Cookies

Cookies : Cookies can store small pieces of information on the client side. The information could be something like the User ID that the user uses to log on to the site. It can also contain other info. such as the time when user last logged onto the site. Cookies are dependent on browser settings of client m/c. Client may disable cookies being written on the client m/c.

Emergence of Cookies → As HTTP is a stateless protocol. There is no way to maintain connection or user information across multiple requests to the same server by the same client (in absence of database on web server). Netscape addressed this issue on web with Cookies. A cookie is a small piece of text data set by a web Server that resides on the client's machine. Once it has been set, the client automatically returns the cookie to the web server with each request that it makes. This allows the server to place values it wishes to "remember" in the cookie, and have access to them when creating a response.

During each transaction, the server has the opportunity to modify or delete any cookies it has already set and also has the ability to set new cookies.

The most common use of this information is the identification of individual users. eg. a site will have a user log-in & will then set a cookie containing the appropriate user name. After that whenever the user makes a request to that particular site, the browser sends the username cookie in addition to the usual info. to the server.

Domain Name

Secur

Syntax for setting cookies is

document.cookie = "cookie_name = Cookie value;
path = <pathname>; domain = <domain>;
expires = <sometime>; Secure;"

Cookie is created as name-value pair. The cookie parameters path, domain expires & secure are optional.

syntax = document.cookie = "key1 = value1; key2 = value2; expires = date";

eg. document.cookie = "userid = John; domain = www.ibm.com;"

Functionality of Cookies Parameters

Parameter Name	Functionality
1. Name	Name of the cookie.
2. value	Value of the cookie.
3. Path	Specifies the directory in which the cookie can be stored. Specifying the directory where the cookie is to be stored gives us the ability to have more than one copy of the same cookie in two different directories. Normally the default path for a client side cookie is the current working directory

	Specifies the lifetime of a cookie The value is given as Greenwich Mean Time.
Domain Name	Used to identify a cookie that is stored on a machine. It is used to identify a particular cookie created by a specific server.
Secure	Indicates that the cookie is only to be returned over a secure (HTTPS) connection.

Cookies in JavaScript :→

Cookies can be accessed in JavaScript by the cookie property of document object. The cookie property is both readable & writeable.

Setting Cookies :

When you assign a string to document.cookie, the browser parses it as cookie & add to its list of cookies.

eg.

```
document.cookie = "username=fred; expires=Sun, 01-Dec-2
                   08:00:00 GMT; path=/home";
```

Type of Cookies : 1. Session 2. Persistent

Session Cookies : Cookies that are set without the expires field are called session cookies. It is called session cookies bcoz they are kept for only the current browser session; they are destroyed when the user quits the browser.

Persistent Cookies → Cookies that are not session cookies are called persistent cookies bcoz the browser keeps them until their expiration date is reached, at which time they are destroyed.

Cookie Limitations

1. The total no. of cookies a browser can store at a time is limited to several hundred.
2. Total no. of cookies a browser can store at a time from particular site is 20.
3. Cookie size is almost 4000 characters.

Security Issues with Cookies

Cookies reside on the user's m/c there is nothing stopping the user from modifying a cookie's value after it is set by server. So don't keep sensitive information in cookies without encryption.

Uses of Cookies :—

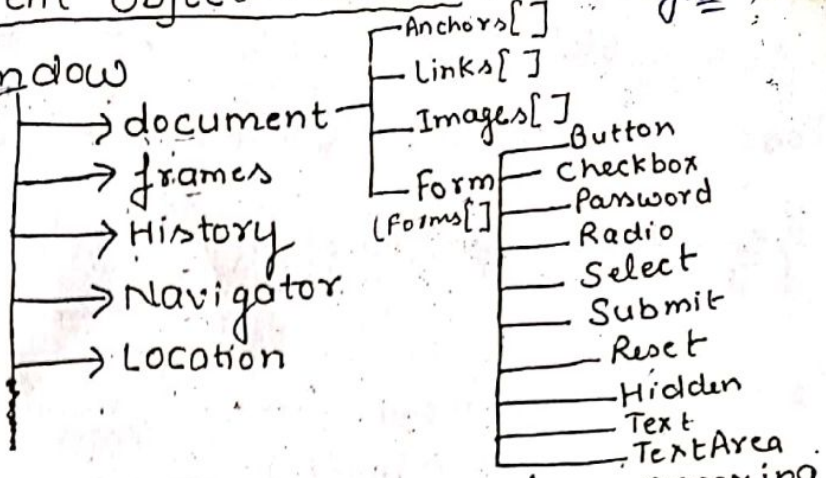
1. Redirect the user from one page to other.
2. One-Time Pop-Ups
3. Customizations

Document Object Model

Jyoti

2-(II)

Object based on DOM are also known as Browser objects. The DOM describe hierarchy of object which represent diff. element in browser.



Window : Represents the top level object for accessing any object other than the Navigator object. The window object represents the current browser instance. It has properties & methods. The following are some of the properties of this object :

- name : Provides the name of the window handle
- status : Provides the window status text msg. appearing on the browser window.
- defaultstatus : Accesses the default window status text message. We can also change the defaultstatus message.

window.property name

Method

- open : Opens a new browser window. This takes three parameters, viz. the URL of the document to be loaded, the name of the window handle and the physical properties of the window. various properties that taken by window.open method
  - Left : Represents left X coordinate on the screen in pixels, to open window.
  - Top : Rep. top Y coordinate on screen
  - width : Specifies the width of window in pixels.
  - Height : " " height " " .
  - scrollbars : Yes or No
  - location : Yes or No
  - Directories : Yes or No
  - Status : Yes or No

· Menubar : Yes or No
· Resizable : Yes or No

Close : This method closes the Java Script Window.

```
eg. nw = window.open("test.htm", "temp", "left=200,
        top=200, height=300, width=500
        scrollbar=no, location=no, resizable=
        no, menubar=no");
    window.close();
```

In case of browsers with frames, there are some more properties

· top : Refers to top most browser in the frame hierarchy.
· self : Refers to the window where the script exists.
· parent : Refers to the frame that is an immediate parent in the frame hierarchy.
· opener : Refers to the window from where the HTML page was opened.

location Object :→

This object represents the URL loaded into the window. It is used to navigate to a page in a local site or to any other page in the internet. By changing the property of this object, the document loaded in the window can be changed.

Properties :- window.location!

· href → Redirects the page to this property's value.
· host → IP address of host.
· hostname → Name of Host
· Port → Port of Host as in URL
· pathname → pathname of current document.
· protocol → Protocol of current document.

Method : Window.location.replace("http://www.gmail.com")
↓
Redirects to the page to www.gmail.com

window.location.replace method used to the checkbox.
another without displaying URL on the browser.
The URL accessed using this method is not available in the history sequence.

The History Object: Almost similiar to the Location Object and keeps track of the sites visited in the current browser session. The methods available in this object are as follows:

- forward(): Redirects the user to the site that is next in the history sequence. If nothing else is present in the list, the user stays on the same page.
- back(): Redirects the user to the previous site in history sequence. If nothing else is present in the list, user stays on the same page.
- go(number): Redirects the user to the site i.e. at the nth position (n stands for number) in history sequence. If nothing else in list, user stays on the same page.

The Document Object: This object represents the elements available within the visible region of the browser. The properties are as follows:

- fgcolor: Foreground color of the object.
- cookie: It is a harmless data related to the site stored on the client M/c.
- bgcolor: Rep. background color of the object.
- anchor: Rep. array of anchors in the document.
- alinkColor: Rep. color of the links in document.
- vlinkColor: " " " visited links in document.
- title: " title of the document.

4.4.6

Methods

- open( ) : It opens the stream to the Document Object. so that text can be written into it.
- write( ) : To write the text content to the Document object.
- close( ) : Used to close the stream currently open to the Document Object.
- clear ( ) : clear the contents of the Document Object.

1. The Anchor Object : The String object can represent the Anchor object too. The Anchor object does not have any properties or methods defined in it.

The anchor object can be accessed with the help of the Anchors array.

2 → The Link Object : This object works exactly like the Anchor object. All the links are stored in the links array, which can be accessed using an index. document.links.length will give the total number of links present in specific document.

The Image object : This object represents the images appearing in the documents. We can manipulate the properties of the images using this object. All the images inside a particular document are stored in the images array.

document.images[0].src = "first.jpg"

The image object has two important properties viz. name & src. There are other prop. such as width & height. Events are onAbort, OnKeyPress, onKeyDown, OnKeyUp

Method : Window ... to the page to www.gmo

3. Form Object: It represents the HTML form. The forms can be accessed through the forms Array. A particular form can be accessed as follows:

```
document.forms("Myform")
document.forms(0)
```

Property of form object : name i.e. name of the form.

We can access the FormField object is as follows

window.document.formname.elementname.property

1.4. Form Field Object

- Button
- Submit
- Reset
- TextBox
- TextArea
- Hidden
- Password
- Radio
- Checkbox
- Select

Prop. of Form Field Objects:

4.1 TextArea, Text, Password :

- defaultvalue : Rep. the value of form-field when the page is loaded.
- value : Rep. current value of form field.
- name : Rep. the name of Formfield.
- type : Rep. the type attribute of the form field.
- form : Rep. the form holding the form field.

4.4.2 Hidden, Button, Submit & Reset :

Hidden can also hold text. Button, submit & Re... accept Events and are mostly used for event Handling. These have following properties

- .value : Current value of form field.
- .name : name of the form field
- .type : type of the form field.
- .form : Rep. form holding the form field

1.4.3 Radio & Checkbox : These form field objects are used for taking inputs from the User as selections.

Properties :

- .Value : Current value of form field.
- .name : name of form field.
- .type : type of " ".
- .form : Rep. form holding the form
- .Checked : Used to set the checked attribute to true or false.
- .defaultChecked : Reflects the checked attribute.

4.4.4 Select : Properties

- .name, type, form
- .length : Rep. No. of options in the selection list
- .options : Rep. An array of options in a Select object.
- .selectedIndex : Rep. the index of selected item.

HTML form. The forms

The Navigator Object : This rep. the browser being used on client machine. This object has properties & methods that can detect Browser type, version & User Agent.

navigator.appName : Name of Browser
navigator.appVersion : Version of "
navigator.userAgent : Returns UserAgent value of HTTP
navigator.platform : Rep. the System's H/w & S/w Configuration.

5. The Frame Object : Rep. No. of frames in currently open web page in the browser window.

<Input type="Hidden" value="xyz">

Get

```
<html>
<head>
<titil>
</title></head>
<body>
<form name="form1" method=get action=Colorsevlet>
<B> Color </B>
<select name="color" size="1">
<option value="Red"> Red </option>
"Blue"> Blue
</select>
<input type=Submit value="submit">
</form> </select> </body> </html>
```

Color servlet.java

```
import java.io.*
import javax.servlet.*;
import java.servlet.http.*;
public class colorGetServlet extends HttpServlet
{
public void doGet (HttpServletReq.req, HttpServletRes.res)
throws servletException, IOException
{
String color = request.getParameter("color");
response.getContentType("Text/HTML")
PrintWrite pw = response.getwriter
pw.println("<B> The selected color is">
pw.println(color);
pw.close();
}}
```

An XHTML Example

The example below shows an XHTML document with a minimum of required tags:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<title>Title of document</title>
</head>

<body>
</body>

</html>
```

Note: The xmlns attribute in <html>, specifies the xml namespace for a document, and is required in XHTML documents.

XHTML HowTo

The following steps shows how a website can be converted from HTML to XHTML in 6 simple steps:

1. Add a <!DOCTYPE>

Add an XHTML <!DOCTYPE> to the first line of every page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Tip: Your pages must have a DOCTYPE declaration if you want them to validate as XHTML.

2. Add an xmlns Attribute

Add an xmlns attribute to the html element of every page:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Note: The xmlns attribute specifies the xml namespace for a document, and is required in XHTML documents.

3. Change Tags And Attribute Names to Lowercase

A general "find-and-replace" function was executed to replace all uppercase tags with lowercase tags. The same was done for attribute names.

STEP 4: code the corresponding event handler before adding header file <AfxInet.h>. as shown below:

```
#include "stdafx.h"
#include "ftp.h"
#include "ftpDlg.h"
#include <AfxInet.h>

void CFtpDlg::OnButton1()
{
        CInternetSession* pInternetSession;
        pInternetSession=new CInternetSession();
        if (!pInternetSession)
        {
                AfxMessageBox("couldn't establihed the internet
connection",MB_OK);
                return;
        }
        CFtpConnection* pFtpConnection;
        pFtpConnection=pInternetSession-
>GetFtpConnection(CString("ftp.microsoft.com"));
        if(!pFtpConnection)
        {
        AfxMessageBox("not establised tp connection",MB_OK);
        return;
        }
        else
        {
                m_t1="DOWNLODING.....";
                UpdateData(false);
                pFtpConnection-
>GetFile(CString("desclaimer.txt"),CString("desclaimer.txt"));
                pFtpConnection->Close();
                pInternetSession->Close();
                m_t1="DOWNLODING COMPLETE";
                UpdateData(false);
                }
}
```

STEP 5: now execute the program.

Unit-2 (II)

1 Introduction

JavaScript Origins

JavaScript was released by Netscape and Sun Microsystems in 1995. However, JavaScript is not the same thing as Java.

What is JavaScript

- It is a programming language.
- It is an interpreted language.
- It is object-based programming.
- It is widely used and supported
- It is accessible to the beginner.

Uses of JavaScript

- Use it to add multimedia elements.

  With JavaScript you can show, hide, change, resize images, and create image rollovers. You can create scrolling text across the status bar.

- Create pages dynamically

  Based on the user's choices, the date, or other external data, JavaScript can produce pages that are customized to the user.

- Interact with the user

  It can do some processing of forms and can validate user input when the user submits the form.

//PROGRAM:7//

//PROGRAM TO IMPLEMENT STACK OPERATIONS USING ARRAY//

```
#include<stdio.h>
#include<conio.h>
int stack[10],top=-1;
void push();
void pop();
void display();
void main()
{
int ch;
clrscr();
printf(">>>STACK OPERATION<<<\n\n");
do{
printf("\n 1.PUSH");
printf("\n 2.POP");
printf("\n 3.DISPLAY");
printf("\n 4.EXIT");
printf("\n ENTER YOUR CHOICE:");
scanf("%d",&ch);
if(ch==1)
push();
else if(ch==2)
pop();
else if(ch==3)
display();
else if(ch==4)
printf("\n END OF PROGRAM");
}while(ch!=4);
}
void push()
{
if(top>9)
printf("\n STACK OVERFLOW");
else
{
top=top+1;
printf("\nENTER A VALUE:");
scanf("%d",&stack[top]);
}
}
```

//SNEHA SHUBHAM//
//12/ECE/1937//

Writing JavaScript

JavaScript code is typically embedded in the HTML, to be interpreted and run by the client's browser. Here are some tips to remember when writing JavaScript commands.

- JavaScript code is case sensitive
- White space between words and tabs are ignored
- Line breaks are ignored except within a statement
- JavaScript statements end with a semi- colon ;

The SCRIPT Tag

The <SCRIPT> tag alerts a browser that JavaScript code follows. It is typically embedded in the HTML.

```
<SCRIPT language = "JavaScript">
statements
</SCRIPT>
```

SCRIPT Example

- Open "script_tag.html" in a browser.
- View the Source
- Put the cursor after <! – Enter code below → and enter the following:

```
<SCRIPT language = "JavaScript">
alert("Welcome to the script tag test page.")
</SCRIPT>
```

- Save the changes by choosing Save from the File menu.
- Then Refresh the browser by clicking the Refresh or Reload button.

Implementing JavaScript

There are three ways to add JavaScript commands to your Web Pages.

- Embedding code
- Inline code
- External file

External File

You can use the SRC attribute of the <SCRIPT> tag to call JavaScript code from an external text file. This is useful if you have a lot of code or you want to run it from several pages, because any number of pages can call the same external JavaScript file. The text file itself contains no HTML tags. It is call by the following tag:

```
<SCRIPT SRC="filename.js">
</SCRIPT>
```

External Example

- Open "external.html" in a browser
- View the Source
- Put the cursor after <! – Enter code here → and enter:
  ```
  <SCRIPT language = "JavaScript" SRC = "external.js">
  </SCRIPT>
  ```
- Save the changes and Refresh the browser.

//PROGRAM:6//

//PROGRAM FOR BINARY SEARCH//

```
#include<stdio.h>
#include<conio.h>
int main()
{
int a,first,last,middle,n,search,array[100];
clrscr();
printf("Enter the number of elements\n");
scanf("%d",&n);
printf("Enter integers \n");
for(a=0;a<n;a++)
scanf("%d",&array[a]);
printf("Enter the value to be found\n");
scanf("%d",&search);
first=0;
last=n-1;
middle=(first+last)/2;
while(first<=last)
{
if(array[middle]<search)
first=middle+1;
else if(array[middle]==search)
{
printf("%d found at location %d\n",search,middle+1);
break;
}
else
last=middle-1;
middle=(first+last)/2;
}
if(first>last)
printf("%d is not present in the list\n",search);
return (0);
}
```

//SNEHA SHUBHAM//
//12/ECE/1937//

Example

```
<html>
<head>
<script type="text/javascript">
function getCookie(c_name)
{
if (document.cookie.length>0)
 {
 c_start=document.cookie.indexOf(c_name + "=");
 if (c_start!=-1)
 {
 c_start=c_start + c_name.length+1;
 c_end=document.cookie.indexOf(";",c_start);
 if (c_end==-1) c_end=document.cookie.length;
 return unescape(document.cookie.substring(c_start,c_end));
 }
 }
return "";
}
function setCookie(c_name,value,expiredays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
}
function checkCookie()
{
username=getCookie('username');
if (username!=null && username!="")
```



```
 alert('Welcome again ');
 }
else
 {
 username=prompt('Please enter your name:',"");
 if (username!=null && username!="")
 {
setCookie('username',username,365);
 }
 }
 }
 </script>
 </head>
 <body onload="checkCookie()">
 </body>
```

4.12 Features of JavaScript

The following are features of the JavaScript.

1. **Imperative and structured :** JavaScript supports all the structured programming syntax in C (e.g., if statements, while loops, switch statements, etc.). One partial exception is scoping: C-style block-level scoping is not supported (instead, JavaScript has function-level scoping). JavaScript 1.7, however, supports block-level scoping with the let keyword. Like C, JavaScript makes a distinction between expressions and statements. One syntactic difference from C is automatic semicolon insertion, in which the semicolons that terminate statements can be omitted.
2. **Dynamic :** As in most scripting languages, types are associated with values not variables. For example, a variable x could b[illegible]ound to a number, then later rebound to a string. JavaScript supports var[illegible]ways to test the type of an object, including duck typing. JavaScript is a[illegible]s entirely object-based JavaScript objects are associative arrays, augmented with prototypes (see below). Object property names are string keys: so x = 10 and obj["x"] = 10

are equivalent, the dot notation being syntactic sugar. Properties and their values can be added, changed, or deleted at run-time. Most properties of an object (and those on its prototype inheritance chain) can be enumerated using a for...in loop. JavaScript has a small number of built-in objects such as Function and Date. JavaScript includes an eval function that can execute statements provided as strings at run-time.

3. **Functional :** Functions are first-class; they are objects themselves. As such, they have properties and can be passed around and interacted with like any other object.

   Inner functions (functions defined within other functions) are created each time the outer function is invoked, and variables of the outer functions for that invocation continue to exist as long as the inner functions still exist, even after that invocation is finished (e.g. if the inner function was returned, it still has access to the outer function's variables) - this is the mechanism behind closures within JavaScript.

4. **Prototype-based :** JavaScript uses prototypes instead of classes for inheritance. It is possible to simulate many class-based features with prototypes in JavaScript.

   Functions double as object constructors along with their typical role. Prefixing a function call with new creates a new object and calls that function with its local this keyword bound to that object for that invocation. The constructor's prototype property determines the object used for the new object's internal prototype. JavaScript's built-in constructors, such as Array, also have prototypes that can be modified.

   Unlike many object-oriented languages, there is no distinction between a function definition and a method definition. Rather, the distinction occurs during function calling; a function can be called as a method. When a function is called as a method of an object, the function's local this keyword is bound to that object for that invocation.

5. **Vendor-specific extensions :** JavaScript is officially managed by Mozilla, and new language features are added periodically. However, only some non-Mozilla JavaScript engines support these new features:

   - property getter and setter functions (also supported by WebKit, Opera, ActionScript, and Rhino)
   - conditional catch clauses
   - iterator protocol adopted from Python
   - shallow generators/coroutines also adopted from Python
   - array comprehensions and generator expressions also adopted from Python
   - proper block scope via new let keyword
   - array and object destructuring (limited form of pattern matching)

6. **Security :** JavaScript and the DOM provide the potential for malicious authors to deliver scripts to run on a client computer via the web. Browser authors contain this risk using two restrictions. First, scripts run in a sandbox in which they can only perform web-related actions, not general-purpose programming tasks like creating files. Second, scripts are constrained by the same origin policy: scripts from one web site do not have access to information such as usernames, passwords, or cookies sent to another site. Most JavaScript-related security bugs are breaches of either the same origin policy or the sandbox.

4.13 The Common Gateway Interface (CGI)

The common gateway interface, or CGI is a simple protocol that allows web pages to communicate with web-server-based programs. CGI's function is to start a web server-based program, then pass and receive environments variables to and from it. An environment variables is part of an operating system, not just part of a function or a program, as are JavaScript variables. An example of a CGI environment variable is server-name, which contains the domain name or IP address of a web server. A web server based application that process CGI environment variable is called CGI script or CGI program and can be designed to perform a multiple of functions. The CGI protocol's primary purpose is to send the data received from a web page to a program on a server, then to send any response can be database program or some type of custom applications.

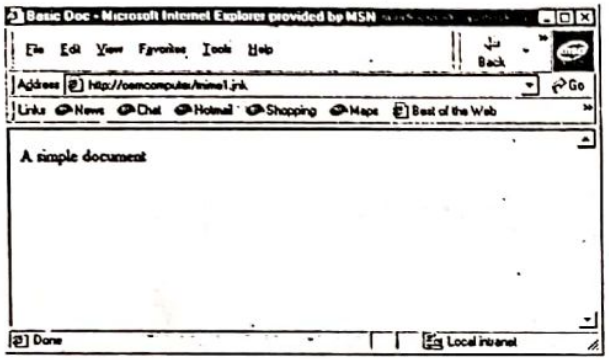
CGI scripts are used to process information entered into HTML forms. HTML forms element are used to pass information that is entered into a form to CGI environment variables. There CGI environment variables are then sent to a CGI Script on a web server. The CGI Script performs some sorting action, such as a query and either send a response back to the requesting web page or generate a new HTML document.

getElementById(P1).Value;

Unit-2 (II) CSS Chapter

7. Here the browser does not recognize the mime type, and so asks me what application I would like to view it in.

8. What happens though if I send the junk file over a server? The answer to that depends on the server. Using Microsoft's Personal Web Server here is what we see when we send the junk file `mime1.jnk` to our client:

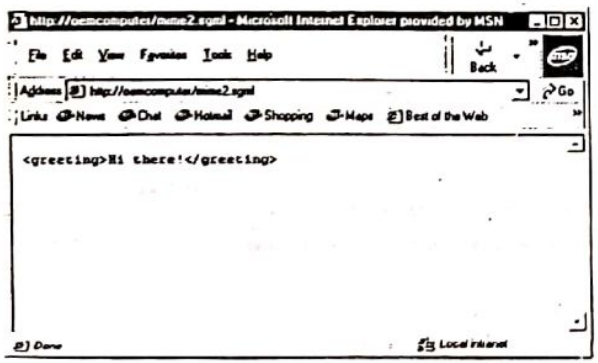


9. The PWS server is clever: it peeks inside the file, sees the root element is `<html>` and delivers the document as `text/html`.

To see the default mechanism in operation we can use the following file:

```
<greeting>Hi there!</greeting>
```

and save it as `mime2.sgml` (or `mime2.jnk`). Now when the server sends it to our client, we see the following:



This is the expected behavior: PWS did not recognize the suffix `.sgml`, and so delivered the file as a `text/plain` mime type.

You can find out more about mime types at `http://www.irvine.com/mime/`

Summary

In this chapter we looked at the data that describes the document and its contents. This is known as meta-data.

We looked at how to use the `<meta>` and `<title>` tags to define a documents content, and also had a brief look at how this could be done by using the emerging RDF syntax. These can be used by 'spiders', programs that automatically 'crawl the web' to compile search engine data (such as that from Google).

The `<link>` and `<base>` tags are used in linking to other documents, such as an external style sheet; the `<base>` tag is used to turn relative links into absolute ones.

We then showed that a namespace is simply the name given to the list of all the elements in a document, and showed how a namespace declaration is used in an XHTML document. The use of other namespaces (such as Dublin Core) was also shown.

We then closed the chapter with a discussion of HTTP headers and mime types.

12 Page 240-241

CSS

Disadvantage

Internal → load again & again
→ single page

9

Style Sheets for the Web

If you have used a computer for any kind of desktop publishing no doubt you will have come across the idea of a style sheet or document template. In its simplest form, a document template consists of a number of styles such as Normal, Heading1, Heading2, and so on. As you type you can apply one of the available styles to the text so that headings assume the correct font size and weight, paragraphs take on the correct margins, and images have the appropriate white space around them.

Style sheets for XHTML work on similar – but not identical – principles. In this chapter we will learn how to compose a simple style sheet for an XHTML document in a language called CSS (Cascading Style Sheets). Using this, we can control the size and colour of text on the screen, define the preferred font for the different components of a document, set their background color or image and control the amount of 'white space' inserted around an element.

In this chapter, we will cover the following:

- The advantages of using a style sheet for your XHTML document
- The CSS style sheet language
- Control of text color (including the text of hypertext links)
- Specifying the background color for the whole or just part of a document
- Specifying the font for a given paragraph, heading or other part of a document
- The control of 'white space' through style sheets
- How to apply a named style with the `class` attribute

inging Style to the Web

Before Style Sheets

In the beginning, HTML offered no control over style at all but was merely for marking up the components of a document in terms of their *function* in the sense of the role they played in the document: heading, list, paragraph, and so on. The way that HTML concentrated on functional aspects of a document of HTML was to its advantage, as it enabled documents to be viewed on all kinds of platforms, from Unix and dumb terminals to PCs and Macintoshes. The idea was that the software at the receiving end – in other the words the browser – could use the HTML markup to see which part of the document performed which function. It could then use its own mechanisms to render the text on the screen as it saw fit. The same document could thus be viewed on all kinds of machines on the 'Net, an idea which brought about the success of the Web.

But people have a natural interest in the look of their documents and, before long, new HTML tags were being added to control aspects of style. During the mid-nineties the Mosaic browser, and later Netscape Navigator, added some simple HTML tags and attributes to control this aspect of web documents, departing from the idea that mark-up was simply about control of document structure. These new tags were instantly popular with everyone apart from the academic purists, who saw them as contaminating the language. "After all," they claimed, "HTML is all about structured documents, and style has nothing to do with structure." They certainly had a point. However, the more popular the Web became, the more the designers wanted control over presentation. Before long all kinds of tricks were in use to make HTML documents behave as they wanted. A classic of its time was the small transparent GIF image, which could be used as a 'spacer' to push text aside, create artificial margins, or to vertically space lines.

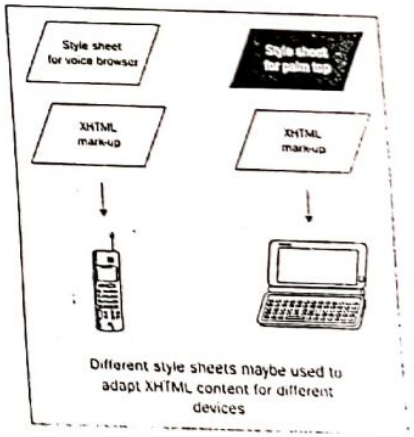
The `<table>` element was even more amenable when it came to assisting with the layout of web pages, with the table cells used to position images and text to give the effect of a multi-column layout. Meanwhile, when it came to headings, designers could hardly be content with the idea that the browser would choose the font for them, and so began to use images for these rather than HTML's `<h1>` - `<h6>` elements. The end result of all these tricks was to greatly complicate the markup, with a page's structure and styling becoming intertwined such that one couldn't be separated from the other.

Style Sheets to the Rescue

In late 1995 there was a meeting of the World Wide Web Consortium and other interested parties to discuss a solution to all these problems: style sheets for the Web. Using a special language, it was advocated that soon everyone would be using style sheets as a separate 'layer' to specify presentational aspects of documents. The rogue HTML tags and tricks employed by web designers would finally be redundant. Although it has taken longer than expected, the web style sheets are finally with us and are widely implemented on all major browsers.

The Advantages of Using Style Sheets

- **You can change the whole look and feel of your document with a few easy edits.** This aspect relates to the process of separating the functional part of the document (the XHTML) from the presentational aspects (the style sheet). When you use Word or another desktop publishing package you use a style sheet or template to present the text properly on the page. If you do this efficiently and in an organized manner then you can attach a new template and re-style the document without fuss. The same principle lies behind the efficient use of a web style sheet. If you structure a document with XHTML and then superimpose a style sheet to specify the presentation then you can easily re-style the whole text by editing the style sheet or swapping it for another one. On the other hand if you sprinkle the text liberally with `<font>` tags, `<br />` (line-break) tags and non-breaking spaces to create the layout, then changing the overall style will be much more tricky and time consuming.
- **The same style sheet can apply to several documents.** You can re-use the style sheet to get the same 'look and feel' for a group of documents; this is extremely useful.
- **The markup is neater and the content is more maintainable.** Nowadays, web pages are often very complex, so much so that if you look at their source it's almost impossible to understand. Instead of looking like a document with tags to structure its components, the page looks more like part of a computer program. This was not the intention of the original designers of the Web. Style sheets free the markup from the tremendous amount of clutter in the form of `<font>` tags, `<br />` tags, GIF images for headings, and other tags controlling presentation. This makes the document easier to mark up, read and maintain.
- **Style sheets make it possible to gear the text to a range of devices rather than just desktops.** With CSS you can make stylistic control a separate process. This approach of keeping structure and presentation apart has tremendous benefits when it comes to fine-tuning web content for different devices such as mobile phones, television, palm-top computers, and so on (this is discussed fully in Chapter 13). For display on a color computer screen, for example, a style sheet might concentrate on color and layout. The style sheet for rendering the very same document into speech (useful if you are accessing via a car phone, for example) would instead focus on the pitch of the voice, the volume, and so on:



Different style sheets maybe used to adapt XHTML content for different devices

- **Documents are faster to download with style sheets.** Lots of people use GIF images for headings and other text which they want to appear large on the screen. These images take some time to download, particularly on slow connections to the Internet. A style sheet gives you a much simpler and faster way of generating headings in a variety of fonts. That said, you do not have quite the same control over the exact look of the headings as you might were you to use a GIF, especially if you want to use an unusual font.

The Cascading Style Sheet (CSS) Language

The most popular language for writing style sheets for XHTML documents is called CSS – the Cascading Style Sheet language. This language was invented by Bert Bos and Håkon Lie at the World Wide Web Consortium with contributions from Chris Lilley, Dave Raggett and many others. The idea was to design a style sheet language which would give authors great flexibility when it came to controlling layout and yet, at the same time, would be easy enough to write that even non-programmers could design their own style sheets for the Web. The idea was that either CSS could be incorporated in the HTML itself – by using special elements and attributes for the purpose – or it could be the content of a completely separate file. This latter arrangement involves linking the style sheet to the document, the style sheet file taking on a different file extension.

Future Directions

CSS has now been through a number of stages of development. The initial specification, CSS1, released in 1996, was primarily concerned with relatively simple style sheet functions such as color, font, background image, and so on. Following on from this, CSS2 (1998) included further and more sophisticated features. These included features associated with page-based layout (for printing), support for downloadable fonts, and the definition of rectangular regions for displaying different parts of documents, giving freedom to lay out documents without resorting to using HTML tables as a means to position items on the screen.

Work on CSS3 has now begun and should be completed during the year 2000. The new version will bring with it many new features organised as modules; some examples of *possible* modules are listed below:

- Values and units
- Aural style sheets
- Columns
- Text flow and related topics
- Paged media
- Vertical text and Ruby
- Color
- Numbering and lists
- Backgrounds
- Text and fonts

CSS3 will, in effect, have all of the features that users expect from a desktop publishing environment, as well as a range of features especially suited to the context of an international and multimedia web.

The XSL Language

This chapter covers the use of the CSS language for styling XHTML documents because this is the most common way to date, and a method suitable for the non-programmer to understand. An alternative style sheet language for programmers for styling XML documents currently being developed is the eXtensible Stylesheet Language or XSL language. Aimed, by and large, at complex documentation projects, XSL has many uses associated with the automatic generation of tables of contents, indexes, reports and other more complex publishing tasks, especially printing.

Support for CSS

For a browser to understand what to do when it finds a CSS style sheet attached to a document it needs not only to understand the syntax of the language itself, but also to know how to render the text and images in the appropriate way on the screen. Support for CSS is now reliable for a number of features but for other features it is far from perfect. The best place to find out which features are supported and which are not is `http://webreview.com/pub/guides/style/style.html`.

The WebReview site is highly recommended. It includes three up-to-date and very useful lists to which all authors should refer. These are as follows:

The Master List
This is the mother of all CSS charts, listing every aspect of the CSS specification and identifying how well it is supported by Netscape 4, Internet Explorer 3 and Internet Explorer 4 for both Mac and Windows 95.

The Safe List
This is the list you'll refer to again and again – every CSS property that is completely or partially supported by all browsers.

The Danger List
This is the list of properties to stay away from – the ones for which support is nonexistent, buggy or quirky.

Why Hasn't Support for CSS Been Better?

When Microsoft's Internet Explorer 3 was released, implementation of CSS was at best experimental. Indeed IE 3 went out with a certain amount of support for CSS even before the specification for CSS1 was finished. Needless to say, the resulting implementation did not tie in with the specification.

Netscape responded by adding a way of setting style properties via JavaScript. However, this was neither thorough nor consistent in its handling of stylistic features. Microsoft then greatly improved their support for style sheets in Internet Explorer 4 although still failed to do a lot of the basics. Netscape grudgingly added some support for CSS syntax to Navigator 4. There were, however, many bugs which meant that a number of properties simply couldn't be used. Microsoft, upon seeing how little real work had been done by Netscape, did not bother in its IE 5 release to patch up or even implement CSS1 properly, let alone CSS 2.

er Times to Come

All in all, then, support for style sheets in the past has been rather disappointing. However, all is changing. Enter now a new company in Norway called OperaSoft who have been diligently working away producing a lightweight browser that fully implements the CSS1 specification (see `http://www.opera.com/index.html`). This has taken them a long time and we hope that by the time this book goes to press that you will be able to benefit from their excellent work. Meanwhile Netscape has gone open source with the Mozilla project and we can expect a very much more thorough implementation of CSS in the next release of Navigator (`http://www.netscape.com`). Furthermore, you can expect to see good support for CSS from other newcomers into the browser market, such as Hewlett Packard's ChaiFarer XML web browser (`http://www.internetsolutions.enterprise.hp.com/chai/chai_farer.html`).

How CSS Style Sheets Work

When an XHTML document is displayed by a browser, ordinarily the browser defaults to using its own ideas about how the text and images should be laid out on the screen. So, for example, one browser might choose Times Roman to display the text for paragraph elements, while another might instead default to a sans serif font such as Arial. What a style sheet does is to tell the browser more specifically how to render the document whether on screen, on paper, or even to voice.

Formatting Objects

The diagram below shows a very simple document in terms of its component parts with heading elements, list elements, paragraph elements and, within these, images and hypertext links. A paragraph, a heading, or a list: these are all examples of formatting objects. These are 'objects' from the perspective of the software that displays the document. Text is said to flow within a formatting object.

Why 'Cascading' Style Sheets?

The name 'Cascading Style Sheets' implies that more than one style sheet can interact together (or merge) together to produce the final look of the document: individual style sheets from different sources can be combined. CSS allows you to write a short style sheet to override a few properties in the browser's built-in style sheet.

Inheritance of Style Sheet Properties

One of the most important aspects of CSS is the way that properties are inherited from one element to another. For instance, suppose you have a paragraph which has been given CSS properties which say that it should be displayed as blue text and in italic. Any elements contained within that paragraph – its child elements – will inherit these properties from their parent paragraph. Bold text within the paragraph will still be blue and italic and, of course, bold. Similarly, any text within the paragraph which has been marked up as `<strong>` will still be blue.

Navigator 4's support for inheritance is not entirely reliable. In particular, it doesn't work well for tables or for elements appearing after a table. Workarounds include using style rules to specify the font family *for <p> and other tags.*

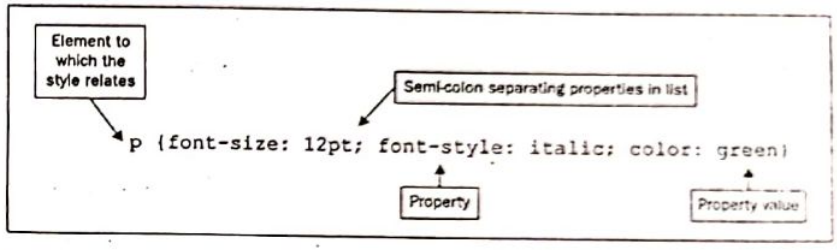
How to Write CSS

Here is a simple example of a stylistic 'rule' written in CSS:

```
h1 {font-size: 12pt; font-weight: bold; color: red;}
```

This example simply states that Heading 1 elements should be 12-point bold red text. We look more at the use of CSS to make text a particular color in more detail during the course of this chapter.

You can see that the syntax for the stylistic rule is relatively straightforward. The element to which the style applies is given first and a property list specifying the style then follows afterwards. The diagram below illustrates this with another example:



The property list is enclosed in curly brackets and successive properties are separated by a semi-colon. The properties themselves consist of a property name and a value, separated by a colon.

It doesn't matter how you space out the rules or where you place new lines. Also, CSS rules are case-insensitive. That means that the following rules have exactly the same effect as each other:

```
p {
  font-size: 120%;
  font-style: italic;
}

p {font-size: 120%; font-style: italic;}

p {FONT-SIZE: 120%; font-style: ITALIC;}

p {font-size:120%;  font-style:italic;}
```

A complete style sheet consists of one or more stylistic rules contained in the `<style>` element, which itself is part of the document head. You can see the `<style>` element in use in the example below

If you want to give the same style to a whole lot of tags you can do this:

```
h2, h3, h4 (color:green;)
```

The effect will be that <h2>, <h3> and <h4> elements will be colored green. This shortcut saves having to write out three rules, one for each tag.

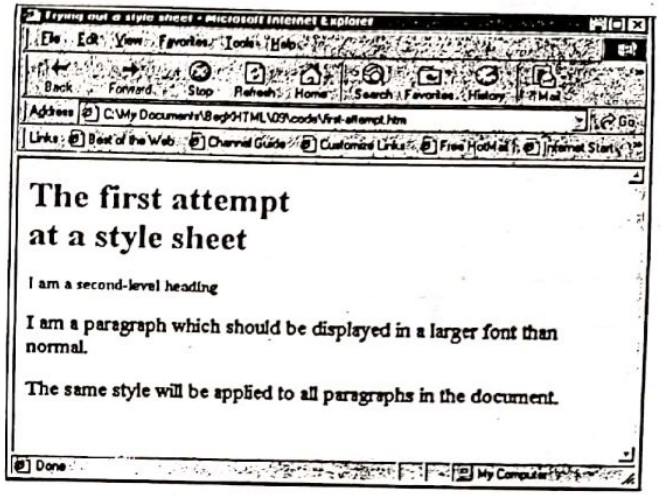
Try It Out – Writing a Simple Style Sheet

Here is a simple style sheet to try out.

1. Take this simple XHTML document and save as `first-attempt.htm`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Trying out a style sheet</title>
    <style type="text/css">
    <!--
      p (font-size:120%;)
      h1 (color:green; )
      h2 (color:green; font-size: 90%;)
      b ( color:red; )
    -->
    </style>
  </head>
  <body>
    <h1>The first attempt<br />
       <b>at a style sheet</b></h1>

    <h2>I am a second-level heading</h2>

    <p>I am a paragraph which should be displayed in a larger font than
normal.</p>

    <p>The same style will be applied to all paragraphs in the document.</p>
  </body>
</html>
```


2. Try out the document on your browser. The style sheet should affect the way the text is displayed and you should see the following or similar:



How It Works

The style sheet inserted in the document has had the effect of making the Heading 1 element green and the size of the Heading 2 element reduced slightly. Meanwhile, because the <b> element was given the color red, it changed the color of the main Heading 1 halfway through the text.

Note that we've enclosed the CSS in comments to hide it from older browsers which may try to render the contents to the screen as text if they don't understand style sheets.

Example of a Linked Style Sheet

Rather than include the style sheet inside the XHTML document, we can create a separate style sheet file and link our XHTML document to it. A separate style sheet file normally has the '.css' file extension. We then use the `<link>` element to attach that style sheet to our XHTML file. In the example above, we can separate the style sheet from the document quite easily.

Try It Out – Example of a Linked Style Sheet

1. Cut and paste the style sheet from the above XHTML document into a new text file, add the comment line to the top, and save the file as `first-link.css` (note that we do not need the `<style>` tags in an external style sheet):

```
/* This is the style sheet to be linked */
p {font-size:120%;}
h1 {color:green; }
h2 {color:green; font-size: 90%;}
b { color:red; }
```

2. Now delete the `<style>` elements from the XHTML document and add in the `<link>` statement, so that the document now looks just like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Trying out a style sheet</title>
    <link rel="stylesheet" href="first-link.css" />
  </head>
  <body>
    <h1>The first attempt<br />
    <b>at a style sheet</b></h1>

    <h2>I am a second-level heading</h2>

    <p>I am a paragraph which should be displayed in a larger font than
normal.</p>

    <p>The same style will be applied to all paragraphs in the document.</p>
  </body>
</html>
```

3. Save as `linking.htm` and display the file in your browser – you should see exactly the same screen as before.

How It Works

The critical line is:

```
<link rel="stylesheet" href="first-link.css" />
```


This simply informs the browser to style the document with the style information contained within the file given as the value of the `href` attribute. One advantage of this method is when, say, one department in a company makes various 'approved' styles which all company documents must follow. The author of the document then simply needs to include one line (the `<link>` statement) to turn their document from a plain XHTML document into one that has a company 'look and feel' to it.

Note that the style sheet does not have to be in the same folder as the XHTML document. For example, in such a company, a manager in one department could author an information document for new recruits and then link it to a style sheet stored in the Human Resources departmental folder as so:

```
<link rel="stylesheet" href="HRdept/new_staff.css" />
```

The path to the file can be as long as necessary. For example, the 'new staff' style information may be stored in a general information folder within the employees folder:

```
<link rel="stylesheet" href="HRdept/employees/gen_info/new_staff.css" />
```

However, you may wish to link to a style sheet stored on a totally different machine, or even one in a totally different country. For example, if you are in the USA but are writing a document for the UK-based branch of the company, you simply give the full URL of the location of the style file as the value of the `href` attribute:

```
<link rel="stylesheet" href="http://www.somecompany.com/UK.css" />
```

Note that this is similar to typing in the address (i.e. the full URL of the location) of an XHTML document into your browser, except that it is for a CSS document this time. For the rest of this chapter, though, we shall just stick to including the style information within the XHTML document for ease of readability.

Common Mistakes in Writing CSS

Here are some of the simple things to get wrong. Remember that writing CSS is a bit like programming in the sense that even a small error can throw the whole thing out and produce surprising results

- Check that you have put semicolons (';') between the properties in the list. This is very easily forgotten.
- Check that you have not forgotten the `</style>` end tag.
- Browsers are very sensitive to markup problems. This applies particularly to tables. The solution is to check that you have indeed written valid XHTML or use the HTMLTidy (Appendix B) program to tidy the markup.
- If you have specified a color by name, make sure that it really is one of the 16 color names allowed by CSS (these are shown later). If it isn't, then the text may be shown in black
- If you have specified a font, again check the spelling. Browsers are not in the slightest bit tolerant of spelling mistakes! Also, remember that font names of more than one word should be quoted and that quotes are not allowed anywhere else.

- On that note, check that you have spelt 'color' in the US-English way (not 'colour' with a 'u', which is the UK-English way).
- Is the problem with your browser rather than with you? Although in our examples we try to use CSS properties that work properly across platforms, it is quite possible that the property you are using does not work properly on your browser. This applies to those using Netscape Navigator 4 and Internet Explorer 3 or 4 – even IE 5 is far from perfect.

Local Styles with the 'style' Attribute

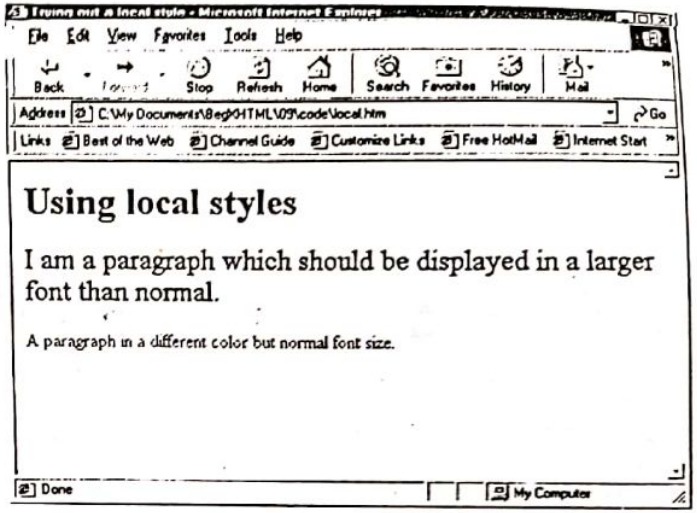
Sometimes you may not want to write a complete style sheet for a document, but you do need to change the style for a small number of elements on a 'one-off' basis. For example, perhaps there is a particular paragraph that you want displayed in a larger italic font. Or maybe a number of words within a paragraph need to be displayed in a different color to draw attention to them. Rather than write a complete style sheet for the document, this kind of styling can be accomplished by using a special attribute designed for the purpose, This is the `style` attribute. Do not make the common mistake of confusing the style *element* (for embedding a style sheet) with the style *attribute* (for defining a local style); the two are quite different.

Try It Out – Using Local Styles

1. Enter the following XHTML document into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Trying out a local style</title>
  </head>
  <body>
    <h1>Using local styles</h1>

    <p style="font-size: 150%">I am a paragraph which should be displayed in a
larger font than normal.</p>

    <p style="color: green">A paragraph in a different color but normal font
size.</p>
  </body>
</html>
```


2. Save the file as `local.htm` and then look at it in your browser. The result should be similar to the following:



How It Works

The `style` attribute has applied a CSS style to the paragraphs in question. In the first paragraph the style attribute was used to apply the `font-size` property and in the second paragraph it was used to apply the `color` property which changed the color of the text to green.

Using the 'style' Attribute with the Element

The `style` attribute can be used with many XHTML elements and you can apply local styles to headings, lists, block-quote elements, and so on. Very useful in this context is the element to indicate a phrase to receive a particular style. Look at the following:

```
<p>This text <span style="color: green">is positively green</span> with <span
style="color: green; font-size: 140%">green</span>
```

The element with the `style` attribute says in effect "start using this style here". Then when the end tag is encountered the style is switched off again. Even if the color for the text has been set by the browser default already, this will be overwritten by setting the color property on the element. You can set multiple properties in this way, for instance, like this:

```
<span style="color: green; font-size: 140%">green</span>
```

...the 'style' Attribute with the Element

...e `style` attribute also comes into its own with the `<div>` element. Suppose you want to give three ...aragraphs in succession a particular style as well as perhaps set the font to red text of a smaller size. The `<div>` element can be used with the `style` attribute like this to mark where the text style should change:

```
<div style="color:red; font-weight:80%">
```

and a closing `</div>` tag marks where the changed style should end.

Note About the 'style' Attribute

You may have worked out for yourself that the use of the `style` attribute removes most of the benefits of style sheets, namely that you can separate style from your document. It's useful for trying things out, because you can simply add the attribute in to most elements and see what effect it'll have. We'll see later how to use named styles to apply more specific styles using classes, and then you can apply those techniques to the `<span>` and `<div>` elements just like you would the `style` attribute. The difference is that the actual definition of the style remains in the style sheet, and you can thus alter it separately from your content.

How CSS Properties Inherit

As explained earlier in this chapter, CSS properties are passed down from one element to another. This example shows this idea in practice.

Try It Out – Showing Inheritance

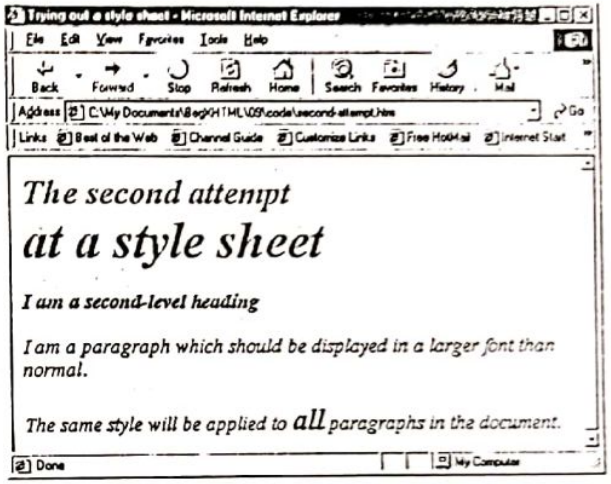
1. Enter this simple XHTML document into your text editor and save as `second-attempt.htm`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Trying out a style sheet</title>
    <style type="text/css">
    <!--
      body {font-style: italic}
      p {font-size:120%}
      h1 {color:green; font-size:200%;}
      h2 {color:green; font-size:120%;}
      b {font-size: 150%}
    -->
    </style>
  </head>
```



```
  <body>
    <h1>The second attempt<br />
    <b>at a style sheet</b></h1>
    <h2>I am a second-level heading</h2>
    <p>I am a paragraph which should be displayed in a larger font than
   normal.</p>
    <p>The same style will be applied to <b>all</b> paragraphs in the
   document.</p>
  </body>
</html>
```

2. Now view the XHTML document containing the style sheet in your browser. The result should be similar to the effect shown below:



How It Works

The inclusion of a style requirement on the `<body>` element has the effect of changing all the text in its descendants; in this case its children headings and paragraphs. If the body is given an italic font style, as in the example above, then all the text is shown italic.

You can also see the result of setting `<b>` (bold text) to 150%. The word 'all' is emboldened and is displayed as such. However it has also inherited the larger font size from its parent element `<p>`. The result? Very large text indeed!

Setting the Color of the Font

You will have seen in the preceding examples how CSS can be used to set the color of the text. This section looks at the use of the `color` property in more detail. The `color` property is very useful and can be applied to almost any element. This means that you can use it not only to set the text color for documents, but individual paragraphs, words and phrases can be given a color of their own.

The easiest way to specify a color is to use one of the standard key color names. These are the 16 colors taken from the Windows VGA palette and are:

aqua	black	blue	fuchsia
grey	green	lime	maroon
navy	olive	purple	red
teal	silver	white	yellow

If you want to specify a color not on this list then you need to use either an RGB value or hexadecimal value. The way this is done is explained in Appendix C. An alternative is to use coded values for colors, giving very precise control. These are used in later examples in this chapter.

Here are some simple examples of the use of the `color` property to set the color of XHTML elements:

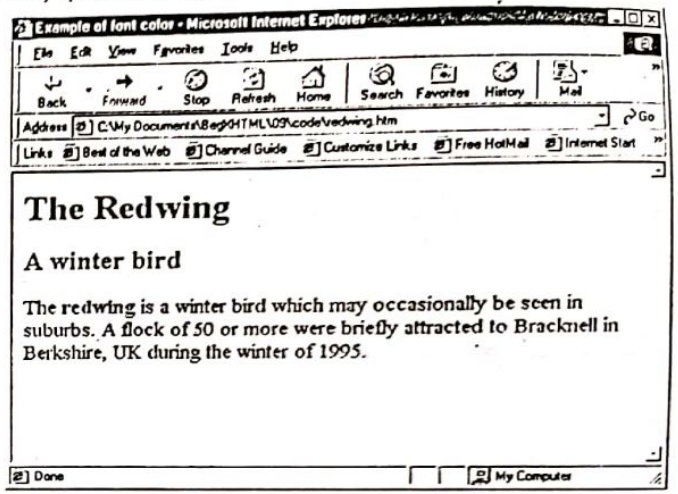
```
body {color: maroon}
  h1 {color: teal}
  h2 {color: olive}
```

Try It Out – Setting the Font Color

1. Type this simple XHTML document into your text editor:

```
<<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Example of font color</title>
    <style type="text/css">    <!--
      body {font-size: 120%; color: blue;}
      p {color:red}
    -->
    </style>
  </head>
  <body>
    <h1>The Redwing</h1>
    <h2>A winter bird</h2>

    <p>The <strong>redwing</strong> is a winter bird which may occasionally be
seen in suburbs. A flock of 50 or more were briefly attracted
to Bracknell in Berkshire, UK during the winter of 1995.</p>
  </body>
</html>
```


2. Save as `redwing.htm`, run the document through your browser and have a look at the results. The whole of the body text is set to blue text in a larger font. The paragraphs are separately specified in red:



How It Works

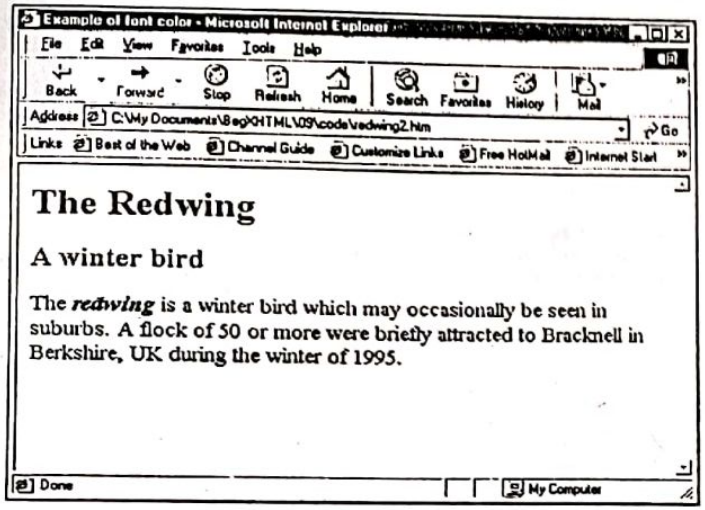
The `font-size` property (which you will come across later) sets the body to 120% larger than normal so all text should be slightly larger than normal. The `color` property is used to set the body of the document to blue which means that all the text might come out in this color. This would be the case were it not for the color property being used to set paragraph elements specifically to red.

Try It Out – Another Example of Font Color

1. Add the following line to the style section of the above document:

```
<style type="text/css">
<!--
   body { font-size: 120%; color: blue;}
   p {color:red}
   strong {color:black; font-style: italic;}
  -->
  </style>
```

Save as `redwing2.htm` and display it in your browser. You should see the following:



ow It Works

CSS properties can be applied to character-level elements (like <b>, and) as well as block elements (such as <p>, <h1> and <blockquote>). In this example, the element is set to black italic text and locally overrides the red text specified for paragraph elements.

sing 'Hex' Numbers to Specify the Color of Text Precisely

In Appendix C we explain that a common way of specifying color is to use a hexadecimal (or 'hex' for short) code. With CSS, this provides a way to be more precise about the color you want: you are no longer limited to the 16 named colors described earlier. When we specify the color as a hexadecimal value in CSS, we must prefix it with a # as in the example below.

y It Out – Using a Hex Code to Specify Color

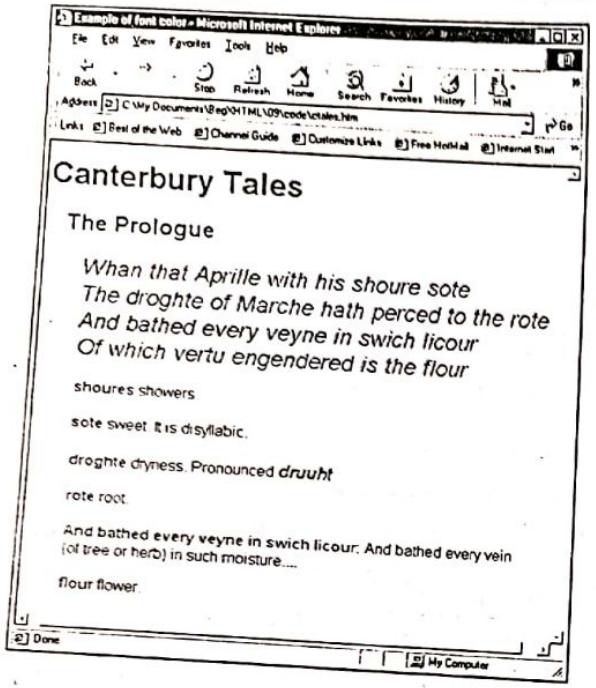
1. Enter the following into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Example of font color</title>
    <style type="text/css">
```



```
<!--
  body {
    margin-left: 10%;
    margin-right: 5%;
    font-family: Trebuchet, Arial, sans-serif
      }
  h1 { margin-left: -8% }
  h2 { margin-left: -4% }
  h2 { color: #006699 }
  em { font-style: italic; font-size:120%;}
  pre { font-style:italic; font-size:150%;}
  b { color:#006699 }
  pre { font-family: Trebuchet, Arial, sans-serif; color:#006699;}
-->
  </style>
</head>
<body>
  <h1>Canterbury Tales</h1>
  <h2>The Prologue</h2>
  <pre>Whan that Aprille with his shoure sote
  The droghte of Marche hath perced to the rote
  And bathed every veyne in swich licour
  Of which vertu engendered is the flour</pre>

  <p><b>shoures</b> showers. </p>

  <p><b>sote</b> sweet. It is disyllabic.</p>

  <p><b>droghte</b> dryness. Pronounced <em>druuht</em> </p>

  <p><b>rote </b>root.</p>

  <p><b>And bathed every veyne in swich licour</b>: And bathed every vein (of
tree or herb) in such moisture....</p>

  <p><b>flour</b> flower.</p>

  </body>
</html>
```

2. Save the file as `ctales.htm` and then run it through your browser. You should see something similar to the screenshot on the following page:



ow It Works

Heading 2 elements have been set to turquoise by giving them the color coded as #006699. The same color has been used for the bold type. Meanwhile emphasized text has been made larger than normal italic. The text of the poem itself is actually preformatted text but the font is specified as Arial (it would normally be Courier) and the size 150%. The `<pre>` element has also been set to italic by using the `<font-style>` property.

etting the Color of Hypertext Links

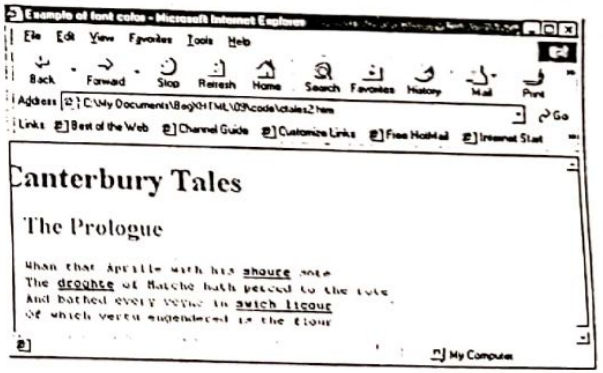
We now move onto the subject of setting the color of hypertext links. A short discussion of hypertext links in terms of page layout and design is given in Chapter 12. You have the option of setting the color of unvisited, visited and active links. An active link is simply a link which you have selected with the cursor. All anchor elements with an `href` attribute will be put into one (and only one)of these three groups and displayed in the appropriate way, as dictated by the style sheet.

Try It Out – Setting the Color of Hypertext Links

1. Type this simple XHTML document into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Example of font color</title>
    <style type="text/css">
    <!--
      h1 { margin-left: -4% }
      h2 { color: #990066 }
      h3 { color: black }
      h4 { color:#990066;}
      pre { color: #990066}
      em { color: #FF6633 }
      A:link { color: black }
      A:visited { color: blue }
      A:active { color: lime }
    -->
    </style>
  </head>
  <body>
    <h1>Canterbury Tales</h1>
    <h2>The Prologue</h2>
    <pre>Whan that Aprille with his <a href=xxx">shoure</a> sote
    The <a href="xxx">droghte</a> of Marche hath perced to the rote
    And bathed every veyne in <a href=yyy>swich licour</a>
    Of which vertu engendered is the flour</pre>
  </body>
</html>
```

2. Save as `ctales2.htm` and run in your browser. This is the result we obtained:



3. Try clicking on a link. As you press down with the cursor the link should go green. After you have used the link (visited it) it should be shown as blue. Of course, none of the links actually point to anything.

How It Works

The CSS `color` property has been associated with a particular kind of hypertext link. These are the critical lines to do the job:

```
A:link { color: black }
```

sets the color of unvisited links to black:

```
A:visited { color: blue }
```

sets the color of visited links to blue:

```
A:active { color: lime }
```

sets the color of active links to lime.

You can of course go further and make hypertext links look completely different from their default style. For example, some people like to remove the underline:

```
A:link {text-decoration: none}
```

Or you can add a background to the hypertext link:

```
A:link {background-color: black; color:yellow}
```

Alternative Method

You can also set the color of hypertext links using the <body> tag attributes. For example:

```
<body bgcolor="white" background="texture.jpg" text="black"
 link="navy" vlink="maroon" alink="red">
```

etting the Background Color

The useful thing about CSS is that it allows you to set the background color for individual parts of the document. Thus you can bring out a certain paragraph by setting the background color to a pastel shade, for example or give a heading special treatment by reversing out the text on a black background. You can also set the background color for the whole document. For example:

```
body { color: black; background: white; }
```


In this example, there are two properties in use: `color` which sets the color of the text, and `background` which sets the color of the page background. Although it is optional, we recommend always adding the semicolon after the last property.

Giving Paragraphs and Headings a Background Color

To give paragraphs, headings, lists and other XHTML elements a background color, simply apply the `background` property to the element in question. For example:

```
p { color: red; background: yellow; }
h2 { color: blue; background: lime; }
```

You can also use the `padding` property to extend the color outside the limits of the text area, as explained later in this chapter.

Setting a Background Texture

Background textures are accomplished by one of two methods. Either you can use attributes belonging to the XHTML <body> element, or you can use CSS. The latter method gives you more control, especially over the position of the background. Using the CSS method you can also decide to only give specific parts of the document a background, for example just a single paragraph or heading.

It is of course possible to create your own, but why not first try downloading a selection of background textures from the Web to play with. Go on-line, load your favorite browser and then search on the words 'backgrounds for Web pages'. A number of references to sites supplying (often free) background textures should come up. For example, Netscape provides a variety of free background images at `http://www.netscape.com/assist/net_sites/bg/ backgrounds.html`. In general it seems to be best to choose either a very light image and then overlay black or some other dark-colored text over it, or to choose a dark background and lay white or other very light text over it. Highly decorative, almost psychedelic backgrounds can make it very hard to read the text.

Try It Out – Inserting a Background Texture for Your Page Within XHTML

1. Type the following 'surreal' XHTML document into your text editor and save it as `haggis.htm`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Example of a textured background</title>
</head>
<body bgcolor="yellow" background="yellow_fabric.gif">
<h1>Episode 7: Trip to Another Dimension</h1>
<p> The mighty Captain Haggis shrugged off his attackers with his
famous ground-black-pepper sneeze. As the evil Cutlery Clan fell to the
floor coughing and sneezing and rubbing at their tearing eyes. Captain
Haggis quickly ate his Dimensional Neaps 'n' Tatties and plunged headlong
into the dreaded Mustard Dimension. </p>
```

```
<p> Does Captain Haggis know no fear? Will he be able to withstand the
scorching mustard heat? And will he be able to rescue the sacred
Three-Legged Haggis in time to save all of Haggisdom from the evil machinations of
the ruthless Cutlery Clan?...</p>

<p> Tune in to next weeks thrilling episode: 'A Mustard Been Here Before'.</p>
</body>
</html>
```

2. The result is a file with the yellow texture repeated across the background:

How It Works

The background attribute specifies the file containing the image to repeat across the background. We have also specified a background color too (using the `bgcolor` attribute) just in case the browser is unable to render the image. If the image you use is very long it will look as though it is merely repeated downwards. If you use a long thin image with a pattern on one side only, the effect is that a border is created down the page.

Using CSS to Position a Background Image

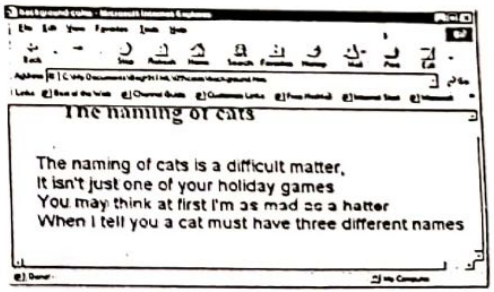
CSS can be used to set the background not only for the whole document, but for specific elements. For example, you can use CSS to set the background image for a heading element. In the next example, CSS is used to allocate a background image to a `<pre>` (preformatted text) element. A single image is used and the browser is told explicitly not to repeat this by using the `background-repeat` property. You can use any light image to demonstrate this for your self; we use the image from the previous example

Try It Out – Using CSS to Insert and Position a Background Image

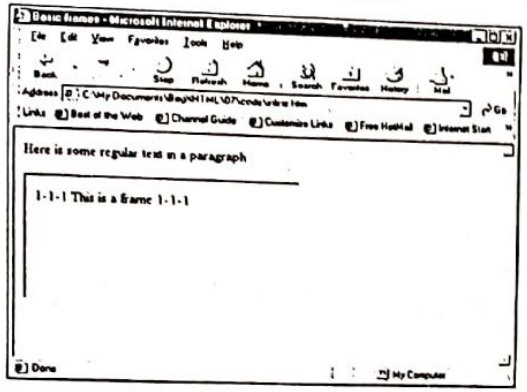
1. Take the following XHTML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>background color</title>
<style type="text/css">
<!--
pre {
       background-color: white;
       background-attachment: fixed;
       background-image: url(yellow_fabric.gif);
       background-position: top left;
       background-repeat: no-repeat;
       padding:2em
      }
  h1 { margin-left: 10% }
  h2 { color: #990066 }
pre { color:black; font-size: 150%; font-family:arial}
-->
</style>

</head>
<body>
<h1>The naming of cats</h1>

<pre>The naming of cats is a difficult matter,
It isn't just one of your holiday games
You may think at first I'm as mad as a hatter
When I tell you a cat must have three different names</pre>
</body>
</html>
```

2. Save the document as `background.htm` and look at it with your browser. What should happen is that the image associated with the `<pre>` element is inserted as a background texture, for example as follows:



2. Run this in your browser – here is what we see in IE5:



IE5 supports `<iframe>` so we see the inline frame; however, Netscape Navigator does not support this feature. There's not really a whole lot to say about iframes, so we'll move on, but it takes most of the same attributes as the `<frame>` element, with some slight differences (like not having to be part of a frameset document).

CSS and Frames

Style sheets can be applied to any of the pages that the frames contain. However applying styling to a frame has no effect what so ever to the pages it contains! In other words, styling is not inherited from the `<frame>` element. The only rudimentary styling that we can employ when using frames is the by using the `marginheight` and `marginwidth` attributes, which we discussed earlier. More on the Cascading Style Sheet (CSS) language can be found in Chapter 9.

The <noframes> Element

Should a browser not support frames then we can still get some kind of display by using the `<noframes>` element. As many devices such as voice readers and small screen devices will not support frames, we should provide an alternative display if we want our pages to reach this audience.

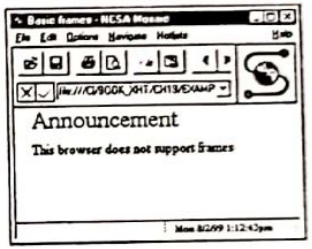
Here is an example of how to do this.

Try It Out – Using the <noframes> Element

1. Add the following into `basic.htm` (our first example) and save it as `noframes.htm`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Basic frames</title>
</head>
<frameset cols="*,*" rows="*,*">
  <frame src="text1.htm" />   <!-- The first frame in the list -->
  <frame src="text2.htm" />   <!-- The second frame in the list -->
  <frame src="text3.htm" />   <!-- The third frame in the list -->
  <frame src="text4.htm" />   <!-- The fourth frame in the list -->
<noframes>
  <h3>Announcement</h3>
  <p>This browser does not support frames</p>
</noframes>
</frameset>
</html>
```

2. Running it in your browser, you should see the same screen as we saw in our very first example. This is because your browser supports frames. However, here is how the above looks in Mosaic 2, which does not support frames:



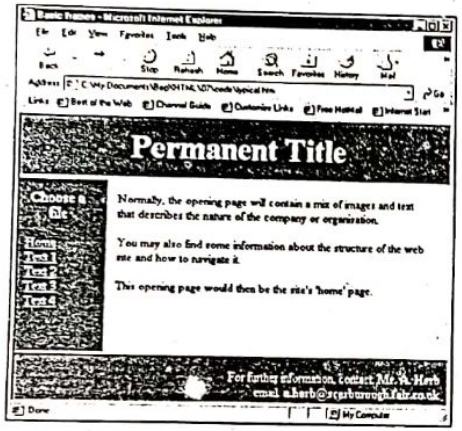
Using Frames for Display

Many sites use frames to great effect, Microsoft for example. Browsing the web will provide other examples. However, we thought that we would just end with an example of one of the more typical layouts. Type this into your editor and call it `typical.htm`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Basic frames</title>
</head>
<frameset rows="80,*,60" >
  <frame src="top1.htm" id="top_frame" name="top_frame" scrolling="no" />
  <frameset cols="20%,*" >
```

```
    <frame src="index1.htm" id="index_view" name="index_view" frameborder="no" />
    <frame src="start_text.htm" id="main_view" name="main_view" frameborder="no"
/>
  </frameset>
  <frame src="bottom1.htm" id="bottom_frame" name="bottom_frame" frameborder="1"
scrolling="no" />
 </frameset>
</html>
```

This example has a permanent header and footer. The left column frame consists of an index, and the right column frame provides a place to show the pages referred to in the index. The screenshot below provides an example of a typical layout:



The individual pages in the frames are styled using CSS, which we don't cover until Chapter 9. If you want the code for these pages, they are all available on the Wrox web site for this chapter; obviously, the names of the files are the values of the `src` attributes in the above code. However, you will be able to make files like these for yourself after you have read Chapter 9.

For a real-life example of a frames layout check out `http://www.ceram.co.uk/`.

Summary

In this chapter we had a look at using frames as a display device to show multiple pages on the same screen. We demonstrated how to do this using the `<frameset>` and `<frame>` element. Note that if you are using frames then you must use the Frameset DTD, and that the `<frameset>` element replaces the `<body>` element. We then looked at how to nest frames, and how to send a document to a particular frame using the `target` attribute. Inline frames were also looked at, and we finished with an example of a fairly typical frame layout design.

...w It Works

CSS has been used to associate a number of background properties with the <pre> element. Although the simplest option is to use the background property to set a background image, here more subtle properties (background-color, background-image, background-repeat, background-attachment and background-position) are used. Using these individual properties you can use CSS to position a background image and to control whether it 'tiles' the background or only appears once. Note that you have to choose an image with a size that can contain the text you wish to be displayed; we deliberately choose a small image here to illustrate that fact.

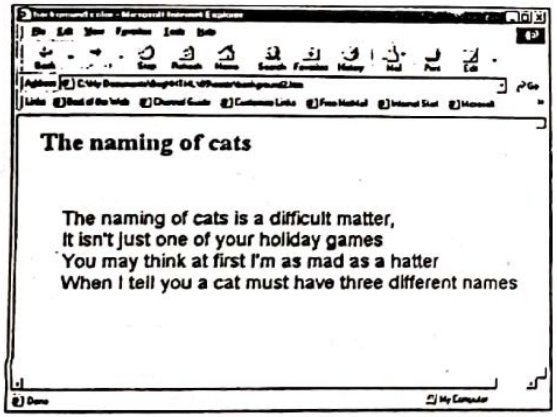
In our next example the image is repeated (or 'tiled') across the background, so we do not have to worry about the image size.

Try It Out – a Background Texture Repeated Behind Text

1. Change the style sheet of the previous document to:

```
<title>background color</title>
<style type="text/css">
<!--
pre {
      background-color: white;
      background-attachment: fixed;
      background-image: url(yellow_fabric.gif);
      background-position: top left;

      padding:2em
     }
  h1 { margin-left: 3% }
  h2 { color: #990066 }
pre { color:black; font-size: 150%; font-family:arial; font-style:bold)
-->
</style>
```

2. Save the document as background2.htm and look at it with your browser. What should happen is that the image associated with the <pre> element is inserted as a background texture, for example as follows:



Style Sheets for the Web

How It Works

In this example, the background consists of a repeated .jpg file which serves as a background texture to a single element in the document. Whereas in the previous example the property background-repeat was used to specifically say that the background should not be repeated again and again, in this case this is exactly the effect we want. The small .jpg file (pastels.jpg) has been tessellated (or 'tiled') across the background. Notice that the text is set to bold by the use of the font-style property, and that the padding property has been used to make the background cover an area outside the normal boundaries of the text. We talk about this later on.

The fixed background attachment means that the background doesn't scroll and stays in place. Notice the use of background-color to set the background to white should the texture not be shown. The margin-left property, used to push the heading in, is explained in the following section.

The Background Color for Table Cells

It's easy to set the color for table cells. All you do is to give the table cell elements a style in the style sheet. For example:

```
td { background: #FFCCFF }
th { background: #FF99FF }
```

This sets the color of all table cells that are table data or table header cells to the hexadecimal colors specified.

Borders, Margins and White Space

Having discussed color and backgrounds we now move on to look at the related subject of improving the look and feel of documents by adding 'white space' and borders around elements. CSS uses the idea of the 'box model' to specify the position of borders, padding, margins and content as shown below. The content box contains the text or image itself, for example a paragraph of text, a list, a heading or a <blockquote> element. Surrounding this is an area called the *padding* into which an image or background color behind the content can spread. If you have some text and you want to extend the color background beyond the border of the text, then you need to set the padding. The border goes between the padding and the margins and this can be set separately. The margins are transparent and reveal the background of the body 'through' them:

Setting the Page Margins

Another way to add white space to a web page is to go for larger margins. You can set the left and right margins with the `margin-left` and `margin-right` properties.

Try It Out – Altering the Page Margins

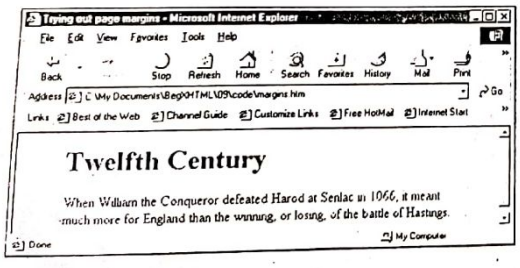
1. Enter the following XHTML document into your text editor, save the file as `margins.htm` and run it in your browser:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"><head>
<title>Trying out page margins</title>
</head>
<body>
<h1>Twelfth Century </h1>
<p>When William the Conqueror defeated Harod at Senlac in 1066, it meant much more
for England than the winning, or losing, of the battle of Hastings.</p>
</body>
</html>
```

2. Now add a style sheet into the head section as so:

```
<head>
<title>Trying out page margins</title>
<style type="text/css">
<!--
body { margin-left: 10%; margin-right: 10%; }
-->
</style>
```

3. Save the file as `margins.htm` again and look at it with your browser. You should see that the margins of the whole document have been made larger. For example:



How It Works

The style sheet sets both margins to 10% of the window width, and the margins will scale when you resize the browser window. You can experiment with the style sheet by changing the margin value to 15%, to -5% (negative value) and so on. If you construct a style for paragraphs for example like this:

```
p { margin-left: 3%}
```

then all paragraphs will be indented with regards to the document as a whole.

To make headings a little more distinctive, you can make them start *within* the margin set for the <body> element, for example:

```
body { margin-left: 10%; margin-right: 10%; }
  h1 { margin-left: -8%;}
  h2,h3,h4,h5,h6 { margin-left: -4%; }
```

The margins for the headings are additive to the margins for the body. Negative values are used to move the start of the headings to the left of the margin set for the body.

Controlling the White Space Above and Below a Heading

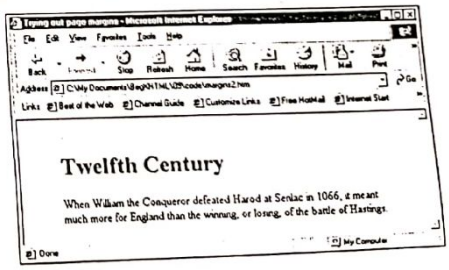
The `margin-top` property specifies the space above and the `margin-below` specifies the space below.

Try It Out – More Space Between a Heading and Text

1. Using the XHTML file from the previous example, add extra space between the heading and the text below it by inserting the following additional style into your style sheet.

```
<style type="text/css">
<!--
body { margin-left: 10%; margin-right: 10%; }
h1 { margin-top: 10%; margin-bottom: 5%; }
-->
</style>
```

2. Save the file as `margins2.htm`. You should see something similar to the following when displayed in the browser:



ce between the heading 'Twelfth Century' and the paragraph below it is larger, as is the space the heading (compare it with the previous screenshot).

ace Before or After a Specific Heading

To specify the space above a particular heading, you should create a named style for the heading. This idea is covered later in the chapter, but because the mechanism is useful in our current context, we will explain briefly what is involved.

What you do is to use the `class` attribute in the markup, for example:

```
<h2 class="subsection">Getting started</h2>
```

The style rule is then written as:

```
h2.subsection { margin-top: 8em; margin-bottom: 3em; }
```

The rule starts with the tag name, a dot and then the value of the `class` attribute. Be careful to avoid placing a space before or after the dot. If you, do the rule won't work. There are other ways to set the styles for a particular element but the `class` attribute is the most flexible.

When a heading is followed by a paragraph, the value of `margin-bottom` for the heading isn't added to the value of `margin-top` for the paragraph. Instead, the maximum of the two values is used for the spacing between the heading and paragraph. This subtlety applies to `margin-top` and `margin-bottom` regardless of which tags are involved.

First-Line Indent

Sometimes you may want to indent the first line of each paragraph. The following style rule emulates the traditional way paragraphs are rendered in novels.

ry It Out – First-Line Indent

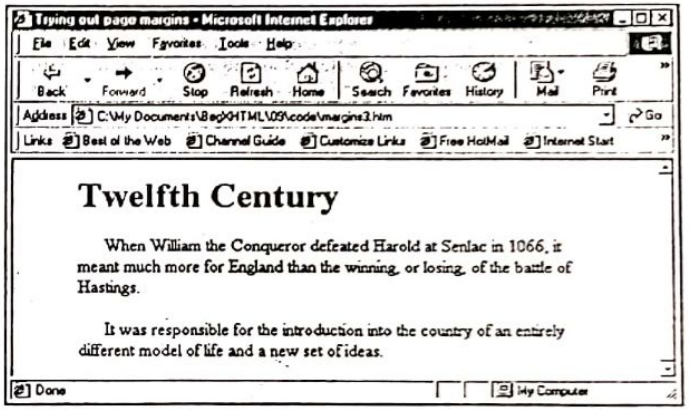
1. Enter the following XHTML document into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Trying out page margins</title>
<style type="text/css">
<!--
body { margin-left: 10%; margin-right: 10%; }
    p { text-indent:5%}
-->
</style>
</head>
<body>
```



```
<h1>Twelfth Century </h1>

<p>When William the Conqueror defeated Harold at Senlac in 1066, it meant much
more for England than the winning, or losing, of the battle of
Hastings.</p>

<p>It was responsible for the introduction into the country of an entirely
different model of life and a new set of ideas.

</body>
</html>
```

2. Save as `margins3.htm`. You should see the following or similar when you view the file in your browser. The first line of each paragraph is indented:



Borders

You can easily add a border around a heading, list, paragraph or a group of these enclosed with a `<div>` element. For instance:

```
div.box { border: solid; border-width: thin; padding: 0.2em; }
```

which can be used with markup such as:

```
<div class="box"> Some text, paragraphs, or whatever </div>
```

The content within this `<div>` element will be enclosed in a box with a thin line around it. Although borders are potentially useful, be careful not to use them where they would add to the clutter of the page, and distract from the text.

There is a limited choice of border types:

dotted	dashed	solid	double
groove	ridge	inset	outset

The `border-width` property sets the width. Its values include thin, medium and thick as well as a specified width e.g. 0.1em. The `border-color` property allows you to set the color.

CSS Properties Used to Control Fonts

Having introduced the topic of fonts we can now go on to a discussion about how to specify the font for your XHTML document. There are a number of CSS properties for this very purpose. Here, we will look at a selection of these, concentrating on the simplest ways to achieve effects. For those who would prefer a much more detailed appreciation of the subject, there are a number of pointers from the W3C site (at `http://www.w3.org`), under 'CSS'. The properties we use here are reliably implemented by the major browser vendors. Use these properties rather than the old-fashioned `<font>` tag which not only clutters your markup but also makes it very difficult to make simple font changes across a whole document in an easily managed way.

The CSS font properties that we cover are:

- `font-size`
- `font-style`
- `font-weight`
- `line-height`
- `font-family`

Setting the Font Size

You can control the font size with the `font-size` property. The simplest way to specify the font size is with percentage values, which is the method we use in this chapter. These are interpreted by the browser as being relative to the size of the parent element of the element you're describing. So, for example, if you specify the font size of bold text as 120% within a 10-point paragraph, then the bold text will be 12-point (because 120% of 10 is 12).

If you are accustomed to an ordinary desktop publishing environment then you will be more familiar with using point size to define the size of the text on a page. You may be surprised to learn, then, that specifying the font size in terms of points is not a reliable method with style sheets for Web documents. With CSS of course you can specify point sizes like this:

```
h1 {font-size: 20pt}
h2 {font-size: 18pt}
h3 {font-size: 14pt}
```


The trouble is that when it comes to displaying the document, *different browsers will show the text in different sizes*. For example, 7-point Verdana may look like 6-point on some computer screens with the result that the font is actually too small to read.

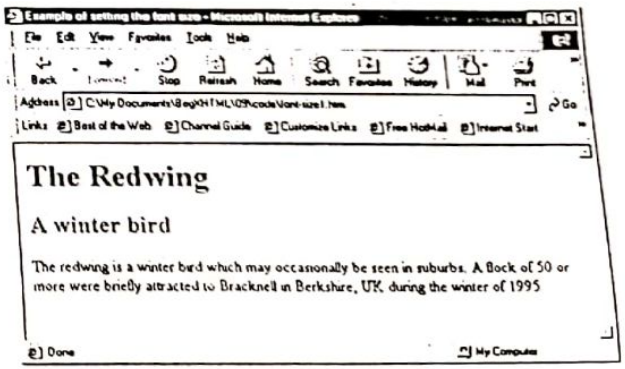
> **In general, a font size will appear smaller on Netscape Navigator than on Microsoft's Internet Explorer.**

Try It Out – Specifying the Font-Size

1. Enter the following XHTML document into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">  <head>
    <title>Example of setting the font size</title>
    <style type="text.css">
    <!--
      body {font-size: 140%}
    -->
    </style>
  </head>
  <body>
    <h1>The Redwing</h1>
    <h2>A winter bird</h2>

    <p>The redwing is a winter bird which may occasionally be seen in
suburbs. A flock of 50 or more were briefly attracted to Bracknell
in Berkshire, UK during the winter of 1995.</p>
  </body>
</html>
```

2. Save the file as `font-size1.htm` and have a look at it on your browser:



How It Works

The `font-size` property has set the body of the document to a larger font size. Setting the body text means that you affect all the elements in the body, including paragraphs and headings. You can experiment decreasing the font size to see how small the font size gets before the text becomes hard to read. Try setting it to 70%, for example. You may also like to try using points instead of percentage to specify the font. For example, change the style of the <body> tag from `body (font-size: 140%)` to `body (font-size:16pt)`.

Setting the Font Size for a Specific XHTML Element

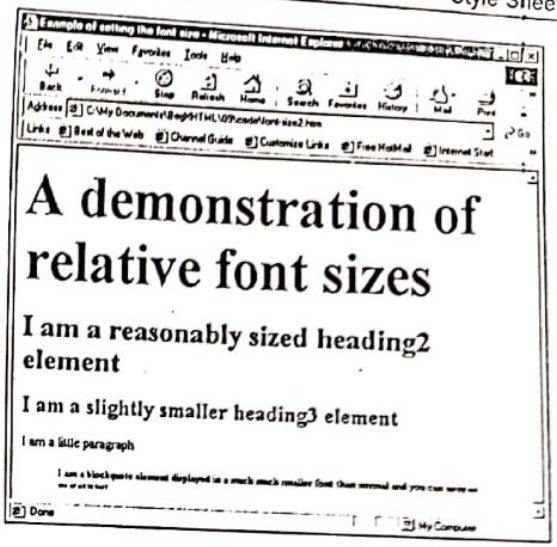
You can also set the font so that it's different for all headings, paragraphs, block-quote elements, bold text, unordered list, and so on. This example shows how different heading elements can be given different font sizes.

Try It Out – Setting the Font Size for Heading Elements

1. Enter the following XHTML document into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Example of setting the font size</title>
    <style type="text/css">
    <!--
      p (font-size:90%)
      h1 (font-size:380% }
      h2 (font-size: 200%)
      h3 (font-size: 150%)
      blockquote (font-size: 70%)
      b (font-size:70%)
    -->
    </style>
  </head>
<body>
  <h1>A demonstration of relative font sizes</h1>
  <h2>I am a reasonably sized heading2 element</h2>
  <h3>I am a slightly smaller heading3 element</h3>
  <p>I am a little paragraph</p>

  <blockquote>I am a blockquote element displayed in a much much
smaller font than normal and you can <b>hardly see me at all
in fact</b></blockquote>
  </body>
</html>
```

2. Save the file as `font-size2.htm` and have a look at it on your browser. The result of our own experiment is shown opposite:



How It Works

The heading elements have been displayed larger or smaller depending on the percentage value they were given in the `font-size` property. The font size – certainly when expressed as a percentage – is taken relative to the enclosing element. So, for example, the size of the element is taken as relative to the <p> element that encloses it. The size of a <p> element is meanwhile taken relative to the enclosing <body> element.

Font Weight

You can set the weight of the font with the `font-weight` property. This instructs the browser to use a lighter or bolder font, as you feel appropriate. One way of specifying the weight is to use a number to indicate how dark the font should be. Choose from 100 (light) through 200, 300, 400 and so on, up to 900, which is dark. The use of numbers to indicate font weight is now supported by IE4, IE5 and by Netscape Navigator 4. Another method involves using the keywords bold, bolder, lighter, and normal

Here are some simple examples:

```
h1 (font-weight: bolder)
h2 (font-weight: 300)
h3 (font-weight: lighter)
h4 (font-weight: 900)
```

Font Style

The font style relates to whether the text is italic, oblique or normal. The first two options are actually the same thing: some typographers use the word 'italic' whereas others prefer 'oblique'.

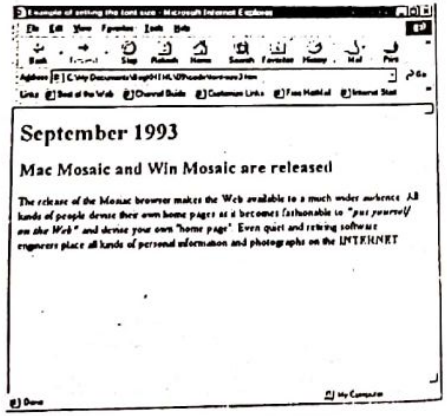
Most browsers render the (emphasis) element in italic and the element in bold.

Try It Out – Changing the Font Style

1. Type this simple XHTML document into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Example of setting the font size</title>
    <style type="text/css">
    <!--
      p { font-style: normal }
      em { font-style: italic; font-weight: bold; }
      strong { text-transform: uppercase; font-weight: bold; }
    -->
    </style>
  </head>
  <body>
    <h1>September 1993</h1>
    <h2>Mac Mosaic and Win Mosaic are released</h2>
    <p>The release of the Mosiac browser makes the Web available to a much wider
audience. All kinds of people devise their own home pages as it becomes
fashionable to <em>"put yourself on the Web"</em> and devise your own "home page".
Even quiet and retiring software engineers place all kinds of personal information
and photographs on the <strong>Internet</strong>.</p>
  </body>
</html>
```

2. Save the file as `font-size3.htm` and have a look at it on your browser:



Forcing the Text to be Lower or Upper Case

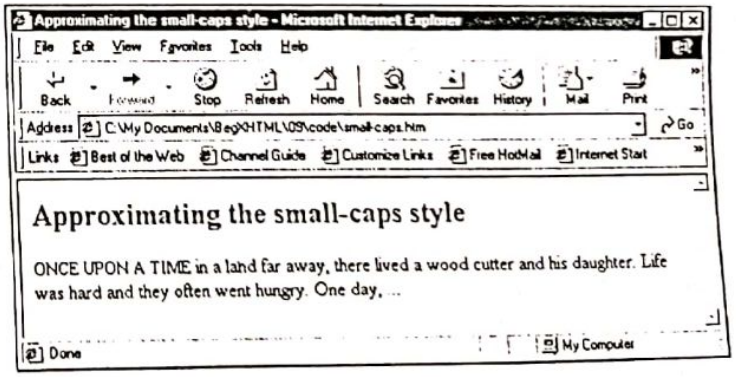
The CSS `text-transform` property can be used to force the text to be all in uppercase or all in lowercase (for example, `text-transform: lowercase`). It is sometimes useful to transform the first word or so in a paragraph into uppercase at a slightly reduced point size.

Try It Out: Making the First Words of a Paragraph Upper Case

1. Enter the following XHTML document into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Approximating the small-caps style</title>
    <style type="text/css">
    <!--
      span {text-transform: uppercase; font-size: 90%}
    -->
    </style>
  </head>
  <body>
    <h2>Approximating the small-caps style</h2>

    <p><span>Once upon a time</span> in a land far away, there lived a wood cutter
and his daughter. Life was hard and they often went hungry. One day, ...</p>
  </body>
</html>
```

2. Save the file as `small-caps.htm` and look at it on your browser. You should see something similar to the following:



How It Works

The element is used to single out a group of words to receive the upper case style. The style for this element says that the text should be transformed to upper case and be 90% of the usual size. This has the effect of making the text look like 'small caps'.

Line-Height

The `line-height` property specifies how far apart the lines in a paragraph or other element should be displayed. A line height of 200% effectively doubles the distance between lines. The line height is generally equivalent to the point size of the text so that 12-point type has a line height of 12pt. If you increase the line height of a paragraph consisting of 10pt Times by say, 120%, the result will be an increase of line height to 12 pts; if you up the line height to 200% then the result is a 20pt line height, and so on.

Try It Out – Adjusting the Line Height

1. Take the same document as above, and alter it as follows:

```
<style type="text/css">
<!--
  span {text-transform: uppercase; font-size: 90%}
  p { line-height: 200%}
-->
</style>
```

2. Save the file and look at it on your browser. You should see that the text enclosed in <p> elements is now double spaced.

Setting the Font Name

Many browsers are running on machines with lots of fonts already installed. You can take advantage of this to make your text look distinctive. CSS allows you to name the font you want to use. Let's assume you want to make a wedding invitation on the Web, and want to use the font called 'Nuptial BT'. You can use the following CSS rule:

```
body { font-family: "Nuptial BT" }
```

Try It Out – Specifying the Font Name

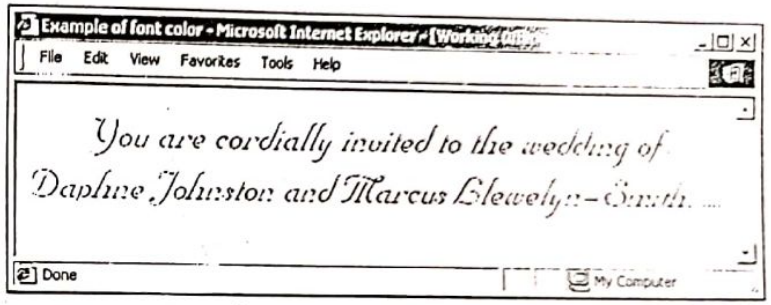
1. Enter the following XHTML document and accompanying style sheet into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Example of font color</title>
    <style type="text/css">
```



```
    <!--
      body { font-family: "Nuptial BT";
      font-size: 300%; text-align: center }
    -->
    </style>
  </head>
  <body>
    <p>You are cordially invited to the wedding of<br />
Daphne Johnston and Marcus Llewelyn-Smith, ....</p>
  </body>
</html>
```

2. Save the file as `wedding.htm` and look at it on your browser. You should see a result similar to the following:



How It Works

The `font-family` property specified the font to use. Should this not be available on the computer viewing the document, a browser default font will be used instead.

Specifying Alternative Fonts for the Browser to Use

Suppose you want to use Garamond as the font for the entire document. This is the critical line you might insert into your style sheet:

```
body {font-family: Garamond, "Times New Roman", serif}
```

However, suppose Garamond is not a type available on the computer displaying the document? Without an alternative the browser will simply use a default font and the document may look less pleasing as a result. In CSS you can, in fact, specify no end of alternative font families in order of preference. The browser will start with the first one in the list (Garamond in this example) and if that is not available, the browser will try the next one along (Times New Roman here), and so on. If none of the fonts in the list are available, the browser will stick with the current font

Try It Out – Specifying a List of Alternative Fonts

1. Enter the following XHTML document into your text editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Style Sheet History</title>
    <style type="text/css">
    <!--
      h1{ font-family: Georgia, "Times New Roman", Garamond, Times, serif }
      p { font-family: Verdana, Arial, Helvetica, sans-serif }
    -->
    </style>
  </head>
  <body>
    <h1>November 1995</h1>
    <p>Style sheets for HTML documents begin to take shape Bert Boss, Håkon Lie,
Dave Raggett, Chris Lilley and others from the
World Wide Web consortium convene in Versailles near Paris to discuss the
deployment of Cascading Style Sheets (the name infers that more than one style
sheet can interact to produce the final look of the document). Using a special
language, the CSS group advocated that everyone would soon be able to write simple
styles for HTML. The SGML community suggest an alternative a LISP-like
language called DSSSL.</p>
  </body>
</html>
```

2. Save the file as `fonts.htm` and look at it on your browser. What you see will depend on which fonts you have on your machine.

How It Works

The fact is that many browsers may not have access to your preferred font. To cope with this, CSS allows you to specify a list of fonts in order of preference. CSS also provides three generic font family names you can use: *serif*, *sans-serif*, and *monospace*. These are valuable as a means of last resort for selecting the font. The 'serif' is a very general category, included as the ultimate fallback.

If the font name contains a space, such as `"Times New Roman"` we recommend you enclose the name in quotation marks as shown above. Note that font names are case insensitive, but it is good practice to use the correct case.

Inventing your Own Named Styles

This topic has been left until late in the chapter as the idea of named styles comes into its own only once you have experimented with CSS, and feel fluent in the use of at least a range of properties.

In many word processing packages you select a named style from a list and apply it to a paragraph or other text. For example, in Word, you have styles such as Heading 1, Body Text, and Footer, and you can also define your own. In CSS you can similarly invent a style and then name it for subsequent use in a document. The trick is accomplished by using a special kind of CSS notation of which here are three simple examples:

```
p.special { color: red; background: yellow }
p.small { font-size:90%}
p.large{ font-size:200%}
```

The first 'rule' says that all paragraphs named 'special' should be red text on a yellow background. The second says that all those named 'small' should be in a font 90% smaller than usual. Meanwhile paragraphs named 'large' should be in a font twice as large as normal size.

But how, once you have invented a named style, do you apply it to a particular piece of text? You apply a named style by means of the `class` attribute, which can be used with all XHTML elements. Here is a paragraph marked up in such a way that it will take on the style defined as 'special'.

```
<p class="special">This paragraph will be displayed with red text on a yellow
background </p>
```

The class attribute is so called because it is used to identify a given class of elements, in this case all paragraphs of type 'special'.

You can devise named styles for elements other than paragraphs. For example you might choose to devise a named style for Heading 2 elements, for block-quote elements, and so on. This idea is demonstrated in the example that follows.

Try It Out – Inventing and Applying your own Named Styles

In this example, a number of named styles are devised called 'color', 'small' and 'normal'. These in effect consist of three styles for three different kinds of paragraphs. Having invented the styles and placed them in the style sheet they are applied with the `class` attribute as need be by the paragraphs in the document.

1. Enter the following style sheet consisting of three named styles for paragraphs and insert it into a document between the `<style>` and `</style>` tags in the usual way. The document you choose can be any simple XHTML document with a selection of paragraph elements. If you prefer you can use the document that we use below.