10

JavaScript: Arrays

10.1 Introduction

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

10.2 Arrays

- An array is a group of memory locations
  - All have the same name and normally are of the same type (although this attribute is not required in JavaScript)
- Each individual location is called an element
- We may refer to any one of these elements by giving the array's name followed by the position number of the element in square brackets ([])

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

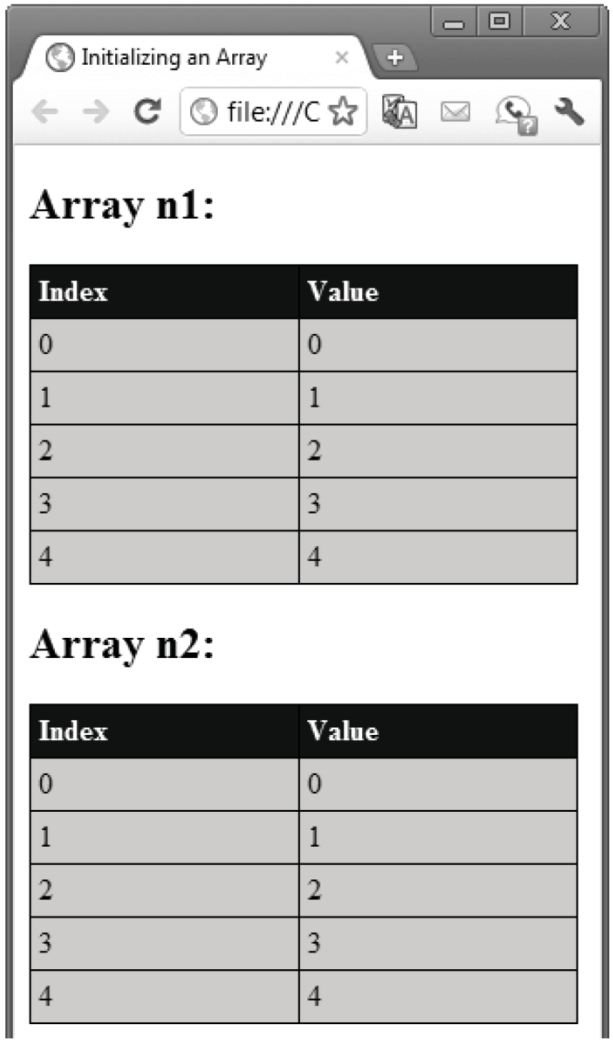
- JavaScript arrays are **`Array` objects**.
- You use the **new operator** to create an array and to specify the number of elements in an array.
- The new operator creates an object as the script executes by obtaining enough memory to store an object of the type specified to the right of new .

10.4 Examples Using Arrays

- Zero-based counting is usually used to iterate through arrays
- JavaScript reallocates an `Array` when a value is assigned to an element that is outside the bounds of the original `Array`

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 10.3: InitArray.html -->
4  <!-- Web page for showing the results of initializing arrays. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Initializing an Array</title>
9        <link rel = "stylesheet" type = "text/css" href = "tablestyle.css">
10       <script src = "InitArray.js"></script>
11    </head>
12    <body>
13       <div id = "output1"></div>
14       <div id = "output2"></div>
15    </body>
16 </html>
```

Fig. 10.3 | Web page for showing the results of initializing arrays. (Part 1 of 2.)



```
1  // Fig. 10.4: InitArray.js
2  // Create two arrays, initialize their elements and display them
3  function start()
4  {
5     var n1 = new Array( 5 ); // allocate five-element array
6     var n2 = new Array(); // allocate empty array
7
8     // assign values to each element of array n1
9     var length = n1.length; // get array's length once before the loop
10
11    for ( var i = 0; i < length; ++i )
12    {
13       n1[ i ] = i;
14    } // end for
15
16    // create and initialize five elements in array n2
17    for ( i = 0; i < 5; ++i )
18    {
19       n2[ i ] = i;
20    } // end for
21
22    outputArray( "Array n1:", n1, document.getElementById( "output1" ) );
23    outputArray( "Array n2:", n2, document.getElementById( "output2" ) );
24 } // end function start
```

Fig. 10.4 | Create two arrays, initialize their elements and display them. (Part 1 of 2.)

```
25
26    // output the heading followed by a two-column table
27    // containing indices and elements of "theArray"
28    function outputArray( heading, theArray, output )
29    {
30       var content = "<h2>" + heading + "</h2><table>" +
31          "<thead><th>Index</th><th>Value</th></thead><tbody>";
32
33       // output the index and value of each array element
34       var length = theArray.length; // get array's length once before loop
35
36       for ( var i = 0; i < length; ++i )
37       {
38          content += "<tr><td>" + i + "</td><td>" + theArray[ i ] +
39             "</td></tr>";
40       } // end for
41
42       content += "</tbody></table>";
43       output.innerHTML = content; // place the table in the output element
44    } // end function outputArray
45
46    window.addEventListener( "load", start, false );
```

Fig. 10.4 | Create two arrays, initialize their elements and display them. (Part 2 of 2.)

10.4 Examples Using Arrays (Cont.)

Using an Initializer List

- 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

10.4.2 Initializing Arrays with Initializer Lists

- The example in Figs. 10.5–10.6 creates three `Array` objects to demonstrate initializing arrays with initializer lists.
- Figure 10.5 is nearly identical to Fig. 10.3 but provides three `div`s in its body element for displaying this example's arrays.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 10.5: InitArray2.html -->
4  <!-- Web page for showing the results of initializing arrays. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Initializing an Array</title>
9        <link rel = "stylesheet" type = "text/css" href = "tablestyle.css">
10       <script src = "InitArray2.js"></script>
11    </head>
12    <body>
13       <div id = "output1"></div>
14       <div id = "output2"></div>
15       <div id = "output3"></div>
16    </body>
17 </html>
```

Fig. 10.5 | Web page for showing the results of initializing arrays. (Part 1 of 2.)

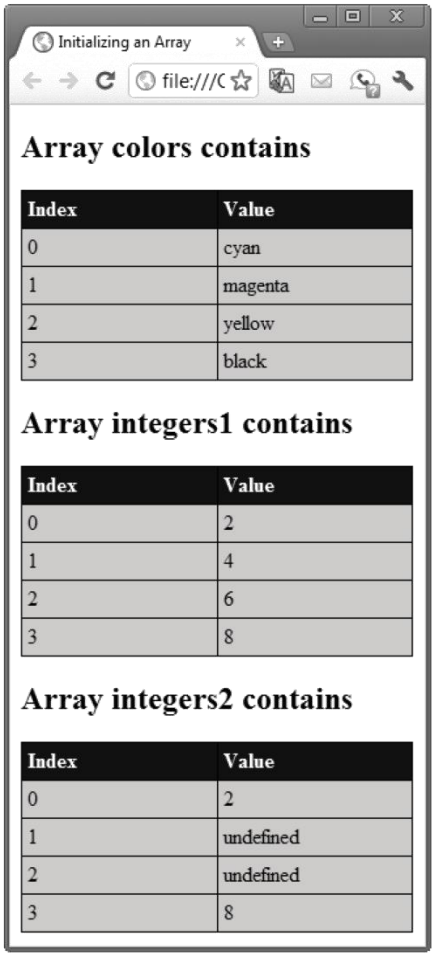


Fig. 10.5 | Web page for showing the results of initializing arrays. (Part 2 of 2.)

```
1  // Fig. 10.6: InitArray2.js
2  // Initializing arrays with initializer lists.
3  function start()
4  {
5     // Initializer list specifies the number of elements and
6     // a value for each element.
7     var colors = new Array( "cyan", "magenta","yellow", "black" );
8     var integers1 = [ 2, 4, 6, 8 ];
9     var integers2 = [ 2, , , 8 ];
10
11    outputArray( "Array colors contains", colors,
12       document.getElementById( "output1" ) );
13    outputArray( "Array integers1 contains", integers1,
14       document.getElementById( "output2" ) );
15    outputArray( "Array integers2 contains", integers2,
16       document.getElementById( "output3" ) );
17 } // end function start
18
```

Fig. 10.6 | Initializing arrays with initializer lists. (Part 1 of 2.)

```
19  // output the heading followed by a two-column table
20  // containing indices and elements of "theArray"
21  function outputArray( heading, theArray, output )
22  {
23     var content = "<h2>" + heading + "</h2><table>" +
24        "<thead><th>Index</th><th>Value</th></thead><tbody>";
25
26     // output the index and value of each array element
27     var length = theArray.length; // get array's length once before loop
28
29     for ( var i = 0; i < length; ++i )
30     {
31        content += "<tr><td>" + i + "</td><td>" + theArray[ i ] +
32           "</td></tr>";
33     } // end for
34
35     content += "</tbody></table>";
36     output.innerHTML = content; // place the table in the output element
37  } // end function outputArray
38
39  window.addEventListener( "load", start, false );
```

Fig. 10.6 | Initializing arrays with initializer lists. (Part 2 of 2.)

10.4.3 Summing the Elements of an Array with for and for…in

- The example in Figs. 10.7–10.8 sums an array's elements and displays the results.
- The document in Fig. 10.7 shows the results of the script in Fig. 10.8.
- JavaScript's for…in Repetition Statement
  - Enables a script to perform a task for each element in an array

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 10.7: SumArray.html -->
4  <!-- HTML5 document that displays the sum of an array's elements. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Sum Array Elements</title>
9        <script src = "SumArray.js"></script>
10    </head>
11    <body>
12       <div id = "output"></div>
13    </body>
14 </html>
```

Fig. 10.7 | HTML5 document that displays the sum of an array's elements.

```
1  // Fig. 10.8: SumArray.js
2  // Summing the elements of an array with for and for...in
3  function start()
4  {
5     var theArray = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
6     var total1 = 0, total2 = 0;
7
8     // iterates through the elements of the array in order and adds
9     // each element's value to total1
10    var length = theArray.length; // get array's length once before loop
11
12    for ( var i = 0; i < length; ++i )
13    {
14       total1 += theArray[ i ];
15    } // end for
16
17    var results = "<p>Total using indices: " + total1 + "</p>";
18
```

Fig. 10.8 | Summing the elements of an array with `for` and `for...in`. (Part 1 of 2.)

```
19      // iterates through the elements of the array using a for... in
20      // statement to add each element's value to total2
21      for ( var element in theArray )
22      {
23         total2 += theArray[ element ];
24      } // end for
25
26      results += "<p>Total using for...in: " + total2 + "</p>";
27      document.getElementById( "output" ).innerHTML = results;
28   } // end function start
29
30   window.addEventListener( "load", start, false );
```

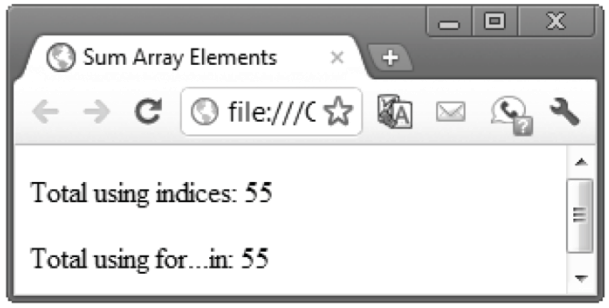


Fig. 10.8 | Summing the elements of an array with `for` and `for...in`. (Part 2 of 2.)

10.8 Sorting Arrays with `Array` Method `Sort`

- **Sorting** data
  - Putting data in a particular order, such as ascending or descending
  - One of the most important computing functions

10.8 Sorting Arrays with Array Method sort (Cont.)

- 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 argument the name of a function that compares its two arguments and returns
    - a negative value if the first argument is less than the second argument,
    - Zero if the arguments are *equal*, or
    - a positive value if the first argument is *greater than* the second

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 10.15: Sort.html -->
4  <!-- HTML5 document that displays the results of sorting an array. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Array Method sort</title>
9        <link rel = "stylesheet" type = "text/css" href = "style.css">
10       <script src = "Sort.js"></script>
11    </head>
12    <body>
13       <h1>Sorting an Array</h1>
14       <p id = "originalArray"></p>
15       <p id = "sortedArray"></p>
16    </body>
17 </html>
```

Fig. 10.15 | HTML5 document that displays the results of sorting an array. (Part 1 of 2.)

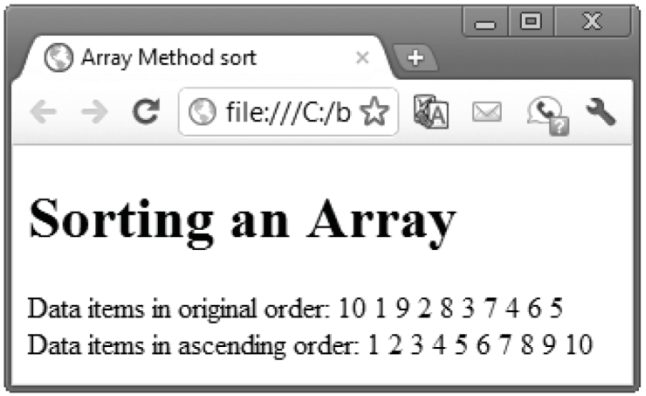


Fig. 10.15 | HTML5 document that displays the results of sorting an array. (Part 2 of 2.)

```
1  // Fig. 10.16: Sort.js
2  // Sorting an array with sort.
3  function start()
4  {
5     var a = [ 10, 1, 9, 2, 8, 3, 7, 4, 6, 5 ];
6
7     outputArray( "Data items in original order: ", a,
8        document.getElementById( "originalArray" ) );
9     a.sort( compareIntegers );  // sort the array
10    outputArray( "Data items in ascending order: ", a,
11       document.getElementById( "sortedArray" ) );
12 } // end function start
13
14 // output the heading followed by the contents of theArray
15 function outputArray( heading, theArray, output )
16 {
17    output.innerHTML = heading + theArray.join( " " );
18 } // end function outputArray
19
```

Fig. 10.16 | Sorting an array with sort. (Part 1 of 2.)

```
20   // comparison function for use with sort
21   function compareIntegers( value1, value2 )
22   {
23      return parseInt( value1 ) - parseInt( value2 );
24   } // end function compareIntegers
25
26   window.addEventListener( "load", start, false );
```

Fig. 10.16 | Sorting an array with `sort`. (Part 2 of 2.)

10.9 Searching Arrays with `Array` Method `indexOf`

- It's often necessary to determine whether an array contains a value that matches a certain *key value*.
- The process of locating a particular element value in an array is called *searching*.
- The `Array` object in JavaScript has built-in methods `indexOf` and `lastIndexOf` for searching arrays.
  - Method `indexOf` searches for the first occurrence of the specified key value
  - Method `lastIndexOf` searches for the last occurrence of the specified key value.
- If the key value is found in the array, each method returns the index of that value; otherwise, `-1` is returned.

10.9 Searching Arrays with `Array` Method `indexOf` (Cont.)

- Every `input` element has a **`value` property** that can be used to get or set the element's value.

Optional Second Argument to `indexOf` and `lastIndexOf`

- You can pass an optional second argument to methods `indexOf` and `lastIndexOf` that represents the index from which to start the search.
- By default, this argument's value is 0 and the methods search the entire array.
- If the argument is greater than or equal to the array's length, the methods simply return -1.
- If the argument's value is negative, it's used as an offset from the end of the array.

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 10.17: search.html -->
4  <!-- HTML5 document for searching an array with indexOf. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Search an Array</title>
9        <script src = "search.js"></script>
10    </head>
11    <body>
12       <form action  = "#">
13          <p><label>Enter integer search key:
14             <input id = "inputVal" type = "number"></label>
15             <input id = "searchButton" type = "button" value = "Search">
16          </p>
17          <p id = "result"></p>
18       </form>
19    </body>
20 </html>
```

Fig. 10.17 | HTML5 document for searching an array with `indexOf`. (Part 1 of 2.)

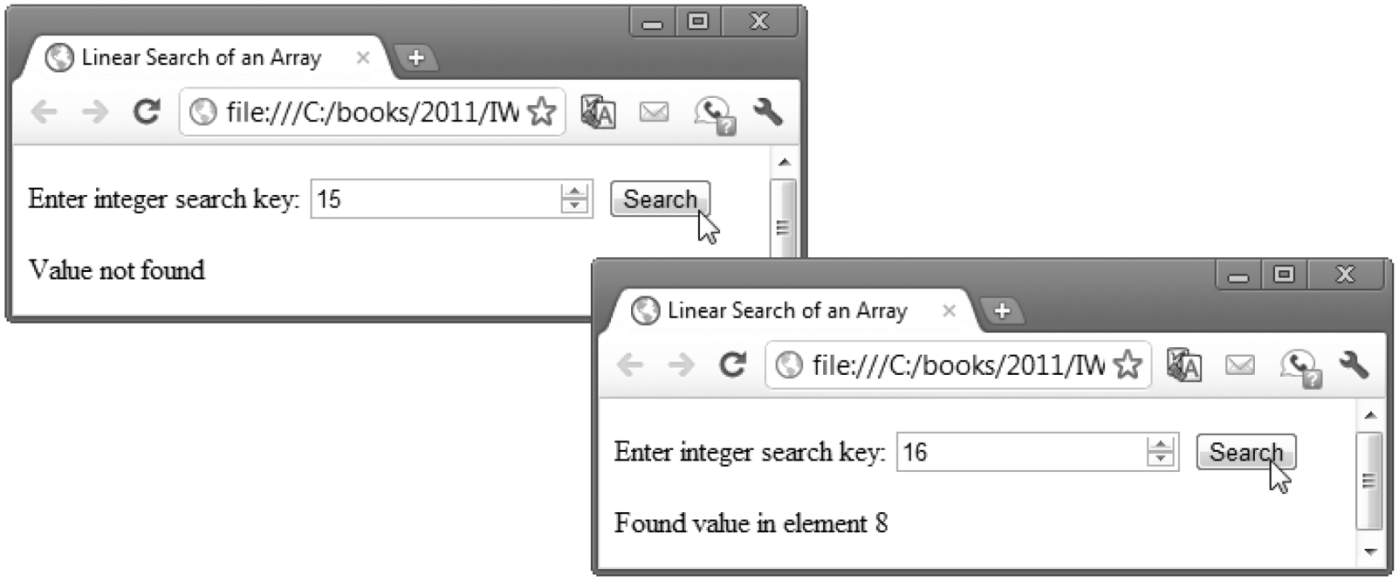


Fig. 10.17 | HTML5 document for searching an array with `indexOf`. (Part 2 of 2.)

```
1  // Fig. 10.18: search.js
2  // Search an array with indexOf.
3  var a = new Array( 100 );  // create an array
4
5  // fill array with even integer values from 0 to 198
6  for ( var i = 0; i < a.length; ++i )
7  {
8     a[ i ] = 2 * i;
9  } // end for
10
11 // function called when "Search" button is pressed
12 function buttonPressed()
13 {
14    // get the input text field
15    var inputVal = document.getElementById( "inputVal" );
16
17    // get the result paragraph
18    var result = document.getElementById( "result" );
19
20    // get the search key from the input text field then perform the search
21    var searchKey = parseInt( inputVal.value );
22    var element = a.indexOf( searchKey );
23
```

Fig. 10.18 | Search an array with `indexOf`.

```
24      if ( element != -1 )
25      {
26         result.innerHTML = "Found value in element " + element;
27      } // end if
28      else
29      {
30         result.innerHTML = "Value not found";
31      } // end else
32   } // end function buttonPressed
33
34   // register searchButton's click event handler
35   function start()
36   {
37      var searchButton = document.getElementById( "searchButton" );
38      searchButton.addEventListener( "click", buttonPressed, false );
39   } // end function start
40
41   window.addEventListener( "load", start, false );
```

Fig. 10.18 | Search an array with `indexOf`.

10.10 Multidimensional Arrays

- To identify a particular two-dimensional multidimensional array element
  - Specify the two indices
  - 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[ row ][ column ]`
  - a is the name of the array
  - `row` and `column` are the indices that uniquely identify the row and column

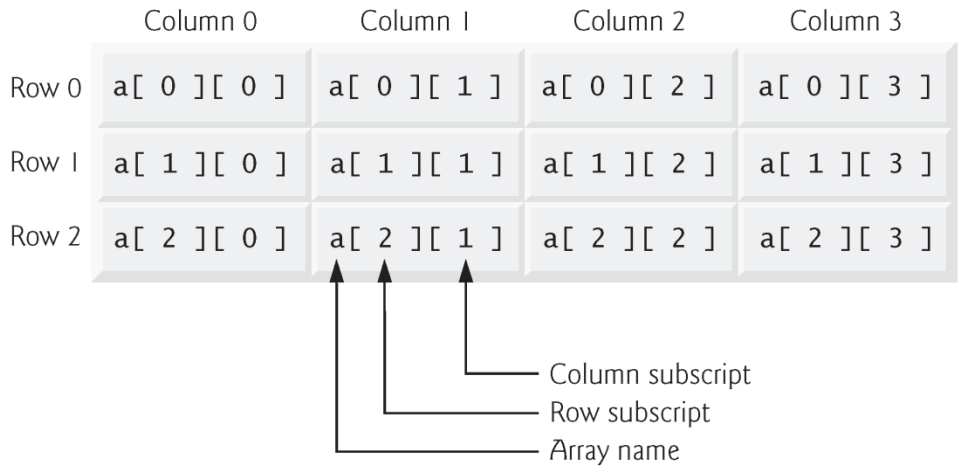


Fig. 10.19 | Two-dimensional array with three rows and four columns.

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

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 10.20: InitArray3.html -->
4  <!-- HTML5 document showing multidimensional array initialization. -->
5  <html>
6     <head>
7        <meta charset = "utf-8">
8        <title>Multidimensional Arrays</title>
9        <link rel = "stylesheet" type = "text/css" href = "style.css">
10       <script src = "InitArray3.js"></script>
11    </head>
12    <body>
13       <h2>Values in array1 by row</h2>
14       <div id = "output1"></div>
15       <h2>Values in array2 by row</h2>
16       <div id = "output2"></div>
17    </body>
18 </html>
```

Fig. 10.20 | HTML5 document showing multidimensional array initialization. (Part 1 of 2.)

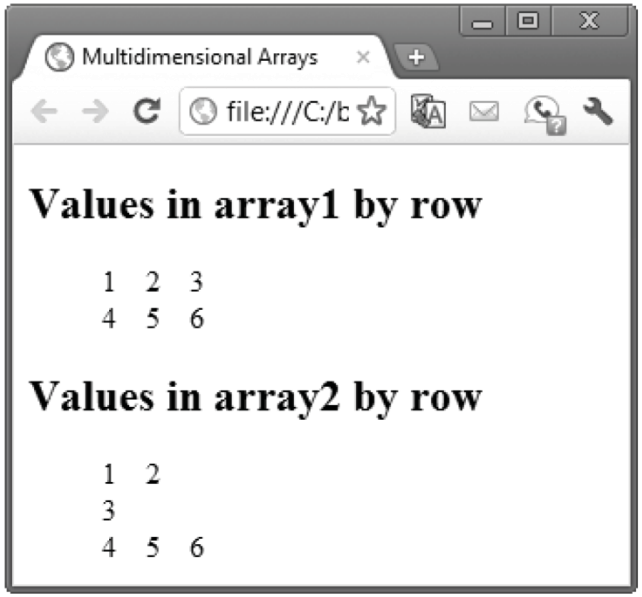


Fig. 10.20 | HTML5 document showing multidimensional array initialization. (Part 2 of 2.)

```
1  // Fig. 10.21: InitArray3.js
2  // Initializing multidimensional arrays.
3  function start()
4  {
5     var array1 = [ [ 1, 2, 3 ], // row 0
6                    [ 4, 5, 6 ] ]; // row 1
7     var array2 = [ [ 1, 2 ], // row 0
8                    [ 3 ], // row 1
9                    [ 4, 5, 6 ] ]; // row 2
10
11    outputArray( "Values in array1 by row", array1,
12       document.getElementById( "output1" ) );
13    outputArray( "Values in array2 by row", array2,
14       document.getElementById( "output2" ) );
15 } // end function start
16
```

Fig. 10.21 | Initializing multidimensional arrays. (Part 1 of 2.)

```
17  // display array contents
18  function outputArray( heading, theArray, output )
19  {
20     var results = "";
21
22     // iterates through the set of one-dimensional arrays
23     for ( var row in theArray )
24     {
25        results += "<ol>"; // start ordered list
26
27        // iterates through the elements of each one-dimensional array
28        for ( var column in theArray[ row ] )
29        {
30           results += "<li>" + theArray[ row ][ column ] + "</li>";
31        } // end inner for
32
33        results += "</ol>"; // end ordered list
34     } // end outer for
35
36     output.innerHTML = results;
37  } // end function outputArray
38
39  window.addEventListener( "load", start, false );
```

Fig. 10.21 | Initializing multidimensional arrays. (Part 2 of 2.)