CSCI 2910
Client/Server-Side Programming

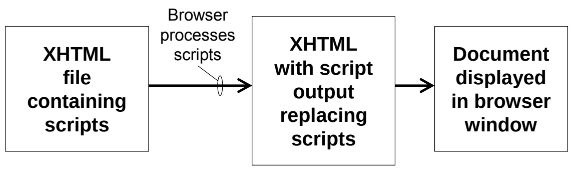
Topic: JavaScript Part 2

Today's Goals

Today's lecture will cover:
- More objects, properties, and methods of the DOM
- The Math object
- Introduction to form validation

Intermediate File vs. HTML Output

A sometimes difficult concept is that the output of a JavaScript script is not output to the browser window, but instead is output to the "intermediate" HTML file that the browser will interpret for display.


XHTML file containing scripts
Browser processes scripts
XHTML with script output replacing scripts
Document displayed in browser window

Intermediate File vs. HTML Output (continued)

- Since the JavaScript output is to be interpreted by a browser as HTML, the output must contain tags.
- Example – Assume we want a heading level 1 with a line break in the middle:
  - Wrong:
    ```
    document.write("This is my \n page
    title");
    ```
  - Right:
    ```
    document.write("<h1>This is my <br />
    page title</h1>");
    ```

write vs. writeln

- There are two document object methods used to write
  - `document.write(`*`string`*`)`
  - `document.writeln(`*`string`*`)`
- The only difference between the two is that writeln appends a carriage return/linefeed (\n) to the end of the string when printing to the intermediate file.

Prompting as Page Loads

- Remember that scripts within the body are executed as they are encountered
- You can take advantage of this by prompting the user for information as the page loads using a function such as window.prompt().

Prompting as Page Loads (continued)

```
<body>
  <script language= "JavaScript"
  type="text/JavaScript">
  <!--
     var head_color;
     head_color = window.prompt("What color
          would you like to display these
          headings in? (Enter web color)");
     document.writeln("<h1 style=\"color:" +
          head_color + "\">" + "My Title" +
          "</h1>");
  //-->
  </script>
 </body>
```

Double vs. Single Quotes

- As with any language that relies heavily on the use of output strings, we must have a way to identify quotation marks within a string without affecting the way the interpreter views the string.
- In JavaScript, there are three ways to embed quotation marks within a string:
  - use single quotes within a string identified using double quotes
  - use double quotes within a string identified using single quotes
  - use the JavaScript escape characters \' or \"

Double vs. Single Quotes (cont.)

Examples:

- `document.write("<a class='menu'>");`
- `document.write('<a class="menu">');`
- `document.write("<a class=\"menu\">");`

All three methods should work regardless of browser

Declaring Variables

- Variables are declared using the keyword ***var***
- Example:

```
var int_value, string_value
```

- When variables are declared, they are not assigned a default value, unless specified by the programmer
- All variables in JavaScript can contain a value of any data type, i.e., JavaScript does not rigorously follow types an will try to convert between types
- ***null*** is a valid variable value

Parsing Functions

- *parseInt(string, radix)* -- returns the first integer in the string. The radix argument specifies the base in which the number is represented in the string, e.g., 16 (hexadecimal), 10 (decimal), or 2 (binary).
- Example:

```
parseInt("313 Gilbreath", 10);
```

would return 313
- If the first character is not a number, then the function returns "NaN" indicating the value is not a number.

Parsing Functions (continued)

- *parseFloat(string)* – returns the first floating point number in the string.
- Example:

```
parseFloat("2.98% of students");
```

would return 2.98
- If the first character is not a number, then the function returns "NaN" indicating the value is not a number. This includes characters such as $ or #.

isNaN()

- isNaN(value) – returns a true or false based on whether value represents a number or not.
- "value" can be a string containing a number.
- Helpful with validation of forms.
- Examples:
  - isNaN("David Tarnoff") would return true
  - isNaN(4*5) would return false
  - isNaN("315") would return false

unescape()

- In some cases, strings are encountered that have certain characters replaced with escape characters.
- For example, a URL often replaces spaces with %20 and the '@' symbol with %40.
- unescape(encodedstring) – goes through a string replacing escape characters with original characters.

unescape() (continued)

For example:

```
document.write(
  unescape("My%20e-
  mail%20is%3A%20tarnoff%40etsu.edu%21"));
```

would output as:

```
My e-mail is: tarnoff@etsu.edu!
```

Math Object

- JavaScript provides this utility object for your use in scripting.
- The Math object isn't part of the DOM, i.e., it is not a conceptual component of a web page.
- The Math object is a stand alone object provided for use with mathematical operations

Math Object Properties

- *Math.E* – returns the base of natural logarithms, i.e., e ≈ 2.7183
- *Math.LN10* – returns the natural logarithm of 10, i.e., ln(10) ≈ 2.3026
- *Math.LN2* – returns the natural logarithm of 2, i.e., ln(2) ≈ 0.6931
- *Math.LOG10E* – returns the base 10 logarithm of e, i.e., $\log_{10}(e) \approx 0.4343$
- *Math.LOG2E* – returns the base 2 logarithm of e, i.e., $\log_2(e) \approx 1.4427$
- *Math.PI* – returns the ratio of the circuference of a circle to its diameter, i.e., pi ≈ 3.1416
- *Math.SQRT1_2* – returns the value of 1 divided by the square root of 2, i.e., 1/(√2) ≈ 0.7071
- *Math.SQRT2* – returns the square root of 2, i.e., √2 ≈ 1.4142

Math Object Methods

- *Math.abs(x)* – returns the absolute value of x
- *Math.acos(x)* – returns the arccosine of x as a numeric value between 0 and PI radians
- *Math.asin(x)* – returns the arcsine of x as a numeric value between -PI/2 and PI/2 radians
- *Math.atan(x)* – returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
- *Math.atan2(y, x)* – returns the arctangent of the quotient of its arguments
- *Math.ceil(x)* – returns the smallest integer greater than or equal to x
- *Math.cos(x)* – returns the cosine of x where x is in radians
- *Math.exp(x)* – returns the value of e^x where e is Euler's constant

Math Object Methods (continued)

- *Math.floor(x)* – returns the largest integer less than or equal to x
- *Math.log(x)* – returns the natural logarithm of x
- *Math.max(x, y)* – returns the greater of x and y
- *Math.min(x, y)* – returns the lesser of x and y
- *Math.pow(x, y)* – returns the value of x^y
- *Math.random()* – returns a pseudo-random number between 0 and 1
- *Math.round(x)* – rounds x to the nearest integer
- *Math.sin(x)* – returns the sine of x where x is in radians
- *Math.sqrt(x)* – returns the square root x
- *Math.tan(x)* – returns the tangent of x where x is in radians

Accessing Data from Forms

- Before we get to the point where we're trying to access data, let's talk a little about the form object and its properties and methods
- One way to "point" to a specific form object is to access the document object forms array.

  document.forms[n]
- The most reliable way to reference a form object is to consistently identify everything with the *name* and *id* attributes.

  document.*formname*
  document.forms["*formname*"]

Form Object Properties

- *action* – Returns the URL address to which the form's data will be submitted.
- *length* – Returns the number of elements in the form.
- *method* – Returns a string specifying data submission method, i.e., either 'get' or 'post'.
- *target* – Returns the target window where the form's response will appear.

Form Object Methods

- *reset( )* – Resets the form to its default values. (Same result as clicking the reset button.)
- *submit( )* – Submits the form's data. (Same result as clicking the submit button.)

Accessing Element Values

- One way to "point" to a specific element in a form is to access the element array under the form object.

  document.formname.elements[n]

  where n equals the position the element holds in the order that the elements were added to the form. Huh?
- The most reliable way to reference an element of a form is to consistently identify everything with the *name* and *id* attributes.

  document.*formname.elementname*
  document.forms["*formname*"].*elementname*

Form Element Object Properties

- *defaultValue* – sets or returns a string representing the default value of the element.
- *name* – sets or returns the element's name or id attribute.
- *type* – returns the element's type property.
- *value* – sets or returns the element's value attribute. Works differently for different elements.

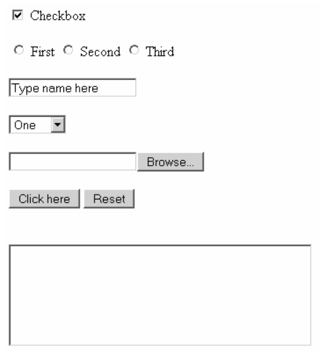
Form Element Object Methods

- *blur( )* – removes the focus from the specified element
- *click( )* – simulates a mouse-click for some elements
- *focus( )* – returns focus to the specified element

The Use of *value*

```
<form name="userinput" id="userinput">
<input type="checkbox" name="gen_check" id="gen_check" checked="checked" />
    Checkbox<br /><br />
<input type="radio" name="gen_radiobutton" id="gen_radiobutton" value="1" /> First
<input type="radio" name="gen_radiobutton" id="gen_radiobutton" value="2" /> Second
<input type="radio" name="gen_radiobutton" id="gen_radiobutton" value="3" /> Third
    <br /><br />
<input type="text" name="gen_text" id="gen_text" value="Type name here"><br /><br />
<select name="gen_select" id="gen_select" size="1">
   <option value="one">One</option>
   <option value="2">Two</option>
   <option value="3">Three</option>
   <option value="four">Four</option>
</select><br /><br />
<input type="file" name="gen_file" id="gen_file" size="20" /><br /><br />
<input type="button" onClick = "printVals()" name="gen_button" id="gen_button"
    value="Click here" />
<input type="reset" value="Reset" />
<br /><br /><br />
<textarea cols="40" rows="6" name="output" id="output" /></textarea>
</form>
```

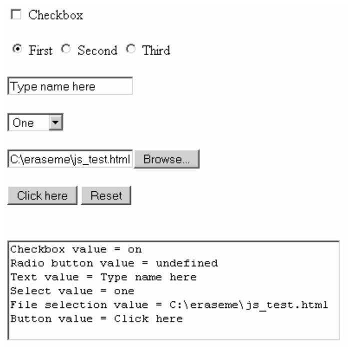
The Use of *value* (continued)



The Use of *value* (continued)

```
var output_string;
function printVals()
{
      output_string="Checkbox value = " +
                        document.userinput.gen_check.checked + "\n"
            + "Radio button value = " +
                        document.userinput.gen_radiobutton.value + "\n"
            + "Text value = " +
                        document.userinput.gen_text.value + "\n"
            + "Select value = " +
                        document.userinput.gen_select.value + "\n"
            + "File selection value = " +
                        document.userinput.gen_file.value + "\n"
            + "Button value = " +
                        document.userinput.gen_button.value + "\n";
      document.userinput.output.value=output_string;
}
```

The Use of *value* (continued)



```
Checkbox value = on
Radio button value = undefined
Text value = Type name here
Select value = one
File selection value = C:\eraseme\js_test.html
Button value = Click here
```

The Use of *value* (continued)

- Some of the values make sense, e.g., the text value equaled the text in the box.
- Some of the values did not make sense
  - Checkbox value always equals "on"
  - Radio buttons always equal "undefined"
  - Button value equals the text on the button

The Use of *value* (continued)

- Solutions
  - To read the checkbox values, use the property *checked* – returns "true" or "false"
  - Associate an *onClick* event for each radio button that modifies a variable
  - Associate an *onClick* event for the button to indicate when it is pressed.

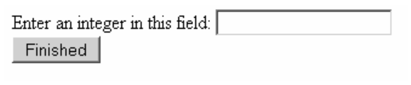
Form Validation

- A very common application of client-side scripts is for validating the data users have entered on a form.
- For example, we would like to make sure that the user has not done something like entered the word "dog" where the form asked for an age.
- The functions covered over the past two lectures will allow us to access form data and verify that it is correct.

Simple Number Verification

The form below creates a text box and a button.

```
<form name="sample" id="sample">
Enter an integer in this field:
<input type="text" size="20"
  name="justanumber" id="justanumber"
  onblur="integercheck()" /><br />
<input type="button" value="Finished" />
</form>
```



Simple Number Verification (continued)

```
<script language="JavaScript"
  type="text/javascript">
function integercheck()
{
  if (isNaN(document.sample.justanumber))
  {
     window.alert("This field requires an
  integer!");
     document.sample.justanumber.focus();
  }
}
</script>
```

Checking for '@' in E-mail

```
function emailcheck(email_string)
{
   if(email_string.indexOf("@")==-1)
      window.alert("Not a valid
           email address!");
}
```