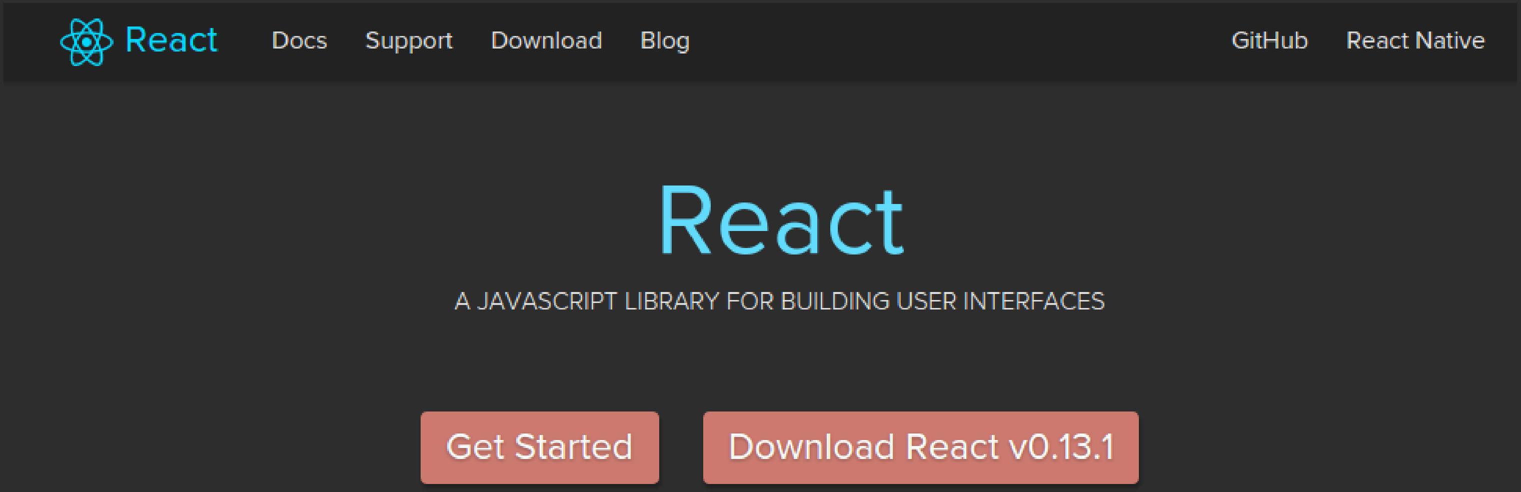

React Docs Support Download Blog GitHub React Native

React

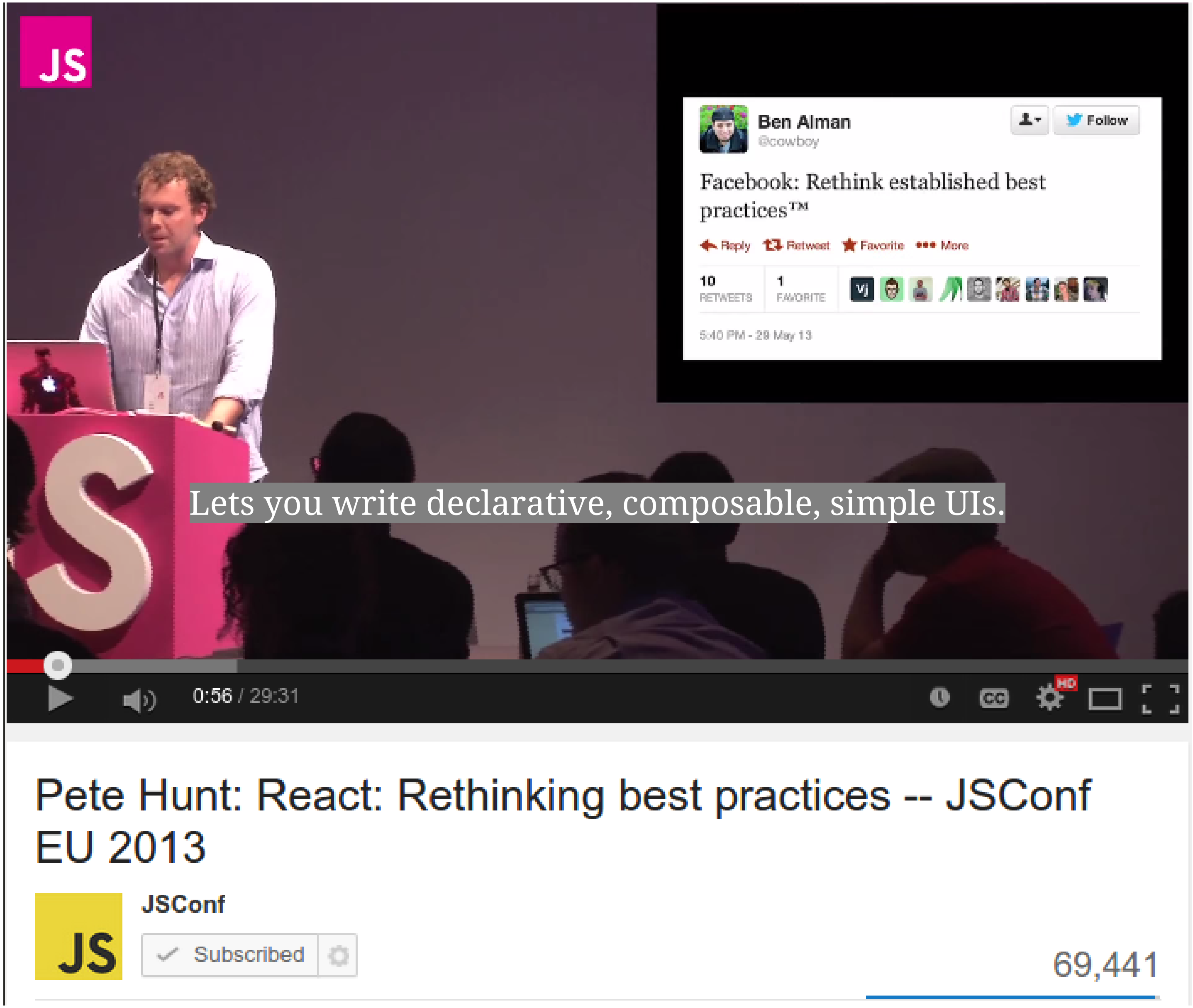
A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Get Started

Download React v0.13.1

Some React Users!

- Facebook: comments, chat, search, notifications... + new JS development. React Native: f8 app, Facebook Groups, Facebook Ads Manager.
- Instagram
- AirBnB: airbnb.com/resolutions
- Asana: React + typescript
- Atlassian: Hipchat
- BBC: Homepage
- Brackets: Project Tree (code editor)
- Codeacademy: Learning environment (slides)
- Flipboard: created react-canvas
- Khan Academy: equation editor, student exercises, admin panels.
- Netflix
- Pivotal: Pivotal UI
- Reddit: reddit gifts
- The New York Times: 2014 Red Carpet Project, Festival de Cannes, world cup
- WhatsApp: web app
- Wired: using in articles for live updates
- XKCD: xkcloud
- Yahoo: Yahoo! mail
- Zendesk



Ben Alman
@cowboy

Follow

Facebook: Rethink established best practices™

Reply Retweet Favorite More

10 RETWEETS 1 FAVORITE

5:40 PM - 28 May 13

Lets you write declarative, composable, simple UIs.

0:56 / 29:31

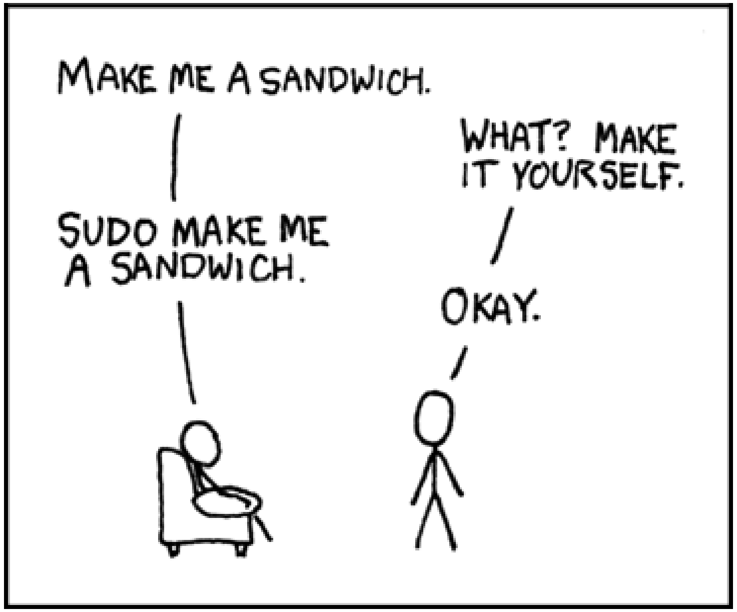
Pete Hunt: React: Rethinking best practices -- JSConf EU 2013

JSConf

Subscribed

69,441

Declarative



Declarative vs. Imperative

- Focus on *what* needs to be done (not implementation details).

```
var numbers = [1,2,3,4,5];
```

```
// triple each number in the list and only get the even ones.
```

```
// imperative
var tripledEvens = [];
for (var i = 0; i < numbers.length; i++) {
    var newNumber = numbers[i] * 3;
    if (newNumber % 2 === 0) {
        tripledEvens.push(newNumber);
    }
}
```

```
// declarative/functional
function triple(n) { return n * 3; }
function evens(n) { return n % 2 === 0; }
var tripledEvens = numbers.map(triple).filter(evens);
```

- Allows us to talk about problems at a higher level (abstraction).

```
-- Declarative: describe what should happen

SELECT * from dogs
INNER JOIN owners
WHERE dogs.owner_id = owners.id
```

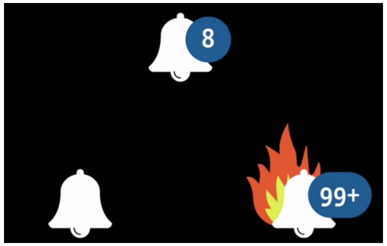
```
// Imperative: explain how it happens

// dogs = [{name: 'Fido', owner_id: 1}, {...}, ... ]
// owners = [{id: 1, name: 'Bob'}, {...}, ...]
var dogsWithOwners = [];
var dog, owner;

for (var di = 0; di < dogs.length; di++) {
    dog = dogs[di];

    for (var oi = 0; oi < owners.length; oi++) {
        owner = owners[oi];
        if (owner && dog.owner_id === owner.id) {
            dogsWithOwners.push({
                dog: dog,
                owner: owner
            });
        }
    }}
}
```

- Easier to reason, maintain, test

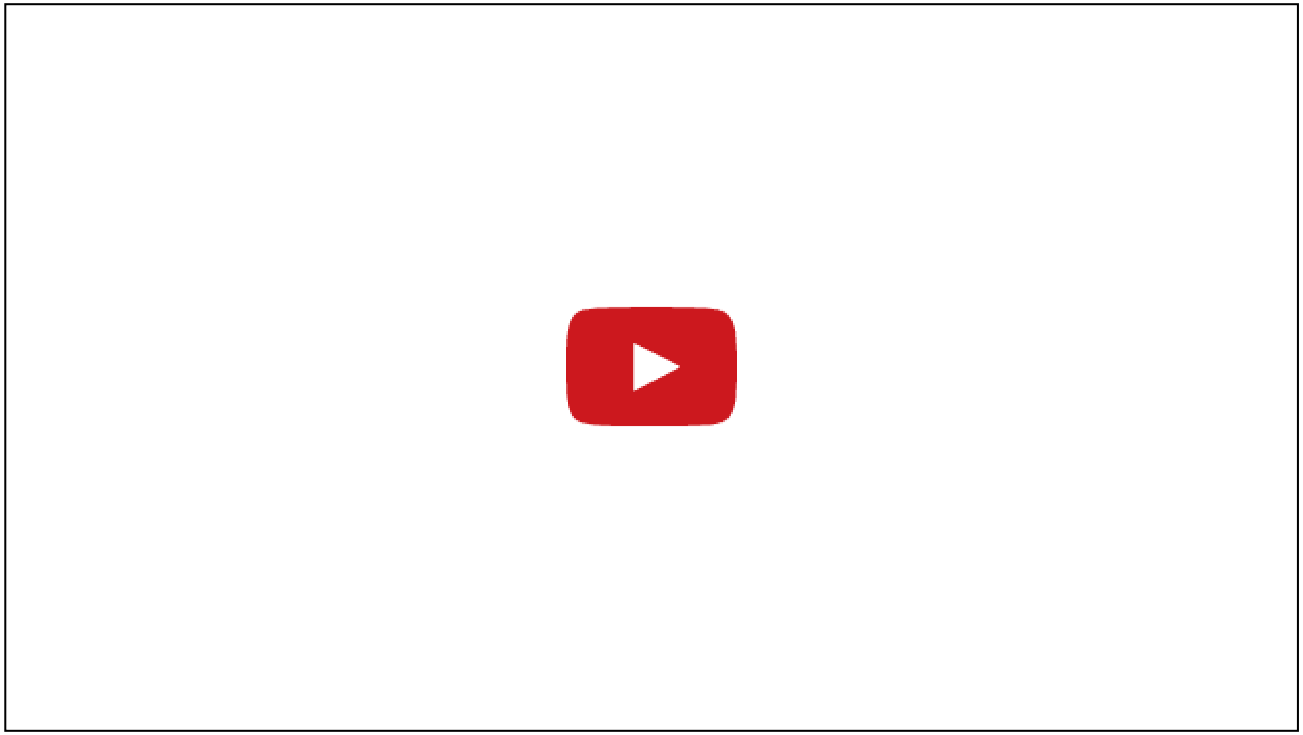
http://latentflip.com/imperative-vs-declarative/

So what? We use imperatives to change the DOM.

```
var p = document.createElement("p");
document.body.appendChild(elements);
document.body.removeChild(elements);
document.body.addChild(elements123);
// move elements, modify elements...
```



```
// Example of a notification UI

if (count > 99) {
    if (!hasFire()) {
        addFire();
    }
} else {
    if (hasFire()) {
        removeFire();
    }
}
if (count === 0) {
    if (hasBadge()) {
        removeBadge();
    }
    return;
}
if (!hasBadge()) {
    addBadge();
}
var countText = count > 99 ? '99+' : count.toString();
getBadge().setText(countText);

// Retained mode (DOM, svg)
// Can start at 10:10 in the video.
```



React wraps the imperative DOM with a declarative layer

```
if (count === 0) {
```

```
    return <div className="bell"></div>;
```

```
} else if (count < 99) {
```

```
    return (
        <div className="bell">
            <span className="badget">{count}</span>
        </div>
    );
```

```
} else {
```

```
    return (
        <div className="bell onFire">
            <span className="badge">99+</span>
        </div>
    );
}
```

Virtual DOM

Diff Alg

```
var MyComponent = React.createClass({
    // What is returned is just a javascript object.
    render: function() {
        if (this.props.first) {
            return <div className="first"><span>A Span</span></div>;
        } else {
            return <div className="second"><p>A Paragraph</p></div>;
        }
    }
});
```

```
// None to first
Create node: <div className="first"><span>A Span</span></div>
```

```
// First to second
Replace attribute: className="first" by className="second"
Replace node: <span>A Span</span> by <p>A Paragraph</p>
```

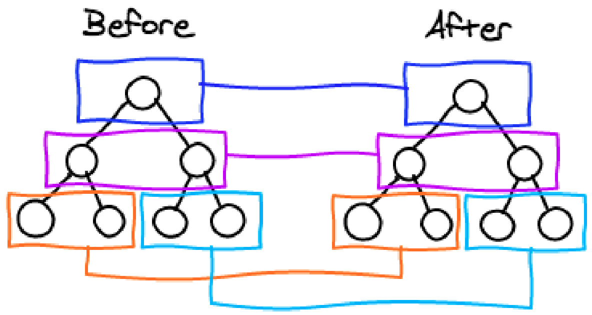
```
// Second to none
Remove node: <div className="second"><p>A Paragraph</p></div>
```

React finds the minimal set of operations between the two render trees (previous and current state).

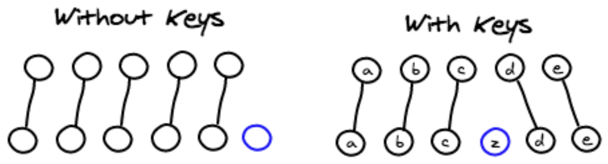
http://calendar.perfplanet.com/2013/diff/

Optimizations to go from `O(n^3)` to `O(n)`

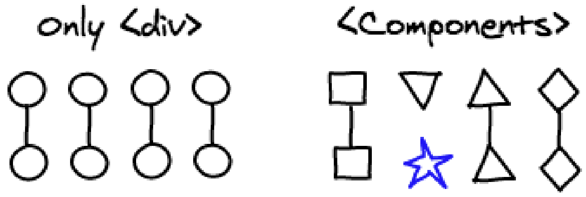
Level by Level



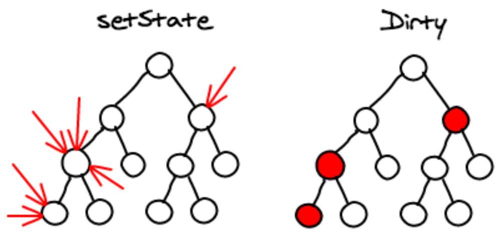
List Diffing



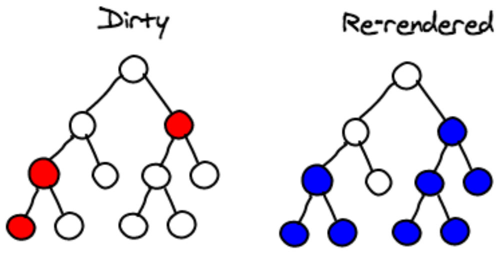
Component Diffing



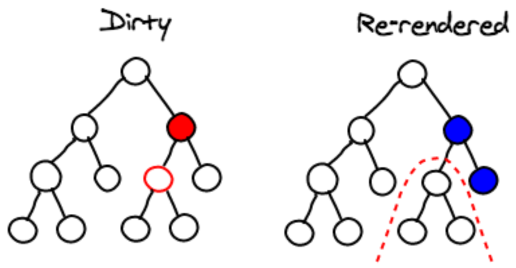
Batching



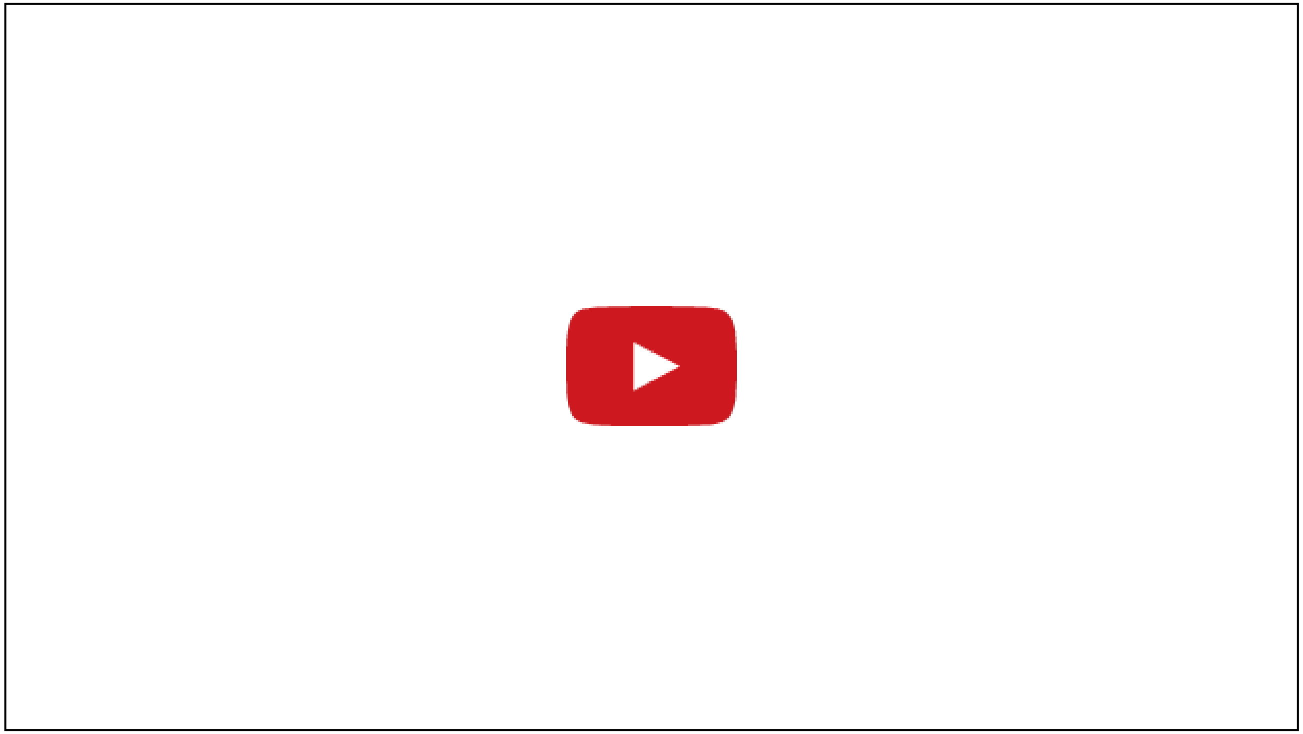
Sub-tree Rendering



Selective Sub-tree Rendering (`shouldComponentUpdate`)



Demo (4:47)



https://youtu.be/z5e7kWSHWTg?t=4m47s

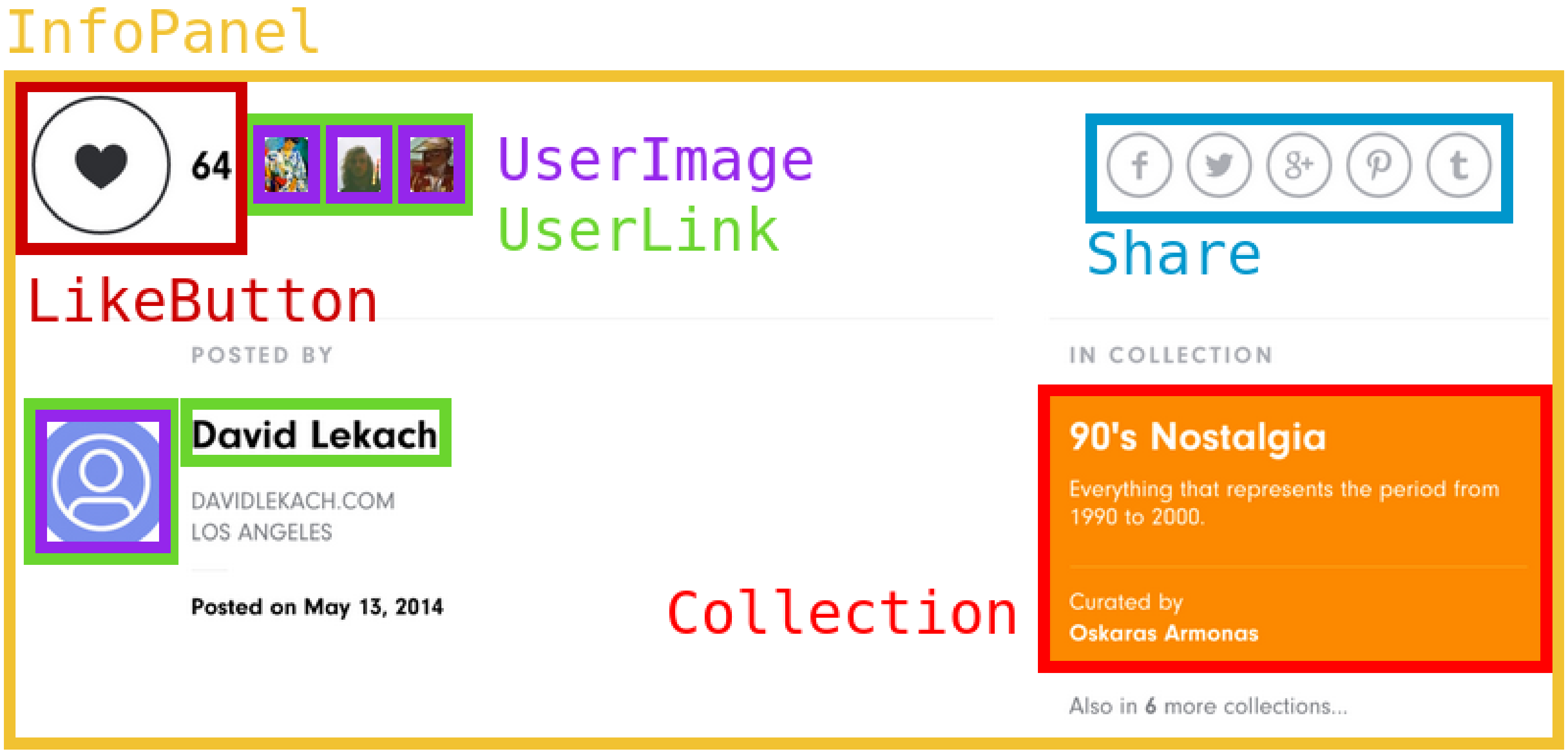
React Inspired

- Angular2 rendering engine will use vDOM
- Ember Glimmer engine will use vDOM (PR)
- React Native: framework to write iOS (and later Android) apps using React.
- ComponentKit: a iOS library by Facebook to make apps similar to React
- React Canvas: adds the ability for React components to render to

Similar to React

- Riot.js
- Mithril
- Om: ClojureScript interface to React
- Mercury: (uses `virtual-dom`)
- Elm-html: HTML in Elm (uses `virtual-dom`)

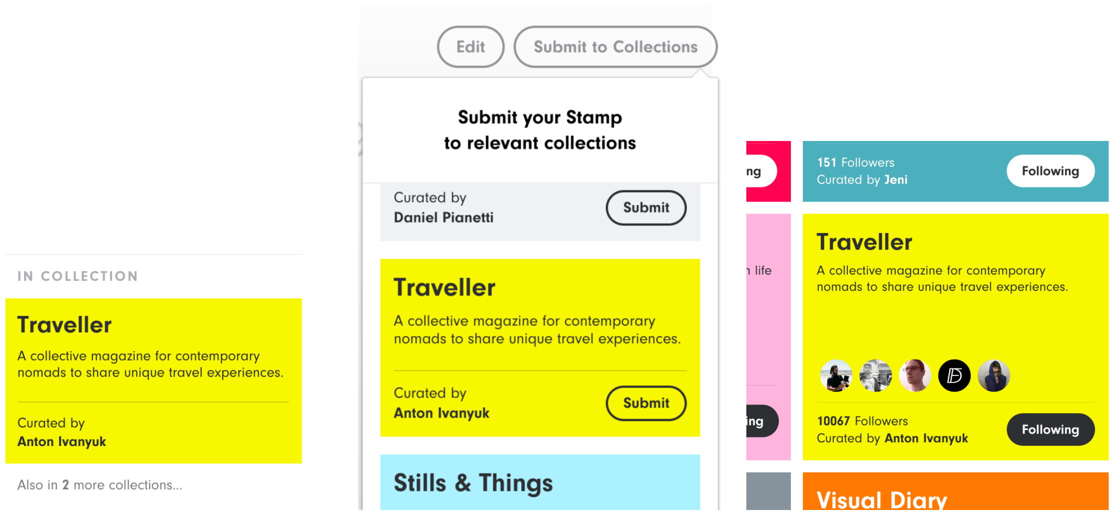
Composition



Components: similar to creating your own element (`h1`, `select`, `img`)

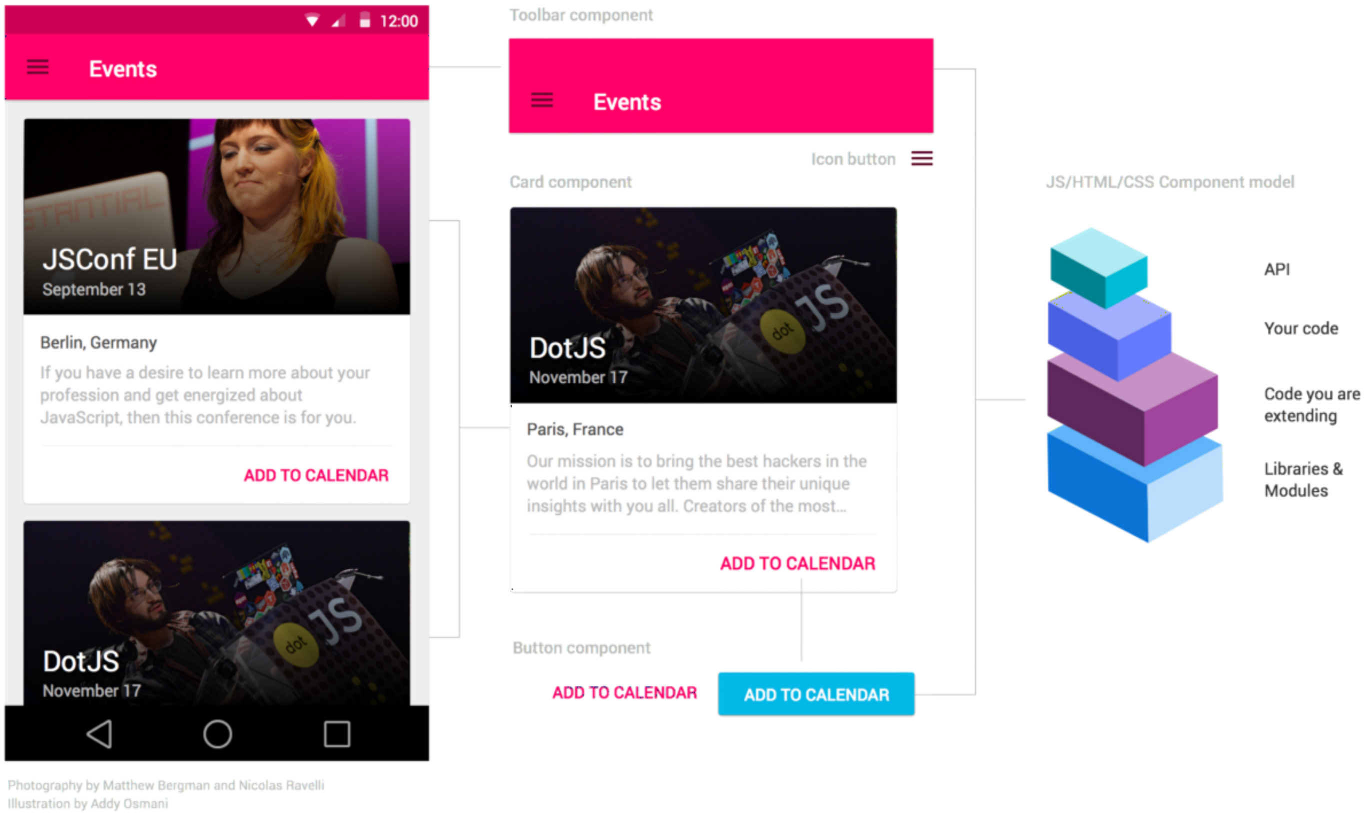
- An element has it's own structure, default styles, and functionality (no interference between components).
- A component uses JSX/HTML, CSS, and JS.

```
// Allows you to use a component with a simple require
define(['components/toolbar'], function (Toolbar) {})

// commonjs/node
var Toolbar = require('components/toolbar');

// es6 modules
import Toolbar from 'components/toolbar';
```


Edit
Submit to Collections
Submit your Stamp to relevant collections
Curated by Daniel Pianetti
Submit
Traveller
A collective magazine for contemporary nomads to share unique travel experiences.
Curated by Anton Ivanyuk
Submit
Stills & Things
IN COLLECTION
Traveller
A collective magazine for contemporary nomads to share unique travel experiences.
Curated by Anton Ivanyuk
Also in 2 more collections...
151 Followers
Curated by Jeni
Following
Traveller
A collective magazine for contemporary nomads to share unique travel experiences.
10067 Followers
Curated by Anton Ivanyuk
Following
Visual Diary

http://slides.com/danabramov/components-react-flux-wip#/16

An app can be broken down into simple components.


12:00
Events
JSConf EU
September 13
Berlin, Germany
If you have a desire to learn more about your profession and get energized about JavaScript, then this conference is for you.
ADD TO CALENDAR
DotJS
November 17
Toolbar component
Events
Icon button
Card component
DotJS
November 17
Paris, France
Our mission is to bring the best hackers in the world in Paris to let them share their unique insights with you all. Creators of the most...
ADD TO CALENDAR
Button component
ADD TO CALENDAR
ADD TO CALENDAR
JS/HTML/CSS Component model
API
Your code
Code you are extending
Libraries & Modules
Photography by Matthew Bergman and Nicolas Ravelli
Illustration by Addy Osmani

React components, Ember.Components, Angular directives, Web Components…

https://medium.com/@addyosmani/javascript-application-architecture-on-the-road-to-2015-d8125811101b

Templates (logic in HTML)

```
// Handlebars
<ul id="comments">
    {{#each comments}}
        <li>{{text}}</li>
    {{/each}}
</ul>
```

```
// Kendo
<script id="javascriptTemplate" type="text/x-kendo-template">
    <ul>
    # for (var i = 0; i < data.length; i++) { #
        <li>#= data[i] #</li>
    # } #
    <ul>
<script>
```

```
// Manual DOM creation
var out = "<ul>";
    for(var i=0, l=items.length; i<l; i++) {
        out = out + "<li>" + options.fn(items[i]) + "</li>";
    }
return out + "</ul>";
```

Write HTML-like syntax for structure (rather than learning another DSL)

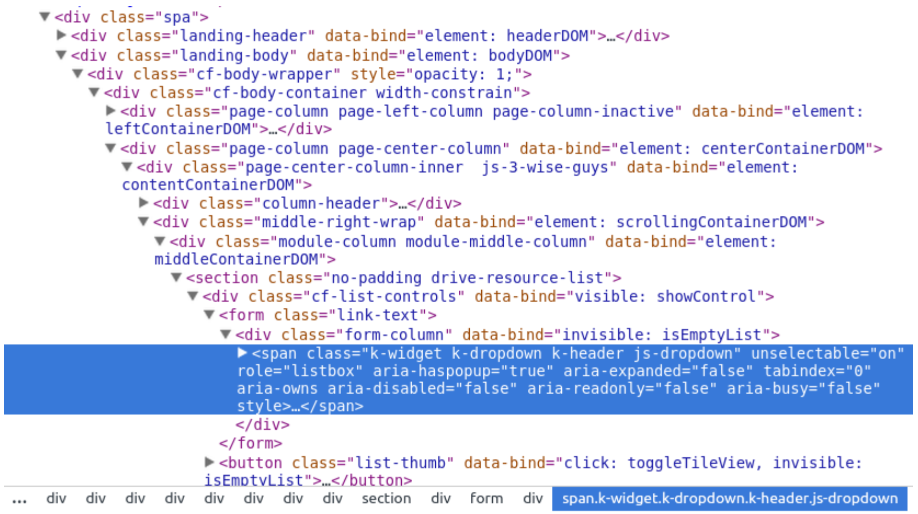
```
// React (ES5)
return (
    var items = this.props.items;
    var listItems = items.map(function(i) {
        return <li>{i}</li>;
    });
    <ul>
        {listItems}
    </ul>
);
```

```
// React (ES6)
return (
    <ul>
        {this.props.items.map((i) => <li>{i}</li>)}
    </ul>
);
```

```
var Avatar = React.createClass({
  render: function() {
    return (
      <div>
        <ProfilePic username={this.props.username} />
        <ProfileLink username={this.props.username} />
      </div>
    );
  }
});
var ProfilePic = React.createClass({
  render: function() {
    return (
      <img src={'http://graph.facebook.com/' + this.props.username + '/picture'} />
    );
  }
});
var ProfileLink = React.createClass({
  render: function() {
    return (
      <a href={'http://www.facebook.com/' + this.props.username}>
        {this.props.username}
      </a>
    );
  }
});
React.render(
  <Avatar username="pwh" />,
  document.getElementById('example')
);
```

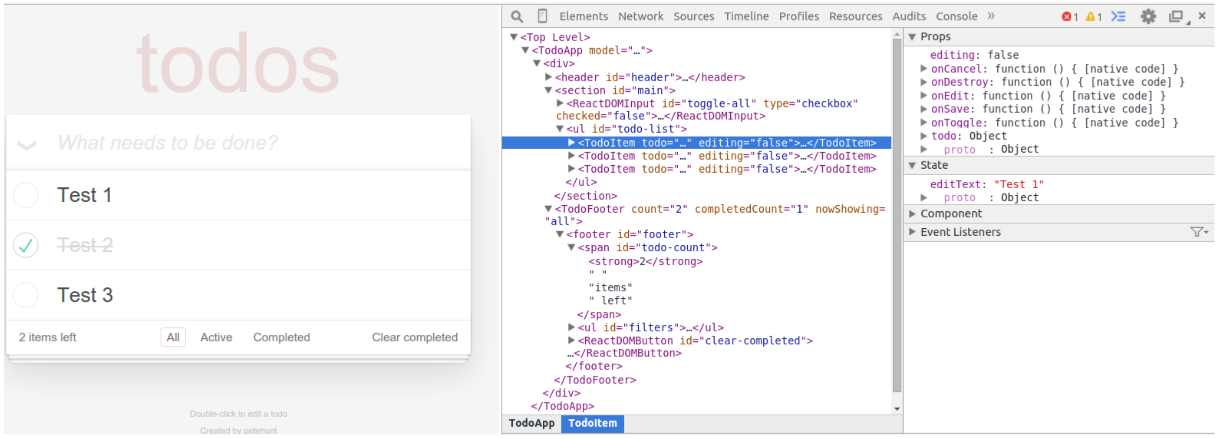
https://facebook.github.io/react/docs/multiple-components.html#composition-example

React Dev Tools (Github)



```
▼<div class="spa">
  ▶<div class="landing-header" data-bind="element: headerDOM">…</div>
  ▼<div class="landing-body" data-bind="element: bodyDOM">
    ▼<div class="cf-body-wrapper" style="opacity: 1;">
      ▼<div class="cf-body-container width-constrain">
        ▶<div class="page-column page-left-column page-column-inactive" data-bind="element:
        leftContainerDOM">…</div>
        ▼<div class="page-column page-center-column" data-bind="element: centerContainerDOM">
          ▼<div class="page-center-column-inner  js-3-wise-guys" data-bind="element:
          contentContainerDOM">
            ▶<div class="column-header">…</div>
            ▼<div class="middle-right-wrap" data-bind="element: scrollingContainerDOM">
              ▼<div class="module-column module-middle-column" data-bind="element:
              middleContainerDOM">
                ▼<section class="no-padding drive-resource-list">
                  ▼<div class="cf-list-controls" data-bind="visible: showControl">
                    ▼<form class="link-text">
                      ▼<div class="form-column" data-bind="invisible: isEmptyList">
                        ▶<span class="k-widget k-dropdown k-header js-dropdown" unselectable="on"
                        role="listbox" aria-haspopup="true" aria-expanded="false" tabindex="0"
                        aria-owns aria-disabled="false" aria-readonly="false" aria-busy="false"
                        style>…</span>
                        </div>
                      </form>
                    ▶<button class="list-thumb" data-bind="click: toggleTileView, invisible:
                    isEmptyList">…</button>
```

… div div div div div div div div section div form div span.k-widget.k-dropdown.k-header.js-dropdown



```
▼<Top Level>
  ▼<TodoApp model="…">
    ▼<div>
      ▶<header id="header">…</header>
      ▼<section id="main">
        ▶<ReactDOMInput id="toggle-all" type="checkbox"
        checked="false">…</ReactDOMInput>
        ▼<ul id="todo-list">
          ▶<TodoItem todo="…" editing="false">…</TodoItem>
          ▶<TodoItem todo="…" editing="false">…</TodoItem>
          ▶<TodoItem todo="…" editing="false">…</TodoItem>
        </ul>
      </section>
      ▼<TodoFooter count="2" completedCount="1" nowShowing=
      "all">
        ▼<footer id="footer">
          ▼<span id="todo-count">
              <strong>2</strong>
              " "
              "items"
              " left"
            </span>
          ▶<ul id="filters">…</ul>
          ▶<ReactDOMButton id="clear-completed">
          …</ReactDOMButton>
        </footer>
      </TodoFooter>
    </div>
  </TodoApp>
```

```
▼ Props
    editing: false
  ▶ onCancel: function () { [native code] }
  ▶ onDestroy: function () { [native code] }
  ▶ onEdit: function () { [native code] }
  ▶ onSave: function () { [native code] }
  ▶ onToggle: function () { [native code] }
  ▶ todo: Object
  ▶   proto  : Object
▼ State
    editText: "Test 1"
  ▶   proto  : Object
▶ Component
▶ Event Listeners
```

JSX Specification

- HTML to JSX Converter

```
<div class="awesome" style="border: 1px solid red">
  <label for="name">Enter your name: </label>
  <input type="text" id="name" />
</div>

<div className="awesome" style={{border: '1px solid red'}}>
  <label htmlFor="name">Enter your name: </label>
  <input type="text" id="name" />
</div>
```

- Supported tags/attributes
- Differences from HTML (reserved words in JS; React matches the JS DOM API)
  - `for` -> `htmlFor`,
  - `class` -> `className`

JSX Complier

- It's just syntax. You can use anything in javascript!
- JSX Complier

Turns

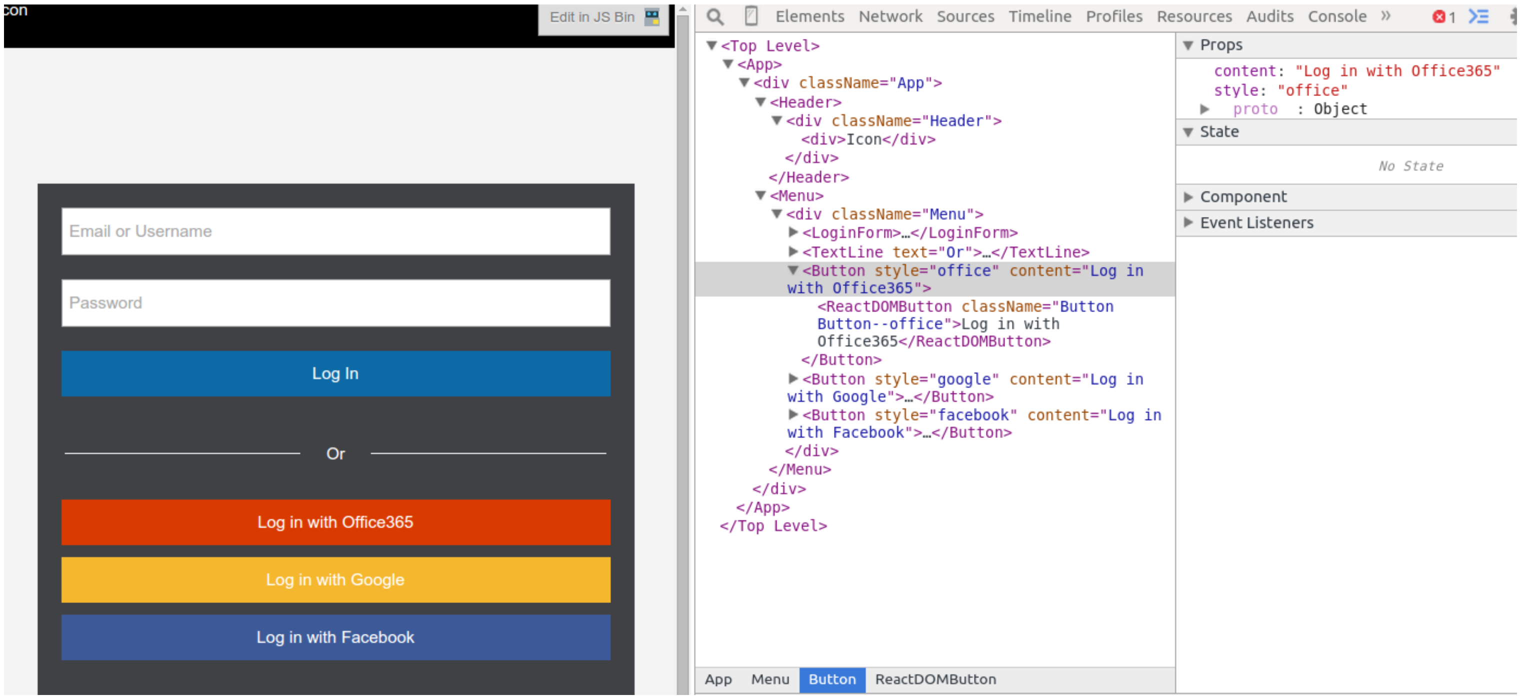
```
<div>Hello {this.props.name}</div>;
```

into

```
React.DOM.div(null, "Hello ", this.props.name),
React.createElement("div", null, "Hello ", this.props.name);
```

- Just another build step like uglify
  - Can use jsx-transform, react-tools, babel

Example



```
▼<Top Level>
  ▼<App>
    ▼<div className="App">
      ▼<Header>
        ▼<div className="Header">
           <div>Icon</div>
         </div>
       </Header>
      ▼<Menu>
        ▼<div className="Menu">
          ▶<LoginForm>…</LoginForm>
          ▶<TextLine text="Or">…</TextLine>
          ▼<Button style="office" content="Log in
          with Office365">
             <ReactDOMButton className="Button
             Button--office">Log in with
             Office365</ReactDOMButton>
           </Button>
          ▶<Button style="google" content="Log in
          with Google">…</Button>
          ▶<Button style="facebook" content="Log in
          with Facebook">…</Button>
         </div>
       </Menu>
     </div>
   </App>
 </Top Level>
```

```
content: "Log in with Office365"
style: "office"
▶ proto : Object
```

Simple

React API

(Demo: http://jsbin.com/yefabumixi/1/edit), increment the number.

So the next would be http://jsbin.com/yefabumixi/2/edit

Create a custom component (your own `select`):

```
var NewButton = React.createClass({});
```

Render a component to a container:

```
React.render(<NewButton />, document.body);
```

Props

```
// Props are like html attributes and are seen as immutable
<NewButton color={ 'red' } />
```

```
getDefaultProps: function() { return { color: 'green' } }
```

```
// Validate component usage (only in development)
propTypes: { color: React.PropTypes.string }
```

https://facebook.github.io/react/docs/top-level-api.html

State

```
// Unlike props, state can change over time
this.state.numClicks
```

```
getInitialState: function () { return { numClicks: 0 } }
```

Change State

```
// from handleClick()
this.setState({
  numClicks: this.state.numClicks + 1
})
```

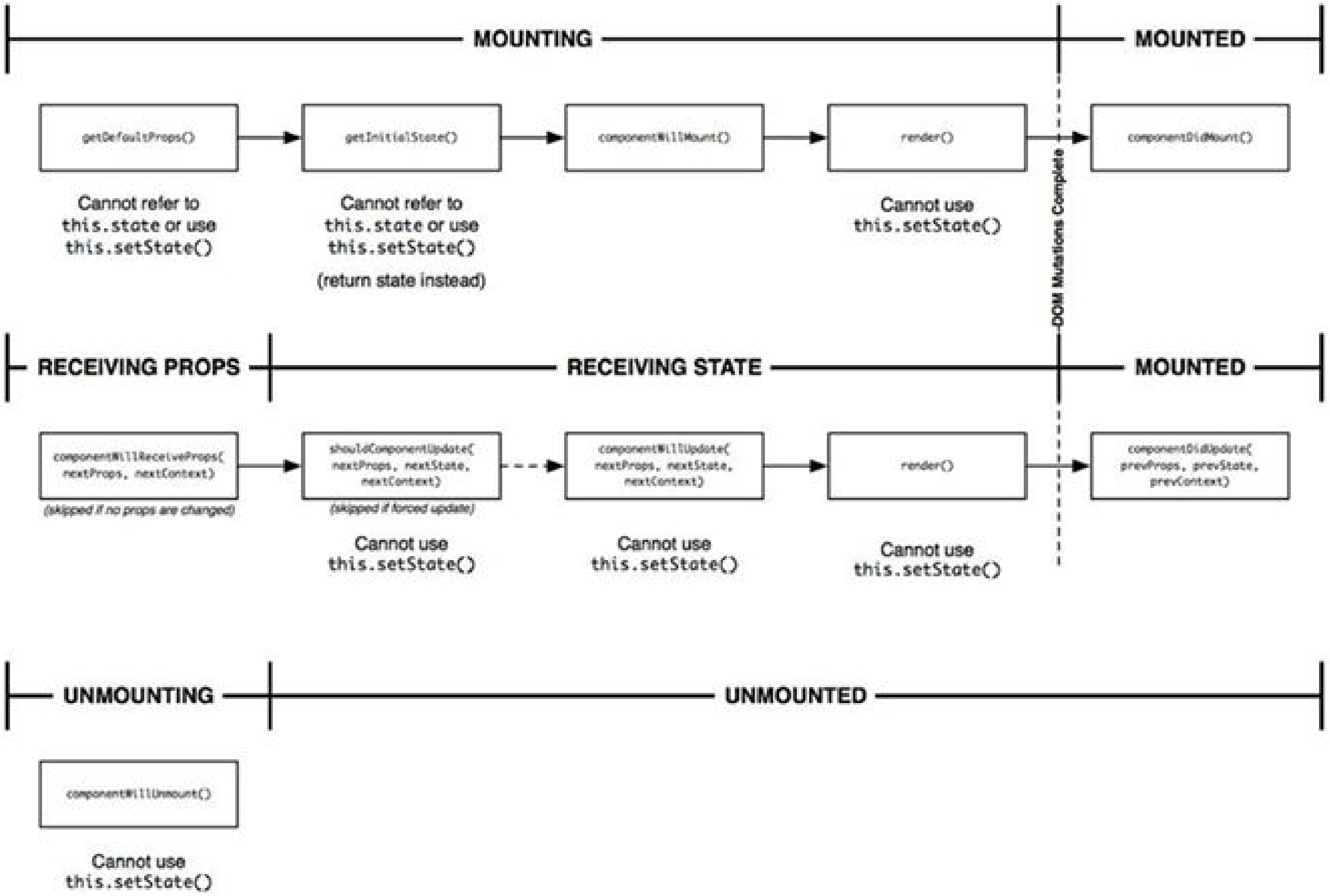
Render Function

```
// render function
render: function() {
    return (<div onClick={this.handleClick}>
      Num clicks: {this.state.numClicks}
    </div>);
  }
```

- Mixins (React 0.12), React.renderToString, statics, contexts, ES6 Classes

Component Lifecycle methods

```
// beforeRender
componentWillMount: function() {
  // initial code to be invoked once
}

// afterRender
componentDidMount: function() {
  // can use 3rd-party libraries like Kendo, setTimeout, AJAX
  var node = this.getDOMNode(); // get access to the actual DOM
}

// Destroy
componentWillUnmount: function() {
  // can unbind/remove event listeners here
}

// Before props change
componentWillReceiveProps: function(nextProps) {}

// Before state change
componentWillUpdate: function(nextProps) {}

// after DOM is updated
componentDidUpdate: function(prevProps, prevState) {}

// Whether to re-render or not
shouldComponentUpdate: function(nextProps, nextState) {
  // return true or false
}
```


MOUNTING
MOUNTED
getDefaultProps()
getInitialState()
componentWillMount()
render()
componentDidMount()
Cannot refer to this.state or use this.setState()
Cannot refer to this.state or use this.setState()
(return state instead)
Cannot use this.setState()
DOM Mutations Complete
RECEIVING PROPS
RECEIVING STATE
MOUNTED
componentWillReceiveProps(nextProps, nextContext)
(skipped if no props are changed)
shouldComponentUpdate(nextProps, nextState, nextContext)
(skipped if forced update)
componentWillUpdate(nextProps, nextState, nextContext)
render()
componentDidUpdate(prevProps, prevState, prevContext)
Cannot use this.setState()
Cannot use this.setState()
Cannot use this.setState()
UNMOUNTING
UNMOUNTED
componentWillUnmount()
Cannot use this.setState()

Misc

Seperate data-fetching and rendering concerns (see Relay)

```
class CommentListContainer extends React.Component {
  constructor() { this.state = { comments: [] } }
  componentDidMount() {
    $.ajax({
      url: "/my-comments.json",
      dataType: 'json',
      success: function(comments) {
        this.setState({comments: comments});
      }.bind(this)
    });
  }
  render() {
    return <CommentList comments={this.state.comments} />;
  }
}
```

```
class CommentList extends React.Component {
  constructor(props) { super(props); }
  renderComment({body, author}) {
    return <li>{body}—{author}</li>;
  }
  render() {
    return <ul> {this.props.comments.map(renderComment)} </ul>;
  }
}
```

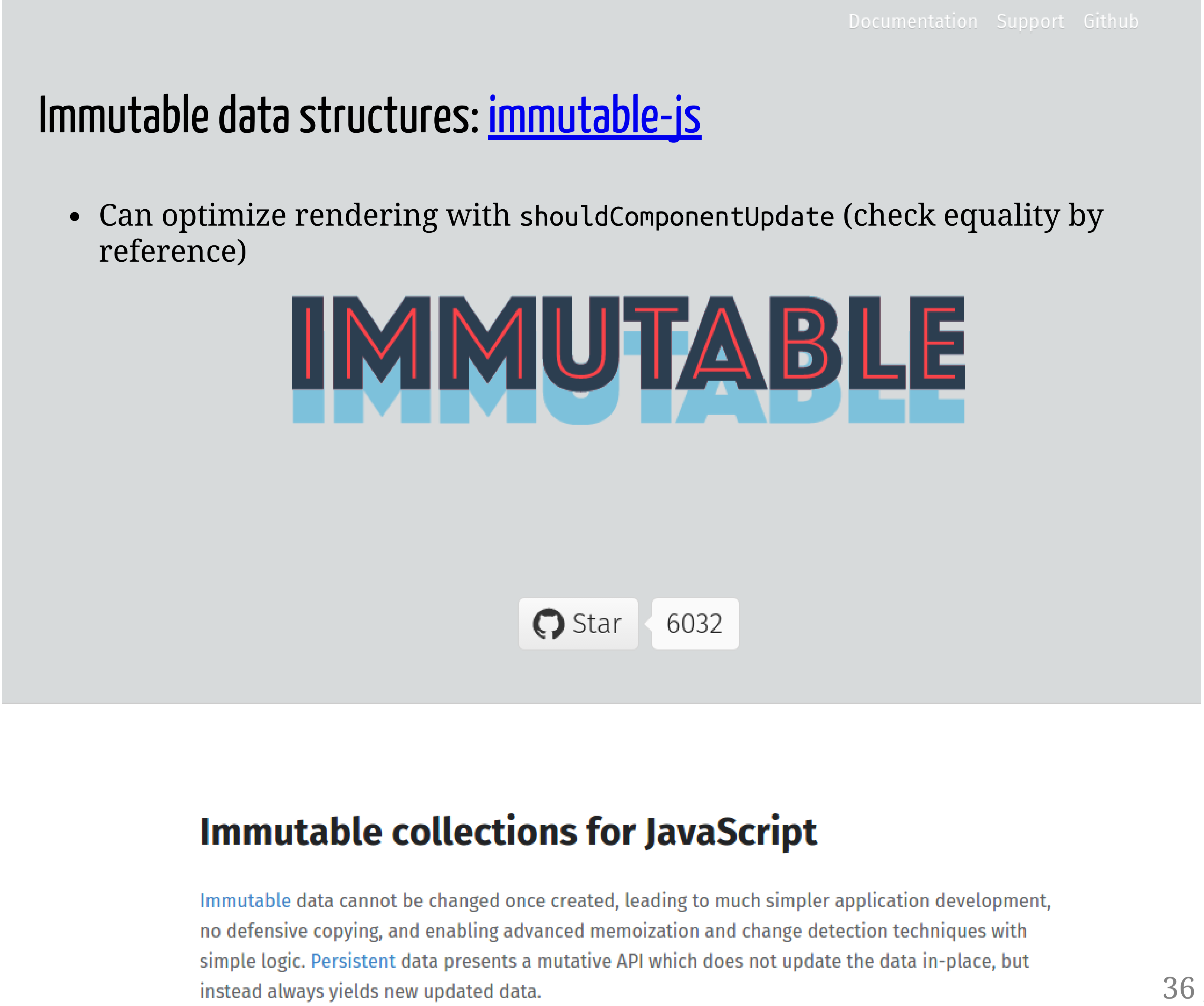
https://medium.com/@learnreact/container-components-c0e67432e005

Seperation of concerns: props (immutable) vs. state (changable)

- Encourages less state in less places
  - "shared mutable state is the root of all evil"
  - Can have stateless components and stateful components
- Simpler to test: Jest
  - Test a component by rendering as a string and and comparing the output. (Events can be created synthetically)

Immutable data structures: immutable-js

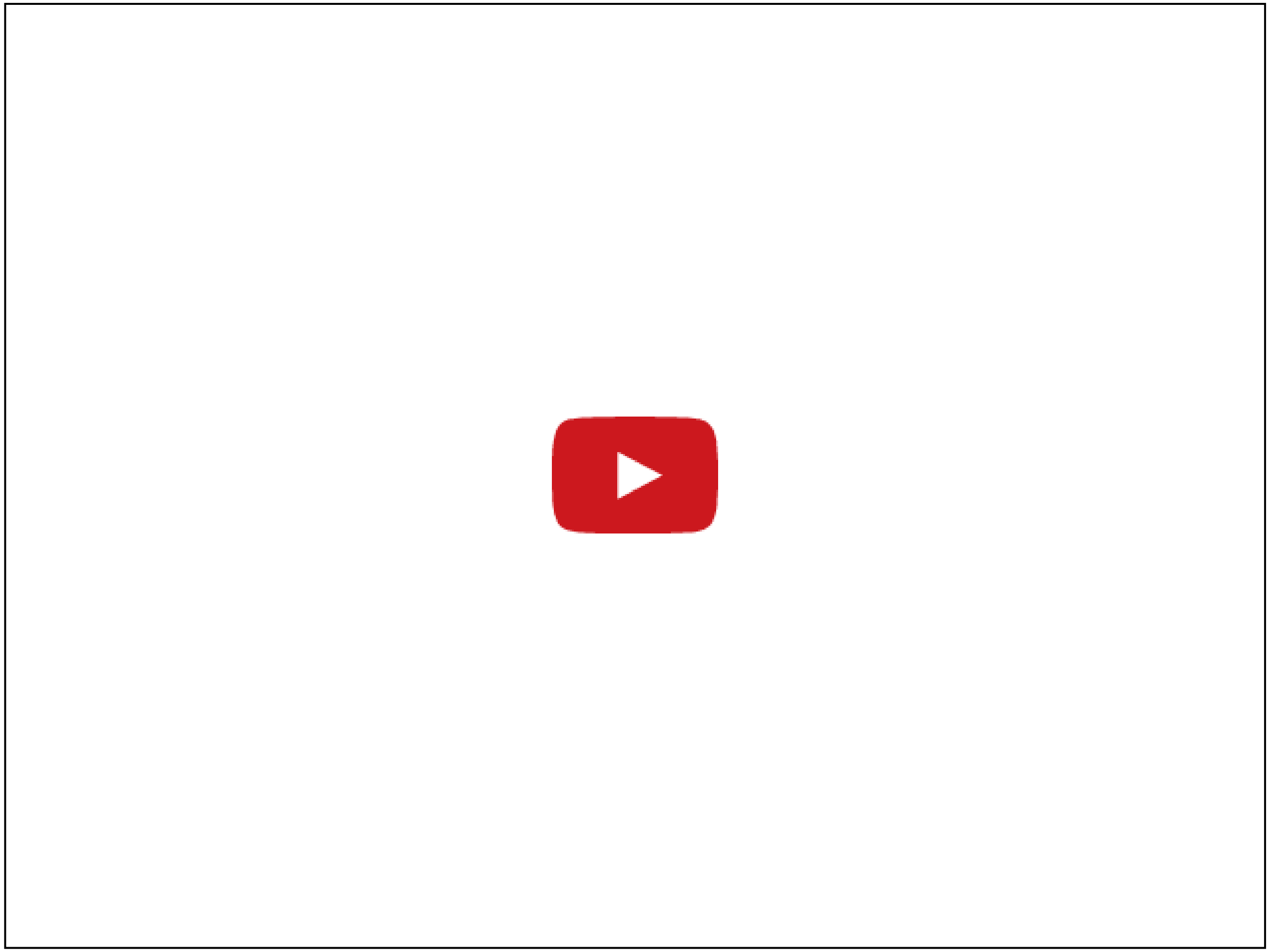
- Can optimize rendering with `shouldComponentUpdate` (check equality by reference)



Immutable collections for JavaScript

Immutable data cannot be changed once created, leading to much simpler application development, no defensive copying, and enabling advanced memoization and change detection techniques with simple logic. Persistent data presents a mutative API which does not update the data in-place, but instead always yields new updated data.

- Easy to snapshot (the view is a function of the state)
- Easy to incorporate undo/redo

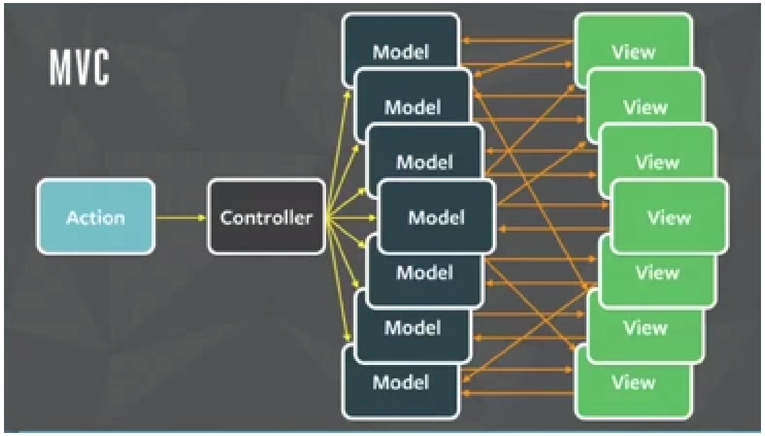


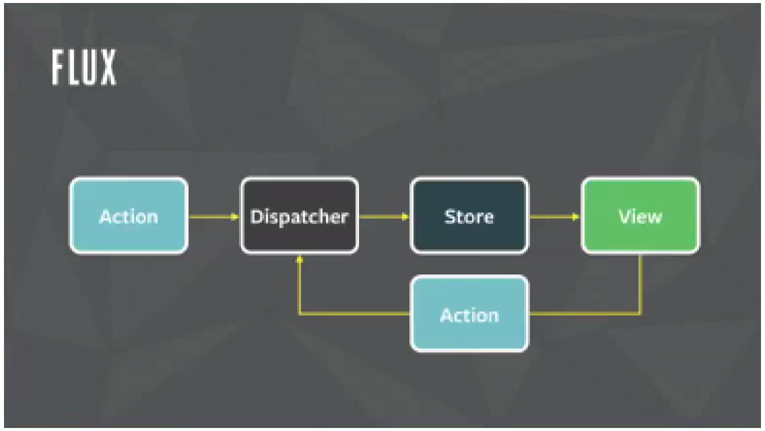
Server rendering (`React.renderToString`)

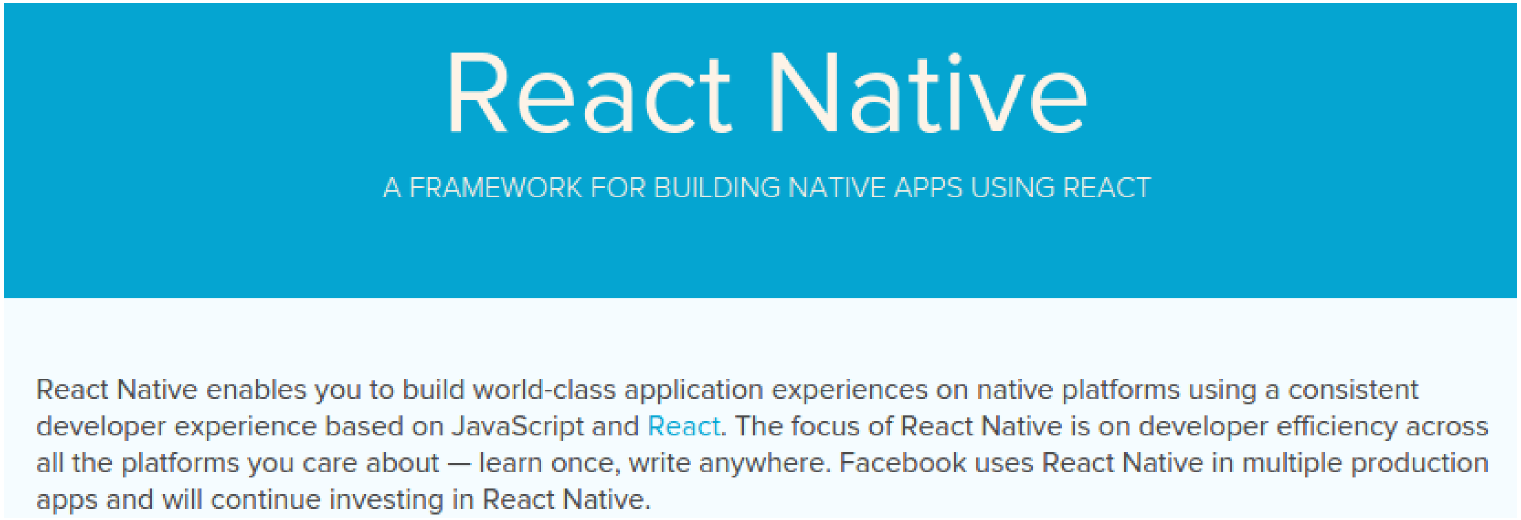
- A faster initial experience for the user (remove extra http requests).
- Isomorphic JavaScript: shared javascript code (on client and server)
- Was difficult to do SEO for client apps since they render on DOM load until google bot changed

Data

- Normally data is passed down through components and events flow up
- Flux: a global event system with one way data flow



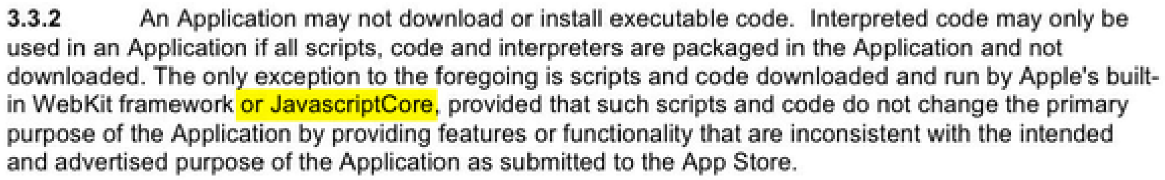



React Native

A FRAMEWORK FOR BUILDING NATIVE APPS USING REACT

React Native enables you to build world-class application experiences on native platforms using a consistent developer experience based on JavaScript and React. The focus of React Native is on developer efficiency across all the platforms you care about — learn once, write anywhere. Facebook uses React Native in multiple production apps and will continue investing in React Native.

- Ability to push updates without review


3.3.2 An Application may not download or install executable code. Interpreted code may only be used in an Application if all scripts, code and interpreters are packaged in the Application and not downloaded. The only exception to the foregoing is scripts and code downloaded and run by Apple's built-in WebKit framework or JavascriptCore, provided that such scripts and code do not change the primary purpose of the Application by providing features or functionality that are inconsistent with the intended and advertised purpose of the Application as submitted to the App Store.

- Native iOS Components, Asynchronous Execution, Flexbox, Polyfills, Extensible