OBJECTS IN JAVASCRIPT

Javascript Objects

When considering Javascript objects, its worth remembering that javascript is not confined to HTML, It can be used to manipulate XML, which in turn can be used to mirror the table structures found in a database. In fact many websites and Content Management Systems use databases to store information to be displayed on the site and HTML has become a popular *front end* for such databases.

Javascript objects, in common with other object orientated programming languages, have both **properties** and **methods**. For example the **built-in** window object has properties of width and height as well as methods (or functions)to open and close the window. This is just one of the many objects built into the Javascript language. As well as the Javascript built-in objects, you can create your own custom objects to suit any particular application.

> There is a slight clash of terminology here with the Object Orientated world. In the context of objects, **properties are variables** created inside an object while **methods are functions** created inside an object.

Dot notation

In Javascript the properties and methods of an object are accessed using the dot notation as shown here. Methods are distinguished by the brackets after the objectMethodName. The brackets must exist, even when there are no arguments.

.

```
objectName.propertyName;
objectName.methodName( arguments );
```

The following example retrieves the length property of a string, and the method, toUpperCase() converts the string to upper cas characters.

```
var alphabet = "abcdefghijklmnopqrstuvwxyz";
var length = alphabet.length;      // the number of characters
alphabet = alphabet.toUpperCase(); //convert string to upper case
```

To illustrate some of the advantages for Object orientated design consider an application for bookshop website. The site might provide a table listing selected books, for example,

ISBN	Title	Author
0-593-01518-5	A brief history of time	Hawking.SW
0-09-977170-5	The emporer's new mind	Penrose.R
0-71122-229-0	The lakeland fells	Wainwright.A

A book is an object, which in this example has the properties of isbn, title and author. We could create variables for ISBN (a code identifying the book), title and author, and use a multi dimensional array to hold the information.

```
var books = new Array();
  books[0] = new Array('ISBN','Title','Author');
  books[1] = new array('0-593-01518-5','A brief history of
time','Hawking.SW');
  books[2] = new Array('0-09-977170-5','The emporer's new
mind','Penrose.R');
  books[3] = new Array('0-71122-229-0','The lakeland
fells','Wainwright.A');
```

Notice that the zero element has been used to define the structure of the table, e.g. books[0][0] = 'ISBN'; books[0][1] = 'Title' and books[0][2] = 'Author'. In fact each element in the table forms what is known as a ***name:value*** pair which is the basis of an object definition. I.e. ISBN:0-593-01518-5, Title:A brief history of time.

It is now a small step to convert the array definition into an object using the now familiar dot notation.

```
var book = new Array();
    book[1].isbn = '0-593-01518-5';
    book[1].title = 'A brief history of time';
    book[1].author = 'Hawking.SW';
```

Alternatively this could have been defined directly using an objects literal notation. E.g.

```
var book = new Array();
  book[1] = {isbn:0-593-01518-5,title:A brief history of
time,author:Hawking.SW};
  book[2] = {isbn:0-09-977170-5,title:The emporer's new
mind,author:Penrose.R};
```

```
  book[3] = {isbn:0-71122-229-0,title:The lakeland
fells,author:Wainwright.A};
```

The right hand side of the expression is now a comma separated list of ***name:value*** pairs.

Finally the book object can be created using a **constructor** function

```
  function book(isbn,title,author)
  {
    this.isbn = isbn;
    this.title = title;
    this.author = author;
  }

var bookList[1] = new book(0-593-01518-5,A brief history of
time,Hawking.SW);
    bookList[2] = new book(0-09-977170-5,The emporer's new
mind,Penrose.R);
    bookList[3] = new book{0-71122-229-0,The lakeland
fells,Wainwright.A};
```

Javascript objects employ the concept of **information hiding** which is a way of managing complexity. I.e. it is not necessary to understand how the window object works, in order to be able to use its its **public** window open() method.

Prototyping - Adding a properties or methods to an object

```
function showBook(id,list)
{
  var out = document.getElementById(id);
  var msg =
"<table><tr><td>ISBN</td><td>Title</td><td>Author</td></tr>
  for (var i = 0;i<list.length;i++)
  {
    msg += "<tr><td>" + list[i].isbn + "</td><td>" + list[i].title
    msg += "</td><td>" + list[i].author + "</td></tr>";
  }
  msg += "</table>";
  out.innerHTML = msg;

}

function book(isbn,title,author,bookList)
```

```
{
  this.isbn = isbn;
  this.title = title;
  this.author = author;
  this.showBook = bookList;
}
```

Classes (or instanceof)

In the Object Orientated design (OOD) world, a class is a group of objects that have been derived from a class constructor. For example you may define

```
   //creates a null object with no properties or methods.
   var student = new Object;
```

Using this definition you can then specify new **instances of** this object. e.g.

```
   var tom, dick, helen = new student;
```

The command **instanceof** can be used to verify that tom is an instanceof person. Because Javascript has no class constructor, classes in not as prominent in Javascript as they are in other object oriented programming languages, such as C++ or Java. However this does not prevent a programmer from building class structures, with classes, sub classes and inheritance, features found in the aforementioned programming languages. Since objects can be nested within other objects in OOD speak these become **sub classes**. which then **inherit** the properties and methods of its parent object (class).

Source : http://www.soslug.org/node/1731