JavaScript: Arrays

Outline

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

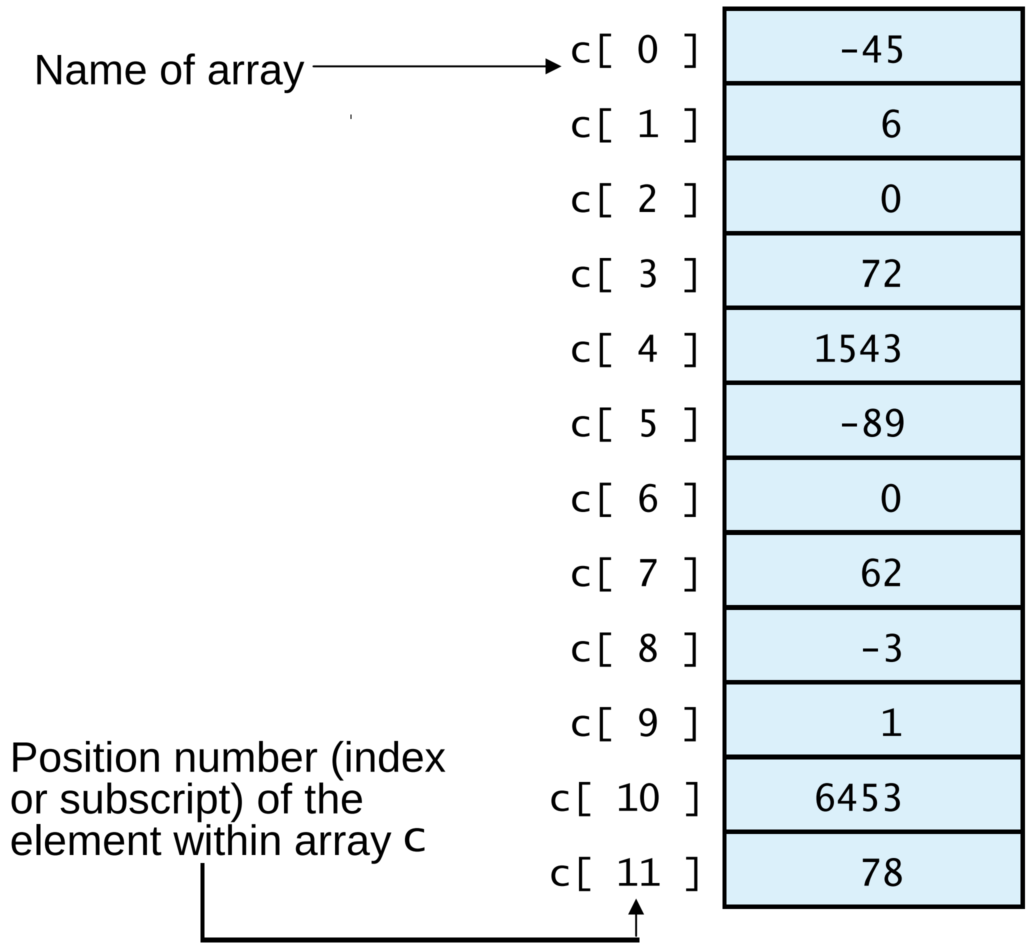


Fig. 11.1 A 12-element array.

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

```
<?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> eler
         // 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;
```

Outline

InitArray.html
(1 of 3)

`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).

<u>Outline</u>

```

         // 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>" );
         doc



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

The for loop adds
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.

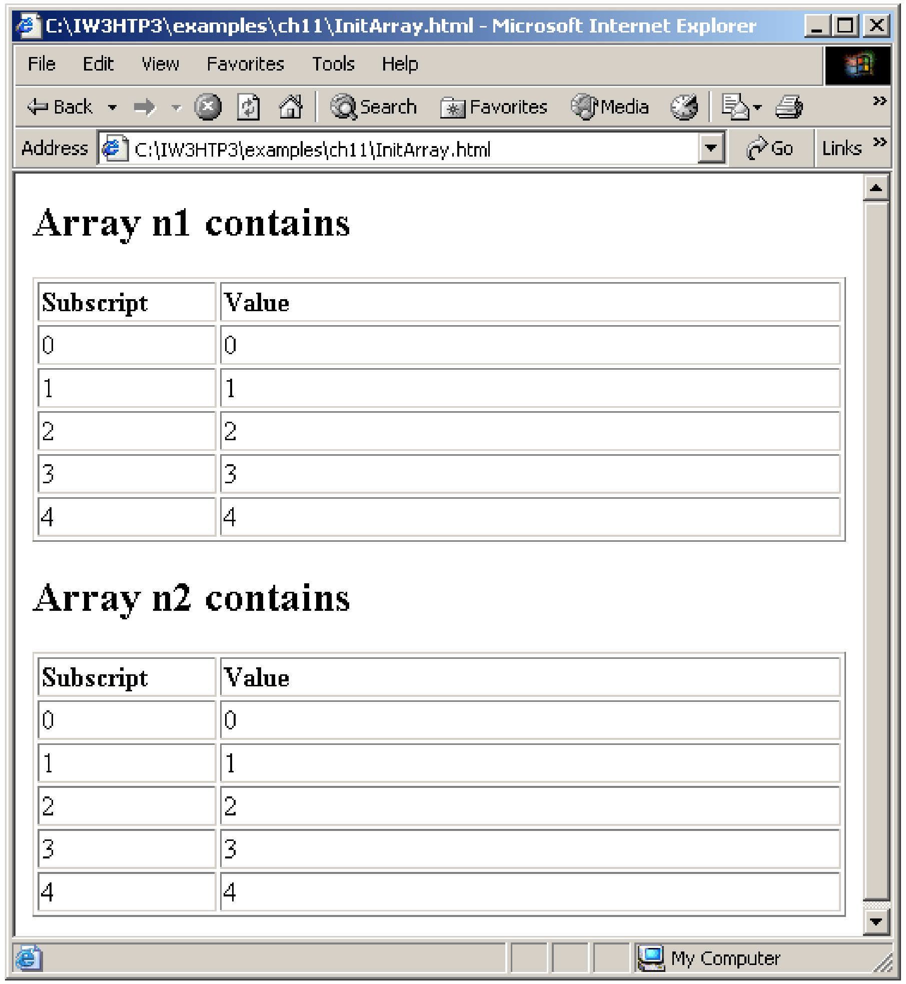
```

            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 = "initializeArrays()"></body>
</html>
```

Outline

InitArray.html
(1 of 3)

11.4 Examples Using Arrays

Fig. 11.3 Initializing the elements of an array.


C:\IW3HTP3\examples\ch11\InitArray.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\InitArray.html Go Links
My Computer

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

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`.

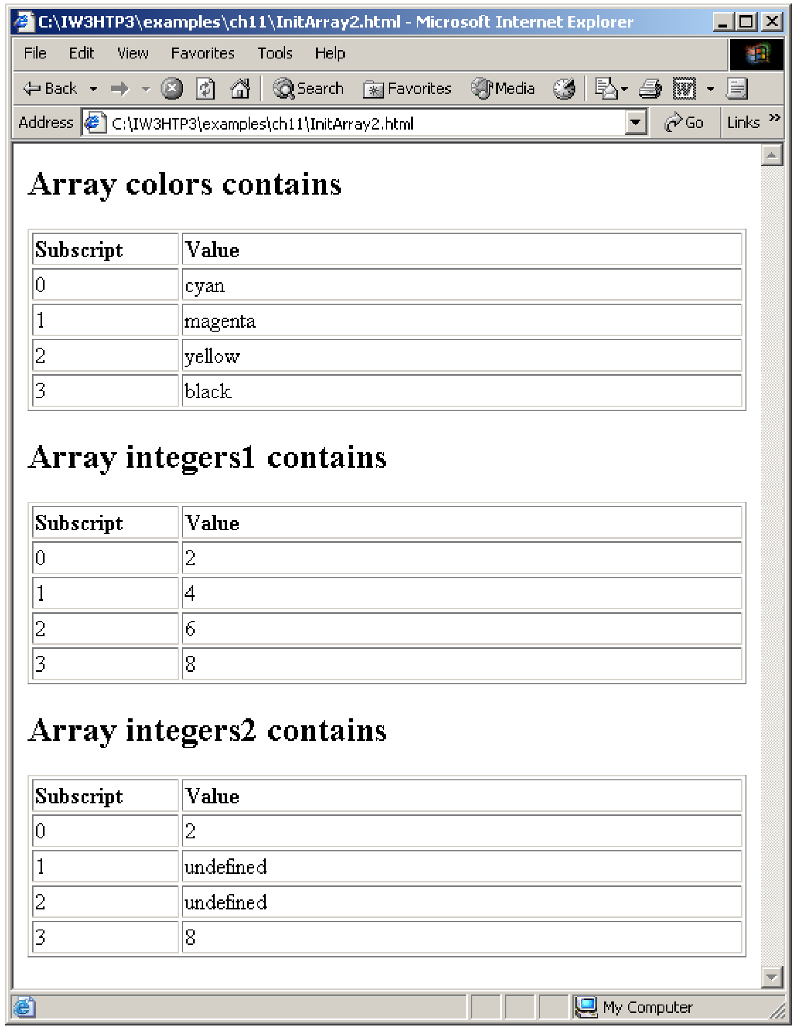
```

         // 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>
```

Outline

InitArray2.html (2 of 2)

11.4 Examples Using Arrays

Fig. 11.4 Initializing the elements of an array.


C:\IW3HTP3\examples\ch11\InitArray2.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\InitArray2.html
Go Links
My Computer

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

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

```
<?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`.

Outline

SumArray.html (1 of 2)

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

            document.writeln( "<br />Total using for...ir
               total2 );
         }
         // -->
      </script>

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

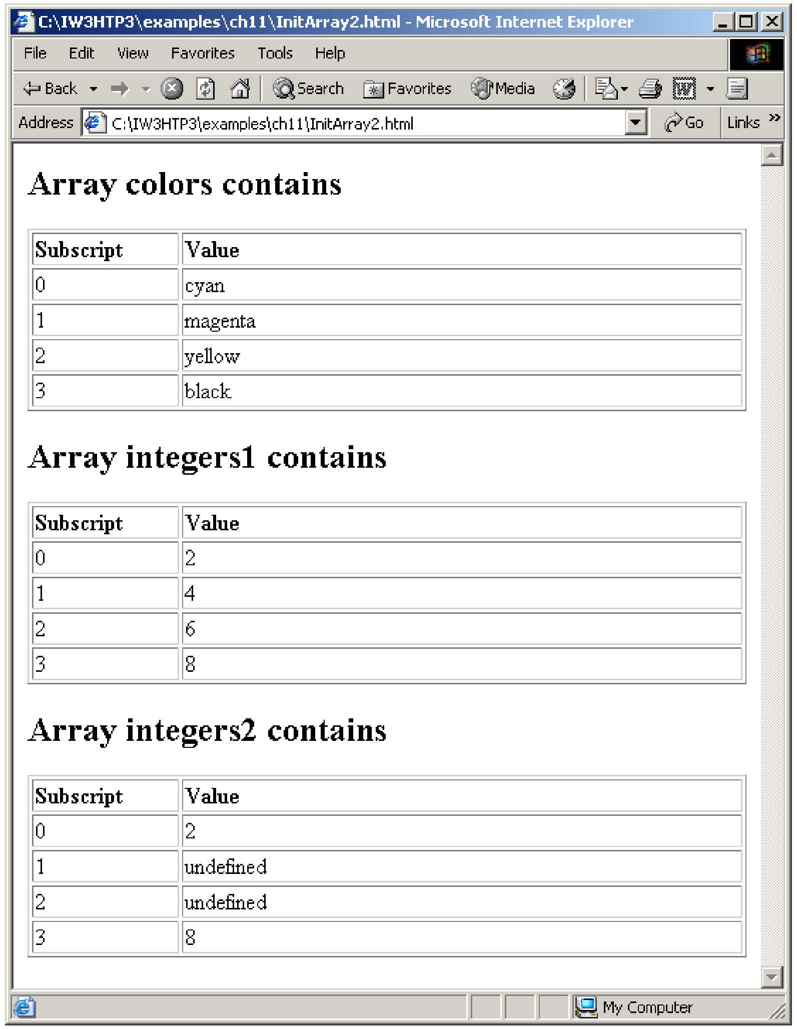
Outline

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

(2 of 2)

11.4 Examples Using Arrays

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


C:\IW3HTP3\examples\ch11\InitArray2.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\InitArray2.html
Go Links
My Computer

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

11.4 Examples Using Arrays

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

```
<?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.

Outline

RollDie.html
(1 of 2)

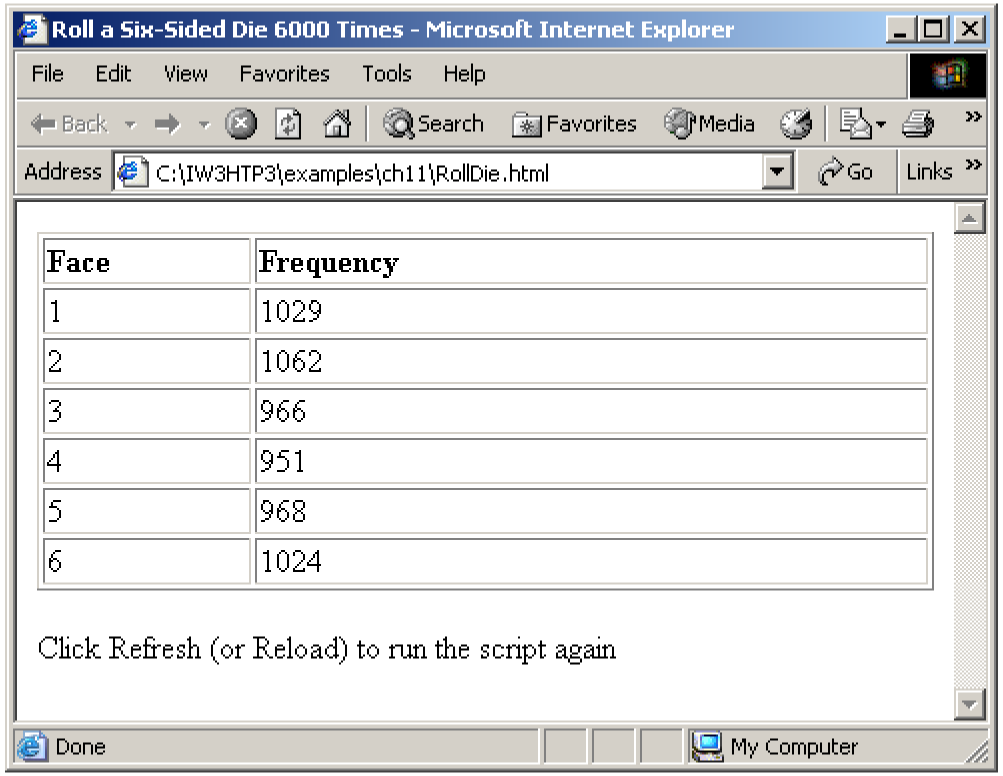
```
         document.writeln( "<table border = \"1\""  +
            "width = \"100%\">" );
         document.writeln( "<thead><th width = \"100\"" +
            " align = \"left\">Face<th align = \"left\">" +
            "Frequency</th></thead></tbody>" );

         for ( face = 1; face < frequency.length; ++face )
            document.writeln( "<tr><td>" + face + "</td><td>" +
               frequency[ face ] + "</td></tr>" );

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

   </head>
   <body>
      <p>Click Refresh (or Reload) to run the script again</p>
   </body>
</html>
```

Outline

RollDie.html
(2 of 2)

11.4 Examples Using Arrays

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


Roll a Six-Sided Die 6000 Times - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\RollDie.html
Go Links
Done
My Computer

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

Click Refresh (or Reload) to run the script again

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)

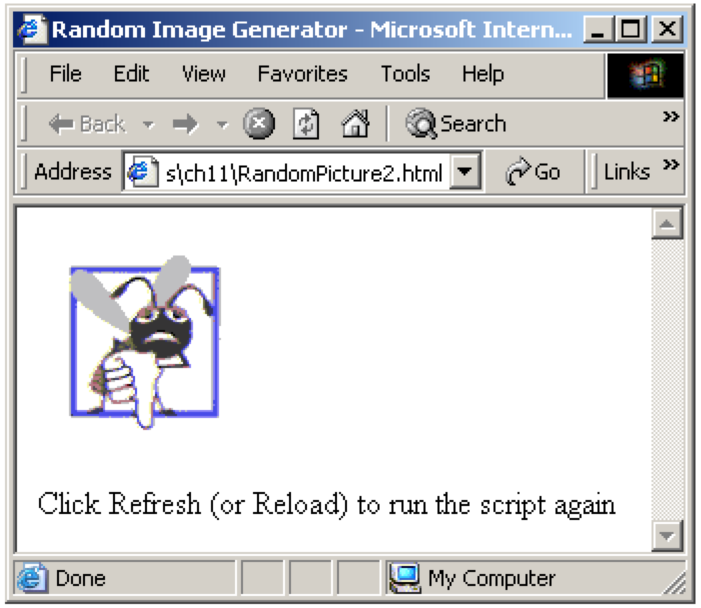
```

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

   </head>

   <body>
      <p>Click Refresh (or Reload) to run the script again</p>
   </body>
</html>
```

Outline

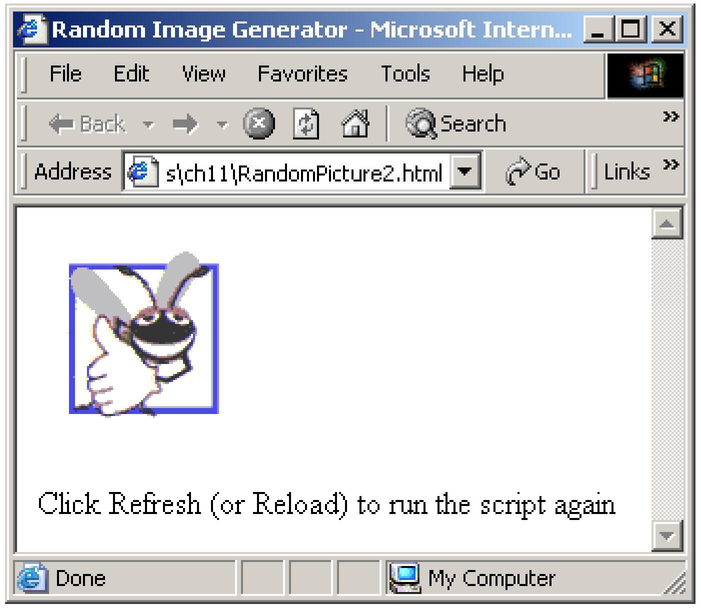
RandomPicture2.html
(2 of 2)

11.5 Random Image Generator Using Arrays

Fig. 11.7 Random image generation using arrays.


Random Image Generator - Microsoft Intern...
File Edit View Favorites Tools Help
Back Search
Address s\ch11\RandomPicture2.html Go Links
Done My Computer

Click Refresh (or Reload) to run the script again


Random Image Generator - Microsoft Intern...
File Edit View Favorites Tools Help
Back Search
Address s\ch11\RandomPicture2.html Go Links
Done My Computer

Click Refresh (or Reload) to run the script again

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

```
<?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

PassArray.html (1 of 3)

Outline

ssArray.html

(2 of 3)

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

      document.writeln( "<h2
         "element call-by-va
         "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 />" );
   }

```

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.

```
         // 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 );
         }
         // -->
      </script>

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

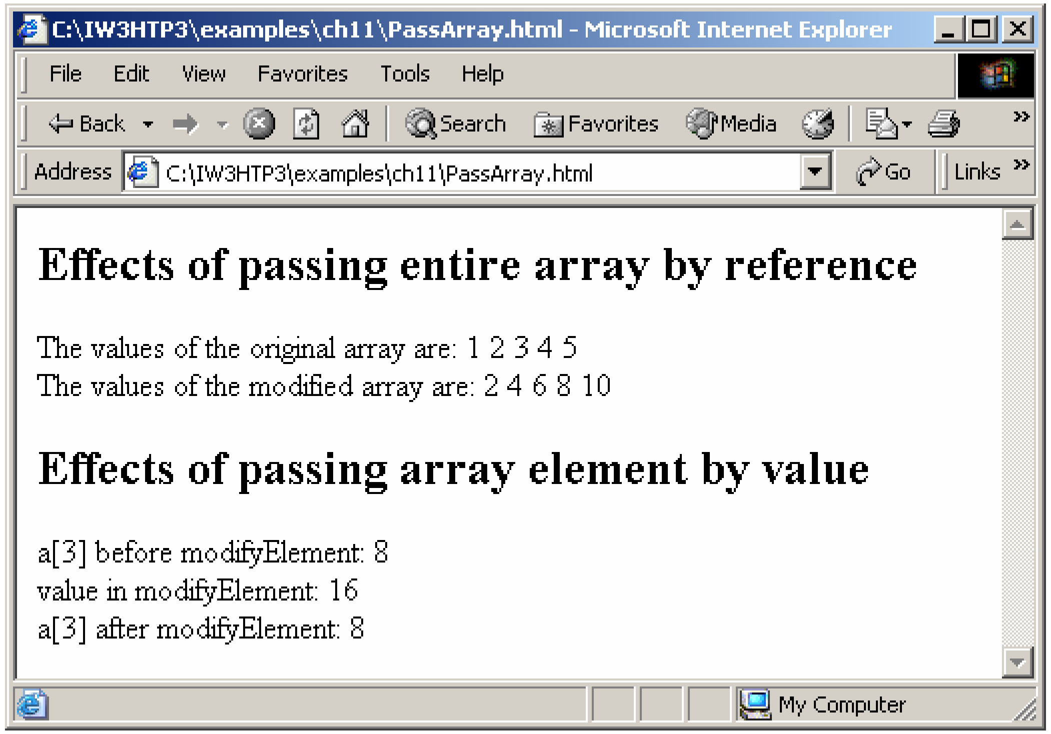
Multiply each element in `theArray` by 2.

Outline

PassArray.html (3 of 3)

11.7 Passing Arrays to Functions

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


C:\IW3HTP3\examples\ch11\PassArray.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\PassArray.html
Go Links
My Computer

```
Effects of passing entire array by reference

The values of the original array are: 1 2 3 4 5
The values of the modified array are: 2 4 6 8 10

Effects of passing array element by value

a[3] before modifyElement: 8
value in modifyElement: 16
a[3] after modifyElement: 8
```

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.

```

         // 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>
```

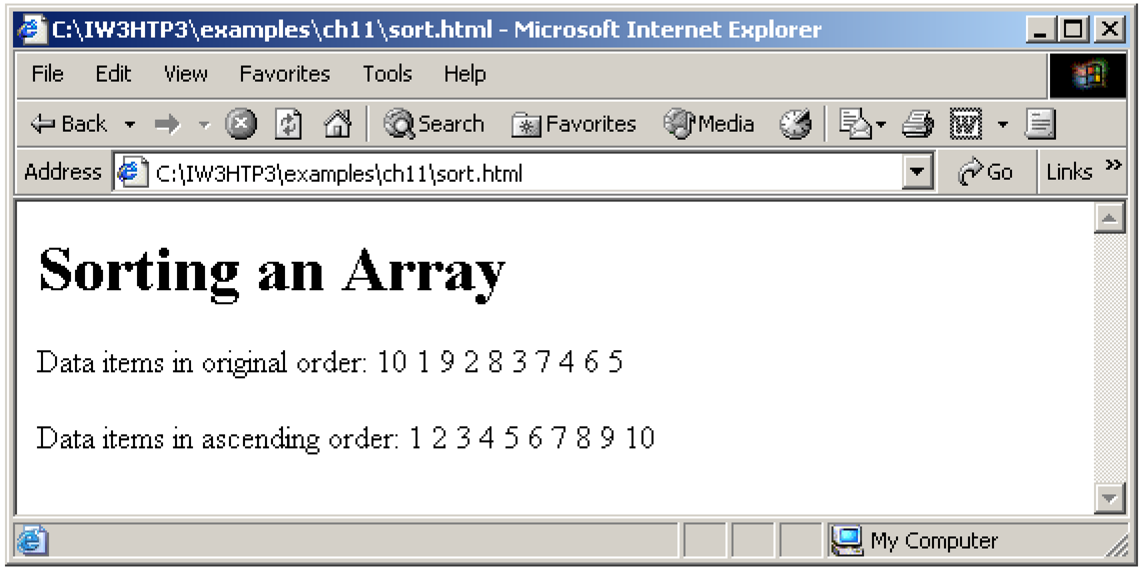
Function `compareIntegers` calculates the difference between the integer values of its arguments.

Outline

Sort.html
(2 of 2)

11.8 Sorting Arrays

Fig. 11.9 Sorting an array with `sort`.


C:\IW3HTP3\examples\ch11\sort.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\sort.html
Go Links
My Computer

Sorting an Array

```
Data items in original order: 10 1 9 2 8 3 7 4 6 5

Data items in ascending order: 1 2 3 4 5 6 7 8 9 10
```

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.

```
      // 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 a
         // be passed to a method for search
         var element = linearSearch( a, parseInt( searchKey ) );

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

```

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 (2 of 3)

```
         // 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" 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 />
         <input name = "result" type = "text" size = "30" /></p>
      </form>
   </body>
</html>
```

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

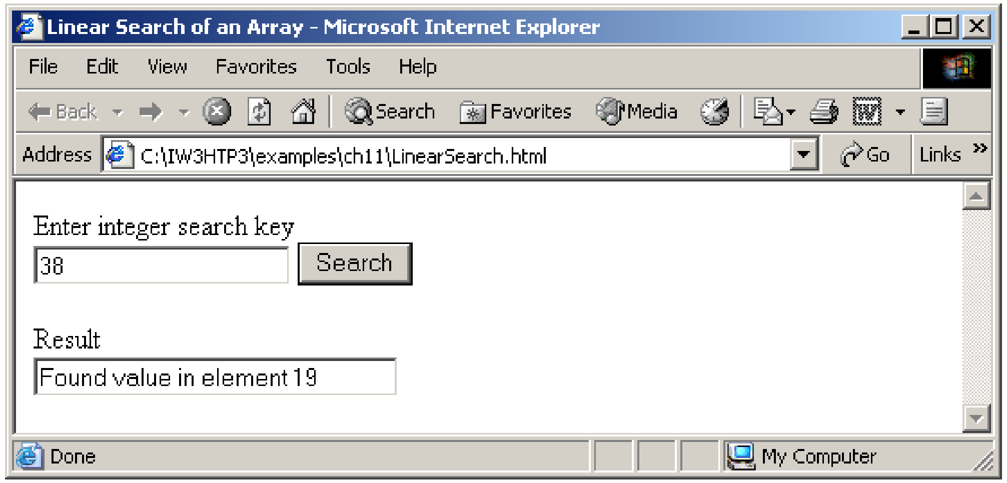
Function `linearSearch` compares each each element with a search key.

Outline

LinearSearch.html (3 of 3)

11.9 Searching Arrays: Linear Search and Binary Search

Fig. 11.10   Linear search of an array.


Linear Search of an Array - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\LinearSearch.html
Go Links
Done
My Computer

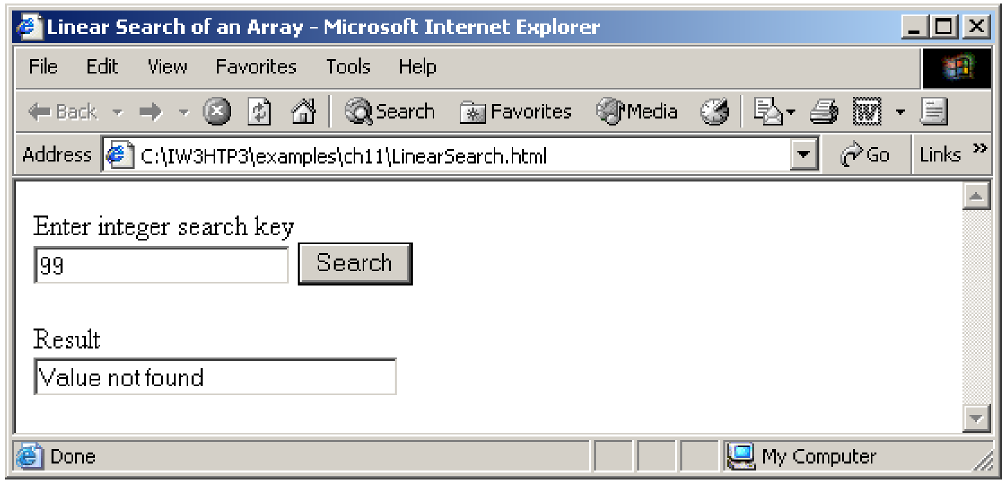
Enter integer search key

38

Search

Result

Found value in element 19


Linear Search of an Array - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\LinearSearch.html
Go Links
Done
My Computer

Enter integer search key

99

Search

Result

Value not found