HTML Concepts

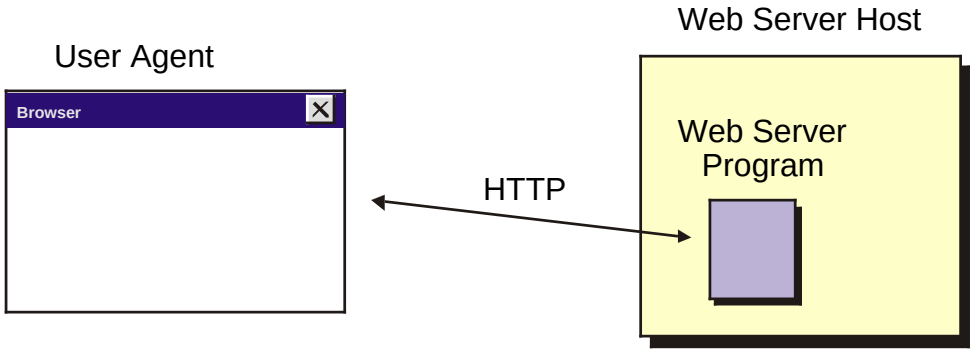
- HTML Forms
- Javascript Fundamentals
- Cookies
- Cascading Style Sheets

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

HTML and HTTP

- Browsers send requests to Web servers, which respond by providing the content requested



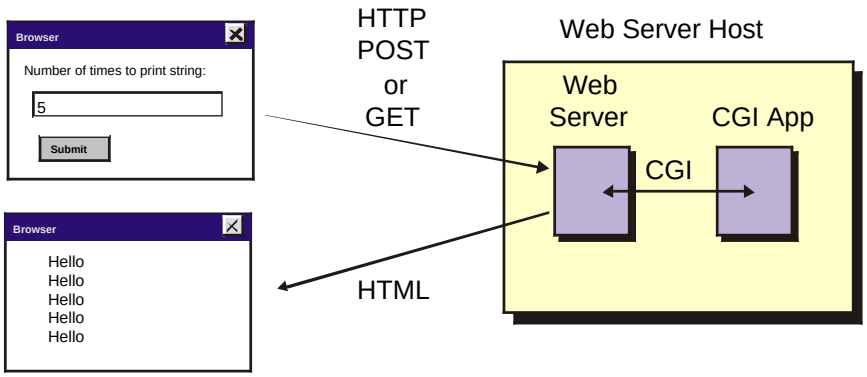
As we've already seen, the Web is a client/server environment.

The browser, or user-agent, acts as a client, sending requests to the Web server. The Web server responds by providing the content to satisfy the request.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

HTML Forms

- HTML defines **forms** that let users enter or request information via the browser
- When the user submits the form, the web server invokes a program that responds to the form, typically by generating a new HTML page



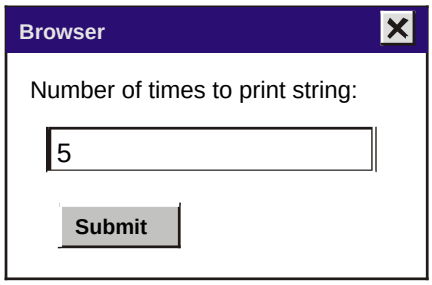
HTML forms are the basis for most Web applications. They let users request or pass data to the application running on the server. The form will specify a GET or POST action that the browser will transmit to the server when the user presses the Submit button. In response, the application generates an HTML page back to the client, which can then display it.

Traditionally, HTML forms have been processed by programs that the Web server invokes using a technique known as the Common Gateway Interface. You can think of CGI as adding programmability to Web pages so applications can generate dynamic content

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Writing an HTML Form

```
1    <form
2      action="http://localhost/cgi-bin/hello.pl">
3      <p>Number of times to print string:</p>
4      <p><input type="text" name="count"></p>
5      <p><input type="submit"></p>
6    </form>
```



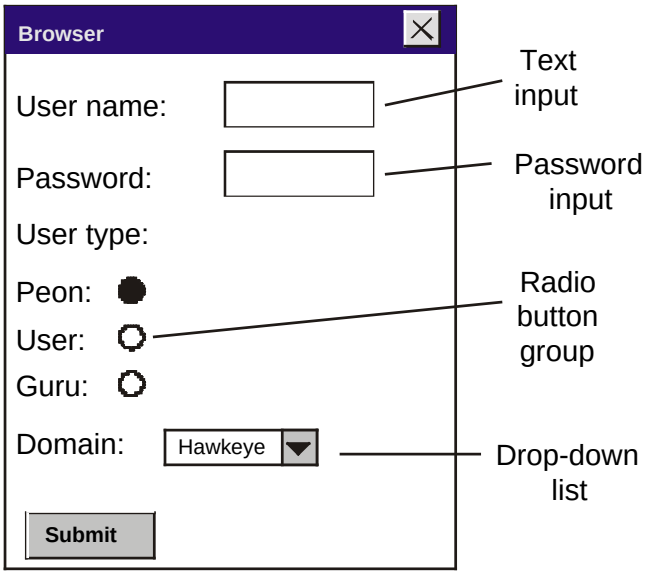
This page shows a sample HTML form that invokes a Perl-language program via CGI. Later, we will see how to write Java programs that respond to such forms. This form asks a user how many times to print some string: the returned HTML page will display the string using that count.

This form defines two input "widgets": a text entry field and a Submit button. Note that we assign a name to the entry field of "count" -- the CGI program can retrieve the entry field contents using that name.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Form Input Widgets

- Forms typically contain one more input devices into which the user can enter text or select a choice



Each of the logical widgets will have a name and value -- the browser will pass the name/value pair to the program that processes the form. The HTML author defines the name but the value comes from the user's typing or selection.

Password input boxes "mask" the text that user types so that someone looking over the user's shoulder cannot read the password.

Radio buttons act as a group -- the user can select only one of the group. Note that there's also a checkbox widget (not shown) that acts as a simple on/off selection.

This form contains a "select" box, which presents a list of choices. The default behavior is to allow only a single selection, but the HTML author can set the select box to multiple selection.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Sample HTML Form

```
1    <form
2      action="http://localhost/cgi-bin/login.pl"
3      method="post">
4      User name: <input type="text" name="username"><br>
5      Password:  <input type="password" name="password"><br>
6      User type:<br>
7      Peon<input type="radio" name="usertype" value="peon"
8                  checked="true"><br>
9      User<input type="radio" name="usertype" value="user"><br>
10     Guru<input type="radio" name="usertype" value="guru"><br>
11     Domain: <select name="domain">
12       <option>Hawkeye</option>
13       <option>Wolverine</option>
14     </select><br>
15     <input type="submit" value="Submit">
16   </form>
```


This listing shows the HTML fragment for the form shown on the previous page.

Note how each of the radio buttons has the same "name" -- that's what organizes them into a group.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

HTTP Request Headers

- When a browser sends a request to a Web server, the message includes not only the request, but HTTP *headers* that contain additional data about the request

Request to http://www.descriptor.com/about.html

```
GET /about.html HTTP/1.1
Host: www.descriptor.com

Accept: text/xml,text/html;*/*
Accept-Language: en-us,en
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.6)
```

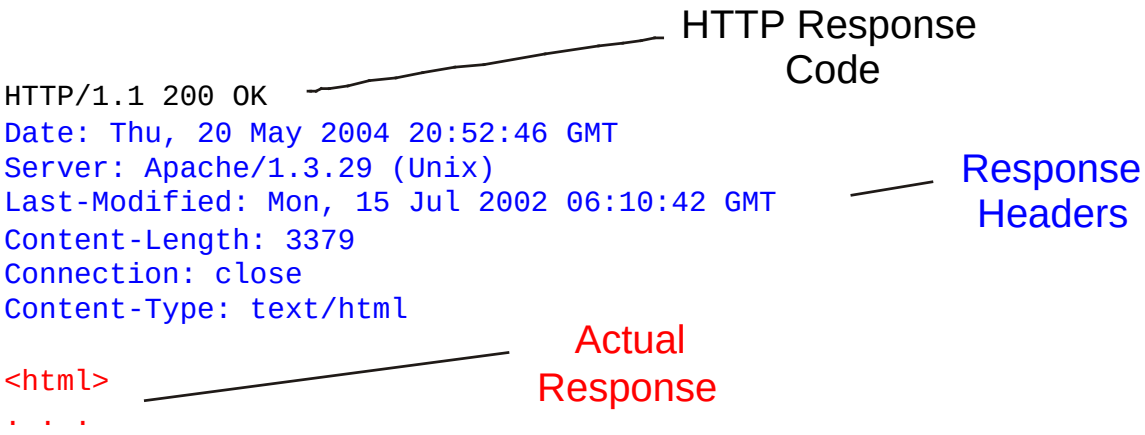
Headers

This page shows a few of the headers for a simple request. Note that the browser indicates what types of content it can handle, what national language the user has configured and a "user-agent" string that in some fashion identifies the browser.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

HTTP Response Headers

- The Web server's response also contains headers that can be processed by the browser

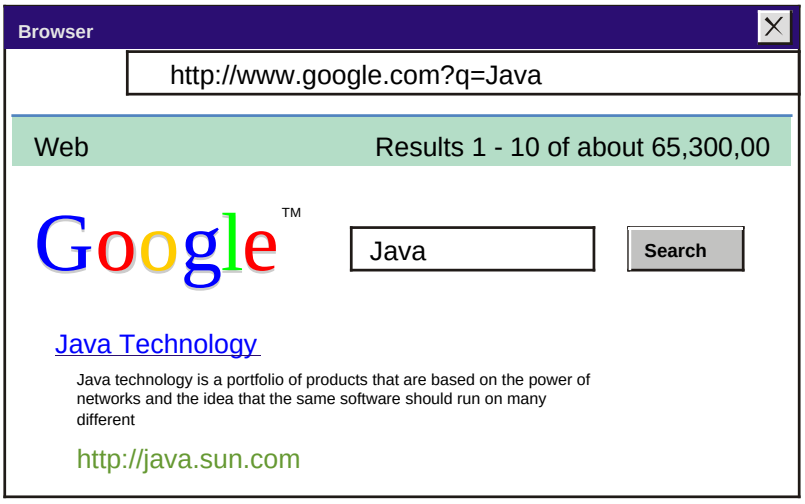


When the browser receives a response, it examines any relevant headers and then renders the returned HTML.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

GET versus POST

- The **GET** method is intended for requests that do not change the state of the server
- GET requests pass any input form parameters as part of the URL, while **POST** requests pass form parameters separately



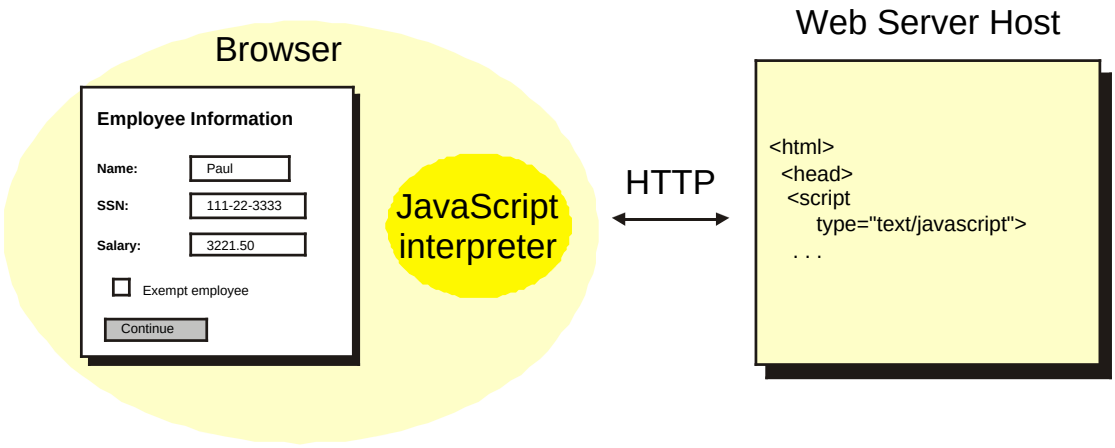
When you write an HTML form, the "form" tag itself has an optional "method" attribute that lets you specify "get" or "post". If you don't specify, the default is "get".

Here we show a Web browser after the results of a search on the Google search engine. The original form had a single text-input widget for the search phrase -- note how the entered string, "Java" appears in the browser's URL bar. This is proof that the input form used the GET method.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Javascript Fundamentals

- JavaScript lets you write programs that run on the browser rather than on the Web server
- JavaScript has a similar syntax to Java, but is not a fully fledged programming language like Java



You embed JavaScript in your HTML pages, but the scripts actually run in the browser, which is considered "client-side".

JavaScript lets you write Web pages with programmability -- for example, you could write JavaScript to validate the contents of a form when the user presses the submit button. That would save a network round-trip if, for example, the user didn't enter data into a required field.

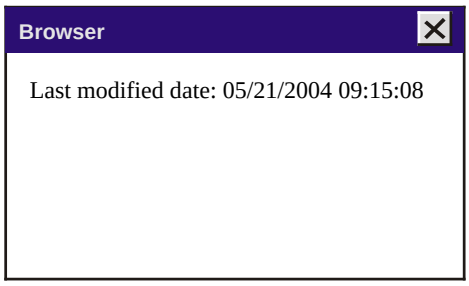
JavaScript is supported by most modern browsers such as Microsoft Internet Explorer and Mozilla.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Sample JavaScript

- A common way to write JavaScript is to embed it within a **script** element

```
1    <html>
2      <body>
3        <script type="text/javascript">
4          document.write ( "Last modified date: " )
5          document.write ( document.lastModified )
6        </script>
7      </body>
8    </html>
```



You can embed Javascript either within the "head" or "body" of an HTML page. Here, the script element is within the body since we want the script to generate HTML output.

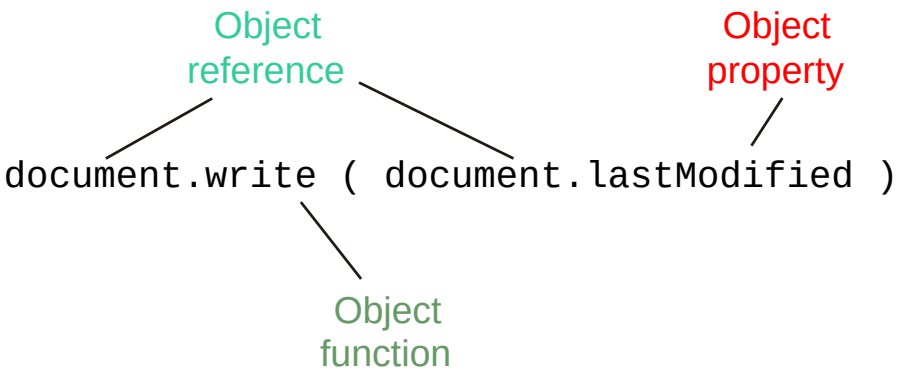
Note that we identify the scripting language using a MIME type -- some browsers support other scripting languages, but Javascript works on most.

You can also store scripts in external files and reference them using the "src" attribute. That lets you reuse the same script in multiple HTML pages.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

JavaScript Objects

- JavaScript **objects** represent parts of the document and allow you to call **functions** on them or manipulate their **properties**
- Unlike in Java, you cannot create your own kinds of objects –– instead, JavaScript programs can only use pre-defined objects



Here we see a line from a script shown earlier. The "document" reference is the name of the object that represents the entire HTML page.

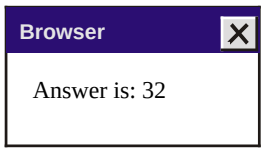
The "write" function lets you insert some text into the document, which the browser eventually displays. You can tell that "write" is a function since it uses parenthesis. The "document.lastModified" is a parameter to the function.

The "document.lastModified" property is a read-only property that tells you when the source HTML page was last changed.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Writing JavaScript Functions

- To avoid duplicating code in scripts, you can write your own **functions** that you can call from scripts
- You generally define functions in the *head* of the HTML page




```
1   <html>
2     <head>
3       <script type="text/javascript">
4          function sum ( num1, num2 )
5          {
6             return num1 + num2
7          }
8       </script>
9     </head>
10    <body>
11      <script type="text/javascript">
12        document.write ( "Answer is: " )
13        document.write ( sum ( 15, 17 ) )
14      </script>
15    </body>
16  </html>
```


Functions let you organize your scripting code to make it easier to read and maintain.

Functions can accept zero or more arguments. To call a function, you must use parenthesis around the argument values.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

JavaScript Variables

- Any script can define **variables** to hold data -- you can reference variables across script elements
- Unlike in Java, variables are *untyped*

```
1    <html>
2      <head>
3        <script type="text/javascript">
4          var mod_date = document.lastModified
5        </script>
6      </head>
7      <body>
8        <script type="text/javascript">
9          document.write ( "Last modified date: " )
10         document.write ( mod_date )
11       </script>
12     </body>
13   </html>
```


Variables are "scratch pad" storage for saving the intermediate results of calculations, user-input data and so forth. Though it's not strictly required, it's a good practice in JavaScript to define variables before use.

By "untyped", we mean that any JavaScript variable can store characters, numbers and so forth. This is different than in Java, where each variable can hold only a single kind of data. Type languages are a little tougher to program, but tend to be more reliable since the compiler finds type-related mistakes before the program ever runs.

If you define a variable within a function, the variable is considered "local" and can only be used by that function.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

JavaScript Events

- JavaScript programs can respond to **events**, such as the user submitting a form
- You typically handle events by writing a function

Event	Description
onsubmit	The user pressed a form's Submit button
onfocus onblur	The user tabbed to or from an input widget on a form
onclick	The user clicked once on a button, checkbox etc.
onchange	The user changed the text or selection in an text-input field or drop-down list

---

Event handlers let your JavaScript programs react to user actions.

To handle an event, you write the event's name as an attribute on the source object's definition, for example:

```
<input type="text" onfocus="myfunction()">
```

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

JavaScript Events Example

```
1    <html>
2      <head>
3        <script type="text/javascript">
4           function confirmSubmit()
5           {
6              var ok = confirm ( "Really submit?" );
7              return ok;
8           }
9        </script>
10     </head>
11     <body>
12       <form action="http://localhost/test"
13             onsubmit="return confirmSubmit()">
14         <input type="Submit">
15       </form>
16     </body>
17   </html>
```


Here we show a JavaScript sample that uses the "onsubmit" event in line 13. When the user submits the form, the browser will call our "confirmSubmit" function, which in turn calls the built-in "confirm" function that displays a message box with OK and Cancel buttons and the specified text. If the user presses OK, "confirm" returns true, otherwise it returns false.

Our "confirmSubmit" function returns the true or false value which the browser then uses to determine whether or not to actually submit the form.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Using Javascript to Validate a Form

```
1    <html>
2      <head>
3        <script type="text/javascript">
4           function validateMyForm()
5           {
6             var theForm = document.MyForm
7             var email = theForm.email.value
8             var ok = true
9             if ( email.length == 0 )
10            {
11              ok = false
12              alert ( "Email must not be empty" )
13            }
14            return ok
15          }
16       </script>
17     </head>
```



This listing starts the JavaScript validation example. In lines 4 to 15, we define a function that determines whether the user entered text into an input widget or not. In lines 6 we retrieve the reference to the form object and then in line 7, we obtain whatever text the user entered into the "email" text widget.

In line 9, we use the JavaScript "if" statement to test the length of the email string. If the string has no characters, we display an alert message box. Note that the function returns either true or false depending on whether the email string had any characters.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Validating a Form, cont'd

```
18   <body>
19       <form action="http://localhost/test"
20             name="MyForm"
21             onsubmit="return validateMyForm()">
22         Email: <input type="text" name="email"><br>
23         <input type="Submit">
24       </form>
25     </body>
26   </html>
```

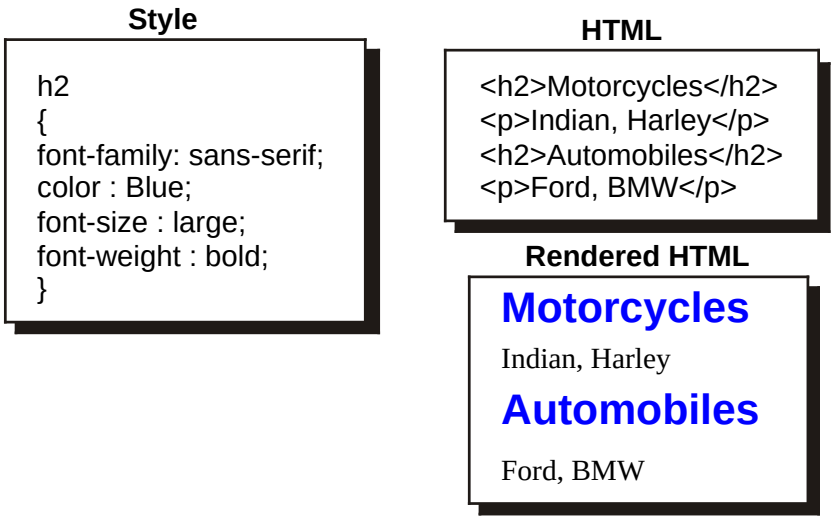

Continuing the listing from the previous page, here we show the form definition -- note in line 20, we give the form a name so our function can retrieve its reference. Then in line 21, we ask to be notified on "submit" events, in which case we call the function defined on the previous page.

Note in line 22 how we assigned the text widget the name "email" -- recall that the validation function uses that name to retrieve any text the user entered into the widget.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

What are Cascading Style Sheets?

- Cascading style sheets (CSS) let you define **styles** that govern the appearance of elements in one or more HTML documents
- Major benefit: You can change the look of an entire Web site by modifying one file!



Cascading Style Sheets let you separate content from presentation. The CSS specification is administered by the W3C (www.w3.org) and is, at the time of this writing (June 2004), currently at CSS level 2, revision 1. CSS level 3 is under development.

CSS support varies across browsers. While most of what we cover in this chapter should work on most modern browsers, it's a good practice to test more advanced CSS pages in the browsers you expect your users to run.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Defining Styles

- There are three places where you can define styles:
    - In-line within a tag
    - In a page's **head** tag (document-level)
    - In an external file
- Using an external file lets you share the style definitions among multiple HTML files

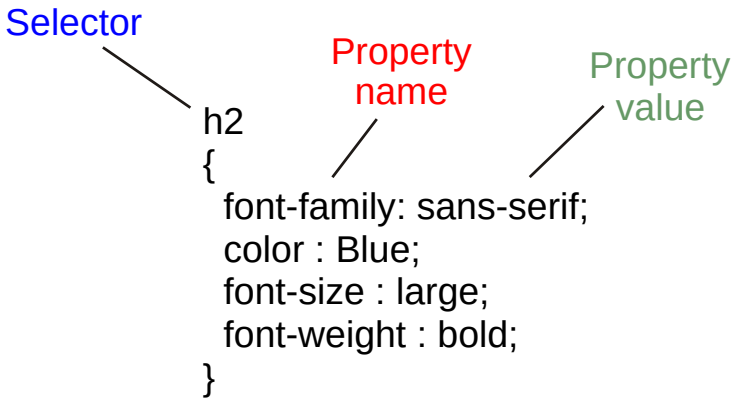
The "virtual style sheet" consists of all of the styles defined in each of the locations listed here. If a style is defined in multiple places differently, the most local one takes precedence -- i.e. a document-level style takes priority over a style defined in an external style sheet.

Using in-line styles is not very common, so we will not cover it here.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Basic Style Syntax

- To define a style, you write a **selector** and enclose the style's properties within curly braces



The selector selects one or more tags in an HTML document. You can list multiple tags in a selector, separated by commas if you want the same properties applied to all of the tags.

The curly braces define the boundary of the style definition and contains one or more property definitions. If you define multiple properties, each definition must be delimited by a semicolon.

To define a property, you specify the property name, a colon, and the property value.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Style Classes

- Style classes let you define different sets of properties for the same selector

Style

```
p.intro
{
 font-family :
  "Courier New",
   Courier, monospace;
}
p.summary
{
 font-family :
    Arial,
    Helvetica, sans-serif;
}
```

HTML

```
<p class="intro">Here we cover vehicles.</p>
<h2>Motorcycles</h2>
<p>Indian, Harley</p>
<h2>Automobiles</h2>
<p>Ford, BMW</p>
<p class="summary">Here we are done.</p>
```

Rendered HTML

Here we cover vehicles.

Motorcycles

Indian, Harley

Automobiles

Ford, BMW

Here we are done.

Style classes are an optional part of CSS that give you a bit of flexibility that makes it easier to maintain common styles.

In this figure, we define two classes for the "p" tag: an "intro" class that uses monospaced text and a "summary" class which uses a sans-serif font. (Note how in the font-family tag you can specify multiple fonts, from more specific to more generic. The browser will apply the first font it can from the list.)

Then, in the HTML document, we define paragraphs that use the classes defined in the stylesheet. Note also that we have paragraphs using the "default" class with no special styles.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Using Document-Level Styles

- HTML 4.0 defines a **style** tag in which you can define styles for elements within the page
- The *style* tag must be nested within the *head* tag

```
1    <html>
2      <head>
3        <style>
4          h2
5          {
6            font-family: sans-serif;
7            color : Blue;
8            font-size : large;
9            font-weight : bold;
10         }
11         </style>
12       </head>
13       <body>
14         . . .
15       </body>
16   </html>
```


Document-level styles let you define styles that are local to a single HTML page. That way, you can define styles in one page that have no conflicts with styles defined in other pages.

The bad news is that if you have multiple pages in which you want to share common styles, using document-level stylesheets, you must define the common styles in each page. That's a maintenance issue.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Using External Style Sheets

- External style sheets reside in a separate file, typically with .css extension
- To reference the external style sheet, write a **link** tag within the *head* tag

my.css

```
h2
{
font-family: sans-serif;
color : Blue;
font-size : large;
font-weight : bold;
}
```

```
1    <html>
2      <head>
3        <link href="my.css"
4              rel="stylesheet"
5              type="text/css">
6      </head>
7      <body>
8        . . .
9      </body>
10   </html>
```


To reference a stylesheet from an HTML page, you write a "link" tag within the document's "head" tag. In the link tag, you reference the stylesheet, which could be a local or remote (e.g. http) location and the type of the relationship, which is stylesheet. Note that we also specify the MIME type for CSS so the browser understands that we are using CSS.

External stylesheets provide the most flexibility since you can reference the same stylesheet from multiple HTML pages. That way, if you ever decide to change a font or color, you only need to change it in one place.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Text Properties

Property name	Description	Possible Values
color	Text color	#RRGGBB, red, blue, etc.
text-align	Alignment	left, center, right, justify,
text-indent	Indentation	Length in pixels, ems, etc.
text-decoration	Adds feature	none,underline,overline, line-through,blink




```
1    p.test-text
2    {
3    color : #9933FF;
4    text-align : left;
5    text-decoration : line-through;
6    text-indent : 10px;
7    }
```


This page shows a summary of commonly used text properties. Note that it is not an exhaustive list -- see the W3C site for a complete list.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Font Properties

Property name	Description	Possible Values
font-family	Prioritized list	specific name,generic
font-size	Size	xx-small, small,medium large,xx-large, etc.
font-style	Italics?	normal,italic,oblique
font-weight	Boldness	normal,bold,bolder lighter, 100 to 900

Hello

```
1    p.test-fonts
2    {
3    font-family: Arial, Helvetica,
4      sans-serif;
5    font-size: xx-large;
6    font-style: oblique;
7    font-weight: 800;
8    }
```


This page shows a summary of commonly used font properties. Note that it is not an exhaustive list -- see the W3C site for a complete list.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Margin Properties

Property name	Description	Possible Values
margin-bottom	Bottom margin	pixels, ems, etc.
margin-top	Top margin	pixels, ems, etc.
margin-left	Left margin	pixels, ems, etc.
margin-right	Right margin	pixels, ems, etc.

```
1    p.test-margin
2    {
3    margin-left: 10px;
4    margin-right: 20px;
5    }
```

This is a test of the margin CSS properties. The quick brown fox jumped over the lazy dog with white spots.

This is some normal text for comparison. The quick brown fox jumped over the lazy dog with white spots.

This page shows a summary of commonly used margin properties. Note that it is not an exhaustive list -- see the W3C site for a complete list.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Border Properties

Property name	Description	Possible Values
border-left	Left border width, style and color	Width: thin,medium,thick, pixels,ems, etc. Style: dotted, dashed, note, etc. Color: #RRGGBB, red, blue, etc.
Same for top, bottom and right		

```
1    p.test-border
2    {
3    border-top: medium solid green;
4    border-bottom: medium solid green;
5    }
```

This is a test of the border CSS properties. The quick brown fox jumped over the lazy dog with white spots.

This is some normal text for comparison. The quick brown fox jumped over the lazy dog with white spots.

This page shows a summary of commonly used border properties. Note that it is not an exhaustive list -- see the W3C site for a complete list.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Background Properties

Property name	Description	Possible Values
background-color	Color	#RRGGBB, red, blue, etc.
background-image	Image	Reference to image file
background-position	Image position	x,y, top left, top center, etc.
background-repeat	Repeat image?	repeat,repeat-x,repeat-y no-repeat

```
1    p.test-background-image
2    {
3    background-image : url(Greenstone.gif);
4    background-repeat : repeat;
5    }
```


This page shows a summary of commonly used color properties. Note that it is not an exhaustive list -- see the W3C site for a complete list.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

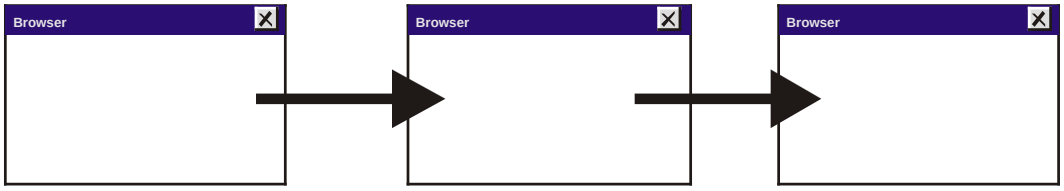
Other CSS Properties

- Padding
- Positioning
- Lists

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Maintaining State Across Requests

- Unlike FTP or Telnet, HTTP conversations do not maintain a long-lived connection between the client and server
- That's great for normal Web browsing, but not good for Web applications that need to maintain state between conversations (e.g. a shopping cart or a secure login)



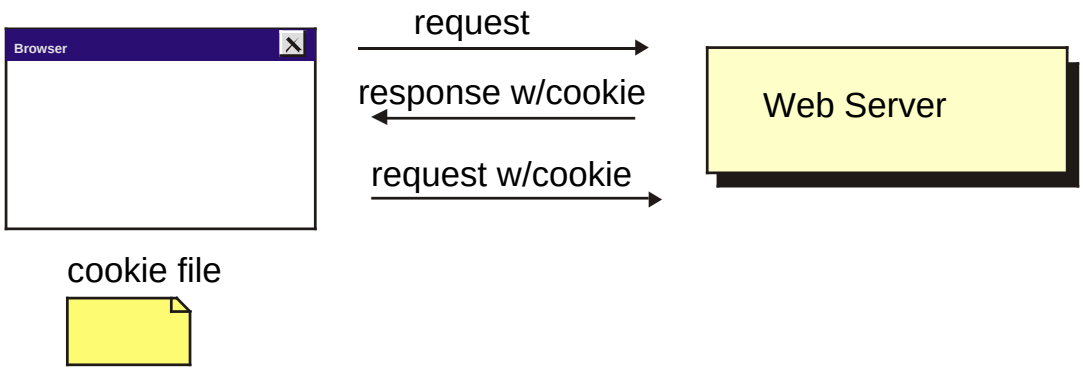
HTTP was originally designed to be a simple protocol, so the connection between browsers and the web server is transient -- essentially, the browser must open a new connection for each request. This is different than with Telnet for example, where the client and server maintain a long-lived connection.

In the early days of the Web, HTTP's statelessness was not an issue. But e-commerce and related applications need to maintain state across requests. The classic example is the shopping cart, which must "remember" its contents until the order is complete.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Cookies

- Cookies are small chunks of information created by a browser at the request of a Web page setting a header
- On subsequent requests, the browser passes the cookie(s) thus allowing the Web application to remember state



Cookies were invented by Netscape to allow Web-based applications to maintain state. Each cookie has a name, a value and an expiration time.

When a Web application returns a response with cookies, the browser stores the cookie information in a convenient (for the browser) location and then returns the cookie back on subsequent requests to the same server.

There are two basic types of cookies -- "session" cookies that expire when the user closes the browser and "persistent" cookies, where the Web application specifies how long the cookie should stay around. Most Web applications use session cookies for maintaining state and persistent cookies for remembering things like user preferences (fonts, colors and like).

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Headers for Cookies

Response

```
HTTP/1.0 200 OK
Date: Fri, 21 May 2004 16:50:27 GMT
Server: Microsoft-IIS/5.0
Set-Cookie: SWID=DFA1D6EF-16FA-4A6C-A180-88D8034A43B1
. . .
```

Subsequent Request (assuming cookie accepted)

```
GET /index.html HTTP/1.1
Host: msn.espn.go.com

Accept: text/xml,text/html;*/*
Accept-Language: en-us,en
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.6)
Cookie:  SWID=DFA1D6EF-16FA-4A6C-A180-88D8034A43B1
```


Here we show a response from http://msn.espn.go.com, which is a sports-oriented Web site. Note that the site requests the browser to set a cookie with name "SWID" -- the name means nothing to the browser, but if allowed, the browser saves the cookie information in a file or in memory.

On subsequent requests to the same domain, the browser simply passes the cookie information as an HTTP header. The receiving Web application can then retrieve the value of the cookie, given its name.

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Chapter Summary

In this chapter, you learned:

- About HTML forms
- How to write simple JavaScript programs
- CSS fundamentals
- About cookies

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems

Copyright © Descriptor Systems, 2001-2003. Course materials may not be reproduced in whole or in part without prior written consent of Descriptor Systems