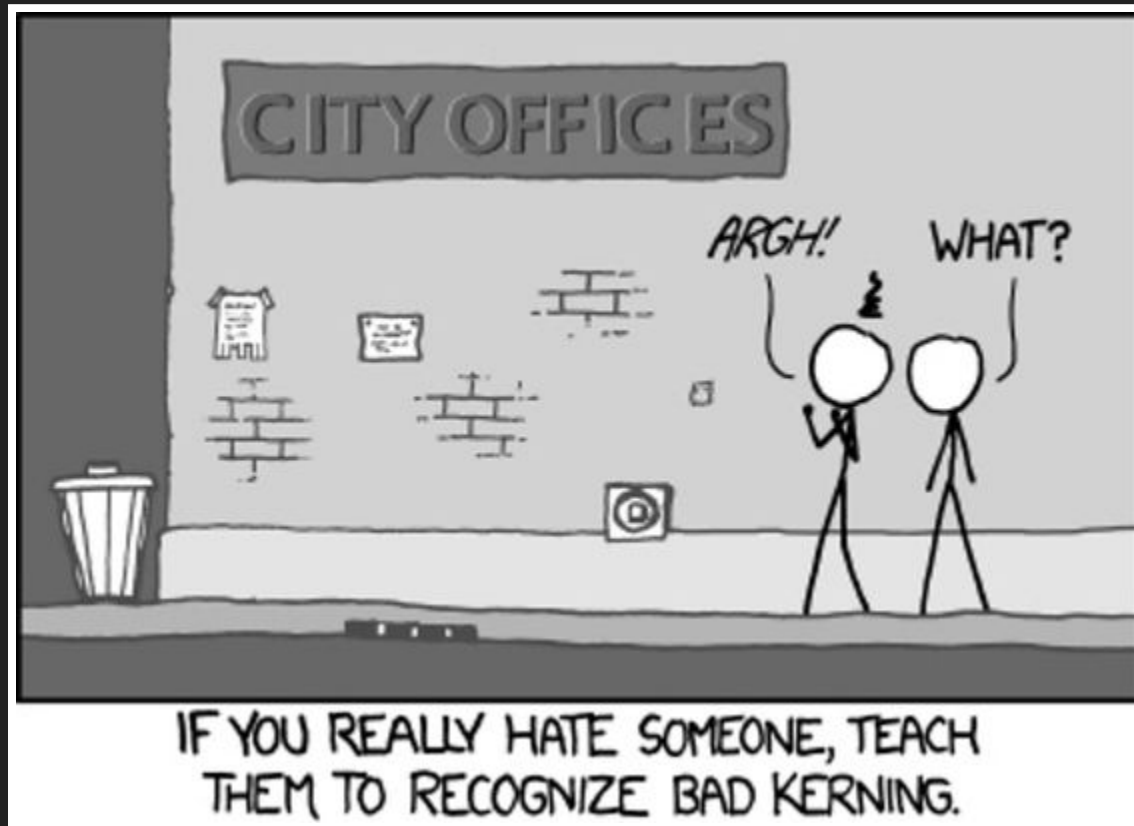


# WHAT IS FUNCTIONAL PROGRAMMING?

@KRISAJENKINS

# FIRST, DO SOME HARM



# EVERY FUNCTION YOU WRITE

...has two sets of inputs & outputs.

# THE FAMILIAR PAIR

```
public int add(int x, int y) {  
    return x + y;  
}
```

# THE HIDDEN PAIR

---

```
public boolean processMessages(int n) {  
    int count = InboxManager.process(n);  
  
    return (count >= 1);  
}
```

---

# HIDDEN INPUTS AND OUTPUTS

Are called 'side-effects'.

# HIDDEN INPUTS AND OUTPUTS

Are called 'side-causes' and 'side-effects'.

# A SIDE-EFFECT SPOTTER'S GUIDE



# SPOT THE SIDE-EFFECT

```
todoList.remaining = function() {  
  var count = 0;  
  angular.forEach(todoList.todos, function(todo) {  
    count += todo.done ? 0 : 1;  
  });  
  return count;  
};
```

# SPOT THE SIDE-CAUSE

---

```
todoList.addTodo = function() {  
    todoList.todos.push({text:todoList.todoText, done:false});  
    todoList.todoText = '';  
};
```

# SIDE-EFFECTS ARE THE COMPLEXITY ICEBERG

---

```
public boolean processMessage(Queue queue) {...}
```

# **SIDE-EFFECTS DESTROY TESTING**

There are no black boxes any more.

# **SIDE-EFFECTS DESTROY COMPONENTS**

Nothing exists in isolation.

# SIDE-EFFECTS DESTROY COMPOSITION

Things cannot be plugged together.

# SO, WHAT IS FUNCTIONAL PROGRAMMING?

# FUNCTIONAL PROGRAMMING IS

- Eliminating side-effects where you can.
- Controlling them where you can't.
- So that every component describes a relationship between its inputs and outputs.



# FUNCTIONAL PROGRAMMING IS

Eliminating & controlling side-effects.

# FUNCTIONAL PROGRAMMING ENABLES

- Easier reasoning
- Easier testing.
- Easier reuse.
- Easier concurrency.

**WAIT, YOU DIDN'T MENTION...**

**MAP, REDUCE, FOLD ...**

# HIGHER ORDER FUNCTIONS

# MONADS

**LOTS OF OTHER THINGS**

**ONE MORE TIME**



# FP

Functional Programming is about eliminating side-effects  
where you can, controlling them where you can't.

# THANK YOU

Twitter & more: @krisajenkins

Blog: <http://blog.jenkster.com/>