercase letter ‘m’) whose size depends on the current font. If the picture size is specified in ems, when you blow up the text the figures also get bigger.

1.8 Braille

Braille is a means of reading textual documents by touch. One Braille character consists of an array with three rows and two columns; in each position there is either a dot or not. This simple cell can rapidly be recognised by a skilled Braillist; a larger cell size would provide more character combinations but would be slower and more error prone to read. There are thus 64 (2^6) combinations, barely enough to cover letters and numbers let alone special characters. Braille cells are used in combinations to get around this. For example, one character means that the following cells are numbers instead of letters

There is an SGML document type (DTD) called ICADD (International Committee for Accessible Document Design) which is explicitly designed to make documents available to the print disabled. It is designed to preserve as much of the information about headings, lists, tables and so on; it should not generate the equivalent of a plain text file. An ICADD document can be converted to large print, speech, or Braille. Automatic techniques exist to convert documents from one DTD to another, once the mapping between the various entities in the two documents has been designed and implemented. Such a mapping exists for conforming HTML level 2.0 documents and ICADD documents.

Thus, provided you have written your HTML document correctly, it can be automatically translated from HTML level 2.0 to ICADD and then to Grade II Braille, suitable for printing on a Braille printer or reading with an electronic Braille display. All without the author of the original HTML document having to know anything about Braille - all the hard work has already been done by the designers of the DTDs and conversion software.

2 Active images

An active image responds to positional input from the reader. Only a single point can be selected, not an area.The first known example of a page with an active image was the World Map server at XEROX Paolo Alto Research Centre (PARC). It was created in June 1993 by a researcher called Steve Putz, as an experiment in interactive information retrieval. It subsequently won two categories in the first Best of the Web competition – *Most Technical Merit* and *Best Use of Interaction*. Since then, standardised interfaces (CGI), handlers (NCSA *imagemap* and W3O *htimage*) and tools (*mapedit*) have made the process quite painless.

2.1 Writing the HTML

Here is a fragment that produces a fixed link:

```
<A HREF="http://site/path/fixed.html">
<IMG SRC="http://site/path/foo.gif ALT="text"></A>
```

and here is one that produces an active link

```
<A HREF="http://site/path/active/image">
<IMG SRC="http://site/path/foo.gif ALT="text" ISMAP></A>
```

The obvious change is the new attribute for the `IMG` tag – `ISMAP`. This tells browsers that are able to, that the pixel coordinates of the selected point should be passed back to the given URL as a search query. Browsers that can display images but are unable to determine a selected point, such as text browsers that display all inline images as external links, will select the link with no search string.

In either case there is only a single anchor, which points to a program on the server side that decodes the selected point and causes the browser to load the relevant URL. The program is generally activated using the Common Gateway Interface, but for now, we will just look at what you have to do to create a page with an active image, not how it is implemented.

The URL for the link which causes a program to be run on the server is constructed like this. The protocol, site (and port number, if not the default) are the same as with ordinary links. The path is a path with which the server can tell you want to run a program. Then there is the name of the active image handling program – in this case, it is called ‘active’. The last part of the URL is the image identifier – a way to identify which active image has been selected, as the same program is used to deal with all of them. This is simply a name that you specify, in this example it is ‘image’.

2.2 Defining the links

The active image handler must map the selected point to one of a finite list of URLs which the browser will then load. It would clearly be laborious and inefficient to define a URL for each and every point in the image. Generally, geometric regions (circles, rectangles, irregular polygons) are defined in pixel coordinates and the decoding program determines which if any of these has been selected. It also helps to define a default URL which will be returned if none of the shapes has been selected.

The process does vary slightly depending on which program is used to do the decoding. The two most popular programs used for this task are the `htimage` program provided with the CERN/W3O server and the imagemap program provided with the NCSA server. Both of these programs require a map file to be written which contains the definitions of these shapes and the corresponding URL for each shape. The decoding program uses the image identifier to find out the name of this file from a master list of all known active images. Here is such a file as the CERN/W3O program *htimage* would like it:

```
default http://nowhere/
#Carbon one
circle (158,175) 17 http://site/path/green-carbon.html
#big blue one
circle (346,185) 23 http://some.site/path/blue-one.html
#The entire molecule
poly (168,35) (132,52) (124,83) (91,114) (51,126) (24,153)
(20,211) (35,244) (62,268) (94,274) (130,252) (164,269) (192,272)
(213,262) (234,237) (240,212) (253,205) (262,192) (273,209)
(282,236) (309,261) (335,270) (364,266) (387,256) (413,232)
(426,209) (431,183) (429,164) (423,140) (401,118) (372,99)
(368,89) (366,59) (347,32) (322,19) (294,19) (280,24) http://
other.site/path/whole-thing.html
```

This defines three shapes; two circles and a polygon. The circles are described by their centre (x,y) and radius. The polygon is described by the position of each vertex.

2.3 Designing and testing an active map

It should be apparent that finding and typing in all these pixel coordinates will be tedious and error prone. It also makes it less likely that, if a shape is slightly the wrong size or a little in the wrong place, it will be corrected. Clearly a program should be doing this work. Such programs exist for the major Unix platforms and for MS-Windows, at least. They allow the shapes to be fine tuned and tested, and will write out a suitable map file.

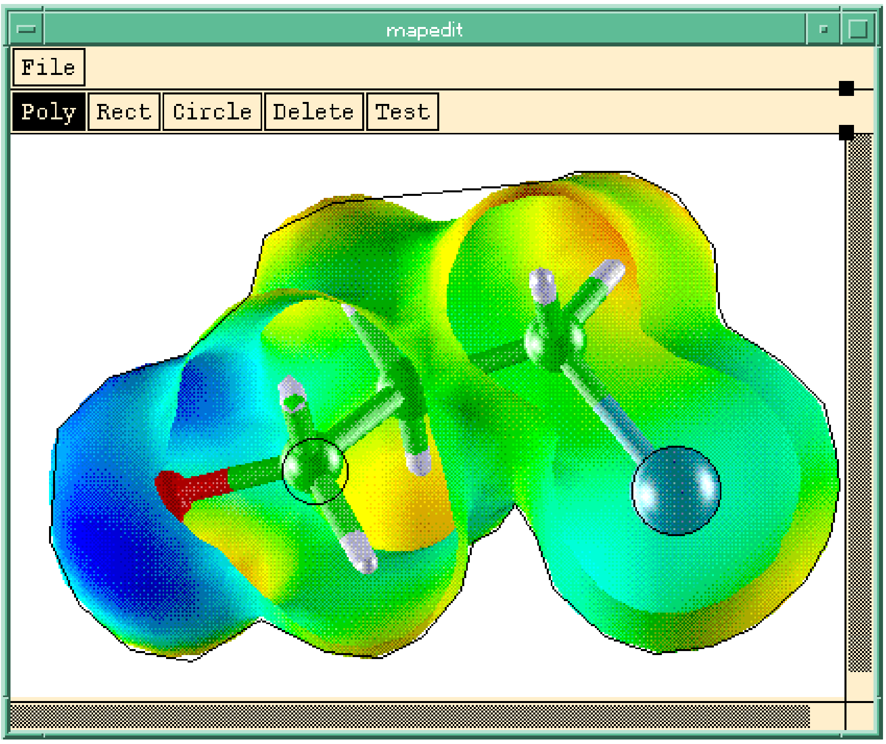


Figure 1: Mapedit, an active image development tool.

In the example above, the image shows a cutaway view of a molecule. The large polygon defines the whole molecule, so that any selection that does not match it must have selected the background. That being the case, how can the two circles be selected?

The first matching shape is used, so if you have overlapping or enclosed shapes put the smaller ones earlier in the file. It is useful to think of the shapes being stacked one on top of another, with the first shape in the file on top.

To test an active map, use the map editor to ensure that you have accurate shapes in the right place and that they are stacked in the right order. Check you have a default URL for selections that miss. Lastly, save out the map file, register the map file with the server master list (see your server documentation about this) and perform a final test with a Web browser. Ensure, or course, that the inline image is the same image that you used to generate the map file!

2.4 Portability

Remember that not all clients can display imagemaps. At present, the best solution is to include a link to a text-only page that contains a list of all the URL's that the imagemap references, with suitable explanatory text for each one. For the future, active images in HTML 3.0 may be handled either on the client side or the server side. Server side active images will be portable between HTML 2.0 and HTML 3.0.

3 Forms

A form is an HTML entity for accepting input from the reader of a document. In most cases this is then sent off to a program on a Web server for processing. To use forms you must do two things: construct the form, and write a program to process the results. Unlike active images, where a standard program could be used to handle all of them, each form generally requires a different program.

3.1 The `FORM` tag

There can be more than one form in a document, but as with all other HTML entities they cannot overlap. Neither can forms be nested inside other forms. The FORM tag is a container. It is used like this:

```
<FORM METHOD=POST ACTION="http://site/path/program/etc/">
the form contents go here
</FORM>
```

The `METHOD` can be either GET or POST. In most cases POST is the best choice; GET suffers from a restriction in the amount of data that can be sent to the form handler.If no method is supplied, the default is GET. The `ACTION` URL is how the form processing program is invoked, so the protocol part of the URL is always HTTP unless you have some rather special requirements.

3.2 Input fields

In addition to all the usual stuff that can go in the body of a document – headings, paragraphs, lists, and so on – a `FORM` tag can also contain three new entities: `INPUT`, `SELECT` and `TEX-`

TAREA. It does not have to contain any of them, but is rather useless without at least one!

The INPUT tag is used for several quite different input types, so it has TYPE attribute which describes what sort of input object is required. There are eight different types of INPUT entity:

- TEXT, for typing in a line of text
- PASSWORD for typing in text which is not shown on screen
- CHECKBOX, for selecting zero, one, or more items from several options
- RADIO, for selecting a single item from several options
- SUBMIT, for sending the form data to the ACTION URL
- IMAGE, for indicating a position on an image *and* sending the form data
- RESET, for setting the form back in its default state
- HIDDEN, for input that is not shown to the reader.

The NAME attribute is how we distinguish between the various form elements. The SELECT and TEXTAREA entities need one, and so do all INPUT entities, apart from SUBMIT and RESET (although they usually have a NAME too). The following example uses several of these input types:

```
<FORM METHOD=POST ACTION="http://site/path/program/etc/">
<INPUT TYPE=TEXT NAME="UserName" LENGTH="12">
<INPUT TYPE=PASSWORD NAME="UserPass" LENGTH="12" MAXLENGTH="12"
VALUE="Guest" >
<INPUT TYPE=CHECKBOX NAME="Journal" VALUE="CG">
Computers and Graphics<BR>
<INPUT TYPE=CHECKBOX NAME="Journal" VALUE="TOG">
Transactions on Graphics<BR>
<INPUT TYPE=CHECKBOX NAME="Journal" VALUE="Forum" CHECKED>
Computer Graphics Forum</P>
<INPUT TYPE=RADIO NAME="Who" VALUE="UG" CHECKED>Undergraduate<BR>
<INPUT TYPE=RADIO NAME="Who" VALUE="PG">Postgraduate<BR>
<INPUT TYPE=RADIO NAME="Who" VALUE="ST">Staff</P>
<INPUT TYPE=IMAGE NAME="Thing" SRC="thing.gif" ALIGN=TOP>
<INPUT TYPE=HIDDEN NAME="User" VALUE="Joe">
<INPUT TYPE=HIDDEN NAME="Piccy.x" VALUE="88">
<INPUT TYPE=HIDDEN NAME="Piccy.y" VALUE="132">
<INPUT TYPE=SUBMIT VALUE="Do it">
<INPUT TYPE=RESET VALUE="Clear form">
</FORM>
```

The SELECT tag is used as a handy alternative to CHECKBOX when there are lots of choices and you want to keep the form compact. The SELECT element is often rendered as a a scrolling list, or sometimes as a pop-up or pull-down menu. You can specify how many items are visible at a time. SELECT is a container, and you put OPTION elements into it to represent the alternatives. SELECT has the NAME attribute, and all the OPTION elements in it take that name.

```
<SELECT NAME="Issue">
<OPTION>March 1993
```

```
<OPTION>April 1993
and so on
<OPTION> May 1995
<OPTION SELECTED>June 1995
</SELECT>
```

Lastly, the `TEXTAREA` tag is used for multi-line text entry fields. It is a container, and anything inside it is taken as the default contents.

```
<textarea name="Comments" rows="3" cols="30">
My comment about your service is:

</textarea>
```

It is not currently possible to upload the contents of external files, short of pasting them into a `TEXTAREA` field. This problem is being addressed by several researchers, and is the subject of an Internet Draft..

4 Server-side programs

The earliest server side programs, such as the Xerox World Map server, were run with special custom servers, separate from the main server which was used solely for document delivery. Later, server developers created easier ways to trigger the server top run on external programs. These methods were still very server specific. For example the Plexus server, which is written in the interpreted language Perl, could accept other Perl modules and start them running with particular URLs. This made it hard to move such programs between servers.

4.1 CGI

The Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers. Gateways are programs that serve data which is not directly readable by a client program, such as a database of protein structures held in PDB format, by converting it to an acceptable forms, such as an HTML page, a PostScript file, some images, or a combination of these. Gateways typically serve a subset of the data they are interfacing to; selection of this subset is typically by an HTML form

Gateways can be run by themselves, but are designed to be run by the http server. The server sets up an environment, shown in Figure 2.

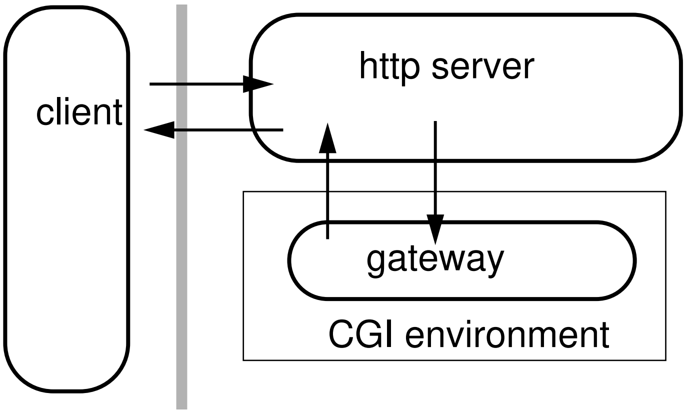


Figure 2: Communication between client, server, and CGI gateway.

4.2 Talking to the CGI program

When a program is run standalone, it is common practice to use the command line to pass extra information. In the case of CGI programs, this is not usually possible. Instead, CGI programs pick up their information from the CGI environment variables.

Query strings are generated by forms that use GET, and by `ISINDEX` text entry fields. Both of these generate a URL of the form:

```
http://site/path/file?some-query
```

The environment variable `QUERY_STRIN`G would be set to some-query in this instance. Another way to pass information to a CGI program, often used when one program is dealing with output from several forms, is to put extra information in the path of the URL:

```
<p>Looking at the
<a href="http://site/cgi-bin/building/elevation=front">
front</a> of the building</p>
```

This will tell my program to generate a front view. The same program could be re-used in different parts of the document to generate other views, simply by changing the extra information:

```
<p>Now we see the
<a href="http://site/cgi-bin/building/elevation=side/
which=left">
left side</a> of the building</p>
```

This extra information is put into the `PATH_INFO` CGI environment variable. In the first instance this would contain the string

```
/elevation=front
```

Other environment variables can be queried to find out the server name, the access method (GET or PUT), or the script name (handy if several soft links all point to the same program).

Forms that use POST send an object to the sever, currently with the MIME type `applica-`

`tion/x-www-form-urlencoded`. A CGI program can check the `CONTENT_TYPE` variable to check it has been sent data from a form; it can find the size of the POSTed object from the `CONTENT_LENGTH` variable and read that many bytes from the standard input.

4.3 Generating output

Anything a CGI program writes to standard output is sent back to the client by the server. Anything it writes to standard error goes in the server error log. Once the CGI program terminates, the server computes the length of the output, generates the headers, and sends the output.

The server will parse the output of the CGI program looking for headers that trigger special actions. It will in general pass on any additional headers which may be generated. A blank line should be produced by the CGI program to separate any headers it produces from the object body. There is one header that must always be produced unless there is no object body: the MIME type of the body.

Many servers will also parse the object body, and compute a content length which is inserted into the header. This saves the CGI program from having to do a lot of housekeeping. So a CGI program that is generating an HTML response need only write:

```
Content-type: text/html

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html><head><Title> and so on
```

to standard output, and the server will do the rest. Of course the output need not be HTML; it can be anything, such as an image, a CGM, a PostScript file, or a sound file, for example. As HTTP is an 8 bit clean transport, the application simply writes the binary data to standard output.

Lastly, it is possible that you will want to output one of a fixed set of existing HTML files. This is readily done by simply generating a string like this:

```
Location: http://some.site/path/samples/Tester.html
```

The server will generate a redirection error, and the browser will automatically load the specified URL.

5 HTML 3.0

The currently implemented level of HTML is 2.0. The specification of HTML 3.0 is, at the time of writing (early 1995) fairly stable and there is a reference implementation in the shape of *Arena*, from W3 Organisation. It will be useful, then, to briefly look at major differences from HTML 2.0 and how HTML 3.0 is likely to alter the way documents are designed and presented.

5.1 Maths

There is support for SGML maths. which is fairly simple to translate to and from existing systems such as LaTeX. It is designed to allow WYSIWYG editing and export to symbolic maths systems such as Maple and Mathematica. The following fragment:

```
<math>y=sum_i=1_^n^{1<over>1+x_i_}</math>
```

would display something like this:

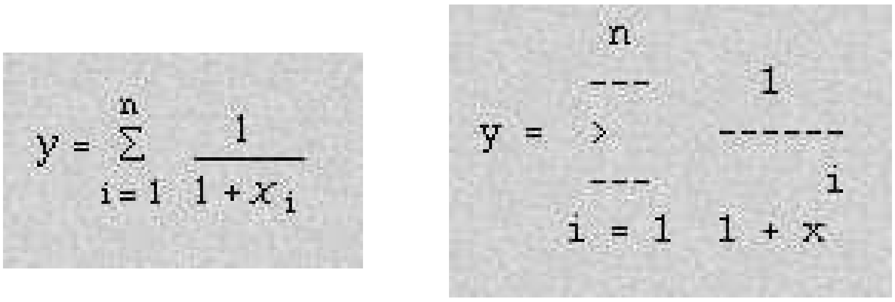


Figure 3: Maths displayed on graphical (left) and text (right) browsers.

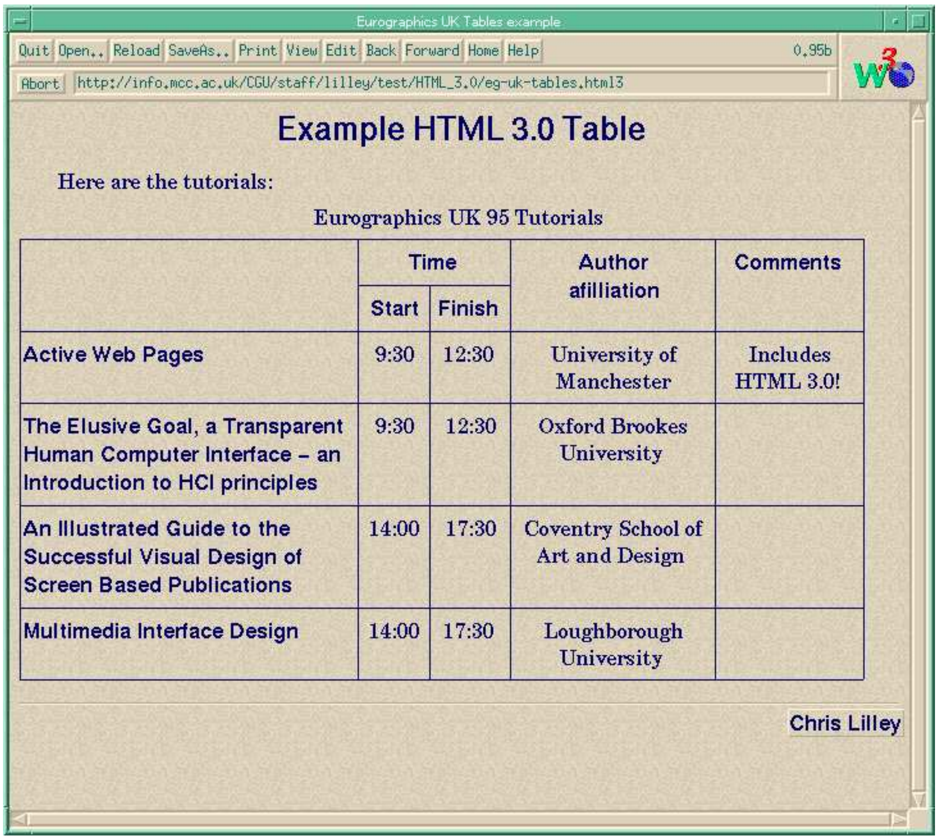
HTML 3.0 maths has also been designed to be parsed from left to right, so that it can be ‘spoken aloud’. This desirable property allows it to be rendered on output devices for the print disabled, such as text-to-speech converters or braille boards.

5.2 Tables

HTML 3.0 has full support for tables, which can be readily converted to and from popular wordprocessing formats. Tables are automatically sized according to the width of the browser window and the contents of each cell. The HTML 3.0 DTD includes attributes from the ICADD DTD, assisting text to speech or text to braille conversion. Tables can have all sorts of things in them - lists, images, forms, even other tables if you really want.

```
<TABLE BORDER>
<CAPTION>Eurographics UK 95 Tutorials</CAPTION>
<TR><TH ROWSPAN=2><TH COLSPAN=2>Time
<TH ROWSPAN=2>Author<BR>afilliation<TH ROWSPAN=”2”>Comments
<TR><TH>Start<TH>Finish
<TR><TH ALIGN=LEFT>Active Web Pages <TD>9:30<TD>12:30
<TD>University of Manchester<TD>Includes HTML 3.0!
<TR><TH ALIGN=LEFT>The Elusive Goal, a Transparent Human Computer
Interface - an Introduction to HCI principles
<TD>9:30<TD>12:30<TD>Oxford Brookes University
<TR><TH ALIGN=LEFT>An Illustrated Guide to the Successful Visual
Design of Screen Based Publications <TD>14:0<TD>17:30<TD>Coventry
School of Art and Design
<TR><TH ALIGN=LEFT>Multimedia Interface Design <TD>14:0<TD>17:30
<TD>Loughborough University
</TABLE>
```

shows one way this might be displayed in a browser:



Eurographics UK Tables example

Quit Open.. Reload SaveAs.. Print View Edit Back Forward Home Help 0.95b

Abort http://info.mcc.ac.uk/CGU/staff/lilley/test/HTML_3.0/eg-uk-tables.html3

Example HTML 3.0 Table

Here are the tutorials:

Eurographics UK 95 Tutorials

	Time		Author afilliation	Comments
	Start	Finish		
Active Web Pages	9:30	12:30	University of Manchester	Includes HTML 3.0!
The Elusive Goal, a Transparent Human Computer Interface – an Introduction to HCI principles	9:30	12:30	Oxford Brookes University	
An Illustrated Guide to the Successful Visual Design of Screen Based Publications	14:00	17:30	Coventry School of Art and Design	
Multimedia Interface Design	14:00	17:30	Loughborough University	

Chris Lilley

Figure 4: Example HTML 3.0 table rendered by Arena.

The same table could also be displayed on a text-only device, using dashes and horizontal bar characters to frame the table.

5.3 Figures

The `IMG` tag, introduced with HTML 1.0, was a short term measure intended to rapidly get graphics into HTML. This was essential for the Scientific Visualisation applications being developed by the Mosaic project at the US National Centre for Supercomputing Applications. `ALT` text was added, somewhat as an afterthought, to provide a simple substitute text string for text-only displays. This string can contain no HTML markup and is very limited.

In HTML 3.0, `IMG` still exists but it is restricted to simple graphics displayed on the current text line. The FIG element is for figures; these can be aligned left, centre or right, or placed in the margins. In contrast to `IMG`, `FIG` behaves as you would expect a figure to behave in a DTP application. Text flows around figures, they can have captions which stay with them, and copyright statements can be linked to the caption. `FIG`

```
<FIG align=left src=”http://site/path/file.jpeg”>
<caption align=center><small>This is a caption   
<A HREF=”copyright.html3”> &copy; MAN-T&amp;EC 1995</a>
</small></caption>
<h2>Congratulations!</h2>
<p align=right>You only see this text if you cannot display
pictures, so you are using a text only browser.</p>
</FIG>
```

Active images can continue to use `IMG` and pass links to the server for decoding. However they may alternatively use the `FIG` element, which is a container, so fully marked up text can be used as an alternative for clients that do not display images. The link also contains the area information and URLs (that are currently held in the map file on the server), and it is the *client* program that maps pointer positions to URLs. This also provides enough information for the client to display a list of the possible URLs.

In HTML 3.0, the shapes are given in abstract coordinates in the range 0 to 1, rather than in pixel coordinates. This is because clients may choose to resize images for display; also, figures are not restricted to bitmapped images. Vector graphics such as CGM may also be used, which are resolution independent, and thus positions in pixel coordinates are not possible. Circles, rectangles and polygons can be used as active areas.

Further reading

CGI programmers reference:
<http://www.halcyon.com/hedlund/cgi-faq/>

HTML 2.0:
<ftp://www.ics.uci.edu/pub/ietf/html/html2/htmlspec281194_1.html>

HTML 3.0:
<http://www.hpl.hp.co.uk/people/dsr/html/Contents.html>

HTML Check toolkit:
<http://www.hal.com/~markg/HaLSoft/html-check/index.html>

HTTP 1.0:
<http://www.w3.org/hypertext/WWW/Protocols/HTTP1.0/draft-fielding-http-spec-01.html>

ICADD converter:
<http://www.ucla.edu/ICADD/html2icadd-form.html>

Mapedit:
<http://sunsite.unc.edu/boutell/mapedit/mapedit.html>

Acknowledgements

This paper and tutorial are based on materials written by the author as part of a pair of courses '*Internet Skills for Bioinformatics*' which are being developed by Manchester and North HPC Training & Education Centre (MAN T&EC), funded by JISC New Technologies Initiative. I would like to thank Andy Grant, Manager of MAN T&EC for his support. I would also like to thank Ruth Loebl of the Royal National Institute for the Blind for helpful information about Braille and speech output devices, and Mark Gaither of HaL Computer Systems for assistance with HTML 2.0 and 3.0 validation software without which parts of this paper and tutorial would be somewhat speculative. Lastly, thanks to Tim Berners-Lee (for inventing the Web, which is five years old this month!), Dan Connolly (for driving HTML 2.0 standardisation) and Dave Raggett (for devising HTML 3.0).