Functional and Object-Oriented Javascript

or

The Javascript Marty Doesn't Want You to Know

aka

How to make your Javascript less like this:

```
function init() {
	var items = document.getQuerySelectorAll(".thing");
	for(var i = 0; i < items.length; i++) {
		items[i].addEventListener("click",clickthing);
	}
}

function clickthing() {
	this.innerHTML = "clicked";
	this.style.color = "red";
}

window.onload = init;
```

And more like this:

```
$(function() {
	$(".thing").click(function() {
		$(this).text("clicked").css({"color":"red"});
	});
});
```

What?

Functional Javascript with anonymous methods and methods-as-variables.

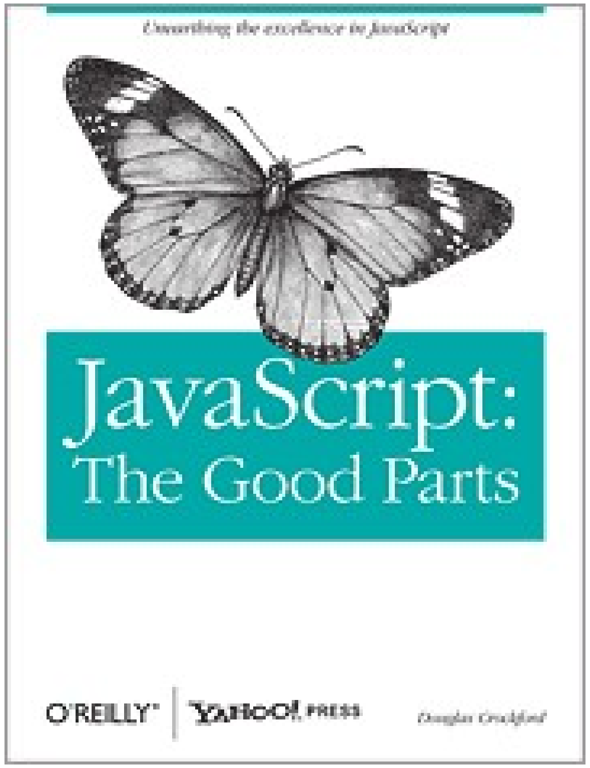
Objective Javascript with both Java-like and JSON notation.

Javascripts objects-as-hashes.

Why?

Most JS code out there won't look like Java code
(Will use at least some of the things listed above)

The big payoff: jQuery
(But jQuery won't make much sense without this!)



Functional Javascript

Functions are variables too!

```
function test() {
   console.log(“test!”);
}
```

is the same* as...

```
var test = function() {
   console.log(“test!”);
}
```

Functions are variables too!
And what can we do with variables?

*actually there are a few minor differences involving the order the code is loaded, but don't mind me

Functional Javascript

```
var test = function() {
   console.log("test");
};

function caller(fn) {
   fn();
}

caller(test);
```

What should this bit of code do?

Then - how do we get this work if test has a parameter?

Functional Javascript

A more realistic example:

```
function isEven(x) {
   return x%2==0;
}
```

Given this, write a method removeEvens(a) that takes an array (of numbers)
and returns a copy of the passed in array, removing all the even elements. Then, generalize it to odds, or every third, etc.

```
function removeEvens(a) {
   var newa = [];
   ...
   return newa;
}
```

Functional Javascript

A more realistic example:

```
function removef(a,f) {
    var newa = [];
    for(var i = 0; i < a.length; i++) {
        if (!f(a[i])) {
            newa.push(a[i]);
        }
    }
    return newa;
}

function isEven(i) {
    return i % 2 == 0;
}

removef([1,2,3,4,5],isEven); //removes all evens
```

What we just implemented here is filter, a basic operation in functional programming. (Javascript arrays have this by default, so we'll use that instead)

Functional Javascript

Array.filter

```
[1,2,3,4,5,6].filter(function(e){ return e%2==0});
//returns array of all evens in the array

[-1,2,-3,4,-5,6].filter(function(e){ return e > 0});
//returns array of all the positives in the array

[“abc”,”defg”,”hi”].filter(function(e){
   return e.length == 2
});
//returns array of all strings of length 2
```

A few more to try:
Array of strings, remove all empty strings
Array of numbers, remove all contained in another array.

Functional Javascript

Array.map

Another common functional operator is called map.
Think of this one as mapping between one array to another, given a mapping function.

```
[1,2,3,4].map(function(i){ return i+1; })
//returns [2,3,4,5]

["abc","bbbc","d"].map(function(i){ return i.length; })
//returns [3,4,1]
```

A few more to try:
Array of numbers, map to their char (String.fromCharCode(i))
Array of strings, filter out the empty's and map to first letter

Functional Javascript

Array.reduce

Another common functional operator is called reduce.
Think of it as combining all the elements of an array into one item, given an accumulator function. The syntax is different:

```
a.reduce(function(prev,cur){
   return running_sum;
}, initial_value);
```

```
[1,2,3,4,5].reduce(function(prev,cur) {
   return prev+cur;
}, 0);
//returns sum of all elements in the array
```

```
["abc","bbbc","d"].reduce(function(prev,cur) {
   return prev+cur;
}, "");
//what do you think this does?
```

Functional Javascript

Array.forEach

This one is pretty self explanatory.
(It's the foreach loop in javascript!)
It doesn't return anything.

```
[1,2,3].forEach(function(i) {
   console.log(i);
});
```

Note about all these functions (except foreach):
They don't modify the array. Instead, they return a new array.

Why Functional Javascript?

It's shorter.

Functions have **closure**, loops don't.
(You can use this to write shorter and better* code)

**opinion*

```
<script>
window.onload = function() {
      var btns = Array.prototype.slice.call(document.querySelectorAll(".btn"));
      btns.forEach(function(btn){
            var btnval = btn.innerHTML;
            btn.addEventListener("click",function() {
                  console.log(btnval);
            });

            btn.innerHTML = "CLICK";
      });

      /*var btns = document.querySelectorAll(".btn");
      for(var i = 0; i < btns.length; i++) {
            var btn = btns[i];
            var btnval = btn.innerHTML;
            btn.addEventListener("click",function() {
                  console.log(btnval);
            });

            btn.innerHTML = "CLICK";
      }*/
}
</script>
```

The for loop and the array.forEach will behave differently for this html page.

```
<button class="btn">5</button>
<button class="btn">4</button>
<button class="btn">7</button>
```

Object-Oriented Javascript

How do you make an object in java?

```
Object o = new Object();
```

How do you make an object in java***script***?

The same way!

```
var o = new Object();
```

Object-Oriented Javascript

Did you know you could do this?

```
var o = new Object();
o.lolwut = 5;
console.log(o.lolwut);
```

What do you think this'll print?

Lesson:
Javascript objects can hold anything you store in them.

Object-Oriented Javascript

Did you know you could do this?

```
var o = new Object();
o["lolwut"] = 5;
console.log(o.lolwut);
```

What do you think this'll print?

Lesson:
o["name"] and o.name notation are almost* equivalent

**The almost comes from o["string with space"], can't do this the other way.*

Object-Oriented Javascript

Another notation

```
var o = {lol:5, lolwut:"lolwut", lel:[1,2,3]};
```

Is equivalent to...

```
var o = new Object();
o.lol = 5;
o["lolwut"] = "lolwut";
o.lel = [1,2,3];
```

The former is commonly called "***JSON***" notation.

Object-Oriented Javascript

Methods and Fields

This is object-oriented programming, so where are my methods and fields?

```
var o = new Object();
o.lolwut = 5;
o.test = function() {
   console.log("test");
};
```

And now you can go...

```
o.lol();
```

But how do we access fields?

Object-Oriented Javascript

Methods and Fields

This is object-oriented programming, so where are my methods and fields?

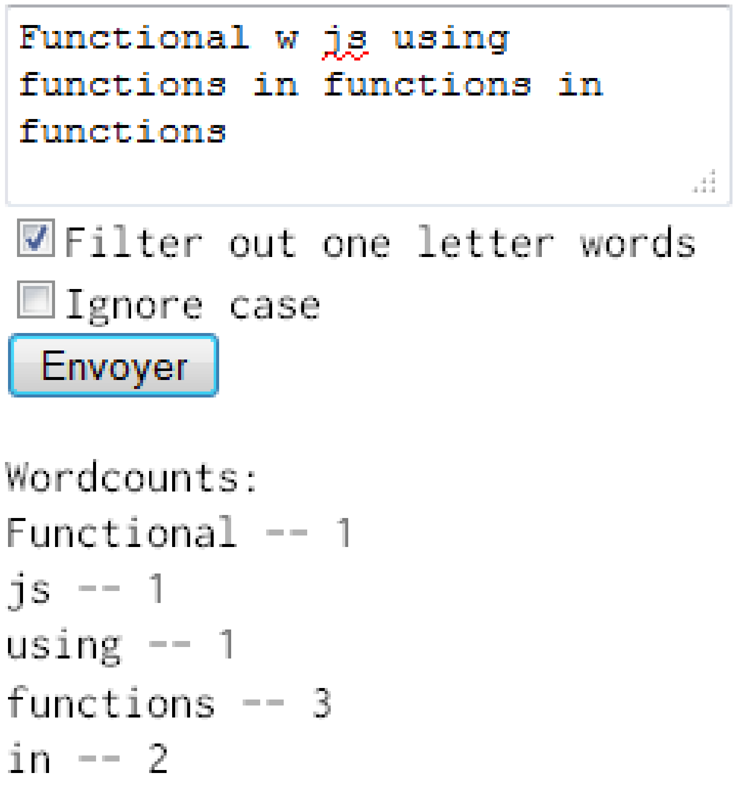
```
var o = new Object();
o.lolwut = 5;
o.test = function() {
   console.log(this.lolwut);
};
```

And now you can go...

```
o.lol();
```

Example

Count and display the occurrences of every word of a textfield.



```
Functional w js using
functions in functions in
functions
```

[✓] Filter out one letter words

[ ] Ignore case

```
Wordcounts:
Functional -- 1
js -- 1
using -- 1
functions -- 3
in -- 2
```

Hint: create a new object as your dictionary. Then, loop through all the words (split the text on space). Increment the dictionary at the word by 1.