

WWW Technologies

Eleni Stroulia

HTML, CSS, Javascript

1

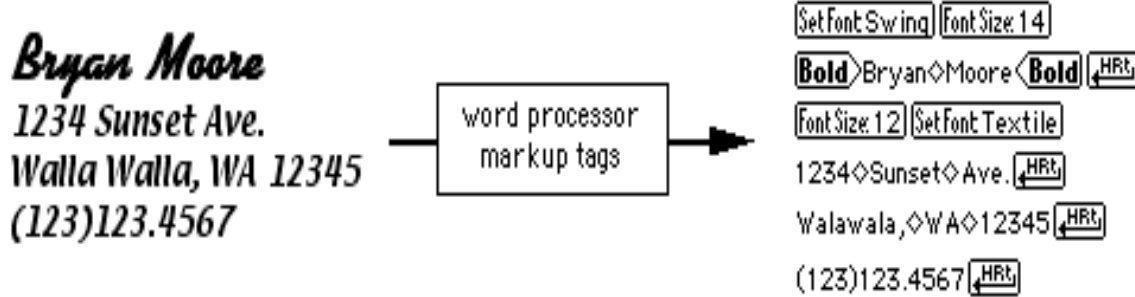
Markup On The Front End

- In the old days, authors scribbled markup instructions on paper in the margins of hand-written works.
- With the advent of electronic documents, the markup instructions are included in the binary files.
- Word processing files are binary blobs. The markup languages are proprietary. Only the programmers know how the markup instructions are encoded as bytes.

HTML, CSS, Javascript

2

How Word Processors work



- Markup is embedded within the text.
 - RTF
 - Tex/Latex
 - Postscript

<http://matwww.ee.tut.fi/hmintro/chap3.html>

HTML, CSS, Javascript

3

Procedural vs. Declarative markup

- Browsers render HTML documents by interpreting the markup commands interspersed in the document text
 - They produce high quality results because they are robust to erroneous markups
- However HTML markup is fundamentally procedural
 - The logical structure of a document is NOT expressed in HTML
 - Extracting data from HTML documents is extremely brittle process
 - Changes in the style require revising all markup commands
- XML is a declarative markup language
 - It declares the logical structure of a document
 - CSS or XSL can be used to specify the style to be used for rendering

HTML, CSS, Javascript

4

Data Formats Taxonomies

- Binary v. Textual
 - Binary: more compact; can be consumed more rapidly by software with little or no conversion.
 - Textual: more portable and interoperable; can be directly read and understood by human beings.
- Final-form v. Reusable
 - Final-form data formats are not designed to allow modification or uses other than that intended by their designers; e.g. PDF, legal transaction formats
 - XML-based data formats are more reusable and repurposable than the alternatives
- Composable v. Standalone
 - PDF; it is not typically embedded in representations encoded in other formats.
 - SOAP, which is designed explicitly to contain a "payload" in some non-SOAP vocabulary

HTML, CSS, Javascript

5

HTML Markup

- The Web was created to be text based (ASCII,Unicode 8,...) so the markup language used to format Web pages is all text.
- Here is how the address on the previous slide would be formatted in HTML, the markup language used to format Web pages.

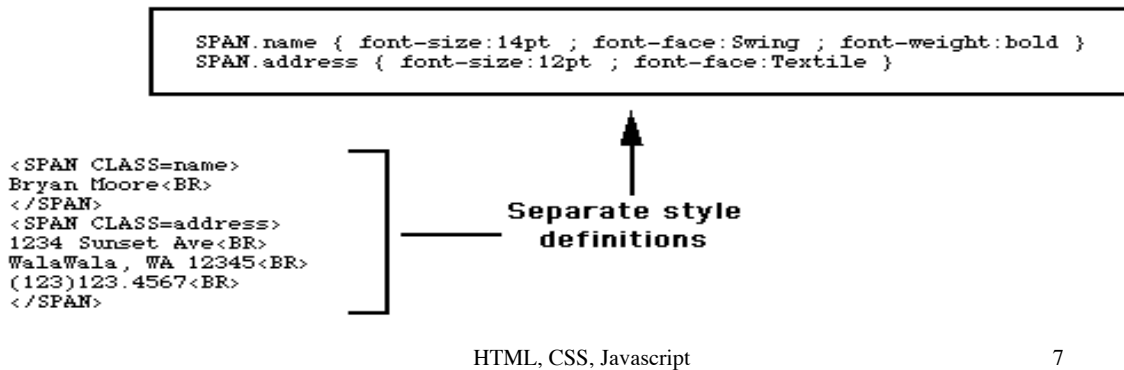
```
<FONT SIZE=14 FACE=Swing>
<B>Bryan Moore</B><BR>
</FONT>
<FONT SIZE=12 FACE=Textile>
1234 Sunset Ave<BR>
WalaWala, WA 12345<BR>
(123)123.4567<BR>
</FONT>
```

HTML, CSS, Javascript

6

HTML and CSS

- The purpose of CSS (Cascading Style Sheets) is to help separate style rules from the data. Using the CSS formatting below, one change to the style rules would change 100 instances of formatting in the document.



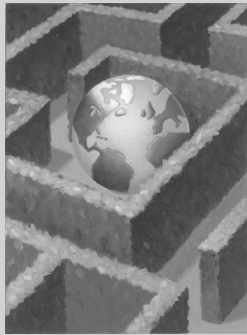
7

XML to the rescue

- XML (eXtensible Markup Language) features a total separation of data content and style.
- The address is shown below, formatted using an invented XML vocabulary.

```
<address id="101">
  <name>Bryan Moore</name>
  <street>1234 Sunset Ave</street>
  <city>WalaWala</city>
  <state>WA</state>
  <zip>12345</zip>
  <phone>(123)123.4567</phone>
</address>
```

- XML is meant to describe what the data is, not what it should look like when rendered by software. XML is covered in detail in Chapters 16 and 17.



core
WEB
programming

HyperText Markup Language (HTML)

Designing Documents for the
World Wide Web

Training Courses: Java, JSP, Servlets, Struts, & JSF:
<http://courses.coreservlets.com>

© 2001-2005 Marty Hall, Larry Brown <http://www.corewebprogramming.com>

9

Agenda

- Introduction to HTML
- Creating and publishing a Web page
- Validating a document
- Main HTML elements
- Block-level HTML elements
- Text-level HTML element
- Creating hypertext links
- Adding images to documents
- Building tables

The World Wide Web

- **Definitions**

- The World Wide Web
 - The set of computers on the Internet that support HTTP
 - Not a separate network.
- HTTP
 - The HyperText Transfer Protocol.
 - The language used by a WWW client (e.g. Netscape, Internet Explorer) to request documents from a WWW server (i.e. the program running at Web sites like amazon.com or yahoo.com)
- HTML
 - The HyperText Markup Language
 - The language used to design web pages

HyperText Markup Language

- **Text Mixed with Markup Tags**

- Tags Enclosed in Angle Brackets
(`<H1>Introduction</H1>`)

- **What Does Markup Describe?**


- Appearance
- Layout
- Content (Can't Enforce an Exact Look)

- **Changes in HTML 3.2 to HTML 4.0**

- Standardization of frames
- Deprecation of formatting elements (vs. style sheets)
- Improved cell alignment and grouping in tables
- Mouse and keyboard events for nearly all elements
- Internationalization features

HTML Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Home Page for Marty Hall</TITLE>
...
<BODY BGCOLOR="#FDF5E6" TEXT="#000000" LINK="#551A8B" ALINK="#FF0000">
<CENTER>
<TABLE BORDER=3 BGCOLOR="#3366FF">
  <TR><TD><STRONG CLASS="title">Home Page
</TABLE>
<P>
<TABLE>
  <TR><TD><STRONG><B>Marty Hall</B><BR>
    President<BR>
    <A HREF="http://www.coreservlets.com">www.coreservlets.com</A><BR>
    coreservlets.com, Inc.<BR>
    6 Meadowsweet Ct., Suite B1<BR>
    Reisterstown, MD 21136-6020<BR>
    <I>email:</I>
    <A HREF="mailto:hall@coreservlets.com">hall@coreservlets.com</A><BR>
    <I>Phone:</I> (410) 429-5535<BR>
    <I>Fax:</I> (410) 429-4931</STRONG></TD>
    <TD><IMG SRC="images/Marty-JHU-Head-Small.jpg"
      ALT="Marty" WIDTH="225" HEIGHT="263" HSPACE="5"></TD></TR>
</TABLE>...
```



13 HTML, CSS, Javascript www.corewebprogramming.com

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/~hall/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

Creating and Publishing a Web Page, cont.

4. Validate the Document

- Check the syntax using a formal HTML validator
 - <http://www.htmlhelp.com/tools/validator/>
 - <http://validator.w3.org/>
- The version of HTML against which the document is validated is based on the DOCTYPE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```
- The WWW Consortium recently added advice that Web pages include information on the character set, even though ASCII or Latin-1 is the default. The validator at <http://validator.w3.org/> gives warnings if you omit this. You can ignore such warnings if you wish.

15

HTML, CSS, Javascript

www.corewebprogramming.com

HTML Document Template

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Title</TITLE>
</HEAD>

<BODY>
<H1>Main Heading</H1>

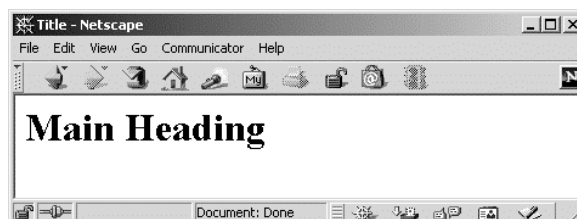
<!-- Rest of page goes here -->

</BODY>
</HTML>
```

← Goes on browser's title bar
May not appear in printouts

← Main heading. Often used as title
Appears in printouts

← HTML comment
Replace with body of
WWW page



16

HTML, CSS, Javascript

www.corewebprogramming.com

Main HTML Elements

1. DOCTYPE

2. HTML

3. HEAD

- TITLE element required
- Optional elements:
 - BASE
 - META
 - BGSOUND
 - SCRIPT, NOSCRIPT
 - STYLE
 - LINK

Main HTML Elements (Continued)

4. BODY Element

- <BODY BGCOLOR="YELLOW">
- HTML Attributes and Attribute Values
 - BACKGROUND
 - BGCOLOR
 - TEXT
 - LINK, VLINK, ALINK
 - OnLoad, OnUnload, OnFocus, OnBlur

5. Elements inside BODY element

- <BODY>
Remaining HTML elements
</BODY>

META Element

- **Records document information, forwards and refreshes pages**
 - NAME="author"
 - NAME="keywords"
 - NAME="description"
 - HTTP-EQUIV="refresh"

META Element, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>News Headlines</TITLE>
  <META HTTP-EQUIV="REFRESH"
    CONTENT="3600">
</HEAD>

<BODY>
<H1 ALIGN="CENTER">News Headlines</H1>

<H2>National News</H2>
Blah, blah, blah.

<H2>International News</H2>
Yadda, yadda, yadda.

</BODY>
</HTML>
```

Block-Level Elements

- **Headings**
 - H1 ... H6
 - ALIGN
- **Basic Text Sections**
 - P
 - ALIGN
 - PRE
 - WIDTH
 - ADDRESS
 - BLOCKQUOTE

Block-Level Elements, cont.

- **Lists**
 - OL
 - LI
 - UL
 - LI
 - DL
 - DT
 - DD
- **Tables and Forms (Postponed)**
- **Misc.**
 - HR
 - DIV
 - CENTER
 - MULTICOL (Netscape only)

Headings

- **Heading Types**

- `<H1 ...> ... </H1>`
- `<H2 ...> ... </H2>`
- `<H3 ...> ... </H3>`
- `<H4 ...> ... </H4>`
- `<H5 ...> ... </H5>`
- `<H6 ...> ... </H6>`

- **Attributes: ALIGN**

- Values: LEFT (default), RIGHT, CENTER

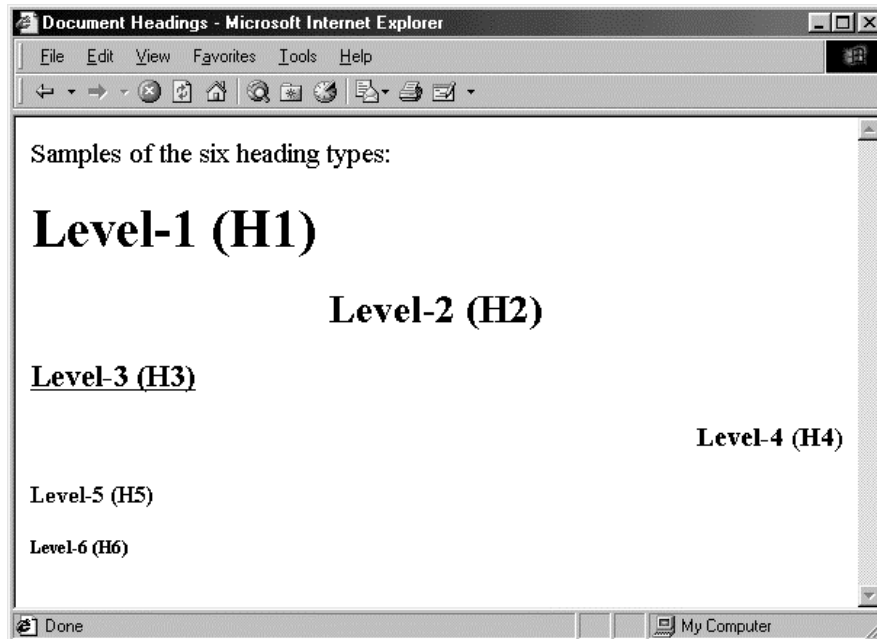
- **Nesting tags**

- Headings and other block-level elements can contain text-level elements, but *not* vice versa

Headings, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Document Headings</TITLE>
</HEAD>
<BODY>
  Samples of the six heading types:
  <H1>Level-1 (H1)</H1>
  <H2 ALIGN="CENTER">Level-2 (H2)</H2>
  <H3><U>Level-3 (H3)</U></H3>
  <H4 ALIGN="RIGHT">Level-4 (H4)</H4>
  <H5>Level-5 (H5)</H5>
  <H6>Level-6 (H6)</H6>
</BODY>
</HTML>
```


Headings, Result



25

HTML, CSS, Javascript

www.corewebprogramming.com

P – The Basic Paragraph

- **Attributes: ALIGN**

- LEFT (default), RIGHT, CENTER. Same as headings.
- Whitespace ignored (use
 for line break)
 - Consecutive <P>'s do not yield multiple blank lines
- End Tag is Optional:

```
<BODY>
```

```
<P>
```

```
Paragraph 1
```

```
</P>
```

```
<P>
```

```
Paragraph 2
```

```
</P>
```

```
<P>
```

```
Paragraph 3
```

```
</P>
```

```
</BODY>
```

Fully-Specified

```
<BODY>
```

```
Paragraph 1
```

```
<P>
```

```
Paragraph 2
```

```
<P>
```

```
Paragraph 3
```

```
</BODY>
```

Equivalent with Implied Tags

26

HTML, CSS, Javascript

www.corewebprogramming.com

Preformatted Paragraphs

- **The PRE Element**

- `<PRE> ... </PRE>`

- **Attributes: WIDTH**

- Expected width in characters. Not widely supported.

- **Problem: Special Characters**

`<PRE>`

```
if (a<b) {  
    doThis();  
} else {  
    doThat();  
}  
</PRE>
```

Desired Character	HTML Required
<	<
>	>
&	&
"	"
Non-breaking space	

27

HTML, CSS, Javascript

www.corewebprogramming.com

OL: Ordered (Numbered) Lists

- **OL Element**

- ``

- `...`

- `...`

- ``

- Attributes: TYPE, START, COMPACT

- **List entries: LI**

- `<LI ...> ... ` (End Tag Optional)

- Attributes: (When inside OL) VALUE, TYPE

A sample list:

```
<OL>  
    <LI>List Item One  
    <LI>List Item Two  
    <LI>List Item Three  
</OL>
```

A sample list:

```
1. List Item One  
2. List Item Two  
3. List Item Three
```

28

HTML, CSS, Javascript

www.corewebprogramming.com

Nested Ordered Lists

```
<OL TYPE="I">
<LI>Headings
  <LI>Basic Text Sections
    <LI>Lists
      <OL TYPE="A">
        <LI>Ordered
          <OL TYPE="1">
            <LI>The OL tag
              <OL TYPE="a">
                <LI>TYPE
                <LI>START
                <LI>COMPACT
              </OL>
            <LI>The LI tag
          </OL>
        <LI>Unordered
          <OL TYPE="1">
            <LI>The UL tag
            <LI>The LI tag
          </OL>
        <LI>Definition
          <OL TYPE="1">
            <LI>The DL tag
            <LI>The DT tag
            <LI>The DD tag
          </OL>
        <LI>Miscellaneous
      </OL>
    </LI>
  </LI>
</OL>
```

```
I. Headings
II. Basic Text Sections
III. Lists
  A. Ordered
    1. The OL tag
      a. TYPE
      b. START
      c. COMPACT
    2. The LI tag
  B. Unordered
    1. The UL tag
    2. The LI tag
  C. Definition
    1. The DL tag
    2. The DT tag
    3. The DD tag
IV. Miscellaneous
```

29

HTML, CSS, Javascript

www.corewebprogramming.com

UL: Unordered (Bulleted) Lists

- **UL Element**

```
- <UL>
  <LI>...
  <LI>...

  </UL>
```

- **Attributes: TYPE, COMPACT**

- TYPE is DISC, CIRCLE, or SQUARE

- **List entries: LI (TYPE)**

- TYPE is DISC, CIRCLE, or SQUARE

A sample list:

```
<UL>
  <LI>List Item One
  <LI>List Item Two
  <LI>List Item
```

A sample list:

```
• List Item One
• List Item Two
• List Item Three
```

30

HTML, CSS, Javascript

www.corewebprogramming.com

UL: Custom Bullets

```
<UL TYPE="DISC">
  <LI>The UL tag
    <UL TYPE="CIRCLE">
      <LI>TYPE
        <UL TYPE="SQUARE">
          <LI>DISC
          <LI>CIRCLE
          <LI>SQUARE
        </UL>
      <LI>COMPACT
    </UL>
  <LI>The LI tag
    <UL TYPE="CIRCLE">
      <LI>TYPE
        <UL TYPE="SQUARE">
          <LI>DISC
          <LI>CIRCLE
          <LI>SQUARE
        </UL>
      <LI>VALUE
    </UL>
  </UL>
```

Unordered Lists

- The UL tag
 - TYPE
 - DISC
 - CIRCLE
 - SQUARE
 - COMPACT
- The LI tag
 - TYPE
 - DISC
 - CIRCLE
 - SQUARE
 - VALUE

Text-Level Elements

• Physical Character Styles

- B, I, TT, U, SUB, SUP, SMALL, BIG, STRIKE, S, BLINK
- FONT
 - SIZE
 - COLOR
 - FACE
- BASEFONT
- SIZE

• Logical Character Styles

- EM, STRONG, CODE, SAMP, KBD, DFN, VAR, CITE

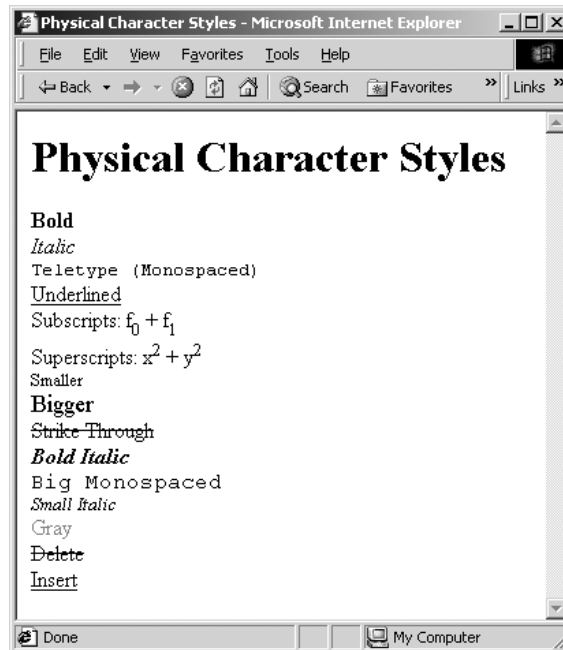
Text-Level Elements (Continued)

- **Hypertext Links**
 - A
 - HREF, NAME, TARGET, ...
- **Images**
 - IMG
 - SRC (required), ALT, ALIGN, WIDTH, HEIGHT, HSPACE, VSPACE, BORDER, USEMAP, ISMAP
- **Misc. Text-Level Elements**
 - BR (Explicit line break)
 - AREA (Client-side image maps)
 - APPLET (Java),
 - ...

Physical Character Styles, Example

```
...
<H1>Physical Character Styles</H1>
<B>Bold</B><BR>
<I>Italic</I><BR>
<TT>Teletype (Monospaced)</TT><BR>
<U>Underlined</U><BR>
Subscripts: f<SUB>0</SUB> + f<SUB>1</SUB><BR>
Superscripts: x<SUP>2</SUP> + y<SUP>2</SUP><BR>
<SMALL>Smaller</SMALL><BR>
<BIG>Bigger</BIG><BR>
<STRIKE>Strike Through</STRIKE><BR>
<B><I>Bold Italic</I></B><BR>
<BIG><TT>Big Monospaced</TT></BIG><BR>
<SMALL><I>Small Italic</I></SMALL><BR>
<FONT COLOR="GRAY">Gray</FONT><BR>
<DEL>Delete</DEL><BR>
<INS>Insert</INS><BR>
...
```

Physical Character Styles, Result



35

HTML, CSS, Javascript

www.corewebprogramming.com

Logical Character Styles, Example

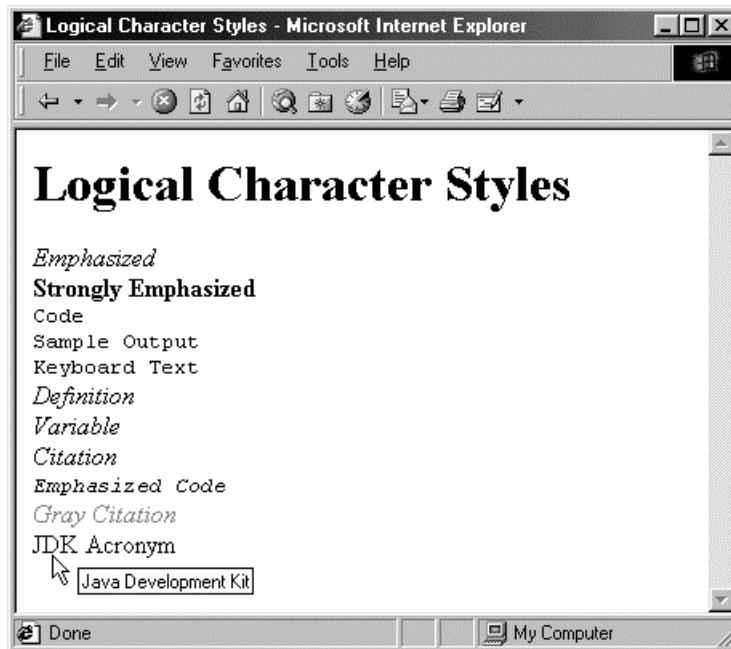
```
...
<H1>Logical Character Styles</H1>
<EM>Emphasized</EM><BR>
<STRONG>Strongly Emphasized</STRONG><BR>
<CODE>Code</CODE><BR>
<SAMP>Sample Output</SAMP><BR>
<KBD>Keyboard Text</KBD><BR>
<DFN>Definition</DFN><BR>
<VAR>Variable</VAR><BR>
<CITE>Citation</CITE><BR>
<EM><CODE>Emphasized Code</CODE></EM><BR>
<FONT COLOR="GRAY"><CITE>Gray Citation</CITE></FONT><BR>
<ACRONYM TITLE="Java Development Kit">JDK Acronym</ACRONYM>
...
```

36

HTML, CSS, Javascript

www.corewebprogramming.com

Logical Character Styles, Result



37

HTML, CSS, Javascript

www.corewebprogramming.com

Hypertext Links

- **Links can contain images and other text-level elements (i.e., `<A HREF...> ... `)**
- **Link to Absolute URL**
 - Use a complete URL beginning with `http://`
Java is discussed in
``
Chapter 2``.
- **Link to Relative URL**
 - Use a filename or relative path to filename
 - Interpreted wrt location of current file
 - Java is discussed in
``Chapter 2``.

38

HTML, CSS, Javascript

www.corewebprogramming.com

Hypertext Links (Continued)

- **Link to Section**

- Use a section name (see below) preceded by #
Images are discussed in
`Section 2.`

- **Link to Section in URL**

- Use absolute or relative URL, then #, then section name
Images are discussed in
`
Sec. 2 of Chap. 1.`

- **Naming a Section**

- Use `` and do not include the pound sign
`<H2>Images</H2>`

39

HTML, CSS, Javascript

www.corewebprogramming.com

IMG: Embedding Images

- **Example**

`<IMG SRC="SomeFile.gif" ALT="My Dog"
WIDTH=400 HEIGHT=300>`

- **Attributes:**

- SRC (required)
- ALT (technically required)
- ALIGN (see `<BR CLEAR="ALL">`)
- WIDTH, HEIGHT
- HSPACE, VSPACE
- BORDER
- USEMAP, ISMAP

40

HTML, CSS, Javascript

www.corewebprogramming.com

Image Alignment, Example

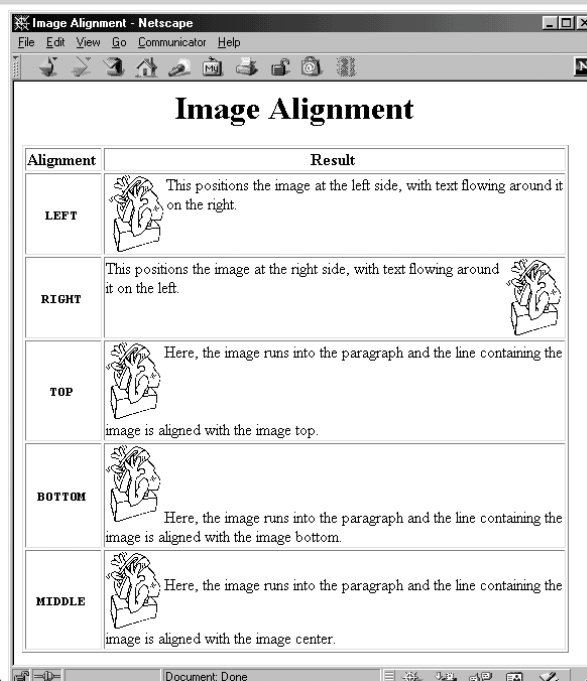
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>Image Alignment</TITLE></HEAD>
<BODY>
<H1 ALIGN="CENTER">Image Alignment</H1>
<TABLE BORDER=1>
  <TR><TH>Alignment
    <TH>Result
  <TR><TH><CODE>LEFT</CODE>
    <TD><IMG SRC="rude-pc.gif" ALIGN="LEFT"
      ALT="Rude PC" WIDTH=54 HEIGHT=77>
      This positions the image at the left side,
      with text flowing around it on the right.
  <TR><TH><CODE>RIGHT</CODE>
    <TD><IMG SRC="rude-pc.gif" ALIGN="RIGHT"
      ALT="Rude PC" WIDTH=54 HEIGHT=77>
      This positions the image at the right side,
      with text flowing around it on the left.
  ...
</TABLE>
</BODY>
</HTML>
```

41

HTML, CSS, Javascript

www.corewebprogramming.com

Image Alignment, Result



42

HTML, CSS, Javascript

www.corewebprogramming.com

Tables

- **Template**

```
<TABLE BORDER=1>
  <CAPTION>Table Caption</CAPTION>
  <TR><TH>Heading1</TH>      <TH>Heading2</TH></TR>
  <TR><TD>Row1 Col1 Data</TD><TD>Row1 Col2 Data</TD></TR>
  <TR><TD>Row2 Col1 Data</TD><TD>Row2 Col2 Data</TD></TR>
  <TR><TD>Row3 Col1 Data</TD><TD>Row3 Col2 Data</TD></TR>
</TABLE>
```

Heading1	Heading2
Row1 Col1 Data	Row1 Col2 Data
Row2 Col1 Data	Row2 Col2 Data
Row3 Col1 Data	Row3 Col2 Data

TABLE Element Attributes

- **ALIGN**

- The ALIGN attribute gives the horizontal alignment of the table as a whole
- Legal values are LEFT, RIGHT, and CENTER, with LEFT being the default

- **BORDER**

- This specifies the width in pixels of the border around the table
- This is in addition to the border around each cell (the CELSPACING).
- The default is zero, which also results in the visible 3D divider between cells being turned off

- **CELLSPACING**

- This gives the space in pixels between adjacent cells. Drawn as a 3D line if BORDER is nonzero, otherwise empty space in the background color is used
- The default is usually about 3

TABLE Element Attributes (Continued)

- **CELLPADDING**
 - CELLPADDING determines the empty space, in pixels, between the cell's border and the table element
 - The default is usually about 1
- **WIDTH**
 - This specifies the width of the table, either in pixels (<TABLE WIDTH=250>) or as a percentage of the current browser window width (<TABLE WIDTH="75%">)
- **BGCOLOR**
 - Specify the background color of the table TABLE (also legal for TR, TD, and TH)
- **BORDERCOLOR, BORDERCOLORDARK,**
- **BORDERCOLORLIGHT**
 - Non standard attributes supported by IE to specify the colors to use for the borders

TABLE Element Attributes (Continued)

- **BACKGROUND**
 - This nonstandard attribute supported by IE gives an image file that will be tiled as the background of the table
 - You might want to use style sheets instead.
- **RULES**
 - HTML 4.0 attribute that specifies which inner dividing lines are drawn
 - All are drawn if this attribute is omitted
 - Legal values are NONE, ROWS, COLS, and ALL
- **FRAME**
 - Specifies which outer borders are drawn
 - All four are drawn if this attribute is omitted
 - Legal values are BORDER or BOX (all), VOID (none), ABOVE (top), BELOW (bottom), HSIDES (top and bottom, despite the somewhat confusing name), VSIDES (left and right), LHS (left), and RHS (right)

Table CAPTION

- **Attribute**
 - ALIGN (Values: TOP, BOTTOM)
- **Usage**
 - An enclosing borderless table may give more flexibility than the built-in CAPTION.

Heading1	Heading2
Row1 Col1 Data	Row1 Col2 Data
Row2 Col1 Data	Row2 Col2 Data
Row3 Col1 Data	Row3 Col2 Data

TR: Table Row

- **TR is used to define each row in the table**
- **Each row will then contain TH and/or TD entries**
- **ALIGN**
 - ALIGN (legal values LEFT, RIGHT, or CENTER) is used to set the default horizontal alignment for table cells
- **VALIGN**
 - VALIGN (legal values TOP, MIDDLE, or BOTTOM) is used to set the default vertical alignment for table cells
- **BGCOLOR**
 - Sets the color for the table row, overriding any values set for the table as a whole via the BGCOLOR attribute of TABLE
- **BORDERCOLOR, BORDERCOLORDARK,**
- **BORDERCOLORLIGHT**
 - Supported only by Internet Explorer, these specify the colors to use for the row borders

Table Cells: TH and TD

- **COLSPAN**

- COLSPAN defines a heading or cell data entry that spans multiple columns

```
<TABLE BORDER=1>
<TR><TH COLSPAN=2>Col 1&2 Heading
      <TH>Col3 Heading
<TR><TD>Col1 Data
      <TD>Col2 Data
      <TD>Col3 Data
</TABLE>
```

Col 1&2 Heading		Col3 Heading
Col1 Data	Col2 Data	Col3 Data

Table Cells: TH and TD (Continued)

- **ROWSPAN**

- ROWSPAN defines a heading or cell data entry that spans multiple rows; similar to COLSPAN

- **ALIGN**

- LEFT, RIGHT, CENTER, JUSTIFY and CHAR.
- E.g., the following aligns entries on a decimal point

```
<TD ALIGN="CHAR" CHAR=".">
```

- **VALIGN**

- TOP, BOTTOM, MIDDLE

- **WIDTH, HEIGHT**

- Values in pixels only (no percentages officially allowed)

- **NOWRAP**

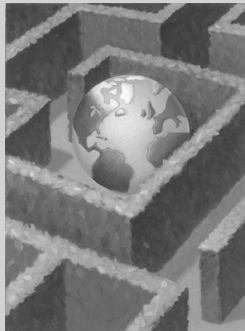
- Use with caution

- **BGCOLOR, BACKGROUND**

- Same as for TABLE and TR

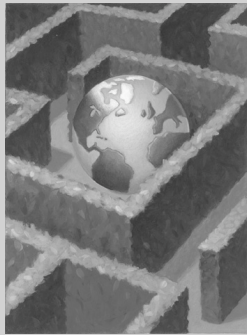
Summary

- A DOCTYPE is required to validate the document
- HTML document should have an enclosing HTML element, a HEAD (TITLE is required) and a BODY
- Documents are composed of block-level and text-level elements
 - Text-level elements must be inside block-level elements, not vice versa
- Hypertext links, ****, can be absolute or relative
 - A link to a named section is denoted by *#section*
- Tables are composed of main table element, **<TABLE>**; rows, **<TR>**; table headers, **<TH>**; and table data, **<TD>**
 - Use BGCOLOR to give background colors to tables, rows, or cells
 - Use ROWSPAN or COLSPAN to join cells



core
WEB
programming

Questions?



core
WEB
programming

HTML Frames

Training Courses: Java, JSP, Servlets, Struts, & JSF:
<http://courses.coreservlets.com>

© 2001-2005 Marty Hall, Larry Brown <http://www.corewebprogramming.com>

53

Agenda

- Advantages and disadvantages of frames
- FRAME template
- Defining rows and cols in a FRAMESET
- Common FRAME and FRAMESET attributes
- Nested frames
- Targeting a document to a named FRAME cell

54

Frame Advantages

- **Certain parts of the interface (e.g., a TOC) are always on the screen**
- **Can avoid retyping common sections of multiple Web pages**
- **Consistent use across a large site sometimes simplifies user navigation**
- **A convenient way to mix text-oriented HTML with Java applets**
- **Image maps are more convenient if the map image remains on screen and only the results section changes**

Frame Disadvantages

- **The meaning of the “Back” and “Forward” buttons can be confusing**
- **Poorly designed frames can get the user lost**
- **Hard to find real URL of a page you want**
 - Printing problems!
- **Hard to bookmark "configuration"**
- **Some very old browsers do not support frames**
- **Security**
 - Hackers can insert frame cells into your pages in some circumstances, perhaps stealing information intended for your site

Frame Template

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD><TITLE>Document Title</TITLE></HEAD>

<FRAMESET ...>
  <!-- FRAME and Nested FRAMESET Entries -->
  <NOFRAMES>
    <BODY>
      <!-- Stuff for non-Frames browsers -->
    </BODY>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

FRAMESET Attributes

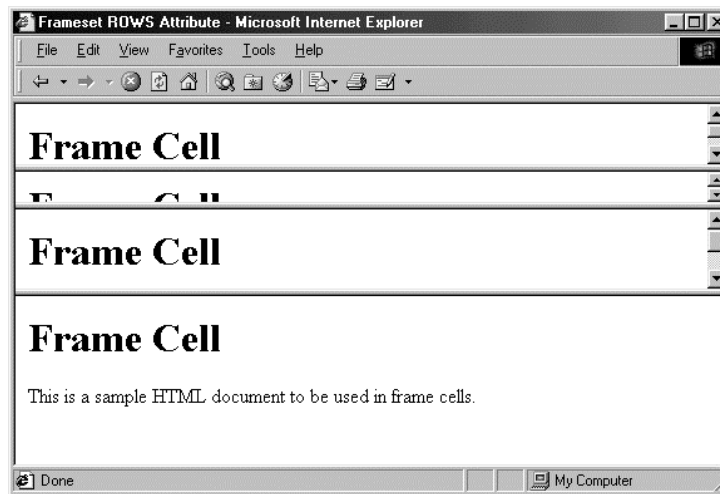
- **COLS, ROWS**

- A comma-separated list of pixel values, percentages, and weighted remainders
- FRAMESET entries should *always* specify at least two rows or columns. Netscape problems if not!
- Examples

```
<FRAMESET ROWS="50,10%,*,2*">
  ...
</FRAMESET>
```

```
<FRAMESET COLS="25%,*,*">
  ...
</FRAMESET>
```

FRAMESET ROWS, Example



```
<FRAMESET ROWS="50,10%,*,2*">  
...  
</FRAMESET>
```

59

HTML, CSS, Javascript

www.corewebprogramming.com

FRAMESET Attributes (Continued)

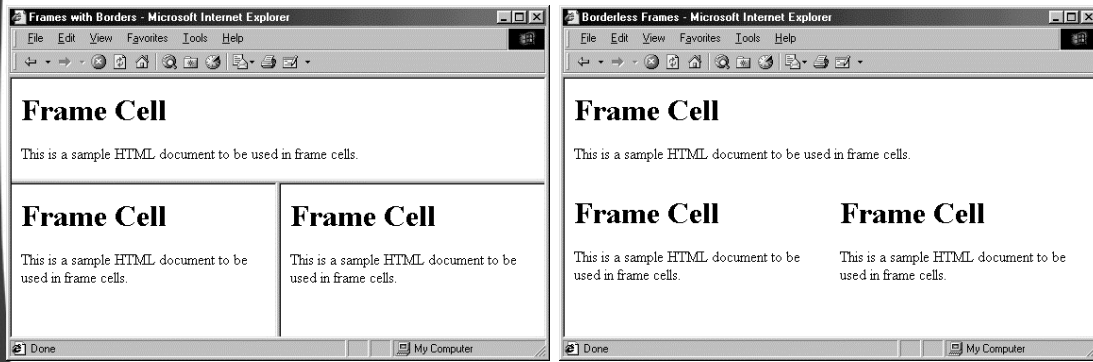
- **FRAMEBORDER**
 - Indicates whether borders will be drawn *between* frame cells
 - YES or 1 specifies borders; NO or 0 specifies no border
 - Can be overridden by FRAMEBORDER settings in individual FRAME entries
 - Often used in conjunction with BORDER=0 and FRAMESPACING=0
- **BORDER (Netscape), FRAMESPACING (IE)**
 - Specify the thickness of the border between cells
 - Apply to outermost FRAMESET only
- **BORDERCOLOR**
 - Sets the color of the border between cell, using either a hex RGB value or color name

60

HTML, CSS, Javascript

www.corewebprogramming.com

Frame Border, Examples



FRAME: Specifying Content of Frame Cells

- **SRC**
 - URL of the document to place in the frame cell
- **NAME**
 - Supplies destination for TARGET attribute of hypertext links
- **FRAMEBORDER, BORDERCOLOR**
- **MARGINWIDTH, MARGINHEIGHT**
 - Specifies the left/right and top/bottom cell margins, respectively
- **SCROLLING**
 - Indicates whether cells should have scrollbars
- **NORESIZE**
 - Disables the ability to resize the frame cells

Frame Example 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD><TITLE>Frame Example 1</TITLE></HEAD>
<FRAMESET ROWS="55%,45%">
  <FRAMESET COLS="*,*,*">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
  </FRAMESET>

  <FRAMESET COLS="*,*">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
  </FRAMESET>

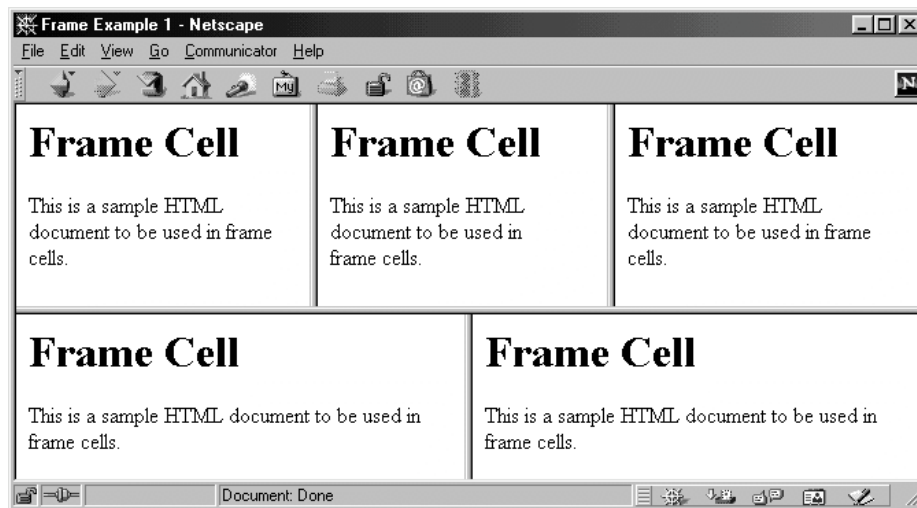
<NOFRAMES>
  <BODY>
    Your browser does not support frames. Please see
    <A HREF="Frame-Cell.html">non-frames version</A>.
  </BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

63

HTML, CSS, Javascript

www.corewebprogramming.com

Frame Example 1, Result



64

HTML, CSS, Javascript

www.corewebprogramming.com

Frame Example 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD><TITLE>Frame Example 2</TITLE></HEAD>

<FRAMESET COLS="55%,45%">
  <FRAMESET ROWS="*,*,*">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
  </FRAMESET>

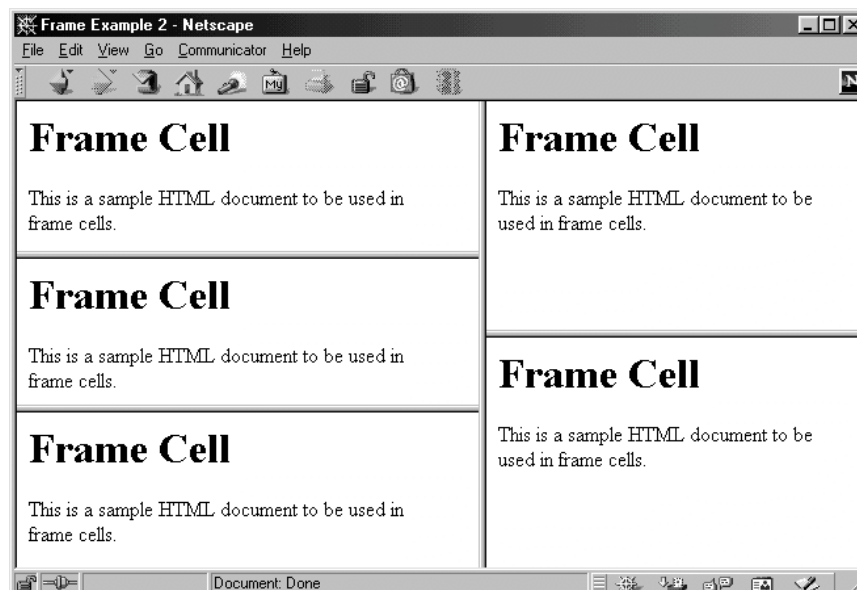
  <FRAMESET ROWS="*,*">
    <FRAME SRC="Frame-Cell.html">
    <FRAME SRC="Frame-Cell.html">
  </FRAMESET>
</FRAMESET>
<NOFRAMES>
  <BODY>
    Your browser does not support frames. Please see
    <A HREF="Frame-Cell.html">nonframes version</A>.
  </BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

65

HTML, CSS, Javascript

www.corewebprogramming.com

Frame Example 2, Result



66

HTML, CSS, Javascript

www.corewebprogramming.com

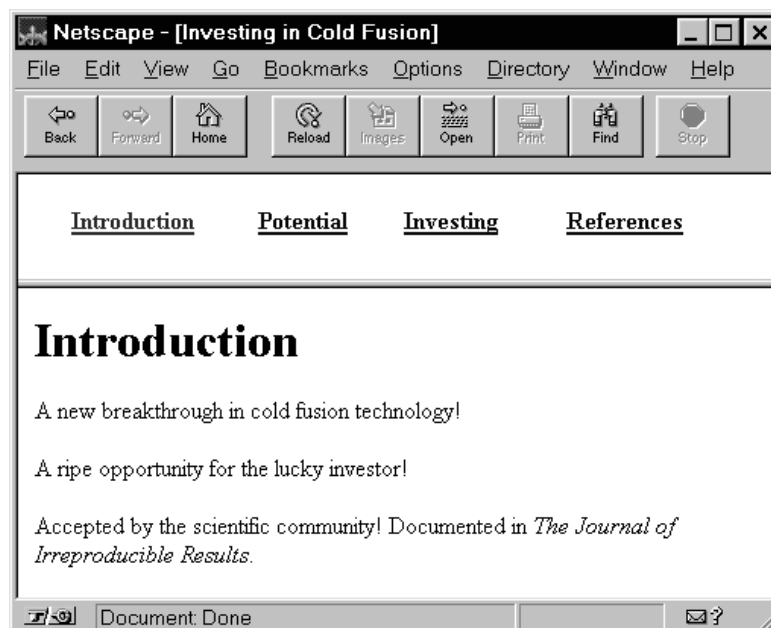
Targeting Frame Cells

- Specify the cell in which to place a page referenced by a hyperlink
- The NAME Attribute of FRAME

```
<FRAME SRC="..." NAME="cellName">
```
- The TARGET Attribute of A HREF

```
<A HREF="..." TARGET="cellName">
```

Targeting Example



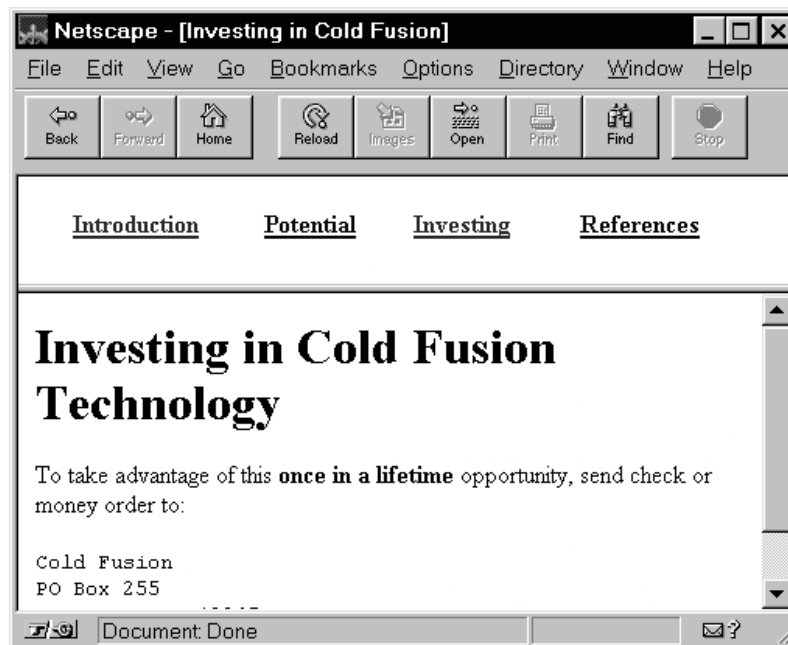
Cold-Fusion.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
  <TITLE>Investing in Cold Fusion</TITLE>
</HEAD>
<FRAMESET ROWS="75,*">
  <FRAME SRC="TOC.html" NAME="TOC">
  <FRAME SRC="Introduction.html" NAME="Main">
<NOFRAMES>
  <BODY>
    This page requires Frames. For a non-Frames version,
    <A HREF="Introduction.html">the introduction</A>.
  </BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

TOC.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Table of Contents</TITLE>
</HEAD>
<BODY>
<TABLE WIDTH="100%">
  <TR><TH><A HREF="Introduction.html" TARGET="Main">
    Introduction</A></TH>
    <TH><A HREF="Potential.html" TARGET="Main">
    Potential</A></TH>
    <TH><A HREF="Investing.html" TARGET="Main">
    Investing</A></TH>
    <TH><A HREF="References.html" TARGET="Main">
    References</A></TH></TR>
</TABLE>
</BODY>
</HTML>
```

Targeting Example, Results



71

HTML, CSS, Javascript

www.corewebprogramming.com

Predefined Frame Names

- **_blank**
 - Load document into a new browser window
- **_top**
 - Causes the linked document to take up the whole browser window
 - Document will not be contained in a frame cell
- **_parent**
 - Places document in the *immediate* FRAMESET parent
 - Same as **_top** if no nested frames
- **_self**
 - Place document in current cell
 - Only necessary to override a BASE entry

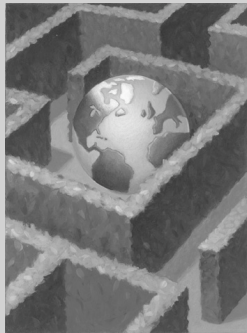
72

HTML, CSS, Javascript

www.corewebprogramming.com

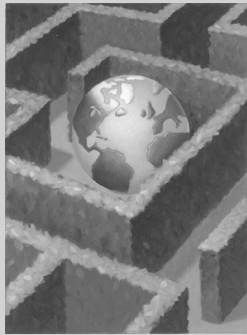
Summary

- **Frames require a Frameset DOCTYPE for validation**
- **A FRAMESET can be divided either into columns or rows**
 - To create both rows *and* columns use nested FRAMESETs
- **By giving a FRAME a name, documents can be targeted to the named frame cell**
 - `<FRAME ... NAME="...">`
 - ``
- **There are four predefined frame names**
 - `_blank`, `_top`, `_parent`, and `_self`



core
WEB
programming

Questions?



core
WEB
programming

Cascading Style Sheets

75

© 2001-2005 Marty Hall, Larry Brown <http://www.corewebprogramming.com>

Agenda

- **Specifying style sheet rules**
- **External and inline style specifications**
- **Creating new HTML elements through style sheet classes**
- **Specifying font and text properties**
- **Controlling foreground and background properties**
- **Netscape LAYERs**
- **Creating layers through style sheets**

76

HTML, CSS, Javascript

www.corewebprogramming.com

Benefits of Cascading Style Sheets

- **Powerful and flexible way to specify the formatting of HTML elements**
 - Can define font, size, background color, background image, margins, etc.
- **Share style sheets across multiple documents or entire Web site**
- **Can specify a class definition for a style, effectively defining new HTML elements**
- **Rules are applied in a hierarchical manner (precedence rules)**

Separating content from look-and-feel style

Cascading Style Sheets

- **CSS, Level 1 (1996)**
 - Concerned with applying simple styles to HTML elements
 - <http://www.w3.org/TR/REC-CSS1>
- **CSS, Level 2 (1998)**
 - Supports media-specific style sheets (visual browsers, aural devices, printers, braille devices)
 - <http://www.w3.org/TR/REC-CSS2>
- **CSS, Level 3 (draft 2001)**
 - Focused on modularization of the CSS specification
 - <http://www.w3.org/TR/css3-roadmap>
- **Note:**
 - CSS1 is supported by Netscape and Internet Explorer 4.x and above
 - See <http://www.webreview.com/style/css1/charts/mastergrid.shtml> for a summary of browser compatibility

Specifying Style Rules

- General form of rule

```
selector { property: value }
```

or

```
selector { property1: value1;  
           property2: value2;  
           ...  
           propertyN: valueN }
```

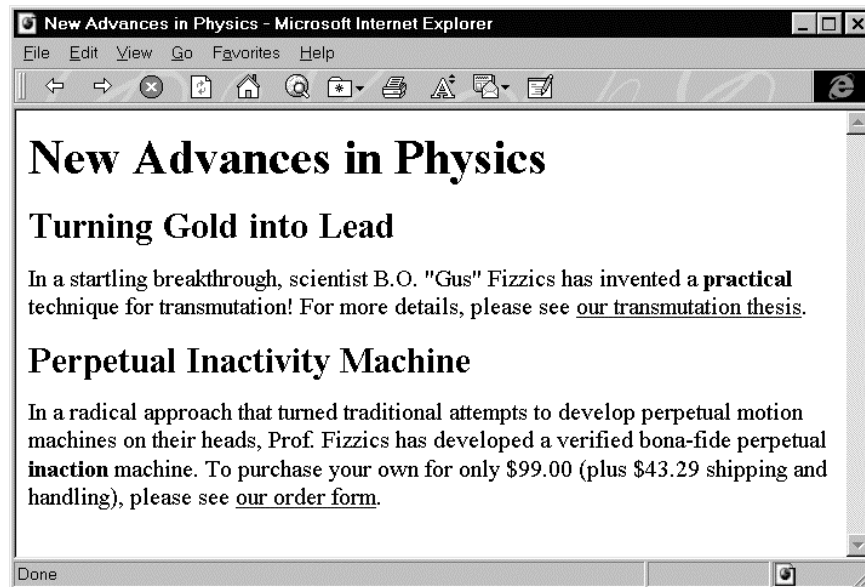
- Example

```
H1 { text-align: center;  
     color: blue }
```

Fizzics1.html, Example (no style sheet)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">  
<HTML>  
<HEAD>  
  <TITLE>New Advances in Physics</TITLE>  
</HEAD>  
<BODY>  
  <H1>New Advances in Physics</H1>  
  
  <H2>Turning Gold into Lead</H2>  
  In a startling breakthrough, scientist B.O. "Gus" Fizzics  
  has invented a <STRONG>practical</STRONG> technique for  
  transmutation! For more details, please see  
  <A HREF="give-us-your-gold.html">our transmutation thesis</A>.  
  ...  
  
</BODY>  
</HTML>
```

Fizzics1.html, Result (no style sheet)



81

HTML, CSS, Javascript

www.corewebprogramming.com

Fizzics2.html, Example (with internal style sheet)

- Style information

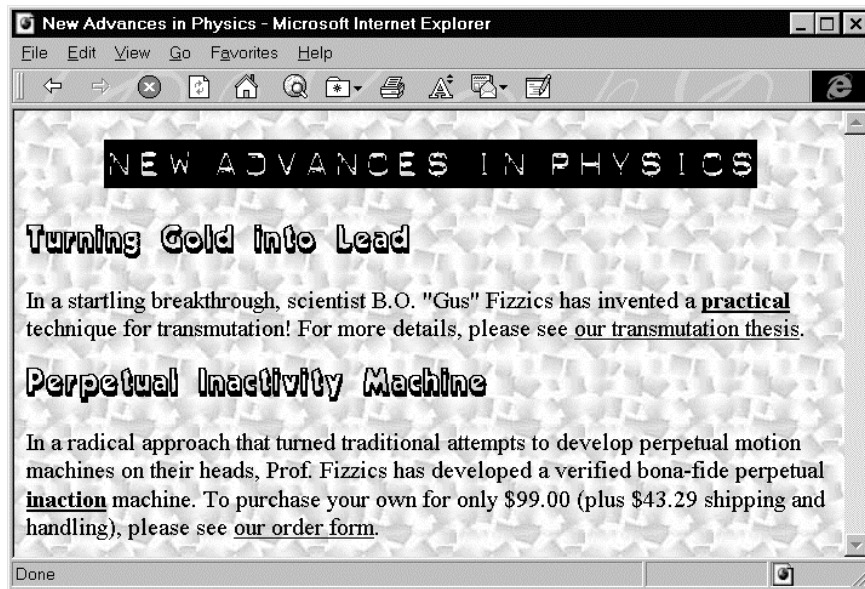
```
<HEAD>
  <TITLE>Document Title</TITLE>
  <STYLE TYPE="text/css">
    <!--
    BODY { background: URL(images/confetti-background.jpg) }
    H1 { text-align: center;
          font-family: Blackout }
    H2 { font-family: MeppDisplayShadow }
    STRONG { text-decoration: underline }
    -->
  </STYLE>
</HEAD>
```

82

HTML, CSS, Javascript

www.corewebprogramming.com

Fizzics2.html, Result (with internal style sheet)



83

HTML, CSS, Javascript

www.corewebprogramming.com

External Style Sheets

- Specify link to external style sheet in the HEAD section of the HTML document

```
<LINK REL=STYLESHEET
      HREF="Sitestyle.css" // Absolute or relative link
      TYPE="text/css">
```

- Sitestyle.css

```
/* Example of an external style sheet */

H1 { text-align: center;
     font-family: Arial
}
H2 { color: #440000;
     text-align: center;
     font-family: Arial Black, Arial, Helvetica, sans-serif
}
...
```

84

HTML, CSS, Javascript

www.corewebprogramming.com

Inline Style Specification

- Use the **STYLE** attribute defined for each HTML element to directly specify the style
- Example

```
...
<H1>New Advances in Physics</H1>
<P STYLE="margin-left: 0.5in;
        margin-right: 0.5in;
        font-style: italic">
This paper gives the solution to three
previously unsolved problems: turning lead into gold,
antigravity, and a practical perpetual motion machine.
...
```

Defining Style Classes

- To define an *element* style class proceed the HTML element by a period and class name

```
// Define an "abstract" paragraph type
P.abstract { margin-left: 0.5in;
             margin-right: 0.5in;
             font-style: italic }
```

- To use, supply the name of the style class in the **CLASS** attribute of the HTML element

```
<H1>New Advances in Physics</H1>
<P CLASS="abstract">
This paper gives the solution to three previously
unsolved problems: turning lead into gold,
antigravity, and a practical perpetual motion machine.
```


Defining Style Classes

- To define a *global* style class, omit the element name

```
// Style available to all elements
.blue { color: blue; font-weight: bold }
```

- To use, simply specify the style class in the CLASS attribute of the HTML element

```
<H2 CLASS="blue">A Blue Heading</H2>

<!-- Apply to a section of text -->
This text is in the default color, but
<SPAN CLASS="blue">this text is blue.</SPAN>
```

Defining Styles through User-Defined IDs

- An ID is like a class but can be applied only once in a document

```
<HEAD>
<TITLE>...</TITLE>
<STYLE TYPE="text/css">
<!--
#foo { color: red }
-->
</STYLE>
</HEAD>
<BODY>
...
<P ID="foo">
...
</BODY>
```


Style Sheet Precedence Rules

1. Rules marked “important” have the highest priority (rarely used)

```
H1 { color: black !important;
    font-family: sans-serif }
```

2. Author rules have precedence over reader rules

- Style sheet rules override browser preferences

3. More specific rules have precedence over less specific rules

```
#foo { ... }          // ID selector highest priority
P.big H1 { ... }      // Class higher over element
P STRONG { ... }      // Two tags higher than single tag
STRONG { ... }
```

4. In case of tie, the last rule has priority

Useful Font Properties

• font-weight

- Relative weight (boldness) of font
- normal | lighter | bold | bolder | 100 | 200 | ... | 900
 - H1 { font-weight : 200 }
 - H2 { font-weight : bolder }

• font-style

- Font face type within a family
- normal | italic | oblique
 - P { font-style : normal }
 - TH { font-style : italic }

• font-size

- Either relative or absolute size of font
- pt, pc, in, cm, mm | em, ex, px, % | xx-large | x-large | large | medium | small | x-small | xx-small | smaller | larger
 - STRONG { font-size: 150% }
 - P { font-size: 14pt }
 - P { font-size: xx-large }

• font-family

- Typeface family for the font
 - H1 { font-family: Arial }

CampBearClaw.html, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
    Transitional//EN">
<HTML>
<HEAD>
    <TITLE>Camp Bear Claw</TITLE>
    <LINK REL=STYLESHEET HREF="CampBearClaw.css"
        TYPE="text/css">
</HEAD>
<BODY>
<H1>Camp Bear Claw</H1>
We have the following activities:
<H2 CLASS="archery">Archery</H2>
<H2 CLASS="arts">Arts and Crafts</H2>
<H2 CLASS="horseback">Horseback Riding</H2>
<H2 CLASS="hiking">Hiking</H2>
<H2 CLASS="campfire">Campfire Song Times</H2>
<H2 CLASS="java">Java Programming</H2>
</BODY>
</HTML>
```

91

HTML, CSS, Javascript

www.corewebprogramming.com

CampBearClaw.css

```
H1 { text-align: center;
      font-family: Funstuff }
H2.archery { font-family: ArcheryDisplay }
H2.arts { font-family: ClampettsDisplay }
H2.horseback { font-family: Rodeo }
H2.hiking { font-family: SnowtopCaps }
H2.campfire { font-family: Music Hall }
H2.java { font-family: Digiface }
```

92

HTML, CSS, Javascript

www.corewebprogramming.com

CampBearClaw.html, Result



93

HTML, CSS, Javascript

www.corewebprogramming.com

Useful Text Properties

- **text-decoration**
 - Describes text “decorations” that are added to the text of an element
 - none | underline | overline | line-through | blink
 - `P { text-decoration: underline }`
- **vertical-align**
 - Determines how elements are positioned vertically
 - top | bottom | baseline | middle | sub | super | text-top | text-bottom | %
- **text-align**
 - Determines how paragraphs are positioned horizontally
 - left | right | center | justify
- **text-indent**
 - Specifies the indentation of the first line of the paragraph
 - +/- pt, pc, in, cm, mm | +/- em, ex, px, %
 - `P { text-indent: -25px } /* Hanging indent */`
- **line-height**
 - Specifies the distance between two consecutive baselines in a paragraph
 - normal | number | pt, pc, in, cm, mm | em, ex, px,
 - `.double { line-height: 200% }`
 - `.triple { line-height: 3 } /* 3x the font size */`
 - `DIV { line-height: 1.5em }`

94

HTML, CSS, Javascript

www.corewebprogramming.com

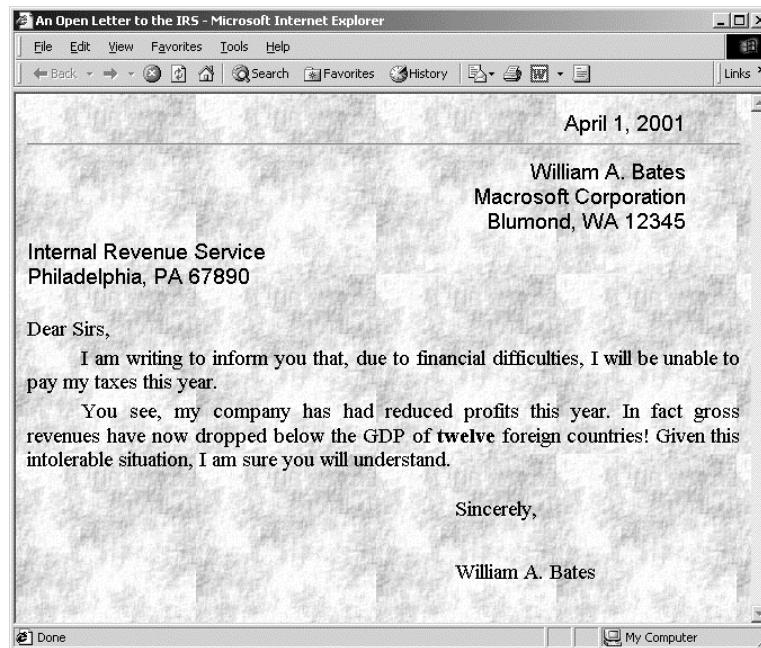
Bates.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>An Open Letter to the IRS</TITLE>
  <LINK REL=STYLESHEET HREF="Bates.css" TYPE="text/css">
</HEAD>
<BODY BACKGROUND="images/bond-paper.jpg">
<P CLASS="rhead">
April 1, 2001
<HR>
<P CLASS="rhead">
William A. Bates<BR>
Macrosoft Corporation<BR>
Blumond, WA 12345
<P CLASS="lhead">
Internal Revenue Service<BR>
Philadelphia, PA 67890
<P>
Dear Sirs,
<P CLASS="body">
I am writing to inform you that, due to financial difficulties,
...
```

Bates.css

```
P { margin-top: 5px }
P.rhead { text-align: right;
          margin-right: 0.5in;
          font-family: sans-serif }
P.lhead { font-family: sans-serif }
P.body { text-align: justify;
          text-indent: 0.5in }
P.foot { margin-left: 60%;
          line-height: 300% }
```

Bates.html



97

HTML, CSS, Javascript

www.corewebprogramming.com

Useful Foreground and Background Properties

- **color**
 - Color of the text or foreground color
 - color-name | #RRGGBB | #RGB | rgb(rrr, ggg, bbb) | rgb(rrr%, ggg%, bbb%)
 - P { color : blue }
 - H1 { color : #00AABB }
 - H3 { color : rgb(255, 0, 0) } /* red */
- **background-image**
 - none | url(filename)
 - Specifies an image to use as the background of region
 - H2 { background-image: url(Bluedrop.gif); }
- **background-repeat**
 - Specifies how to tile the image in the region
 - repeat | repeat-x | repeat-y | norepeat
 - BODY { background-image: url(Bluedot.gif);
 - background-repeat: repeat-x; }
- **background**
 - Lets you combine properties in a single entry
 - P { background: url(wallpaper.jpg) repeat-x }

98

HTML, CSS, Javascript

www.corewebprogramming.com

Cabinets.html, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
    Transitional//EN">
<HTML>
<HEAD>
    <TITLE>Joe's Cabinets</TITLE>
    <LINK REL=STYLESHEET HREF="Cabinets.css" TYPE="text/css">
</HEAD>
<BODY>
<CENTER>
<TABLE WIDTH=360 HEIGHT=199>
    <TR><TD ALIGN="CENTER" CLASS="banner">Joe's Cabinets
</TABLE>
</CENTER>
<P>
Welcome to Joe's Cabinets. We specialize in
<UL>
    <LI>Custom Cabinets
    <LI>Kitchen Remodeling
</UL>
</BODY>
</HTML>
```

99

HTML, CSS, Javascript

www.corewebprogramming.com

Cabinets.css

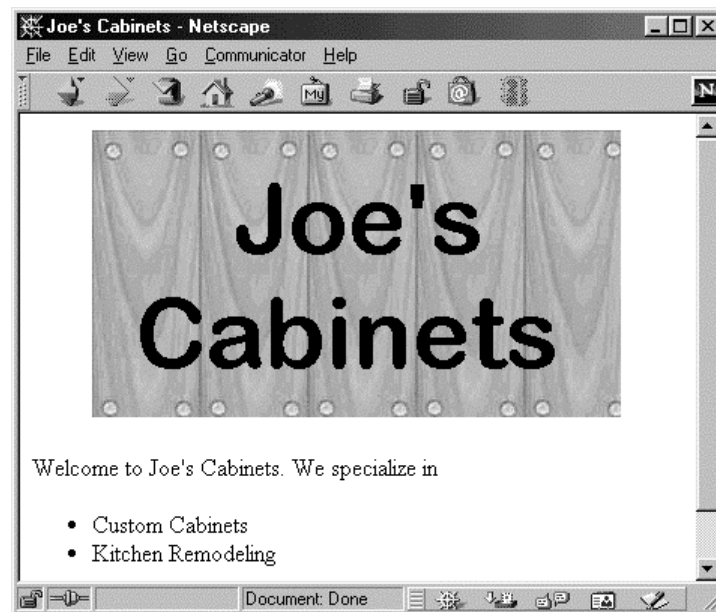
```
.banner { background: url(images/boards.jpg) repeat-x;
           font-size: 50pt;
           font-family: Arial Rounded MT Bold }
```

100

HTML, CSS, Javascript

www.corewebprogramming.com

Cabinets.html, Result



101

HTML, CSS, Javascript

www.corewebprogramming.com

Properties of the Bounding Box

- **CSS assume that all elements result in one or more rectangular regions (bounding box)**
- **Styles can specify the margins, borders, and padding of the bounding box**

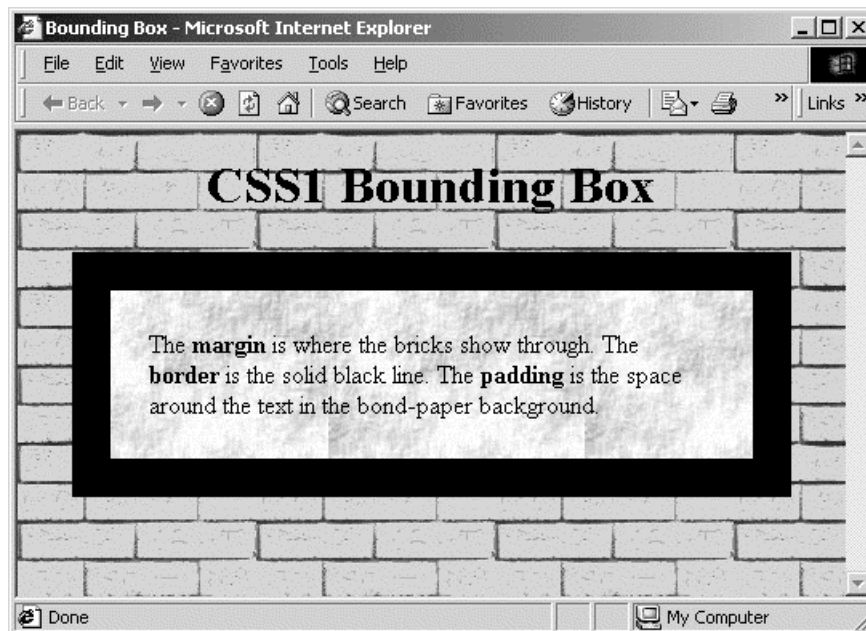
```
P { margin: 0.25in;  
    border: 0.25in solid black;  
    padding: 0.25in;  
    background: URL(images/bond-paper.jpg) }
```

102

HTML, CSS, Javascript

www.corewebprogramming.com

The Bounding Box



103

HTML, CSS, Javascript

www.corewebprogramming.com

Images and Floating Elements

- **width, height**
 - Specify a fixed size for an element (usually an image)
 - auto | pt, pc, in, cm, mm | em, ex, px
 - `IMG.bullet { width: 50px; height: 50px }`
- **float**
 - This property lets elements float into the left or right margins where the text wraps around
 - none | left | right

104

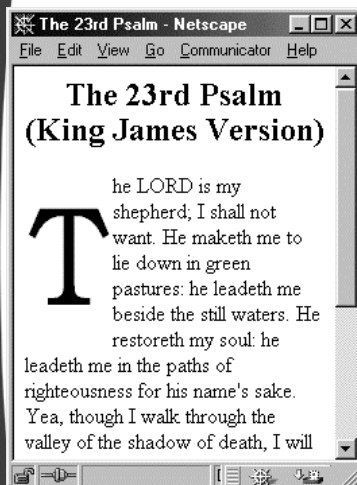
HTML, CSS, Javascript

www.corewebprogramming.com

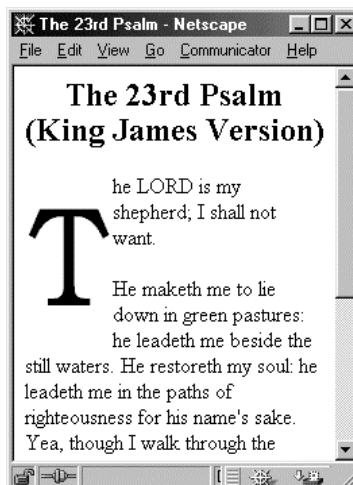
Psalm23.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>The 23rd Psalm</TITLE>
  <STYLE>
    <!--
    SPAN { float: left;
          font-family: "Cushing Book";
          font-size: 75pt }
    -->
  </STYLE>
</HEAD>
<BODY>
  <H2 ALIGN="CENTER">
    The 23rd Psalm (King James Version)</H2>
  <SPAN>T</SPAN>he LORD is my shepherd; I shall not want.
  He maketh me to lie down in green pastures: he leadeth me
  beside the still waters. He restoreth my soul: he leadeth me
  in the paths of righteousness for his name's sake. Yea,
```

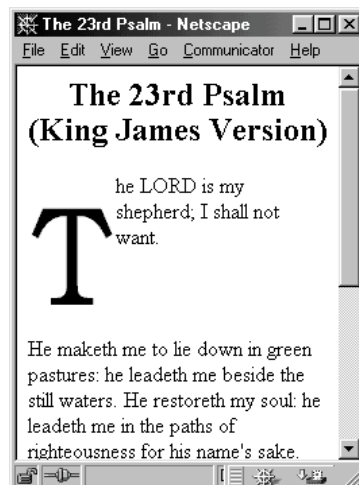
Psalm23.html, Result



The float property can be used to implement "drop caps"



Adding a <P> element simply continues the flow



Adding <P STYLE="clear: left"> forces the next paragraph to start after the floating element

Appendix, Length Units

Unit	Description
cm	Centimeters (absolute unit)
em	The height of the current font (relative unit)
ex	The height of the letter "x" in the current font (relative unit)
in	Inches (absolute unit)
mm	Millimeters (absolute unit)
pc	Picas; 6 picas per inch; 12 points per pica (absolute unit)
pt	Points; 72 points per inch (absolute unit)
px	Pixels (relative unit)

Summary

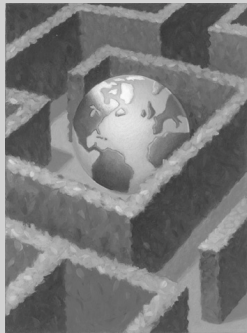
- **Through style sheets you can specify the general formatting of HTML elements**
- **Use external style sheets to share styles across all documents in the Web site**
- **Class definitions allow you to define multiple styles for an HTML element**
- **LAYERs are only supported by Netscape 5; however, a viable alternative are style sheet layers**

CSS Resources

- A short introduction to HTML and CSS
 - Getting started with HTML:
<http://www.w3.org/MarkUp/Guide/>
 - Adding a touch of style:
<http://www.w3.org/MarkUp/Guide/Style.html>
 - More advanced features:
<http://www.w3.org/MarkUp/Guide/Advanced.html>
- W3Schools tutorial
<http://www.w3schools.com/css/default.asp>
- The Cascading Style Sheets home page at W3C
<http://www.w3.org/Style/CSS/>

HTML, CSS, Javascript

122



core
WEB
programming

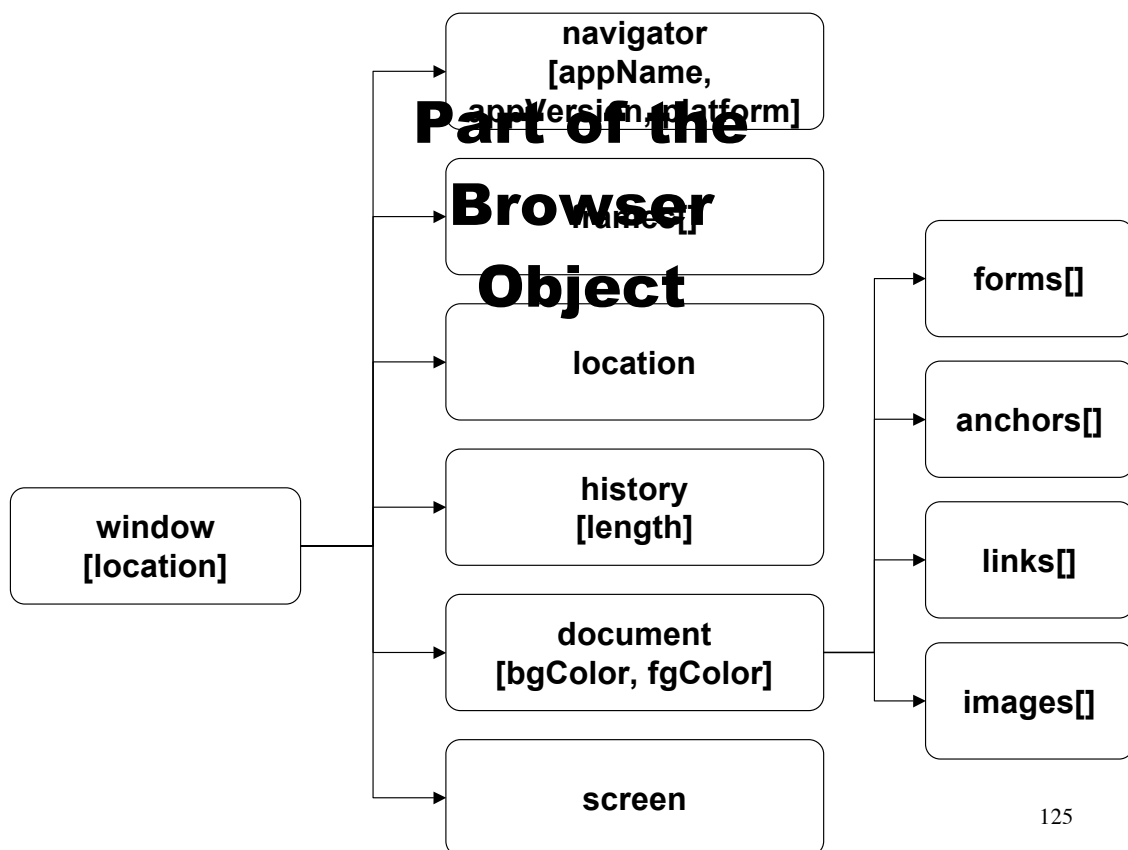
Questions?

The Browser Object

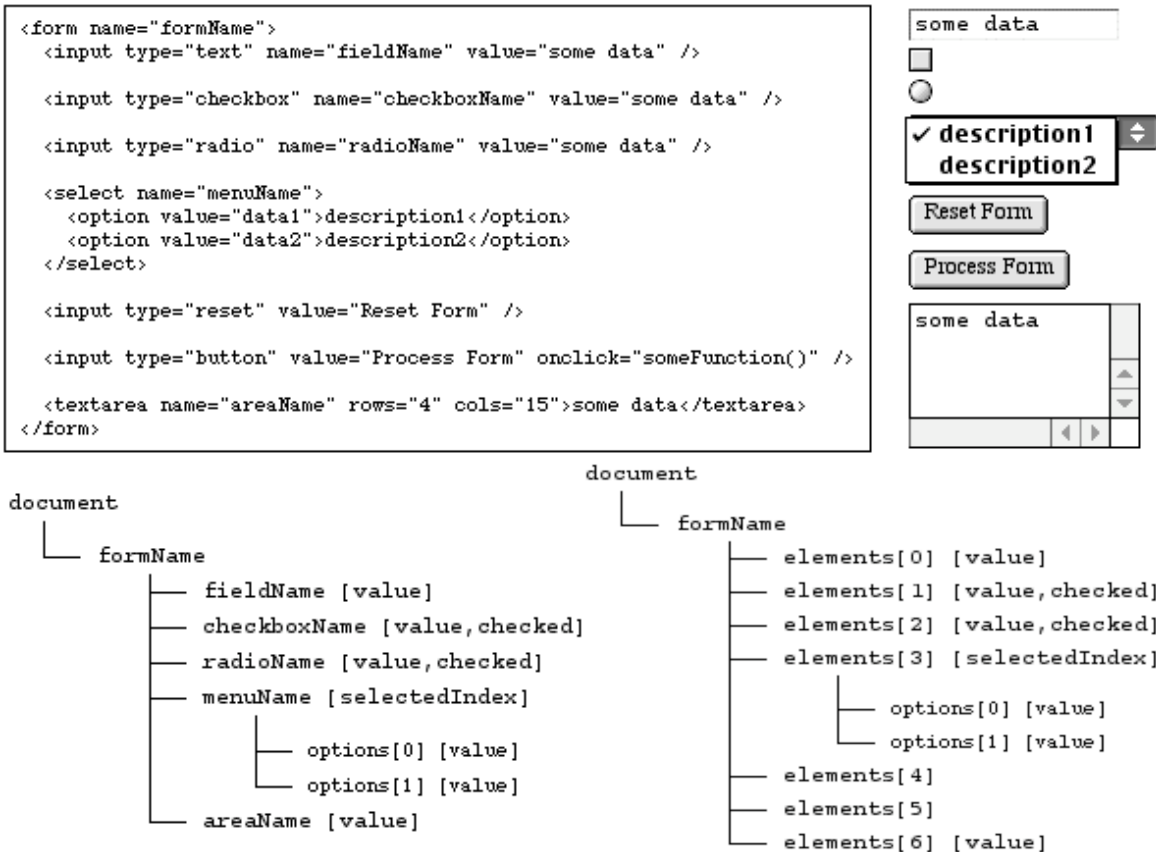
Eleni Stroulia

HTML, CSS, Javascript

124



125



Event handlers

- Event Handlers (listeners) are special properties of the various objects.
 - window [onfocus, onblur, ...]
 - document [onload, onunload, ...]
 - image [onmouseover, onmouseout, ...]
 - link [onclick, ...]

Client-side data processing

- These form elements are incapable of holding numeric data.
 - Text field -- a string with no new line characters
 - Text area -- a string potentially with new line characters
- There is no built-in way to test which one(s) are checked; you have to loop over them and simply test each one.
 - Checkbox -- Boolean Variable
 - Radio Button -- Boolean Variable
- Menus
 - Single selection -- Works like a single selection group of radio buttons.
 - Multiple selection -- Works like a group of checkboxes.
 - The selectedIndex property of the menu holds the currently (first/last) selected menu index -- if multiple selection element, have to iterate to find user's choice.

Data Validation

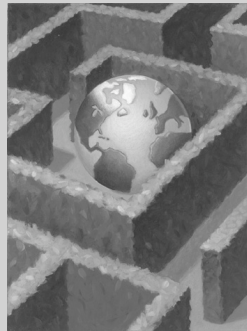
```
<form name="formname"
      method="GET"
      action="http://www.cknuckles.com/cgi/echo.cgi">
  ...
  <input type="submit" value="Submit Form" />
</form>

<script language="JavaScript">
  document.formname.onSubmit=verify;
  // has to be after def. of submit button

  function verify() { ... }
</script>
```

Validation using string object

String	
Properties	Methods
length	<code>charAt(index)</code> returns character at specified index
	<code>indexOf(char)</code> returns index of first occurrence of char, otherwise returns -1 (char can be a string)
	<code>lastIndexOf(char)</code> returns index of last occurrence of char, otherwise returns -1 (char can be a string)
	<code>match(/pattern/)</code> returns true if pattern is found in string (use i command and modifier for case insensitive matching)
	<code>replace(/pattern/,str)</code> replaces first occurrence of pattern with the replacement str (use g command and modifier for global replacement)
	<code>split(char)</code> returns an array of substrings, splitting around the delimiting char
	<code>substr(start,length)</code> returns a substring from start index of specified length in characters
	<code>toLowerCase()</code> replaces all caps with lower case characters
	<code>toUpperCase()</code> replaces all lower case characters with caps



core
WEB
programming

HTML Forms

Sending Data to Server-Side
Programs

Agenda

- Sending data from forms
- The FORM element
- Text controls
- Push buttons
- Check boxes and radio buttons
- Combo boxes and list boxes
- File upload controls
- Server-side image maps
- Hidden fields
- Grouping controls
- Tab ordering

132

HTML, CSS, Javascript

www.corewebprogramming.com

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>

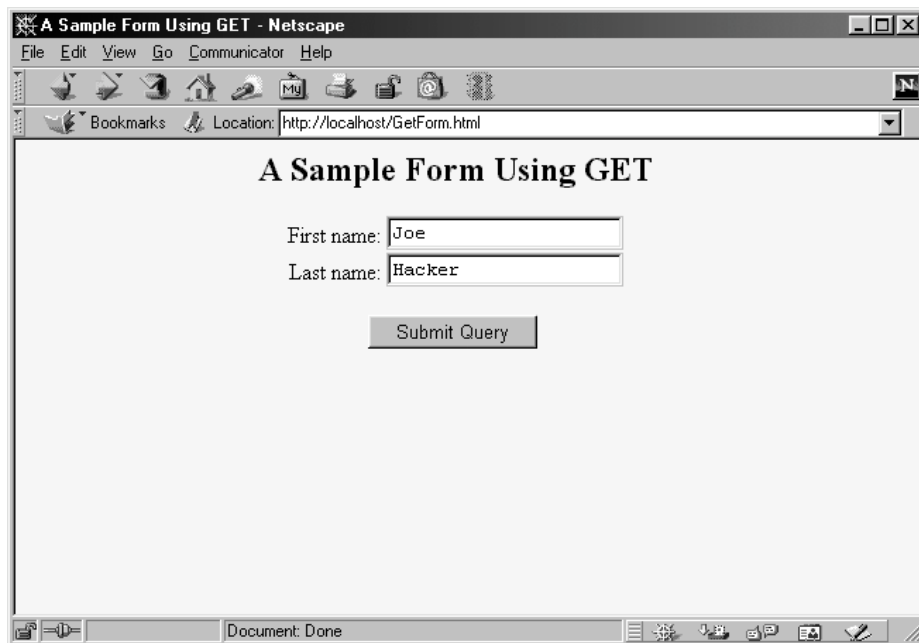
</BODY></HTML>
```

133

HTML, CSS, Javascript

www.corewebprogramming.com

Initial Result

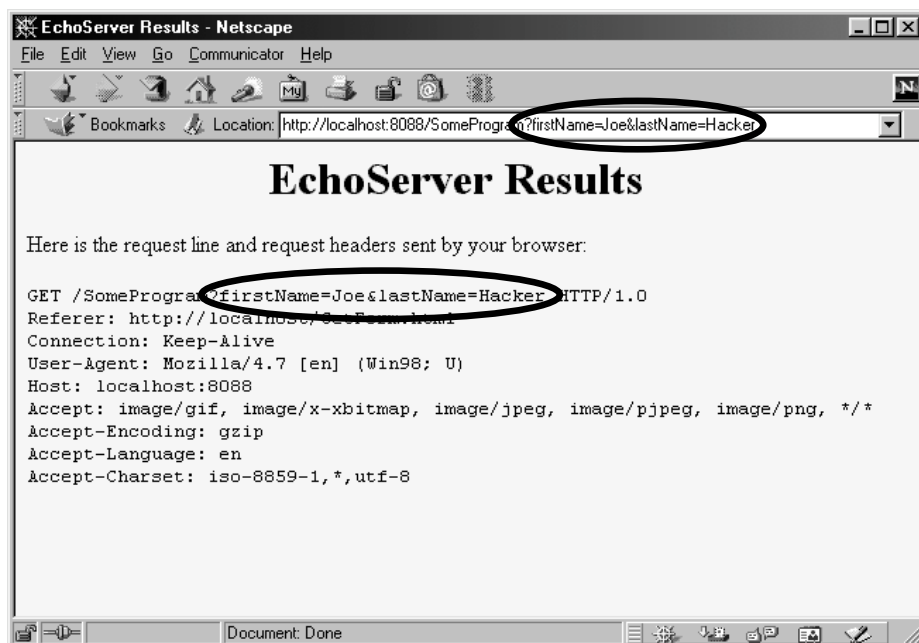


134

HTML, CSS, Javascript

www.corewebprogramming.com

Submission Result



135

HTML, CSS, Javascript

www.corewebprogramming.com

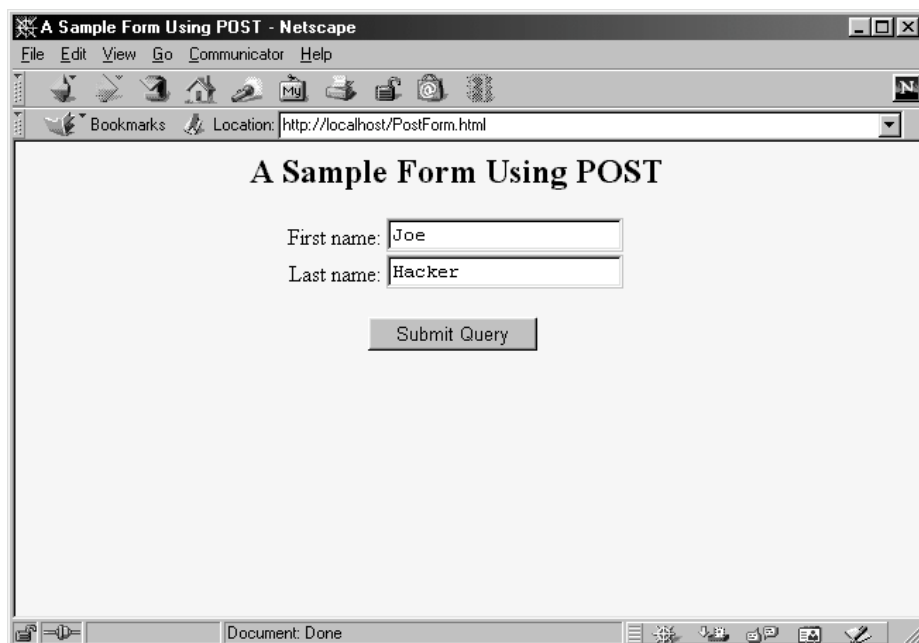
Sending Data with POST

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

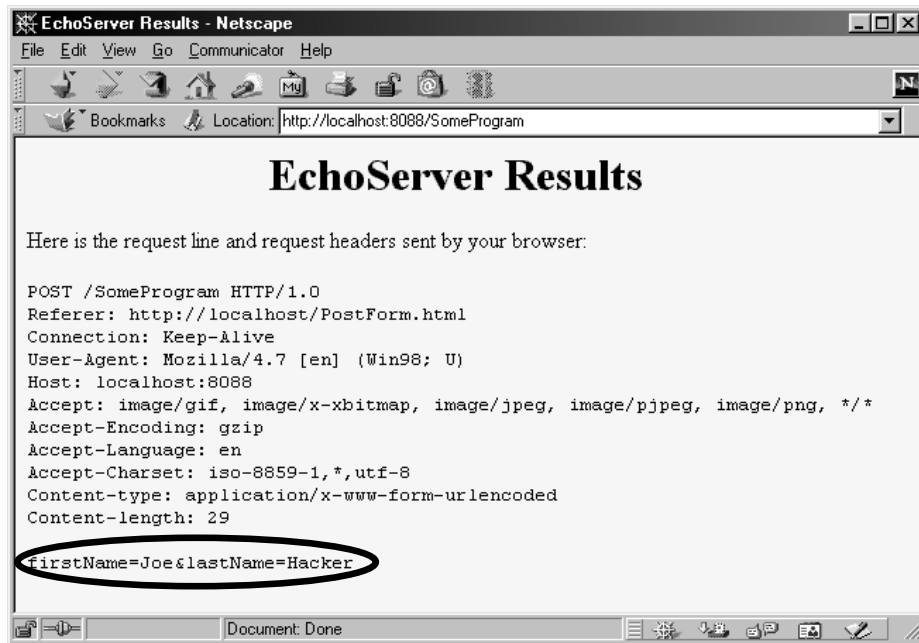
<FORM ACTION="http://localhost:8088/SomeProgram"
      METHOD="POST">
  <CENTER>
    First name:
    <INPUT TYPE="TEXT" NAME="firstName" VALUE="Joe"><BR>
    Last name:
    <INPUT TYPE="TEXT" NAME="lastName" VALUE="Hacker"><P>
    <INPUT TYPE="SUBMIT">
  </CENTER>
</FORM>

</BODY></HTML>
```

Initial Result



Submission Result

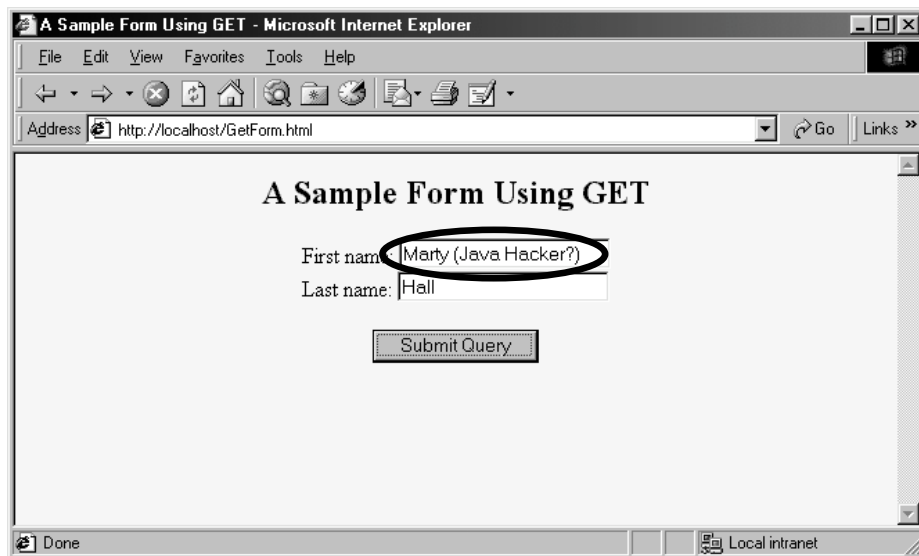


138

HTML, CSS, Javascript

www.corewebprogramming.com

URL Encoding: Original Form

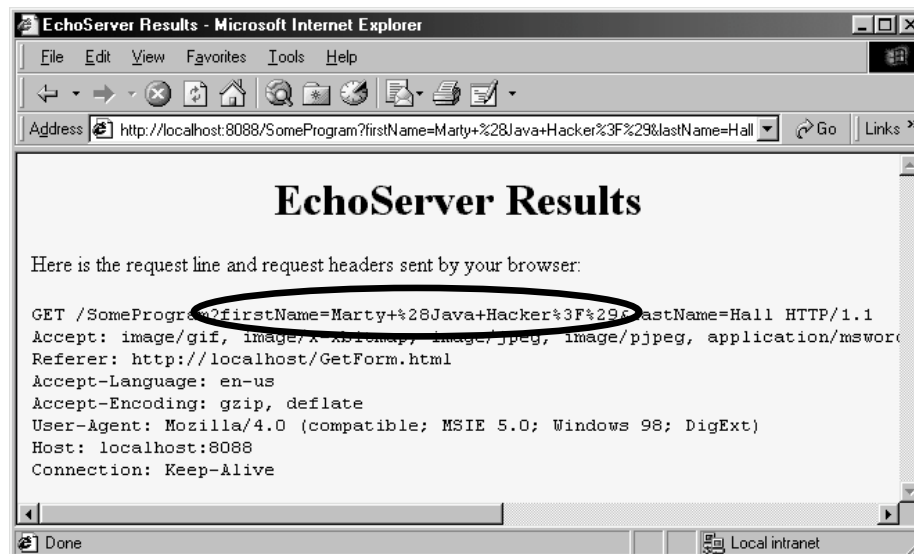


139

HTML, CSS, Javascript

www.corewebprogramming.com

URL Encoding: Result



140

HTML, CSS, Javascript

www.corewebprogramming.com

Text Controls

- **Textfields**

- `<INPUT TYPE="TEXT" NAME="..." ...>`
 - VALUE can give original value

- **Password Fields**

- `<INPUT TYPE="PASSWORD" NAME="..." ...>`
 - Always use POST

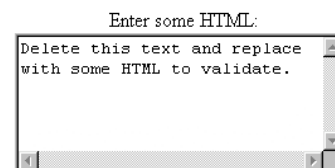
Enter Password:

- **Text Areas**

- `<TEXTAREA NAME="..." ROWS="..." COLS="...">`

`</TEXTAREA>`

- Interpretation of regular HTML tags turned off between `<TEXTAREA...>` and `</TEXTAREA>`



141

HTML, CSS, Javascript

www.corewebprogramming.com

Push Buttons

- **Submit Buttons**

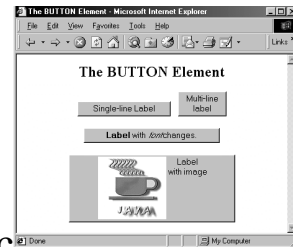
- `<INPUT TYPE="SUBMIT" ...>`
 - Use `NAME` if you have multiple buttons
 - Use `VALUE` to change button's label

- **Reset Buttons**

- `<INPUT TYPE="RESET" ...>`
 - Use `VALUE` to change button's label

- **JavaScript Buttons**

- `<INPUT TYPE="BUTTON" onClick="someJavaScriptFunction()" ...>`



- **Fancy Buttons**

- `<BUTTON TYPE="SUBMIT" ...>HTML</BUTTON>`
 - Internet Explorer and Netscape 6 only

142

HTML, CSS, Javascript

www.corewebprogramming.com

Using Multiple Submit Buttons

`<CENTER>`

Item:

`<INPUT TYPE="TEXT" NAME="Item" VALUE="256MB SIMM">
`

`<INPUT TYPE="SUBMIT" NAME="Add" VALUE="Add Item to Cart">`

`<INPUT TYPE="SUBMIT" NAME="Delete" VALUE="Delete Item from Cart">`

`</CENTER>`

143

HTML, CSS, Javascript

www.corewebprogramming.com

Check Boxes

- **Format**

- `<INPUT TYPE="CHECKBOX" NAME="..." ...>`
 - The `CHECKED` attribute makes it initially checked
 - Name/value pair sent only if checkbox is checked when form is submitted

- **Example code**

```
<P>
<INPUT TYPE="CHECKBOX" NAME="noEmail" CHECKED>
Check here if you do <I>not</I> want to
get our email newsletter
```

- **Example result**

☒ Check here if you do *not* want to get our email newsletter

Radio Buttons

- **Format**

- `<INPUT TYPE="RADIO" NAME="..." VALUE="..."...>`
 - All radio buttons in a group should have same `NAME`
 - Only one button in a group can be pressed; pressing a different one causes previous one to pop out

- **Example**

```
<DL>
<DT>Credit Card:
<DD><INPUT TYPE="RADIO" NAME="creditCard"
        VALUE="visa">
    Visa
<DD><INPUT TYPE="RADIO" NAME="creditCard"
        VALUE="mastercard">
    Master Card
    ...
</DL>
```

Credit Card:

☐ Visa

☐ Master Card

☒ Java Smart Card

☐ American Express

☐ Discover

Combo Boxes

- **Format**

- SELECT gives NAME
- OPTION gives VALUE

Favorite language:

- **Example**

Favorite language:

Favorite language:

```
<SELECT NAME="language">
  <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>
```



List Boxes

- **Format**

- Identical to combo boxes, but specify MULTIPLE

- **Example**

Languages you know:


```
<SELECT NAME="language" MULTIPLE>
  <OPTION VALUE="c">C
  <OPTION VALUE="c++">C++
  <OPTION VALUE="java" SELECTED>Java
  <OPTION VALUE="lisp">Lisp
  <OPTION VALUE="perl" SELECTED>Perl
  <OPTION VALUE="smalltalk">Smalltalk
</SELECT>
```

Languages you know:



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*
- **Hidden fields**
 - Preset NAME and VALUE sent with form submission..
- **Grouping Controls**
 - FIELDSET lets you visually group forms.
 - Internet Explorer and Netscape 6 only.
- **Tab order control**
 - TABINDEX (Internet Explorer and Netscape 6 only)

148

HTML, CSS, Javascript

www.corewebprogramming.com

Summary

- **General process**
 - FORM uses ACTION to specify base URL
 - Input elements each have a NAME
 - User enters values
 - When form submitted, URL is
baseURL?name1=value1&name2=value2&...
 - For POST requests, name/value pairs sent on separate line
(not part of URL)
- **Textfields**
 - <INPUT TYPE="TEXT" ...>
- **Submit Buttons**
 - <INPUT TYPE="SUBMIT" ...>

149

HTML, CSS, Javascript

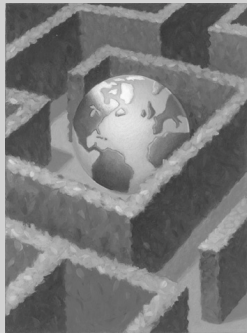
www.corewebprogramming.com

HTML Resources

- HTML tutorial at W3schools <http://www.w3schools.com/html>
- A short introduction to HTML and CSS
 - Getting started with HTML: <http://www.w3.org/MarkUp/Guide/>
 - Adding a touch of style: <http://www.w3.org/MarkUp/Guide/Style.html>
 - More advanced features: <http://www.w3.org/MarkUp/Guide/Advanced.html>
- NCSA--A Beginner's Guide to HTML Home Page
<http://archive.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>
- The HTML home page at W3C <http://www.w3.org/MarkUp/>
- HTML 4.0
 - <http://www.w3.org/TR/html4/cover.html>
 - <http://www.w3.org/TR/REC-html40/>
 - <http://www.htmlhelp.com/reference/html40/>
- HTML forms - <http://www.w3.org/TR/html4/interact/forms.html>

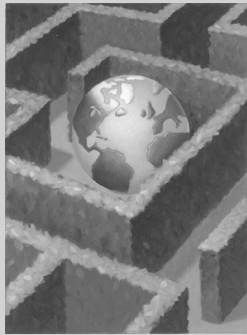
HTML, CSS, Javascript

150



core
WEB
programming

Questions?



core
WEB
programming

JavaScript

Adding Dynamic Content
to Web Pages

152

© 2001-2005 Marty Hall, Larry Brown <http://www.corewebprogramming.com>

Agenda

- **Generating HTML Dynamically**
- **Monitoring User Events**
- **Basic JavaScript Syntax**
- **Applications**
 - Using JavaScript to customize Web pages
 - Using JavaScript to make pages more dynamic
 - Using JavaScript to validate CGI forms
 - Using JavaScript to manipulate HTTP cookies
 - Using JavaScript to interact with and control frames
 - Controlling applets and calling Java from JavaScript
 - Accessing JavaScript from Java

153

HTML, CSS, Javascript

www.corewebprogramming.com

Generating HTML Dynamically

- **Idea**
 - Script is interpreted as page is loaded, and uses `document.write` or `document.writeln` to insert HTML at the location the script occurs

- **Template**

```
...
<BODY>
Regular HTML

<SCRIPT TYPE="text/javascript">
<!--
Build HTML Here
// -->
</SCRIPT>

More Regular HTML
</BODY>
```

A Simple Script

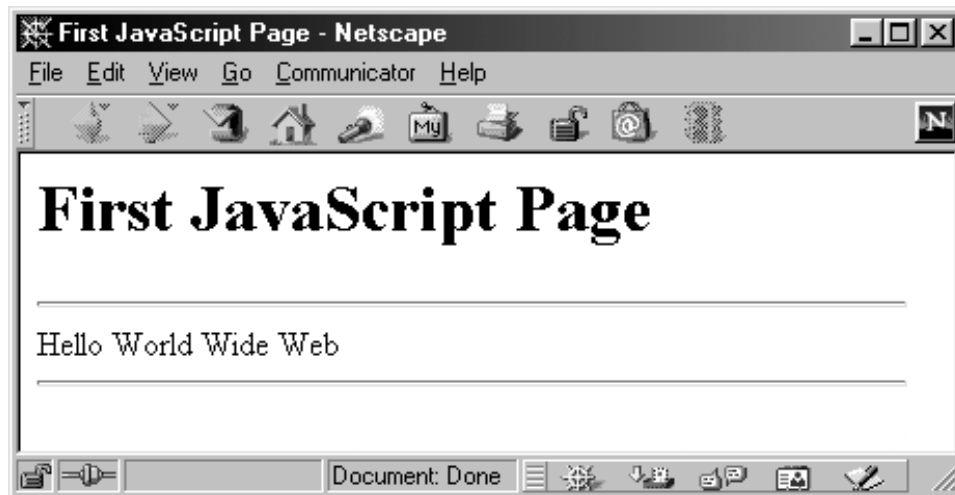
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
  <TITLE>First JavaScript Page</TITLE>
</HEAD>

<BODY>
<H1>First JavaScript Page</H1>

<SCRIPT TYPE="text/javascript">
<!--
document.write("<HR>");
document.write("Hello World Wide Web");
document.write("<HR>");
// -->
</SCRIPT>

</BODY>
</HTML>
```

Simple Script, Result



Extracting Document Info with JavaScript, Example

```
<HTML>
<HEAD>
  <TITLE>Extracting Document Info with JavaScript</TITLE>
</HEAD>
<BODY BGCOLOR="WHITE">
  <H1>Extracting Document Info with JavaScript</H1>
  <HR>

  <SCRIPT TYPE="text/javascript">
    <!--

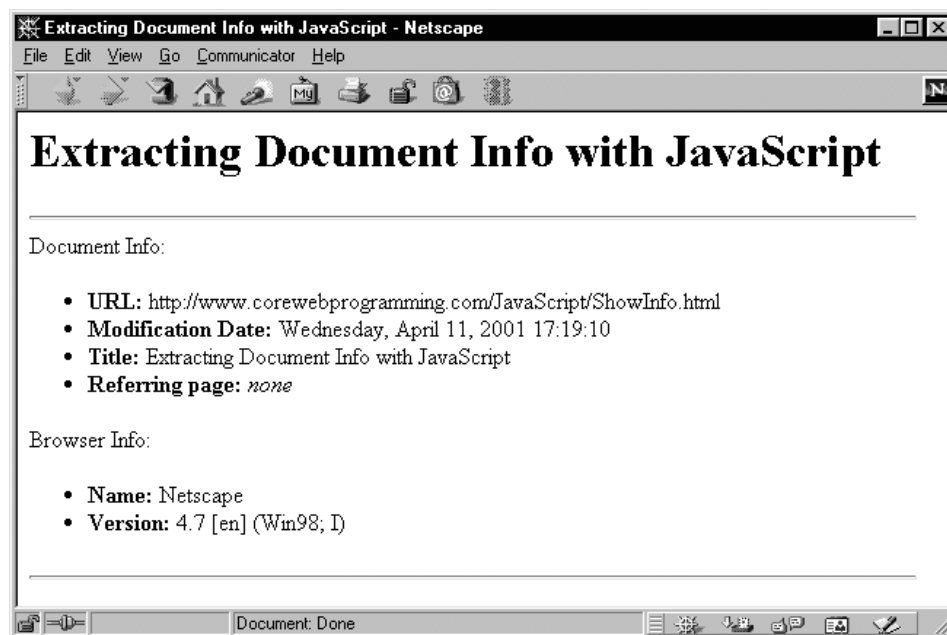
    function referringPage() {
      if (document.referrer.length == 0) {
        return("<I>none</I>");
      } else {
        return(document.referrer);
      }
    }
  </SCRIPT>
```

Extracting Document Info with JavaScript, Example, cont.

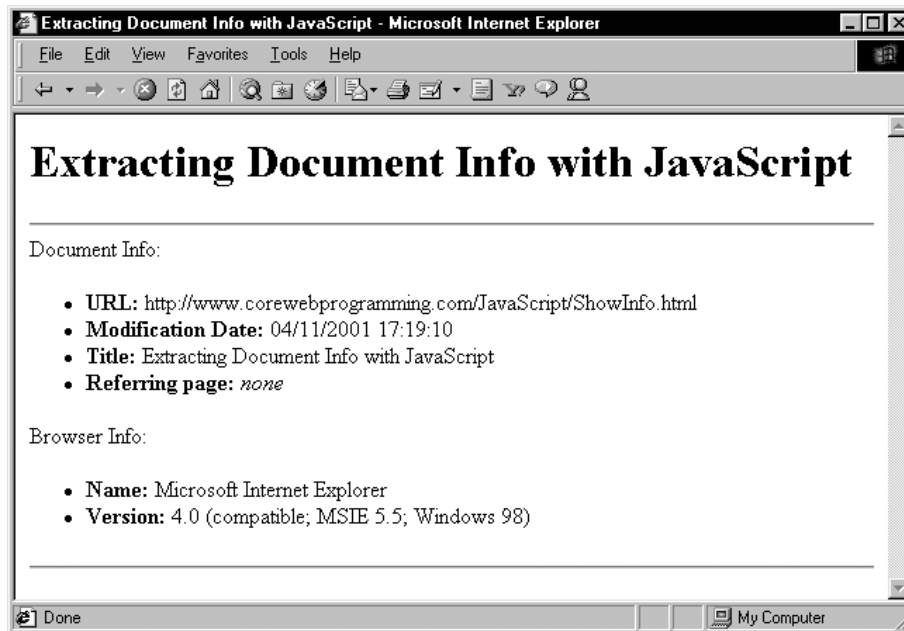
```
...
document.writeln
("Document Info:\n" +
"<UL>\n" +
"  <LI><B>URL:</B> " + document.location + "\n" +
"  <LI><B>Modification Date:</B> " + "\n" +
    document.lastModified + "\n" +
"  <LI><B>Title:</B> " + document.title + "\n" +
"  <LI><B>Referring page:</B> " + referringPage() + "\n" +
"</UL>");
document.writeln
("Browser Info:" + "\n" +
"<UL>" + "\n" +
"  <LI><B>Name:</B> " + navigator.appName + "\n" +
"  <LI><B>Version:</B> " + navigator.appVersion + "\n" +
"</UL>");
// -->
</SCRIPT>

<HR>
</BODY>
</HTML>
```

Extracting Document Info with JavaScript, Result



Extracting Document Info with JavaScript, Result



160

HTML, CSS, Javascript

www.corewebprogramming.com

Multi-Browser Compatibility

1. Use Language Attribute

```
<SCRIPT LANGUAGE="JavaScript">
<!--
languageVersion = "1.0";
// -->
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript1.1">
<!--
languageVersion = "1.1";
// -->
</SCRIPT>

...

<SCRIPT LANGUAGE="JavaScript1.5">
<!--
languageVersion = "1.5";
// -->
</SCRIPT>
```

Note: Don't include that attribute `TYPE="text/javascript"`

161

HTML, CSS, Javascript

www.corewebprogramming.com

Multi-Browser Compatibility, cont.

2. Use Vendor/Version Info

- navigator.appName
- navigator.appVersion

Monitoring User Events

• Use Various onXxx Attributes

- onClick
- onLoad
- onMouseOver
- onFocus
- etc.

User Events, Example

```
<HTML>
<HEAD>
  <TITLE>Simple JavaScript Button</TITLE>
  <SCRIPT TYPE="text/javascript">
    <!--
    function dontClick() {
      alert("I told you not to click!");
    }
    // -->
  </SCRIPT>
</HEAD>

<BODY BGCOLOR="WHITE">
  <H1>Simple JavaScript Button</H1>

  <FORM>
    <INPUT TYPE="BUTTON"
      VALUE="Don't Click Me"
      onClick="dontClick()" ">
  </FORM>
</BODY>
</HTML>
```

User Events, Result



JavaScript Syntax: Dynamic Typing

- **Idea**

- Like Lisp, values are typed, not variables
- A value is only checked for proper type when it is operated upon

- **Example**

```
var x = 5; // int
x = 5.5; // float
x = "five point five"; // String
```

JavaScript Syntax: Function Declarations

1. Declaration Syntax

- Functions are declared using the function reserved word
- The return value is not declared, nor are the types of the arguments
- Examples:

```
function square(x) {
    return(x * x);
}

function factorial(n) {
    if (n <= 0) {
        return(1);
    } else {
        return(n * factorial(n - 1));
    }
}
```

JavaScript Syntax: Function Declarations, cont.

2. First Class Functions

- Functions can be passed and assigned to variables
- Example

```
var fun = Math.sin;  
alert("sin(pi/2)=" + fun(Math.PI/2));
```



JavaScript Syntax: Objects and Classes

1. Fields Can Be Added On-the-Fly

- Adding a new property (field) is a simple matter of assigning a value to one
- If the field doesn't already exist when you try to assign to it, JavaScript will create it automatically.
- For instance:

```
var test = new Object();  
test.field1 = "Value 1"; // Create field1 property  
test.field2 = 7; // Create field2 property
```

JavaScript Syntax: Objects and Classes, cont.

2. You Can Use Literal Notation

- You can create objects using a shorthand “literal” notation of the form

```
{ field1:val1, field2:val2, ... , fieldN:valN }
```

- For example, the following gives equivalent values to `object1` and `object2`

```
var object1 = new Object();  
object1.x = 3;  
object1.y = 4;  
object1.z = 5;  
  
object2 = { x:3, y:4, z:5 };
```

JavaScript Syntax: Objects and Classes, cont.

3. The "for/in" Statement Iterates Over Properties

- JavaScript, unlike Java or C++, has a construct that lets you easily retrieve all of the fields of an object
- The basic format is as follows:

```
for(fieldName in object) {  
    doSomethingWith(fieldName);  
}
```

- Also, given a field name, you can access the field via `object["field"]` as well as via `object.field`

Field Iteration, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>For/In Loops</TITLE>

<SCRIPT TYPE="text/javascript">
<!--

function makeObjectTable(name, object) {
  document.writeln("<H2>" + name + "</H2>");
  document.writeln("<TABLE BORDER=1>\n" +
    "  <TR><TH>Field<TH>Value");
  for(field in object) {
    document.writeln ("  <TR><TD>" + field +
      "<TD>" + object[field]);
  }
  document.writeln("</TABLE>");
}
// -->
</SCRIPT>
```

Field Iteration, Example

```
...
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>For/In Loops</H1>

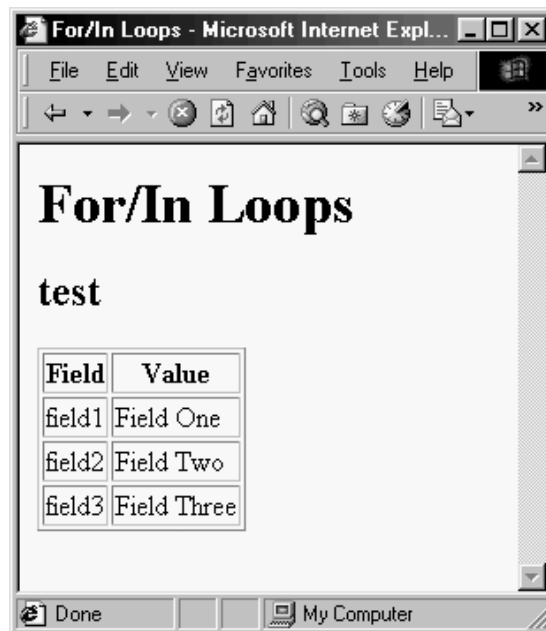
<SCRIPT TYPE="text/javascript">
<!--

var test = new Object();
test.field1 = "Field One";
test.field2 = "Field Two";
test.field3 = "Field Three";
makeObjectTable("test", test);

// -->
</SCRIPT>

</BODY>
</HTML>
```

Field Iteration, Result



The for/in statement iterates over object properties

174

HTML, CSS, Javascript

www.corewebprogramming.com

JavaScript Syntax: Objects and Classes, cont.

4. A “Constructor” is Just a Function that Assigns to “this”

- JavaScript does not have an exact equivalent to Java’s class definition
- The closest you get is when you define a function that assigns values to properties in the `this` reference
- Calling this function using `new` binds `this` to a new Object
- For example, following is a simple constructor for a Ship class

```
function Ship(x, y, speed, direction) {  
    this.x = x;  
    this.y = y;  
    this.speed = speed;  
    this.direction = direction;  
}
```

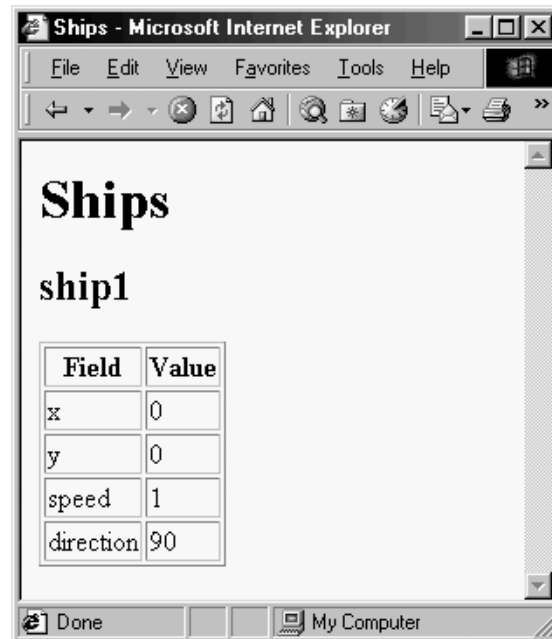
175

HTML, CSS, Javascript

www.corewebprogramming.com

Constructor, Example

```
var ship1 =  
    new Ship(0, 0, 1, 90);  
makeObjectTable("ship1", ship1);
```



176

HTML, CSS, Javascript

www.corewebprogramming.com

JavaScript Syntax: Objects and Classes, cont.

5. Methods Are Function-Valued Properties

- No special syntax for defining methods of objects
- Instead, you simply assign a function to a property

177

HTML, CSS, Javascript

www.corewebprogramming.com

Class Methods, Example

- Consider a version of the Ship class that includes a move method

```
function degreesToRadians(degrees) {  
    return(degrees * Math.PI / 180.0);  
}  
  
function move() {  
    var angle = degreesToRadians(this.direction);  
    this.x = this.x + this.speed * Math.cos(angle);  
    this.y = this.y + this.speed * Math.sin(angle);  
}  
  
function Ship(x, y, speed, direction) {  
    this.x = x;  
    this.y = y;  
    this.speed = speed;  
    this.direction = direction;  
    this.move = move;  
}
```

Class Methods, Result

```
var ship1 = new Ship(0, 0, 1, 90);  
makeObjectTable("ship1 (originally)", ship1);  
ship1.move();  
makeObjectTable("ship1 (after move)", ship1);
```

Field	Value
x	0
y	0
speed	1
direction	90
move	function move() { var angle = degreesToRadians(this.direction); this.x = this.x + this.speed * Math.cos(angle); this.y = this.y + this.speed * Math.sin(angle); }

Field	Value
x	6.123031769111886e-17
y	1
speed	1
direction	90
move	function move() { var angle = degreesToRadians(this.direction); this.x = this.x + this.speed * Math.cos(angle); this.y = this.y + this.speed * Math.sin(angle); }

JavaScript Syntax: Objects and Classes, cont.

5. Arrays

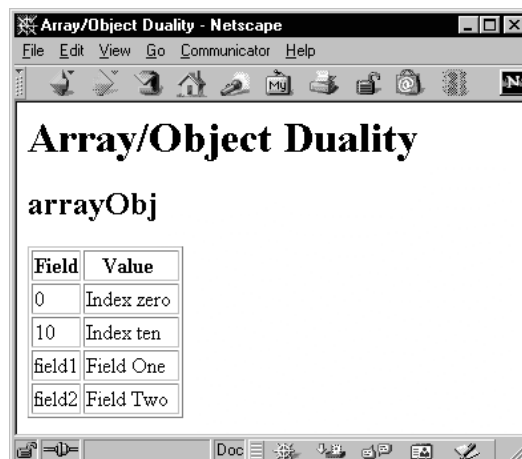
- For the most part, you can use arrays in JavaScript a lot like Java arrays.
 - Here are a few examples:

```
var squares = new Array(5);
for(var i=0; i<squares.length; i++) {
    vals[i] = i * i;
}
// Or, in one fell swoop:
var squares = new Array(0, 1, 4, 9, 16);
var array1 = new Array("fee", "fie", "fo", "fum");
// Literal Array notation for creating an array.
var array2 = [ "fee", "fie", "fo", "fum" ];
```
 - Behind the scenes, however, JavaScript simply represents arrays as objects with numbered fields
 - You can access named fields using either `object.field` or `object["field"]`, but numbered fields only via `object[fieldNumber]`

Array, Example

```
var arrayObj = new Object();
arrayObj[0] = "Index zero";
arrayObj[10] = "Index ten";
arrayObj.field1 = "Field One";
arrayObj["field2"] = "Field Two";
```

```
makeObjectTable("arrayObj",
               arrayObj);
```



Application: Adjusting to the Browser Window Size

- **Netscape 4.0 introduced the `window.innerWidth` and `window.innerHeight` properties**
 - Lets you determine the usable size of the current browser window

Determining Browser Size, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Strawberries</TITLE>

  <SCRIPT TYPE="text/javascript">
    <!--
    function image(url, width, height) {
      return('<IMG SRC="' + url + '"' +
        ' WIDTH=' + width +
        ' HEIGHT=' + height + '>');
    }

    function strawberry1(width) {
      return(image("Strawberry1.gif", width, Math.round(width*1.323)));
    }

    function strawberry2(width) {
      return(image("Strawberry2.gif", width, Math.round(width*1.155)));
    }
    // -->
  </SCRIPT>
</HEAD>
```

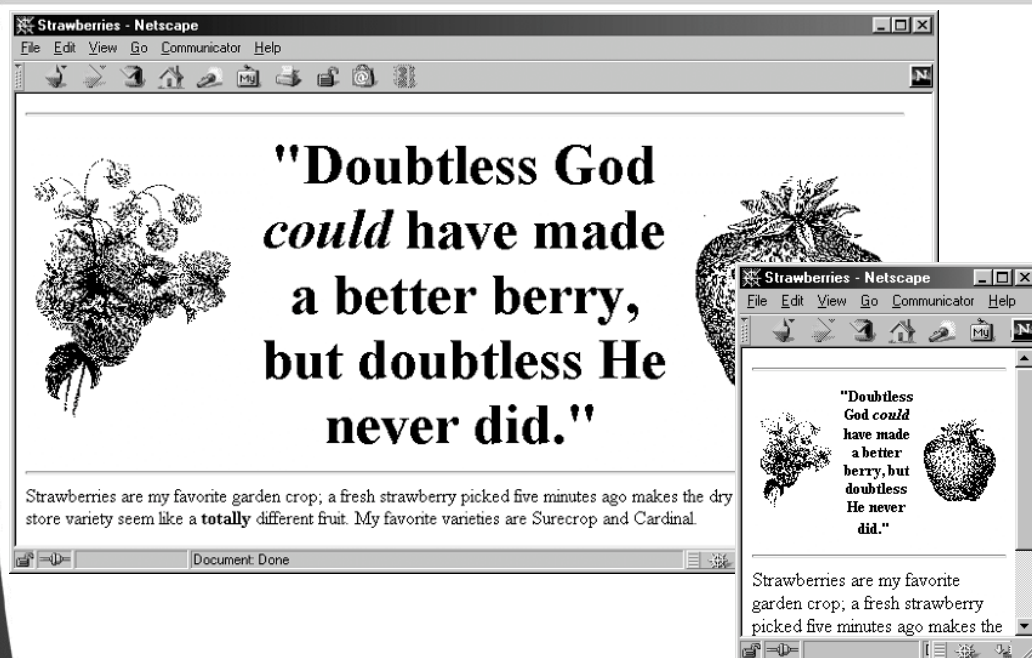
Determining Browser Size, Example, cont.

```
...
<SCRIPT TYPE="text/javascript">
<!--
    var imageWidth = window.innerWidth/4;
    var fontSize = Math.min(7, Math.round(window.innerWidth/100));

document.writeln
    ('<TABLE>\n' +
      '  <TR><TD>' + strawberry1(imageWidth) + '\n' +
      '    <TH><FONT SIZE=' + fontSize + '>\n' +
      '      "Doubtless God <I>could</I> have made\n' +
      '        a better berry, but doubtless He\n' +
      '        never did."</FONT>\n' +
      '    <TD>' + strawberry2(imageWidth) + '\n' +
      '  </TABLE>');
// -->
</SCRIPT>
<HR>

Strawberries are my favorite garden crop; a fresh ...
</BODY>
</HTML>
```

Determining Browser Size, Results



Application: Using JavaScript to Make Pages Dynamic

- **Modifying Images Dynamically**
 - The `document.images` property contains an array of Image objects corresponding to each IMG element in the current document
 - To display a new image, simply set the SRC property of an existing image to a string representing a different image file

Modifying Images, Example

- The following function changes the first image in a document

```
function changeImage() {  
    document.images[0].src = "images/new-image.gif";  
}
```

- Referring to images by name is easier:

```
<IMG SRC="cool-image.jpg" NAME="cool"  
    WIDTH=75 HEIGHT=25>
```

```
function improveImage() {  
    document.images["cool"].src = "way-cool.jpg";  
}
```

Modifying Images: A Clickable Image Button, Example

```
<SCRIPT TYPE="text/javascript">
<!--
imageFiles = new Array("images/Button1-Up.gif",
                        "images/Button1-Down.gif",
                        "images/Button2-Up.gif",
                        "images/Button2-Down.gif");
imageObjects = new Array(imageFiles.length);
for(var i=0; i<imageFiles.length; i++) {
    imageObjects[i] = new Image(150, 25);
    imageObjects[i].src = imageFiles[i];
}

function setImage(name, image) {
    document.images[name].src = image;
}

```

Modifying Images: A Clickable Image Button, Example

```
function clickButton(name, grayImage) {
    var origImage = document.images[name].src;
    setImage(name, grayImage);
    var resetString =
        "setImage('" + name + "', '" + origImage + "')";
    setTimeout(resetString, 100);
}
// -->
</SCRIPT>

</HEAD>
...
<A HREF="location1.html"
    onClick="clickButton('Button1', 'images/Button1-Down.gif')">
<IMG SRC="images/Button1-Up.gif" NAME="Button1"
    WIDTH=150 HEIGHT=25></A>

<A HREF="location2.html"
    onClick="clickButton('Button2', 'images/Button2-Down.gif')">
<IMG SRC="images/Button2-Up.gif" NAME="Button2"
    WIDTH=150 HEIGHT=25></A>
...

```

Highlighting Images Under the Mouse, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
    Transitional//EN">

<HTML>
<HEAD>
    <TITLE>High Peaks Navigation Bar</TITLE>

<SCRIPT TYPE="text/javascript">
<!--

// Given "Foo", returns "images/Foo.gif".
function regularImageFile(imageName) {
    return("images/" + imageName + ".gif");
}

// Given "Bar", returns "images/Bar-Negative.gif".
function negativeImageFile(imageName) {
    return("images/" + imageName + "-Negative.gif");
}
```

Highlighting Images Under the Mouse, Example, cont.

```
// Cache image at specified index. E.g., given index 0,
// take imageNames[0] to get "Home". Then preload
// images/Home.gif and images/Home-Negative.gif.

function cacheImages(index) {
    regularImageObjects[index] = new Image(150, 25);
    regularImageObjects[index].src =
        regularImageFile(imageNames[index]);
    negativeImageObjects[index] = new Image(150, 25);
    negativeImageObjects[index].src =
        negativeImageFile(imageNames[index]);
}

imageNames = new Array("Home", "Tibet", "Nepal",
    "Austria", "Switzerland");
regularImageObjects = new Array(imageNames.length);
negativeImageObjects = new Array(imageNames.length);

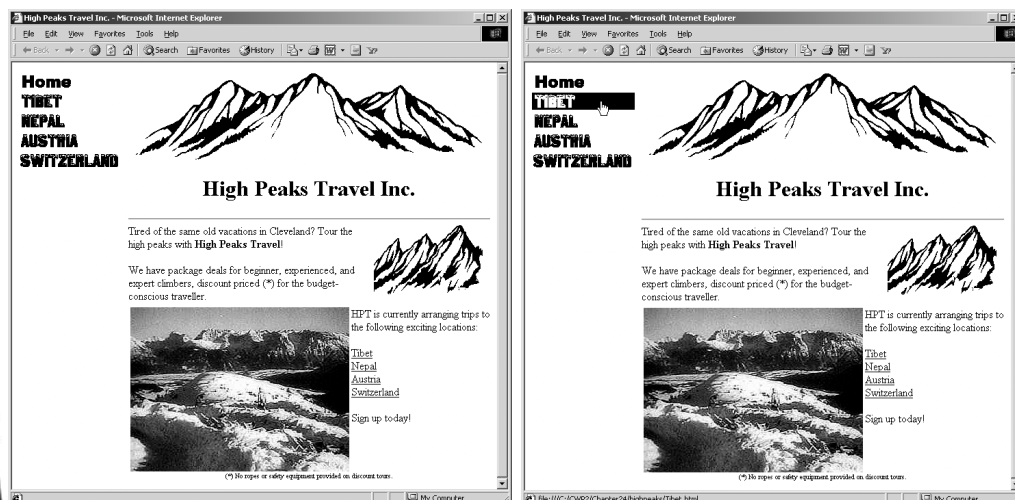
// Put images in cache for fast highlighting.
for(var i=0; i<imageNames.length; i++) {
    cacheImages(i);
}
```

Highlighting Images Under the Mouse, Example, cont.

```
...
function highlight(imageName) {
    document.images[imageName].src = negativeImageFile(imageName);
}

function unHighlight(imageName) {
    document.images[imageName].src = regularImageFile(imageName);
}
// -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="WHITE">
<TABLE BORDER=0 WIDTH=150 BGCOLOR="WHITE"
    CELLPADDING=0 CELLSPACING=0>
  <TR><TD><A HREF="Tibet.html"
    TARGET="Main"
    onMouseOver="highlight('Tibet')"
    onMouseOut="unHighlight('Tibet')">
    <IMG SRC="images/Tibet.gif"
    NAME="Tibet"
    WIDTH=150 HEIGHT=25 BORDER=0>
  </A>
  ...
```

Highlighting Images Under the Mouse, Result



Making Pages Dynamic: Moving Layers

- Netscape 4 introduced “layers” – regions that can overlap and be positioned arbitrarily
- JavaScript 1.2 lets you access layers via the `document.layers` array, each element of which is a `Layer` object with properties corresponding to the attributes of the `LAYER` element
- A named layer can be accessed via `document.layers["layer name"]` rather than by using an index, or simply by using `document.layerName`

Moving Layers, Example

- Descriptive overlays slowly “drift” to final spot when button clicked

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Camps on K-3</TITLE>

  <SCRIPT TYPE="text/javascript">
    <!--
    function hideCamps() {
      // Netscape 4 document model.
      document.layers["baseCamp"].visibility = "hidden";
      document.layers["highCamp"].visibility = "hidden";
      // Or document.baseCamp.visibility = "hidden";
    }

    function moveBaseCamp() {
      baseCamp.moveBy(1, 3);
      if (baseCamp.pageX < 130) {
        setTimeout("moveBaseCamp()", 10);
      }
    }
  </SCRIPT>
</HEAD>
<BODY>
```


Moving Layers, Example, cont.

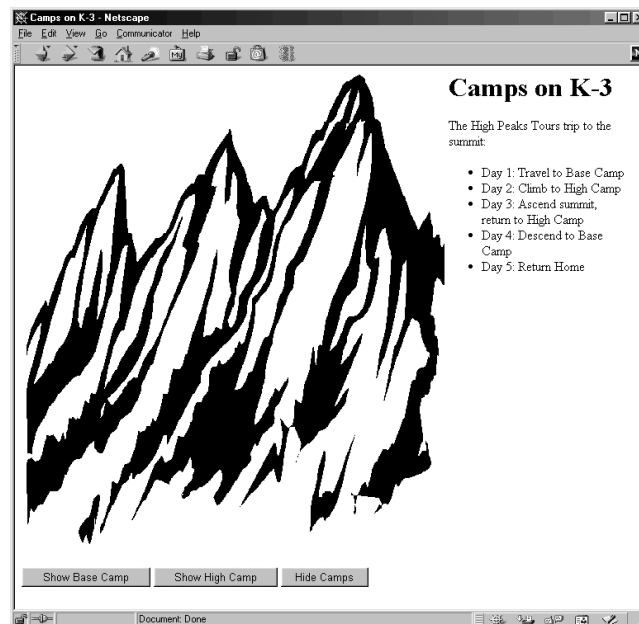
```
function showBaseCamp() {
    hideCamps();
    baseCamp = document.layers["baseCamp"];
    baseCamp.moveToAbsolute(0, 20);
    baseCamp.visibility = "show";
    moveBaseCamp();
}
function moveHighCamp() {
    highCamp.moveBy(2, 1);
    if (highCamp.pageX < 110) {
        setTimeout("moveHighCamp()", 10);
    }
}
function showHighCamp() {
    hideCamps();
    highCamp = document.layers["highCamp"];
    highCamp.moveToAbsolute(0, 65);
    highCamp.visibility = "show";
    moveHighCamp();
}
// -->
</SCRIPT>
```

Moving Layers, Example, cont.

```
<LAYER ID="highCamp" PAGEX=50 PAGEY=100 VISIBILITY="hidden">
    <TABLE>
        <TR><TH BGCOLOR="WHITE" WIDTH=50>
            <FONT SIZE="+2">High Camp</FONT>
            <TD><IMG SRC="images/Arrow-Right.gif">
        </TD>
    </TR>
    </TABLE>
</LAYER>
<LAYER ID="baseCamp" PAGEX=50 PAGEY=100 VISIBILITY="hidden">
    <TABLE>
        <TR><TH BGCOLOR="WHITE" WIDTH=50>
            <FONT SIZE="+2">Base Camp</FONT>
            <TD><IMG SRC="images/Arrow-Right.gif">
        </TD>
    </TR>
    </TABLE>
</LAYER>

<FORM>
    <INPUT TYPE="Button" VALUE="Show Base Camp"
        onClick="showBaseCamp()">
    <INPUT TYPE="Button" VALUE="Show High Camp"
        onClick="showHighCamp()">
    <INPUT TYPE="Button" VALUE="Hide Camps"
        onClick="hideCamps()">
</FORM>
```


Moving Layers, Result

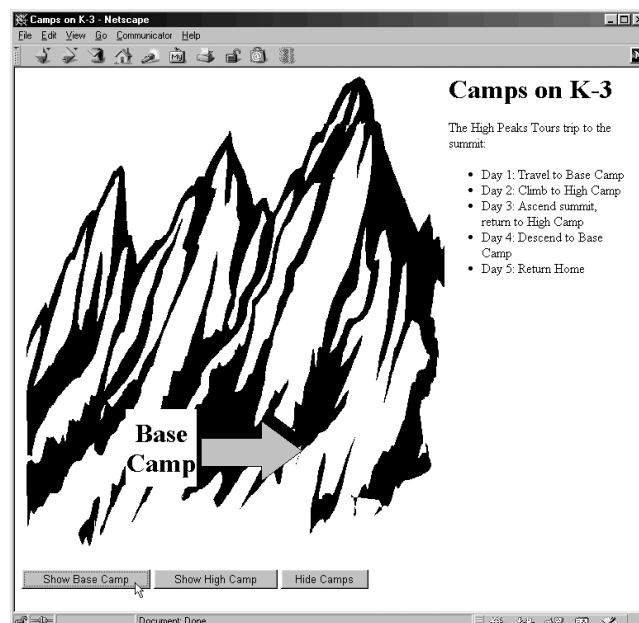


198

HTML, CSS, Javascript

www.corewebprogramming.com

Moving Layers, Result



199

HTML, CSS, Javascript

www.corewebprogramming.com

Application: Using JavaScript to Validate CGI Forms

1. Accessing Forms

- The `document.forms` property contains an array of `Form` entries contained in the document
- As usual in JavaScript, named entries can be accessed via name instead of by number, plus named forms are automatically inserted as properties in the document object, so any of the following formats would be legal to access forms

```
var firstForm = document.forms[0];  
// Assumes <FORM NAME="orders" ...>  
var orderForm = document.forms["orders"];  
// Assumes <FORM NAME="register" ...>  
var registrationForm = document.register;
```

Application: Using JavaScript to Validate CGI Forms, cont.

2. Accessing Elements within Forms

- The `Form` object contains an `elements` property that holds an array of `Element` objects
- You can retrieve form elements by number, by name from the array, or via the property name:

```
var firstElement = firstForm.elements[0];  
// Assumes <INPUT ... NAME="quantity">  
var quantityField = orderForm.elements["quantity"];  
// Assumes <INPUT ... NAME="submitSchedule">  
var submitButton = register.submitSchedule;
```

Checking Form Values Individually, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>On-Line Training</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
...
// When the user changes and leaves textfield, check
// that a valid choice was entered. If not, alert
// user, clear field, and set focus back there.

function checkLanguage() {
  // or document.forms["langForm"].elements["langField"]
  var field = document.langForm.langField;
  var lang = field.value;
  var prefix = lang.substring(0, 4).toUpperCase();
  if (prefix != "JAVA") {
    alert("Sorry, '" + lang + "' is not valid.\n" +
          "Please try again.");
    field.value = ""; // Erase old value
    field.focus();    // Give keyboard focus
  }
}
```

202

HTML, CSS, Javascript

www.corewebprogramming.com

Checking Form Values Individually, Example, cont.

```
// -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>On-Line Training</H1>

<FORM ACTION="cgi-bin/registerLanguage" NAME="langForm">
To see an introduction to any of our on-line training
courses, please enter the name of an important Web
programming language below.
<P>
<B>Language:</B>
<INPUT TYPE="TEXT" NAME="langField"
       onFocus="describeLanguage()"
       onBlur="clearStatus()"
       onChange="checkLanguage()">
<P>
<INPUT TYPE="SUBMIT" VALUE="Show It To Me">
</FORM>

</BODY>
</HTML>
```

203

HTML, CSS, Javascript

www.corewebprogramming.com

Checking Form Values Individually, Results



Checking Values When Form is Submitted, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Camp Registration</TITLE>
  <SCRIPT TYPE="text/javascript">
    <!--
    function isInt(string) {
      var val = parseInt(string);
      return(val > 0);
    }

    function checkRegistration() {
      var ageField = document.registerForm.ageField;
      if (!isInt(ageField.value)) {
        alert("Age must be an integer.");
        return(false);
      }
      ...
      // Format looks OK. Submit form.
      return(true);
    }
  // -->
</SCRIPT>
```

Checking Values When Form is Submitted, Example, cont.

```
<BODY BGCOLOR="WHITE">
<H1>Camp Registration</H1>

<FORM ACTION="cgi-bin/register"
      NAME="registerForm"
      onSubmit="return (checkRegistration())">
Age: <INPUT TYPE="TEXT" NAME="ageField"
      onFocus="promptAge()"
      onBlur="clearStatus()">

<BR>
Rank: <INPUT TYPE="TEXT" NAME="rankField"
      onFocus="promptRank()"
      onBlur="clearStatus()">

<BR>
Serial Number: <INPUT TYPE="TEXT" NAME="serialField"
      onFocus="promptSerial()"
      onBlur="clearStatus()">

<P>
<INPUT TYPE="SUBMIT" VALUE="Submit Registration">
</FORM>

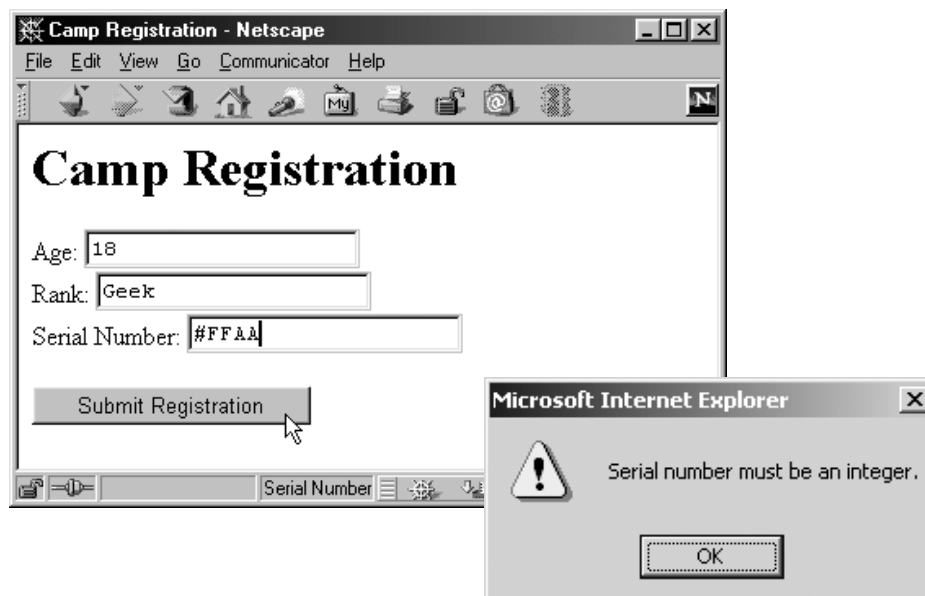
</BODY>
</HTML>
```

206

HTML, CSS, Javascript

www.corewebprogramming.com

Checking Values When Form is Submitted, Results



207

HTML, CSS, Javascript

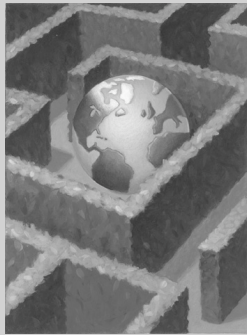
www.corewebprogramming.com

Summary

- **JavaScript permits you to:**
 - Customize Web pages based on the situation
 - Make pages more dynamic
 - Validate HTML form input
 - Manipulate cookies
 - Control frames
 - Integrate Java and JavaScript

Javascript Resources

- W3schools JavaScript tutorial
<http://www.w3schools.com/js/default.asp>
- JavaScript.com <http://www.javascript.com/>
- The JavaScript source
<http://javascript.internet.com>
- The JavaScript kit <http://javascriptkit.com/>



core
WEB
programming

Questions?