

Webpack入門

Webpackを使うと何が幸せか

- JavaScriptの依存関係を解決して1ファイルにまとめてくれる
- グローバル空間に変数が出ることなくプログラムを書ける
- sassや画像などなんでも1ファイルにまとめられる

その前にNode.jsとnpmを知っておこう！

Node.js



Node.js

JavaScriptのサーバーサイド用言語

2009年に登場

npm

Node.jsのモジュールパッケージ管理ツール

今ではフロントエンドのパッケージもここで管理されている

これが流行りだす前は `bower` というTwitterが開発したツールが使われていた

npm で管理されているモジュールもインポートしてみよう

パッケージのインストール

```
npm install smartphoto --save
```


Webpackの設定

Webpackのインストール

```
npm i webpack --save-dev
```

webpack-cliのインストール

```
npm i webpack-cli --save-dev
```

package.jsonにコマンドを追加

```
"scripts": {  
  "build": "webpack"  
}
```

一度実行してみよう

```
npm run build
```

```
ERROR in Entry module not found: Error: Can't resolve './src' i
```

./src/index.jsを作成

```
console.log('hello world');
```

```
npm run build
```

/dist/main.js が生成されている。

Webpack4ではエントリーポイントと、アウトプットファイルの指定がなくてもOK

エントリーポイントを指定したい場合

```
"scripts": {  
  "build": "webpack ./src/index.js --output ./dest/bundle.js"  
}
```

developmentモードと、buildモード

```
"dev": "webpack --mode development",  
"build": "webpack --mode production"
```

npm run devを実行

実行スピードが早いですが、minifyされてなかったり最適化されていない。
素早く実行結果を確認したいときに有効

```
npm run dev
```

webpack-dev-serverの利用

webpack-dev-serverの利用

```
npm i webpack-dev-server --save-dev
```

```
"start": "webpack-dev-server --mode development --open",  
"build": "webpack --mode production"
```

```
npm run start
```

Babelの使用

- babel-core
- babel-loader
- babel-preset-env

が必要

Babelの使用

先ほどのパッケージをインストール

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

Babelの使用

.babelrcの設定

```
{  
  "presets": [  
    "env"  
  ]  
}
```


Babelの使用

webpack.config.jsの作成

```
module.exports = {  
  module: {  
    rules: [  
      {  
        test: /\.js$/,  
        exclude: /node_modules/,  
        use: {  
          loader: "babel-loader"  
        }  
      }  
    ]  
  }  
};
```

arrow functionが解決されているのがわかる

hello.js

```
export default (name) => {  
  console.log(`Hello! ${name}`);  
}
```

main.js

```
import hello from './hello';  
hello('daigo'); // hello daigo
```

Reactも使いたい

Reactも使いたい

```
npm i react react-dom --save
```

React も使いたい

```
npm i babel-preset-react --save-dev
```

.babelrc

```
{  
  "presets": ["env", "react"]  
}
```

Reactも使いたい

index.js

```
import React from "react";
import ReactDOM from "react-dom";
const App = () => {
  return (
    <div>
      <p>React here!</p>
    </div>
  );
};
export default App;
ReactDOM.render(<App />, document.getElementById("app"));
```

ソースコード

<https://github.com/steelydylan/frontend-nagoya-webpack-handson>

```
git clone git@github.com:steelydylan/frontend-nagoya-webpack-handson
npm install
```

していただければ今日の日でも環境が整います

ありがとうございました。