Lesson Introduction

Now that we've talked a little bit about arrays and objects and how they can be used differently to store different types of data, I'd like to discuss some techniques that you can use with arrays.

First of all, arrays support a number of very useful methods such as `push`, `pop`, `unshift`, and `shift`. These allow you to add and remove data from the beginning and the end of an array. The ability to manipulate your data this way provides a number of opportunities.

Push, Pop, Unshift, and Shift

Let's use the `push` method.

```
var steps = ["brainstorm", "narrow", "prototype", "test", "propose"];
steps.push("profit");
console.log(steps);
// ["brainstorm", "narrow", "prototype", "test", "propose", "profit"]
```

As you see the "profit" is now at the end of our initial array.

```
var popped = steps.pop();
console.log(steps);
// ["brainstorm", "narrow", "prototype", "test", "propose"]
console.log(popped); //"profit"
```

We see the "profit" has been popped off the end, so the `pop` method removes the last item and also returns the value of that last item.

There's a parallel to the `push` command called `unshift`.

```
steps.unshift("drink");
console.log(steps);
// ["drink", "brainstorm", "narrow", "prototype", "test", "propose"]
```

`unshift` does the same thing that `push` does except that it puts the new term in the array at the beginning.

And now `shift`:

```
var shifted = steps.shift();
console.log(steps);
// ["brainstorm", "narrow", "prototype", "test", "propose"]
console.log(shifted); //"drink"
```

It simply removes the first item from the array and returns its value.

Arrays As Stacks

Stacks are one of two techniques for using sequential arrays. With a stack, you add items to a sequential list and take them off in the opposite order in which they were added, so the last one added is the first one that you take off.

This is a frequent pattern in computer programs and you can achieve this using arrays with the `push` and `pop` methods.

```
var stack = [];
stack.push("urgent");
stack.push("super urgent");
stack.push("ultra urgent");
console.log(stack); //["urgent", "super urgent", "ultra urgent"]
```

Imagine we always want to deal with the last thing that they told us first:

```
var current = stack.pop();
console.log(current); //"ultra urgent"
console.log(stack); //["urgent", "super urgent"]
```

Arrays As Queues

Similarly, we can use arrays to simulate **queues** that are designed to make sure nothing gets missed from the very beginning to the very end of the list. By taking the item that was put on the list the longest to go and dealing with it first. As you might have figured out we can use `push` and `shift` to make a queue.

```
var queue = [];
queue.push("brainstorm");
queue.push("narrow");
queue.push("prototype");
console.log(queue); //["brainstorm", "narrow", "prototype"]

var current = queue.shift();
console.log(current); //"brainstorm"
console.log(queue); //["narrow", "prototype"]
```