Chapter 2 (Part B)

A JavaScript subset for jQuery users

The relational operators

Operator	Description	Example
`==`	Equal	`lastName == "Hopper"` `testScore == 10`
`!=`	Not equal	`firstName != "Grace"` `months != 0`
`<`	Less than	`age < 18`
`<=`	Less than or equal	`investment <= 0`
`>`	Greater than	`testScore > 100`
`>=`	Greater than or equal	`rate / 100 >= 0.1`

The syntax of the global isNaN method

```
isNaN(expression)
```

Examples of the isNaN method

```
isNaN("Hopper")      // Returns true
isNaN("123.45")      // Returns false
```

The logical operators in order of precedence

Operator	Description	Example
`!`	NOT	`!isNaN(age)`
`&&`	AND	`age > 17 && score < 70`
`||`	OR	`isNaN(rate) || rate < 0`

How the logical operators work

- Both tests with the AND operator must be true for the overall test to be true.
- At least one test with the OR operator must be true for the overall test to be true.
- The NOT operator switches the result of the expression to the other Boolean value.
- To override the order of precedence when two or more logical operators are used in a conditional expression, you can use parentheses.

Terms

- conditional expression
- relational operator
- compound conditional expression
- logical operator

The syntax of the if statement

```
if ( condition-1 ) { statements }
[ else if ( condition-2 ) { statements }
  ...
  else if ( condition-n ) { statements } ]
[ else { statements } ]
```

An if statement with an else clause

```
if ( age >= 18 ) {
    alert ("You may vote.");
} else {
    alert ("You are not old enough to vote.");
}
```

An if statement with else if and else clauses

```
if ( isNaN(rate) ) {
    alert ("You did not provide a number for the rate.");
} else if ( rate < 0 ) {
    alert ("The rate may not be less than zero.");
} else if ( rate > 12 ) {
    alert ("The rate may not be greater than 12.");
} else {
    alert ("The rate is: " + rate + ".");
}
```

An if statement with a compound conditional expression

```
if ( isNaN(userEntry) || userEntry <= 0 ) {
    alert ("Please enter a number greater than zero.");
}
```

How to test a Boolean variable

To see if it's true

```
if ( isValid == true ) { }
if ( isValid ) { }              // same as isValid == true
```

To see if it's false

```
if ( isValid == false ) { }
if ( !isValid == true ) { }
if ( !isValid ) { }             // same as !isValid == true
```

The syntax of a while loop

```
while ( condition ) { statements }
```

A while loop that adds the numbers from 1 through 5

```
var sumOfNumbers = 0;
var numberOfLoops = 5;
var counter = 1;
while (counter <= numberOfLoops) {
    sumOfNumbers += counter;
    counter++;                      // adds 1 to counter
}
alert(sumOfNumbers);                // displays 15
```

The syntax of a do-while loop

```
do { statements } while ( condition );
```

A do-while loop that gets a user entry until it is a number

```
do {
    var investment = prompt(
        "Enter investment amount as xxxxx.xx", 10000);
    investment = parseFloat(investment);
    if ( isNaN(investment) ) {
        alert("Investment must be a number");
    }
while ( isNaN(investment) );
```

The syntax of a for loop

```
for ( counterInitialization;
      condition;
      incrementExpression ) {
    statements
}
```

A for loop that calculates the future value of an investment

```
var investment = 10000;
var annualRate = 7.0;
var years = 10;
var futureValue = investment;
for ( var i = 1; i <= years; i++ ) {
    futureValue += futureValue * annualRate / 100;
}
alert (futureValue);                    // displays 19672
```

Terms

- if statement
- if clause
- else if clause
- else clause
- nested if statements
- while statement
- while loop
- do-while statement
- do-while loop
- for statement
- for loop
- counter for a loop
- index for a loop

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 = [];
```

A statement that creates an array

```
var totals = [];
```

The syntax for referring to an element of an array

```
arrayName[index]
```

Statements that refer to elements in an array

```
totals[2]          // refers to the third element
totals[1]          // refers to the second element
```

How to add values to an array

```
totals[0] = 141.95;
totals[1] = 212.25;
totals[2] = 411;
```

The syntax for getting the length property of an array

```
arrayName.length
```

A statement that gets the length of an array

```
var count = totals.length;
```

How to add a value to the end of an array

```
totals[totals.length] = 135.75;
```

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 in the array

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

The message that's displayed

Code that puts four totals in an array

```
var totals = [];
totals[0] = 141.95;
totals[1] = 212.25;
totals[2] = 411;
totals[3] = 135.75;
```

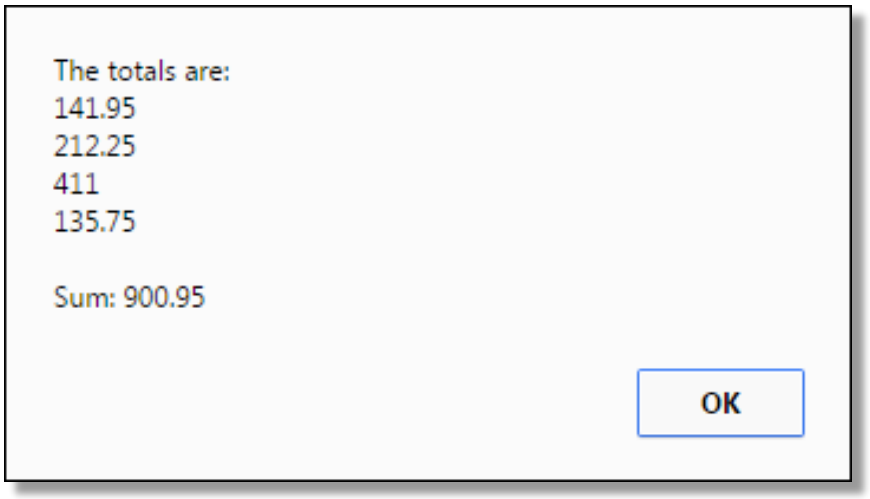
Code that sums the totals in the array

```
var sum = 0;
for (var i = 0; i < totals.length; i++) {
    sum += totals[i];
}
```

Code that displays the totals and the sum

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

The message that's displayed



Terms

- array
- length of an array
- index for an array

The syntax for a function expression

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

A function expression with one parameter that returns a DOM element

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

A statement that calls the $ function

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

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

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

A statement that calls the showYear function

```
showYear();
```

The syntax for a function declaration

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

A function declaration with two parameters that returns a value

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

A statement that calls the calculateTax function

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

Terms

- function
- calling a function
- function expression
- function declaration
- parameter
- passing parameters
- return statement

A function that uses a local variable named tax

```
var calculateTax = function ( subtotal, taxRate ) {
    var tax = subtotal * taxRate;      // tax is local
    tax = tax.toFixed(2);
    return tax;
}
alert("Tax is " + tax);                // causes an error
```

A function that uses a global variable named tax

```
var tax;                                    // tax is global
var calculateTax = function ( subtotal, taxRate ) {
    tax = subtotal * taxRate;
    tax = tax.toFixed(2);
}
alert("Tax is " + tax);                     // does not cause error
```

A function that inadvertently uses a global variable named tax

```
var calculateTax = function ( subtotal, taxRate ) {
    tax = subtotal * taxRate;  // tax is global
    tax = tax.toFixed(2);
}
alert("Tax is " + tax);        // does not cause error
```

The same function in strict mode

```
"use strict";                  // the strict mode directive
var calculateTax = function ( subtotal, taxRate ) {
    tax = subtotal * taxRate;  // error is thrown
    tax = tax.toFixed(2);
}
alert("Tax is " + tax);        // causes an error
```

Best coding practices for variables

- Use local variables whenever possible.
- Use strict mode.
- Declare the variables for a function at the start of the function.

Terms

- scope
- local variable
- local scope
- global variable
- global scope
- strict mode

Common events

Object	Event
window	load
button	click
control or link	focus
	blur
control	change
	select
element	click
	dblclick
	mouseover
	mousein
	mouseout

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.");
}
```

Terms

- event handler
- attach an event handler

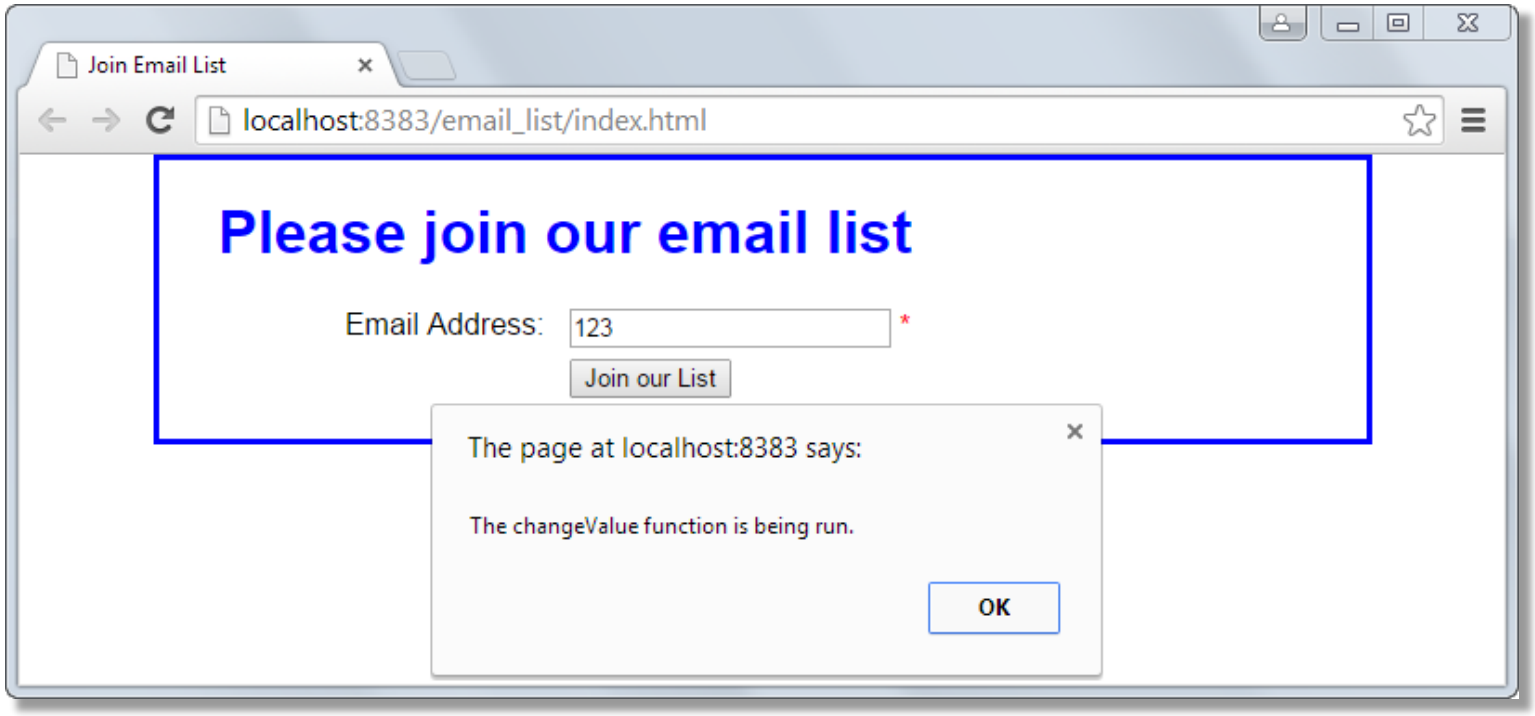
The HTML for a page

```
<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>
```

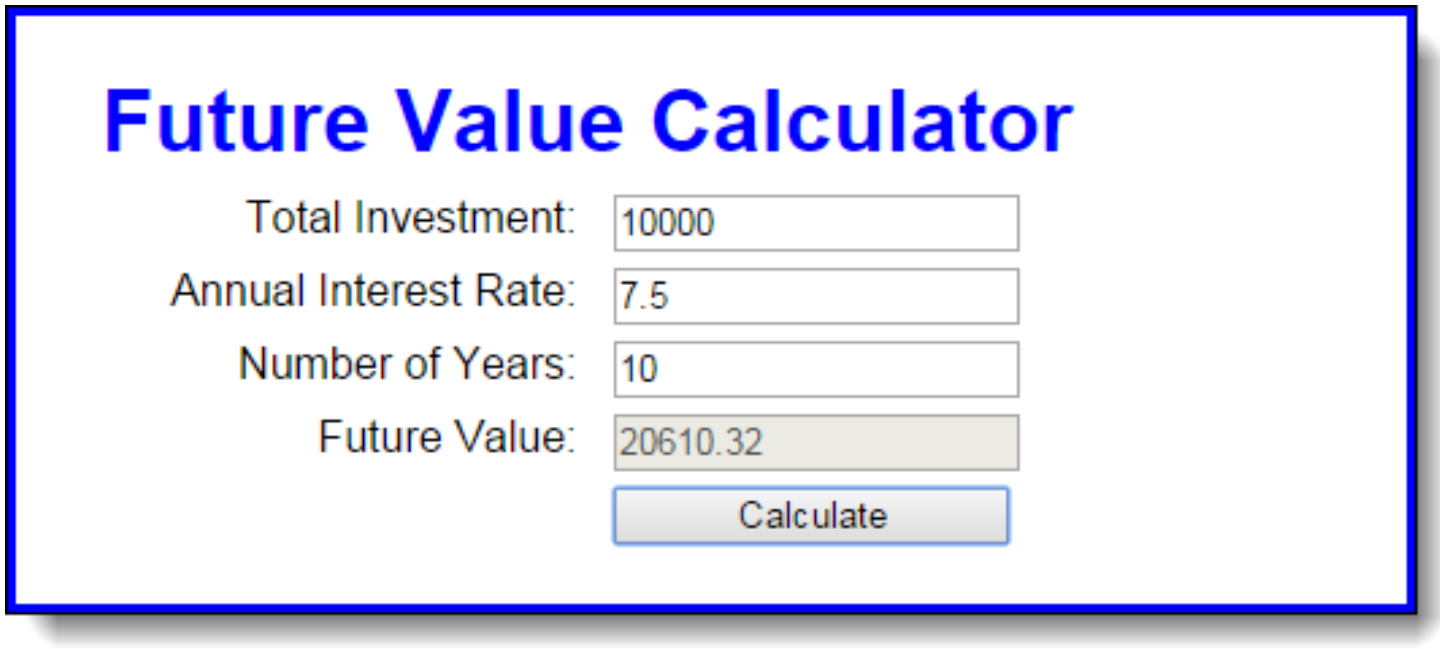
JavaScript that attaches two event handlers in the onload event handler

```
// 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 change event of the text box
var changeValue = function () {
    alert("The changeValue function is being run.");
}

// the event handler for the load event
window.onload = function () {
    $("join_list").onclick = joinList;
    $("email_address").onchange = changeValue;
}
```

The web browser after the Email Address has been changed



The Future Value application



The HTML for the application (index.html)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Future Value Calculator</title>
    <link rel="stylesheet" href="future_value.css">
    <script src="future_value.js"></script>
</head>
<body>
    <main>
        <h1>Future Value Calculator</h1>

        <label for="investment">Total Investment:</label>
        <input type="text" id="investment"><br>
```

The HTML for the application (continued)

```
            <label for="rate">Annual Interest Rate:</label>
            <input type="text" id="annual_rate"><br>

            <label for="years">Number of Years:</label>
            <input type="text" id="years"><br>

            <label for="future_value">Future Value:</label>
            <input type="text" id="future_value"
                   disabled><br>

            <label> </label>
            <input type="button" id="calculate"
                   value="Calculate"><br>
    </main>
</body>
</html>
```

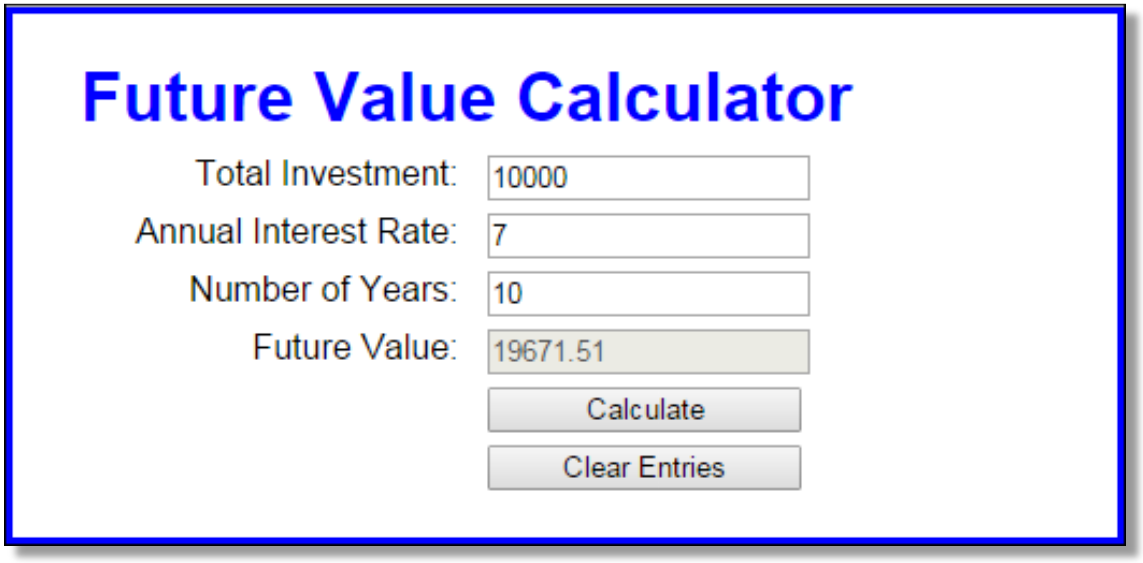
The JavaScript for the application (future_value.js)

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

var calculateFV = function(investment, rate, years) {
    var futureValue = investment;
    for (var i = 1; i <= years; i++ ) {
        futureValue += futureValue * rate / 100;
    }
    futureValue = futureValue.toFixed(2);
    return futureValue;
}
```

The JavaScript for the application (continued)

```
var processEntries = function() {
    var investment = parseFloat( $("investment").value );
    var rate = parseFloat( $("annual_rate").value );
    var years = parseInt( $("years").value );

    if (isNaN(investment) || isNaN(rate) || isNaN(years)) {
        alert("One or more entries is invalid");
    }
    else {
        $("future_value").value =
            calculateFV(investment, rate, years);
    }
}
window.onload = function () {
    $("calculate").onclick = processEntries;
    $("investment").focus();
}
```

Exercise 2-1 Enhance the Future Value application



Provide better data validation and add a clear button

Exercise 2-2 Build a Miles Per Gallon application