Testing web apps

Miroslav Jančařík, Filip Šátek
miroslav.jancarik@firma.seznam.cz
filip.satek@firma.seznam.cz

Introduction

- Your name and surname
- Your position
- Where do you work
- Your experience
- Your expectations

Rules

- No cell phone
- No notebook
- One person speaks at a time

Schedule

- ES6/React
- Unit tests
- Integration tests
- Regression tests
- E2E tests
- Performance tests

Why ES6


M MM 2359

ES6

- Let and Const
- Assignment Destructuring
- Spread operator and rest parameters
- Arrow functions
- Template literals
- Promises
- Async/await
- Classes
- Modules

Let and Const

- **let and const are alternatives to var** for declaring variables

- **let is block-scoped** instead of lexically scoped to a function

- let is not hoisted to the top of the block, while var declarations are hoisted to top of the function

- const is also block-scoped, not hoisted, and constrained by TDZ semantics

Assignment Destructuring

- let color = car.color;
- let wheels = car.wheels;
- let **{ wheels } = car** === let **wheels = car.wheels**
- let **{ color: paint } = car** === let **paint = car.color**

Spread operator and rest parameters

- Rest parameters are a better alternative to arguments
  - **function foo (bar, ...rest) {}**
  - Removes the need for .apply when calling functions, **fn(...[1, 2, 3])** is equivalent to **fn(1, 2, 3)**

Arrow functions

- The this keyword is inherited from the parent context

- **param => returnValue**

- [1, 2].map(**x => x * 2**) vs **[1,2].map(function(x) { return x * 2; })**

Promises

node95.js

```
var floppy = require('floppy');

floppy.load('disk1', function (data1) {
    floppy.prompt('Please insert disk 2', function () {
        floppy.load('disk2', function (data2) {
            floppy.prompt('Please insert disk 3', function () {
                floppy.load('disk3', function (data3) {
                    floppy.prompt('Please insert disk 4', function () {
                        floppy.load('disk4', function (data4) {
                            floppy.prompt('Please insert disk 5', function () {
                                floppy.load('disk5', function (data5) {
                                    // if node.js would have existed in 1995
                                });
                            });
                        });
                    });
                });
            });
        });
    });
});

```

Promises

- The Promise object is used for **asynchronous operations**. A Promise represents an operation that might not have completed yet, but is expected in the future.

Async/await

- An **async** function can contain **await** expression that pauses the execution of the **async** function and waits for the passed Promise's resolution, and then resumes the **async** function's execution and returns the resolved value.

Classes

- Not “traditional” classes, **syntax sugar** on top of prototypal inheritance
- Syntax similar to declaring objects, **class Car {}**
- Instance methods
- Static methods
- Constructor method
- Prototypal inheritance with a simple syntax **class Car extends Vehicle**

Modules

- **export default value** exports a default binding
- **export var car = 'car'** exports a named binding
- **export** { car, vehicle } exports all bindings using a single declaration

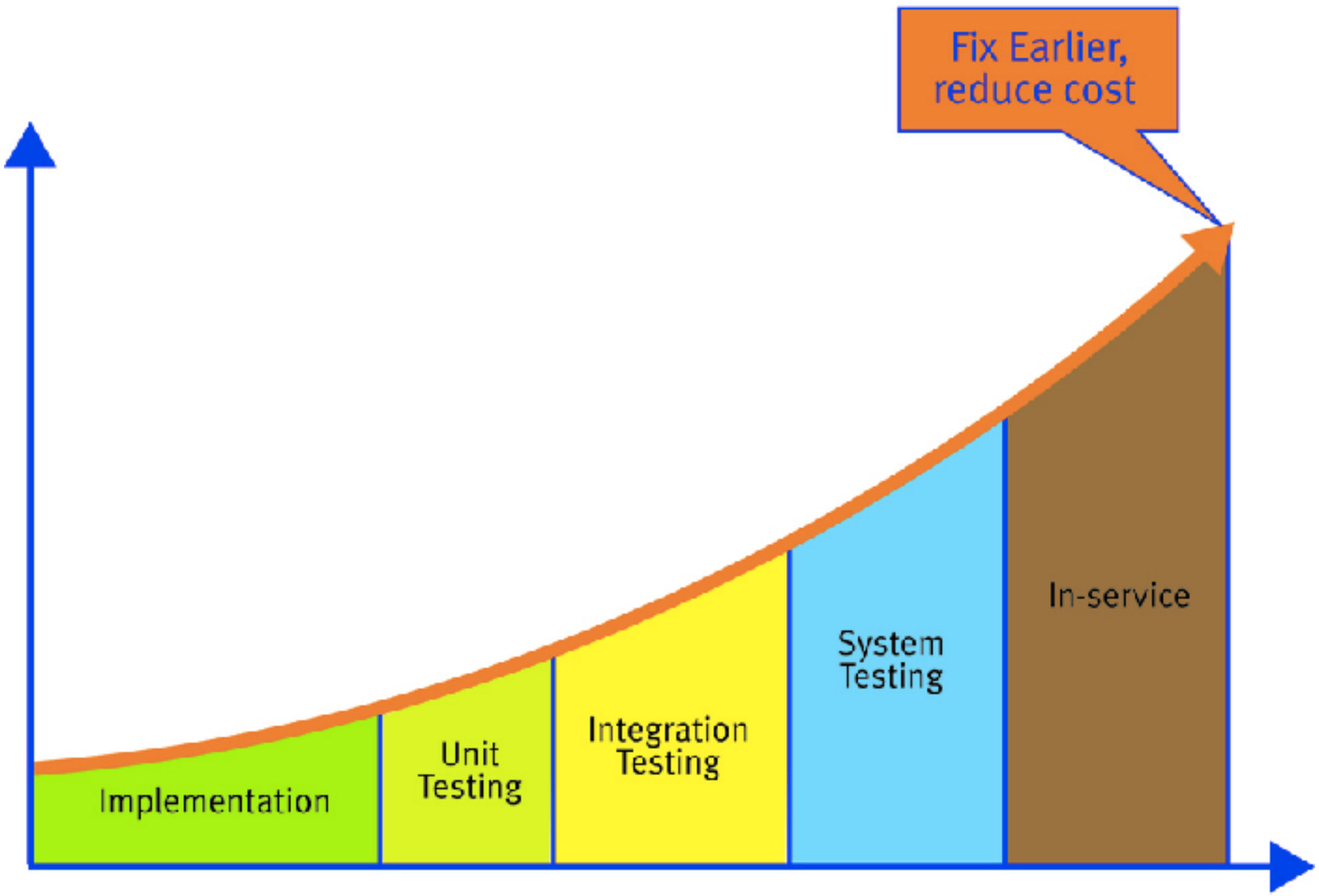
Modules

- **import 'car'** loads the car module into the current module
- **import car from 'car'** assigns the default export of the car module to a local car variable
- **import { car, vehicle } from 'machine'** imports named exports car and vehicle from the machine module
- **import { car as baseCar } from 'machine'** imports named export car but aliased as a baseCar variable
- **import { default } from 'car'** also imports the default export
- **import { default as car } from 'car'** imports the default export aliased as car
- **import machine, { car, vehicle } from 'machine'** mixes default machine with named exports car and vehicle in one declaration
- **import * as machine from 'machine'** imports all exports from the machine module as a machine variable (an object containing all exported bindings)

Why writing tests


BCE ECB EZB EKT
50
EURO
10
THE UNITED STATES OF AMERICA
F 34512877F
WASHINGTON, D.C.

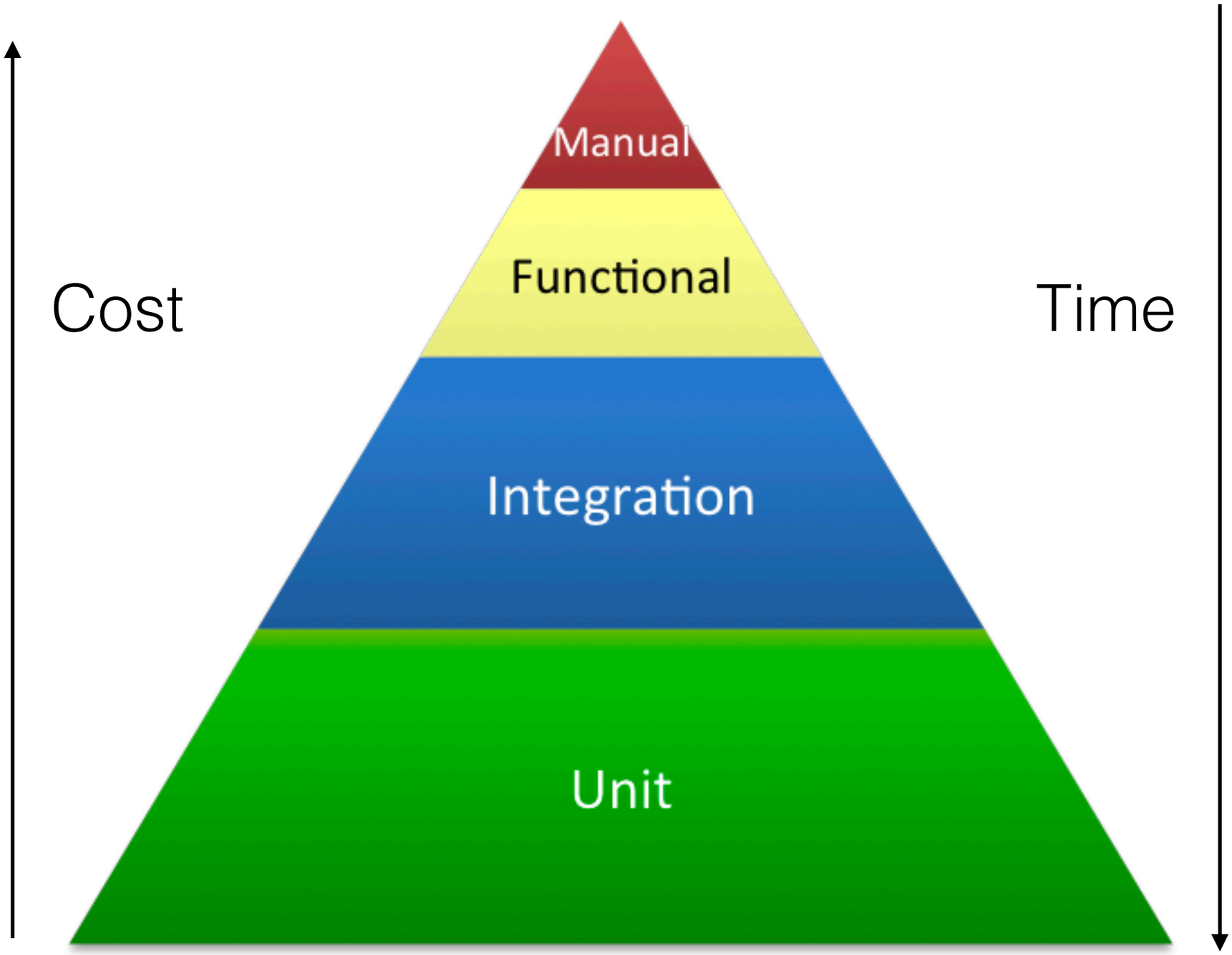
Bug cost


Fix Earlier,
reduce cost
Implementation
Unit
Testing
Integration
Testing
System
Testing
In-service

Consequence

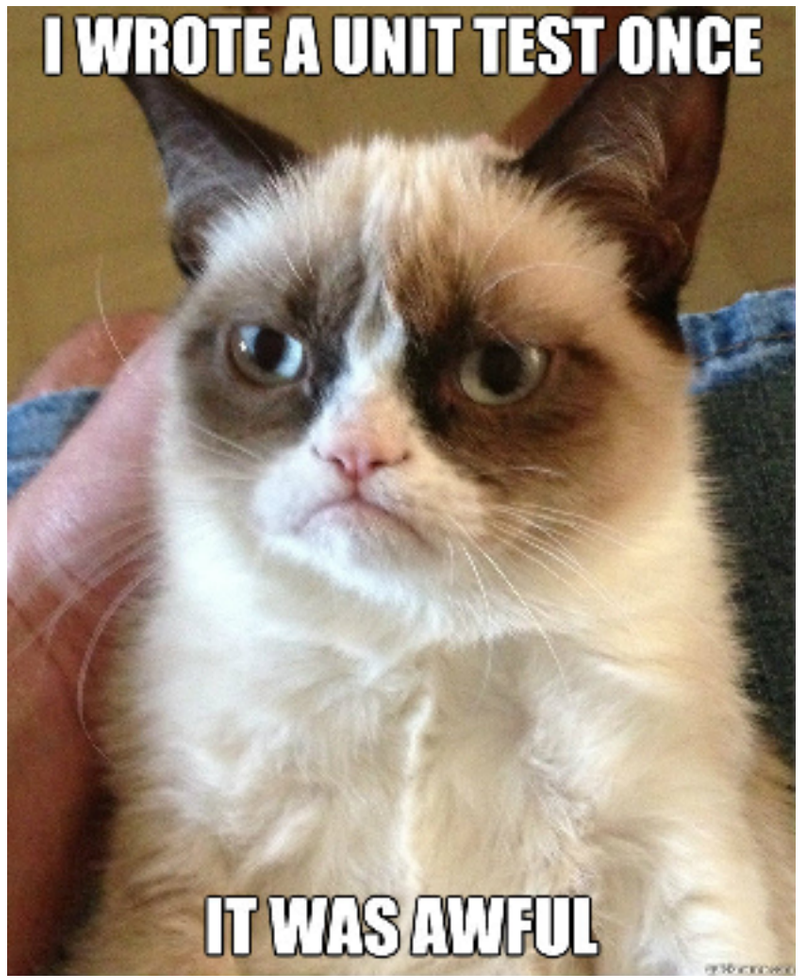
- No stress
- Easy maintenance
- Boost develop

Test pyramid

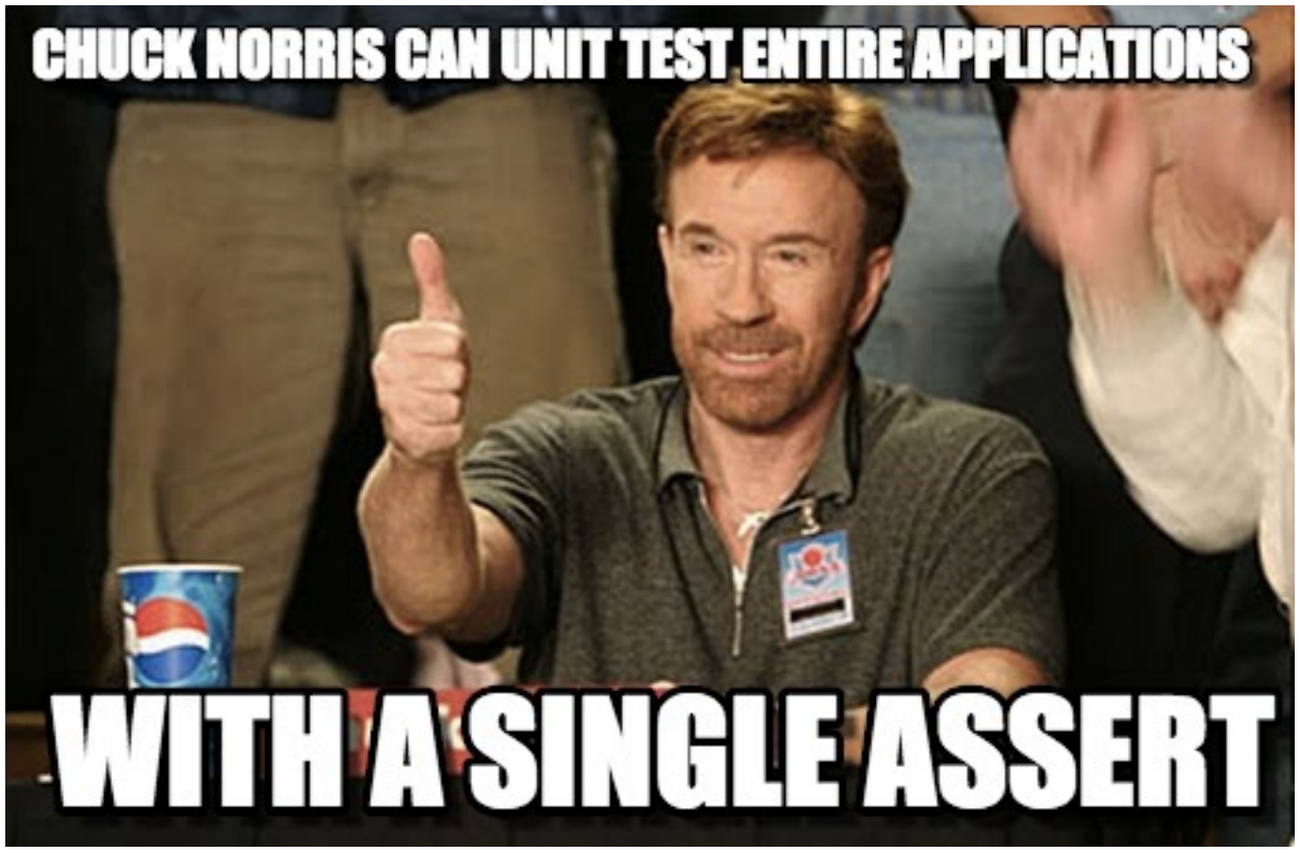


Unit tests

Unit tests



Unit tests


CHUCK NORRIS CAN UNIT TEST ENTIRE APPLICATIONS
WITH A SINGLE ASSERT

Unit tests

- Easy
- Quick
- Isolated

Unit tests

- Arrange
- Act
- Assert

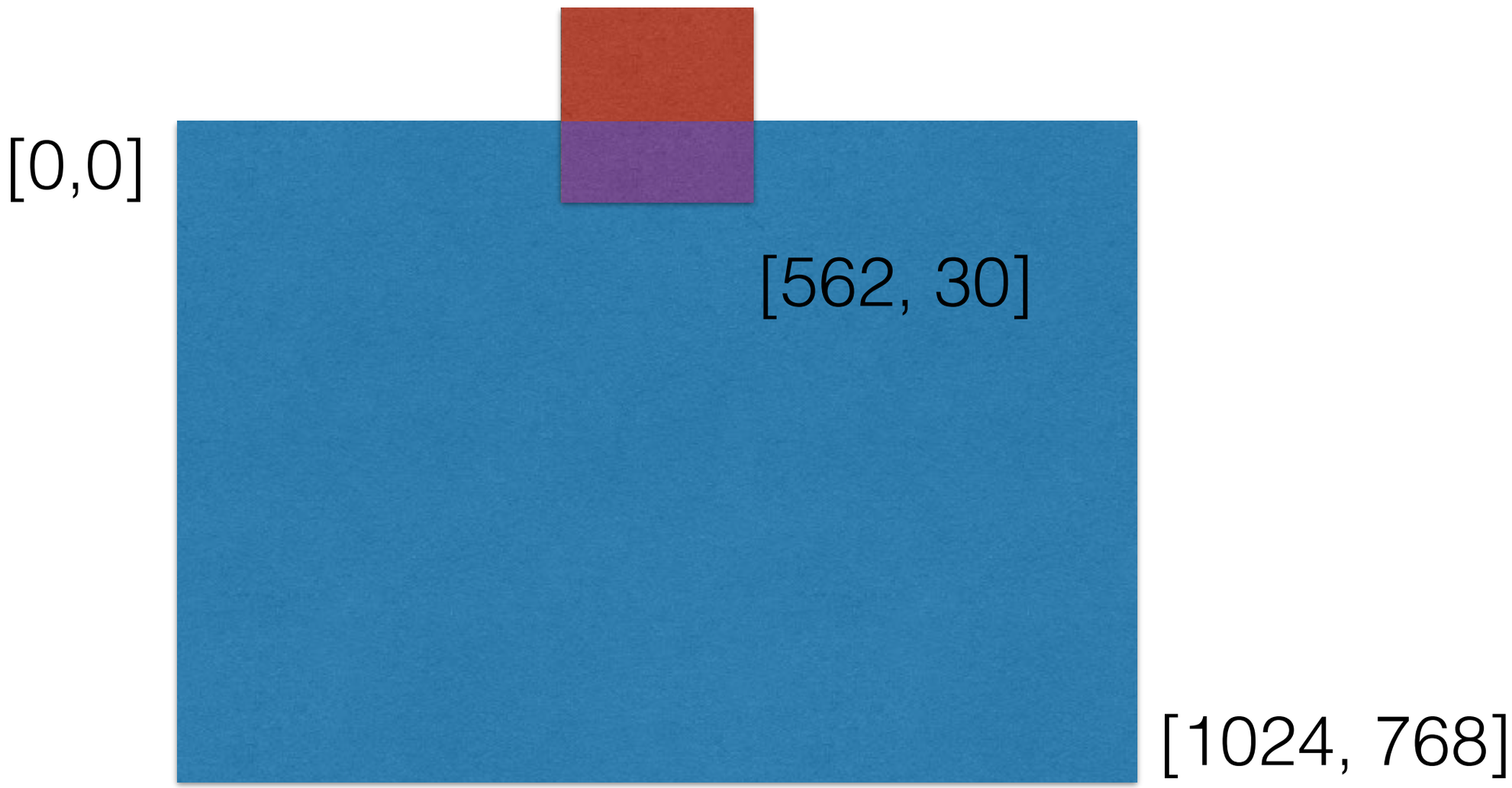
Demo

Workshop 1

$\tilde{x} = x_{\left(\frac{n+1}{2}\right)}$, je-li n *liché* číslo,

$\tilde{x} = \frac{x_{\left(\frac{n}{2}\right)} + x_{\left(\frac{n}{2}+1\right)}}{2}$, je-li n *sudé* číslo.

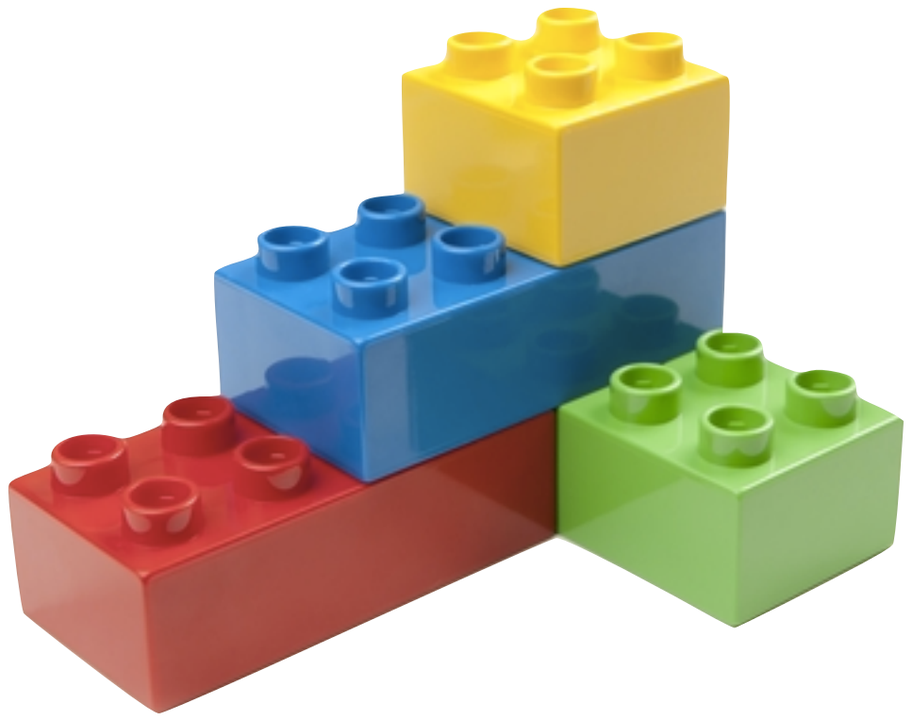
Workshop 2



Other possibilities

- Property based testing
- Stryker mutator

Integration tests



React

- Javascript library for building user interfaces
- **Virtual DOM**
- It’s all about components
- Data flow

React Top level API

- **Component** - base class for ES6

ReactDOM Top level API

- **render** - renders a ReactElement into the DOM

React API

https://reactjs.org/

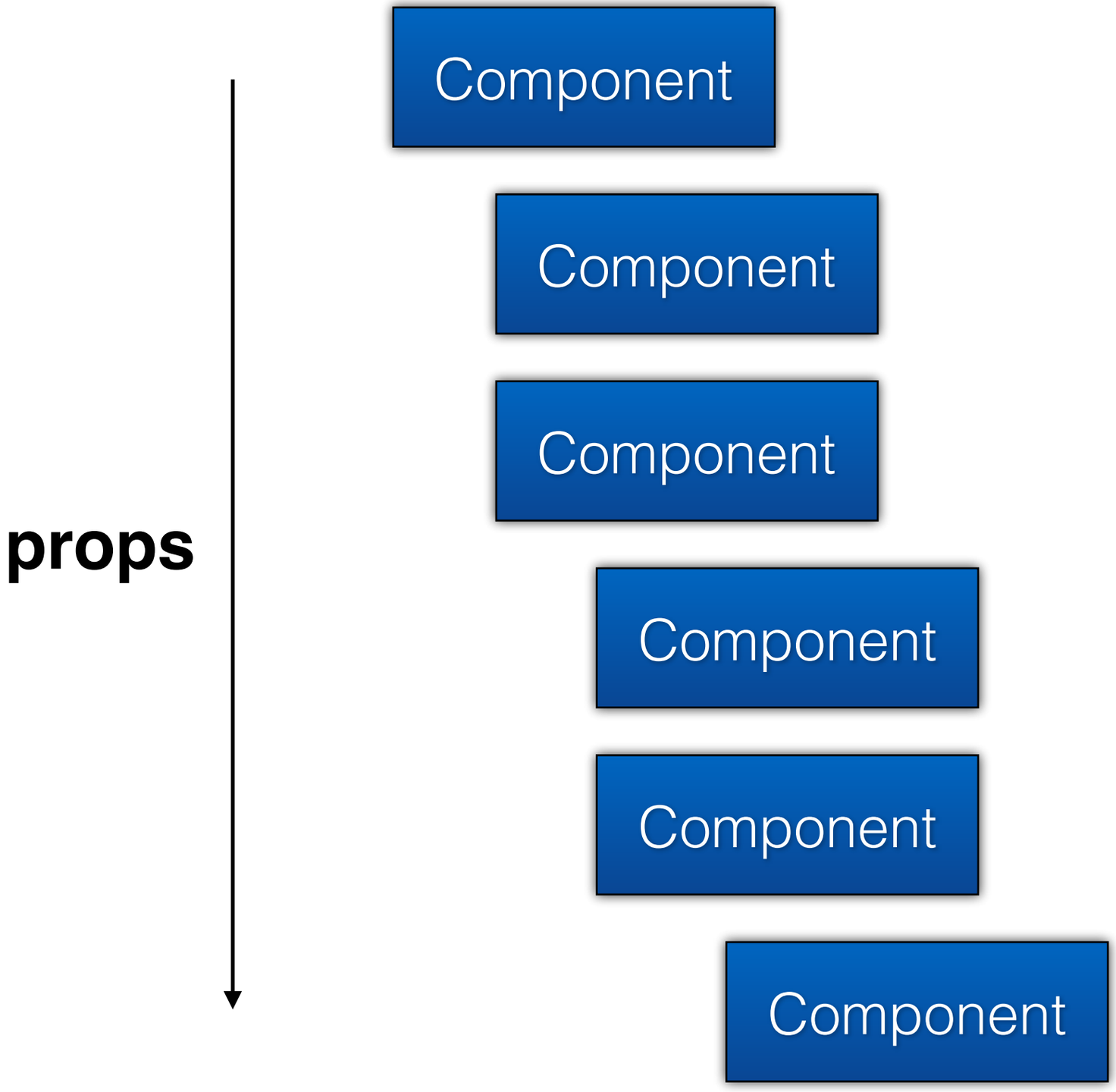
JSX Gotchas

- JSX is not HTML
- escapes all the strings (hard to inject custom HTML)
- Components' render() method must return a single React element, Fragment, array or null

Component properties

- **props** - data passed in from a parent component
- **state** - component's internal data
- **refs** - references to DOM elements (and other components) rendered by the component

Components


Component
Component
Component
Component
Component
Component
props

Component API

- **setState** - shallow merge of the provided state object into the current state

Component Lifecycle

http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/