**Настройка окружения**

Создаю файл webpack.config.js

Иницыализирую git

git push -u origin master

git checkout -b 'webpack' // создаю новую ветку

git push -u origin webpack // залить в новую ветку

git checkout master // переключиться обратно на ветку, например после pull reqesta

git pull // скачать обнавленный репозиторий в ветку в которой нахожусь

npm init

в package.json добавляю свойство "private": true

устанавливаю webpack, смотреть на офф сайте guides->geitting stsrted

npm install webpack webpack-cli --save-dev

задаю базовые настройки webpack в файле webpack.config.js

const ***path*** = require('path') //модуль возвращает строчку с абсолютным путем от корневой папки  
  
// експортирую обьект с конфигурациями  
module.exports = {  
 context: ***path***.resolve(\_\_dirname, 'src'), //указываем где лежат исходники  
 mode: 'development', // указываю режим разработки (по умолчанию это разработка)  
 entry: './index.js', // указываю входную точку  
 output: {  
 filename: 'bundle.[hash].js', // указываю название конечного файла и хеширую  
 path: ***path***.resolve(\_\_dirname, 'dist') // указываю конечную папку куда буду все складывать  
 }  
}

прописываю в package.json в scripts стартовую команду, и для продакшена

"scripts": {  
 "test": "echo \"Error: no test specified\" && exit 1",  
 "start": "webpack",  
 "build": "webpack --mode production"  
}

Запуск команда npm run start, npm run build

Устанавливаю плагины для webpack:

Для работы с html

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

для перетаскивания фавиконок

npm install copy-webpack-plugin --save-dev

чтобы выносить css в отдельный файл

npm install --save-dev mini-css-extract-plugin

чистит нужную папку

npm install --save-dev clean-webpack-plugin

результат webpack.config.js:

const ***path*** = require('path') //модуль возвращает строчку с абсолютным путем от корневой папки  
const { CleanWebpackPlugin } = require('clean-webpack-plugin') // чистит нужную папку (dist)  
const ***HTMLWebpackPlugin*** = require('html-webpack-plugin') // подключаю плагин для работы с html  
const ***CopyPlugin*** = require("copy-webpack-plugin") // для перетаскивания фавиконок  
const ***MiniCssExtractPlugin*** = require('mini-css-extract-plugin') // чтобы выносить css в отдельный файл  
  
// експортирую обьект с конфигурациями  
module.exports = {  
 context: ***path***.resolve(\_\_dirname, 'src'), //указываем где лежат исходники  
 mode: 'development', // указываю режим разработки (по умолчанию это разработка)  
 entry: './index.js', // указываю входную точку  
 output: {  
 filename: 'bundle.[hash].js', // указываю название конечного файла и хеширую  
 path: ***path***.resolve(\_\_dirname, 'dist') // указываю конечную папку куда буду все складывать  
 },  
 resolve: {  
 extensions: ['.js'], // какие файлы грузить по умолчанию  
 alias: {  
 '@': ***path***.resolve(\_\_dirname, 'src'),  
 '@core': ***path***.resolve(\_\_dirname, 'src/core')  
 } // alias - сокращения для путей к папкам и файлам  
 },  
 plugins: [ // набор плагинов  
 new CleanWebpackPlugin(),  
 new ***HTMLWebpackPlugin***({  
 template: 'index.html'  
 }), // указываю откуда брать шаблон для html  
 new ***CopyPlugin*** ({  
 patterns: [  
 {  
 from: ***path***.resolve(\_\_dirname, 'src/favicon.ico'), // откуда  
 to: ***path***.resolve(\_\_dirname, 'dist') // куда  
 }  
 ]  
}), // чтобы переносить фавикон

new ***MiniCssExtractPlugin***({  
 filename: 'bundle.[hash].css'  
 })  
 ]  
}

устанавливаю лоадеры:

преаброзовывает scss в js

npm install sass-loader sass --save-dev

преаброзовывает css в js

npm install --save-dev css-loader

babel лоадер

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

{ // babel лоадер  
 test: /\.m?js$/,  
 exclude: /node\_modules/,  
 use: {  
 loader: "babel-loader",  
 options: {  
 presets: ['@babel/preset-env']  
 }  
 }  
}

Также устанавливаю npm install @babel/preset-env --save-dev

Результат :

const ***path*** = require('path') //модуль возвращает строчку с абсолютным путем от корневой папки  
const { CleanWebpackPlugin } = require('clean-webpack-plugin') // чистит нужную папку (dist)  
const HTMLWebpackPlugin = require('html-webpack-plugin') // подключаю плагин для работы с html  
const ***CopyPlugin*** = require("copy-webpack-plugin") // для перетаскивания фавиконок  
const ***MiniCssExtractPlugin*** = require('mini-css-extract-plugin') // чтобы выносить css в отдельный файл  
  
// експортирую обьект с конфигурациями  
module.exports = {  
 context: ***path***.resolve(\_\_dirname, 'src'), //указываем где лежат исходники  
 mode: 'development', // указываю режим разработки (по умолчанию это разработка)  
 entry: './index.js', // указываю входную точку  
 output: {  
 filename: 'bundle.[hash].js', // указываю название конечного файла и хеширую  
 path: ***path***.resolve(\_\_dirname, 'dist') // указываю конечную папку куда буду все складывать  
 },  
 resolve: {  
 extensions: ['.js'], // какие файлы грузить по умолчанию  
 alias: {  
 '@': ***path***.resolve(\_\_dirname, 'src'),  
 '@core': ***path***.resolve(\_\_dirname, 'src/core')  
 } // alias - сокращения для путей к папкам и файлам  
 },  
 plugins: [ // набор плагинов  
 new CleanWebpackPlugin(),  
 new HTMLWebpackPlugin({  
 template: 'index.html'  
 }), // указываю откуда брать шаблон для html  
 new ***CopyPlugin***({  
 patterns: [  
 {  
 from: ***path***.resolve(\_\_dirname, 'src/favicon.ico'), // откуда  
 to: ***path***.resolve(\_\_dirname, 'dist') // куда  
 }  
 ]  
 }), // чтобы переносить фавикон  
 new ***MiniCssExtractPlugin***({  
 filename: 'bundle.[hash].css'  
 })  
 ],  
 module: {  
 rules: [ // лоадеры  
 { // css лоадер  
 test: /\.s[ac]ss$/i, // тестируем либо sass либо scss  
 use: [  
 ***MiniCssExtractPlugin***.loader, // переменная лоадер плагина MiniCssExtractPlugin  
 "css-loader", // потом через css лоадер  
 "sass-loader", // сначала пропускаеться через sass лоадер  
 ],  
 },  
 { // babel лоадер  
 test: /\.m?js$/, // пропускаем js через babel лоадер  
 exclude: /node\_modules/,  
 use: {  
 loader: "babel-loader",  
 options: {  
 presets: ['@babel/preset-env']  
 }  
 }  
 }  
 ],  
 }  
}

устанавливаю пакет sudo npm i -D crose-env –

определяет в какой OS мы работаем и правильно задает системную переменную :

***process***.env.NODE\_ENV

в package.json в start и build указываю какое значение задать переменной NODE\_ENV

в зависимости от режиме сборки

"start": "cross-env NODE\_ENV=development webpack",  
"build": "cross-env NODE\_ENV=production webpack --mode production"

Добавляю soucemap

devtool: isDev ? 'source-map' : false, // добавляет карту кода если мы в режиме рвазработки

Устанавливаю dev-server

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

добавляю в package.json

"start": "cross-env NODE\_ENV=development webpack-dev-server --open",

Добавляю в webpack.config.js

devServer: {  
 port: 3000,  
 hot:isDev  
},

Добавляю поливил babel npm install --save @babel/polyfill

Добавляю полифил к входной точке

entry: ['@babel/polyfill', './index.js'], *// указываю входную точку*

Устанавливаю пакеты eslint

npm i eslint eslint-loader babel-eslint

Устанавливаю конфигурацию eslint от Google

npm install --save-dev eslint eslint-config-google

**Верстка**

Устанавливаю normalize.css [npm](https://www.npmjs.org/package/normalize.css/) install normalize.css

Подключаю normalize в index.scss через тильду

@import "~normalize.css";