



# WRITING BEAUTIFUL JAVASCRIPT TESTS

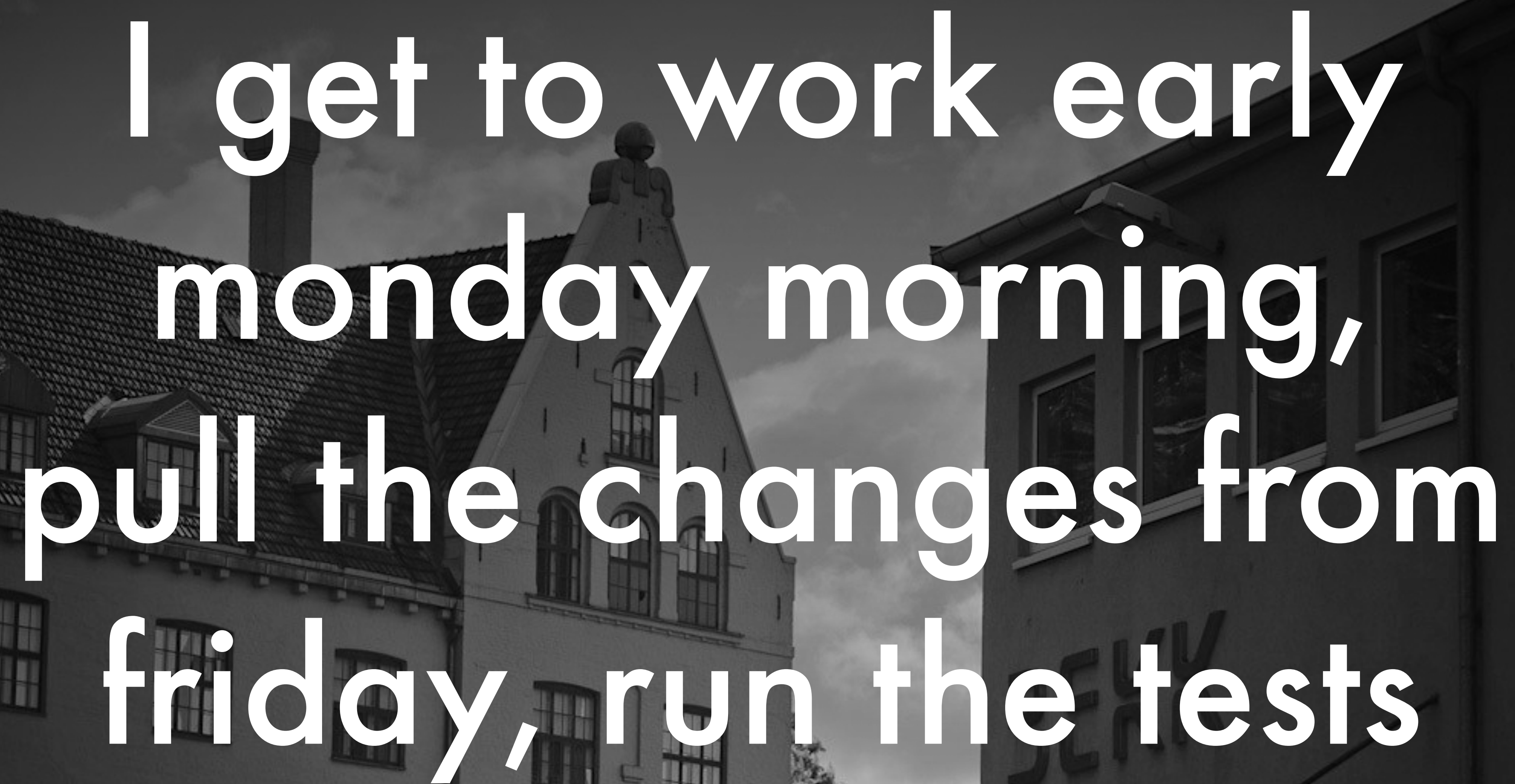
@KIMJOAR





#exampletime






I get to work early  
monday morning,  
pull the changes from  
friday, run the tests



Jasmine Spec Runner v2.0.2 x  
localhost:8000/SpecRunner.html

 **Jasmine** 2.0.2 finished in 0.037s

.....x.....

**55 specs, 1 failure** raise exceptions

[Spec List](#) | [Failures](#)

**Order with life insurance with child insurance validates**

Expected false to be true.

```
Error: Expected false to be true.  
  at stack (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1304:17)  
  at buildExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1281:14)  
  at Spec.Env.expectationResultFactory (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:473:18)  
  at Spec.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:271:34)  
  at Expectation.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:431:21)  
  at Expectation.toBe (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1220:12)  
  at Object.<anonymous> (http://localhost:8000/spec/OrderSpec.js:200:41)  
  at attemptSync (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1620:12)  
  at QueueRunner.run (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1608:9)  
  at QueueRunner.execute (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1595:10)
```



Jasmine Spec Runner v2.0.2 x  
localhost:8000/SpecRunner.html

Jasmine 2.0.2 finished in 0.037s

55 specs, 1 failure raise exceptions

Spec List | Failures

One or more with life instances and with child instances violates

Expected value to be true

Error: Expected false to be true

at stack (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1304:17)  
at buildExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1281:14)  
at Spec.Env.expectationResultFactory (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:473:18)  
at Spec.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:271:34)  
at Expectation.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:431:21)  
at Expectation.toBe (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1220:12)  
at Object.<anonymous> (http://localhost:8000/spec/01\_rSpec.js:100:41)  
at attemptSync (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1100:12)  
at QueueRunner.run (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1608:23)  
at QueueRunner.run (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1595:10)

# WHY WHO PUSHED A FAILING TEST!?







, however, is green





, however, is green



## EXAMPLE OF WHAT A TEST MIGHT LOOK LIKE

```
describe('cart', function() {
  beforeEach(function() {
    // called before every test
    // (often called setup)
  });

  it('calculates correct total price', function() {
    var cart = new Cart();

    cart.addItem({ name: 'Sleeping bag', price: 299, quantity: 2 });
    cart.addItem({ name: 'Tent', price: 499, quantity: 1 });

    expect(cart.totalPrice()).toEqual(1097);
  });

  afterEach(function() {
    // called after every test
    // (often called teardown)
  });
});
```



## EXAMPLE OF WHAT A TEST MIGHT LOOK LIKE

```
describe('cart', function() {
  var cart;


  beforeEach(function() {
    cart = new Cart();
  });

  describe('with two orders', function() {
    beforeEach(function() {
      cart.addItem({ name: 'Sleeping bag', price: 299, quantity: 2 });
      cart.addItem({ name: 'Tent', price: 499, quantity: 1 });
    });

    it('calculates correct total price', function() {
      expect(cart.totalPrice()).toEqual(1097);
    });
  });
});
```



Jasmine Spec Runner v2.0.2 x  
localhost:8000/SpecRunner.html

 **Jasmine** 2.0.2 finished in 0.037s

.....x.....

**55 specs, 1 failure** raise exceptions

[Spec List](#) | [Failures](#)

**Order with life insurance with child insurance validates**

Expected false to be true.

```
Error: Expected false to be true.  
  at stack (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1304:17)  
  at buildExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1281:14)  
  at Spec.Env.expectationResultFactory (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:473:18)  
  at Spec.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:271:34)  
  at Expectation.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:431:21)  
  at Expectation.toBe (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1220:12)  
  at Object.<anonymous> (http://localhost:8000/spec/OrderSpec.js:200:41)  
  at attemptSync (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1620:12)  
  at QueueRunner.run (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1608:9)  
  at QueueRunner.execute (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1595:10)
```

Jasmine Spec Runner v2.0.2 x  
localhost:8000/SpecRunner.html

Jasmine 2.0.2 finished in 0.037s

55 specs, 1 failure raise exceptions

Spec List | Failures

**Order with life insurance with child insurance validates**

Expected false to be true.

```
Error: Expected false to be true.  
  at stack (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1304:17)  
  at buildExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1281:14)  
  at Spec.Env.expectationResultFactory (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:473:18)  
  at Spec.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:271:34)  
  at Expectation.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:431:21)  
  at Expectation.toBe (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1220:12)  
  at Object.<anonymous> (http://localhost:8000/spec/OrderSpec.js:200:41)  
  at attemptSync (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1620:12)  
  at QueueRunner.run (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1608:9)  
  at QueueRunner.execute (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1595:10)
```



Jasmine Spec Runner v2.0.2 x  
localhost:8000/SpecRunner.html

Jasmine 2.0.2 finished in 0.037s

55 specs, 1 failure raise exceptions

Spec List | Failures

**Order with life insurance with child insurance validates**

Expected false to be true.

```
Error: Expected false to be true.  
  at stack (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1304:17)  
  at buildExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1281:14)  
  at Spec.Env.expectationResultFactory (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:473:18)  
  at Spec.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:271:34)  
  at Expectation.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:431:21)  
  at Expectation.toBe (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1220:12)  
  at Object.<anonymous> (http://localhost:8000/spec/OrderSpec.js:200:41)  
  at attemptSync (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1620:12)  
  at QueueRunner.run (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1608:9)  
  at QueueRunner.execute (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1595:10)
```

55 specs, 1 failure raise exceptions

Spec List | Failures

**Order with life insurance with child insurance validates**

Expected false to be true.

```
Error: Expected false to be true.  
  at stack (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1304:17)  
  at buildExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1281:14)  
  at Spec.Env.expectationResultFactory (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:473:18)  
  at Spec.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:271:34)  
  at Expectation.addExpectationResult (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:431:21)  
  at Expectation.toBe (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1220:12)  
  at Object.<anonymous> (http://localhost:8000/spec/OrderSpec.js:200:41)  
  at attemptSync (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1620:12)  
  at QueueRunner.run (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1608:9)  
  at QueueRunner.execute (http://localhost:8000/lib/jasmine-2.0.2/jasmine.js:1595:10)
```



```
it('validates', function() {  
    expect(order.isValid()).toBe(true);  
});
```

**WHY**


**DID**

**IT**

**FAIL?**

# WHY

```
beforeEach(function() {  
  var ci = new Product({  
    type: 'CI',  
    price: 105  
  });  
  cart.addItem(ci);  
});
```

 CHILD INSURANCE

# DID

# IT

**DOES THIS CHANGE THE ORDER?**

# FAIL?



```
beforeEach(function() {  
  beneficiary1.amount = 1500000;  
  beneficiary2.amount = 500000;  
  
  var li = new Product({  
    type: 'LI',  
    price: 399,  
    amount: beneficiary1.amount  
      + beneficiary2.amount  
  });  
  
  cart.addItem(li);  
  order.setPolicyHolder(policyHolder2);  
  order.addBeneficiary(beneficiary1);  
  order.addBeneficiary(beneficiary2);  
});
```

**WHY**

**DID**

**IT**

**FAIL?**

```
beforeEach(function() {
  policyHolder1 = new PolicyHolder({
    name: "Ola Nordmann",
    ssn: 07099451429,
    telephoneNumber: "+4795732501",
    email: "me@example.org",
    address: {
      street: "Toftes gate 17",
      postCode: 0556,
      postalArea: "Oslo"
    }
  });
  policyHolder2 = new PolicyHolder({
    name: "Testy Testeson",
    ssn: 17099926629,
    telephoneNumber: "+4743032501",
    email: "me2@example.org",
    address: {
      street: "Ravnkollbakken 3",
      postCode: 0970,
      postalArea: "Oslo"
    }
  });

  beneficiary1 = {
    name: "Testy2",
    ssn: "04037335466"
  }
  beneficiary2 = {
    name: "Testy3",
    ssn: "18049938744"
  }
  beneficiary3 = {
    name: "Testy4",
    ssn: "21050682312"
  }

  cart = new Cart();

  order = new Order({
    cart: cart,
    policyHolder: policyHolder1,
    withdrawalDay: 15
  });
});
```

I  
HAVE  
NO  
IDEA



```
beforeEach(function() {  
    // ...  
  
    policyHolder2 = new PolicyHolder({  
        name: "Testy Testeson",  
        ssn: 17099926629,  
        telephoneNumber: "+4743032501",  
        email: "me2@example.org",  
        address: {  
            street: "Ravnkollbakken 3",  
            postCode: 0970,  
            postalArea: "Oslo"  
        }  
    });  
  
    // ...  
});
```

**THE PROBLEM?**

**He got too  
old to have  
child insurance**

A black and white photograph of a ship's mast and rigging against a cloudy sky with a bright sun flare. The text is overlaid in white, bold, sans-serif font.

You will,  
at some point,  
**HATE** your tests





We need to  
learn more  
about  
writing tests





WHY DO  
WE TEST?



**because  
testing in  
production sucks**



But seriously,

**IT DEPENDS**



HI! I'M @KIMJOAR



I'M A CONSULTANT AT BEKK IN OSLO, NORWAY  
I WORK IN 5-10 PERSON TEAMS  
FOR 3 MONTHS TO SEVERAL YEARS  
FOR LARGE COMPANIES  
ON INTERACTIVE AND COMPLEX APPS



A blurred black and white photograph of a modern office interior. In the foreground, a person is walking from left to right, their figure out of focus. In the background, several other people are visible, some standing and some walking, also blurred. The office has a clean, minimalist aesthetic with white walls and a polished floor. On the left wall, there are large, stylized letters that appear to be 'U', 'E', and 'K'. A long white cabinet or desk is visible in the mid-ground. The ceiling has several pendant lights hanging down. The overall atmosphere is one of a busy, active workspace.

*TODAY:*  
PART PHILOSOPHY  
PART TECHNIQUES  
SHARING EXPERIENCES  
BOTH GOOD AND BAD





WHY DO

I TEST?





REGRESSIONS  
COMMUNICATION  
DESIGN

# REGRESSIONS





A lighthouse with a red roof sits on a concrete pier in the middle of a dark blue body of water. The sky is filled with large, white, fluffy clouds. In the background, there are low hills and a small town. The text "we're building more complex and interactive apps" is overlaid in white, bold, sans-serif font across the center of the image.

we're building  
more complex and  
interactive apps



A lighthouse with a red roof sits on a concrete pier in the middle of a dark, choppy sea. The sky is filled with large, white, fluffy clouds. In the background, a low-lying coastline with some buildings and trees is visible under a hazy sky.

often in  
larger teams



A lighthouse with a red roof and white body stands on a small, rocky island in the middle of a dark, choppy sea. The sky is filled with large, white, fluffy clouds. The text "less afraid of changing code" is overlaid in white, sans-serif font across the center of the image.

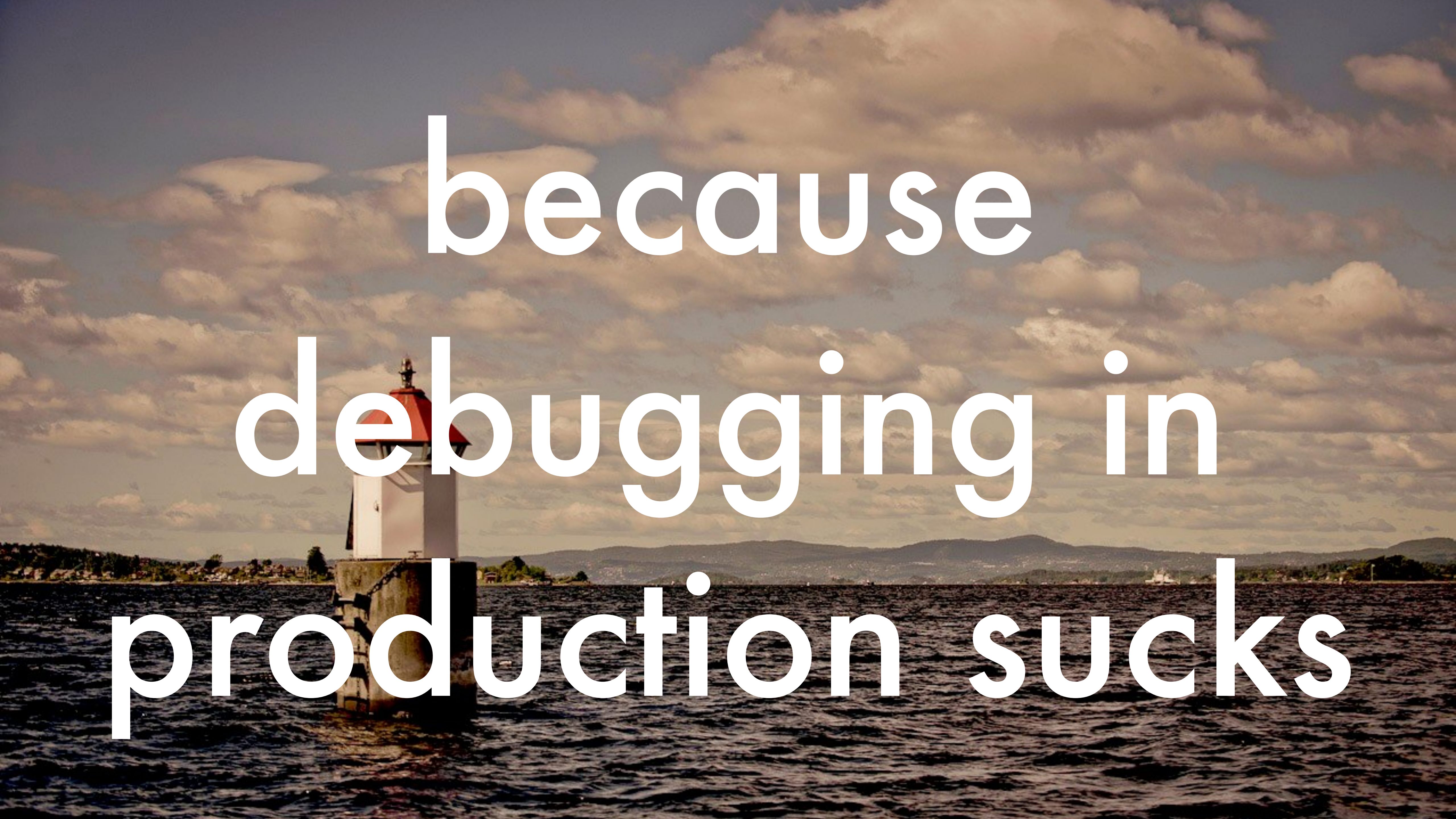
less afraid of  
changing code



A lighthouse with a red top and white body stands on a concrete pier in the middle of a dark, choppy sea. The sky is filled with large, textured clouds in shades of grey and brown, suggesting a dramatic or overcast day. In the background, a low-lying coastline with some buildings and trees is visible under the horizon.

less afraid of  
changing other  
people's code





because  
debugging in  
production sucks



fix a bug once







# COMMUNICATION



why did we  
do it like this?





what should  
it do?







DESIGN





immediate  
feedback





keep the  
code simple





REGRESSIONS  
COMMUNICATION  
DESIGN



so . . . what's  
the problem?





it's easy to write  
really bad tests



TRY TO

HOW I WRITE

MY TESTS



# FOCUS ON THE READER





**TIME SPENT READING CODE > TIME SPENT WRITING CODE**



**IN A TEAM YOU SPEND A LOT OF  
TIME READING OTHERS' CODE**



NO SETUP





A black and white photograph of a dark, choppy sea. In the foreground, a white zebra crossing is visible on a dark surface, possibly a pier or a boat deck. The water is dark and textured with small waves. The overall mood is somber and contemplative.

**DRY**

ISN'T ALWAYS THE ANSWER



The background of the image is a dark, textured surface of water with small ripples. In the lower foreground, there is a zebra crossing with white diagonal stripes on a dark pavement.

**DRYING UP LINES**

**vs**

**DRYING UP CONCEPTS**



"create tiny universes with  
minimal conceptual  
overhead"

–Jay Fields





CREATION

HELPER





```
it('is valid when policy holder is young enough', function() {  
    var order = createOrder({  
        cart: createCart([{ type: 'CI' }]),  
        policyHolder: createPolicyHolder({ age: 14 })  
    });  
  
    expect(order.isValid()).toBe(true);  
});
```



```
it('is not valid when policy holder is too old', function() {  
    var order = createOrder({  
        cart: createCart([{ type: 'CI' }]),  
        policyHolder: createPolicyHolder({ age: 15 })  
    });  
  
    expect(order.isValid()).toBe(false);  
});
```



A scenic view of a lighthouse on a rocky island. The lighthouse is white with a red band and a green roof. In the background, there is a dense forest of green trees. In the foreground, there is a body of water with some boats and buildings along the shore. The sky is a mix of light and dark clouds.

CREATION

HELPER

WITH GOOD DEFAULTS



REVEAL

INTENT





# WHAT ARE WE TESTING HERE?

```
it('validates', function() {  
    expect(order.isValid()).toBe(true);  
});
```



A scenic view of a lighthouse on a rocky island. The lighthouse is white with a red band and a black top. The island is covered in green trees. In the background, there are more buildings and a body of water. The sky is a mix of light and dark clouds.

REVEAL

INTENT

FOCUS ON WHY, NOT HOW



DETERMINISTIC





```
beforeEach(function() {
  policyHolder1 = new PolicyHolder({
    name: "Ola Nordmann",
    ssn: 07099451429,
    telephoneNumber: "+4795732501",
    email: "me@example.org",
    address: {
      street: "Toftes gate 17",
      postCode: 0556,
      postalArea: "Oslo"
    }
  });
  policyHolder2 = new PolicyHolder({
    name: "Testy Testeson",
    ssn: 17099926629,
    telephoneNumber: "+4743032501",
    email: "me2@example.org",
    address: {
      street: "Ravnkollbakken 3",
      postCode: 0970,
      postalArea: "Oslo"
    }
  });

  beneficiary1 = {
    name: "Testy2",
    ssn: "04037335466"
  }
  beneficiary2 = {
    name: "Testy3",
    ssn: "18049938744"
  }
  beneficiary3 = {
    name: "Testy4",
    ssn: "21050682312"
  }

  cart = new Cart();

  order = new Order({
    cart: cart,
    policyHolder: policyHolder1,
    withdrawalDay: 15
  });
});
```

# REMEMEMBER

# THIS ONE?



# NO CONDITIONALS

NO FOR, IF OR WHILE



Midnight

12:01 am

1:00 am

11:00 am

Noon

12:01 pm

1:00 pm

11:59 pm





```
it("handles dates correctly", function() {
  for (var mins = 0; mins < 1440; mins++) {
    var d = new Date(2014, 1, 1, 0, 0, mins, 0)
    var event = new Event({ date: d });

    if (d.getHours() == 0 && d.getMinutes() == 0) {
      expect(event.date).toEqual("Midnight");
    }
    else if (d.getHours() == 12 && d.getMinutes() == 0) {
      expect(event.date).toEqual("Noon");
    }
    else {
      expect(event.date).toEqual(moment(d).format("h:mm a"));
    }
  }
});
```



```
it("handles dates correctly", function() {
  for (var mins = 0; mins < 1440; mins++) {
    var d = new Date(2014, 1, 1, 0, 0, mins, 0)
    var event = new Event({ date: d });

    if (d.getHours() == 0 && d.getMinutes() == 0) {
      expect(event.date).toEqual("Midnight");
    }
    else if (d.getHours() == 12 && d.getMinutes() == 0) {
      expect(event.date).toEqual("Noon");
    }
    else {
      expect(event.date).toEqual(moment(d).format("h:mm a"));
    }
  }
});
```

**CAN YOU SPOT THE BUG?**



```
it("handles dates correctly", function() {  
  for (var mins = 0; mins < 1440; mins++) {  
    var d = new Date(2014, 1, 1, 0, 0, mins, 0)  
    var event = new Event({ date: d });  
  
    if (d.getHours() == 0 && d.getMinutes() == 0) {  
      expect(event.date).toEqual("Midnight");  
    }  
    else if (d.getHours() == 12 && d.getMinutes() == 0) {  
      expect(event.date).toEqual("Noon");  
    }  
    else {  
      expect(event.date).toEqual(moment(d).format("h:mm a"));  
    }  
  }  
});
```

This is seconds,  
not minutes



**CAN YOU SPOT THE BUG?**



```
it("handles noon", function() {
  var event = new Event({
    date: createDate({ hours: 12, mins: 0 })
  });

  expect(event.date).toEqual("Noon");
});

it("handles midnight", function() {
  var event = new Event({
    date: createDate({ hours: 0, mins: 0 })
  });

  expect(event.date).toEqual('Midnight');
});
```





SEE IT FAIL

YOU DON'T WANT TO FAIL AT FAILING



QuickTime Player File Edit View Window Help

Jasmine Spec Runner v2.0. x

localhost:8000/SpecRunner.html

Jasmine 2.0.2 finished in 0.003s

1 spec, 0 failures raise exceptions

Player  
increments count when song is finished

```
8 describe("Player", function() {
7
6   it("increments count when song is finished", function() {
5     var player = new Player();
4
3     expect(player.countDone).toBe(0);
2
1     player.play('song name', function() {
0       expect(player.countDone).toBe(1);
1     });
2   });
3
4 });
```

PlayerSpec.js 9,43 All

"PlayerSpec.js" 13L, 291C written

THIS TEST WILL  
NEVER FAIL



The image shows a side-by-side view of a web browser and a code editor. The browser window on the left displays the Jasmine Spec Runner v2.0.2 interface. It shows a single spec for the 'Player' object: 'increments count when song is finished'. The status bar indicates '1 spec, 0 failures raise exceptions'. The code editor on the right shows the source code for 'PlayerSpec.js'. The code defines a 'describe' block for 'Player' with an 'it' block for the test. The test code is as follows:

```
8 describe("Player", function() {
7
6   it("increments count when song is finished", function() {
5     var player = new Player();
4
3     expect(player.countDone).toBe(0);
2
1     player.play('song name', function() {
0       expect(player.countDone).toBe(3);
1     });
2   });
3
4 });
```

A red box highlights the `expect(player.countDone).toBe(3);` line in the code editor. The text 'THIS TEST WILL NEVER FAIL' is overlaid in large white letters on the bottom right of the code editor.

PlayerSpec.js [+] 9,43 All



The image shows a browser window on the left displaying the Jasmine test runner results. It reports '1 spec, 1 failure' and shows the failure details for the spec 'Player increments count when song is finished'. The error message is 'Expected 0 to be 3.' with a stack trace pointing to the test code.

The browser window on the right shows the source code for 'PlayerSpec.js'. The code is as follows:

```
9 describe("Player", function() {
8
7   it("increments count when song is finished", function(done) {
6     var player = new Player();
5
4     expect(player.countDone).toBe(0);
3
2     player.play('song name', function() {
1       expect(player.countDone).toBe(3);
0       done();
1     });
2   });
3
4 });
```

A white arrow originates from the `done();` line (line 0) and points to the error message 'Expected 0 to be 3.' in the browser window.

# ASYNC TESTS NEED TO TELL WHEN THEY ARE DONE

PlayerSpec.js 10,19 All  
"PlayerSpec.js" 14L, 315C written



Remember, the most  
important thing your  
tests do is

**F**  
**A**  
**I**  
**L**



# BEWARE OF ASYNC







# ARRANGE ACT ASSERT

```
it('calculates total price', function() {  
  // ARRANGE  
  var cart = new Cart();  
  
  // ACT  
  cart.addItem(...);  
  cart.addItem(...);  
  
  // ASSERT  
  expect(cart.totalPrice()).toEqual(1097);  
});
```



**and perhaps most importantly ...**



Tests **SHOULD** to  
be stupidly  
simple

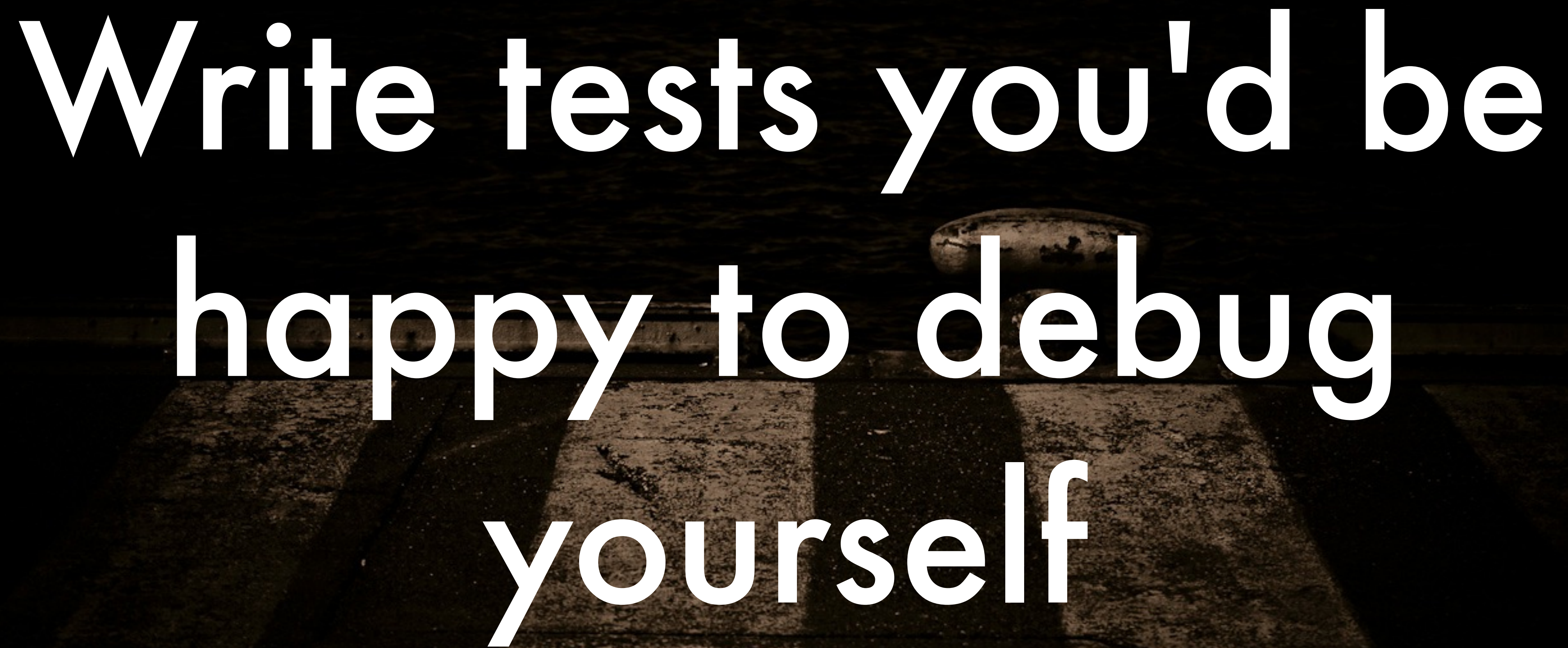




What happens when  
we don't understand  
our tests?







Write tests you'd be  
happy to debug  
yourself



# MAINTAINING YOUR TESTS

A person with a backpack is seen from behind, standing on a grassy hillside. They are looking out over a vast coastal landscape. In the foreground, there is a lush green field. The middle ground features a body of water, likely the ocean, with several rocky islands and peninsulas. The background shows rugged, dark mountains under a bright blue sky with scattered white clouds. The overall scene is serene and scenic, suggesting a remote or natural environment.



# POOR QUALITY TESTS

CAN SLOW DEVELOPMENT  
TO A CRAWL








FEEL THE PAIN

IF IT DOESN'T FEEL GOOD, FIX IT!



A black and white photograph of a row of houses with a cloudy sky and a bird in flight. The houses are in the foreground, and the sky is filled with clouds. A single bird is flying in the upper center of the frame. The text 'REFRACTOR YOUR TESTS' is overlaid in large, white, sans-serif font across the middle of the image.

# REFRACTOR YOUR TESTS



3

.

.

1







150000:50000

LINES OF TESTS

LINES OF APP CODE



***"... test the areas that you  
are most worried about  
going wrong"***

**- Martin Fowler**





WHAT IS THE COST OF TESTING?  
WHAT IS THE VALUE OF TESTING?





LESS

DOGMA-DRIVEN

TESTING, PLEASE



A black and white photograph of a row of houses with a cloudy sky and a bird in flight. The text is overlaid on the image.

**DON'T ACCEPT  
INCOMPREHENSIBLE  
TESTS**



A black and white photograph of a row of houses with a bird flying in the sky above them. The houses are in the foreground, and the sky is filled with clouds. A bird is visible in the upper left portion of the sky. A street light is visible in the center of the image, positioned between the two lines of text.

KEEP 'EM

SIMPLE



TESTS CAN  
BE AWESOME





HOW CAN WE  
*INCREASE* THE  
VALUE?







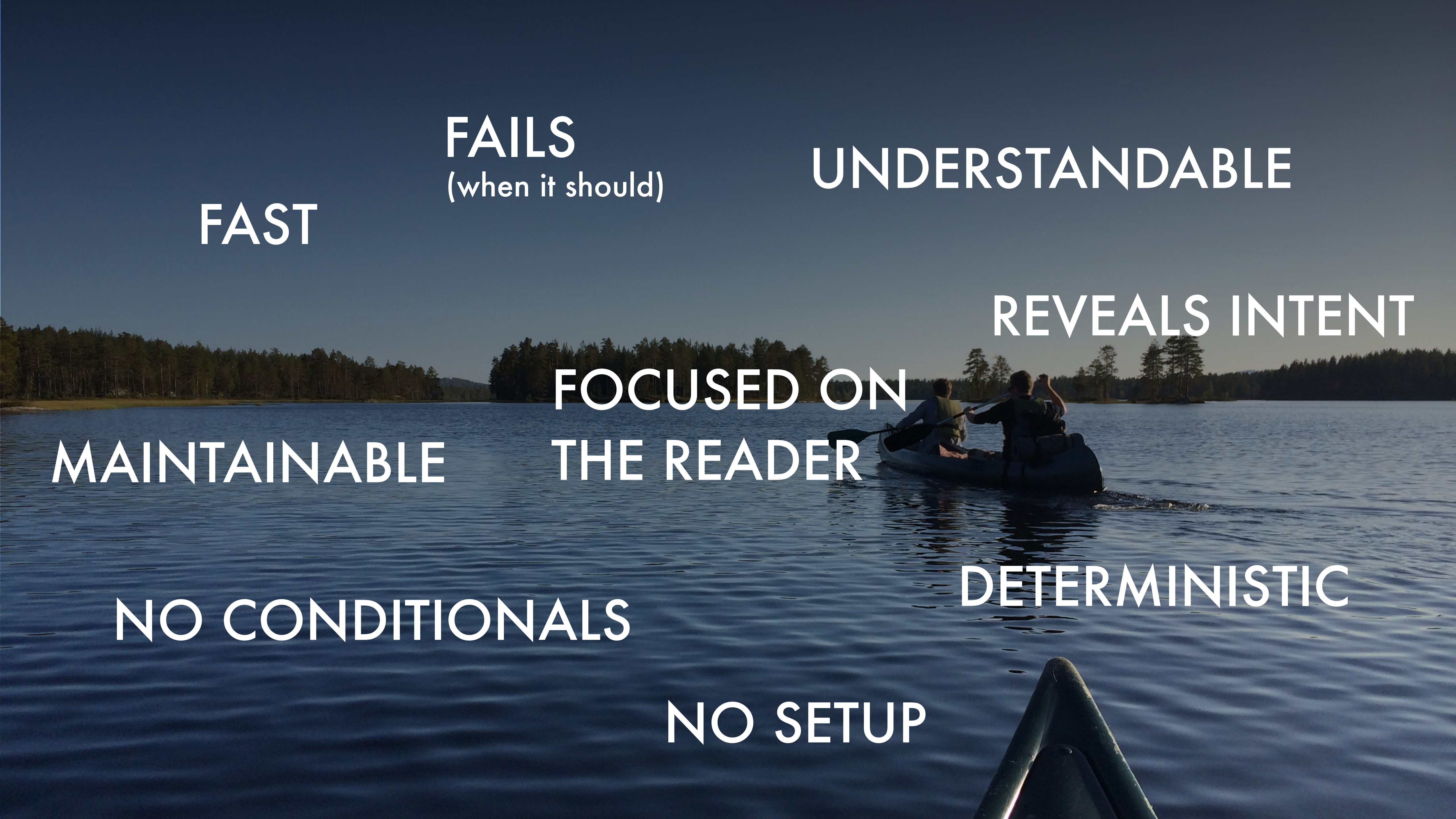
SO - WHAT IS A  
BEAUTIFUL TEST?



A person is kayaking on a calm lake. The kayaker is in the foreground, seen from behind, wearing a dark jacket and a life vest. The water is a deep blue with gentle ripples. In the background, a dense forest of tall, thin trees lines the shore under a clear, light blue sky. The text "I've got no definitive answer" is overlaid in white, sans-serif font across the middle of the image.

I've got no definitive answer





**FAST**

**FAILS**  
(when it should)

**UNDERSTANDABLE**

**REVEALS INTENT**

**FOCUSED ON  
THE READER**

**MAINTAINABLE**

**DETERMINISTIC**

**NO CONDITIONALS**

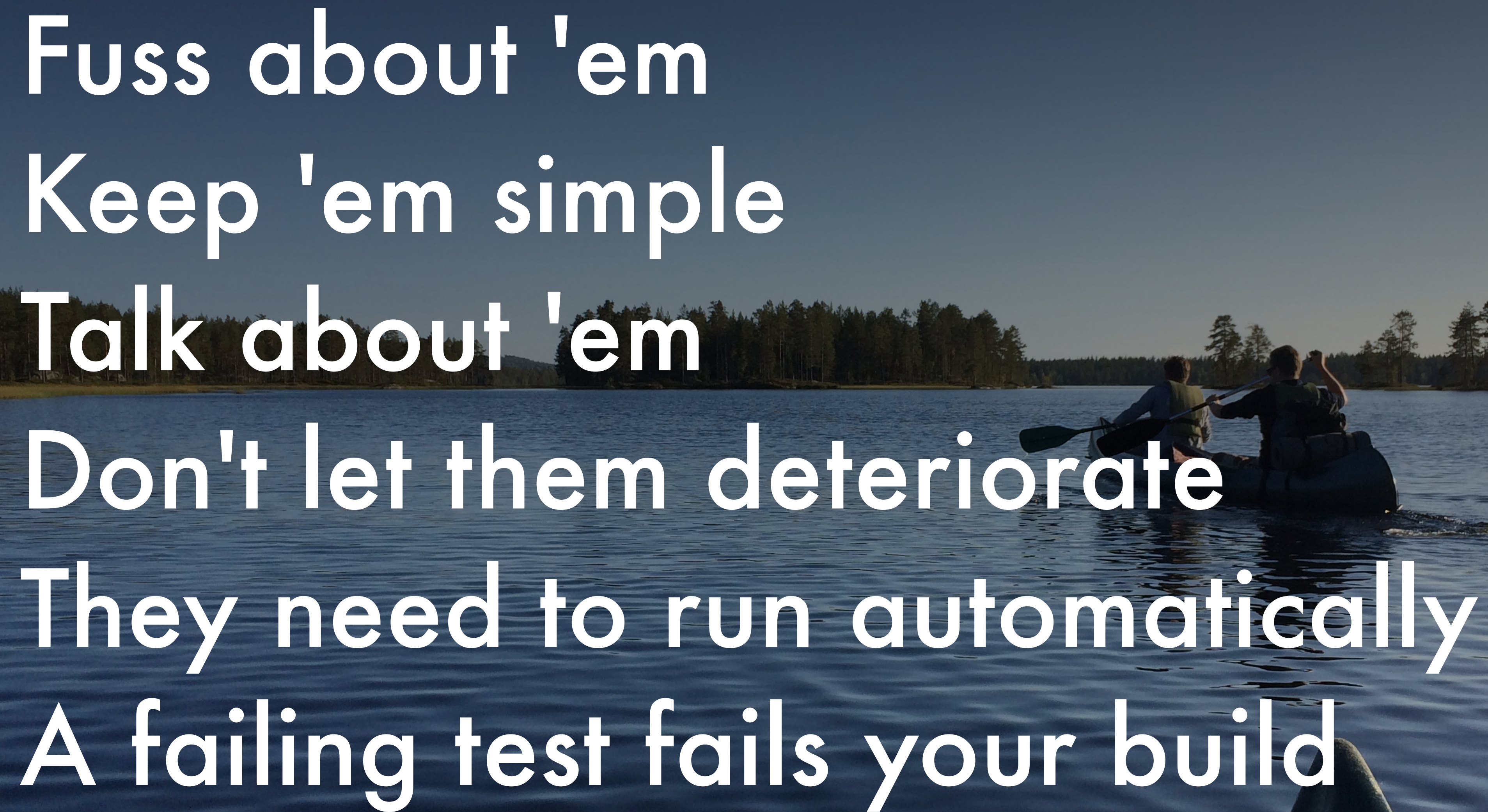
**NO SETUP**



A photograph of two people kayaking on a calm lake. The kayakers are in the middle ground, moving away from the viewer. The water is dark blue with gentle ripples. In the background, there is a dense line of evergreen trees along the shore under a clear, light blue sky. The tip of the kayaker's paddle is visible in the bottom center of the frame.

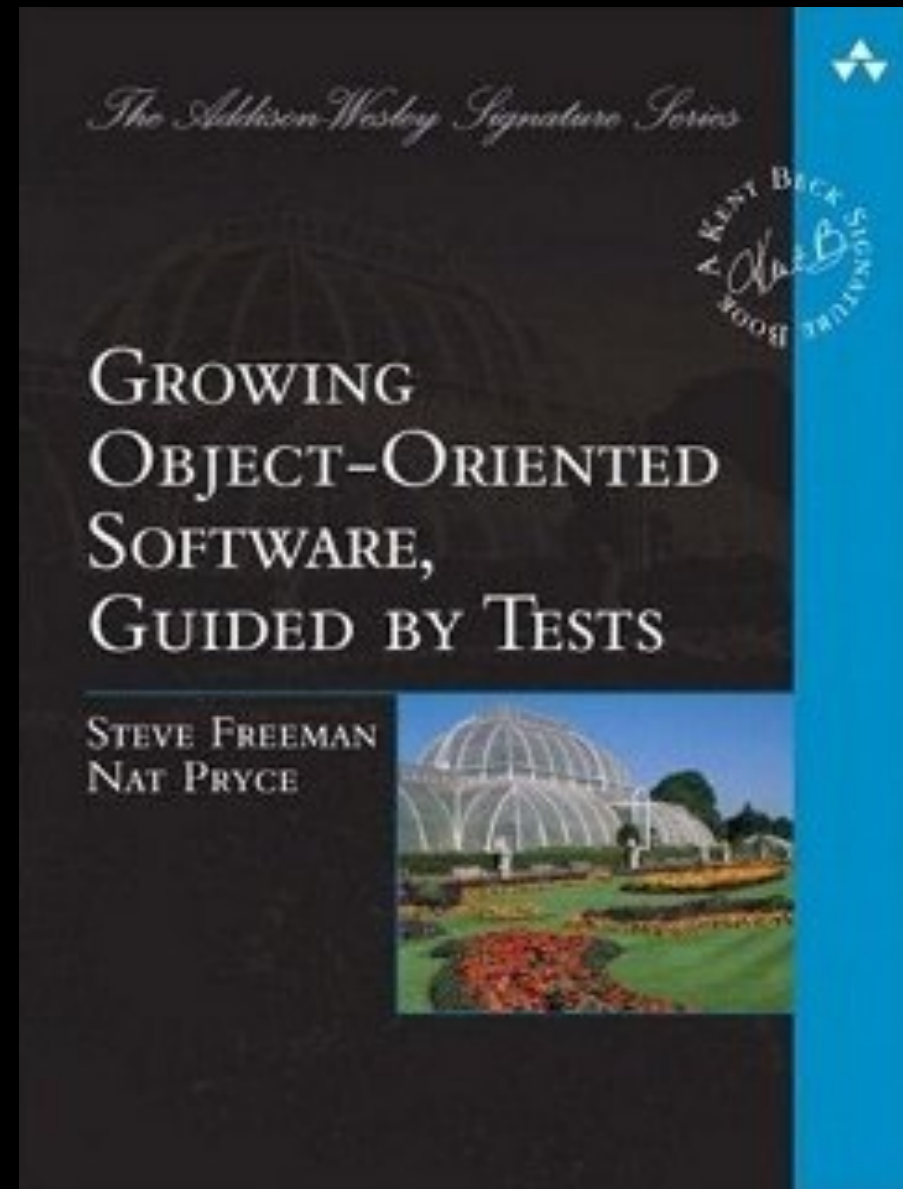
Focus on the craft, not the tools



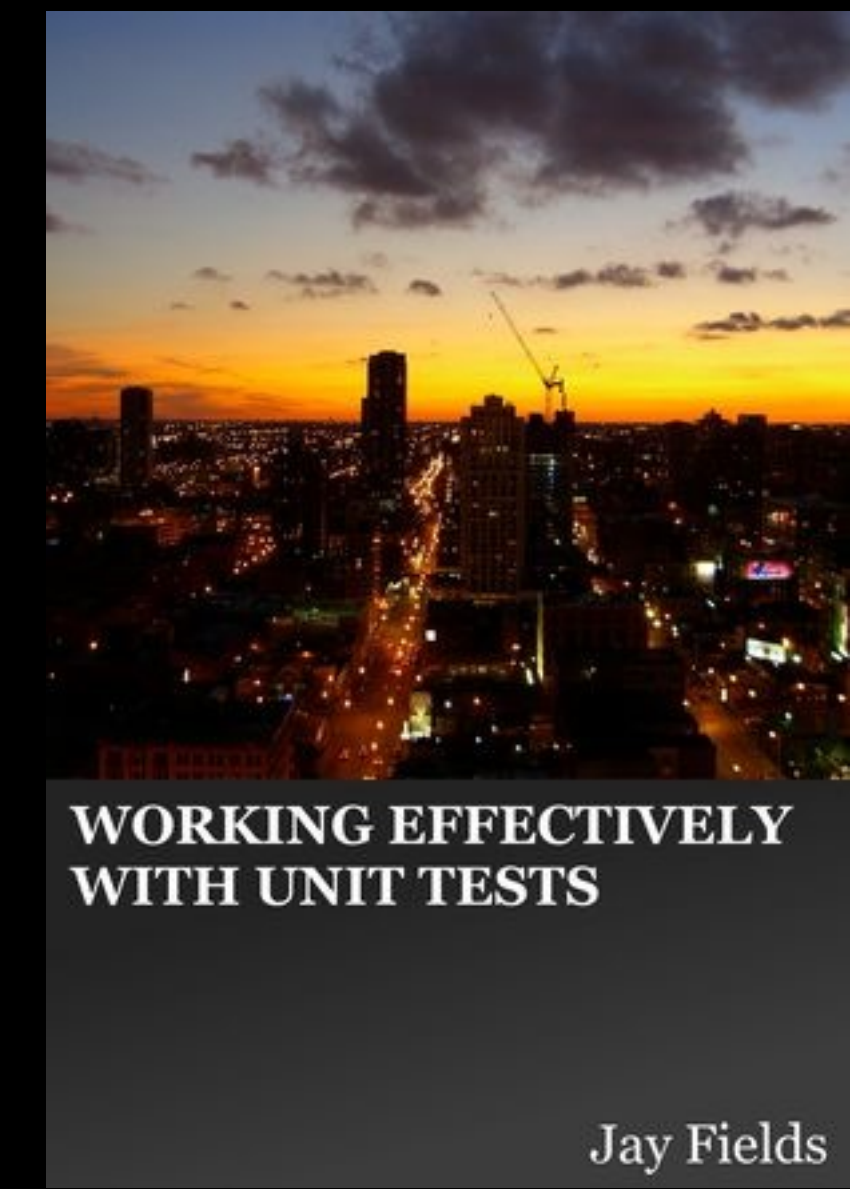


Fuss about 'em  
Keep 'em simple  
Talk about 'em  
Don't let them deteriorate  
They need to run automatically  
A failing test fails your build

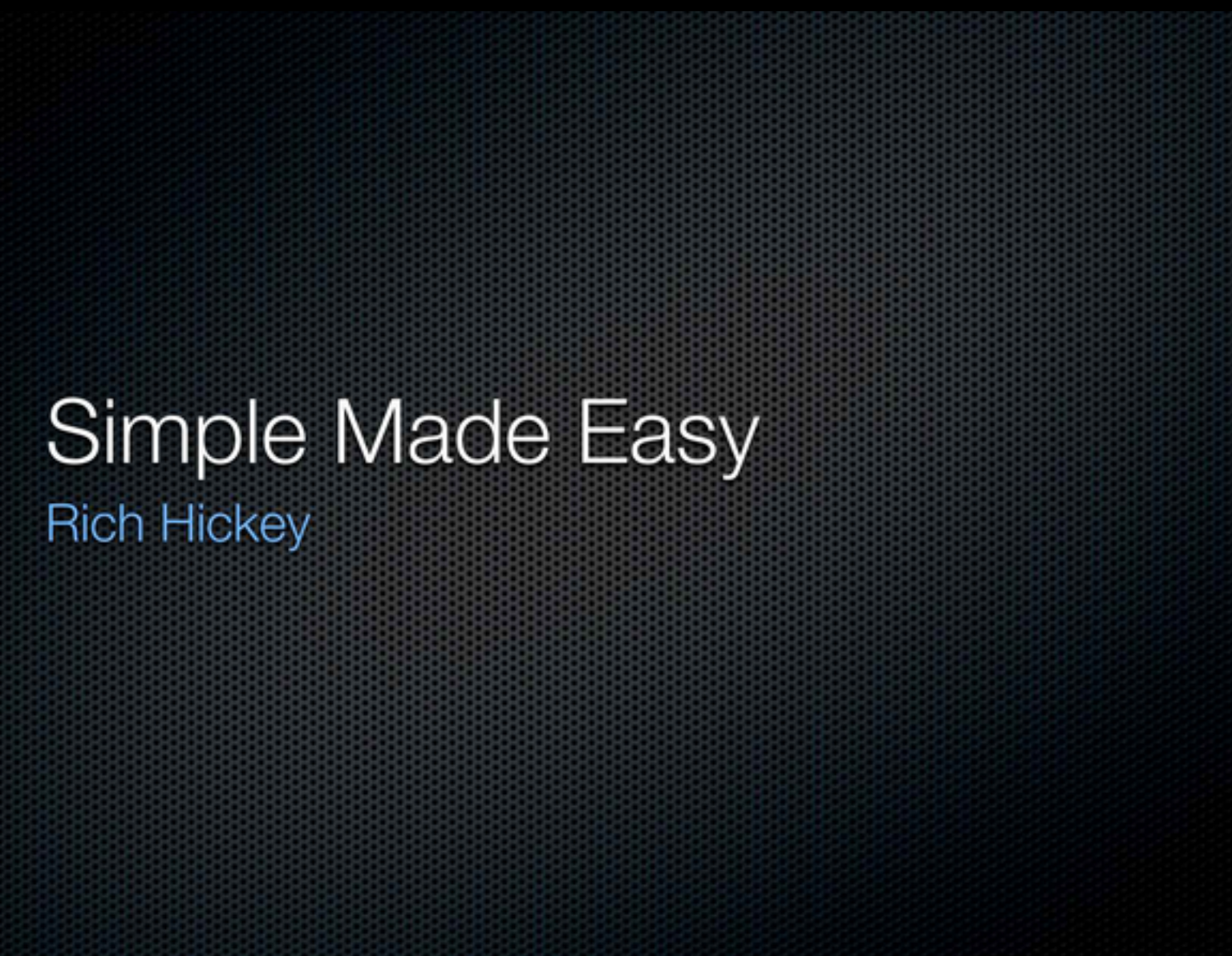




WANT  
TO



LEARN  
MORE?





# THANK YOU

I'm here until Sunday – let me know  
if you want to talk JavaScript

[kimjoar.net](http://kimjoar.net)