NATIVE APP-ENTWICKLUNG "THE FACEBOOK WAY"

- HANDS-ON REACT-NATIVE

10.06.2017

WILLKOMMEN!

- AppWithUs
- Wir entwickeln professionelle mobile Applikationen mit React Native
- > 10 Apps entwickelt
  - z.B. Avipeo - soziales Netzwerk für die Luftfahrt
  - SnapHere - finde Snapchatter in deiner Umgebung
- sitzen in Hamburg (Altona)

WER WIR SIND

Sebastian Germesin

Sebastian Gronewold

Ansgar Mertens

Dominic Melius

Christoph Byza

WAS PASSIERT HIER HEUTE

• kurze Einführung in Javascript (ES6)	• 20 min	• Sebastian Ge.
• Theoretische Einführung in React + React Native	• 40 min	• Sebastian Ge.
• Livecoding-Vorführung	• 45 min	• Sebastian Gr.
• App-Projekt vorstellen	• 20 min	• Christoph By.
• <lunch-break> (ca. 12:45 - 13:30)	• 45 min	• everyone
• Coding (ca. 13:30 - 18:30)	• 5 Stunden	• everyone
• Debriefing	• 30 min	• Sebastian Ge.

NOCH KURZ DAVOR

- Es gibt Pizza / Salat / Nudeln heute mittag, das müssen wir mal kurzen "organizen"
- Zudem wollen wir mal kurz abcheck, wer JS/ React/ usw kann...

JAVASCRIPT ES6

- new bindings
- destructuring of objects
- arrow functions
- classes
- import

JAVASCRIPT - ES6 BINDINGS

```
1   var x = 5;
2   x = 3;
3   console.log(x); // => 3
4
5   let y = 19;
6   y = 18;
7   console.log(y); // => 18
8   let y = 17; // => SyntaxError: Identifier 'y' has already been declared
9
10  const z = 5;
11  z = 2; // => TypeError: Assignment to constant variable.
12
```

JAVASCRIPT - ES6 DESTRUCTURING

```
let cat = {
  name: 'Kitty',
  age: 3,
  height: 31
};

console.log(cat.name); // => 'Kitty'
console.log(cat['age']); // => 3

delete cat.height;
console.log(cat.height) // => undefined
```

```
let { name } = cat;
console.log(name); // => 'Kitty'

let othercat = {
  ...cat,
  name: 'Garfield',
};
console.log(othercat.name); // => 'Garfield';
console.log(othercat.height); // => 31;
```

JAVASCRIPT - ES6 ARROW FUNCTIONS

```
1  function mult (a, b) {
2    return a * b;
3  }
4  console.log(mult(4,5)); // => 20
5
6  //ES6
7  const multES6 = (a, b) => a * b;
8  console.log(multES6(4,5)); // => 20
9
```

```
1  let x = ['ReactJS', 'ReactNative', 'Cordova'];
2  let y = x.filter(n => n.indexOf('React') === 0);
3  let z = x.filter(n => !n.indexOf('React')); // => nice, but not understandable for others
4  console.log(y); // => [ 'ReactJS', 'ReactNative' ]
5  console.log(z); // => [ 'ReactJS', 'ReactNative' ]
6
```

JAVASCRIPT - ES6 CLASSES

```
class Point {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }

  toString() {
    return '(' + this.x + ', ' + this.y + ')';
  }
}
```

```
class ColorPoint extends Point {
  constructor(x, y, color) {
    super(x, y);
    this.color = color;
  }

  toString() {
    return super.toString() + ' in ' + this.color;
  }
}

let cp = new ColorPoint(25, 8, 'green');
console.log(cp.toString()); // '(25, 8) in green'

console.log(cp instanceof ColorPoint); // true
console.log(cp instanceof Point); // true
```

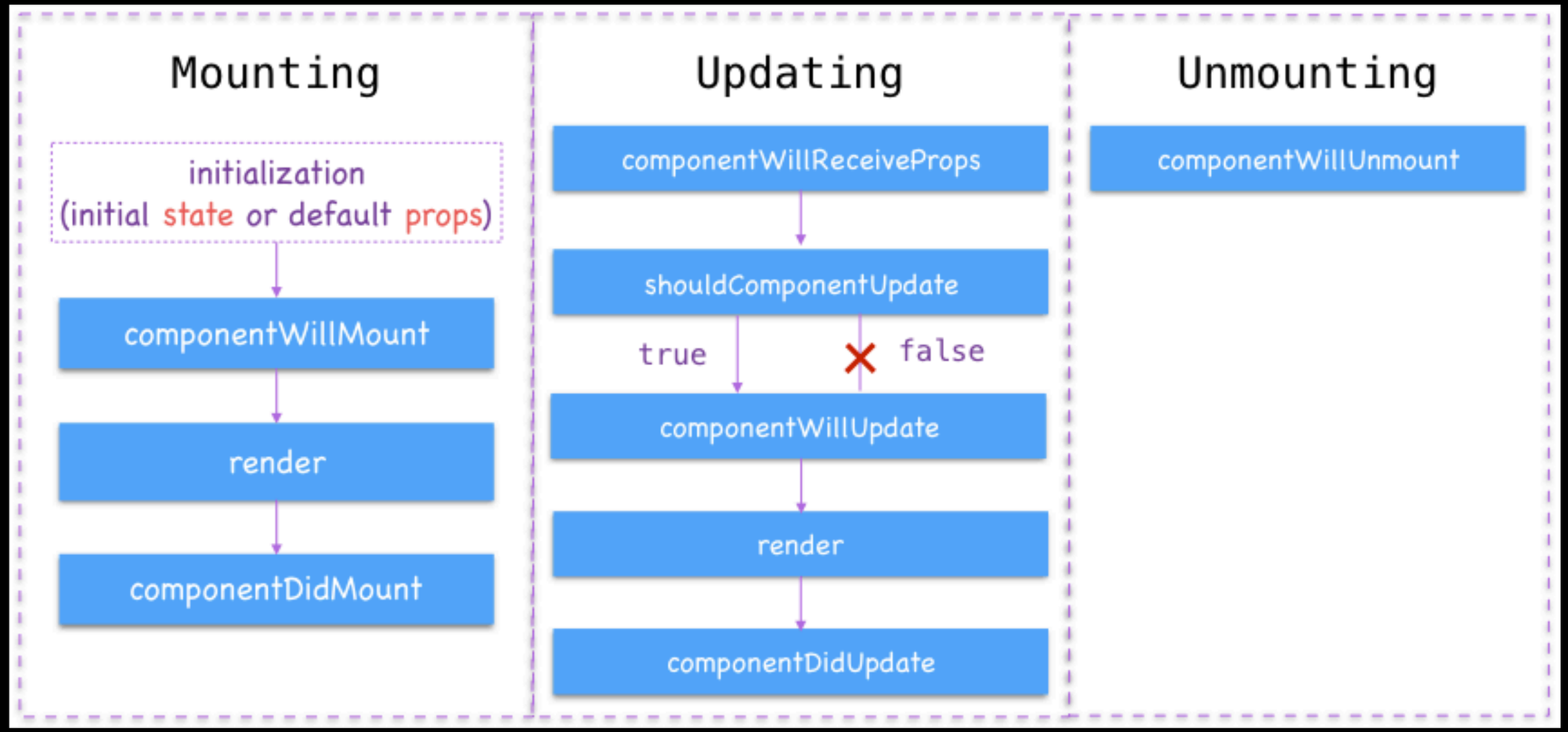
JAVASCRIPT - ES6 MODULE IMPORT/EXPORT

```
1  module.exports = {
2    someVal: 42,
3    someFn: () => console.log('test')
4  };
5
6  import { someVal } from './somefile';
7  import SomeModule from './somefile';
8  // someVal, SomeModel.someFn();
9
```

EINFÜHRUNG REACTJS

- ReactJS wurde 2011 von Facebook entwickelt und 2013 "open-sourced"
- Hauptbestandteile:
  - Flexible (kleine) JavaScript library
  - Große Community (> 10.000 npm repositories)
  - JSX
- Seit 2015 auch erweitert für Mobile Anwendungen (React Native)

COMPONENT - LIFECYCLE



COMPONENT - LIFECYCLE

```
 1 import .... Component {
 5
10   componentWillMount() {
11     ...
12   }
13
14   componentDidMount() {
15     ...
16   }
17
18   componentWillReceiveProps(nextProps) {
19     ...
20   }
...
41 }
```

```
 1 import .... Component {
 5
22   componentWillUpdate(nextProps, nextState) {
23     ...
24   }
25
26   shouldComponentUpdate(nextProps, nextState) {
27     ...
28   }
29
30   componentWillUnmount() {
31     ...
32   }
33
34   render() {...
40   }
41 }
```

WAS IST JSX

- Abkürzung von JavaScript extension syntaX
- https://facebook.github.io/jsx/

> "THE PURPOSE OF THIS SPECIFICATION IS TO DEFINE A CONCISE AND FAMILIAR SYNTAX FOR DEFINING TREE STRUCTURES WITH ATTRIBUTES. A GENERIC BUT WELL DEFINED SYNTAX ENABLES A COMMUNITY OF INDEPENDENT PARSERS AND SYNTAX HIGHLIGHTERS TO CONFORM TO A SINGLE SPECIFICATION."

JSX - BEISPIEL

```
1  var dropdown =
2    <Dropdown>
3      A dropdown list
4      <Menu>
5        <MenuItem>Do Something</MenuItem>
6        <MenuItem>Do Something Fun!</MenuItem>
7        <MenuItem>Do Something Else</MenuItem>
8      </Menu>
9    </Dropdown>;
10
11 render(dropdown);
```

KLEINES BEISPIEL

```
1   import React from 'react';
2   import { render } from 'react-dom';
3
4   const Todo = ({todo}) => {
5     return (<a href="#" className="list-group-item">{todo.text}</a>);
6   }
7
8   const TodoList = ({todos}) => {
9     const todoNode = todos.map(todo => {
10      return (<Todo todo={todo} key={todo.id}/>)
11    });
12    return (
13      <div className="list-group">
14        {todoNode}
15      </div>
16    );
17  }
18
```

```
19
20  class TodoApp extends React.Component{
21
22    state = {
23      data: []
24    }
25
26    componentDidMount(){
27      //TODO: load data from API
28    }
29
30    render() {
31      return (
32        <div>
33          <TodoList
34            todos={this.state.data}
35          />
36        </div>
37      );
38    }
39  }
40  render(<TodoApp />, document.getElementById('container'));
```

EINFÜHRUNG REACT NATIVE

React Native hat die gleichen Ansätze wie ReactJS, aber ist von der darunterliegenden Technik sehr unterschiedlich - mit Absicht 🤓

REACT NATIVE VS. IONIC/XAMARIN/...

- React Native benutzt weitestgehend native Elemente und steuert diese über JavaScript an. Daher ergibt sich eine **natürliche** Reaktionszeit von UI-Elementen

- Hybride Apps auf Basis von Ionic o.ä. basieren auf einer WebView, welche native Elemente nachbildet. Daraus erfolgt ein leicht-verzögertes UI.

REACT NATIVE VS. TRULY-NATIVE

- React Native setzt auf einen Ansatz, der es ermöglicht, Apps für Android und iOS zu bauen und dabei den Großteil des geschriebenen Codes wiederzuverwenden (> 80%).

- "Truly Native" bedeutet, eine App für iOS Geräte in z.B. Swift zu entwickeln und anschließend, die gleiche App in Java für das Android System zu bauen. Dadurch entsteht ein Overhead von 100%, welcher die Entwicklung sehr teuer macht und Releasezyklen vergrößert.

COMPONENT - INIT

```
 1 import React, { Component } from 'react';
 2 import { View, Platform } from 'react-native';
 3
 4 export default class Moin extends Component {
...
24 });
```

COMPONENT - RENDERING

```
 1 import ...
 3
 4 export default class Moin extends Component {
...
10   render() {
11     return (
12       <View style={{ backgroundColor: 'red'}}>
13       </View>
14     );
15   }
16 };
```

COMPONENT - STATE

```
 1 import ...
 3
 4 export default class Moin extends Component {
 5
 6   state = {
 7     text: 'Hello',
 8   }
 9
10   render() {
11     return (
12       <View style={...}>
13         <Text>{this.state.text}</Text>
14       </View>
15     );
16   }
17 }
```

COMPONENT - UPDATE STATE

```
1  import ...
3
4  export default class Moin extends Component {
5
6    state = {
7      text: 'Hello',
8    }
9
10   componentDidMount() {
11     this.setState({
12       text: 'Test',
13     });
14   }
```

COMPONENT - PROPERTIES

```
 1 import ...;
 2
 3 export default class Moin extends Component {
 4
 5   render() {
 6     return (
 7       <View style={styles.container}>
 8         <Text>{this.state.text}</Text>
 9       </View>
10     );
11   }
12 }
```

COMPONENT - PROPERTIES

```
 1 import ...;
 2
 3 export default class Moin extends Component {
 4
 5   render() {
 6     return (
 7       <View style={styles.container}>
 8         <SomeComponent
 9           value={this.state.text}>
10         </SomeComponent>
11       </View>
12     );
13   }
14 }
```

COMPONENT - UPDATE STATE

```
 1 import ...;
 2
 3 export default class SomeComponent extends
 4 Component {
 5
 6   render() {
 7     return (
 8       <Text>
 9         {this.props.value}
10       </Text>
11     );
12   }
13 }
```

COMPONENT - TOUCHABLE ELEMENTS

```
1 import ...;
2
3 export default class SomeComponent extends
4 Component {
5
6   render() {
7     return (
8       <TouchableOpacity onPress={this.props.onClick}>
9         ...
10      </TouchableOpacity>
11    );
12  }
13 }
```

COMPONENT - CHILD -> PARENT

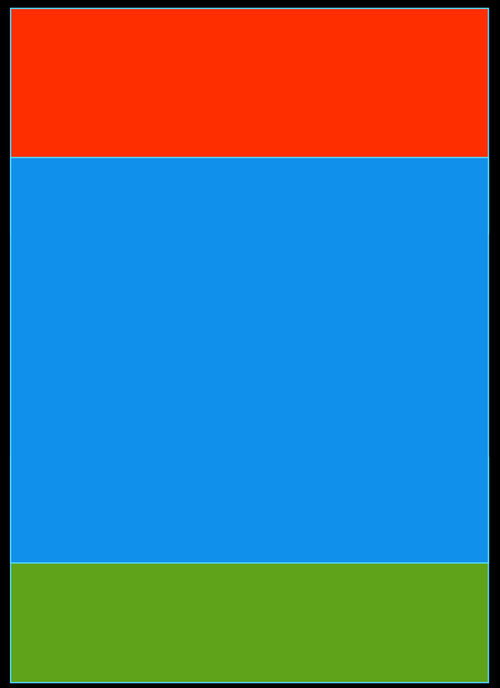
```
 1 import ...;
 2
 3 export default class Moin extends
 4 Component {
 5
 6   render() {
 7     return (
 8       <SomeComponent
             onClick={() => this.setState({text: 'test'})}>
 9         ...
10       </SomeComponent>
11     );
12   }
13 }
```

COMPONENT - DIMENSIONS, PLATFORM

```
 1 import { Dimensions, Platform } from 'react-native'
 3
 4 export default class Moin extends Component {
 5
 6   render() {
 7     return (
 8       <Text>
 9         {Dimensions.get('window').height}
10       </Text>
11       <Text>
12         {Dimensions.get('window').width}
13       </Text>
14       <Text>
15         {Platform.OS}
16       </Text>
17     );
18   }
19 }
```

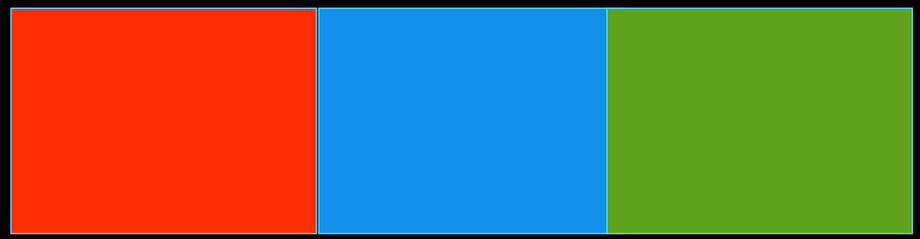
COMPONENT - FLEXBOX

```
1 <View style={{flex: 1}}>
2   <View style={{flex: 1, backgroundColor: 'red'}} />
4   <View style={{flex: 2, backgroundColor: 'blue'}} />
6   <View style={{flex: 1, backgroundColor:  green'}} />
8 </View>
```



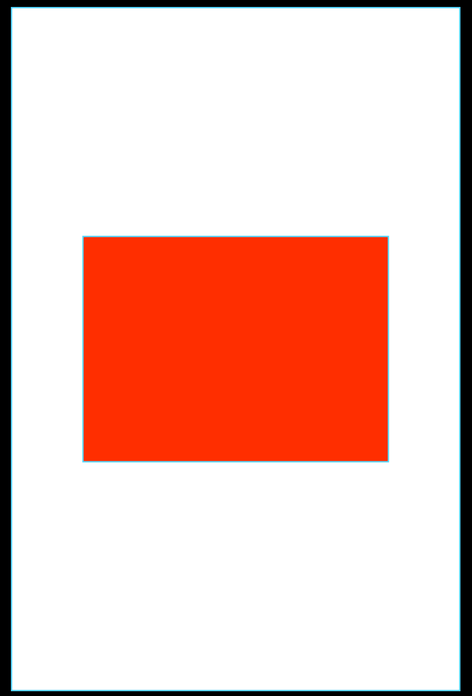
COMPONENT - FLEXBOX

```
1 <View style={{flex: 1, flexDirection: 'row'}}>
2   <View style={{flex: 1, backgroundColor: 'red'}} />
4   <View style={{flex: 1, backgroundColor: 'blue'}} />
6   <View style={{flex: 1, backgroundColor:  green'}} />
8 </View>
```



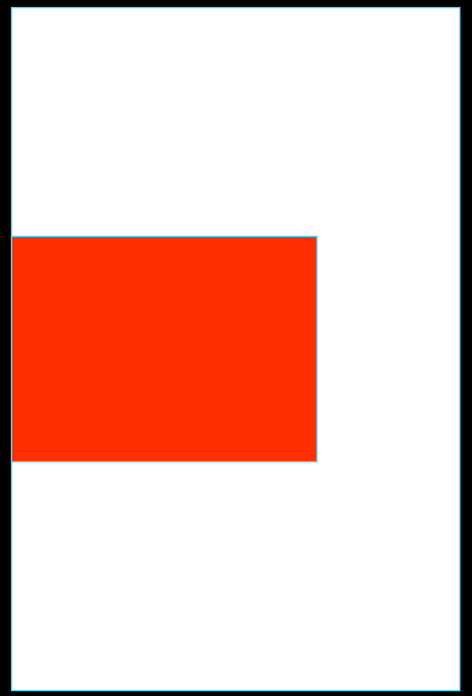
COMPONENT - FLEXBOX

```
1 <View style={{flex: 1, alignItems: 'center', justifyContent: 'center'}}>
3   <View style={{width: 40, height: 40, backgroundColor: 'red'}} />
5 </View>
```



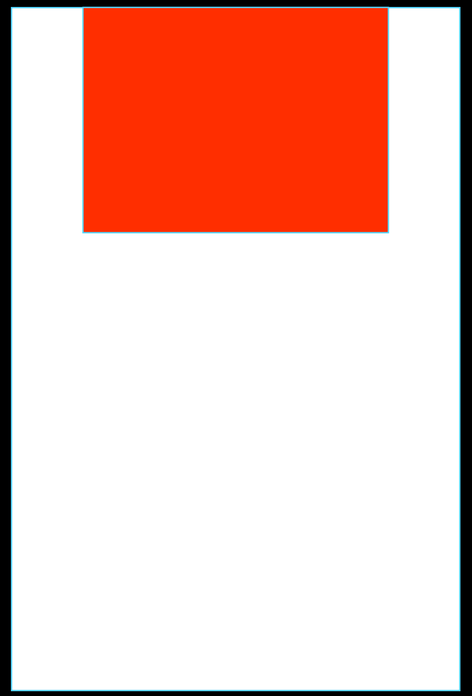
COMPONENT - FLEXBOX

```
1 <View style={{flex: 1, alignItems: 'flex-start', justifyContent: 'center'}}>
3   <View style={{width: 40, height: 40, backgroundColor: 'red'}} />
5 </View>
```



COMPONENT - FLEXBOX

```
1 <View style={{flex: 1, alignItems: 'center', justifyContent: 'flex-start'}}>
3   <View style={{width: 40, height: 40, backgroundColor: 'red'}} />
5 </View>
```



COMPONENT - LISTVIEW

```
1  import { ListView } from 'react-native';
2
3  export default class ChatMessages extends Component {
4
5    state = {
6      dataSource: new ListView.DataSource({rowHasChanged: (r1, r2) => r1 !== r2}),
8    };
9
10   renderRow = (data) => {
11     return <View />
12   };
13
14   render() {
15     let { dataSource } = this.state;
16     return (
17       <ListView
18         dataSource={dataSource.cloneWithRows(this.props.list)}
19         renderRow={this.renderRow}
20       />
21     );
22   }
23 }
```

COMPONENT - ~~LISTVIEW~~ FLATLIST

```
1  import { FlatList } from 'react-native';
2
3  export default class ChatMessages extends Component {
4
5    // FYI: this.props.list = [{key: 'a'}, {key: 'b'}];
6
8
9
10   renderRow = (info) => {
11     return <View>...</View>;
12   };
13
14   render() {
15
16     return (
17       <FlatList
18         data={this.props.list}
19         renderItem={this.renderRow}
20       />
21     );
22   }
23 }
```

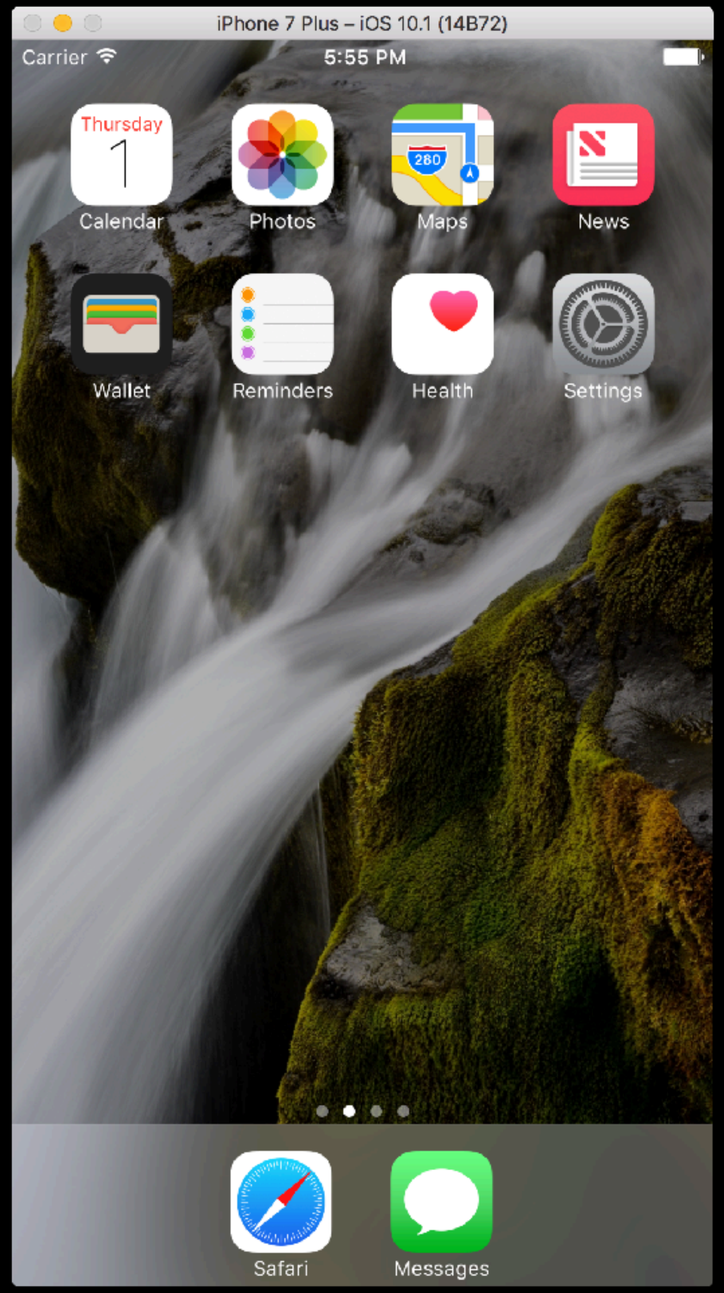
Soviel zur Theorie.....

Jetzt kommt LiveCoding.....

Jetzt kommt das Briefing.....

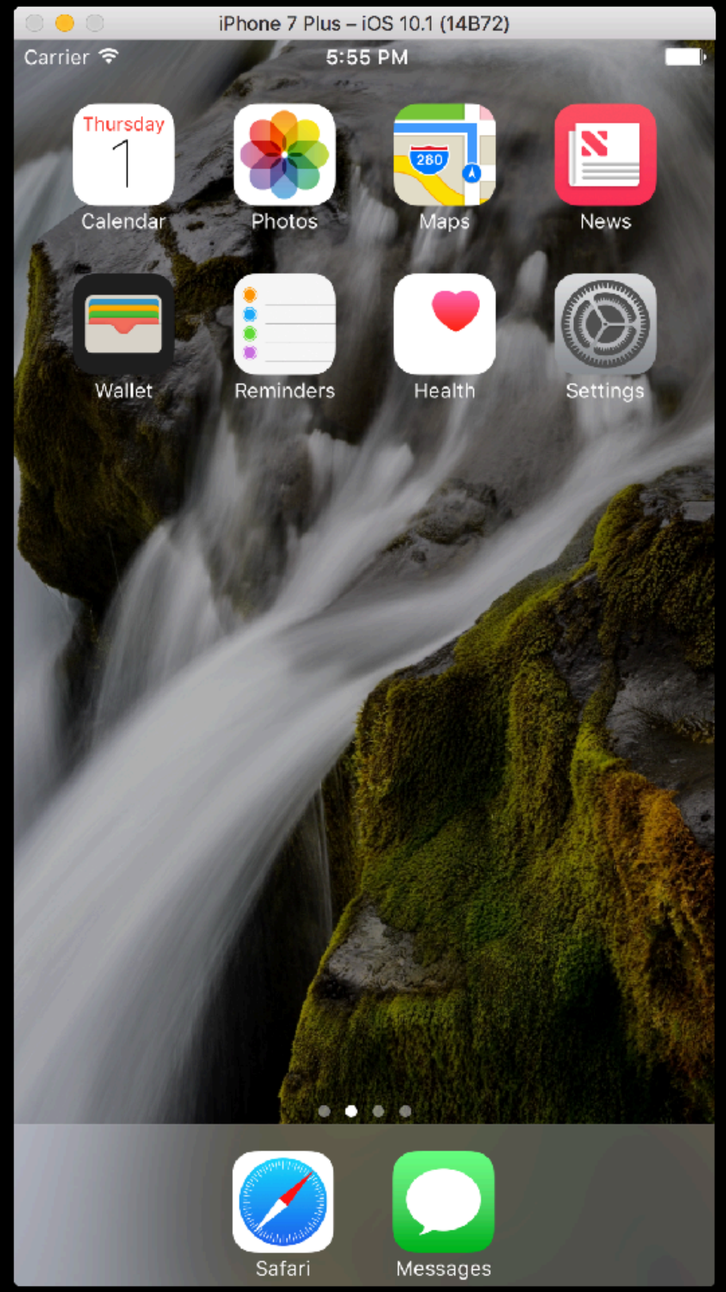
HANDS-ON BRIEFING

- Wir entwickeln gemeinsam eine Multi-Timer-App
  - Einrichten mehrerer Timer
  - Starten/Stoppen/Resetten
  - Performance-Optimierung



HANDS-ON BRIEFING

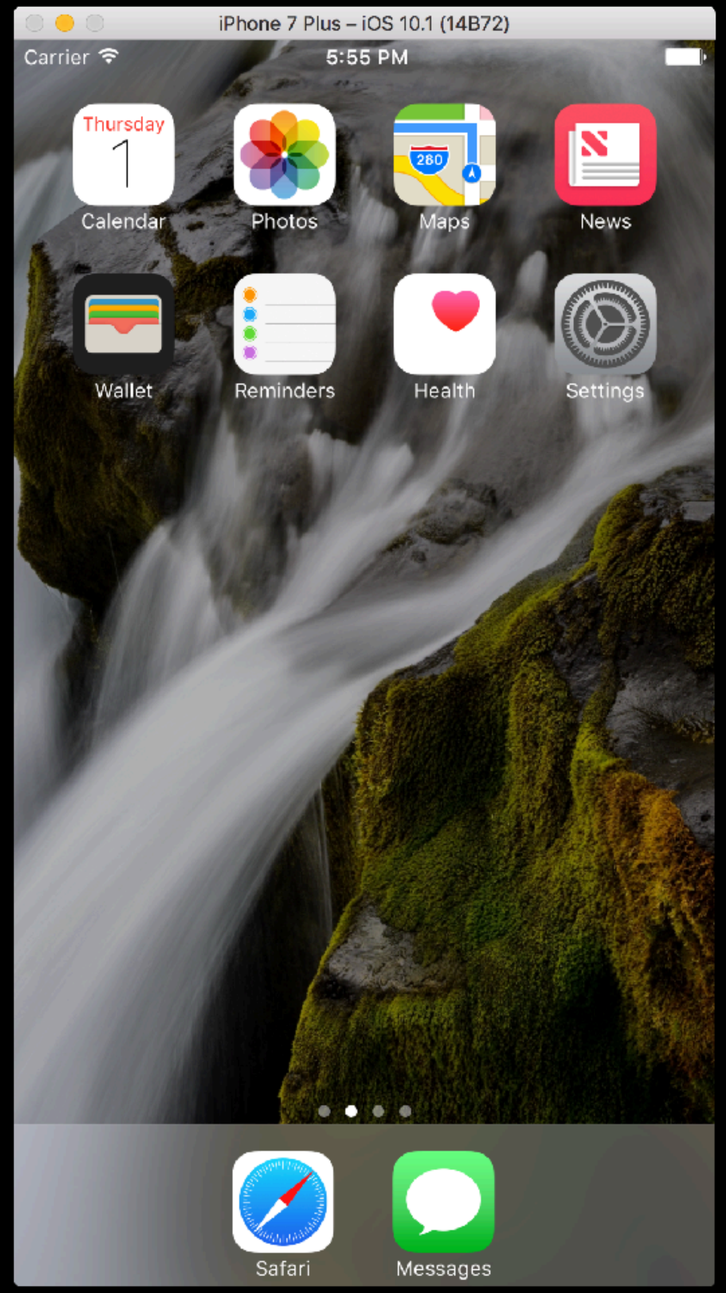
- möglichst Gruppenarbeit
  - 1-2 Personen pro Rechner
- wir "kommen rum" und helfen euch dabei



HANDS-ON BRIEFING

- **Step 1**: Installiert alle Dependencies
  - (stehen auch in der README.md)

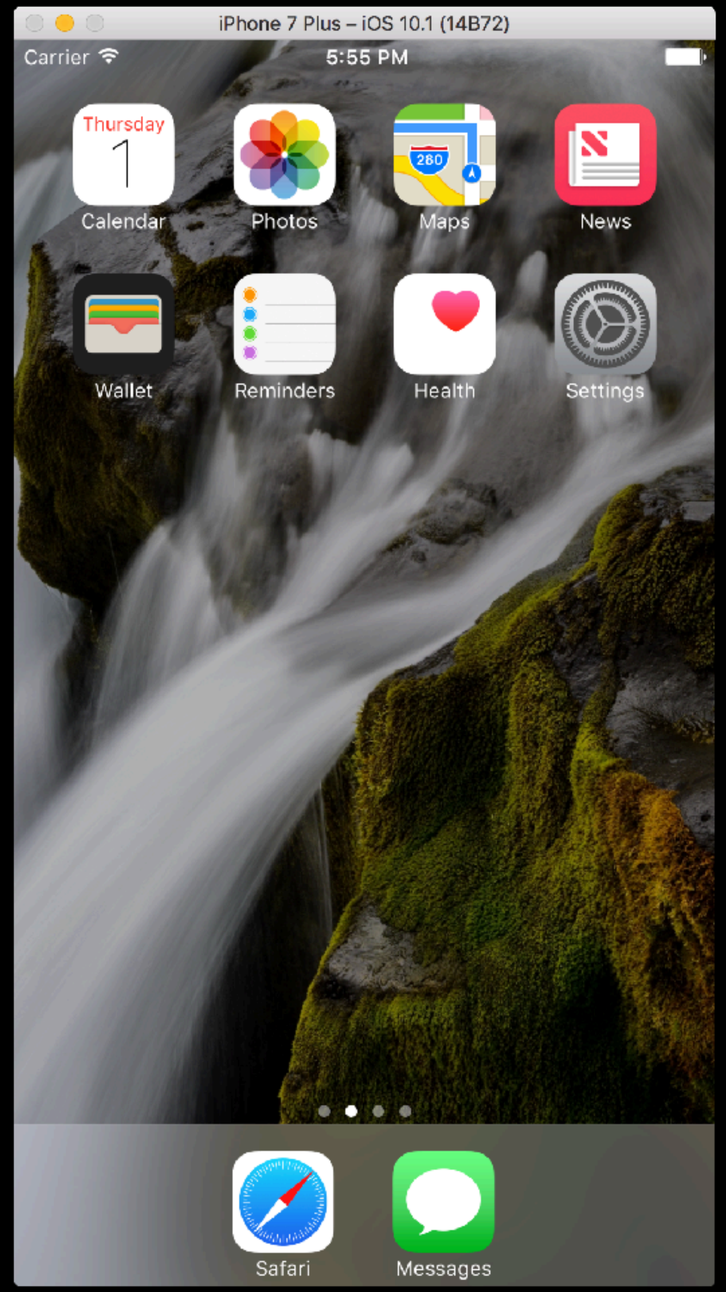
```
# install homebrew

/usr/bin/ruby -e "$(curl -fsSL https://
raw.githubusercontent.com/Homebrew/install/master/
install)"

brew install node watchman

npm install -g react-native-cli

npm install
```



HANDS-ON BRIEFING

- **Step 2**: Create App Starter

```
react-native init MultiTimer
```



HANDS-ON BRIEFING

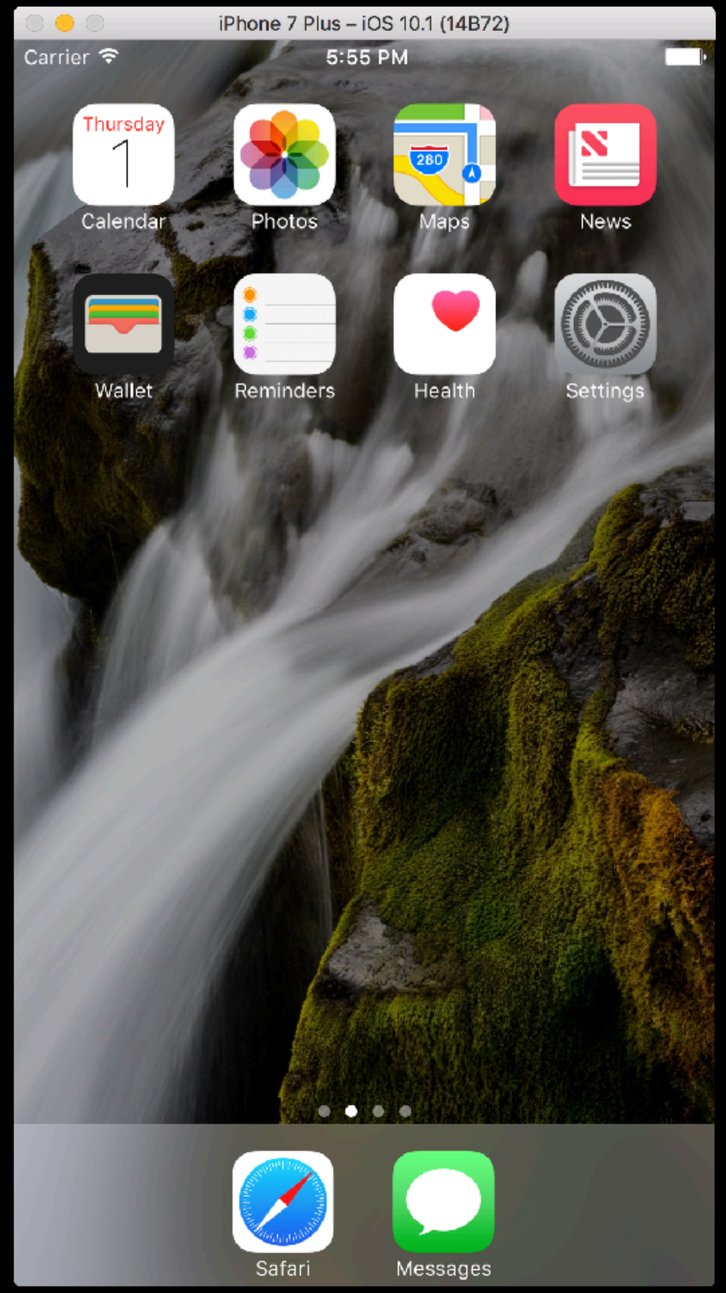
- **Step 3**: Im Simulator/Emulator installieren

iOS

```
react-native run-ios
```

Android

```
# either connect your
Android device via USB, or
start an emulator

android avd

react-native run-android
```

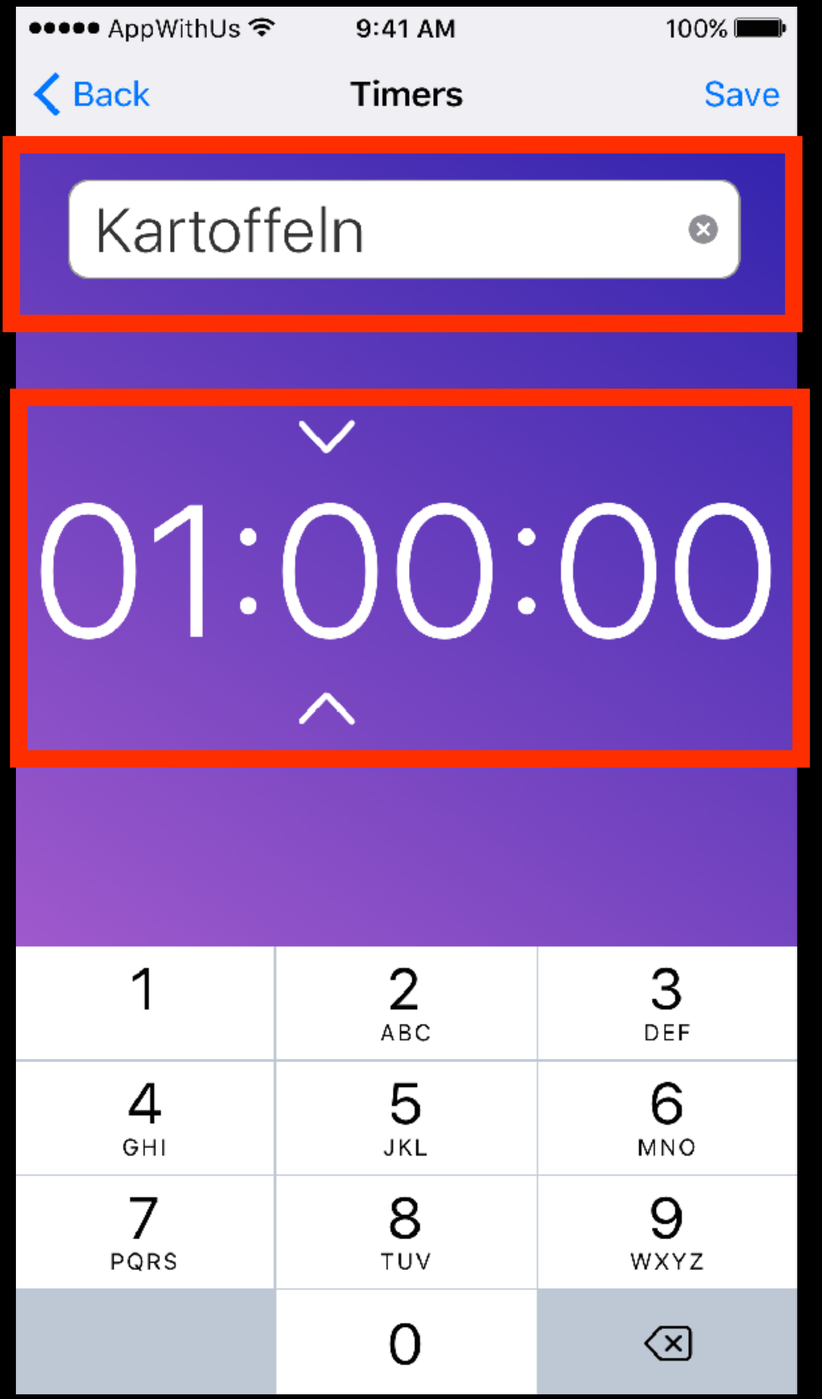


HANDS-ON BRIEFING

- **Step 4**: MainView
- Anforderungen:
  - Liste der aktuellen Timer mit jeweiligem Status
  - Möglichkeit einen Timer zu erstellen
- Pluspunkte:
  - Hinblick auf Performance

HANDS-ON BRIEFING

- **Step 5**: CreateView
- Components:
  - NameInput
  - TimerInput
- Stolperfallen:
  - KeyboardAvoidingView
  - Properties / State



LOS GEHT'S

DEMO

- Jeder zeigt seine Lösung und berichtet kurz!
  - 5 Minuten pro Team

DEBRIEF

- Wir wollen wissen:
  - Was habt ihr gelernt?
  - Was lief gut?
  - Was lief nicht so gut?
  - Was sollte das nächste Mal anders sein?

VIELEN DANK