# Advanced JavaScript

JavaScript? Again?
redbrick.dcu.ie/~edu/AdvancedJS.pdf

# Voy

Admin
Past Webmaster

Github: VoyTechnology

# Plan

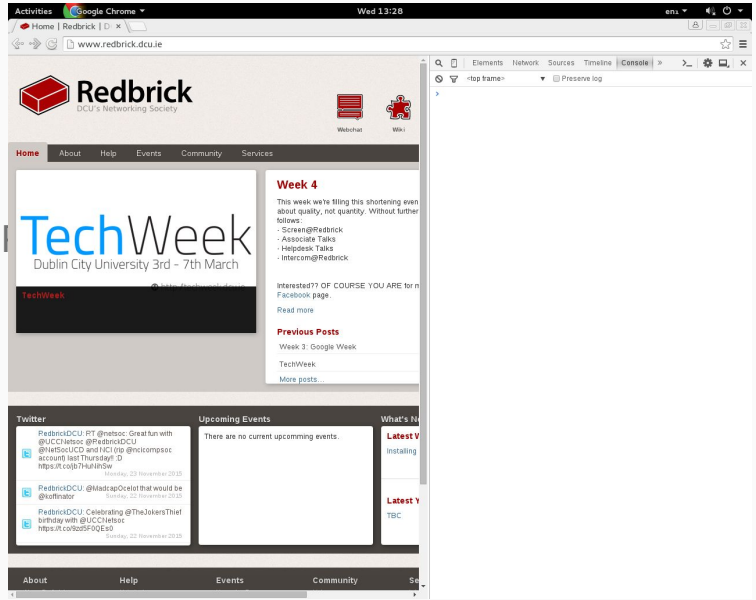
Today’s talk will include

- Revision of basics
- Objects
- Classes
- Prototyping
- Async
- Nice things
- Pizza?

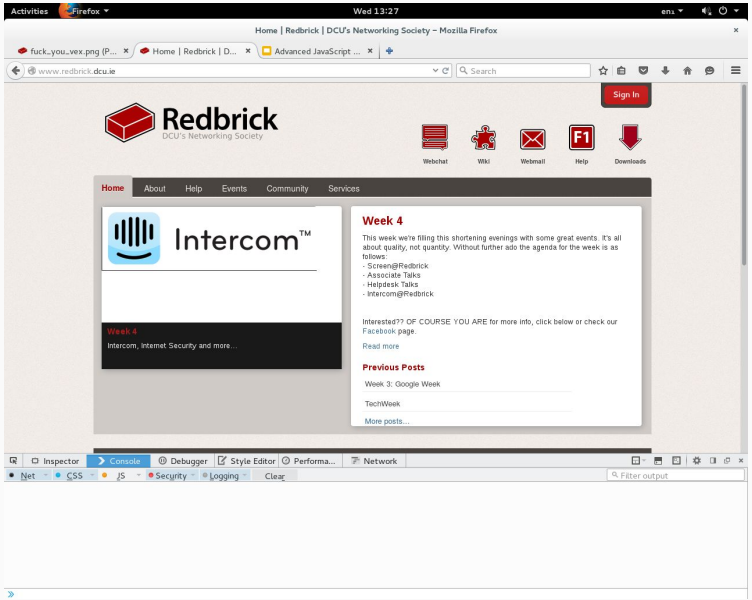
# Revision

# Dev Console

## Chrome




## Firefox



# Making JavaScript files

Make a new file, call it *main.js* or whatever you want

All code we will work on today will be saved there

Make a HTML file and paste this:

```
<!DOCTYPE html>
<html>
    <head><script src=”main.js”></script></head>
</html>
```

# Revision: Hello World

```
var hello = “Hello World”

function printWhatever(something){
        console.log(something);
}



printWhatever(hello);
```

# data types

number: -4, 0, 1, 0.34, 4.352e+12

string: ‘Hello World’

array: [1,2,3]

object: {hello: “world”}

function: hello()

boolean: true

undefined: null, undefined, “var x;”

# Functions

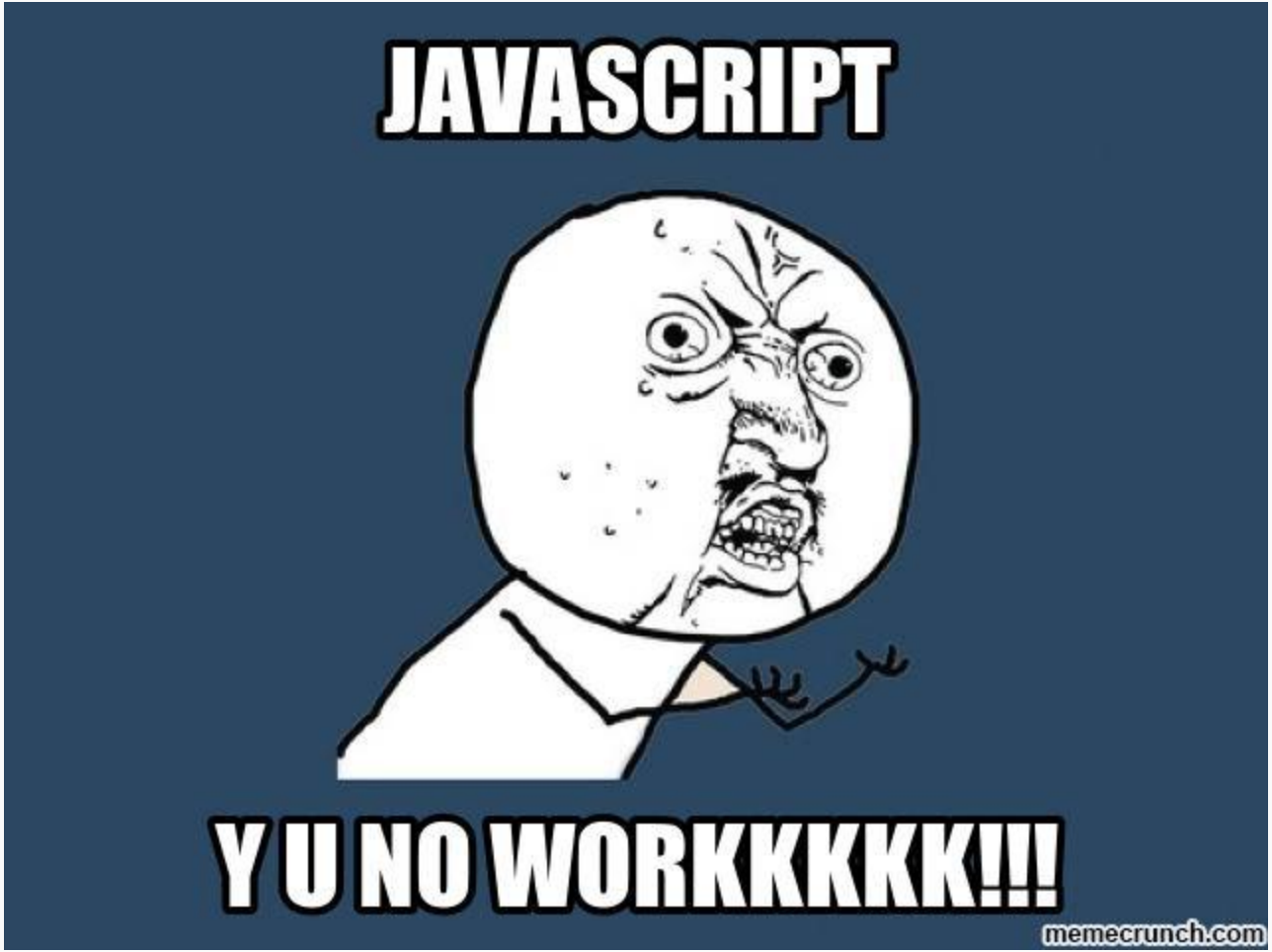
```
function myFunc(){}                    // Empty Function

function myFunc(){return true;}        // Function returning true

function myFunc(arg){ return arg;}     // Function taking a argument

var f = function(){}                   // Anonymous function assigned to f

(function(){console.log("hi");})()     // Anonymous function in closure
```


JAVASCRIPT
Y U NO WORKKKKKK!!!
memecrunch.com

# Class - Object - Prototype

# Objects: Basic Properties

Instance of *something*

Contain methods

Contain variables

Everything is public

# Objects

Everything is an object*

Empty Object: {}

Key=>Value: {hello: “world”}

Can store any data type:

- { thing:{} }
- { multiple: [1,2,3] }
- {
  name: “Wojtek”,
  interests: [“space”, “servers”]
  }

# *Everything is an object? What?

```
var array = [1,2,3,4]

array.push(5) // [1,2,3,4,5]
```

# Using Objects

```
var data = {
        code: 200,
        message: “OK”
}
console.log(data.code) // 200
data.message = “Not OK… :(”

console.log(data) // Object: { code: 200, message: “Not OK… :(” }
```

# Classes

They don’t really exists… yet*

But, you have this:

```
function Car(make){
        this.make = make;
}

var tesla = new Car(‘tesla’);

console.log(tesla) // Car {make: “tesla”}

console.log(tesla.make) // “tesla”
```

# *Future Classes (ECMAScript 6)

```
class Car {
	constructor(make) {
		this.make = make
	}
}
```

Coming to browsers near you in… whenever everybody implements it

# But we are not there yet

rzft.co/BLhZe

# Classes: Naive way

```
function Car(make){
	this.make = make;
	this.drive = function(){
		console.log(“I drive a” + make);
	}
}

myCar = new Car(“tesla”);
myCar.drive(); // I drive a tesla
```

# Prototyping

- Proper way of making a class
- faster
- more memory efficient

```
function Car(make){
        this.make = make;
}

Car.prototype.drive = function(){
        console.log(“I drive a ” + this.make);
}

Car.prototype.getMake = function(){
        return this.make;
}

Car.prototype.setMake = function(newMake){
        this.make = newMake;
}

var tesla = new Car(‘tesla’);
tesla.drive();
tesla.getMake();
zoe = tesla
zoe.setMake(‘zoe’);
```

# Async

# Async

What does it do?

Doesn’t block

Continues on with tasks

Revisits the task when it’s done

# Async

**ADVANTAGES**

- Non-blocking
- Concurrent

**DISADVANTAGES**

- Hard to understand
- Limited scope
- Nested functions

# BUT WHY?

# Consider this

**PHP** (yes, PHP)

```
$data = $_GET[‘data’];
$done = sendToDB($data);                    // This will take a while
echo $done;
echo “hi”
```

The above is blocking, meaning nothing else can be ran while sendToDB() is running

# JavaScript Async: Example 1

```
var data = req.get[‘data’];
sendToDB(data, function(){
        res.send(true);
});
res.send(“hi”);
```

# JavaScript Async: Example 2

```
// jQuery (frontend helper/framework)
var submit = $('#submit');
submit.click(function(){
        $.post(formData, function(status){
                statusBox.text(status);
        });
});
updateTimeBox();
```

# Writing Async Code

```
function sumOf(n, callback){
        var total = 0;
        for(var i = 1; i < n; i++){ total += i }
        callback(total);
}

sumOf(Math.pow(2,51), function(result){
        console.log(result);
});
```


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

# Nice things

# Time for fun

You are all experts now

Let’s cover:

- JSON
- Errors
- Error catching
- Type casting
- Writing JavaScript
- JS: Not only in browsers

# JSON

JavaScript Object Notation (JavaScript object as a string)

```
var data = {code: 200, message: “OK”};
var json = JSON.stringify(data);

console.log(json)

var newData = JSON.parse(json);
console.log(newData);
```

# JSON

Don’t

Do not stringify a object containing a method. It will fail

# Errors

undefined.

```
var a = [1,2];
console.log( a[1000000] );
```

# Error Catching

Gotta Catch ‘em All

```
var data = {}
var bad = “{code:200}”
try {
        data = JSON.parse(bad);
} catch( err ){
        console.log(“Bad json”);
}
```

# Error Throwing

Make your own errors

```
var baby = “baby”
try{
        if(baby != “mine”){
                throw baby;
        }
} catch(bby){
        console.log(bby + “ caught”);
}
```

# Type Casting

```
var raw = “Hello user 42”;
var arr = raw.split(“ ”);

var id = arr[2];
console.log(id); // “42”
```

---

```
var id = Number(arr[2]);
console.log(id) // 42
```

# Type Casting

| | |
|---|---|
| x.toString() | Converts x to string |
| String() | ^^ same as above |
| Number(x) | Converts x to number |
| Boolean(x) | Converts x to Boolean |

# JS: Not only in browser

| | |
|---|---|
| Servers | nodejs.org |
| Apps | phonegap.com |
| Drones | nodecopter.com |
| Desktop Apps | nwjs.io |


WROTE JAVASCRIPT
CODE WORKED
DIYLOL.COM

# Thank You