Basic HTML Programming

HTML

HTML stands for **Hypertext Markup Language**.

HTML files are basically special **text** files:

- Contain special control sequences or **tags** that control how text is to be formatted.
- HTML files are the "source code" for Web Browsers
  - A browser reads the HTML file and
  - Tries to display it using the tags to control layout.
- **Text** file created by:
  - Any text editor — FREE: **BBEdit Lite**
  - Special HTML editors — freeware — expensive: **Dreamweaver**




Creating HTML Documents and Managing Web site

Text Editors

- BBEditLite
- Notepad, wordpad
- Other text editing programs

WYSIWYG Editors

- MS Frontpage
- Macromedia Dreamweaver
- Others exist

First use text editor programs to learn HTML Basics. Starting with WYSIWYG hinders learning process. Only use WYSIWYG when you know basics.




Creating Your Own Web Pages

The process in creating permanent WWW (HTML + related scripts (later)) pages in this course is basically

- Create, and test, local files on the Macintosh Computer or share UNIX files directly across a locally (samba) mounted shared folder.
- Save HTML documents with extension .html, e.g. sport.html
- Ultimately you should store Permanent Copies of files on Department's UNIX System/Web Server.




Back
Close

Storing and Serving Files on School UNIX WWW Server

- Once you are happy with the format of the local (Mac based files) and you want a permanent Web page you should FTP the file to your personal UNIX file space.

OR

- If you have mounted you UNIX files via SAMBA

There are **2** places where you may store HTML files on your personal UNIX WWW file space.

NOTE: There is a Difference




Two Special Sub-Directories in your Home Directory

`project_html` — Files place in this directory will be viewable only within the department. Ideal for coureswork

- Use URL:

  `http://project.cs.cf.ac.uk/A.B.Surname`

  where **A.B.Surname** is your long email name to reference files from the Web.

`public_html` — Files place in this directory will be viewable on the whole Internet. This is where you would create your **Home Page**.

- Use URL:

  `http://users.cs.cf.ac.uk/A.B.Surname`

  where **A.B.Surname** is your long email name to reference files from the Web.




Making Your Web Space Available

The directories should have been created for you but you wil have to PUBLISH your pages on the COMSC Information Server.

- You agree to abide by University/School Regulations when you PUBLISH your pages
- To PUBLISH your pages, go to URL:

  `http://www.cs.cf.ac.uk/user/`

  and click on PUBLISH button to make your pages available.
- UNPUBLISH available also.




Back
Close

Further Information of your School's Web Space

Further information on user and project Web pages at:

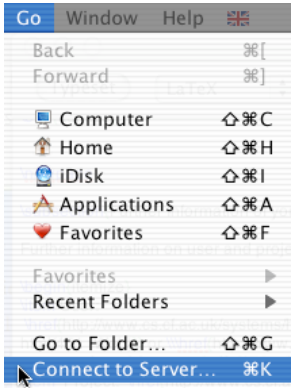
- User: http://www.cs.cf.ac.uk/systems/html/451 (Web) or http://www.cs.cf.ac.uk/systems/pdfs/451.pdf (PDF)
- Project: http://www.cs.cf.ac.uk/systems/html/452 (Web) or http://www.cs.cf.ac.uk/systems/pdfs/452.pdf (PDF)




UNIX File space and Storing Files on UNIX: SAMBA

Two ways to transfer your files on UNIX Web Space

Mount Unix Files via Samba



- Use **Macintosh Finder GO Connect to Server...** to mount your web space.

- Choose the COMSC network and the `claros` (or most other machines)
- Select and Store files in in `project_html` or `public_html` directory.

UNIX File space and Storing Files on UNIX: FTP

If will need to transfer files from the Macintosh to UNIX and alternative is to use FTP (STP from outside School):

- Fire up **Fetch Application** or command line FTP/SFTP.
- FTP to `ftp.cs.cf.ac.uk`
- Login into your own file space by using your on UNIX User ID (**e.g.** scm...) and password.
- Change the directory to your `project_html` or `public_html` directory.
- **Put** the file(s) in this directory.

Creating Your Own Home Page and other files for the WWW

- Create the file using BBEdit (or another) and save it to disk. If the file is
  intended to be your home page save it as `index.html`.
- Make sure file is correct by viewing it locally in Web Browser on Macintosh (or PC!).
- Use Samba/FTP to access your UNIX File space.
- Save file in the `public_html` directory. It **Must be placed** here.
- `index.html` should be the "*home*" file for every subdirectory of your web space — Browser always look for this file if just a directory is referenced, E.g.:

  `http://users.cs.cf.ac.uk/A.B.Surname/`: You home page is assumed to be `index.html`

  `http://users.cs.cf.ac.uk/A.B.Surname/SubDIR`: `index.html` is assumed to be present in the `SubDIR`.




The Best Way to Learn HTML

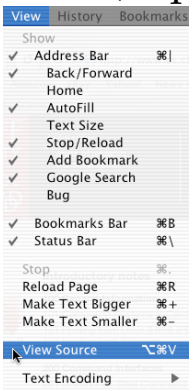
The best way to learn HTML (or any programming language) is by example.

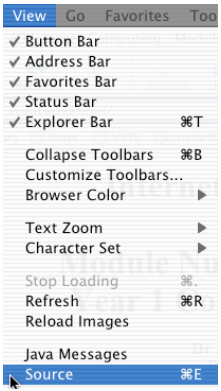
You can read many books but practice, i.e. writing your own HTML pages and learning from example WWW pages on line, is the best way to learn tips and constructs.




Viewing HTML Source in your Browser

- Find a Web page you like or wish to learn how it is formatted.
- Make sure this Web page is currently being viewed by Web Browser.
- You can view the WWW page by selecting the **View Source** (Safari) item or **Source** (Explorer) item from the **View** Menu.





- Compare the HTML with the browser display of the Page.
- Portions of the file may be selected with the mouse (click and drag mouse) and then Copied and Pasted into other documents (E.g. BBEdit/Dreamweaver windows.




Anatomy of Any HTML Document

Every HTML document consists of two elements:

- **Head** elements — provides page title and general page formatting commands
- **Body** elements — put the main HTML text in this part.




HTML Tags

All HTML commands or tags have the following form:

<name_of_tag>...</name_of_tag>

Tags control the structure, formatting and hypertext linking or HTML pages.

Tags are made active by <name_of_tag> and must be made inactive by an associated </name_of_tag>.

HTML is not case sensitive — tags can be upper or lower case letters (even mixtures of cases) — Not recommended.




Basic HTML Page Structure

We can now meet or first three HTML tags `html`, `head` and `body`

Note that these specify the basic anatomy of every HTML page.

```
<html>
<head>
head elements go here
</head>
<body>
body elements go here
</body>
</html>
```

NOTE:

- `<html>` is the first tag of any HTML page. It indicates that the contents of the page is in HTML.
- `</html>` has to be the last tag of any HTML page




Basic HTML Coding

Head elements

- `<head>....</head>` tag delimits head part of document.
- `<title>....</title>` Defines the title of the Web page.
- Ever Web page should have a `title`
  - Displayed as Title of Web Browser Window
  - Used in Bookmarks or Hot lists to identify page
  - Make title succinct but meaningful
  - Only one title per page
  - Only plain text in title (no other tags).
  - Usually `<body>` first level one header same as `title` (see below).




The Body Element

- `<body>....</body>` tag delimits body part of document.
- All other commands that constitute web page **nested** inside body.
- `Body` must follow `head`.




Headings

- Headings are used to title sections and subsection of a document.
- HTML has 6 levels of headings labelled `h1, h2, ..., h6`.
- Delimit headings between the `<hn>....</hn>` tags where $n = 1 \ldots 6$
- The first heading should be `<h1>` item
  In most documents the first heading is the same as the page `title`.
- Headings are displayed in larger/bolder fonts than normal body text.
- Increment headings linearly — do not skip.




Example of HTML headings

```
<html>
<head>
<title> HTML Heading Levels
</title>
</head>


<body>
<h1> This is a level 1 heading </h1>

<p>
This is not a heading. It is a paragraph.
</p>
<h2> This is a level 2 heading </h2>
<h3> This is a level 3 heading </h3>
<h4> This is a level 4 heading </h4>
<h5> This is a level 5 heading </h5>
<h6> This is a level 6 heading </h6>
</body>
</html>
```





Which looks like this when viewed through a browser:

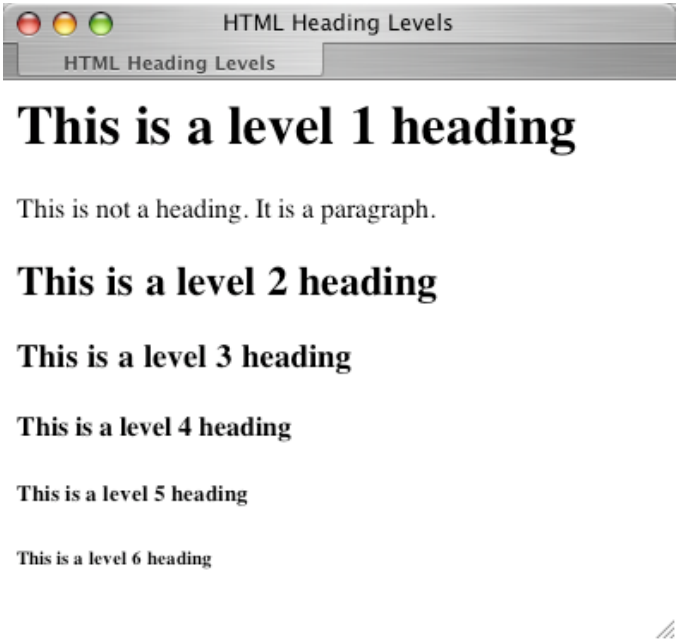


Figure 11: HTML Heading Levels Example




HTML Comments

Comments delimited by:

```
<!-- ......... -->
```

- Ignored by browser – No formatting function
- Like all good programming practice:

Use meaning comments in your HTML

Simple comment example:

```
<!-- THIS IS A COMMENT -->
```





Paragraphs

- `<p> ....</p>` tag delimts a paragraph.
- HTML ignores most carriage returns in a file — so must use `<p>` or `<br>` tag to get a newline in the browser.
- Text is wrapped until a `<p>` or `</p>` encountered.
  - HTML assumes that if a `<p>` is encounterd before a `</p>` then a paragraph should be inserted. (Old HTML Legacy)
  - Bad practice to leave out `</p>`.
- Paragraphs can be aligned — `LEFT, CENTER, RIGHT` – with the `ALIGN` attribute via

```
<p align=center>
```

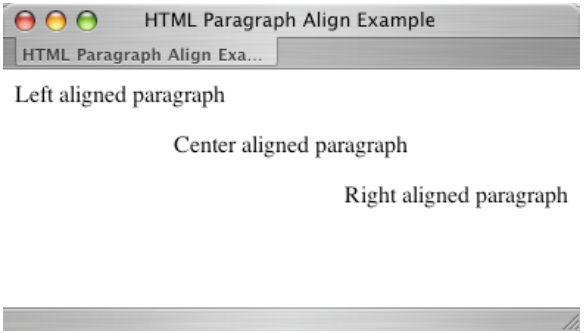




Paragraph Align Example

```
<p align>
<!-- THIS IS A COMMENT -->
<!-- Default align is left -->
Left aligned paragraph
</p>

<p align = center>

Center aligned paragraph
</p>

<p align = right>

Right aligned paragraph
</p>
```

Which looks like this when viewed through a browser:



Links and Anchors

Linking to Other Documents — The Bread and Butter of the Web

Regions of text can be linked to other documents via the anchor, `<a>`, tag which has the following format:

`<a href=''`*filename_or_URL*`''> link text </a>`

- The opening `<a>` tag has a `href` attribute that is used to specify the link to URL or local file.
- Text between the `<a>` and `</a>` (closing tag) is highlighted by the browser to indicate the hyperlink.

- Depending on browser and web page configuration highlight style can differ.




Relative and Absolute Links

There is a subtle and very important between the links in the previous example:

Relative links — refer to a page in relation to the current document

- sub-directories and included files can be specified in the relative link.
- Makes for very portable web pages.
  Whole directory systems can be moved easily.
- e.g.

```
<a href="\Sport\Football.html">Football</a>
```

Absolute links — reference files based on the absolute location on the local file system or WWW.

- e.g.

```
<a href="http://www.bbc.co.uk/Sport/Football.html">Fo
```


Anchors — Jumping to specific places in a document

Anchors are special places within documents that can be linked to.

- Anchors may placed anywhere in a document with

```
<a name = "anchor_name">Anchor Position</a>
```

- Anchors **within the same document** are referred to by

```
<a href = "#anchor_name">Go to anchor</a>
```

- Anchors **in the external document** are referred to by

```
<a href = "link#anchor_name">
```

where `link` may a relative, absolute or remote URL link.




Anchor Example - Same code for external or internal:

```
<ul>
 <li><a href="#apples">apples</a></li>
 <li><a href="#oranges">oranges</a></li>
 <li><a href="#bananas">bananas</a></li>
</ul>
<h2>Information</h2>
<p><a name="apples">
 Apples are green
</p>
<p><a name="oranges">
 Oranges are orange
</p>
<p><a name="bananas">
 Bananas are yellow
</p>
```





Lists

HTML supports a variety of lists.

Unordered or Bulleted lists

- `<ul> ... </ul>` delimits list.
- `<li>` indicates list items.
- Closing `</li>` is not strictly required. (Old HTML Legacy) But recommended.

```
<ul>
<li> apples. </li>
<li> bananas.</li>
</ul>
```

Which looks like this when viewed through a browser:

- apples.
- bananas.

Ordered or Numbered lists

- `<ul> ... </ul>` delimits list.
- `<li>` indicates list items.
- Closed with `</li>`.

For Example:

```
<ol>
<li> apples.</li>
<li> bananas.</li>
</ol>
```

Which looks like this when viewed through a browser:

1. apples.
2. bananas.




Preformatted Text

The <PRE> tag generates text in a fixed width font and causes spaces, new lines and tabs to be significant. Often used for program listings.

Example:

```
<pre>
   This is preformatted       text.

New lines, spaces etc. are
significant.
</pre>
```

which looks like this when viewed through a browser:

```
   This is preformatted       text.

New lines, spaces etc. are
significant.
```

In-Line Images

All browsers can display in-line images that are in JPEG or GIF format.

- Use the `img` tag with `src` attribute to include an image in you HTML page:

  <img src=image_link>

  where image_link is the the relative, absolute or remote URL link of the image file.

- Include `alt=''replacement''` attribute for browsers that may not be set to display graphics, where `replacement` is a meaningful short text description.
  - Some people preserve Bandwidth/ Download time by turning off image display in a browser
  - If image is corrupted or URL is "misdirected" — perhaps some files/directories moved.
  - `Lynx` text only browser
  - Special needs Browsers for Blind/Partially sighted people
- Images can be `align`ed like paragraphs.
- Images can be resized in pixel size or percentage with `width` and/or `height` attributes.

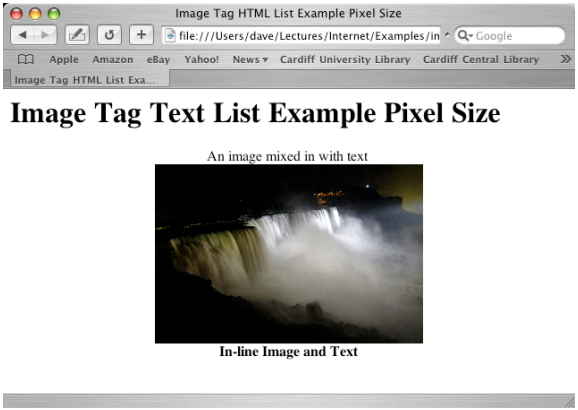



In-Line Image Example Pixel Size

HTML:

```
<p align = center>
An image mixed in with text <br>

<img src="niagara.jpg" align=middle
width = 300 height = 200
alt="Niagara Falls">
</p>
```

Which looks like this when viewed through a browser:



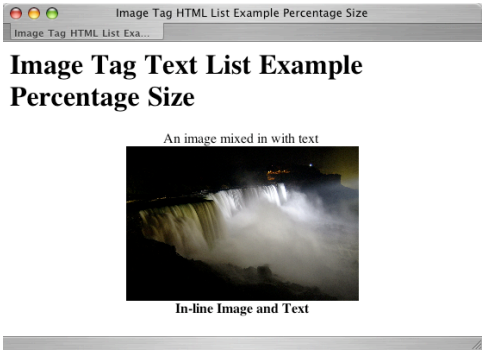
In-Line Image Example Percentage Size

HTML:

- **Note:** Only have set `width` here to preserve image aspect.

```
<p align = center>
An image mixed in with text <br>

<img src="niagara.jpg" align=middle
width = 50%
alt="Niagara Falls">
</p>
```

Which looks like this when viewed through a browser:



Differences between Absolute Pixel and Percentage Image Size?

- Pixel size fixes size of image — if window resizes too small then scrolling will be invoked.
- When you resize browser window always rescales image to fit window – to within a reasonable minimum size.




External Images, Sounds, Video

External Images will be loaded into their own page as a simple URL. The `href` field within the `anchor` tag is used.

These are easily included by using
link anchor
link anchor
link anchor

```
<a href="http://www.bbc.co.uk/sound_file.wav">MADE UP LINK
```

Embedded (in-line) Audio, Midi and Video

To include Audio, MIDI and Video Elements in a web page use the `embed` tag, syntax:

```
 <EMBED SRC="media_url"
HEIGHT=
WIDTH=
AUTOPLAY= TRUE/FLASE
LOOP= TRUE/FALSE>
```

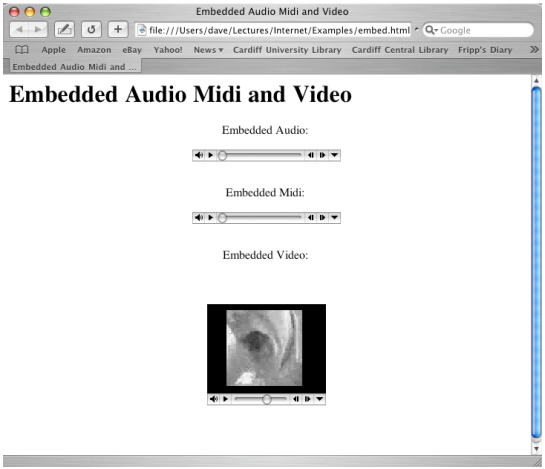
- `height`, `width` as before
- `autoplay = TRUE` forces media to play on page load
- `loop = TRUE` forced media to loop continously on page




Example Embedded Audio, Midi and Video

```
<p align = center>
Embedded Audio: <br>
<EMBED SRC="Shaggy.wav"
HEIGHT=50 WIDTH=200
AUTOPLAY=FLASE LOOP=FALSE>
</p>

<p align = center>
Embedded Midi: <br>
<EMBED SRC="mars.mid"
HEIGHT=50 WIDTH=200
AUTOPLAY=FALSE LOOP=FALSE>
</p>

<p align = center>
Embedded Video: <br>
<EMBED SRC="JawsII.mov"
   HEIGHT=250 WIDTH=200
   AUTOPLAY=TRUE LOOP=FALSE>
</p>
```

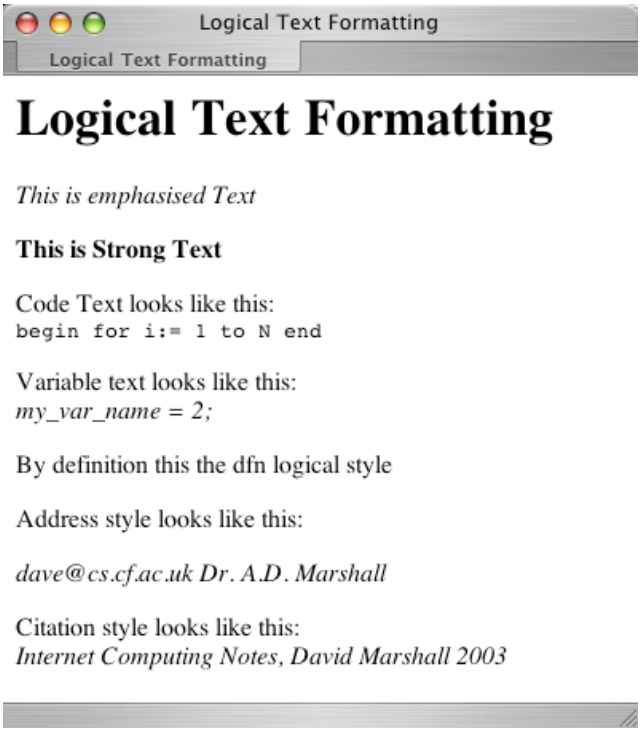
Which looks like this when viewed through a browser:



Logical Character Tags Examples

```
<em>
This is emphasised Text
</em>
 <strong>
This is Strong Text
</strong>

 Code Text looks like this:<br>
<code>
begin

for i:= 1 to N

end
</code>

Variable text looks like this:<br>
<var>
my_var_name = 2;
</var>

<dfn>
By definition  this the dfn logical style
</dfn>

Address style looks like this:
<ADDRESS>
dave@cs.cf.ac.uk Dr. A.D. Marshall
</ADDRESS>
</p>

Citation style looks like this:<br>
<cite>
Internet Computing Notes, David Marshall 2003
</cite>
```

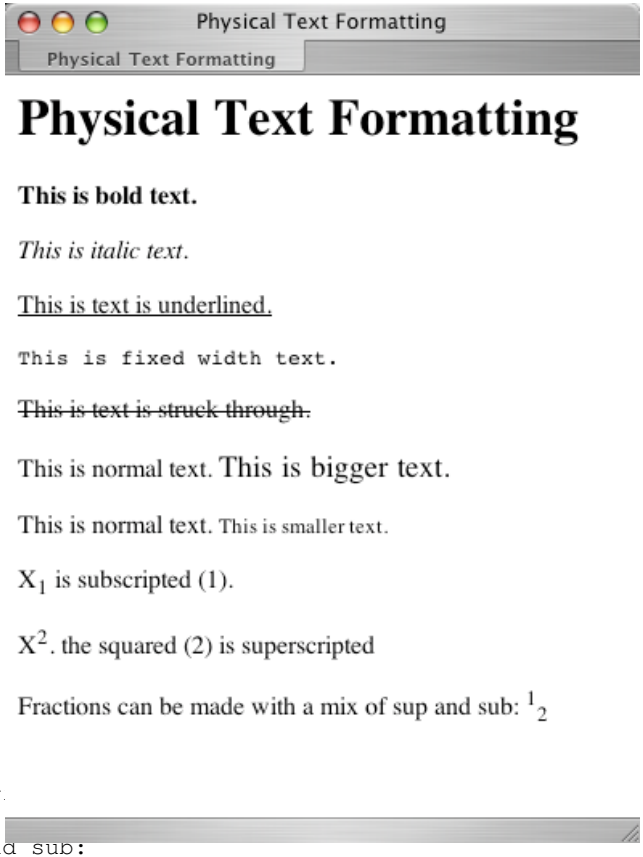
Which looks like this:



Physical Character Tags Examples

```
<b>
This is bold text.
</b>

<i>
This is italic text.
</i>

<u>
This is text is underlined.
</u>

<tt>
This is fixed width text.
</tt>

<s>
This is text is struck through.
</s>

This is normal text.
<big>
This is bigger text.
</big>

This is normal text.
<small>
This is smaller text.
</small>

X<sub>1</sub> is subscripted (1).

X<sup>2</sup>. the squared (2) is superscr

Fractions can be made with a mix of sup and sub:
<sup>1</sup><sub>2</sub>
```

Which looks like this:



Special Characters

Certain characters need to be referred to in a special way.

These include:

- Character strings that represent special symbols, *e.g.*
  - &amp for &
  - &lt for <
  - &gt for >
  - &quot for double quote (“)

Horizontal rules and Line breaks

There are two tags that can be used to control the layout of your page.

- Horizontal Rule `<hr>`
- Line break `<br>` — inserts a end of line where it appear

Neither have a closing tag or associated text.
Their use is fairly straightforward.




Horizontal Rule `<hr>`

The `<hr>` has 4 attributes that may be associated with it.

- The `size` attribute to specify thickness of line in pixels (pixels are individual dots displayed on the screen).

For example:

```
<b>2 Pixels</b><br>
<hr size=2>
<b>4 Pixels</b><br>
<hr size=4>
<b>8 Pixels</b><br>
<hr size=8>
<b>16 Pixels</b><br>
<hr size=16>
```

Which looks like this:

Can also add/remove shading, alter alignment and change width e.g.

```
<hr noshade align=right width=50%>
```

Fonts and Font Sizes

The `<font>` tag is used to change the font size and type face of text enclosed between the begin and end tag.

- The `size` attribute changes the size of the font. Allowed values are 1 to 7.
  - size attributes can be incremented or decrements with + operator **within the above range**.

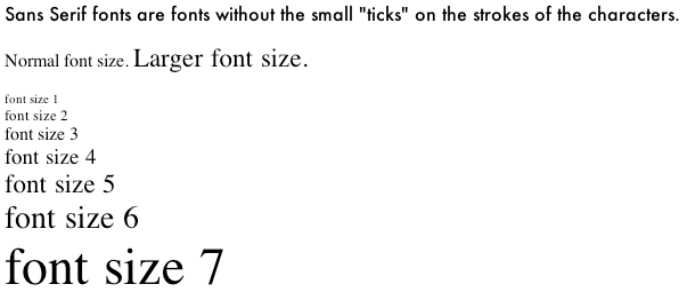
    E.g `size = +2`.
- The `face` attribute to select a type face.
  E.g. `face = "futura,helvetica"`, face = ”courier”




Example uses of `font` tag: `face` and absolute `size`

```
<P><font face="Futura,Helvetica">Sans Serif fonts are fonts
without  the small "ticks" on the strokes of the characters.
 </font></P>

<P>Normal font size. <font size=5>Larger font size.</font></P>

<font size=1>font size 1</font><br>
<font size=2>font size 2</font><br>
<font size=3>font size 3</font><br>
<font size=4>font size 4</font><br>
<font size=5>font size 5</font><br>
<font size=6>font size 6</font><br>
<font size=7>font size 7</font><br>
```

looks looks like this






Web Page Backgrounds

You can do a few simple yet effective things to spice up your web pages.

It is straightforward to

- Change the colour of your background.
- Make a (GIF or JPEG) image a background




Colour in HTML

Colour is widely used in many HTML contexts.

We, briefly, introduce the concept of colour in HTML here.

There are two ways to specify colour:

- Use **Hexadecimal numbers** to specify each red, green and blue component.
- Use one of a set of **predefined colour names**.




Hexadecimal Colour Representation in HTML

Colour for each red, green and blue colour component is represented by a range 0 (0 Hex) to 255 decimal (FF hex) where

- 0 indicates zero colour component
- 255 indicates full colour component
- Other values a proportion of a the colour value.




Example Hexadecimal Colours

Thus, we can form:

Red	Green	Blue	Colour
0	0	0	Black
255	255	255	White
255	0	0	Red (full)
0	255	0	Green (full)
0	255	0	Blue (full)
x	x	x	x = 0-255
			Shade of grey
255	0	255	Magenta
0	255	255	Yellow

and so on ...




Back
Close

Predefined Colours in HTML

You can choose from predefined colours, for example:

Black, White, Green, Maroon, Olive, Navy, Purple, Gray, Red, Yellow, Blue, Teal, Lime, Aqua, Fuchsia and Silver.

- Colour names are easier to remember. The only drawback is a restricted choice compared with several million possible colours possible with hexadecimal representation.
- The fidelity and ultimate rendering of colour will depend on the hardware you run the browser on.




Setting the Background Colour of Your Browser

To change the background colour you must set the BGCOLOR attribute in the BODY tag.

- To specify a hexadecimal number you must put a # before the number and then two hex digits for each red, green and blue component respectively.

  For example, **whitehex.html**:

```
<BODY BGCOLOR = "#FFFFFF">
```

  gives us a white background.




Some Example Named Colour Backgrounds

To set BGCOLOR with a **predefined name** simply refer to the **name of one of the allowed colours**.

For example, green.html:

```
<BODY BGCOLOR = "green">
```

gives us a green background.

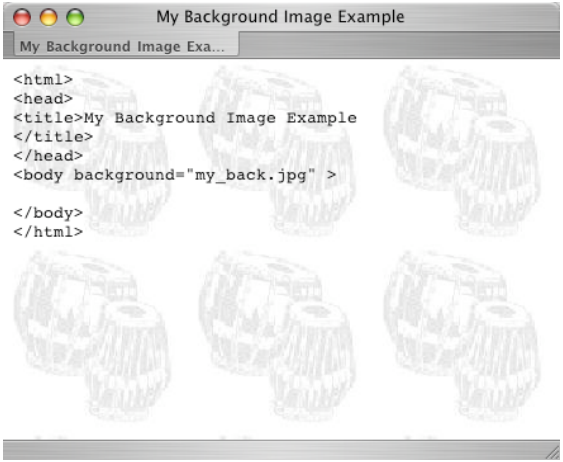



Image Backgrounds

You can use an image as a background for your web pages.

To do this use the BACKGROUND attribute of the BODY tag, for example, **my_back.html**:

```
<body background="my_back.gif">
```



URLs (relative or absolute) can be supplied:

```
<body background=
   "images/my_back.gif">

<body background=
  "/server_images/my_back.gif">

<body background=
   "http://www.myimageserver/my_back.gif">
```

Using Image Backgrounds

Some precautions should be taken when using images:

- Keep the size of the images small
- Browsers **tile** images:
  - Small images are repeated in rows and columns to fill the web page.
  - This saves on downloading overheads of large images
  - Images should have patterns that "flow" between tiles
  - GIF and JPEG image formats allowed
- Browsers **cache** images and web pages so reuse a backgrounds on several pages.
- Reusing a background creates a consistency to pages providing a "web site" overall image.




Scrolled Image Backgrounds

Image backgrounds cane be Scroll controlled via the `bgproperties` attribute of the `BODY` tag

- If `bgproperties="fixed"` is set when (Web Page) text is scrolled background image remains fixed

  For example, **my_back_fixed.html** :

```
<body  background="my_back.jpg" bgproperties="fixed">
```

- **Otherwise** the background image also scrolls

  For example, `my_back_scroll.html`.




Text Colour in HTML

You can change the colour of text on your web page in much the same way as you set the background colour.

- You can change the colour of all the text on page by setting the `TEXT` attribute of the `BODY` tag.
- For example redtexthex.html:

```
<body text = "#FF0000">
```

or, redtext.html:

```
<body text = "red">
```

gives us red text on our page.




Multicoloured Text

Setting the Text colour in the BODY still only gives one colour for all text on a page:

- You can also colour individual portions of text with the <FONT> tag by setting the COLOR attribute.

For Example, **fontcol.html**:

```
Multicoloured text:
<br><br>

<font color = "#FF0000">
This is RED text
</font><br><br>

<font color = "#00FF00">
This is GREEN text
</font><br><br>

<font color = "#0000FF">
This is Blue text
</font><br><br>
```

Which gives:

Multicoloured text:

This is RED text

This is GREEN text

This is Blue text

CGI Scripting

What is a CGI Script?

A **CGI script** is any program that runs on a web server.

Why CGI Scripts:

CGI stands for **C**ommon **G**ateway **Interface**

CGI defines a standard way in which information may be passed to and from the browser and server.

Any program or script that can process information according to the CGI specification (part of HTTP protocol) can, in theory, be used to code a CGI script.




The Role of CGI

The role of this CGI script is to:

- Accept the data which the user inputs and
- Do something with it.
- Usually, send a reply back to user.

Figure 26: The Common Gateway Interface




Writing and Running CGI Scripts

CGI scripts can exist in many forms — depending upon what the server supports.

CGI scripts can be compiled programs or batch files or any executable entity. For simplicity we will use the term **script** for all CGI entities.

Typically CGI scripts are written in:

- Perl scripts — **colorgreen The method we adopt**. Most common too
- C/C++ programs
- Unix Scripts

colorred We will concentrate on Perl in this course.

CGI scripts therefore have to be written (and maybe compiled) and **checked for errors** before they are run on the server.




Calling a CGI Script

CGI can be called and run in a variety of ways on the server.

The 2 most common ways of running a CGI script are:

- From an HTML Form — the `ACTION` attribute of the form specifies the CGI script to be run.
- Direct URL reference — A CGI script can be run directly by giving the URL explicitly in HTML.
  - Arguments (values) may be required by the script this will have to passed in.
  - We will see how to do this shortly.

One other way CGI scripts are called is in **Server-side include** HTML commands.

- This is something we will leave until later.




Creating CGI Scripts

We will be creating CGI scripts in Perl.

Perl has become the default language for creating CGI scripts as it has many useful features and a rich set of libraries.

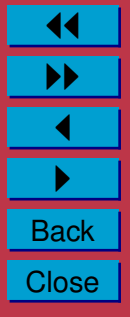
On Mac OS X/UNIX perl scripts are executed as **scripts**

- Perl Scripts are interpreted — **no need to compile**
- A special program, **the Perl Interpreter**, supplied on the system.

On PC (also LINUX/Solaris)

- ActivePerl is used (Industry Standard)

Perl is a freely available for most platforms — see www.perl.org or www.perl.com



Perl CGI Script Development Cycle

The basic cycle of perl script development recommended for **this course** is:

1. Write and create Perl scripts on Local Machine (Mac OS X/PC/LINUX).
2. Test, run and debug Perl script Local Machine (Mac OS X/PC/LINUX)
   - **Possibly not fully functional CGI test**
   - **But test for syntax and basic output**.
3. For permanent storage or permanent Web distribution, Samba/FTP perl script and HTML to School's UNIX/LINUX Web Server
   - As with HTML home pages there are two distinct ways to serve CGI (Local (Project) and Global (Public)) scripts.
   - HTML files must be placed in special directories.
   - Perl scripts must be place in special (sub) directories (`cgi-bin` subdirectory from HTML directory — **more soon**).




Setting up CGI Scripts to run on a server

As mentioned above and in similar fashion HTML code need to located in a special place in order to run and operate properly.

CGI Scripts on School's Web Server

The process of installing CGI scripts is similar to that of HTML pages, except different (sub)directories and URLs are used.

`project_html/cgi-bin` — Files placed in this directory will be accessible **only within** the school.

- Use URL:

  `http://www.cs.cf.ac.uk/project/A.B.Surname/cgi-bin`

  where **A.B.Surname** is your long email name to reference files from HTML forms or directly.
- Associated HTML files (**i.e.** ones whose `FORM ACTION` calls the CGI script) must still be placed in the `project_html` directory (one (sub)directory level above.




World Wide CGI Scripts on School's Web Server

`public_html/cgi-bin` — Files placed in this (sub)directory will be viewable on the **whole Internet.**

- Use URL:

```
http://www.cs.cf.ac.uk/User-bin/A.B.Surname
```

where **A.B.Surname** is your long email name to reference scripts from HTML or direct URL.




Setting up `cgi-bin` (sub)directories

- `cgi-bin` (sub)directories should already be created for you.
- You will have register your `project` and `public_html/cgi-bin` directory on the School's Web Server.
- CGI scripts placed here will need their access permission changed.
  - See more information of following slide

Further information on user and project Web CGI pages at:

- User: http://www.cs.cf.ac.uk/systems/html/451 (Web) or http://www.cs.cf.ac.uk/systems/pdfs/451.pdf (PDF)
- Project: http://www.cs.cf.ac.uk/systems/html/452 (Web) or http://www.cs.cf.ac.uk/systems/pdfs/452.pdf (PDF)




Back
Close

Configuring and Running Individual CGI scripts

Simply place (FTP) the CGI script in the `public` or `project_html/cgi-bin` subdirectories.

Every CGI script will need to have certain access modes changed. This can be done from Macintosh (with Fetch) or on UNIX/LINUX (via Telnet connection).

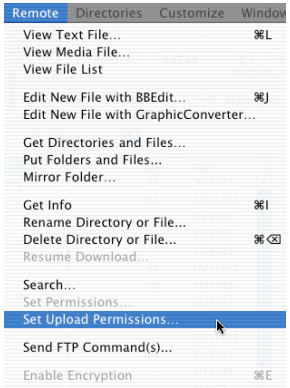
CGI scripts have a maximum CPU runtime of 30 secs — after which they are terminated.

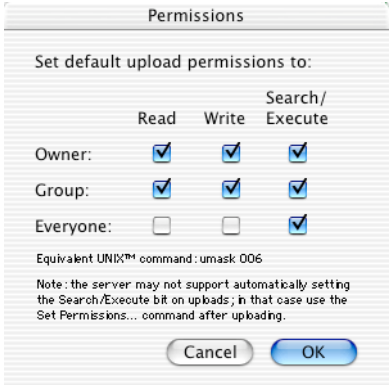



Setting File Access Modes – Fetch FTP Application

To set file access modes from **colorgreen Fetch (just before FTP transfer)**:

- Select the **Set Upload Permissions...** Menu item from the **Remote** Menu.
- A new window appears. Click on and set the **Owner** and **Group** **read, write** and **search/execute** permissions and set the **Everyone** **search/execute** permission





Setting File Access Modes – UNIX/LINUX (via Telnet)

To Set CGI file permissions from **UNIX**:

- Assume we have create the CGI script called `test1.pl`.
- `test1.pl` must reside in directory `project` or `public_html/cgi-bin` (or copy it there).
- To change the mode of the script to make it executable and accessible by the Web server type (from your top level directory in example below):

```
chmod +x public_html/cgi-bin/test1.pl
touch public_html/cgi-bin/test1.pl
chmodwww public_html/cgi-bin/test1.pl
```

- If you `cd` into `public_html/cgi-bin` or `project_html/cgi-bin` then you need not type full paths
- `chmodwww test1.pl` may be replaced by `chmod og+w test1.pl`

You should now be able to access the script using URL:

`http://www.cs.cf.ac.uk/user(project)/A.B.Surname/cgi-bin`