

IPA
1st Semester, 2007-2008

Internet 1

Ch. 12

JavaScript: Arrays

attasr@ipa.edu.sa



Introduction

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

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

Arrays

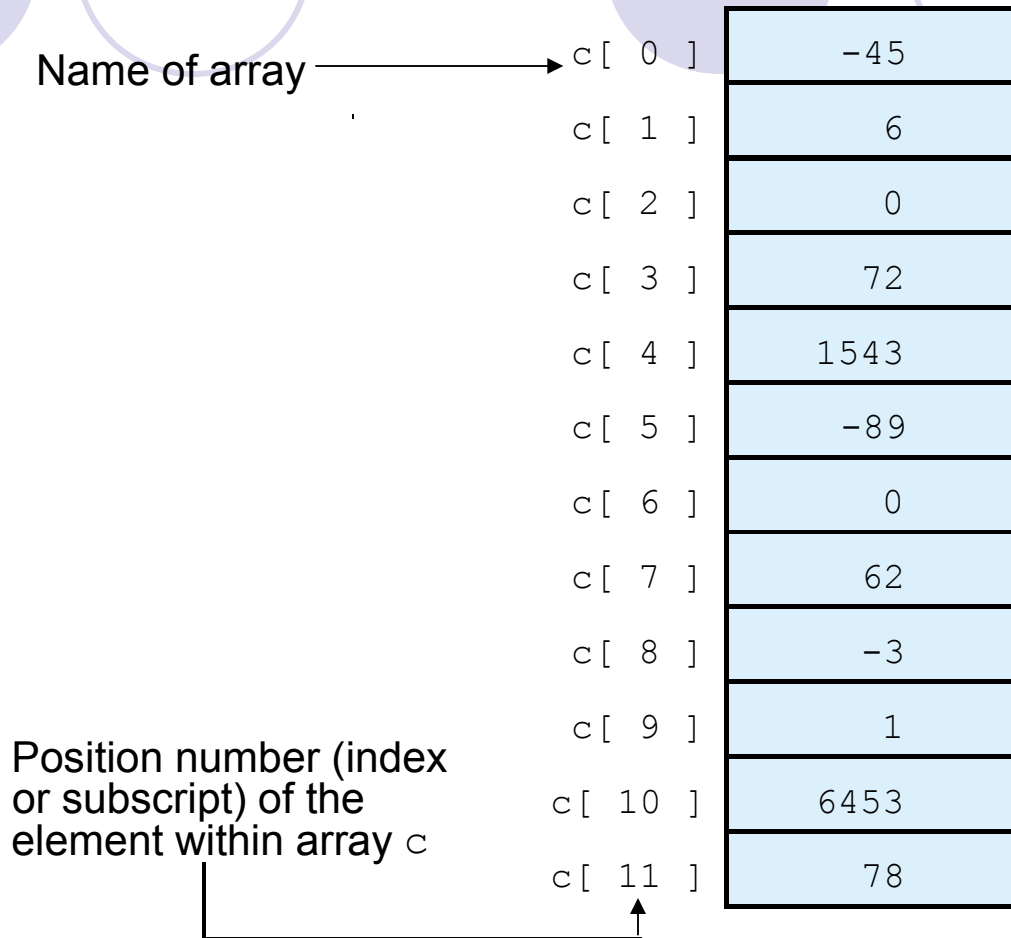


Fig. 11.1 A 12-element array.

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.

Declaring and Allocating Arrays

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

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

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

```

1  <?xml version = "1.0"?>
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. 11.3: InitArray.html -->
6  <!-- Initializing an Array      -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9      <head>
10         <title>Initializing an Array</title>
11
12         <script type = "text/javascript">
13             <!--
14             // this function is called when the <body> element
15             // onload event occurs
16             function initializeArrays()
17             {
18                 var n1 = new Array( 5 );    // allocate array n1
19                 var n2 = new Array();       // allocate array n2
20
21                 // assign values to each element of Array n1
22                 for ( var i = 0; i < n1.length; ++i )
23                     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).


```
24 // create and initialize five-elements in Array n2
```

```
26 for ( i = 0; i < 5; ++i )
```

```
27     n2[ i ] = i;
```

The for loop adds
each element to its s

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

```
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 {
```

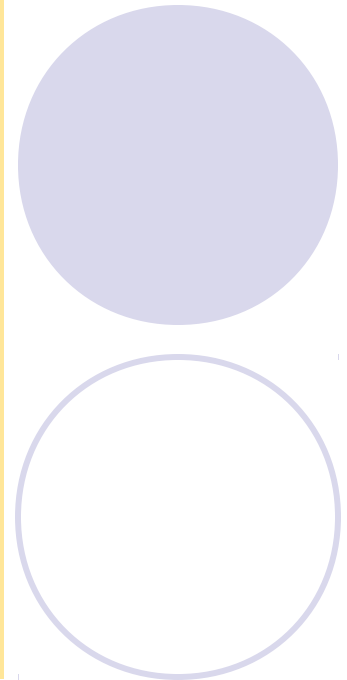
```
37     document.writeln( "<h2>" + header + "</h2>" );
```

```
38     doc  
39  
40  
41     doc  
42  
43
```

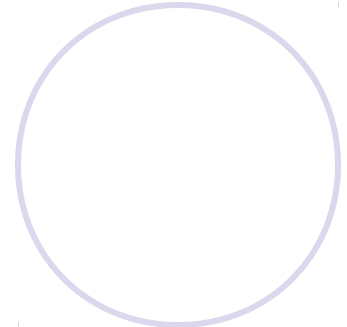
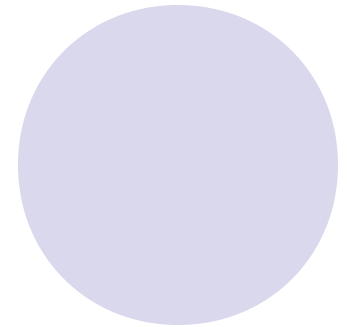
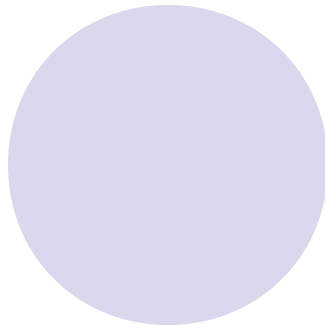
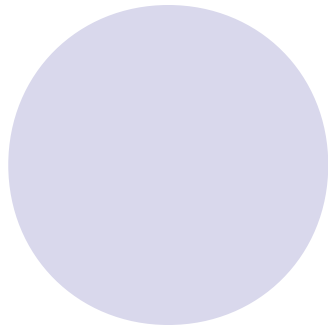
The second time function outputArray is
called, variable header gets the value of
"Array n2 contains" and variable
theArray gets the value of n2.

```
    align = \ "left\ ">Subscript</th>
```

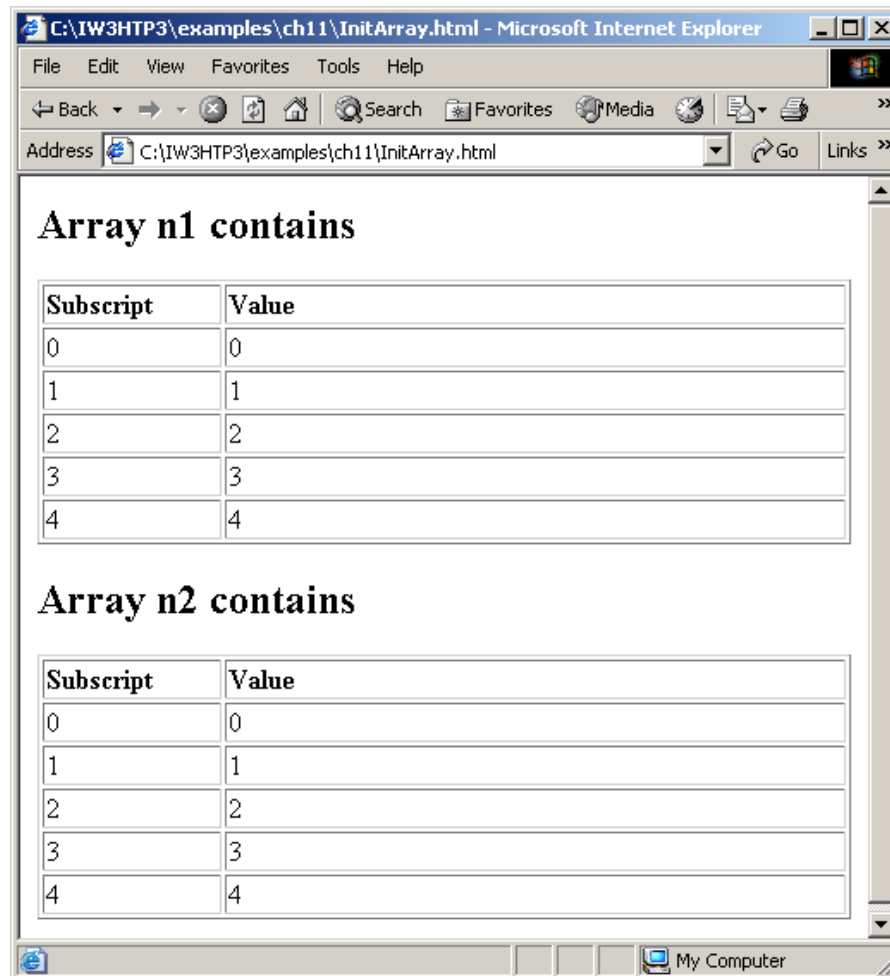
```
    "<th align = \ "left\ ">Value</th></thead><tbody>" );
```



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



Examples Using Arrays



C:\IW3HTP3\examples\ch11\InitArray.html - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media Print

Address C:\IW3HTP3\examples\ch11\InitArray.html Go Links

Array n1 contains

Subscript	Value
0	0
1	1
2	2
3	3
4	4

Array n2 contains

Subscript	Value
0	0
1	1
2	2
3	3
4	4

My Computer

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”

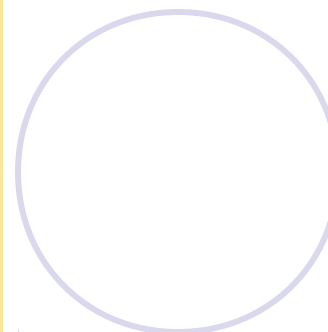
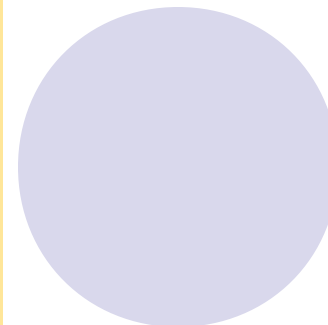
```
var n = [ 10, 20, , 40, 50 ];
```

```
1 <?xml version = "1.0"?>
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. 11.4: InitArray2.html -->
6 <!-- Initializing an Array with a Declaration -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10         <title>Initializing an Array with a Declaration</title>
11
12     <script type = "text/javascript">
13         <!--
14         function start()
15         {
16             // Initializer list specifies n
17             // value for each element.
18             var colors = new Array( "cyan", "magenta",
19                                     "yellow", "black" );
20             var integers1 = [ 2, 4, 6, 8 ];
21             var integers2 = [ 2, , 8 ];
22
23             outputArray( "Array colors contains", colors );
24             outputArray( "Array integers1 contains", integers1 );
25             outputArray( "Array integers2 contains", integers2 );
26         }
```

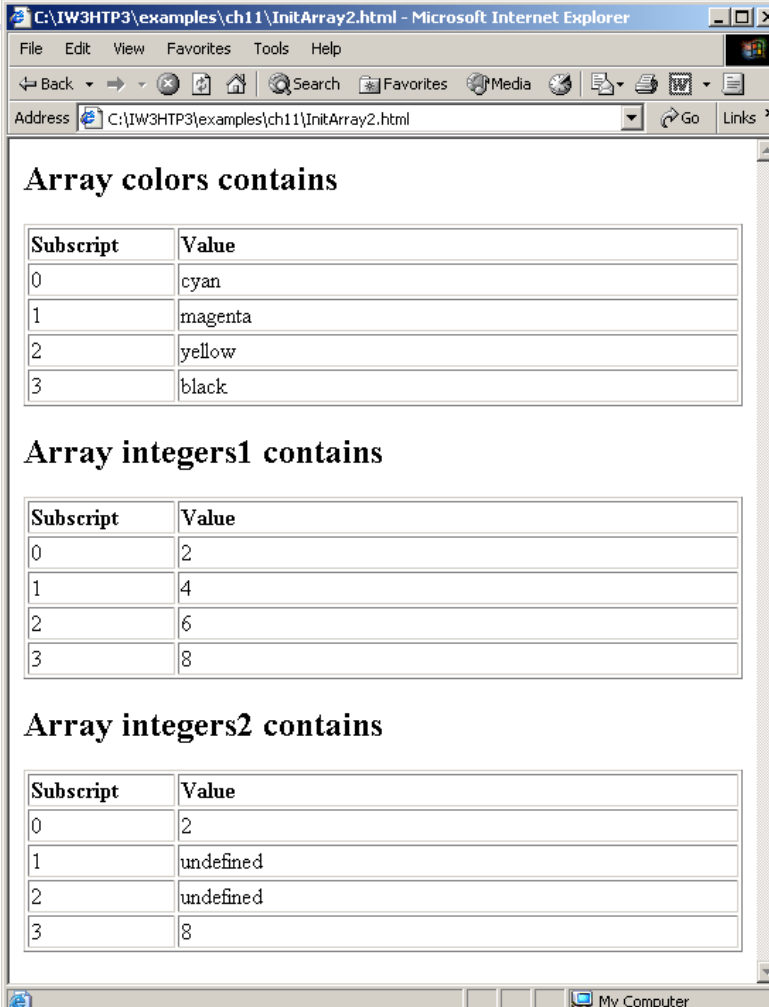
Array integers1 is initialized using an initializer list.

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

```
27 // output "header" followed by a two-column table
28 // containing subscripts and elements of "theArray"
29 function outputArray( header, theArray )
30 {
31     document.writeln( "<h2>" + header + "</h2>" );
32     document.writeln( "<table border = \"1\" \" +
33         \"width = \"100%\">" );
34     document.writeln( "<thead><th width = \"100\" \" +
35         \"align = \"left\">Subscript</th>\" +
36         \"<th align = \"left\">Value</th></thead><tbody>" );
37
38     for ( var i = 0; i < theArray.length; i++ )
39         document.writeln( "<tr><td>" + i + "</td><td>" +
40             theArray[ i ] + "</td></tr>" );
41
42     document.writeln( "</tbody></table>" );
43 }
44 // -->
45 </script>
46
47 </head><body onload = "start()"></body>
48 </html>
```



Examples Using Arrays



The screenshot shows a Microsoft Internet Explorer window with the address bar displaying `C:\IW3HTP3\examples\ch11\InitArray2.html`. The page content consists of three sections, each with a title and a table:

Array colors contains

Subscript	Value
0	cyan
1	magenta
2	yellow
3	black

Array integers1 contains

Subscript	Value
0	2
1	4
2	6
3	8

Array integers2 contains

Subscript	Value
0	2
1	undefined
2	undefined
3	8

Examples Using Arrays

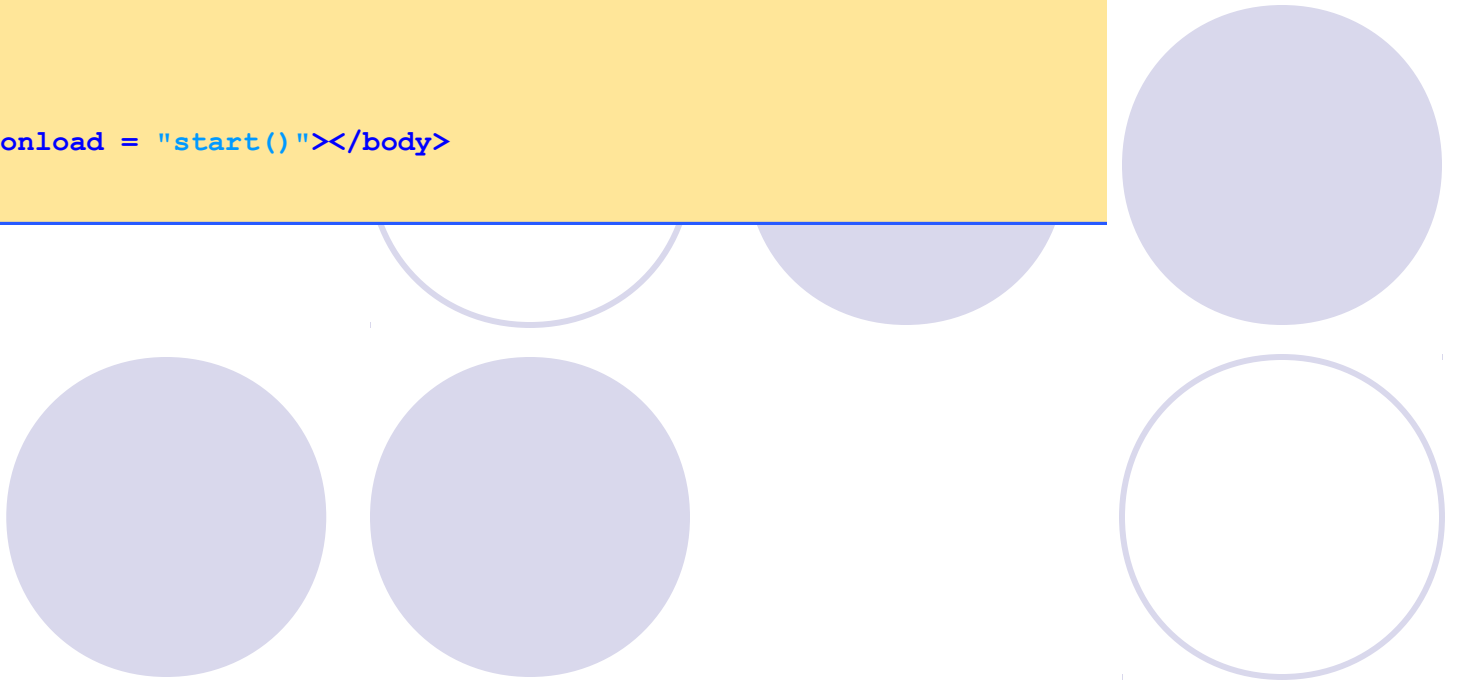
- `for...in` statement
 - Perform an action for each element in an array
 - Iterates over array elements
 - Assigns each element to specified variable one at a time
 - Ignores non-existent elements


```
1 <?xml version = "1.0"?>
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. 11.5: SumArray.html      -->
6 <!-- Summing Elements of an Array -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10         <title>Sum the Elements of an Array</title>
11
12         <script type = "text/javascript">
13             <!--
14             function start()
15             {
16                 var theArray = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
17                 var total1 = 0, total2 = 0;
18
19                 for ( var i = 0; i < theArray.length; i++ )
20                     total1 += theArray[ i ];
21
22                 document.writeln( "Total using subscripts: " + total1 );
23
```

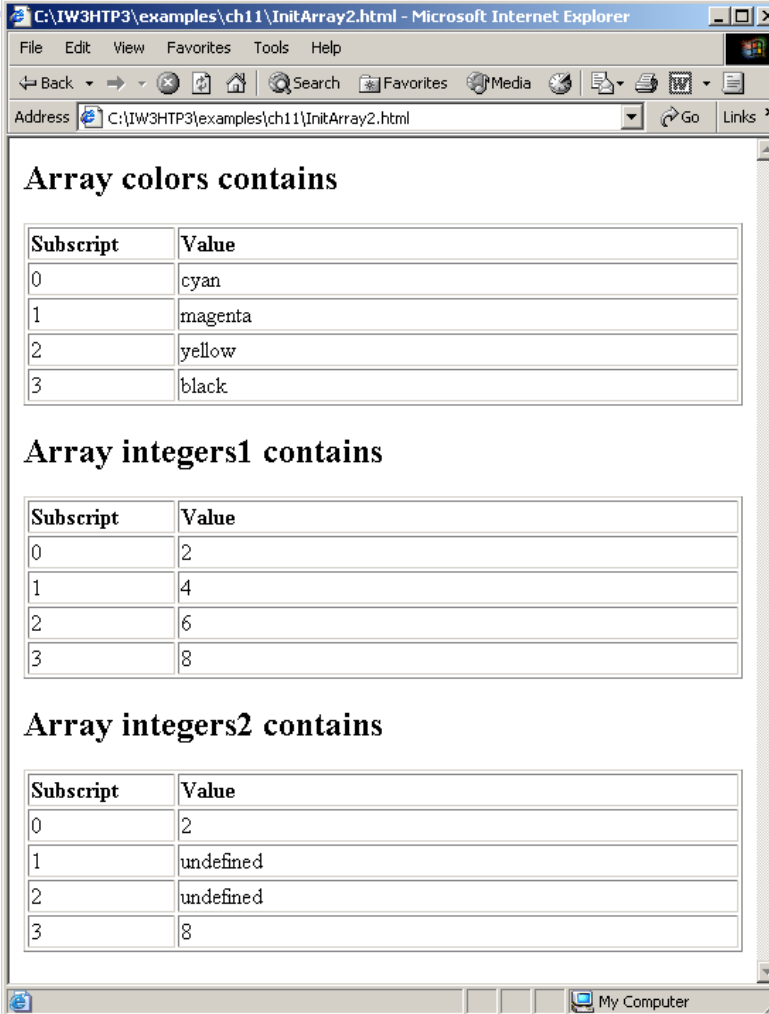
The for loop sums the values contained in the 10-element integer array called `theArray`.

```
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`.



Examples Using Arrays



The screenshot shows a Microsoft Internet Explorer browser window with the address bar displaying `C:\IW3HTP3\examples\ch11\InitArray2.html`. The page content consists of three sections, each with a title and a table:

Array colors contains

Subscript	Value
0	cyan
1	magenta
2	yellow
3	black

Array integers1 contains

Subscript	Value
0	2
1	4
2	6
3	8

Array integers2 contains

Subscript	Value
0	2
1	undefined
2	undefined
3	8

Examples Using Arrays



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

```

1 <?xml version = "1.0"?>
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. 11.6: RollDie.html          -->
6 <!-- Roll a Six-Sided Die 6000 Times -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10         <title>Roll a Six-Sided Die 6000 Times</title>
11
12         <script type = "text/javascript">
13             <!--
14             var face, frequency = [ 0, 0, 0, 0, 0, 0 ];
15
16             // summarize results
17             for ( var roll = 1; roll <= 6000; ++roll ) {
18                 face = Math.floor( 1 + Math.random() * 6 );
19                 ++frequency[ face ];
20             }
21

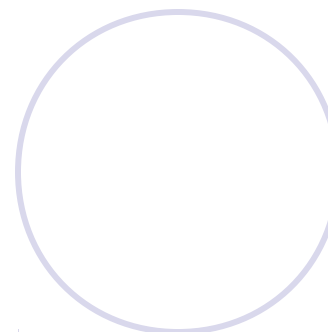
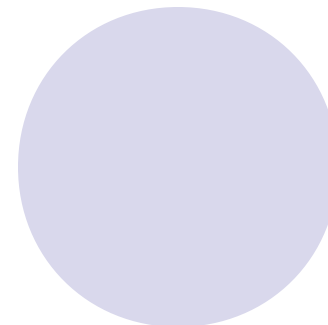
```

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

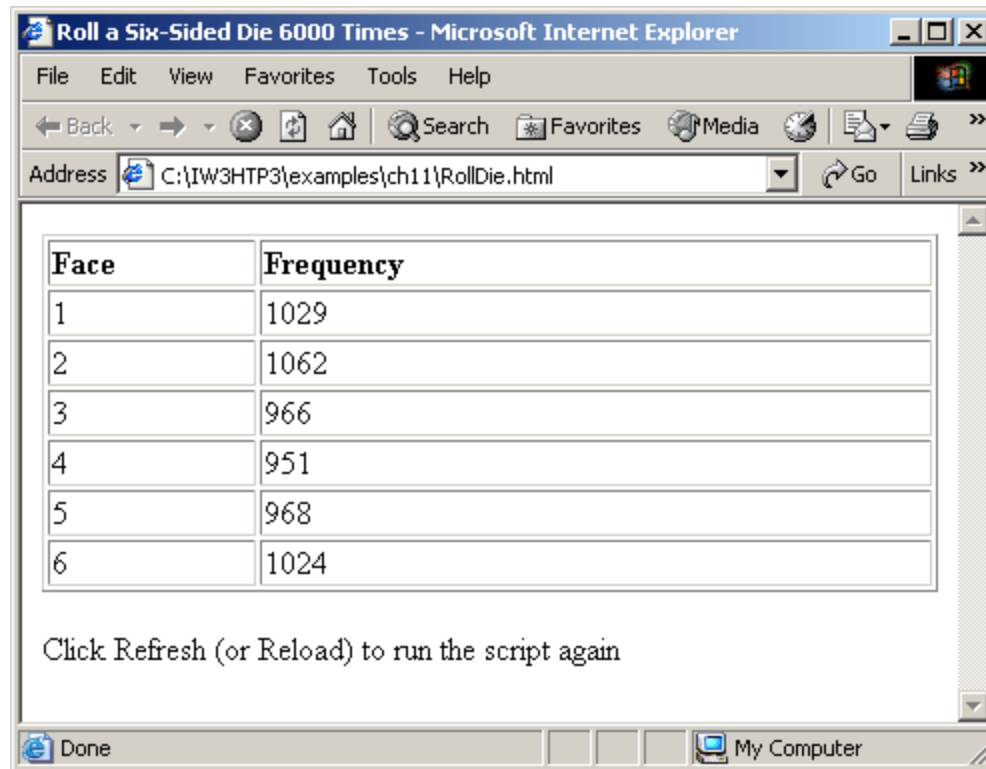
```

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>

```



Examples Using Arrays



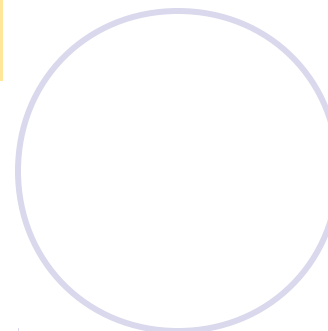
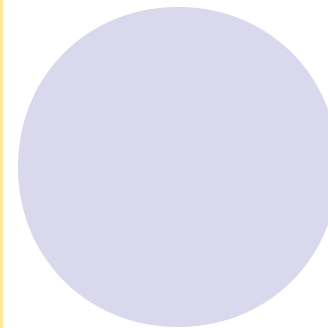
Face	Frequency
1	1029
2	1062
3	966
4	951
5	968
6	1024

Click Refresh (or Reload) to run the script again

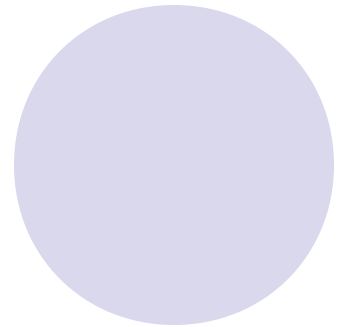
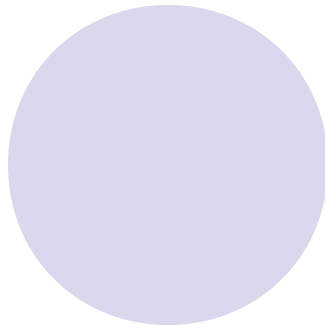
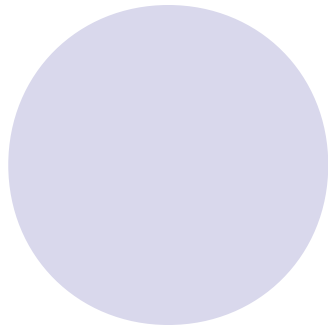
Random Image Generator Using Arrays

- 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

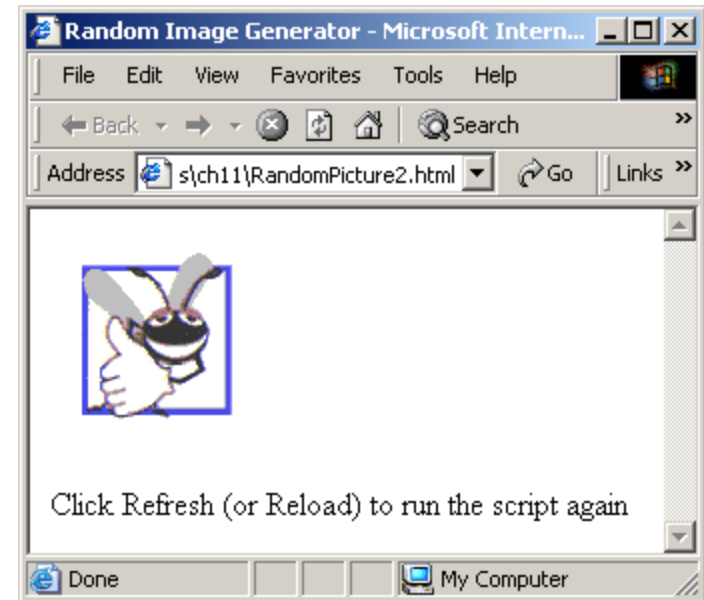
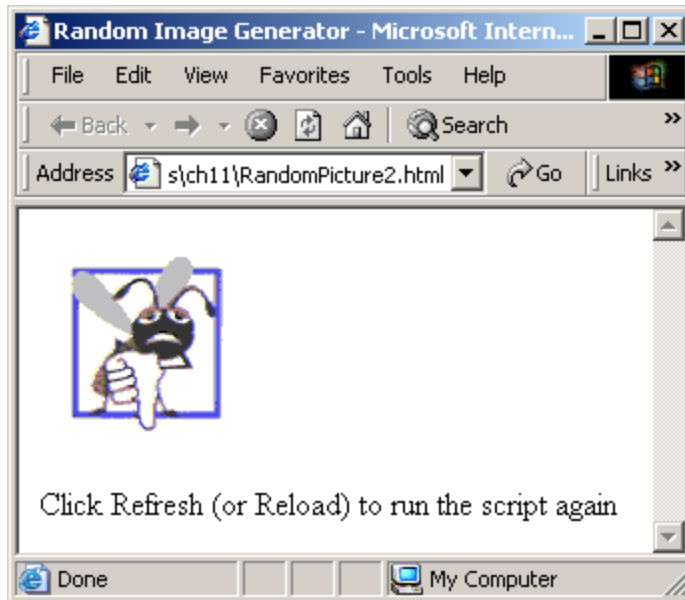

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5 <!-- Fig. 11.7: RandomPicture2.html    -->
6 <!-- Randomly displays one of 7 images -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10         <title>Random Image Generator</title>
11
12         <script type = "text/javascript">
13             <!--
14             var pictures =
15                 [ "CPE", "EPT", "GPP", "GUI", "PERF", "PORT", "SEO" ];
```



```
16     document.write ( "<img src = \"\" +
17         pictures[ Math.floor( Math.random() * 7 ) ] +
18         \".gif\" width = \"105\" height = \"100\" />" );
19     // -->
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>
```



Random Image Generator Using Arrays



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

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

```

1 <?xml version = "1.0"?>
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. 11.8: PassArray.html -->
6 <!-- Passing Arrays          -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10     <title>Passing Arrays and Individual Array
11       Elements to Functions</title>
12
13     <script type = "text/javascript">
14       <!--
15       function start()
16       {
17         var a = [ 1, 2, 3, 4, 5 ];
18
19         document.writeln( "<h2>Effects of passing entire " +
20           "array call-by-reference"
21         outputArray(
22           "The values of the original array are: ", a );
23
24         modifyArray( a ); // array a passed call-by-reference
25

```

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

Function modifyArray multiplies each element by 2.

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

```
29 document.writeln( "<h2  
30     "element call-by-va
```

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

```
31     "a[3] before modifyElement: " + a[ 3 ] );
```

```
32  
33 modifyElement( a[ 3 ] );
```

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

```
35 document.writeln(  
36     "<br />a[3] after modifyElement: " + a[ 3 ]  
37 }  
38
```

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

```
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
```

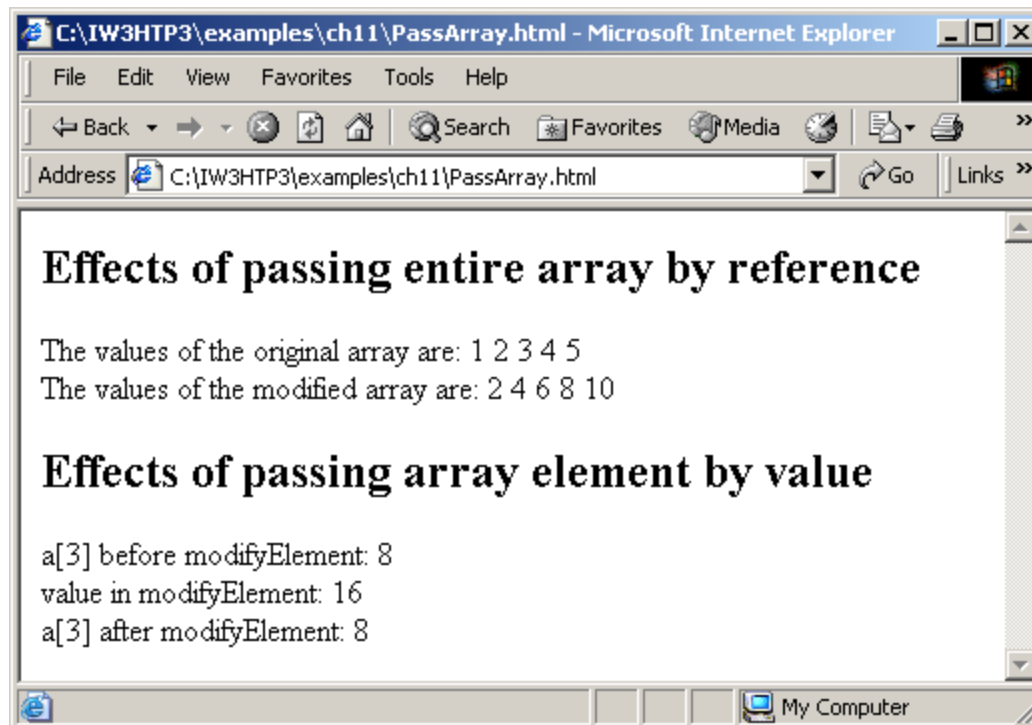
Method `join` takes as its argument a string containing a separator that should be used to separate the elements of the array in the string that is returned.

```
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>
```

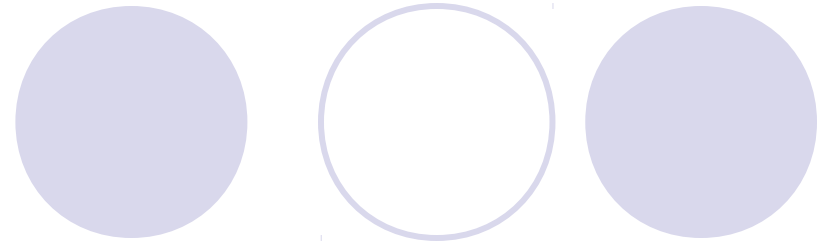
Multiply each element in theArray by 2.



Passing Arrays to Functions



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

```

1 <?xml version = "1.0"?>
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. 11.9: sort.html -->
6 <!-- Sorting an Array      -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10         <title>Sorting an Array with Array Method sort</title>
11
12         <script type = "text/javascript">
13             <!--
14             function start()
15             {
16                 var a = [ 10, 1, 9, 2, 8, 3, 7, 4, 6, 5 ];
17
18                 document.writeln( "<h1>Sorting an Array</h1>" );
19                 outputArray( "Data items in original order: ", a );
20                 a.sort( compareIntegers ); // sort the array
21                 outputArray( "Data items in ascending order: ", a );
22             }

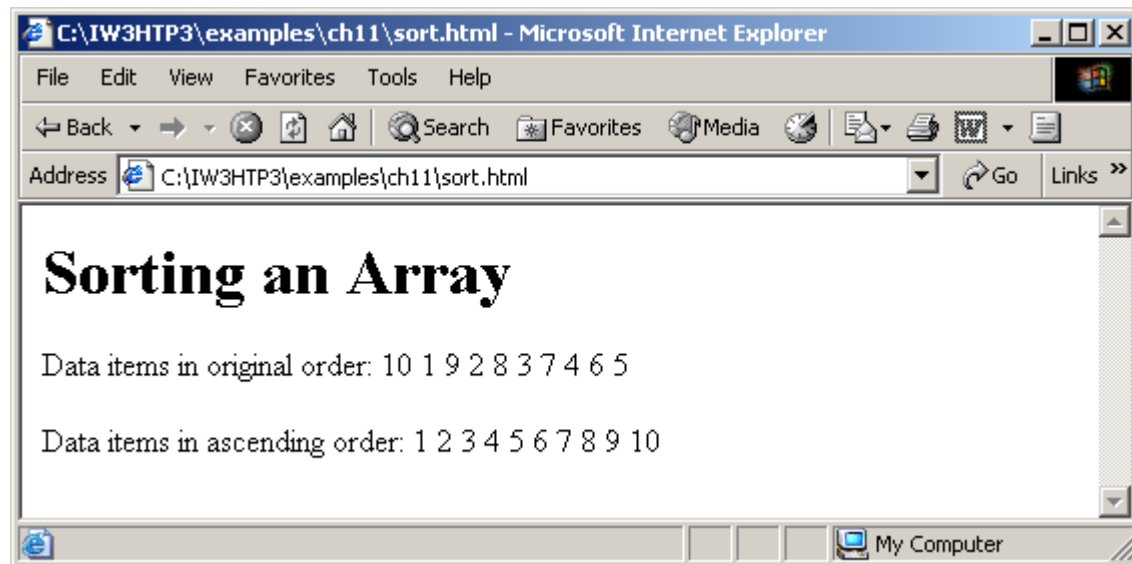
```

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.

```
23 // outputs "header" followed by the contents of "theArray"
24 function outputArray( header, theArray )
25 {
26     document.writeln( "<p>" + header +
27         theArray.join( " " ) + "</p>" );
28 }
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.

Sorting Arrays



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

```
1 <?xml version = "1.0"?>
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. 11.10: LinearSearch.html -->
6 <!-- Linear Search of an Array -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10         <title>Linear Search of an Array</title>
11
12         <script type = "text/javascript">
13             <!--
14             var a = new Array( 100 ); // create an Array
15
16             // fill Array with even integer values from 0 to 198
17             for ( var i = 0; i < a.length; ++i )
18                 a[ i ] = 2 * i;
19
```

Array a is initiated with 100 elements.

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

20 // function called when "Search" button is pressed

21 function buttonPressed()

22 {

23 var searchKey = searchForm.inputVal.value;

24 // Array a is passed to linearSearch even though it

25 // is a global variable. Normally a

26 // be passed to a method for search

27 var element = linearSearch(a, parseInt(searchKey));

28 if (element != -1)

29 searchForm.result.value =

30 "Found value in element

31 else

32 searchForm.result.value = "Value not found";

33 }

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.


```

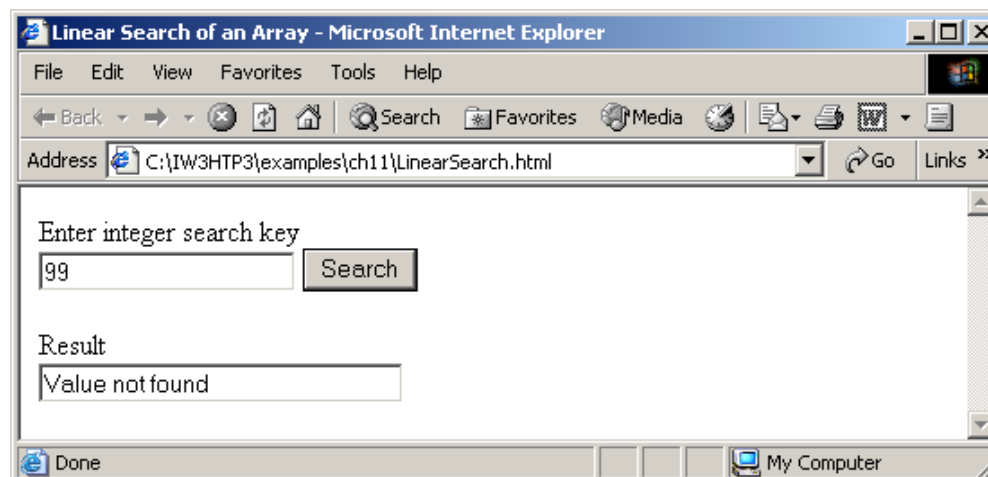
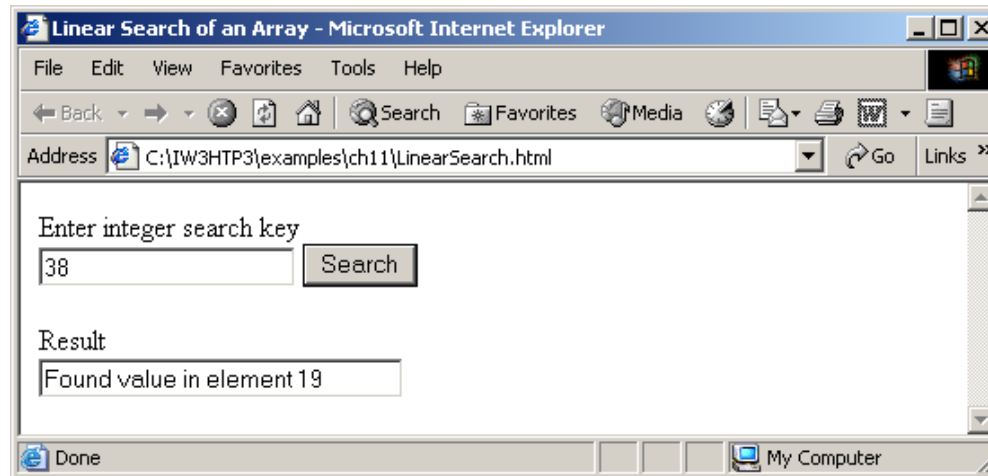
37 // Search "theArray" for the specified "key" value
38 function linearSearch( theArray, key )
39 {
40     for ( var n = 0; n < theArray.length; ++n )
41         if ( theArray[ n ] == key )
42             return n;
43
44     return -1;
45 }
46 // -->
47 </script>
48
49 </head>
50
51 <body>
52     <form 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 searchKey

Function linearSearch compares each
each element with a search key.

Searching Arrays: Linear Search and Binary Search



```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4
5 <!-- Fig. 11.11 : BinarySearch.html -->
6 <!-- Binary search -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10         <title>Binary Search</title>
11
12         <script type = "text/javascript">
13             <!--
14             var a = new Array( 15 );
15
16             for ( var i = 0; i < a.length; ++i )
17                 a[ i ] = 2 * i;
18
```

Array a is initialized with 15 elements.

19 // function called when "Search" button is pressed

20 function buttonPressed()

21 {

22 var searchKey = searchForm.inputVal.value;

24 searchForm.result.value =

25 "Portions of array searched\n";

27 // Array a is passed to binarySearch

28 // is a global variable. This is done because

29 // normally an array is passed to a method

30 // for searching.

31 var element =

32 binarySearch(a, parseInt(searchKey));

34 if (element != -1)

35 searchForm.result.value +=

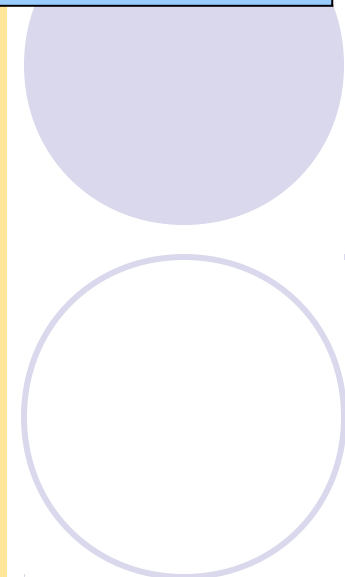
36 "\nFound value in element " + element;

37 else

38 searchForm.result.value += "\nValue not found";

39 }

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



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 being

53 // during each iteration of the

54 // search loop.

55 buildOutput(theArray, low, middle, high);

56
57 if (key == theArray[middle]) // match

58 return middle;

59 else if (key < theArray[middle])

60 high = middle - 1; // search low end of array

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 than the middle elements, the high subscript is set to middle + 1.

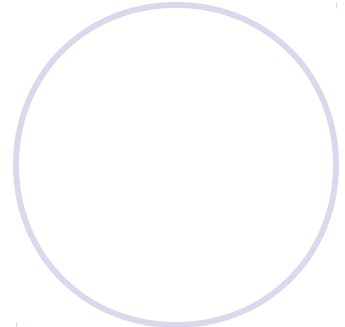
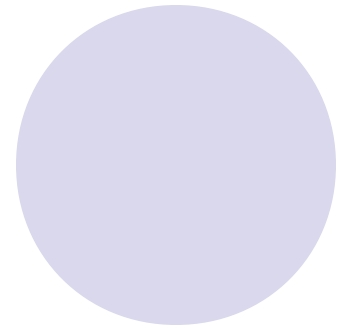
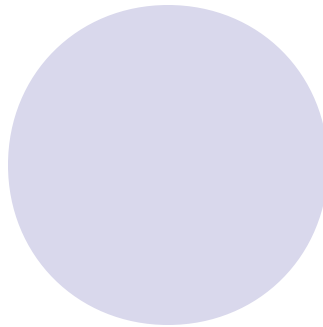
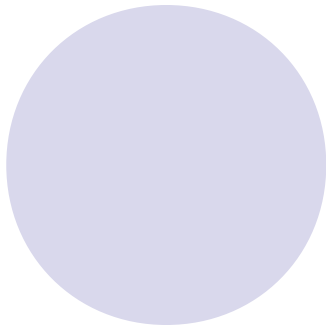
```

64         return -1;    // searchKey not found
65     }
66
67
68     // Build one row of output showing the current
69     // part of the array being processed.
70     function buildOutput( theArray, low, mid, high )
71     {
72         for ( var i = 0; i < theArray.length; i++ ) {
73             if ( i < low || i > high )
74                 searchForm.result.value += "<br>";
75             // mark middle element in output
76             else if ( i == mid )
77                 searchForm.result.value += theArray[ i ] +
78                     ( theArray[ i ] < 10 ? "*" : " " );
79             else
80                 searchForm.result.value += theArray[ i ] +
81                     ( theArray[ i ] < 10 ? " " : " " );
82         }
83
84         searchForm.result.value += "\n";
85     }
86     // -->
87 </script>
88 </head>
89

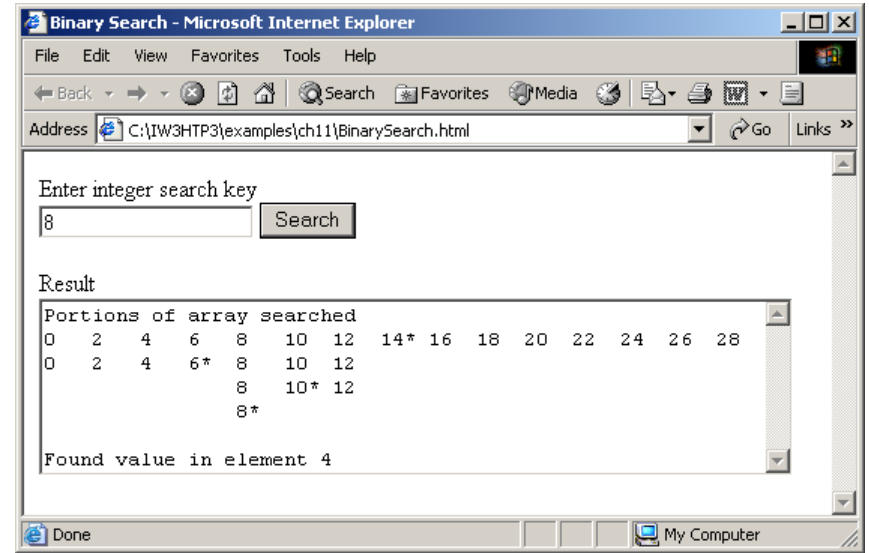
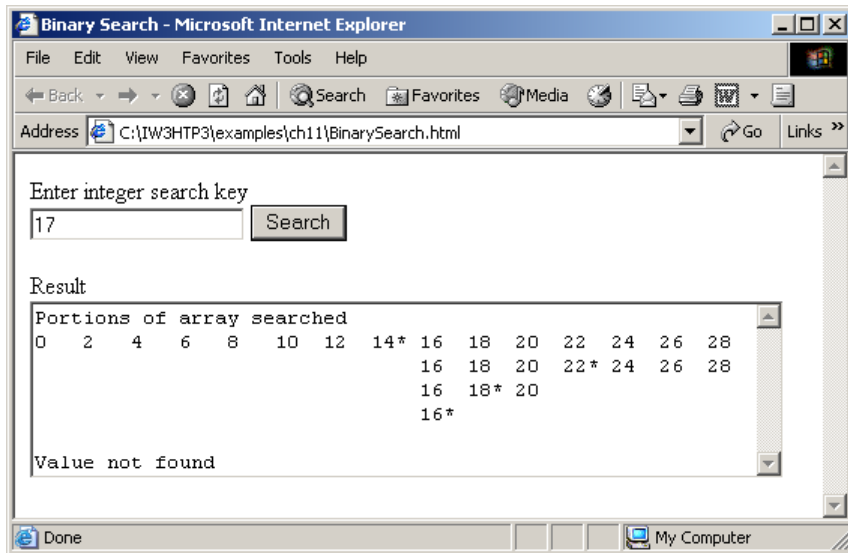
```

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

```
90 <body>
91     <form name = "searchForm" action = "">
92         <p>Enter integer search key<br />
93         <input name = "inputVal" type = "text" />
94         <input name = "search" type = "button" value =
95             "Search" onclick = "buttonPressed()" /><br /></p>
96         <p>Result<br />
97         <textarea name = "result" rows = "7" cols = "60">
98         </textarea></p>
99     </form>
100 </body>
101</html>
```



Searching Arrays: Linear Search and Binary Search



Multidimensional Arrays



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

Multidimensional Arrays

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Diagram illustrating the structure of a two-dimensional array with three rows and four columns. The array is represented as a table with rows and columns. The first column is labeled 'Column 0', the second 'Column 1', the third 'Column 2', and the fourth 'Column 3'. The rows are labeled 'Row 0', 'Row 1', and 'Row 2'. Each cell in the table contains a reference to an element in the array, such as 'a[0][0]' for the element at Row 0, Column 0. Below the table, three arrows point to the first row of the table, labeled 'Column subscript (or index)', 'Row subscript (or index)', and 'Array name' respectively, indicating the components of the array notation.

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

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 ] ] ;
```

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 );
```

```

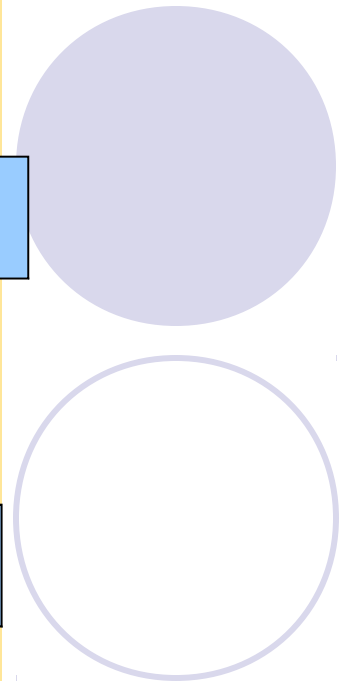
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 11.13: InitArray3.html          -->
6 <!-- Initializing Multidimensional Arrays -->
7
8 <html xmlns = "http://www.w3.org/1
9     <head>
10         <title>Initializing Multidimensional Arrays</title>
11
12         <script type = "text/javascript"
13             <!--
14             function start()
15             {
16                 var array1 = [ [ 1, 2, 3 ],      // first row
17                               [ 4, 5, 6 ],      // second row
18                               var array2 = [
19                                   [ 1, 2, 3 ],
20                                   [ 4, 5, 6 ] ]; // third row
21
22                 outputArray( "Values in array1 by row", array1 );
23                 outputArray( "Values in array2 by row", array2 );
24             }

```

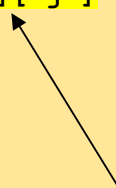
Array array1 provides six initializers in two rows.

Array array2 provides six initializers in three rows.

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

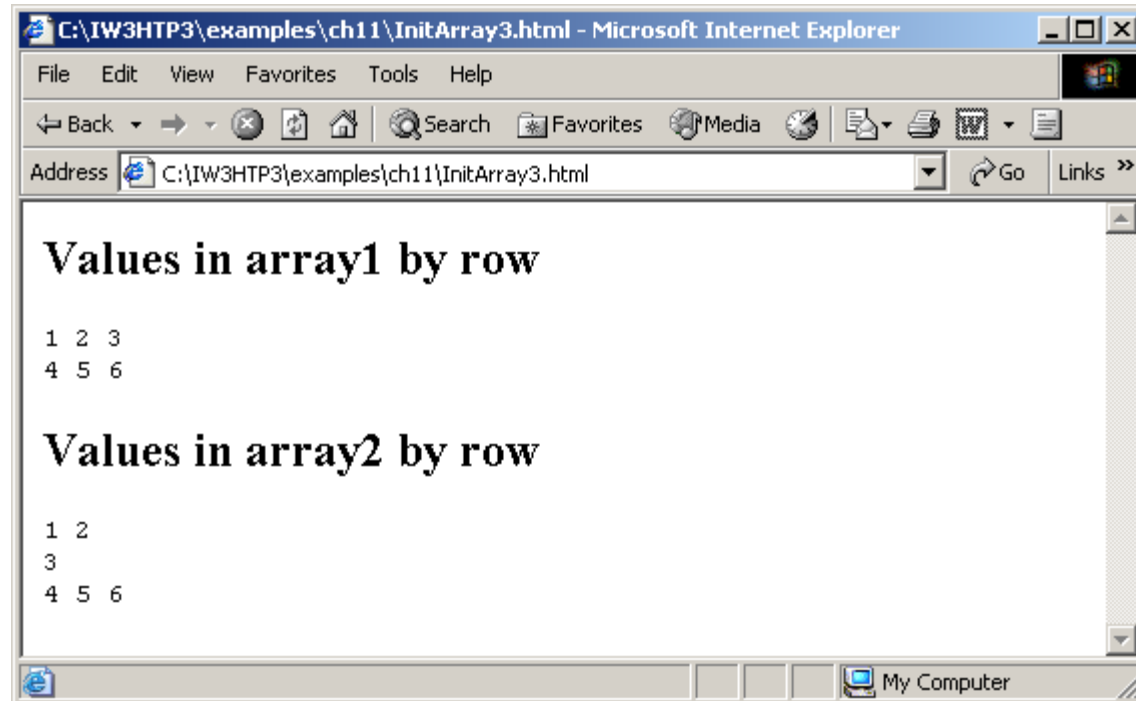


```
25 function outputArray( header, theArray )
26 {
27     document.writeln( "<h2>" + header + "</h2><tt>" );
28
29     for ( var i in theArray ) {
30
31         for ( var j in theArray[ i ] )
32             document.write( theArray[ i ][ j ] + " " );
33
34         document.writeln( "<br />" );
35 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



Referencing the multidimensional array theArray.

Multidimensional Arrays



Different array manipulations using `for` and `for/in`

```
var total = 0;
for (var row = 0; row < a.length; ++row )
    for (var col = 0; col < a[ row ].length; ++col )
        total += a[ row ][ col ];
```

identical to

```
var total = 0;
for (var row in a )
    for (var col in a[ row ] )
        total += a[ row ][ col ];
```

Both statements total the elements of the array,
one row at a time

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

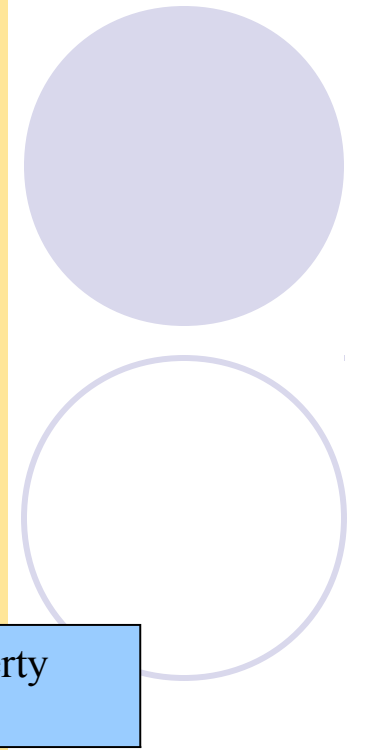
- XHTML Forms

- Contain controls, including radio buttons

- `action` property specifies what happens when submitted

- Can call JavaScript code

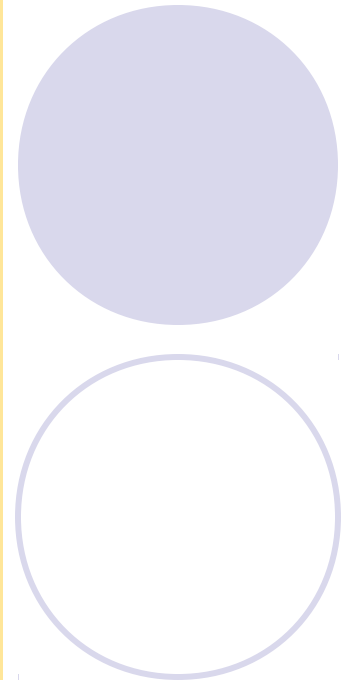
```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5 <!-- Fig. 11.14: quiz.html -->
6 <!-- Online Quiz -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9 <head>
10 <title>Online Quiz</title>
11
12 <script type = "text/JavaScript">
13
14     function checkAnswers()
15     {
16         // determine whether the answer is correct
17         if ( myQuiz.radioButton[ 1 ].checked )
18             document.write( "Congratulations, your answer is correct" );
19         else // if the answer is incorrect
20             document.write( "Your answer is incorrect" );
21     }
22
23 </script>
24
25 </head>
```



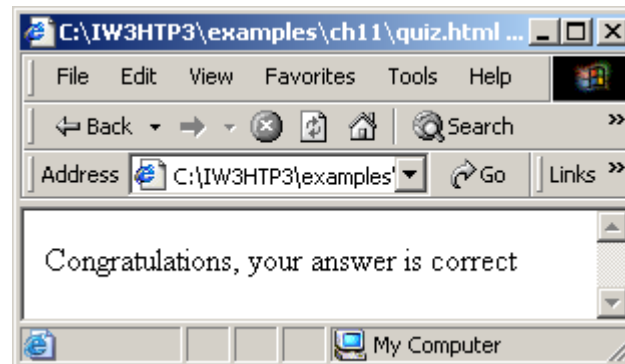
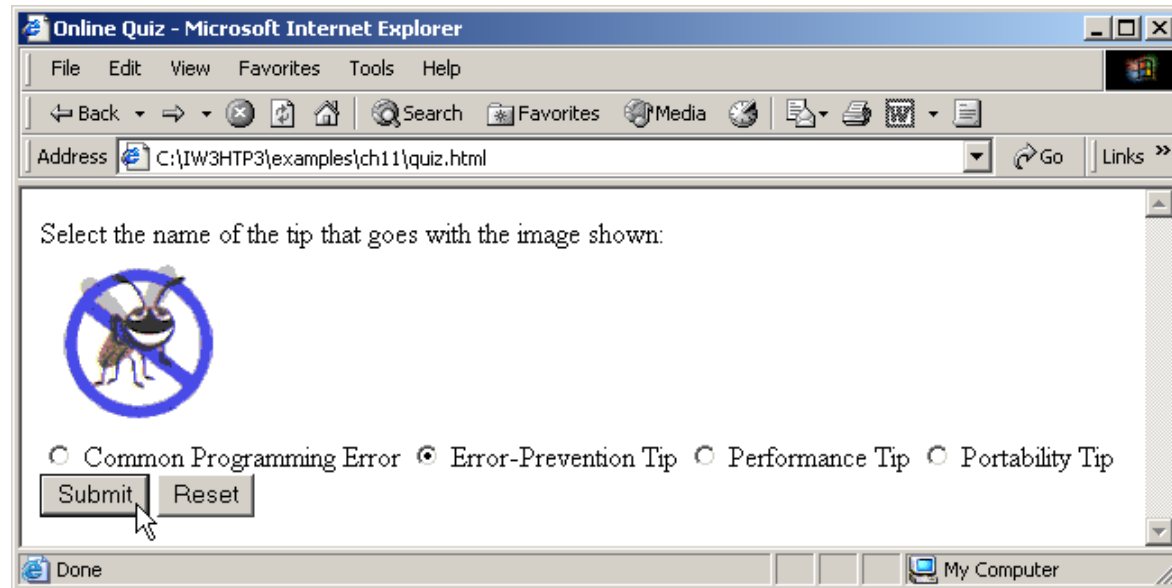
Determining the value of property
checked.

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

Call the checkAnswers function
when the form is submitted.



Building an Online Quiz



Assignment 9

- 1) Exercise # 11.10.

Due Date for A # 9:

- Next Monday before your lecture.