INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

Module 3 (I-III): Client-Side Scripting (JavaScript)

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

- Objectives
  - Understand the role of scripting languages in dynamic documents
  - Learn how to write scripts using JavaScript
  - Learn the core constructs of the JavaScript language

- Outline
  - Introduction
  - Client-Side Scripting
  - Overview of the JavaScript Language
  - Some Built-in Objects
  - Functions and Event Handlers
  - Arrays

Introduction

- Traditionally web pages are static, i.e. never change unless the Web page itself is changed
  - Appropriate for pages where the content and styling seldom change and where the visitor is merely a passive reader of page content.
  - Not appropriate for dynamic pages where layout, styling, and content need to change in response to visitor actions and desires.
- Examples of dynamic effects
  - Put dynamic text into an HTML page
  - Change the visual appearance of a hyperlink to indicate user actions
  - Change a picture size or lightness when the user clicks on accompanying buttons
  - Display a block of text (e.g. revealing words definitions) when moving the mouse on top of the underlined terms being defined
  - Carry out processing tasks through interaction with the user (e.g. create a calculator, or games)
  - Display current date and time
  - Opening pages in customized windows (e.g. specify whether the browser's menu bar, status bar or whatever should be present)
  - Show contextual information in the status bar
  - Validating inputs to fields before submitting a form

Introduction (cont.)

- Dynamic HTML (or DHTML) is a collection of technologies to change static Web pages into dynamic Web pages that
  - React to events initiated by the user or by the Web page itself
    - So you can enhance page interactivity
  - Making decisions or repetitions
  - Dynamically changing elements and styles
  - Generate alters, documents, etc
- DHTML pages requires familiarity with four main topics
  - HTML/XHTML
  - CSS
  - The browser's Document Object Model (DOM)
    - the collection of HTML/XHTML elements appearing on a Web page
  - JavaScript
    - a scripting language that allows adding real programming to web pages

Client-Side Scripting

- A scripting language is a lightweight programming language
  - such as JavaScript, VBScript, Jscript, Perl, etc.
  - Allow making web pages more animated and more responsive to user interaction
  - Interpreted not compiled; thus the code is executed as the page is downloaded and rendered
  - The script code can be executed either on the server (server-side script) or on the client/browser (client-side script)

Client-Side Scripting (cont.)

- What is client-side script?
  - Code embedded in Web pages along with XHTML and CSS styles, downloaded from the Web server to the browser, and then executes locally on the client
  - Including code within a web page can lead to a number of features to enhance the appearance and functionality of Web pages
    - Without the need to send information to the Web Server
  - The ability to interpret and run JavaScript code is built into modern desktop browsers

Client-Side Scripting (cont.)

- Why client-side script?
  - Enhance the appearance and add cool effects, e.g. animation
    - With the DOM, it gives access to all the elements on a web page; so you can create, modify and remove elements in the page dynamically.
  - Create UI constructs not inherent in HTML (i.e., special formatting features that go beyond HTML)
    - Drop-down and pull-out menus
    - Tabbed dialogs
  - Data validation before going out to the server
  - Boost interactivity and reduce the response time
  - Better performance and scalability: less burden on the server and the Internet

The JavaScript Language

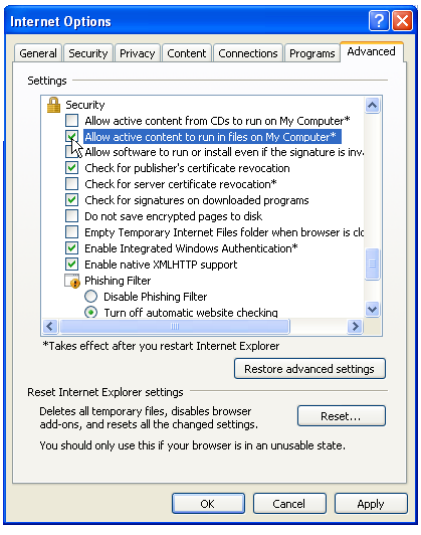
- Object-based scripting language released in the Fall of 1995
- Originally developed by Netscape and named LiveScript
- Later, Netscape & Sun Microsystems Collaboration renamed LiveScript to JavaScript
- Jscript is Microsoft's implementation of JavaScript
- JavaScript is completely different from Java except for some syntactical similarities
- Together with DOM, it works with the objects associated with a Web page document
  - the window
  - the document
  - the elements
    - such as forms, images, hyperlinks, etc

JavaScript vs. Java

JavaScript	Java
Object Oriented Programming (OOP) language created by people at Netscape	Object Oriented Programming (OOP) language created by James Gosling at Sun Microsystems
Contains a much smaller and simpler set of commands than does Java	Versatile and richer set of commands and features
Interpreted (not compiled) by client.	Compiled bytecodes downloaded from server, executed on client.
Code integrated with, and embedded in, HTML.	Applets distinct from HTML (accessed from HTML pages).
Cannot automatically write to hard disk.	Cannot automatically write to hard disk.
Variable data types not declared (dynamic typing).	Variable data types must be declared (static typing).
Object-oriented. No distinction between types of objects. Inheritance is through the prototype mechanism, and properties and methods can be added to any object dynamically.	Class-based. Objects are divided into classes and instances with all inheritance through the class hierarchy. Classes and instances cannot have properties or methods added dynamically.

Enabling the Browser

- Before you can run code examples with JavaScript on your computer, you may need to change your browser's security settings.
  - IE7 prevents scripts on the local computer from running by default
  - FF2 enables JavaScript by default
- Java-enabled browser is not automatically a JavaScript-enabled browser



Enabling JavaScript in Internet Explorer 7

Placement of JavaScripts

- Similar to CSS, there are three ways to use JavaScript:
  - Embedded: JavaScript can be placed in the `<head>` or in the `<body>` of an HTML document using the <style> element

```
<script type="text/javascript">
<!--
  //JavaSript Code Goes here
//  -->
</script>
```

    - JavaScript *functions* should be defined in the `<head>`
      - This ensures that the function is loaded before it is needed
    - JavaScript in the `<body>` will be executed as the page loads
    - Forgetting the ending `</script>` tag for a script may prevent the browser from interpreting the script properly and may prevent the document from loading properly
  - External: JavaScript can be put in a separate `.js` file

```
<script src="myJavaScriptFile.js"></script>
```

    - Put this HTML wherever you would put the actual JavaScript code
    - An external .js file lets you use the same JavaScript on multiple HTML pages
    - The external `.js` file cannot itself contain a `<script>` tag
  - Inline: Place JavaScript code as part of an event attached to an XHTML element

Example 1

```
<html>
<head>
    <title>First Example in Javascript</title>
</head>

<body>
    <script type="text/javascript">
          alert("Welcome to JavaScript!");
    </script>
</body>
</html>
```

Example 2

```
<html>
<head>
    <title>Another JavaScript Example </title>
</head>
<body>
Hello!!!<br>
<script>
document.write("Welcome to JavaScript!<br>");
</script>
Have fun...<br>
</body>
</html>
```

General Remarks

- Most JavaScript syntax and statements are borrowed from C and Java, so we won't spend much time on it
- JavaScript is case sensitive; all keywords are lowercase
- JavaScript uses C-style comments: // and /* */
- Some old browsers may not recognize script tags
  - They ignore the script tags but display the included JavaScript code
  - To get them ignore the whole thing, use html comment tag:

```
<script type="text/javascript">
    <!--
        document.write("Hello World!")
    //-->
    </script>
```

  - To get JavaScript to ignore the HTML close comment, - ->, the // starts a JavaScript comment, which extends to the end of the line

General Remarks (cont.)

- Every statement should end with a semicolon (although it is optional unless two or more statements appear on the same line
- The interpreter notifies the user of a syntax error when it attempts to execute the statement containing the error (at runtime)
  - Recall that syntax errors are violations of the rules of the programming language
  - Examples
    - Forgetting one of the delimiters of a multiline comment
    - Nesting multiline comments
    - Splitting a statement in the middle of a string (to avoid this, use + operator to concatenate strings)
    - Splitting a statement in the middle of an identifier
    - Having a space between symbols such as ==, !=, >= and <=
    - Reversing the operators !=, >= and <=
    - etc

Primitive data types

- JavaScript has three "primitive" data types: number, string, and boolean
  - Everything else is an object
- Numbers are always stored as floating-point values
  - Hexadecimal numbers begin with 0x
  - Some platforms treat 0123 as octal, others treat it as decimal
- Strings may be enclosed in either single quotes or double quotes
  - Strings can contains escape characters such as \n (newline), \" (double quote), \' (single quote), etc.
- Booleans are either true or false
  - 0, "0", empty strings, undefined, null, and NaN are false, other values are true

Variables

- Variables are declared with a `var` statement:
  - var pi = 3.1416, x, y, name = "Dr. ABC" ;
  - Variables names must begin with a letter or underscore
  - Variable names are case-sensitive
  - Variables are untyped (they can hold values of any type)
  - The word var is optional (but it's good style to use it)
  - Some programmers prefer to declare each variable on a separate line.
- Variables declared within a function are local to that function (accessible only within that function)
- Variables declared outside a function are global (accessible from anywhere on the page)

Operators

- Arithmetic operators:
  + - * / % ++ --
- Comparison operators:
  < <= == != >= >
- Logical operators:
  && || ! (&& and || are short-circuit operators)
- Bitwise operators:
  & | ^ (XOR) ~ (NOT) << >> (Shifts binary bits to right, discarding bits shifted off) >>> (Shifts binary bits to right, discarding bits shifted off and shifting in zeros from left.)
- Assignment operators:
  += -= *= /= %= <<= >>= >>>= &= ^= |=
- String operator:
  + (concatenation)

Operators (cont.)

- The conditional operator:
  `condition ? value_if_true : value_if_false`
- Special equality tests:
  - == and != try to convert their operands to the same type before performing the test
  - === and !== consider their operands unequal if they are of different types
  - Using x=3 and y="3":
    - x==y returns true
    - x===y returns false
- Additional operators:
  new typeof void delete

Operators (cont.)

- Precedence and associativity

Operators	Associativity	Type
() [] .	left to right	highest
++ -- !	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
&&	left to right	logical AND
||	left to right	logical OR
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Statements

- Assignment, e.g.
  - `greeting = "Hello, " + name;`
- Compound statement

```
{ statement;
   ...;
   statement;
}
```

- If statement

```
if (condition)
    statement;
```

- If –else statement

```
if (condition)
    statement;
else
    statement;
```


Statements (cont.)

- Loop statements

```
// while loop
while (condition)
    statement;

// do-while loop
do
    statement
while (condition);

// for loop
for (initialization; condition; increment)
    statement;
```

Statements (cont.)

- The switch statement:

```
switch (expression){
      case label :
          statement;
          break;
      case label :
          statement;
          break;
      ...
      default :
          statement;
  }
```

- Other familiar statements:
  - break;
  - continue;
  - The empty statement, as in ;; or { }

Exceptions

- Almost the same as in Java
- To create and throw an exception, use

```
throw expression;
```

  - Where expression is the value of the exception, and can be of any type (often, it's a literal String)
- To handle exceptions, use

```
try {
    statements to try
} catch (e) {
    exception-handling statements
} finally {      // optional, as usual
    code that is always executed
}
```

  - Notice that no type declaration for e; hence there is only one catch clause

Exceptions (cont.)

- To catch specific exceptions, use

```
try {
    statements to try
} catch (e if test1) {
    exception-handling for the case that test1 is true
} catch (e if test2) {
    exception-handling for when test1 is false and test2 is true
} catch (e) {
    exception-handling for when both test1 and test2 are false
} finally {        // optional, as usual
    code that is always executed
}
```

  - Typically, the test would be something like
    e == "InvalidNameException"

Referencing XHTML Elements

- An XHTML element must be assigned an **id** for a script to refer to it:
  <*tag* **id**="*idValue*"...>
  - The assigned *idValue* value must be unique and composed of alphanumerics excluding spaces
- Once an id is assigned, the XHTML object can be referenced in a script:
  document.getElementById("*idValue*")
- An alternate is the notation
  document.all.*idValue*,
- In some cases only the value itself is needed
  idValue

Getting and Setting Style Properties

- DHTML is created commonly by changing the style properties of XHTML elements

  -- Get a current style property:
  document.getElementById("*id*").**style.*property***

  -- Set a different style property:
  document.getElementById("*id*").**style.*property*** = *value*

- For example, given

  <h2 id="Head" style="color:blue">This is a Heading</h2>

- We can change the color property as

  document.getElementById("Head").style.color = "red"

JavaScript Functions

- Work just like subprograms in other languages
- The parentheses following the name of a method contain the arguments that the method requires to perform its task (or its action)

```
<script type="text/javascript">
function ChangeStyle() {
   document.getElementById("MyTag").style.fontSize = "14pt";
   document.getElementById("MyTag").style.fontWeight = "bold";
   document.getElementById("MyTag").style.color = "red";
}
</script>
```

- You can call the function in any statement or to handle user events, e.g.

  This is a paragraph that has its styling changed.

JavaScript Functions (cont.)

- Recursive functions
  - Example

```
// Recursive factorial function
function factorial( number ){
   if(number <=1)
      return 1;
   else
      return number * factorial (number -1);
}
```


Some Built-in Objects

- JavaScript includes built-in objects besides those associated with specific elements of a Web page, e.g.
  - Number
  - Boolean
  - Math
  - String
  - Date
  - Array

http://msconline.maconstate.edu/tutorials/JSDHTML/default.htm

The Number Object

- In JavaScript, all numbers are floating point
- Special predefined numbers:
  - Infinity, Number.POSITIVE_INFINITY
    - the result of dividing a positive number by zero
  - Number.NEGATIVE_INFINITY
    - the result of dividing a negative number by zero
  - NaN, Number.NaN (Not a Number)
    - the result of dividing 0/0
  - NaN is unequal to everything, even itself
    - There is a global isNaN() function
  - Number.MAX_VALUE
    - the largest representable number
  - Number.MIN_VALUE
    - the smallest (closest to zero) representable number

The Boolean Object

- The boolean values are true and false
- When converted to a boolean, the following values are also false:
  - 0
  - "0" and '0'
  - the empty string, '' or ""
  - undefined
  - null
  - NaN

The Math Object

- Can be accessed as Math.property, e.g.
  - x=Math.pow(3,2); // x=9
- Allows many common mathematical calculations including (all prefixed with Math as above):
  - Math.abs(x) : absolute value
  - Math.ceil(x) and Math.floor(x) : smallest integer not less than x and largest integer not greater than x
  - Math.cos(x), Math.exp(x), Math.log(x), Math.sin(x), Math.tan(x) : trigonometric and log rhythmic functions
  - Math.min(x,y) or Math.max(x,y) // returns the minimum or maximum of values x and y
  - Math.pow(x,y) //raises x to the power y
  - Math.round(x) // rounds to nearest integer
  - Math.sqrt(x) // returns square root

Strings and Characters

- In JavaScript, a string is a primitive type
- As mentioned strings are surrounded by either single quotes or double quotes
- There is no "character" type
- Special characters are:

`\0`	NUL	`\v`	vertical tab
`\b`	backspace	`\'`	single quote
`\f`	form feed	`\"`	double quote
`\n`	newline	`\\`	backslash
`\r`	carriage return	`\xDD`	Unicode hex *DD*
`\t`	horizontal tab	`\xDDDD`	Unicode hex *DDDD*

Some string methods

Method	Description
bold()	Changes the text in a string to bold.
italics()	Changes the text in a string to italic.
strike()	Changes the text in a string to strike-through characters.
sub()	Changes the text in a string to subscript.
sup()	Changes the text in a string to superscript.
toLowerCase()	Changes the text in a string to lower-case.
toUpperCase()	Changes the text in a string to upper-case.
fixed()	Changes the text in a string to fixed (monospace) font.
fontcolor ("color")	Changes the color of a string using color names or hexadecimal values.
fontsize("n")	Changes the size of a string using font sizes 1 (smallest) - 7 (largest).
link("href")	Formats a string as a link.

Some string methods (cont.)

Method	Description
charAt(index)	Returns the character at position index in the string.
charCodeAt(index)	Returns the Unicode or ASCII decimal value of the character at position index in the string.
indexOf("chars")	Returns the starting position of substring "chars" in the string. If "chars" does not appear in the string, then -1 is returned.
lastIndexOf("chars")	Returns the starting position of substring "char" in the string, counting from end of string. If "chars" does not appear in the string, then -1 is returned.
slice(index1[,index2])	Returns a substring starting at position index1 and ending at (but not including) position index2. If index2 is not supplied, the remainder of the string is returned.
split(delimiter)	Splits a string into separate substrings which are copied as individual elements into a new array object. The delimiter identifies the separator character for splitting the string but it is not included in the substrings. The array object does not need to be prior declared.
substr(index[,length])	Returns a substring starting at position index and including length characters. If no length is given, the remaining characters in the string are returned.
substring(index1,index2)	Returns a substring starting at position index1 and ending at (but not including) position index2.
toString()	Converts a value to a string.
toFixed(n)	Returns a string containing a number formatted to n decimal digits.
toPrecision(n)	Returns a string containing a number formatted to n total digits.

The Date Object

- Permits you to work with date and time settings taken from the system clock of the user's PC.
- By default creates an object with the computer's current date and time, e.g.

  now = new Date();
  - variable now contains current date and time
  - months are expressed 0-11; 0 being Jan., 11 being Dec.
- Dates are actually stored as an integer representing the number of milliseconds since January 1st, 1970
  - Negative values indicate dates before this date
- Once you have a date object you can set the date, or read the date in a number of useful formats
  - now.setFullYear(2003, 0, 31); /* Jan 31st, 2003 */
  - now.setHours(13, 13, 13); /* 1:13:13 PM, local time zone */

The Date Object (cont.)

Method	Description
`getDate()`	Returns the day of the month.
`getDay()`	Returns the numeric day of the week (Sunday = 0).
`getMonth()`	Returns the numeric month of the year (January = 0).
`getYear()` `getFullYear()`	Returns the current year.
`getTime()`	Returns the number of milliseconds since January 1, 1970.
`getHours()`	Returns the military hour of the day.
`getMinutes()`	Returns the minute of the hour.
`getSeconds()`	Returns the seconds of the minute.
`getMilliseconds()`	Returns the milliseconds of the second.

Basic Input and Output

- Programming languages need to start with some data and manipulate it
  - `confirm` asks a yes/no question in a dialog box
  - `prompt` prompts the user to type in some information into a text field inside the dialog box
- Sources of data can include:
  - Files
  - Databases
  - User (keyboard & mouse typically)
  - Variable assignments (ex: pi=3.14159)
  - Javascript objects, e.g. the date object
- Example

```
UsrName= prompt("What is your name?","Enter your name here");
```


Basic Input and Output (cont.)

- After a program manipulates the input data with various statements it usually creates an output of some kind
- Source of output may include:
  - Files
  - Database
  - Display or Printer
  - Devices (sound card, modems etc)
  - Javascript Objects
    - Via Object Methods

Event Handlers

- There are numerous page events and associated event handlers that need to be learned to create DHTML
- Mouse Events

onclick	The mouse button is clicked and released with the cursor positioned over a page element.
ondblclick	The mouse button is double-clicked with the cursor positioned over a page element.
onmousedown	The mouse button is pressed down with the cursor positioned over a page element.
onmousemove	The mouse cursor is moved across the screen.
onmouseout	The mouse cursor is moved off a page element.
onmouseover	The mouse cursor is moved on top of a page element.
onmouseup	The mouse button is released with the cursor positioned over a page element.

Example 1

```
<script type="text/javascript">
function Multiply() {
  var No1 = prompt("Enter the first number:", "");
  var No2 = prompt("Enter the second number:", "");
  var Product = No1 * No2;
   Str = No1 +" * "+ No2 + " = " ;
  alert(Str + Product.toFixed(2));
}
</script>
<input type="button" value="Get Number" onclick="Multiply()"/>
```

Example 2

```
<script type="text/javascript">
 function Subtract() {
   document.getElementById("Output").value =
    document.getElementById("FirstNo").value -
    document.getElementById("SecondNo").value;
 }
</script>
<input id="FirstNo" type="text" value="10" style="width:50px"/>
<input id="SecondNo" type="text" value="20" style="width:50px"/>
<input type="button" value=" = " onclick="Subtract()"/>
<input id="Output" type="text" style="width:50px"/>
```


Example 3

```
<script type="text/javascript">
  function TextSize() {
    var ReturnedValue = window.confirm("Larger text?");
    if (ReturnedValue == true) {
         document.body.style.fontSize = "12pt";
    } else {
         document.body.style.fontSize = "10pt";
    }
  }
</script>
<input type="button" value="Set Text" onclick="TextSize()">
```

Event Handlers

- Another way is to use inline scripts, i.e. put the script inside the event handler itself:

```
<p id="MyTag"
onclick="document.getElementById('MyTag').style.fontSize='14
pt'; document.getElementById('MyTag').style.fontWeight='bold';
document.getElementById('MyTag').style.color='red'"> This is a
paragraph that has its color changed.</p>
```

- Note
  - The <script> tag is not necessary in this case
  - Quoted values inside the script must be enclosed in single quotes (apostrophes) to alternate and differentiate the sets of quote marks.
  - Amount of use and convenience dictate whether to use functions or inlining
  - The paragraph "MyTag" (containing the script) refers to itself in the script

The Array Object

- In JavaScript the Array object is used to store a collection of related data items in a single variable
- Memory locations are allocated to it using the `new` operator, e.g.

```
var names = new Array(5)
```

  - Creates a new array of five elements
  - Each element has an initial value `undefined`
  - If the size is eliminated, an empty array is created
- The first element in an array is at index 0
- Arrays in JavaScript are "dynamic" entities (objects) that can change size after they are created
  - JavaScript reallocates an Array when a value is assigned to an element that is outside the bounds of the original Array
  - Elements between the last element of the original Array and the new element have undefined values
  - Referring to an element outside the *Array* bounds is normally a logical error.
- Each array has a length attribute that be used to get the size of the array, e.g.

```
arrayname.length
```

Passing Arrays to Functions

- In the function call, specify the array name, e.g.

```
outputArray(names);
```

  - Passes the array n to the function outputArray()

- In the function header, specify a parameter that will receive the array, e.g.

```
function outputArray(b)
```

- Arrays are objects so they are passed to functions by reference
  - Note that numbers, boolean values and strings are passed to functions by value

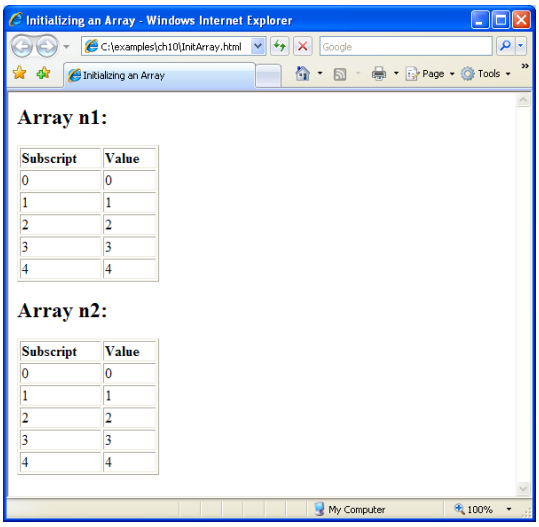
Example

```
8     <head>
9        <title>Initializing an Array</title>
10       <style type = "text/css">
11          table { width: 10em }
12          th    { text-align: left }
13       </style>
14       <script type = "text/javascript">
15          <!--
16          // create (declare) two new arrays
17          var n1 = new Array( 5 ); // allocate five-element Array
18          var n2 = new Array(); // allocate empty Array
19
20          // assign values to each element of Array n1
21          for ( var i = 0; i < n1.length; ++i )
22             n1[ i ] = i;
23
24          // create and initialize five elements in Array n2
25          for ( i = 0; i < 5; ++i )
26             n2[ i ] = i;
27
28          outputArray( "Array n1:", n1 );
29          outputArray( "Array n2:", n2 );
```

Example (cont.)

```
31          // output the heading followed by a two-column table
32          // containing subscripts and elements of "theArray"
33          function outputArray( heading, theArray )
34          {
35             document.writeln( "<h2>" + heading + "</h2>" );
36             document.writeln( "<table border = \"1\"" );
37             document.writeln( "<thead><th>Subscript</th>" +
38                "<th>Value</th></thead><tbody>" );
39
40             // output the subscript and value of each array element
41             for ( var i = 0; i < theArray.length; i++ )
42                document.writeln( "<tr><td>" + i + "</td><td>" +
43                   theArray[ i ] + "</td></tr>" );
44
45             document.writeln( "</tbody></table>" );
46          } // end function outputArray
47       // -->
48         </script>
49      </head><body></body>
50 </html>
```


Example (cont.)



Initializing Array Elements

- The initial values of an array can be specified in two ways:
  - using a comma-separated initializer list enclosed in square brackets `[]`, e.g.

```
var n =[4, 2, 5, 6, 10];
```

  - as arguments in the parentheses following `new Array`, e.g.

```
var n = new Array(4, 2, 5, 6, 10);
```

- The array's size is determined by the number of values listed
- If an element value is not known, use a comma as a place holder, e.g.

```
var n =[4, 2,, 6, 10];
```


Accessing Array Elements

- You can iterate through all elements of an array using for-loop or for-in loop
  - But `for...in` loop skips any undefined elements
- Example using for-loop

```
18          for ( var i = 0; i < theArray.length; i++ )
19             total1 += theArray[ i ];
```

- Example using for-in loop

```
25          for ( var element in theArray )
26             total2 += theArray[ element ];
```

Random Image Generator Using Arrays

- Uses a pictures array to store the names of the image files as strings
- Accesses the array using a randomized index

```
16          var pictures =
17             [ "CPE", "EPT", "GPP", "GUI", "PERF", "PORT", "SEO" ];

22     document.write( "<img src = \"" +
23             pictures[ Math.floor( Math.random() * 7 ) ] + ".gif\" />" );
```


Some Array Operations

- Sorting an array using the sort method

  var n=[4, 5, 2, 10];

  n.sort();

  - With no arguments, the method uses string comparisons to determine the sorting order of the Array elements
  - It can take as its optional argument the name of a function (called the comparator function) that compares its two arguments and returns a negative value, zero, or a positive value, if the first argument is less than, equal to, or greater than the second, respectively
    - Note that functions in JavaScript are data; they can be assigned to variables, stored in `Arrays` and passed to functions just like other data

Example

```
12          var a = [ 10, 1, 9, 2, 8, 3, 7, 4, 6, 5 ];
13
14          document.writeln( "<h1>Sorting an Array</h1>" );
15          outputArray( "Data items in original order: ", a );
16          a.sort( compareIntegers );  // sort the array
17          outputArray( "Data items in ascending order: ", a );
18
19          // output the heading followed by the contents of theArray
20          function outputArray( heading, theArray )
21          {
22             document.writeln( "<p>" + heading +
23                theArray.join( " " ) + "</p>" );
24       }// end function outputArray
```

```
26          // comparison function for use with sort
27          function compareIntegers( value1, value2 )
28          {
29             return parseInt( value1 ) - parseInt( value2 );
30          } // end function compareIntegers
```


Some Array Operations (cont.)

- To sort an array numerically:

```
n.sort(function(a, b){return a - b;})
```

- `join` method of an Array
  - Returns a string that contains all of the elements of an array, separated by the string supplied in the function's argument
  - If an argument is not specified, the empty string is used as the separator

```
document.writeln(
        theArray.join( " " ) + "<br />" );
```

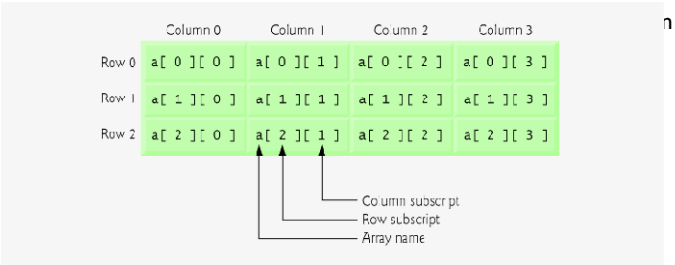
- `concat` method can be used to join two arrays

```
document.write(arr1.concat(arr2));
```

Some Array Operations (cont.)

- arrayObject.reverse()
  - Reverses the order of the elements in an array
- arrayObject.slice(start [, end])
  - Returns selected elements from an existing array
- arrayObject.push()
  - Adds one or more elements to the end of an array and returns the new length
- arrayObject.pop()
  - Removes and returns the last element of an array
- arrayObject.shift()
  - Removes and returns the first element of an array
- arrayObject.toString()
  - Converts an array to a string and returns the result
- arrayObject.splice(index,howmany,elem1,.....,elemX)
  - used to remove and add new elements to an array.

Multidimensional Arrays

- Multidimensional arrays are maintained as arrays of arrays, e.g.
  - A 2D array is a 1D array; each element is another 1D array
  - Hence rows of a two-dimensional array can vary in length
- Two-dimensional array element accessed using an element name of the form `a[ i ][ j ]`
  - a is the name of the array
  - 

Two-Dimensional Arrays (cont.)

- Arrays are allocated dynamically with the `new operator`
  - `Example`

```
var b;
b = new Array (2); // two rows
b[0] = new Array(3); // three elements in the first row
b[1] = new Array(5); //five elements in the second row
```

- 2D arrays can be initialized in declarations like a 1D array, with values grouped by row in square brackets
  - Examples

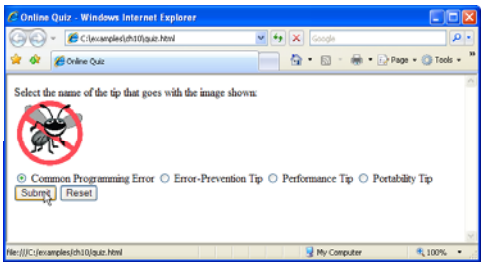
```
var b = [[1, 2], [3, 4]];
var b = [[1, 2], [3, 4, 5]];
```

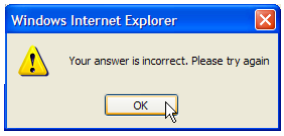

Two-Dimensional Arrays (cont.)

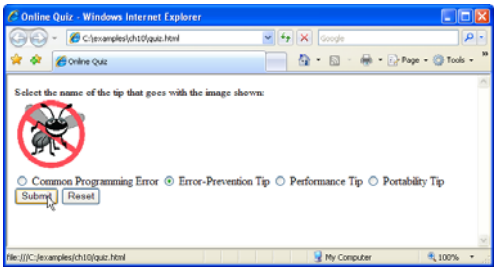
- Example

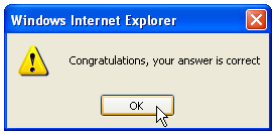
```
for ( var i in theArray )
{
      for ( var j in theArray[ i ] )
            document.write( theArray[ i ][ j ] + " " );
      document.writeln( "<br />" );
}
```

Example: Building an Online Quiz









Example: Building an Online Quiz (cont.)

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Online Quiz</title>
10       <script type = "text/JavaScript">
11          <!--
12          function checkAnswers()
13          {
14             var myQuiz = document.getElementById( "myQuiz" );
15
16             // determine whether the answer is correct
17             if ( myQuiz.elements[ 1 ].checked )
18                alert( "Congratulations, your answer is correct" );
19             else // if the answer is incorrect
20                alert( "Your answer is incorrect. Please try again" );
21          } // end function checkAnswers
22          -->
23       </script>
24    </head>
25    <body>
26       <form id = "myQuiz" onsubmit = "checkAnswers()" action = "">
27          <p>Select the name of the tip that goes with the
28             image shown:<br />
29             <img src="EPT.gif" alt="mystery tip"/>
30             <br />
```

Example: Building an Online Quiz (cont.)

```
32            <input type = "radio" name = "radiobutton" value = "CPE" />
33            <label>Common Programming Error</label>
34
35            <input type = "radio" name = "radiobutton" value = "EPT" />
36            <label>Error-Prevention Tip</label>
37
38            <input type = "radio" name = "radiobutton" value = "PERF" />
39            <label>Performance Tip</label>
40
41            <input type = "radio" name = "radiobutton" value = "PORT" />
42            <label>Portability Tip</label><br />
43
44            <input type = "submit" name = "submit" value = "Submit" />
45            <input type = "reset" name = "reset" value = "Reset" />
46         </p>
47      </form>
48   </body>
49 </html>
```


Q & A

References

- Some useful links with examples and other resources:
  - *Internet and World Wide Web How to Program*, 4/e, H. M. Deitel, P. J. Deitel, and A. B. Goldberg, Pearson Education Inc., 2008. Chapters 6-13.
  - *Web Development and Design Foundations with XHTML*, 4/e, Pearson Education Inc. 2009. Chapter 14.
  - http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.3/guide/intro.html
  - http://msconline.maconstate.edu/tutorials/JSDHTML/default.htm
  - http://www.javascript.com
  - Beginning JavaScript Tutorials http://www.pageresource.com/jscript/index.html
  - JavaScript Tutorial for the Total Non-Programmer http://www.webteacher.com/javascript/
  - More Beginning JavaScript Tutorials http://echoecho.com/javascript.htm
  - Core JavaScript 1.5 Reference Manual http://www.webreference.com/javascript/reference/core_ref
  - The JavaScript Source http://javascript.internet.com