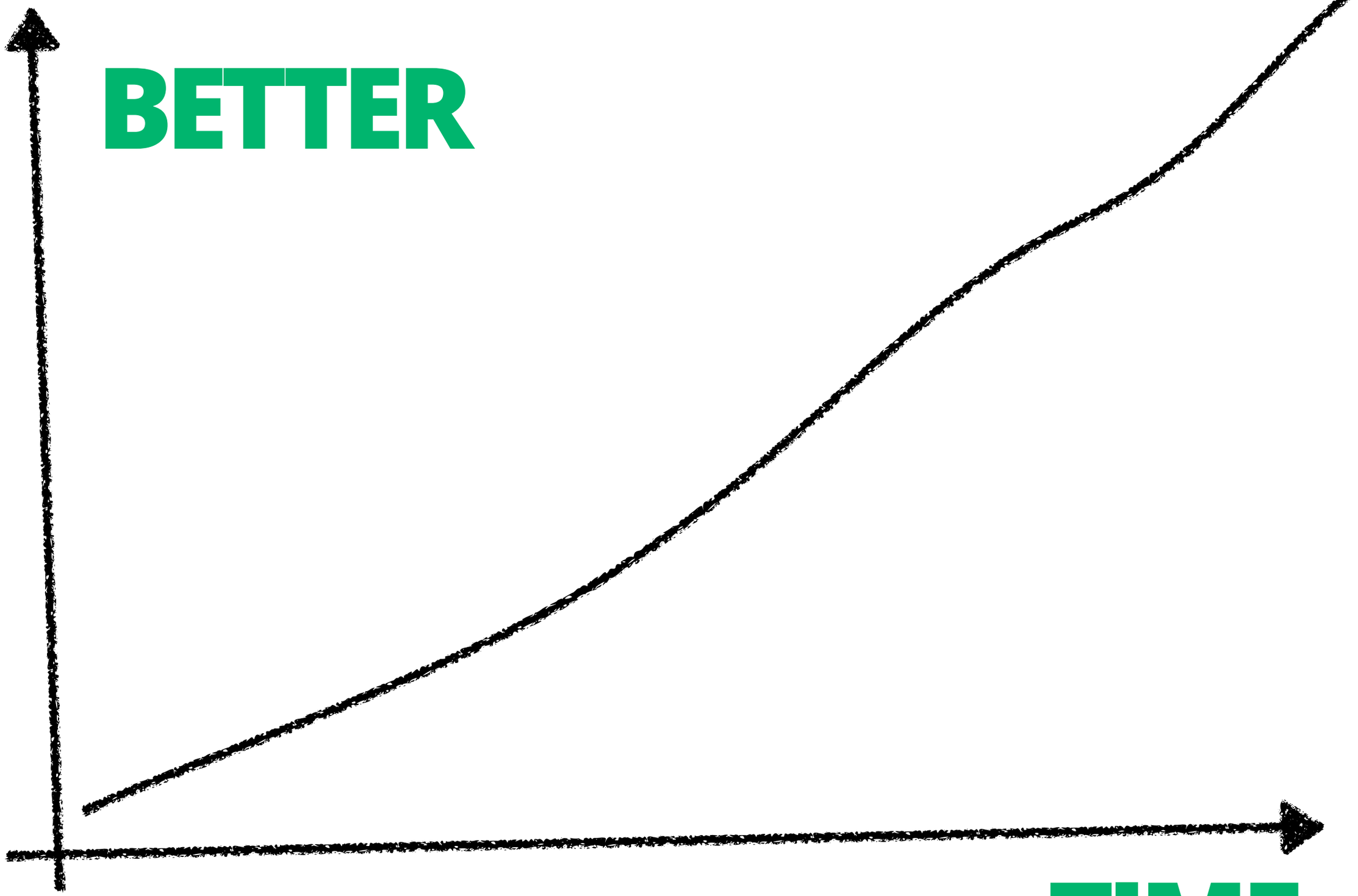
ThoughtWorks®

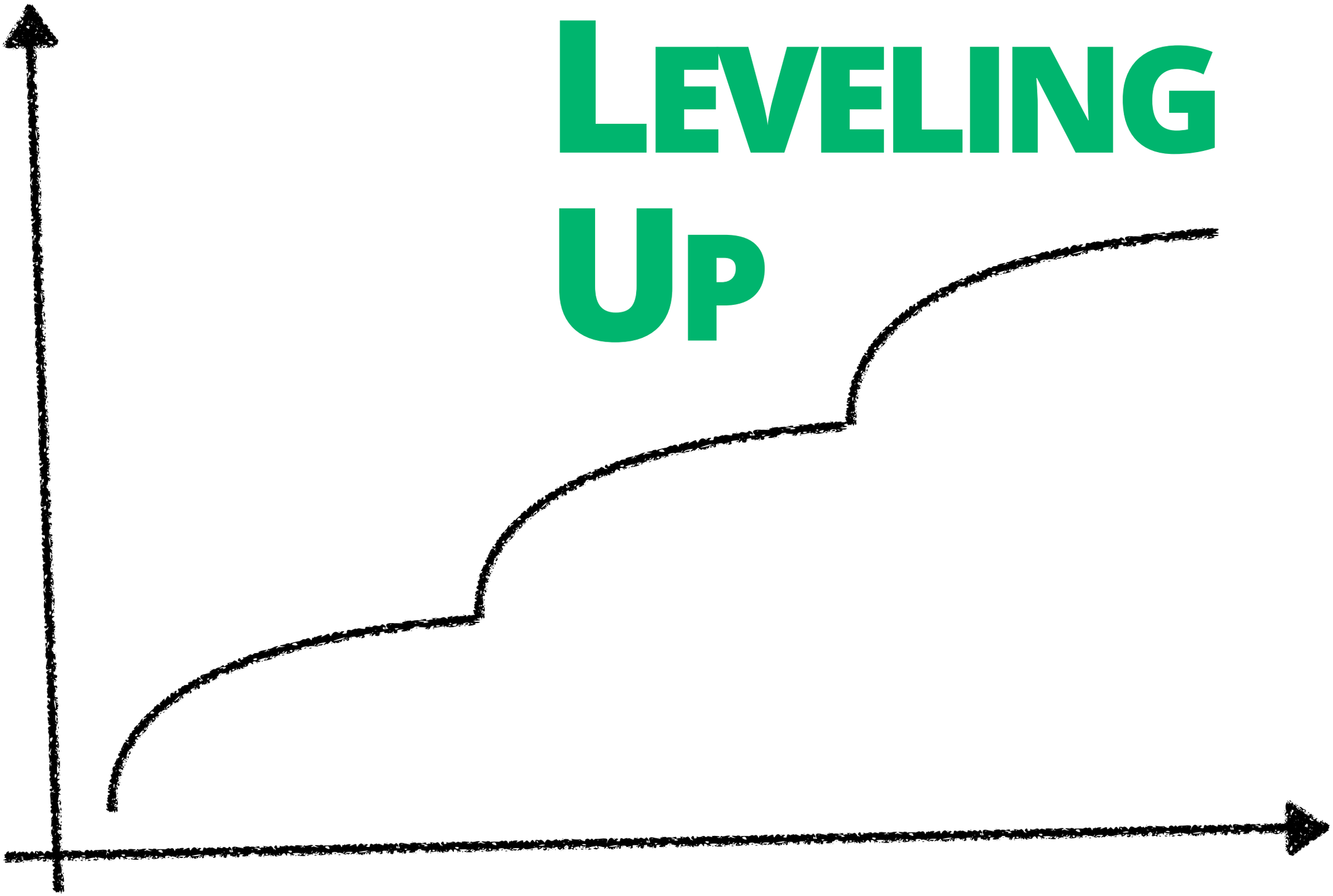
DIFFERENT.JS

Pete Hodgson
@ph1 | phodgson@thoughtworks.com



@ph1

Leveling Up



@ph1

Pete Hodgson
@ph1, Glorified Typist

ThoughtWorks®

315 Montgomery Street, 16th Floor | San Francisco, California | 94104
T: +1 415 869 3000 C: +1 707 206 8897
E: phodgson@thoughtworks.com

this

@ph1


this!!!!
@ph1

this!!!!

```
function myClass(message) {
  this.message = message;
  var self = this;
  $('button').click(function(){
    self.onClick();
  });
}
```

@ph1

this

@ph1

~~this~~

@ph1

Object Oriented Programming

state & behavior,

traveling together through time

@ph1

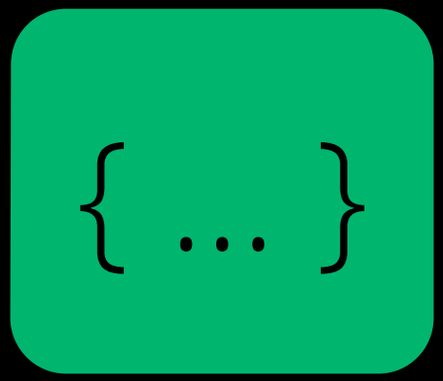
A PERSON

```
george = new Person()

george.eatCake() // om nom nom nom

george.stillHungry() // returns true or false
```

```
var george = new Person();
```

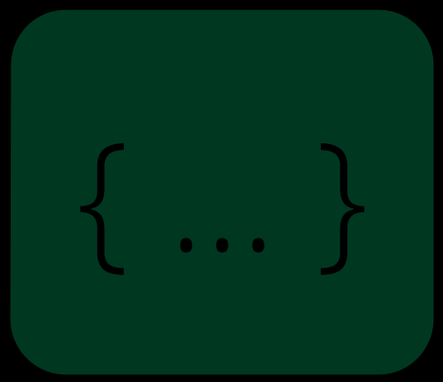
@ph1

```
var george = new Person();
```

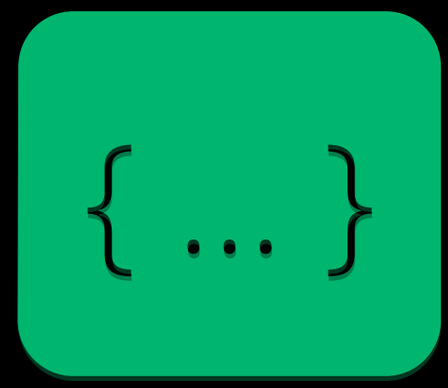


@ph1

```
var george = new Person();
```




```
george.eatCake();
```



@ph1

```
var george = new Person();
```

{ ... }

```
george.eatCake();
```

{ ... }

{ ... }

```
if( george.stillHungry() ){
  // blah
}
```

@ph1

```
var george = new Person();
```

{ … }

```
george.eatCake();
```

{ … }

{ … }

```
if( george.stillHungry() ){
  // blah
}
```

@ph1

WITH PROTOTYPES / THIS

```
function Person(){
  this.cakesEaten = 0;
}

Person.prototype.eatCake = function(){
  this.cakesEaten += 1;
};

Person.prototype.stillHungry = function(){
  return this.cakesEaten < 3;
};
```

WITH PROTOTYPES / THIS

```
function Person(){
  this.cakesEaten = 0;
}

Person.prototype.eatCake = function(){
  this.cakesEaten += 1;
};

Person.prototype.stillHungry = function(){
  return this.cakesEaten < 3;
};
```

@ph1

WITH CLOSURES

```
function createPerson(){
  var cakesEaten = 0;
  return {
   eatCake: function(){
      cakesEaten += 1;
    },
    stillHungry: function(){
      return cakesEaten < 3;
    }
  };
}
```

```
function
}
Person
};
Person
};
```

```
george = new Person()

cakes = ["coffee","cheese","bundt","angel"]
cakes.forEach(george.eatCake)
```

@ph1

```
george = new Person()

cakes = ["coffee","cheese","bundt","angel"]
cakes.forEach(george.eatCake)

george.stillHungry() // => true
```

@ph1

```
george = new Person()

cakes = ["coffee","cheese","bundt","angel"]
cakes.forEach(george.eatCake)

george.stillHungry() // => true

george.cakesEaten // => 0
```

@ph1

```
george = createPerson()

cakes = ["coffee","cheese","bundt","angel"]
cakes.forEach(george.eatCake)

george.stillHungry() // => false
```

@ph1

~~this~~

@ph1


@ph1

level up!

@ph1

LEVEL UP

- ☐ no return values *(yes, really!)*
- ☐ only 1 parameter
- ☐ all functions less than 5 lines long
- ☐ test-first everything
- ☐ no tests
- ☐ immutable objects
- ☐ no “methods” (functions on objects)
- ☐ functional reactive

Pete Hodgson
@ph1
phodgson@thoughtworks.com

thanks!

slides: http://bitly.com/differentjs