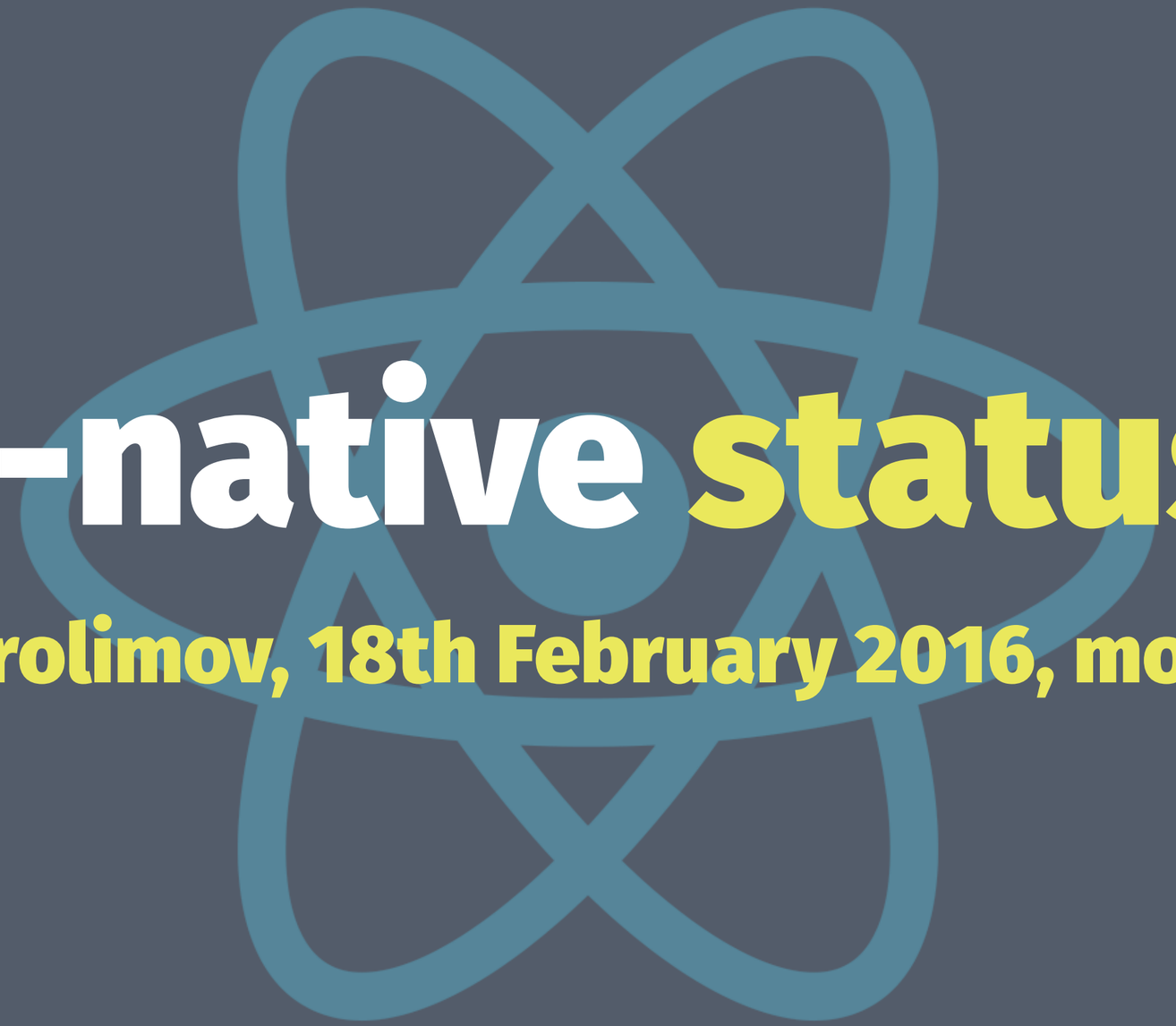


react-native status quo

Christoph Jerolimov, 18th February 2016, mobile.cologne

Agenda

→ Motivation & Concept

→ Stylesheets / Flexbox

→ Demo

→ Status Quo

→ Integration opportunities

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.

Get started with React Native

Native Components

With React Native, you can use the standard platform components such as UITabBar on iOS and Drawer on Android. This gives your app a consistent look and feel with the rest of the platform ecosystem, and keeps the quality bar high. These components are easily incorporated into your app using their React component counterparts, such as TabBarIOS and DrawerLayoutAndroid.

```
// iOS

var React = require('react-native');
var { TabBarIOS, NavigatorIOS } = React;

var App = React.createClass({
  render: function() {
    return (
      <TabBarIOS>
```

Motivation

→ Share know how (React), maybe team / code

→ Better developer experience (DX) than native

Write once, run anywhere

Initially Java, but html5 too

~~Write once, run anywhere~~

Learn once, use anywhere

Share know how: React

Declarative UI
(Unidirectional data flow)

A view-only library
(the view in MVC, but MVC is not required)

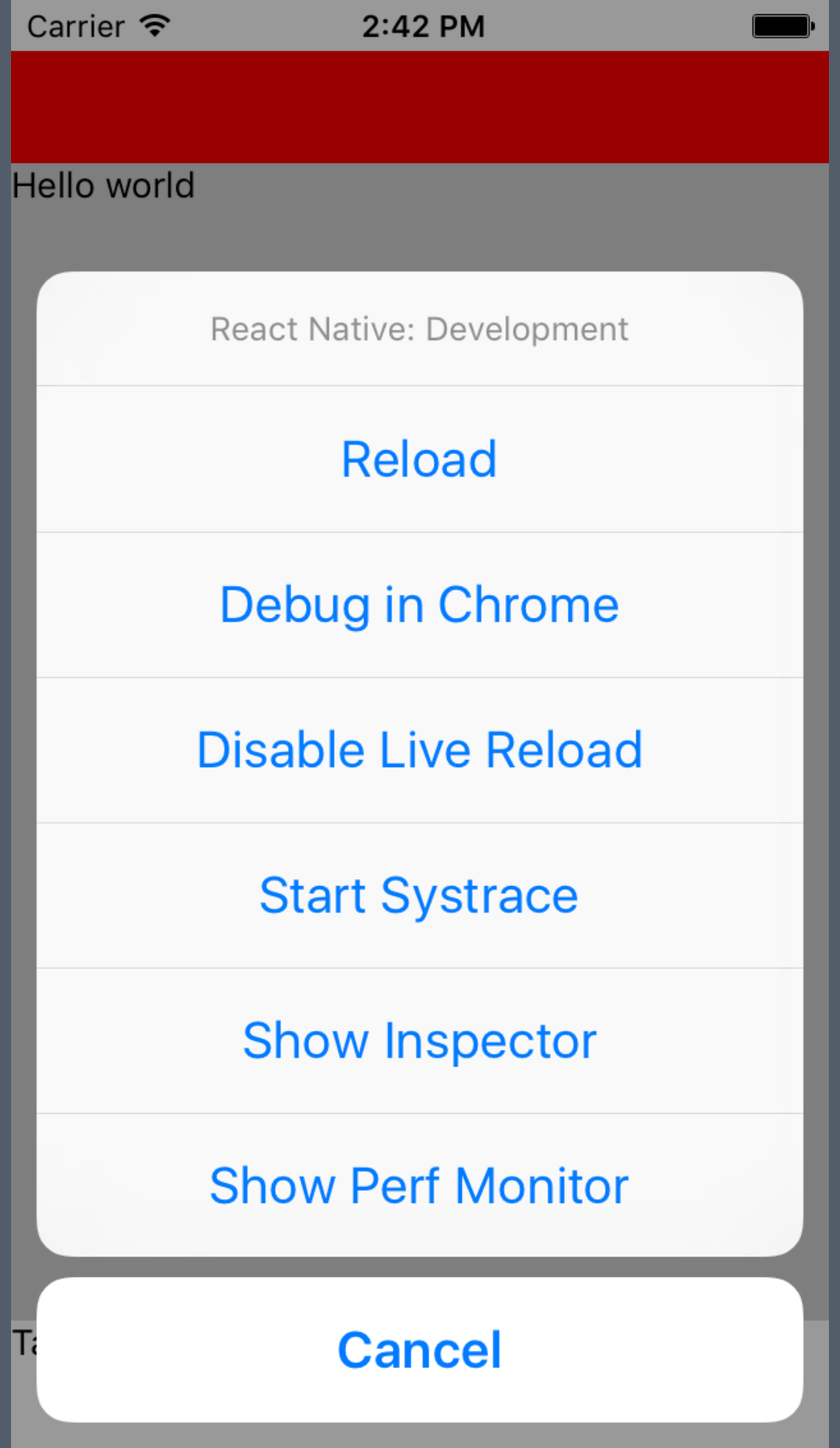
Automatically updates the view hierarchy

Better developer Experience

→ "HTML- & CSS-like" => JSX + Flexbox

→ Hot reloading (⌘R) & Live Reload

→ Debugger, UI Inspector, Profiling


Carrier
2:42 PM
Hello world
React Native: Development
Reload
Debug in Chrome
Disable Live Reload
Start Systrace
Show Inspector
Show Perf Monitor
Cancel

Solution

→ Reuse react.js (web) to render the view hierarchy

→ Renders native views (no WebView!)

→ Polyfills for networking (fetch), Geolocation, …

→ (Easy) Integration options in *both* directions

In development

→ You write **"modern" javascript** in your favorited editor

→ Babel transform the sources (ES6 and more...)

→ App communicates with a local http server

In production

→ Precompiled, minified JS bundled within the app

→ Code updates are technical possible.. and allowed

Technical

→ Based on a **minimal JS VM: JavaScriptCore**

→ JS controls the native UI

→ JS renders the "virtual DOM" as JSON

→ JS <-> Native bridge (multithreaded)

→ Native part renders UI based on this JSON

Supported platforms

→ Android 4.1+, >= 93 % [1]

→ iOS 7+, >= 96 % [2] [3]

[1] https://developer.android.com/about/dashboards/index.html

[2] https://david-smith.org/iosversionstats/

[3] https://developer.apple.com/support/app-store/

Getting Started[4]

→ Requires Node.js 4+, nvm is recommended

→ OSX is the common dev platform (at FB)

→ Linux and Windows should work[5]

→ Android SDK[6] for Android / Xcode 7+ for iOS

[4] http://facebook.github.io/react-native/docs/getting-started.html

[5] http://facebook.github.io/react-native/docs/linux-windows-support.html

[6] http://facebook.github.io/react-native/docs/android-setup.html

View components

View, Text, TextInput, Image, Switch, ScrollView,
PickerIOS, ProgressBarAndroid, ProgressViewIOS,
WebView, ListView, Navigator, NavigatorIOS, Modal,
MapView, RefreshControl, TabBarIOS,
ActivityIndicatorIOS, DatePickerIOS,
DrawerLayoutAndroid, PullToRefreshViewAndroid,
SegmentedControlIOS, SliderIOS, TouchableHighlight,
TouchableOpacity, TouchableWithoutFeedback, **...**

Other APIs / modules

ActionSheetIOS, **Alert,** AlertIOS, **Animated,**
AppRegistry, AppState, AppStateIOS, AsyncStorage,
BackAndroid, CameraRoll, Dimensions,
IntentAndroid, InteractionManager, LayoutAnimation,
LinkingIOS, **NetInfo, PanResponder,**
PushNotificationIOS, StatusBarIOS, StyleSheet,
ToastAndroid, VibrationIOS, **...**

RefreshControl

Edit on GitHub

This component is used inside a ScrollView to add pull to refresh functionality. When the ScrollView is at `scrollY: 0`, swiping down triggers an `onRefresh` event.

Props

View props...

android **colors** [[object Object]]

The colors (at least one) that will be used to draw the refresh indicator.

android **progressBackgroundColor** color

The background color of the refresh indicator.

ios **tintColor** color

The color of the refresh indicator.

ios **title** string

The title displayed under the refresh indicator.

A React Component

Extends `React.Compontent`

External immutable *props*
vs
Internal private *state*

Must implement at least the `render()`-method

Optional methods to handle the lifecycle/updates
(componentWillMount ... componentWillUnmount)

Hello world

```
class HelloWorld extends Component {
    render() {
        return <Text>Hello World</Text>;
    }
}

AppRegistry.registerComponent('MyApp', () => HelloWorld);
```

JSX state example

```
class Blink extends React.Component {
    constructor(props) {
        super(props);
        this.state = { visible: true }
    }
    componentWillMount() {
        this.interval = setInterval(() => {
            this.setState({ visible: !this.state.visible });
        }, 1000);
    }
    componentWillUnmount() {
        this.clearInterval(this.interval);
    }
    render() {
        const style = { opacity: this.state.visible ? 1 : 0 };
        return <Text style={ style }>{ this.props.children }</Text>
    }
}
```

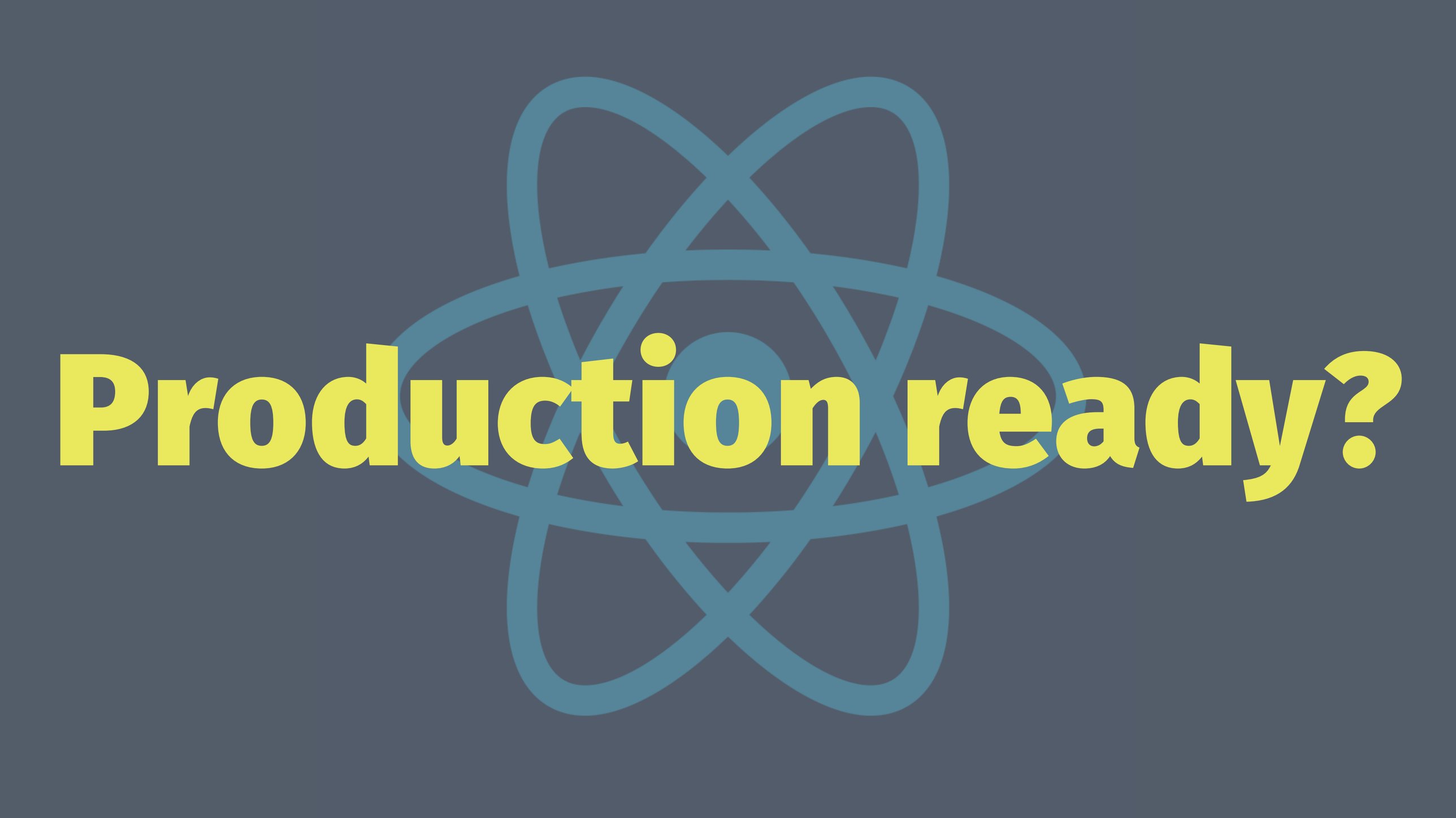
Stylesheets

```
const bold = {
    fontWeight: 'bold' // A string!
};

const styles = StyleSheet.create({
    bold: {
        fontWeight: 'bold'
    }
});

<View style={{ borderWidth: 1, borderColor: 'red' }}>
    <Text style={ bold }>Hello World</Text>
    <Text style={ styles.bold }>Hello World</Text>
</View>
```

Flexbox

```
// Grow 100% with childs 50%, 30% and 20%
<View style={{ flex: 1, flexDirection: 'row' }}>
    <View style={{ flex: 0.5, backgroundColor: 'red' }} />
    <View style={{ flex: 0.3, backgroundColor: 'blue' }} />
    <View style={{            backgroundColor: 'green' }} />
</View>;

// Grow 100% where first and last child is fix
<View style={{ flex: 1 }}>
    <View style={{ height: 64, backgroundColor: 'red' }} />
    <View style={{             backgroundColor: 'blue' }} />
    <View style={{ height: 50, backgroundColor: 'green' }} />
</View>;
```

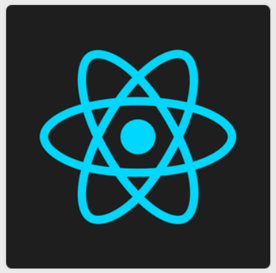
Demo


Production ready?

ProductPains

Post

1



React Native

A framework for building native apps with React

facebook.github.io/react-native

REP

Recent Activity | Top | Fixed

REACT NATIVE
191 Get Android to feature parity with iOS 22

REACT NATIVE
172 React-Native to depend on React 4

REACT NATIVE
115 Offload some animations from JS thread for better perf 8

REACT NATIVE
82 Fix Various ListView Issues 8

REACT NATIVE

Status & Roadmap

→ 0.x - But production ready if your brave.

→ Some components are not yet available on Android
(MapView for example, but community projects are available for all common problems!)

→ Android M permissions

→ Performance and API improvements


Erik Schlegel
@erikschlegel1
Follow
Speaking at @reactconf next month to talk about what the Microsoft Developer Experience group is doing with @reactnative.
RETWEETS
12
LIKES
17
Vj
6:02 AM - 31 Jan 2016

Navigation

pain: Navigator / NavigatorIOS / DrawerLayoutAndroid

better: ExNavigator by James @Ide

NavigationExperimental ??

→ tip: Make your navigation stack serializable

→ If you need two, make it twice

Performance

→ Native UI, e.g. ScrollView

→ Smooth animations

→ Sometimes laggy, e.g. ListView,
missing estimated cell height?

→ Never as fast as *optimized* native code

Platform switch

Auto-select component based on a file suffix:

```
CustomShoppingCardItem.android.js
CustomShoppingCardItem.ios.js
```

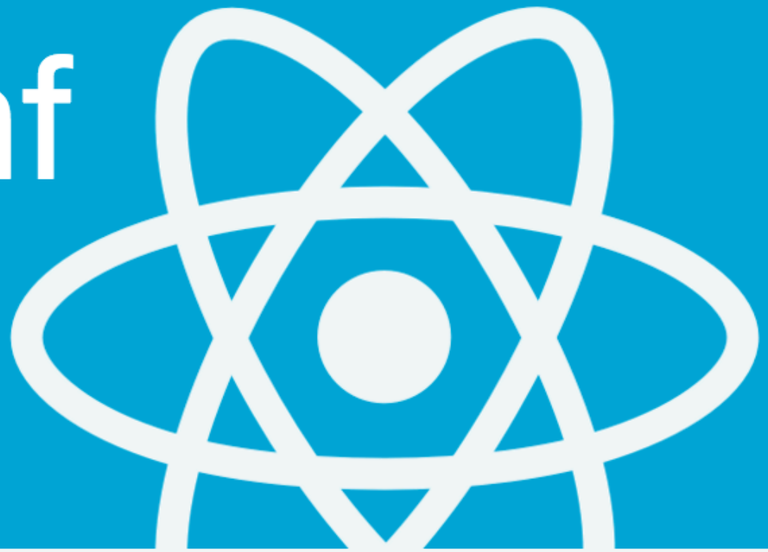
Or a good old platform switch:

```
import { Platform } from 'react-native';

if (Platform.OS === 'android') {
    // ...
} else {
    // ...
}
```

React.js Conf

FEBRUARY 22 & 23, 2016

SAN FRANCISCO, CA



Overview

At React.js Conference 2015, we announced React Native, GraphQL, and Relay, and the community response blew us away.

React opens a world of new possibilities such as server-side rendering, real-time updates, different rendering targets like svg, canvas, iOS, and Android.

We need to rethink how we write applications using one-way data flow, immutable data structures, the full power of JavaScript and languages targeting it.

Join us at React.js Conf to shape the future of client-side applications!

Questions?

Thank you