Looping and Arrays

Today's Outline

- Questions
- Overview of types of loops
- For-Loops
- Arrays

Types of Loops

Loops are used to execute code repeatedly.

There are four main types of loops in javascript:

1. For Loops
2. While Loops
3. For-In Loops
4. Do-While Loops

Today we will only be concentrating on For Loops

For-Loops

We use for loops when we know how many times we want to execute the same code.

```
for ( var i = 0; i < 10; i++ ) {

    console.log(i);

}
```

For-Loops - Parts

There are five main parts to a javascript for-loop.

```
for ( var i = 0; i < 10; i++ ) {
      console.log(i);
}
```

1. keyword for - for
2. initialization - var i = 0;
3. condition - i < 10;
4. increment / decrement - i++;
5. code block - console.log(i);

For-Loops

Steps

1. Initialization assigns a value to the “looping” variable (i). This happens before the looping starts.
2. The condition is tested for true or false. If true, then the code will execute. If false, then the code in between the looping brackets will not run.
3. After the code in the loop executes, the increment / decrement statement is run. The variable is then re-assigned to the new value. (i.e. i++ or i--).
4. The condition is then again tested and code will be re-run with the new variable value.
5. This will continue to happen until the condition is false.

For-Loops - Demo

Incrementing

Decrementing

Skipping Numbers

Running full blocks of code

For-Loops

For-Loops can begin and end with any number you want. You don't need to be constrained to starting with 0.

Loops can run in ascending or descending order.

You do not need to loop sequentially. The following is perfectly valid.

```
for ( var i = -10; i < 100; i += 70 ) {
        console.log(i);
}
```

What's the result of this code?

Arrays

Arrays are used to stored multiple values in a single variable.

They can be also be thought of as collections or lists.

When you need to store a lot of variables or an unknown amount of variables, then you need to use an array.

Arrays - Creation

Create an array by using opening and closing square brackets.

```
var myArray = [ ];
```

You can also create an array with variables already stored inside. Commas separate items in an array.

```
var cars = [ “Mercedes”, “Pontiac Grand AM” ];
```

The cars array now has two strings inside.

Arrays - Access

To find an item in an array, you use square brackets with an integer to get the item at the index.

```
var cars = [ “Mercedes”, “Pontiac Grand Am” ];

console.log( cars[0] );        // Mercedes

console.log( cars[1] );        // Pontiac Grand Am
```

Items in an array start at index 0.

Arrays - Access

```
var cars = [  “Mercedes”, “Pontiac Grand Am”   ];
```

0 1

Arrays - Types

Arrays don’t have to be limited to one type. Strings, numbers, booleans and more complex objects can all be stored in the same array. You can even put arrays inside of arrays.

```
var myArray = [ 1, “Hello”, true  ];

console.log(myArray[0]);    // 1

console.log(myArray[1]);    // Hello

console.log(myArray[2]);    // true
```

Arrays - Properties

An Array, like everything else in Javascript, is an object. Objects can have properties and functions. Objects will be covered in more detail in a future lesson.

To find out how many items are in an array, we can ask the “length” property.

```
var cars = [ “Mercedes”, “Pontiac Grand AM” ];

console.log( cars.length );
```

What is the value printed out?

Arrays - Demos

Creation

Accessing

How many items in an array?

Arrays - Adding Items

You can add items to an array by using the length property and square brackets.

```
var cars = [ “Mercedes”, “Pontiac Grand AM” ];

cars[cars.length] = “BMW”;

console.log( cars );  //  [ “Mercedes”, “Pontiac Grand AM”, “BMW” ];
```

This is the same as writing…

```
cars[2] = “BMW”;
```

Arrays - Adding Items

When using an index to add items, you will create an array of empty items if you use an index too far in advance.

```
var cars = [“Mercedes”, “Pontiac Grand AM” ];

cars[5] = “BMW”;

console.log(cars);

// [ “Mercedes”, “Pontiac GrandAM”, undefined, undefined, “BMW” ]
```

Arrays - Adding Items

You can also use the push() method to add an element to the array.

push() adds items to the end of the array.

```
var cars = [ “Mercedes”, “Pontiac” ];

cars.push(“BMW”);

console.log(cars[2]);    //     BMW
```

Arrays - Removing Items

The simplest way to remove an element from an an array is by calling the pop() method.

The pop() method removes the last element of the array.

```
var cars = [ “Mercedes”, “Pontiac” ];

cars.pop();

console.log( cars ); // Mercedes
```

Arrays - Removing Items

The shift() method removes an item from the beginning of the array.

```
var cars = [ “Mercedes”, “Pontiac” ];

cars.shift();

console.log( cars ); // Pontiac
```

Arrays - Demo

Adding Items

Removing Items

Arrays and For-Loops

When we want to look at all the items in the array, we can use a for-loop. Since we know the length of an array using the “length” property, we know how long to loop.

```
var cars = [ “Mercedes”, “Pontiac”, “BMW” ];

for (var i = 0; i < cars.length; i++) {
        console.log(cars[i]);
}
```

What is printed out?

Arrays and For-Loops

```
var cars = [ “Mercedes”, “Pontiac”, “BMW” ];
for (var i = 0; i < cars.length; i++) {
        console.log(cars[i]);
}
```

This is the equivalent of doing…

```
var cars = [ “Mercedes”, “Pontiac”, “BMW” ];
for (var i = 0; i < 3; i++) {
        console.log(cars[i]);
}
```

Summary

- For-Loops are used to execute code multiple times
- Arrays are used to store multiple values in one variable
- You can use for-loops to look at the values inside of an array

Workshop

1. Large Group Coding - Find Item in an Array
2. Smaller Group Coding - Add all numbers in an array
3. Smaller Group Coding - Reverse words in a sentence

Citations

http://www.w3schools.com/js/js_arrays.asp