Introduction react-native

Webworkers Cologne, 27th january 2016, Christoph Jerolimov

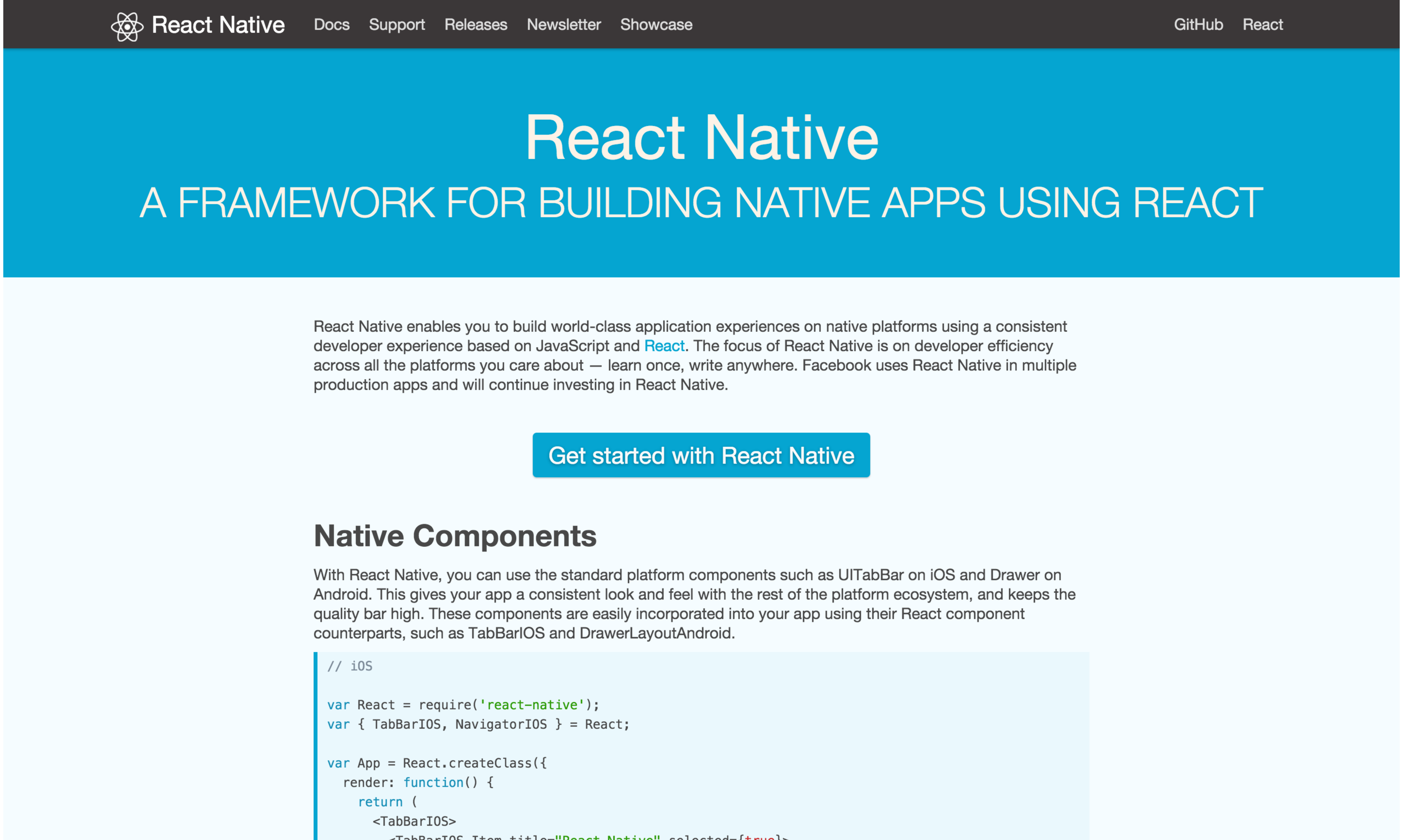
Agenda

→ Motivation & Concept

→ Components / Native Components

→ Stylesheets / Flexbox

→ Demo



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>
        <TabBarIOS.Item title="React Native" selected={true}>
```

Motivation

→ Share code and know how (React)

→ Better developer experience (DX) than native

Write once, run anywhere

Initially Java, but html5 too

~~Write once, run anywhere~~

Learn once, use anywhere

Solution

→ Reuse react.js to render a VDOM / view hierarchy

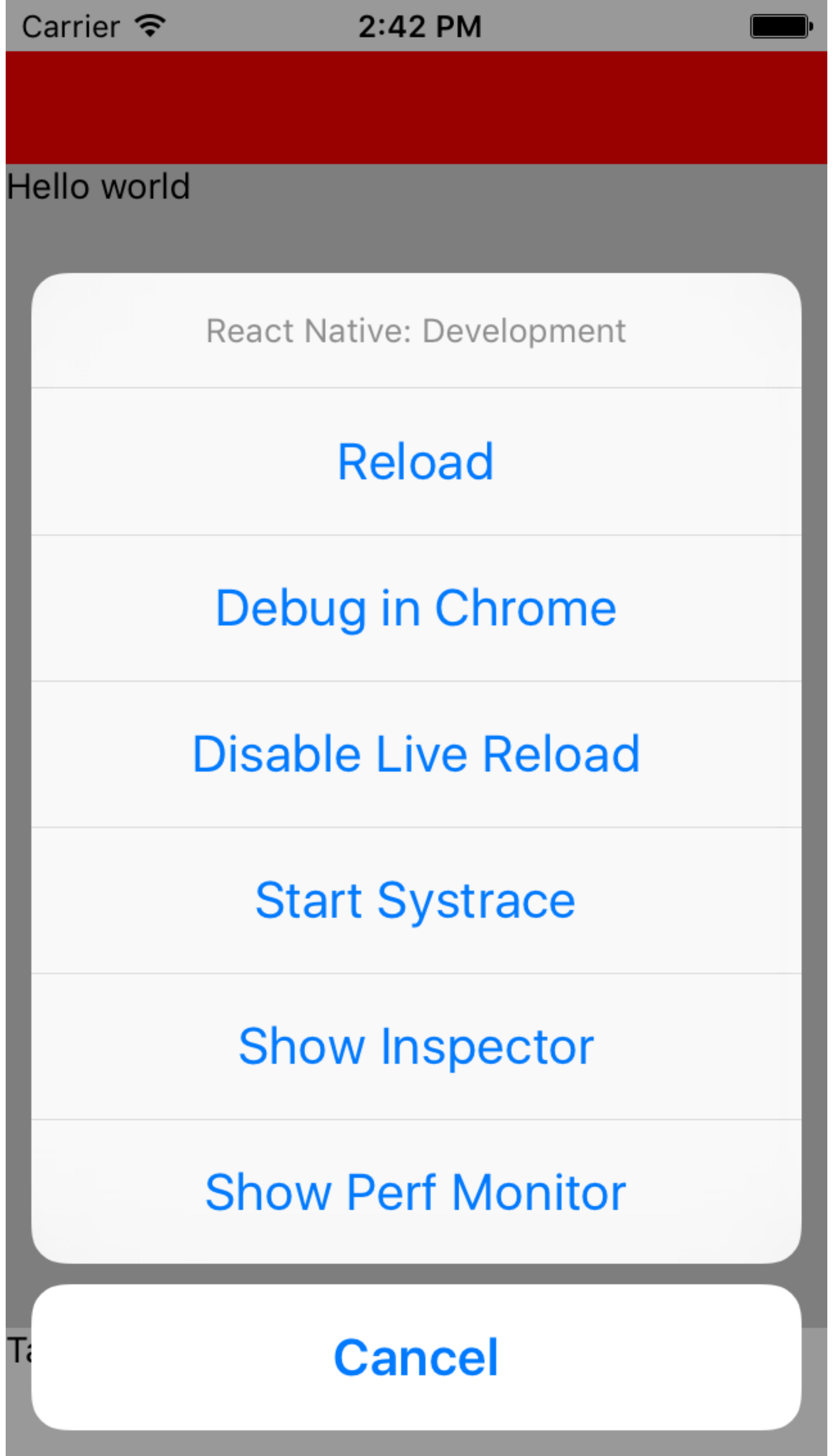
→ But renders them as native views (no WebView!)

→ Polyfills for networking (fetch), Geolocation, ...

→ (Easy) Integration options in *both* directions

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

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 most problems)
- → Android M permissions
- → Performance and API improvements

Performance

- → Native UI, Fast, Responsive
- → Smooth animations
- → Complex gestures
- → Everything runs asynchronous
- → Bridge is batched
- → Never as fast as *optimized* native code

In development

- → You write 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**, **...**

Hello world

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

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

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

Platform switch

Auto-select component based on a file suffix:

```
Slider.android.js
Slider.ios.js
```

Or a good old platform switch:

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

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

Questions?

Thank you