# Chapter 4

How to work with JavaScript objects, functions, and events

# Objectives

- How to use objects to work with data
- How to use functions
- How to handle events
- Two illustrative applications

# How to use objects to work with data

# How to use the window and document objects

- *The **window object*** is the global object, and JavaScript lets you omit the object name and dot operator when referring to the window object.
- The ***document object*** is the object that lets you work with the Document Object Model(DOM).

# How to use the window and document objects (cont.)

### Another method of the window object that displays a dialog box

| Method | Description |
|---|---|
| `confirm(string)` | Displays a dialog box that contains the string in the parameter, an OK button, and a Cancel button. If the user clicks OK, true is returned. If the user clicks Cancel, false is returned. |

### Two methods of the window object for working with numbers

| Method | Description |
|---|---|
| `parseInt(string)` | Converts the string that's passed to it to an integer data type and returns that value. If it can't convert the string to an integer, it returns NaN. |
| `parseFloat(string)` | Converts the string that's passed to it to a decimal data type and returns that value. If it can't convert the string to a decimal value, it returns NaN. |

### Examples of window methods

```
confirm("Are you sure you want to delete it?");

var entryA = prompt("Enter any value", 12345.6789);
entryA = parseInt(entryA);                          // entryA = 12345

var entryB = prompt("Enter any value", 12345.6789);
entryB = parseFloat(entryB);                        // entryB = 12345.6789
```

# How to use the window and document objects (cont.)

## Three methods of the document object

| Method | Description |
|---|---|
| `getElementById(id)` | Gets the HTML element that has the id that's passed to it and returns that element. |
| `write(string)` | Writes the string that's passed to it into the document. |
| `writeln(string)` | Writes the string and advances to a new line. |

## Examples of document methods

```
// returns the object for the HTML element
var rateBox = document.getElementById("rate");

// writes a line into the document
document.writeln("Today is " + today.toDateString());
```

# How to use Textbox and Number object

- The **Textbox object** is one of the DOM object.
- When you assign a numeric value to a variable, a **Number object** is created. The you can use the Number method with the variable.

# How to use Textbox and Number object (cont. )

### One method of the Textbox object

| Method | Description |
|---|---|
| `focus()` | Moves the cursor into the text box, but doesn't return anything. |

### Two properties of the Textbox object

| Property | Description |
|---|---|
| `value` | A string that represents the contents of the text box. |
| `disabled` | A Boolean value that controls whether the text box is disabled. |

### One method of the Number object

| Method | Description |
|---|---|
| `toFixed(digits)` | Returns a string representation of the number after it has been rounded to the number of decimal places in the parameter. |

# How to use Textbox and Number object (cont. )

## HTML tags that define two text boxes

```
<input type="text" id="first_name">
<input type="text" id="sales_amount">
```

## How to use the value property to get the value from a text box

### Without chaining

```
var firstName = document.getElementById("first_name");
firstName = firstName.value;
```

### With chaining

```
var firstName = document.getElementById("first_name").value;
```

## How to use the parseFloat method to get a number value from a text box

### Without chaining

```
var salesAmount = document.getElementById("sales_amount");
salesAmount = salesAmount.value;
salesAmount = parseFloat(salesAmount);
```

### With chaining

```
var salesAmount = parseFloat(document.getElementById("sales_amount").value);
```

# How to use Date and String objects

## The syntax for creating a JavaScript object and assigning it to a variable

```
var variableName = new ObjectType();
```

### A statement that creates a Date object

```
var today = new Date();
```

## A few of the methods of a Date object

| Method | Description |
|---|---|
| `toDateString()` | Returns a string with the formatted date. |
| `getFullYear()` | Returns the four-digit year from the date. |
| `getDate()` | Returns the day of the month from the date. |
| `getMonth()` | Returns the month number from the date. The months are numbered starting with zero. January is 0 and December is 11. |

## Examples that use a Date object

```
var today = new Date();                 // creates Date object with current date
alert ( today.toDateString() );         // displays Fri Mar 09 2012 on 3/9/2012
alert ( today.getFullYear() );          // displays 2012
alert ( today.getDate() );              // displays 9
alert ( today.getMonth() );             // displays 2, not 3 for March
```

# How to use Date and String objects (cont.)

## One property of a String object

| Method | Description |
|---|---|
| `length` | Returns the number of characters in the string. |

## A few of the methods of a String object

| Method | Description |
|---|---|
| `indexOf(search,position)` | Searches for the first occurrence of the search string starting at the position specified or zero if position is omitted. If found, it returns the position of the first character, counting from 0. If not found, it returns -1. |
| `substr(start,length)` | Returns the substring that starts at the specified position (counting from zero) and contains the specified number of characters. |
| `toLowerCase()` | Returns a new string with the letters converted to lowercase. |
| `toUpperCase()` | Returns a new string with the letters converted to uppercase. |

## Examples that use a String object

```
var name = "Ray Harris";
var nameUpper = name.toUpperCase();      // nameUpper = RAY HARRIS
var nameLength = name.length;            // nameLength = 10
var index = name.indexOf(" ");           // index = 3
var firstName = name.substr(0, index);   // firstName = Ray
```

# How to use functions

# How to create and call a function expression

- A **function** is a block of statements that perform an action.
- It can receive parameters and return a value by issuing a return statement.

# How to create and call a function expression (cont.)

**The syntax for a function**

```
var variableName = function(parameters) {
    // statements that run when the function is executed
}
```

**A function with no parameters that doesn't return a value**

```
var showYear = function() {
    var today = new Date();
    alert( "The year is " + today.getFullYear() );
}
```

**How to call the function**

```
showYear();
```

# How to create and call a function expression (cont.)

## A function with one parameter that returns a DOM element

```
var $ = function (id) {
    return document.getElementById(id);
}
```

### How to call the function

```
var emailAddress1 = $("email_address1").value;
```

## A function with two parameters that returns a value

```
var calculateTax = function ( subtotal, taxRate ) {
    var tax = subtotal * taxRate;
    tax = parseFloat( tax.toFixed(2) );
    return tax;
}
```

### How to call the function

```
var subtotal = 85.00;
var taxRate = 0.05;
var salesTax = calculateTax( subtotal, taxRate );   // calls the function
alert(salesTax);                                    // displays 4.25
```

# How to create and call a function declaration

- A **function declaration** is one that is coded with a name and is'nt assigned to a variable.
- The syntax for a function declaration

```
function functionName (parameters) {
    // statements that run when the function is executed
}
```

- A function declare with no parameters that does'nt return a value

```
function() showYear {
    var today = new Date();
    alert( "The year is " + today.getFullYear() );
}
```

**How to call the function**

```
showYear();
```

# How to create and call a function declaration (cont.)

- A function declare with one parameter that return a DOM element

```
function $(id){
    return document.getElementById(id);
}
```

How to call the function

```
var emailAddress1 = $("email_address1").value;
```

- A function declare with one parameter that return a DOM element

```
function calculateTax(subtotal, taxRate){
    var tax = subtotal * taxRate;
    tax = tax.toFixed(2);
    return tax;
}
```

How to call the function

```
var subtotal = 85.00;
var taxRate = 0.05;
var salesTax = calculateTax(subtotal, taxRate);
alert(salesTax);
```

# How to use local and global variable

- Variables that are created inside a function are **local variables**, and local variables can only be refered to by the code within the function.
- Variables created outside of function are **global variables**, and the code in all functions has access to them.
- The scope of a variable or function determines what code has access to it.

# How to use local and global variable (cont.)

**A function that uses a local variable named tax**

```
var calculateTax = function ( subtotal, taxRate ) {
    var tax = subtotal * taxRate;      // tax is a local variable
    tax = parseFloat( tax.toFixed(2) );
    return tax;
}
```

**Referring to a local variable from outside the function causes an error**

```
alert("Tax is " + tax);                // causes error
```

**A function that uses a global variable named tax**

```
var tax;                               // tax is a global variable
var calculateTax = function ( subtotal, taxRate ) {
    tax = subtotal * taxRate;
    tax = parseFloat( tax.toFixed(2) );
}
```

**Referring to a global variablefrom outside the function doesn't cause an error**

```
alert("Tax is " + tax);                // will not cause error
```

# How to use local and global variable (cont.)

**A function that inadvertently uses a global variable named tax**

```
var calculateTax = function ( subtotal, taxRate ) {
    tax = subtotal * taxRate;   // no var keyword so tax is treated as global
    tax = parseFloat( tax.toFixed(2) );
}
```

**Referring to the tax variable from outside the function doesn't cause an error...but it should!**

```
alert("Tax is " + tax);                    // will not cause error
```

# How to use strict mode

- When you use strict mode, if you forget to code the var keyword in the variable declaration or if you misspell a variable name that has been declared, the JavaScript engine will throw an error.
- The strict mode directive

```
"use strict"; //goes at the top of a file or function
```

- A function that inadvertently uses a global variable named tax

```
var calculateTax = function(subtotal, taxRate){
    tax = subtotal * taxRate;
    tax = tax.toFixed(2);
};
```

Reference te tax variable outside the function

```
alert("Tax is " + tax);    //will not cause error
```

# How to use strict mode (cont.)

- The same function in strict mode

```
“use strict”;
var calculateTax = function(subtotal, taxRate){
   tax = subtotal * taxRate;
   tax = tax.toFixed(2);
};
```

Reference te tax variable outside the function

```
alert(“Tax is ” + tax); //cause error
```

# When to use local and global variable

- Best coding practices
  - Use local variables whenever possible.
  - Use var keyword to declare all variables
  - Use strict mode
  - Declare the variables that are used in a function at the start of the function.

# How to handle events

# Event and Event handler

- Event is a action from user or another system like click, mouse over...
- Event handler is a functions is  executed when an event occurs.

# Event and Event handler (cont.)

## Common events

| Object | Event | Occurs when... |
|---|---|---|
| `window` | `load` | The document has been loaded into the browser. |
| `button` | `click` | The button is clicked. |
| `control or link` | `focus` | The control or link receives the focus. |
| | `blur` | The control or link loses the focus. |
| `control` | `change` | The user changes the value in the control. |
| | `select` | The user selects text in a text box or text area. |
| `element` | `click` | The user clicks on the element. |
| | `dblclick` | The user double-clicks on the element. |
| | `mouseover` | The user moves the mouse over the element. |
| | `mousein` | The user moves the mouse into the element. |
| | `mouseout` | The user moves the mouse out of the element. |

# How to attach and event handler to an event

- The syntax for attaching an event handler

```
objectVariable.oneventName = eventHandlerName;
```

- An event handler named joinList

```
var joinList = function(){
    alert("The statements for the function go here")
};
```

How to attach the event handler to the click event of a button

```
$("submit_button").onclick = joinList;
```

How to attach the event handler to the double-click event of a text box

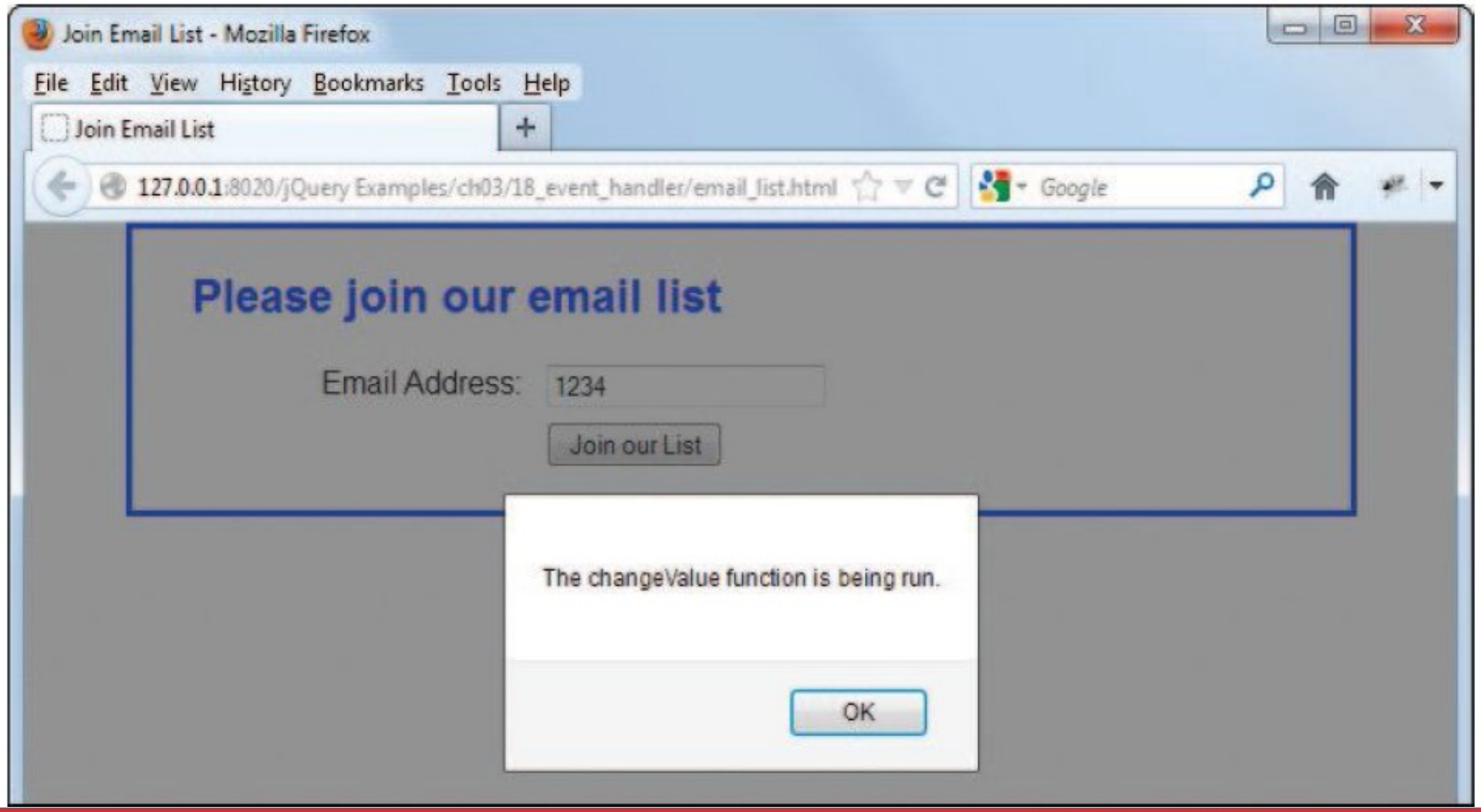
```
$("text_box_1").ondblclick = joinList;
```

- How to create and attach an event handler in one step

```
Window.onload = function(){
    alert("This is the window onload event handler function.");
}
```

# How to use an onload event handler to attach other event handlers

**The web browser after the Email Address has been changed**



# How to use an onload event handler to attach other event handlers (cont.)

## The HTML

```
<h1>Please join our email list</h1>
<label for="email_address">Email Address:</label>
<input type="text" id="email_address" name="email_address"><br>

<label> </label>
<input type="button" id="join_list" value="Join our List"><br>
```

## The JavaScript

```
// the $ function
var $ = function (id) {
    return document.getElementById(id);
}
// the event handler for the click event of the button
var joinList = function () {
    alert("The joinList function is being run.");
}
// the event handler for the onchange event of the text box
var changeValue = function () {
    alert("The changeValue function is being run.");
}
// the event handler for the onload event that attaches two event handlers
window.onload = function () {
    $("join_list").onclick = joinList;            // attaches 1st handler
    $("email_address").onchange = changeValue;    // attaches 2nd handler
}
```

# The Miles Per Gallon application

&

# The Email List application

Page 128 - 135

# Summary

- *The **window object*** is the global object, and JavaScript lets you omit the object name and dot operator when referring to the window object.
- The ***document object*** is the object that lets you work with the Document Object Model(DOM).
- A **function** is a block of statements that perform an action. It can receive parameters and return a value by issuing a return statement.
- Event is a action from user or another system like click, mouse over...
- Event handler is a functions is  executed when an event occurs.