Working with Objects

Mohamed Mukthar Ahmed

Working with Objects

Outline

- Working with objects
- The window object
- The document object
- The Textbox & Number objects
- The Date & String objects
- Document Object Model (DOM)
- Anonymous Functions
- Named Functions
- Scope of variables
- Event Handling
  - Common Events
- Using onload Event Handler
- Application Code Analysis

Mohamed Mukthar Ahmed

Working with Objects

- We learnt the syntax for using the methods and properties of an object
- Let's learn some more objects, methods and properties that are needed for working with data

The window Object

- With the window apart from **alert** and **prompt** methods we have yet another method **confirm**

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.

The document Object

- The **document** object let's us work with Document Object Model (**DOM**) that represents all the HTML elements of the page
- It is the highest object in the DOM structure
- The **getElementById()** method is commonly used to get the object for the HTML element

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.

Mohamed Mukthar Ahmed

The document Object

- The **document** object let's us work with Document Object Model (**DOM**) that represents all the HTML elements of the page
- It is the highest object in the DOM structure
- The **getElementById()** method is commonly used to get the object for the HTML element

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());
```

Mohamed Mukthar Ahmed

The Textbox & Number Objects

- When we use the **getElementById()** method to get a textbox, the method return is a Textbox object
- The Textbox **value** property is use to get the value.

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 Textbox object

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

The Textbox & Number Objects

- When we declare a numeric variable, a Number object is created
- The **toFixed**(digits) method of the Number object

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.

The Textbox & Number Objects

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);
```

Other examples of chaining

```
var salesAmount =
    parseFloat(document.getElementById("sales_amount").value).toFixed(2);
document.getElementById("first_name").value = ""; // clear a text box
document.getElementById("first_name").focus();    // move focus to a text box
```


The Date & String Objects

- There isn't a primitive data type for dates. Thus we need to create a **Date** object before we use it's methods
- When we create a **Date** object, it is initialized with the current date and time

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.

The Date & String Objects

- When we declare a string variable, a **String** object is automatically created

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.

Mohamed Mukthar Ahmed

Document Object Model (DOM)

- As the browser loads an HTML page, it builds a DOM
- The DOM is a hierarchical collection of nodes in the web brower's memory that represents the current page
- JavaScript can modify the web page in the browser by modifying the DOM. Whenever the DOM is changed the browser displays the result
- To modify the text for an HTML element, we can use the **firstChild** property to get the descendent node and then the **nodeValue** property to access the text

The syntax for changing the text node for an element

```
elementObject.firstChild.nodeValue = "The text for the element";
```

An example that puts a message in the span element

```
document.getElementById("email_address_error").firstChild.nodeValue =
    "This entry is required";
```

Mohamed Mukthar Ahmed

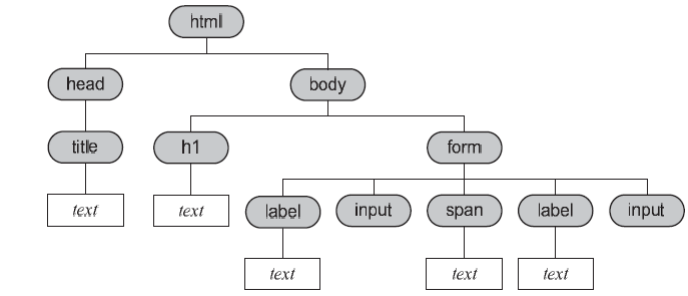
Document Object Model (DOM)

The code for a web page

```
<!DOCTYPE html>
<html>
<head>
    <title>Join Email List</title>
</head>
<body>
    <h1>Please join our email list</h1>
    <form id="email_form" name="email_form"
          action="join.html" method="get">
        <label for="email_address">Email Address:</label>
        <input type="text" id="email_address">
        <span id="email_address_error">*</span><br>
        <label> </label>
        <input type="button" id="join_list" value="Join our List">
    </form>
</body>
</html>
```


Document Object Model (DOM)

The DOM for the web page



Anonymous Function

- A JavaScript application needs to handle **events**
- We need to code and invoke functions that handle an event
- A **function** is a block of statements that can receive ***parameters*** and return a value by issuing the **return** statement
- Technically a function has no name, it is stored in a variable and is referenced to by the variable name. Thus it is called **Anonymous Function**
- Any anonymous function must be coded before any statement that calls it. Otherwise, an error will occur

Anonymous Function

The syntax for an anonymous function

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

An anonymous 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();
```

An anonymous 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;
```

Anonymous Function

An anonymous 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
```


Named Function

- Anonymous functions are commonly used to handle events
- JavaScript also allows us code Named Functions
- A named function is not stored in a variable and its name is coded after the keyword **function**
- In contrast to an anonymous function, a named function doesn't have to be coded before any statement calls it

The syntax for a named function

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

Named Function

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

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

How to call the function

```
showYear();
```

A named function with two parameters that returns a value

```
function calculateTax ( 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
```


Scope of Variables

- The scope of a variable or function determines what code has access to it
- Variables created inside a function have **local** scope
- Variables created outside the functions have **global** scope
- If we forget to code the **var** keyword in a variable declaration, JavaScript assumes that the variable is global
  - Debugging problem
- It is better to pass local variables from one function to another as parameters than use global variables
  - Easier to understand with less chances of errors

Event Handling

- JS applications commonly respond to user actions. These actions are called **events**
- Functions that handle the events are called **event handlers**. i.e. An event handler is a function that's executed when an event occurs
- To make that happen, we need to attach the functions to the events
- When we attach an event handler to an event, we don't code the parentheses after the function name

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.

- When we code an event for an event handler, we precede the event name with **on**.
- Example: onclick is used for click event

Event Handling

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

```
document.getElementById("submit_button").onclick = joinList;
```

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

```
document.getElementById("text_box_1").ondblclick = joinList;
```


Using onload Event Handler

- The event handler for the **onload** event of the window object can be used to attach the event handlers of other events after the DOM has been built
- NOTE : This function starts with **window.onload** so it is executed after the page is loaded and the DOM has been built

Using onload Event Handler

The HTML

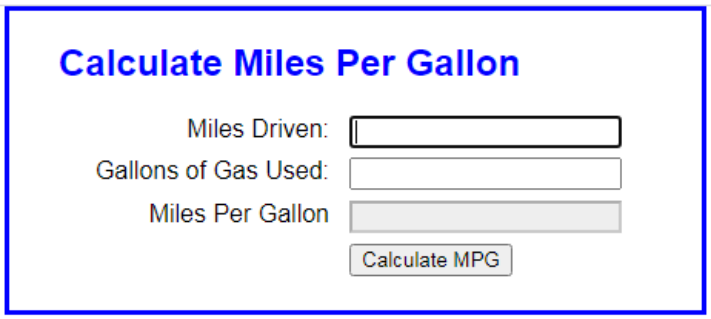
```
<h1>Please join our email list</h1>
<label for="email_address">Email Address:</label>
<input type="text" id="email_address" name="email_address"><br>

<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
}
```

Mohamed Mukthar Ahmed

MPG Application



Mohamed Mukthar Ahmed

MPG Application

```
8     <script>
9         var $ = function (id) {
10            return document.getElementById(id);
11        }
12        var calculateMpg = function () {
13            var miles = parseFloat($("miles").value);
14            var gallons = parseFloat($("gallons").value);
15
16            if (isNaN(miles) || isNaN(gallons)) {
17                alert("Both entries must be numeric");
18            }
19            else {
20                var mpg = miles / gallons;
21                $("mpg").value = mpg.toFixed(1);
22            }
23        }
24        window.onload = function () {
25            $("calculate").onclick = calculateMpg;
26            $("miles").focus();
27        }
28    </script>
```

Mohamed Mukthar Ahmed

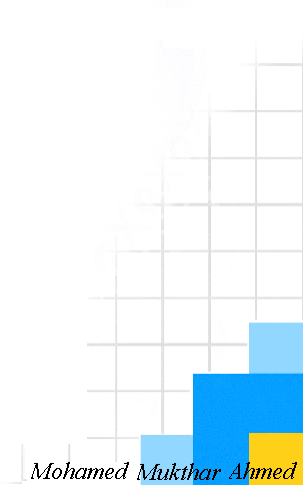
MPG Application

```
30 <body>
31     <section>
32         <h1>Calculate Miles Per Gallon</h1>
33         <label for="miles">Miles Driven:</label>
34         <input type="text" id="miles"><br>
35         <label for="gallons">Gallons of Gas Used:</label>
36         <input type="text" id="gallons"><br>
37         <label for="mpg">Miles Per Gallon</label>
38         <input type="text" id="mpg" disabled><br>
39         <label> </label>
40         <input type="button" id="calculate" value="Calculate MPG"><br>
41     </section>
42 </body>
```

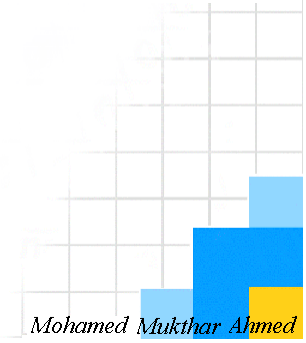
Mohamed Mukthar Ahmed

MPG Application

```
2  body {
3      font-family: Arial, Helvetica, sans-serif;
4      background-color: white;
5      margin: 0 auto;
6      width: 450px;
7      border: 3px solid blue;
8  }
9  h1 {
10     color: blue;
11 }
12 section {
13     padding: 0 2em 1em;
14 }
15 label {
16     float: left;
17     width: 11em;
18     text-align: right;
19     padding-bottom: .5em;
20 }
21 input {
22     margin-left: 1em;
23     margin-bottom: .5em;
24 }
```



Email List Application



Email List Application

```
<body>
    <section>
        <h1>Please join our email list</h1>
        <form id="email_form" name="email_form" action="join.html" method="get">
            <label for="email_address1">Email Address:</label>
            <input type="text" id="email_address1" name="email_address1">
            <span id="email_address1_error">*</span><br>

            <label for="email_address2">Re-enter Email Address:</label>
            <input type="text" id="email_address2" name="email_address2">
            <span id="email_address2_error">*</span><br>

            <label for="first_name">First Name</label>
            <input type="text" id="first_name" name="first_name">
            <span id="first_name_error">*</span><br>

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

Mohamed Mukthar Ahmed

Email List Application

```
1  var $ = function (id) {
2      return document.getElementById(id);
3  }
4  var joinList = function () {
5      var emailAddress1 = $("email_address1").value;
6      var emailAddress2 = $("email_address2").value;
7      var isValid = true;
8      // validate the first entry
9      if (emailAddress1 == "") {
10         $("email_address1_error").firstChild.nodeValue =
11                                     "This field is required.";
12         isValid = false;
13     } else {
14         $("email_address1_error").firstChild.nodeValue = "";
15     }
16     // validate the second entry
17     if (emailAddress2 == "") {
18         $("email_address2_error").firstChild.nodeValue =
19                                     "This field is required.";
20         isValid = false;
21     } else if (emailAddress1 !== emailAddress2) {
```

Mohamed Mukthar Ahmed

Email List Application

```
22          $("email_address2_error").firstChild.nodeValue =
23                                  "This entry must equal first entry.";
24          isValid = false;
25      } else {
26          $("email_address2_error").firstChild.nodeValue = "";
27      }
28      // validate the third entry
29      if ($("first_name").value == "") {
30          $("first_name_error").firstChild.nodeValue =
31                                      "This field is required.";
32          isValid = false;
33      } else {
34          $("first_name_error").firstChild.nodeValue = "";
35      }
36      // submit the form if all entries are valid
37      if (isValid) {
38          $("email_form").submit();
39      }
40  }
41  window.onload = function () {
42      $("join_list").onclick = joinList;
43      $("email_address1").focus();
44  }
```


Mohamed Mukthar Ahmed