Chapter 2

Getting started with JavaScript

Objectives

- How to code JavaScript
- The JavaScript syntax
- How to work with JavaScript data
- How to use objects, methods and properties
- Two illustrative application

How to code JavaScript

JavaScript example

- JavaScript in HTML document

```
<p>
    <script>
        var today = new Date();
        document.write("Current date: ");
        document.write(today.toDateString());
    </script>
</p>
<p>
    <script>
        var today = new Date();
        document.write("&copy; ");
        document.write(today.getFullYear());
        document.write(", San Joaquin Valley Town Hall")
    </script>
</p>
```

Three ways to include JavaScript in a web page

- A script element in the head section that loads an *external JavaScript*

```
<script src="calculate_mpg.js"></script>
```

- A script element that embedded JavaScripts in the head section

```
<head>
    ...
    <script>
        alert("The Calculate MPG application");
        var miles = prompt("Enter miles driven");
        miles = parseFloat(miles);
        var gallons = prompt("Enter gallons of gas used");
        gallons = parseFloat(gallons);
        var mpg = miles/gallons;
        mpg = parseInt(mpg);
        alert("Miles per gallon = " + mpg);
    </script>
</head>
```

Three ways to include JavaScript in a web page (cont.)

- A script element that embedded JavaScripts in the body section

```
<p>
    <script>
        var today = new Date();
        document.write("Current date: ");
        document.write(today.toDateString());
    </script>
</p>
<p>&copy; 
    <script>
        var today = new Date();
        document.write( today.getFullYear() );
    </script>
, San Joaquin Valley Town Hall
</p>
```

The result of the JavaScript in a web browser

Current date: Mon Mar 12 2012

© 2012 , San Joaquin Valley Town Hall"

The JavaScript Syntax

How to code JavaScript statements

- Syntax:

```
Var function_name = function(){
   statements;
}
```

A block of JavaScript code

```
var joinList = function () {
    var emailAddress1 = $("email_address1").value;
    var emailAddress2 = $("email_address2").value;

    if (emailAddress1 == "") {
        alert("Email Address is required.");
    } else if (emailAddress2 == "") {
        alert("Second Email Address is required.");
    } else if (emailAddress1 !== emailAddress2) {
        alert("Second Email entry must equal first entry.");
    } else if ($("first_name").value == "") {
        alert("First Name is required.");
    } else {
        $("email_form").submit();
    }
}
```

How to code JavaScript statements

- A JavaScript statements has a syntax that’s similar to the syntax of Java
- The basic syntax rules:
  - JavaScript is case-sensitive
  - Each JavaScript statement ends with a semicolon.
  - JavaScript ignores extra whitespace within statements

How to create identifiers

- Identifiers are the name given to variables, functions, objects, properties and methods.
- Rules for creating identifiers
  - Identifiers can only contain letters, numbers, the underscore, and the dollar sign.
  - Identifiers can't start with a number.
  - Identifiers are case-sensitive.
  - Identifiers can be any length.
  - Identifiers can't be the same as *reserved words*.
  - Avoid using global properties and methods as identifiers. If you use one of them, you won't be able to use the global property or method with the same name.

How to create identifiers (cont.)

Valid identifiers in JavaScript

```
subtotal          index_1          $
taxRate           calculate_click  $log
```

Camel casing versus underscore notation

```
taxRate           tax_rate
calculateClick    calculate_click
emailAddress      email_address
firstName         first_name
futureValue       future_value
```

Naming recommendations

- Use meaningful names for identifiers. That way, your identifiers aren't likely to be reserved words or global properties.
- Be consistent: Either use camel casing (taxRate) or underscores (tax_rate) to identify the words within the variables in your scripts.
- If you're using underscore notation, use lowercase for all letters.

How to create identifiers (cont.)

Reserved words in JavaScript

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

- You can’t create an identifier with a reserved words.

How to use comments

- *Comments* let you add descriptive notes to your code.
- *Comments* are ignored when JavaScript is executed.
- *JavaScripts* provides two forms of comments:
  - Single line comment:

    ```
    //This is single line comement
    ```

  - Block comment:

    ```
    /* This is
    Block comment */
    ```

How to use comments (cont.)

```
/* this onload function sets up the events that display and hide
   the text that follows a series of h2 headings
*/
window.onload = function () {
    var fiveReasons = $("five_reasons");          // gets a div element

    // gets the h2 and div elements within the div element
    var h2Headings = fiveReasons.getElementsByTagName("h2");
    var divTags = fiveReasons.getElementsByTagName("div");

    var i, headingNode, divNode;
    for (i = 0; i < h2Headings.length; i++ ) {    // one loop for each h2
        headingNode = h2Headings[i];
        divNode = divTags[i];

        // Attaches an event handler for each h2
        headingNode.onclick = function () {
            var h2 = this;
            if (h2.nextElementSibling.getAttribute("class") == "closed") {
                h2.nextElementSibling.setAttribute("class", "open");
            }
            else {
                h2.nextElementSibling.setAttribute("class", "closed");
            }
        }
    }
}
```

How to work with JavaScript data

The primitive data types

- Date type decide what type of data will store in a variable.
- Javascript's primitive data types

Data type	Description
Number	Represents an integer or a decimal value that can start with a positive or negative sign.
String	Represents character (string) data.
Boolean	Represents a Boolean value that has two possible state true or false.

The primitive data types (cont.)

Examples of number values

```
15                  // an integer
-21                 // a negative integer
21.5                // a decimal value
-124.82             // a negative decimal value
-3.7e-9             // floating-point notation for -0.0000000037
```

Examples of string values

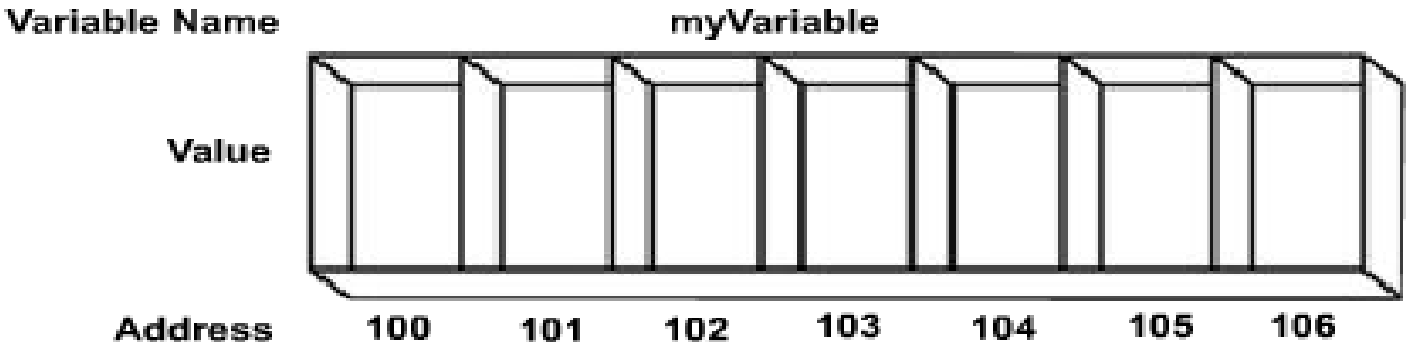
```
"JavaScript"        // a string with double quotes
'String Data'       // a string with single quotes
""                  // an empty string
```

The two Boolean values

```
true                // equivalent to true, yes, or on
false               // equivalent to false, no, or off
```

How to use variables

- A variable is a location in the computer's memory is stored and retrieved later



How to use variables

- Syntax to declare and assign value to a variable

```
var variableName;
variableName = value;
```

- Examples

```
var counter;
counter = 1;
var sum, average;
sum = 0; average = 0;
```

How to use variables (cont.)

- Syntax to declare and assign value to a variable in one statement

```
var variableName = value;
```

- Examples

```
var counter = 1;
var subtotal = 74.95;
var name = “Joshep”;
var isValid = false;
```

How to code arithmetic expressions

Common arithmetic operators

Operator	Description	Example	Result
+	Addition	5 + 7	12
-	Subtraction	5 - 12	-7
*	Multiplication	6 * 7	42
/	Division	13 / 4	3.25
%	Modulus	13 % 4	1
++	Increment	counter++	adds 1 to counter
--	Decrement	counter--	subtracts 1 from counter

The order of precedence for arithmetic expressions

Order	Operators	Direction	Description
1	++	Left to right	Increment operator
2	--	Left to right	Decrement operator
3	* / %	Left to right	Multiplication, division, modulus
4	+ -	Left to right	Addition, subtraction

How to code arithmetic expressions (cont.)

Examples of precedence and the use of parentheses

```
3 + 4 * 5            // Result is 23 since the multiplication is done first
(3 + 4) * 5          // Result is 35 since the addition is done first

13 % 4 + 9           // Result is 10 since the modulus is done first
13 % (4 + 9)         // Result is 0  since the addition is done first

1000 + 1000 * .05    // Result is 1050 since the multiplication is done first
1000 + (1000 * .05)  // Result is still 1050
```

How to use arithmetic expressions in assignment statements

- Code that calculates sale tax

```
var subtotal = 200;
var taxPercent = .5;
var taxAmount = subtotal * taxPercent;
var total =  subtotal + taxAmount;
```

How to use arithmetic expressions in assignment statements (cont.)

- The most useful compound assignment operators

Operator	Description
+=	Adds the result of the expression to the variable.
-=	Subtracts the result of the expression from the variable.
*=	Multiplies the variable value by the result of the expression.

- Statements that use the compound assignment operators

```
var subtotal = 74.95;
Subtotal += 20.00;
var counter = 10;
counter -= 1;
var price = 100;
price *= .8;
```

How to work with strings variable

- The concatenation operators for strings

Operator	Description
+	Concatenates two values.
+=	Adds the result of the expression to the end of the variable

- Some of the escape sequences that can be used in strings

Operator	Description
\n	Starts a new line in a string.
\"	Puts a double quotation mark in a string.
\'	Puts a single quotation mark in a string.

How to work with strings variable (cont.)

How to declare string variables and assign values to them

```
var firstName = "Ray", lastName = "Harris";    // assigns two string values
var fullName = lastName + ", " + firstName;    // fullName is "Harris, Ray"
```

How to code compound assignment statements with string data

```
var firstName = "Ray", lastName = "Harris";
var fullName = lastName;                       // fullName is "Harris"
fullName += ", ";                              // fullName is "Harris, "
fullName += firstName;                         // fullName is "Harris, Ray"
```

How to code compound assignment statements with mixed data

```
var months = 120;
message = "Months: ";
message += months;                             // message is "Months: 120"
```

How escape sequences can be used in a string

```
var message = "A valid variable name\ncannot start with a number.";
var message = "This isn\'t the right way to do this.";
```

How to declare Boolean variables and assign values to them

```
var isValid = false;                           // Boolean value is false
```

How to use objects, methods, and properties

Introduction to object, methods, and properties

- **Object** is a collection of methods and properties.
- A **method** performs a function or does an action.
- A **property** is a data item that relates to the object.
- The **window object** is a global object for JavaScript.

Introduction to object, methods, and properties (cont.)

Common methods of the window object

Method	Description
`alert(string)`	Displays a dialog box that contains the string that's passed to it by the parameter along with an OK button.
`prompt(string,default)`	Displays a dialog box that contains the string in the first parameter, the default value in the second parameter, an OK button, and a Cancel button. When the user enters a value and clicks OK, that value is returned as a string. Or if the user clicks Cancel, null is returned.
`print()`	Issues a print request to the browser.

One property of the window object

Property	Description
`location`	The URL of the current web page.

Introduction to object, methods, and properties (cont.)

The syntax for calling a method of an object

```
objectName.methodName(parameters)
```

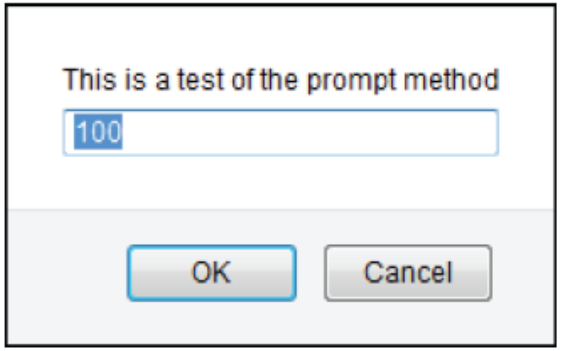
A statement that calls the alert method of the window object

```
window.alert("This is a test of the alert method");
```

A statement that calls the prompt method with the object name omitted

```
var userEntry = prompt("This is a test of the prompt method", 100);
```

The prompt dialog box that's displayed



The syntax for accessing a property of an object

```
objectName.propertyName
```

A statement that displays the location property of the window object

```
alert(window.location);        // Displays the URL of the current page
```

How to use the parseInt() and parseFloat() method

The parseInt and parseFloat methods of the window object

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 that use the parseInt and parseFloat methods

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

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

var entryC = prompt("Enter any value", "Hello");
alert(entryC);                                  // displays Hello
entryC = parseInt(entryC);
alert(entryC);                                  // displays NaN
```

How to use the write() and writelin() method of the document object

- The write() and writeln() methods of the document object

Method	Description
write(string)	Writes the string that’s passed to it into the document.
writeln(string)	Writes the string that’s passed to it into the document ending with a new line character.

How to use the write() and writelin() method of the document object (cont.)

- Example

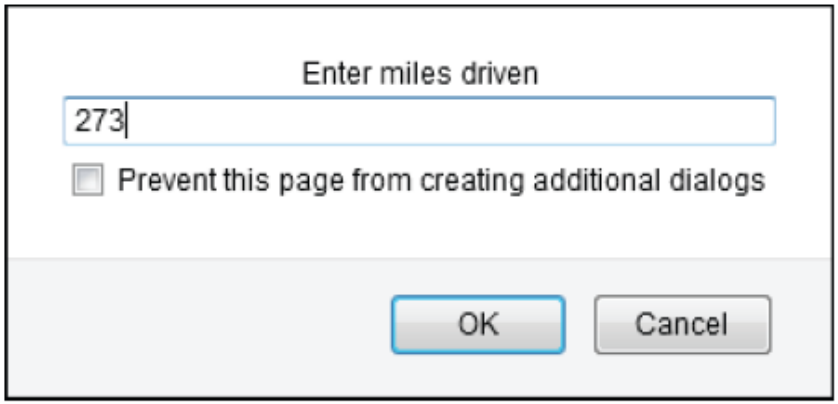
```
<body>
   <script>
    var today =  new Date()
    document.write(“<h1>Welcome to our site!</h1>”);
    document.write(“Today is ”);
    document.write(today.toDateString());
    document.write(“<br>”);
    document.writeln(“Today is ”);
    document.writeln(today.toDateString());
    document.write(“<br>”);
   </script>
</body>
```

Two illustrative application

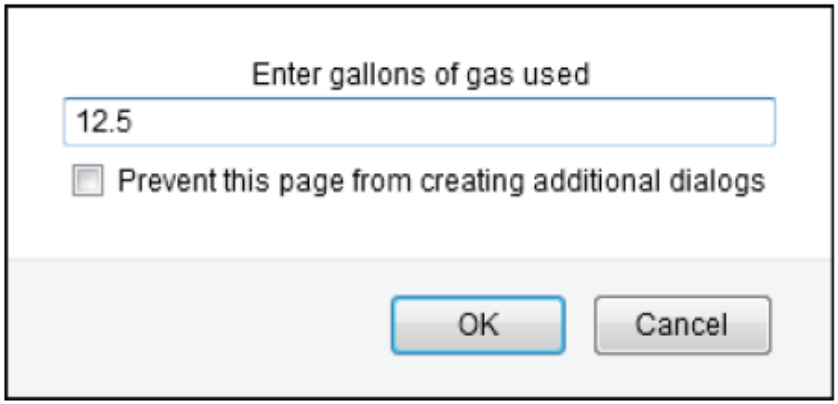
The Miles Per Gallan application

The dialog boxes for the Calculate MPG application

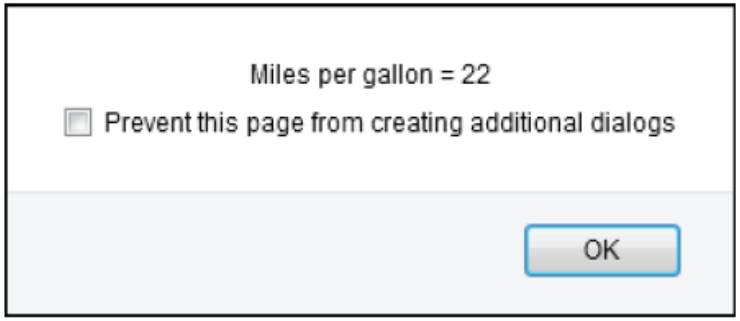
The first prompt dialog box



The second prompt dialog box



The alert dialog box that displays the result



The Miles Per Gallan application (cont)

The HTML and JavaScript for the application

```
<!doctype html>
<html>
<head>
    <title>The Calculate MPG Application</title>
    <script>
        alert("The Calculate MPG application");
        var miles = prompt("Enter miles driven");
        miles = parseFloat(miles);
        var gallons = prompt("Enter gallons of gas used");
        gallons = parseFloat(gallons);
        var mpg = miles/gallons;
        mpg = parseInt(mpg);
        alert("Miles per gallon = " + mpg);
    </script>
</head>
<body>
<section>
    <h1>This page is displayed after the JavaScript is executed</h1>
</section>
</body>
</html>
```

The Test Scores application

- On page 80 and 81

Summary

- There are three ways to include JavaScript in a web page: external scripts, embedded in head section and embedded in body section.
- A JavaScript statements has a syntax that’s similar to the syntax of Java.
- Javascript’s primitive data types: Number, String, Boolean.
- A variable is a location in the computer’s memory is stored and retrieved later.
- **Object** is a collection of methods and properties.
- A **method** performs a function or does an action.
- A **property** is a data item that relates to the object.
- The **window object** is a global object for JavaScript.