Anonymous Functions & Callbacks

Unlike Python, *functions* are expressions

That means you can save them to variables!

Here’s how you learned to define and call functions:

```
function adamsFunc(name) {
    return “what’s up ” + name;
}
var greeting = adamsFunc( “Adam” );
```

You can do the same with an anonymous function!

```
var adamsFunction = function(name) {
    return “what’s up ” + name;
}
var greeting = adamsFunction( “Adam” );
```

Saving functions to variables allows us to pass functions into other functions!

First let’s define an “add” function and a “subtract” function

```
 var add = function(num1, num2) {
      return num1 + num2;
}

var subtract = function(num1, num2) {
      return num1 - num2;
}
```

Now let’s define a “calc” function

```
var calc = function(num1, num2, operation) {
    return operation(num1, num2);
}
```

Hmmmmm… operation is an input parameter,
but we’re calling it like it’s a function!

We can pass “add” and “subtract” into “calc” as we please!

(we call add and subtract “callbacks”)

```
var addResult = calc(2, 4, add); // Equals 6

var subResult = calc(2, 4, subtract); // Equals -2
```

Arrays have built in functions that make use of callbacks

It would behoove you to learn about them!

Array.forEach()

- Iterates over your array and calls a function for each element
- The element is passed to your callback function

Anonymous callback function!

```
var arr = [0, 1, 2, 3, 4, 5];

arr.forEach( function(element) {
        console.log( element + “! “);
} );

// 0!
// 1!
// 2!
// 3!
// 4!
// 5!
```

You can make your callback function do whatever you want!

Array.map()

- Iterates over your array and creates a new array
- The new array's elements are determined by your call back function

Anonymous callback function!

```
var arr = [0, 1, 2, 3, 4, 5];

var newArr = arr.map( function(element) {
        return element + 2;
} );

console.log(newArr); // [ 2, 3, 4, 5, 6, 7 ]
```

Array.filter()

- Iterates over your array and creates a new array containing only the elements you want
- Your callback should return ‘true’ for elements you want, ‘false’ otherwise

```
var arr = [0, 1, 2, 3, 4, 5];

var newArr = arr.filter( function(element) {
      return element < 2;
} );
```

Returns true for numbers less than 2, false otherwise

```
console.log(newArr); // [ 0, 1 ]
```

Array.each()

- Iterates over your array and returns true if every element passes your condition
- Your callback should return ‘true’ for elements that pass your condition, false otherwise

```
var arr = [0, 1, 2, 3, 4, 5];

var allLT2 = arr.each( function(element) {
        return element < 2;
} );
```

Returns true for numbers less than 2, false otherwise

```
console.log(allLT2); // false
```

Array.some()

- Iterates over your array and returns true if at least one element passes your condition
- Your callback should return ‘true’ for elements that pass your condition, false otherwise

```
var arr = [0, 1, 2, 3, 4, 5];

var someLT2 = arr.some( function(element) {
        return element < 2;
} );
```

Returns true for numbers less than 2, false otherwise

```
console.log(someLT2); // true
```