Javascript

Asynchronous functions and libraries

by Kevin Faulhaber

Asynchronous

= non-blocking

Synchronous vs Asynchronous

```
var stock = 0,
    prev = 0,
    curr = 1;

for (var i = 0; i < 10000; i++) {

    stock = curr;
    curr = curr + prev;
    prev = stock;

    console.log(curr);
}

console.log('We are done!');
```

```
var stock = 0,
    prev = 0,
    curr = 1;
setTimeout(function () {
    for (var i = 0; i < 10000; i++) {

        stock = curr;
        curr = curr + prev;
        prev = stock;

        console.log(curr);
    }
}, 0);

console.log('We are done!');
```

List of asynchronous native codes

- setInterval (*jQuery*)
- setTimeout
- requestAnimationFrame (*Browser games*)
- XMLHttpRequest (*AJAX*)
- WebSocket
- Worker

Callbacks

How do we know when an asynchronous task gets completed?

Callback function in action

While the fib function is happening we can do other stuff in the mean time.


```
var fib = function (cb) {

    var stock = 0,
        prev = 0,
        curr = 1;

    setTimeout(function () {
        for (var i = 0; i < 10000; i++) {
            stock = curr;
            curr = curr + prev;
            prev = stock;

            console.log(curr);
        }
        cb();
    }, 0);
}

var callback = function () {
    console.log('We are done');
}

fib(callback);
```

Library

What is a Javascript library?

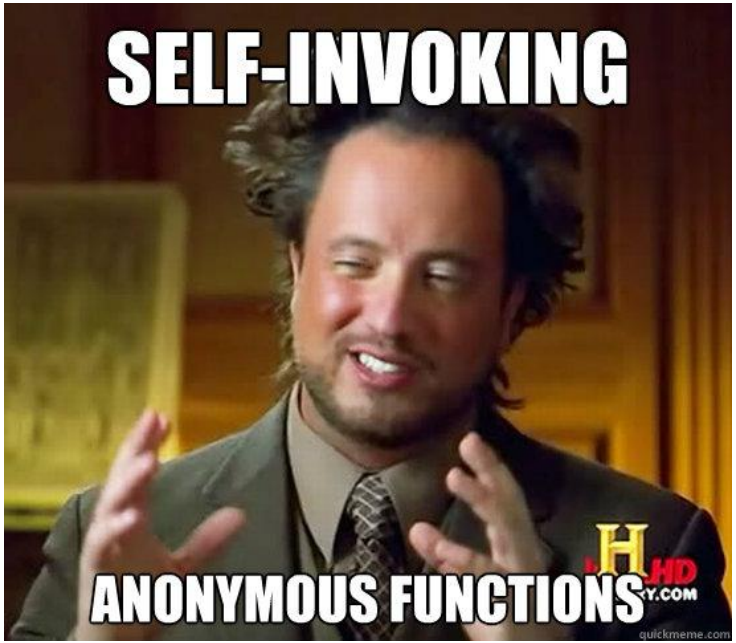
Self invoking anonymous function

Javascript libraries are usually found within a self invoking anonymous function.

```
1 (function(){
2     /* code here */
3 })();
```

Why?

Yeah...



Difference between Global and Local

Global variable = a variable that can be used anywhere within the program

Local variable = a variable that can only be accessed within a function

```
1 var globalVariable;
2
3 function someFunction() {
4     var localVariable;
5 }
```

Public and private (½)

Private variable/function = a variable/function that can only be accessed from within a function

```
1 (function() {
2     var private_var;
3
4     function private_function() {
5       //code
6     }
7 })()
```

Public and private (2/2)

Public variable/function = a variable/function that can be accessed from outside and inside a function

Private variable/function = a variable/function that can only be accessed from within a function

JavaScript

```
var example1 = (function(){
    var private_function = function(){
        console.log('Private function.');
    }
    return {
        public_function: function(){
            private_function();
        }
    }

})();

example1.private_function();
example1.public_function();
```

Self instantiating library

```javascript
(function(){
    var myLib = function(o){
        if(!(this instanceof myLib)){
            return new myLib(o);
        }

        /* ... */
    }

})()
```

To make life easy for the user, lets let the library auto instantiate itself.

Yeah...


Auto instantiating
classes functions
Thingies
HD
HISTORY.COM

Checking parameters

Check to see if user really added a parameter

```
(function () {
    var myLib = function (o) {
        /* ... */
        if (typeof o === 'undefined') {
            o = {};
        }
        /* ... */

    }

})()
```

Setting Attributes

We set class attributes from the parameters.

```javascript
(function(){
    var myLib = function(o){
        /* ... */
        this.attribute1 = o.attribute1 || null;
        this.attribute2 = o.attribute2 || null;
        /* ... */
    }
})()
```

Creating class functions

?

```javascript
(function () {
    var myLib = function (o) {
        /* ... */
    }
    myLib.prototype.function1 = function () {
        console.log('My first class function: ' + this.attribute1);
    };
})()
```

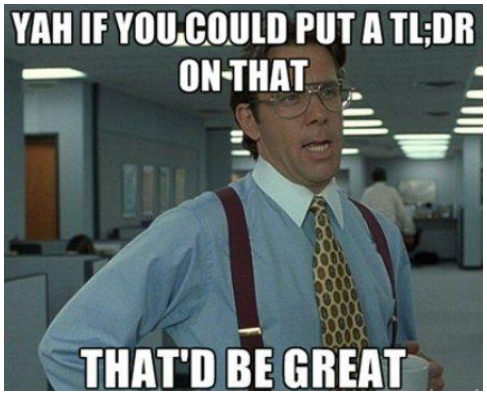
OR

JavaScript

```javascript
(function () {
    var myLib = function (o) {
        /* ... */
        this.function1 = function(){
            console.log('My first class function: ' + this.attribute1);
        };
    }
})()
```

Creating the “fn” shortcut

“prototype” is too long, let's shorten it.




```
(function () {
    var myLib = function (o) {
        /*...*/
    }
    myLib.fn = myLib.prototype = {
        init: function(){}
    }
    myLib.fn.function1 = function () {
        console.log('My first class function: ' + this.attribute1);
    };
})()
```

Exporting

Attach “myLib” to the window be able to use it globally.

```
(function () {
    var myLib = function (o) {
        if (!(this instanceof myLib)) {
            return new myLib(o);
        }
        if (typeof o === 'undefined') {
            o = {};
        }
        this.attribute1 = o.attribute1 || null;
        this.attribute2 = o.attribute2 || null;
    }
    myLib.fn = myLib.prototype = {
        init: function () {}
    }
    myLib.fn.function1 = function () {
        console.log('My first class function: ' + this.attribute1);
    };

    window.myLib = myLib;
})()
```

But wait there’s more...

Asynchronous function + library in one?

Create a library “Person”

Create a simple person library that:

- Has a constructor that accepts an obj “{}” parameter
  - **Obj** must contain the ***name, age, gender, and age*** of the person we want to create
- Auto instantiates itself
- Checks if the obj “{}” parameter is well defined
- Create the “fn” shortcut
- Create a **method** or **plugin** that transforms all the attributes to one string and returns it
- alert() or console.log() the string.

Complete the personHandler library

Click me!

Complete this library by making all the big **for** loops asynchronous!