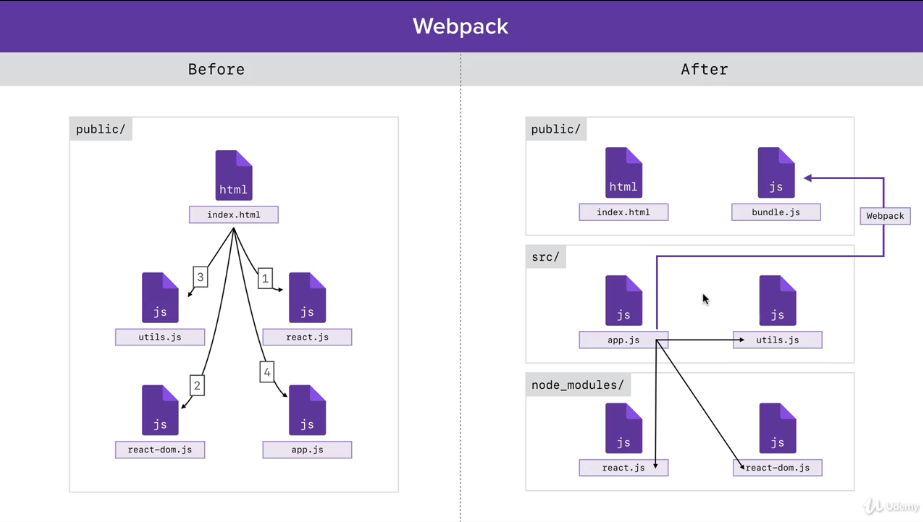
**WEBPACK**



WebPack в отличии от других пакетных менеджеров (Grunt, Gulp) может собирать файлы из разрозненных частей (через javascript module exports) в один большой файл, который при указании прогоняет через Babel, минифицирует и вставляет на фронтенд под названием bundle.js

# УСТАНОВКА WEBPCK (ЛОКАЛЬНО)

Через yarn:

yarn add webpack

yarn add live-server

yarn add webpack-dev-server

yarn add babel-core babel-loader

yarn add babel-plugin-transform-class-properties

После этого нужно в **package.json** добавить скрипт,

build - для продакшн версси, единожды запускает webpack, прогоняет все файлы через себя и babel, выливает в bundle.js

dev-server - для девелопмент версии, запускает веб-сервер, все файлы держит в оперативке

"scripts": {

"serve": "live-server public/",

"build": "webpack",

"dev-server": "webpack-dev-server"

},

# WEBPACK.CONFIG.JS

Все настройки по всем установленным модулям webpack берет из файла **webpack.config.js**.

Необходимо создать файл в корне проекта, в него пишем:

const path = require('path');

module.exports = {

entry: "./src/app.js",

output: {

path: path.join(\_\_dirname, 'public'),

filename: "bundle.js"

},

module: {

rules: [{

loader: 'babel-loader',

test: /\.js$/,

exclude: /node-modules/

}]

},

devtool: 'cheap-module-eval-source-map',

devServer: {

contentBase: path.join(\_\_dirname, 'public')

}

};

# .babelrc

После этого необходимо в корне проекта создать **.babelrc**, куда добавить пресеты babel

{

"presets": ["env", "react"],

"plugins": ["transform-class-properties"]

}

# ЗАПУСК

После этого можно запустить webpack, который возьмет файл src/app.js (из webpack.config.js) и выльет его в папку public/bundle.js. Это версия для продакшна

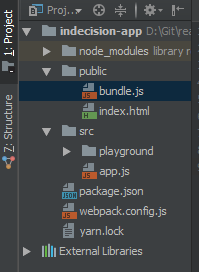
yarn run webpack

Для девелопмента, что бы запустить веб-сервер, пишем,

yarn run webpack-dev-server

# ПОДРОБНО

Структура должна выглядеть так, если рабочий основной файл - src/app.js, а итоговый файл - bundle.js:



# УСТАНОВКА DEV-SERVER

webpack имеет свой дев-сервер, который работает быстро, потому что заточен именно под webpack. А еще он не требет физического расположения конечного собранного файла типа bundle.js: сервер сконфигурирован так, что все держит в памяти.

Из-под корня проекта пишем:

yarn add webpack-dev-server

# BABEL В WEBPACK

Устанавливаем babel core и babel-loader

yarn add babel-core babel-loader

После установки в webpack.config.js необходимо их добавить в rules. Дописываем:

module: {

rules: [{

loader: 'babel-loader',

test: /\.js$/,

exclude: /node-modules/

}]

},

devtool: 'cheap-module-eval-source-map'

**loader** - модуль, который заинтсллили

**test** - какие файлы модуль должен искать? (тут все файлы, у которых окончание .js)

**exclude** - папка для исключения поиск в

**devtool** - babel-source maps для отображения ошибок помодульно. Подробнее - <https://webpack.js.org/configuration/devtool/#devtool>

После этого необходимо в корне проекта создать **.babelrc**, куда добавить пресеты и плагины babel.

{

"presets": ["env", "react"],

"plugins": ["transform-class-properties"]

}