Putting the Fun in Functions

Or in fundamental

or in da Funk

Review

```
function fn(param){
}

var fn = function( param ){
};

var a = “a”;
fn(a);
```

Functional Expressions

```
var foo = function(param){
  doSomeStuffWithParam(param);
};
```

Function declarations

```
function foo(param){
  doSomeStuffWithParam(param);
}
```

Differences?

```
var name = “Jeff”;

var greeting = greet( name );

function greet(param){
  return “Hey, ” + param;
}

console.log( greeting ); //=> “Hey, Jeff”
```

```
var name = “Jeff”;

var greeting = greet( name ); //=>
TypeError: greet is not a function

var greet = function(param){
  return “Hey, ” + param;
}

console.log( greeting );
```

Wait, what?

Functional Declarations load before the rest of your code

```
var name = “Jeff”;

var greeting = greet( name );

function greet(param){
   return “Hey, ” + param;
} // RUNS FIRST

console.log( greeting ); //=> “Hey, Jeff”
```

Three Different Ways to Refer to Functions

Functions, Methods, Constructor Calls

```
var foo = function(){};
// foo is a function
```

```
var obj = {
   foo: function(){
   }
};
// obj.foo is a method
```

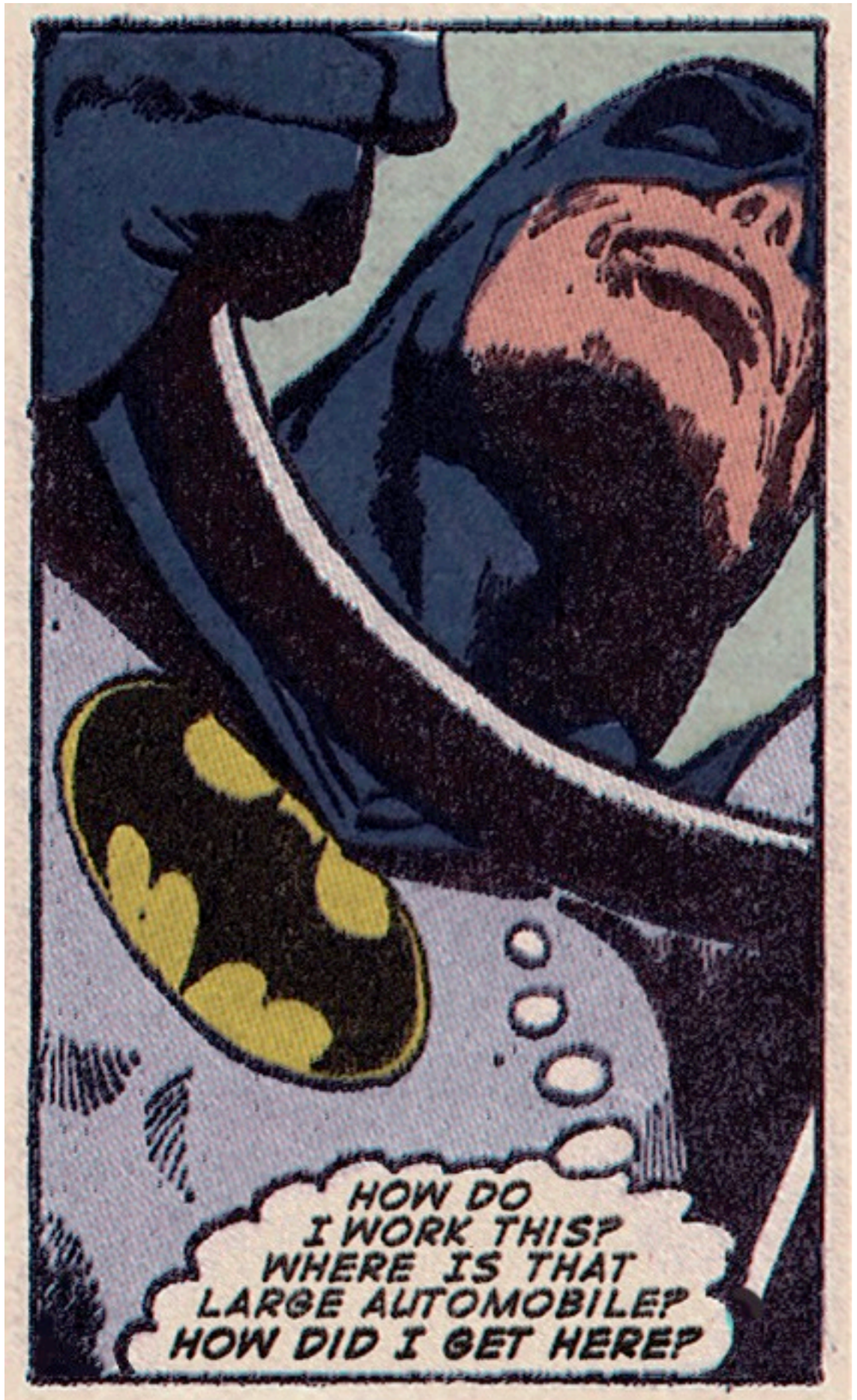
```
var Obj = function(){};
Obj.prototype.foo = function(){};
var o = new Obj();
//new Obj is a constructor call
```

Calling Functions

```
foo(a, b);
foo.call( undefined, a, b );
foo.apply( undefined, [a, b]);
```

```
obj.foo( a, b );
obj.foo.call( obj, a, b );
obj.foo.apply( obj, [a, b] );
```

```
obj.foo( a, b );
obj.foo.call( obj, a, b );
obj.foo.apply( obj, [a, b] );
```


HOW DO
I WORK THIS?
WHERE IS THAT
LARGE AUTOMOBILE?
HOW DID I GET HERE?

That first argument changes the context of the function called

That doesn't make a ton of sense right now

We'll come back to that

Higher Order Functions

```
var foo = function( bar ){
  console.log( bar );
};

var doMath = function( num1, num2, fn ){
  var sum = num1 + num2;
  fn( sum );
};

doMath( 3, 4, foo );
// 7;
```

Array.prototype.map

Remember, this is a function on an instance of an Array

```
var double = function(num){
  return num * 2;
};

[4, 2, 5, 1, 9, 5].map( double ); //=> [8,4,10,2,18,10];
```

Call vs. Apply

```
fn(); =>
fn.call( undefined ); =>
fn.apply( undefined, [] );

fn( 1, 2, 3 ) =>
fn.call( undefined, 1, 2, 3 ); =>
fn.apply( undefined, [1,2,3] );
```

Why bother?

```
var hello = function(){
  console.log( “Hello, ” + this.name );
};

var Person = function( name ){
  this.name = name;
};

var Dog = function( name ){
  this.name = name;
};

var p = new Person( “Jeff” );
var d = new Dog( “Lyle” );

hello.call(p); //=> “Hello, Jeff”
hello.call(d); //=> “Hello, Lyle”
```

ANOTHER EXAMPLE

```
//Concat these two arrays

var arr = [1,2,3,4,5];
var arr2 = [6,7,8,9,0];
```

// Most common response (when this was an interview question for me)

```
for( var i = 0, l = arr2.length; i < l; i++ ){
  arr.push( arr2[i] );
}
```

Syntax

```
arr.push(element1, ..., elementN)
```

Parameters

`element1, ..., elementN`

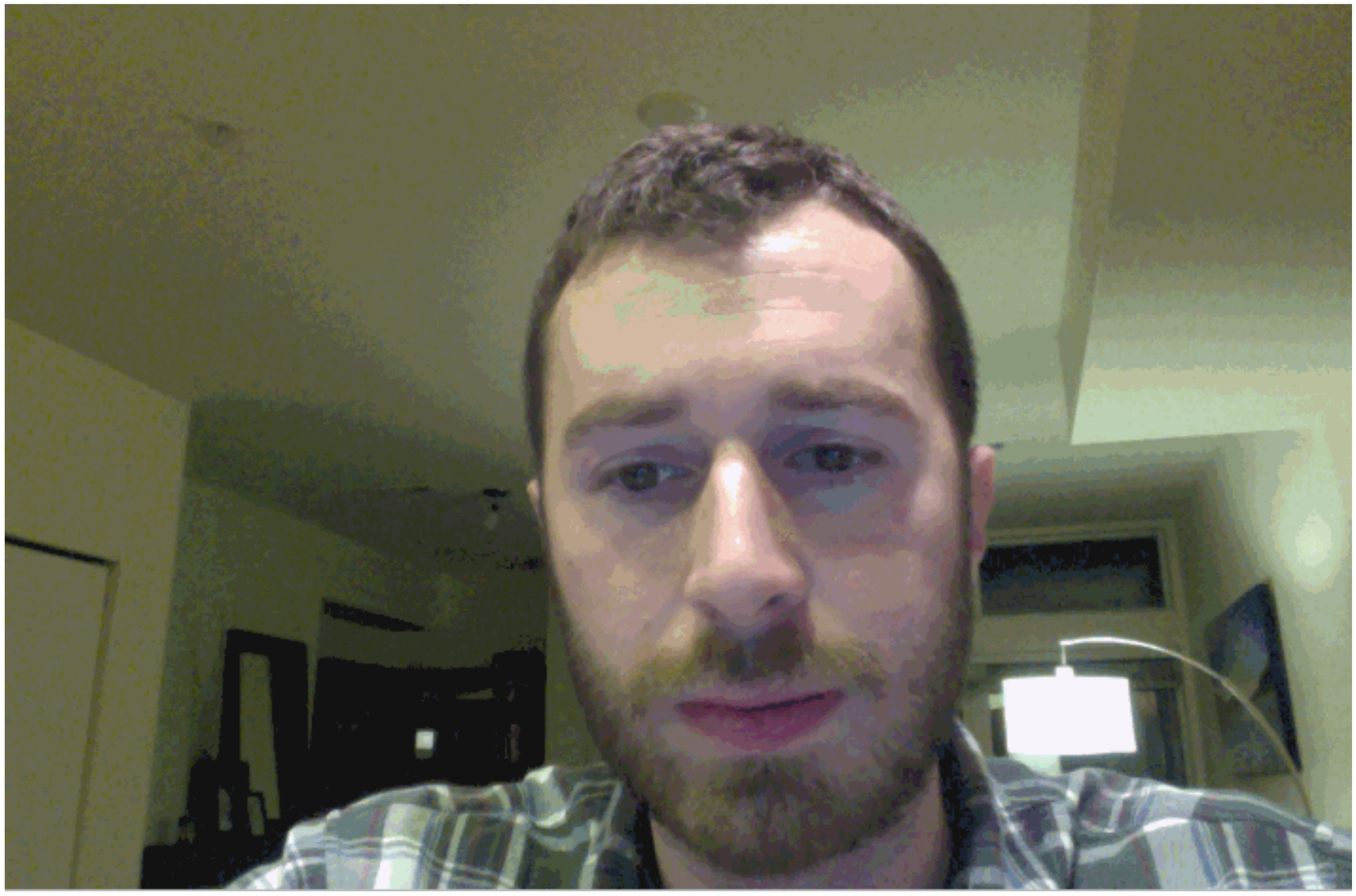
The elements to add to the end of the array.

```
// Instead use apply!
arr.push.apply(arr, arr2);
arr; //=> [1,2,3,4,5,6,7,8,9,0]
```

Arguments

A pseudo-array of the arguments that are passed into a function

A pseudo-array of the arguments that are passe
function



Time for a solution.

Array.prototype.slice

Array-like objects

`slice` method can also be called to convert Array-like objects / collections to a new Array. You just bind the method to the object. The `arguments` inside a function is an example of an 'array-like object'.

```
function list() {
  return Array.prototype.slice.call(arguments, 0);
}

var list1 = list(1, 2, 3); // [1, 2, 3]
```

```
var raise = function( str ){
  return str.toUpperCase();
};

var yellsArguments = function(){
  var args = Array.prototype.slice.call( arguments, 0 );
  console.log( args.map( raise ).join( " " ) );
};

yellsArguments( “hello”, “how”, “are”, “you”, “today?” );
//=> "HELLO HOW ARE YOU TODAY?"
```

Let's do this piece by piece

```
var raise = function( str ){
  return str.toUpperCase();
};
```

```
var howdy = “hey”;

raise( howdy ); //=> “HEY”;
```

```
var yellsArguments = function(){
  var args = Array.prototype.slice.call( arguments, 0 );
  console.log( args.map( raise ).join( " " ) );
};
```

```
var yellsArguments = function(){
  // do stuff
};
```

```
var args = Array.prototype.slice.call( arguments, 0 );

// Changes arguments pseudo-array into a real one!
```

```
console.log( args.map( raise ).join( " " ) );
```

```
var args = [ “hello”, “how”, “are”, “you”, “today?” ];

args.map( raise );

//=> [“HELLO”, “HOW”, “ARE”, “YOU”, “TODAY?”]
```

```
var raise = function( str ){
  return str.toUpperCase();
};

var yellsArguments = function(){
  var args = Array.prototype.slice.call( arguments, 0 );
  console.log( args.map( raise ).join( " " ) );
};

yellsArguments( “hello”, “how”, “are”, “you”, “today?” );
//=> "HELLO HOW ARE YOU TODAY?"
```

Activity

- Learn about Array.prototype.sort
- Figure out how to use [].map to square all numbers in an array