andrena OBJECTS

Experts in agile software engineering

Hybride App-Entwicklung mit React-Native, React und Redux

Ein Erfahrungsbericht

Agenda

- Motivation und Entscheidung
- Einführung in Frameworks
- Qualitätssicherung
- Demo
- Zusammenfassung

Erwartungen an den Vortrag

- Was bietet der Vortrag?
  - Erfahrungsbericht
  - Überblick über die Frameworks
  - Erfahrungen/Tipps im Bereich Testing
- Was bietet der Vortrag nicht?
  - Jahrelange Erfahrung mit react-native
  - Jahrelange Erfahrung in der App-Entwicklung

Motivation und Entscheidung

Motivation

- Gemeinsame Codebasis für Android und iOS
- Entwicklung von nativen Apps ohne Vorkenntnisse
- Entwicklung in bekannter Sprache (JavaScript)
- Einsatz moderner Technologien

Auf der Suche nach dem passenden Framework

Warum React-Native

- Pro
  - Große Community
  - Test-Driven-Development
  - Redux (Trennung von Logik & Anzeige)
- Contra
  - Hohe Einstiegshürde

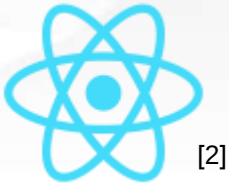
[1]

andrena
OBJECTS

Einführung in die Frameworks

React, React-Native und Redux

ReactJS

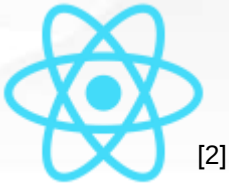


[2]

JavaScript-Framework für (Web)-User-Interfaces

- Ermöglicht deklarative Views
- Komponenten-basiert
- Effizientes Rendering durch Virtual DOM
- Deterministisches Rendering
- Ist *nur* die View
- Ermöglicht JSX

Was ist JSX?



[2]

Syntactic sugar für Markup in JavaScript

Ohne JSX:

```
const MyReactComponent = () =>
    React.createElement('div', { id: 42 }, 'A message');
```

Mit JSX:

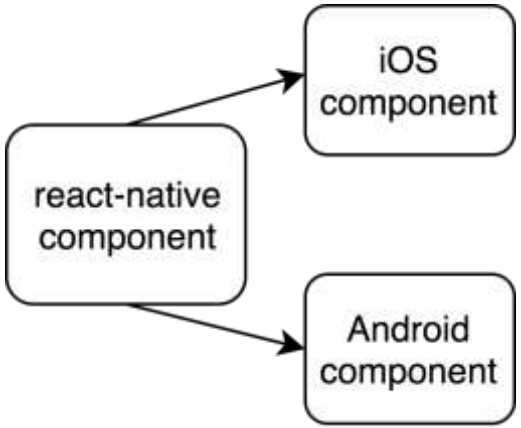
```
const MyReactComponent = () =>
    <div id="42">A Message</div>;
```

React-Native

[1]

Framework zum Bau nativer Apps in JavaScript

- Liefert React-Native-JavaScript-Komponenten
- Glue-Code zwischen JavaScript und Android/iOS
- Ermöglicht eigene React-Native-Komponenten
- Styling mit CSS und insb. Flexbox
- Buildtools
- Entwicklungstools



React-Native

[1]

Framework zum Bau nativer Apps in JavaScript

Beispiel-Komponenten

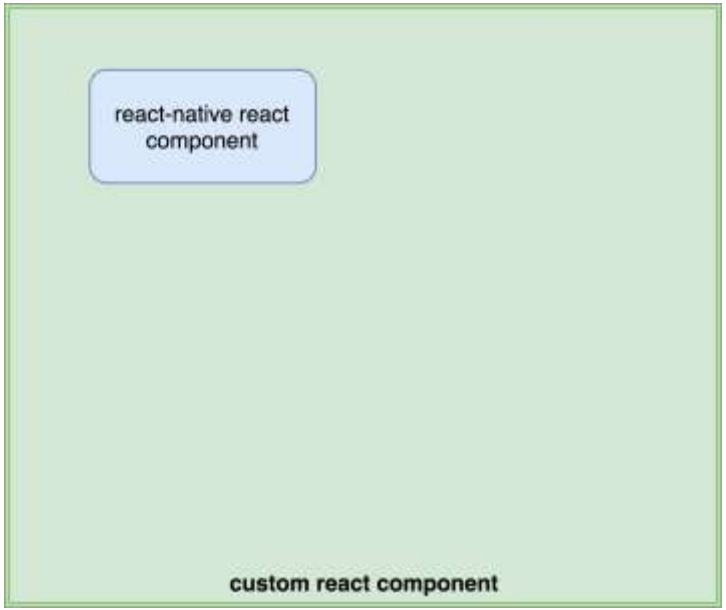
- Text, Button, InputField, …
- Slider, Picker, ActivityIndicator, ...
- View, ScrollView, FlatList, ...

Beispiel-APIs

- Vibration
- Gelocation
- AsyncStorage

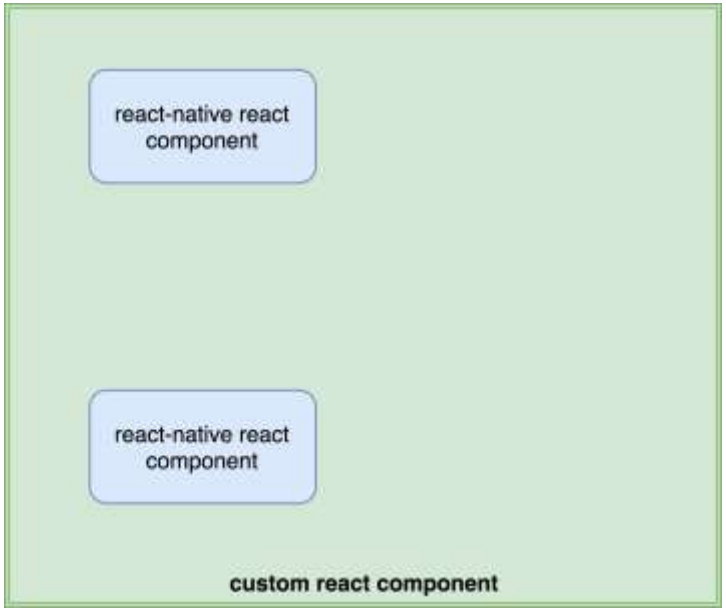
React-Native + React

React ist der Glue-Code der React-Native-Komponenten



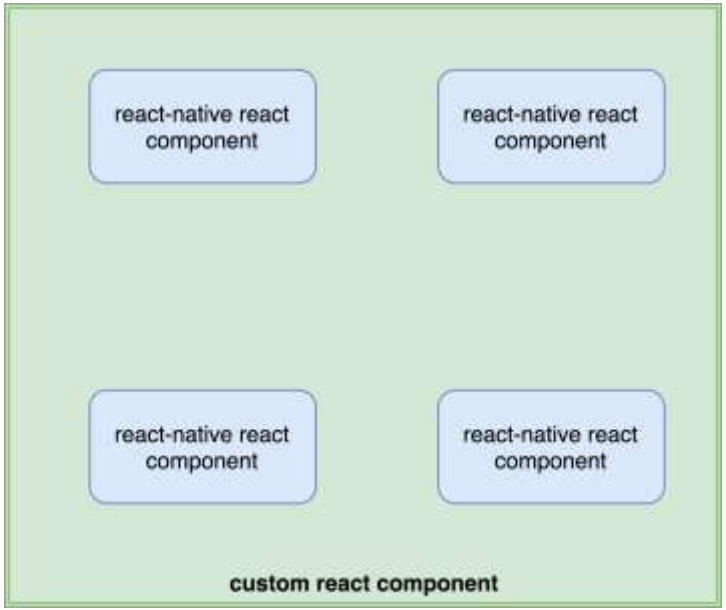
React-Native + React

React ist der Glue-Code der React-Native-Komponenten



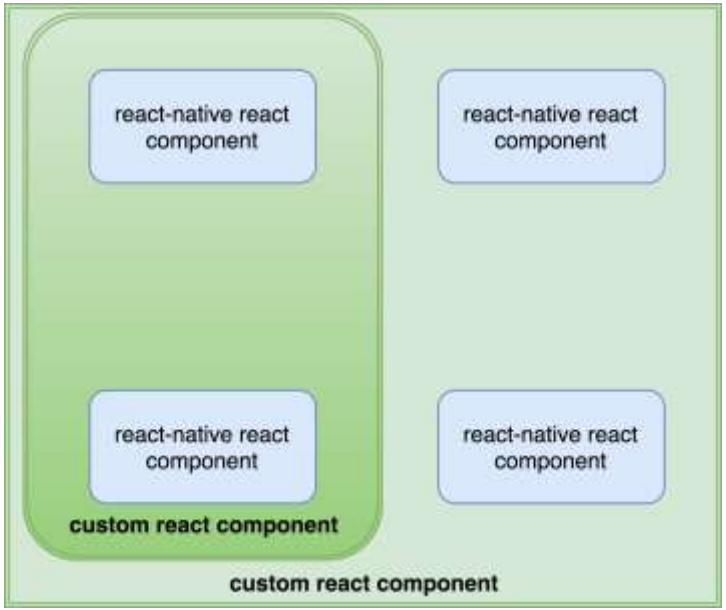
React-Native + React

React ist der Glue-Code der React-Native-Komponenten



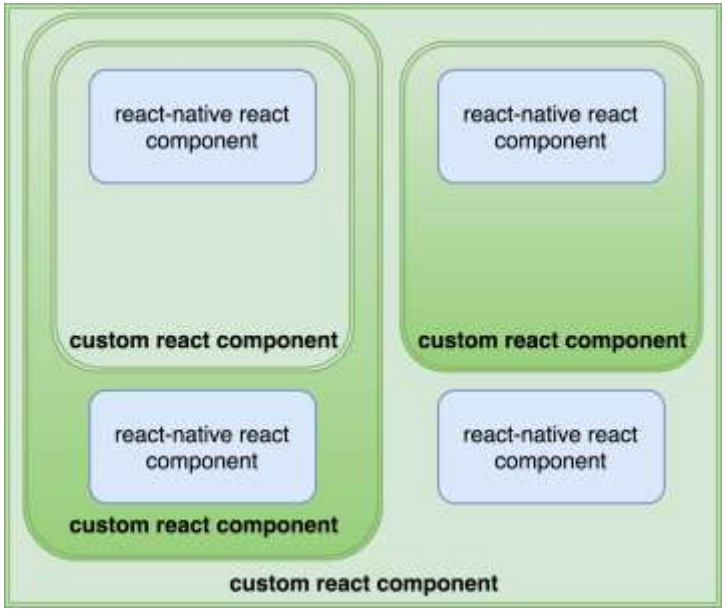
React-Native + React

React ist der Glue-Code der React-Native-Komponenten



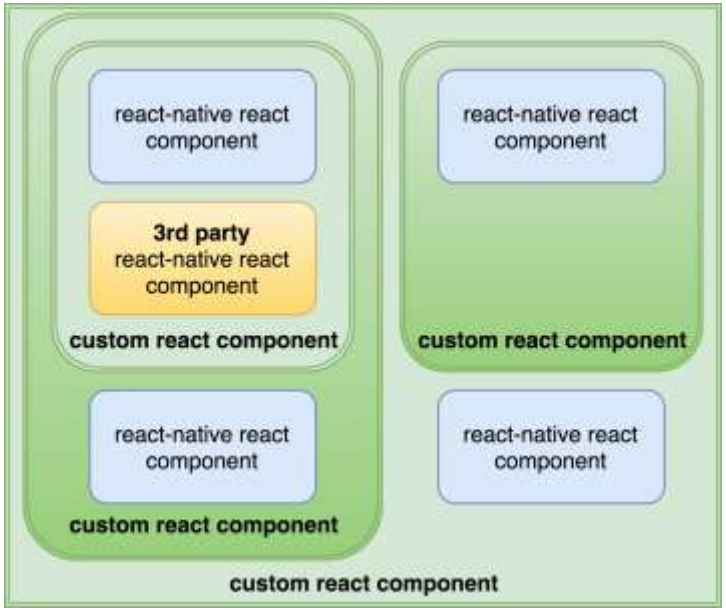
React-Native + React

React ist der Glue-Code der React-Native-Komponenten



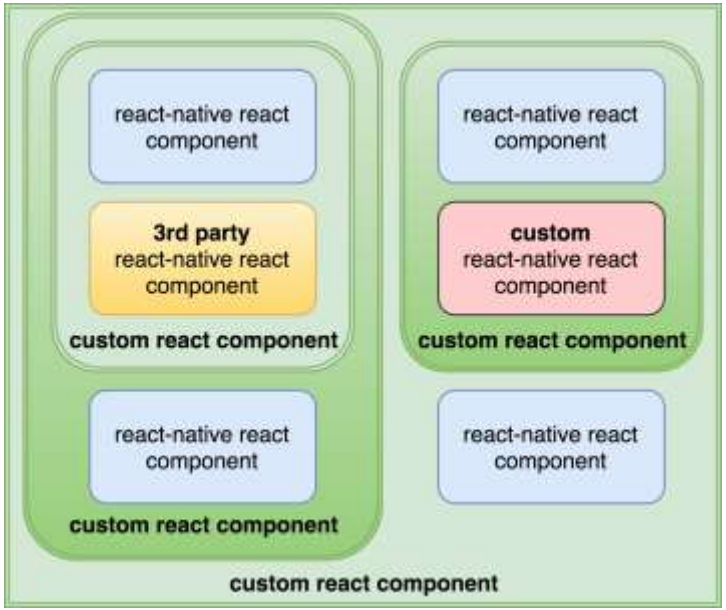
React-Native + React

React ist der Glue-Code der React-Native-Komponenten



React-Native + React

React ist der Glue-Code der React-Native-Komponenten



React-Native + React

Warte...

Wenn React und React-Native *nur* die View bilden,
was ist das dann mit meinen
Daten, Controller, Server-Calls etc.?

Redux

"Predictable state container"

- Ein einziger Zustands-Container für die gesamte Anwendung
- Verhalten bei Zustandsänderungen sind vorhersehbar
- 3 Prinzipien
  - "Single source of truth"
  - "State is read-only"
  - "Changes are made with pure functions"

Quelle: http://redux.js.org/docs/introduction/ThreePrinciples.html

Redux - 3 Grundbausteine

1.) Actions

- Eine **Action** ist ein JavaScript-Objekt mit:
  - Type
  - Payload (Optional)

- **Beispiel:**

```
{
    type: 'ADD_TODO',
    id: 42,
    text: 'Nächste Folie zeigen',
}
```


Redux - 3 Grundbausteine

2.) State

- Der **Zustand** ist ein JavaScript-Objekt
  - Er lässt sich hierarchisch aufbauen
  - Der ***Store*** hält den Zustand und stellt Methoden zur Verfügung

- **Beispiel-State**:

  Todos

  Todo1, Todo2, …

  VisibilityFilter

Redux - 3 Grundbausteine

3.) Reducer

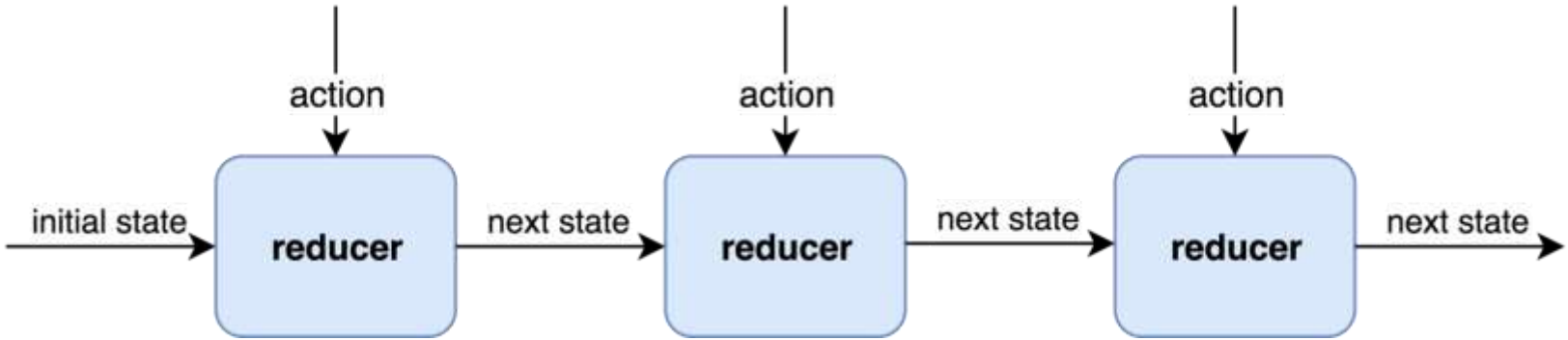
- Eine Action beschreibt, *was* sich verändern soll.
- Ein **Reducer** beschreibt, ***wie*** sich der Zustand verändern soll.
- Ein Reducer verhält sich **deterministisch**

- Root-Reducer besteht aus Teil-Reducer, z.B.
  - todosReducer für Todos state
  - visibilityFilterReducer für visibilityFilter state

Redux - 3 Grundbausteine

3.) Reducer

- **Input**:Current state, Action
- **Output**: Next state



Redux - Middleware

Jede Action durchläuft jede Middleware

- Erweiterungspunkt für vielfältige Anwendungsgebiete (z.B. Logging)
- Notwendig für Seiteneffekte (z.B. asynchrone Anfragen)

Redux - Action Creator

Funktion zum Erzeugen von Actions

- Details einer Action (innere Struktur) werden gekapselt

```
export function addTodo(text) {
        return {
                type: 'ADD_TODO',
                text,
        };
}
```

Quelle: http://redux.js.org/docs/api/bindActionCreators.html

Empfehlung zu Redux - ducks pattern

Konvention zur Strukturierung von Redux-Modulen in einer Datei

- **default export** ist **Reducer**
- **Action Creator** sind **named exports**
- Action types haben die Form “**npm-module-or-app/reducer/ACTION_TYPE**”
- Action types dürfen exportiert werden

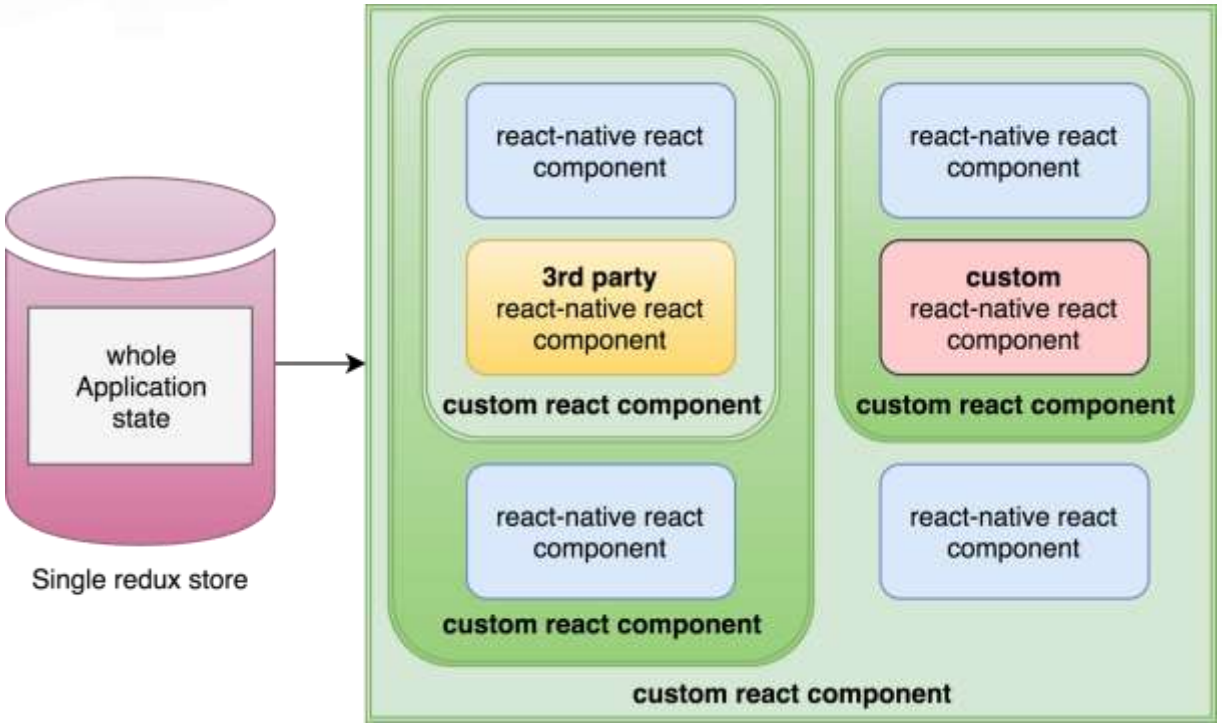
- Empfehlung: **Selectors** (Funktion für Zugriff auf Daten) sind **named exports**

Quelle: https://github.com/erikras/ducks-modular-redux

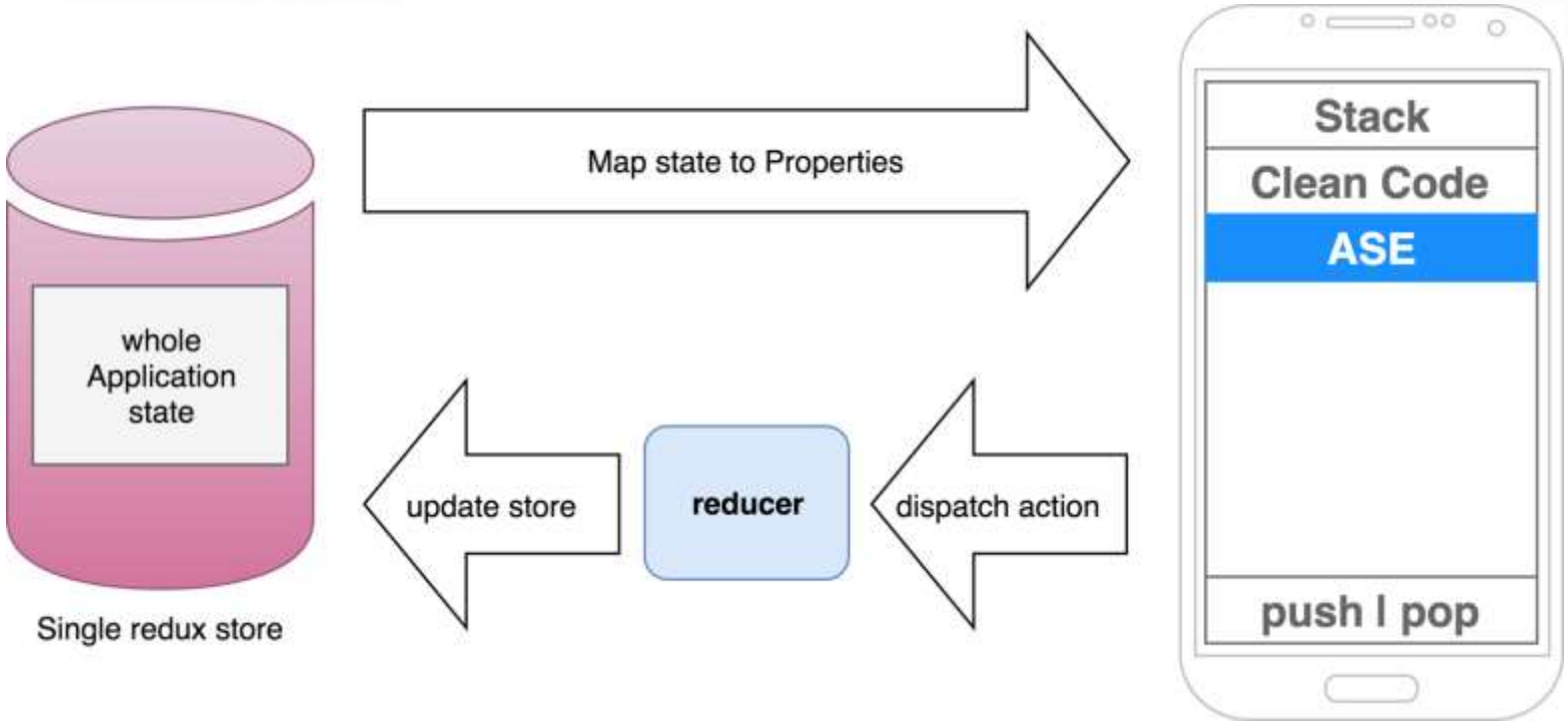
Weitere Empfehlungen zu Redux

- Einführung (“must see”)
  - Dan Abramov: “Getting Started with Redux” https://egghead.io/courses/getting-started-with-redux
- Middlewares:
  - Asynchronität (redux-thunk, redux-saga, redux-observables)
  - redux-logger
  - redux-persist
- Trennung in Presentational und Container Components

React-Native + React + Redux



React-Native + React + Redux



andrena
OBJECTS

Qualitätssicherung

Code-Standards, Automatisierte Unittests und Continuous Integration

Code-Standards

- Einheitliche Formatierung und Design Pattern
  - IDE Unterstützung
  - ESLint

Jest - Automatisiertes Testen

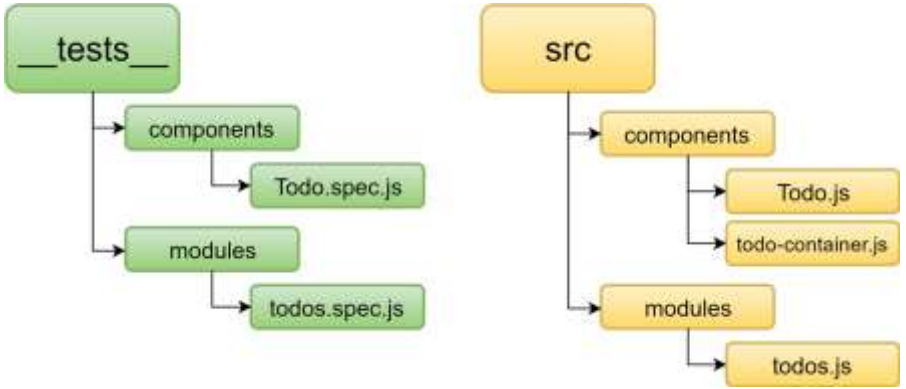
- Auf React angepasstes Testing-Framework
- Basierte ursprünglich auf Jasmine

- Wir verwenden Jest für **Unittests**
  - Schnelle Testausführung (wenige Sekunden)
  - Einfache Anzeige der Coverage und der uncovered Lines
  - Snapshot-Tests
- Tests orientieren sich an http://redux.js.org/

Jest - Automatisiertes Testen

Testaufbau

- Aufbau des Testcodes parallel zum Produktivcode möglich
- Anordnung in Testsuites
- *Beispiel:*




```
describe('A Component', () => {
 it('behaves as expected', () =>{
    // test content
    expect(true).toBe(true);
 });
});
```

andrena OBJECTS

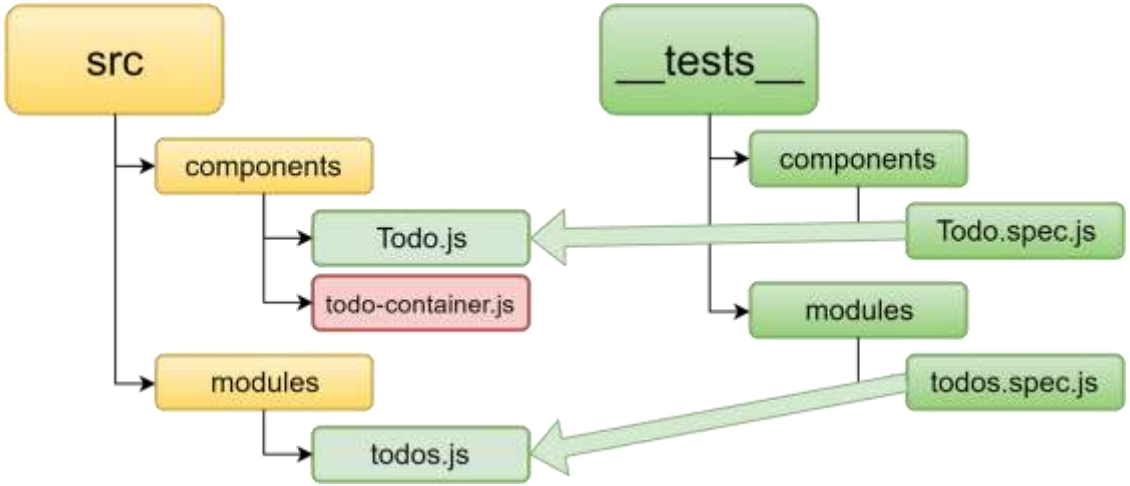
Jest - Automatisiertes Testen

Was testen wir mit Unittests?

- Komponenten (React, React-Native)
- Module (Redux)

… und was nicht?

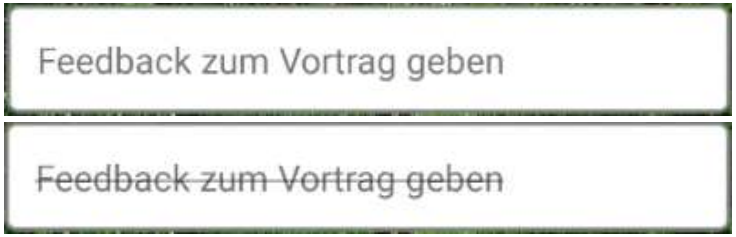
- Container
- App-Setup



Jest - Automatisiertes Testen

Komponenten (React + React-Native)

- Darstellung
- Funktionalität
  - *Beispiel:* Todo




```
<TouchableHighlight onPress={props.onPress}>
  <Text style={
    {textDecorationLine: props.completed ? 'line-through' : 'none'}
  }>
    {props.children}
  </Text>
</TouchableHighlight>
```

Jest - Automatisiertes Testen

Komponenten - Darstellung

- Snapshottests
  - Serialisierung der Komponente / Vergleich mit Serialisierung
  - Snapshottest sind auf alle serialisierbaren Objekte anwendbar

```
it('renders an uncompleted Todo correctly in a snapshot', () => {
  const todoJson = renderer.create(
    <Todo completed={false} onPress={doNothing}>
      my uncompleted Task
    </Todo>
  ).toJSON();
    expect(todoJson).toMatchSnapshot();
});
```

Jest - Automatisiertes Testen

Was ist ein Snapshot?

```
exports[`Todo renders an uncompleted task correctly ...
<View ...
<Text
 style={...
     Object {
       "textDecorationLine": "none",
     },
>
 my uncompleted Task
</Text>
```

Jest - Automatisiertes Testen

Komponenten - Funktionalität

- Enzyme
  - Simulation von Verhalten

```
it('triggers the onPress action by pressing the component', () => {
  const mockedOnPress = jest.fn();
  const toDoWrapper = shallow(
    <Todo completed onPress={mockedOnPress}>my completed Task</Todo>
  );
  expect(mockedOnPress).not.toHaveBeenCalled();
  toDoWrapper.simulate('press');
  expect(mockedOnPress).toHaveBeenCalledTimes(1);
});
```

Jest - Automatisiertes Testen

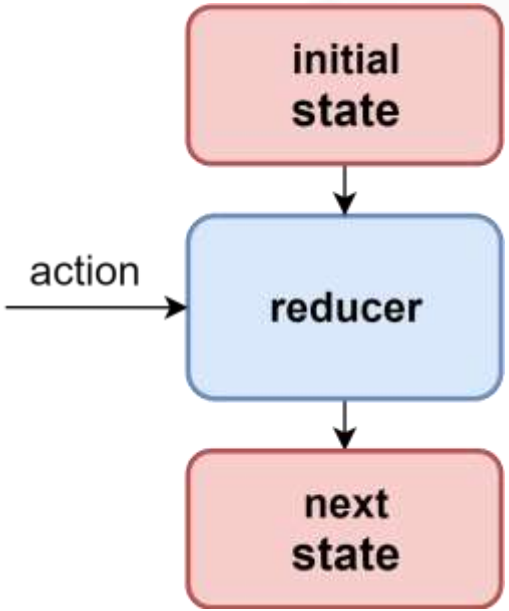
Reducer, Actions und Action Creator (Redux)

- Reducer und Actions Creator können separat getestet werden
  - Erzeugen Action Creater die erwartete Action
  - Gibt der Reducer bei einer Action den korrekten next state zurück
  - Beispiele auf redux.js.org
- Wir testen Reducer direkt mit Action Creator
  - Tests werden weniger fragil
  - *Beispiel:* Änderung/Umbenennung des Action-Type

Jest - Automatisiertes Testen

Reducer und Action Creator

- Welcher Zustand wird zurückgegeben
- *Beispiel:* Änderung der todos bei addTodo(`'message'`)

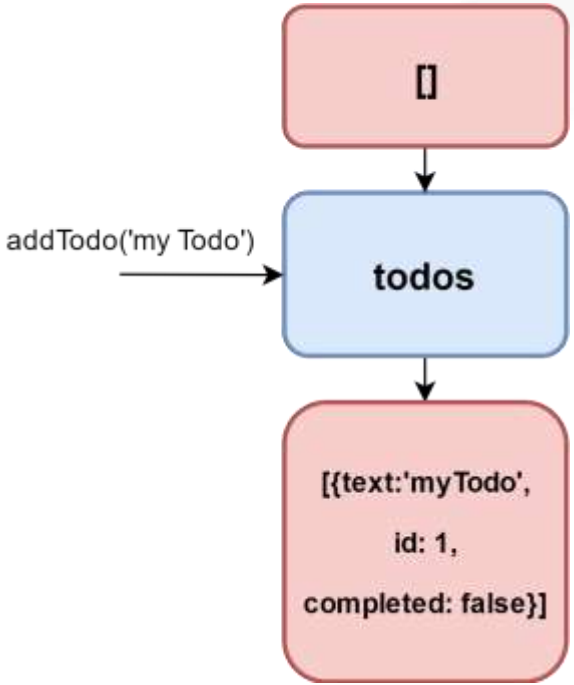



```
const todos = (state = {}, action) => {
    switch (action.type) {
        case ADD_TODO:
            return [
                ...state,
                todo({}, action),
            ];
...
```

Jest - Automatisiertes Testen

Reducer und Action Creator

- Erwartetes Verhalten bei addTodo()




```
it('adds correctly a Todo', () => {
    const initialState = [];
    const action = addTodo('my first Todo');
    const expectedState = [{
        text: 'my first Todo',
        id: 1,
        completed: false,
    }];
    expect(toDosReducer(initialState, action)).toEqual(expectedState);
});
```

Jest - Automatisiertes Testen

Async Action Creator (Thunk, Sagas etc.)

- Eine Action dispatched weitere Actions an

```
export function addDoneTodo() {
    return (dispatch) => {
        dispatch(addTodo('Something I already did'));
        dispatch(toggleTodo(lastTodoId));
    };
}
```

Jest - Automatisiertes Testen

Async Action Creator (Thunk, Sagas etc.)

- Gemockter Redux-Store
  - Zugriff und Überprüfung der Actions

```
it('dispatches actions correctly for addDoneTodo', () => {
        const store = mockStore();
        const expectedActions = [
                addTodo('Something I already did'),
                toggleTodo(lastTodoId)
        ];
    store.dispatch(addDoneTodo());
    expect(store.getActions()).toEqual(expectedActions);
});
```

Continuous Integration

Anforderungen an CI

- Bei jeder eingecheckten Änderung werden automatisch
  - die Tests ausgeführt,
  - die Metriken zur Codequalität ermittelt,
  - die Android und die iOS App gebaut und
  - die Installationsdateien zur Verfügung stellen.

Continuous Integration

Umsetzung mit Jenkins (Multibranch-Pipeline)

- Jenkinsfile (Buildconfiguration) ist eingecheckt
  - Buildprozess ist in Abschnitte (stages) unterteilt
  - Abschnitte können speziellen nodes zugeteilt werden
  - z.B. “Build iOS” auf macOS node
  - Zugriff auf Umgebungsvariablen (Branchname, Buildnummer etc.)
  - ermöglicht u.A. branchspezifisches Verhalten

Continuous Integration

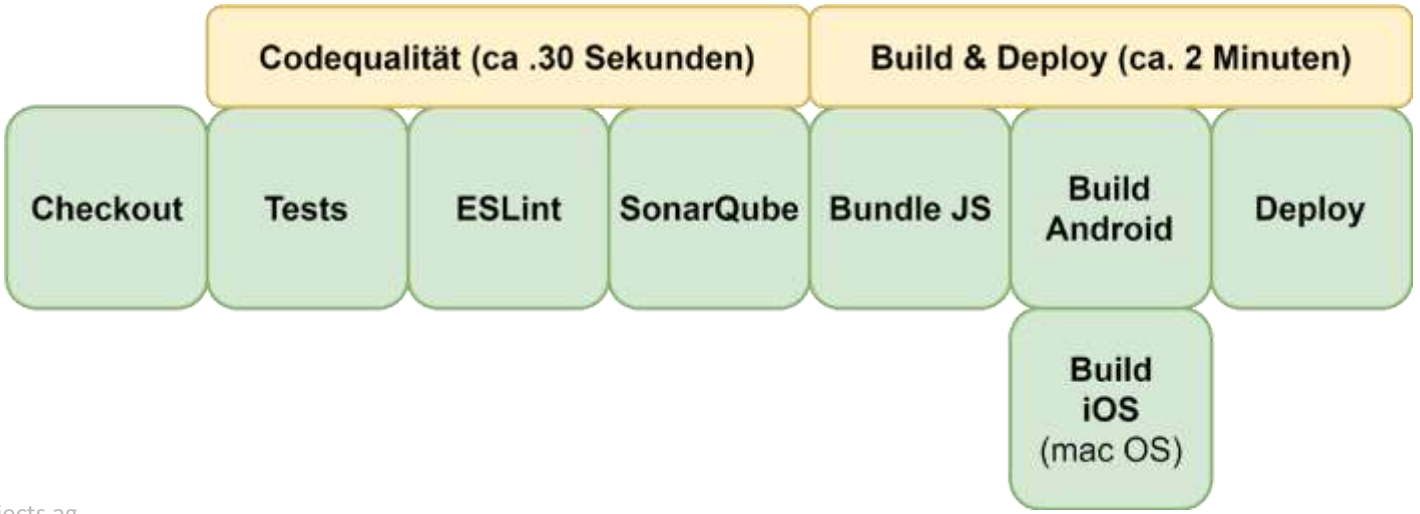
Umsetzung mit Jenkins (Multibranch-Pipeline)

- Umsetzung möglichst mit kleinen npm-/yarn-Skripten
  - yarn install
  - yarn test
  - yarn lint
- Weitere Build-Tools wenn nötig
  - React-Native CLI
  - Gradle (für Android)
  - XCode-Buildtools (für iOS)

Continuous Integration

Umsetzung

- Jenkins (Multibranch-Pipeline)



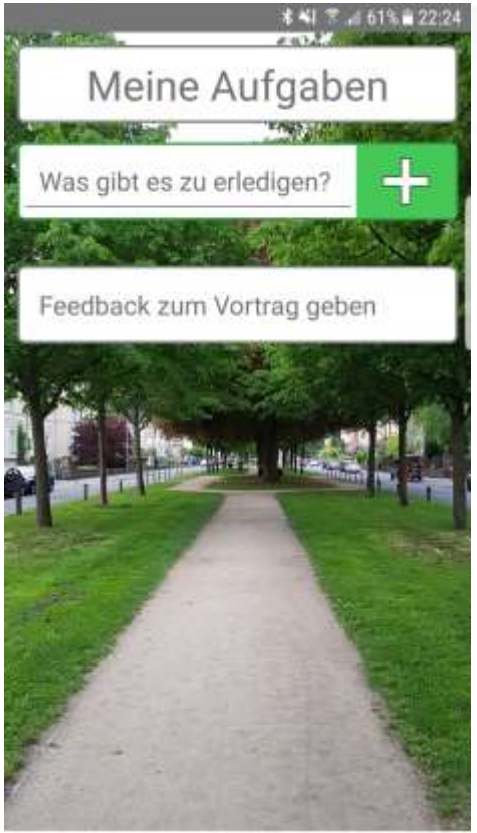
Demo

Todo-List App

Demo

Einblick in die App-Entwicklung

- *Beispiel:* Todo-List App (http://redux.js.org)
- Test-Driven-Development
  - Testausführung in wenigen Sekunden
  - Jest als Testing-Framework
- Hot-Reloading
- Debugging
  - Redux (Trennung von Logik & Anzeige)

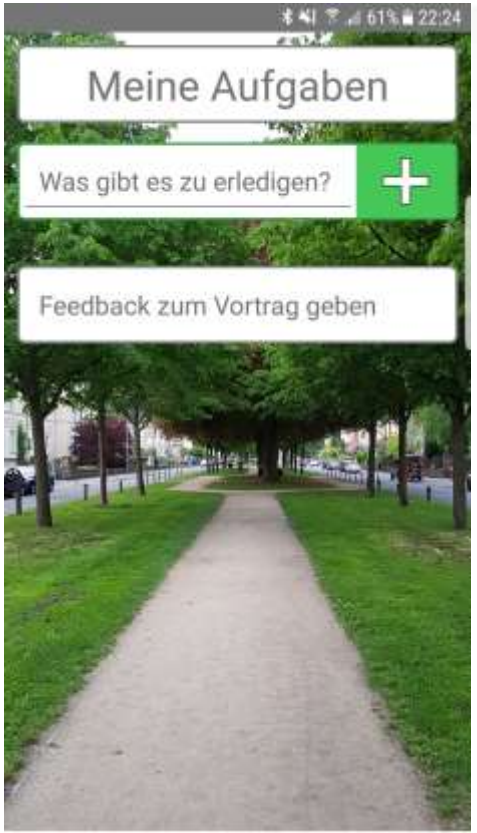


Demo

Einblick in die App-Entwicklung

- Wie (schnell) werden Tests ausgeführt?
- Wie sehe ich die Testabdeckung?
- Wie funktionieren Snapshottests?

- Wie schnell sehe ich Änderungen?
- Wie debugge ich die App?



Zusammenfassung

Zusammenfassung

Was spricht gegen React-Native + Redux

- Es geht nicht immer ohne Android-/iOS-Know-How
- Einstiegshürde hoch für nicht-JavaScript-Entwickler
- Redux zu verstehen und richtig anwenden zu können kostet Zeit
- Bleeding Edge
  - Regelmäßig “Breaking Changes”
  - Navigationskonzepte immer noch nicht ausgereift

Zusammenfassung

Was spricht für React-Native + Redux

- Entwicklung nativer Apps in JavaScript ist möglich
- TDD und schnelle Testausführung
- Hohe Testbarkeit
- Robustheit
- CI/CD schnell und einfach
- Aktive Community

Codebeispiele

- ToDos-App
  - https://github.com/raphKn/ToDos
- React-Native mit Redux Starter
  - https://github.com/rroehrig/ReactNativeReduxSeed

Bilderquellen

[1] React Logo, facebook.github.io/react, ergänzt um “React-Native”-Untertitel
https://github.com/facebook/react/blob/master/LICENSE-docs

[2] React Logo, facebook.github.io/react
https://github.com/facebook/react/blob/master/LICENSE-docs