Chapter 16

How to work with Arrays

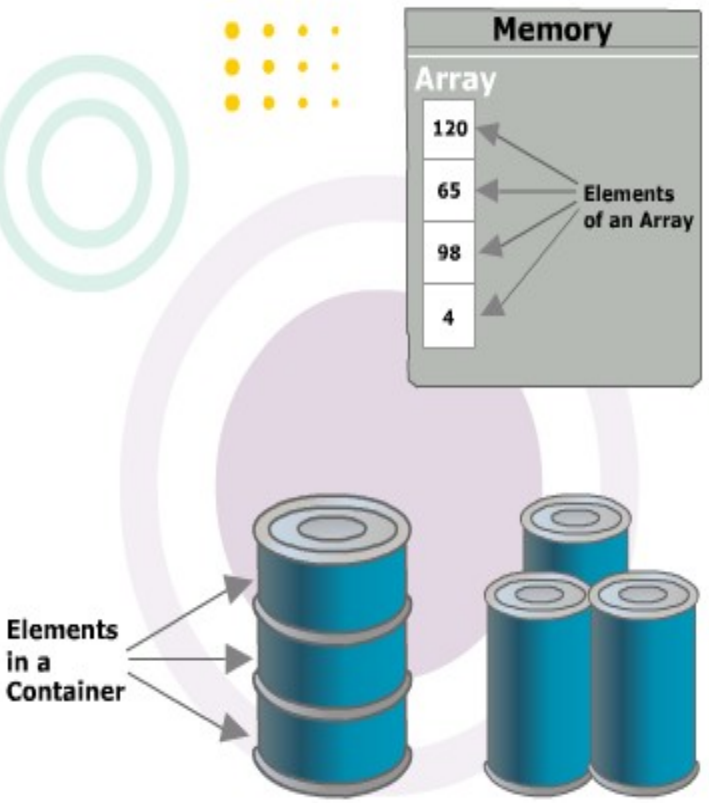
Objectives

- How to create and use an array
- How to use methods of an Array object
- The Task List application
- Other skills for working with arrays
- The Task List 2.0 application

How to create and use an array

Introduction to Array

- An array can store one or more elements
- The length of an array is the number of elements in the array
- Each of the element in the array is accessible through an integer value called a subscript
- An array subscript begins from zero and is also called array index



How to create an array

The syntax for creating an array

Using the new keyword with the Array object name

```
var arrayName = new Array(length);
```

Using the brackets literal

```
var arrayName = [];
```

The syntax for creating an array and assigning values in one statement

Using the new keyword with the Array object name

```
var arrayName = new Array(arrayList);
```

Using the brackets literal

```
var arrayName = [arrayList];
```

How to create an array and assign values in one statement

```
var rates = new Array(14.95, 12.95, 11.95, 9.95);
var names = ["Ted Lewis", "Sue Jones", "Ray Thomas"];
```

How to create an array (cont.)

The syntax for referring to an element of an array

```
arrayName[index]
```

Code that refers to the elements in an array

```
rates[2]          // Refers to the third element in the rates array
names[1]          // Refers to the second element in the names array
```

How to assign values to an array by accessing each element

How to assign rates to an array that starts with four undefined elements

```
var rates = new Array(4);
rates[0] = 14.95;
rates[1] = 12.95;
rates[2] = 11.95;
rates[3] = 9.95;
```

How to assign strings to an array that starts with no elements

```
var names = [];
names[0] = "Ted Lewis";
names[1] = "Sue Jones";
names[2] = "Ray Thomas";
```

How to add and delete array elements

One property and one operator for an array

Property	Description
`length`	The number of elements in an array.
Operator	**Description**
`delete`	Deletes the contents of an element and sets the element to undefined, but doesn't remove the element from the array.

How to add an element to the end of an array

```
var numbers = [1, 2, 3, 4];       // array is 1, 2, 3, 4
numbers[numbers.length] = 5;      // array is 1, 2, 3, 4, 5
```

How to add an element at a specific index

```
var numbers = [1, 2, 3, 4];     // array is 1, 2, 3, 4
numbers[6] = 7;                 // array is 1, 2, 3, 4, undefined, undefined, 7
```

How to delete a number at a specific index

```
var numbers = [1, 2, 3, 4];       // array is 1, 2, 3, 4
delete numbers[2];                // array is 1, 2, undefined, 4
```

How to use for loops to work with arrays

Code that puts the numbers 1 through 10 into an array

```
var numbers = [];
for (var i = 0; i < 10; i++) {
    numbers[i] = i + 1;
}
```

Code that displays the numbers array created above

```
var numbersString = "";
for (var i = 0; i < numbers.length; i++) {
    numbersString += numbers[i] + " ";
}
alert (numbersString);
```

The message that's displayed

How to use for loops to work with arrays (cont.)

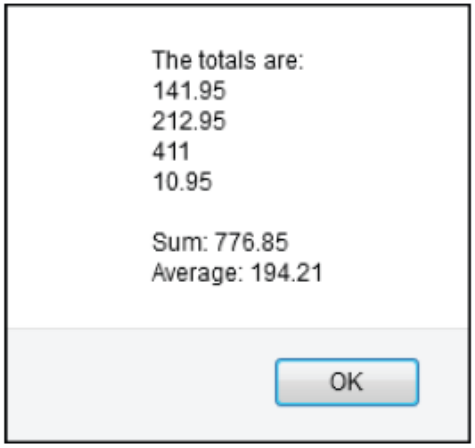
Code that computes the sum and average of an array of totals

```
var totals = [141.95, 212.95, 411, 10.95];
var sum = 0;
for (var i = 0; i < totals.length; i++) {
    sum += totals[i];
}
var average = sum / totals.length;
```

Code that displays the totals array, the sum, and the average

```
var totalsString = "";
for (var i = 0; i < totals.length; i++) {
    totalsString += totals[i] + "\n";
}
alert ("The totals are:\n" + totalsString + "\n" +
       "Sum: " + sum.toFixed(2) + "\n" + "Average: " + average.toFixed(2) );
```

The message that's displayed



How to use for-in loops to work with arrays

The syntax of a for-in loop

```
for (var elementIndex in arrayName) {
    // statements that access the elements
}
```

A for-in loop that displays the numbers array in a message box

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
var numbersString = "";
for (var index in numbers) {          // The start of the for-in loop
    numbersString += numbers[index] + " ";
}
alert(numbersString);
```

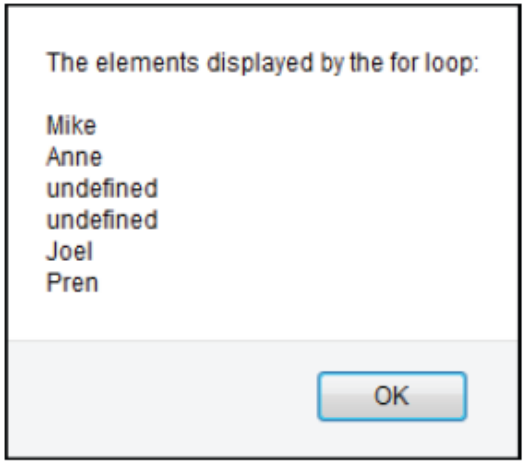
The message that's displayed

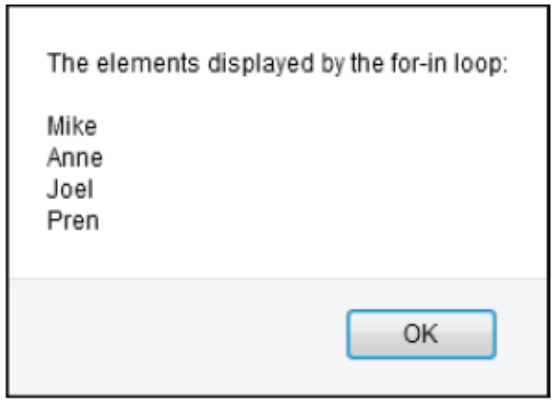
How to use for-in loops to work with arrays (cont.)

Code that shows the difference between for and for-in loops

```
var names = ["Mike", "Anne", "Ray"];
names[4] = "Joel";                  // Array is Mike, Anne, Ray, undefined, Joel
names[names.length] = "Pren";       // Pren is added to the array
delete names[2];                    // Ray is deleted from the array

var namesString1 = "The elements displayed by the for loop:\n\n";
for (var i = 0; i < names.length; i++) {
    namesString1 += names[i] + "\n"; }    // Includes undefined elements

var namesString2 = "The elements displayed by the for-in loop:\n\n";
for (var i in names) {
    namesString2 += names[i] + "\n"; }    // Omits undefined elements

alert (namesString1);
alert (namesString2);
```

The messages that are created by the for and the for-in loops





How to use the methods of an Array object

Methods of an Array object that access simple parameters

Method	Description
`push(element_list)`	Add one or more elements to the end of the array and returns the new length of the array.
`pop()`	Remove the last element in the array, decrements the length and returns the element that it removed.
`unshift(element_list)`	Add one or more elements to the begining of the array and returns the new length of the array.
`shift()`	Remove the first element in the array, decrements the length and returns the element that it removed.
`reverse()`	Reverses the order of the elements in the array.
`splice(start, number)`	Remove the number of elements from index is start and returns the elements that were removed.
`splice(start, number, element_list)`	Remove the number of elements from index is start, replace them by element in element_list and returns the elements that were removed.

Methods of an Array object that access simple parameters(cont.)

Method	Description
`slice(start,number)`	Return a new array with number of element in number parameter and from index in start parameter.
`concat(array_list)`	Return a new array that consists of the original array concatenated with the array in the array list.
`join([separator])`	Convert all the elements of array to string and concatenates them separated with separator parameter
`toString()`	Same join() method with separator is comma.
`toLocaleString()`	Same toString() method but using a locale specific separator
`isArray(object)`	Checks whether the object passed to it is an array.
`indexOf(value, start)`	Return the first index at which value is found. Return -1 if the value is not found.
`lastIndexOf(value, start)`	Return the last index at which value is found. Return -1 if the value is not found.

Methods of an Array object that access functions as parameters

Method	Description
`sort([comparision function])`	Accepts an optional function to change the default sort order, if no parameter ascending sort will be apply.
`forEach(function, this)`	Accepts a function that is executed once for each element. Returns a value of undefined.
`every(function, this)`	Accepts a function that tests each element in the array to meet a specific condition. Returns true if all element pass the test, false otherwise.
`some(function, this)`	Accepts a function that tests each element in the array to meet a specific condition. Returns true if at least element passes the test, false otherwise.
`map(function, this)`	Accepts a function that is execute one for each element, and returns a new array containing the result of each function call.
`filter(function, this)`	Accepts a function that is execute one for each element, and returns a new array containing the element that meet the specific condition of the function.

Methods of an Array object that access functions as parameters (cont.)

Method	Description
`reduce(function, init)`	Accept a function that returns all the elements reduced to one value, processed in ascending order.
`reduceRight(function, init)`	Accept a function that returns all the elements reduced to one value, processed in descending order.

Examples of the Array methods

How to use the push and pop methods to add and remove elements

```
var names = ["Mike", "Anne", "Joel"];
names.push("Ray", "Pren");          // names is Mike, Anne, Joel, Ray, Pren
var removedName = names.pop();      // removedName is Pren
alert (names.join());               // displays Mike,Anne,Joel,Ray
```

How to use the unshift and shift methods to add and remove elements

```
var names = ["Mike", "Anne", "Joel"];
names.unshift("Ray", "Pren");       // names is Ray, Pren, Mike, Anne, Joel
removedName = names.shift();        // removedName is Ray
alert (names.join());               // displays Pren,Mike,Anne,Joel
```

How to use the join and toString methods

```
var names = ["Mike", "Anne", "Joel", "Ray"];
alert (names.join());               // displays Mike,Anne,Joel,Ray
alert (names.join(", "));           // displays Mike, Anne, Joel, Ray
alert (names.toString());           // displays Mike,Anne,Joel,Ray
```

Examples of the Array methods (cont.)

- How to use the sort() method
  - For Alphanumeric sorting

```
var names=[“Grace”, “Charles”, “Ada”, “Alan”, “Linus”];
names.sort();    //names is Ada,Alan,Charles,Grace,Linus
```

  - For numeric sorting in ascending sequence

```
var comparision = function(x, y){
    return x-y;
};
Var numbers = [520, 33, 9, 199];
numbers.sort(comparision);    //numbers is 9,33,199,520
```

Examples of the Array methods (cont.)

- How to use the map() method

```
var numbers=[1, 4, 9, 16];
var squared = numbers.map(function( value ){
    return value * value;
});
var root = numbers.map( Math.sqrt ); //Root is 1,2,3,4
```

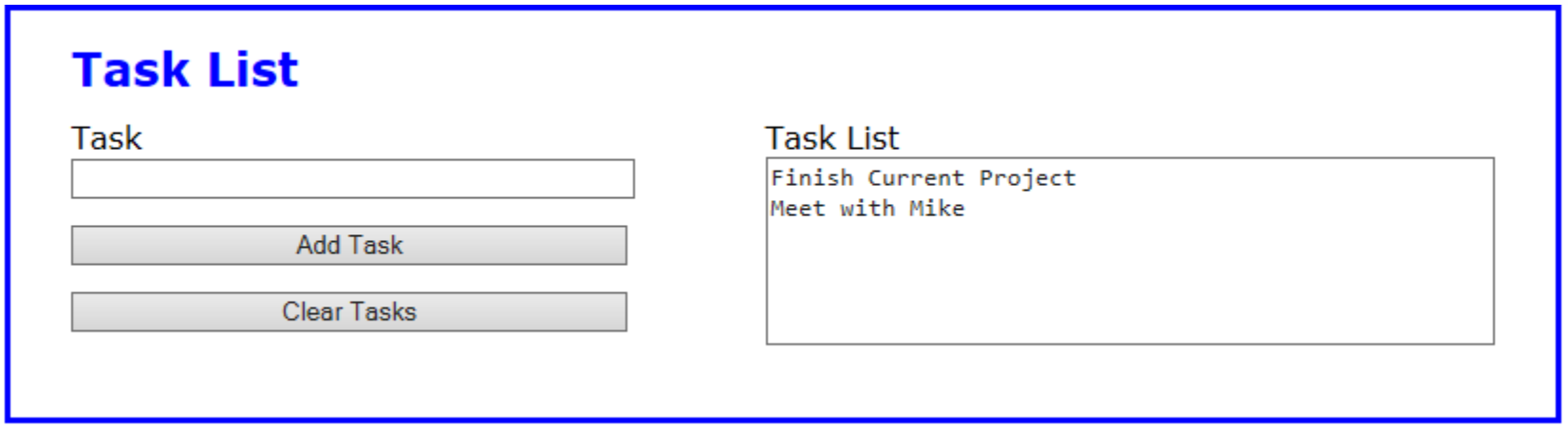
- How to use filter() method

```
var numbers =
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20];
var checkPrime = function( value ){
    var isPrime = true;
    for( var i=2; i<value; i++){
        if(value % i === 0){
            isPrime = false; break;
        }
    }
    return isPrime;
}
Var prime = numbers.filter(checkPrime);
                //Prime is 1,2,3,7,11,13,17,19
```

The Task List application

The Task List application

- The User Interface



The Task List application

- The HTML Code

```
<main>
  <h1>Task List</h1>
  <div id="tasks">
      <label for="task_list">Task List</label><br>
      <textarea id="task_list" rows="6" cols="50"></textarea>
  </div>

  <label for="task">Task</label><br>
  <input type="text" name="task" id="task"><br>

  <input type="button" name="add_task" id="add_task" value="Add Task"><br>
  <input type="button" name="clear_tasks" id="clear_tasks" value="Clear Tasks">
</main>
```

The Task List application

- The JavaScript Code

```
$(document).ready(function(){
    var tasks = [];

    var displayTaskList = function() {
        tasks.sort();

        $("#task_list").val( tasks.join("\n") );
        $("#task").focus();
    };

    $("#add_task").click(function() {
        var textbox = $("#task");
        var task = textbox.val();
        if (task === "") {
            alert("Please enter a task.");
            textbox.focus();
        } else {
            // add new task to tasks array
            tasks.push( task );

            // clear task text box and re-display tasks
            textbox.val( "" );
            displayTaskList();
        }
    });

    $("#clear_tasks").click(function() {
        tasks = [];
        $("#task_list").val( "" );
        $("#task").focus();
    });

    // set focus on initial load
    $("#task").focus();
});
```

Other skills for working with arrays

How to use a String method to create an array

- A String method that creates an array

Method	Description
`split(separator,limit)`	Split a string into an array.

- Example use split() method to create an array

```
var fullName= “Grace M Hopper”;
var nameParts = fullName.split(“ ”); //Create an array
alert(nameParts.lenght);      //display 3
alert(nameParts)       //display Grace,M,Hopper
var lastName = nameParts[nameParts.length -1];
alert(lastName);
```

How to create and use an associative array

- When you create an associative array, you use string as the indexes instead of numbers.
- How to create an associative array with 4 elements

```
var item = [];
item[“itemCode”] = 123;
item[“itemName”] = “HTML5”;
item[“itemCost”] = 54.5;
item[“itemQuantity”] = 5;
alert(item.length);              //display 0
alert(Object.keys(item).length); //display 4
```

How to create and use an associative array (cont.)

- How to add an element to the associative array

```
item[“lineCost”] =(item[“itemCost”] *
        item[“itemQuantity”]).toFixed(2);
```

- How to retrieve and display the elements in the associative array

```
alert(“Item elements: \n” +
    ”\nCode = ” + item[“itemCode”] +
    “\nName = ”+ item[“itemName”] +
    “\nCost = ” + item[“itemCost”] +
    “\nQuantity = ” + item[“itemQuantity”] +
    “\nLine Cost =” + item[“lineCose”]);
```

How to create and use an array of arrays

- How to create and use an array of arrays
  - Code that create an array of arrays

```
var testScores = [];
var testScores[0] = [80, 82, 90, 87, 85];
var testScores[1] = [79, 80, 74];
var testScores[2] = [93, 95, 89, 100];
var testScores[3] = [60, 72, 75, 71];
```

  - Code that refer elements in the array of arrays

```
alert(testScores[0][1]);   //display 82
alert(testScores[2][3]);   //display 100
```

How to create and use an array of arrays (cont.)

- How to create and use an array of associative arrays
  - Code that create an array of arrays

```
var invoice = [];
```

  - Code that add an associative array to invoice array

```
invoice[0] = [];
invoice[0][“itemCode”] = 123;
invoice[0][“itemName”] = “HTML5”;
invoice[0][“itemCost”] = 54.5;
invoice[0][“itemQuantity”] = 5;
```

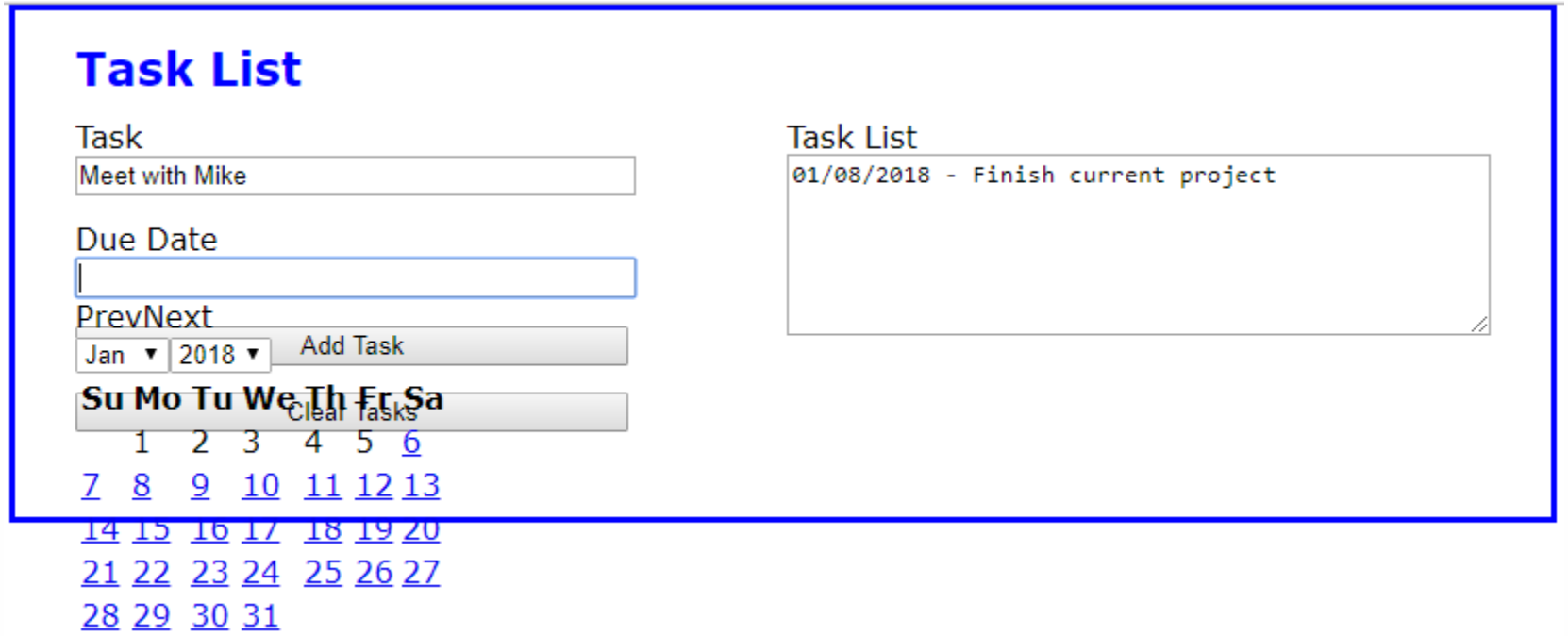
  - Code that refer elements in the array of associative arrays

```
alert(invoice[0][“itemCode”]);//display 123
alert(invoice[0][“itemName”]);//display HTML5
```

The Task List 2.0 application

The Task List 2.0 application

- The User Interface



The Task List 2.0 application

- The HTML Code

```
<main>
  <h1>Task List</h1>
  <div id="tasks">
      <label for="task_list">Task List</label><br>
      <textarea id="task_list" rows="6" cols="50"></textarea>
  </div>

  <label for="task">Task</label><br>
  <input type="text" name="task" id="task"><br>

  <label for="due_date">Due Date</label><br>
  <input type="text" name="due_date" id="due_date"><br>

  <input type="button" name="add_task" id="add_task" value="Add Task"><br>
  <input type="button" name="clear_tasks" id="clear_tasks" value="Clear Tasks">
</main>
```

The Task List 2.0 application

- The JavaS

```
$(document).ready(function(){
    var displayTaskList = function() {
        var taskString = localStorage.C16tasks || "";
        if (taskString.length > 0) {
            // create array to hold task arrays
            var tasks = [];

            // split string on first delimiter. Then, loop array, split
            // strings on second delimiter, and store array in tasks array
            var interim = taskString.split( "|" );
            for (var i = 0; i < interim.length - 1; i++) {
                tasks.push( interim[i].split( "~~" ));
            }

            // sort array of arrays by due date
            tasks.sort(function(arr1, arr2) {
                var a = new Date(arr1[1]); // 2nd element of first array
                var b = new Date(arr2[1]); // 2nd element of second array

                if ( a < b ) { return -1; }
                else if ( a > b ) { return 1; }
                else { return 0; }
            });

            // reduce sorted array of arrays to a single string
            taskString = tasks.reduce( function( prev, current ) {
                return prev + current[1] + " - " + current[0] + "\n";
            }, ""); // pass initial value for prev parameter
        }

        // display tasks string and set focus on task text box
        $("#task_list").val( taskString );
        $("#task").focus();
    };
```

The Task List 2.0 application

- The JavaScript Code

```
$("#add_task").click(function() {
    var task = $("#task").val();
    var dueDate = $("#due_date").val();

    if (task === "" || dueDate === "") {
        alert("Please enter a task and due date.");
        $("#task").focus();
    } else {
        // retrieve tasks and create array for new task
        var taskString = localStorage.C16tasks || "";
        var newTask = [task, dueDate];

        // add new task to end of task string in local storage
        localStorage.C16tasks = taskString + newTask.join( "~~" ) + "|";

        // clear task text boxes and re-display tasks
        $("#task").val("");
        $("#due_date").val("");
        displayTaskList();
    }
});
```

The Task List 2.0 application

- The JavaScript Code

```
    $("#clear_tasks").click(function() {
        localStorage.removeItem("C16tasks");
        $("#task_list").val("");
        $("#task").focus();
    });

    $("#due_date").datepicker({
        changeMonth: true,
        changeYear: true,
        minDate: 0
    });

    // display tasks on initial load
    displayTaskList();
});
```

Summary

- An **array** can store one or more elements. The length of an array is the number of elements in the array.
- Each of the element in the array is accessible through an integer value called a subscript. Subscript start from 0.
- You can use **for loops** or **for-in loops** to works with array elements.
- You can use the **split() method** of a String object to create an array from substrings within a string.
- An **associative array** uses strings for the indexes instead of numbers.
- In an **array of arrays**, each element in one array contains another array.

The End.