Hello webpack!

Daniel Mies

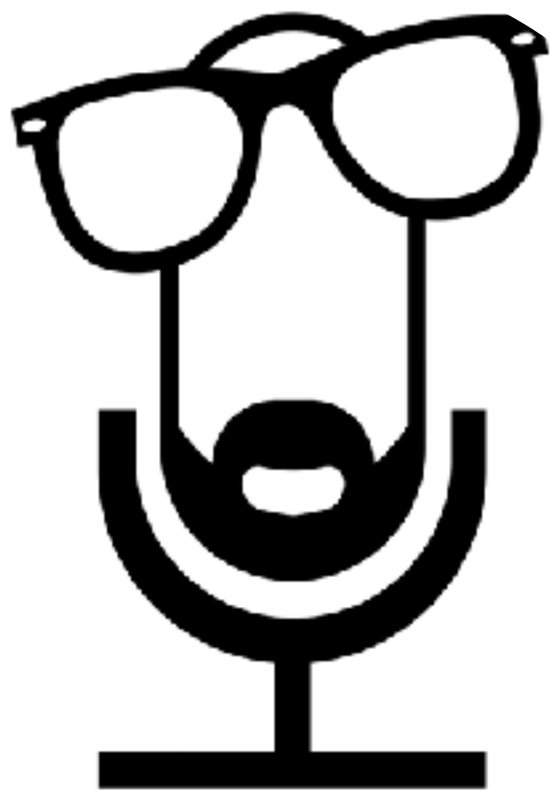
codecentric AG

About me

- Daniel Mies
- Software Craftsman
- codecentric AG
- @danielmies

Podcast: Herr Mies will's wissen

- Tech Interviews
- alle 2 Wochen
- 30 - 45 Minuten
- mies.me



Agenda

- What is webpack
- Getting started
- Modules
    - Using more than one language
    - Working with assets
- Plugins
    - better CSS handling
    - generating HTML templates
- Useful tools

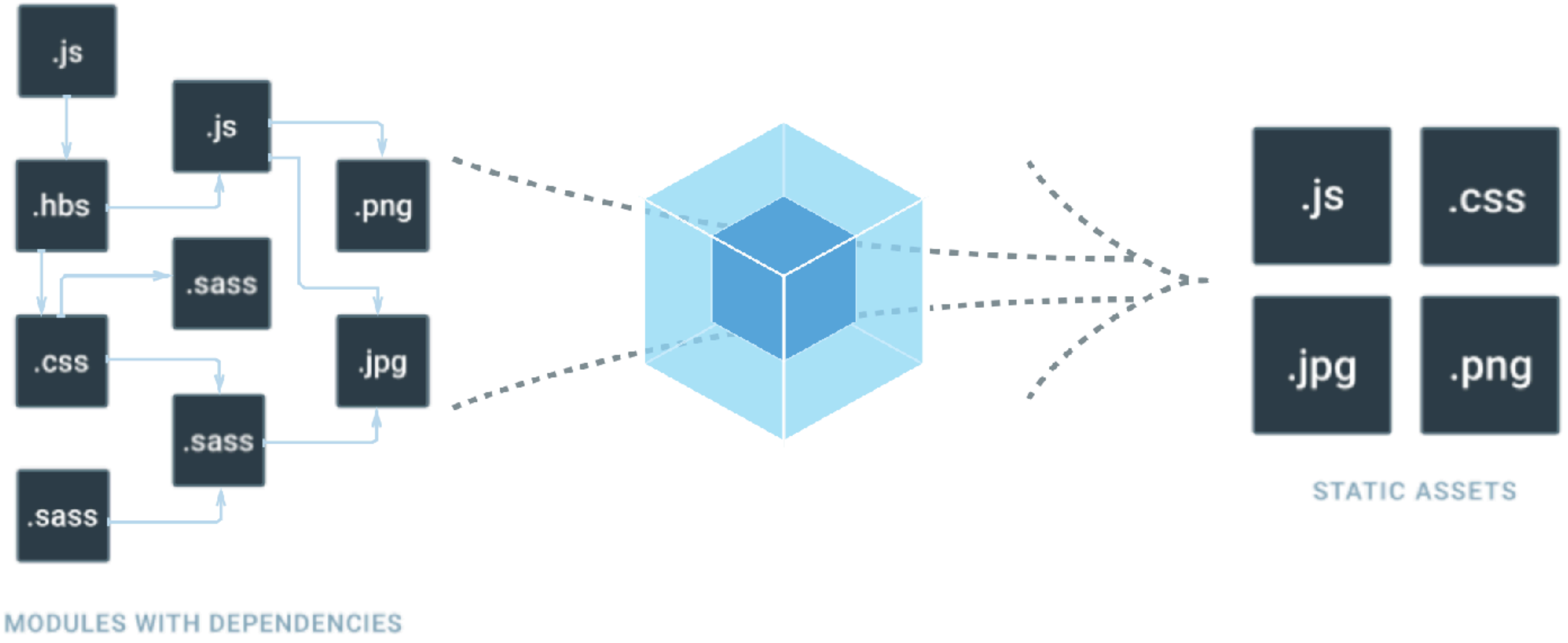
Who are you?

What is webpack?

What is webpack?

webpack is a **module bundler**. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset.

https://github.com/webpack/webpack



https://webpack.js.org

About webpack

- current version: 3.6
- official website: https://webpack.js.org
- github: https://github.com/webpack/webpack
- installation:

```
npm install --save-dev webpack
```

Getting started

Project setup

- setup a npm project:

```
npm init -y
```

- install webpack:

```
npm install --save-dev webpack
```

- install http-server to show something:

```
npm install --save-dev http-server
```

- for this sample only:

```
npm install --save-dev left-pad
```

Project setup

- setup a npm project:

```
npm init -y
```

- install webpack:

```
npm install --save-dev webpack
```

- install http-server to show something:

```
npm install --save-dev http-server
```

- for this sample only:

```
npm install --save-dev left-pad
```

webpack will bundle all your code so this library is **only needed during development** and not during runtime. Many examples do this wrong.

package.json

```
{
  "name": "01_getting-started",
  "version": "1.0.0",
  "scripts": {
    "build": "webpack",
    "start": "http-server"
  },
  "devDependencies": {
    "http-server":"^0.10.0",
    "left-pad": "^1.1.3",
    "webpack": "^3.6.0"
  }
}
```

Basic HTML

```
<!-- ./index.html -->
<html>
<head>
  <title>Hello Webworker</title>
</head>
<body>
  <script src="./dist/bundle.js"></script>
</body>
</html>
```

Our JavaScript application

```
// ./src/app.js
var leftPad = require("left-pad");

function component() {
  var title = leftPad("Hello Webworker!", 20, "+");

  var element = document.createElement("h1");
  element.innerHTML = title;

  return element;
}

document.body.appendChild(component());
```

codecentric

Our JavaScript a

We bundle left-pad here to demonstrate how node modules can be used

```
// ./src/app.js
var leftPad = require("left-pad");

function component() {
  var title = leftPad("Hello Webworker!", 20, "+");

  var element = document.createElement("h1");
  element.innerHTML = title;

  return element;
}

document.body.appendChild(component());
```

Our JavaScript a

We bundle left-pad here to demonstrate how node modules can be used

```
// ./src/app.js
var leftPad = require("left-pad");

function component() {
  var title = leftPad("Hello Webworker!", 20, "+");

  var element = document.createElement("h1");
  element.innerHTML = title;

  return element;
}

document.body.appendChild(component());
```

`component()` creates a h1 with some text and returns it

webpack.config.js

```
module.exports = {
  entry: "./src/app.js",
  output: {
    filename: "./dist/bundle.js"
  }
};
```

webpack.config.js

```
module.exports = {
  entry: "./src/app.js",
  output: {
    filename: "./dist/bundle.js"
  }
};
```

A simple way to define an entry point. You can use multiple endpoints and there are more options to define them.

webpack.config.js

```
module.exports = {
  entry: "./src/app.js",
  output: {
    filename: "./dist/bundle.js"
  }
};
```

A simple way to define an entry point. You can use multiple endpoints and there are more options to define them.

There are more ways to define the output, too. For every entry an output is generated. Start simple with one entry/outpoint.

codecentric

Demo

Entry point

Entry point

- webpack needs a file to start with

Entry point

- webpack needs a file to start with
- define one or more files, that are needed to build your application

Entry point

- webpack needs a file to start with
- define one or more files, that are needed to build your application

```
module.exports = {
  entry: "./src/app.js",
  // ...
};
```

Entry point

- webpack needs a file to start with
- define one or more files, that are needed to build your application

```
module.exports = {
  entry: "./src/app.js",
  // ...
};
```

```
module.exports = {
  entry: ["./src/app.js"],
  // ...
};
```

Entry point

- webpack needs a file to start with
- define one or more files, that are needed to build your application

```
module.exports = {
  entry: "./src/app.js",
  // ...
};
```

```
module.exports = {
  entry: ["./src/app.js"],
  // ...
};
```

```
module.exports = {
  entry: {
    app: "./src/app.js"
  },
  // ...
};
```

Output

Output

- your bundled modules

Output

- your bundled modules
- webpack generates a bundle for every entry point

Output

- your bundled modules
- webpack generates a bundle for every entry point
- there will be more generated files later (`output.path` is needed for them)

Output

- your bundled modules
- webpack generates a bundle for every entry point
- there will be more generated files later (`output.path` is needed for them)

```
const path = require("path");


module.exports = {
  // ...
  output: {
    path: path.resolve(__dirname, "dist"),
    filename: "bundle.js"
  },
  // ...
```

More languages

Adding ES6 and more

Adding ES6 and more

- webpack can be extended by loaders for ES6, TypeScript and more

Adding ES6 and more

- webpack can be extended by loaders for ES6, TypeScript and more
- projects can have loaders for multiple languages

Adding ES6 and more

- webpack can be extended by loaders for ES6, TypeScript and more
- projects can have loaders for multiple languages
- every loader must be configured

Project setup

- add babel and the babel-loader

```
npm install --save-dev babel-loader babel-core babel-preset-env
```

package.json

```
{
  "name": "02_es6",
  "version": "1.0.0",
  "scripts": {
    "build": "webpack",
    "start": "http-server"
  },
  "devDependencies": {
    "babel-core": "^6.26.0",
    "babel-loader": "^7.1.2",
    "babel-preset-env": "^1.6.0",
    "http-server":"^0.10.0",
    "webpack": "^3.6.0"
  }
}
```

Our JavaScript application

```
// component.js
export const component = (type, text) => {
  var element = document.createElement(type);
  element.innerHTML = text;

  return element;
};

// app.js
import { component } from "./component";

document.body.appendChild(component("h1", "Hello Webworker"));
```

webpack.config.js

```
module.exports = {
  // ... (like before)
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /(node_modules)/,
        use: {
          loader: "babel-loader",
          options: { presets: ["env"]}
        }
      }
    ]
  }
};
```

Demo

More languages

if we want more languages, e.g. TypeScript, we can add them via npm:

```
npm install --save-dev typescript ts-loader
```

add a **tsconfig.json** to your project to make TypeScript work (`tsc --init`)

Our JavaScript application

```
// component.ts
export const component = <K extends keyof HTMLElementTagNameMap>(
  type: K,
  text: string
) => {
  var element = document.createElement(type);
  element.innerHTML = text;

  return element;
};
// app.js
import { component } from "./component.ts";

document.body.appendChild(component("h1", "Hello Webworker"));
```

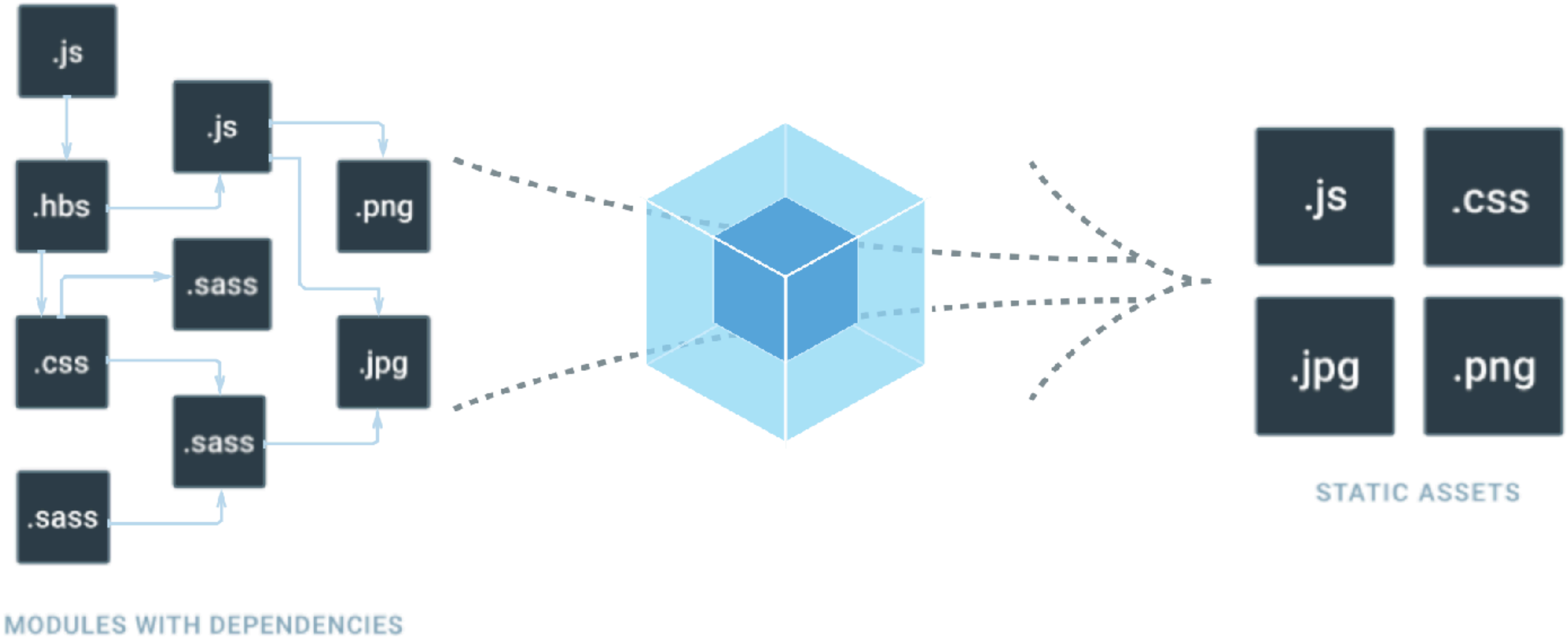
webpack.config.js

```
module.exports = {
  // ... (like before)
  module: {
    rules: [
      // ...
      {
        test: /\.ts$/,
        exclude: /(node_modules)/,
        use: {
          loader: "ts-loader",
        }
      }
    ]
  }
};
```

Demo

codecentric

Static Assets



https://webpack.js.org

Using webpack to add CSS

Using webpack to add CSS

- you are not limited to use JavaScript in your application

Using webpack to add CSS

- you are not limited to use JavaScript in your application
- webpack allows to load all kinds of assets, like CSS, SASS or images, too

Using webpack to add CSS

- you are not limited to use JavaScript in your application
- webpack allows to load all kinds of assets, like CSS, SASS or images, too
- all you need is a loader that supports your type of asset

Project setup

add some more loaders:

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

Some CSS

```
// ./src/style.css
.hello {
  color: red;
}
```

HTML

```
<!-- ./index.html -->
<html>
<head>
  <title>Hello Webworker</title>
  <!-- no CSS here! -->
</head>
<body>
  <script src="./dist/bundle.js"></script>
</body>
</html>
```

Our JavaScript application

```
// ./src/component.js
import "./style.css";

export const component = (type, text) => {
  var element = document.createElement(type);
  element.innerHTML = text;
  element.className = "hello";

  return element;
};
```

webpack.config.js

```
module.exports = {
  // ... (like before)
  module: {
    rules: [
      {
        test: /\.css$/,
        use: ["style-loader", "css-loader"]
      }
    ]
  }
};
```

webpack.config.js

```
module.exports = {
  // ... (like before)
  module: {
    rules: [
      {
        test: /\.css$/,
        use: ["style-loader", "css-loader"]
      }
    ]
  }
};
```

1

webpack.config.js

```
module.exports = {
  // ... (like before)
  module: {
    rules: [
      {
        test: /\.css$/,
        use: ["style-loader", "css-loader"]
                           2              1
      }
    ]
  }
};
```

Demo

CSS and JS loader

```
// ./src/component.js
import "./style.css";

export const component = (type, text) => {
  var element = document.createElement(type);
  element.innerHTML = text;
  element.className = "hello";

  return element;
};
```

CSS and JS loader

```
// ./src/component.js
import "./style.css";


export cons                                    > {
  var eleme                                    type);
  element.innerHTML = text;
  element.className = "hello";

  return element;
};
```

the **CSS loader** will load this file and transform it into a JS module

CSS and JS loader

```
// ./src/component.js
import "./style.css";

export cons                                           > {
  var eleme                                           type);
  element.innerHTML = te
  element.className = "h

  return element;
};
```

the **CSS loader** will load this file and transform it into a JS module

the **style loader** will add a script to the generated output which loads this css into the webpage

codecentric

And more

And more

- there is support for other languages like SASS and LESS

And more

- there is support for other languages like SASS and LESS
- you can even load other files and use them within your bundle

Plugins

Plugins

- allow to add more functionality to your build

Project setup

- start with the last sample
- add the extract-text-webpack-plugin:
  ```
  npm install --save-dev extract-text-webpack-plugin
  ```
- add the clean-webpack-plugin
  ```
  npm install --save-dev clean-webpack-plugin
  ```

HTML

```
<!-- ./index.html -->
<html>
<head>
  <title>Hello Webworker</title>
  <link rel="stylesheet" href="./dist/styles.css" />
</head>
<body>
  <script src="./dist/bundle.js"></script>
</body>
</html>
```

webpack.config.js

```
module.exports = {
  // ... (like before)
  module: {
    rules: [
      // ...
      {
        test: /\.css$/,
        use: ExtractTextPlugin.extract({
          fallback: "style-loader",
          use: "css-loader"
        })
      }
    ]
  },
  plugins: [
    new CleanWebpackPlugin(["./dist"]),
    new ExtractTextPlugin("./styles.css")
};
```

Demo

Project setup

- add the extract-text-webpack-plugin:

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

HTML Template

```
<!-- ./templates/index.ejs -->
<html>
<head>
  <title>Hello Webworker</title>
</head>
<body>
</body>
</html>
```

webpack.config.js

```
module.exports = {
  // ... (like before)
    module: {
    rules: [/* ... */]
  },
  plugins: [
    // ...
    new HtmlWebpackPlugin({
      filename: "./index.html",
      template: "./templates/index.ejs"
    })
  ]
};
```

Demo

Useful tools

source maps

- webpack can generate source maps for your application
- just configure them as **`devTool`**

webpack.config.js

```
module.exports = {
  entry: // ...
  output: // ...
  devTool: "inline-source-map",
  // ...
```

webpack-dev-server

- you can use webpack with a `--watch` flag
- or you can use webpack-dev-server which gives you a simple server with live reloading

```
npm install --save-dev webpack-dev-server
```

webpack.config.js

```
module.exports = {
  entry: // ...
  output: // ...
  devServer: {
    contentBase: "./dist"
  },
  // ...
```

webpack.config.js

```
module.exports = {
  entry: {
    app: "./src/app.js"
  },
  output: {
    path: path.resolve(__dirname, "dist"),
    filename: "[name].bundle.js?[hash]"
  },
  devServer: {
    contentBase: "./dist"
  },
  // ...
```

webpack.config.js

```
module.exports = {
  entry: {
    app: "./src/
  },
  output: {
    path: path.re    ve(__dirname, "dist"),
    filename: "[name].bundle.js?[hash]"
  },
  devServer: {
    contentBase: "./dist"
  },
  // ...
```

`[name]` is replaced by the key in entry. Here `[name]` will be replaced by `app`.

webpack.config.js

```
module.exports = {
  entry: {
    app: "./src/
  },
  output: {
    path: path.re    ve(__dirname, "dist"),
    filename: "[name].bundle.js?[hash]"
  },
  devServer: {
    contentBase: "./dist"
  },
  // ...
```

`[name]` is replaced by the key in entry. Here `[name]` will be replaced by **`app`**.

`[hash]` is generated by webpack. This helps a lot to avoid caching problems.

package.json

```
{
  "name": "07_devserver",
  "scripts": {
    "build": "webpack",
    "start": "webpack-dev-server --open"
  },
  "devDependencies": {
    //...
    "webpack": "^3.6.0",
    "webpack-dev-server": "^2.8.2"
  }
}
```

Demo

And more

What else do you need?

What else do you need?

- **webpack-dev-middleware**: if you want to use webpack within your node server

What else do you need?

- **webpack-dev-middleware**: if you want to use webpack within your node server
- a **production build** (without source-maps)

What else do you need?

- **webpack-dev-middleware**: if you want to use webpack within your node server
- a **production build** (without source-maps)
- **chunks** / **code splitting**: move libraries into single file so they get cached

What else do you need?

- **webpack-dev-middleware**: if you want to use webpack within your node server
- a **production build** (without source-maps)
- **chunks** / **code splitting**: move libraries into single file so they get cached
- more **loaders**, e.g. for images and fonts

What else do you need?

- **webpack-dev-middleware**: if you want to use webpack within your node server
- a **production build** (without source-maps)
- **chunks** / **code splitting**: move libraries into single file so they get cached
- more **loaders**, e.g. for images and fonts

this sample build is more than enough to get started

Questions?

Thanks!

codecentric.de

mies.me