JavaScript Basics & HTML DOM

Sang Shin
Java Technology Architect
Sun Microsystems, Inc.
sang.shin@sun.com
www.javapassion.com

Disclaimer & Acknowledgments

- Even though Sang Shin is a full-time employee of Sun Microsystems, the contents here are created as his own personal endeavor and thus does not necessarily reflect any official stance of Sun Microsystems on any particular technology
- Acknowledgments
  - The contents of this presentation was created from JavaScript tutorial from www.w3cschools.com

Topics

- What is and Why JavaScript?
- How and Where do you place JavaScript code?
- JavaScript language
- JavaScript functions
- JavaScript events
- JavaScript objects
- JavaScript HTML DOM objects
- Closure (need to be added)

What is and Why JavaScript?

What is JavaScript?

- Was designed to add interactivity to HTML pages
- Is a scripting language (a scripting language is a lightweight programming language)
- JavaScript code is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

What can a JavaScript do?

- JavaScript gives HTML designers a programming tool
- JavaScript can put dynamic text into an HTML page
- JavaScript can react to events
- JavaScript can read and write HTML elements
- JavaScript can be used to validate input data
- JavaScript can be used to detect the visitor's browser
- JavaScript can be used to create cookies

How and Where Do You Place JavaScript Code?

How to put a JavaScript code into an HTML page?

- Use the <script> tag (also use the type attribute to define the scripting language)

```
<html>
<head>
<script type="text/javascript">
...
</script>
</head>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```

Where Do You Place Scripts?

- Scripts can be in the either <*head*> section or <*body*> section
- Convention is to place it in the <*head*> section

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
```

Referencing External JavaScript File

- Scripts can be provided locally or remotely accessible JavaScript file using *src* attribute

```
<html>
<head>
<script language="JavaScript"
        type="text/javascript"
        src="http://somesite/myOwnJavaScript.js">
</script>
<script language="JavaScript"
        type="text/javascript"
        src="myOwnSubdirectory/myOwn2ndJavaScript.js">
</script>
```

JavaScript Language

JavaScript Variable

- You create a variable with or without the var statement

```
var strname = some value
strname = some value
```

- When you declare a variable within a function, the variable can only be accessed within that function
- If you declare a variable outside a function, all the functions on your page can access it
- The lifetime of these variables starts when they are declared, and ends when the page is closed

JavaScript Popup Boxes

- Alert box
  - User will have to click "OK" to proceed
  - alert("sometext")
- Confirm box
  - User will have to click either "OK" or "Cancel" to proceed
  - confirm("sometext")
- Prompt box
  - User will have to click either "OK" or "Cancel" to proceed after entering an input value
  - prompt("sometext","defaultvalue")

JavaScript Language

- Conditional statement
  - if, if.. else, switch
- Loop
  - for loop, while loop
- try...catch
- throw

JavaScript Functions (which behave like Java methods)

More on Functions in other Presentation

JavaScript Funcitons

- A JavaScript function contains some code that will be executed only by an event or by a call to that function
  - To keep the browser from executing a script as soon as the page is loaded, you can write your script as a function
- You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
- Functions can be defined either <head> or <body> section
  - As a convention, they are typically defined in the <head> section

Example: JavaScript Function

```
<html>
<head>
<script type="text/javascript">
   // If alert("Hello world!!") below had not been written within a
   // function, it would have been executed as soon as the page gets loaded.
   function displaymessage() {
        alert("Hello World!")
   }
</script>
</head>

<body>
<form>
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form>
</body>
</html>
```

JavaScript Events

Events & Event Handlers

- Every element on a web page has certain events which can trigger invocation of event handlers
- Attributes are inserted into HTML tags to define events and event handlers
- Examples of events
  - A mouse click
  - A web page or an image loading
  - Mousing over a hot spot on the web page
  - Selecting an input box in an HTML form
  - Submitting an HTML form
  - A keystroke

Events

- onabort - Loading of an image is interrupted
- onblur - An element loses focus
- onchange - The content of a field changes
- onclick - Mouse clicks an object
- ondblclick - Mouse double-clicks an object
- onerror - An error occurs when loading a document or an image
- onfocus - An element gets focus
- onkeydown - A keyboard key is pressed

Events

- onkeypress - A keyboard key is pressed or held down
- onkeyup - A keyboard key is released
- onload - A page or an image is finished loading
- onmousedown - A mouse button is pressed
- onmousemove - The mouse is moved
- onmouseout - The mouse is moved off an element
- onmouseover - The mouse is moved over an element
- onmouseup - A mouse button is released

Events

- onreset - The reset button is clicked
- onresize - A window or frame is resized
- onselect - Text is selected
- onsubmit - The submit button is clicked
- onunload - The user exits the page

onload & onUnload Events

- The *onload* and *onUnload* events are triggered when the user enters or leaves the page
- The onload event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information
- Both the onload and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page.

onFocus, onBlur and onChange

- The onFocus, onBlur and onChange events are often used in combination with validation of form fields.
- Example: The *checkEmail()* function will be called whenever the user changes the content of the field:

```
<input type="text" size="30"
id="email" onchange="checkEmail()">;
```

Example & Demo: onblur

```
<html>
<head>
<script type="text/javascript">
    function upperCase() {
        var x=document.getElementById("fname").value
        document.getElementById("fname").value=x.toUpperCase()
    }
</script>
</head>

<body>

Enter your name:
<input type="text" id="fname" onblur="upperCase()">

</body>
</html>
```

onSubmit

- The *onSubmit* event is used to validate all form fields before submitting it.
- Example: The *checkForm()* function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be canceled. The function *checkForm()* returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.html"
onsubmit="return checkForm()">
```

Example & Demo: onSubmit

```
<html>
<head>
<script type="text/javascript">
    function validate() {
       //  return true or false based on validation logic
    }
</script>
</head>

<body>
       <form action="tryjs_submitpage.htm" onsubmit="return validate()">
          Name (max 10 chararcters): <input type="text" id="fname" size="20"><br />
          Age (from 1 to 100): <input type="text" id="age" size="20"><br />
          E-mail: <input type="text" id="email" size="20"><br />
          <br />
          <input type="submit" value="Submit">
       </form>
</body>
</html>
```

onMouseOver and onMouseOut

- onMouseOver and onMouseOut are often used to create "animated" buttons.
- Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver event');return false">
<img src="w3schools.gif" width="100" height="30">
</a>
```

JavaScript Objects

JavaScript Object

- JavaScript is an Object Oriented Programming (OOP) language
- A JavaScript object has properties and methods
  - Example: *String* JavaScript object has *length* property and *toUpperCase()* method

```
<script type="text/javascript">

var txt="Hello World!"
document.write(txt.length)
document.write(txt.toUpperCase())

</script>
```

JavaScript Built-in Objects

- String
- Date
- Array
- Boolean
- Math

JavaScript Object vs. Java Object

- Simlarities
  - Both has properties and methods
- Differences
  - JavaScript object can be dynamically typed while Java object is statically typed
  - In JavaScript, properties and methods are dynamically added

JavaScript Objects; 3 Different Ways of Creating JavaScript Objects

Sun microsystems

Creating Your Own JavaScript Objects

- 3 different ways
  - Create a direct instance of an object by using built-in constructor for the *Object* class
  - Create a template (Constructor) first and then create an instance of an object from it
  - Create object instance as Hash Literal

Option 1: Creating a Direct Instance of a JavaScript Object

- By invoking the built-in constructor for the Object class

```
personObj=new Object(); // Initially empty with no properties or methods
```

- Add properties to it

```
personObj.firstname="John";
personObj.age=50;
```

- Add an anonymous function to the *personObj*

```
personObj.tellYourage=function(){
    alert(“This age is ” + this.age);
}
// You can call then tellYourage function as following
personObj.tellYourage();
```

Option 1: Creating a Direct Instance of a JavaScript Object

- Add a pre-defined function

```
function tellYourage(){
    alert("The age is" + this.age);
}
personObj.tellYourage=tellYourage;
```

- Note that the following two lines of code are doing completely different things

```
// Set property with a function
personObj.tellYourage=tellYourage;
// Set property with returned value of the function
personObj.tellYourage=tellYourage();
```

Option 2: Creating a template of a JavaScript Object

- The template defines the structure of a JavaScript object in the form of a function
- You can think of the template as a constructor

```
function Person(firstname,lastname,age,eyecolor) {
        this.firstname=firstname;
        this.lastname=lastname;
        this.age=age;
        this.tellYourage=function(){
            alert("This age is " + this.age);
        }
}
```

Option 2: Creating a template of a JavaScript Object

- Once you have the template, you can create new instances of the object

```
myFather=new Person("John","Doe",50,"blue");
myMother=new Person("Sally","Rally",48,"green");
```

- You can add new properties and functions to new objects

```
myFather.newField = “some data”;
myFather.myfunction = function() {
alert(this["fullName"] + ” is ” + this.age);
}
```

Option 3: Creating JavaScript Object as a Hash Literal

- Create *personObj* JavaScript object

```
var personObj = {
        firstname: "John",
        lastname: "Doe",
        age: 50,
        tellYourage: function () {
              alert(“The age is ” + this.age );
        }
        tellSomething: function(something) {
              alert(something);
        }
 }

personObj.tellYourage();
personObj.tellSomething(“Life is good!”);
```

JavaScript Objects:
Hash (Associative Array)

JavaScript Object is an Associative Array (Hash)

- A JavaScript object is essentially an associative array (hash) with fields and methods, which are keyed by name

```
{
    firstname: "John",
    lastname: "Doe",
    age: 50,
    tellYourage: function () {
        alert("The age is " + this.age );
    },
    tellSomething: function(something) {
        alert(something);
    }
}
```

- The following two lines of code are semantically equivalent

```
myObject.myfield = "something";
myObject['myfield'] = "something";
```

JavaScript Objects: Classes, Objects, Inheritance

JavaScript has No built-in concept of Inheritance

- JavaScript has a concept of objects and classes (like in Java) but no built-in concept of inheritance (unlike in Java)
  - Every JavaScript object is really an instance of the same base class, a class that is capable of binding member fields and functions to itself at runtime

JavaScript Objects: prototype

prototype

- A prototype is a property of every JavaScript object
- Functions and properties can be associated with a constructor's property
- When a function is invoked with *new* keyword, all properties and methods of the prototype for the function are attached to the resulting object

prototype

```
// Constructor of the MyObject
function MyObject(name, size){
    this.name=name;
    this.size=size;
}
// Add a function to the prototype
MyObject.prototype.tellSize=function{
    alert("size of " + this.name+" is " + this.size);
}

// Create an instance of the object. The new object has tellSize() method.
var myObj=new MyObject("Sang", "30 inches");
myObj.tellSize();
```

JavaScript Objects: Functions Again

A function is a first-class JavaScript Object

- Functions are a bit like Java methods
  - They have arguments and return values
- A function is a first-class object in JavaScript (unlike in Java)
  - Can be considered as a descendant of Object
  - Can do everything a regular JavaScript object can do such as storing properties by name
  - Function objects can have other function objects as methods

A function can take Variable arguments

- You can call *myfunction()* or *myfunction(20)*

```
function myfunction(value){
    if (value){
        this.area=value;
    }
    return this.area;
}
```

JavaScript Objects: Context

HTML DOM Objects

HTML DOM

- The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML documents
- All HTML elements, along with their containing text and attributes, can be accessed through the DOM.
  - The contents can be modified or deleted, and new elements can be created.
- The HTML DOM is platform and language independent
  - It can be used by any programming language like Java, JavaScript, and VBScript

HTML DOM Objects

- Anchor object
- Document object
- Event object
- Form and Form Input object
- Frame, Frameset, and IFrame objects
- Image object
- Location object
- Navigator object

HTML DOM Objects

- Option and Select objects
- Screen object
- Table, TableHeader, TableRow, TableData objects
- Window object

Document Object

Document Object: Write text to the output

```
<html>
<body>

<script type="text/javascript">
document.write("Hello World!")
</script>

</body>
</html>
```

Document Object: Write text with Formatting to the output

```
<html>
<body>

<script type="text/javascript">
    document.write("<h1>Hello World!</h1>")
</script>

</body>
</html>
```

Document Object: Use getElementById()

```
<html>

<head>
<script type="text/javascript">
    function getElement() {
        var x=document.getElementById("myHeader")
        alert("I am a " + x.tagName + " element")
    }
</script>
</head>

<body>
<h1 id="myHeader" onclick="getElement()">Click to see what element I am!</h1>
</body>

</html>
```

Document Object: Use getElementsByName()

```
<html>
<head>
<script type="text/javascript">
    function getElements() {
        var x=document.getElementsByName("myInput")
        alert(x.length + " elements!")
    }
</script>
</head>

<body>
<input name="myInput" type="text" size="20"><br />
<input name="myInput" type="text" size="20"><br />
<input name="myInput" type="text" size="20"><br />
<br />
<input type="button" onclick="getElements()" value="How many elements named
    'myInput'?">
</body>
</html>
```

Document Object: Return the innerHTML of the first anchor in a document

```
<html>
<body>

<a name="first">First anchor</a><br />
<a name="second">Second anchor</a><br />
<a name="third">Third anchor</a><br />
<br />

InnerHTML of the first anchor in this document:
<script type="text/javascript">
    document.write(document.anchors[0].innerHTML)
</script>

</body>

</html>
```

Document Object: Access an item in a collection

```
<html>
<body>
<form id="Form1" name="Form1">
Your name: <input type="text">
</form>
<form id="Form2" name="Form2">
Your car: <input type="text">
</form>

<p>
To access an item in a collection you can either use the number or the name of the item:
</p>

<script type="text/javascript">
document.write("<p>The first form's name is: " + document.forms[0].name + "</p>")
document.write("<p>The first form's name is: " + document.getElementById("Form1").name
    + "</p>")
</script>

</body>
</html>
```

Event Object

Event Object: What are the coordinates of the cursor?

```
<html>
<head>
<script type="text/javascript">
    function show_coords(event) {
        x=event.clientX
        y=event.clientY
        alert("X coords: " + x + ", Y coords: " + y)
    }
</script>
</head>

<body onmousedown="show_coords(event)">
<p>Click in the document. An alert box will alert the x and y coordinates of the
    cursor.</p>
</body>

</html>
```

Event Object: What is the unicode of the key pressed?

```
<html>
<head>
<script type="text/javascript">
    function whichButton(event) {
        alert(event.keyCode)
    }

</script>
</head>

<body onkeyup="whichButton(event)">
<p><b>Note:</b> Make sure the right frame has focus when trying this example!</p>
<p>Press a key on your keyboard. An alert box will alert the unicode of the key
    pressed.</p>
</body>

</html>
```

Event Object: Which element was clicked?

```
<html>
<head>
<script type="text/javascript">
function whichElement(e) {
    var targ
    if (!e) var e = window.event
    if (e.target) targ = e.target
        else if (e.srcElement) targ = e.srcElement
    if (targ.nodeType == 3) // defeat Safari bug
        targ = targ.parentNode
    var tname
   tname=targ.tagName
   alert("You clicked on a " + tname + " element.")
}
</script>
</head>

<body onmousedown="whichElement(event)">
<p>Click somewhere in the document. An alert box will alert the tag name of the element you clicked on.</p>

<h3>This is a header</h3>
<p>This is a paragraph</p>
<img border="0" src="ball16.gif" width="29" height="28" alt="Ball">
</body>

</html>
```

Event Object: Which event type occurred?

```
<html>
<head>

<script type="text/javascript">
    function whichType(event) {
        alert(event.type)
    }
</script>
</head>

<body onmousedown="whichType(event)">

<p>
Click on the document. An alert box will alert which type of event occurred.
</p>

</body>
</html>
```

JavaScript Basics

Sang Shin
Java Technology Architect
Sun Microsystems, Inc.
sang.shin@sun.com
www.javapassion.com