ARRAY DATA STRUCTURE

Javascript Array Object

An **array** is a **data structure** common to all modern programming languages. Arrays are **temporary** data structures, that is they only exist so long as the program is running. Once the program is terminated any data contained in an array is lost. To use the built in Array object you must first create an **instance** of it using the **new** keyword. An array can be manipulated as a single entity, or any single member of the array (called an **array element**) can be accessed independently. Array elements are usually of the same data type, (but they need not be; because Javascript is a loosely typed language)..

Each element is located by an **index** (or address) pointing to the appropriate **cell** number. For example, consider a **one dimensional array** to contain 8 integer numbers:-

```
var myList = new Array(8);
```

The above declaration has informed the compiler to make available 8 cells (locations in memory) that can each hold an integer. To assign the number 8 to the first element of the array is done by the following assignment.
Notice that the first cell index starts at zero (0).

```
myList[0] = 8;
```

Once the array has been declared cell values can be assigned in any order. For example the next cell to be assigned a value might be cell 3. After assigning these values you can imagine the memory positions looking like this.
Until elements are assigned a value they have the value **undefined**.

index	0	1	2	3	4	5	6	7
value	8			5			5	5

Data is generally entered into an array by one of two methods, namely by assignment as shown above, or by using a loop. The first method is fine if you only have a few numbers to enter, but the second is much more widely

used. The following code fragment shows how to initialise all the array cell values to 0.

An array is simply a **container**, or **list** of similar things.

```
  myVar = new Array();  // a new Array called myVar.
```

Array can be defined in one of two ways. For example if the number of items is short, and maybe fixed as well, then the array can be defined as follows.

```
  var myColor = new Array("Red", "Green", "Blue", "Cyan",
"Magenta",
                          "Yellow", "Black", "White" ) ; or more
simply
  var myColor = ["Red", "Green", "Blue", "Cyan", "Magenta",
                 "Yellow", "Black", "White"];
```

Array elements can be specified one at a time. For example,

```
  var myColor = new Array();
  myColor[0] = "Red";
  myColor[1] = "Green";
  myColor[2] = "Blue";
  myColor[3] = "Cyan";
  myColor[4] = "Magenta";
  myColor[5] = "Yellow";
  myColor[6] = "Black";
  myColor[7] = "White";
```

Notice that you don't have to specify the size, (as you might in Visual Basic for example) though you can if you want to. The array can grow incrementally as new elements are added. Each item in the array is defined by its position. by default the first position is 0, then 1 and so on. The value in the square brackets is the **index**, or position of the item in the array.
It is a common mistake to confuse the address (e.g. 4) with the data (Magenta). A single element of an array can be assigned to a variable

```
  var favouriteColor = myColor[4];
```

Alternatively I can pass the whole array to another variable. For example

```
  var colorRange = new Array();
colorRange = myColor;
```

To determine the length (or number of elements) in an array, use the length **property** for example,

```
  var size = myColor.length;
```

Source : http://www.soslug.org/node/1754