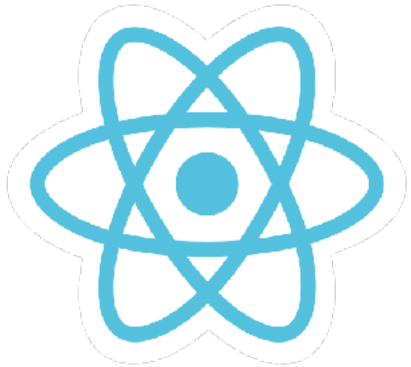
Teufelszeug React-Native

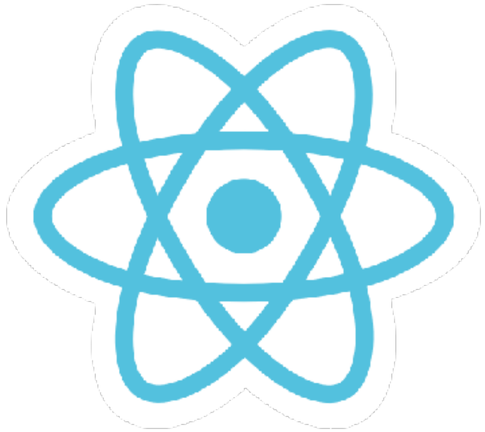
Düsseldorf iOS Meetup & CocoaHeads Düsseldorf

21.03.2017 - Christoph Jerolimov



React & React-native

Flexbox Layout



React-Native

The good & the bad parts

- Crossplatform
- “Developer experience” -vs- Bleeding Edge (fun?)
- (JavaScript)

Issues

- Neither Code nor know how sharing possible
- Native try & error process is broken (slow)
- WebView User Experience is bad

“Write once, run anywhere.”

— First Java, and later HTML 5

“**Learn** once, **use** anywhere.”

— react

HelloWorld.js

```
import React, { Component, Text } from 'react-native';



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

HelloWorld2.js

```
import React, { Component } from 'react-native';

import HelloWorld from ‘./HelloWorld‘;


class HelloWorld2 extends Component {
    render() {
        return <HelloWorld />
    }
}
```

Idea

- Reuse React.js to **render a view hierarchy**
- Render ***native*** **views (no WebView)**
- Polyfills for networking (fetch), Geolocation, …
- (Easy) Integration options in *both* directions

React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Get Started Download React v0.14.3

JUST THE UI

Lots of people use React as the V in MVC. Since React makes no assumptions about the rest of your technology stack, it's easy to try it out on a small feature in an existing project.

VIRTUAL DOM

React abstracts away the DOM from you, giving a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native.

DATA FLOW

React implements one-way reactive data flow which reduces boilerplate and is easier to reason about than traditional data binding.

A Simple Component

React components implement a `render()` method that takes input data and returns what to display. This example uses an XML-like syntax called JSX. Input data that is passed into the component can be accessed by `render()` via `this.props`.

JSX is optional and not required to use React. Try clicking on "Compiled JS" to see the raw JavaScript code produced by the JSX compiler.

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>
```

React

- **Declarative UI**
  (Unidirectional data flow)

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

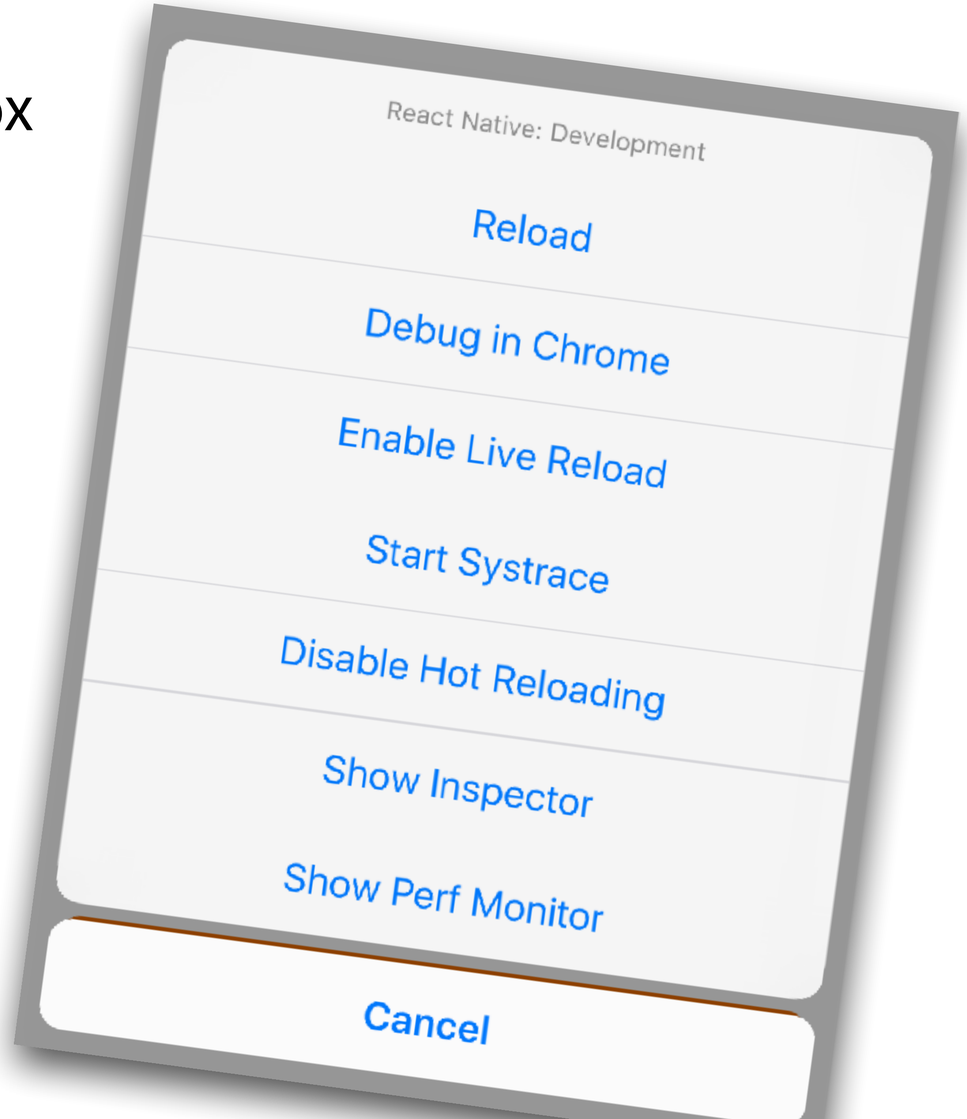
- **Automatically updates the view hierarchy**

Uni-directional data flow

```
class HelloWorld extends Component {
  render() {
    return (
      <View>
        <Text> { this.props.title } <Text>
        <Text> { this.state.value } <Text>
        <Button onPress={ this.changeValue } />
      </View>
    );
  }
}
```

Developer Experience? ⌘D

- “HTML- & CSS-like” => JSX + Flexbox
- (Fast) Reload your app (⌘R)
- Live Reload & Hot-Reloading
- Debugger, UI Inspector, Profiling
- ….



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.

In Development

- You write **“modern” javascript** in your favorite editor
- *Babel* transform your sources (ES 6 and more …)
- App communicates with a **local http server**

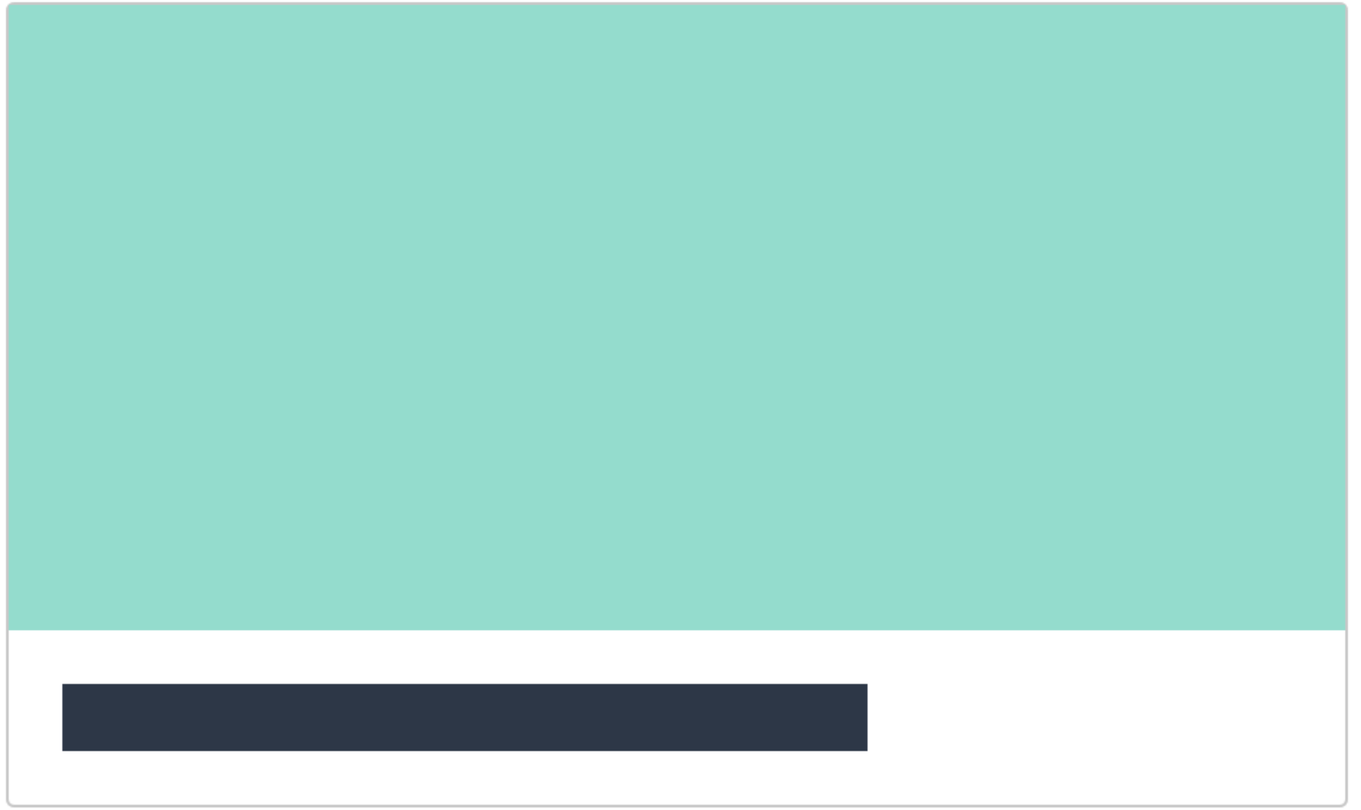
In Production

- **Pre”compiled"**, minified javascript
- **Bundled** within the app container
- JS and asset updates are possible… and allowed

Technical

- JS was executed in a **minimal “JavaScript VM”**
  (JavaScriptCore is part of the WebKit OSS project)
- “Virtual DOM” is a JSON representation of the UI state
- JS <=> Natives bridge (multithreaded)
- Native part renders UI based on the JSON.

Flexbox Layout



Stylesheet

```
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>;
```

Platform Switch

Good old platform switch:

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

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

Auto-select component based on a file suffix:

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

Navigation…

- NavigatorIOS
- Navigator
- ExNavigator
- NavigatorExperimental
- “react-navigation” <= I’m here
- “react-router”
- Airbnb “native-navigation”

Q&A