Outline
11.1 Introduction
11.2 Arrays
11.3 Declaring and Allocating Arrays
11.4 Examples Using Arrays
11.5 Random Image Generator Using Arrays
11.6 References and Reference Parameters
11.7 Passing Arrays to Functions
11.8 Sorting Arrays
11.9 Searching Arrays: Linear Search and Binary Search
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

![Array Diagram]

**Fig. 11.1** A 12-element array.
## 11.2 Arrays

<table>
<thead>
<tr>
<th>Operators</th>
<th>Associativity</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>left to right</td>
<td>highest</td>
</tr>
<tr>
<td>[</td>
<td>left to right</td>
<td>multiplicative</td>
</tr>
<tr>
<td>.</td>
<td>left to right</td>
<td>additive</td>
</tr>
<tr>
<td>++</td>
<td>right to left</td>
<td>unary</td>
</tr>
<tr>
<td>--</td>
<td>right to left</td>
<td>relational</td>
</tr>
<tr>
<td>!</td>
<td>right to left</td>
<td>equality</td>
</tr>
<tr>
<td>*</td>
<td>left to right</td>
<td>logical AND</td>
</tr>
<tr>
<td>/</td>
<td>left to right</td>
<td>logical OR</td>
</tr>
<tr>
<td>%</td>
<td>right to left</td>
<td>conditional</td>
</tr>
<tr>
<td>==</td>
<td>left to right</td>
<td></td>
</tr>
<tr>
<td>!=</td>
<td>left to right</td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>left to right</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>=</td>
</tr>
<tr>
<td>? :</td>
<td>right to left</td>
<td>conditional</td>
</tr>
<tr>
<td>= += -= *= /= %=</td>
<td>right to left</td>
<td>assignment</td>
</tr>
</tbody>
</table>

**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

```javascript
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
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).
```javascript
// 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( "<thead><th width =\"100\">Subscript</th><th align =\"left\">Value</th></thead><tbody>" );
    /* output contents of theArray */
    document.writeln( "</tbody><thead><th width =\"100\">Subscript</th><th align =\"left\">Value</th></thead><tbody>" );
    document.writeln( "</tbody></table>" );
}
```

The for loop adds five elements to Array n2 and initialize each element to its subscript number (0 to 4).

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

The second time function outputArray 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>
11.4 Examples Using Arrays

Fig. 11.3 Initializing the elements of an array.

<table>
<thead>
<tr>
<th>Subscript</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Array n1 contains

<table>
<thead>
<tr>
<th>Subscript</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Array n2 contains
11.4 Examples Using Arrays

• Possible to declare and initialize in one step
  – Specify list of values
    • Initializer list
      ```javascript
      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”
      ```javascript
      var n = [ 10, 20, , 40, 50 ];
      ```
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">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
<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);
}
</script>
</head>
</html>

The for loop sums the values contained in the 10-element integer array called theArray.
for (var element in theArray)
    total2 += theArray[element];

document.writeln("<br />Total using for...in: 
    total2");

// -->
</script>

</head><body onload = "start()"></body>
</html>
11.4 Examples Using Arrays

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

Array colors contains

<table>
<thead>
<tr>
<th>Subscript</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>cyan</td>
</tr>
<tr>
<td>1</td>
<td>magenta</td>
</tr>
<tr>
<td>2</td>
<td>yellow</td>
</tr>
<tr>
<td>3</td>
<td>black</td>
</tr>
</tbody>
</table>

Array integers1 contains

<table>
<thead>
<tr>
<th>Subscript</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

Array integers2 contains

<table>
<thead>
<tr>
<th>Subscript</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>undefined</td>
</tr>
<tr>
<td>2</td>
<td>undefined</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>
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/javascript">
        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];
        }
    </script>
</head>
</html>

Referencing Array frequency replaces the switch statement used in Chapter 10’s example.
document.writeln("<table border = "1" width = "100%">
  <thead><tr><th align = "left">Face</th><th align = "left">Frequency</th></tr></thead>
  <tbody>
    <tr><td>1</td><td> frequency[ face ]</td></tr>
    <tr><td>2</td><td> frequency[ face ]</td></tr>
    <tr><td>3</td><td> frequency[ face ]</td></tr>
    <tr><td>4</td><td> frequency[ face ]</td></tr>
    <tr><td>5</td><td> frequency[ face ]</td></tr>
    <tr><td>6</td><td> frequency[ face ]</td></tr>
  </tbody></table>

// -->
</script>
</head>
<body>
  <p>Click Refresh (or Reload) to run the script again</p>
</body>
</html>
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">
<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'];
</script>
</head>
</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
<?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</h2>
  "The values of the original array are: ", a );

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

  outputArray( a );

  // array a passed call-by-reference

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

</script>
</head>
</html>
```javascript
// 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.

```javascript
modifyElement( a[3] );
```


```javascript
document.writeln(
    "<br />a[3] after modifyElement: " + a[3];
}
```

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.
```javascript
// 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.
11.7 Passing Arrays to Functions

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

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
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 );
}

Function compareIntegers calculates the difference between the integer values of its arguments.
11.8 Sorting Arrays

Fig. 11.9 Sorting an array with `sort`.

```
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
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 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";
}

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.
// 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()" />
</p>

    <p>Result<br />
    <input name = "result" type = "text" size = "30" />
</form>
</body>
</html>
11.9 Searching Arrays: Linear Search and Binary Search

Fig. 11.10 Linear search of an array.