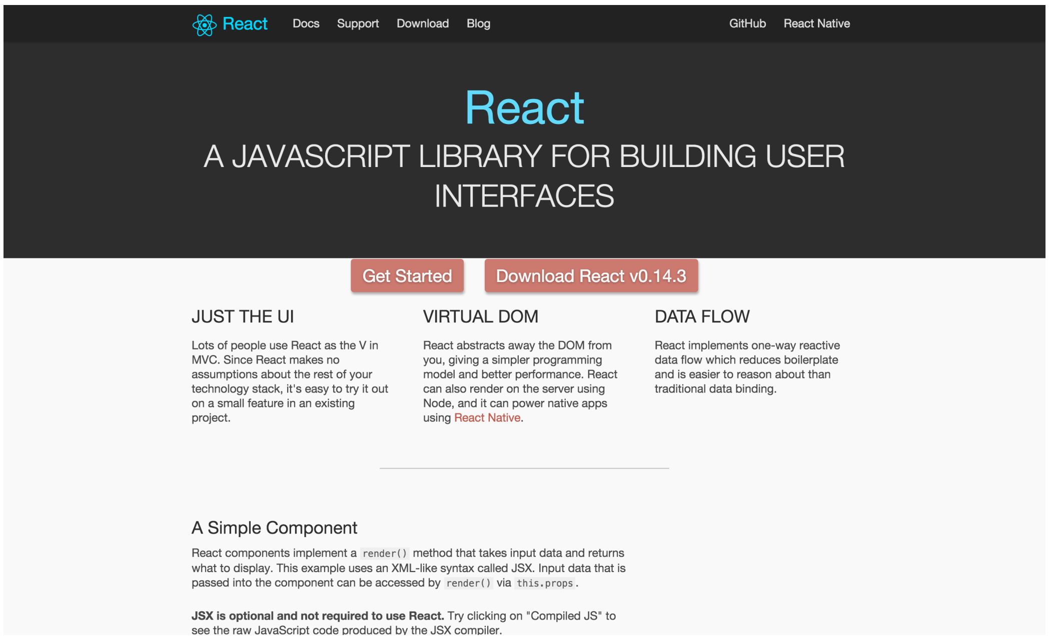
[fit] Teufelszeug

[fit] react-native

[fit] Cocoaheads, Köln, August 15th 2016, Christoph Jerolimov

Agenda

-
-
- Motivation & Concept
- Native Components / Stylesheets / Flexbox



React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

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.

^What's react.js, a lib, no framework Easy to integrate, also on small parts Pushished mid 2013 by Facebook Mitte 2013 von Facebook veröffentlicht First reactions were rather skeptical. JSX is JS superset

React

Declerative UI In general, a uni-directional data flow.

A view-only library The view in MVC, but MVC is not required.

Automatically updates the DOM, when necessary The browser is just *one possible* rendering engine.

^Components are capsulated, reuseable, testable units Simple to understand and maintain

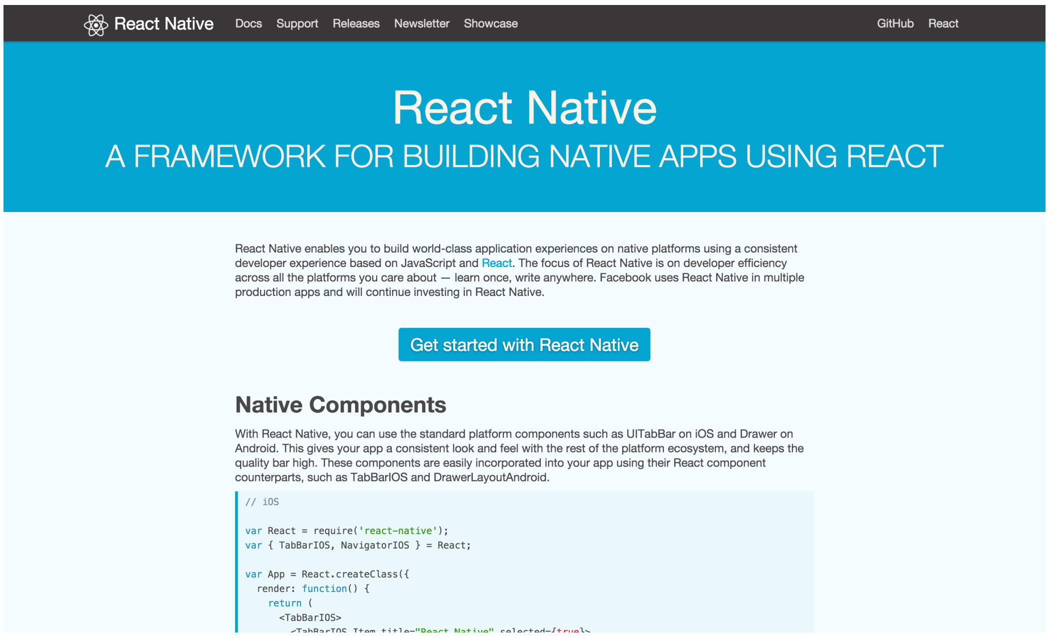
React-native

Declerative UI In general, a uni-directional data flow.

A view and bridging library View, geolocation, network, ...

Automatically updates the view hierarchy, when necessary The browser is just *one possible* rendering engine.

^Components are capsulated, reuseable, testable units Simple to understand and maintain



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.

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}>
```

^Announced Jan 2015, first version published spring 2015 Sep 2015 Android support GitHub rank #27, ~23.500 stars, 420+ contributors

React JSX example

JSX is a JS superset and supports sub-components (and DOM elements) inline:

```javascript import React, { Component } from 'react';

class HelloWorld extends Component { render() { return Hello World; } }

// Usage: ```
```

React-native JSX example

JSX is a JS superset and supports sub-components inline:

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

class HelloWorld extends Component { render() { return Hello World; } }

// Usage: ```
```

JSX property example

Usage of external properties, not only strings:

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

class Hello extends Component { render() { return Hello {{ this.props.person.firstname }}; } }

// Usage: ```
```

}

JSX state example

```js class Blink extends Component { componentWillMount() { setInterval(() => { this.setState({ visible: !this.state.visible }); }, 1000); }

```
render() {
    const style = { opacity: this.state.visible ? 1 : 0 };
    return <Text style={ style }>{ this.props.children }</T
ext>
}
```

}

// Usage: ```
```

Components

Every Element is/extends a react `Component`

External immutable **props**
vs
Internal private **state**

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

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

(There are other ways to define a component...)

Virtual DOM / view hierarchy

DOM manipulations are slow.

Render method generates a VDOM ("JSON")
and calculates a diff to reduce DOM manipulations.

render (UI-State$_n$) => VDOM$_n$

diff (VDOM$_{n-1}$, VDOM$_n$) => DOM updates...

^The virtual DOM is a JS object, not bindet directly to the DOM. This allow performance hacks etc.

Reasons for react-native

- Reuse knowhow: build feature instead of platform teams
- Increase **native dev** developer experience
- Easy **integration in both** directions
    - Integrate react-native view into native VH.
    - Integrate native view into react-native VH.
- Better UX than a WebView

Developer Experience

- "HTML- & CSS-like" => JSX + Flexbox
- Hot reloading (⌘R) & Live Reload
- Debugger, UI Inspector, Profiling

^Modern JavaScript w/ optional Flow (or TypeScript) ^Decrease turn around times, write it, test it, try again.

How does it work?

- Based on a **minimal JS VM: JavaScriptCore** (EcmaScript 5)
  - Android 4.1+, >= 96 % [1]
  - iOS 7+, >= 97 % ^2
- **JS <-> Native bridge** (multithreaded)
  - JS renders the "virtual DOM" -> JSON
  - Native part renders the native UI <- JSON

^JSX is part of WebKit iOS 7 includes a shared version already Android bundles the library w/ the app (3,5 MB)

^Use a inter process model by default and can also run the app in a remote process, for example in the Chrome (for Debugging)

^Also common native targets

Getting Started[^1]

- Requires Node.js 4+, nvm is recommended
  - for Android development: Android SDK[2]
  - for iOS development: Xcode 7+ (read as: a Mac)
- OSX is the common dev platform (at FB)
- but Linux and Windows should work[3]

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

[fit] Demo

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.

Stylesheets

```javascript 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>
```
```

~~~
---

![fit](./react-logo-1000-transparent.png)

## **Flexbox**

```javascript
    // 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>;
~~~

Navigation

pain: Navigator / NavigatorIOS / DrawerLayoutAndroid

better: ExNavigator by James @Ide

upcoming: NavigationExperimental

tip: Make your navigation stack serializable

Performance

- Native UI, e.g. ScrollView
- Smooth animations
- ListView has no estimated cell yet

Platform switch

Auto-select component based on a file suffix:

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

Or a good old platform switch:

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

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

```

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

[fit] Questions?

[fit] Questions?

[fit] Thank you!

---

1. http://facebook.github.io/react-native/docs/getting-started.html↩
2. http://facebook.github.io/react-native/docs/linux-windows-support.html↩
3. http://facebook.github.io/react-native/docs/android-setup.html↩