Object Oriented JavaScript

Objective

- Quick Intro to JavaScript
- JavaScript Objects
- JavaScript Function
- Object Oriented JavaScript

JavaScript is not Java.

- JavaScript was developed as a web-based scripting language which would exist exclusively on the World Wide Web. It was designed for short scripts which would be easily editable by web developers. The only relation JavaScript bears to its namesake is a similar syntax.

Wow factor of JavaScript

- Real programming to your Web Pages.
- It adds interactivity to static HTML
- It can react to events
- Helpful for Validation
- Helpful to detect visitor's browser to provide better and predictive services

Programing features

- **It has all the Operators like any other language.**
  - Arithmetic Operators (+ - * / % ++ --)
  - Assignment Operators (= += -= *= /= %=)
  - Comparison operators (== != < > <= >=)
  - Logical operators (&&(logical and) ||(logical or) !(not))
  - Conditional operator(?:)
- **It also has**
  - Conditional Statements : (if , if … else, switch… case)
  - Iterative statements (while, do… while, for, for … in)

Code to understand Syntax

- **if … else Statement**
  - Syntax:

```
if (condition)
 {
 code to be executed if condition is true
 }
else
 {
 code to be executed if condition is not true
 }
```

- **switch statement**
  - Syntax:

```
switch(n)
{
case 1:
 execute code block 1
 break;
case 2:
 execute code block 2
 break;
default:
 code to be executed if n is different from case 1 and 2
}
```

```
do
  {
    code to be executed
  }while (var<=endvalue);
```

```
<script type="text/JavaScript">
      var i=0;
while (i<=5)
 {
   document.write("The number is " + i);
   document.write("<br />");
   i++;
 }
</script>
```

```
<script type="text/JavaScript">
 var i=0;
 for (i=0;i<=5;i++)
 {
  document.write("The number is " + i);
  document.write("<br />");
 }
</script>
```

```
<script type="text/JavaScript">
    var x;
    var names = new Array();
     names [0] = “Sameer";
     names [1] = “Swati";
     names [2] = “Dolly";

for (x in names )
 {
      document.write(names [x] + "<br />");
 }
</script>
```

Built-In JavaScript objects

- String
- Date
- Math
- Boolean

String Object methods

- `var txt = new String(string);`

  or

- `var txt = string;`
- Implied properties
  - Length

Method	Description
charAt()	Returns the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
split()	Splits a string into an array of substrings

Date Object

- Date object is used to work with date. and time
- Creating Date Object
  - new Date()
  - Example:

```
<script type="text/JavaScript">
    var d = new Date();
    document.write(d);
</script>
```

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
setMinutes()	Set the minutes (from 0-59)

Example on Date Comparing

```
<script type="text/JavaScript">
    var myDate = new Date();

    //set date to 14-Jan-2010
    myDate.setFullYear(2010, 0, 14);
    var today = new Date();

    if (myDate > today)
        alert("Today is before 14th January 2010");
    else
        alert("Today is after 14th January 2010");
</script>
```

Array Object

- Array object is used to store multiple values in a single variable

- **Creating/ Accessing array**

```
// regular array (add an optional integer)
var names=new Array();
names[0]="Swati";
names[1]="Varsha";
names[2]="Dolly";

for (i=0;i<names.length;i++)
{
        document.write(names[i] + "<br />");
}
```

Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
join()	Joins all elements of an array into a string
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
sort()	Sorts the elements of an array

Array Example

- Join 3 arrays into single array

```
<script type="text/JavaScript">
    var teamLeads = ["Janes", "Roschelle"];
    var developers = ["Smith", "Jacob", "Raman"];
    var testers = ["Lovleen", "Iram"];
    var project_resource = teamLeads.concat(developers ,
    testers );
    document.write(project_resource );
</script>
```

More Example

Example: Numerically sorting of data

```
<script type="text/JavaScript">
    function sortNumber(a, b) {
        return a - b;
    }

    var n = [ "10", "5", "40", "25", "100", "1" ];
    document.write(n.sort(sortNumber));
</script>
```

Math Object

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical constants and methods.
- **Using Math Object's properties/methods**

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

Method	Description
abs(num)	Returns the absolute value of num
ceil(num)	Returns num, rounded upwards to the nearest integer
max(n1,n2,n3,n4)	Returns the number with the highest value
pow(x,y)	Returns the value of x to the power of y
sqrt(num)	Returns the square root of num

JavaScript Functions

- Function treated as an object – a "first-class function"
- Functions can be instantiated (created)
- returned by other functions
- Stored as an array element or object property or assigned to a variable
- A JavaScript function stored as an object property is called a "method"
- Functions can be nested

- Normal declaration of a function:

```
function myFunc(parm1, parm2) {
  parm1 = parm1 + parm2;
  return parm1;
  }
```

- A function with no paramters still uses the parentheses

```
function myFunc(){
    return "Some Text";
};
```

- Function Defines an object "calc":

```
var calc {};
```

  - Adds a function property to the object just created; a method:

```
calc.add = function(a,b){return (a + b)};
```

  - Use of the function:

```
var c = calc.add(2,3);
```

Object-oriented?

- To qualify as object-oriented, programming languages must provide support for the following:
  - Data Abstraction
  - Encapsulation
  - Data protection
  - Inheritance

JavaScript is not an object-oriented language.

- JavaScript does not support data abstraction in the form of Classes, neither is there support for data protection.
- However, JavaScript is defined as an object-based language.

The JavaScript Object

- Objects in JavaScript are merely lists of properties, functions, and other objects
- There are three types of objects in JavaScript:
- Native: strictly defined within the language. (Number, String, Image, etc.)
- Host: Pre-defined by another source. Related to the script's environment.
- User-defined

Host Objects

- Very often, JavaScript is used to interact with the DOM (Document Object Model), a hierarchy of all objects in a web document. Manipulating the DOM is one way to create dynamic web content. Every web browser can have its own version of the DOM, which can make scripting a bit problematic.

User-defined objects

- Users can define their own objects by writing constructor function…

```
function Frog(name, color) {
this.name = name;
this.color = color;
}
```

- …and instantiating that function.

```
var myFrog = new Frog("Mothra II", "#004400")
```

User-defined objects

- You may add, remove, or alter your object's properties and methods at any time:

```
myFrog.size = "10px"
myFrog.kid = new Frog("Mothra III", "#337F00")
function sing(){
    Document.write("It's not easy being ")
    Document.write(this.color)
}
myFrog.croak = sing
delete myFrog.size
```

User-Defined Objects

- While you can instantiate an object many times, the individual objects can be changed to anything at will. Objects instantiated by the same constructor won't necessarily bear any resemblance to eachother.
- There is no way to check two objects for similar 'type'.

JavaScript: Object Based

- Objects are at the very heart of JavaScript. It is a language created to alter things that have already been defined. Objects drive the language, and it is impossible to write useful JavaScript without them.

JavaScript Functions

- All functions are executed as a method of some object
- The ‘this’ keyword is a reference to the object which is executing the method
  - 1) Global object is ‘this’ when function is defined in the page as a global function
  - 2) DOM Event Handler – ‘this’ references the DOM element that hooked up the event
    - If you are in an HTML element nested in another element having an event and you click on the inner element, the click event will “bubble” up to the outer element that subscribed to the event and fire off. ‘this’ will refer to the outer element, not the element which triggered the event.
  - 3) Constructor function – ‘this’ refers to the object being created
- Properties of objects *cannot* be private in scope.

JavaScript Functions

- Anonymous functions
  - You can define and use a function without giving it a name. Use the keyword 'function' instead of a name.
  - Example: you can assign a function to a variable and execute this variable like a function:

```
var myFunc = function(parm1, parm2){parm1 = parm1 +
  parm2; return parm1;}
```

  - Example: when you are assigning a function to a 'click' event, you can just define the function there instead of referencing one that was already declared
- Closure
  - When nesting functions, variables declared in an outer function can be seen by the inner functions

(See function2.html)

JavaScript Functions

- You can assign a function to a variable or an object property – a pointer – and use the pointer just like the function. So, you can have multiple ways to execute the same function. If the function changes, then all the pointers (variables) have the new functionality.

```
var c;
function myFunc(p1, p2){return p1 + p2};
var abc = myFunc;
c = abc("Hello", "World");
```

  (See function1.html)

- Since a function is an object, you can create properties for it.