Tutorial 10: Programming with JavaScript

Computer Science Department
University of Central Florida

COP 3175 – Internet Applications

Objectives

- Learn the history of JavaScript
- Create a script element
- Understand basic JavaScript syntax
- Write text to a Web page with JavaScript
- Learn about JavaScript data types

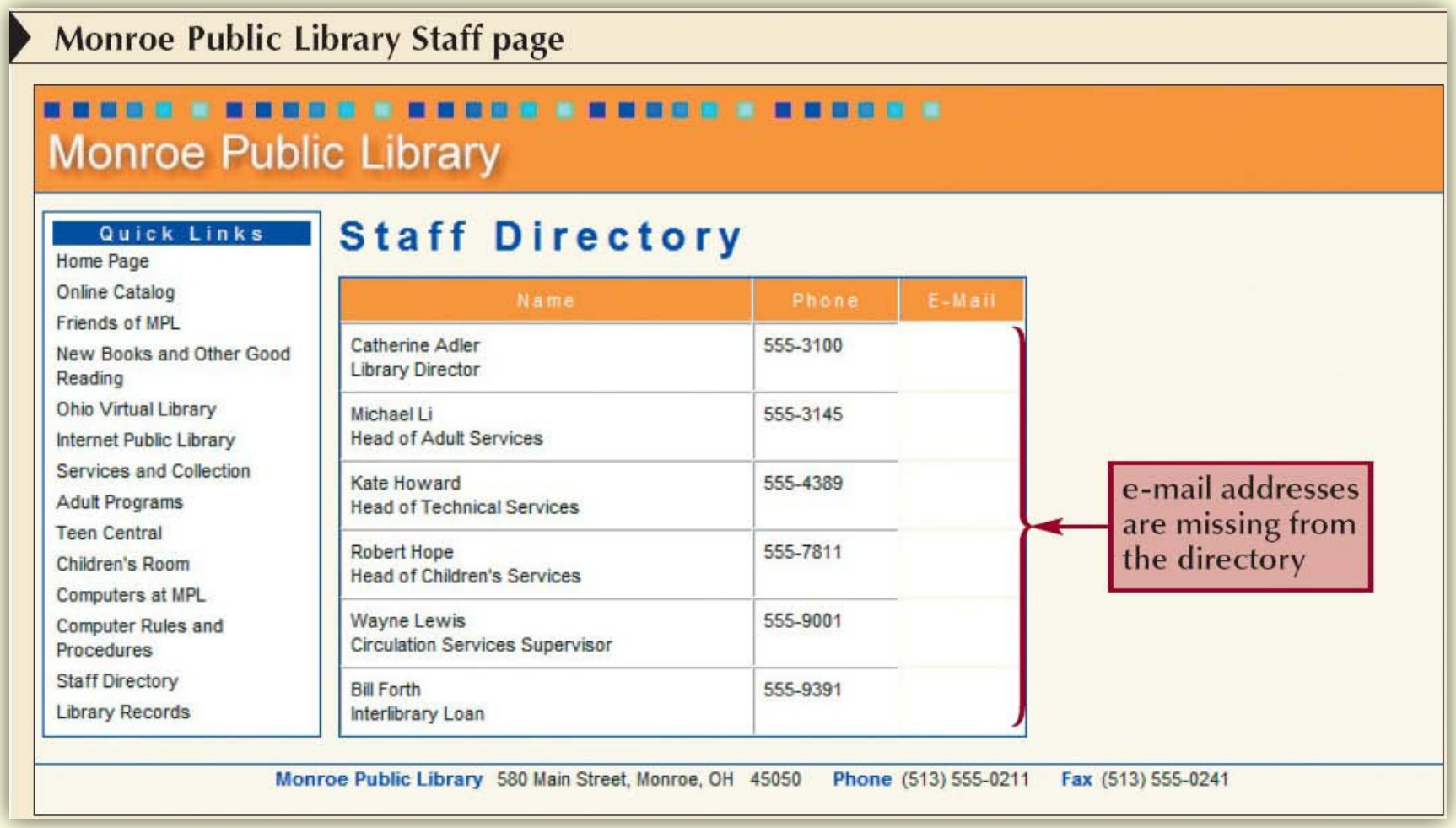
Objectives

- Declare and work with variables
- Create and call a JavaScript function
- Access an external JavaScript file
- Add comments to JavaScript code
- Learn about basic debugging techniques and tools

Tutorial 10 Website

- Monroe Public Library
  - Kate Howard is the in-house IT guru
    - but she's no Web developer
  - The Library's Web page has the email addresses of all staff available online
    - as such, these emails can be scanned from the HTML files and used by spammers
  - Kate wants to somehow scramble the email address within the actual HTML code
  - At the same time, she wants them viewable when the page is rendered by the browser

Tutorial 10 Website

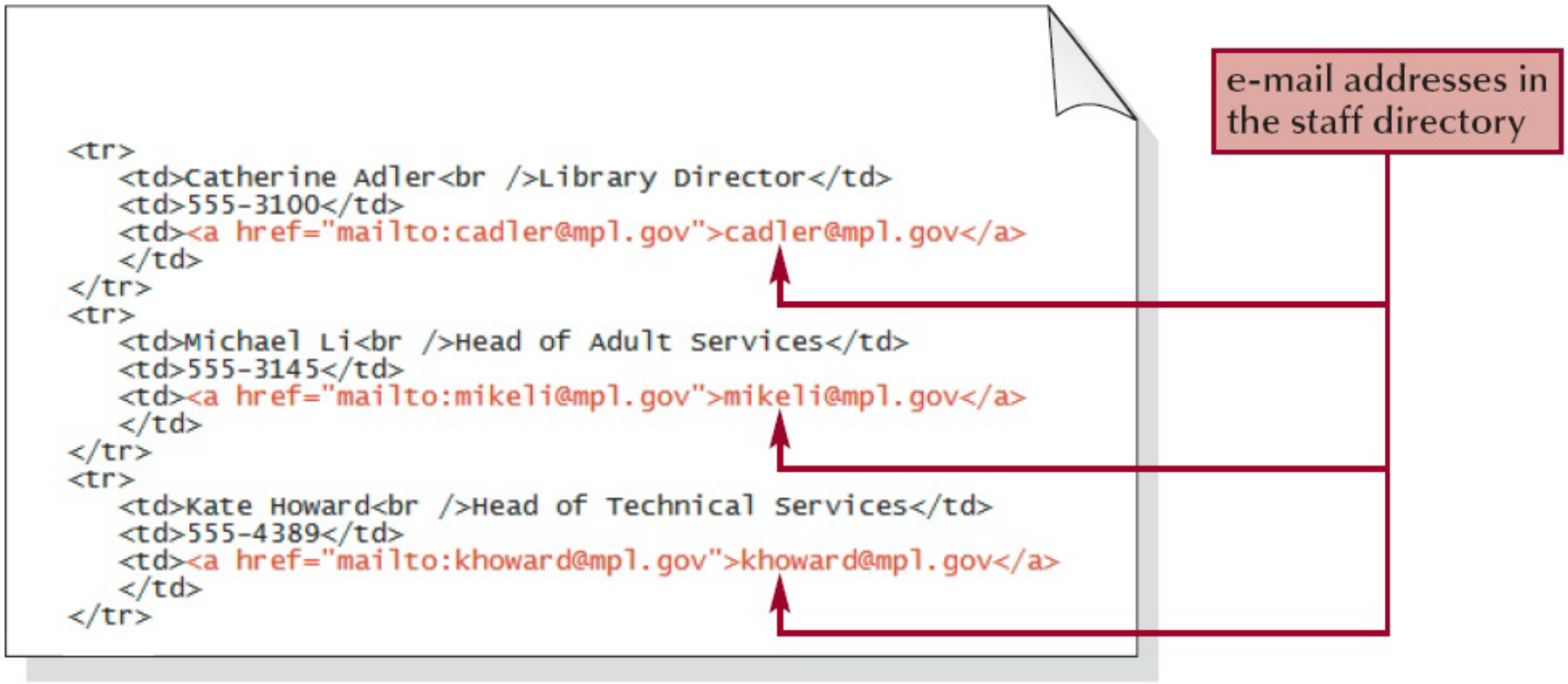


Tutorial 10 Website

- Email Harvesters
  - One way spammers collect email addresses is through the use of email harvesters
    - These are programs that scan documents
      - usually Web pages
    - and search the documents for email addresses
    - Once an address is found, it is added to a database, which is then used for sending spam
  - So when you put your email address directly into HTML code, you are giving easy access for email harvesting programs to snatch your email
    - and spam you

Tutorial 10 Website

Harvesting e-mail addresses




```
<tr>
   <td>Catherine Adler<br />Library Director</td>
   <td>555-3100</td>
   <td><a href="mailto:cadler@mpl.gov">cadler@mpl.gov</a>
   </td>
</tr>
<tr>
   <td>Michael Li<br />Head of Adult Services</td>
   <td>555-3145</td>
   <td><a href="mailto:mikeli@mpl.gov">mikeli@mpl.gov</a>
   </td>
</tr>
<tr>
   <td>Kate Howard<br />Head of Technical Services</td>
   <td>555-4389</td>
   <td><a href="mailto:khoward@mpl.gov">khoward@mpl.gov</a>
   </td>
</tr>
```

Tutorial 10 Website

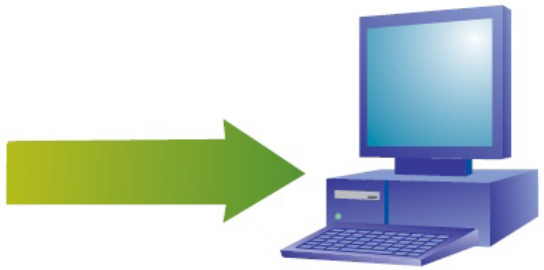
- Scrambling the Email addresses:
  - Kate decides not to place the addresses directly into the HTML file
  - Instead, she wants them scrambled up within the actual Web page code (HTML file)
  - But when the browser loads and renders the page, the email addresses should unscramble
    - This mechanism will thwart most email harvesters, although it is not 100% effective
    - But it's still a TON better than simply leaving your email directly in the HTML code

Tutorial 10 Website

Scrambling e-mail addresses

```
<script type="text/javascript">
   showEM("reldac","vog.lpm");
</script>
```

e-mail address scrambled with JavaScript, keeping it from appearing in the page code



the browser runs a JavaScript program to unscramble the e-mail address …

… that the end user can view

Tutorial 10 Website

- Scrambling the Email addresses:
  - Neither HTML or XHTML has features that allow for the scrambling (or unscrambling) of addresses
  - You must write a program to do this for you
  - Now entering the scene…

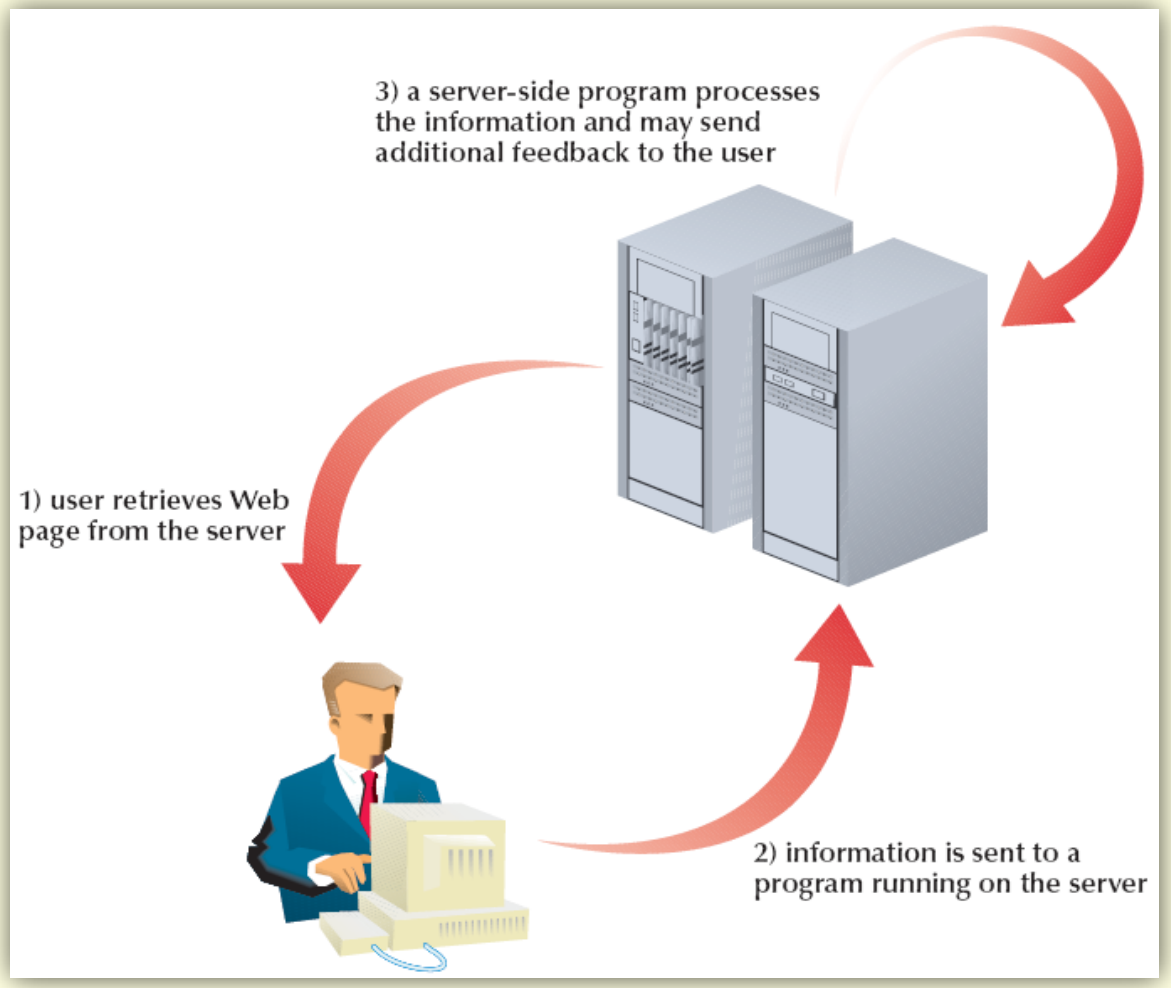
- JavaScript

Introducing JavaScript

- Server-Side and Client-Side Programming
  - Programming on the Web comes in two types:
  1) Server-side programming
     - Programs are placed on the server that hosts a Web site
     - Advantages are that these programs have access to databases that run on the server and are not accessible to end users
       - Online banking, credit card transactions, discussion groups
     - Disadvantages are that these programs use Web server resources
       - If the system is overloaded, the end user may sit waiting through long delays

Introducing JavaScript

Server-Side Programming

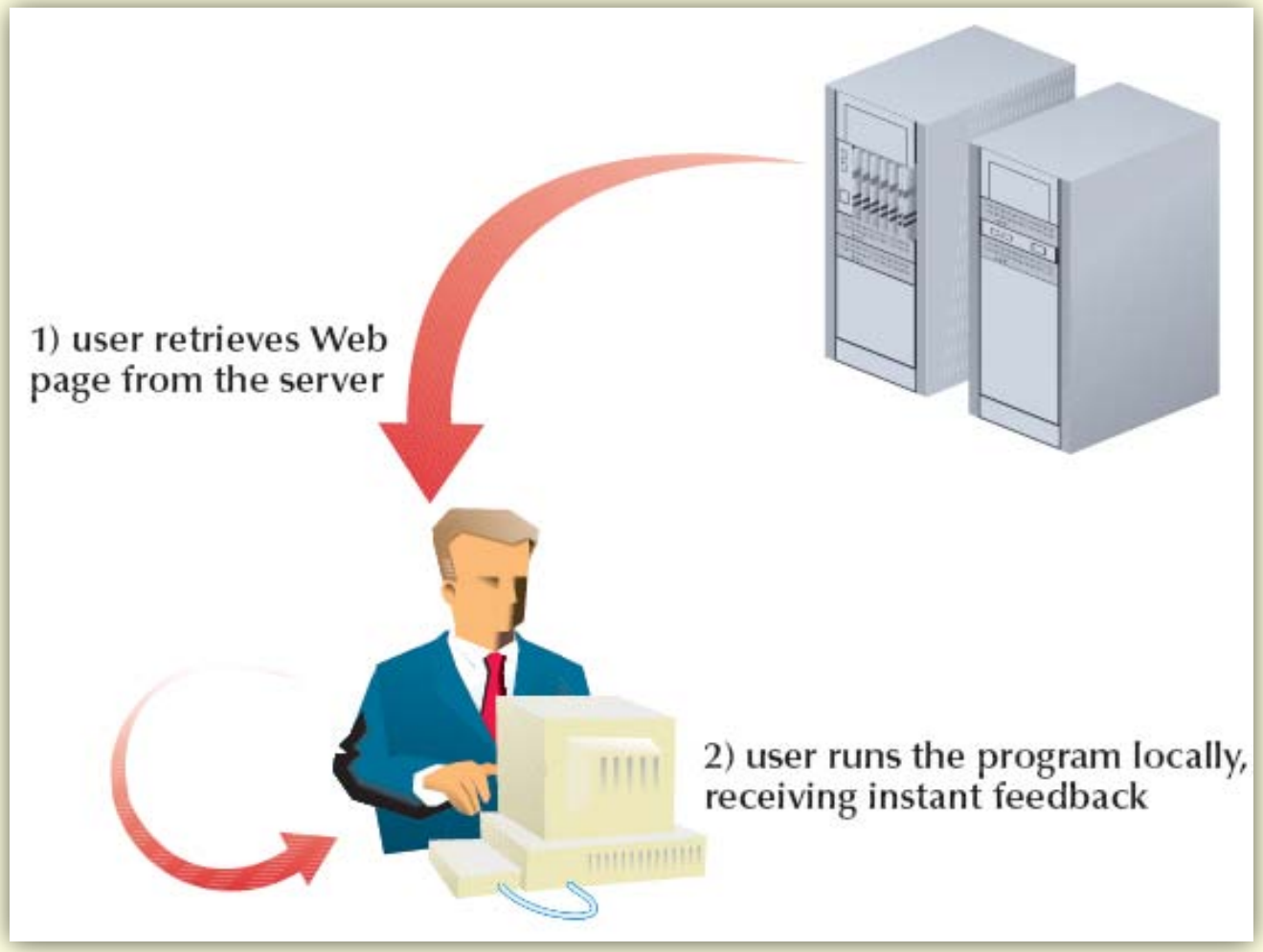


Introducing JavaScript

- Server-Side and Client-Side Programming
  - Programming on the Web comes in two types:
  2) Client-side programming
    - Programs are executed on each user's computer
      - computing is thereby distributed so that the server is not overloaded
    - Advantages are that the programs are often more responsive since they run locally on the user's machine
    - Disadvantages are that you lose access to the databases on the central server
  - Often a combination of server-side and client-side programming works best

Introducing JavaScript

Client-Side Programming



Introducing JavaScript

- The Development of JavaScript
  - Several programming languages can run on the client side
  - One such language is JavaScript
    - JavaScript is a subset of Java
      - So it's similar to Java, yet different in many important ways
    - Java itself is a compiled language
      - meaning, the code must be submitted to a compiler that translates it into executable code
      - For Java, this compiled code is called the Java applet
      - So for Java, you need both the compiler and an application or operating system that can run the compiled code

Introducing JavaScript

- The Development of JavaScript
  - Several programming languages can run on the client side
  - One such language is JavaScript
    - JavaScript is an **interpreted language**
      - meaning, the program code is executed directly without compiling it
    - You only need two things to use JavaScript:
      1) a text editor to write the JavaScript commands
      2) a Web browser to run the commands and display the results
    - This means that JavaScript can be inserted directly into an HTML or XHTML file or placed in a separate text file that is linked to the Web page

Introducing JavaScript

- Comparing Java and JavaScript

Java	JavaScript
A compiled language	An interpreted language
Requires the JDK (Java Development Kit) to create the applet	Requires a text editor
Requires a Java virtual machine or interpreter to run the applet	Requires a browser that can interpret JavaScript code
Applet files are distinct from the HTML and XHTML code	JavaScript programs are integrated with HTML and XHTML code
Source code is hidden from the user	Source code is accessible to the user
Powerful, requiring programming knowledge and experience	Simpler, requiring less programming knowledge and experience
Secure; programs cannot write content to the hard disk	Secure; programs cannot write content to the hard disk; however, there are more security holes than in Java
Programs run on the client side	Programs run on the client side

Introducing JavaScript

- The Development of JavaScript
  - IE supports a different version of JavaScript called Jscript
    - Very similar to JavaScript, although some JavaScript commands are not supported in Jscript, and vice versa
  - The European Computer Manufacturers Association (ECMA) develops scripting standards
    - The current standard is called ECMAScript
      - but browsers still generally call it JavaScript

Introducing JavaScript

- External vs Direct Placement
  - JavaScript programs can be placed directly into an HTML file
  - or they can be saved in an external file
  - There are advantages and disadvantages of both methods
    - Just like those of embedded vs external CSS style sheets
  - For this tutorial, we examine both methods and begin by looking at JavaScript code entered directly into the HTML file

Introducing JavaScript

- Working with the Script Element
  - Scripts are entered using the script element
  - Syntax:

```
<script type="mime-type">
 script commands
</script>
```

    - where *mime-type* defines the language in which the script is written and *script commands* are commands written in the scripting language
  - The MIME type for JavaScript is text/javascript

Introducing JavaScript

- Placing the Script Element
  - When a browser sees the script element, it treats any lines within the element as commands to run
    - These commands are processed in the order in which they appear within the HTML file
    - There are no limit to the number of commands
    - and no limit to the number of script elements in a page
  - Scripts can be placed in the head or body section
    - When placed in the body section, a browser interprets and runs them as it loads the different elements of the Web page

Introducing JavaScript

- Writing a JavaScript Statement
  - Every JavaScript program contains a series of statements (commands)
    - Each statement is a single line that indicates an action
    - Each statement should end in a semicolon
  - Syntax:

    ```
    JavaScript statement;
    ```

    - where *JavaScript statement* is the code the browser runs
  - The semicolon is the official way of notifying the browser that the statement is complete

Introducing JavaScript

- JavaScript and XML Parsers
  - JavaScript within an XHTML file can cause issues
    - XHTML parsers attempt to process symbols in the JavaScript code
  - Solution:
    - Place the code within a CDATA section
    - See page 571 for more info
  - Problem:
    - CDATA section is not well supported by browsers
  - Solution:
    - Use external JavaScript files (coming shortly…)

Tutorial 10 Website

- A simple program:
  - We experiment with JavaScript by using it to write a naïve program that simply writes the text of the email address into the Web page
    - We'll build on this simple program during the rest of the tutorial
  - Here's how we add our "program" to the page:

```
<tr>
   <td>Catherine Adler<br />Library Director</td>
   <td>555-3100</td>
   <td>
      <script type="text/javascript">
         document.write("cadler@mpl.gov");
      </script>
   </td>
</tr>
```

Tutorial 10 Website – Result

Staff Directory

Name	Phone	E-Mail
Catherine Adler Library Director	555-3100	cadler@mpl.gov
Michael Li Head of Adult Services	555-3145	
Kate Howard Head of Technical Services	555-4389	
Robert Hope Head of Children's Services	555-7811	
Wayne Lewis Circulation Services Supervisor	555-9001	
Bill Forth Interlibrary Loan	555-9391	

Tutorial 10 Website

- A simple program:
  - The document.write() statement tells the browser to send the text string “cadler@mpl.gov” to the Web page document
  - Note:
    - The placement of the script element tells the browser where to place this string
    - Because the script is placed between the opening and closing <td> tags, the text generated by the script is placed there as well

Writing Output to the Web Page

- The document.write() Method
  - This “method” is one of the basic ways that JavaScript uses to send output to a Web document
  - Why is it called a method?
    - In JavaScript, many commands involve working with objects in the Web page
  - An **object** is any item—from the browser window itself to a document displayed in the browser to an element displayed within the document
    - Even the mouse pointer, the window scrollbars, or the browser application itself can be treated as an object

Writing Output to the Web Page

- The document.write() Method
  - A **method** is a process by which JavaScript manipulates or acts upon the properties of an object
  - In this case, we used the write() method to write new text into the document object

Writing Output to the Web Page

- The document.write() Method
  - General Syntax:

    `document.write("text");`

    - where *text* is a string of characters that you want written to the Web document
  - The text string can also include HTML tags
  - Example:

    `document.write("<h1>blah</h1>");`

    - This text (blah) and the markup tags are placed into the document and rendered by the browser
      - just as if it had been entered directly into the HTML file

Tutorial 10 Website

- Kate wants the email to appear as a hyperlink
  - So we have to use <a> tags
    - cadler@mpl.gov
  - Note:
    - This text string requires the use of quotation marks for the href value (a rule of HTML and required by XHTML)
    - Also, text strings created by the document.write() method must be enclosed in quotes as well
      - So we'll have two sets of quotes
        - We have to place one set within the other
      - We end up using both double quotes and single quotes

Tutorial 10 Website

- Kate wants the email to appear as a hyperlink
  - The following JavaScript code encloses the href attribute value in single quotes and uses double quotes to mark the entire text to be written to the Web page document

```
document.write("<a href='mailto:cadler@mpl.gov'>");
document.write("cadler@mpl.gov");
document.write("</a>");
```

  - Note that the code is placed into three commands
    - We could have used one long text string, but that may be difficult to read
    - A browser treats these 3 commands as one long string

Tutorial 10 Website

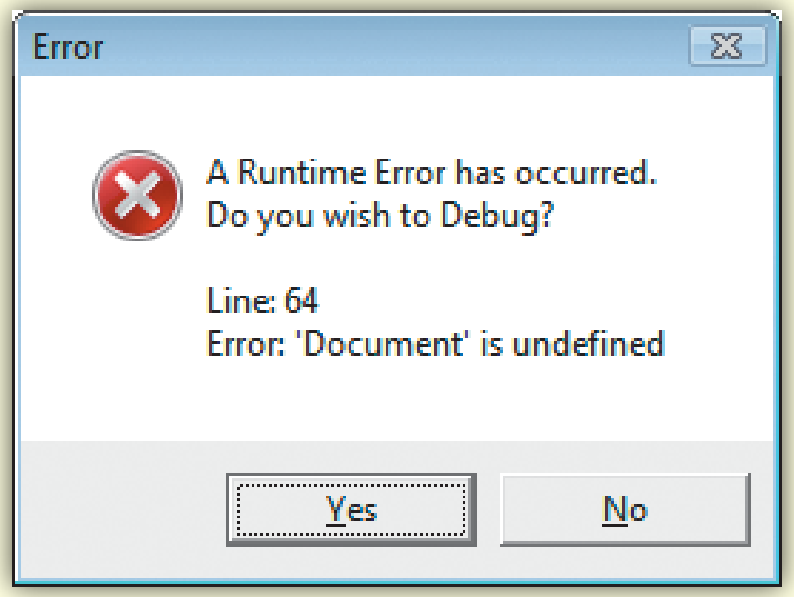
- Adding the email link for Catherline Adler
  - So now we simply add the two other document.write() commands
    - Thereby making the email as a link

```
<tr>
   <td>Catherine Adler<br />Library Director</td>
   <td>555-3100</td>
   <td>
      <script type="text/javascript">
         document.write("<a href='mailto:cadler@mpl.gov'>");
         document.write("cadler@mpl.gov");
         document.write("</a>");
      </script>
   </td>
</tr>
```

Tutorial 10 Website – Result

Understanding JavaScript Syntax

- There are several rules for JavaScript syntax
  - JavaScript is case sensitive
    - Document.write("");
    - is not recognized by the browser and results in an error



Understanding JavaScript Syntax

- There are several rules for JavaScript syntax
  - Ignores most occurrences of extra white space
    - You can indent your code as shown on the last example
  - However, you cannot place line breaks within a single statement
    - Example:

```
document.write("<a href='mailto:cadler@mpl.gov'>
cadler@mpl.gov
</a>");
```

    - This would not be allowed
      - There are ways to break lines (see page 576), but it is not good practice to do so (accidental errors introduced)

Browser Support

Supporting Non-JavaScript Browsers

For browsers that don't support scripts or that have their support for client-side scripts disabled, you can specify alternative content using the noscript element. The syntax of the noscript element is

```
<noscript>
   alternative content
</noscript>
```

where *alternative content* is the content a browser should display in place of accessing and running the script. For example, the following code displays a text message indicating that the page requires the use of JavaScript:

```
<script type="text/javascript">
  JavaScript statements
</script>
<noscript>
   <p>This page requires JavaScript. Please turn on JavaScript
      if your browser supports it and reload the page.</p>
</noscript>
```

Browsers that support client-side scripts and have that support enabled ignore the content of the noscript element.

Working with Variables

- The document.write() method becomes powerful when used with variables
  - A variable is a named item in a program that stores information
    - Most JavaScript programs use variables to represent values and text strings
  - Variables can store information created in one part of the program and use that info elsewhere
  - Variable values can also change as the program runs, enabling the program to display different values under varying conditions

Working with Variables

- Declaring a JavaScript Variable
  - Declaring a variable tells the JavaScript interpreter to reserve memory space for a variable
  - Syntax:

    ```
    var variable;
    ```

    - where *variable* is the name assigned to the variable
  - Example:
    - The following statement creates a variable called emLink (short for email link):

      ```
      var emLink
      ```

    - You can declare multiple variables by using commas

      ```
      var emLink, userName, emServer;
      ```

Working with Variables

- Declaring a JavaScript Variable
  - Limits on variable names:
    - The first character must be a letter of an underscore (_)
    - The remaining characters can be letters, numbers or additional underscore characters
    - Variable names cannot contain spaces
    - Finally, you cannot use words that JavaScript has reserved for other purposes
    - Example:
      - You cannot give a variable the name of “document.write”
    - Variable names are case sensitive
      - emLink and emlink are two different variables
    - Common mistake is forgetting this fact!

Working with Variables

- Assigning a Value to a Variable
  - Once a variable is declared, we can give it a value
  - Syntax:

    ```
    variable = value;
    ```

    - where variable is the variable name and value is the value assigned to the variable
  - Example:
    - The following statement stores the string cadler into the variable named userName:

      ```
      userName = “cadler”;
      ```
  - You can combine several declaratios:

    ```
    var userName = “cadler”, emServer = “mpl.gov”;
    ```

Working with Variables

- Assigning a Value to a Variable
  - Note:
    - Declaring variables with the var keyword is not required
    - The first time you use a variable, JavaScript creates the variable in computer memory
    - The following works just as well:

      ```
      userName = “cadler”;
      ```

    - Although not required, it is good programming practice to use the var command to make variables obvious
  - Suggestion:
    - To make your code easier to read, place all variable declarations at the beginning of your program
      - also good programming practice

Tutorial 10 Website

- Adding two JavaScript Variables:
  - We add two variables to the page
    - userName and emServer
  - Shortly, we'll modify the page further, showing how we use those variables in the script

```
<tr>
   <td>Catherine Adler<br />Library Director</td>
   <td>555-3100</td>
   <td>
      <script type="text/javascript">
         var userName = "cadler";
         var emServer = "mpl.gov";

         document.write("<a href='mailto:cadler@mpl.gov'>");
         document.write("cadler@mpl.gov");
         document.write("</a>");
      </script>
   </td>
</tr>
```

Working with Variables and Data

- Working with Data Types
  - JavaScript variable can store different types of information, which is referred to as its **data type**
  - The following data types are supported:
    - Numeric variables
    - Text/String variables
    - Boolean variables
    - Null variables

Working with Variables and Data

- Working with Data Types
  - Numeric variables
    - any number, such as 13, 22.5, or -3.14159
    - numbers can also be expressed in scientific notation
    - numeric values are specified without quotation marks

      ```
      var year = 2007;
      ```
  - Text/String variables
    - Any group of characters such as “Hello” or “whats up”
    - Strings must be enclosed in double or single quotation marks, but not both
      - **`‘Hello’`** is acceptable
      - **`“Hello’`** is not

Working with Variables and Data

- Working with Data Types
  - Boolean variables
    - Indicates the truth or falsity of a statement
    - Only two possible values: true or false
    - Used in programs that act differently based on different conditions
  - Null variables
    - Indicates that no value has yet been assigned to the variable
    - You can assign the keyword null explicitly:

      ```
      var emLink = null;
      ```

    - or just declare the variable and do not give it a value

Working with Variables and Data

- Working with Data Types
  - The data type is determined by the context in which the variable is used
    - This means that a variable can change from one data type to another within a single program
    - Example:
      - Month = 5;
      - Month = “March”;
      - the variable Month started out as a numeric variable then becomes a string variable
    - This type of language is called a weakly typed language
      - Since the variables are not strictly tied to a specific data type

Working with Variables and Data

- Working with Data Types
  - Weakly Typed Languages can lead to unpredictable results
    - Examples:

      `var total = 5 + 4;`

      - 9 gets stored into the variable total
    - The + symbol concatenates strings:

      `var emLink = “cadler” + “@” + “mpl.gov”;`

      - stores the text string cadler@mpl.gov into the emLink var
    - What about this?
      - x = 5;
      - y = “4”;
      - z = x + y;

The string “54” gets stored into the z variable since the y variable stores “4” as a string.

Tutorial 10 Website

- Creating the emLink variable
  - Now we can see the + symbol in action
    - We make a variable called emLink
    - We then save the email into it

```
<script type="text/javascript">
   var userName = "cadler";
   var emServer = "mpl.gov";
   var emLink = userName + "@" + emServer;

   document.write("<a href='mailto:cadler@mpl.gov'>");
   document.write("cadler@mpl.gov");
   document.write("</a>");
</script>
```

    - Note: we aren't printing the emLink variable just yet
      - that is coming up…

Working with Variables and Data

- Working with Data Types
  - Once you've created a variable, you can use it in place of the value it contains
  - Example:

```
var libName = "Monroe Public Library";
document.write(libName);
```

  - You can also use the + symbol to combine a variable with a text string

```
var libName = "Monroe Library";
document.write("<p>Welcome to the " + libName + "</p>");
```

Tutorial 10 Website

- Using the emLink variable
  - Now that we have this emLink variable
  - We can print it instead of the actual email address

```
<script type="text/javascript">
   var userName = "cadler";
   var emServer = "mpl.gov";
   var emLink = userName + "@" + emServer;

   document.write("<a href='mailto:" + emLink + "'>");
   document.write(emLink);
   document.write("</a>");
</script>
```

    - Since the email address is stored within emLink, that address now printed via the emLink variable

Brief Interlude: FAIL Pics

Daily UCF Bike Fail

Creating a JavaScript Function

- What is a function and what is its purpose?
  - Thus far, we focused only on one email address
  - However, there are five other staff members for the Library
  - If we wanted to use JavaScript to write the emails for all members, we'd have to repeat the same code five separate times
  - Thankfully, JavaScript provides a simpler way of doing just this
    - Functions

Creating a JavaScript Function

- What is a function and what is its purpose?
  - When you want to reuse the same set of JavaScript commands throughout your Web page, you can store the commands in a **function**
  - A function is a collection of commands that performs an action or returns a value
  - Every function is identified by a function name and a set of commands the function runs
  - Some functions also require parameters
    - variables associated with the function

Creating a JavaScript Function

- What is a function and what is its purpose?
  - General Syntax of a JavaScript function:

```
function function_name(parameters){
      JavaScript commands
}
```

    - where *function_name* is the name of the function, *parameters* is a comma-separated list of variables used in the function, and *JavaScript* commands are the statements run by the function
    - function names are case sensitive
    - A function name must begin with a letter or underscore and cannot contain any spaces

Creating a JavaScript Function

- Example function named showMsg()
  - Simply writes a paragraph to a Web document

```
function showMsg() {
        document.write("<p>Welcome to the Monroe Library</p>");
}
```

    - There are no parameters to this function
    - If the name of the library was stored in a function parameter named libName, the showMsg() function would look as follows:

```
function showMsg(libName) {
        document.write("<p>Welcome to the" + libName +"</p>");
}
```

Tutorial 10 Website

- Adding a function to show emails
  - We add a JavaScript function in the header
    - This will allow us to reuse this function for each person in the staff

```
    <title>Monroe Public Library</title>
    <link href="mplstyles.css" rel="stylesheet" type="text/css" />

    <script type="text/javascript">
       function showEM(userName, emServer) {
          var emLink = userName + "@" + emServer;
          document.write("<a href='mailto:" + emLink + "'>");
          document.write(emLink);
          document.write("</a>");
       }
    </script>
</head>
```

    - Note that the variables userName and emServer are passed to this function as parameters

Creating a JavaScript Function

- Calling a Function
  - When a browser encounters a function, it bypasses it without executing any code
    - The function is only executed when it is “called” by another JavaScript command
  - Syntax for calling a JavaScript funciton:

    `function_name(parameter values)`

    - where *function_name* is the name of the function and *parameter values* is a comma separated list of values that match the parameters of the function

Creating a JavaScript Function

- Calling a Function
  - Example:
    - To call the showMsg() function using the text string "Monroe Public Library" as the value of the libName parameter, you run the following command:

      ```
      showMsg("Monroe Public Library");
      ```

    - The command would call the showMsg() function and the following would output to the HTML document:

      ```
      <p>Welcome to the Monroe Public Library</p>
      ```

Creating a JavaScript Function

- Calling a Function
  - Parameter values can also be variables
  - The following commands store the library name as a variable called libText and then calls the showMsg() function using that variable

```
var libText = "Cutler Public Library";
showMsg(libText);
```

  - Result printed to the HTML document:

```
<p>Welcome to the Cutler Public Library</p>
```

Tutorial 10 Website

- Calling the showEM() function:
  - Within the actual Web table, we write one JavaScript command that calls our function
    - Remember: the function was written in the header

```
<tr>
   <td>Catherine Adler<br />Library Director</td>
   <td>555-3100</td>
   <td>

      <script type="text/javascript">
         showEM("cadler","mpl.gov");
      </script>

   </td>
</tr>
```

    - Note: the result is the same as earlier, but now we can reuse this showEM() function for other staff addresses

Tutorial 10 Website

Adding other addresses:

```
<tr>
   <td>Michael Li<br />Head of Adult Services</td>
   <td>555-3145</td>
   <td>
      <script type="text/javascript">
         showEM("mikeli","mpl.gov");
      </script>
   </td>
</tr>
<tr>
   <td>Kate Howard<br />Head of Technical Services</td>
   <td>555-4389</td>
   <td>
      <script type="text/javascript">
         showEM("khoward","mpl.gov");
      </script>
   </td>
</tr>
<tr>
   <td>Robert Hope<br />Head of Children's Services</td>
   <td>555-7811</td>
   <td>
      <script type="text/javascript">
         showEM("rhope","mpl.gov");
      </script>
   </td>
</tr>
<tr>
   <td>Wayne Lewis<br />Circulation Services Supervisor</td>
   <td>555-9001</td>
   <td>
      <script type="text/javascript">
         showEM("wlewis","mpl.gov");
      </script>
   </td>
</tr>
<tr>
   <td>Bill Forth<br />Interlibrary Loan</td>
   <td>555-9391</td>
   <td>
      <script type="text/javascript">
         showEM("bforth","mpl.gov");
      </script>
   </td>
</tr>
```

Tutorial 10 Website – Result

Name	Phone	E-Mail
Catherine Adler Library Director	555-3100	cadler@mpl.gov
Michael Li Head of Adult Services	555-3145	mikeli@mpl.gov
Kate Howard Head of Technical Services	555-4389	khoward@mpl.gov
Robert Hope Head of Children's Services	555-7811	rhope@mpl.gov
Wayne Lewis Circulation Services Supervisor	555-9001	wlewis@mpl.gov
Bill Forth Interlibrary Loan	555-9391	bforth@mpl.gov

Creating a Function to Return a Value

- Functions can return values
  - The other use of functions is to return a calculated value
    - To do so, the function must include a **return statement**
  - Syntax:

```
function function_name(parameters){
      JavaScript commands
      return value;
}
```

    - where *value* is the calculated value that is returned by the function

Creating a Function to Return a Value

- Functions can return values
  - Example:
    - Following function calculates the area of a rectangle by multiplying the region's length and width:

```
function CalcArea(length, width) {
      var area = length*width;
      return area;
}
```

    - In this function, the value of the area variable is returned by the function and sent back to where the function was called from
    - So you have to create a variable receive this value

Creating a Function to Return a Value

- Functions can return values
  - Example:
    - Following code uses the function to calculate the area of a rectangle whose dimensions are 8 x 6 units:

```
var x = 8;
var y = 6;
var z = CalcArea(x,y);
```

    - The first two commands assign values to the x and y variables, respectively
    - The third command calls the CalcArea function
      - It sends the values in the x and y variables as parameters to the function, and it receives the result, saving it into **z**

Tutorial 10 Website

- We are on the right track
  - We're using JavaScript to write the emails
  - But the text of the username and mail server can still be read from the document
  - Ideally, we want to somehow hide that text
  - Solution:
    - Within the HTML file, we will type the usernames and mail server **backwards**
    - Then we will call a function that will reverse this string
    - So mail harvesters will find a jumbled string (reversed)
    - But then it will appear normal in the Web browser

Accessing External JavaScript Files

- External JavaScript files
  - Thus far, we've only looked at JavaScript placed within the HTML document
  - Another, arguably better alternative, is to place the JavaScript code in an external file
  - The code to access an external script file is:

```
<script src="url" type="mime-type"></script>
```

    - where *url* is the URL of the external document and *mime-type* is the language of the code in the file
  - To access code in the spam.js file, add:

```
<script type="text/javascript" src="spam.js"></script>
```

Tutorial 10 Website

- Function to reverse strings
  - We use the following function to reverse strings:

```
function stringReverse(textString) {
      if (!textString)
            return '';
      var revString='';
      for (i = textString.length-1; i>=0; i--)
            revString+=textString.charAt(i);
      return revString;
}
```

    - The exact details of this function are beyond the scope of this tutorial (and course)
    - The beauty is that this is OKAY. We can still call and use this function for our needs!!!

Tutorial 10 Website

- Function to reverse strings
  - Using the stringReverse function:
  - If we call the function as follows:

```
userName = stringReverse("reldac");
emServer = stringReverse("vog.lpm");
```

    - the userName variable would have the value cadler
    - the emServer variable would have the value mpl.gov

  - So our strings are getting reversed!

Tutorial 10 Website

- Function to reverse strings
  - Using the stringReverse function:
  - If we call the function as follows:

```
userName = stringReverse("reldac");
emServer = stringReverse("vog.lpm");
```

    - the userName variable would have the value cadler
    - the emServer variable would have the value mpl.gov

  - So our strings are getting reversed!

Accessing External JavaScript Files

Using an external script file

```
function stringReverse(textString) {
   if (!textString) return '';
   var revString='';
   for (i = textString.length-1; i>=0; i--)
      revString+=textString.charAt(i)
   return revString;
}
```

```
<title>Monroe Public Library</title>
<link href="mplstyles.css" rel="stylesheet" type="text/css" />
<script src="spam.js" type="text/javascript"></script>
<script type="text/javascript">
   function showEM(userName, emServer) {
      var emLink = userName + "@" + emServer;
      document.write("<a href='mailto:" + emLink + "'>");
      document.write(emLink);
      document.write("</a>");
   }
</script>
</head>
```

Tutorial 10 Website

- Adding the external file to the header:
  - We first add the code to link to the external JavaScript file

```
    <title>Monroe Public Library</title>
    <link href="mplstyles.css" rel="stylesheet" type="text/css" />

    <script src="spam.js" type="text/javascript"></script>

    <script type="text/javascript">
       function showEM(userName, emServer) {
          var emLink = userName + "@" + emServer;
          document.write("<a href='mailto:" + emLink + "'>");
          document.write(emLink);
          document.write("</a>");
       }
    </script>
</head>
```

Tutorial 10 Website

- Testing the stringReverse() function:
  - Now we call the stringReverse function from our document

```
<script type="text/javascript">
   function showEM(userName, emServer) {

      userName = stringReverse(userName);
      emServer = stringReverse(emServer);

      var emLink = userName + "@" + emServer;
      document.write("<a href='mailto:" + emLink + "'>");
      document.write(emLink);
      document.write("</a>");
   }
</script>
```

Tutorial 10 Website – Result

Name	Phone	E-Mail
Catherine Adler Library Director	555-3100	reldac@vog.lpm
Michael Li Head of Adult Services	555-3145	ilekim@vog.lpm
Kate Howard Head of Technical Services	555-4389	drawohk@vog.lpm
Robert Hope Head of Children's Services	555-7811	epohr@vog.lpm
Wayne Lewis Circulation Services Supervisor	555-9001	siwelw@vog.lpm
Bill Forth Interlibrary Loan	555-9391	htrofb@vog.lpm

Tutorial 10 Website

- Not quite what we want!
  - The email addresses are shown, in the browser and, therefore, to the user, in backwards order
  - And the emails are still in correct order within the code, which isn't what we want
  - What we want:
    - The emails within the code should be reversed to thward email harvesters
    - Then the stringReverse() function will do its magic, reversing the emails for the browser (user)
    - So we need to enter the usernames and email servers in reverse order

Tutorial 10 Website

- Changing userName and emServer values:

```
<tr>
   <td>Catherine Adler<br />Library Director</td>
   <td>555-3100</td>
   <td>
      <script type="text/javascript">
         showEM("reldac","vog.lpm");
      </script>
   </td>
</tr>

<tr>
   <td>Michael Li<br />Head of Adult Services</td>
   <td>555-3145</td>
   <td>
      <script type="text/javascript">
         showEM("ilekim","vog.lpm");
      </script>
   </td>
</tr>
<tr>
   <td>Kate Howard<br />Head of Technical Services</td>
   <td>555-4389</td>
   <td>
      <script type="text/javascript">
         showEM("drawohk","vog.lpm");
      </script>
   </td>
</tr>
```

Tutorial 10 Website – Result

Commenting & Debugging

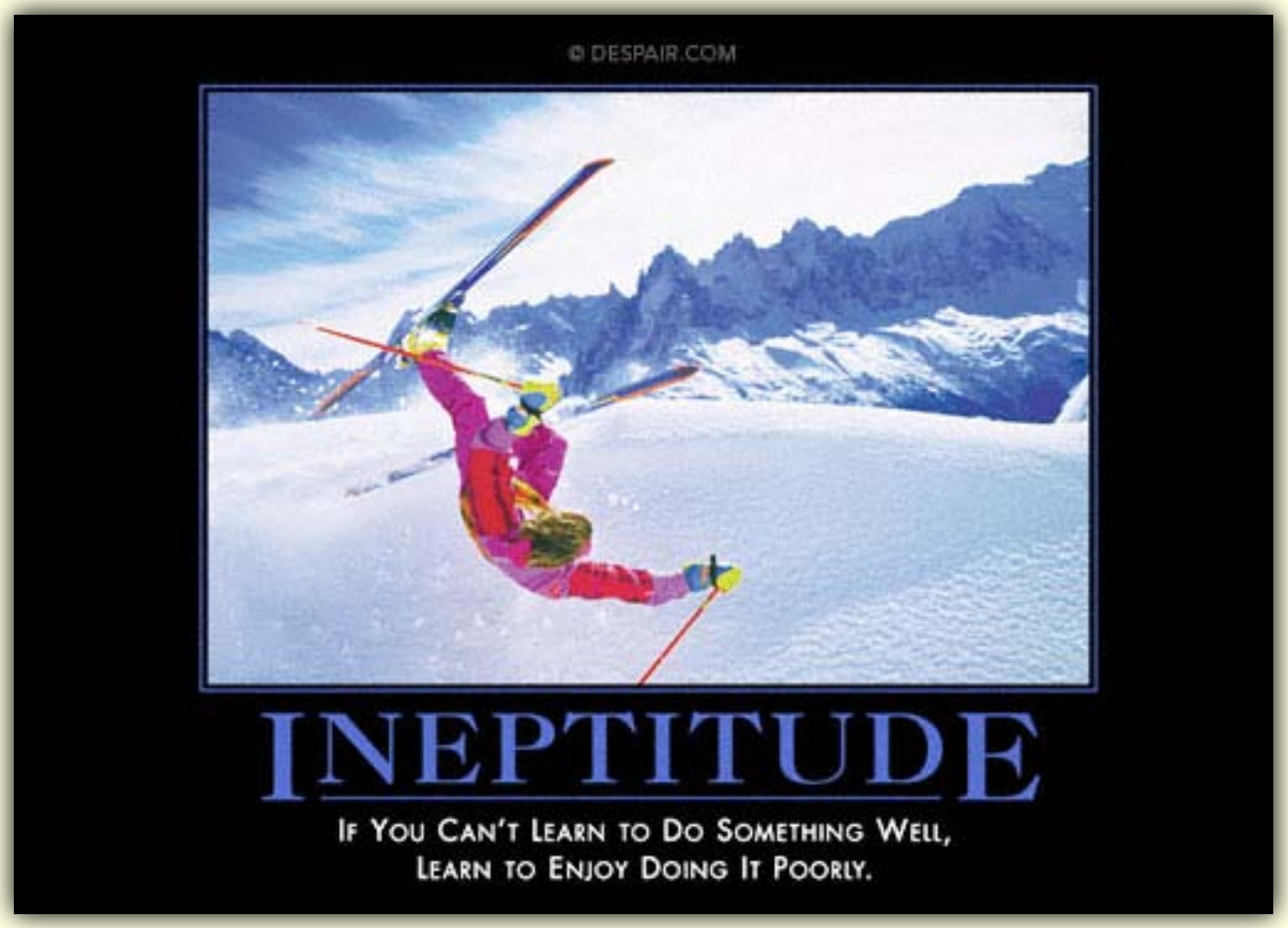
- The remainder of Tutorial 10
  - Discusses how to properly comment your code
  - And how to debug your code

  - See pages 594 – 603 for more information

Tutorial 10: Programming with JavaScript

WASN'T THAT PHENOMENAL!

Daily Demotivator



Tutorial 10: Programming with JavaScript

Computer Science Department
University of Central Florida

COP 3175 – Internet Applications