Arrays are a way of storing a lot variables that are very similar or related. Arrays allow the developer to group these variables together and treat them as a group instead of having to work with them as individual variables.

***For example:*** *The following script list four books as individual variables. This works ok if you only have a few books. What happens when you have 100 books? Do you really want to create a variable for each book?*

<script>

var book1 = “Javascript”;

var book2 = “CSS and Styling”;

var book3 = “HTML5 New Way”;

var book4 = “Advanced CSS Styles”;

…

</script>

Instead of storing numerous variables that are very similar we can group them together into an array. This allows us to store all of the information in one variable called books. The books variable has one row for each book stored in the array. We can use as many rows as we need.

<script>

var books = new Array(); //creates an array variable called books

books[0] = “Javascript”; //these lines load the values into rows of the array

books[1] = “CSS and Styling”;

books[2] = “HTML5 New Way”;

books[3] = “Advanced CSS Styles”;

</script>

**Create an Array variable**

In Javascript we have an Array object that defines, describes and provides tools to work with arrays. When we create an array variable we are really making an Array object. So in the example above we are making a **books** variable. Javascript and the developer can treat the variable as an Array object.

You can find more information about the Array object in the Javascript reference. I would recommend the DHTML Definitive Reference and W3Schools as good starting points.

**Row of an array**

Each value stored in an array variable has its own row. This is very similar in layout and structure to a table or a spreadsheet. Each row in this array has a number or index associated with it. In Javascript like most of the object oriented languages the first row starts with **0**.

**Putting values into an array**

In the example shown above the first book was loaded into row 0 or the first row of the books array. The next book was loaded into row 1 of the array. Because we start counting rows with 0 things seem a little off but it will work out.

The Javascript command to create a new uses the var command to create a new variable. You may call an array variable anything you wish. The keyword ‘new’ tells Javascript that you are creating a new Array object and using the Array( ) method to create the object. The new array object will called whatever name you gave the variable.

Once an array variable is created we can load the rows. In general you load the rows of an array starting with row 0 and work your way up. Some arrays can be loaded at the same time they are created. This works well when you have an array that rarely changes and does not have too many values to be stored.

***For example:*** *The two examples shown below create an array variable called products. The first one creates and loads the array in the same command. The second command creates the array and then manually loads a value into each row of the array.*

<script>

var products = new Array(“Pen”, “Paper”,”Pencil”); //creates and loads the array

var products = new Array( ); //create the array

products[0] = “Pen”; //load the first value

products[1] = “Paper”; //load the second value

products[2] = “Pencil”; //load the third value

</script>

**Getting values out of an array**

You can use a row of an array just like a variable. You use the index or row number to indicate which value you wish to access.

<script>

alert(products); //displays all the values stored in the array

alert( products[1] ); //displays Paper because that is the value stored in row 1

</script>

<script>

document.write( products[0] ); //writes the value of “Pen” out to the document object

document.write( “<p>” + products[2] + “</p>” ); //writes out a paragraph with Pencil

</script>

**Using a loop to output an array**

A common use of an array is to store data and then use the array to create content on a page. One of the most useful properties of the Array object is the .length property. This property allows us to know how many rows are in an array. This makes processing the contents of an array with a loop very convenient. Since we know how many rows we can use the for loop to process the rows or values in an array.

***.length A property of the Javascript Array object. It returns the number of rows in an array***

*Given our products array in the example above review these examples.*

<script> **//displays each value in the array as a paragraph on the page.**

for( x=0; x<products.lenght; x++)

{

document.write( “<p>” + products[x] + “</p>”);

}

</script>

<ul>

<script> **//displays each value in the array as a list item within a list.**

for( x=0; x<products.lenght; x++)

{

document.write( “<li>” + products[x] + “</li>”);

}

</script>

</ul>

<select name=”productsList”>

<script> **//displays each value in the array as an option within a select statement.**

for( x=0; x<products.lenght; x++)

{

document.write( “<option>” + products[x] + “</option>”);

}

</script>

</select>

**Parallel arrays**

In many cases we need to group several pieces of information together for each product or service, etc. In this case we can use multiple arrays that work together. One array will hold the name of the product. Another array will hold the prices of the products. The name and the price are stored in different arrays but they are stored in the same row of those arrays.

We have a table of the following information. We need to put the product and its corresponding price in an array.

|  |  |
| --- | --- |
| **Product** | **Price** |
| Pen | 1.29 |
| Paper | .19 |
| Pencil | .67 |

In Javascript we are going to create two arrays. One for the product and one for the price. The price in row 0 will correspond to the product in row 0, etc. The price array is parallel to the product array. In other words the rows of each array work together like in a spreadsheet.

<script>

var products = new Array( ); //create the product array

products[0] = “Pen”; //load Pen

products[1] = “Paper”; //load Paper

products[2] = “Pencil”; //load Price

var price = new Array( ); //create the price array

price[0] = 1.29 //load the price of a Pen

price[1] = .19; //load the price of a Paper

price[2] = .67; //load the price of a Pencil

</script>

Search an array to find a value

When processing arrays we often need to locate a certain value within the array. Or we need to know if the value is already in the array before we add it to the array. This is called searching the array. A loop is used to read through each row in the array. A if statement is used to compare the value of the current row to the desired value. If they match that value is in the array. If not the value is not located within the array.

For example: Using a loop to process each row of the array looking for a value. In this case is ‘Paper’ already in the product array?

<script>

var products = new Array( ); //create the product array

products[0] = “Pen”; //load Pen

products[1] = “Paper”; //load Paper

products[2] = “Pencil”; //load Price

var requestedProduct = prompt(“What product are you looking for?”,””); //Paper

for(x=0; x<product.length; x++)

{

if( product[x] == requestProduct)

{

alert(requestedProduct + “ is already in the Product array”);

alert(“It is located at row “ + x);

}

}

</script>

For example: Find a product and display its price. This questions uses a loop to search through the product array. If it finds the requested product it will use the parallel array price to display the price that corresponds to the product. If nothing is located it will display a ‘Not Found’ message.

<script>

function findPrice()

{

var inProduct = prompt("What product do you want?",""); //product to search for

var foundRow="not"; //local variable

for(x=0; x<products.length; x++)

{

if(inProduct == products[x])

{

foundRow = x; //located the product at row x

break; //stop the loop and continue with next command

}

}//end loop search for product name

if(foundRow == "not") //check to see if we found the product

{

alert("Sorry, " + inProduct + " is not listed in our products");

}

else

{

alert("The price for " + products[foundRow] + " is " + price[foundRow]);

}

}//end findPrice function

findPrice(); //runs the function

</script>

**Modify values in an array**

Once values are in an array they can be accessed, displayed and updated. In order to change the value of the row you located the desired value either by the row number or index or by search through the array looking for the desired value.

Once you have located the row that contains the value you wish to update you can assign a new value, modify the existing value or replace the current value with “” or 0.

<script>

product[1] = “Printer Paper”; //changes “Paper” to “PrinterPaper”

price[1] += .10; //changes .19 to .29 price[1] = price[1]+.10

product[4] = “Stapler”; //Adds new row to product array

price[4]=4.87; //adds stapler price to the parallel price array

</script>