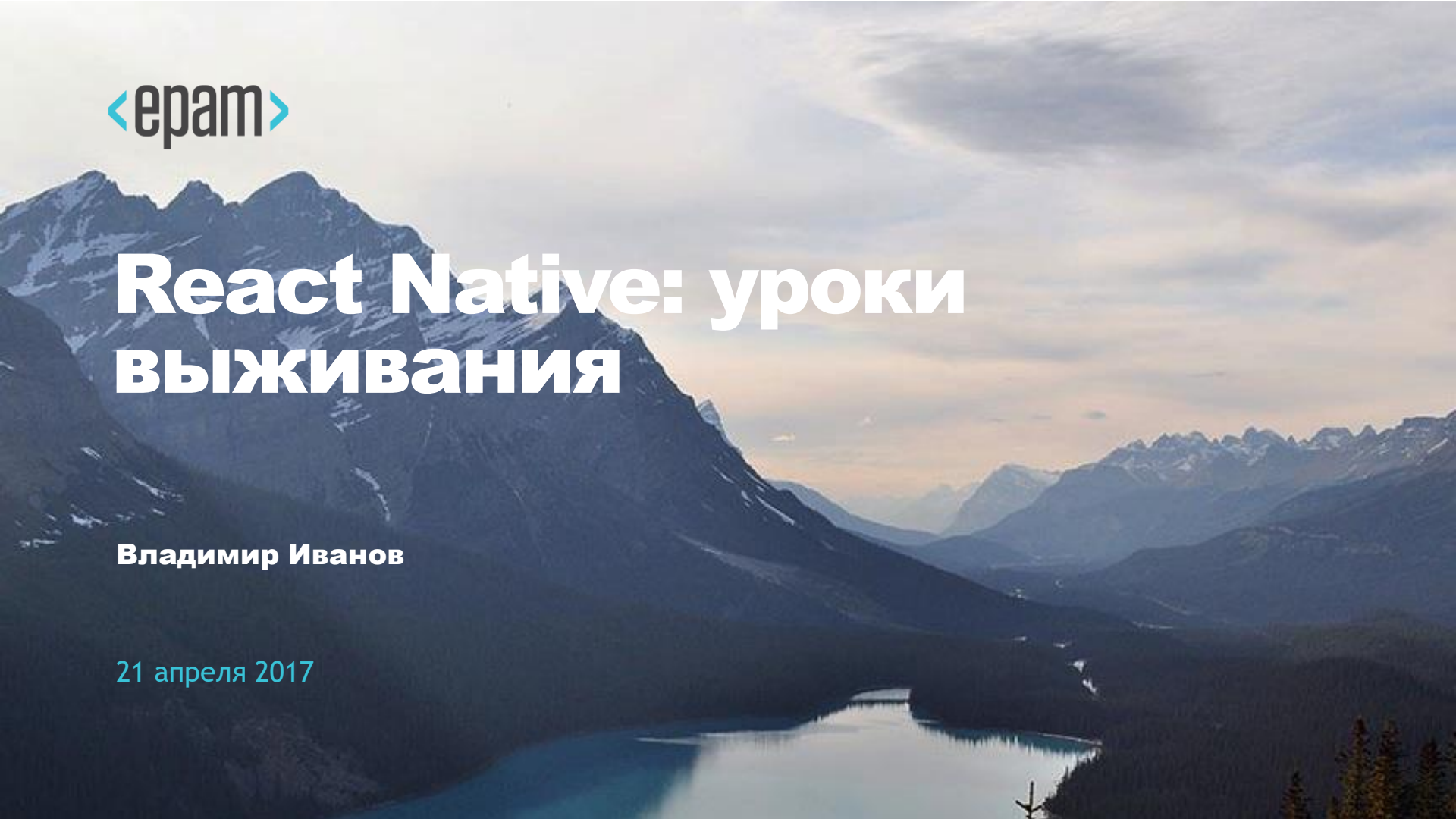


React Native: уроки выживания

Владимир Иванов

21 апреля 2017

Слайд с улыбкой

Владимир Иванов

- Ведущий разработчик
- Более 6 лет в Android разработке
- Широкий интерес в Mobile
- Отец изумительного сына

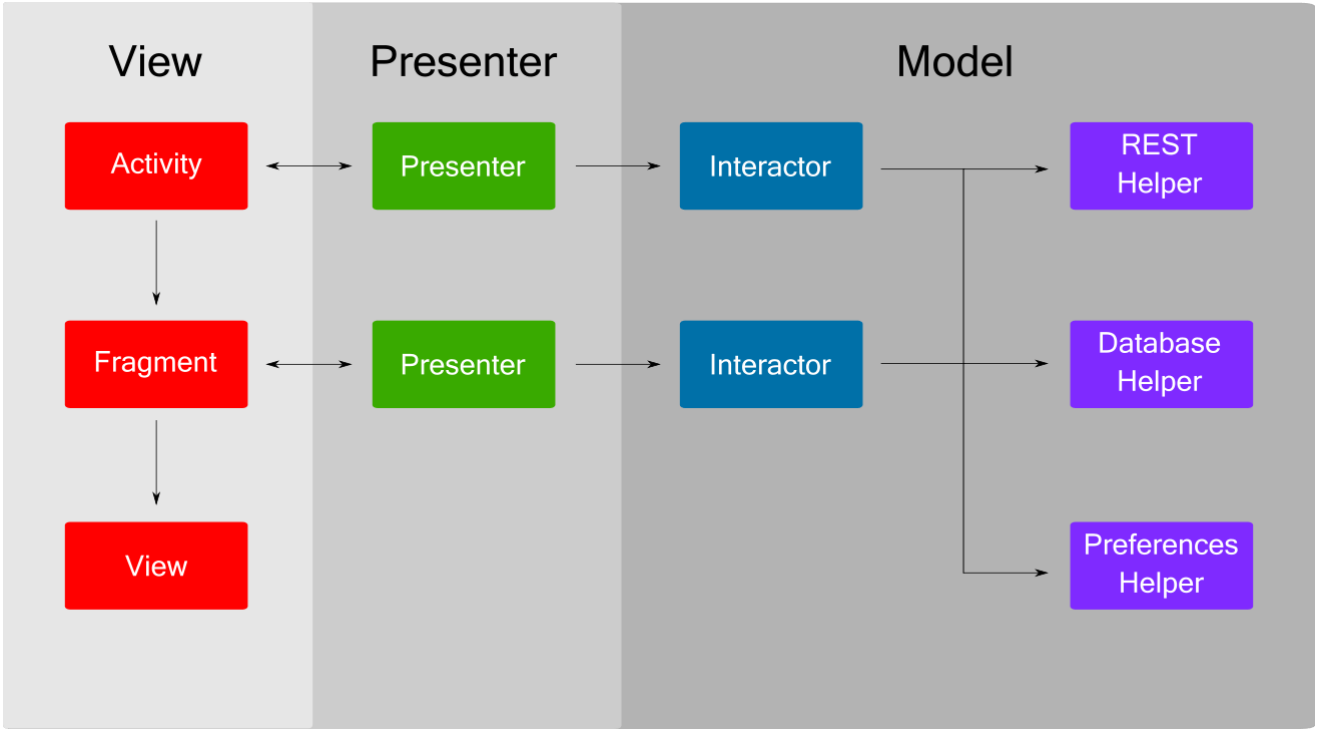
QA

- У кого есть коммерческие проекты на RN?
- У кого есть проекты в сторе?
- У кого есть хоть какой-нибудь проект?
- **Что самое трудное?**

Что самое трудное?

- Локальные персистентные данные
- Локализация
- Сложный UI
- Много бизнес-логики

Подход нативной разработки

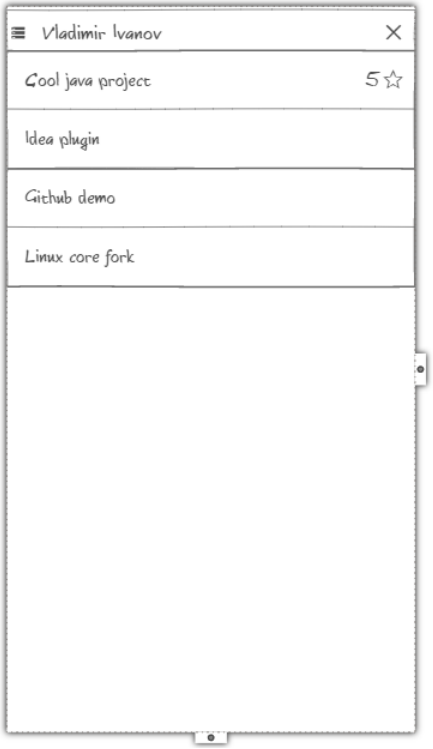


Сложный UI

1. React Native – это View!
2. Вынесение бизнес-логики
3. Управление состоянием

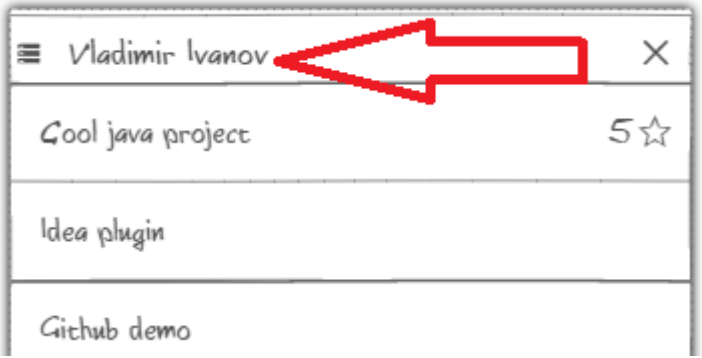
Простой клиент для гитхаба

- Логин
- Список репозиториев со звездочками



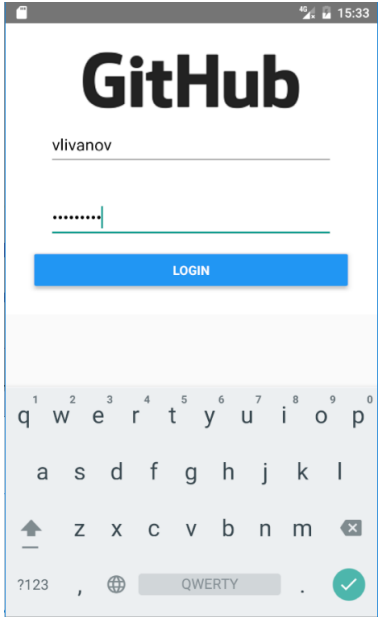
Как отображать пользователя?

- Задача: Имя пользователя может быть нужно на любом экране



Наивная имплементация

- Передавать пользовательсся через props




```
<View>
    {this.state.user && <Repositories user={this.state.user} auth={this.state.auth} />}
    {!this.state.user && <LoginScreen
        onLogin={(user, auth) => {
            console.log(`[App] user = ${JSON.stringify(user)}`);
            this.setState({ user, auth });
        }}/>}
</View>);
```

```
.then(result => {
    console.log('fetch: ' + JSON.stringify(result));
    if (result.status === 200) {
        console.log('fetch: success, name = ' + result.name)
        this.props.onLogin(JSON.parse(result._bodyInit), base64);
    } else {
        this.setState({ error: `Failed to login with ${result.status}` })
    }
})
```

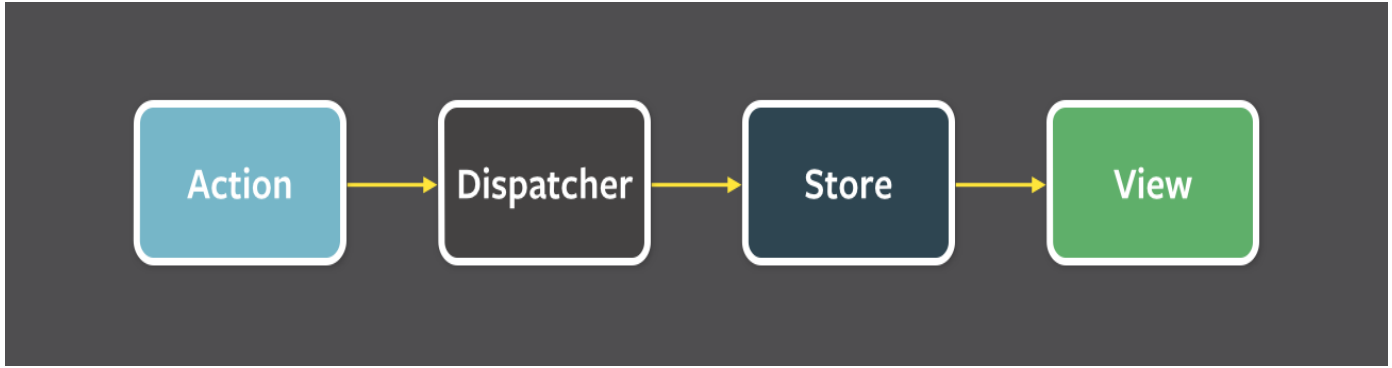
Проблемы наивной имплементации

- Для любой вложенности > 1 передача коллбэков – ад
- Бизнес-логика логина находится внутри компонента(View)
- Данные нужно передавать между компонентами, что возможно далеко не всегда

ВНЕШНЕЕ СОСТОЯНИЕ: FLUX&REDUX

FLUX подход – однонаправленный поток данных

- Dispatcher
- Store
- View



Redux – реализация Flux

- > 12 имплементаций Flux
  - (Flummox, Delorean, Reflux)
- Redux – лучший, потому что
  - Хорошо документирован
  - Лучше всего реализован
  - Хорошо масштабируется
  - Большое коммьюнити

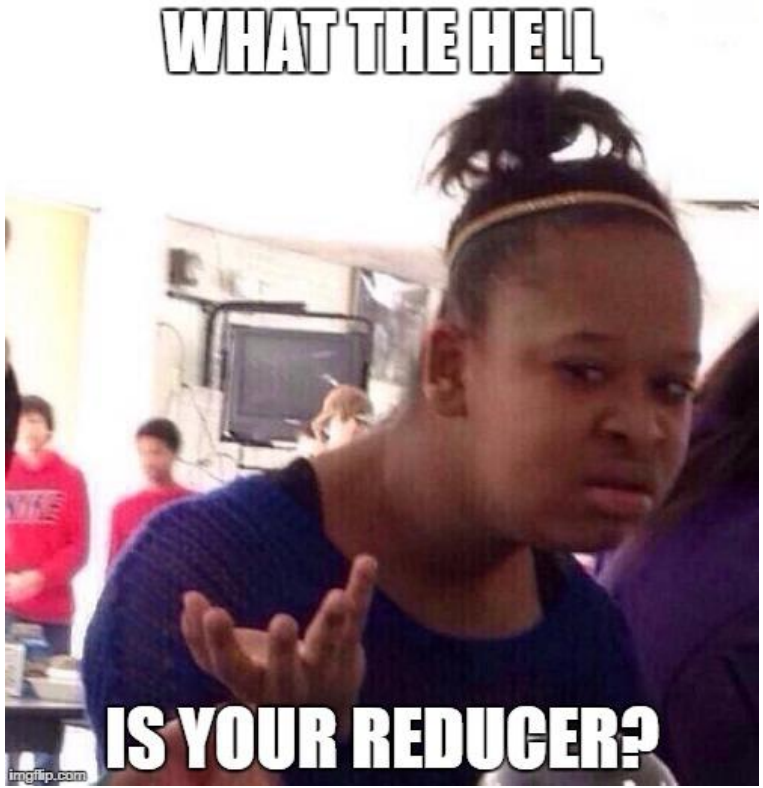
Проблема

- Состояние приложения – это много данных
- Когда данных много, мы теряем над ними контроль
- Изменяемость и асинхронность: Кола с Ментосом

Философия REDUX

- Источник истины один
- Стейт только чтения
- Изменения происходят с помощью чистых функций(редьюсеров)

Философия REDUX



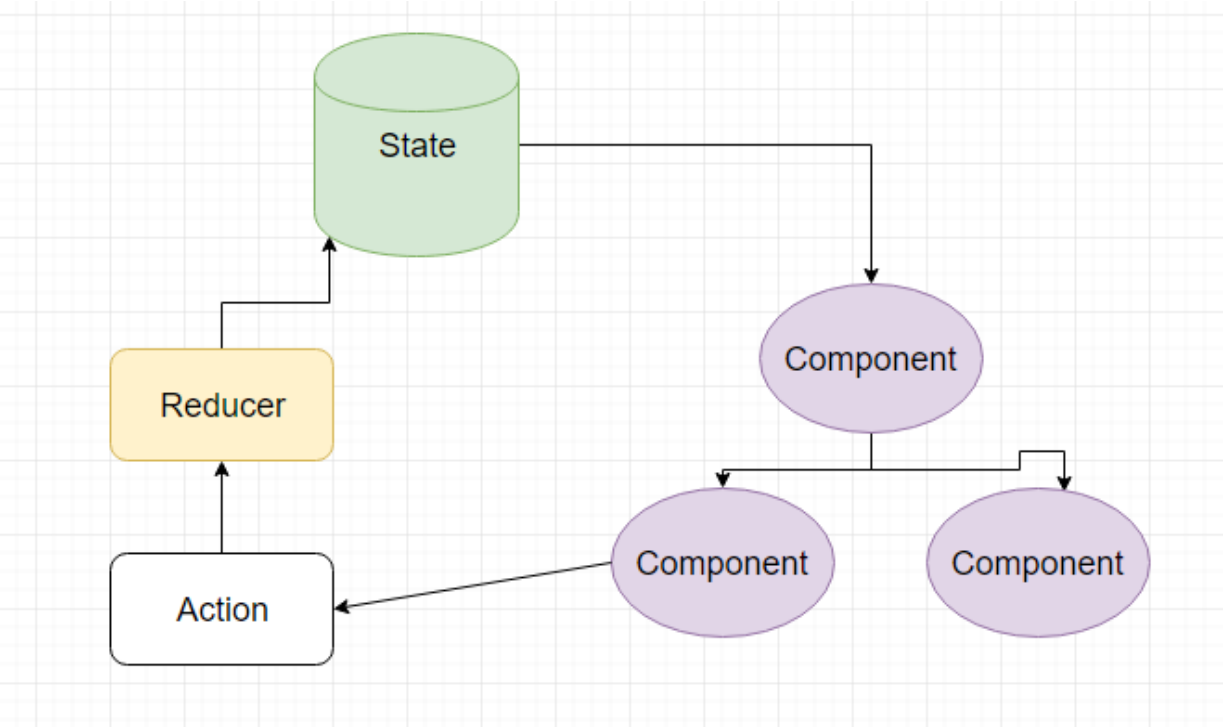
Редьюсер – это слово из bullshit bingo!

architecture	action	danabramov	react	reducer
ref	state	connect	actioncreator	state
unidirection	props	CUSTOM BINGO (free square)	dataflow	store
reactnative	jest	flux	context	immutable
component	fatarrowfunction	mapStateToProps	redux	thunk

https://goo.gl/ZRiXbU

Инструменты REDUX

- Action
- Store
- Action creator
- Reducer



Инструменты REDUX: Action

- Action – POJO с полем type


```
{
   "type":"LOGIN_FORM_SUBMITTED",
   "payload":{
      "login":"github",
      "password":"fancypassword"
   }
}
```

Инструменты REDUX: Store

- Store – хранилище для State
- Store работает с редьюсерами

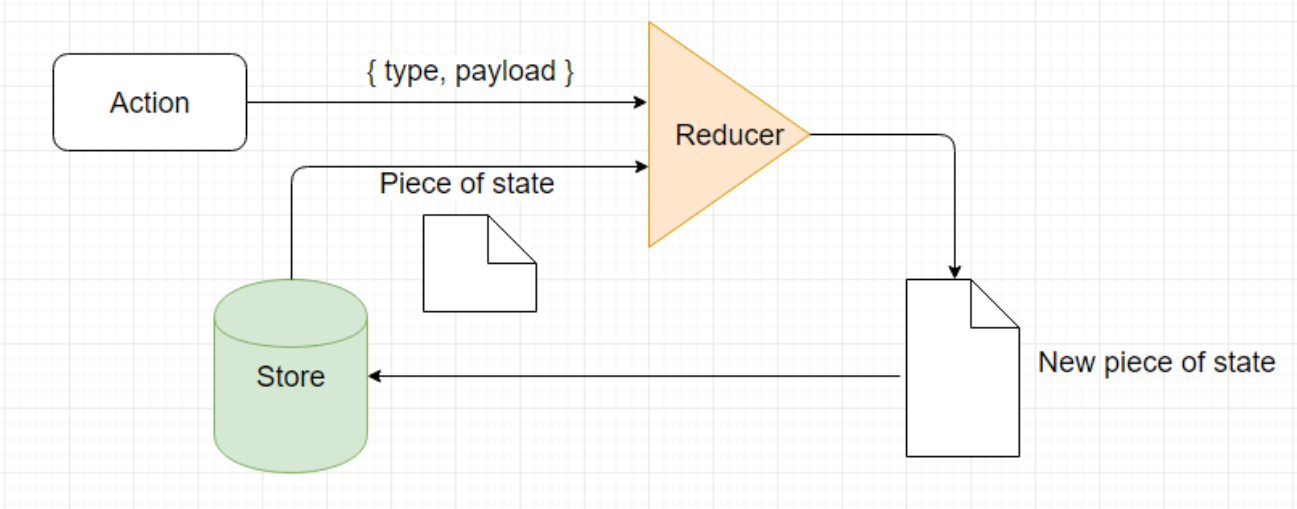
App.js

```
1  import React, {Component} from 'react';
2  import {createStore} from 'redux';
3
4  import Provider from "react-redux/src/components/Provider";
5
6  import reducers from './src/reducers';
7  import Root from "./src/Root";
8
9  export default class App extends Component {
10
11     constructor(props) {
12         super(props);
13         this.state = {user: undefined, auth: undefined}
14     }
15
16     render() {
17         return (
18             <Provider store={createStore(reducers)}>
19                 <Root />
20             </Provider>);
21     }
22 };
23
```

Инструменты REDUX: Reducer

- Reducer: чистая функция, которая возвращает новое состояние после применения Action.

- В Store может быть(и должно быть!) много редьюсеров.



Инструменты REDUX: Reducer

2 параметра: текущее состояние и действие

1 возвращаемое значение: новое состояние

LoginReducer.js

```
import {
    LOGIN_START,
    LOGIN_SUCCESS,
    LOGIN_ERROR,
} from '../actions/ActionTypes';

const INITIAL_STATE = {
    user: null,
    auth: null,
    error: null,
    loading: false
};

export default (state = INITIAL_STATE, action) => {
    switch (action.type) {
        case LOGIN_START:
            return { ...state, loading: true };
        case LOGIN_ERROR:
            return { loading: false, error: action.payload };
        case LOGIN_SUCCESS:
            return {
                loading: false,
                user: action.payload.user,
                auth: action.payload.auth,
                error: null
            };
        default:
            return INITIAL_STATE;
    }
}
```

Инструменты REDUX: Reducer

Комбинирование редьюсеров (reducers/index.js) :

LoginReducer.js | index.js

```
1  import { combineReducers } from 'redux';
2  import LoginReducer from './LoginReducer';
3
4  export default combineReducers({
5      login: LoginReducer
6  });
```

Инструменты REDUX: Reducer

Доступ до состояния редьюсера(LoginScreen.js) :

```
const mapStateToProps = (state) => {
    return { error: state.login.error };
};

export default connect(mapStateToProps, { loginStart, loginEnd, loginError })(LoginScreen);
```

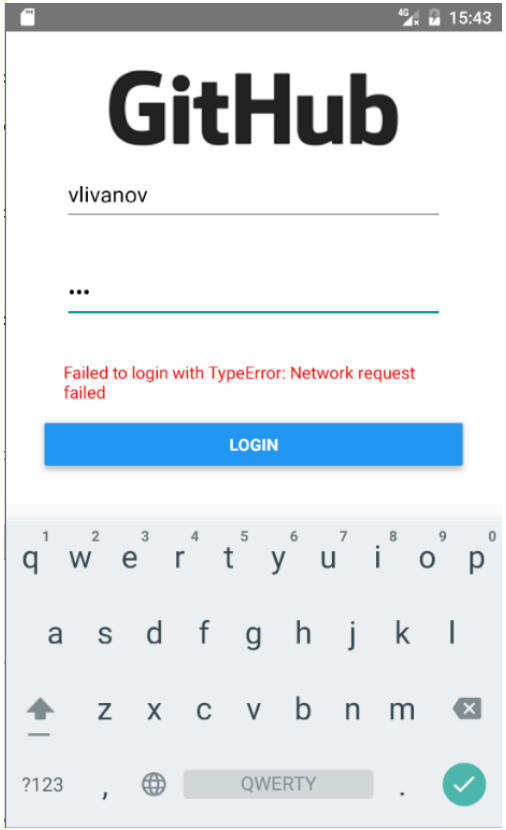
Инструменты REDUX: ActionCreator

Фабрики экшнов - это функции, которые возвращают экшны:

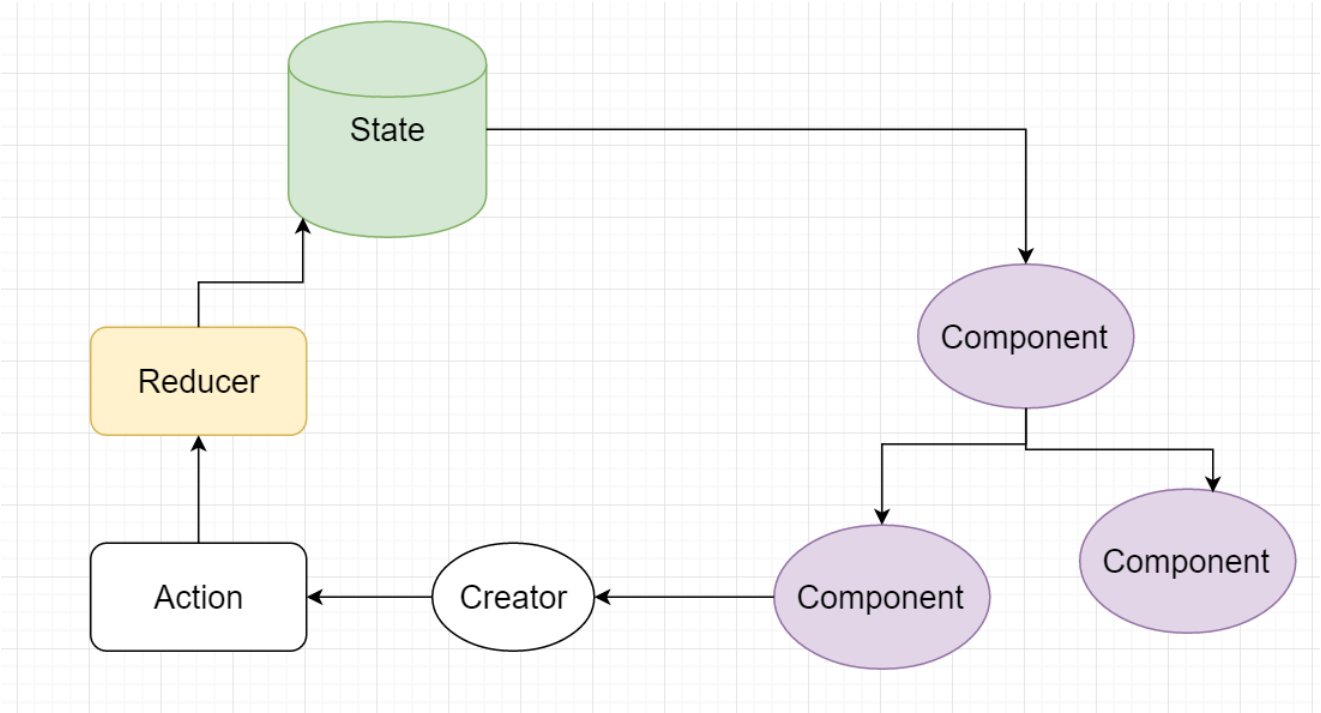
LoginActions.js

```
import {
    LOGIN_START,
    LOGIN_SUCCESS,
    LOGIN_ERROR,
} from './ActionTypes';

export const loginStart = () => {
    return { type: LOGIN_START };
};

export const loginEnd = ({ user, auth }) => {
    return { type: LOGIN_SUCCESS, payload: { user, auth } };
};

export const loginError = (error) => {
    return { type: LOGIN_ERROR, payload: error };
};

```

Демо

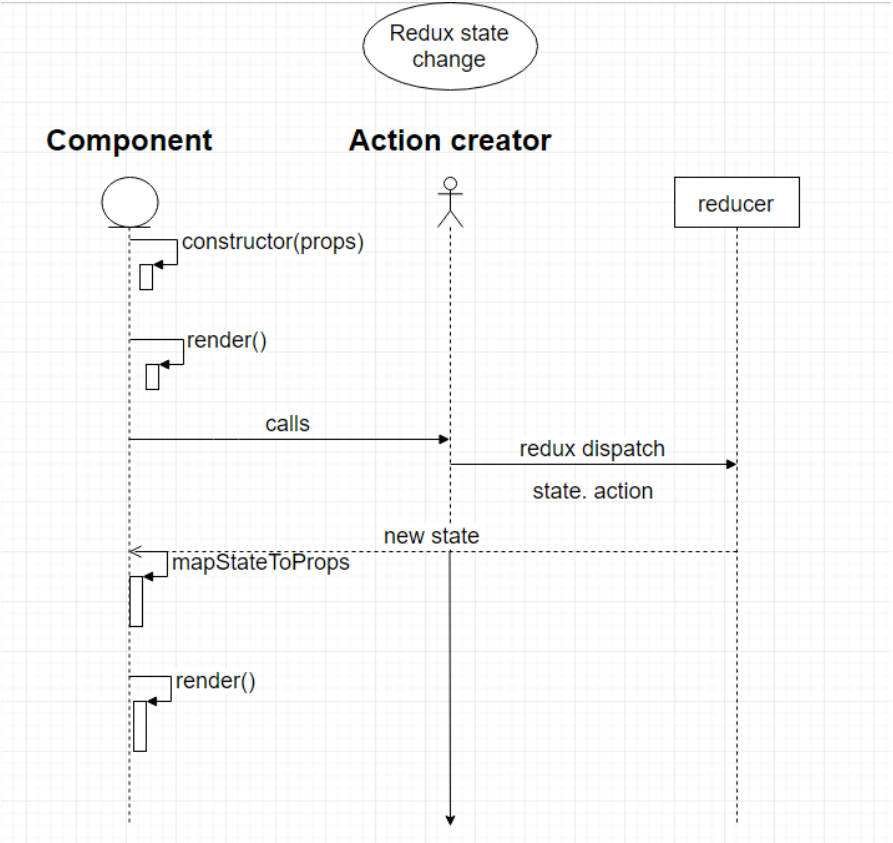

15:43
GitHub
vlivanov
...
Failed to login with TypeError: Network request failed
LOGIN
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123
QWERTY

Инструменты REDUX: полный цикл



Инструменты REDUX: полный цикл

1. constructor(props)
2. render()
3. view tap
4. action creator
5. reducer()
6. mapStateToProps(newState)
7. componentWillReceiveProps
8. render()



Проблема

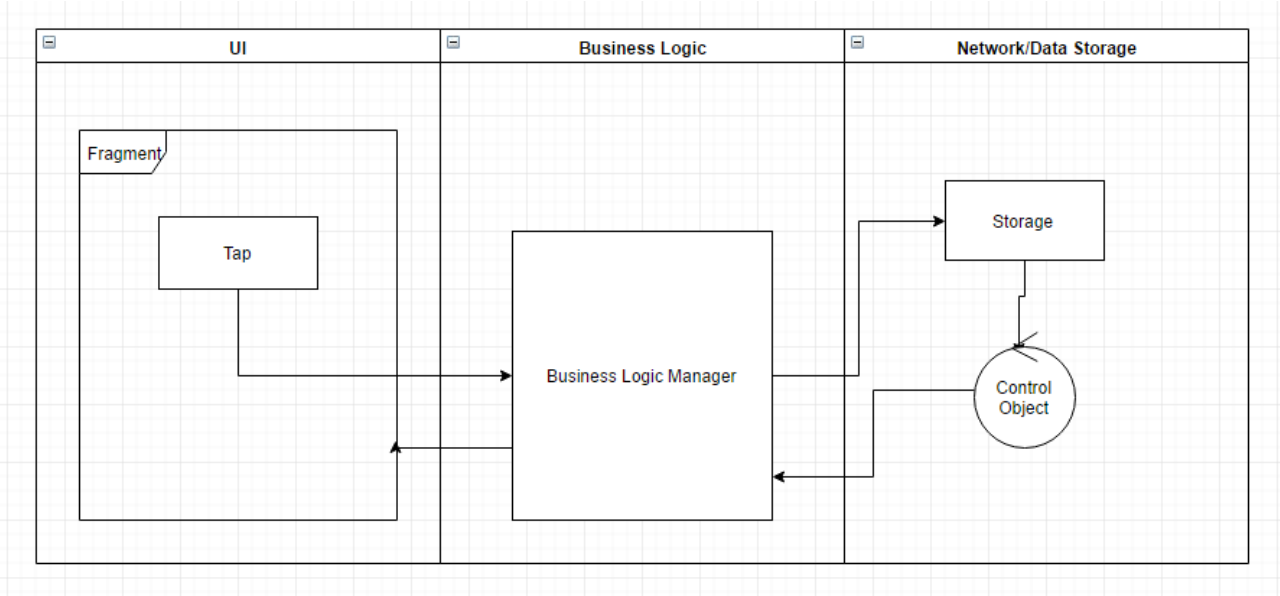
Бизнес-логика по прежнему находится в компоненте

Проблема

АСИНХРОННЫЕ ЭКШНЫ

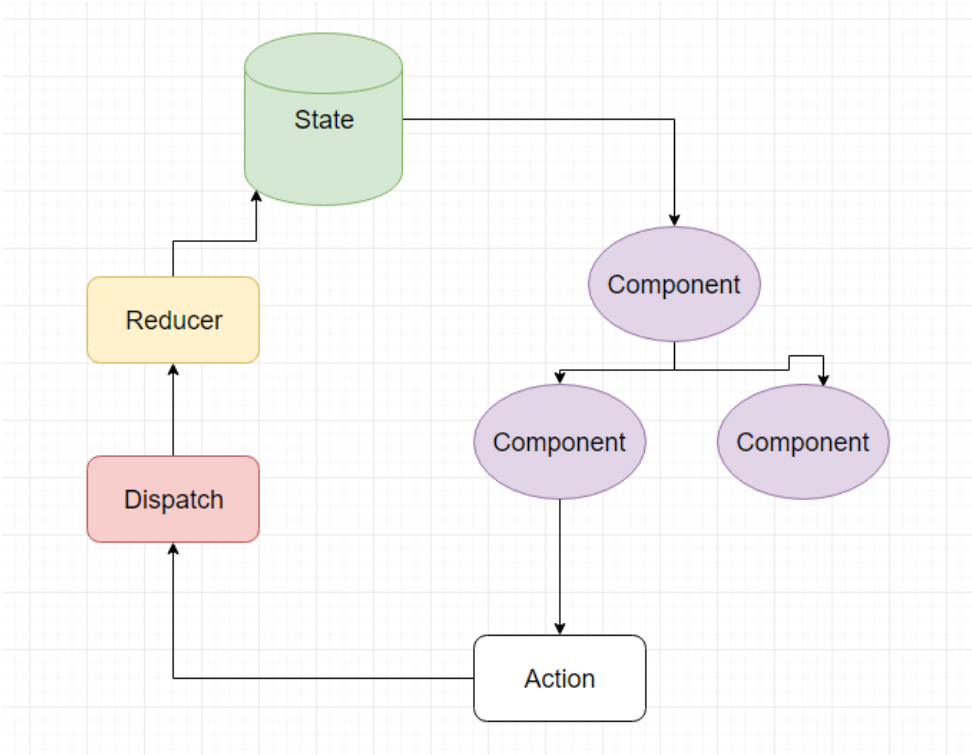
Асинхронные экшны

- Сеть или хранилище данных асинхронные



Асинхронные экшны

- Redux actions – синхронные
- Dispatching



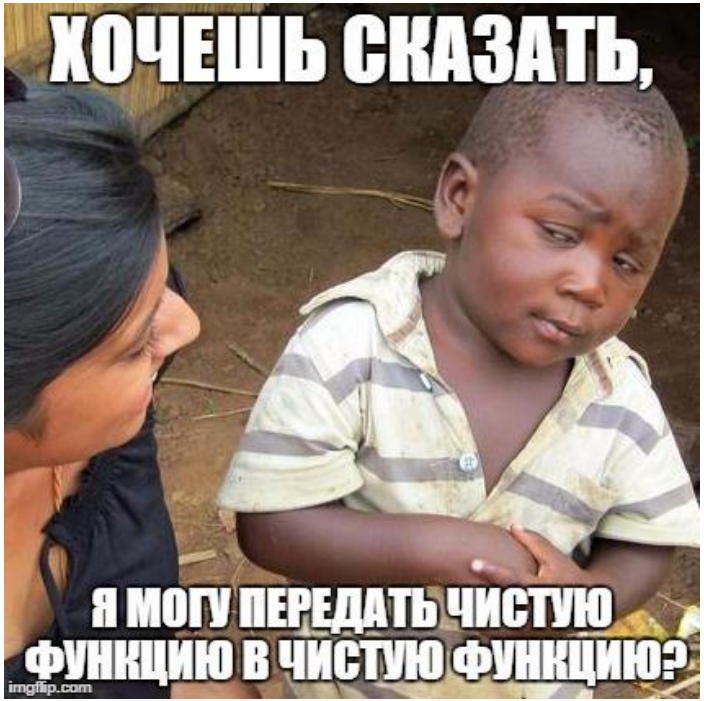
Асинхронные экшны

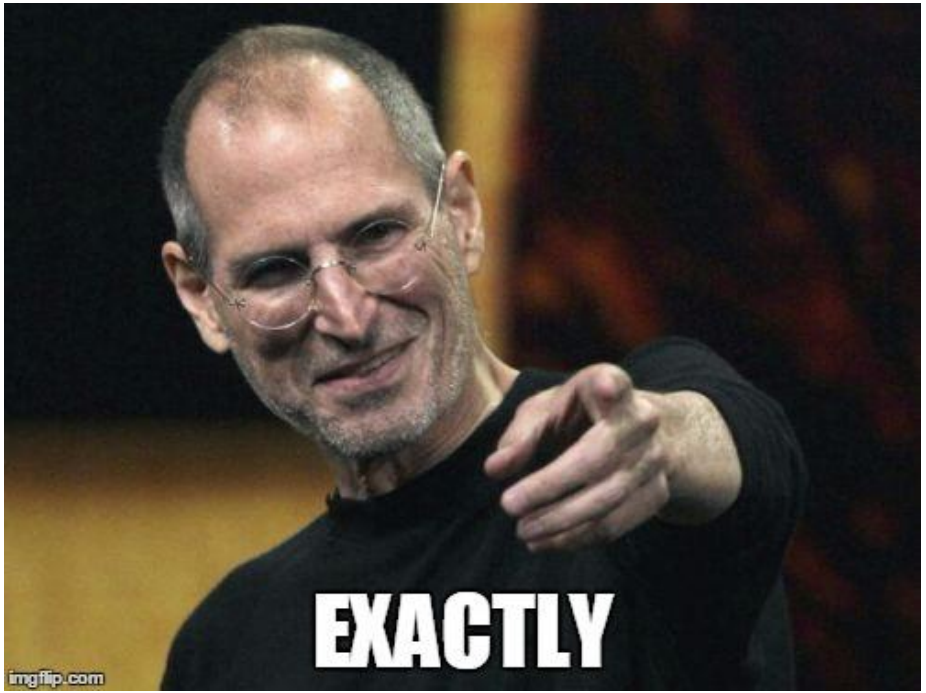
- не action, а function(dispatch).

```
6
7   export const loginStart = () => {
8       return { type: LOGIN_START };
9   };
```

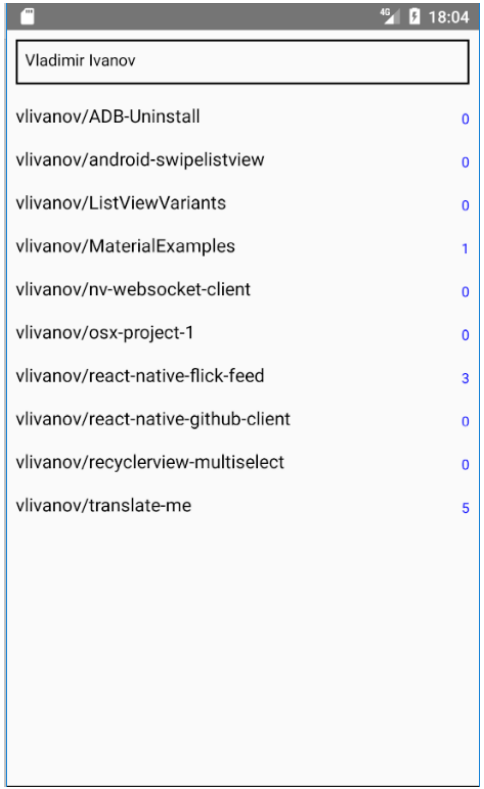
```
9   export const loginStart = (login, password) => {
10      return (dispatch) => {
11          dispatch({ type: LOGIN_START });
12          let base64 = encode(`${login}:${password}`);
13          fetch('https://api.github.com/user', {
14              method: 'GET',
15              headers: {"Accept": 'application/json'...},
20          })
21              .then(result => {
22                  console.log('fetch: ' + JSON.stringify(result));
23                  if (result.status === 200) {
24                      console.log('fetch: success, name = ' + result.name);
25                      dispatch({"type": LOGIN_SUCCESS...});
32                  } else {
33                      dispatch({"type": LOGIN_ERROR...});
37                  }
38              })
39              .catch(error => {
40                  dispatch({"type": LOGIN_ERROR...});
44              })
45              .done();
46      };
47  };
48
```

Асинхронные экшны





Демо


18:04
Vladimir Ivanov
vlivanov/ADB-Uninstall 0
vlivanov/android-swipelistview 0
vlivanov/ListViewVariants 0
vlivanov/MaterialExamples 1
vlivanov/nv-websocket-client 0
vlivanov/osx-project-1 0
vlivanov/react-native-flick-feed 3
vlivanov/react-native-github-client 0
vlivanov/recyclerview-multiselect 0
vlivanov/translate-me 5

Идентфикаторы компонент

- JSX, траспейлинг: нет идентификаторов
- Как выставить фокус при открытии экрана?
- Как валидировать форму?
  - Понять, что данные невалидны
  - Показать ошибку именно у неверного поля

Идентфикаторы компонент

- Понять, что данные невалидны, легко – у нас есть стейт
- Как проставить ошибку полю?
  - Можно хранить состояние ошибки где-нибудь в Store, это неудобно переиспользовать(помним про сложный UI)
  - Можно поискать, как обращаться к полю напрямую, чтобы менять его стейт
- В случае фокуса не поможет даже Store

Идентфикаторы компонент

- Решение: использования свойства ref.

```
<View style={styles.inputUsername}>
    <TextInput
        ref="loginInput"
        autoCorrect={false}
        placeholder={strings.loginScreen_login}
        editable
        style={[styles.input, styles.greyFont]}
        onChangeText={(text) => this.setState({ username: text })}
        value={this.state ? this.state.username : ''}
    />
</View>
```

```
<View style={styles.inputUsername}>
    <TextInput
        ref={(id) => { this.loginInput = id; }}
        autoCorrect={false}
        placeholder={strings.loginScreen_login}
        editable
        style={[styles.input, styles.greyFont]}
        onChangeText={(text) => this.setState({ username: text })}
        value={this.state ? this.state.username : ''}
    />
</View>
```

Идентификаторы компонент

- Использование ref:

```
56      render() {
57          const {container, textInput} = styles;
58
59          return (
60              <View style={container}>
61                  <View style={{alignItems: 'center'}}>
62                      <Image
63                          source={require('./../../GitHub-Logo.png')}
64                      />
65                  </View>
66                  <Form ref={(id) => { this.form = id; }}>
67                      {this.renderInputs()}
68                  </Form>
69                  {this.renderError()}
70                  <Button title='Login' onPress={() => {
71                      if (this.form.validate()) {
72                          this.onLogin();
73                      }
74                  }}/>
75              </View>
76          )
77      }
78  }
```

Context

- Проблема: передача пропертей при большой вложенности компонентов(уровень вложенности > 3)
- Пример: коллбэки навигации
- Решение: передача контекста

```
render() {
    const goBack = this.props.goBack;
    return (
        <View style={styles.container}>
            <Toolbar goBack={goBack}/>
            <View style={styles.content}>
                <Header goBack={goBack}/>
                <Body goBack={goBack}/>
                <Footer goBack={goBack}/>
            </View>
        </View>
    );
}
```

Context

- childContextTypes – как PropTypes, только contextTypes

Context types

```
22
23      getChildContext() {
24          return {
25              register: this.register,
26              unregister: this.unregister,
27              components: this.components,
28              errors: this.state.errors,
29              validateState: this.validateState
30          };
```

```
5
6       static childContextTypes = {
7           register: PropTypes.func.isRequired,
8           unregister: PropTypes.func.isRequired,
9           components: PropTypes.object,
10          errors: PropTypes.object,
11          validateState: PropTypes.func.isRequired
12      };
13
```

Context – это вам не Activity!

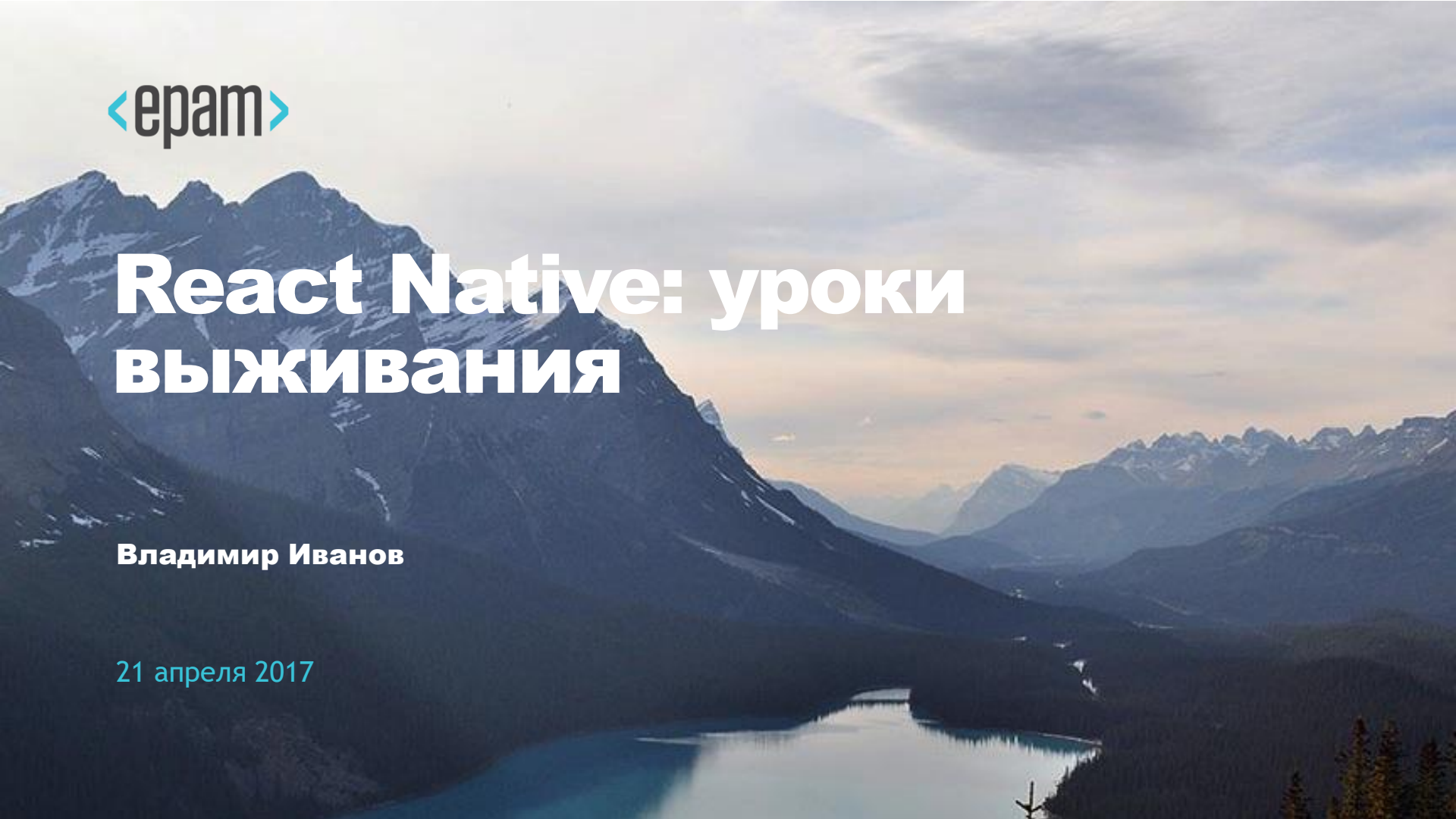
- Не надо путать React Context с Андроидом контекстом:
  - Никаких Activity, Appliction etc.
  - Контекст – это просто обьект, спущенный по иерархии
  - Контекстов может быть несколько

Контекстов может быть несколько

```
57
58      render() {
59          const {container, textInput} = styles;
60
61          return (
62              <View style={container}>
63                  <Localization lang={this.lang}>
64                      <View style={{alignItems: 'center'}}>
65                          <Image
66                              source={require('./../../GitHub-Logo.png')}
67                          />
68                      </View>
69                      <Form ref={(id) => { this.form = id; }}>
70                          {this.renderInputs()}
71                      </Form>
72                      {this.renderError()}
73                      <Button title='Login' onPress={() => {
74                          if (this.form.validate()) {
75                              this.onLogin();
76                          }
77                      }}/>
78                  </Localization>
79              </View>
80          )
81      }
82  }
```

Ссылки

- https://github.com/vlivanov/react-native-github-client/tree/master/github
- https://twitter.com/dan_abramov
- https://github.com/reactjs/redux
- https://facebook.github.io/react/docs/context.html



React Native: уроки выживания

Владимир Иванов

21 апреля 2017