

NILS HARTMANN | <https://nilshartmann.net>



React

IN ENTERPRISE-ANWENDUNGEN

Slides: <http://bit.ly/bedcon-react>

Source Code: <https://github.com/nilshartmann/react-greeting-example>

NILS HARTMANN

Programmierer aus Hamburg

Java

JavaScript, TypeScript, React

Trainings, Workshops

nils@nilshartmann.net

@NILSHARTMANN

Enterprise

Anwendungen

Enterprise Anwendungen

- Große Code-Basis
- Viele Entwickler, mehrere Teams
- Langliebig

Herausforderungen: beim Entwickeln

- Wartbarkeit
- Mehrere Entwickler und Teams müssen Code bearbeiten und verstehen
 - Am besten auch noch nach einem Jahr
- Neue Team-Mitglieder müssen Code verstehen

Herausforderungen: zur Laufzeit

- Performance
- Bundle Größen
 - Auswirkung auf Start der Anwendung

Beispiel Anwendung

<https://github.com/nilshartmann/react-greeting-example>

Code Struktur

für React-Anwendungen

Es gibt kein "richtig" oder "falsch"

- Möglichst viel Logik in Komponente lassen
 - Sehr einfach zu verstehen
- Möglichkeiten, render zu strukturieren:
 - Mehrere Funktionen
 - Mehrere Komponenten
- "Private" Komponenten uU in selber Datei lassen
- Falls Datei zu groß, in eigenes Modul verschieben, aber im selben Ordner lassen
- Alles was fachlich zu einer Einheit gehört, in einem Ordner lassen
 - Keine Trennung nach technischen Artefakten

Komponente als Self-Contained System

- Möglichst viel Logik in Komponente lassen
 - Sehr einfach zu verstehen
- Möglichkeiten, render zu strukturieren:
 - Mehrere Funktionen
 - Mehrere Komponenten
- "Private" Komponenten uU in selber Datei lassen
- Falls Datei zu groß, in eigenes Modul verschieben, aber im selben Ordner lassen
- Alles was fachlich zu einer Einheit gehört, in einem Ordner lassen
 - Keine Trennung nach technischen Artefakten

Code-Basis strukturieren: Component Folder Pattern

<https://medium.com/styled-components/component-folder-pattern-ee42df37ec68>

Große Anwendungen in mehrere Module zerlegen

- Publish nach Nexus
- Guter Kandidat: Komponentenbibliothek
- yarn link zum einfachen Testen
- Storybook zum Präsentieren der Komponenten

Fehler-Handling

Fehler vermeiden & behandeln

REACT STRICT MODE

Neue Komponente: `React.StrictMode` (16.3)

- Nur im Development Modus aktiv
- Warnt vor typischen React-Fehlern
 - Zum Beispiel Lifecycle-Hooks, die deprecated sind
- Wird künftig wohl noch weitere Prüfungen geben

```
class GreetingApp extends React.Component {  
  render() {  
    return <React.StrictMode>  
      // hier kommt die Anwendung  
    </React.StrictMode>;  
  }  
}
```

REACT STRICT MODE

Neue Komponente: `React.StrictMode` (16.3)

- Nur im Development Modus aktiv
- Warnt vor typischen React-Fehlern
 - Zum Beispiel Lifecycle-Hooks, die deprecated sind
- Wird künftig wohl noch weitere Prüfungen geben

```
class GreetingApp extends React.Component {  
  render() {  
    return <React.StrictMode>  
      // hier kommt die Anwendung  
    </React.StrictMode>;  
  }  
}
```

```
⊗ ▶ Warning: Unsafe lifecycle methods were found within a strict-mode tree: warning.js:33  
    in Switch (created by GreetingApp)  
    in div (created by GreetingApp)  
    in Router (created by BrowserRouter)  
    in BrowserRouter (created by GreetingApp)  
    in GreetingApp  
  
componentWillMount: Please update the following components to use componentDidMount instead: Route  
  
componentWillReceiveProps: Please update the following components to use static  
getDerivedStateFromProps instead: Chart, Route  
  
Learn more about this warning here:  
https://fb.me/react-strict-mode-warnings
```

ERROR BOUNDARIES

Globale Fehlerbehandlung: componentDidCatch

- Lifecycle Hook seit React 16.2
- Fängt Fehler auf, die beim rendern in einer **unterliegenden** Komponente aufgetreten sind

```
class ErrorHandler extends React.Component {
```

Lifecycle Hook
(könnte Fehler z.B. auch
an einen Server
schicken/alarmieren)

```
  componentDidCatch(error, errorInfo) {  
    this.setState({ hasError: true })  
  }
```

```
}
```


ERROR BOUNDARIES

Globale Fehlerbehandlung: componentDidCatch

- Lifecycle Hook seit React 16.2
- Fängt Fehler auf, die beim rendern in einer **unterliegenden** Komponente aufgetreten sind

```
class ErrorHandler extends React.Component {  
  
  componentDidCatch(error, errorInfo) {  
    this.setState({ hasError: true })  
  }  
  
  render() {  
    return this.state.hasError ?  
      <h1>Ein Fehler aufgetreten!</h1>  
      :  
      this.props.children;  
  }  
}
```

Lifecycle Hook
(könnte Fehler z.B. auch
an einen Server
schicken/alarmieren)

Fehlermeldung
anzeigen

ERROR BOUNDARIES

Globale Fehlerbehandlung: componentDidCatch

- Verwendung

```
class GreetingApp extends React.Component {  
  
  render() {  
    return <ErrorHandler>  
      <FachlicheKomponenten .... />  
    </ErrorHandler>  
  }  
}
```

TypeScript

für React-Anwendungen

TYPESCRIPT UND REACT: PROPERTIES

Komponenten Props als Typen in TypeScript

```
function GreetingList(props: GreetingListProps) {  
  return <table>{props.greetings.map(...)}</table>;  
}
```

```
interface Greeting { . . . };  
interface GreetingListProps {  
  greetings: Greeting[]  
};
```

Typ definieren

Überprüfung zur
Compile-Zeit
(auch direkt in der IDE)

```
[ts]  
Type '{ greetings: string; }' is not assignable to type 'IntrinsicAttributes & GreetingListProps'.  
  Type '{ greetings: string; }' is not assignable to type 'GreetingListProps'.  
    Types of property 'greetings' are incompatible.  
      Type 'string' is not assignable to type 'Greeting[]'.
```

```
(JSX attribute) greetings: string
```

```
<GreetingList greetings="hello" />
```

TYPESCRIPT UND REACT: PROPERTIES & STATE

Komponenten-Klassen - 1

- Types für Properties und State definieren

Typ für Props

```
interface GreetingComposerProps {  
  initialName: string;  
  onSave(newGreeting: NewGreeting) => void;  
};
```

Typ für State

```
interface GreetingComposerState = {  
  name: string;  
  greeting: string;  
};
```

TYPESCRIPT UND REACT: PROPERTIES & STATE

Komponenten-Klassen - 2

- Types in Komponenten-Klasse angeben

Typ für Props

```
interface GreetingComposerProps {  
  initialName: string;  
  onSave(newGreeting: NewGreeting) => void;  
};
```

Typ für State

```
interface GreetingComposerState = {  
  name: string;  
  greeting: string;  
};
```

Typen angeben

```
class GreetingComposer extends React.Component  
  <GreetingComposerProps, GreetingComposerState> {  
  . . .  
}
```

Komponenten-Klassen - 3

- State initialisieren (optional)

```
class GreetingComposer extends React.Component
  <GreetingComposerProps, GreetingComposerState> {

  readonly state: GreetingComposerState = {
    greeting: "",
    name: ""
  }
  . . .
}
```

TYPESCRIPT UND REACT: PROPERTIES & STATE

Typische Fehler, die durch TypeScript aufgedeckt werden

Potentielle Fehler

```
// Properties sind read-only
this.props.initialName = null;

// Nur bekannte Properties dürfen verwendet werden
const x = this.props.not_here;

// State muss vollständig initialisiert werden
this.state = {name: ""}; // greeting fehlt

// this.state darf nur im Konstruktor verwendet werden
this.state.name = ""; // außerhalb des Cstr

// Elemente im State müssen korrekten Typ haben
this.setState({name: 7}); // 7 is not a string

// Unbekannte Elemente dürfen nicht in den State
gesetzt werden
this.setState({notHere: 'invalid'});
```


Build Optimierung

Bundle-Größe optimieren

Problem: Große Bundle-Datei

Name	Größe
 main.js	5.8 MB

- Browser muss viele Daten laden (**Netzwerk!**)
- Browser muss große JavaScript-Datei parsen (**CPU!**)
- Browser muss das JavaScript ausführen (**CPU!**)
- ...erst jetzt ist die Anwendung bereit!

BUNDLE OPTIMIERUNG


Schritt 1: Production-Mode von Webpack

Kommandozeile

```
webpack --mode production
```

oder webpack.config.js:

```
module.exports = {  
  entry: . . . ,  
  output: { . . . },  
  mode: "production"  
}
```

Name	▼ Änderungsdatum	Größe
 main.js	Heute	3.3 MB

WARNING in entrypoint size limit: The following entrypoint(s) combined as set size exceeds the recommended limit (244 KiB). This can impact web performance.

Entrypoints:

main (3.31 MiB)

main.js

Schritt 1: Production-Mode von Webpack

Hintergrund **TODO**
welche Plug-ins etc



Schritt 2: Source Maps auslagern

- Browser lädt Source Maps erst wenn benötigt

webpack.config.js:

```
module.exports = {  
  entry: . . . ,  
  output: { . . . },  
  mode: "production",  
  devtool: "source-map" // z.B. oder none  
}
```

Ergebnis...

	main.js	Heute	567.7 KB
	main.js.map	Heute	2 MB

- In Webpack 4 Standard im Production Mode (wenn nichts anderes konfiguriert ist)

BUNDLE OPTIMIERUNG



Schritt 2: Source Maps auslagern

- Browser lädt Source Maps erst wenn benötigt

webpack.config.js:

```
module.exports = {  
  entry: . . . ,  
  output: { . . . },  
  mode: "production",  
  devtool: "source-map" // z.B. oder none  
}
```

Ergebnis...

	main.js	Heute	567.7 KB
	main.js.map	Heute	2 MB

...aber immer noch groß

```
WARNING in entrypoint size limit: The following entrypoint(s) combined as  
set size exceeds the recommended limit (244 KiB). This can impact web per  
formance.  
Entrypoints:  
  main (568 KiB)  
    main.js
```

Schritt 3: Externe Libs nicht ins Application Bundle

- Stattdessen z.B. über CDN laden
- Können sehr gut gecacht werden (vom Browser u.a.)
- Mehrere parallele Requests zum Laden möglich

webpack.config.js:

```
module.exports = {  
  externals: {  
    "react": "React",  
    "react-dom": "ReactDOM",  
    . . .  
  }  
}
```

index.html

```
<script src="https://.../react.prod.min.js" >  
<script src="https://.../react-dom.prod.min.js" />
```



BUNDLE OPTIMIERUNG

Schritt 3: Externe Libs nicht ins Application Bundle

- Stattdessen z.B. über CDN laden
- Können sehr gut gecacht werden (vom Browser u.a.)
- Mehrere parallele Requests zum Laden möglich

webpack.config.js:

```
module.exports = {  
  externals: {  
    "react": "React",
```

Name	▼ Änderungsdatum	Größe
 main.js.map	Heute	281.6 KB
 main.js	Heute	69.7 KB

index.html

```
<script src="https://.../react.prod.min.js" >  
<script src="https://.../react-dom.prod.min.js" />  
<script src="https://.../react-dom.prod.min.js" />
```


Bonus: Commit-Id in index.html

Code Splitting

Asynchrone Importe

Asynchrones Laden von Modulen

- Erlaubt das dynamische Nachladen von Code-Teilen
- Beim Start der Anwendung nur direkt benötigte Teile laden (Minimalversion)
- Weitere Teile werden erst bei Benutzerinteraktion oder im Hintergrund geladen
- Basiert auf `dynamic import` (Nicht Standard, aber Stage 3)

<https://reactjs.org/docs/code-splitting.html>

<https://webpack.js.org/guides/code-splitting/>

CODE SPLITTING

Beispiel: Ein Modul dynamisch importieren - 1

Ein JS Modul
(calculator.js)

```
// calculator.js  
export default function calculator(a, b) {  
  return a+b;  
}
```

CODE SPLITTING

Beispiel: Ein Modul dynamisch importieren - 2

Ein JS Modul
(calculator.js)

```
// calculator.js
export default function calculator(a, b) {
  return a+b;
}
```

Zum Vergleich: **statischer**
Import

```
// Verwender
import calculator from "./calculator";
console.log(calculator(7,8));
```

CODE SPLITTING

Beispiel: Ein Modul dynamisch importieren - 3

Ein JS Modul
(calculator.js)

```
// calculator.js
export default function calculator(a, b) {
  return a+b;
}
```

Zum Vergleich: statischer
Import

```
// Verwender
import calculator from "./calculator";
console.log(calculator(7,8));
```

1. Modul wird asynchron geladen

2. Modul steht zur Verfügung

```
// Verwender (asynchroner Import)
import("./calculator").
  then(calculatorModule => {
    const calculator = calculatorModule.default;
    console.log(calculator(7, 8));
  })
}
```

Nachladen von React-Komponenten

- Erfordert die Darstellung von Platzhaltern, bis die eigentliche Komponente geladen ist
- Wenn die eigentliche Komponente geladen ist, muss die umschließende Komponente neu dargestellt werden
 - Geladene Komponente kann in den State gesetzt werden
 - Alternativ *forceUpdate* zum neuen rendern
- Fertige Lösung: react-loadable
(<https://github.com/jamiebuilds/react-loadable>)

CODE SPLITTING

Beispiel: Eine React-Komponente nachladen - 1

Eine Loader-Komponente

```
class LoadableGreetingComposer extends React.Component {  
  state = {};
```

Ziel-Komponente laden

```
  async componentDidMount() {  
    const result = await import("./GreetingComposer");  
    this.setState  
      ({ GreetingComposer: result.default });  
  }
```

```
}
```


CODE SPLITTING

Beispiel: Eine React-Komponente nachladen - 2

Eine Loader-Komponente

```
class LoadableGreetingComposer extends React.Component {  
  state = {};
```

Ziel-Komponente laden

```
  async componentDidMount() {  
    const result = await import("./GreetingComposer");  
    this.setState(  
      ({ GreetingComposer: result.default });  
    );  
  }
```

Ziel-Komponente anzeigen oder Platzhalter

```
  render() {  
    if (this.state.GreetingComposer) {  
      return <this.state.GreetingComposer {...this.props} />;  
    }  
  
    return <span>Loading...</span>;  
  }  
}
```

CODE SPLITTING

Beispiel: Eine React-Komponente nachladen - 3

Eine Loader-Komponente	<pre>class LoadableGreetingComposer extends React.Component { state = {};</pre>
Ziel-Komponente laden	<pre> async componentDidMount() { const result = await import("./GreetingComposer"); this.setState(({ GreetingComposer: result.default });); }</pre>
Ziel-Komponente anzeigen oder Platzhalter	<pre> render() { if (this.state.GreetingComposer) { return <this.state.GreetingComposer {...this.props} />; } return Loading...; } }</pre>
Verwendung	<pre><LoadableGreetingComposer initialName="Klaus" /></pre>

CODE SPLITTING

Webpack: Ausgabe-Datei benennen

Gewünschten
Namen beim
Import als
Kommentar angeben

```
class LoadableGreetingComposer extends React.Component {  
  state = {};  
  
  async componentDidMount() {  
    const result =  
      await import(/*GreetingComposer*/ "../GreetingComposer");  
    this.setState  
      ({ GreetingComposer: result.default });  
  }  
  
  . . .  
}
```

Fertige Lösung: React Loadable

- <https://github.com/jamiebuilds/react-loadable>
- Fehler-Handling
- Vor-Laden von Komponenten
- Unterstützung für Server-Site Rendering

CODE SPLITTING

Fertige Lösung: React Loadable

- <https://github.com/jamiebuilds/react-loadable>
- Fehler-Handling
- Vor-Laden von Komponenten
- Unterstützung für Server-Site Rendering

Eine Loading-Komponente

```
function LoadingComponent({error}) {  
  return error ? "Error 😞" : "Loading...";  
}
```

CODE SPLITTING

Fertige Lösung: React Loadable

- <https://github.com/jamiebuilds/react-loadable>
- Fehler-Handling
- Vor-Laden von Komponenten
- Unterstützung für Server-Site Rendering

Eine Loading-Komponente

```
function LoadingComponent({error}) {  
  return error ? "Error 😞" : "Loading...";  
}
```

```
import Loadable from "react-loadable";
```

Die Loader-Komponente

```
const LoadableComposer = Loadable({  
  loader: () => import("./GreetingComposer"),  
  loading: LoadingComponent  
});
```

CODE SPLITTING

Fertige Lösung: React Loadable

- <https://github.com/jamiebuilds/react-loadable>
- Fehler-Handling
- Vor-Laden von Komponenten
- Unterstützung für Server-Site Rendering

Eine Loading-Komponente

```
function LoadingComponent({error}) {  
  return error ? "Error 😞" : "Loading...";  
}
```

```
import Loadable from "react-loadable";
```

Die Loader-Komponente

```
const LoadableComposer = Loadable({  
  loader: () => import("./GreetingComposer"),  
  loading: LoadingComponent  
});
```

Verwendung

```
<LoadableComposer />
```

CODE SPLITTING

Beispiel: Code Splitting mit React Router und React Loadable

```
class LoadingComponent { . . . }

const LoadableAdminPage = Loadable({
  loader: () => import("./pages/AdminPage"),
  loading: LoadingComponent
});

const LoadableDisplayPage = Loadable({
  loader: () => import("./pages/DisplayPage"),
  loading: LoadingComponent
});

const GreetingApp = () => (
  <Router>
    <Route path="/greet/:greetingId" component={LoadableDisplayPage} />
    <Route path="/" component={LoadableAdminPage} />
  </Router>
);
```


CODE SPLITTING

Beispiel: Code Splitting mit React Router und React Loadable

```
class LoadingComponent { . . . }
```

```
const LoadableAdminPage = Loadable({  
  loader: () => import("./pages/AdminPage"),  
  loading: LoadingComponent  
});
```

```
const LoadableDisplayPage = Loadable({  
  loader: () => import("./pages/DisplayPage"),  
  loading: LoadingComponent  
});
```

```
const GreetingApp = () => (  
  <Router>  
    <Route path="/greet/:greetingId" component={LoadableDisplayPage} />  
    <Route path="/" component={LoadableAdminPage} />  
  </Router>  
);
```

Demo

- <http://localhost:8000>

CODE SPLITTING





Beispiel: Code Splitting mit React Router und React Loadable

```
class LoadingComponent { . . . }
```

```
const LoadableAdminPage = Loadable({  
  loader: () => import("./pages/AdminPage"),  
  loading:  
});
```

```
const LoadableDisplayPage = Loadable({  
  loader: () => import("./pages/DisplayPage"),  
  loading:  
});
```

```
const GreetingApp = () => (  
  <Router>  
    <Route path="/greet/:greetingId" component={LoadableDisplayPage} />  
    <Route path="/" component={LoadableAdminPage} />  
  </Router>  
);
```

Name	Größe
 AdminPage.js	6.7 KB
 DisplayPage.js	3 KB
 GreetingComposer.js	1.4 KB
 main.js	65.6 KB

Caching

CACHING

Browser Request vermeiden: JavaScript im Browser cachen

- Erfordert eindeutige Dateinamen
 - Hash-Namen mit Webpack generieren

Eindeutige Namen
erzeugen:

```
module.exports = {  
  output: {  
    filename: "[name].[chunkhash].js",  
    . . .  
  }  
  
}
```

CACHING

Browser Request vermeiden: JavaScript im Browser cachen

- Erfordert eindeutige Dateinamen
 - Hash-Namen mit Webpack generieren
 - index.html mit erzeugten Files generieren

Eindeutige Namen
erzeugen:

```
module.exports = {  
  output: {  
    filename: "[name].[chunkhash].js",  
    . . .  
  }  
}
```

index.html generieren:

```
plugins: [  
  new HtmlWebpackPlugin({  
    filename: . . .  
    template: . . .  
  })  
]  
}
```

CACHING

Browser Request vermeiden: JavaScript im Browser cachen

- Cache-Header im Server setzen

Beispiel: nginx

```
map $sent_http_content_type $expires {  
    text/html                epoch;  
    application/javascript    max;  
    text/css                  max;  
}  
  
server {  
    listen ...;  
    . . .  
  
    expires $expires;  
}
```

CACHING

Browser Request vermeiden: JavaScript im Browser cachen

- Cache-Header im Server setzen

Beispiel: nginx

Demo

- <http://localhost:9000>

```
$expires {  
    epoch;  
    max;  
    max;
```

```
server {  
    listen ...;  
    . . .  
  
    expires $expires;  
}
```

E2E Tests

Testen im Browser

TestCafe & Puppeteer

- TC steuert diverse Browser fern (über Adapter)
 - kein Selenium!
- Lassen sich per npm/yarn installieren
- Laufen headless im Jenkins Build mit
- TestCafe bringt "selectors" für React Components mit

TestCafe: Test

- Code Beispiel Testcafe
- Code Beispiel Ausführen mit Package Json
- Code Beispiel für Screenshots und Debugging (eventuell live)

Vielen Dank!

<http://bit.ly/bedcon-react>

Fragen?

[HTTPS://NILSHARTMANN.NET](https://nilshartmann.net) | @NILSHARTMANN