YOU DON'T KNOW

JAVASCRIPT


7 OCTOBER 2017
VEER YOU GOT MEMES?
I GOT 100+ FOLLOW @VEERMEME
INSTAGRAM HANDLE

HMMM...

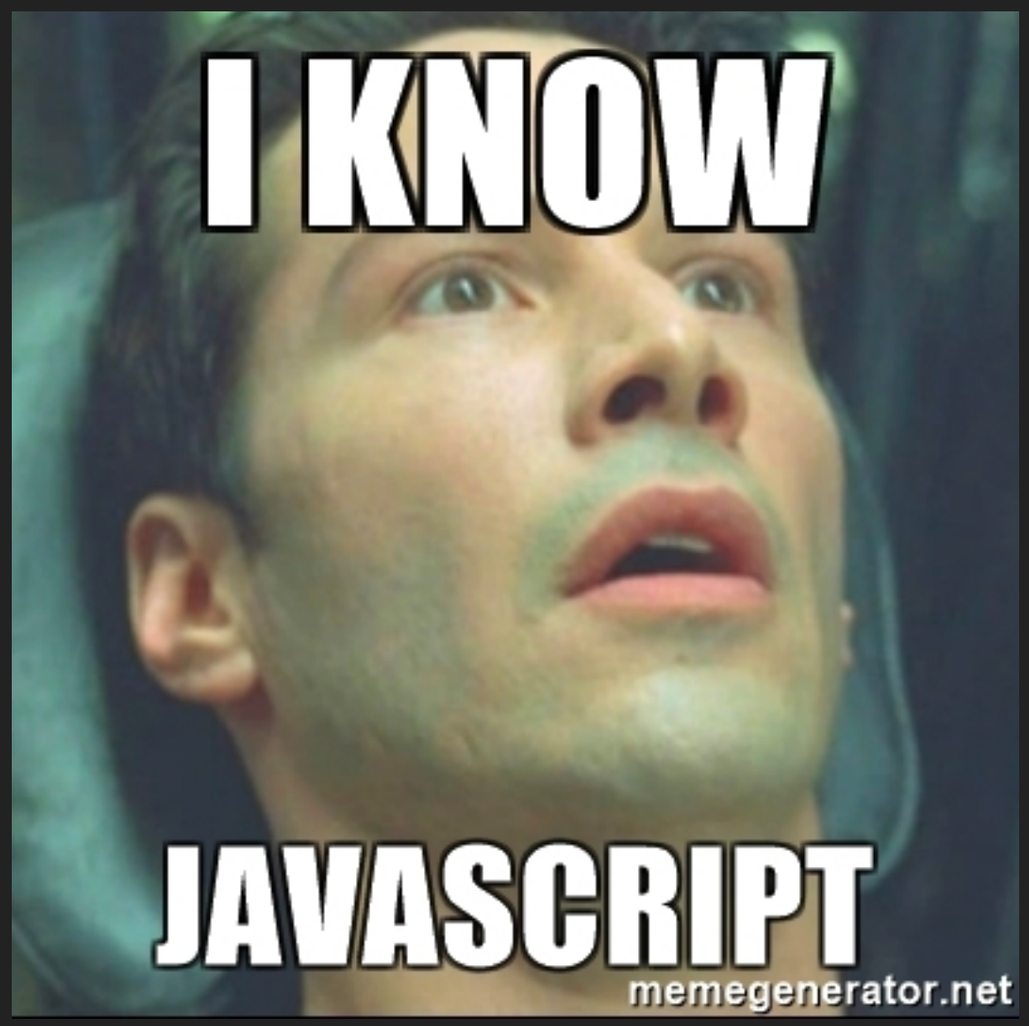
WHO ADDED THIS SLIDE...

VEEEEEERRRR!!

SORRY . . .
PLEASE CONTINUE. . .

VEER

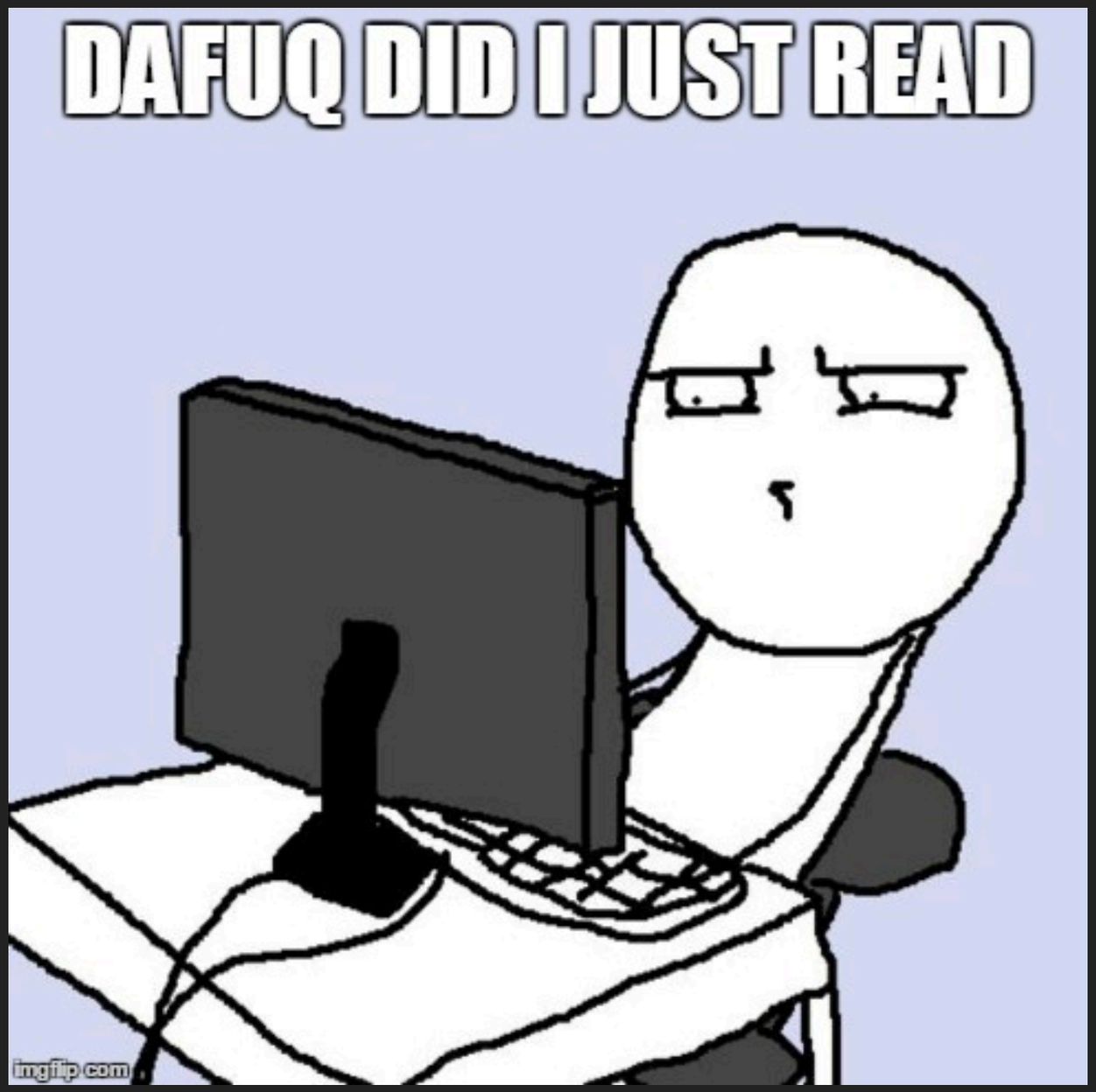

JAVASCRIPT
JAVASCRIPT EVERYWHERE


I KNOW
JAVASCRIPT
memegenerator.net

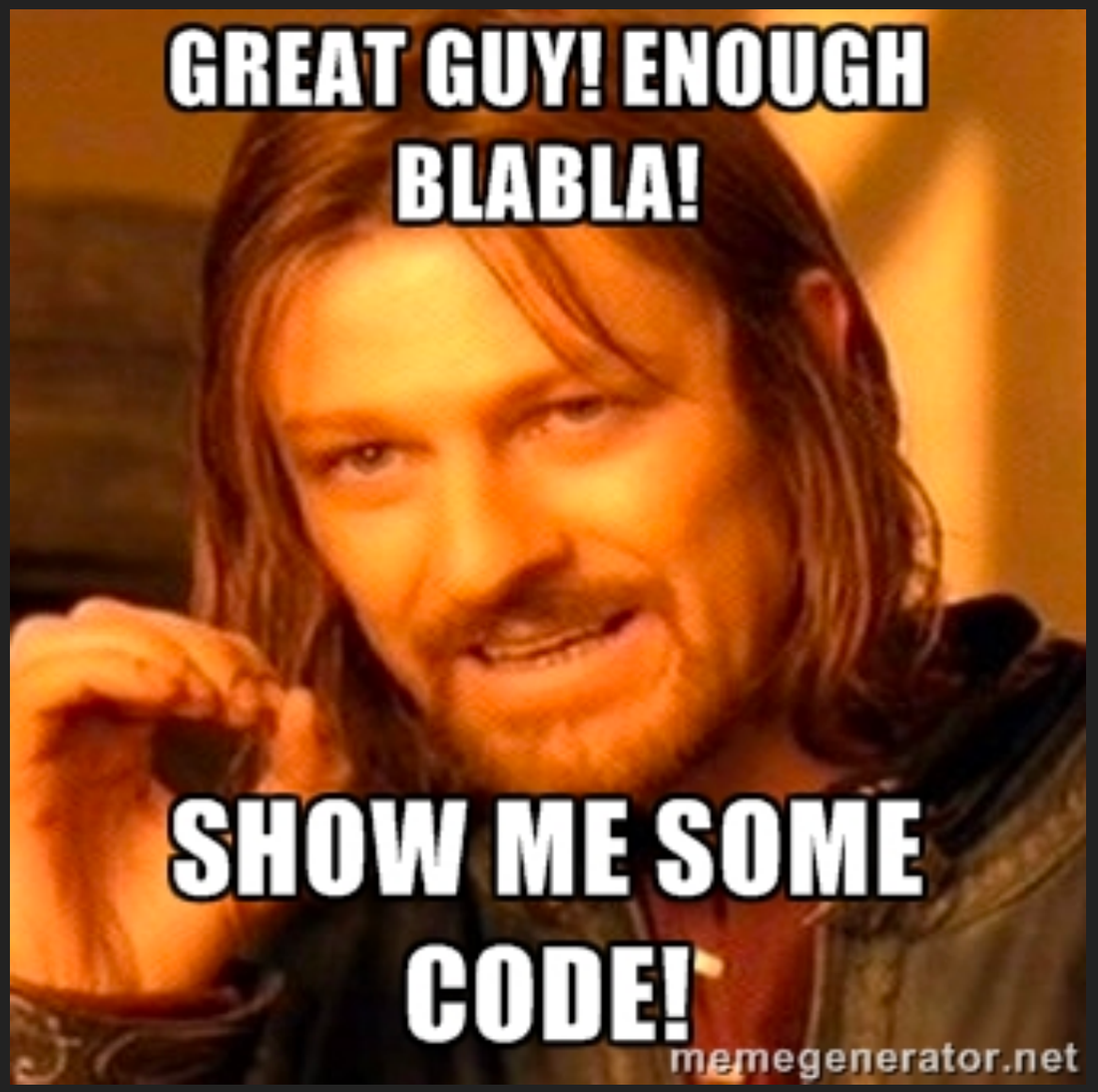
“WHATS IS JAVASCRIPT?”

aaaa...hmmm..haaa...

A SINGLE-THREADED NON-BLOCKING ASYNCHRONOUS CONCURRENT LANGUAGE


DAFUQ DID I JUST READ
imgflip.com

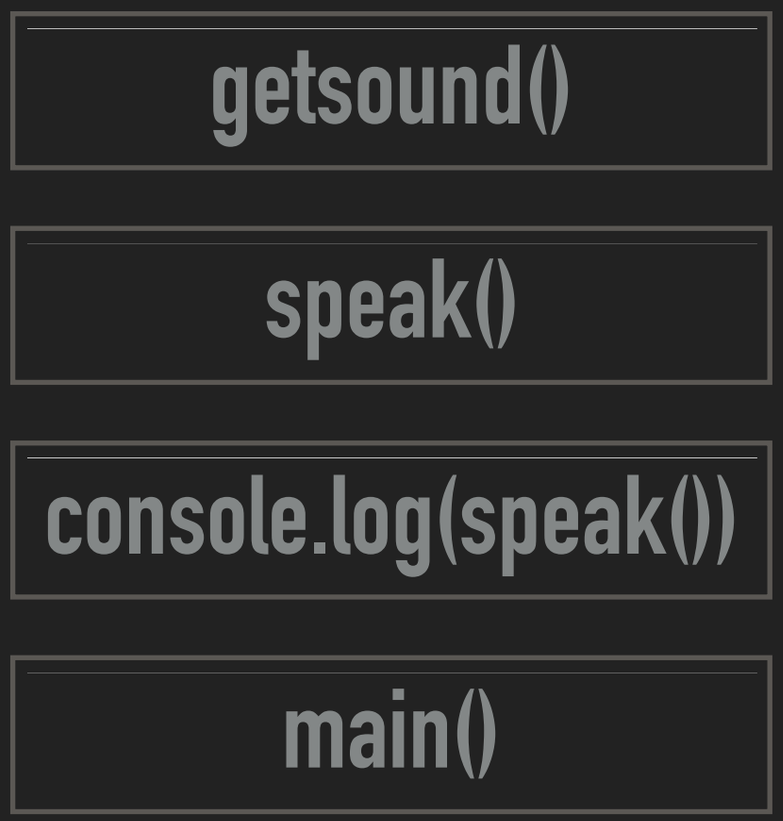
IT HAS
A CALL STACK
AND EVENT LOOP
A CALLBACK QUEUE
& SOME OTHER APIS


GREAT GUY! ENOUGH BLABLA!
SHOW ME SOME CODE!
memegenerator.net

CODE

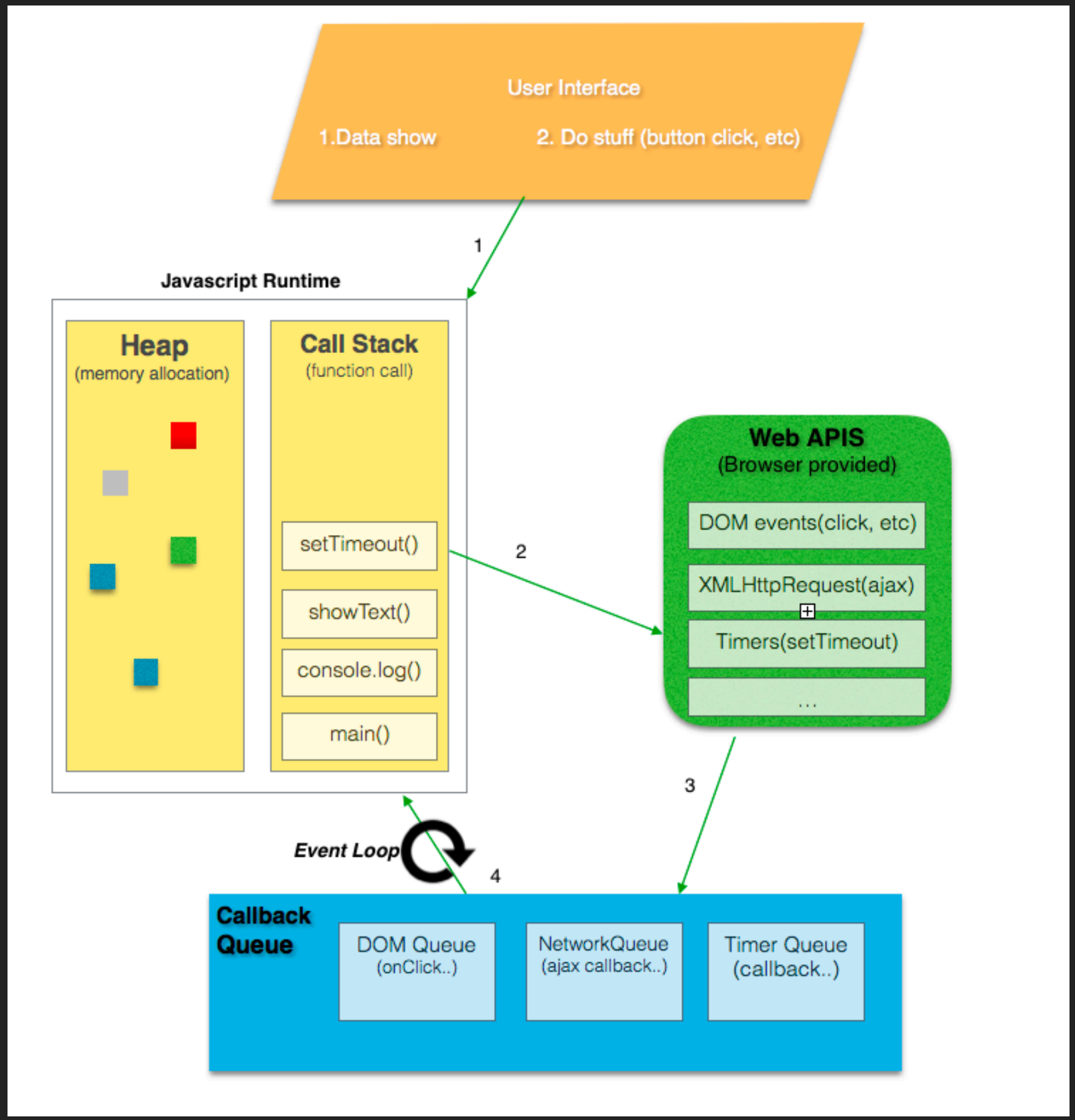
```
1  function getsound() {
2      return "bow bow!";
3  }
4
5  function speak() {
6      return getsound();
7  }
8
9  console.log(speak());
```

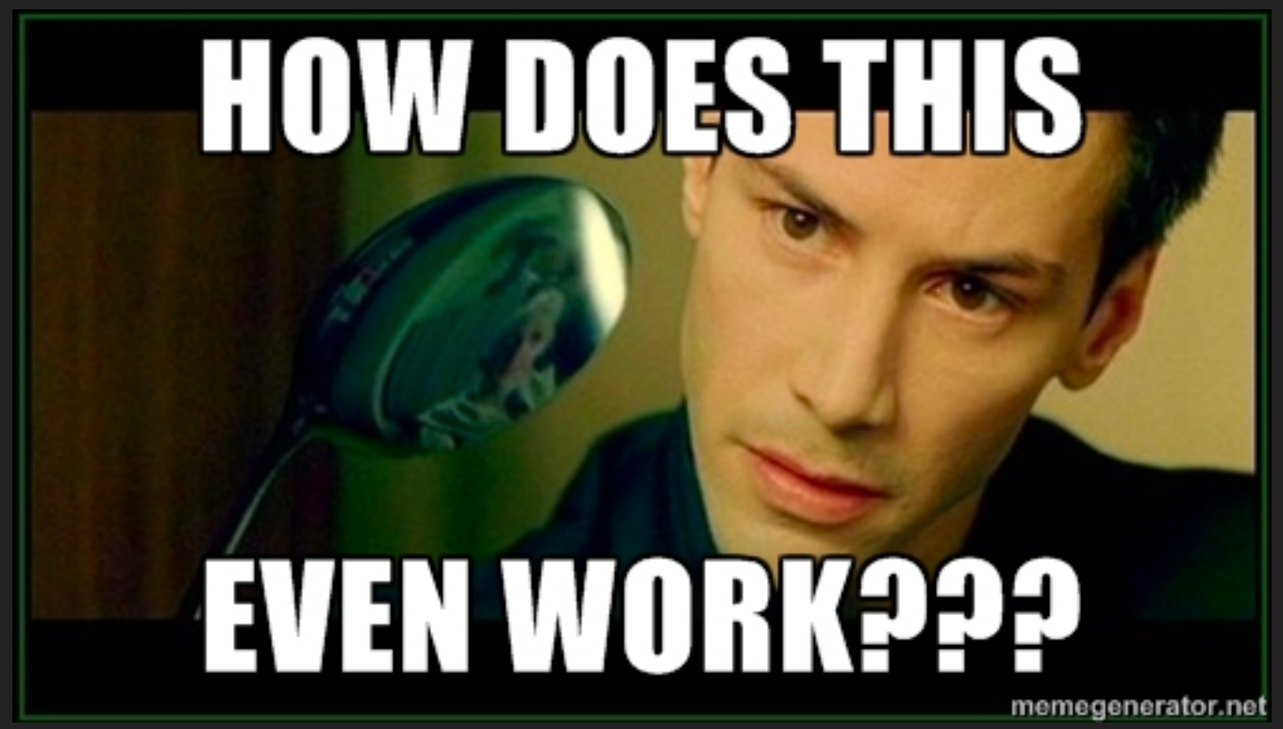
CALL STACK



OUTPUT

```
bow bow!
```


User Interface
1.Data show
2. Do stuff (button click, etc)
1
Javascript Runtime
Heap
(memory allocation)
Call Stack
(function call)
setTimeout()
showText()
console.log()
main()
2
Web APIS
(Browser provided)
DOM events(click, etc)
XMLHttpRequest(ajax)
Timers(setTimeout)
...
3
Event Loop
4
Callback Queue
DOM Queue
(onClick..)
NetworkQueue
(ajax callback..)
Timer Queue
(callback..)


HOW DOES THIS
EVEN WORK???
memegenerator.net

YA ..I DON'T BELIEVE YOU.. SHOW PROOF!

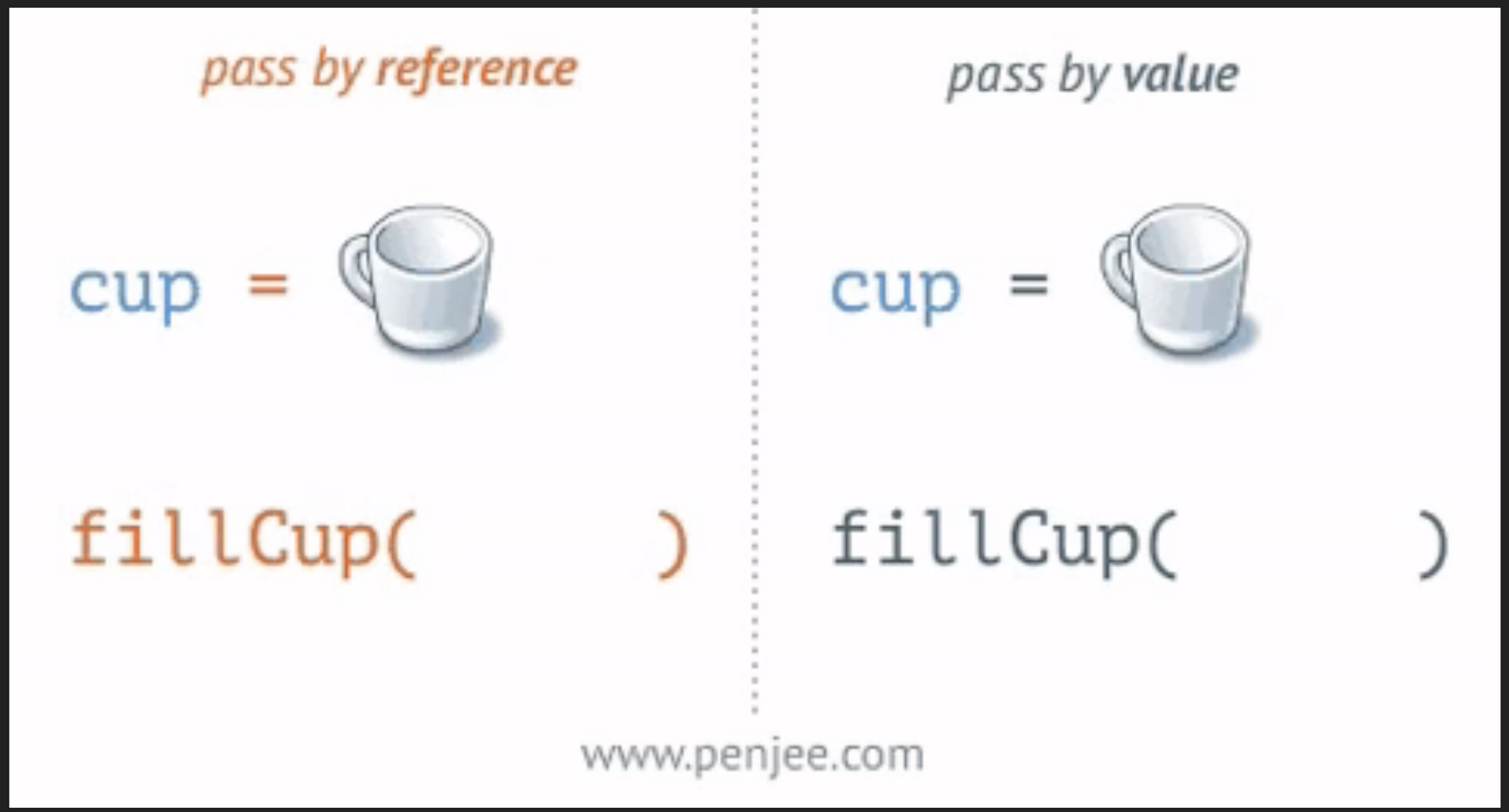
Veer

OK!

“PASS BY REFERENCE / VALUE”

Ofcourse its value..isn’t it

REFERENCE / VALUE



WHATS THE OUTPUT?

```
1  //Strings , Numbers or Boolean
2  var firstname ="veer";
3  var lastname = firstname
4  firstname="vikas"
5
6  console.log("firstname",firstname);
7  console.log("lastname",lastname);
```

```
firstname vikas
lastname veer
```

WHATS THE OUTPUT?

```
1  var dailyfood=["burger","pizza","noodles"];
2
3  var weekendfood=dailyfood;
4
5  weekendfood.push("pav bhaji");
6
7  console.log("weekendfood",weekendfood);
8  console.log("dailyfood",dailyfood);
```

```
weekendfood ▶ (4) ["burger", "pizza", "noodles", "pav bhaji"]
dailyfood ▶ (4) ["burger", "pizza", "noodles", "pav bhaji"]
```

WHATS THE OUTPUT?

```
1  var food = {
2      pav: "bhaji",
3      puri: "bhaji"
4  }
5  var myfood = food;
6  myfood.puri = "shrikhand";
7  console.log("food:", food);
8  console.log("myfood:", myfood);
```

```
food: ▶ {pav: "bhaji", puri: "shrikhand"}
myfood: ▶ {pav: "bhaji", puri: "shrikhand"}
```

BY VALUE

STRING

NUMBER

BOOLEAN

BY REFERENCE

ARRAY

OBJECT

HOW TO PASS ARRAY BY VALUE

```
1  var dailyfood=["burger","pizza","noodles"];
2
3  //Makes a copy
4  var weekendfood=dailyfood.slice();
5
6  weekendfood.push("pav bhaji");
7  console.log("After First Push..");
8  console.log("weekendfood:",weekendfood);
9  console.log("dailyfood:",dailyfood);
```

It’s a shallow copy

Works only for generics i.e string etc, but not for array of object

SHALLOW COPY

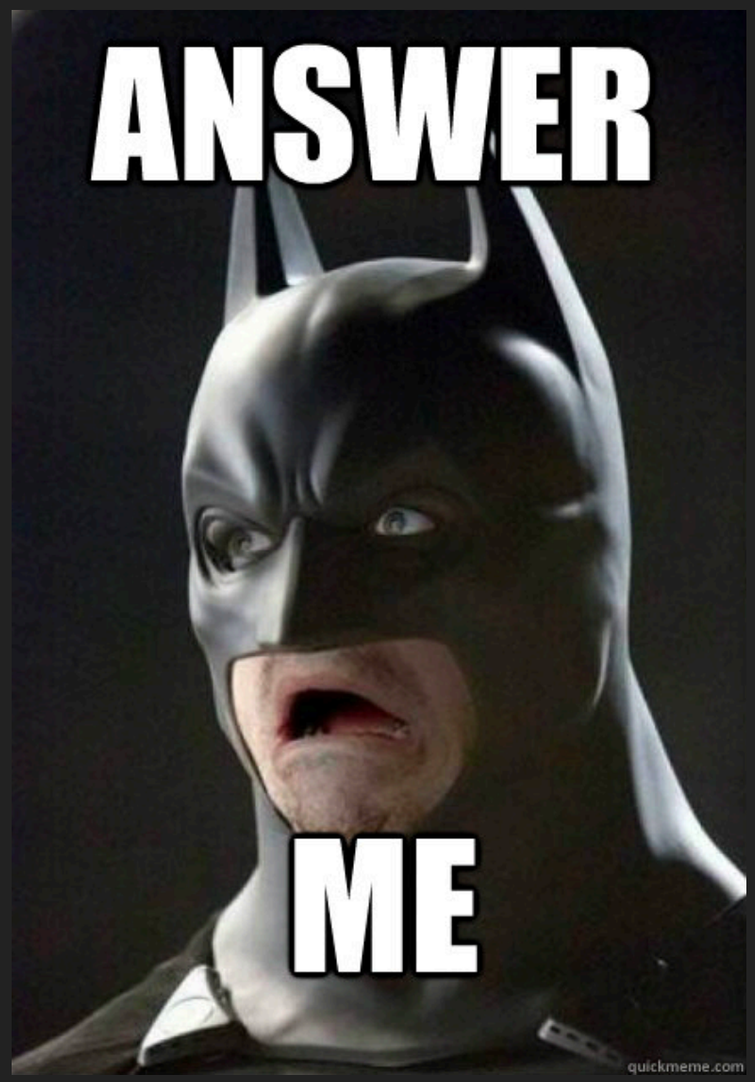
```
1  var dailyfood=["burger","pizza","noodles",[1,2,3]];
```

```
After First Push..
weekendfood: ▶ (5) ["burger", "pizza", "noodles", Array(3), "pav bhaji"]
dailyfood: ▶ (4) ["burger", "pizza", "noodles", Array(3)]
```

```
After Second Push..
weekendfood: ▶ (5) ["burger", "pizza", "noodles", Array(4), "pav bhaji"]
dailyfood: ▶ (4) ["burger", "pizza", "noodles", Array(4)]
```

```
10
11
12  weekendfood[3].push(4);
13  console.log("After Second Push..");
14  console.log("weekendfood:",weekendfood);
15  console.log("dailyfood:",dailyfood);
```

HOW TO PASS OBJECT BY VALUE



HOW TO PASS OBJECT BY VALUE

```
var food = {
    pav: "bhaji",
    puri: "bhaji"
}
var myfood = Object.assign({}, food);
myfood.puri = "shrikhand";
console.log("food:",food);
console.log("myfood:",myfood);
```

```
food: ▶ {pav: "bhaji", puri: "bhaji"}
```

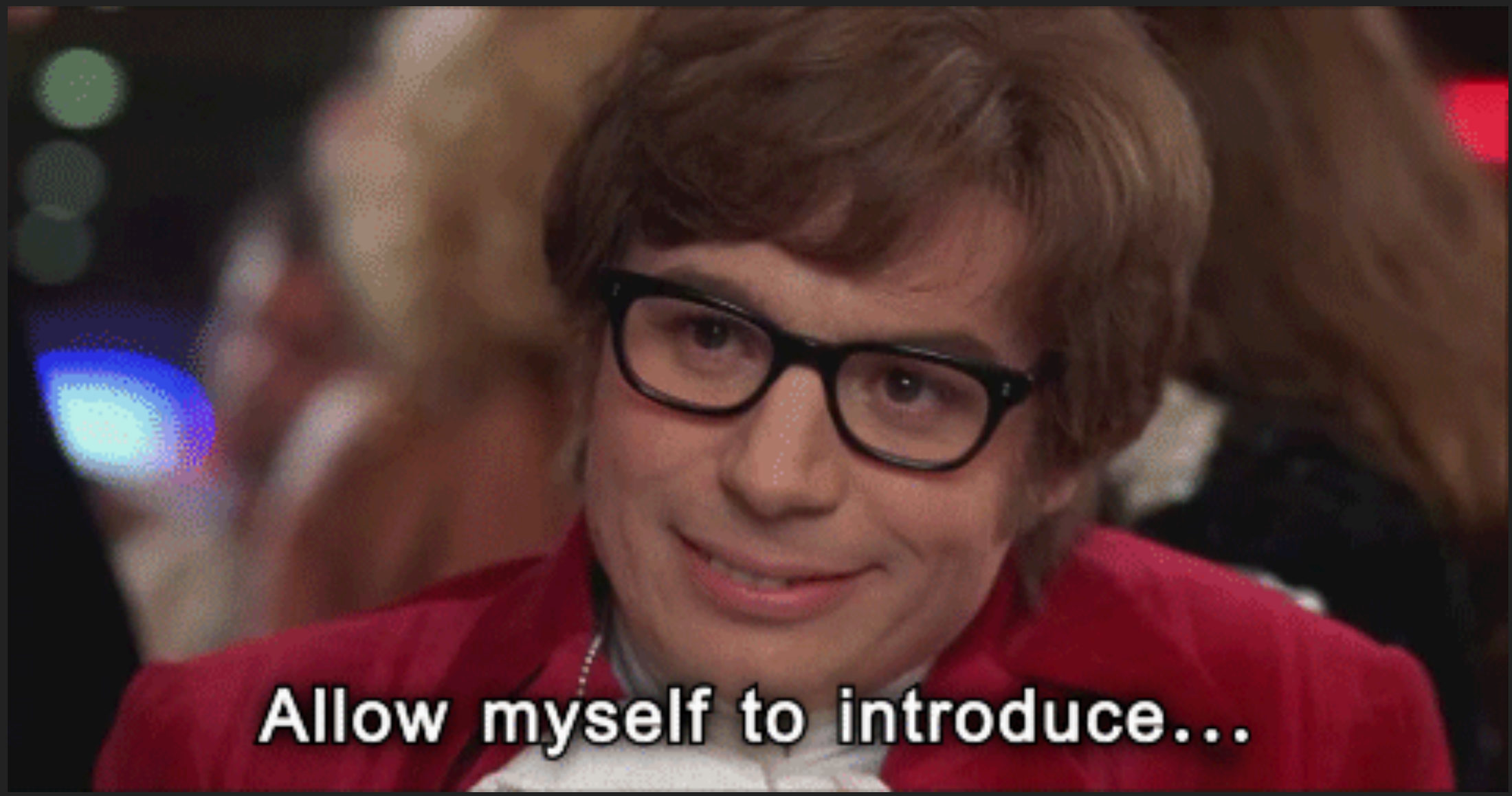
```
myfood: ▶ {pav: "bhaji", puri: "shrikhand"}
```

Object.assign() is useful for merging objects or cloning them shallowly.

“THIS”

WTF is this?

WHEN HE SAYS MYSELF HE REALLY MEANS 'THIS'



A VARIABLE
WITH THE VALUE
OF THE OBJECT
THAT INVOKES THE FUNCTION

THE VALUE OF THIS IS BASED ON THE CONTEXT IN WHICH THE FUNCTION IS CALLED AT RUNTIME

GRRRRR....STILL NOT GETTING IT... :(

Veer

WHATS THE OUTPUT?

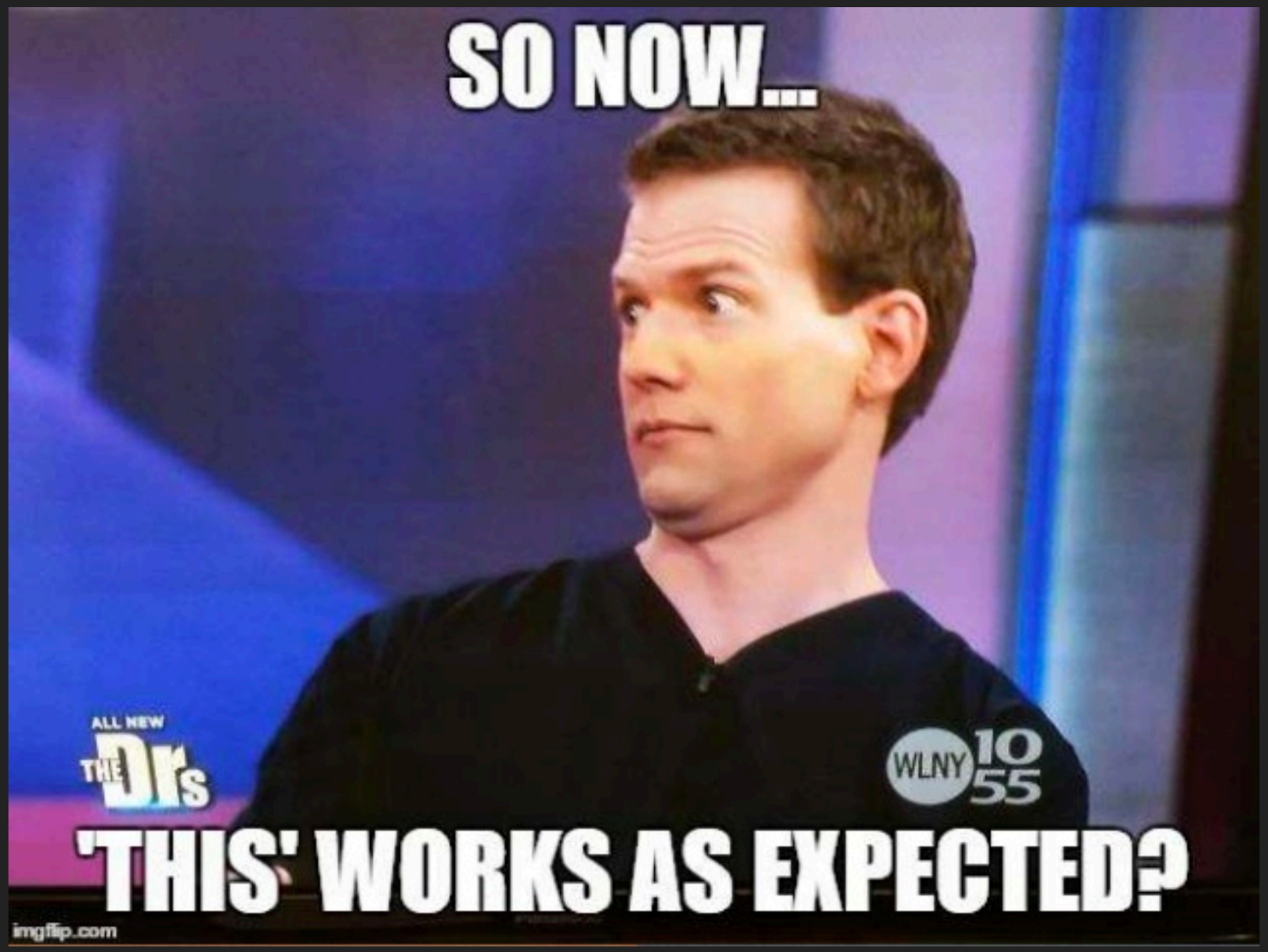
```
1  var sound = "bow bow";
2  var playSound = function() {
3      console.log("Playing.." + this.sound);
4  }
5
6  var cat = {
7      sound: "meow..meow..",
8      playSound: playSound
9  }
10
11 playSound();
12 cat.playSound();
```

```
Playing..bow bow
Playing..meow..meow..
```

WHATS THE OUTPUT?

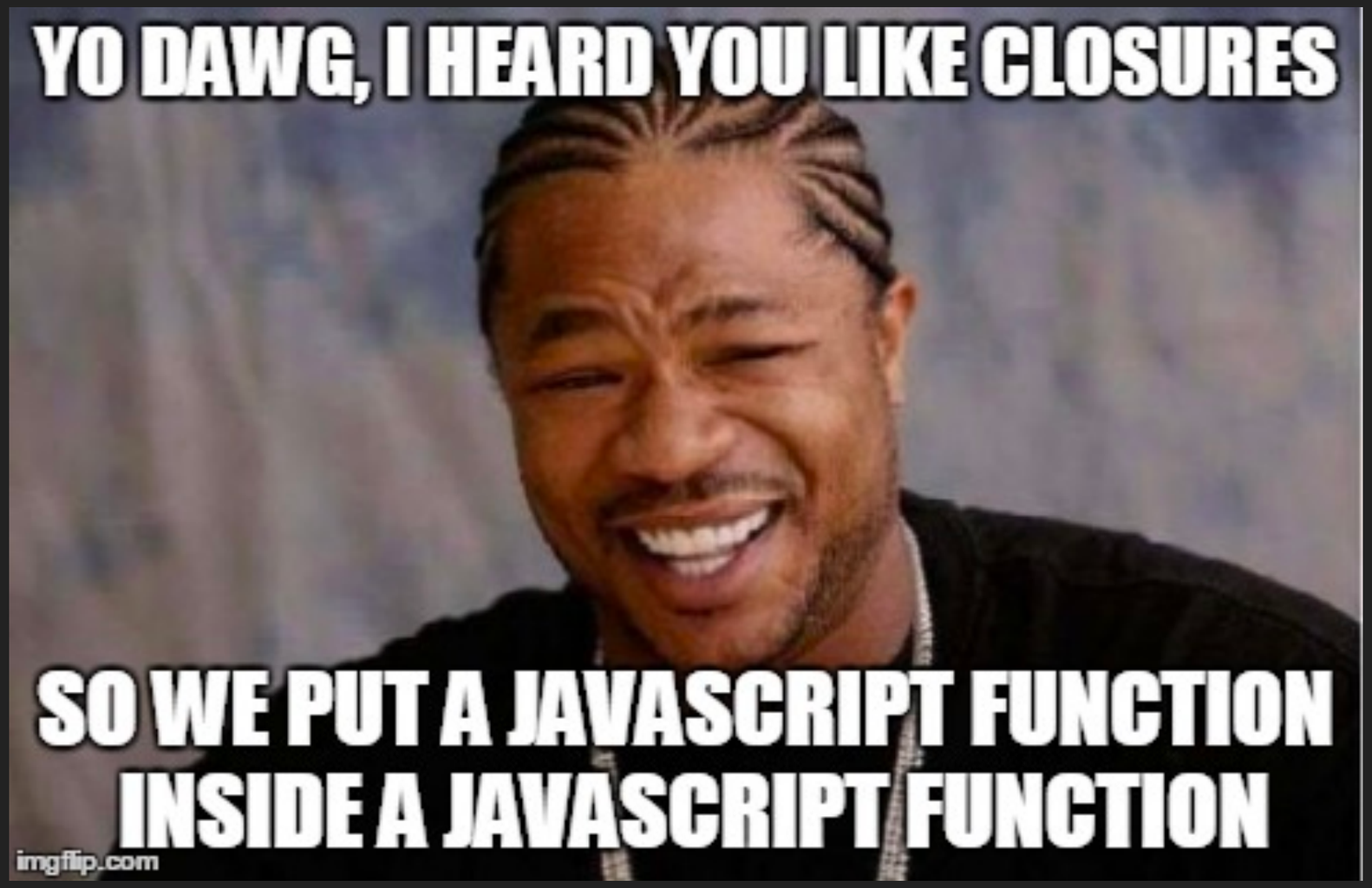
```
1  var dog = {
2      sound: "bow bow",
3      bark: function() {
4          console.log("Barking.." + this.sound);
5          var barkLoudly = function() {
6              console.log("Barking Loudly.." + this.sound);
7          }();
8      }
9  }
10
11 dog.bark();
```

```
dog.bark();
Barking..bow bow
Barking Loudly..undefined
```


SO NOW...
ALL NEW
THE Drs
WLNY 10 55
'THIS' WORKS AS EXPECTED?
imgflip.com

“CLOSURE”

Love this song by chainsmokers ..Ehh?


YO DAWG, I HEARD YOU LIKE CLOSURES
SO WE PUT A JAVASCRIPT FUNCTION INSIDE A JAVASCRIPT FUNCTION
imgflip.com

A LOCAL VARIABLE OF THE FUNCTION – KEPT ALIVE AFTER THE FUNCTION IS RETURNED

THE RETURNED FUNCTION HAS ACCESS TO THE PARENT SCOPE EVEN IF THE PARENT FUNCTION HAS ENDED

GRRRRR....
NO CLOSURE TO THIS... :(

Veer

WHATS THE OUTPUT?

```
1  function makeFun() {
2      var name = "veer";
3
4      function displayName() {
5          console.log("My name is " + name);
6      };
7      return displayName;
8  }
9  var boy = makeFun();
10 boy();
```

```
My name is veer
```

WHATS THE OUTPUT?

```
1  function makeAdder(x) {
2      return function(y) {
3          return x + y;
4      };
5  }
6
7  var add5to = makeAdder(5);
8  var add100to = makeAdder(100);
9
10 console.log(add5to(10));
11 console.log(add100to(10));
```

“PROTOTYPE”

i won’t say a word !

A PROTOTYPE IS A COLLECTION OF PROPERTIES AND METHODS THAT CAN BE AUTOMATICALLY ATTACHED TO A OBJECT WHEN CREATED

EVERY JAVASCRIPT FUNCTION HAS A PROTOTYPE PROPERTY

PRIMARILY USED FOR

INHERITANCE

GRRRRR....
NOT ABLE TO INHERIT.... :(

Veer

```
var activities = {
    speak: function() {
        console.log(this.sound);
    },
    getType: function() {
        console.log(this.type);
    }
}
function animal(type,sound)
{
    this.type=type;
    this.sound=sound;
}

animal.prototype=activities;

var doberman = new animal("dog","bow bow");
var veer = new animal("human","bababa-babab-nana");

console.log("doberman:");
doberman.getType();
doberman.speak();

console.log("veer:");
veer.getType();
veer.speak();
```

WHATS THE OUTPUT?

```
doberman:
dog
bow bow
veer:
human
bababa-babab-nana
```

YOU CAN ADD PROPERTIES TO PROTOTYPE ANY TIME

THE PROTOTYPE CHAIN WILL FIND THE NEW PROPERTY AS EXPECTED

```
var activities = {
    speak: function() {
        console.log(this.sound);
    },
    getType: function() {
        console.log(this.type);
    }
}

function animal(type, sound) {
    this.type = type;
    this.sound = sound;
}

animal.prototype = activities;

var doberman = new animal("dog", "bow bow");
var veer = new animal("human", "bababa-babab-nana");

console.log("doberman:");
doberman.getType();
doberman.speak();

console.log("veer:");
veer.getType();
veer.speak();

activities.run = function() {
    console.log(this.type + " is running");
}

doberman.run();
veer.run();
```

WHATS THE OUTPUT?

```
doberman:
dog
bow bow
veer:
human
bababa-babab-nana
dog is running
human is running
```

YOU CAN EXTEND NATIVE OBJECTS

```
Array.prototype.hello = function() {
    console.log("hello i am array")
}
;
var data = [1, 2, 3, 4];
data.hello();
```

"MODULE PATTERN"

PATTERNS IN JAVASCRIPT
..SERIOUSLY?

MAINTAINABILITY
NAMESPACING
REUSABILITY

IMMEDIATELY INVOKING FUNCTION EXPRESSION (IIFE)

```
(function() {
    console.log("I am invoked")
}());
```

GLOBAL VARIABLE IMPORT

```
var globalvar = [1, 2, 3];
(function(a) {
    console.log("I am using global variable" + a.length);
}(globalvar));
```

OBJECT INTERFACE

```
var animal = (function(s) {
    var sound = s;
    return {
        playSound: function() {
            console.log(sound);
        }
    }
}("bow bow"))
console.log("Access Sound:" + animal.sound);
animal.playSound();
```

```
Trying to access sound undefined
bow bow
```

COMMONJS MODULE (NODE JS)

```
function myModule() {
  this.hello = function() {
    return 'hello!';
  }

  this.goodbye = function() {
    return 'goodbye!';
  }
}

module.exports = myModule;
```

```
var myModule = require('myModule');

var myModuleInstance = new myModule();
myModuleInstance.hello(); // 'hello!'
myModuleInstance.goodbye(); // 'goodbye!'
```

Explicit dependencies
Loads module synchronously


WHY SO
SYNCHRONOUS?
memegenerator.net

AMD
(ASYNCHRONOUS MODULE DEFINITION)

```
1  define(['myModule', 'myOtherModule'], function(myModule, myOtherModule) {
2
3      console.log(myModule.hello());
4  });
```

Explicit dependencies
Loads module asynchronously

WAKE HIM UP!

“FUNCTIONS”

Ahh! I know this
Dont’t I?

FUNCTION CONSTRUCTOR

Every JavaScript function is actually a `Function` object.

Syntax

```
new Function ([arg1[, arg2[, ...argN]],] functionBody)
```

Parameters

`arg1, arg2, ... argN`

Names to be used by the function as formal argument names. Each must be a string that corresponds to a valid JavaScript identifier or a list of such strings separated with a comma; for example "`x`", "`theValue`", or "`a,b`".

`functionBody`

A string containing the JavaScript statements comprising the function definition.

FUNCTION CONSTRUCTOR

```
var playSong =  new Function(
                    "line1",
                    "line2",
                    "line3",
                "return line1+' '+line2+' '+line3"
            );
console.log(playSong("tamma", "tamma", "looge"));
```

Can be used instead of eval.

Jquery Uses to covert JSON String to Object

BIND

```
var pokemon = {
    firstname: 'Pika',
    lastname: 'Chu '
};

var pokemonName = function() {
    var fullname = this.firstname + ' ' + this.lastname;
    console.log(fullname+ 'I choose you!');
};

// creates new object and binds pokemon.
//'this' of pokemon === pokemon now
var logPokemon = pokemonName.bind(pokemon);

logPokemon(); // 'Pika Chu I choose you!'
```

When called, has its ‘this’ keyword set to the provided value

It explicitly lets you define the value of ‘this’

CALL / APPLY

```
1  var pokemon = {
2      firstname: 'Pika',
3      lastname: 'Chu '
4  };
5
6  var pokemonName = function(snacks,hobby) {
7      var fullname = this.firstname + ' ' + this.lastname;
8      console.log(fullname+ "loves "+snacks+" and "+hobby);
9  };
10
11 pokemonName.call(pokemon,'sushi','algorithms');
12 pokemonName.apply(pokemon,['sushi','algorithms']);
```

call() method calls the function with a given this value and arguments provided individually

apply() method calls the function with a given this value and an array of all arguments

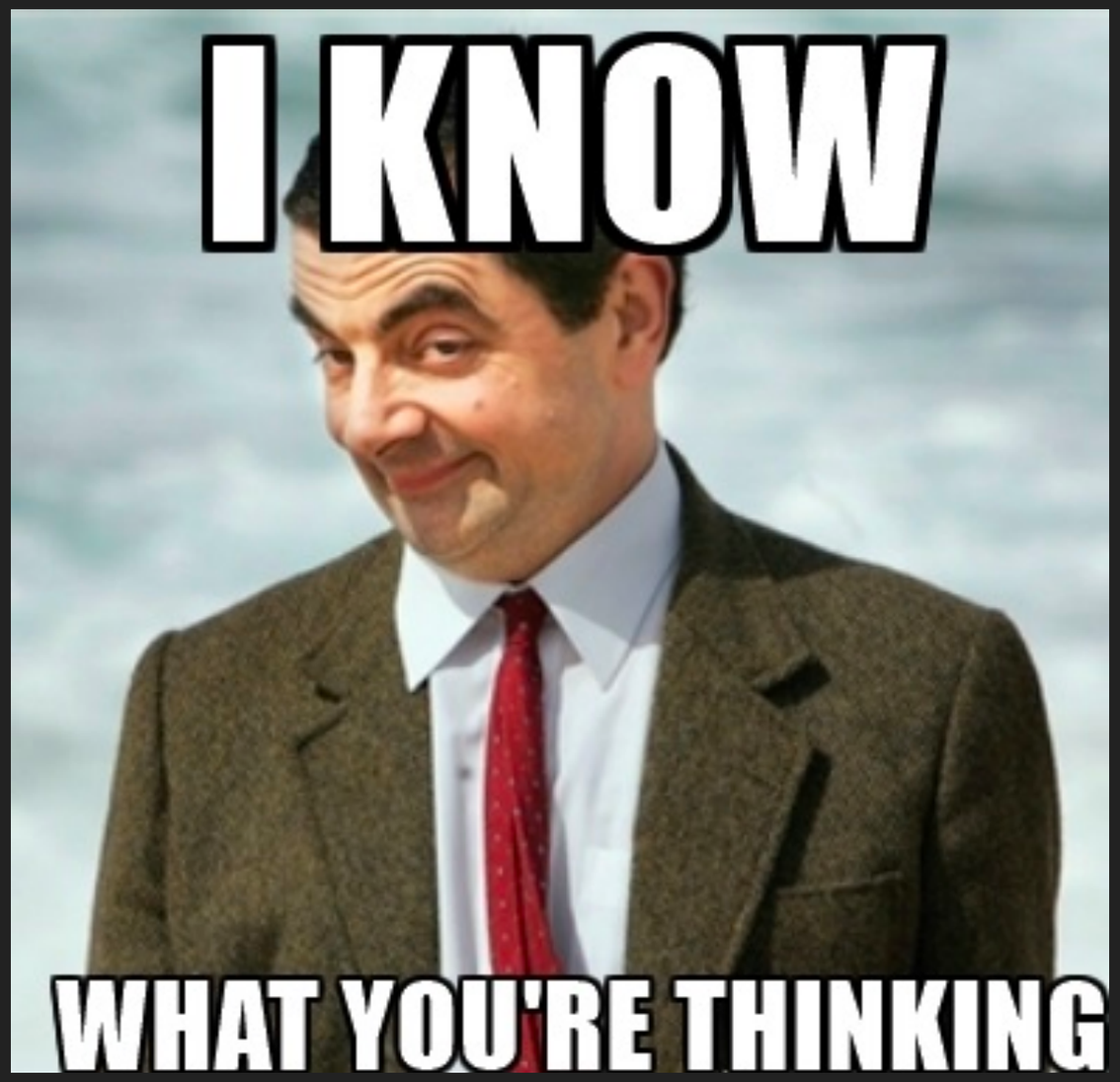
DIFFERENCE

BIND & CALL/APPLY

Call accepts additional parameters

Call executes the function right away

Call does not make a copy of the function it is being called on


I KNOW
WHAT YOU'RE THINKING

LETS TAKE A BREAK!

Vikas

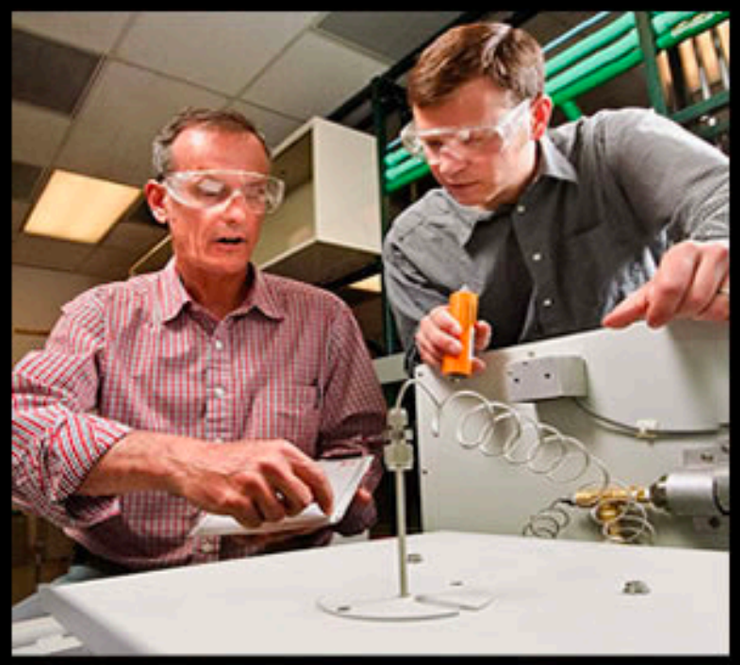
Web Developer

What my friends think I do

What my Mum thinks I do

What society thinks I do

What my boss thinks I do



What I think I do

What I actually do

andyzoric.com

WEB DEVELOPER WITH A JOB

WEB DEVELOPER WITHOUT A JOB

BIG CLAP FOR PAKYA..

THANKS FOR BEING A SPORT

“EVAL”

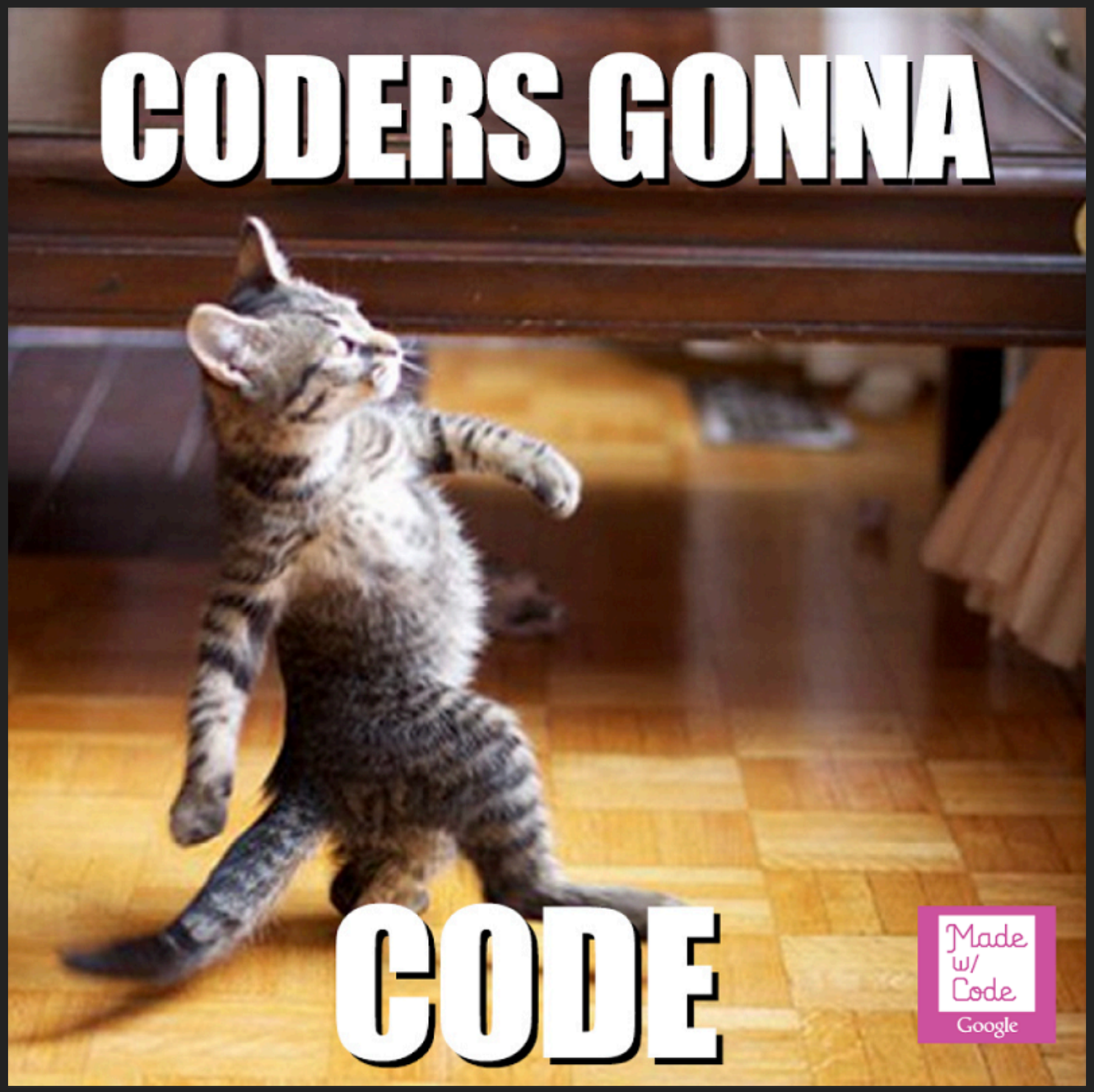
I Heard its Evil..

CHECK THIS FOR MORE INFO

Click

“CLEAN CODE”

Swach Code Abhiyaan :P


CODERS GONNA
CODE
Made w/ Code
Google

RETHINK LOOP

Lets forget for loop

```
var people = [
    {name:"Veer, Prabhu", age:2,gender:"M" ,occupation:"dancer"},
    {name:"Vikas, Prabhu", age:34,gender:"M",occupation:"boss"},
    {name:"Mohini, Memane", age:25,gender:"F",occupation:"coder"},
    {name:"Rahul, Nair", age:3000,gender:"M",occupation:"robot"},
    {name:"Devendra, Walnaj", age:25,gender:"M",occupation:"scientist"}
];
```

RETHINK LOOP-FILTER

```
//Filter for people eligible for voting
var voters = people.filter(function(person) {
    return person.age > 18 ? true : false;
})
console.log("Voters");
console.table(voters);
```

RETHINK LOOP-MAP

```
16  //Get full names with occupation
17  var descriptions = people.map(function(person)
18  {
19      return person.name +" is a "+person.occupation;
20  });
21  console.log("Descriptions");
22  console.table(descriptions);
```

RETHINK LOOP-SORT

```
25  //Sort by age or each person
26  var youngtoolder=people.sort(function(last,next){
27      return (last.age>next.age)? 1:-1;
28  })
29  console.log("Younger");
30  console.table(youngtoolder)
```

RETHINK LOOP-REDUCE

```
33  //get the total age
34  var totalage = people.reduce(function(total, person) {
35      return total + person.age
36  }, 0)
37  console.log("Total Age:" + totalage);
```

RETHINK LOOP – SUMMARY

```
map, filter, and reduce
explained with emoji 😂

map([🐮, 🌽, 🐔, 🌽], cook)
=> [🍔, 🍟, 🍗, 🍿]

filter([🍔, 🍟, 🍗, 🍿], isVegetarian)
=> [🍟, 🍿]

reduce([🍔, 🍟, 🍗, 🍿], eat)
=> 💩
```

RETHINK LOOP – TASKS

Get people with gender male & scientist

Sort by last name

Get total count of male & female

RETHINK IF – TERNARY

if else pattern

```
1  function postTransaction() {
2
3      if (isTransactionvalid()) {
4          return saveTransaction();
5      } else {
6          return rejectTransaction();
7      }
8  }
```

condition ? expr1 : expr2

```
11 function postTransaction() {
12     return isTransactionvalid() ?
13         saveTransaction() : rejectTransaction();
14 }
```

RETHINK IF – TERNARY

What about else if

```
17  function postTransaction(transdata)
18  {
19      if(transdata.assetclass=="MF"){
27      }
28      else{
29          return rejectTransaction(transdata);
30      }
31  }
```

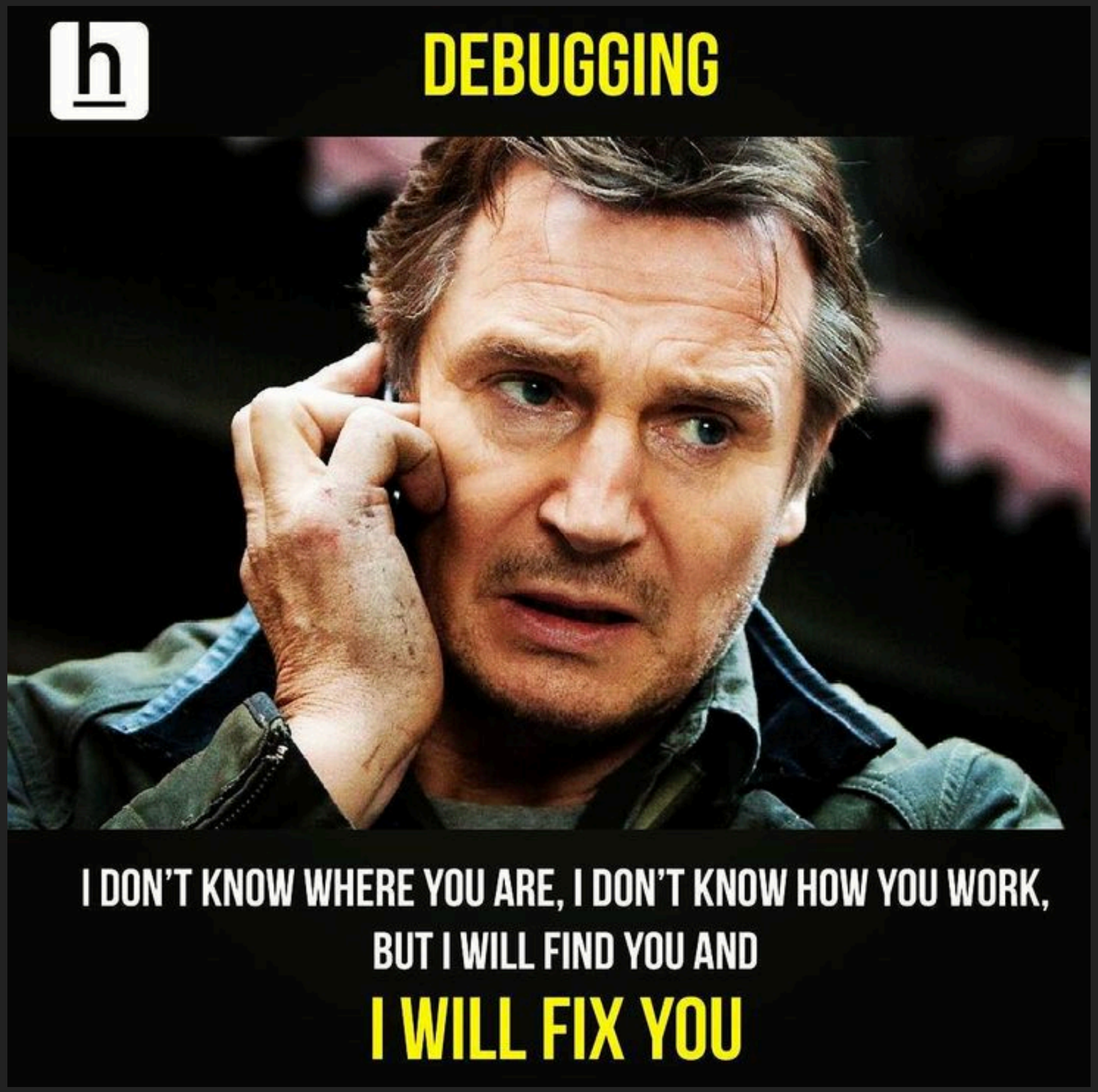
```
33  function postTransaction(transdata)
34  {
35      return
36      transdata.assetclass=="MF"    ? saveMFTransaction(transdata);
37      transdata.assetclass=="EQ"    ? saveEQTransaction(transdata);
38      transdata.assetclass=="DEPO"  ? saveDEPOTransaction(transdata);
39                                    : rejectTransaction(transdata);
40  }
```

RETHINK SWITCH

```
var dataset={
    "EQ":getEQData,
    "MF":getMFData,
    "DEPO":getDEPOData,
    "default":getBlankData
}
function getDataset(transdata)
{
    return
    dataset[transdata.assetclass]
    ?dataset[transdata.assetclass].call(this,transdata);
    :dataset.default.call(this.transdata)
}
```

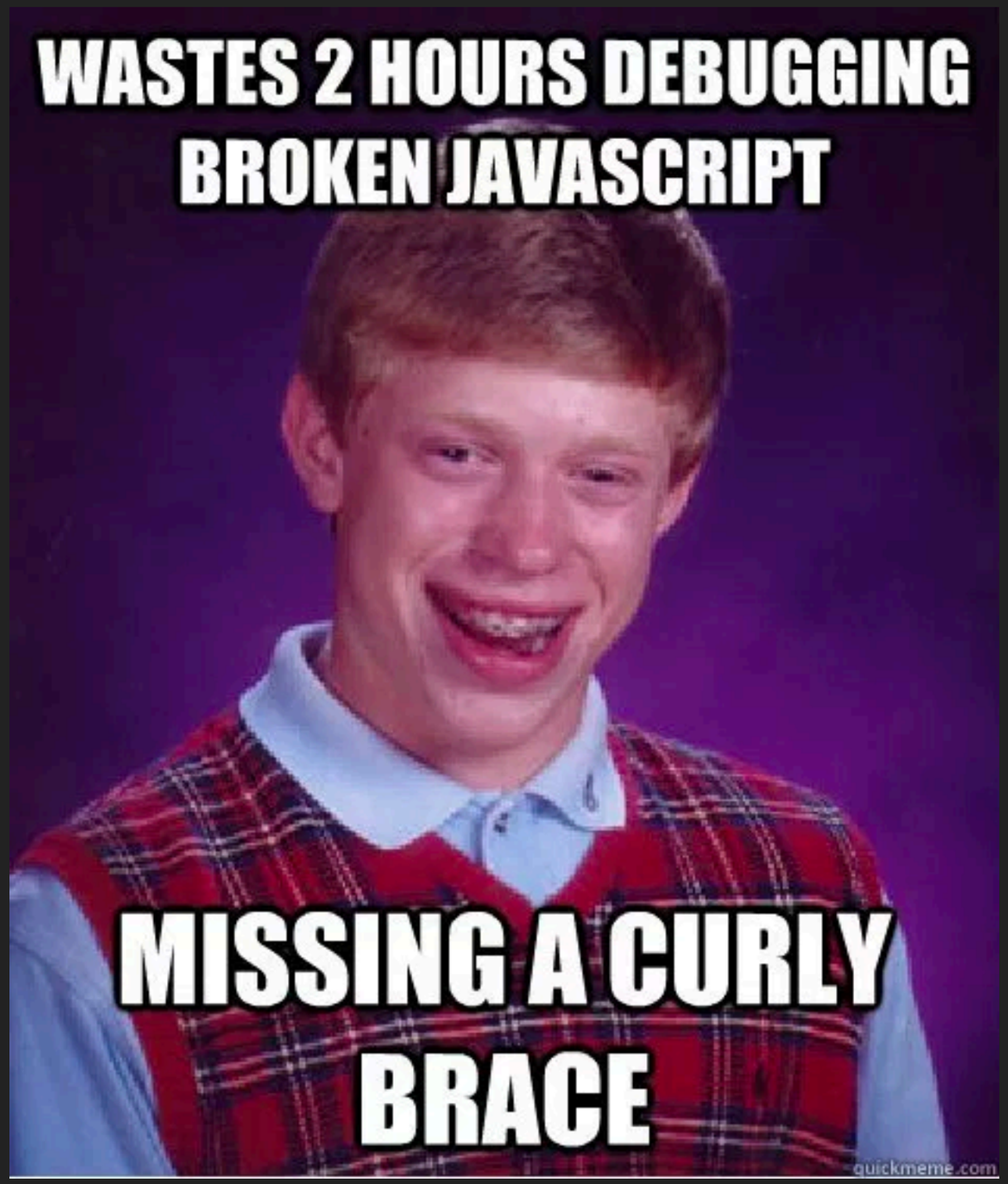
“DEBUG”

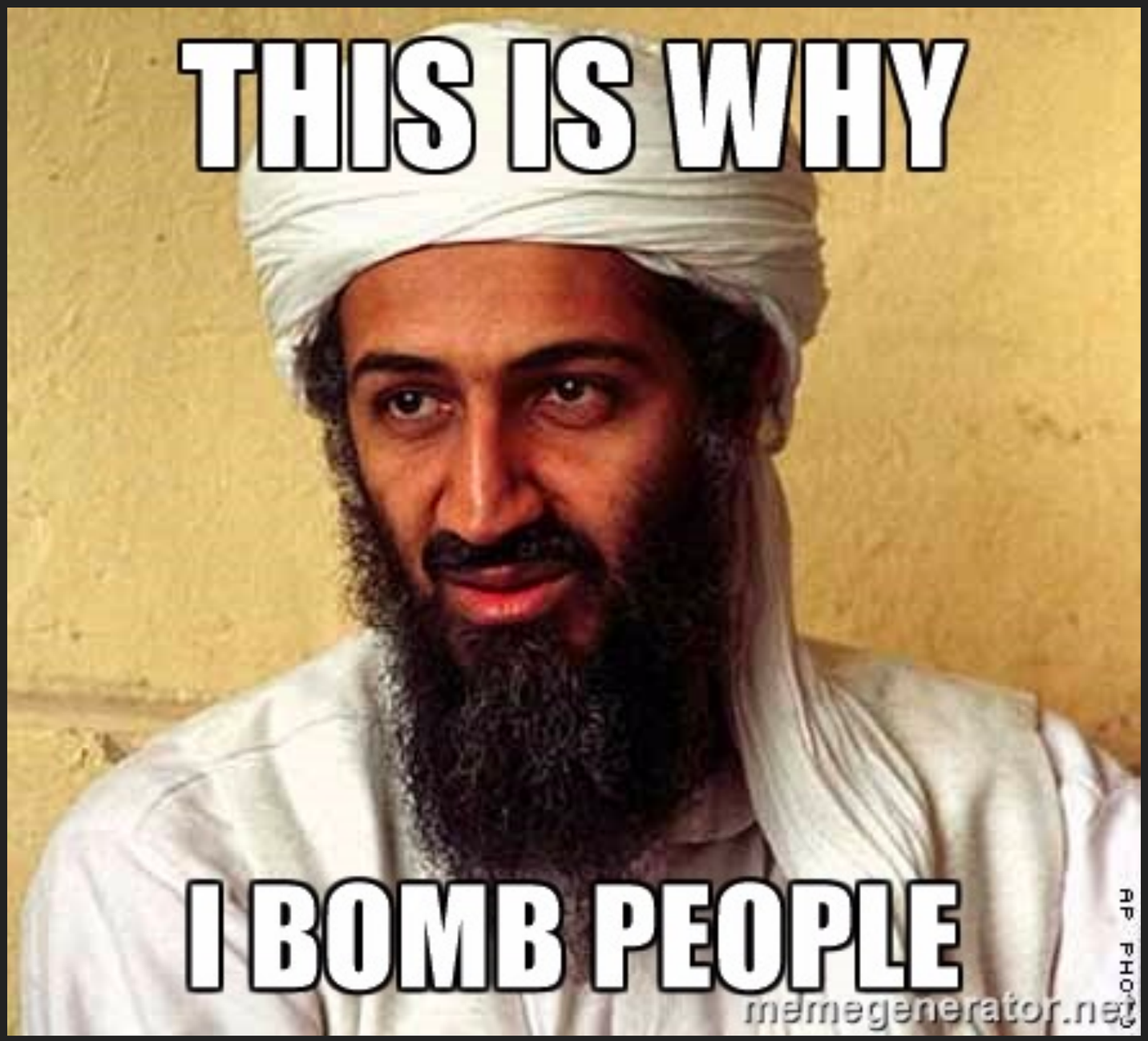
Its not a bug..its a feature


DEBUGGING
I DON'T KNOW WHERE YOU ARE, I DON'T KNOW HOW YOU WORK,
BUT I WILL FIND YOU AND
I WILL FIX YOU

HOW WILL YOU FIX IT

ADD ALERT AT EACH LINE.


WASTES 2 HOURS DEBUGGING
BROKEN JAVASCRIPT
MISSING A CURLY
BRACE
quickmeme.com


THIS IS WHY
I BOMB PEOPLE
memegenerator.net
AP PHOTO


SCOTTY, I NEED
MORE POWER!
memegenerator.net


ONE DOES NOT SIMPLY DO WEB DEVELOPMENT
WITHOUT CHROME DEVTOOLS
imgflip.com

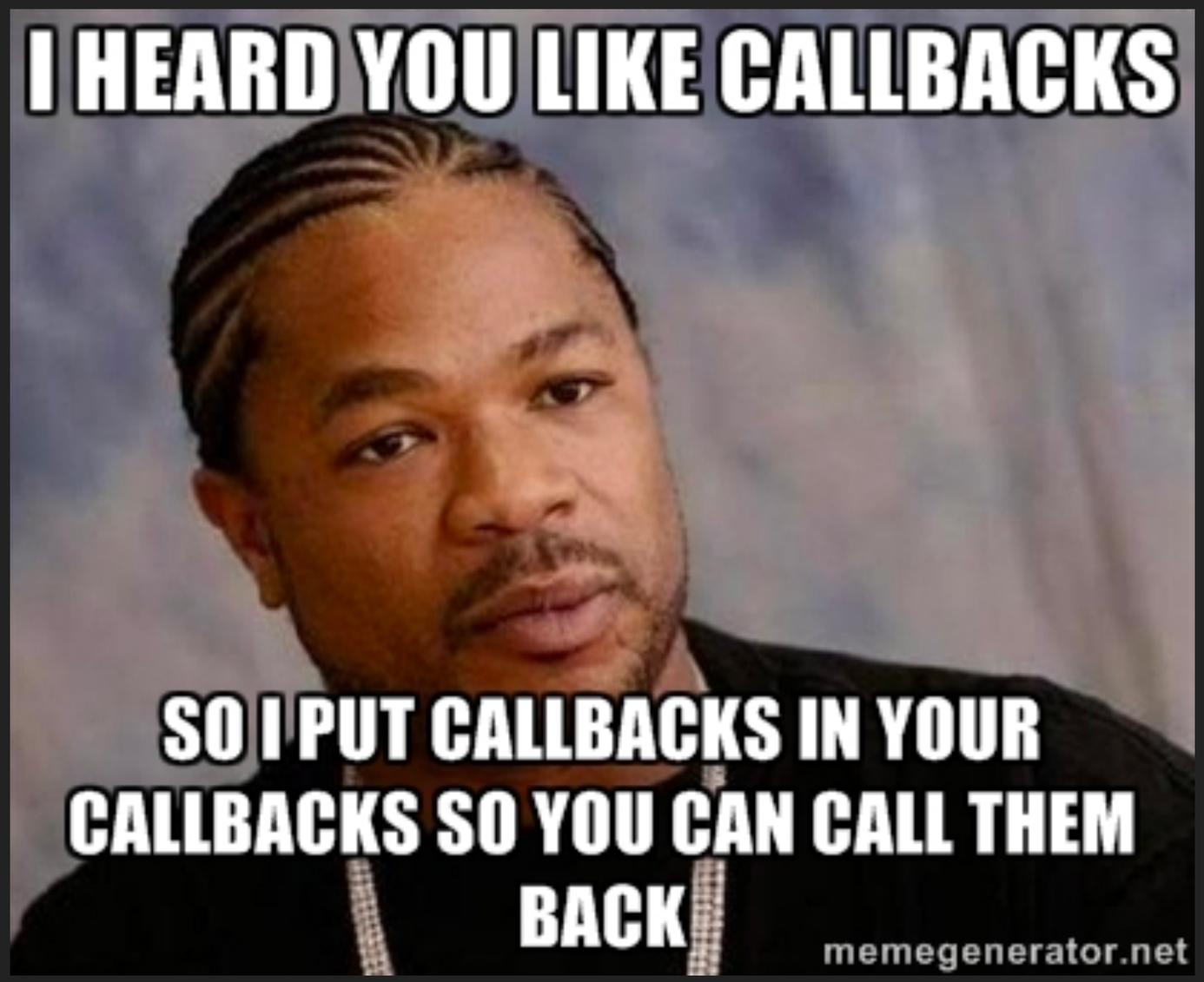
LETS GO TO

CHROME DEVELOPER TOOLS

“CALLBACK HELL”

Ohhhhh!


You want to hear a JavaScript joke?
I'll callback later.
quickmeme.com


I HEARD YOU LIKE CALLBACKS
SO I PUT CALLBACKS IN YOUR CALLBACKS SO YOU CAN CALL THEM BACK
memegenerator.net

```
a(function (resultsFromA) {
  b(resultsFromA, function (resultsFromB) {
    c(resultsFromB, function (resultsFromC) {
      d(resultsFromC, function (resultsFromD) {
        e(resultsFromD, function (resultsFromE) {
          f(resultsFromE, function (resultsFromF) {
            console.log(resultsFromF);
          })
        })
      })
    })
  })
});
```


PROMISES?
imgflip.com

WHAT IS A PROMISE?

A OBJECT THAT MAY PRODUCE A SINGLE VALUE IN THE FUTURE

A PROMISE IS AN ASYNCHRONOUS VALUE

PROMISE HAS 3 STATES

FULFILLED

REJECTED

PENDING


SHOW ME THE CODE
AND I NEED IT NOW!

CREATE PROMISE

```
1  var pinkypromise = new Promise(function(resolve, reject) {
2      setTimeout(function() {
3          resolve("Hi dad, i kept my promise");
4      }, 1000);
5  });
```

CALL PROMISE

```
pinkypromise
    .then(function(res) {
        console.log(res);
        console.log("Good Job");
    })
    .catch(function(res){
        console.error("Please keep your promise");
    })
```

PROMISE CHAINING

```
17  pinkypromise
18      .then(function(res) {
19          console.log(res);
20          return Promise.resolve("Good Job");
21      })
22      .then(function(res) {
23          console.log(res)
24          return new Promise(function(resolve, reject) {
25              setTimeout(function() {
26                  reject("Opps Dad, i broked my promise")
27              }, 500)
28          })
29      })
30      .then(function(res) {
31          console.log("Please keep your promise .. fingers crossed")
32      })
33      .catch(function(err) {
34          console.error(err);
35          console.log("Ok try next time.")
36      })
```

MULTIPLE PROMISE

```
65  //Multiple Promise
66  Promise.all([pinkypromise, didhomework])
67      .then(function(data) {
68          console.table(data);
69      })
70      .catch(function(err) {
71          console.error(err)
72      })
```

ASYNC AWAIT

```
75  //Async Await
76  async function asyncawait()
77  {
78      try
79      {
80          var promise1 = await pinkypromise;
81          var promise2 = await didhomework;
82          var promise3 = await getupearly(promise1+"\n"+promise2);
83          console.log(promise3);
84
85      }
86      catch(e)
87      {
88          console.error(e);
89      }
90  }
91  asyncawait();
```

“ES6”

WOW!

ARROW FUNCTION

```
1  //ES5
2  function Bored(name)
3  {
4      return name+" is bored";
5  }
6
7  //ES6
8  const Bored =(name)=>name+" is bored";
9
10 Bored("Pakya");
```

No need to specify function keyword

Implicit return at last line

Brings clarity & code reduction

TEMPLATE LITERALS

```
1  //ES5
2  function Bored(name)
3  {
4      return name+" is bored";
5  }
6
7  //ES6
8  const Bored =(name)=>`${name} is bored`;
9
10 Bored("Pakya");
```

Use the tilde sign to enclose the string

No more + sign or “ for concatenation

All scope variables can be accessed

SPREAD OPERATOR

```
19  //Es6 Spread Operator
20  function length(...nums) {
21      console.log(nums.length);
22  }
23  length(3,4,5,6,7,8) // 6;
24
25
26  function total(x, y, z) {
27      console.log(x + y + z);
28  }
29  total(1,2,3);
30  total.apply(null, [1,2,3]);
31  total(...[1,2,3])
```

MANIPULATING OBJECTS

```
1  //ES5 Object copy
2  var obj1 = { a: 1, b: 2 }
3  var obj2 = { a: 2, c: 3, d: 4}
4  var obj3 = Object.assign(obj1, obj2)
```

```
6  //ES6 Object copy
7  const obj1 = { a: 1, b: 2 }
8  const obj2 = { a: 2, c: 3, d: 4}
9  const obj3 = {...obj1, ...obj2}
```

```
11 //ES5 Object Extracting
12 var obj1 = { a: 1, b: 2, c: 3, d: 4 }
13 var a = obj1.a
14 var b = obj1.b
15 var c = obj1.c
16 var d = obj1.d
```

```
18 //ES6 Object Extracting
19 const obj1 = { a: 1, b: 2, c: 3, d: 4 }
20 const {
21   a,
22   b,
23   c,
24   d
25 } = obj1
```

```
27 //ES5 Object Assignment
28 var a = 1
29 var b = 2
30 var c = 3
31 var d = 4
32 var obj1 = { a: a, b: b, c: c, d: d }
```

```
34 //ES6 Object Assignment
35 var a = 1
36 var b = 2
37 var c = 3
38 var d = 4
39 var obj1 = { a, b, c, d }
```

MODULES

```
1  //ES6 Export Module
2  var myModule = { x: 1, y: function(){ console.log('This is ES5') }}
3  module.exports = myModule;
4
5  //ES6 Export Module
6  const myModule = { x: 1, y: () => { console.log('This is ES5') }}
7  export default myModule;
8
9
10 //ES5 Module Import
11 var myModule = require('./myModule');
12
13 //ES6 Module Import
14 import myModule from './myModule';
```

“BACKBONEJS”

“Backbone is not a complete framework. It's a set of building blocks. It leaves much of the application design, architecture and scalability to the developer, including memory management, view management, and more.”
-Derick Bailey

BACKBONE.JS

WHAT IT IS?

Provides client side app structure

It contains Models to represent data

It contains Views to hook up Models to the DOM

Synchronises data to/from server

IT'S A MV* FRAMEWORK

MODEL,VIEW & *

MODEL

Application data and business rules

It also contains events which notifies any change in data

Views are updated using these event notifications

This gives you one source of truth, which is not the user interface.

VIEW

User interacts with View

It observes any data change event in model

It reads and edits Model

ROUTER

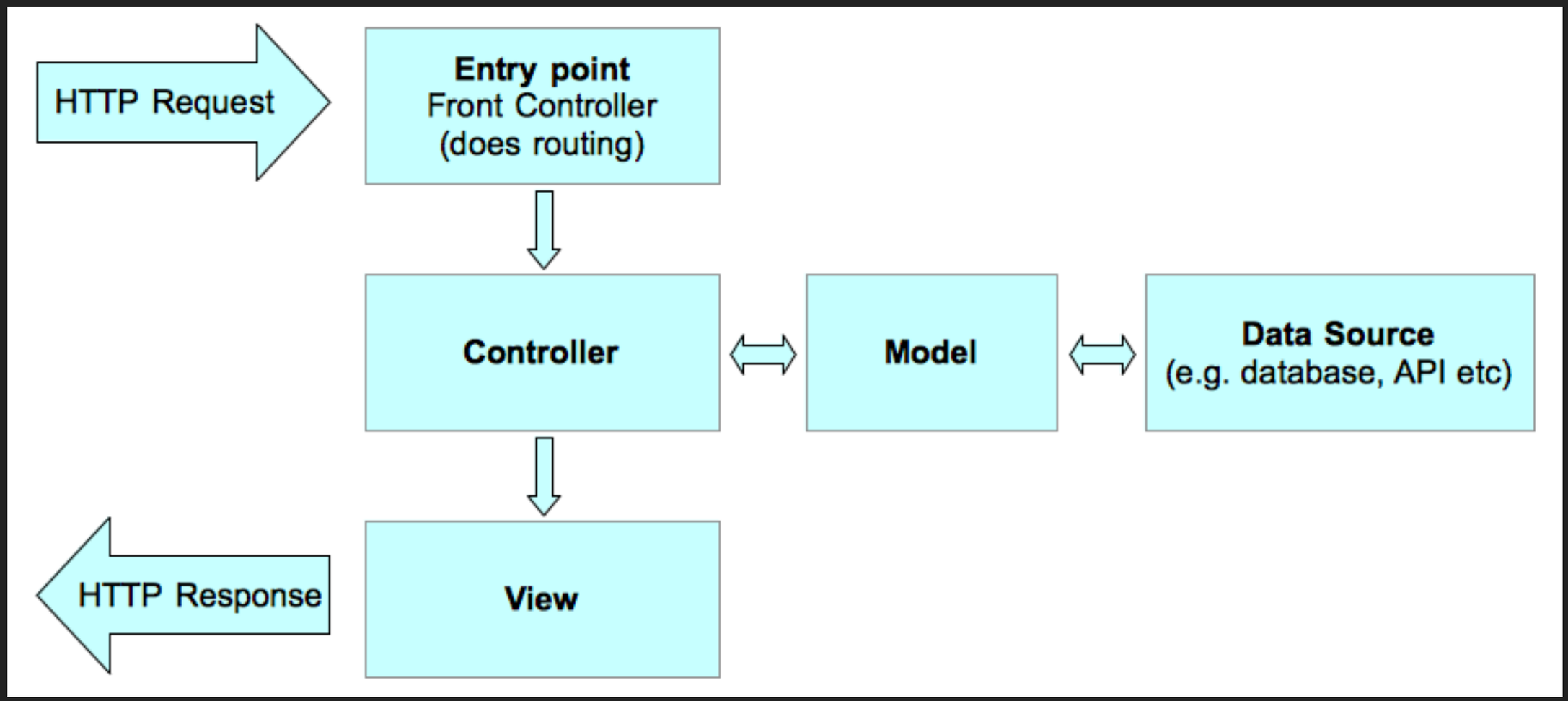
Everything after the hashtag is considered as router

Maps your URL to function

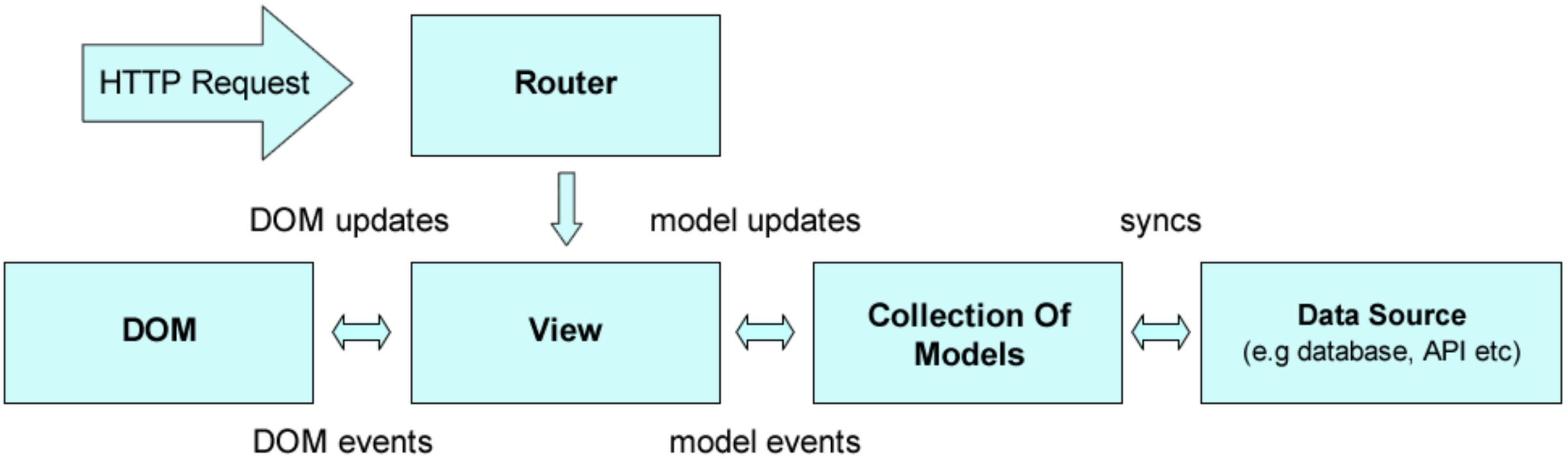
Manages application state like bookmarking a particular view


WHAT DOES IT ALL MEAN
I SO CONFUSE
Kevin Ball

HTTP REQUEST/RESPONSE LIFECYCLE FOR SERVER-SIDE MVC



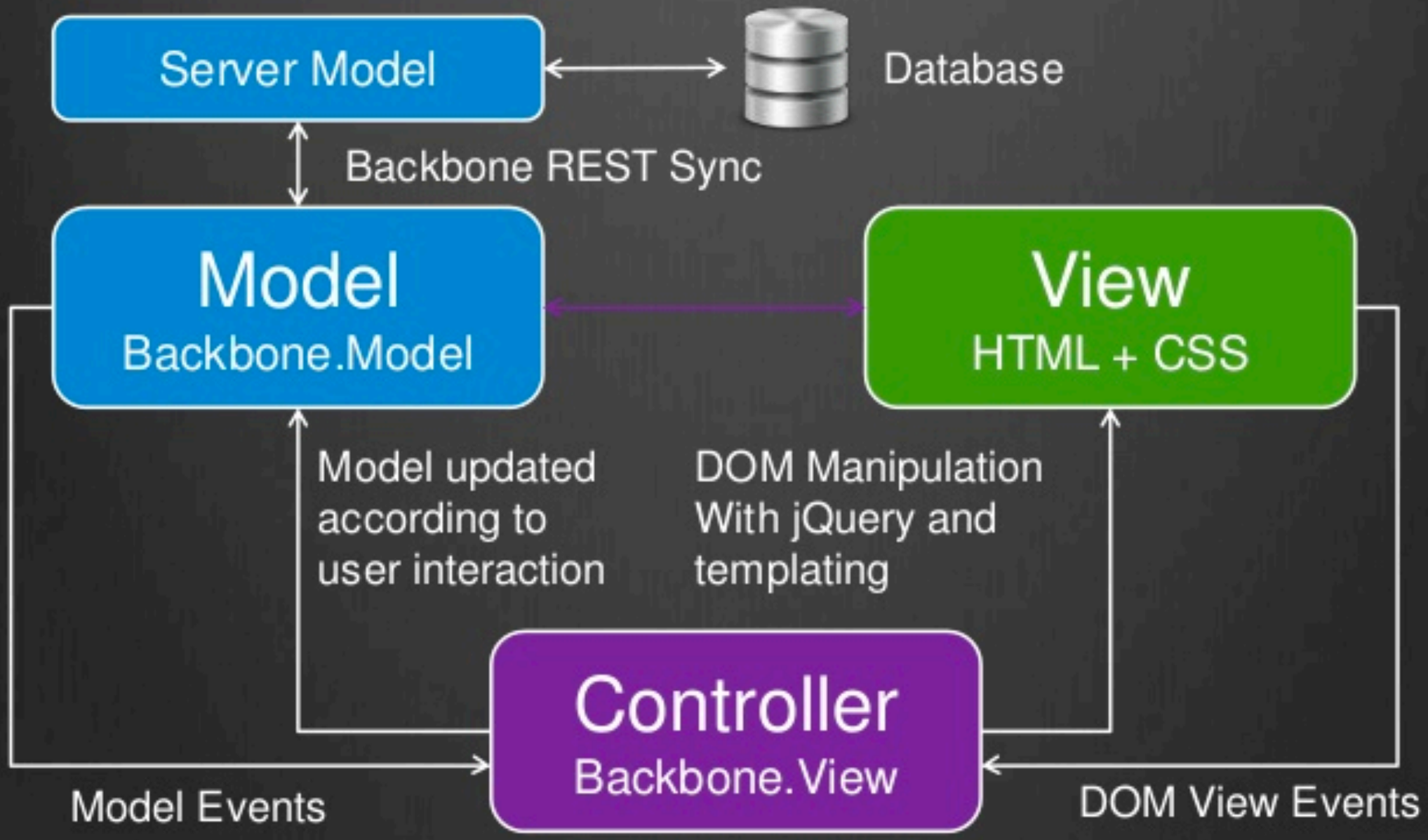
MVC AS IMPLEMENTED BY BACKBONEJS

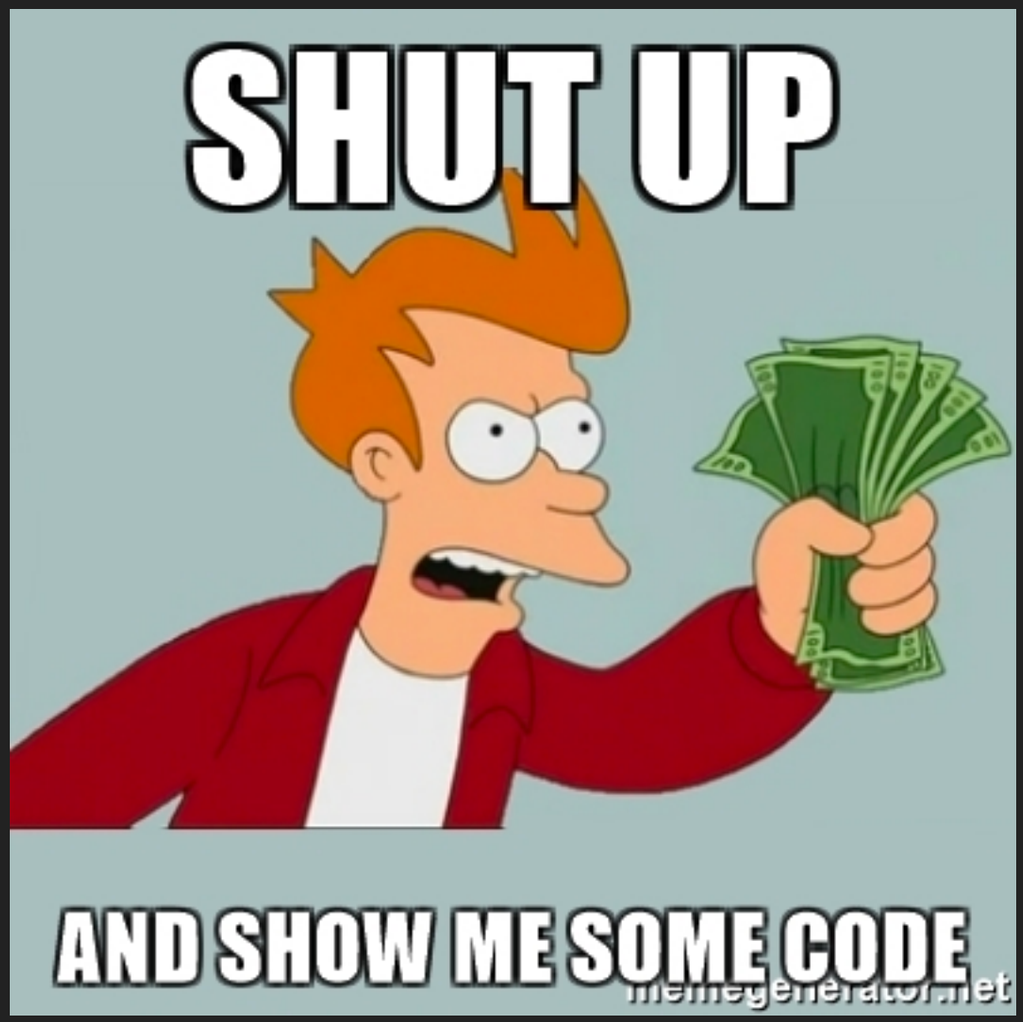


DEPENDENCIES

JQUERY & UNDERSCORE

Backbone MVC Architecture




SHUT UP
AND SHOW ME SOME CODE


THANKS!
quickmeme.com