JavaScript Functions and Arrays

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).

Display an alert box on the basis of function:

```
<html>
<head>
<script>
function myFunction()
{
alert("Hello! I am an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="myFunction()" value="Show alert box" />
</body>
</html>
```

Display a confirm box on the basis of function:

```
<body>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
var x;
var r=confirm("Press a button!");
if (r==true)
  {
  x="You pressed OK!";
  }
else
  {
  x="You pressed Cancel!";
  }
document.getElementById("demo").innerHTML=x;
}
</script>
</body>
```

Display a prompt box on the basis of function:

```
<html>
<body>
<p>Click the button to demonstrate the prompt box.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
var x;
var name=prompt("Please enter your name","Harry Potter");
if (name!=null)
  {  x="Hello " + name + "! How are you today?";
  document.getElementById("demo").innerHTML=x;
  }
}
</script>
</body>
</html>
```

Calling a function with two arguments:

```
<script>
function myfunction(txt)
{
alert(txt);
}
</script>
<form>
<input type="button" onclick="myfunction('Good Morning!')" value="In the
    Morning">
<input type="button" onclick="myfunction('Good Evening!')" value="In the
    Evening">
</form>
<p>
When you click on one of the buttons, a function will be called with two arguments.
</p>
```

Function that returns a value:

```
<html>
<head>
<script>
function myFunction()
{
return ("Hello world!");
}
</script>
</head>
<body>

<script>
document.write(myFunction())
</script>

</body>
</html>
```

Function that returns a value:

```
<html>
<head>
<script>
function pro(a,b)
{
return a*b;
}
</script>
</head>
<body>

<script>
document.write(pro(8,5));
</script>

</body>
</html>
```

Taking Values from textbox:

```
<html>
    <head>
        <script type="text/javascript">
            function add()
            {
                var x=document.getElementById("first").value;
                var y=document.getElementById("second").value;
                var z = parseInt(x) + parseInt(y);
                alert(z);
            }
        </script>
    </head>
    <body>
        <form method="get" action="#">
            Enter First Name: <input type="text" name="first" id="first"> <br/>
            Enter First Name: <input type="text" name="second" id="second"> <br/>
            <input type="button" value="Show Result" onClick="add()">  
            <input type="reset" value="reset">
        </form>
    </body>
</html>
```

Array

An indexed list of elements

We said that a variable is a container that holds a value.

Similarly, an Array can be considered a container as well, but this one can hold multiple values

```
var student1, student2, student3, student4 ;

student1 = “Waseem” ;
student2 = “varun” ;
student3 = “ankit” ;
student4 = “Deepak” ;

document.write( student1 ) ;
document.write( student2 ) ;
document.write( student3 ) ;
document.write( student4 ) ;
```

```
student = new Array( 4 )  ;   //array declaration

student[ 0 ] = "Waseem" ;
student[ 1 ] = "varun" ;
student[ 2 ] = "ankit" ;
student[ 3 ] = "Deepak" ;
```

Can you see the advantage of using arrays along with the 'for' loop?

```
for ( x = 0 ; x < 4 ; x = x + 1 ) {
        document.write( student[ x ] ) ;
}
```

```
student = new Array( 4 )  ;   //array declaration

student[ 0 ] = "Waseem" ;
student[ 1 ] = "varun" ;
student[ 2 ] = "ankit" ;
student[ 3 ] = "Deepak" ;
```

Can you see the advantage of using arrays along with the 'for' loop?

```
for ( x = 0 ; x < 4 ; x = x + 1 ) {
        document.write( student[ x ] ) ;
}
```

Arrays in JavaScript

- In JavaScript, arrays are implemented in the form of the 'Array' object
- The key property of the 'Array' object is 'length', i.e the number of elements in an array
- Two of the key 'Array' methods are:
  - reverse( )
  - sort( )
- Elements of an array can be of any type; you can even have an array containing other arrays

JavaScript Arrays are Heterogeneous

Unlike many other popular languages, a JavaScript Array can hold elements of multiple data types, simultaneously

```
a = new Array( 9 ) ;
b = new Array( 13 ) ;

b[ 0 ] = 23.7 ;
b[ 1 ] = " Continental Hotel" ;
b[ 2 ] = a ;
```

JavaScript Arrays are Heterogeneous

Unlike many other popular languages, a JavaScript Array can hold elements of multiple data types, simultaneously

```
a = new Array( 9 ) ;
b = new Array( 13 ) ;

b[ 0 ] = 23.7 ;
b[ 1 ] = " Continental Hotel" ;
b[ 2 ] = a ;
```

JavaScript Arrays are Heterogeneous

Unlike many other popular languages, a JavaScript Array can hold elements of multiple data types, simultaneously

```
a = new Array( 9 ) ;
b = new Array( 13 ) ;

b[ 0 ] = 23.7 ;
b[ 1 ] = " Continental Hotel" ;
b[ 2 ] = a ;
```

Array Methods: reverse( )

Reverses the order of the elements

```
x = new Array ( 4 ) ;
x[ 0 ] = "Waseem" ;
x[ 1 ] = "Waqar" ;
x[ 2 ] = "Saqlain" ;
x[ 3 ] = "Shoaib" ;
x.reverse( ) ;
x.sort( ) ;
for ( k = 0 ; k < x.length; k = k + 1 ) {
        document.write( x[ k ] + "<BR>") ;
}
```

Saqlain
Shoaib
Waqar
Waseem

Is this the required result?

Array Methods

- Push
- Pop
- Unshift
- Shift
- Reverse
- Sort
- Join
- splice

join

- var fruits = ["Banana", "Orange","Apple", "Mango"]; document.getElementById("demo").innerHTML = fruits.join(" * ");

- Result

Banana * Orange * Apple * Mango

pop

- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.pop();

// Removes the last element ("Mango") from fruits

- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  var x = fruits.pop();

// the value of x is "Mango"

push

- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.push("Kiwi");

// Adds a new element ("Kiwi") to fruits

- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  var x = fruits.push("Kiwi");

// the value of x is 5

shift

- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.shift();

// Removes the first element "Banana" from fruits

unshift

- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.unshift("Lemon");

// Adds a new element "Lemon" to fruits

var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon"); // Returns 5

splice

- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.splice(2, 0, "Lemon", "Kiwi");

- The first parameter (2) defines the position **where** new elements should be **added** (spliced in).
- The second parameter (0) defines **how many** elements should be **removed**.
- The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be **added.**