1. **Подготовка**
2. В консоли прописываем

npm init

1. Заполняем поля следующим образом

package name: (hello-sass)

version: (1.0.0)

description: hello sass

entry point: (index.js)

test command:

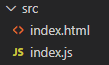
git repository:

keywords:

author: user <user@mail.ru>

license: (ISC)

1. Структура проекта (пример)



1. **Установка**
2. В консоли прописываем

npm install -D webpack webpack-cli

1. Создаём файл

*webpack.config.js*

1. **Настройка**
2. В файле webpack.config.js прописываем

const path = require('path')

module.exports = {

entry: './path/to/my/entry/file.js', //точка входа, может быть несколько точек в массиве

output: {

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

filename: 'my-first-webpack.bundle.js' //можно использовать следующую структуру

filename: '[name].[contenthash].js',

}

};

1. Для сборки используем команду

*npx webpack --config webpack.config.js*

1. Импорт и экспорт классов осуществляется следующим образом

*export default class Post …*

в основном файле .js импортируем import Post from ‘./Post’

1. Пример конфиг файла
2. const path = require('path')
3. const HTMLWebpackPlugin = require('html-webpack-plugin')
4. const {CleanWebpackPlugin} = require('clean-webpack-plugin')
5. module.exports = {
6. context: path.resolve(\_\_dirname, 'src'),
7. mode: 'development',
8. entry: {
9. main: ['@babel/polyfill', './index.js'],
10. analytics: './analytics.js'
11. },
12. output: {
13. filename: '[name].[contenthash].js',
14. path: path.resolve(\_\_dirname, 'dist')
15. },
16. plugins: [
17. new HTMLWebpackPlugin({
18. template: './index.html'
19. }),
20. new CleanWebpackPlugin()
21. ],
22. module: {
23. rules: [
24. {
25. test: /\.css$/,
26. use: ['style-loader', 'css-loader']
27. },
28. {
29. test: /\.less$/,
30. use: ['style-loader', 'css-loader', 'less-loader']
31. },
32. {
33. test: /\.s[ac]ss$/,
34. use: ['style-loader', 'css-loader', 'sass-loader']
35. },
36. {
37. test: /\.(png|jpg|svg|gif)$/,
38. use: ['file-loader']
39. },
40. {
41. test: /\.(ttf|woff|woff2|eot)$/,
42. use: ['file-loader']
43. },
44. {
45. test: /\.m?js$/,
46. exclude: /node\_modules/,
47. use: {
48. loader: 'babel-loader',
49. options: {
50. presets: ['@babel/preset-env'],
51. plugins: [
52. '@babel/plugin-proposal-class-properties'
53. ]
54. }
55. }
56. }
57. ]
58. }
59. }
60. Используем для хэширования файлов (строчка 9)

*filename: '[name].[contenthash].js'*

1. **Подключение плагинов (строчка 16)**
   1. **Плагин для HTML**
2. Установка через терминал

npm install –D html-webpack-plugin

1. Подключаем в конфиге

const HTMLWebpackPlugin = require('html-webpack-plugin')

1. Прописываем в plugins

new HTMLWebpackPlugin({

            template: './index.html'

*//*путь для файла

* 1. **Плагин для очистки папки dist**

1. Установка через терминал

*npm install –D clean-webpack-plugin*

1. Подключение в конфиге

const {CleanWebpackPlugin} = require('clean-webpack-plugin')

1. Прописываем в plugins

new CleanWebpackPlugin()

1. **Скрипты для Webpack**
2. В файле package.json добавляем новый скрипт

*"scripts": {*

*"dev": "webpack --mode development",… //*сборка проекта в режиме разработки

*"* *build ": "* *webpack --mode production ",… //*сборка проекта в режиме продакшина (минифицированные файлы)

1. Для запуска пишем

*npm run dev/npm run build*

1. Пример скриптов

"scripts": {

    "dev": "webpack --mode development",

    "build": "webpack --mode production",

    "watch": "webpack --mode development --watch"

  },

1. **Контекст**

Говорим webpack, что работаем с конкретной папкой. Указывать в конфиге её больше не нужно (не будет работать)

*context: path.resolve(\_\_dirname, 'src')*

1. **Лоадеры**

Все папки должны находиться в папке src!!!

Пример конфига с лоадерами

module: {

        rules: [

            {

                test: /\.css$/,

                use: ['style-loader', 'css-loader']

            },

            {

                test: /\.less$/,

                use: ['style-loader', 'css-loader', 'less-loader']

            },

            {

                test: /\.s[ac]ss$/,

                use: ['style-loader', 'css-loader', 'sass-loader']

            },

            {

                test: /\.(png|jpg|svg|gif)$/,

                use: ['file-loader']

            },

            {

                test: /\.(ttf|woff|woff2|eot)$/,

                use: ['file-loader']

            },

            {

                test: /\.m?js$/,

                exclude: /node\_modules/,

                use: {

                loader: 'babel-loader',

                options: {

                    presets: ['@babel/preset-env'],

                    plugins: [

                        '@babel/plugin-proposal-class-properties'

                    ]

                        }

                    }

            }

                ]

            }

В конфиге прописываем следующую структуру

module: {

rules: [

{ test: /\.txt$/, //если файл имеет след. Расширение, то используем

use: 'raw-loader' } //след. лоадеры

]

}

* 1. **Css**

{

                test: /\.css$/,

                use: ['style-loader', 'css-loader']

            },

Порядок в webpack очень важен, т.к. он идёт справа налево 🡸

1. Установка данных пакетов осуществляется с помощью команды:

*npm install –D style-loader css-loader*

1. В файле index.js импортируем

import './styles/styles.css'

* 1. **Less**

{

                test: /\.less$/,

                use: ['style-loader', 'css-loader', 'less-loader']

            },

1. Необходимо установить сам less

*npm install –D less*

1. Установка данного пакета осуществляется с помощью команды:

*npm install –D less-loader*

1. В файле index.js импортируем

import './styles/less.less'

* 1. **Sass**

{

                test: /\.s[ac]ss$/, //[ac] пишется для файлов .sass и .scss

                use: ['style-loader', 'css-loader', 'sass-loader']

            },

1. Установка данных пакетов осуществляется с помощью команды:

*npm install –D node-sass sass-loader*

1. В index.js импортируем

import './styles/scss.scss'

* 1. **Установка библиотек Css**

1. Команда

*npm install normalize.css*

1. импортируем библиотеку в файле style.css

*@import “~normalize.css”; //~ - означает, что мы переходим в nod-модули*

1. **Защита от публикации пакета**

В файле package.json вместо поля main: прописываем поле private: со значением true

"private": true

**Сжатие Css, Js, HTML**

////////////////////

1. **Babel**

[https://babeljs.io/setup#installation](https://babeljs.io/setup%23installation)

1. Для установки прописываем в консоль

*npm install --save-dev babel-loader @babel/core*

1. В конфиге прописываем

{

                test: /\.m?js$/,

                exclude: /node\_modules/,

                use: {

                loader: 'babel-loader',

                options: {

                    presets: ['@babel/preset-env'],

                    plugins: [

                        '@babel/plugin-proposal-class-properties'

                    ]

                        }

                    }

            }

1. Устанавливаем пресет для babel

*npm install –save-dev @babel/preset-env*

1. Добавляем следующую запись в файл package.json

"browserslist": "> 0.25%, not dead",

1. Подключаем полифилы. В консоли пишем

*npm install --save @babel/polyfill*

1. Модифицируем поле main:

main: ['@babel/polyfill', './index.js'],

* 1. **Добавление плагинов в babel, наример**

*npm install –D @babel/plugin-proposal-class-properties*

В конфиге добавляем поле plugins:

{

                test: /\.m?js$/,

                exclude: /node\_modules/,

                use: {

                loader: 'babel-loader',

                options: {

                    presets: ['@babel/preset-env'],

                    plugins: [ //добавляем поле

                        '@babel/plugin-proposal-class-properties'

                    ]

                        }

                    }

            }