inovex

React 👰 💒 TypeScript

Eine glückliche Ehe

Johann Böhler

Karlsruhe, 25.6.2018

Johann Böhler

Full-Stack Entwickler

Alles so halb, nichts so richtig

Case Studies

Komponenten-
BIBLIOTHEK

Scrumlr

React

- Komponentenbasierte Library für Web UIs
  - Framework im Baukastenprinzip
- JSX Syntax für Komponenten, angelehnt an HTML
- Mittlerweile MIT (seit September 2017)

JSX

```
<div>
    <h1 className="heading">Title</h1>
    <Article>{content}</Article>
</div>
```

JSX

```
<div>
    <h1 className="heading">Title</h1>
    <Article>{content}</Article>
</div>
```

```
<div className="article">
    {children}
</div>
```

Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.

Martin Fowler

***Refactoring: Improving the Design of Existing Code*, 1999**

Statische Typisierung

- Gesteigertes Verständnis? Vielleicht ...
  - Definitiv aber Teil der Dokumentation
- Statische Typen können Fehler reduzieren
  - ... müssen aber nicht, siehe Quelle
- IDE Unterstützung
  - Assistenz
  - Refactoring

Vergleich

TypeScript vs. Flow

- Programmiersprache
- Starke Verbreitung
- Performance
- Meist einheitliche Codebase, Migration aus JS möglich

- Nur Validierung
- Mäßige Verbreitung, mehr in React Native
- Langsam
- Optional, Migration aus JS problemlos
- Typgenauer für React

Quickstart

```
npm install -g create-react-app
create-react-app my-app --scripts-version=react-scripts-ts
```

Zu beachten

Veränderungen gegenüber JavaScript

› .jsx wird zu .tsx
› PropTypes ➡️ ♻️ ➡️ 🗑️

Zu beachten

Veränderungen gegenüber TypeScript

- **<>**-Casts werden zu **as**-Casts
  - `<MyClass> someObject`
  - `someObject as MyClass`

- https://github.com/Microsoft/TypeScript-React-Starter/issues/8
  `import * as ...`

```jsx
// src/components/Hello.jsx

import React from 'react';
import PropTypes from 'prop-types';

const Hello = ({name, enthusiasmLevel = 1}) => {
  if (enthusiasmLevel <= 0) {
    throw new Error('You could be a little more enthusiastic. :D');
  }
  return (
    <div className="hello">
      <div className="greeting">
        Hello {name + Array(enthusiasmLevel + 1).join('!')}
      </div>
    </div>
  );
};

Hello.propTypes = {
  name: PropTypes.string.isRequired,
  enthusiasmLevel: PropTypes.number
};
```

```
// src/components/Hello.tsx

import * as React from 'react';

interface Props {
  name: string;
  enthusiasmLevel?: number;
}

const Hello: React.SFC<Props> = ({name, enthusiasmLevel = 1}) => {
  if (enthusiasmLevel <= 0) {
    throw new Error('You could be a little more enthusiastic. :D');
  }
  return (
    <div className="hello">
      <div className="greeting">
        Hello {name + Array(enthusiasmLevel + 1).join('!')}
      </div>
    </div>
  );
};
```

```
// src/components/Hello.tsx

import * as React from 'react';

interface Props {
  name: string;
  enthusiasmLevel?: number;
}

function Hello({name, enthusiasmLevel = 1}: Props) {
  if (enthusiasmLevel <= 0) {
    throw new Error('You could be a little more enthusiastic. :D');
  }
  return (
    <div className="hello">
      <div className="greeting">
        Hello {name + Array(enthusiasmLevel + 1).join('!')}
      </div>
    </div>
  );
}
```

```tsx
// src/components/Hello.tsx

import * as React from 'react';

interface Props {
  name: string;
  enthusiasmLevel?: number;
}

class Hello extends React.Component<Props, {}> {
  render() {
    if (this.props.enthusiasmLevel <= 0) {
      throw new Error('You could be a little more enthusiastic. :D');
    }
    return (
      <div className="hello">
        <div className="greeting">
          Hello {this.props.name + Array(this.props.enthusiasmLevel + 1).join('!')}
        </div>
      </div>
    );
  }
}
```

Was ist mit ...

Default Props

```
// src/components/Greeting.tsx

import * as React from 'react';

export interface Props {
  name?: string;
  className?: string;

  // some other props
  [key: string]: any;
}

export class Greeting extends React.Component<Props, {}> {
  static defaultProps: Partial<Props> = {
    name: 'Stranger‘
  };

  render() {
    return <span>Hello {this.props.name}</span>;
  }
}
```

Was ist mit ...

State

```
// src/components/Greeting.tsx

import * as React from 'react';

export interface Props {
  initialCount: number;
}

export interface State {
  count: number;
}

export class Counter extends React.Component<Props, State> {
  constructor(props: Props) {
    super(props);
    this.state = {count: props.initialCount};
  }

  increment = () => {
    this.setState({count: this.state.count + 1});
  };

  render() {
    return (
      <>
       <span>Count {this.state.count}</span>
        <button onClick={this.increment}>Increment</button>
      </>
    );
  }
}
```

Exkurs

Generics

```
// src/generics.ts

function identity(arg: any): any {
  return arg;
}

const output: string = <string> identity("myString");;
```

```typescript
// src/generics.ts

function identity(arg: any): any {
  return arg;
}

function identity<T>(arg: T): T {
  return arg;
}

const output: string = identity<string>('myString');
```

```typescript
// src/generics.ts

function logArrayLenght<T>(arg: Array<T>): Array<T> {
  console.log(arg.length);
  return arg;
}

function logActionType<T extends Action>(arg: T): T {
  console.log(arg.type);
  return arg;
}
```

```typescript
// src/union.ts

interface A {
  name: string;
}

interface B {
  age: number;
}

type Union = A & B;

const person: Union = {
  name: 'Max Mustermann‘,
  age: 12
};
```

Was ist mit ...

Higher Order Components

```tsx
// src/hoc/WithLoading.tsx

interface WithLoadingProps {
  loading: boolean;
}

const withLoading = <P extends object>(Component: React.ComponentType<P>) =>
  class WithLoading extends React.Component<P & WithLoadingProps> {
    render() {
      const {loading, ...props} = this.props as WithLoadingProps;
      return loading ? <LoadingSpinner /> : <Component {...props} />;
    }
  };
```

```tsx
// src/hoc/WithLoading.tsx

interface WithLoadingProps {
  loading: boolean;
}

const withLoading = <P extends object>(Component: React.ComponentType<P>): React.SFC<P & WithLoadingProps> =>
  ({loading, ...props}: WithLoadingProps) => (loading ? <LoadingSpinner /> : <Component {...props} />);
```

Was ist mit ...

Redux

```
// src/types/index.tsx

export interface StoreState {
  languageName: string;
  enthusiasmLevel: number;
}
```

```
// src/constants/index.tsx

export const INCREMENT_ENTHUSIASM = 'INCREMENT_ENTHUSIASM';
export type INCREMENT_ENTHUSIASM = typeof INCREMENT_ENTHUSIASM;

export const DECREMENT_ENTHUSIASM = 'DECREMENT_ENTHUSIASM';
export type DECREMENT_ENTHUSIASM = typeof DECREMENT_ENTHUSIASM;
```

```
// src/actions/index.tsx

import * as constants from '../constants';

export interface IncrementEnthusiasm {
  type: constants.INCREMENT_ENTHUSIASM;
}

export interface DecrementEnthusiasm {
  type: constants.DECREMENT_ENTHUSIASM;
}

export type EnthusiasmAction = IncrementEnthusiasm | DecrementEnthusiasm;

export function incrementEnthusiasm(): IncrementEnthusiasm {
  return {type: constants.INCREMENT_ENTHUSIASM};
}

export function decrementEnthusiasm(): DecrementEnthusiasm {
  return {type: constants.DECREMENT_ENTHUSIASM};
}
```

```
// src/reducers/index.tsx

import {EnthusiasmAction} from '../actions';
import {StoreState} from '../types/index';
import {DECREMENT_ENTHUSIASM, INCREMENT_ENTHUSIASM} from '../constants/index';

export function enthusiasm(state: StoreState, action: EnthusiasmAction): StoreState {
  switch (action.type) {
    case INCREMENT_ENTHUSIASM:
      return {...state, enthusiasmLevel: state.enthusiasmLevel + 1};
    case DECREMENT_ENTHUSIASM:
      return {...state, enthusiasmLevel: Math.max(1, state.enthusiasmLevel - 1)};
  }
  return state;
}
```

```
// src/components/Hello.container.tsx

import {connect, Dispatch} from 'react-redux';
import {StoreState} from '../types/index';
import * as actions from '../actions/';
import {Hello} from './Hello';

export function mapStateToProps({enthusiasmLevel, languageName}: StoreState) {
  return {
    enthusiasmLevel,
    name: languageName
  };
}

export function mapDispatchToProps(dispatch: Dispatch<actions.EnthusiasmAction>) {
  return {
    onIncrement: () => dispatch(actions.incrementEnthusiasm()),
    onDecrement: () => dispatch(actions.decrementEnthusiasm())
  };
}

export default connect(mapStateToProps, mapDispatchToProps)(Hello);
```

Was ist mit ...

Redux und Props

Connect

Props in vier Bausteinen

- Initiale Props
- Props mit State
- Props mit Dispatch
- Props gemerged

```
export declare function connect<TStateProps, no_dispatch, TOwnProps>(
  mapStateToProps: MapStateToPropsParam<TStateProps, TOwnProps>
): ComponentDecorator<TStateProps, TOwnProps>;

export declare function connect<no_state, TDispatchProps, TOwnProps>(
  mapStateToProps: null | undefined,
  mapDispatchToProps: MapDispatchToPropsParam<TDispatchProps, TOwnProps>
): ComponentDecorator<TDispatchProps, TOwnProps>;

export declare function connect<TStateProps, TDispatchProps, TOwnProps>(
  mapStateToProps: MapStateToPropsParam<TStateProps, TOwnProps>,
  mapDispatchToProps: MapDispatchToPropsParam<TDispatchProps, TOwnProps>
): ComponentDecorator<TStateProps & TDispatchProps, TOwnProps>;

export declare function connect<TStateProps, no_dispatch, TOwnProps, TMergedProps>(
  mapStateToProps: MapStateToPropsParam<TStateProps, TOwnProps>,
  mapDispatchToProps: null | undefined,
  mergeProps: MergeProps<TStateProps, undefined, TOwnProps, TMergedProps>
): ComponentMergeDecorator<TMergedProps, TOwnProps>;

export declare function connect<no_state, TDispatchProps, TOwnProps, TMergedProps>(
  mapStateToProps: null | undefined,
  mapDispatchToProps: MapDispatchToPropsParam<TDispatchProps, TOwnProps>,
  mergeProps: MergeProps<undefined, TDispatchProps, TOwnProps, TMergedProps>
): ComponentMergeDecorator<TMergedProps, TOwnProps>;

export declare function connect<no_state, no_dispatch, TOwnProps, TMergedProps>(
  mapStateToProps: null | undefined,
  mapDispatchToProps: null | undefined,
  mergeProps: MergeProps<undefined, undefined, TOwnProps, TMergedProps>
): ComponentMergeDecorator<TMergedProps, TOwnProps>;

...
```

index.d.ts von @types/react-redux

```
export interface OwnProps {
  // ...
}

export interface StateProps {
  // ...
}

export type Props = OwnProps & StateProps;

export const mapStateToProps = (state: any, ownProps: OwnProps): Props {
  // ...
}

connect<StateFocusedCardProps, null, OwnFocusedCardProps>(mapStateToProps)(Component);
```

```
export declare function connect<TStateProps, no_dispatch, TOwnProps>(
  mapStateToProps: MapStateToPropsParam<TStateProps, TOwnProps>
): ComponentDecorator<TStateProps, TOwnProps>;

export declare function connect<no_state, TDispatchProps, TOwnProps>(
  mapStateToProps: null | undefined,
  mapDispatchToProps: MapDispatchToPropsParam<TDispatchProps, TOwnProps>
): ComponentDecorator<TDispatchProps, TOwnProps>;

export declare function connect<TStateProps, TDispatchProps, TOwnProps>(
  mapStateToProps: MapStateToPropsParam<TStateProps, TOwnProps>,
  mapDispatchToProps: MapDispatchToPropsParam<TDispatchProps, TOwnProps>
): ComponentDecorator<TStateProps & TDispatchProps, TOwnProps>;

export declare function connect<TStateProps, no_dispatch, TOwnProps, TMergedProps>(
  mapStateToProps: MapStateToPropsParam<TStateProps, TOwnProps>,
  mapDispatchToProps: null | undefined,
  mergeProps: MergeProps<TStateProps, undefined, TOwnProps, TMergedProps>
): ComponentMergeDecorator<TMergedProps, TOwnProps>;

export declare function connect<no_state, TDispatchProps, TOwnProps, TMergedProps>(
  mapStateToProps: null | undefined,
  mapDispatchToProps: MapDispatchToPropsParam<TDispatchProps, TOwnProps>,
  mergeProps: MergeProps<undefined, TDispatchProps, TOwnProps, TMergedProps>
): ComponentMergeDecorator<TMergedProps, TOwnProps>;

export declare function connect<no_state, no_dispatch, TOwnProps, TMergedProps>(
  mapStateToProps: null | undefined,
  mapDispatchToProps: null | undefined,
  mergeProps: MergeProps<undefined, undefined, TOwnProps, TMergedProps>
): ComponentMergeDecorator<TMergedProps, TOwnProps>;

...
```

Was ist mit ...

Styled Components

```tsx
// src/components/MyH1.tsx

import * as React from 'react';
import {default as styled} from 'styled-components';

const MyH1 = styled.h1`
  color: red;
`;
```

```
// src/theme/index.ts

import * as styledComponents from 'styled-components';

const {
  default: styled,
  css,
  injectGlobal,
  keyframes,
  ThemeProvider
} = styledComponents as styledComponents.ThemedStyledComponentsModule<IThemeInterface>;

export interface IThemeInterface {
  primaryColor: string;
}

export const theme = {
  primaryColor: '#e9e9eb'
};

export default styled;

export {css, injectGlobal, keyframes, ThemeProvider};
```

```tsx
// src/components/quote.tsx

import styled from '../theme';

const Quote = styled.h1`
  color: ${props => props.theme.primaryColor};
  font: 400 36px/1.4 'cardo';
  font-style: italic;
  font-weight: normal;
  text-align: left;
  text-indent: -10px;
  max-width: 800px;
  width: 80%;
  margin: 0 auto;
`;

export default Quote;
```

https://github.com/patrick91/Styled-Components-Typescript-Example/blob/master/src/components/quote.tsx

Was ist mit ...

Enzyme

```
interface ShallowWrapper<P = {}, S = {}> extends CommonWrapper<P, S> {}
interface ReactWrapper<P = {}, S = {}> extends CommonWrapper<P, S> {}

...

const ShallowRendered: ShallowWrapper<Props, State> = shallow(<Hello />);
const MountRendered: ReactWrapper<Props, State> = mount(<Hello />);
```

Was ist mit ...

Code Style

TSLint

Prettier

TypeScript Support

- MobX
- React Router
- react-transition-group
- Firebase
- Jest, Mocha, Chai, Sinon
- Moment.js
- Fetch & Axios
- Styleguidist & Storybook
- ...

Vielen Dank

Johann Böhler

inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe

jboehler@inovex.de
0173 3181 182