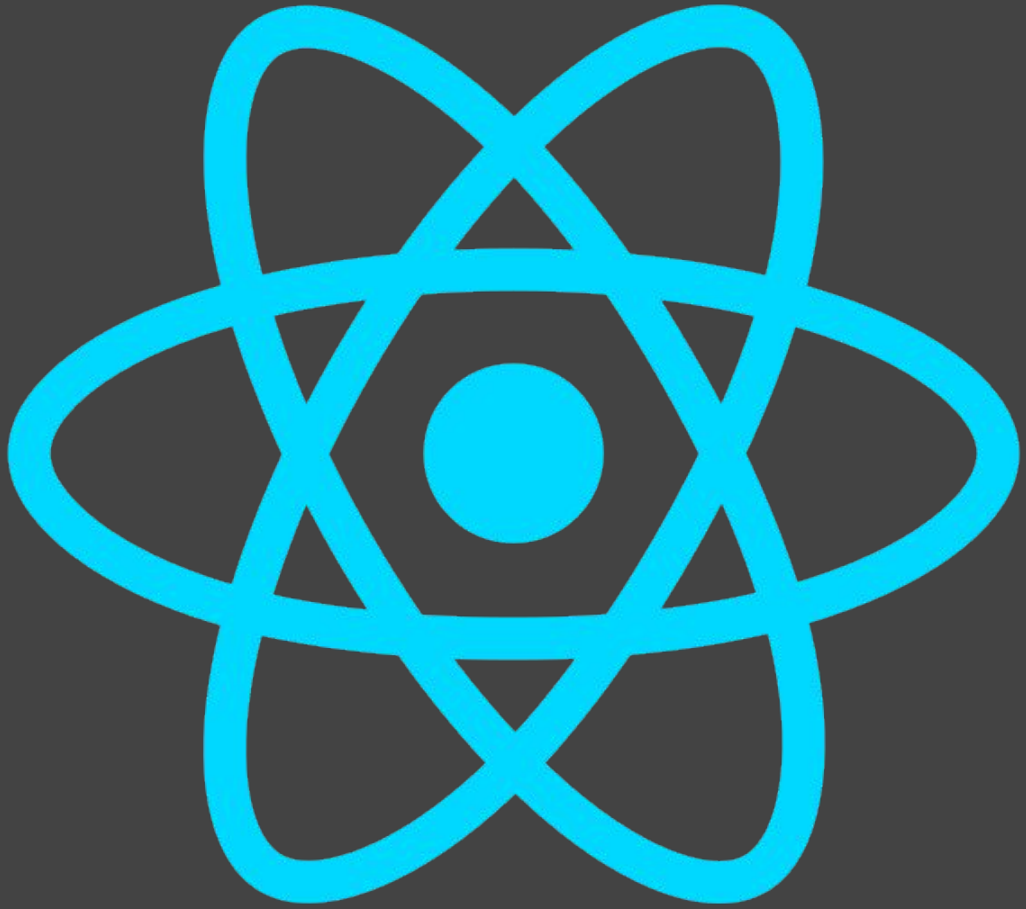


Manuel Mauky

@manuel_mauky
www.lestard.eu
github.com/lestard

Saxonia Systems
So geht Software.

JUG Görlitz

Kurzer Werbeblock

27.06.2018 - 19:00 Uhr

Vortrag: Immutable Data mit Java

JUG Görlitz

Vortrag bei der Java User Group Görlitz im iJUG e.V.

Immutable Data - Unveränderliche Daten im Griff

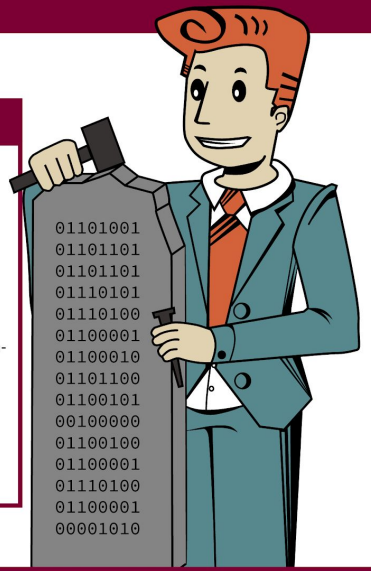
Manuel Mauky, Saxonia Systems AG

Thema

Unveränderliche Datenstrukturen sind ein wichtiges Werkzeug der funktionalen Programmierung, haben aber auch in der OOP einige Vorteile.

Da Java als Sprache aber keine besondere Unterstützung für Immutables mitbringt, müssen EntwicklerInnen auf Patterns und Bibliotheken zurückgreifen, um auf der JVM in den Genuss von unveränderlichen Datenstrukturen zu kommen.

Im Vortrag möchte ich neben einer grundsätzlichen Einführung in die Thematik auch einige dieser Techniken vorstellen und ihre Vor- und Nachteile besprechen.



Mittwoch, 27.06.2018
19:00 Uhr
Hochschule Zittau/Görlitz
Haus GII, Raum 0.10
Brückenstraße 1
02826 Görlitz

JUG Görlitz
http://www.jug-gr.de

iJUG Member

Mobile Apps?

Mobile Apps? - Welche Möglichkeiten existieren?

Native

Android
iOS

Mobile Apps? - Welche Möglichkeiten existieren?

Native

Android
iOS

Web App

Single-Page-App im Browser

Mobile Apps? - Welche Möglichkeiten existieren?

Native

Android
iOS

HTML5 Hybrid

Web-App in native Wrapper/WebView

- Cordova
- PhoneGap
- Ionic

Web App

Single-Page-App im Browser

Mobile Apps? - Welche Möglichkeiten existieren?

Native

Android
iOS

HTML5 Hybrid

Web-App in native Wrapper/WebView

- Cordova
- PhoneGap
- Ionic

Andere Hybrid

- JavaFX

Web App

Single-Page-App im Browser

Mobile Apps? - Welche Möglichkeiten existieren?

Native

Android
iOS

Native Cross-Plattform

- **React-Native**
- NativeScript
- Xamarin

HTML5 Hybrid

Web-App in native Wrapper/WebView

- Cordova
- PhoneGap
- Ionic

Andere Hybrid

- JavaFX

Web App

Single-Page-App im Browser

Vergleich mit HTML5-Cross-Plattform

Gemeinsamkeiten

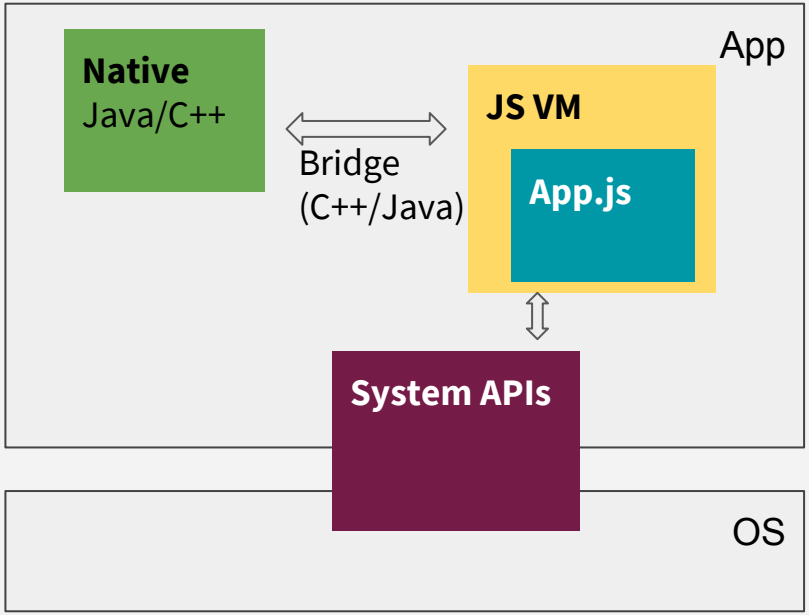
- JavaScript
- CSS*
- natives Package (apk, ipa)
- App Stores

Unterschiede

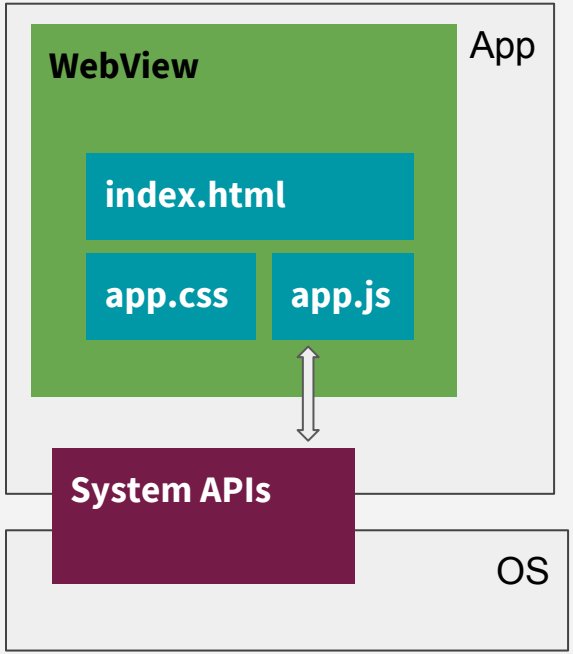
- kein HTML
- keine WebView
- React-Native rendert native Componenten

Vergleich mit HTML5-Cross-Plattform

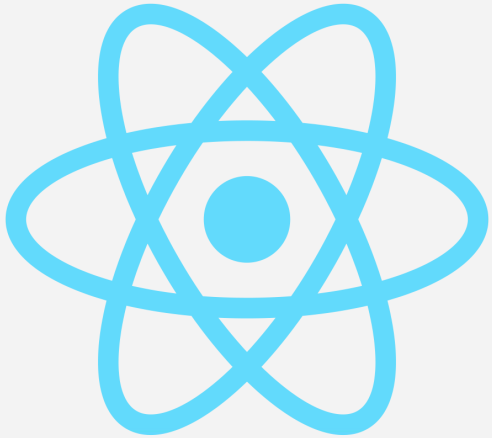
React-Native



Hybrid App



React-Native

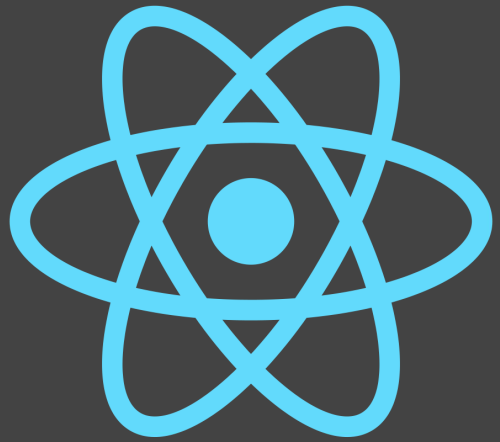


React → Erstellung von UI-Komponenten

React-Native → React für native Anwendungen zu nutzen

Entwickelt von Facebook seit 2013

OpenSource seit 2015



React

React DOM

```
import React from 'react'

class HelloWorld extends React.Component {
    render() {
        return (
            <div>
                <p>Hello {this.props.name}</p>
            </div>
        )
    }
}

// usage
<HelloWorld name='Hugo' />
```

React DOM

```
import React from 'react'

const HelloWorld = (props) => (
    <div>
        <p>Hello {props.name}</p>
    </div>
)

// usage
<HelloWorld name='Hugo' />
```

Composition

```
import React from 'react'

const HelloWorld = (props) => (
    <div>
        <p>Hello {props.name}</p>
    </div>
)

const HelloList = (props) => (
    <ul>
        <li><HelloWorld name='Hugo' /></li>
        <li><HelloWorld name='Marlene' /></li>
        <li><HelloWorld name='Luise' /></li>
    </ul>
)
```

```
import React from 'react'

const HelloWorld = (props) => (
    <div>
        <p>Hello {props.name}</p>
    </div>
)

const HelloList = (props) => (
    <ul>
        { props.names.map(name => <li><HelloWorld name={name} /></li>) }
    </ul>
)

// usage
<HelloList names={['Hugo', 'Marlene', 'Luise']} />
```

```
import React from 'react'

const HelloList = (props) => {
    if(props.names.length === 0) {
        return <p>Niemand da!</p>
    } else {
        return <ul>
                { props.names.map(name => (
                    <li>
                        <HelloWorld name={name} />
                    </li>
                )) }
            </ul>
    }
}

// usage
<HelloList names={['Hugo', 'Marlene', 'Luise']} />
```

```
import React from 'react'

const HelloWorld = (props) => (
    <div>
        <p>Hello {props.name}</p>
    </div>
)

const HelloList = (props) => (
    <ul>
        { props.names.map(name => <li><HelloWorld name='Hugo' /></li>) }
    </ul>
)

// usage
<HelloList names={['Hugo', 'Marlene', 'Luise']} />
```

React-Native

```
import React from 'react'
import { View, Text, FlatList } from 'react-native'

const HelloWorld = (props) => (
    <View>
        <Text>Hello {props.name}</Text>
    </View>
)

const HelloList = (props) => (
    <FlatList
        data={props.names}
        renderItem={({item}) => <HelloWorld name={item} />}
    />
)

// usage
<HelloList names={['Hugo', 'Marlene', 'Luise']} />
```

~~"Write once, run anywhere"~~

"Learn once, write anywhere"

Komponenten

View

Text

TextInput

DatePicker

Switch

Slider

Image

Button

Modal

StatusBar

Picker

ListView

ScrollView

WebView

APIs

Accessibility

Alert

Clipboard

Geolocation

Animation

Network-Info

Storage

Camera

Share

Timers

Vibration

Keyboard

Apps Bauen

Apps bauen

```
import { Text, View, AppRegistry } from 'react-native'

const App = (props) => (
    <View>
        <Text>Hallo Welt</Text>
    </View>
);


AppRegistry.registerComponent('myapp', () => App);
```

Apps bauen

```
import { Text, View, AppRegistry } from 'rea

const App = (props) => (
    <View>
        <Text>Hallo Welt</Text>
    </View>
);


AppRegistry.registerComponent('myapp', () =>
```

Styling

Styling

```
const styles = StyleSheet.create({
     container: {
          height: 500,
          backgroundColor: 'coral'
     },
     text: {
          fontSize: 40,
          color: '#fff8dc'
     }
})

const App = (props) => (
     <View style={styles.container}>
          <Text style={styles.text}>Hallo Welt>
     </View>
);
```

Styling

```
const styles = StyleSheet.create({
     container: {
          height: 500,
          backgroundColor: 'coral'
     },
     text: {
          fontSize: 40,
          color: '#fff8dc'
     }
})

const App = (props) => (
     <View style={styles.container}>
          <Text style={styles.text}>Hallo Welt>
     </View>
);
```

Layout

Layout? FlexBox!

```
const styles = StyleSheet.create({
     container: {
          flex: 1,
          justifyContent: 'center',
          backgroundColor: 'coral'
     },
     text: {
          fontSize: 40,
          color: '#fff8dc'
     }
})

const App = (props) => (
     <View style={styles.container}>
          <Text style={styles.text}>Hallo Welt>
     </View>
);
```

Layout? FlexBox!

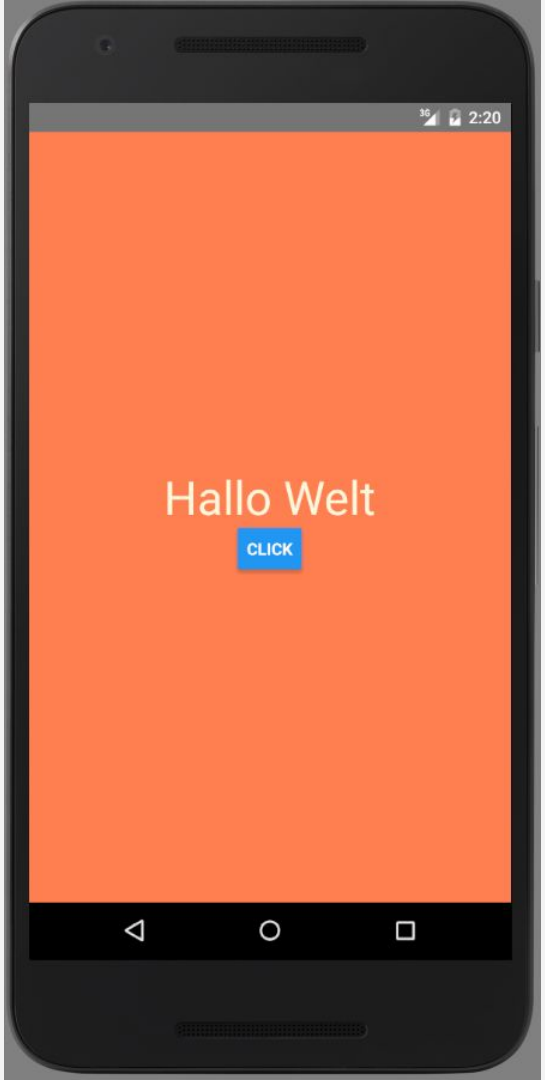
```
const styles = StyleSheet.create({
     container: {
          flex: 1,
          justifyContent: 'center',
          backgroundColor: 'coral'
     },
     text: {
          fontSize: 40,
          color: '#fff8dc'
     }
})

const App = (props) => (
     <View style={styles.container}>
          <Text style={styles.text}>Hallo Welt>
     </View>
);
```

Layout? FlexBox!

```
const styles = StyleSheet.create({
     container: {
          flex: 1,
          justifyContent: 'center',
          alignItems: 'center',
          backgroundColor: 'coral'
     },
     text: {
          fontSize: 40,
          color: '#fff8dc'
     }
})

const App = (props) => (
     <View style={styles.container}>
          <Text style={styles.text}>Hallo Welt>
     </View>
);
```



Plattform-Spezifische Komponenten

Plattformspezifische Komponenten

```
import { Text, Platform } fron 'react-native'

const HelloWorld = (props) => {
     let greeting;

     if(Platform.OS === 'ios') {
          greeting = 'Welcome on iOS';
     } else {
          greeting = 'Hi Android user';
     }

     return <Text>{greeting}</Text>
}
```

Plattformspezifische Komponenten

```
import { Text, Platform } fron 'react-native'

const HelloWorld = (props) => {
     let greeting;

     if(Platform.OS === 'ios') {
          greeting = 'Welcome on iOS';
     } else {
          greeting = 'Hi Android user';
     }

     return <Text>{greeting}</Text>
}
```

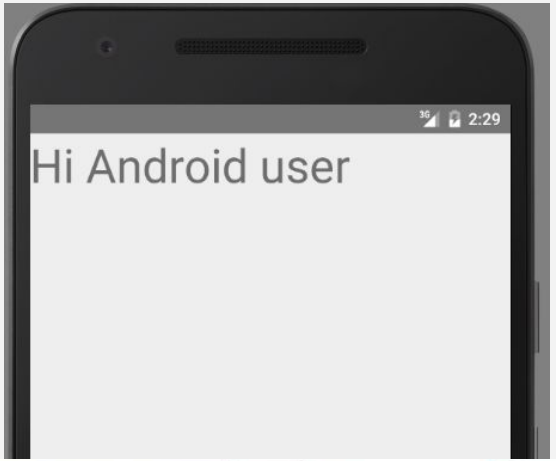
Plattformspezifische Komponenten

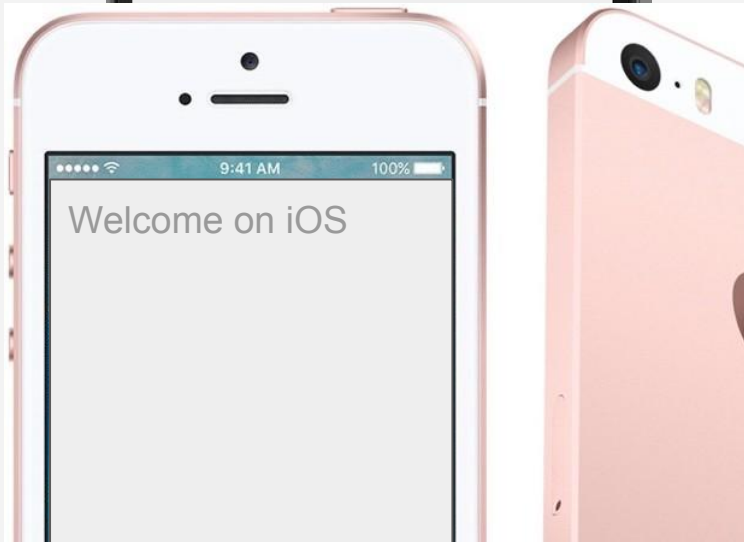
```
import { Text, Platform } fron 'react-native'

const HelloWorld = (props) => {
     let greeting;

     if(Platform.OS === 'ios') {
          greeting = 'Welcome on iOS';
     } else {
          greeting = 'Hi Android user';
     }

     return <Text>{greeting}</Text>
}
```



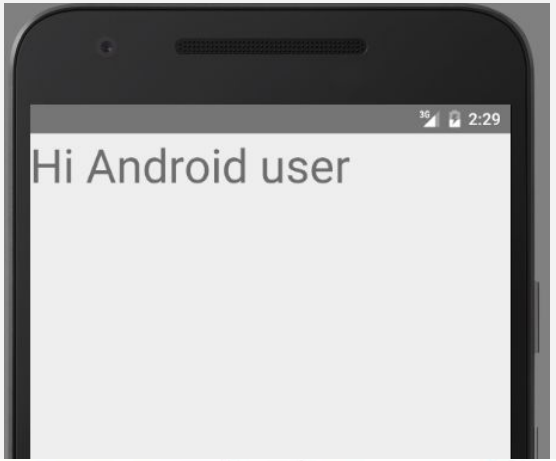


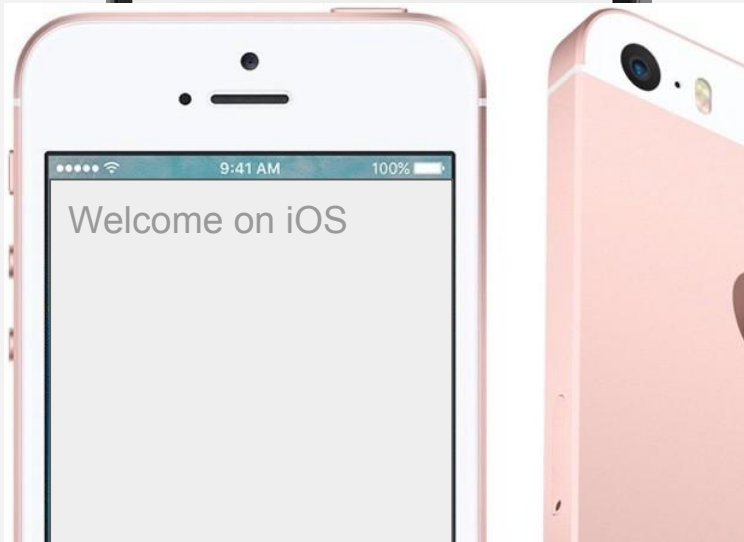
```
// HelloWorld.ios.js
const HelloWorld = (props) => (
     <Text>Welcome on iOS</Text>
)


// HelloWorld.android.js
const HelloWorld = (props) => (
     <Text>Hi Android user</Text>
)


// SomeOtherComponent.js
import HelloWorld from 'HelloWorld'

const SomeOtherComponent = (props) => (
     <View>
          <HelloWorld />
     </View>
)
```





Netzwerk

- Fetch API (Spec in Progress)

```
fetch('https://blubba.io/api')
    .then((response) => response.json())
    .then((json) => {
        console.log("data:", json)
    });
```

Demo

Fazit

Plattformen

React DOM: Browser

React-Native:

- Android, iOS offiziell Supported
- Windows 10 support von Microsoft
- MacOS durch Community
- Ubuntu durch Canonical :-(

Vorteile React-Native

- Eine Art zu Programmieren auf mehreren Plattformen
- React + Redux
- ~~OOP~~ → Funktional
- Dev-Tools
  - Time-Travel
  - Live-Reloading
- Aktive Community
- Schnelle Ergebnisse auf verschiedenen Plattformen

Vergleich mit Native

- Performance
  - prinzipiell vergleichbar
  - bei Native mehr Tuning-Möglichkeiten
- Mehr Componenten/APIs bei Native
- Näher an der Hardware
- Dev-Tools (GUI-Builder)

Vergleich mit HTML5-Cross-Platform

- unterstützte Plattformen
  - React-Native: Android + iOS
  - Cordova: Android, iOS, Windows Phone, BlackBerry, Symbian, …
- Performance
- Look and Feel

Vergleich mit Browser-Single-Page-Apps

- kein AppStore (Vorteil und Nachteil)
- Auffindbar über Web
- unterstützte Plattformen
- Performance

Fragen?

@manuel_mauky
github.com/lestard
www.lestard.eu

Saxonia Systems
So geht Software.