CSC 498: Web Programming

Haidar Harmanani
Department of Computer Science and Mathematics
Lebanese American University
Byblos, 1401 2010 Lebanon

SGML

- SGML: Standard Generalized Markup Language
  - an international standard (ISO 8879) published in 1986.
  - prescribes a standard format for embedding descriptive markup within a document.
  - specifies a standard method for describing the structure of a document.
    - allows you to set up hierarchical models for each type of document you produce.
    - supports different document structures for different information types, e.g. technical manuals, parts catalogs, design specifications, reports and memos.

SGML

- SGML documents are
  - ASCII format: human and machine-readable
  - independent of any specific hardware or software
  - portable
- Can be broken into three components:
  - structure
  - content
  - style

SGML

- Structure:
  - defined in a file called the Document Type Definition (DTD)
  - describes the structure of a document, much like a database schema describes the types of information it handles and the relationships between fields
    - e.g. the address information should contain street, city, country fields
  - specifies rules for the relationships between elements
    - e.g. the street field must go first before the city field.
    - e.g. the country field is compulsory and can not be blanks.

SGML

```
<address>
        <street> street name </street>
        <city> city name </city>
</address>
```

- Content:
  - content is the information itself
  - identify the content's position within the DTD structure by using "tagging."
    - These tags mark the beginning and end of each part of the structure.
    - e.g. <street> 276 Castle Peak Road </street>
    - "<street>" indicates the start of a street name, and "</street>" indicates the end
  - Nesting of tags reveal the structure of a document.

SGML

- Style:
  - Define how different fields should be displayed.
    - e.g. The style defines that "city field in the address should be all in upper case and in red colored font".
    - Content: <city> hong kong </city>
    - Display: HONG KONG
  - SGML itself has nothing to do with setting standards for style, so most systems still rely on proprietary methods.

SGML

- Benefits of SGML:
  - Increased productivity
  - Reusability
  - Information longevity
  - Improved data integrity
  - Better data control
  - Shareability
  - Portability of information
  - Flexibility beyond traditional publishing

SGML

- Two implementations of SGML standard:
  - HTML: Hypertext Markup Language
  - XML: Extensible Markup Language

	HTML	XML
Structure	Not defined	Defined in DTD files
Content	♣Limited number of tags ♣Tags are defined in HTML specifications ♣Tags are not case sensitive	♣Unlimited number of tags ♣Tags are defined in DTD ♣Tags are case sensitive
Style	Not defined	Defined in XSL (style) files

SGML

- HTML tags are limited, so we need to write content using available HTML tags
- e.g. writing address as

  <P> 276 Castle Peak Road </P>

  <P> Hong Kong </P>
- With XML, we can define new tags for writing address fields.
- e.g. it will be more descriptive to write

  <street> 276 Castle Peak Road </street>

  <city> Hong Kong </city>

Hypertext & HTML

- HyperText Markup Language (HTML) is the language for specifying the static content of Web pages
  - hypertext refers to the fact that Web pages are more than just text
    - can contain multimedia, provide links for jumping within & without
  - markup refers to the fact that it works by augmenting text with special symbols (tags) that identify structure and content type
- HTML is an evolving standard (as new technology/tools are added)
  - HTML 1 (Berners-Lee, 1989): very basic, limited integration of multimedia
    - in 1993, Mosaic added many new features (e.g., integrated images)
  - HTML 2.0 (IETF, 1994): tried to standardize these & other features, but late
    - in 1994-96, Netscape & IE added many new, divergent features
  - HTML 3.2 (W3C, 1996): attempted to unify into a single standard
    - but didn't address newer technologies like Java applets & streaming video
  - HTML 4.0 (W3C, 1997): current standard
    - attempted to map out future directions for HTML, not just react to vendors
  - XHTML 1.0 (W3C, 2000): HTML 4.01 modified to conform to XML standards
  - XHTML 1.1 (W3C, 2001): "Modularization" of XHTML 1.0

Web Development Tools

- Many high-level tools exist for creating Web pages
  - e.g., Microsoft FrontPage [phased out], Miscrosoft Web Expression, Netscape Composer, Adobe PageMill, Adobe Golive [?], Adobe DreamWeaver, HotDog, iWeb, …
  - Many applications have "save to HTML" options (e.g., Word)
  - for most users who want to develop basic, static Web pages, these are fine
- So, why are we learning low-level HTML using a basic text editor?
  - may want low-level control
  - may care about size/readability of pages
  - may want to "steal" page components and integrate into existing pages
  - may want dynamic features such as scripts or applets
  - May want to edit a page created by a web authoring tool

Tags vs. Elements

- HTML specifies a set of tags that identify structure and content type
  - tags are enclosed in < >

     specifies an image
  - most tags come in pairs, marking a beginning and ending
    - <title> and </title> enclose the title of a page
- Example: an HTML *element* is an object enclosed by a pair of tags

  `<title>My Home Page</title>` is a TITLE element

  `<b>This text appears bold.</b>` is a BOLD element

  `<p>Part of this text is <b>bold</b>.</p>`
  is a PARAGRAPH element that contains a BOLD element

Structural Elements

- an HTML document has two main structural elements
  - HEAD contains setup information for the browser & the Web page
    - e.g., the title for the browser window, style definitions, JavaScript code, …
  - BODY contains the actual content to be displayed in the Web page

```
<html>
 <!-- Version information --
 -- File: page01.html --
 -- Author: COMP519 --
 -- Creation: 15.08.06 --
 -- Description: introductory page --
 -- Copyright: U.Liverpool --
 -->
 <head>
  <title>My first HTML document</title>
 </head>
 <body>
  Hello world!
 </body>
</html>
```

view page

HTML documents begin and end with <html> and </html> tags

Comments appear between <!-- and -->

HEAD section enclosed between <head> and </head>

BODY section enclosed between <body> and </body>

** Find more info on HTML docs!*

Text Layout

```
<html>
<!-- COMP519 page02.html 30.08.05 -->
<head>
  <title>Text Layout</title>
</head>

<body>
  <p>
  This is a paragraph of text<br/>
  made up of two lines.
  </p>

  <p>
  This is another paragraph with a
    GAP   between
  some of the words.
  </p>

  <p>
     This paragraph is<br/>
  indented on the first line<br/>
  but not on subsequent lines.
  </p>
</body>

</html>
```

- for the most part, layout of the text must be left to the browser
  - every sequence of whitespace is interpreted as a single space
  - browser automatically wraps the text to fit the window size
- can override some text layout
  - can cause a line break using the
 tag (no closing tag)
  - can specify a new paragraph (starts on a new line, preceded by a blank line) using …
  - can force a space character using the symbol for a non-breaking space:

view page

Separating Blocks of Text

```
<html>
<!-- COMP519  page03.html  15/08/06 -->
<head>
  <title>Blocks of Text</title>
</head>

<body>
  <h1>Major heading 1</h1>
  <p>
  Here is some text.
  </p>

  <h2>Subheading</h2>
  <p>
  Here is some subtext.
  </p>

  <hr/>

  <h1>Major heading 2</h1>
  <p>
  Here is some more text.
  </p>
</body>

</html>
```

- can specify headings for paragraphs or blocks of text
  - <h1>…</h1> tags produce a large, bold heading
  - <h2>…</h2> tags produce a slightly smaller heading
    - ...
  - <h6>…</h6> tags produce a tiny heading
- can insert a horizontal rule to divide sections
  - <hr/> draws line across window
  - <hr width="50%" /> sets width
  - <hr size="10" /> sets thickness

view page

The Basic Web page – A Worked Example

```
<html>
<!-- COMP519  page22.html  15.08.06 -->
   <head>
      <title> Bill Smiggins Inc. </title>
   </head>
   <body>
      <h1>Bill Smiggins Inc.</h1>
      <h2>About our Company...</h2>
      <p>This Web site provides clients, customers,
         interested parties and our staff with all of
         the information that they could want on
         our products, services, success and failures.
      </p>
      <hr/>
      <h3> Products </h3>
      <p> We are probably the largest
      supplier of custom widgets, thingummybobs, and bits
      and pieces in North America. </p>
      <hr width="50%"/>
   </body>
</html>
```

view page

Text Appearance

```
<html>
 <!-- COMP519  page25.html  15.08.06 -->
 <head>
  <title>Text Variations and Escape Sequences<
/title>
 </head>
 <body>
  <h1>Text Variations</h1>
  <p>We can use <b>simple</b> tags to
     <i>change</i> the appearance of
     <strong>text</strong> within
     <tt>Web pages</tt>.
     Even super<sup>script</sup>
     and sub<sub>scripts</sub> are
      <em>supported</em></p>

   <h1>Text Escape Sequences</h1>
  <p>
    &amp; &lt; &gt; &quot; &copy;
  </p>
 <h1>Preformatted text</h1>
      <pre>
         University of Liverpool
         Department of Computer Science
         Chadwick Building, Peach Street
         Liverpool, L69 7ZF, UK
      </pre>
</body>
```

- can specify styles for fonts
  - … specify bold
  - … specify italics
  - <tt>… </tt> specify typewriter-like (fixed-width) font
  - <big>… </big> increase the size of the font
  - <small>… </small> decrease the size of the font
  - … put emphasis
  - … put even more emphasis
  - _… specify a subscript
  - […] a superscript
  - <pre>…</pre> include ready-formatted text
  - & < > " © escape characters used in HTML control

Lists

```
<html>
<!-- COMP519 page07.html  15.08.06 -->
<head>
  <title>Simple Lists</title>
</head>
<body>
<p>

<ul style="list-style-type: square;">
 <li> ... first list item... </li>
 <li> ... second list item... ... </li>
</ul>
<dl>
 <dt> Dweeb </dt>
   <dd> young excitable person who may
  mature into a <em>Nerd</em> </dd>
 <dt> Hacker </dt>
     <dd> a clever programmer </dd>
 <dt> Nerd </dt>
    <dd> technically bright but
        socially inept person </dd>
</dl>
 <ol style="list-style-type: upper-roman;" >
  <li value="30">makes item number 30.</li>
  <li value="40">makes item number 40.</li>
  <li> makes item number 41. </li>
</ol>
</p>
</body>
```

- there are 3 different types of list elements
  - … specifies an ordered list (using numbers or letters to label each list item)
    - identifies each list item
    - *can set type of ordering, start index*
  - … specifies unordered list (using a bullet for each)
    - identifies each list item
  - <dl>…</dl> specifies a definition list <dt> identifies each term
    - <dd> identifies its definition

* We will learn more about the "style" attributes soon enough.

view page

Hyperlinks

```
<html>
<!-- COMP519 page08.html  15.08.06 -->

<head>
  <title>Hyperlinks</title>
</head>

<body>
  <p>
  <a href="http://www.liv.ac.uk">
  The University of Liverpool</a>
  <br/>
  <a href="page07.html" target="_blank">
   Open page07 in a new window</a>
  </p>
</body>

</html>
```

view page

- perhaps the most important HTML element is the hyperlink, or ANCHOR
  - …
    - where URL is the Web address of the page to be displayed when the user clicks on the link
    - *if the page is accessed over the Web, must start with* http://
    - *if not there, the browser will assume it is the name of a local file*
  - …
    - causes the page to be loaded in a new
    - Window

* *Find more info on attribute TARGET*

Hyperlinks (cont.)

```
<html>
<!-- COMP519  page09.html  15.08.06 -->

<head>
  <title>Internal Links in a Page</title>
</head>

<body>
  <p>
  [ <a href="#HTML">HTML</a> |
    <a href="#HTTP">HTTP</a> |
    <a href="#IP">IP</a> |
    <a href="#TCP">TCP</a> ]
  </p>
  <p>
  Computer acronyms:
  <dl>
   <a name="HTML"></a><dt>HTML</dt>
   <dd>HyperText Markup Language
   <a name="HTTP"></a><dt>HTTP</dt>
   <dd>HyperText Transfer Protocol…</dd>
   <a name="IP"></a><dt>IP</dt>
   <dd>Internet Protocol…</dd>
   <a name="TCP"></a><dt>TCP</dt>
   <dd>Transfer Control Protocol…</dd>
  </dl>
  </p>
</body>
```

- for long documents, you can even have links to other locations in that document
  - …
    - where ident is a variable for identifying this location
  - …
    - will then jump to that location within the file
  - …
    - can jump into the middle of another file just as easily

view page

Images

- Can include images using IMG
  - by default, browsers can display GIF and JPEG files
  - other image formats may require plug-in applications for display
    -
    - again, if file is to be accessed over the Web, must start with http:// (if not, will assume local file)
    - * Find more info on <img/>

```
<html>
<!-- COMP519  page10.html  15.08.06 -->
<head>
  <title>Images</title>
</head>
<body>
<img src="http://www.csc.liv.ac.uk/~martin/teaching/comp519/HTML/Blair.jpg"
 alt="Blair" title="Tony Blair"/>
  <p>Former Prime Minister Tony Charles Lynton Blair</p>
</body>
</html>
```

view page

Tables

- Tables are common tools for arranging complex layout on a Web page
  - a table divides contents into rows and columns
  - by default, column entries are left-justified, so provide for alignment

```
<html>
<!-- COMP519  page11.html  15.08.06 -->
<head>
  <title>Tables</title>
</head>
<body>
<h2>A Simple Table</h2>
  <table>
   <tr>
     <td> Left Column </td>
     <td> Right Column </td>
   </tr>
   <tr>
     <td> Some data </td>
     <td> Some data </td>
   </tr>
 </table>
</body>
</html>
```

<table>…</table> specify a table element

<tr>…</tr> specify a row in the table

<td>…</td> specify table data (i.e., each column entry in the table)

view page

Layout in a Table

```
<html>
<!-- COMP519  page12.html  15.08.06 -->

<head>
  <title>Table Layout</title>
</head>

<body>
  <table style="border: 1px solid;">
     <tr style="text-align: center;">
     <td style="border: 1px solid;">
           Left<br/>Column</td>
     <td style="border: 1px solid;
         vertical-align: top;">
           Right Column</td>
    </tr>
    <tr>
      <td style="border: 1px solid;">
          Some data</td>
      <td style="border: 1px solid;">
          Some data</td>
    </tr>
  </table>
</body>
</html>
```

- can have a border on tables using the "border" attribute
  - <table style= "border: 1px solid;">
  - *increasing the number makes the border thicker*
- can control the horizontal & vertical layout within cells
  - <td style= "text-align:center">
  - <td style= "vertical-align: bottom">
- can apply layout to an entire row
  - <tr style="text-align: center">
  - <tr style="vertical-align: top">

We will explore this more with Cascading Style Sheets.

view page

Table Width

```
<html>
<!-- COMP519  page13.html  15.08.06 -->
<head>
  <title>Table Width</title>
</head>

<body>
<table style="width: 100%;">
    <tr>
      <td>left-most </td>
      <td style="text-align: right;">
              right-most</td>
    </tr>
</table>
</body>
</html>
```

view page

- by default, the table is sized to fit the data
- can override & specify the width of a table relative to the page
  - <table style="width: 60%">

Other Table Attributes

```
<html>
<!-- COMP519  page14.html  15.08.06 -->
<head>
  <title>Table Formatting</title>

 <style type="text/css" media="screen">
     table { border: 1px solid; padding: 1px;}
     th, td {  border: 1px solid; padding: 10px;
               text-align: center; }
  </style>
</head>

<body>
  <table>
    <tr>
      <th>HEAD1</th> <th>HEAD2</th> <th>HEAD3</th>
    </tr>
    <tr>
      <td>one</td> <td>two</td> <td>three</td>
    </tr>
    <tr>
      <td rowspan="2"> four </td>
      <td colspan="2"> five </td>
    </tr>
    <tr>
      <td> six </td> <td> seven </td>
    </tr>
</table>
</body>
```

view page

- can control the space between cells & margins within cells

  This is the "padding" attribute in the table and th,td style sheet declarations (more on this with Cascading style shets).

- can add headings

  `<th>` is similar to `<td>` but displays heading centered in bold

- can have data that spans more than one column

  `<td colspan="2">`

- similarly, can span more than one row

  `<td rowspan="2">`

(This example uses CSS style sheet commands in the page <header>.)

Frames

- Frames provide the ability to split the screen into independent parts
  - Frames are going out of fashion, partly because they interact poorly with web search engines (i.e. search engines cannot generally access the data stored in the inset frame objects).
  - Frames can also "break" the regular behavior of browsers, most notably the "Back" button on the browser can behave in unexpected ways.

Content vs. Presentation

- Most HTML tags define content type, independent of presentation
  - Exceptions?
- Style sheets associate presentation formats with HTML elements
  - CSS1: developed in 1996 by W3C
  - CSS2: released in 1998, but not fully supported by browsers
  - CSS3: under development
  - HTML style sheets are known as Cascading Style Sheets, since can be defined at three different levels
    - (1) inline style sheets apply to the content of a single HTML element
    - (2) document style sheets apply to the whole BODY of a document
    - (3) external style sheets can be linked and applied to numerous documents
    - lower-level style sheets can override higher-level style sheets

How does (CSS) fit in? (1)

- Offer more control over how elements are rendered by writing something like:

```
<FONT style="font-size:12px"> This font is size 12 pixels</FONT>
```

- Not perfect (Netscape renders a fraction smaller) but best you'll get.

How does (CSS) fit in? (2)

- If you don't want browser to underline links and colour them red when rolled over, try this:

```
<a href="info.htm" style="color:black; text-decoration:none;">Info</a>
```

- First you need understand just what CSS actually is, and how to use it.

What exactly is CSS?

- CSS is a language, or a micro-language that's integrated into HTML, principally from the v4 browsers (which means IE4 and NS4).
  - It uses its own syntax
  - It doesn't interfere with old browsers
  - It offers a stronger, more predictable way to control appearances (font sizes to margins, backgrounds, forms, layers etc.)

Where does it come from?

- There is only one CSS, as set down by the World Wide Web Consortium (W3C)
  - www.w3c.org
- These CSS standards are implemented by those who create the browsers (e.g. Firefox, Microsoft, Netscape).

CSS Attribute Structure (1)

- In short, it pairs attributes (e.g. size, color) with values (e.g. 4px, #003366). An attribute picks out a property, the value says what it should be.
- Example:

```
<p style="color: #00FF00; margin-left: 10px">...</p>
```

CSS Attribute Structure (2)

- Here's another example:

```
<P style="width: 50px;
  border-style: solid; border-width: 1px; border-color:
  black; padding: 5px"> Here's some words</P>
```

- This is fantastic! Why? Using basic HTML?

view page

CSS Attribute Structure (3)

- The basic syntax for CSS is:

attribute: value;

such as:

font-weight: bold

color: #330033

- With a semi-colon between pairs:

font-weight: bold; color: #330033

How do we use CSS in HTML? (1)

- You can apply a CSS style directly within a tag in your main body HTML.
- Just use style="..." within the tag (just as you would use color="..." or href="...")and write your CSS between the quotes:

```
<tagname style=" … ">
   Values</tagname>
```

How do we use CSS in HTML? (2)

- An example of the style attribute in work with the H1 tag would be:

```
<H1 style=" … ">
  Your text here</H1>
```

- This method of applying a style to HTML is called inline styles, because the style attribute is in the tag it wants to apply itself to.

How do we use CSS in HTML? (3)

- You can use this **style="..."** syntax in any tag you want providing it makes CSS sense to apply that attribute to that tag.
- e.g. You can't apply a margin to a word in the middle of a sentence, but only to what are called '*block level elements*':
  - <P>, <H1>, <H2> etc.

The Span & Div tags (1)

- But what if you do want to apply a style to a word in the middle of a paragraph - make it blue, bold and italics, say?
- In 'traditional' HTML you would write this:

```
<FONT color="blue">
<B><I>…  </I></B>
</FONT>
```

The Span & Div tags (2)

- In CSS lingo we write it as:

```
style= "color: blue; font-weight: bold;
        font-style: italic"
```

- But now you don't need any of the block tags, where do you put the style="..."?
- Could use any of these tags; but the favoured means is to use the "do nothing" **span** tag:

The Span & Div tags (3)

<SPAN style="color:blue; font-weight:bold; font-style:italic"**>**...

</SPAN>

- <SPAN> has no particular meaning or attachment - it simply means: 'Do these things to whatever's within me'.
- <DIV> is similar, but is for block level elements

Inline Style Sheets

```
<html>
<!-- COMP519  page17.html  15.08.06 -->

<head>
  <title>Inline Style Sheets</title>
</head>

<body>
  <p style="font-family:Arial,sans-serif;
            text-align:right">This is a
  right-justified paragraph in a sans serif
  font (preferably Arial), with some
  <span style="color:green">green text</span>.
  </p>

  <p>And <a style="color:red;
                   text-decoration:none;
                   font-size:larger;"
           href="page01.html">here</a>
     is a formatted link.
  </p>
</body>
</html>
```

view page

- Using the `style` attribute, you can specify presentation style for a single HTML element
  - within tag, list sequence of property:value pairs

font-family:Courier,monospace
font-style:italic
font-weight:bold
font-size:12pt font-size:large font-size:larger

color:red color:#000080
background-color:white

text-decoration:underline
text-decoration:none
text-align:left text-align:center
text-align:right text-align:justify
vertical-align:top vertical-align:middle
vertical-align:bottom

text-indent:5em text-indent:0.2in

Inline Style Sheets (cont.)

```
<html>
<!-- COMP519  page18.html  15.08.06 -->

<head>
  <title>Inline Style Sheets</title>
</head>

<body>
  <p>Here is an image
    <img src="Blair.jpeg" alt="Tony Blair"
         style="margin-left:0.3in;
                margin-right:0.3in;
                vertical-align:middle;
                border-style:double;
                border-color:yellow" />
    embedded in text.
  </p>

  <ol style="list-style-type:upper-alpha">
    <li> one thing</li>
    <li> or another</li>
    <ul style="list-style-type:square;
               whitespace:pre">
      <li> with  this</li>
      <li> or    that</li>
    </ul>
  </ol>
</body>
```

- more style properties & values

margin-left:0.1in margin-right:5%
margin:3em
padding-top:0.1in padding-bottom:5%
padding:3em

border-width:thin border-width:thick
border-width:5
border-color:red
border-style:dashed border-style:dotted
border-style:double border-style:none

whitespace:pre

list-style-type:square
list-style-type:decimal
list-style-type:lower-alpha
list-style-type:upper-roman

view page

Inline Style Sheets (cont.)

```
<html>
<!-- COMP519  page19.html 15.08.06 -->

<head>
  <title> Inline Style Sheets </title>
</head>

<body>
  <table style="font-family:Arial,sans-serif">
    <caption style="color:red;
                    font-style:italic;
                    text-decoration:underline">
       Student data. </caption>
    <tr style="background-color:red">
      <th> name </th> <th> age </th>
    </tr>
    <tr>
      <td> Chris Smith </td> <td> 19 </td>
    </tr>
    <tr>
      <td> Pat Jones </td> <td> 20 </td>
    </tr>
    <tr>
      <td> Doogie Howser </td> <td> 9 </td>
    </tr>
</table>
</body>
```

- Style sheets can be applied to tables for interesting effects

view page

Document Style Sheets

- Inline style sheets apply to individual elements in the page
  - using inline style directives can lead to inconsistencies, as similar elements are formatted differently
    - e.g., we might like for all <h1> elements to be centered
  - inline definitions mix content & presentation
    - ➔ violates the general philosophy of HTML
- alternatively, document style sheets allow for a cleaner separation of content and presentation
  - style definitions are placed in the <head> of the page (within STYLE tags)
  - can apply to all elements, or a subclass of elements, throughout the page

Style Definitions (1)

- Style Definitions are pieces of CSS code placed in the head section of your page.
- In them you can change a particular HTML element (e.g. <p>) to reflect certain styles.

Style Definitions (2)

- A sample style definition:

```
a {
   text-decoration: none; color: black;
   font-family: Verdana; font-size: 12pt;
  }
```

- Whatever styles you define for a tag in the HEAD section, will apply to all instances of that tag in your HTML body (really useful!!)

Document Style Sheets

```
<html>
<!-- COMP519  page20.html  15.08.06 -->

<head>
  <title>Document Style Sheets</title>
  <style type="text/css">
     h1 {color:blue;
         text-align:center}
     p.indented {text-indent:0.2in}
  </style>
</head>

<body>
  <h1> Centered Title </h1>

  <p class="indented">This paragraph will
 have the first line indented, but
 subsequent lines will be flush. </p>

  <p>This paragraph will not be indented.
  </p>

 <h1> The End </h1>

</body>
```

- document style sheets ensure that similar elements are formatted similarly
  - can even define subclasses of elements and specify formatting

  `p.indented` defines subclass of paragraphs
  - inherits all defaults of <p>
  - adds new features

  to specify this newly defined class, place class="ID" attribute in tag

- note how "clean" the <body> is

view page

Document Style Sheets (cont.)

```
<html>
<!-- COMP519  page21.html  15.08.06 -->

<head>
  <title> Inline Style Sheets </title>
 <style type="text/css">
     table {font-family:Arial,sans-serif}
     caption {color:red;
              font-style:italic;
              text-decoration:underline}
     th {background-color:red}
  </style>
</head>

<body>
  <table>
    <caption> Student data. </caption>
    <tr><th> name </th>          <th> age</th></tr>
    <tr><td> Chris Smith </td>   <td> 19 </td></tr>
    <tr><td> Pat Jones </td>     <td> 20 </td></tr>
    <tr><td> Doogie Howser </td> <td> 9 </td></tr>
</table>
</body>
</html>
```

- document style sheets are especially useful in formatting tables
- effectively separates content from presentation

what if you wanted to right-justify the column of numbers? what if you changed your mind?

view page

CSS selectors (1)

- A form of Style Definition used for mouse rollovers
  - Supported by later browsers
  - Can define **a:link**, **a:active**, **a:hover**,
  - Simply applies the selected style when the corresponding action is triggered (e.g. on it's own, when clicked on, on mouse over)

CSS selectors (2)

```
a:link {    text-decoration: none;
            color: black; font-weight: bold    }
a:hover {
            text-decoration: underline;
            color: darkgreen
}
```

- And your plain, black, but bold link will underline and turn dark green on rollover.

Grouping

- In order to decrease repetitious statements within style sheets, grouping of selectors and declarations is allowed.

```
H1, H2, H3, H4, H5, H6 {
  color: red;
  font-family: sans-serif
}
```

Inheritance

- Virtually all selectors which are nested within selectors will inherit the property values assigned to the outer selector unless otherwise modified.
  - A colour defined for the BODY will also be applied to text in a paragraph.

Classes (1)

- There are two ways of creating styles.
  - You can redefine an existing HTML tag
    e.g. <body>, <p>, <font> <a>
  - You can create a brand new style, which you can apply to any tag you wish.
    - HTML does not supply any tag to do a particular type of style for the sub headings in a news page;
    - So instead of redefining a <h2> tag, for example, one can create a **new style** called 'subhead'.
    - This new type of style is called a **class**.

Classes (2)

- The definition for this subhead would be:

```
<style>
<!--
      . subhead {    font-family: Arial;
                     color: #0066CC;
                     font-size: 18pt
                  }
-->
</style>
```

Classes (3)

- Notice the '.' in front of the style name?
  - This is to indicate to the browser that this style is not redefining a HTML tag, but instead creating a class.

Applying a Class to an Element (1)

- Surprisingly enough, it's done through the class=" … " attribute. For example:

```
.warning {
  color: lime; background: #ff80c0  }
```

… used in HTML with the CLASS attribute …

```
<h1 CLASS="warning">STOP!</h1>
```

Applying a Class to an Element (2)

- In this example, the warning class may be applied to any BODY element since it does not have an HTML element associated with it in the style sheet.
- Notice also that we left out the '.' when using the class attribute? (Rule: only put in when defining)

Applying a Class to an Element (3)

- We can also define classes to work only with certain tags. For example:

```
p.news {
font-weight: bolder;
color: red; background: white
  }
```

- In this example, the news class may only be applied to the P element.

Applying a Class to an Element (4)

```
<p class="news">Due to the recent traffic crisis in
  Galway city, the Government have decided to abolish
  the road taxes, and instead introduce rent.</p>
```

- The next example will not work:
  - the news class only works with the <p> tag

```
<h1 class="news">STOP!</h1>
```

Cascading Concept (1)

- As a final note, throughout all this wondrous stuff, you may have been asking: "Yes, but what the heck is '*cascading*'? "
- CSS has a tree-like structure, and one definition or class effects the things that come lower in the structure, or within its scope, unless another CSS rule comes into play, saying something different.

Cascading Concept (2)

- So, for instance, if you define:
  - The <p> tag as: black 12px Arial
  - A .link class as: blue Verdana
- Then think about this:

```
<p>For a more info,
  <a href="..."
   class="link"
        style="color: green">click here</a>
  or else </p>
```

Cascading Concept (3)

- What's going on?!
- This may seem difficult, but in practice it isn't, and makes good sense
  - Plus it gives great flexibility to the designer to manipulate elements at any stage
- So go forth and use it!

Linking to an external Style Sheet (1)

- This method of defining styles in the <head> section of the page is a lot more efficient than defining each item individually, as well as decreasing file size.
- If we have a large website however, we don't want to have to define these styles on every page.

Linking to an external Style Sheet (2)

- CSS lets us create a file, where we can create our styles, and link this style file or **style sheet** to every page in our website, thus eliminating the need to type out the styles for every page.

Linking to an external Style Sheet (3)

- To link a file to a style sheet, place the following code in the head section of our page:

```
<LINK
  REL="stylesheet"
  TYPE="text/css"
  HREF="style.css">
```

Linking to an external Style Sheet (4)

- This creates a link between our webpage and the style sheet, and all styles in the style sheet are adopted for the current webpage.
- It is important to know that external Style Sheets can only contain CSS specific mark-up.
  - In other words, it CANNOT contain any HTML, only what you would normally place in the head section when defining styles.

Modularity & Style Sheets

```
<html>
<!-- COMP519  page26.html 15.08.06 -->

<head>
  <title>Title for Page</title>
  <link rel="stylesheet"
        type="text/css"
        href="myStyle.css"
        title="myStyle“ />
</head>

<body>
 <h1>Centered Title</h1>

 <p class="indented">This paragraph will
 have the first line indented, but
 subsequent lines will be flush.</p>

 <p>This paragraph will not be indented.
 </p>

 <h1>The End</h1>

</body>
</html>
```

```
/* myStyle.css  COMP519 02.09.05 */
h1 {color : blue; text-align : center}
p.indented {text-indent:0.2in}
```

- ideally, the developer(s) of a Web site would place all formatting options in an external style sheet
- all Web pages link to that same style sheet for a uniform look
  - simplifies Web pages since only need to specify structure/content tags

view page

and Tags

- Problem: the font properties apply to whole elements, which are often too large
  - Solution: a new tag to define an element in the content of a larger element -
  - The default meaning of is to leave the content as it is

```
<p> Now is the <span> best time </span> ever! </p>
```

- Use to apply a document style sheet to its content

```
<style type = "text/css">
 .bigred {font-size: 24pt;
 font-family: Ariel; color: red}
 </style>
 <p> Now is the <span class = "bigred">
     best time </span> ever!
 </p>
```

- The tag is similar to other HTML tags, they can be nested and they have id and class attributes

view page

- Another tag that is useful for style specifications:

Used to create document sections (or divisions) for which style can be specified
e.g., a section of five paragraphs for which you want some particular style

Sample Style Sheet (1)

```
/* This is a comment */
a {     font-family: Verdana; font-size: 10pt;
    color: #0000FF; text-decoration: none  }
a:visited {    color: #0000FF;
          text-decoration: none   }
a:hover {    color: #0000FF;
          text-decoration: underline     }
a:active {    color: #FF8000    }
```

Sample Style Sheet (2)

- body { font-family: Verdana; font-size: 10pt; font-style: normal; font-weight: normal; color: #000000; background-color: #FFFFFF }
- td { font-family: Verdana; font-size: 10pt; font-style: normal; font-weight: normal; color: #000000; background-color: #FFFFFF }
- h1 { font-family: Verdana, "Times New Roman", Times, serif; font-size: 9px; color: #666666 }

12 Quick CSS Effects (1)

- Control your text size
  - Ever get really miffed that setting <FONT size="n"> for your text never properly controls the result? And moreover, the sizes 1, 2, 3, 4 and so on don't leave much room for precision.
  - The CSS font-size property offers far more control. Try this:

```
<p style="font-size: 12px">
   Put your text in here</p>
```

12 Quick CSS Effects (2)

- Format all your text in one fell swoop
  - Why mess up your HTML applying font tags to every line of body text? This style rule should do for just about every line of body text you use:

```
p {  font: normal 11px
        Verdana, Arial, Helvetica, sans-serif  }
td {  font: normal 11px
           Verdana, Arial, Helvetica, sans-serif  }
```

12 Quick CSS Effects (3)

- Centralise your heading styles
  - Now you can set up sub-styles for headings, subheadings and so on in just one place - so you can reformat the whole lot by changing just one line. Add this to your style sheet:

```
.subhead {  font-size: 14px; font-weight: bold;  }
```

  - Then for the subhead, write:

```
<p class="subhead">Subhead in here</p>
```

12 Quick CSS Effects (4)

- Get clever with links
  - The default behaviour for links - underlined and turning red on rollover isn't to every designer's taste.

```
a { color: #003366; text-decoration: none }
a:hover {
        color: white;
        background-color: #003366 }
```

  - Now there's no underline, but the links have a blue-green background colour when you roll over.

12 Quick CSS Effects (4)

Property	Values
a:link	Define the style for unvisited links
a:visited	Define the style for visited links
a:active	Define the style for active link (when you click on it)
a:hover	Define the style for hovered link (when mouse move over it)

Pseudo-Elements

```
<html>
<!-- COMP519  page23.html  15.08.06 -->

<head>
  <title>Title for Page</title>
  <style type="text/css">
      a {color : red;
         text-decoration : none;
         font-size : larger}
      a:visited {color : blaack}
      a:active {color : orange}
      a:hover {color : blue}
      p:first-letter {font-size : large;
              color : white;
              background-color : darkblue}
  </style>
</head>

<body>
  <p> Welcome to my Web page.  I am so
  happy you are here.
  </p>
  <p> Be sure to visit
  <a href="http://www.cnn.com">CNN</a>
  for late-breaking news.
  </p>
</body>
```

- pseudo-elements are used to address sub-parts of elements
  - can specify appearance of link in various states
    - `:visited` `:active` `:hover`
  - can specify format of first line in page or paragraph
    - `:first-line`
  - can specify format of first letter in page or paragraph
    - `:first-letter`

- *Danger :* changing the look of familiar elements is confusing

- *Careful :* current browsers do not support all CSS2 features

view page

12 Quick CSS Effects (5)

- Create text margins
  - Margins are a nightmare to create in old HTML, fiddling about with tables and never quite knowing how the widths are going to turn out.
    Forget it: turn to CSS instead:

```
<p style="margin-left: 10px">...</p>
```

  - Simple, eh?

12 Quick CSS Effects (6)

- Funky cursor effects
  - In a browser, you can change the cursor that appears when you roll over a link - or images, text and so on that don't have links. How? Try this style rule:

```
a {    cursor: crosshair    }
```

  - Alternatives to crosshair include hand, text, help, wait, and various resize options: n-resize (for a North angle), ne-resize (for Northeast), and so on.

12 Quick CSS Effects (7)

- Juicy quote
  - Ever wanted to have tempting quotes appearing in large italics through the main text of your articles (or callouts as they call them in the magazine business)? No probs. Just put this before a <P> tag:

```
<DIV style="
  width: 130px;
  float: right;
  color: maroon;
  font-size: 18px;
  font-style: italic;
  font-weight: bold">
&quot;Here's your juicy
  quote&quot;  </div>
```

12 Quick CSS Effects (8)

- Designer forms
  - Web forms look pretty dull in the default scheme of things, but you can easily style them up a bit with CSS. It only works in IE and Netscape after V6, but it's well worth it.
  - The effect appears as a black form input field and the text comes out white.

```
<FORM>
<INPUT type="text"
name="textfield" style="
  color: white;
  background-color: black;
  font: 11px Verdana,Helvetica"
value="Enter your keywords"
size="30">
</FORM>
```

12 Quick CSS Effects (9)

- Form sizes
  - Another form-related stroke of CSS genius: if you test your work on all the browsers and platforms, you'll find it's nigh on impossible to get the widths of inputs, dropdowns and so on the same for all viewers - which is a pain if you're after well-spaced design. You can take 95 per cent control of this using the width CSS property, which works on forms for IE, and using the traditional size="" for NS:

```
<INPUT type="text"
  style="width: 200px" size="30">
```

12 Quick CSS Effects (10)

- Enter keywords...
  - Not really CSS this, but while we're on forms, here's a useful JavaScript trick that people always ask about: how to include words in a form field, such as 'Enter keywords', which clear when they click in the space. Simple:

```
<INPUT type="text" name="textfield" value="Enter
  your keywords" onfocus="this.value="" size="30">
```

  - (After this.value=, that's two single quotes followed by a double quote)

12 Quick CSS Effects (11)

- Cool scrollbars
  - Did you know that in IE you can change the colors of the scollbars at the right and bottom of your page? Pretty cool.
  - Twiddle the colour # references to suit yourself.

```
body {
scrollbar-face-color: #2A314C;
scrollbar-shadow-color: #2A314C;
scrollbar-highlight-color: #2A314C;
scrollbar-3dlight-color: #9AB6C4;
scrollbar-darkshadow-color: #20253A;
scrollbar-track-color: #20253A;
scrollbar-arrow-color: #CCCCCC
}
```

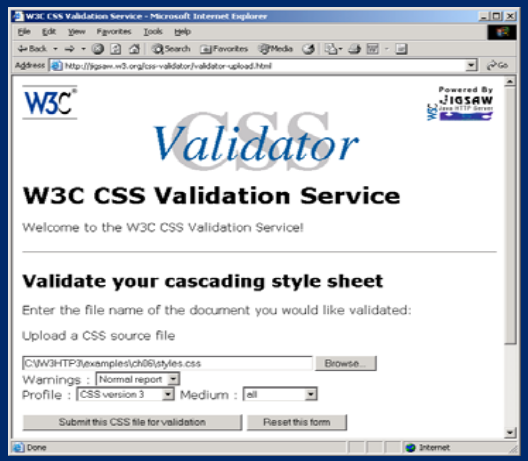
12 Quick CSS Effects (12)

- Background control
  - People often ask about how the pros create the effect where you have a background graphic which doesn't tile and repeat; or which doesn't move when you scroll the page. The answer, of course, is using CSS, and here it is:

```
body {
background-image: url(background.gif);
background-repeat: no-repeat;
background-attachment: fixed
}
```

- Replace background.gif with the URL of the image.
- background-repeat can be no-repeat, repeat, repeat-x or repeat-y.
  Attachment can by fixed or scroll.

Validating your Style Sheet



Forms and Other Issues ☺

Creating and Publishing a Web Page

1. Create an HTML document
2. Place the document in a world-accessible directory (often `public_html` or `www`) on a system running an HTTP server

```
Unix> cd
Unix> chmod a+x .   (Note the ".")
Unix> mkdir public_html
Unix> chmod a+x public_html
```

3. Access the web page through http://hostname/~username/filename
   - E.g. **`http://www.apl.jhu.edu/~lmb/test.html`**
   - If the filename is omitted, a system default filename is assumed (often `index.html`)
     - E.g. **`http://www.apl.jhu.edu/~hall/`** refers to the file **`index.html`** in hall's **`public_html`** directory

What are forms?

- <form> is just another kind of HTML tag
- HTML forms are used to create (rather primitive) GUIs on Web pages
  - Usually the purpose is to ask the user for information
  - The information is then sent back to the server
- A form is an area that can contain form elements
  - The syntax is: <form *parameters*> *...form elements...* </form>
  - Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
    - Other kinds of HTML tags can be mixed in with the form elements
  - A form usually contains a Submit button to send the information in he form elements to the server
  - Forms can be used to send the information to the server or for a simple GUI program

The <form> tag

- The <form *arguments*> … </form> tag encloses form elements (and probably other HTML as well)
- The arguments to form tell what to do with the user input
  - action="*url*" (required)
    - Specifies where to send the data when the Submit button is clicked
  - method="get" (default)
    - Form data is sent as a URL with ?form_data info appended to the end
    - Can be used *only* if data is all ASCII and not more than 100 characters
  - method="post"
    - Form data is sent in the body of the URL request
    - Cannot be bookmarked by most browsers
  - target="*target*"
    - Tells where to open the page sent as a result of the request
    - *target*= _blank means open in a new window
    - *target*= _top means use the same window

The <input> tag

- Most, but not all, form elements use the input tag, with a type="…" argument to tell which kind of element it is
  - type can be text, checkbox, radio, password, hidden, submit, reset, button, file, or image
- Other common input tag arguments include:
  - name: the name of the element
  - value: the "value" of the element; used in different ways for different values of type
  - readonly: the value cannot be changed
  - disabled: the user can't do anything with this element
  - Other arguments are defined for the input tag but have meaning only for certain values of type

Text input

A text field:

```
<input type="text" name="textfield" value="with an initial value">
```

A multi-line text field

```
<textarea name="textarea" cols="24" rows="2">Hello</textarea>
```

A password field:

```
<input type="password" name="textfield3" value="secret">
```

- Note that two of these use the input tag, but one uses textarea

Buttons

- A submit button:

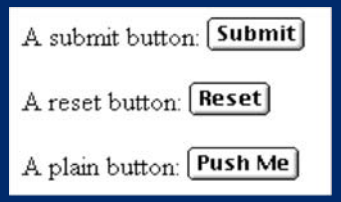
```
<input type="submit" name="Submit" value="Submit">
```

- A reset button:

```
<input type="reset" name="Submit2" value="Reset">
```

- A plain button:

```
<input type="button" name="Submit3" value="Push Me">
```



- submit: send data
- reset: restore all form elements to their initial state
- button: take some action as specified by JavaScript

- Note that the type is input, not "button"

Checkboxes

- A checkbox:

```
<input type="checkbox" name="checkbox"
       value="checkbox" checked>
```

A checkbox: ☑

  - type: "checkbox"
  - name: used to reference this form element from JavaScript
  - value: value to be returned when element is checked
  - Note that there is *no text* associated with the checkbox—you have to supply text in the surrounding HTML

Radio buttons

```
Radio buttons:<br>
  <input type="radio" name="radiobutton" value="myValue1">
  male<br>
  <input type="radio" name="radiobutton" value="myValue2" checked>
  female
```

Radio buttons:
○ male
◉ female

- If two or more radio buttons have the same name, the user can only select one of them at a time
  - This is how you make a radio button "group"
- If you ask for the value of that name, you will get the value specified for the selected radio button
- As with checkboxes, radio buttons do not contain any text

Drop-down menu or list

- A menu or list:

```
<select name="select">
    <option value="red">red</option>
    <option value="green">green</option>
    <option value="BLUE">blue</option>
</select>
```

A menu or list: red

  - Additional arguments:
    - size: the number of items visible in the list (default is "1")
    - multiple: if set to "true", any number of items may be selected (default is "false")

Combo Boxes

- Format
  - SELECT gives NAME
  - OPTION gives VALUE
- Example

```
Favorite language:
<select NAME="language" Multiplr>
  <Option VALUE="c">C
  <Option VALUE="c++">C++
  <Option VALUE="java" SELECTED>Java
  <Option VALUE="lisp">Lisp
  <Option VALUE="perl">Perl
  <Option VALUE="smalltalk">Smalltalk
</select >
```

Languages you know:
C
C++
Java
Lisp
Perl
Smalltalk

Hidden fields

- <input type="hidden" name="hiddenField" value="nyah">
  <-- right there, don't you see it?

A hidden field: <-- right there, don't you see it?

- What good is this?
  - All input fields are sent back to the server, including hidden fields
  - This is a way to include information that the user doesn't need to see (or that you don't want him/her to see)
  - The value of a hidden field can be set programmatically (by JavaScript) before the form is submitted

Other Controls and Options

- File upload controls
  - Lets user select a file and send it to the server
- Server-side image maps
  - User clicks on an image and form gets submitted.
  - Form data gets sent as *name*.x=*x-pos*&*name*.y=*y-pos*

A complete example

```
<html>
<head>
<title>Get Identity</title>
<meta http-equiv="Content-Type" content="text/html;
      charset=iso-8859-1">
</head>
<body>
<p><b>Who are you?</b></p>
<form method="post" action="">
  <p>Name:
    <input type="text" name="textfield">
  </p>
  <p>Gender:
    <input type="radio" name="gender" value="m">Male
    <input type="radio" name="gender" value="f">Female</p>
  </form>
</body>
</html>
```

Who are you?

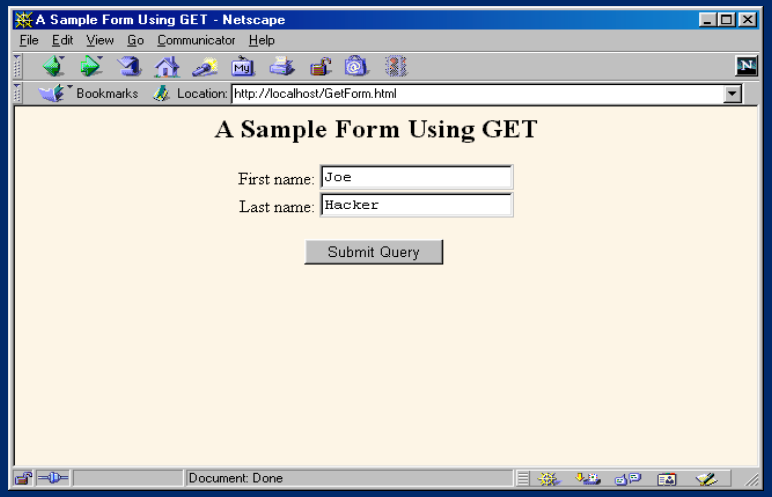
Name:

Gender: ○ Male ○ Female

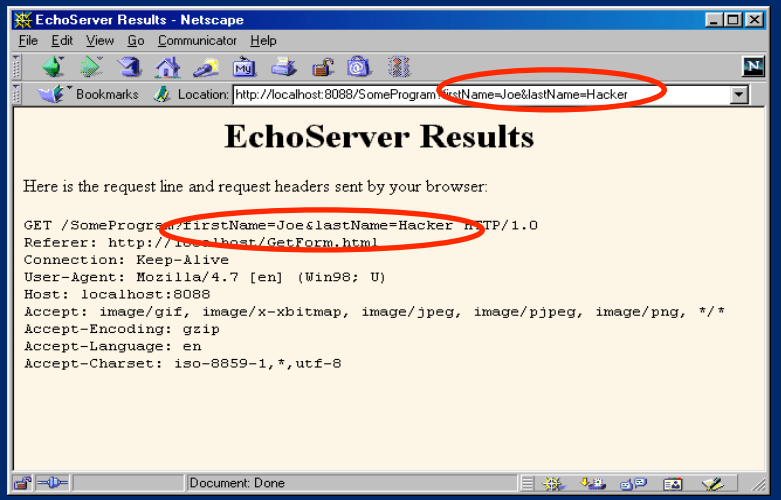
Example 2: Sending Data with GET

```
<BODY BGCOLOR="#FDF5E6">
<H2 ALIGN="CENTER">A Sample Form Using GET</H2>

<FORM ACTION="http://localhost:8088/SomeProgram">
  <CENTER>
  First name:
  <INPUT TYPE="TEXT" NAME="firstName" VALUE="Joe"><BR>
  Last name:
  <INPUT TYPE="TEXT" NAME="lastName" VALUE="Hacker"><P>
  <INPUT TYPE="SUBMIT">
  </CENTER>
</FORM>

</BODY></HTML>
```

Initial Result



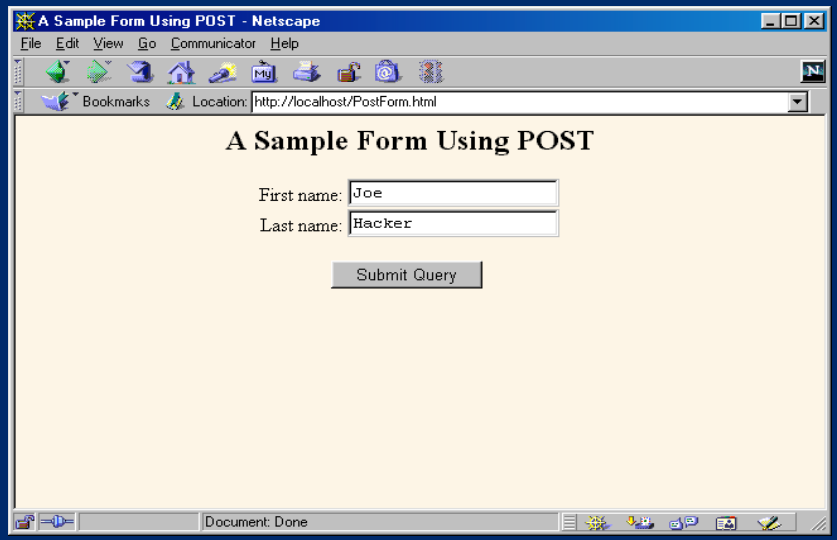
Submission Result



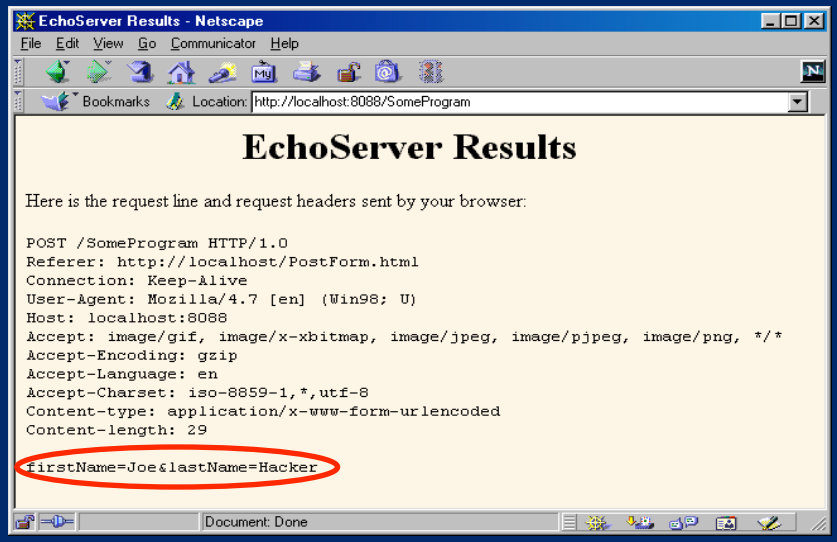
Example 3: Sending Data with POST

```
…
<BODY BGCOLOR="#FDF5E6">
<H2 ALIGN="CENTER">A Sample Form Using POST</H2>

<FORM ACTION="http://localhost:8088/SomeProgram"
      METHOD="POST">
  <CENTER>
  First name:
  <INPUT TYPE="TEXT" NAME="firstName" VALUE="Joe"><BR>
  Last name:
  <INPUT TYPE="TEXT" NAME="lastName" VALUE="Hacker"><P>
  <INPUT TYPE="SUBMIT">
  </CENTER>
</FORM>

</BODY></HTML>
```

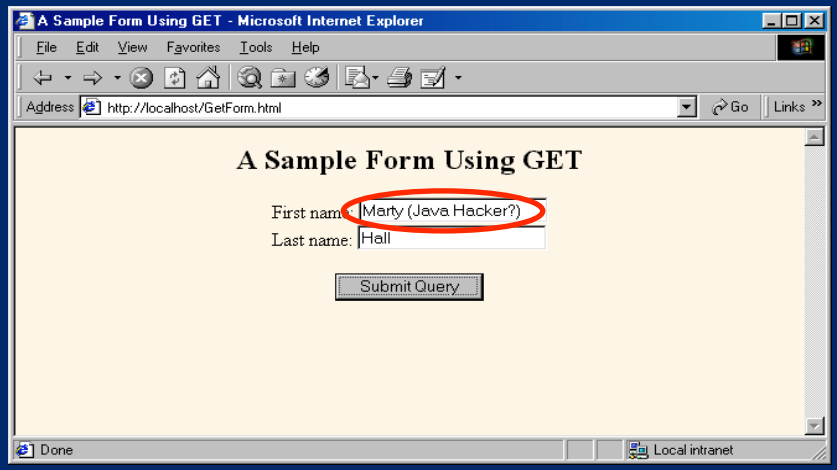
Initial Result



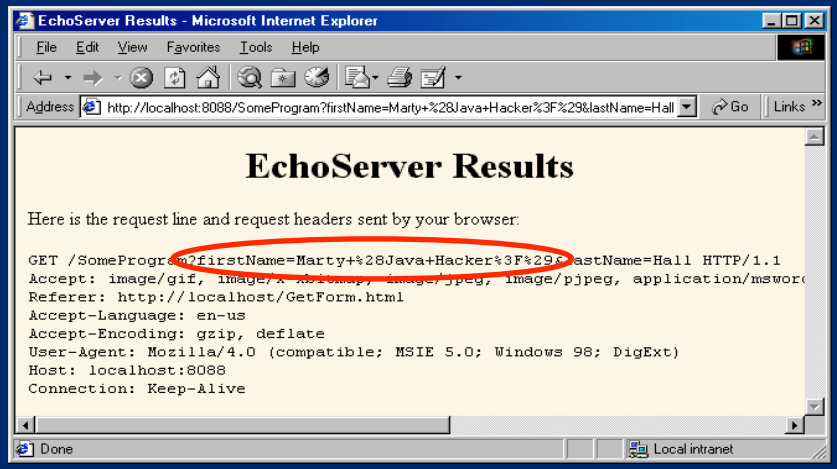
Submission Result



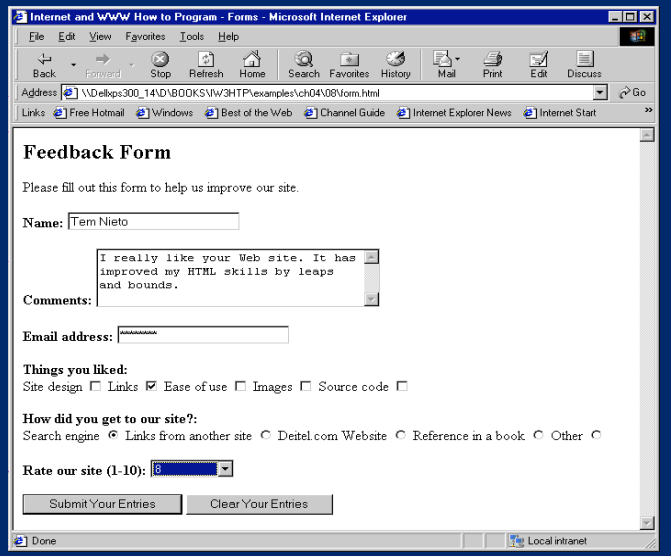
URL Encoding: Original Form



URL Encoding: Result



HTML Form - Example



Final Comments

- HTML and CSS provide lots of neat features,
  - Don't add features that distract from the content of the page
    - use color & fonts sparingly and be careful how elements fit together
      - e.g, no purple text on a pink background, no weird fonts
    - use images only where appropriate
      - e.g., bright background images can make text hard to read
      - e.g., the use of clickable images instead of buttons or links can slow access
    - don't rely on window or font size for layout
      - e.g., font size may be adjusted by viewer, window constrained
    - don't be annoying
      - e.g., no pop-up windows, excessive advertising, silly music
    - break large document into smaller or provide a menu
    - stick to standard features and test several browsers if possible
    - utilize style sheets to make changes easy & ensure consistency

just because you can add a feature doesn't mean you should!