Старт и точки входа

npm init -y

npm i -D webpack-cli@5.1.4 [webpack-dev-server@4.15.1](mailto:webpack-dev-server@4.15.1)

Создаем папку src в ней любой index.js

Далее создаем фаил webpack.config.js и в нем указываем точки входа и выхода.

Режим mode: ‘production’ or ‘development’ – в зависимости от режима собирается билд (для прода максимально сжатый)

Так же в скрипты добавляем 2 команды - для сборки в режиме разработки и в режиме прода

Скрипты в package.json

В скрипт при вызове вебпака можно передавать настройки переменных в виде

webpack –parament value

   "build:dev": "webpack –mode development",

ENV

В конфиг вебпака при передаче env нужно что бы module.exports возвращал не обьект конфига, а функцию которая принимает аргументы (в том числе и еnv) и возвращает обьект конфига, где уже можно обращаться к еnv как env.paramenr

Планины

Html-webpack-plugin – для того что бы скрипты из точки входа подтягивались в тег скрипт фаила index.html

npm i -D [html-webpack-plugin@5.5.3](mailto:html-webpack-plugin@5.5.3)

(в репозитории описание)

Лоадеры

Это определенная цепочка обработчиков, через которые проходят фаила и выполняется какая то обработка кода. Очень важен порядок в лоадерах (работает с конца в начало)

Добавление тайпскрипта

https://webpack.js.org/guides/typescript/

npm install --save-dev typescript@9.5.0 ts-loader@5.2.2

Добавляем ts.config (из ссылки)

Добавляем блок module (лоадеры) и resolve(упрощения при импортах) (из ссылки)

Перепишем сам вебпак на ts

https://webpack.js.org/configuration/configuration-languages/#typescript

npm install --save-dev typescript ts-node @types/node @types/webpack

Исправляем импорты в вебпак конфиге, заменяем module.exports на export default, добавить несколько параметров в tsconfig

Dev Server, Watch режим, Source map

1 Dev server позволяет в режиме разработки при изменении кода перезапускать сборку (что бы не перезапускать сборку каждый раз)

npm i –D webpack-dev-server

Добавляем в конфиг вебпака

import type { Configuration as DevServerConfiguration } from "webpack-dev-server";

    devServer: {

      static: path.resolve(\_\_dirname, "build"),

      port: 3000,

      hot: true,

    },

добавляем скрипт

"start": "webpack serve --env mode=development",

Так же можно прокидывать через env порт на котором будет открываться приложение (к коде есть)

2 Source map - инструмент для того что бы при возникновении ошибки в собранном бандле показывать ошибку как бы в исходном коде. В вебпак конфиг необходимо добавить

devtool: "inline-source-map",

3. Подключение плагинов и всего остального по условию .

При подключении чего либо по условию в зависимости от mode – обязательно во втором случае возвращать undefined (так как оператор && || возвращает false а вебпак с ним не работает )

Пример

const isDev = env.mode === ‘development’

devtool: isDev ? "inline-source-map" : undefined,

REACT

Создаем стандартный для реакта index.tsx. Создаем App.tsx – импортируем его в index. И в вебпак конфиге меняем во входной точне index.ts => index.tsx

Для того что бы каждый раз не импортировать при создании компонент React, в tsconfig добавляем

"jsx": "react-jsx",

Настройка стилей CSS, SCSS

Поключаем в наш App.tsx стили из созданного App.css

Доустанавливаем

Npm I –D style-loader css-loader

Добавлем лоадеры в конфиг

{

          test: /\.css$/i,

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

        },

Для работы с scss добавляем

npm I –D sass sass-loader