WEBPACK

1. *Начало работы*

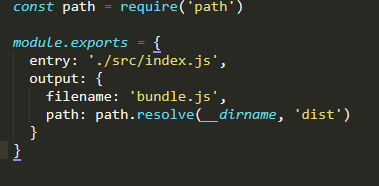
## 1. Первым делом для того, чтобы начать работать с Webpack, надо инициализировать файл package.JSON. Для этого потребуется ввести команду в консоль: npm init

## 2. Если webpack не установлен, то установить его:

## npm I –save-dev webpack , npm I –save-dev webpack-CLI

## 3. Создать папку src/index.js

## 4. Создать следующий файл в корне: webpack.config.js. В нём же прописать следующие команды:



## Entry –указывает входную точку,

## Output – указывает:

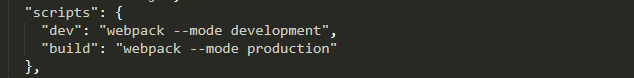
## filename – как будет называться конечный файл,

## path –путь по которому будет лежать конечный файл

2. *Модули и оптимизация сборки*

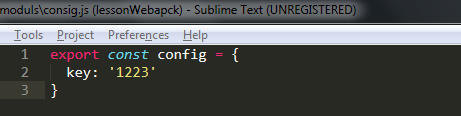
## В webpack существуют два mod`a – это production mod и development mode. Первый предназначен для сборки итогового файла к работе на хостинге, а второй для разработки проекта.

## Для того чтобы использовать эти моды, необходимо в файле package.JSON указать их в разделе scripts.

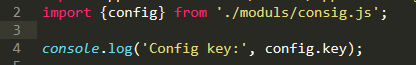


## Виды импорта в js:

## 1-ый вариант:



./src/moduls/config.js

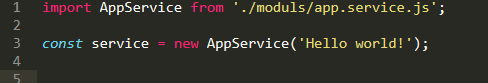


./src/index.js

## 2-ой вариант:



./src/modus/appservice.js



./src/index.js

## 3-ий вариант:



./src/moduls/header-componenets.js

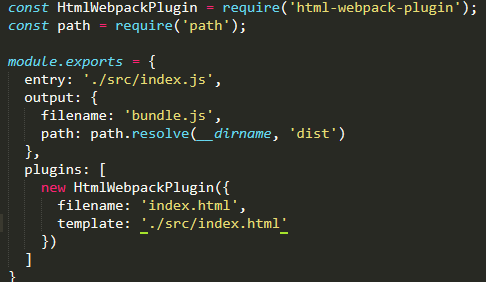


./src/index.js

3*. Плагины и создание html-файла*

## Плагины в webpack`е это такие дополнительные классы, с помощью которых мы можем добавлять функционал, который захотим.

## HtmlWebpackPlugin – плагин, с помощью которого, webpack самостоятельно подключает bundle.js в итоговый html-файл.



## Loader — это инструмент, который определенным образом преобразует исходный код модулей. Лоадер позволяет преобразовывать файлы, при их импорте или «загрузке».

## *! Loader`ы описываются в блоке модулей*

## css-loader – нужен для работы с css файлами. Позволяет только загружать и делать импорты css файлов, а для применения стилей нужен style-loader , но засовывает сами стили в html-файл.

## MiniCssExtractPlugin - Этот плагин извлекает CSS в отдельные файлы.

## [optimize-css-assets-webpack-plugin](https://github.com/NMFR/optimize-css-assets-webpack-plugin) – сжимает css;

## ! при его установке, может не работать uglifyjs-webpack-plugin, так как переопределяет поле optimization в webpack.config.js.

## Как искать решение ошибки в интернете:

## 

## копирую Unexpected token:

## подставляю название плагина uglifyjs-webpack-plugin Unexpected token

## ! для того чтобы uglifyjs-webpack-plugin мог работать надо подключить babel-loader для работы webpack с ES6 синтаксисом.

4*. Работа с библиотеками*

## 1. npm I –save jquery

## 2. импортировать библиотеку import $ from 'jquery' (куда захочу)

5*. Webpack Dev Server*

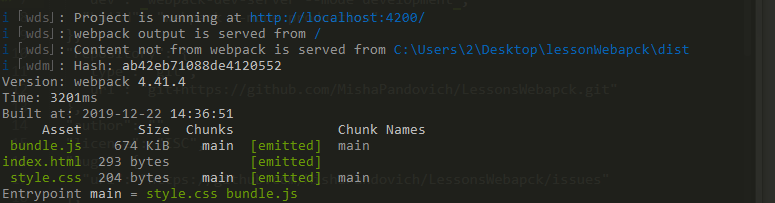
## 1. npm I –save-dev webpack-dev-server

## 2. в файле webpack.config.js указать в поле devServer следующие параметры:



## потом прописать в package.json команду для запуска этого режима





1. *Настройка Babel*

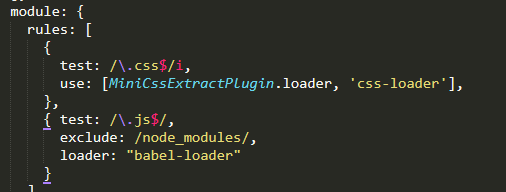
## Устаналвиваем три пакета и полифилл

npm install --save-dev @babel/core @babel/cli @babel/preset-env

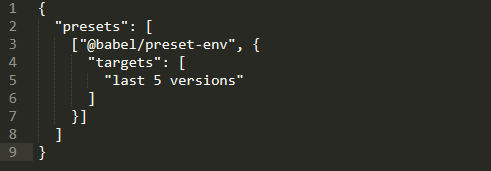
npm install --save @babel/polyfill

## Для работы с плагином необходимо установить babel-loader:

npm i --save-dev babel-loader



## Нужно создать файл .babelrc в корне проекта



## !Можно использовать uglifyjs

1. *Настройка SASS*

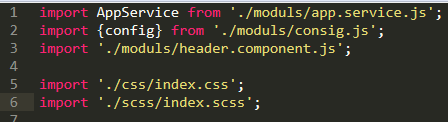
## Устанваливаем два пакета:

## npm I sass-loader node-sass –save-dev

## Прописываем правила в webpack.config.js

## 

## Подключить index.scss в index.js



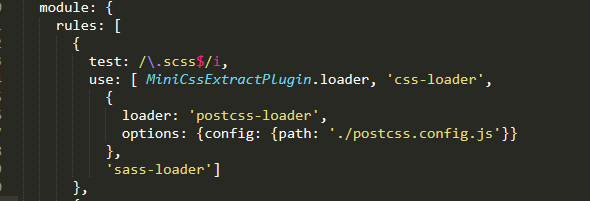
1. *Установка Autoprefixer*

## Прежде всего для работы c Autoprefixer потребуется установить postcss-loader

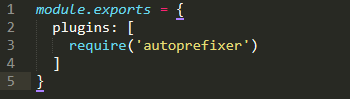
## npm i –save-dev postcss-loader

## npm I –save-dev autoprefixer

1. В файле Webpack.config.js в разделе модуля необходимо изменить правила следующим образом:



1. Следующий шаг это создать файл postcss.config.js со следующим содержимым:



1. В файле package.JSON указать до каких версий браузера, использовать префиксы:

