Reactive testing

Ondrej Bartas at Blueberry.io

github.com/ondrejbartas

Why to test?

- confidence
- live documentation
- helps to define better API

Why I test

- faster development
- single responsibility functions
- no need for [cmd+r]
- it is fun

How to think

- input A1 => output B1
- input A2 => output B2
- input A3 => output B3

What is A

- minimal needed input to produce B

What is B

- did calling Fn with A returned/called expected thing B?

What do you need?

- test runner (ava/mocha)
- command line task **`yarn test`**, **`gulp test`** etc.
- Continuous Integration - Circle CI, Travis CI, Jenkins
- way to run tests in watch mode


11:47
11:47
24

Watch mode? What the hell!

v1

run all tests on file save

Watch mode? What the hell!

V2

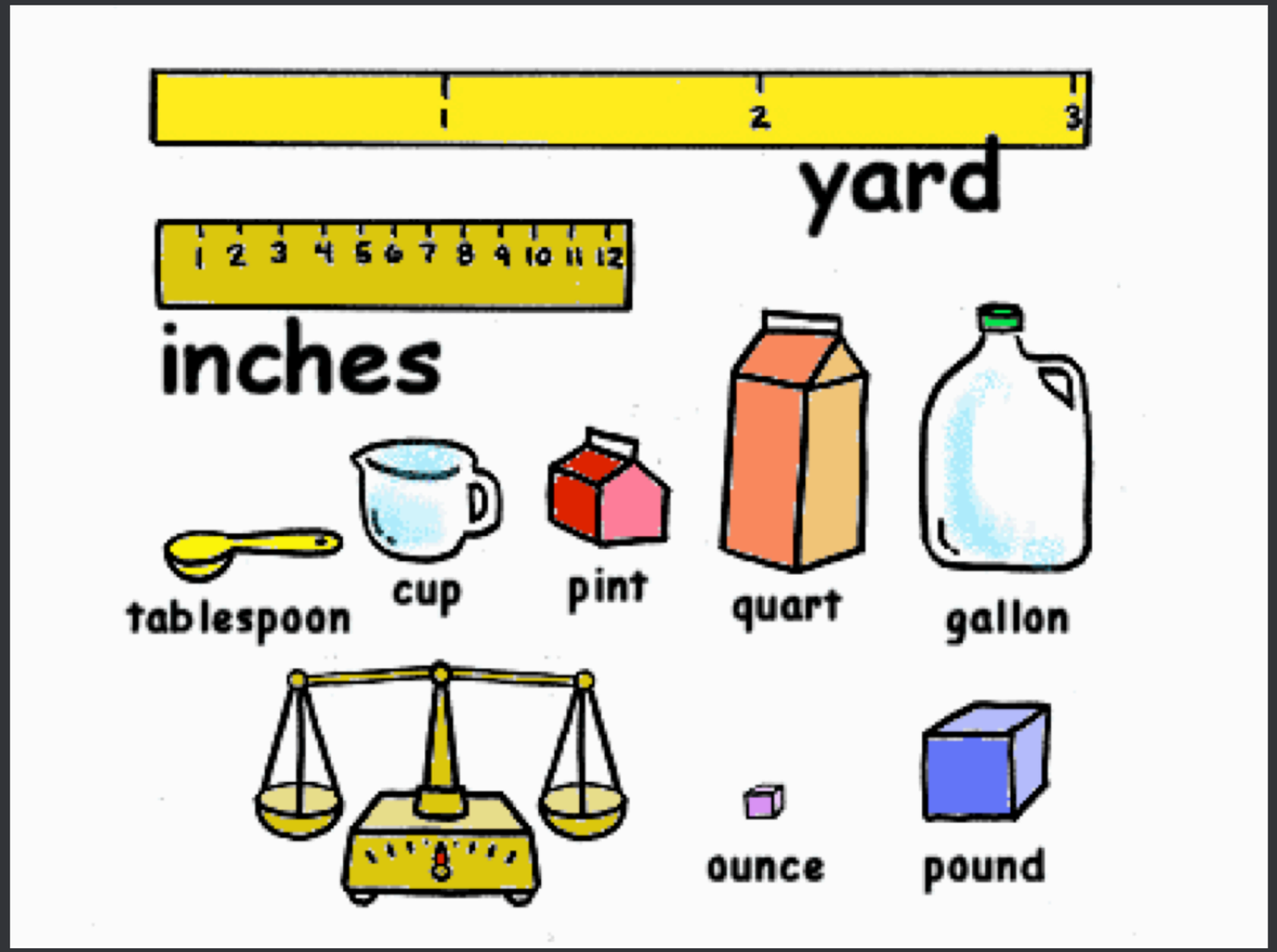
- run tests in saved test file
- ex. when saved **src/test/index.spec.js** run tests in **src/test/index.spec.js**
- run tests in test file associated to saved file
- ex. when saved **src/index.js** run tests in **src/test/index.spec.js**

Start small

Main categories/levels

- unit
- integration
- acceptance

Unit tests


1
2
3
yard
1 2 3 4 5 6 7 8 9 10 11 12
inches
tablespoon
cup
pint
quart
gallon
ounce
pound

1st write test

```
import test from 'ava';
import { incrementCounter } from '../actionCreators';

test('incrementCounter will have right type', t => {
  t.is(incrementCounter().type, 'INCREMENT');
});
```

```
1. incrementCounter will have right type
failed with "(0 , _actionCreators.incrementCounter) is not a function"
   Test.fn (actionCreators.spec.js:5:8)
 _combinedTickCallback (internal/process/next_tick.js:67:7)
 process._tickCallback (internal/process/next_tick.js:98:9)
```

2nd implement

```
export function incrementCounter() {
}
```

```
1. incrementCounter will have right type
failed with "Cannot read property 'type' of undefined"
   Test.fn (actionCreators.spec.js:5:8)
 _combinedTickCallback (internal/process/next_tick.js:67:7)
 process._tickCallback (internal/process/next_tick.js:98:9)
```

```
export function incrementCounter() {
  return {
  }
}
```

```
 1. incrementCounter will have right type

t.is(incrementCounter().type, 'INCREMENT')
                        |
                        undefined


    undefined
      Test.fn (actionCreators.spec.js:5:5)
      _combinedTickCallback (internal/process/next_tick.js:67:7)
      process._tickCallback (internal/process/next_tick.js:98:9)
```

```
export function incrementCounter() {
  return {
    type: ''
  }
}
```

```
 1. incrementCounter will have right type

t.is(incrementCounter().type, 'INCREMENT')
                        |
                        ""

    ""
      Test.fn (actionCreators.spec.js:5:5)
      _combinedTickCallback (internal/process/next_tick.js:67:7)
      process._tickCallback (internal/process/next_tick.js:98:9)
```

```
export function incrementCounter() {
  return {
    type: 'INCREMENT'
  }
}
```

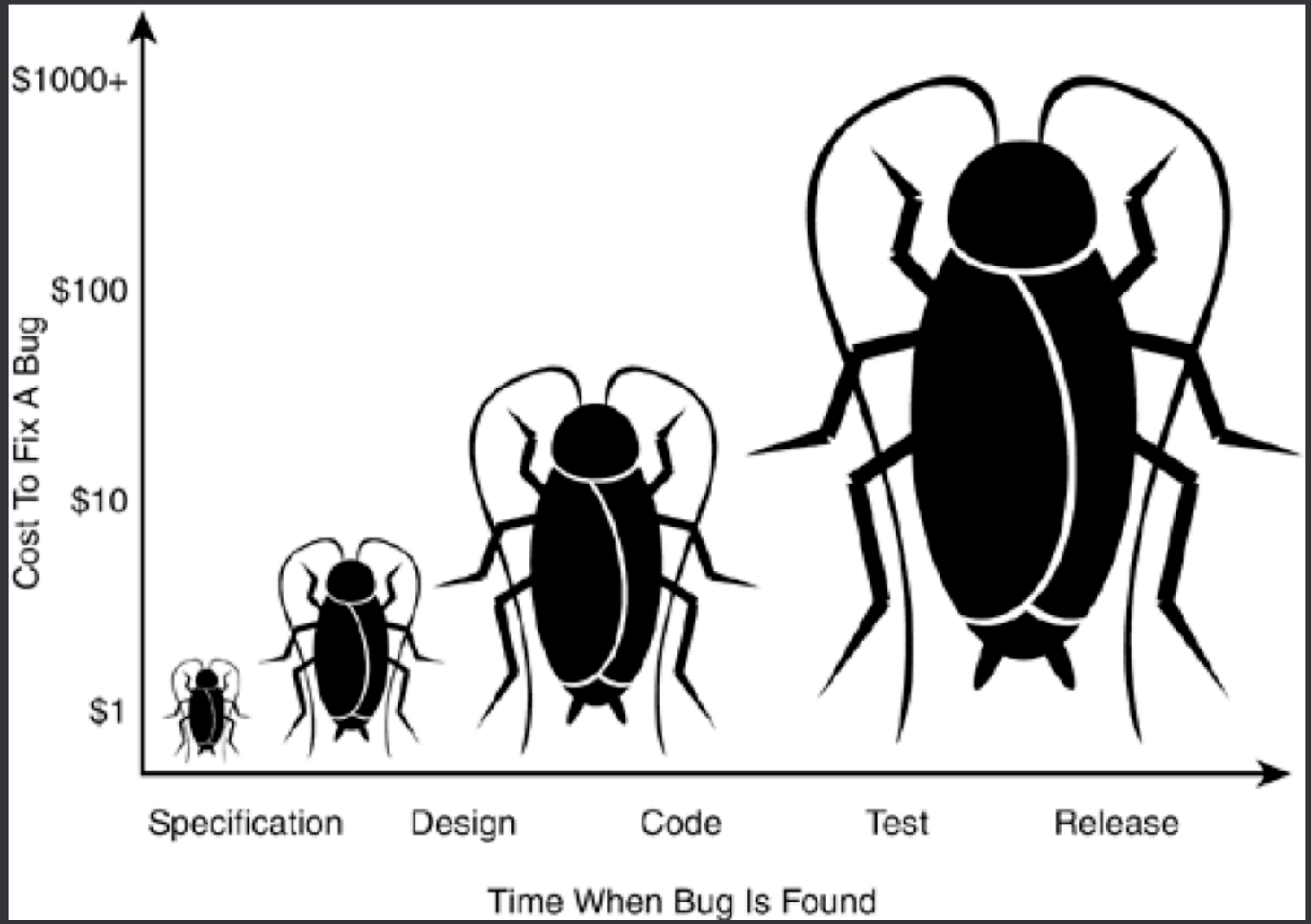
```
1 passed
```

```
export function incrementCounter() {
  return {
    type: 'INCREMENT2'
  }
}
```

```
 1. incrementCounter will have right type

t.is(incrementCounter().type, 'INCREMENT')
                        |
                       "INCREMENT2"

    "INCREMENT2"
      Test.fn (actionCreators.spec.js:5:5)
      _combinedTickCallback (internal/process/next_tick.js:67:7)
      process._tickCallback (internal/process/next_tick.js:98:9)
```

```
export function incrementCounter() {
  return {
    type: 'INCREMENT'
  }
}
```

Refactoring



Age validator from Czech Personal ID

```
import moment from 'moment';

// YYMMDDCCCC -> CCCC is checksum
export default function ageValidation(personalId) {
  // const year = parseInt(personalId.substr(0, 2), 10);
  // const month = parseInt(personalId.substr(2, 2), 10);
  // const day = parseInt(personalId.substr(4, 2), 10);
  // const century = year <= moment().year() - 2000 ? 20 : 19;

  const birth = moment(`${century}${year}-${month}-${day}`);

  return (moment().diff(birth, 'years') < 18) ? 'under 18' : null;
}
```

```
import moment from 'moment';
import test from 'ava';
import ageValidation from '../ageValidation';

function getPersonalID(old) {
  return moment().subtract(old, 'years').format('YYMMDD1234');
}

test('ageValidation should return error when age is under 18', t =>
  t.is(ageValidation(getPersonalID(17)), 'under 18')
);
test('ageValidation should not return error when age is exactly 18', t =>
  t.is(ageValidation(getPersonalID(18)), null)
);
test('ageValidation should not return error when age is over 18', t =>
  t.is(ageValidation(getPersonalID(21)), null)
);
```

New feature - support girls :D

When we create personal ID for girls,
we need to add + 50 to months
so august (8) will became 58
and december (12) will became 62

- boy: 86**08**091111
- girl: 86**58**091111

Update test support function

```
function getPersonalID(old) {
  return moment().subtract(old, 'years').format('YYMMDD1234');
}
```

to

```
function getPersonalID(old, girl = false) {
  const base = moment().subtract(old, 'years').format('YYMMDD1234');
  return girl ?
    `${base.substr(0, 2)}${parseInt(base.substr(2, 1), 10) + 5}${base.substr(4, 7)}` :
    base;
}
```

Add new tests

```
test('ageValidation should return error when age is under 18 and id is girl', t =>
  t.is(ageValidation(getPersonalID(17, true)), 'under 18')
);

test('ageValidation should not return error when girls age is over 18', t =>
  t.is(ageValidation(getPersonalID(20, true)), null)
);
```

```
 4 passed
 1 failed


 1. ageValidation should return error when age is under 18 and id is girl

t.is(ageValidation(getPersonalID(17, true)), 'under 18')
     |              |
     null           "996011234"

    null           "996011234"
      Test.fn (ageValidation.spec.js:30:5)
      _combinedTickCallback (internal/process/next_tick.js:67:7)
      process._tickCallback (internal/process/next_tick.js:98:9)
```

Extraction

```
// YYMMDDCCCC -> CCCC is checksum, girls have MM + 50
export function getAge(personalId) {
  // const year = parseInt(personalId.substr(0, 2), 10);
  // const month = parseInt(personalId.substr(2, 2), 10);
  // const day = parseInt(personalId.substr(4, 2), 10);
  // const century = year <= moment().year() - 2000 ? 20 : 19;

  return moment().diff(moment(`${century}${year}-${month}-${day}`), 'years');
}

export default function ageValidation(personalId) {
  return (getAge(personalId)) < 18) ? 'under 18' : null;
}
```

Still failing

```
 4 passed
 1 failed


 1. ageValidation should return error when age is under 18 and id is girl

t.is(ageValidation(getPersonalID(17, true)), 'under 18')
     |              |
     null           "996011234"

    null          "996011234"
      Test.fn (ageValidation.spec.js:30:5)
      _combinedTickCallback (internal/process/next_tick.js:67:7)
      process._tickCallback (internal/process/next_tick.js:98:9)
```

Variations

```
test('getAge for boy', (t) => {
  t.is(getAge(getPersonalID(5)), 5);
  t.is(getAge(getPersonalID(17)), 17);
  t.is(getAge(getPersonalID(20)), 20);
  t.is(getAge(getPersonalID(65)), 65);
  t.is(getAge(getPersonalID(95)), 95);
});

test('getAge for girl', (t) => {
  t.is(getAge(getPersonalID(5, true)), 5);
  t.is(getAge(getPersonalID(17, true)), 17);
  t.is(getAge(getPersonalID(20, true)), 20);
  t.is(getAge(getPersonalID(65, true)), 65);
  t.is(getAge(getPersonalID(95, true)), 95);
});
```

```
 5 passed
 2 failed


 1. ageValidation should return error when age is under 18 and id is girl

t.is(ageValidation(getPersonalID(17, true)), 'under 18')
     |             |
     null          "996011234"

    null          "996011234"
      Test.fn (ageValidation.spec.js:30:5)
      _combinedTickCallback (internal/process/next_tick.js:67:7)
      process._tickCallback (internal/process/next_tick.js:98:9)


 2. getAge for girl

t.is(getAge(getPersonalID(5, true)), 5)
     |      |
     NaN    "116011234"

    NaN    "116011234"
      Test.fn (ageValidation.spec.js:46:5)
      _combinedTickCallback (internal/process/next_tick.js:67:7)
      process._tickCallback (internal/process/next_tick.js:98:9)
```

Now we know what fails

moment is not able to parse months 5x or 6x

Predefined Personal IDs

```
test('getBirth for girl', (t) => {
  t.is(getBirth('8658091234'), '1986-8-9');
  t.is(getBirth('1558091234'), '2015-8-9');
  t.is(getBirth('1562091234'), '2015-12-9');
});

test('getBirth for boy', (t) => {
  t.is(getBirth('8608091234'), '1986-8-9');
  t.is(getBirth('1508091234'), '2015-8-9');
  t.is(getBirth('1512091234'), '2015-12-9');
});
```

```
// YYMMDDCCCC -> CCCC is checksum
export function getBirth(personalId) {
  // const year = parseInt(personalId.substr(0, 2), 10);
  // const month = parseInt(personalId.substr(2, 2), 10);
  // const day = parseInt(personalId.substr(4, 2), 10);
  // const century = year <= moment().year() - 2000 ? 20 : 19;
  return `${century}${year}-${month}-${day}`;
}

export function getAge(personalId) {
  return moment().diff(moment(getBirth(personalId)), 'years');
}
```

```
 3. getBirth for girl

t.is(getBirth('8658091234'), '1986-8-9')
      |
     "1986-58-9"

    "1986-58-9"
      Test.fn (ageValidation.spec.js:54:5)
      _combinedTickCallback (internal/process/next_tick.js:67:7)
      process._tickCallback (internal/process/next_tick.js:98:9)
```

lets fix it

```
export function getBirth(personalId) {
  ...
    // month % 50 will remove information about girl from ID
  const month = parseInt(personalId.substr(2, 2), 10) % 50;
  ...
}
```


9 passed


WE DID IT!
memegenerator.net

```
import apiCall from './apiCall';

export default function submitForm(values) {
  return async ({ dispatch }) => {
    dispatch({ type: 'FORM_SUBMIT_STARTED' });

    const data = await apiCall(
      '/api/form',
      { method: 'post', body: JSON.stringify(values) }
    );

    if (data.status === 'ok') {
      dispatch({ type: 'FORM_SUBMIT_SUCCESS', payload: data });
    } else {
      dispatch({ type: 'FORM_SUBMIT_ERROR', error: data });
    }

    return {
      type: 'FORM_SUBMIT_FINISHED',
    };
  };
}
```


WELCOME TO YOUR NEW NIGHTMARE.
A NIGHTMARE ON ELM STREET
04.30.10
NEW LINE CINEMA
PLATINUM DUNES
© 2010 WARNER BROS. ENTERTAINMENT INC.
WARNER BROS. PICTURES

What is A?

values from our form

success will be returned for `{ name: "Jake" }`

and we know that `{ name: "wrong" }` will return error

What is B?

that reducer will receive these actions:
- FORM_SUBMIT_STARTED
- FORM_SUBMIT_FINISHED

and based on result of apiCall:
- FORM_SUBMIT_SUCCESS
- FORM_SUBMIT_ERROR

Write description

```
import test from 'ava';

test('submitForm must dispatch FORM_SUBMIT_STARTED', () => {});
test('submitForm must return FORM_SUBMIT_FINISHED', () => {});
test('submitForm must dispatch on success FORM_SUBMIT_SUCCESS', () => {});
test('submitForm must dispatch on error FORM_SUBMIT_ERROR', () => {});
```

How to test dispatch?

```
import sinon from 'sinon';
```

sinon is used for mocking and stubing

and we will use *sinon* for mimicking *dispatch*

```
test('submitForm must dispatch FORM_SUBMIT_STARTED', async () => {
  const dispatch = sinon.stub();
  await submitForm({ name: 'super' })({ dispatch });
});
```

Assert that dispatch was calledWith

```
t.true(dispatch.calledWith({ type: 'FORM_SUBMIT_STARTED' }));
```

but this can produce not helping message at all:

```
  1. submitForm must dispatch FORM_SUBMIT_STARTED

 t.true(dispatch.calledWith({ type: 'FORM_SUBMIT_STARTED2' }))
                  |
                  false
```

we get error (as expected) but message is not helping at all

Use sinon assertion

```
sinon.assert.calledWith(dispatch, { type: 'FORM_SUBMIT_STARTED2' });
```

```
1. submitForm must dispatch FORM_SUBMIT_STARTED
failed with "expected stub to be called with arguments { type: "FORM_SUBMIT_STARTED2" }
 stub({ type: "FORM_SUBMIT_STARTED" }) at _callee$ (/Users/bartas/local_work/blueberry/
 stub({ payload: { status: "ok" }, type: "FORM_SUBMIT_SUCCESS" }) at _callee$ (/Users/ba
   stub({ payload: { status: "ok" }, type: "FORM_SUBMIT_SUCCESS" }) at _callee$ (/Users/
```

they explain what is happening

Extract creator for test

```
async function submit(values) {
  const dispatch = sinon.stub();
  const result = await submitForm(values)({ dispatch });
  return { dispatch, result };
}
```

Assert that dispatch was called

```
test('submitForm must dispatch on success FORM_SUBMIT_SUCCESS', async () => {
  const { dispatch } = await submit({ name: 'John' });
  sinon.assert.calledWith(dispatch, { type: 'FORM_SUBMIT_SUCCESS' });
});
```

```
1. submitForm must dispatch on success FORM_SUBMIT_SUCCESS
failed with "expected stub to be called with arguments { type: "FORM_SUBMIT_SUCCESS" }
 stub({ type: "FORM_SUBMIT_STARTED" }) at _callee$ (/Users/bartas/local_work/blueberry/r
 stub({ payload: { status: "ok" }, type: "FORM_SUBMIT_SUCCESS" }) at _callee$ (/Users/ba
   stub({ payload: { status: "ok" }, type: "FORM_SUBMIT_SUCCESS" }) at _callee$ (/Users/
     Object.fail (/Users/bartas/local_work/blueberry/reactive-testing/node_modules/sinon
```

Sinon assertion calledWithMatch (partial match)

```
test('submitForm must dispatch on success FORM_SUBMIT_SUCCESS', async () => {
  const { dispatch } = await submit({ name: 'John' });
  sinon.assert.calledWithMatch(dispatch, { type: 'FORM_SUBMIT_SUCCESS' });
});
```

```
1 passed
```

What about freaking apiCall?

Stub api (3rd party)

```
import sinon from 'sinon';
import Promise from 'bluebird';
import * as apiActions from '../apiCall';

const api = sinon.stub(apiActions, 'apiCall');
api.returns(Promise.resolve({ status: 'error' }));
sinon.assert.calledWith(api, '/api/form', { method: 'post', body: '{"name":"Error Name"}' });
api.restore() // Do not forget about restore
```

conditional returns

```
api.withArgs('a1').returns('b1');
api.withArgs('a2').returns('b2');
```

full test

```
import sinon from 'sinon';
import Promise from 'bluebird';
import * as apiActions from '../apiCall';

test('submitForm must dispatch on error FORM_SUBMIT_ERROR with stubbed api', async () => {
  const api = sinon.stub(apiActions, 'apiCall');
  api.returns(Promise.resolve({ status: 'error' }));
  const { dispatch } = await submit({ name: 'Error Name' });

  sinon.assert.calledWithMatch(dispatch, { type: 'FORM_SUBMIT_ERROR' });
  sinon.assert.calledWithMatch(api, '/api/form', { method: 'post' });
  sinon.assert.calledWithMatch(api, '/api/form', { body: '{"name":"Error Name"}' });

  api.restore();
});
```


5 passed

nock

For fetch/XHR requests you can use

```
nock('http://your.api.com').post('/api/form')
  .reply(
    400,
    {
      status: 'error',
      errors: {
        name: 'invalid name'
      }
    }
  );
```

Unit testing of React components

```
import React, { Component, PropTypes as RPT } from 'react';

export default class Counter extends Component {
  static propTypes = {
    counter: RPT.number.isRequired,
  }

  render() {
    return (
      <div>
        Counter: {this.props.counter}
      </div>
    );
  }
}
```

Setup

```
import React from 'react';
import ReactDOM from 'react-dom/server';
import test from 'ava';
import { jsdom } from 'jsdom';
import Counter from '../Counter';

global.document = jsdom('<!doctype html><html><body></body></html>');
global.window = global.document.defaultView;

const renderHtml = counter =>
  ReactDOM.renderToStaticMarkup(
    <Counter counter={counter} />,
    global.document.createElement('div')
  );
```

Assertions

```
test('Counter should render without crash', () =>
  renderHtml()
);

test('Counter should contain Counter text', (t) => {
  t.regex(renderHtml(3), /Counter/);
});

test('Counter should contain counter value', (t) => {
  t.regex(renderHtml(3), /3/);
  t.regex(renderHtml(4), /4/);
  t.regex(renderHtml(105), /105/);
});
```

```
export default class Counter extends Component {
  ...
  getSquare() {
    return this.props.counter * this.props.counter;
  }
  ...
}
```

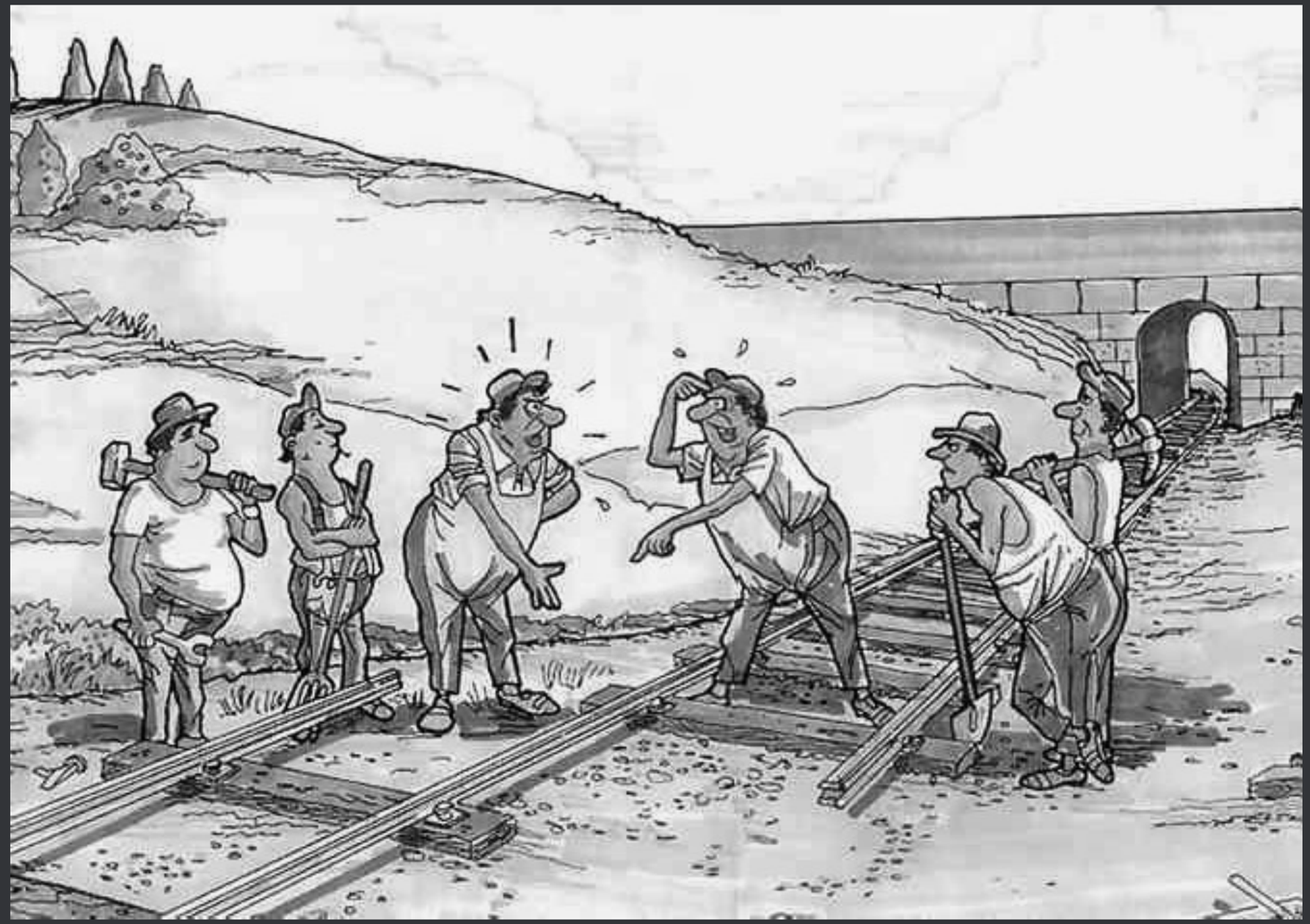
```
 1. Counter getSquare returns correct values

t.is(ReactBrowserDOM.render(<Counter counter={5} />, global.document.createElement('div')).getSquare(), 22)
                                                                                          |
                                                                                          25
```

```
import ReactDOM from 'react-dom';
test('Counter getSquare returns correct values', (t) => {
  t.is(ReactDOM.render(<Counter counter={0} />, document.createElement('div')).getSquare(), 0);
  t.is(ReactDOM.render(<Counter counter={5} />, document.createElement('div')).getSquare(), 25);
  t.is(ReactDOM.render(<Counter counter={10} />, document.createElement('div')).getSquare(), 100);
});
```

What else can be used for component tests:

- Jest
- Enzyme
- Karma

Integration tests



```
import React, { Component, PropTypes as RPT } from 'react';
import { bindActionCreators } from 'redux';
import Counter from './Counter';
import { connect } from 'react-redux';
import { incrementCounter as increment } from './actionCreators';

class CounterPage extends Component {
  static propTypes = {
    counter: RPT.number.isRequired,
    increment: RPT.func.isRequired,
  }

  render() {
    return (
      <Counter
        counter={this.getSquare()}
        increment={this.props.increment}
      />
    );
  }
}

export default connect(
  state => ({ counter: state.counter }),
  dispatch => ({ increment: bindActionCreators(increment, dispatch) })
)(CounterPage);
```

Preparing test

```
import React from 'react';
import ReactDOM from 'react-dom';
import test from 'ava';
import { jsdom } from 'jsdom';
import CounterPage from '../CounterPage';

global.document = jsdom('<!doctype html><html><body></body></html>');
global.window = global.document.defaultView;

const renderHtml = () =>
  ReactDOM.render(
    <CounterPage />,
    global.document.createElement('div')
  );

test('Counter Page should render', () => {
  renderHtml();
});
```

We are missing store in context

```
  1. Counter Page should render
  failed with "Could not find "store" in either the context or props of "Connect(CounterPage)". Either wrap the
 root component in a <Provider>, or explicitly pass "store" as a prop to "Connect(CounterPage)"."
     invariant (/Users/bartas/local_work/blueberry/reactive-testing/node_modules/invariant/invariant.js:42:15)
   new Connect (/Users/bartas/local_work/blueberry/reactive-testing/node_modules/react-redux/lib/components/con
nect.js:131:36)
   /Users/bartas/local_work/blueberry/reactive-testing/node_modules/react/lib/ReactCompositeComponent.js:294:18
   measureLifeCyclePerf (/Users/bartas/local_work/blueberry/reactive-testing/node_modules/react/lib/ReactCompos
iteComponent.js:74:12)
```

MOCK store passed through context

```
import sinon from 'sinon';
import { Provider } from 'react-redux';

const store = {
  getState: () => ({ counter: 5 }),
  dispatch: () => ()
}

const renderHtml = (store) =>
  ReactDOM.render(
    <Provider store={store}>
      <CounterPage />
    </Provider>,
    global.document.createElement('div')
  );
```

Use already tested store

```
import { createStore } from 'redux';
import { Provider } from 'react-redux';

export default function createAppStore(initialState) {
  return createStore(reducers, initialState);
}

const store = createAppStore({ someInitialState: 'values' })

<Provider store={store}>
```

Context of app is given! Use It!

```
const renderAppHtml = (component, initialState) =>
  ReactDOM.render(
    <Provider store={createStore(initialState)}>
      {component}
    </Provider>,
    global.document.createElement('div')
  );
```

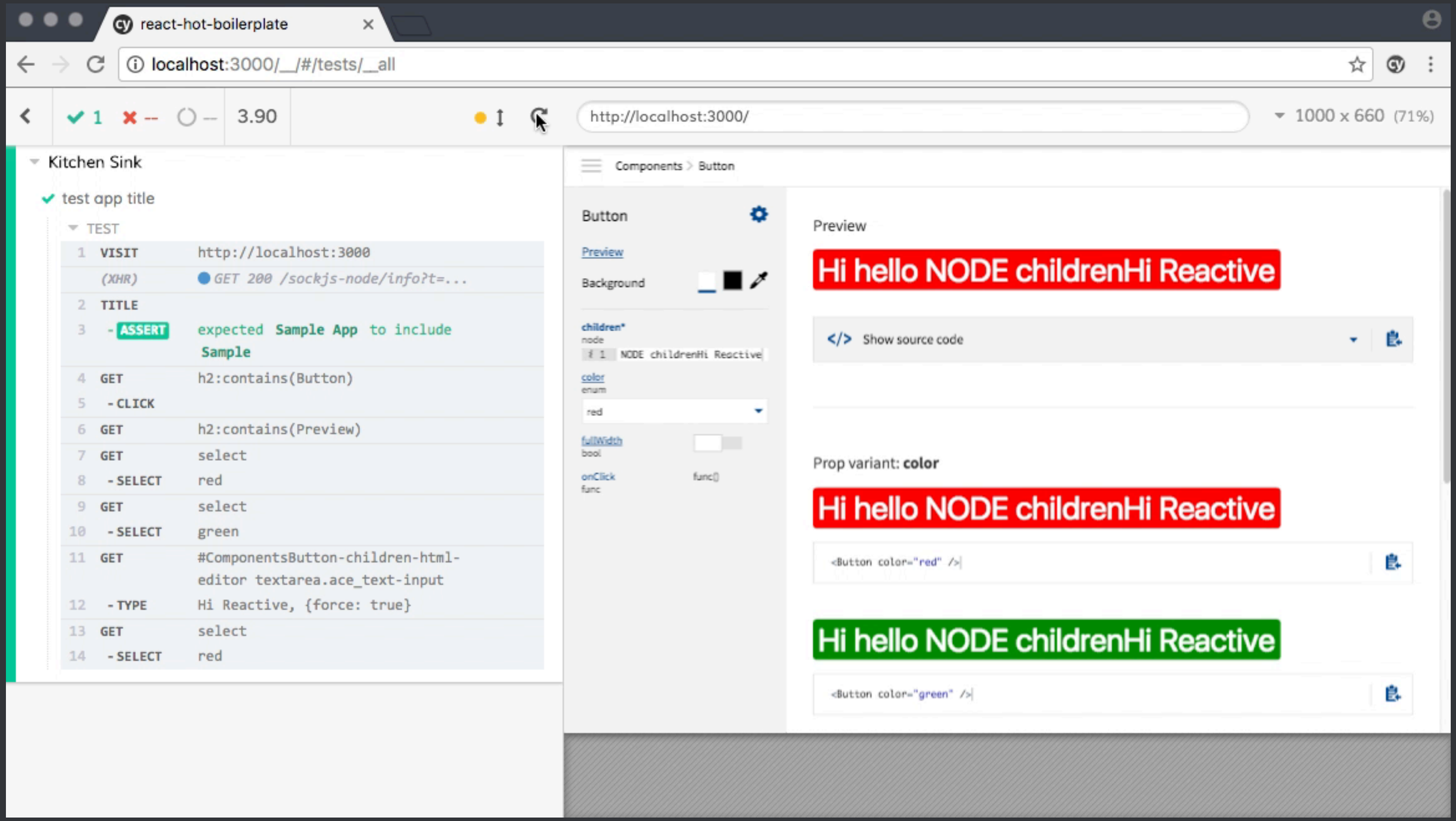
Provide developers with documentation :D

```
test('Counter Page should render', () => {
  renderAppHtml(<CounterPage />, { counter: 7 })
});
```

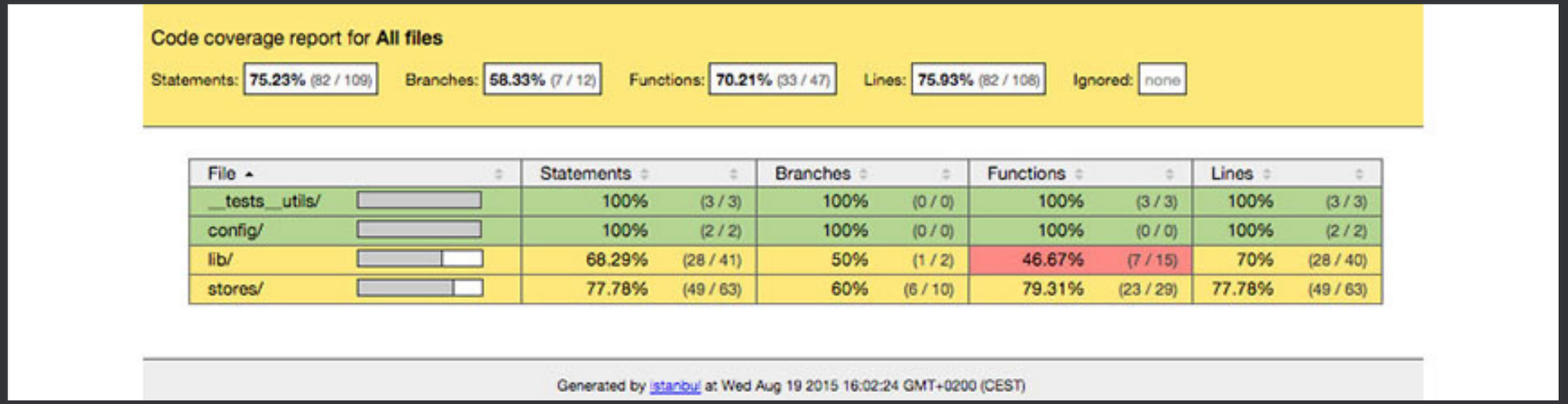

```
test('Counter Page should render counter value', (t) => {
  t.regex(renderAppHtml(<CounterPage />, { counter: 7 }), /Counter: 7/)
});
```

Acceptance/Features

- Cypress
- Nightwatch + Selenium


react-hot-boilerplate
localhost:3000/__/#/tests/__all
1
--
--
3.90
http://localhost:3000/
1000 x 660 (71%)
Kitchen Sink
test app title
TEST
1 VISIT http://localhost:3000
(XHR) GET 200 /sockjs-node/info?t=...
2 TITLE
3 - ASSERT expected Sample App to include Sample
4 GET h2:contains(Button)
5 - CLICK
6 GET h2:contains(Preview)
7 GET select
8 - SELECT red
9 GET select
10 - SELECT green
11 GET #ComponentsButton-children-html-editor textarea.ace_text-input
12 - TYPE Hi Reactive, {force: true}
13 GET select
14 - SELECT red
Components > Button
Button
Preview
Background
children*
node
1 NODE childrenHi Reactive
color
enum
red
fullWidth
bool
onClick
func
func()
Preview
Hi hello NODE childrenHi Reactive
</> Show source code
Prop variant: color
Hi hello NODE childrenHi Reactive
<Button color="red" />
Hi hello NODE childrenHi Reactive
<Button color="green" />

Code coverage



Code coverage report for **All files**

Statements: **75.23%** (82 / 109) Branches: **58.33%** (7 / 12) Functions: **70.21%** (33 / 47) Lines: **75.93%** (82 / 108) Ignored: none

File	Statements		Branches		Functions		Lines	
__tests__utils/	100%	(3 / 3)	100%	(0 / 0)	100%	(3 / 3)	100%	(3 / 3)
config/	100%	(2 / 2)	100%	(0 / 0)	100%	(0 / 0)	100%	(2 / 2)
lib/	68.29%	(28 / 41)	50%	(1 / 2)	46.67%	(7 / 15)	70%	(28 / 40)
stores/	77.78%	(49 / 63)	60%	(6 / 10)	79.31%	(23 / 29)	77.78%	(49 / 63)

Generated by istanbul at: Wed Aug 19 2015 16:02:24 GMT+0200 (CEST)

Thanks

Ondrej Bartas at Blueberry.io

github.com/ondrejbartas/reactive-testing