

Topic: HTML and JavaScript

- Encoding of a web page is an HTML file (*text* file, Notepad-editable), downloaded from server to browser
- Part of text in HTML file is to display
- Part is *tags* to format info or to introduce executable code
- Tagged information can include hyperlinks; names of other HTML files; GIF (graphics)

Exporting from MS Word to HTML

- To insert a hyperlink, select text, right click, and fill in form with URL
- To save as HTML, use *SaveAs*, select *Web Page*
- HTML file may be uploaded to a web server
- Note that while pictures may be embedded in Word files, they may not in HTML files

HTML tags

- Delimited by <, >
- Tags surround text to format it
- *Examples:*
 - **This <i> is </i> HTML**
This *is* HTML
 - **<title> 63.120 </title>**
(Designates text as a page title that appears in title bar in browser)
 - **<p> ... </p>** designates a paragraph
 - **<hr>** (generates horizontal rule)
 - **<a href =**
"http://www.cnn.com">Search
(hyperlink [Search](http://www.cnn.com))

Skeleton HTML file

- **<html>**
<head> ... <title> ... </title> ... </head>
<body> ... </body>
</html>
- Every tag above should appear, in this order, in every HTML file
- Text to display, and other tags, appears in place of “...” above
- File name must have extension *.htm* or *.html*

Hyperlinks

- When user clicks on *link text* of a hyperlink (e.g., blue underlined text)
 - Browser places current HTML file on a *stack* maintained by browser
 - Browser downloads Internet file referenced by hyperlink
 - Browser displays referenced file, whether HTML, PDF, or other
 - Clicking “Back” button returns browser to HTML file at top of stack

Internal hyperlinks

- A hyperlink may link to a location in the same HTML document (e.g., for a table of contents consisting of hyperlinks)
- *Example:*

```
<a href = #ch2> Chapter 2 </a>
...
<a name = ch2></a>
Chapter 2
...
```

Absolute and relative references

- An *absolute reference* is a URL, which specifies the exact location of data from whatever origin
- *Example:*
`FSC`
- Single slashes designate directory/subdirectory relationships
- A relative reference specifies a file's server location in relation to location of current HTML file
- *Example, one directory up from current:*
` Home `

Image and other links

- To display a graphic image, use an *image tag*, e.g.,
``
- Attributes controlled by tags: vertical and horizontal alignment, text wrap
- Formats: GIF (Graphic Image Format), JPG (Joint Photographic Experts Group)

Other tags

- Headings may be generated with named styles
- *Comments* occur between angle-brackets with “!”: `<!-- [Comment] -->`
- *Script* tag enables embedding of code from non-HTML languages, e.g., CGI, JavaScript (see topic 7, JavaScript)

Attributes and elements specified by tags

- Background: `<body bgcolor="yellow">`
- Text and link colors:
`<body text="red" link="green">`
`FSC`
- Rules: `<hr>` (horizontal rule)
- Lists (need `` for each item):
`` (unnumbered, bulleted)
`` (numbered)
`<dl>` (definitional)
- Tables: `<table border>`
- See files `list-example.htm`, `table-example.htm`

Named styles in HTML

```
<!--This example shows how to define a named style
in HTML, analogous to MS Word named styles-->
<html>

<head>
<title>David Keil HW 6</title>
<style> h1
  {color: red; font-family: arial; font-size: 24px}
</style>
</head>

<body>
<h1>This is my web page</h1>
This is text about my web page
<h1>More about my web page</h1>
This is more text about my web page
</body>
</html>
```

Computer Science I Using Java David Keil 1/08

HTML features

- Text links (hypertext)
- Graphic links (e.g., buttons)
- *Hot spots*: areas used as links
- *Forms*: Enable collection of data from user uploaded to server and processes by CGI script
- *Tables*: grids of cells for layout
- *Frames*: areas that scroll independently of the rest of page

Computer Science I Using Java David Keil 1/08

12

Extensions to HTML

- Dynamic HTML: enables animation, mouseover effects
- XML (Extensible Markup Language): supports structuring of data with tags, e.g., for fields of a database
- Scripting languages: JavaScript, VBScript, PerlScript
- Client-side scripts are embedded in HTML code, server-side scripts generate custom HTML files

Subtopic: Programming in a procedural language, JavaScript

- *Motivation:* Working with IT means thinking abstractly about data and operations
- Design, coding, testing of solutions is part of learning problem solving
- JavaScript: A language interpreted on browsers that has all the power of languages like C, C++, Java
- Compiled (offline translation) vs. interpreted (simultaneous translation) code
- Use tags `<script language = "Javascript"> ... </script>`
- The JavaScript text between these tags will execute when browser displays HTML file

Event-driven programming

- *Browsers* and most other apps are *interactive*: alternate input and output
- *Command-line environments*: URL line in browser, Google prompt, DOS or UNIX prompt
- *GUI events*: Windows/Mac/Xwindows
- *Features*: Icons, menus, dialog boxes, windows, buttons, scrollers, check boxes
- *Common feature*: User generates *events*, e.g., clicks, drags, keystrokes, timeouts
- One form of interaction in browser: hyperlinks
- Another is embedding of event-based JavaScript programs in HTML files

Simple JavaScript example

```
<html>  <!-- hello.htm -->
  <head><title>DK-Hello</title></head>
  <body>63.120 says Hello!
    <script language="JavaScript">
      alert("hello");
    </script>
  </body>
</html>
```

- Displays “hello” in an alert box (a kind of dialog)
- `alert` is a JavaScript function (a kind of procedure)
- JavaScript may be used after `script` tag

Simple button

```
<html> <!--button.htm-->
<head><title>63120 Hello</title></head>
<body>
  <input type=button value = "Hello"
        onClick = 'alert("Hi")'>
</body></html>
```

- This code displays “Hi” when “Hello” button is pressed
- **<input>** tag defines an input button object
- *Event handler*: code that specifies application’s response to a particular event, such as user click on a button

Computer Science I Using Java David Keil 1/087

Counting button clicks

```
<html> <head><title>yes-no counter</title></head>
<body>
  /* count-yes.htm Displays Yes, No buttons for user to click,
  counts # clicks on each. Four 'event-handlers' specify the
  program's response to four different input events: Yes, No,
  Stats, Reset. When user presses 'Yes' button, variable
  'num_yes' is incremented. */
  <script language="JavaScript">
    var num_yes=0, num_no=0; // Variables
  </script>
  <td><input type=button value = "Yes"
            onClick = 'num_yes = num_yes + 1'></td>
  <td><input type=button value = "No"
            onClick = 'num_no = num_no + 1'></td>
  <td><input type=button value = "Stats"
            onClick = 'alert("Yes: "+num_yes + " No: "+num_no)'></td>
  <td><input type=button value = "Reset"
            onClick = 'num_yes = num_no = 0'></td>
</body>
</html>
```

Computer Science I Using Java David Keil 1/088

Text input/output

```
<head><title>Input echo</title></head>
<body>
  <form name="Input"><table>
    <!-- Display prompt and get input:-->
    <td>Enter your user name:
    <!-- Generate input-box:-->
      <input type=text    name=user_name
              value=""   size = 15> </td>
    <!-- Wait for button-press, display message:-->
    <td><input type=button value="Done"
          onClick = 'alert("Hello " + user_name.value)''>
      </td>
    <!-- Assigning a value to onClick defines
          JavaScript response to clicking button-->
  </table></form>
</body>
```

JavaScript variables

- Variables have *name*, *type*, *value*
- Must declare a variable to use it
var a;
- Valid *types* (interpretation of bits):
- Number: **var a=2, b=3;**
- String: **var name = "Bill";**
- Truth value: **var greater = (a > b);**
- *Assignment* can change variable's value:
a = b * 4;

Expressions

- *Expressions* have *values*
- Expressions are *literals*, *variables*, or *function calls*, or may be formed with *operators* (+, -, *, /, %)
- *Boolean* expressions have truth values and may be built with *relational operators* (==, !=, <, >, <=, >=) or *logical operators* (!, &&, ||)

Statements

- *Statements* are *executable* (or variable declarations)
- Statements include *assignment*, *if*, *if ... else*, *while*, and compound statements
- *Simple* statements, e.g., assignment, terminate with “;”

Sources

J. Parsons, D. Oja. *Computer Concepts*, 9th ed.
Thomson, 2007.

L. Snyder. *Fluency with Information
Technology*. Addison Wesley, 2006.