Loops & Arrays – Recitation 9

The Array Object:
-An array is used to store multiple values in a single variable.
-For example, if I have a list of courses that I'm taking, I don't want to have to store them all one by one into individual variables.

This is unnecessary:
var course1 = "CS170";
var course2 = "MATH135";
var course3 = "FRENCH101";
var course4 = "CS111";
var course5 = "BUS101";

Instead, let's use an array:
var fallCourses = new Array(); // create a new array object
fallCourses[0] = "CS170";
fallCourses[1] = "MATH135";
fallCourses[2] = "FRENCH101";
fallCourses[3] = "CS111";
fallCourses[4] = "BUS101";

Or we can condense that and do this:
var fallCourses = new Array("CS170","MATH135","FRENCH101","CS111","BUS101");

Things to know about arrays:
-Declare an array by saying: var arr = new Array(); By doing this, you're saying that you want to create an array object that can be used to store things.
-The first slot in an array is the 0^{th} spot (not the 1^{st}!). Then they continue to increment.
-Accessing the nth spot: arr[n]; You can save this in another variable or alert this value. Treat it like you would treat a variable.
-Setting nth spot to new value: arr[n] = 3;
-Arrays can store anything that a variable can store (numbers, strings, other variables, etc)

**Think of arrays like lockers. Each locker has an id number on it, and to access your stuff in a locker, you have to know what id number it is. So let's say, you have your belongings in locker #21, and you have shoes in there that you want to take out. After you take the shoes out, you want to put your backpack in the locker. To do this with code, we can do:

var takeOut = locker[21]; // takeOut will contain whatever was in locker[21] (shoes)
locker[21] = "backpack"; // replace the value in locker[21] with backpack

The 'for' loop:
-Loops are used to run the same code multiple times – each time with a different value
-For example, I want to alert the numbers 1-5. Since we're working with small numbers, you could do:

```
alert(1);
alert(2);
alert(3);
alert(4);
alert(5);
```

But what if I told you to alert the numbers 1-100 or 1-1000? You don't want to have to write that many alerts. Instead we use a loop:

```
for(var i=1; i<=100; i++){
  alert(i);
}
```

First part: I'm saying that we should start at the variable i, which we set equal to 1.
Second part: This is our condition, so we keep executing this loop until i<=100.
Third part: Every time we go through the loop once, we increase i's value by 1.
Within the loop: That's the code we execute every iteration of the loop.

Structure:

```
for(starter; loop condition; increment){
  // What should be done at every iteration of the loop?
}
```

starter: This executes at loop start
loop condition: The condition to check if the loop is finished. It is checked after every execution of the loop body.
increment: An action to perform after every iteration, but before the loop condition is checked

**Think of loops as a person doing the same task in an assembly line. For example, if the person has a job of putting a brand new laptop in a box to get it ready for shipment. If we know that the person has 100 laptops to put into 100 boxes, we can say:

```
for(var i=0; i<100; i++){
  openBox();
  putLaptopInBox();
  closeBox();
}
```

Here I'm calling functions, assuming that they're written already. Also, notice how I start i at 0 and terminate when i<100. The variable i will only hit 99 and then stop.

We can combine loops and arrays now!

Example: print out all the values in this array (iterate through the array)

```
<script>
        var arr = new Array("Hello", "world", "I", "am", "taking", "CS170", "!");
        for (var i=0; i<arr.length; i++){
                document.write(arr[i]);
        }
</script>
```

**Note: arr.length gets me the array length, as expected. The length of arr here is 7, but remember that we're starting at index 0, so the highest index of this array is 6, which is why we do i<arr.length, and not i<=arr.length.