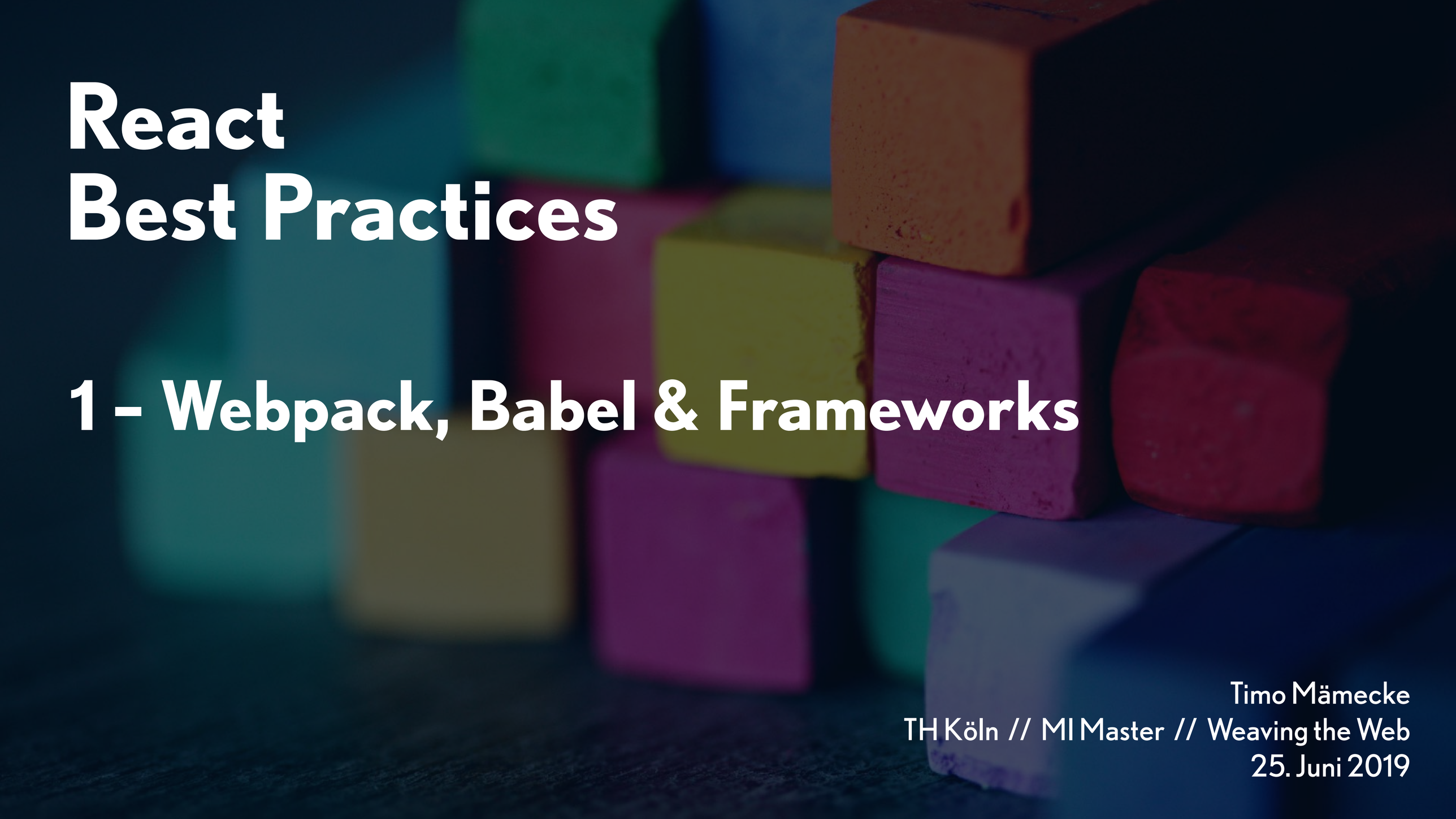


React Best Practices

1 – Webpack, Babel & Frameworks

Timo Mämecke
TH Köln // MI Master // Weaving the Web
25. Juni 2019

Compiled modernes JavaScript
in Browser-kompatibles JavaScript.

Beachtet nur JavaScript.

Packt alle Dateien eines JavaScript-Projekts in ein zusammengehöriges Bundle.

JavaScript, CSS, Bilder, Fonts, …

Nutzt Babel für JavaScript.

Muss ich Babel und Webpack beherrschen?

Man will und kann viel Kontakt damit vermeiden.

Früher oder später kommt man immer in Kontakt damit.

Wissen was es ist und was es macht.

Grob wissen, wie man es konfigurieren kann.

Nicht wissen, wie man es von 0 auf konfiguriert.

Deshalb: kurzer Einblick in Babel & Webpack


BABEL

Was Babel macht

```
import React from 'react'

export default function Greeting(props) {
  return <div>Hello, {props.name}!</div>
}
```

Wir schreiben modernes JavaScript.

BABEL

parsing, transforming, printing

Babel generiert browser-kompatibles JavaScript.

```
Object.defineProperty(exports, "__esModule", {
  value: true
});
exports.default = Greeting;

var _react =
_interopRequireDefault(require("react"));

function _interopRequireDefault(obj) { return obj &&
obj.__esModule ? obj : { default: obj }; }

function Greeting(props) {
  return _react.default.createElement("div", null,
"Hello, ", props.name, "!");
}
```

Warum wir Babel brauchen

Unterschiedlicher Browser-Support für modernes JavaScript.

ES6, ES7, ES8, ES9 … ESNext

Wir nutzen schon Syntax, die noch gar nicht standardisiert wurde.

Wir nutzen JSX, was kein Teil von ECMAScript ist.

Babel konfigurieren

Meist[1] via .babelrc im Projekt-Root

Plugins: definieren wie Code geparsed und transformed wird.

Presets: Sammlung von Plugins.

Presets & Plugins müssen installiert werden.

[1] https://babeljs.io/docs/en/config-files

Beispiel: .babelrc

```
{
  "presets": [
    [
      "@babel/preset-env",
      { "useBuiltIns": "entry" }
    ],
    "@babel/preset-react"
  ],

  "plugins": [
    "@babel/plugin-proposal-decorators"
  ],

  "env": {
    "test": {
      "plugins": [
        "jest-hoist"
      ]
    }
  }
}
```

Preset für Support von neustem JS

Preset speziell für React (z.B. JSX)

Zusätzliche Plugins

Environment-spezifische Overrides


webpack

Was Webpack macht

Liest einen Entrypoint ein (z.B. index.js).

Liest alle Abhängigkeiten (und deren Abhängigkeiten) ein
z.B. andere JavaScript Dateien, CSS, Bilder, Fonts, ...

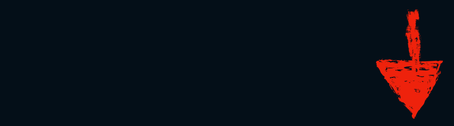
Transformiert alle Dateien mittels verschiedenen Loadern, basierend auf Dateinamen (z.B. *.js Dateien mit babel-loader).

Gibt alle transformierte Dateien in einen Ordner aus.

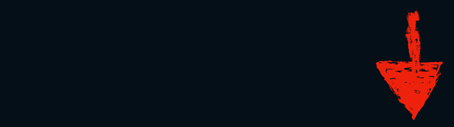
Ergebnis: ein zusammengehöriger Build.

Was Webpack macht

```
import fancyImage from './assets/fancy-image.png'
<img src={fancyImage} />
```




```
const fancyImage = require('./assets/fancy-image.png')
React.createElement('img', { src: fancyImage })
```




```
const fancyImage = '/fancy-image.a32f40.png'
// ./assets/fancy-image.png -> ./build/fancy-image.a32f40.png
React.createElement('img', { src: fancyImage })
```

Beispiel: webpack.config.js

```
module.exports = {
  entry: './index.js',
  output: {
    path: './build',
    filename: 'index.bundle.js'
  },
  module: {
    rules: [
      {
        test: /\.jsx?$/,
        loader: "babel-loader"
      },
      {
        test: [/\.jpe?g$/, /\.png$/],
        loader: "file-loader"
      }
    ]
  }
}
```

Entrypoint des Projekts

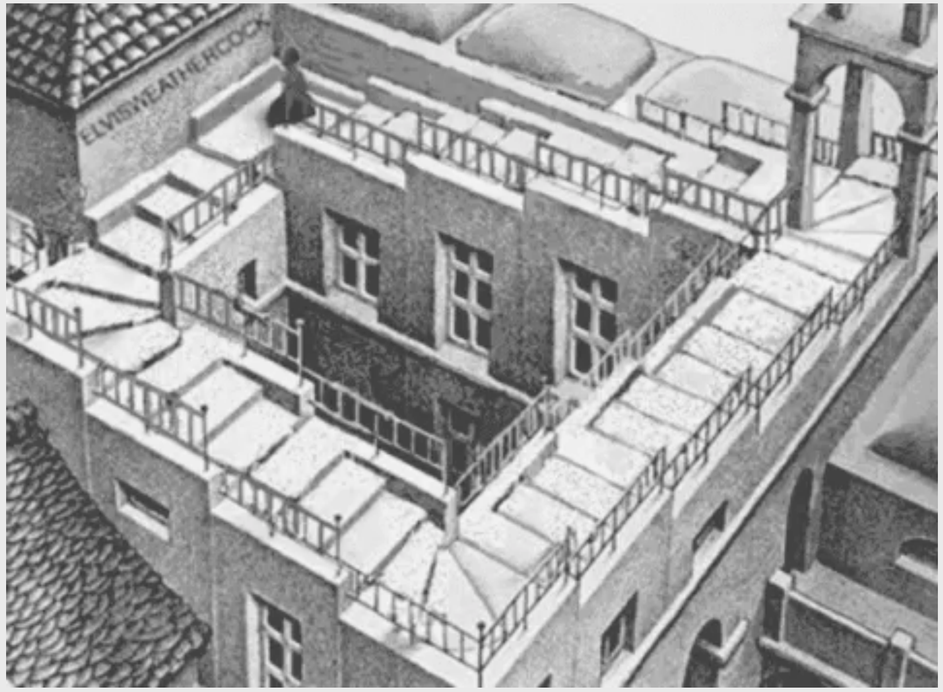
Ausgabeordner

Loader-Regeln

babel-loader für *.js und *.jsx

file-loader für *.jpg, *.jpeg und *.png

* Sehr vereinfachtes Beispiel. Echte configs sind viel komplizierter.



Sinnbild: Webpack komplett selbst konfigurieren


Frameworks

Warum Frameworks?

Keine Zeit mit Konfiguration verschwenden.

Fehler vermeiden, die Builds zu groß oder kaputt machen.

Wenn es unbedingt sein muss: vorhandene Konfigurationen anpassen.
z.B. loader hinzufügen, babel-plugins hinzufügen

Beliebteste Frameworks

Create React App[1] (CRA)

Gatsby

Next.js

Speziell für Ruby on Rails: webpacker

[1] Create React App ist ein Tool, kein Framework.

Create React App

https://facebook.github.io/create-react-app/

"pure React experience". Keine Besonderheiten, sehr einfach zum Starten.

Von Facebook maintained.

Kein eigener Server, baut statische Dateien.[1]

Keine vorgegebene Projektstruktur.

Vorgefertigte und nicht anpassbare[2] Webpack und Babel Configs.

[1] Daher einfaches Deployment: https://facebook.github.io/create-react-app/docs/deployment

[2] Es gibt Projekte, mit denen die Config angepasst werden kann. Unter anderem react-app-rewired, rescripts, carco.
Siehe auch: https://twitter.com/dan_abramov/status/1045809734069170176

Gatsby

https://www.gatsbyjs.org/

Kein eigener Server, baut statische Dateien.

Static Site Generator, der Content aus vielen Quellen beziehen kann.

Pre-rendering von statischem Content (für SEO, Ladezeiten).

Vorgegebene Projektstruktur.

Vorgefertigte, aber anpassbare Webpack und Babel Configs.

Sehr viele Plugins für Gatsby.

Next.js

https://nextjs.org

Unterstützt Server-Side Rendering.

Möglich als Server, integrierbar in eigenen Node-Server, oder als statische Dateien.

Vorgegebene Projektstruktur.

Vorgefertigte, aber anpassbare Webpack und Babel Configs.

Vergleichbar mit einer Erweiterung von Create React App.

Welches nehmen?

Create React App

- wenn man wenig Erfahrung mit React hat und pures React erleben will
- wenn man vorwiegend dynamischen Inhalt hat, der immer asynchron geladen werden muss
- wenn man auf experimentellsten neuen Features verzichten kann

Gatsby

- wenn man schon Erfahrung mit React hat
- wenn man viel statischen Content hat, der ohne Dev-Skills geändert werden kann

Next.js

- wenn man schon Erfahrung mit React hat
- wenn man einen Node Server hat
- sehr komplexe und flexible Anwendungen

Wie entscheiden?

Gemeinsam abwägen, wie Art und Anforderungen des Projektes zu einem Framework passen.

Skills im Team in Entscheidung einbeziehen.

Man braucht nicht immer die tollsten Babel- oder Webpack-Features.

tl;dr

Babel & Webpack sollte man kennen, aber nicht beherrschen.

Keine Webpack- und Babel-Config selbst schreiben. Frameworks oder Starter Kits verwenden. Wenn es sein muss, deren Config anpassen.

Create React App ist kein Framework, erwartet nicht zu viel.

Gatsby ist eine gute Wahl als Framework fur standalone Frontend-Anwendungen, wenn man mit React schon Kontakt hatte.