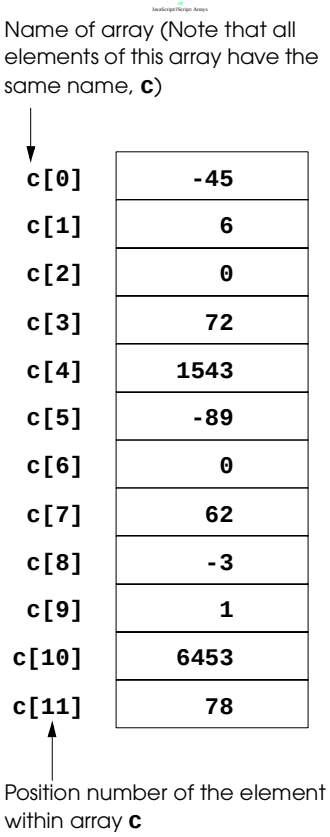


Fig. 12.1 A 12-element array.

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. 12.2 Precedence and associativity of the operators discussed so far.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.3: InitArray.html -->

<HEAD>
<TITLE>Initializing an Array</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   // this function is called when the <BODY> element's
   // ONLOAD event occurs
   function initializeArrays()
   {
      var n1 = new Array( 5 );   // allocate 5-element Array
      var n2 = new Array();      // allocate empty Array

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

      // create and initialize five-elements in Array n2
      for ( i = 0; i < 5; ++i )
         n2[ i ] = i;

      outputArray( "Array n1 contains", n1 );
      outputArray( "Array n2 contains", n2 );
   }

   // 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( "<TR><TD WIDTH = '100'><B>Subscript</B>"
                        + "<TD><B>Value</B></TR>" );

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

      document.writeln( "</TABLE>" );
   }
</SCRIPT>

</HEAD><BODY ONLOAD = "initializeArrays()"></BODY>
</HTML>
```

Fig. 12.3 Initializing the elements of an array (part 1 of 2).

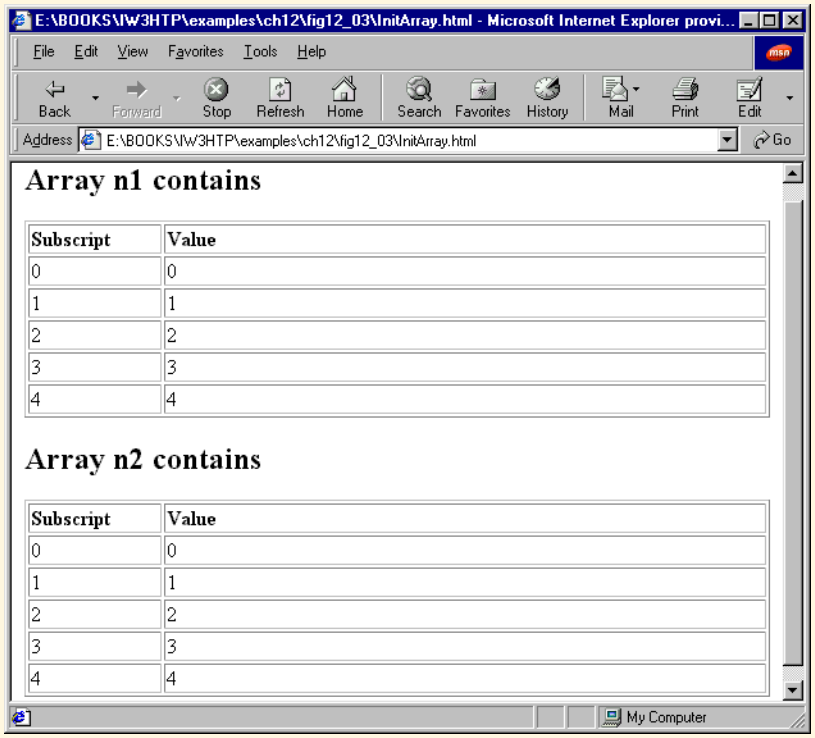


Fig. 12.3 Initializing the elements of an array (part 2 of 2).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.4: InitArray.html -->

<HEAD>
<TITLE>Initializing an Array with a Declaration</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   function start()
   {
      // Initializer list specifies number of elements and
      // 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 );
   }

   // 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( "<TR><TD WIDTH = '100'><B>Subscript</B>"
                  + "<TD><B>Value</B></TR>" );

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

      document.writeln( "</TABLE>" );
   }
</SCRIPT>

</HEAD><BODY ONLOAD = "start()"></BODY>
</HTML>
```

Fig. 12.4 Initializing the elements of an array with a declaration (part 1 of 2).

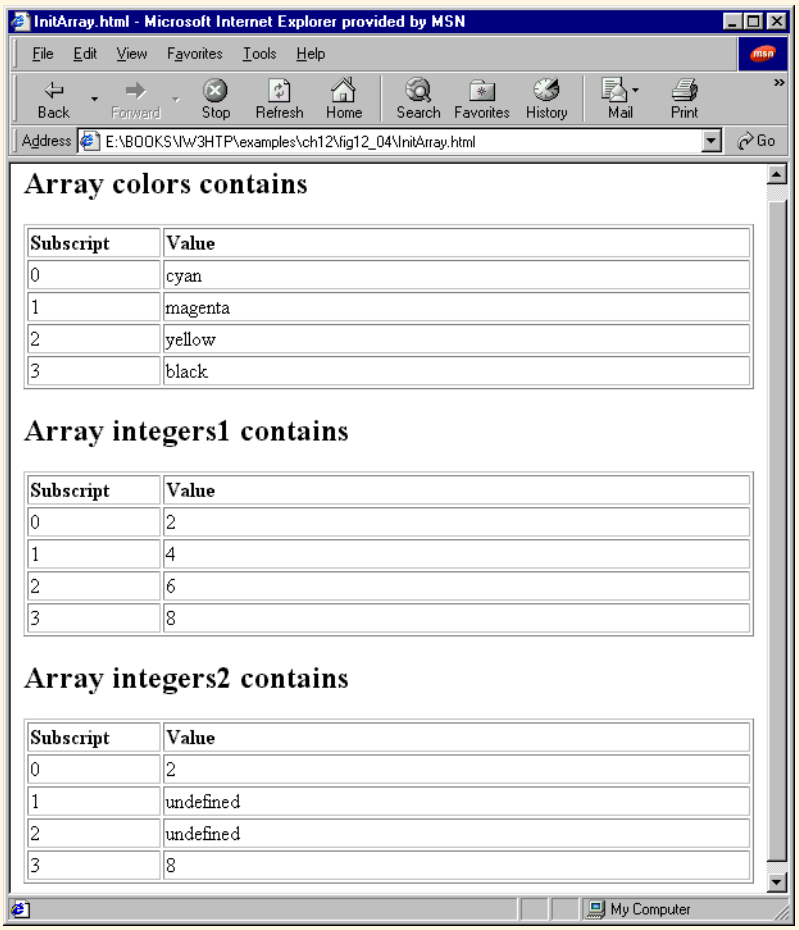


Fig. 12.4 Initializing the elements of an array with a declaration (part 2 of 2).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.5: SumArray.html -->

<HEAD>
<TITLE>Sum the Elements of an Array</TITLE>

<SCRIPT LANGUAGE = "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 );

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

      document.writeln( "<BR>Total using for/in: " + total2 );
   }
</SCRIPT>

</HEAD><BODY ONLOAD = "start()"></BODY>
</HTML>
```

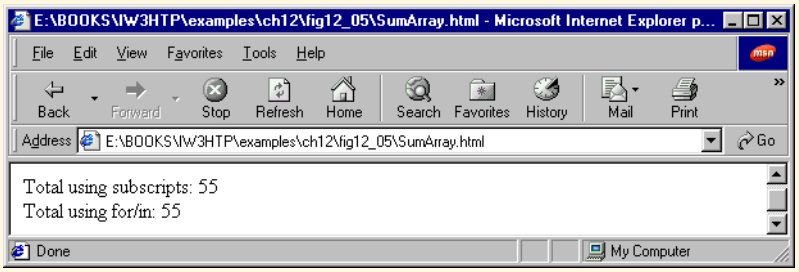
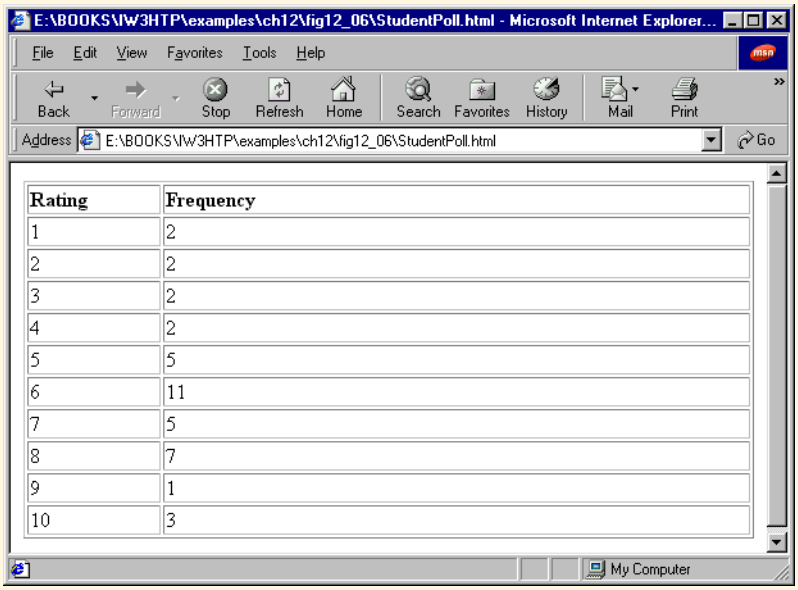


Fig. 12.5 Computing the sum of the elements of an array .

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.6: StudentPoll.html -->

<HEAD>
<TITLE>Student Poll Program</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   function start()
   {
      var responses = [ 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
                        1, 6, 3, 8, 6, 10, 3, 8, 2, 7,
                        6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
                        5, 6, 7, 5, 6, 4, 8, 6, 8, 10 ];
      var frequency = [ , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ];

      for ( var answer in responses )
         ++frequency[ responses[ answer ] ];

      document.writeln( "<TABLE BORDER = '1' WIDTH = '100%'>" );
      document.writeln( "<TR><TD WIDTH = '100'><B>Rating</B>" +
                        "<TD><B>Frequency</B></TR>" );

      for ( var rating = 1;
            rating < frequency.length; ++rating )
         document.writeln( "<TR><TD>" + rating + "<TD>" +
                           frequency[ rating ] + "</TR>" );

      document.writeln( "</TABLE>" );
   }
</SCRIPT>
</HEAD><BODY ONLOAD = "start()"></BODY>
</HTML>
```

Fig. 12.6 A simple student-poll analysis program (part 1 of 2).



Rating	Frequency
1	2
2	2
3	2
4	2
5	5
6	11
7	5
8	7
9	1
10	3

Fig. 12.6 A simple student-poll analysis program (part 2 of 2).

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2  <HTML>
3  <!-- Fig. 12.8: RollDie.html -->
4
5  <HEAD>
6  <TITLE>Roll a Six-Sided Die 6000 Times</TITLE>
7
8  <SCRIPT LANGUAGE = "JavaScript">
9     var face, frequency = [ , 0, 0, 0, 0, 0, 0 ];
10
11    // summarize results
12    for ( var roll = 1; roll <= 6000; ++roll ) {
13       face = Math.floor( 1 + Math.random() * 6 );
14       ++frequency[ face ];
15    }
16
17    document.writeln( "<TABLE BORDER = '1' WIDTH = '100%'>" );
18    document.writeln( "<TR><TD WIDTH = '100'><B>Face</B>" +
19       "<TD><B>Frequency</B></TR>" );
20
21    for ( face = 1; face < frequency.length; ++face )
22       document.writeln( "<TR><TD>" + face + "<TD>" +
23                         frequency[ face ] + "</TR>" );
24
25    document.writeln( "</TABLE>" );
26 </SCRIPT>
27
28 </HEAD>
29 <BODY>
30 <P>Click Refresh (or Reload) to run the script again</P>
31 </BODY>
32 </HTML>
```

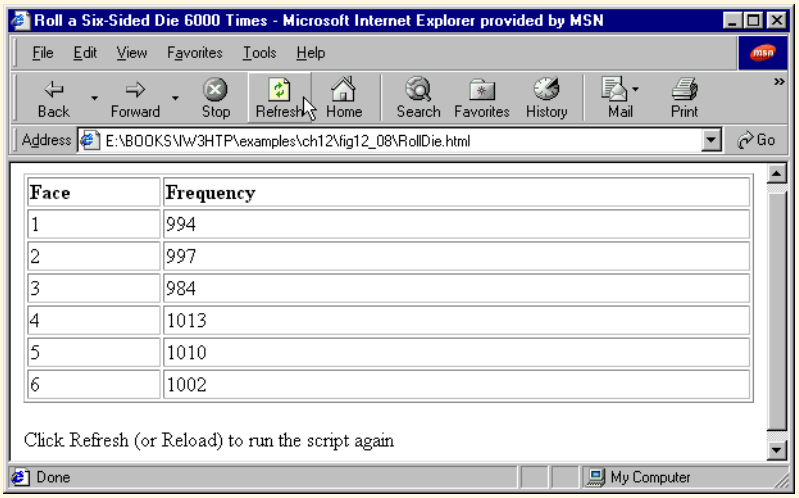


Fig. 12.7 Dice-rolling program using arrays instead of `switch`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.9: PassArray.html -->

<HEAD>
<TITLE>Passing Arrays and Individual Array
       Elements to Functions</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   function start()
   {
      var a = [ 1, 2, 3, 4, 5 ];

      document.writeln( "<H2>Effects of passing entire " +
                        "array call-by-reference</H2>" );
      outputArray(
         "The values of the original array are: ", a );

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

      outputArray(
         "The values of the modified array are: ", a );

      document.writeln( "<H2>Effects of passing array " +
         "element call-by-value</H2>" +
         "a[3] before modifyElement: " + a[ 3 ] );

      modifyElement( a[ 3 ] );

      document.writeln(
         "<BR>a[3] after modifyElement: " + a[ 3 ] );
   }

   // outputs "header" followed by the contents of "theArray"
   function outputArray( header, theArray )
   {
      document.writeln(
         header + theArray.join( " " ) + "<BR>" );
   }

   // function that modifies the elements of an array
   function modifyArray( theArray )
   {
      for ( var j in theArray )
         theArray[ j ] *= 2;
   }

   // function that attempts to modify the value passed
   function modifyElement( e )
   {
      e *= 2;
      document.writeln( "<BR>value in modifyElement: " + e );
```

Fig. 12.8 Passing arrays and individual array elements to functions (part 1 of 2).

```
53      }
54   </SCRIPT>
55
56   </HEAD><BODY ONLOAD = "start()"></BODY>
57   </HTML>
```

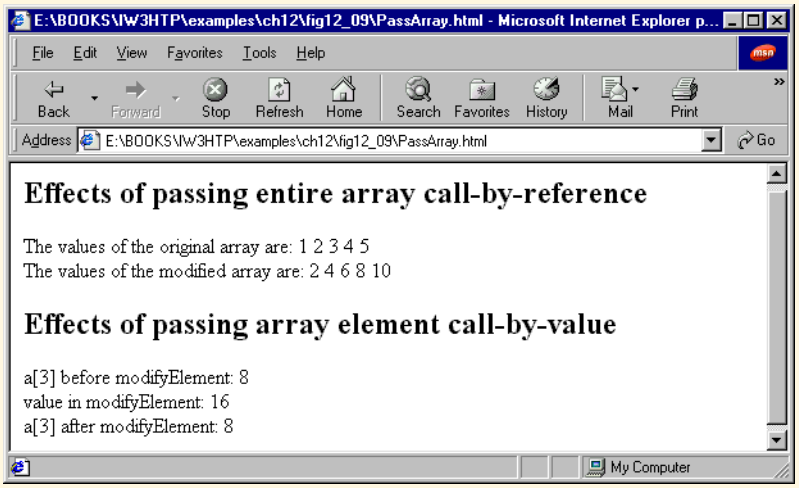


Fig. 12.8 Passing arrays and individual array elements to functions (part 2 of 2).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.10: BubbleSort.html -->

<HEAD>
<TITLE>Sorting an Array with Bubble Sort</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   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 );
      bubbleSort( a );  // sort the array
      outputArray( "Data items in ascending order: ", a );
   }

   // outputs "header" followed by the contents of "theArray"
   function outputArray( header, theArray )
   {
      document.writeln(
         "<P>" + header + theArray.join( " " ) + "</P>" );
   }

   // sort the elements of an array with bubble sort
   function bubbleSort( theArray )
   {
      // control number of passes of theArray
      for ( var pass = 1; pass < theArray.length; ++pass )

         // one pass - control's number of comparison per pass
         for ( var i = 0; i < theArray.length - 1; ++i )

            // perform one comparison
            if ( theArray[ i ] > theArray[ i + 1 ] )
               swap( theArray, i, i + 1 );  // swap elements
   }

   // swap two elements of an array
   function swap( theArray, first, second )
   {
      var hold;  // temporary holding area for swap

      hold = theArray[ first ];
      theArray[ first ] = theArray[ second ];
      theArray[ second ] = hold;
   }
</SCRIPT>

</HEAD><BODY ONLOAD = "start()"></BODY>
</HTML>
```

Fig. 12.9 Sorting an array with bubble sort (part 1 of 2).

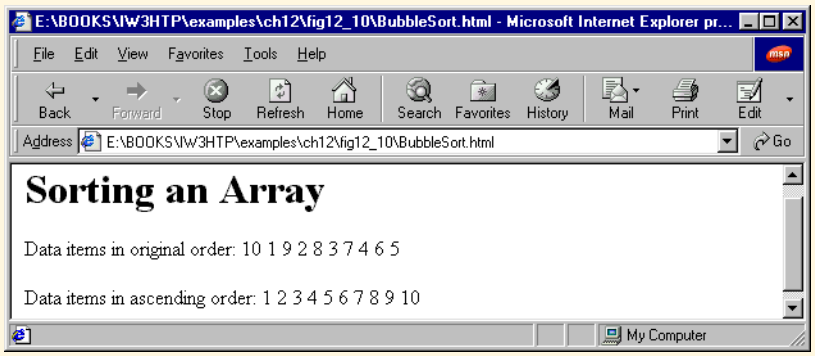


Fig. 12.9 Sorting an array with bubble sort (part 2 of 2).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.11: sort.html -->

<HEAD>
<TITLE>Sorting an Array with Array Method sort</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   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 );
   }

   // outputs "header" followed by the contents of "theArray"
   function outputArray( header, theArray )
   {
      document.writeln( "<P>" + header +
         theArray.join( " " ) + "</P>" );
   }

   // comparison function for use with sort
   function compareIntegers( value1, value2 )
   {
      return parseInt( value1 ) - parseInt( value2 );
   }
</SCRIPT>

</HEAD><BODY ONLOAD = "start()"></BODY>
</HTML>
```

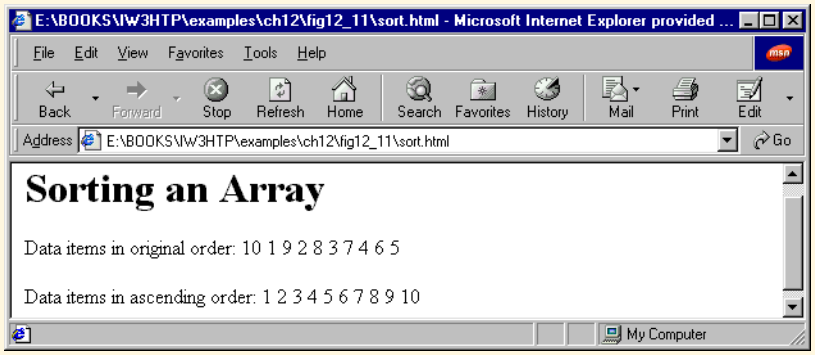
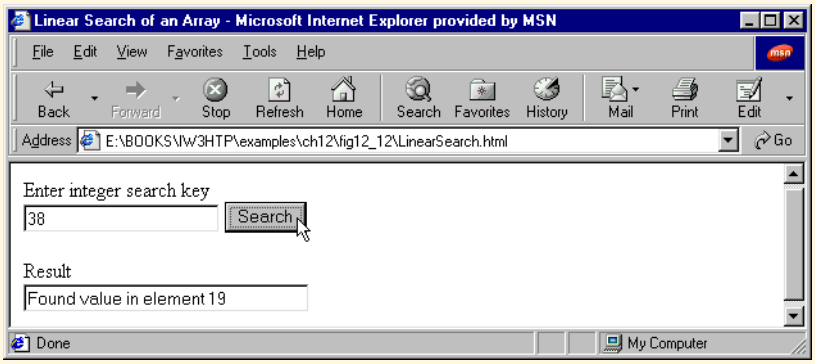


Fig. 12.10 Sorting an array with `sort`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.12: LinearSearch.html -->

<HEAD>
<TITLE>Linear Search of an Array</TITLE>

<SCRIPT LANGUAGE = "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;

   // function called when "Search" button is pressed
   function buttonPressed()
   {
      var searchKey = searchForm.inputVal.value;

      // Array a is passed to linearSearch even though it
      // is a global variable. Normally an array will
      // be passed to a method for searching.
      var element = linearSearch( a, parseInt( searchKey ) );

      if ( element != -1 )
         searchForm.result.value =
            "Found value in element " + element;
      else
         searchForm.result.value = "Value not found";
   }

   // Search "theArray" for the specified "key" value
   function linearSearch( theArray, key )
   {
      for ( var n = 0; n < theArray.length; ++n )
         if ( theArray[ n ] == key )
            return n;

      return -1;
   }
</SCRIPT>

</HEAD>

<BODY>
<FORM NAME = "searchForm">
   <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>
```

Fig. 12.11 Linear search of an array (part 1 of 2).

```
53      <INPUT NAME = "result" TYPE = "text" SIZE = "30"></P>
54   </FORM>
55   </BODY>
56   </HTML>
```



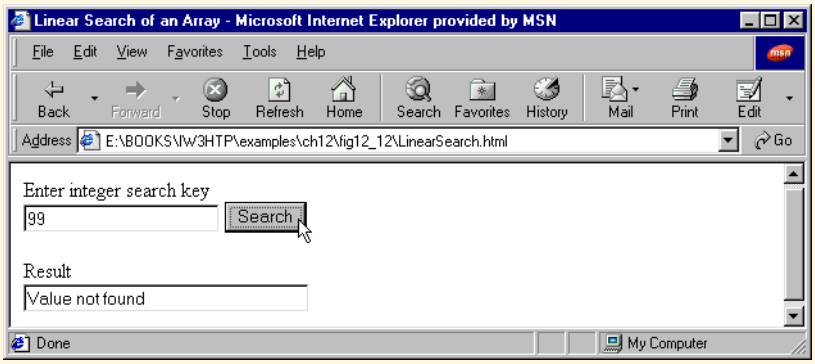


Fig. 12.11 Linear search of an array (part 2 of 2).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.13: BinarySearch.html -->

<HEAD>
<TITLE>Binary Search of an Array</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   var a = new Array( 15 );

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

   // function called when "Search" button is pressed
   function buttonPressed()
   {
      var searchKey = searchForm.inputVal.value;

      searchForm.result.value =
         "Portions of array searched\n";

      // Array a is passed to binarySearch even though it
      // is a global variable. This is done because normally
      // an array is passed to a method for searching.
      var element = binarySearch( a, parseInt( searchKey ) );

      if ( element != -1 )
         searchForm.result.value +=
            "\nFound value in element " + element;
      else
         searchForm.result.value += "\nValue not found";
   }

   // Binary search
   function binarySearch( theArray, key )
   {
      var low = 0;                    // low subscript
      var high = theArray.length - 1; // high subscript
      var middle;                     // middle subscript

      while ( low <= high ) {
         middle = ( low + high ) / 2;

         // The following line is used to display the part
         // of theArray currently being manipulated during
         // each iteration of the binary search loop.
         buildOutput( theArray, low, middle, high );

         if ( key == theArray[ middle ] )  // match
            return middle;
         else if ( key < theArray[ middle ] )
            high = middle - 1;  // search low end of array
```

Fig. 12.12 Binary search of a sorted array (part 1 of 3).

```
53             else
54                low = middle + 1;   // search high end of array
55          }
56
57          return -1;   // searchKey not found
58       }
59
60       // Build one row of output showing the current
61       // part of the array being processed.
62       function buildOutput( theArray, low, mid, high )
63       {
64          for ( var i = 0; i < theArray.length; i++ ) {
65             if ( i < low || i > high )
66                searchForm.result.value += "    ";
67             else if ( i == mid ) // mark middle element in output
68                searchForm.result.value += a[ i ] +
69                   ( theArray[ i ] < 10 ? "*  " : "* " );
70             else
71                searchForm.result.value += a[ i ] +
72                   ( theArray[ i ] < 10 ? "   " : "  " );
73          }
74
75          searchForm.result.value += "\n";
76       }
77    </SCRIPT>
78
79    </HEAD>
80
81    <BODY>
82    <FORM NAME = "searchForm">
83       <P>Enter integer search key<BR>
84       <INPUT NAME = "inputVal" TYPE = "text">
85       <INPUT NAME = "search" TYPE = "button" VALUE = "Search"
86              ONCLICK = "buttonPressed()"><BR></P>
87       <P>Result<BR><TEXTAREA NAME = "result" ROWS = "7" COLS = "60">
88                   </TEXTAREA></P>
89    </FORM>
90    </BODY>
91    </HTML>
```

Fig. 12.12 Binary search of a sorted array (part 2 of 3).

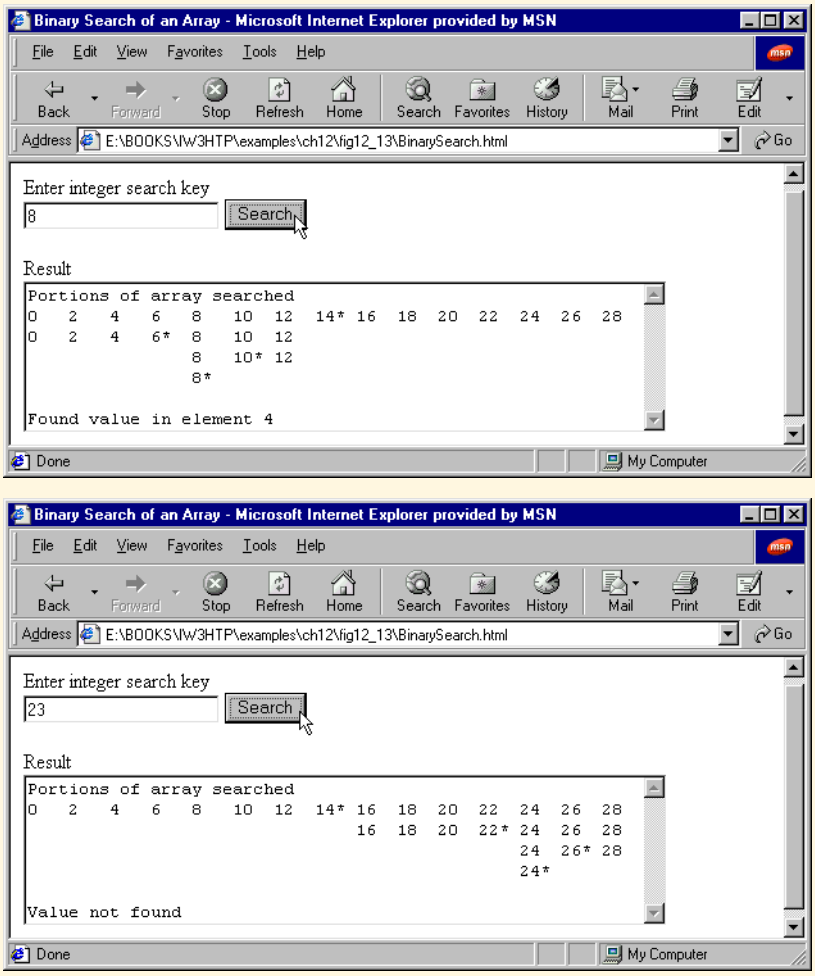


Fig. 12.12 Binary search of a sorted array (part 3 of 3).

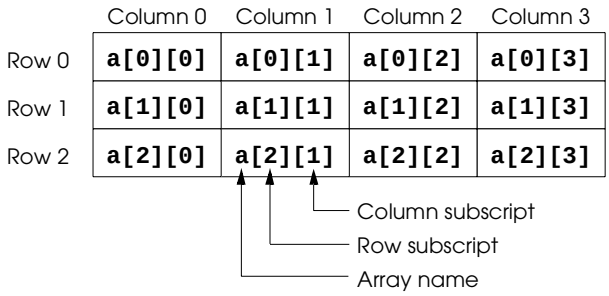


Fig. 12.13 A double-subscripted array with three rows and four columns.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.15: InitArray.html -->

<HEAD>
<TITLE>Initializing Multidimensional Arrays</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   function start()
   {
      var array1 = [ [ 1, 2, 3 ],     // first row
                     [ 4, 5, 6 ] ];   // second row
      var array2 = [ [ 1, 2 ],        // first row
                     [ 3 ],           // second row
                     [ 4, 5, 6 ] ];   // third row

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

   function outputArray( header, theArray )
   {
      document.writeln( "<H2>" + header + "</H2><TT>" );

      for ( var i in theArray ) {

         for ( var j in theArray[ i ] )
            document.write( theArray[ i ][ j ] + " " );

         document.writeln( "<BR>" );
      }

      document.writeln( "</TT>" );
   }
</SCRIPT>

</HEAD><BODY ONLOAD = "start()"></BODY>
</HTML>
```

Fig. 12.14 Initializing multidimensional arrays (part 1 of 2).

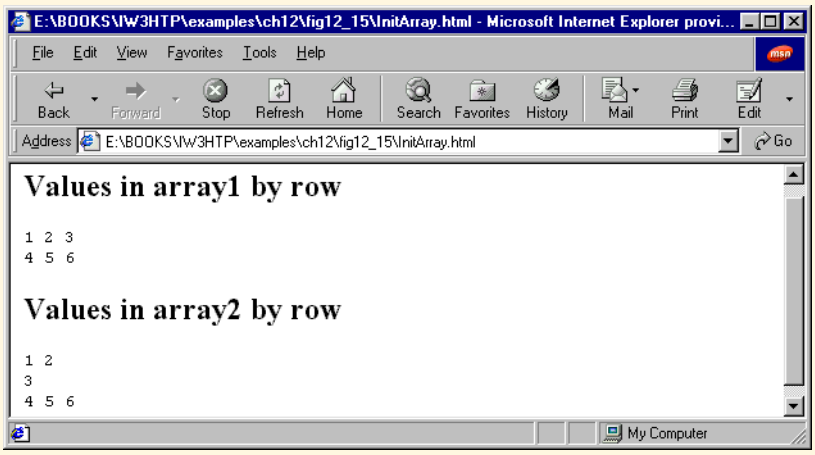


Fig. 12.14 Initializing multidimensional arrays (part 2 of 2).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<!-- Fig. 12.16: DoubleArray.html -->

<HEAD>
<TITLE>Double-subscripted Array Example</TITLE>

<SCRIPT LANGUAGE = "JavaScript">
   function start()
   {
      var grades = [ [ 77, 68, 86, 73 ],
                     [ 96, 87, 89, 81 ],
                     [ 70, 90, 86, 81 ] ];

      document.writeln( "<PRE>" );
      outputArray( grades );

      document.writeln(
         "\n\nLowest grade: " + minimum( grades ) +
         "\nHighest grade: " + maximum( grades ) );

      // calculate average for each student (i.e., each row)
      for ( var i in grades )
         document.write( "\nAverage for student " + i +
            " is " + average( grades[ i ] ) );

      document.writeln( "</PRE>" );
   }

   // find the minimum grade
   function minimum( grades )
   {
      var lowGrade = 100;

      for ( var i in grades )
         for ( var j in grades[ i ] )
            if ( grades[ i ][ j ] < lowGrade )
               lowGrade = grades[ i ][ j ];

      return lowGrade;
   }

   // find the maximum grade
   function maximum( grades )
   {
      var highGrade = 0;

      for ( var i in grades )
         for ( var j in grades[ i ] )
            if ( grades[ i ][ j ] > highGrade )
               highGrade = grades[ i ][ j ];

```

Fig. 12.15 Example of using double-subscripted arrays (part 1 of 3).

```
53          return highGrade;
54       }
55
56       // determine the average grade for a particular
57       // student (or set of grades)
58       function average( setOfGrades )
59       {
60          var total = 0;
1.
2.       for ( var i in setOfGrades )
3.          total += setOfGrades[ i ];
4.
5.       return total / setOfGrades.length;
6.    }
7.
8.    // output grades
9.    function outputArray( grades )
10.   {
11.      document.write( "          " ); // align heads
12.
61          for ( var i in grades[ 0 ] )
62             document.write( "[" + i + "]  " );
63
64          for ( var i in grades ) {
65             document.write( "<BR>grades[" + i + "]   " );
66
67             for ( var j in grades[ i ] )
68                document.write( grades[ i ][ j ] + "   " );
69          }
70       }
71    </SCRIPT>
72
73    </HEAD><BODY ONLOAD = "start()"></BODY>
74    </HTML>
```

Fig. 12.15 Example of using double-subscripted arrays (part 2 of 3).

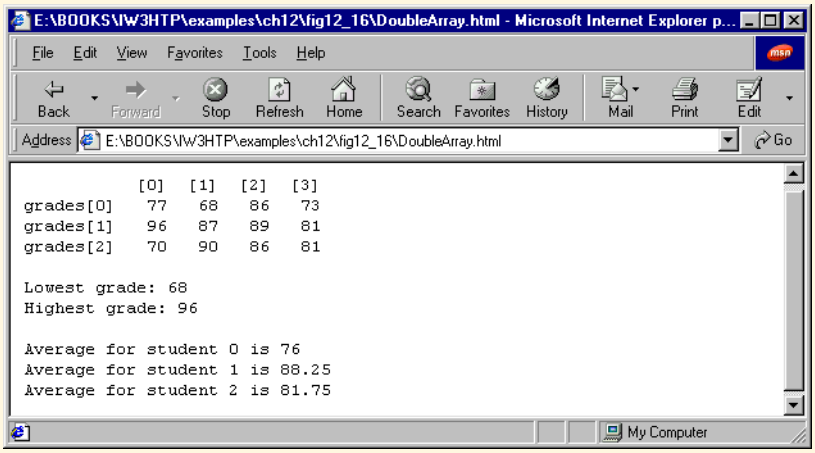


Fig. 12.15 Example of using double-subscripted arrays (part 3 of 3).