facebook

React

reactjs.org

Ben Newman (@benjamn)
Paul O’Shannessy (@zpao)

Components

`<div>, <span>`

`<ActionButton>, <Counter>`

Anatomy of a Component

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
var ActionButton = React.createClass({
  render: function() {





  }
});
```

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
var ActionButton = React.createClass({
  render: function() {





  }
});
```

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
var ActionButton = React.createClass({
  render: function() {
    return (
      <button class="ActionButton">
        <span>button text</span>
      </button>
    );
  }
});
```

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
var ActionButton = React.createClass({
  render: function() {
    return (
      <button class="ActionButton">
        <span>{this.props.text}</span>
      </button>
    );
  }
});
```

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
var ActionButton = React.createClass({
  render: function() {
    return (
      <button class="ActionButton" onClick={this.props.onAction}>
        <span>{this.props.text}</span>
      </button>
    );
  }
});
```

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
var ActionButton = React.createClass({
  render: function() {
    return (
      <button class="ActionButton" onClick={this.props.onAction}>
        <span>{this.props.text}</span>
      </button>
    );
  }
});
```

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
var ActionButton = React.createClass({
  render: function() {
    return (
      <button class="ActionButton" onClick={this.props.onAction}>
        <span>{this.props.text.toUpperCase()}</span>
      </button>
    );
  }
});
```

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
var ActionButton = React.createClass({
  render: function() {
    return (
      <button class="ActionButton" onClick={this.props.onAction}>
        <span>{this.props.text}</span>
      </button>
    );
  }
});
```

```
<ActionButton text="Book flight" onAction={someFunc} />
```

```
<Counter initialCount={4} />
```

```
var Counter = React.createClass({
  getInitialState: function() {
    return {count: this.props.initialCount};
  },
  addToCount: function(delta) {
    this.setState({count: this.state.count + delta})
  },
  render: function() {
    return (
      <div>
        <h3>Count: {this.state.count}</h3>
        <ActionButton text="+1" onAction={this.addToCount.bind(this, 1)} />
        <ActionButton text="-1" onAction={this.addToCount.bind(this, -1)} />
      </div>
    );
  }
});
```

```
<Counter initialCount={4} />
```

```
var Counter = React.createClass({
  getInitialState: function() {
    return {count: this.props.initialCount};
  },
  addToCount: function(delta) {
    this.setState({count: this.state.count + delta})
  },
  render: function() {
    return (
      <div>
        <h3>Count: {this.state.count}</h3>
        <ActionButton text="+1" onAction={this.addToCount.bind(this, 1)} />
        <ActionButton text="-1" onAction={this.addToCount.bind(this, -1)} />
      </div>
    );
  }
});
```

```
<Counter initialCount={4} />
```

```
var Counter = React.createClass({
  getInitialState: function() {
    return {count: this.props.initialCount};
  },
  addToCount: function(delta) {
    this.setState({count: this.state.count + delta})
  },
  render: function() {
    return (
      <div>
        <h3>Count: {this.state.count}</h3>
        <ActionButton text="+1" onAction={this.addToCount.bind(this, 1)} />
        <ActionButton text="-1" onAction={this.addToCount.bind(this, -1)} />
      </div>
    );
  }
});
```

```
<Counter initialCount={4} />
```

```
var Counter = React.createClass({
  getInitialState: function() {
    return {count: this.props.initialCount};
  },
  addToCount: function(delta) {
    this.setState({count: this.state.count + delta})
  },
  render: function() {
    return (
      <div>
        <h3>Count: {this.state.count}</h3>
        <ActionButton text="+1" onAction={this.addToCount.bind(this, 1)} />
        <ActionButton text="-1" onAction={this.addToCount.bind(this, -1)} />
      </div>
    );
  }
});
```

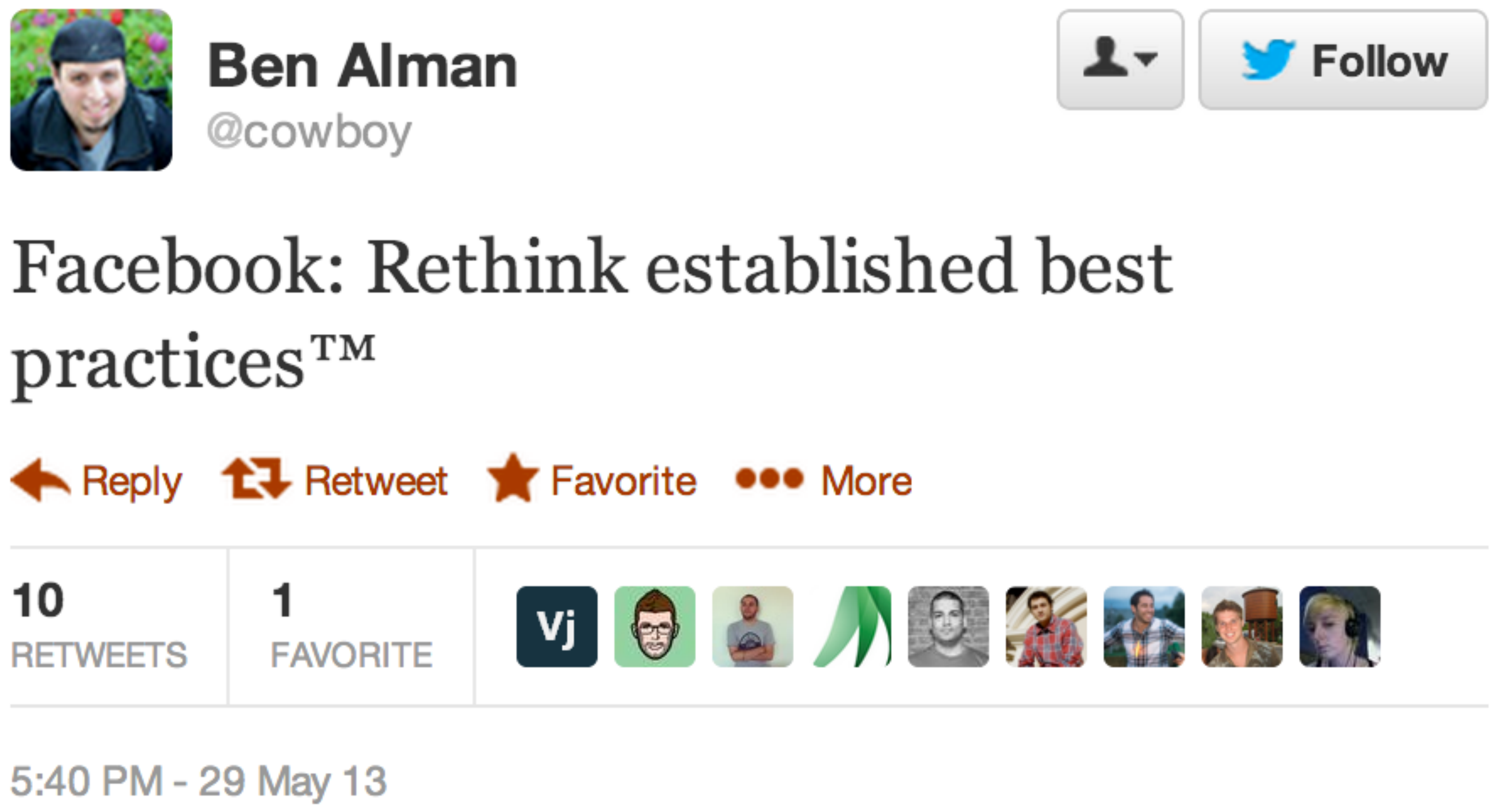
```
<Counter initialCount={4} />
```

```
var Counter = React.createClass({
  getInitialState: function() {
    return {count: this.props.initialCount};
  },
  addToCount: function(delta) {
    this.setState({count: this.state.count + delta})
  },
  render: function() {
    return (
      <div>
        <h3>Count: {this.state.count}</h3>
        <ActionButton text="+1" onAction={this.addToCount.bind(this, 1)} />
        <ActionButton text="-1" onAction={this.addToCount.bind(this, -1)} />
      </div>
    );
  }
});
```

```
<Counter initialCount={4} />
```

What makes React different?

1. Components, not templates
2. Re-render on update
3. Virtual DOM (and events)

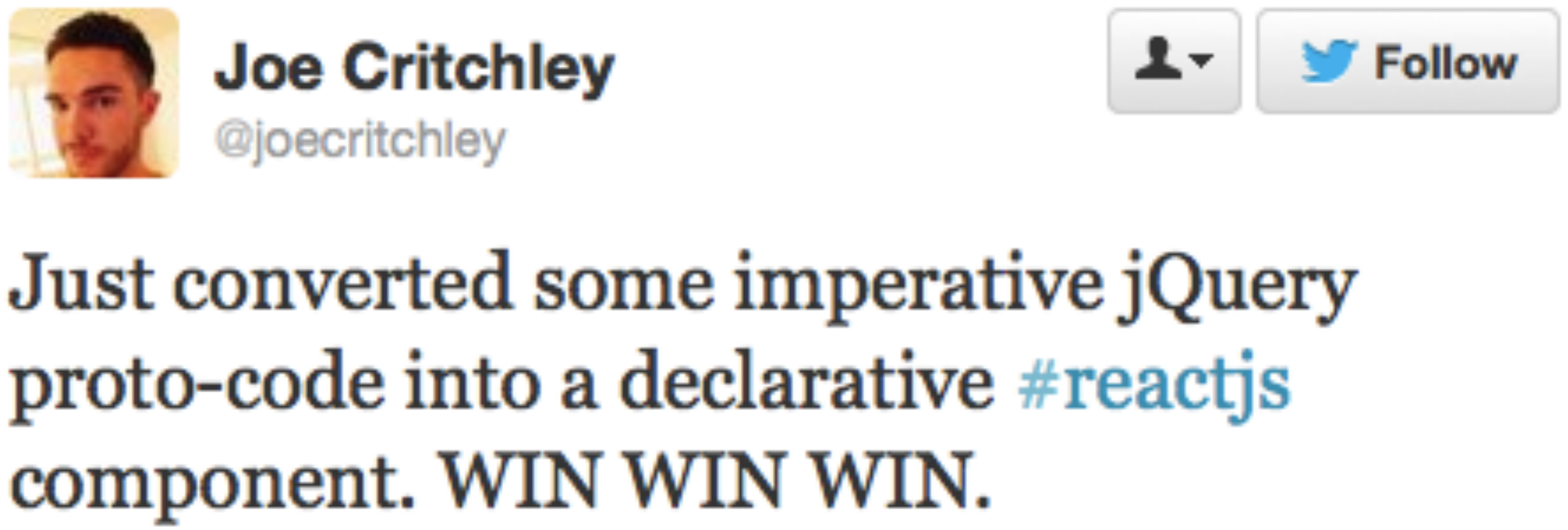


Ben Alman
@cowboy

Facebook: Rethink established best practices™

10 RETWEETS
1 FAVORITE

5:40 PM - 29 May 13


Joe Critchley
@joecritchley

Just converted some imperative jQuery proto-code into a declarative #reactjs component. WIN WIN WIN.

1. Components, not templates

Separation of concerns:

Reduce coupling, increase cohesion.

Coupling is:

“The degree to which each program module relies on each of the other modules.”

http://en.wikipedia.org/wiki/Coupling_(computer_science)

Cohesion is:

“The degree to which elements of a module belong together.”

http://en.wikipedia.org/wiki/Cohesion_(computer_science)

“View model” tightly couples template to display logic.

[{“price”: “7.99”, “product”: “Back scratcher”, “tableRowColor”: “rgba(0, 0, 0, 0.5)”}]

Templates separate technologies, not concerns

React components are loosely coupled and highly cohesive

```
var Counter = React.createClass({
  getInitialState: function() {
    return {count: this.props.initialCount};
  },
  addToCount: function(delta) {
    this.setState({count: this.state.count + delta})
  },
  render: function() {
    return (
      <div>
        <h3>Count: {this.state.count}</h3>
        <ActionButton text="+1" onAction={this.addToCount.bind(this, 1)} />
        <ActionButton text="-1" onAction={this.addToCount.bind(this, -1)} />
      </div>
    );
  }
});
```

```
<Counter initialCount={4} />
```

2. Re-render on every change

```
var Counter = React.createClass({
  getInitialState: function() {
    return {count: this.props.initialCount};
  },
  addToCount: function(delta) {
    this.setState({count: this.state.count + delta})
  },
  render: function() {
    return (
      <div>
        <h3>Count: {this.state.count}</h3>
        <ActionButton text="+1" onAction={this.addToCount.bind(this, 1)} />
        <ActionButton text="-1" onAction={this.addToCount.bind(this, -1)} />
      </div>
    );
  }
});
```

```
<Counter initialCount={4} />
```

Best analogy: Website from 1994

Data changing over time is the root of all evil.

Re-rendering on every change makes things simple.

Every place data is displayed is guaranteed to be up-to-date.

Re-rendering on every change makes things simple.

No magical data binding.

Re-rendering on every change makes things simple.

No model dirty checking.

Re-rendering on every change makes things simple.

No more explicit DOM operations – everything is declarative.

3. Virtual DOM

Won’t rerendering be as slow as molasses?!

React has a virtual DOM (and events system).

Optimized for performance and memory footprint

On every update…

- React builds a new virtual DOM subtree
- …diffs it with the old one
- …computes the minimal set of DOM mutations and puts them in a queue
- …and batch executes all updates

It’s fast!

Because the DOM is slow!

It’s fast!

Computes minimal DOM operations

It’s fast!

Batched reads and writes for optimal DOM performance

It’s fast!

Usually faster than manual DOM operations

It’s fast!

Automatic top-level event delegation (with cross-browser HTML5 events)

It’s fast!

Can do all this at 60fps, even in a (non-JIT) UIWebView on the iPhone.

Why Should **YOU** Use React?

- Can be used for parts of your application
- Plays well with other libraries and technologies (meteor, rails, node)
- Components allow you to split work easily

Learn more and get involved

- http://reactjs.org
- #reactjs on Freenode IRC
- reactjs on Google Groups
- www.facebook.com/careers

More Links

- react-meteor: https://github.com/benjamn/react-meteor
- <ActionButton> demo: http://jsfiddle.net/zpao/EFhy4/
- <Clicker> demo: http://jsfiddle.net/zpao/fk5Pc/

facebook