*JavaScript*
*and AJAX*
2nd Edition

# Tutorial 3

Working with Arrays, Loops,
and Conditional Statements

## Objectives

- Create an array
- Populate and reference values from an array
- Work with array methods
- Work with For loops
- Work with While loops

## Objectives

- Loop through the contents of an array
- Work with If, If... Else, and multiple conditional statements
- Use arrays, loops, and conditional statements to create a table
- Work with break, continue, and label commands

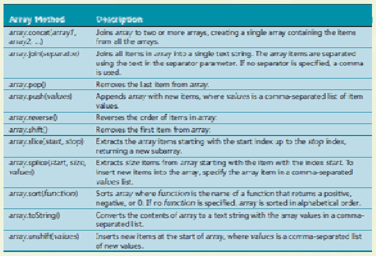
## Working with Arrays

- An **array** is a collection of data values organized under a single name
  - Each individual data value is identified by an **index**
- To create an array:
  - `var array = new Array(length);`
- To populate an array:
  - `array[i] = value;`
  - `var array = [values];`
- To create and populate an array:
  - `var array = new Array(values);`

## Specifying Array Length

- To determine the size of an array, use the property:
  - `array.length`
- To add more items to an array, run the command:
  - `array[i] = value;`
- To remove items from an array, run the command:
  - `array.length = value;`

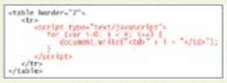
## Using Array Methods



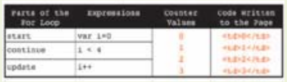
| Array Method | Description |
|---|---|
| *array*.concat(*array1*, *array2*, ...) | Joins *array* to two or more arrays, creating a single array containing the items from all the arrays. |
| *array*.join(*separator*) | Joins all items in *array* into a single text string. The array items are separated using the text in the separator parameter. If no separator is specified, a comma is used. |
| *array*.pop() | Removes the last item from *array*. |
| *array*.push(*values*) | Appends *array* with new items, where *values* is a comma-separated list of item values. |
| *array*.reverse() | Reverses the order of items in *array*. |
| *array*.shift() | Removes the first item from *array*. |
| *array*.slice(*start*, *stop*) | Extracts the array items starting with the *start* index up to the *stop* index, returning a new subarray. |
| *array*.splice(*start*, *size*, *values*) | Extracts *size* items from *array* starting with the item with the index *start*. To insert new items into the array, specify the array item in a comma-separated *values* list. |
| *array*.sort(*function*) | Sorts *array* where *function* is the name of a function that returns a positive, negative, or 0. If no *function* is specified, *array* is sorted in alphabetical order. |
| *array*.toString() | Converts the contents of *array* to a text string with the array values in a comma-separated list. |
| *array*.unshift(*values*) | Inserts new items at the start of *array*, where *values* is a comma-separated list of new values. |

# Working with Program Loops

- A **program loop** is a set of commands that is executed repeatedly until a stopping condition has been met
  - For loop
    - A **counter variable** tracks the number of times a set of commands is run
    - The collection of commands that is run each time through a loop is collectively known as a **command block**



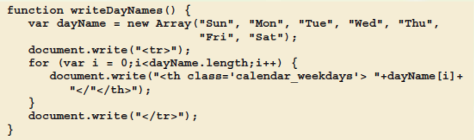
```
<table border="2">
   <tr>
      <script type="text/javascript">
         for (var i=0; i < 4; i++) {
            document.write("<td>" + i + "</td>");
         }
      </script>
   </tr>
</table>
```



| Parts of the For Loop | Expressions | Counter Values | Code Written to the Page |
|---|---|---|---|
| start | var i=0 | 0 | `<td>0</td>` |
| continue | i < 4 | 1 | `<td>1</td>` |
| update | i++ | 2 | `<td>2</td>` |
|  |  | 3 | `<td>3</td>` |

# Working with Program Loops

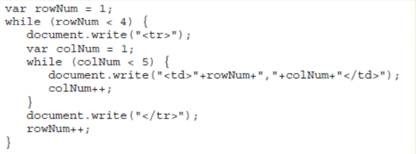
- For loops are often used to cycle through the different values contained within an array



```
function writeDayNames() {
   var dayName = new Array("Sun", "Mon", "Tue", "Wed", "Thu",
                           "Fri", "Sat");
   document.write("<tr>");
   for (var i = 0;i<dayName.length;i++) {
      document.write("<th class='calendar_weekdays'> "+dayName[i]+
         "</"</th>");
   }
   document.write("</tr>");
}
```

# Working with Program Loops

- A while loop does not depend on the value of a counter variable; it runs as long as a specific condition is met



```
var rowNum = 1;
while (rowNum < 4) {
   document.write("<tr>");
   var colNum = 1;
   while (colNum < 5) {
      document.write("<td>"+rowNum+","+colNum+"</td>");
      colNum++;
   }
   document.write("</tr>");
   rowNum++;
}
```

# Creating Program Loops

- To create a For loop, use the syntax:

```
for (start; continue; update) {
    commands
}
```

- To create a While loop, use the following syntax:

```
while (continue) {
    commands
}
```

# Creating Program Loops

- To create a Do/While loop, use the following syntax:

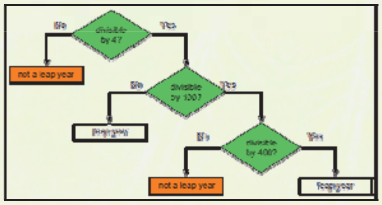
```
do {
    commands
    }
while (continue);
```

- To loop through the contents of an array, enter the For loop:

```
for (var i = 0; i < array.length; i++) {
    commands involving array[i]
}
```

# Working with Conditional Statements

- A **conditional statement** is a statement that runs a command or command block only when certain circumstances are met



## Working with Conditional Statements

- To test a single condition, use the construction:

```
if (condition) {
    commands
}
```

- To test between two conditions, use the following construction:

```
if (condition) {
    commands if condition is true
} else {
    commands if otherwise
}
```

## Working with Conditional Statements

- To test multiple conditions, use the construction:

```
if (condition 1) {
    first command block
} else if (condition 2) {
    second command block
} else if (condition 3) {
    third command block
} else {
    default command block
}
```

## Creating a Switch Statement

- To create a Switch statement to test for different values of an expression, use the structure:

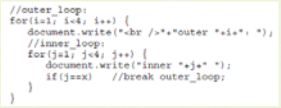
```
switch (expression) {
    case label1: commands1
    break;
    case label2: commands2
    break;
    case label3: commands3
    break;
...
    default: default commands
}
```

## Managing Program Loops and Conditional Statements

- The break command terminates any program loop or conditional statement
- The syntax for the break command is:
  - `break;`
- The continue command stops processing the commands in the current iteration of the loop and jumps to the next iteration
  - `continue;`

## Managing Program Loops and Conditional Statements

- Labels are used to identify statements in JavaScript code so that you can reference those statements elsewhere in a program
  - `label: statement`
  - `break label;`
  - `continue label;`



```
//outer_loop:
for(i=1; i<4; i++) {
   document.write("<br />"+"outer "+i+": ");
   //inner_loop:
   for(j=1; j<4; j++) {
      document.write("inner "+j+" ");
      if(j==x)   //break outer_loop;
   }
}
```

## Using Multidimensional Arrays

- A matrix is a multidimensional array in which each item is referenced by two or more index values
- In a matrix, each value is referenced by a row index number and column index number
- Although matrices are commonly used in various programming languages, JavaScript does not support them