Chapter 11 - JavaScript: Arrays

Outline

Objectives

- In this tutorial, you will learn:
  - To introduce the array data structure.
  - To understand the use of arrays to store, sort and search lists and tables of values.
  - To understand how to declare an array, initialize an array and refer to individual elements of an array.
  - To be able to pass arrays to functions.
  - To be able to search and sort an array.
  - To be able to declare and manipulate multi-dimensional arrays.

11.1 Introduction

- Arrays
  - Data structures of related items
    - Also called Collections
  - Dynamic

11.2 Arrays

- Arrays in JavaScript
  - Each element referenced by a number
    - Start at “zeroth element”
    - Subscript or index
  - Accessing a specific element
    - Name of array
    - Brackets
    - Number of element
  - Arrays know their length
    - `length` property

11.2 Arrays

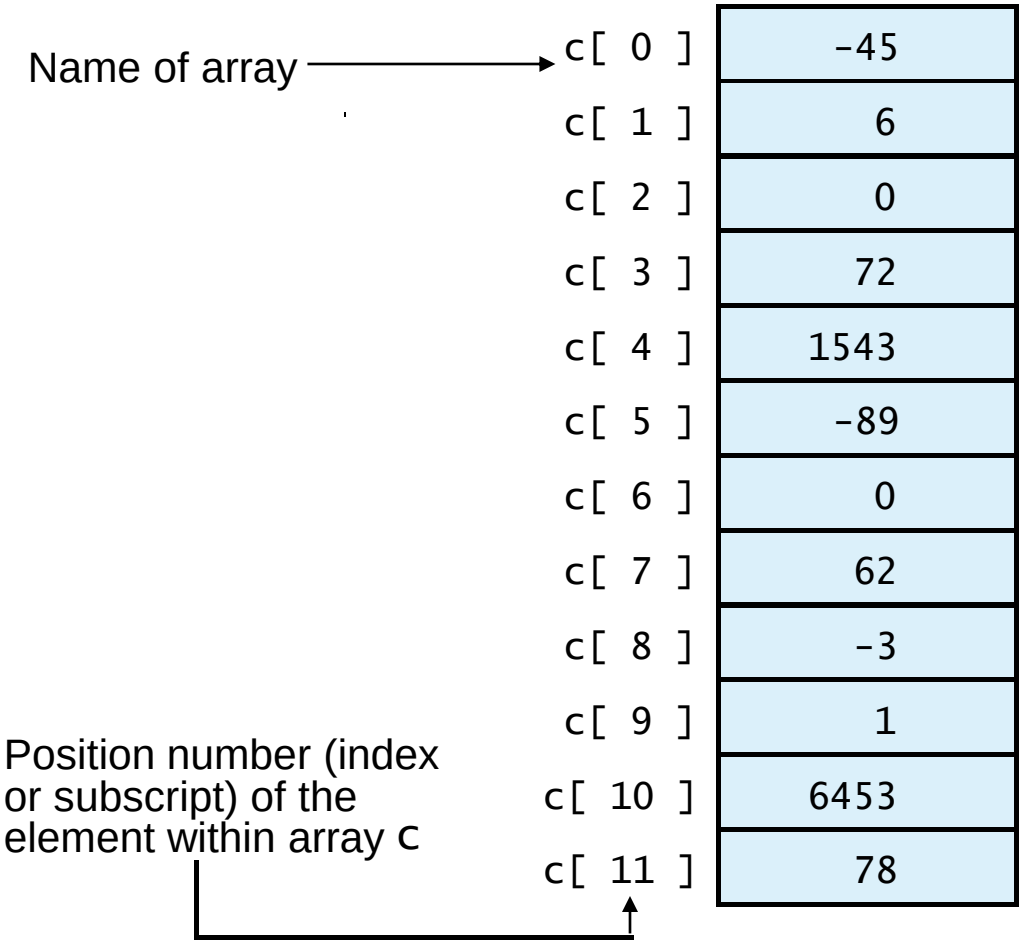


Fig. 11.1 A 12-element array.

11.2 Arrays

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

Fig. 11.2 Precedence and associativity of the operators discussed so far.

11.3 Declaring and Allocating Arrays

- Arrays in memory
  - Objects
  - Operator `new`
    - Allocates memory for objects
    - Dynamic memory allocation operator

```
var c;
c = new Array( 12 );
```

11.4 Examples Using Arrays

- Arrays grow dynamically
  - Allocate more space as items are added
- Must initialize array elements
  - Default value is undefined
  - `for` loops convenient
  - Referring to uninitialized elements or elements outside array bounds is an error

Outline

InitArray.html
(1 of 3)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.3: InitArray.html -->
<!-- Initializing an Array     -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Initializing an Array</title>

      <script type = "text/javascript">
         <!--
         // this function is called when the <body> elem
         // onload event occurs
         function initializeArrays()
         {
            var n1 = new Array( 5 );    // allo
            var n2 = new Array();       // allo

            // assign values to each element of Array n1
            for ( var i = 0; i < n1.length; ++i )
               n1[ i ] = i;
```

`Array n1` has five elements.

`Array n2` is an empty array.

The `for` loop initializes the elements in `n1` to their subscript numbers (0 to 4).

Outline

```
24
25          // create and initialize five-elements in Array n2
26          for ( i = 0; i < 5; ++i )
27             n2[ i ] = i;
28
29          outputArray( "Array n1 contains", n1 );
30          outputArray( "Array n2 contains", n2 );
31       }
32
33       // output "header" followed by a two-column table
34       // containing subscripts and elements of "theArray"
35       function outputArray( header, theArray )
36       {
37          document.writeln( "<h2>" + header + "</h2>" );
38          doc
39
40
41          doc
42
43             "<th align = \"left\">Value</th></thead><tbody>" );
```

The `for` loop adds f
initialize each elem

Each function displays the contents of its respective Array in an XHTML table.

The second time function `ouputArray` is called, variable `header` gets the value of "`Array n2 contains`" and variable `theArray` gets the value of `n2`.

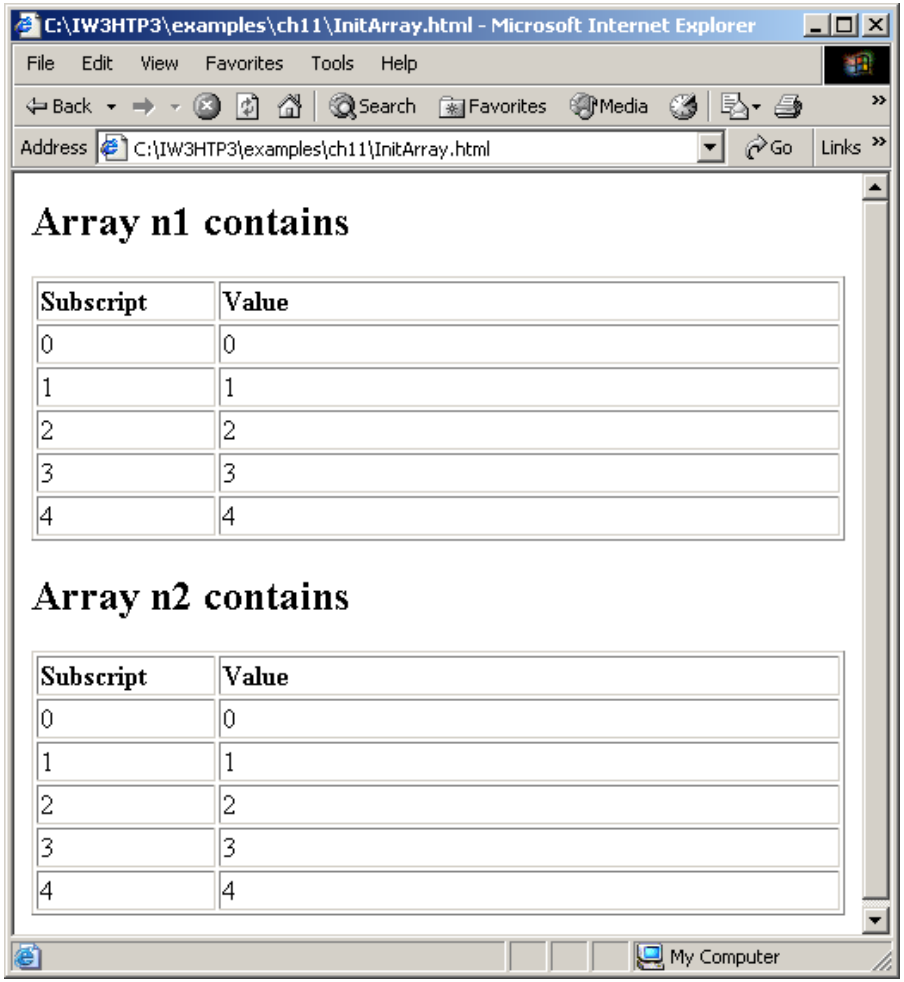
```
44
45          for ( var i = 0; i < theArray.length; i++ )
46             document.writeln( "<tr><td>" + i + "</td><td>" +
47                theArray[ i ] + "</td></tr>" );
48
49          document.writeln( "</tbody></table>" );
50       }
51       // -->
52    </script>
53
54 </head><body onload = "initializeArrays()"></body>
55 </html>
```

Outline

InitArray.html
(1 of 3)

11.4 Examples Using Arrays

Fig. 11.3 Initializing the elements of an array.



11.4 Examples Using Arrays

- Possible to declare and initialize in one step
  - Specify list of values
    - Initializer list

```
var n = [ 10, 20, 30, 40, 50 ];
var n = new Array( 10, 20, 30, 40, 50 );
```

  - Also possible to only initialize some values
    - Leave uninitialized elements blank
    - Uninitialized elements default to “undefined”

```
var n = [ 10, 20, , 40, 50 ];
```

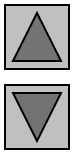
Outline

InitArray2.html
(1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.4: InitArray2.html                -->
<!-- Initializing an Array with a Declaration -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Initializing an Array with a Declaration</title>

      <script type = "text/javascript">
         <!--
         function start()
         {
            // Initializer list specifies n
            // value for each element.
            var colors = new Array( "cyan", "magenta",
               "yellow", "black" );
            var integers1 = [ 2, 4, 6, 8 ];
            var integers2 = [ 2, , , 8 ];

            outputArray( "Array colors contains", colors );
            outputArray( "Array integers1 contains", integers1 );
            outputArray( "Array integers2 contains", integers2 );
         }
```

Array `integers1` is initialized using an initializer list.

Two values are not supplied for `integers2`, which will be displayed as `undefined`.



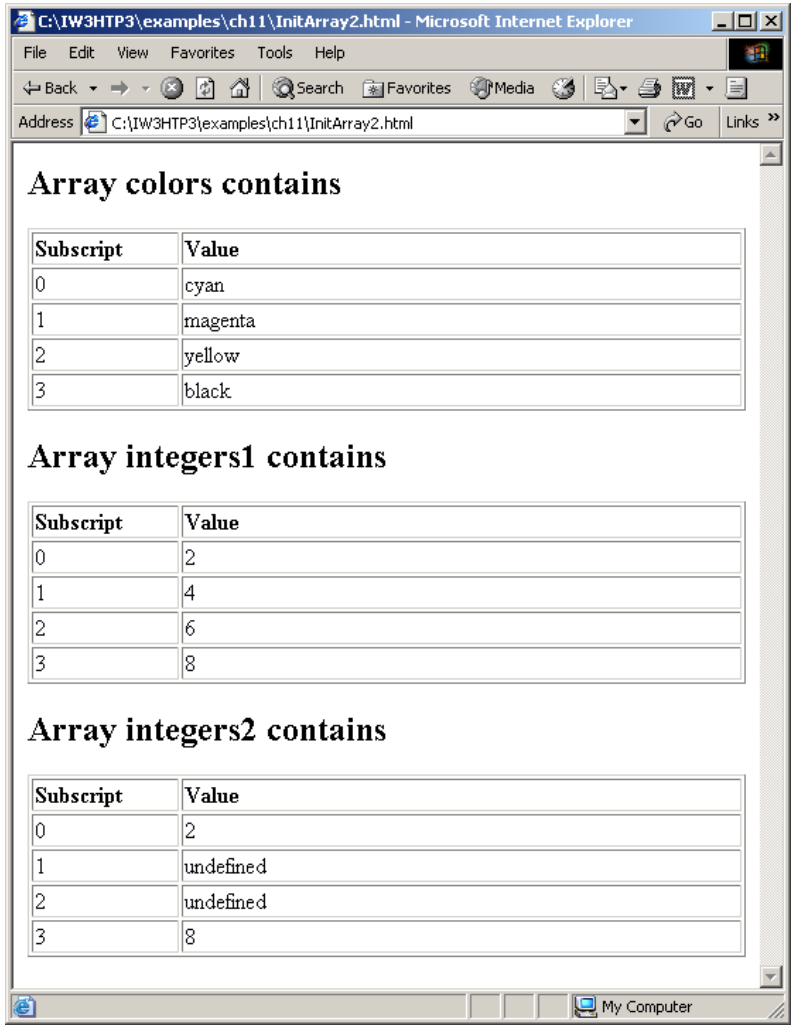
Outline

InitArray2.html
(2 of 2)

```
         // output "header" followed by a two-column table
         // containing subscripts and elements of "theArray"
         function outputArray( header, theArray )
         {
            document.writeln( "<h2>" + header + "</h2>" );
            document.writeln( "<table border = \"1\"" +
               "width = \"100%\">" );
            document.writeln( "<thead><th width = \"100\" " +
               "align = \"left\">Subscript</th>" +
               "<th align = \"left\">Value</th></thead><tbody>" );

            for ( var i = 0; i < theArray.length; i++ )
               document.writeln( "<tr><td>" + i + "</td><td>" +
                  theArray[ i ] + "</td></tr>" );

            document.writeln( "</tbody></table>" );
         }
         // -->
      </script>

   </head><body onload = "start()"></body>
</html>
```

11.4 Examples Using Arrays

Fig. 11.4 Initializing the elements of an array.



11.4 Examples Using Arrays

- `for...in` statement
  - Perform an action for each element in an array
  - Iterates over array elements
    - Assigns each element to specified variable one at a time
  - Ignores non-existent elements

Outline

SumArray.html
(1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.5: SumArray.html     -->
<!-- Summing Elements of an Array -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Sum the Elements of an Array</title>

      <script type = "text/javascript">
         <!--
         function start()
         {
            var theArray = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
            var total1 = 0, total2 = 0;

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

            document.writeln( "Total using subscripts: " + total1 );

```

The `for` loop sums the values contained in the 10-element integer array called `theArray`.

```
24            for ( var element in theArray )
25               total2 += theArray[ element ];
26
27            document.writeln( "<br />Total using for...in
28               total2 );
29         }
30         // -->
31      </script>
32
33   </head><body onload = "start()"></body>
34 </html>
```

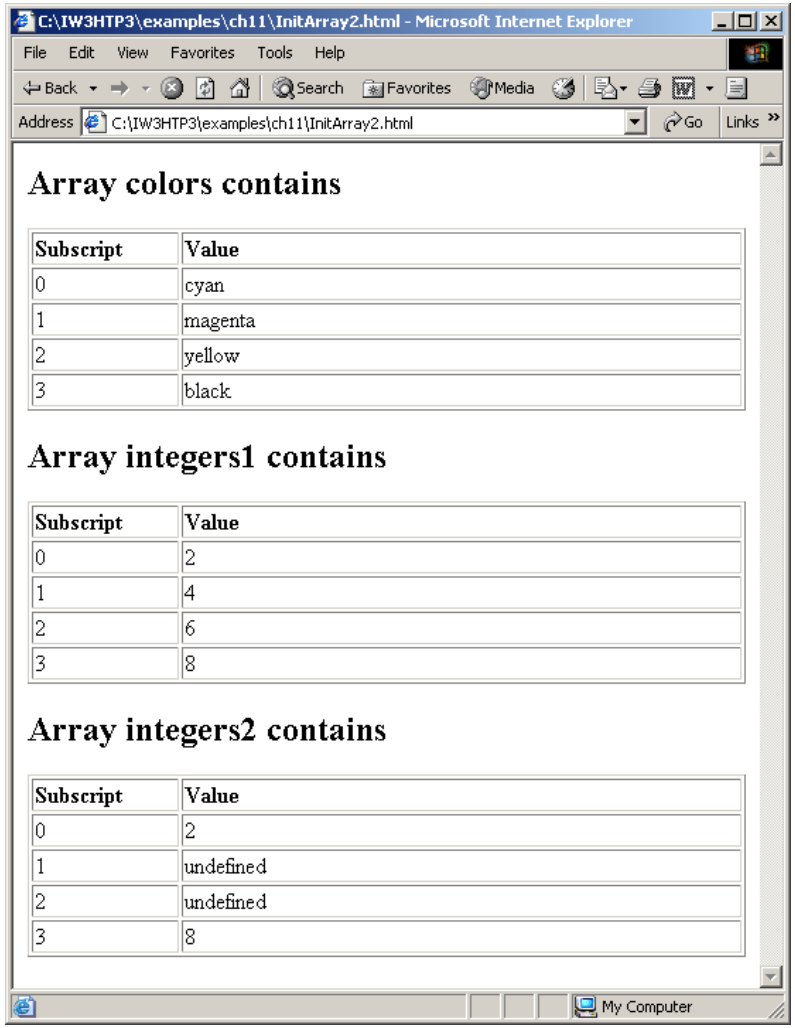
Variable `element` is assigned a subscript in the range of 0 up to, but not including, `theArray.length`.

Outline

(2 of 2)

11.4 Examples Using Arrays

Fig. 11.5 Calculating the sum of the elements of an array.



11.4 Examples Using Arrays

- Arrays can provide shorter and cleaner substitute for `switch` statements
  - Each element represents one case

Outline

RollDie.html
(1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.6: RollDie.html          -->
<!-- Roll a Six-Sided Die 6000 Times -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Roll a Six-Sided Die 6000 Times</title>

      <script type = "text/javascr
         <!--
         var face, frequency = [ , 0, 0, 0, 0, 0, 0 ];

         // summarize results
         for ( var roll = 1; roll <= 6000; ++roll ) {
            face = Math.floor( 1 + Math.random() * 6 );
            ++frequency[ face ];
         }

```

Referencing `Array` `frequency` replaces the `switch` statement used in Chapter 10's example.

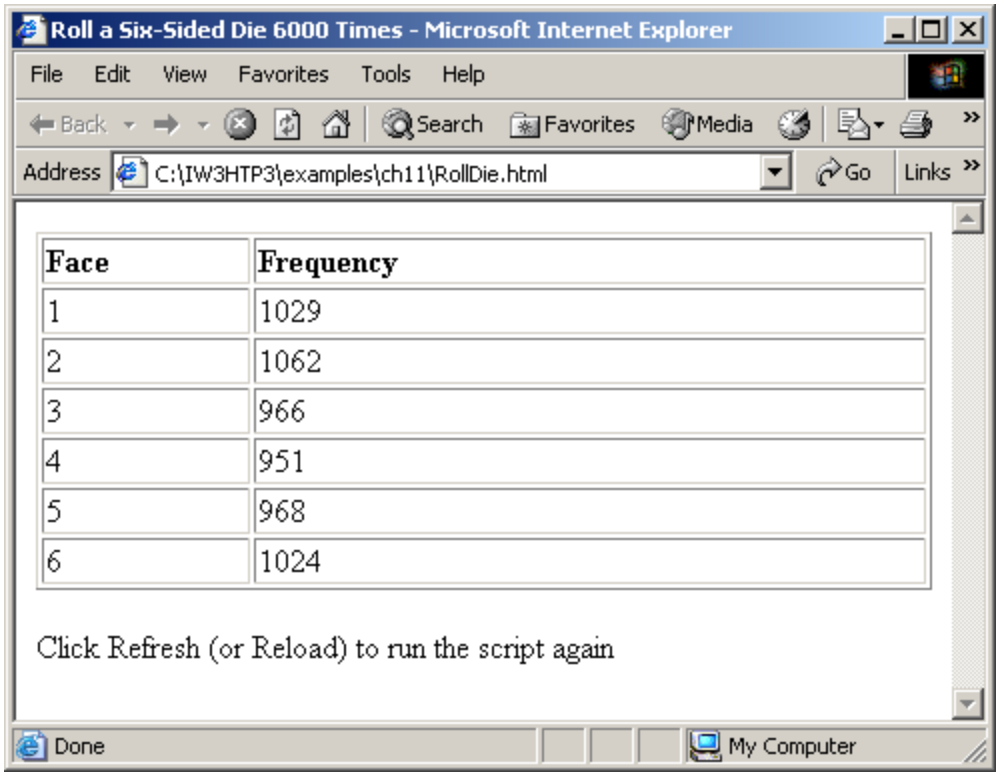
```
22          document.writeln( "<table border = \"1\""  +
23             "width = \"100%\">" );
24          document.writeln( "<thead><th width = \"100\"" +
25             " align = \"left\">Face<th align = \"left\">" +
26             "Frequency</th></thead></tbody>" );
27
28          for ( face = 1; face < frequency.length; ++face )
29             document.writeln( "<tr><td>" + face + "</td><td>" +
30                frequency[ face ] + "</td></tr>" );
31
32          document.writeln( "</tbody></table>" );
33          // -->
34       </script>
35
36    </head>
37    <body>
38       <p>Click Refresh (or Reload) to run the script again</p>
39    </body>
40 </html>
```

Outline

RollDie.html
(2 of 2)

11.4 Examples Using Arrays

Fig. 11.6 Dice-rolling program using arrays instead of a `switch`.



11.5 Random Image Generator Using Arrays

- Cleaner approach than previous version
  - Specify any file name rather than integers 1-7
  - Result of `Math.random` call is index into array of image file names

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 11.7: RandomPicture2.html    -->
<!-- Randomly displays one of 7 images -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Random Image Generator</title>

      <script type = "text/javascript">
         <!--
         var pictures =
            [ "CPE", "EPT", "GPP", "GUI", "PERF", "PORT", "SEO" ];
```

Outline

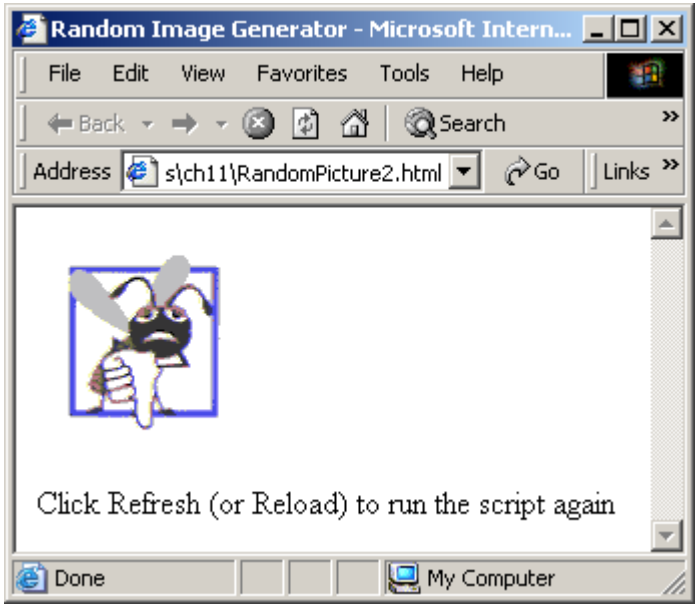
RandomPicture2.html
(1 of 2)

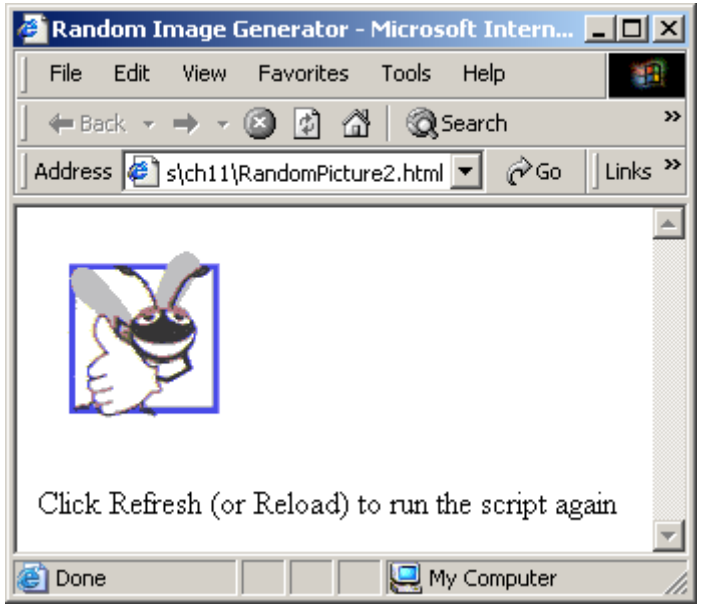
```
16
17          document.write ( "<img src = \"" +
18             pictures[ Math.floor( Math.random() * 7 ) ] +
19             ".gif\" width = \"105\" height = \"100\" />" );
20          // -->
21       </script>
22
23    </head>
24
25    <body>
26       <p>Click Refresh (or Reload) to run the script again</p>
27    </body>
28 </html>
```


RandomPicture2.html
(2 of 2)

11.5 Random Image Generator Using Arrays

Fig. 11.7 Random image generation using arrays.





11.6 References and Reference Parameters

- Two ways to pass parameters
  - Pass-by-value
    - Pass copy of original value
    - Default for numbers and booleans
    - Original variable is unchanged
  - Pass-by-reference
    - How objects are passed, like arrays
    - Pass location in memory of value
    - Allows direct access to original value
    - Improves performance

11.7 Passing Arrays to Functions

- Name of array is argument
  - Not necessary to also pass size of array
    - Arrays know their size
  - Passed by reference
    - Individual elements are passed by value if numbers or booleans
- `Array.join`
  - Creates string containing all array elements
  - Specify separator

Outline

PassArray.html
(1 of 3)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.8: PassArray.html -->
<!-- Passing Arrays            -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Passing Arrays and Individual Array
            Elements to Functions</title>

      <script type = "text/javascript">
         <!--
         function start()
         {
            var a = [ 1, 2, 3, 4, 5 ];

            document.writeln( "<h2>Effects of passing entire " +
               "array call-by-reference</
            outputArray(
               "The values of the original array are: ", a );

            modifyArray( a );  // array a passed call-by-reference

```

The first call to function `outputArray` displays the contents of the `Array` a before it is modified.

Function `modifyArray` multiplies each element by 2.

Outline

ssArray.html
(2 of 3)

```
26          outputArray(
27             "The values of the modified array are: ", a );
28
29          document.writeln( "<h2
30             "element call-by-va
31             "a[3] before modifyElement: " + a[ 3 ] );
32
33          modifyElement( a[ 3 ] );
34
35          document.writeln(
36             "<br />a[3] after modifyElement: " + a[ 3
37       }
38
39       // outputs "header" followed by the contents of "theArray"
40       function outputArray( header, theArray )
41       {
42          document.writeln(
43             header + theArray.join( " " ) + "<br />" );
44       }
45
```

Again, function `outputArray` is called to show that the contents of `Array` a have been modified.

Function `modifyElement` multiplies the contents of `a[ 3 ]` by 2.

The value of `a[ 3 ]` is output to show its contents before it is modified.

Method `join` takes as its argument a string containing a separator that should be used to separate the elements of the array in the string that is returned.

```
46          // function that modifies the elements of an array
47          function modifyArray( theArray )
48          {
49             for ( var j in theArray )
50                theArray[ j ] *= 2;
51          }
52
53          // function that attempts to modify the value passed
54          function modifyElement( e )
55          {
56             e *= 2;
57             document.writeln( "<br />value in modifyElement: " + e );
58          }
59          // -->
60       </script>
61
62    </head><body onload = "start()"></body>
63 </html>
```

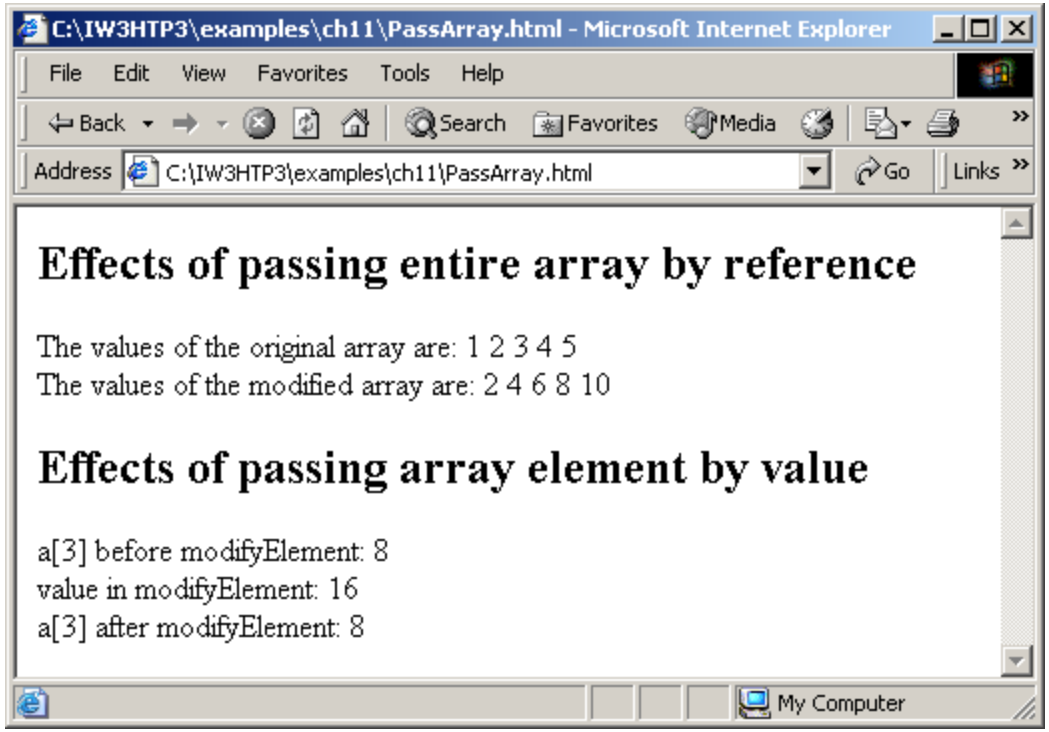
Multiply each element in `theArray` by 2.

Outline

PassArray.html
(3 of 3)

11.7 Passing Arrays to Functions

Fig. 11.8 Passing arrays and individual array elements to functions.



11.8 Sorting Arrays

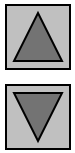
- Sorting
  - Important computing task
- `Array.sort`
  - Defaults to string comparison
  - Optional comparator function
    - Return negative if first argument less than second
    - Return zero if arguments equal
    - Return positive if first argument greater than second

Outline

Sort.html
(1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.9: sort.html -->
<!-- Sorting an Array     -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Sorting an Array with Array Method sort</title>

      <script type = "text/java
         <!--
         function start()
         {
            var a = [ 10, 1, 9, 2, 8, 3, 7, 4, 6, 5 ];

            document.writeln( "<h1>Sorting an Array</h1>" );
            outputArray( "Data items in original order: ", a );
            a.sort( compareIntegers );  // sort the array
            outputArray( "Data items in ascending order: ", a );
         }
```

Method `sort` takes as its optional argument the name of a function that compares two arguments and returns a value of –1, 0 or 1.



Outline

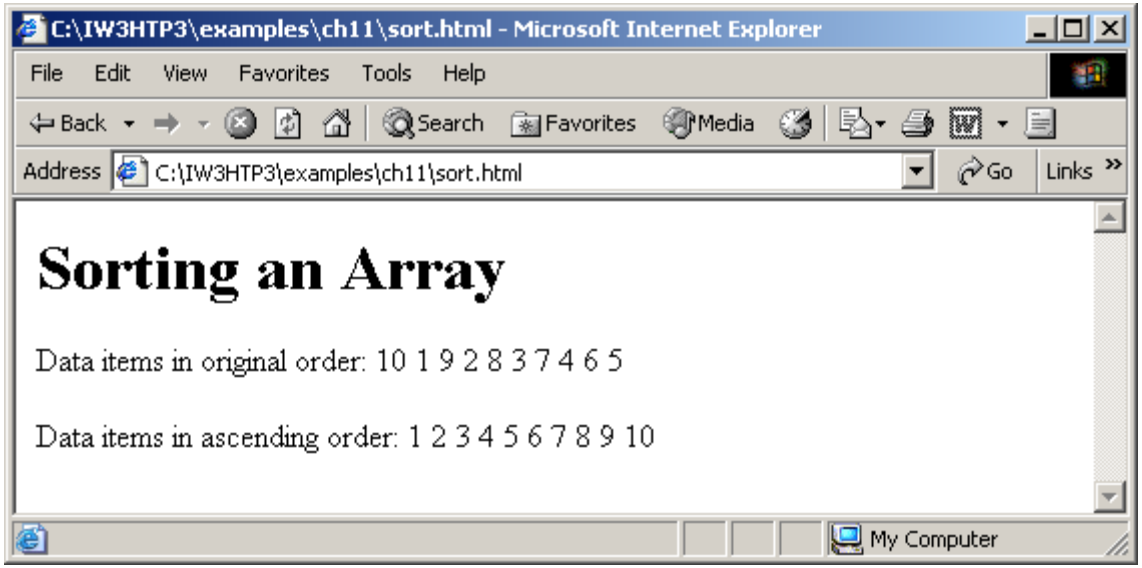
Sort.html
(2 of 2)

```
23
24          // outputs "header" followed by the contents of "theArray"
25          function outputArray( header, theArray )
26          {
27             document.writeln( "<p>" + header +
28                theArray.join( " " ) + "</p>" );
29          }
30
31          // comparison function for use with sort
32          function compareIntegers( value1, value2 )
33          {
34             return parseInt( value1 ) - parseInt( value2 );
35          }
36          // -->
37       </script>
38
39    </head><body onload = "start()"></body>
40 </html>
```

Function `compareIntegers` calculates the difference between the integer values of its arguments.

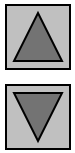
11.8 Sorting Arrays

Fig. 11.9 Sorting an array with `sort`.



11.9 Searching Arrays: Linear Search and Binary Search

- Searching
  - Look for matching key value
- Linear search
  - Iterate through each element until match found
  - Inefficient
    - Worst case scenario, must test entire array
- Binary search
  - Requires sorted data
  - Cuts search range in half each iteration
  - Efficient
    - Only look at small fraction of elements



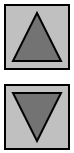
Outline

LinearSearch.html (1 of 3)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.10: LinearSearch.html -->
<!-- Linear Search of an Array     -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Linear Search of an Array</title>

      <script type = "text/javascript">
         <!--
         var a = new Array( 100 );  // create an Array

         // fill Array with even integer values from 0 to 198
         for ( var i = 0; i < a.length; ++i )
            a[ i ] = 2 * i;

```

Array a is initiated with 100 elements.

Array a is populated with the even integers 0 to 198.



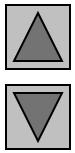
Outline

LinearSearch.html (2 of 3)

```
20      // function called when "Search" button is pressed
21      function buttonPressed()
22      {
23         var searchKey = searchForm.inputVal.value;
24
25         // Array a is passed to linearSearch even though it
26         // is a global variable. Normally a
27         // be passed to a method for search
28         var element = linearSearch( a, parseInt( searchKey ) );
29
30         if ( element != -1 )
31            searchForm.result.value =
32               "Found value in element
33         else
34            searchForm.result.value = "Value not found";
35      }
36
```

Get value of search key from the input field in the XHTML form.

Calling function `linearSearch` and passing it the `Array a` and the value of variable `searchKey` as an integer.



Outline

LinearSearch.html (3 of 3)

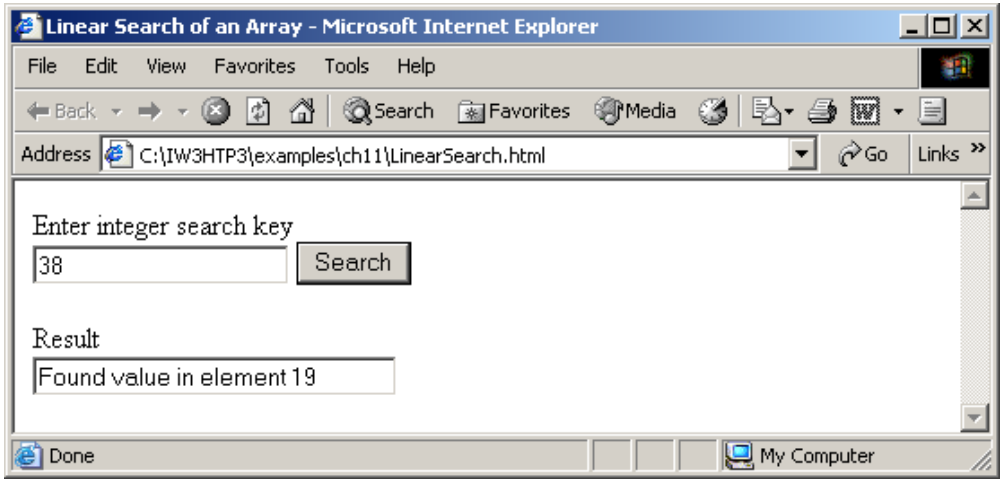
```
37         // Search "theArray" for the specified "key" value
38         function linearSearch( theArray, key )
39         {
40            for ( var n = 0; n < theArray.length; ++n )
41               if ( theArray[ n ] == key )
42                  return n;
43
44            return -1;
45         }
46         // -->
47      </script>
48
49   </head>
50
51   <body>
52      <form name = "searchForm" action  = "">
53         <p>Enter integer search key<br />
54         <input name = "inputVal" type = "text" />
55         <input name = "search" type = "button" value = "Search"
56                onclick = "buttonPressed()" /><br /></p>
57
58         <p>Result<br />
59         <input name = "result" type = "text" size = "30" /></p>
60      </form>
61   </body>
62 </html>
```

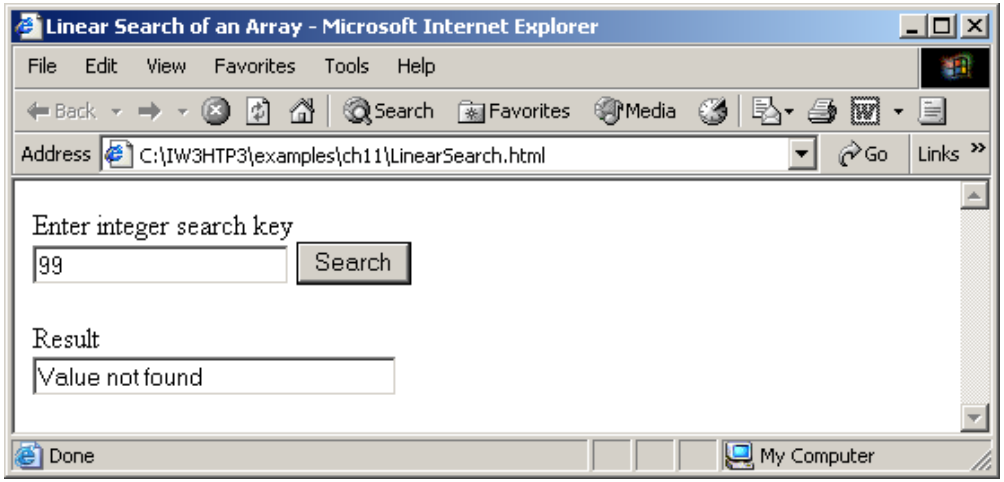
Variable `theArray` gets the value of `Array a` and variable `key` gets the value of variable `search`

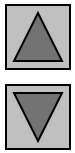
Function `linearSearch` compares each each element with a search key.

11.9 Searching Arrays: Linear Search and Binary Search

Fig. 11.10 Linear search of an array.







Outline

BinarySearch.html (1 of 5)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 11.11 : BinarySearch.html -->
<!-- Binary search                  -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Binary Search</title>

      <script type = "text/javascript">
         <!--
         var a = new Array( 15 );

         for ( var i = 0; i < a.length; ++i )
            a[ i ] = 2 * i;

```

`Array` a is initialized with 15 elements.

Outline

BinarySearch.html (2 of 5)

```
19         // function called when "Search" button is pressed
20         function buttonPressed()
21         {
22            var searchKey = searchForm.inputVal.value;
23
24            searchForm.result.value =
25               "Portions of array searched\n";
26
27            // Array a is passed to binarySearc
28            // is a global variable. This is do
29            // normally an array is passed to a method
30            // for searching.
31            var element =
32               binarySearch( a, parseInt( searchKey ) );
33
34            if ( element != -1 )
35               searchForm.result.value +=
36                  "\nFound value in element " + element;
37            else
38               searchForm.result.value += "\nValue not found";
39         }
40
```

Function `binarySearch` receives two arguments: the `Array` `a` and the search key, `searchKey`.

Outline

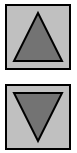
BinarySearch.html (3 of 5)

```
41         // Binary search
42         function binarySearch( theArray, key )
43         {
44            var low = 0;                     // low subscript
45            var high = theArray.length - 1; // high subscript
46            var middle;                      // middle subscript
47
48            while ( low <= high ) {
49               middle = ( low + high ) / 2;
50
51               // The following line is used to display the
52               // part of theArray currently be
53               // during each iteration of the
54               // search loop.
55               buildOutput( theArray, lo
56
57               if ( key == theArray[ middle ] )  // match
58                  return middle;
59               else if ( key < theArray
60                  high = middle - 1; /
61               else
62                  low = middle + 1; // search high end of array
63            }
```

If the `key` matches the `middle` element of a subarray, the subscript of the current element is

If `key` is less than the `middle` element, the high subscript is set to `middle` – 1.

If `key` is greater then the `middle` elements, the high subscript is set to `middle + 1`.



Outline

BinarySearch.html (4 of 5)

```
         return -1;   // searchKey not found
      }

      // Build one row of output showing the current
      // part of the array being processed.
      function buildOutput( theArray, low, mid, high )
      {
         for ( var i = 0; i < theArray.length; i++ ) {
            if ( i < low ||
               searchForm.re
            // mark middle element in output
            else if ( i == mid )
               searchForm.result.value += theArray[ i ] +
                  ( theArray[ i ] < 10 ? "*  " : "* " );
            else
               searchForm.result.value += theArray[ i ] +
                  ( theArray[ i ] < 10 ? "   " : "  " );
         }

         searchForm.result.value += "\n";
      }
      // -->
   </script>
</head>
```

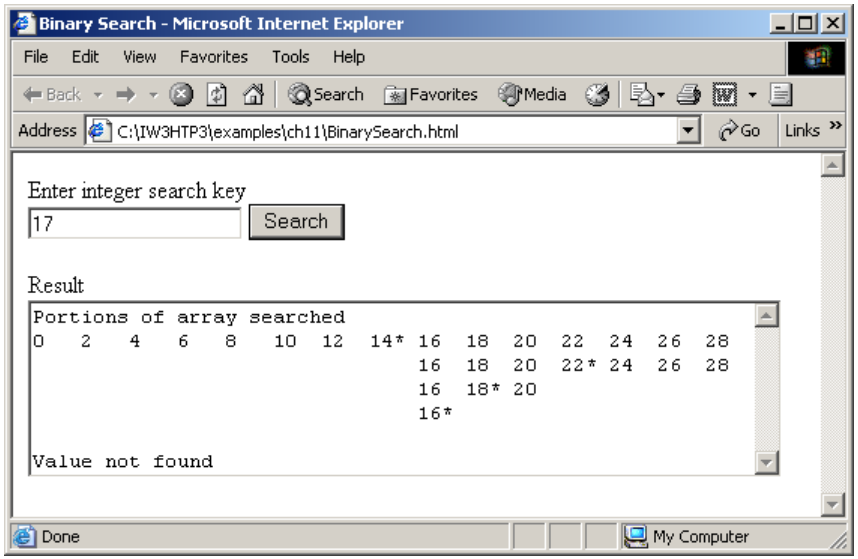
Function `buildOutput` creates the markup that displays the results of the search.

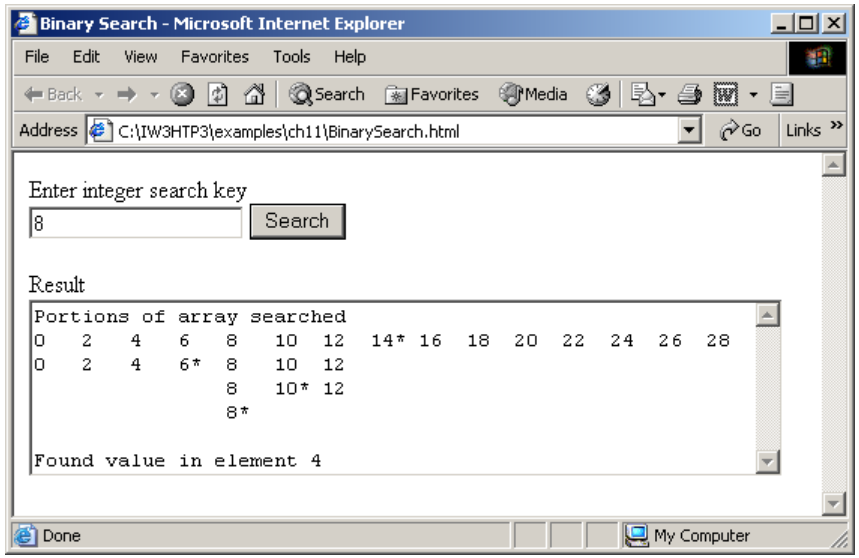
```
   <body>
      <form name = "searchForm" action = "">
         <p>Enter integer search key<br />
         <input name = "inputVal" type = "text" />
         <input name = "search" type = "button" value =
            "Search" onclick = "buttonPressed()" /><br /></p>
         <p>Result<br />
         <textarea name = "result" rows = "7" cols = "60">
         </textarea></p>
      </form>
   </body>
</html>
```


BinarySearch.html
(5 of 5)

11.9 Searching Arrays: Linear Search and Binary Search

Fig. 11.11 Binary search of an array.





11.10 Multidimensional Arrays

- Two-dimensional arrays analogous to tables
  - Rows and columns
    - Specify row first, then column
  - Two subscripts

11.10 Multidimensional Arrays

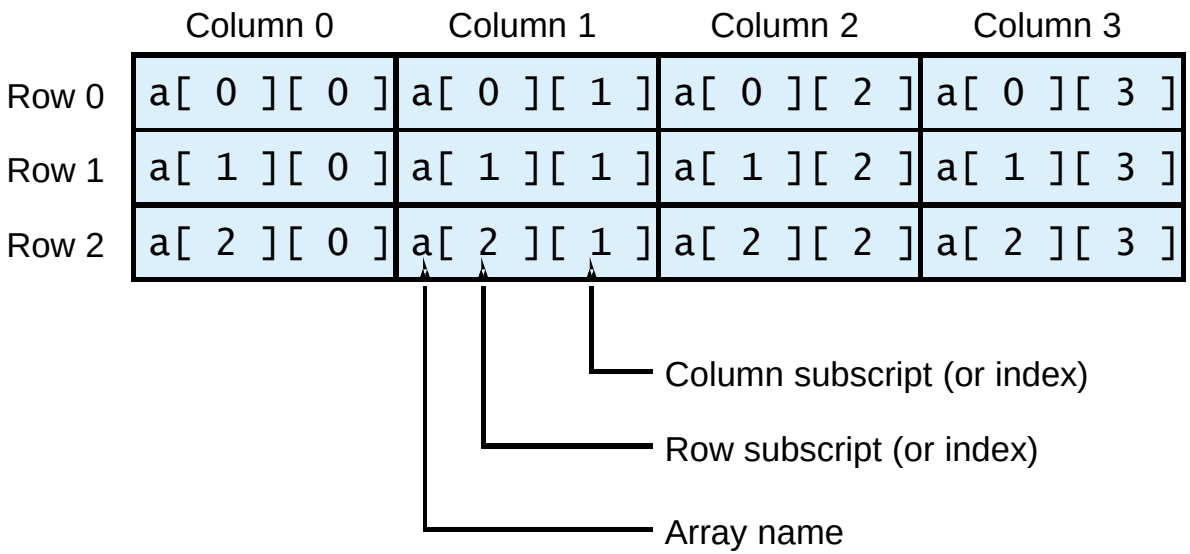


Fig. 11.12 Two-dimensional array with three rows and four columns.

11.10 Multidimensional Arrays

- Declaring and initializing multidimensional arrays
  - Group by row in square brackets
  - Treated as arrays of arrays
  - Creating array `b` with one row of two elements and a second row of three elements:

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

11.10 Multidimensional Arrays

- Also possible to use new operator
  - Create array b with two rows, first with five columns and second with three:

```
var b;

b = new Array( 2 );
b[ 0 ] = new Array( 5 );
b[ 1 ] = new Array( 3 );
```

Outline

InitArray3.html (1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.13: InitArray3.html          -->
<!-- Initializing Multidimensional Arrays -->

<html xmlns = "http://www.w3.org/19
   <head>
      <title>Initializing Multidimensional Arrays</title>

      <script type = "text/javascr
         <!--
         function start()
         {
            var array1 = [ [ 1, 2, 3 ],     // first row
                           [ 4, 5, 6 ] ];   // second row
            var array2 = [

                           [ 4, 5, 6 ] ];   // third row

            outputArray( "Values in array1 by row", array1 );
            outputArray( "Values in array2 by row", array2 );
         }
```

`Array array1` provides six initializers in two rows.

`Array array2` provides six initializers in three rows.

Function `outputArray` displays each array's elements in a Web page.

Outline

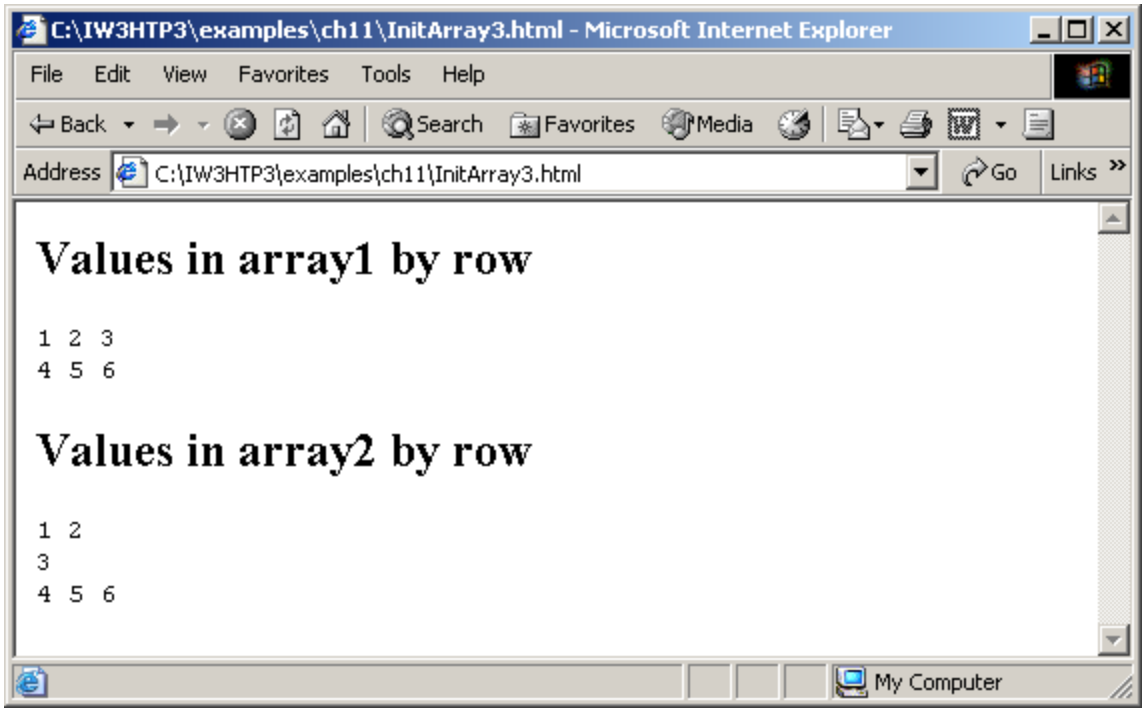
InitArray3.html
(2 of 2)

```
25
26          function outputArray( header, theArray )
27          {
28             document.writeln( "<h2>" + header + "</h2><tt>" );
29
30             for ( var i in theArray ) {
31
32                for ( var j in theArray[ i ] )
33                   document.write( theArray[ i ][ j ] + " " );
34
35                document.writeln( "<br />" );
36             }
37
38             document.writeln( "</tt>" );
39          }
40          // -->
41       </script>
42
43    </head><body onload = "start()"></body>
44 </html>
```

Referencing the multidimensional array `theArray`.

11.10 Multidimensional Arrays

Fig. 11.13 Initializing multidimensional arrays.



11.11 Building an Online Quiz

- Radio buttons
  - Represented as an array
    - Name of radio buttons is name of array
    - One element per button
  - `checked` property is `true` when selected
- XHTML Forms
  - Contain controls, including radio buttons
  - `action` property specifies what happens when submitted
    - Can call JavaScript code

Outline

Quiz.html
(1 of 2)

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- Fig. 11.14: quiz.html -->
<!-- Online Quiz           -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
 <title>Online Quiz</title>

 <script type = "text/JavaScript">

    function checkAnswers()
    {
       // determine whether the answer is correct
       if ( myQuiz.radiobutton[ 1 ].checked )
          document.write( "Congratulations, your answer is correct" );
       else // if the answer is incorrect
          document.write( "Your answer is incorre
    }

 </script>

 </head>
```

Determining the value of property checked.

Outline

Quiz.html
(2 of 2)

```
26
27 <body>
28    <form id = "myQuiz" action = "JavaScript:checkAnswers()">
29       <p>Select the name of the tip that goes with the image shown:<br />
30          <img src="EPT.gif" width="108" height="100" alt="mystery tip"/>
31          <br />
32
33          <input type = "radio" name = "radiobutton" value = "CPE" />
34          <label>Common Programming Error</label>
35
36          <input type = "radio" name = "radiobutton" value = "EPT" />
37          <label>Error-Prevention Tip</label>
38
39          <input type = "radio" name = "radiobutton" value = "PERF" />
40          <label>Performance Tip</label>
41
42          <input type = "radio" name = "radiobutton" value = "PORT" />
43          <label>Portability Tip</label><br />
44
45          <input type = "submit" name = "submit" value = "Submit" />
46          <input type = "reset" name = "reset" value = "Reset" />
47       </p>
48    </form>
49 </body>
50 </html>
```

Call the `checkAnswers` function when the form is submitted.

11.11 Building an Online Quiz

Fig. 11.14 Online quiz graded with JavaScript.