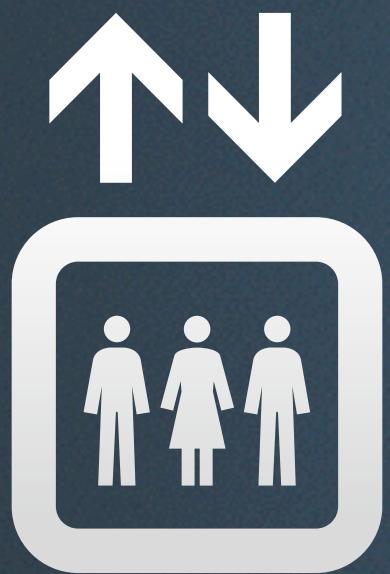


Webpack



move:elevator
→ GROUP

github.com/KevinHerklotz/webpack-demo



Über mich

- ▶ Kevin Herklotz
- ▶ 26 Jahre
- ▶ Frontend-Developer bei move:elevator
- ▶ ca. 1,5 Jahre Berufserfahrung



Inhalt

1. Was ist Webpack
2. Voraussetzungen
3. CLI
4. Loader
5. Konfigurations-Datei
6. Plugins
7. Code Splitting
8. Hot Module Replacement
9. Zusammenfassung



1. Was ist Webpack?

sebastian-gebhardt.de

Home
News
Über mich
Software
TV-Total Emulator
Wallpaper Swap II
One Package Installer
O.P.I. Wizard
WebPack Installer
WebPack Builders
17° FTP
LogoRandomizer
Shutdown Manager
Bug Report

W3C XHTML 1.0 ✓
W3C CSS ✓

WebPack v1.1.3

[Projektentwicklung: Benutzer | Stadium: Release]
[Download | Installation | Konfiguration | Neues Setup]

WebPack ist ein Installer für Windows ab Version 95. Mit WebPack hat man die Möglichkeit, eine komplette Installation aus einer einzigen Datei heraus zu starten.

Der Vorteil an WebPack ist, dass die Installationspakete sehr klein sind im Gegensatz zu anderen Anbietern.

Die Software unterstützt die meisten Installationsroutinen gängigen Funktionen. Unter anderem:

- Aufforderung zum Anlegen einer Planz
- Anzeigen einer Ordner
- Wahl zwischen mehreren Setup-Typen (Normal, Kompakt, Benutzer)
- Auswahl des Installationsordners
- Wahl eines Ordners im Startmenü
- Anpassung der Registrierung
- Erstellen von Verknüpfungen
- Start einer Anwendung nach der Installation
- Dialoge in Deutsch und Englisch
- Fortschrittsanzeige bei der Installation

Download

Webpack wird für zahlreiche Installationen von Software auf dieser Seite vorausgesetzt. Sie benötigen mindestens Windows 95, damit WebPack arbeitet. Unter folgender Adresse bekommen Sie WebPack:

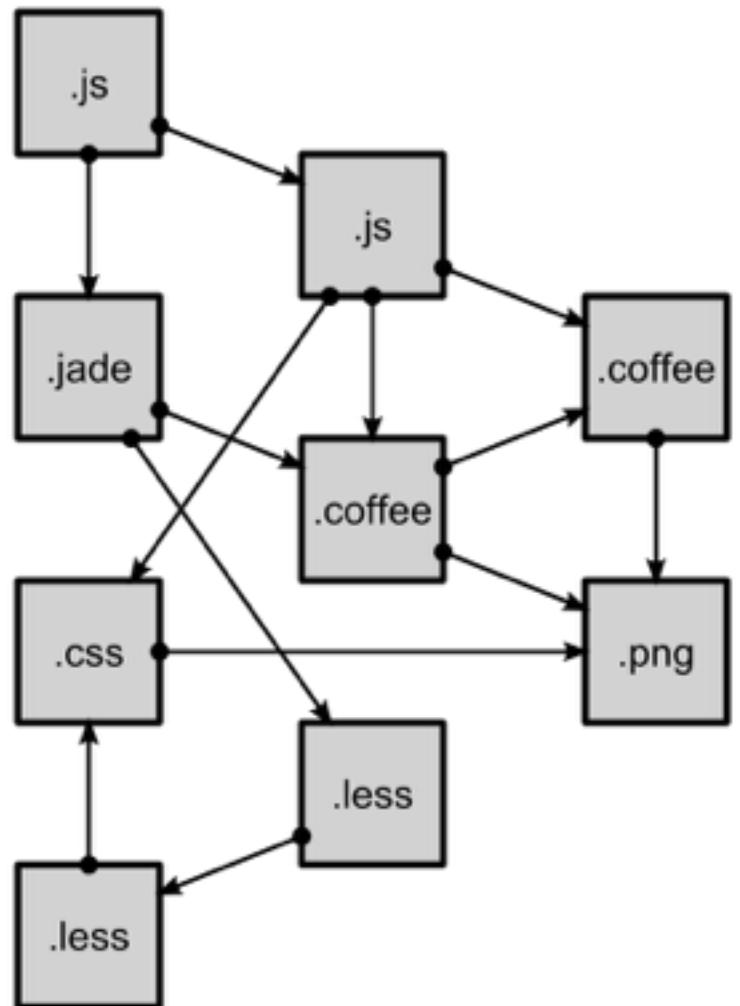
[WebPack Installer v1.1.3 herunterladen]

1. Was ist Webpack?

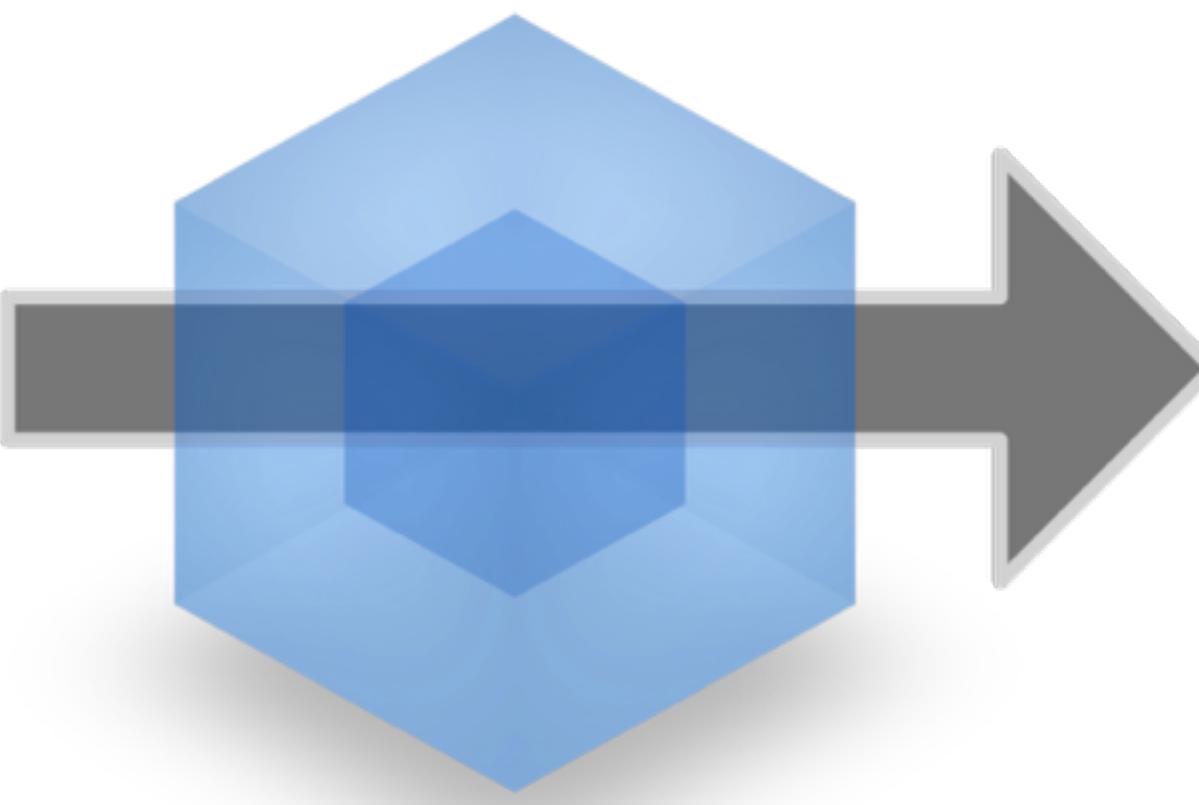
build tool / module bundler

- ▶ teilt Abhängigkeiten in „Chunks“ auf, die bei Bedarf geladen werden

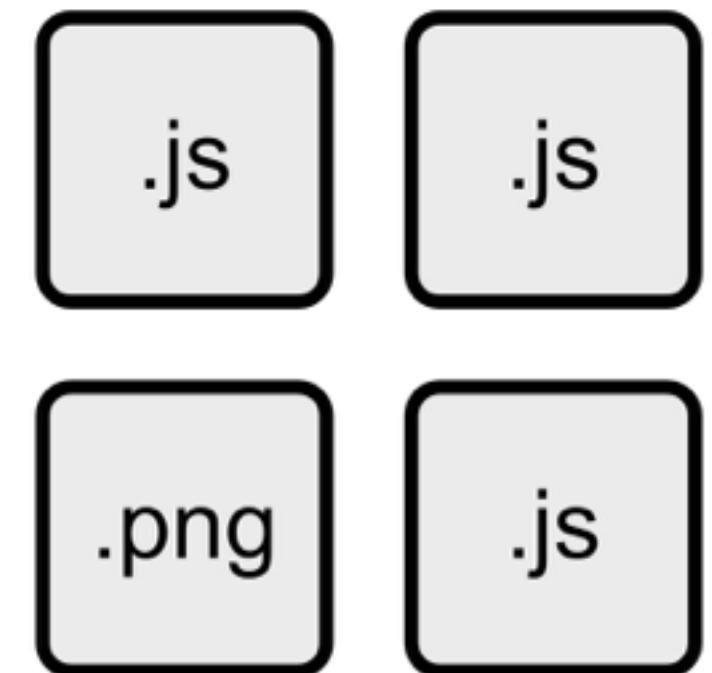




modules
with dependencies



webpack
MODULE BUNDLER



static
assets



1. Was ist Webpack?

build tool / module bundler

- ▶ teilt Abhängigkeiten in „Chunks“ auf, die bei Bedarf geladen werden
- ▶ Werkzeuge für Entwicklung (Source Maps, Hot Reload, Minifizierer, ...)
- ▶ umfangreiche Erweiterbarkeit
- ▶ umfangreiche Konfigurierbarkeit
- ▶ geeignet für sehr große Projekte



2. Voraussetzungen

- ▶ Node.js
- ▶ (npm install webpack -g)
- ▶ (npm install webpack-dev-server -g)



3. CLI

▶ **webpack <entry> <output>**

-d

--debug --devtool source-map --output-pathinfo

-p

--optimize-minimize --optimize-occurrence-order

--watch

Watches all dependencies and recompile on change

--progress

compilation progress to stderr

--display-chunks

Display the separation of the modules into chunks

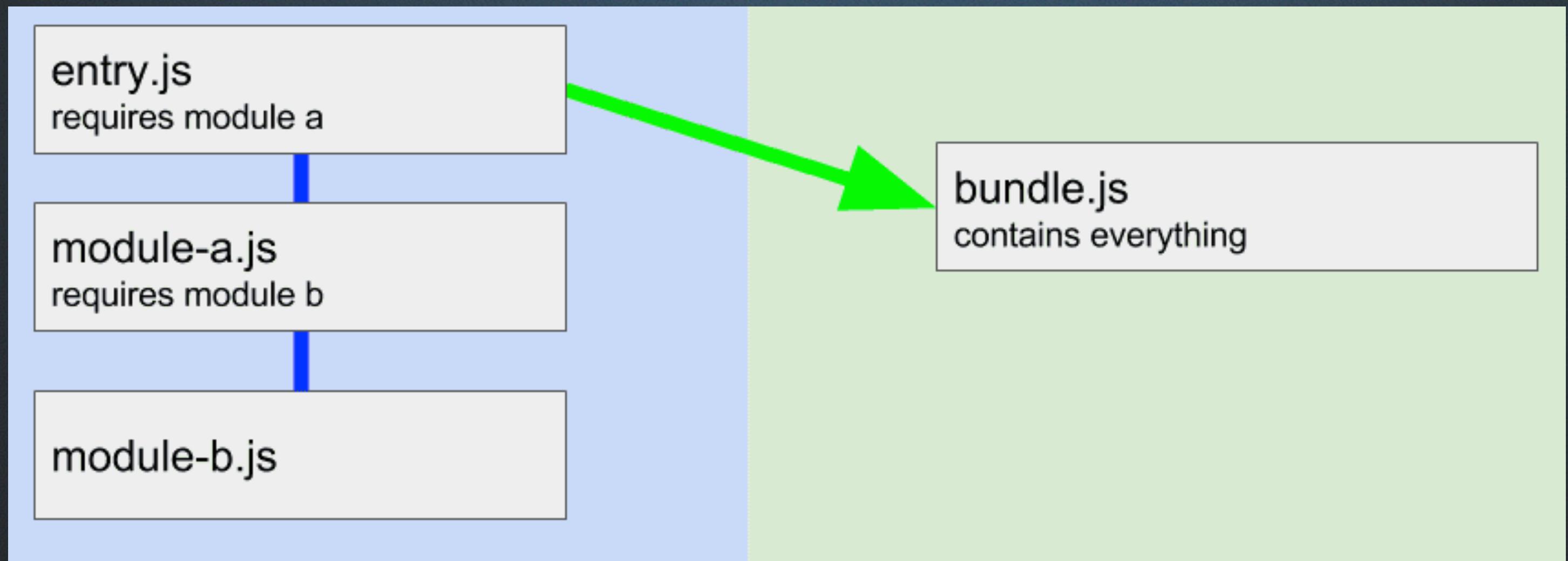
--display-reasons

Show more information about the reasons why a module is included



3. CLI - Beispiel

► `webpack entry.js bundle.js`



► siehe Code-Beispiel 1

4. Loader

- ▶ erweitert das Handling mit Modulen
- ▶ gibt String zurück
- ▶ Beispiele: svg, uglify, babel, jade, twig, mustache, ngtemplate, react-templates, sass, autoprefixer, eslint, ...

```
1. npm install {name}-loader  
2. In JS-Datei: require('{name}!./path/to/file');
```

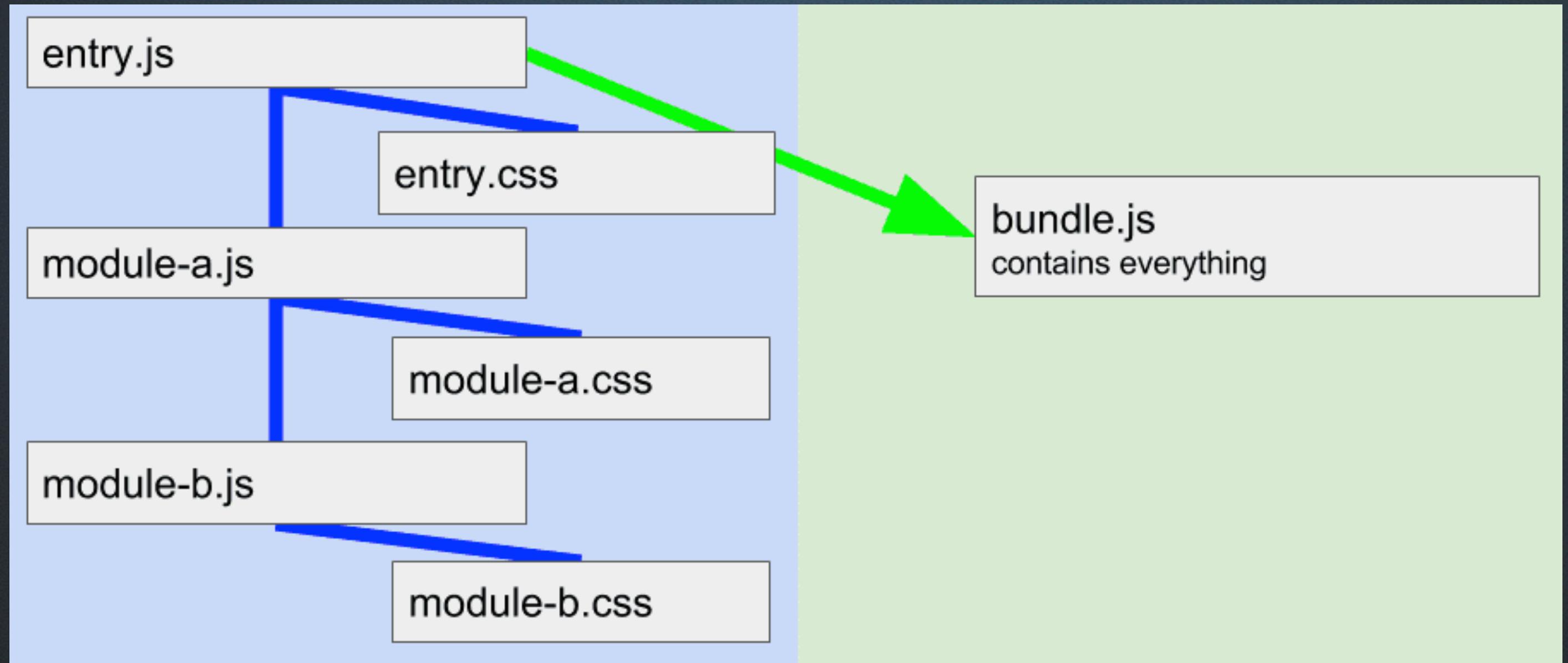
Beispiel CSS:

```
npm i css-loader style-loader -save-dev
```

```
var css = require('style!css!./foo.css');
```



4. Loader



→ siehe Code-Beispiel 2

4. Loader

Code-Beispiel 2

entry.js

```
1 var moduleA = require('./module-a.js');
2 var css = require('style!css!./entry.css');
3
4 console.log('entry.js');
5 document.write('<div class="entry">entry</div>');
```

index.html im Browser



module B
module A
entry



5. Konfigurations-Datei

Beispiel 3

webpack.config.js

```
1 var webpack = require("webpack"),
2     production = process.env.NODE_ENV === 'production';
3
4 module.exports = {
5     entry: ["./entry.js"],
6     output: {
7         filename: "bundle.js"
8     },
9     module: {
10        loaders: [
11            {
12                test: /\.css$/,
13                loaders: ['style', 'css']
14            }
15        ],
16        devtool: production ? false : 'source-map'
17    };
18}
```



5. Konfigurations-Datei

Beispiel 3

package.json

```
12  "scripts": {  
13    "dev": "NODE_ENV=development webpack",  
14    "build": "NODE_ENV=production webpack"  
15  },
```



6. Plugins

- ▶ Erweitern Funktionalitäten von Webpack
- ▶ Beispiele:
 - ▶ progress-plugin
 - ▶ webpack-angular-resource-plugin
 - ▶ html-webpack-plugin
 - ▶ uvm.

→ siehe Code-Beispiel 4: "html-webpack-plugin"



6. Plugins

Beispiel 4

webpack.config.js

```
1  var webpack = require("webpack"),
2      HtmlWebpackPlugin = require('html-webpack-plugin'),
3      production = process.env.NODE_ENV === 'production';
4
5  module.exports = {
6      entry: ["./entry.js"],
7      output: {"filename": "bundle.js"},...
10     module: {
11         loaders: [{"test": /\.css$/...}]
15     },
16     plugins: [
17         new HtmlWebpackPlugin({
18             title: 'My Demo Title',
19             filename: 'demo.html'
20         })
21     ],
22     devtool: production ? false : 'source-map'
23 };
```



6. Plugins

Beispiel 4

```
npm install html-webpack-plugin --save-dev
```

demo.html (generiert)

```
1      <!DOCTYPE html>
2      <html>
3          <head>
4              <meta charset="UTF-8">
5              <title>My Demo Title</title>
6          </head>
7          <body>
8              <script type="text/javascript" src="bundle.js"></script>
9          </body>
</html>
```

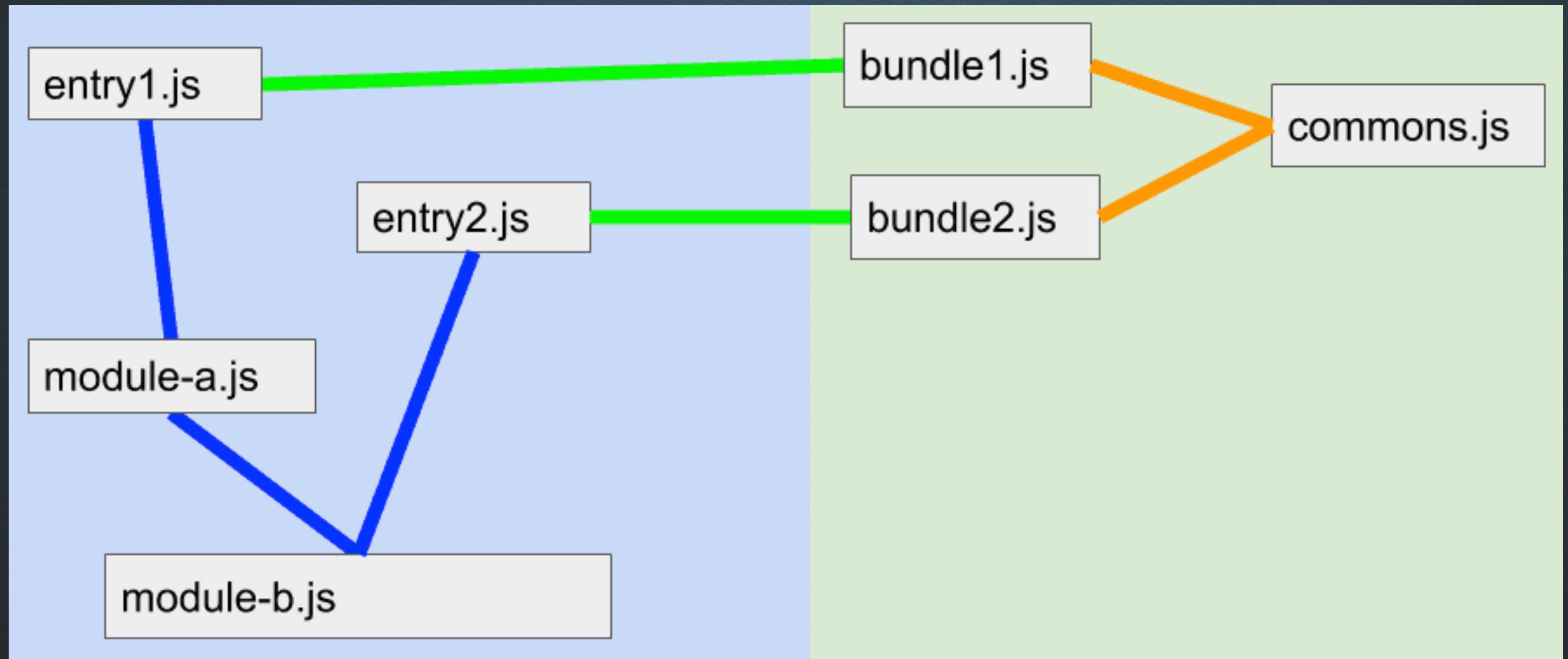


7. Code Splitting

- ▶ optionales Feature
- ▶ Modul Abhängigkeiten in „Chunks“ aufteilen
==> kürzere initiale Ladezeiten
- ▶ nicht gebrauchter Code wird nicht geladen
- ▶ CommonsChunkPlugin:
 - ▶ gleiche Module in verschiedenen Chunks werden automatisch erkannt und nur einmal geladen
 - ▶ bereits in Webpack integriert



7. Code Splitting



→ siehe Code-Beispiel 5

7. Code Splitting

Code Beispiel 5

webpack.config.js

```
4  module.exports = {
5    entry: {
6      bundle1: "./entry1.js",
7      bundle2: "./entry2.js"
8    },
9    output: {
10      path: __dirname + "/dist",
11      filename: "[name].js",
12      publicPath: "/dist/"
13    },
14    module: {....},
15    plugins: [
16      new webpack.optimize.CommonsChunkPlugin("commons", "commons.js")
17    ],
18    devtool: production ? false : 'source-map'
19  };
20
```

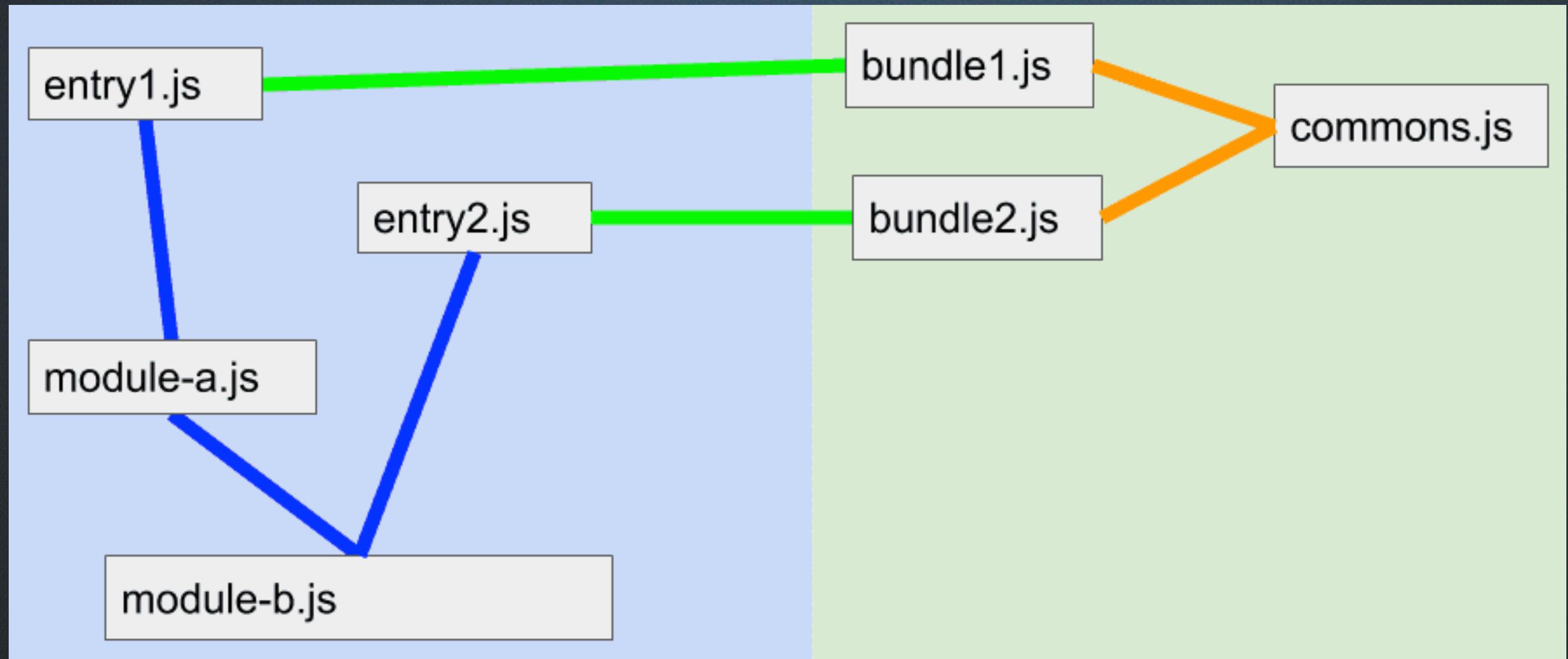


8. Hot Module Replacement

- ▶ "webpack-dev-server"
- ▶ kleiner lokaler Webserver
- ▶ erkennt Änderungen an Euren Quelldateien
 - > transformiert automatisch neu
 - > lädt Eure Webseite neu
- ▶ Übersetzt NUR geänderte Dateien neu



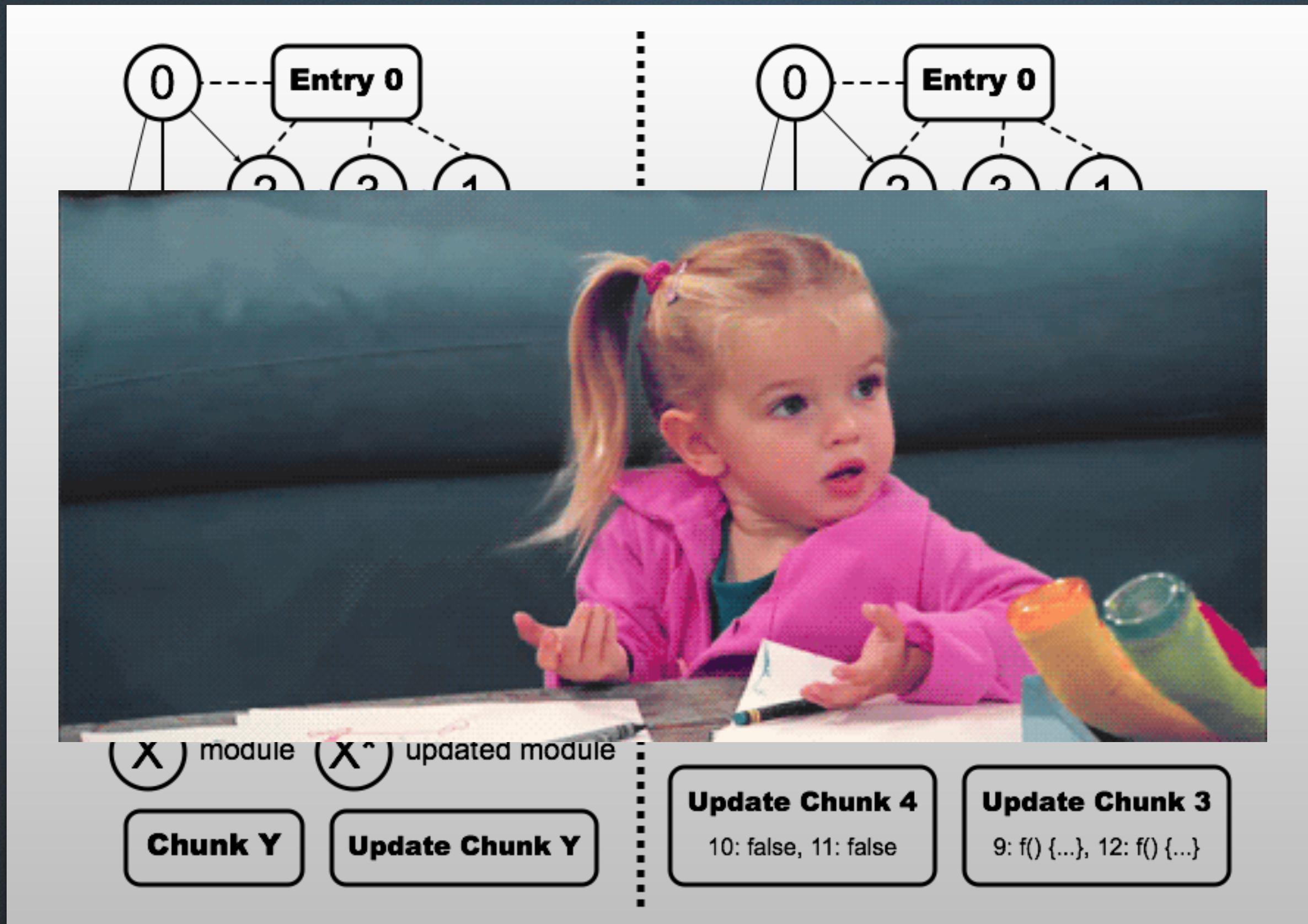
8. Hot Module Replacement



→ siehe Code-Beispiel 6



8. Hot Module Replacement



9. Zusammenfassung

Vorteile

- + geringere Ladezeiten durch Code-Splitting
- + Erweiterbarkeit
- + wachsende Community
- + schneller "Hot Mode"

webpack.github.io/docs/comparison.html

Nachteile

- nur ein "Hobby"-Hauptentwickler
- unausgereifte Dokumentation
- komplexer Einstieg
- teilweise unverständliche API



Das
war's.



Fragen?
Was nutzt ihr?



Bilderquellen

- ▶ <https://webpack.github.io/docs/what-is-webpack.html>
- ▶ <https://github.com/k9ordon/webpack-ftw>
- ▶ <http://giphy.com/gifs/KI01DytIVPEw8>
- ▶ <https://webpack.github.io/docs/hot-module-replacement-with-webpack.html>

