10

JavaScript: Arrays

Outline

10.1 Introduction

- **Arrays**
  - **Data structures consisting of related data items**
  - **Sometimes called collections of data items**
- **JavaScript arrays**
  - **“dynamic” entities that can change size after they are created**

10.2 Arrays (Cont.)

- **The first element in every array is the zeroth element.**
- **The *i*th element of array `c` is referred to as `c[i-1]`.**
- **Array names follow the same conventions as other identifiers**
- **A subscripted array name**
  - **can be used on the left side of an assignment to place a new value into an array element**
  - **can be used on the right side of an assignment operation to use its value**
- **Every array in JavaScript knows its own length, which it stores in its `length` attribute and can be found with the expression *`arrayname`*`.length`**

10.3 Declaring and Allocating Arrays

- **JavaScript arrays are `Array` objects.**
- **Creating new objects using the `new` operator is known as creating an instance or instantiating an object**
- **Operator `new` is known as the dynamic memory allocation operator**

Fig. 10.3 | Initializing the elements of an array (Part 1 of 3).

```
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">
4
5  <!-- Fig. 10.3: InitArray.html -->
6  <!-- Initializing the elements of an array. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
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 );
30
```

Operator `new` allocates an `Array` called `n1` with five elements

Operator `new` allocates an empty `Array` called `n2`

Zero-based counting used in `for` loop to set each element's value equal to its subscript

Five elements added and initialized in `n2`, which dynamically expands

```
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>
```

Outputs the subscript and value of every array element in a table

Fig. 10.3 | Initializing the elements of an array (Part 2 of 3).

10.4 Examples Using Arrays (Cont.)

- **Arrays can be created using a comma-separated initializer list enclosed in square brackets (`[]`)**
  - **The array's size is determined by the number of values in the initializer list**
- **The initial values of an array can be specified as arguments in the parentheses following `new Array`**
  - **The size of the array is determined by the number of values in parentheses**

Fig. 10.4 | Declaring and initializing arrays (Part 1 of 3).

```
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">
4
5  <!-- Fig. 10.4: InitArray2.html -->
6  <!-- Declaring and initializing arrays. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Initializing an Array with a Declaration</title>
10       <style type = "text/css">
11          table { width: 15em }
12          th    { text-align: left }
13       </style>
14       <script type = "text/javascript">
15          <!--
16          // Initializer list specifies the number of elements and
17          // a value for each element.
18          var colors = new Array( "cyan", "magenta","yellow", "black" );
19          var integers1 = [ 2, 4, 6, 8 ];
20          var integers2 = [ 2, , , 8 ];
21
22          outputArray( "Array colors contains", colors );
23          outputArray( "Array integers1 contains", integers1 );
24          outputArray( "Array integers2 contains", integers2 );
25
26          // output the heading followed by a two-column table
27          // containing the subscripts and elements of theArray
28          function outputArray( heading, theArray )
29          {
```

Creates an array with four elements, all of which are defined

Creates an array with four elements, all of which are defined in an initializer list

Creates an array with four elements, two of which reserve space for values to be specified later

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

Fig. 10.4 | Declaring and initializing arrays (Part 2 of 3).

10.6 References and Reference Parameters

- **Two ways to pass arguments to functions (or methods)**
  - **pass-by-value**
  - **pass-by-reference**
- **Pass-by-value**
  - **a *copy* of the argument's value is made and is passed to the called function**
- **In JavaScript, numbers, boolean values and strings are passed to functions by value.**
- **Pass-by-reference**
  - **The caller gives the called function direct access to the caller's data and allows it to modify the data if it so chooses**
  - **Can improve performance because it can eliminate the overhead of copying large amounts of data, but it can weaken security because the called function can access the caller's data**

10.6 References and Reference Parameters (Cont.)

- **All objects are passed to functions by reference**
- **Arrays are objects in JavaScript, so `Array`s are passed to a function by reference**
  - **a called function can access the elements of the caller's original `Array`s.**
- **Name of an array**
  - **actually a reference to an object that contains the array elements and the `length` variable**

10.7 Passing Arrays to Functions

- **Pass an array as an argument to a function**
  - **Specify the name of the array (a reference to the array) without brackets**
- **Although entire arrays are passed by reference, *individual numeric and boolean array elements* are passed *by value* exactly as simple numeric and boolean variables are passed**
  - **Such simple single pieces of data are called scalars, or scalar quantities**
  - **To pass an array element to a function, use the subscripted name of the element as an argument in the function call**
- **`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**

Fig. 10.8 | Passing arrays and individual array elements to functions (Part 1 of 3).

```
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">
4
5  <!-- Fig. 10.8: PassArray.html -->
6  <!-- Passing arrays and individual array elements to functions. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Passing arrays and individual array
10          elements to functions</title>
11       <script type = "text/javascript">
12          <!--
13          var a = [ 1, 2, 3, 4, 5 ];
14
15          document.writeln( "<h2>Effects of passing entire " +
16             "array by reference</h2>" );
17          outputArray( "Original array: ", a );
18
19          modifyArray( a );  // array a passed by reference
20
21          outputArray( "Modified array: ", a );
22
23          document.writeln( "<h2>Effects of passing array " +
24             "element by value</h2>" +
25             "a[3] before modifyElement: " + a[ 3 ] );
26
27          modifyElement( a[ 3 ] ); // array element a[3] passed by value
28
29          document.writeln( "<br />a[3] after modifyElement: " + a[ 3 ] );
30
```

Passes array `a` to function `modifyArray` by reference

Passes array element `a[3]` to function `modifyElement` by value

```
31          // outputs heading followed by the contents of "theArray"
32          function outputArray( heading, theArray )
33          {
34             document.writeln(
35                heading + theArray.join( " " ) + "<br />" );
36          } // end function outputArray
37
38          // function that modifies the elements of an array
39          function modifyArray( theArray )
40          {
41             for ( var j in theArray )
42                theArray[ j ] *= 2;
43          } // end function modifyArray
44
45          // function that modifies the value passed
46          function modifyElement( e )
47          {
48             e *= 2; // scales element e only for the duration of the
49                     // function
50             document.writeln( "<br />value in modifyElement: " + e );
51          } // end function modifyElement
52          // -->
53       </script>
54    </head><body></body>
55 </html>
```

Creates a string containing all the elements in `theArray`, separated by " "

Multiplies each element in `theArray` by 2, which persists after the function has finished

Multiplies the array element by 2, but only for the duration of the function

Fig. 10.8 | Passing arrays and individual array elements to functions (Part 2 of 3).

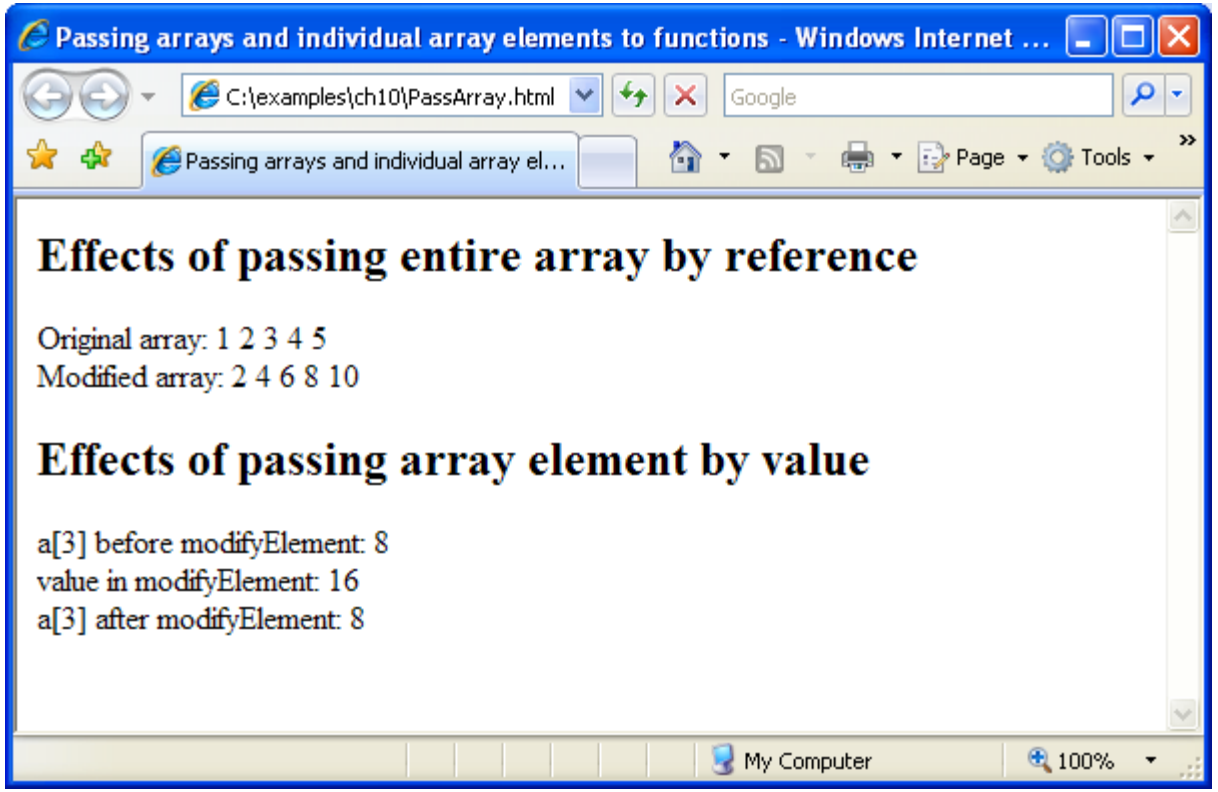

Fig. 10.8 | Passing arrays and individual array elements to functions (Part 3 of 3).

10.8 Sorting Arrays

- **Sorting data**
  - **Putting data in a particular order, such as ascending or descending**
  - **One of the most important computing functions**
- **`Array` object in JavaScript has a built-in method `sort`**
  - **With no arguments, the method uses string comparisons to determine the sorting order of the Array elements**
  - **Method `sort` takes 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**
- **Functions in JavaScript are considered to be data**
  - **They can be assigned to variables, stored in `Array`s and passed to functions just like other data**

Fig. 10.9 | Sorting an array with `sort` (Part 1 of 2).

```
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">
4
5  <!-- Fig. 10.9: Sort.html -->
6  <!-- Sorting an array with sort. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Sorting an Array with Array Method sort</title>
10       <script type = "text/javascript">
11          <!--
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
25
```

Passes function `compareIntegers` to method `a.sort` to arrange the elements of `a` in ascending numerical order

```
26          // comparison function for use with sort
27          function compareIntegers( value1, value2 )
28          {
29             return parseInt( value1 ) - parseInt( value2 );
30          } // end function compareIntegers
31          // -->
32       </script>
33    </head><body></body>
34 </html>
```

Defines a function comparing integers to be passed to method `sort` (to replace the default string comparison function)

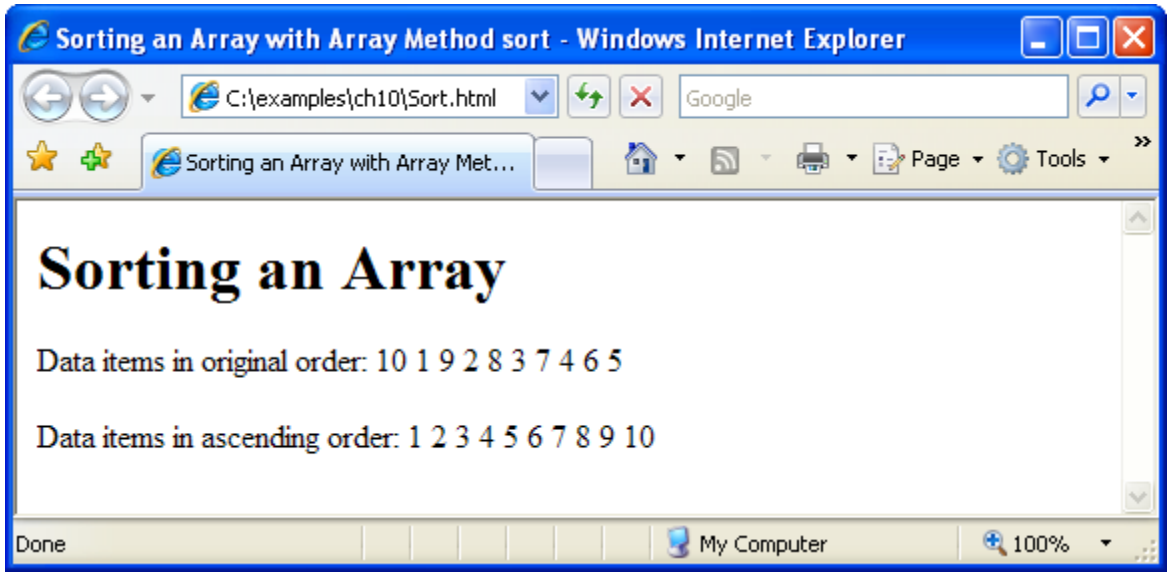

Fig. 10.9 | Sorting an array with **`sort`** (Part 2 of 2).

```
55    <body>
56       <form action  = "">
57          <p>Enter integer search key<br />
58          <input id = "inputVal" type = "text" />
59          <input type = "button" value = "Search"
60                onclick = "buttonPressed()" /><br /></p>
61          <p>Result<br />
62          <input id = "result" type = "text" size = "30" /></p>
63       </form>
64    </body>
65 </html>
```

When the `Search` button is pressed, calls function `buttonPressed`

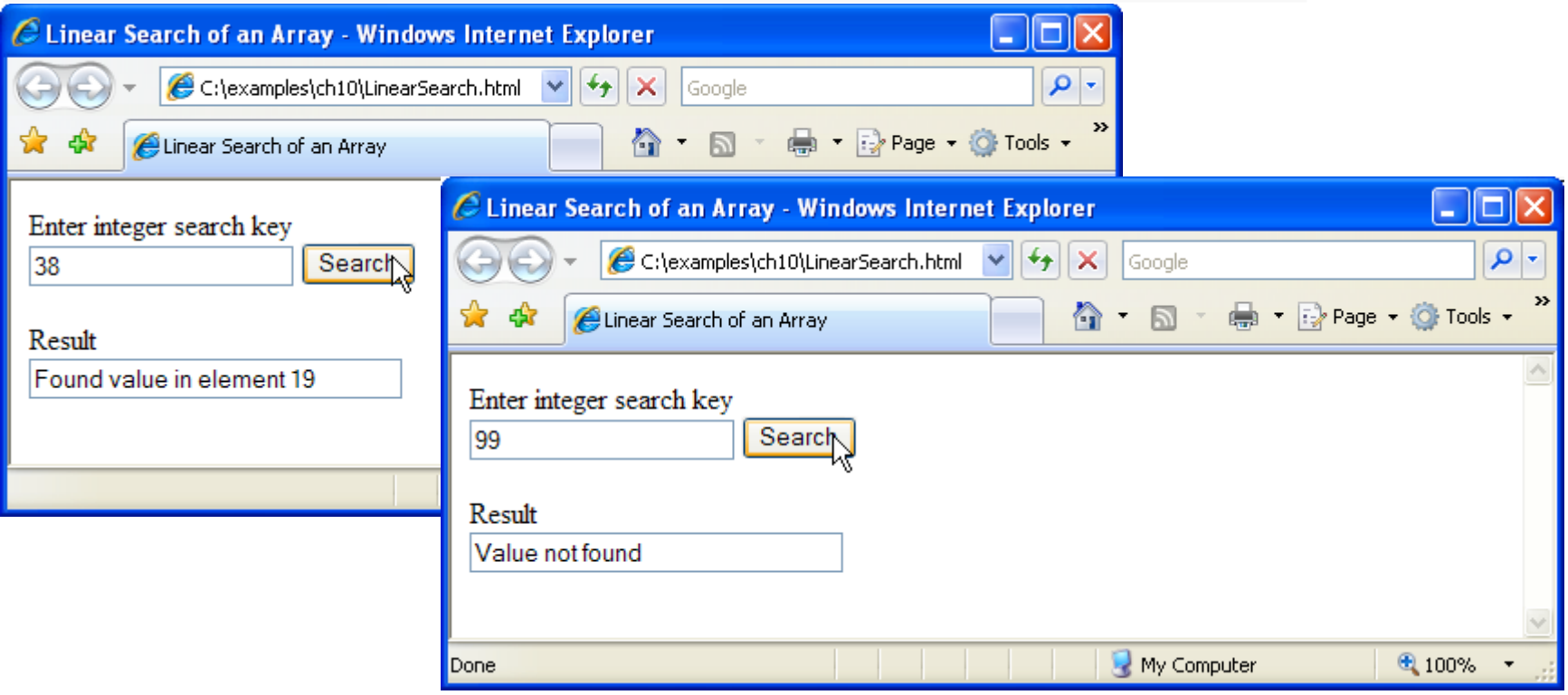

Fig. 10.10 | Linear search of an array (Part 3 of 3).

10.10 Multidimensional Arrays

- **To identify a particular two-dimensional multidimensional array element**
  - **Specify the two subscripts**
  - **By convention, the first identifies the element's row, and the second identifies the element's column**
- **In general, an array with *m* rows and *n* columns is called an *m*-by-*n* array**
- **Two-dimensional array element accessed using an element name of the form `a[ i ][ j ]`**
  - **`a` is the name of the array**
  - **`i` and `j` are the subscripts that uniquely identify the row and column**
- **Multidimensional arrays are maintained as arrays of arrays**

10.10 Multidimensional Arrays (Cont.)

- **Multidimensional arrays can be initialized in declarations like a one-dimensional array, with values grouped by row in square brackets**
  - **The interpreter determines the number of rows by counting the number of sub initializer**
  - **The interpreter determines the number of columns in each row by counting the number of values in the sub-array that initializes the row**
- **The rows of a two-dimensional array can vary in length**
- **A multidimensional array in which each row has a different number of columns can be allocated dynamically with operator `new`**

Fig. 10.13 | Initializing multidimensional arrays (Part 1 of 2).

```
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">
4
5  <!-- Fig. 10.13: InitArray3.html -->
6  <!-- Initializing multidimensional arrays. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Initializing Multidimensional Arrays</title>
10       <script type = "text/javascript">
11          <!--
12          var array1 = [ [ 1, 2, 3 ], // first row
13                         [ 4, 5, 6 ] ]; // second row
14          var array2 = [ [ 1, 2 ], // first row
15                         [ 3 ], // second row
16                         [ 4, 5, 6 ] ]; // third row
17
18          outputArray( "Values in array1 by row", array1 );
19          outputArray( "Values in array2 by row", array2 );
20
21          function outputArray( heading, theArray )
22          {
23             document.writeln( "<h2>" + heading + "</h2><pre>" );
24
25             // iterates through the set of one-dimensional arrays
26             for ( var i in theArray )
27             {
28                // iterates through the elements of each one-dimensional
29                // array
30                for ( var j in theArray[ i ] )
31                   document.write( theArray[ i ][ j ] + " " );
```

Initializes `array1` with an initializer list of sub initializer lists

Initializes `array2` with rows of different lengths

Nested `for…in` statements traverse the arrays by iterating through the sets of one-dimensional arrays, then through the elements of each of those one-dimensional arrays

```
32
33             document.writeln( "<br />" );
34          } // end for
35
36          document.writeln( "</pre>" );
37       } // end function outputArray
38       // -->
39    </script>
40 </head><body></body>
41 </html>
```

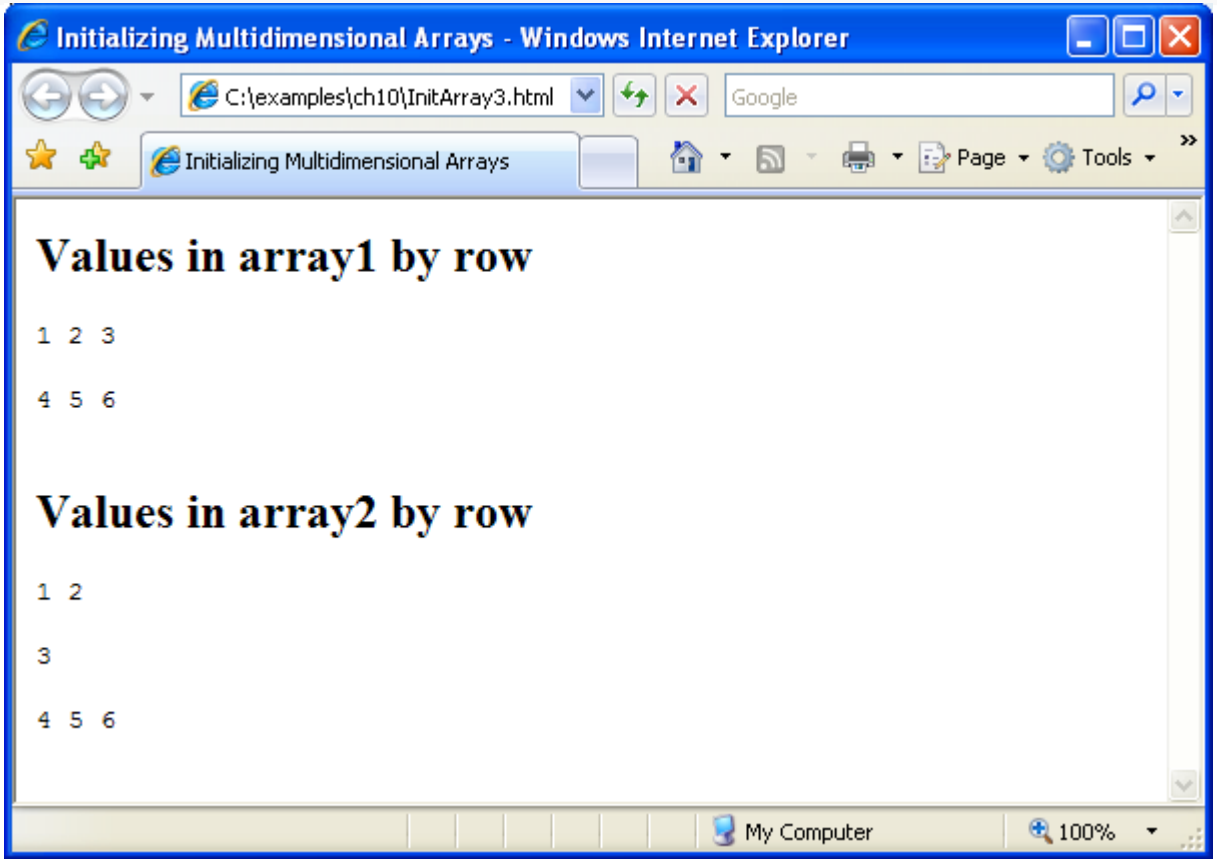

Fig. 10.13 | Initializing multidimensional arrays (Part 2 of 2).

10.11 Building an Online Quiz

- XHTML `form` elements can be accessed individually using `getElementById` or through the `elements` property of the containing `form` object
- `elements` property
  - contains an array of all the `form`'s controls
- Property `checked` of a radio button
  - `true` when the radio button is selected
  - `false` when the radio button is not selected

```
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">
4
5  <!-- Fig. 10.14: quiz.html -->
6  <!-- Online quiz graded with JavaScript. -->
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 />
```

Checks to see if the second radio button of the `myQuiz form` is selected

Fig. 10.14 | Online quiz graded with JavaScript (Part 1 of 3).

```
31
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>
```


Fig. 10.14 | Online quiz graded with JavaScript (Part 2 of 3).

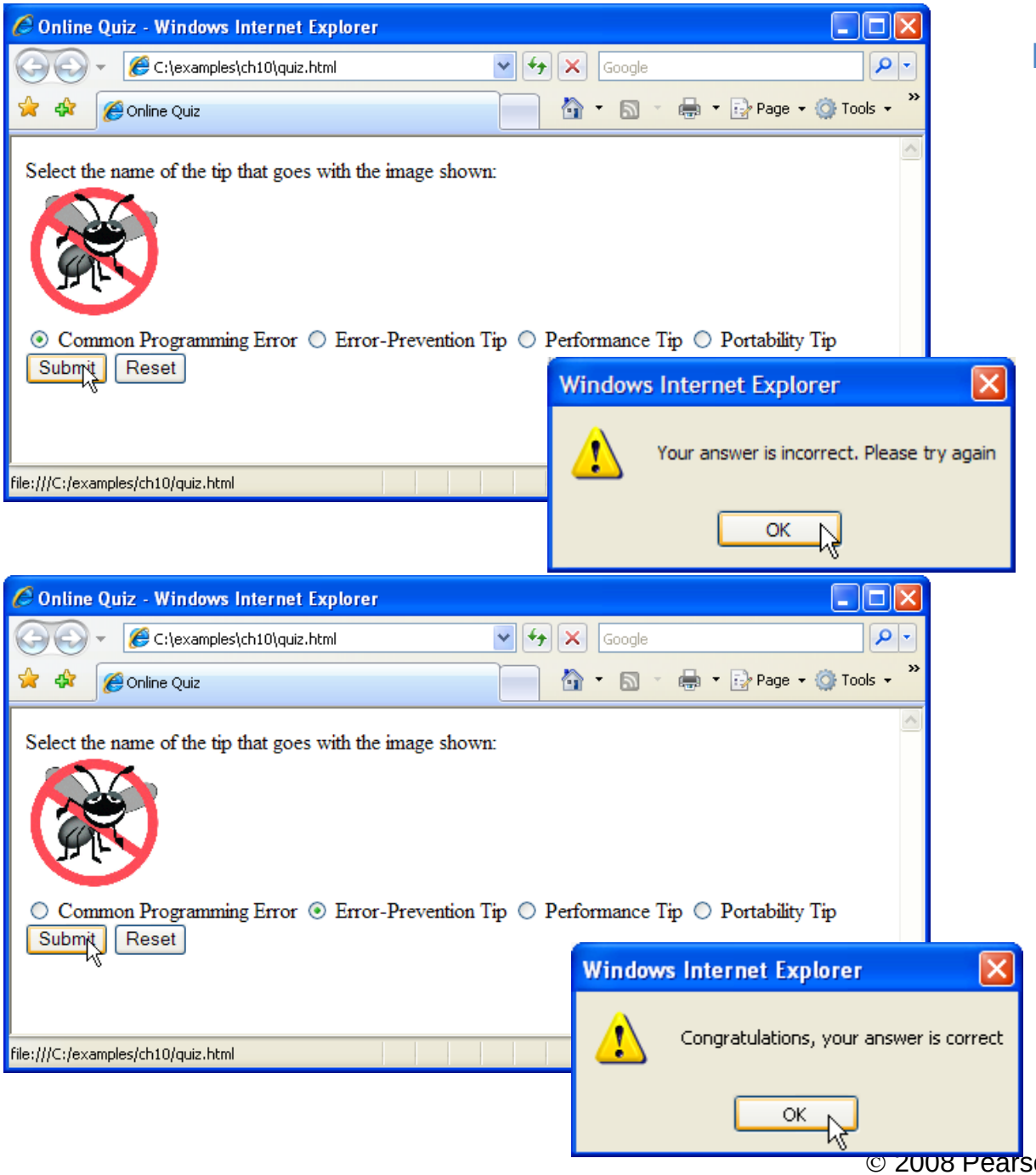

Fig. 10.14 | Online quiz graded with JavaScript (Part 3 of 3).