**WEBPACK-JackCoder-tutorial**

**git init**

**yarn init**

Будем использовать препроцессор stylus к нему подключим 2 плагина:

autoprefixer

....

Одно из отличий Webpack от Gulp это наличие точки входа - index.js, в нем размещаем все подключения библиотек.

Установим:

**yarn add -D webpack webpack-cli webpack-dev-server path**

Далее в корне проекта создаем и настраиваем webpack.config.js

const path = require('path');

module.exports = {

    entry: {

        app: \_\_dirname + '/src/index.js'

    },

    output: {

        filename: '[name].js',

        path: path.resolve(\_\_dirname, './dist'),

        publicPath: '/dist' // Необходимо для dev-server

    },

    devServer: {

        overlay: true //Показ ошибок не в консоли, а в окне браузера

    }

};

Затем в корне проекта создаем index.html и подключаем к нему dist/app.js

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Webpack 4</title>

    <link rel="stylesheet" href="">

</head>

<body>

    <div class="wrapper">

        <h1>Webpack</h1>

    </div>

    <script src="/dist/app.js"></script>

</body>

</html>

Далее создаем src/index.js в нем будут подключатся все наши зависимости, затем, в качестве примера, создаем src/js/common.js

src/js/common.js:

let add = (a, b) => a + b;

console.log(add(2, 6));

И не забываем его подключить в src/index.js:

import './js/common'; // .js указывать не обязательно

**yarn dev**

Важно! в данном режиме (**--mode development**) папка /dist не создается, она хранится в памяти, это одно из кардинальных отличий webpack от gulp, за счет этого достигается его быстрота, так как ему не надо создавать папку /dist на харде, все хранится в RAM.

папка /dist будет создана только в режиме --mode production

**yarn build**

Далее для того чтобы мы могли спокойно использовать ES6 синтаксис и быть уверенными, что он запучтиться в старых браузерах нам необходимо установить Babel:

**yarn add -D @babel/core @babel/preset-env babel-loader**

Затем необходимо добавить и настроить babel в разделе module в webpack.config.js:

module: {

        rules: [

            {

                test: /\.js$/,

                loader: 'babel-loader', // Указываем через что необхожимо обрабатывать js файлы

                exclude: '/node\_modules/'

            }

        ]

    },

Далее в корне проекта надо конфиг для babel - .babelrc:

{

    "presets": [

        "@babel/preset-env"

    ]

}

**yarn build**

теперь в /dist/app.js находится транспилированный js в синтаксис ES5

Далее подключим сторонние js файлы, подключим Vue.js, для начала скачаем пакет vue:

**yarn add -D vue**

Затем покдлючим его, есть 4 способа:

*1й (самый популярный)*

импортировать его в index.js

import 'vue';

поиск начинаеется из папки /src

*2й способ:*

import Vue from 'vue';

это удобно если мы хотим взаимодействовать с переменной Vue, например:

Vue.use();

*3й способ:*

подвесить объект Vue на window, касаемо Vue более предпочтительный

window.Vue = require('vue');

*4й способ, например нам надо установить Bootstrap и нам нужны только стили или наоборот только JS*

import 'bootstrap/dist/css/bootstrap.min.css'

или если хотим взаимодействовать, то

import Bootstrap from 'bootstrap/dist/css/bootstrap.min.css'

Далее добавим в разделе module обработчик для css файлов:

{

                test: /\.css$/,

                loader: 'css-loader', // Указываем через что необхожимо обрабатывать js файлы

            }

В следующем видео сделаем правильную настройку webpack.config.js, так как использовать один файл конфигурации не совсем правильно, лучше всего использовать три файла:

1й для dev

2й для production

3й для merge, чтобы не копировать стили

cdcdd