React.js

Using, automating, testing, and profiting.

Where does React fit in?

Angular

Ember

Could use react here, but since Backbone mutates models with Model.set, it doesn't jibe well with React immutability

Backbone

jQuery

Flux

React

First: Just try it out

---

http://jsfiddle.net/vjeux/kb3gN/

JavaScript 1.7

```
var Hello = React.createClass({
    render: function() {
        return <div>Hello {this.props.name}</div>;
    }
});

React.render(<Hello name="World" />, document.body);

```

This is called “JSX” I also hated it at first. Give it a chance.

JSFiddle is nice, but…

React seed project

---

https://github.com/psbanka/react-talk

- [ ] **make build**: runs browserify and preps CSS and JS code for demo

git tag: **seed**

Build and test hello-world using the seed project

TESTING

I FIND YOUR LACK OF TESTS DISTURBING.

React seed project

---

https://github.com/psbanka/react-talk

- ☐ **make build**: runs browserify and preps CSS and JS code for demo
- ☐ **make test**: runs unit-tests once
- ☐ **make test-watch**: runs unit-tests whenever there's a change

git tag: **seed**

Jest: the React tester

- ☐ Uses jasmine
- ☐ Optional. Can still use Karma or whatever else
- ☐ Runs in node, not in PhanomJS
- ☐ Automatically mocks all required components
- ☐ Runs tests in complete isolation
- ☐ Runs tests in parallel

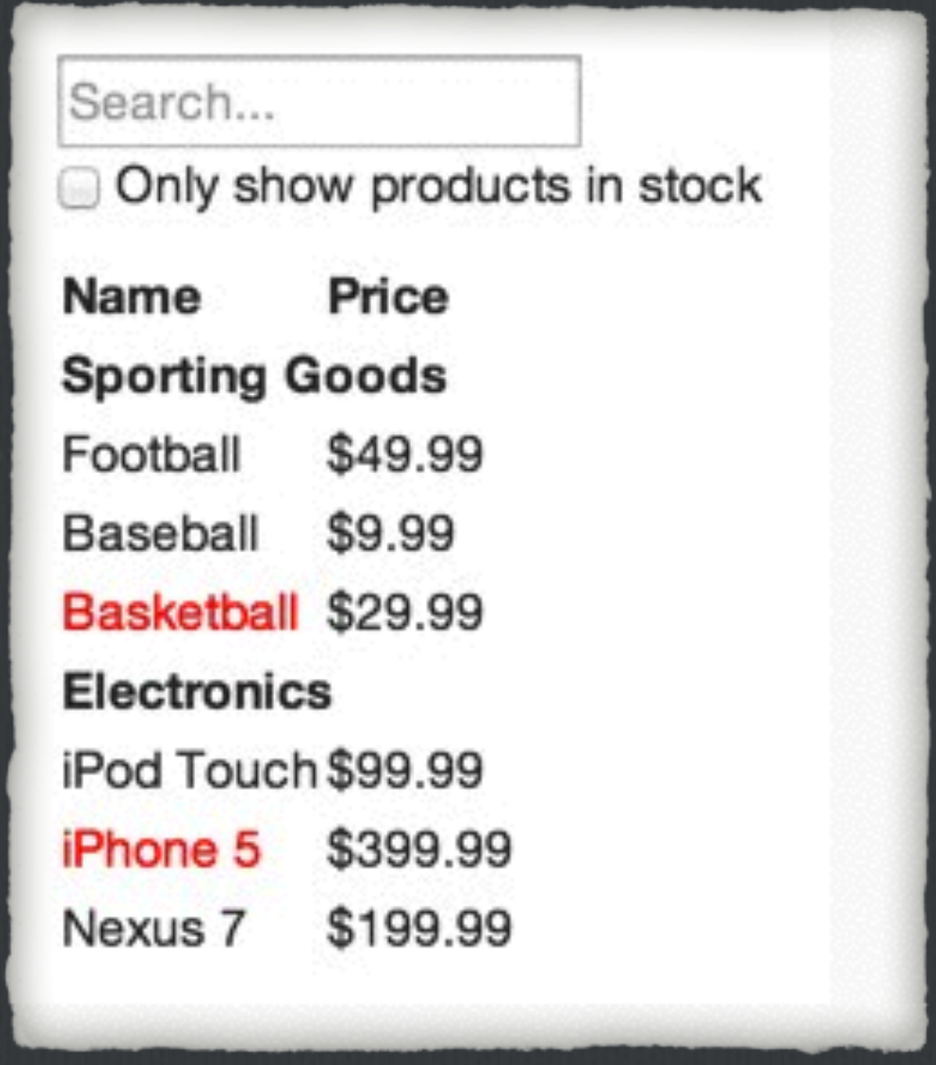
You have to tell it what NOT to mock

instead of React.render(),
use renderIntoDocument

Find the component
and test it!

```
/** @jsx React.DOM */

jest.dontMock('../hello.js');
var React = require('react/addons');
var Hello = require('../hello.js');
var TestUtils = React.addons.TestUtils;

describe('hello', function() {

    it('makes a proper greeting', function() {
        var hello = TestUtils.renderIntoDocument(
            <Hello name="John" />
        );

        var helloItem = TestUtils.findRenderedDOMComponentWithTag(hello, 'div')

        var domNode = helloItem.getDOMNode();
        expect(helloItem.getDOMNode().textContent).toEqual('Hello John');
    });

});
```

Let's do a project!

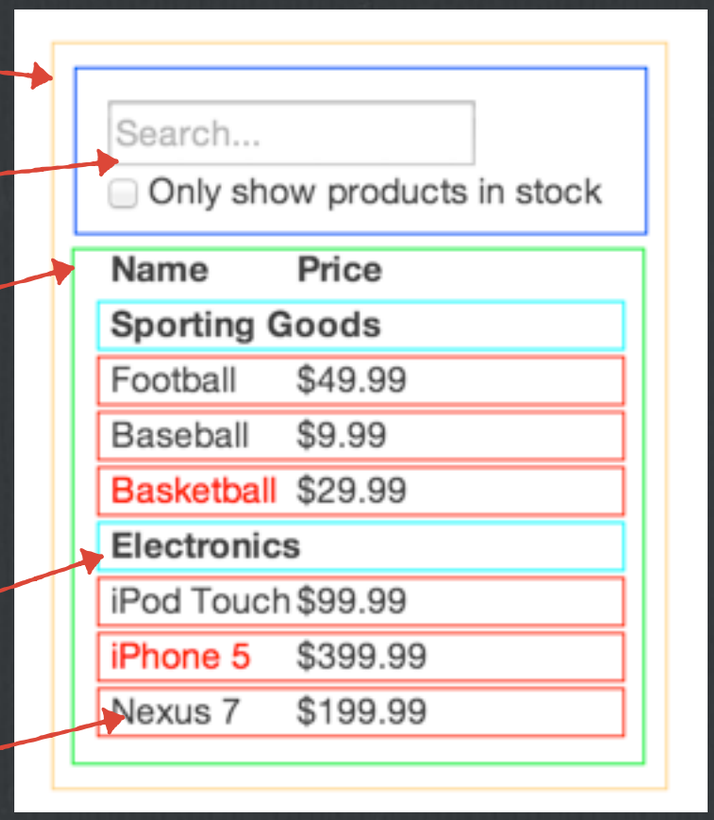


How do you make this work using React?

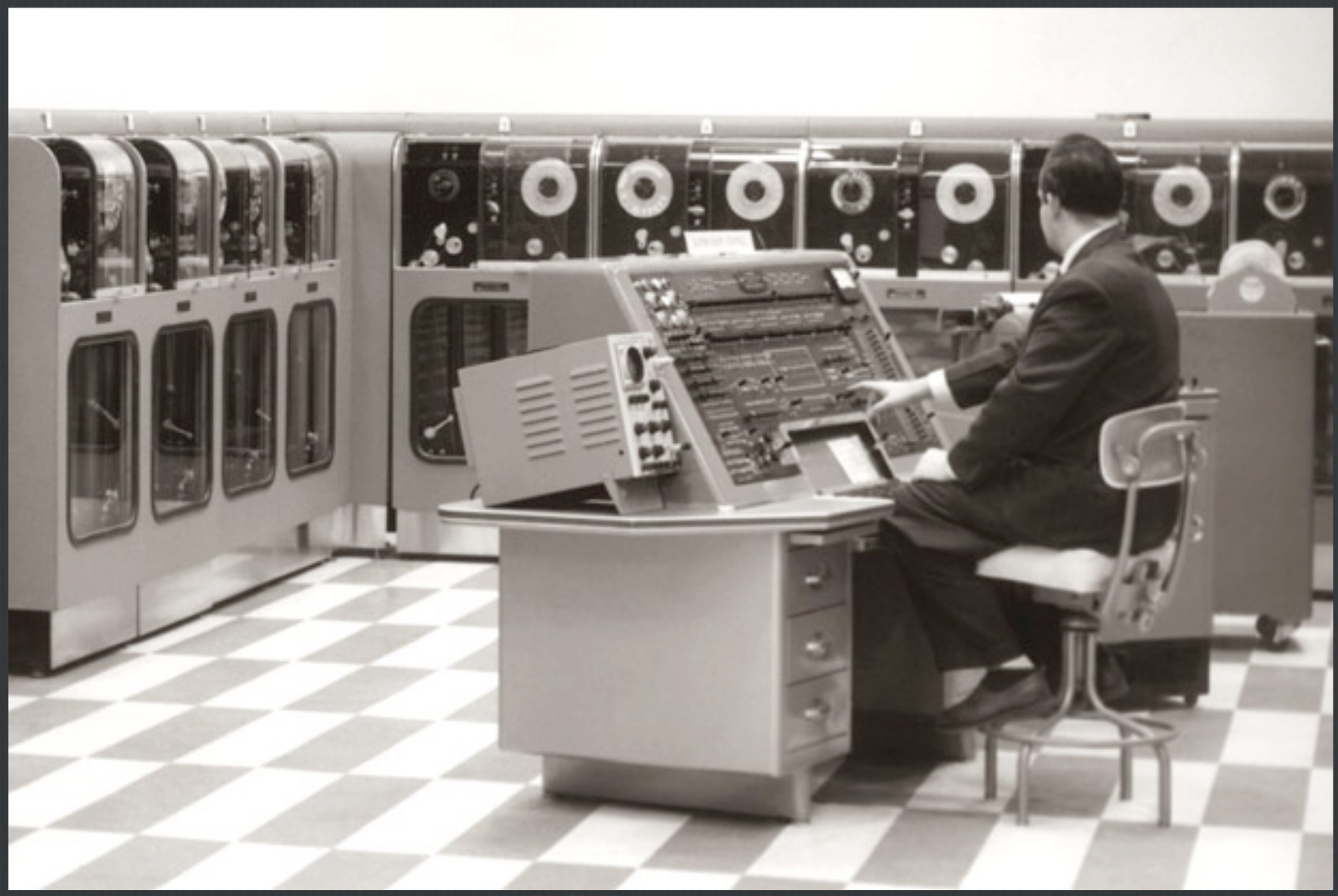
http://facebook.github.io/react/docs/thinking-in-react.html

Break it up

- [ ] FilterableProductTable
  - [ ] SearchBar
  - [ ] ProductTable
    - [ ] ProductCategoryRow
    - [ ] ProductRow



Let’s get to codin’!



Lets make it dynamic

State:

- ☐ Is it passed in from a parent via props? If so, it probably isn't state.
- ☐ Does it change over time? If not, it probably isn't state.
- ☐ Can you compute it based on any other state or props in your component? If so, it's not state.

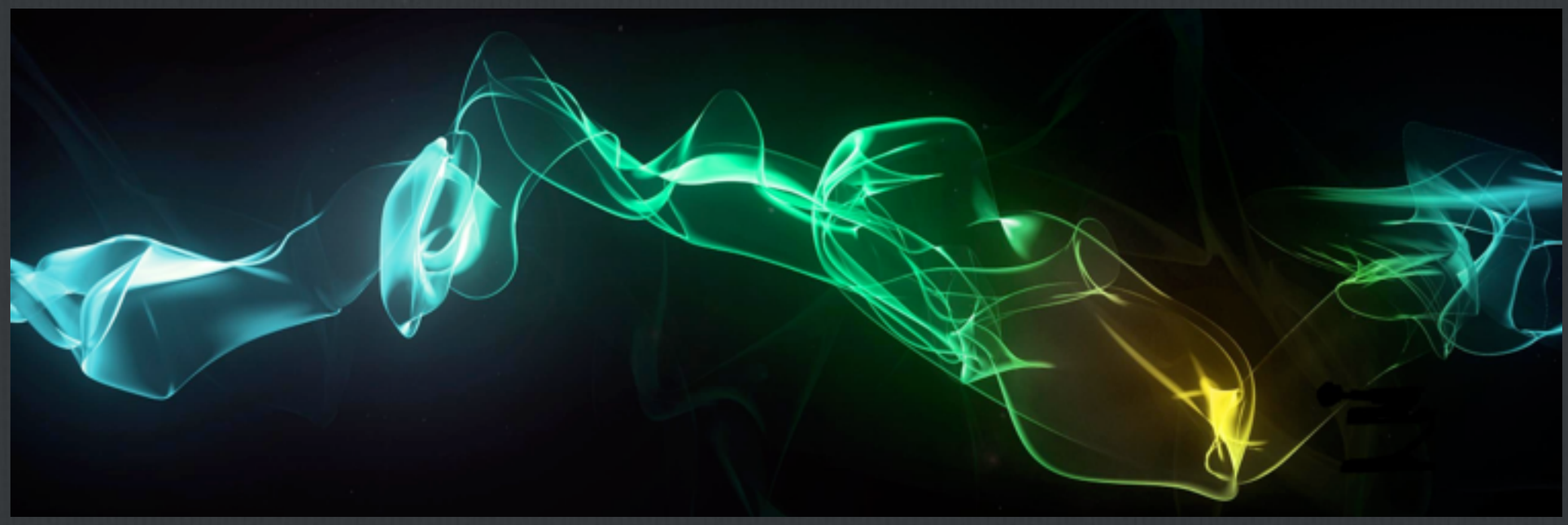
What is state in our case?

- [ ] Think of all of the pieces of data:
  - [ ] The original list of products
  - [ ] The search text the user has entered
  - [ ] The value of the checkbox
  - [ ] The filtered list of products

Make it dynamic

---

git tag: step-4



https://github.com/psbanka/react-talk/commit/2f2259c81a1e4a6cf3753f4c0615f7eee14526f0

React.js has one-way binding

React's one-way binding

JavaScript implements the von Neumann model of computing. Data only flows one way, that is, you can only assign a value on the right side of the equals sign to the identifier expression on the left side. By building a two-way data binding abstraction in JavaScript you are fighting this property of the language. This means that your program becomes harder to reason about correctness (because data flows in and out of modules in multiple directions) and even harder to think about performance (small changes may cause unexpected cascading updates in your application). Debugging these two-way data binding issues is almost impossible.

We need to add an event

---

git tag: **Step-5-complete**

https://github.com/psbanka/react-talk/commit/02538a060f233cf3f5a2a1858548236a74b19909

Summary

Reasons not to use React.js?

- [ ] Yet Another Library (130KB vs 84KB for jQuery)
- [ ] JSX (optional)
  - [ ] love it or hate it
  - [ ] can make the build a little more complex
  - [ ] no eslint support (but they are working on this)

Benefits use React.js?

- ☐ Incremental adoption
- ☐ Composition is king
- ☐ One-way data binding causes fewer problems
- ☐ Testing is easy and integral
- ☐ JSX: love it or hate it
- ☐ Performance