**웹팩 기본**

-프로젝트의 구조를 분석하고 자바스크립트 모듈을 비롯한 관련 리소스들을 찾은 다음,

이를 브라우저에서 이용할 수 있는 번들로 묶고 패킹하는 모듈 번들러

-프로젝트 전체를 한 단위로 분석함

-> 지정된 메인 파일에서 시작해 자바스크립트의 require문과 import 문을 참고해

프로젝트의 모든 의존성을 조사하고 로더를 이용해 처리한 후 번들로 묶은

자바 스크립트 파일을 생성함

-npm으로 설치할 수 있음.

**웹팩 적용하기**

npm init //새로운 폴더를 만들고 그 안에다가 초기화

//package.json 이 생기면 다음과 같이 수정

//수정 전

{

"name": "exweb",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

//이 부분은 웹팩 빌드하는 방식 중에 하나를 쓰기위해서 아래와 같이 수정할 것이다.

},

"author": "",

"license": "ISC"

}

//수정 후

{

"name": "exweb",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

// 이안에 있는 내용을 실행하려면 npm run script값을 입력해야 실행이 된다.

"build": "webpack" //ex) npm run build

},

"author": "",

"license": "ISC"

}

//리액트를 쓰기위해app.js 폴더 안에 다음과 같은 index.html을 하나 만들자

<!DOCTYPE html>

<html lang=”en”>

<head>

//웹팩을 깔면 헤더에 <script 바벨, 리액트, 리액돔> 불러운거 안써도 됨.

//대신 바디부분에 app.js를 써줘야함.

//이걸 CRA가 대신 해주는데 한번 직접 세팅해보자.

<meta charset=”UTF-8”>

<meta charset=”viewport” content=”width=device-width, initial-scale=1.0”>

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

<title>Document</title>

</head>

<body>

<div id=”root”></div>

<script src=”./app,js”></script>

</body>

</html>

//명령어로 react와 react-dom 설치

npm i react react-dom //i는 install, 띄어쓰기로 두개 한번에 설치

//설치를 하면 package.json 파일에 dependencies 부분이 추가됨.

//웹팩 설치, 웹팩과 웹팩-cli을 같이 설치해야함.

npm i -D webpack webpack-cli //-D는 개발용으로만 쓴다는 것. --save-dev 도 가능

//이후 package.json에 devDependencies가 추가됨.

//root 부분에 webpack.config.js 파일을 만든다.

module.exports = {

//0. 별로 중요하지 않음.

name : ‘아무거나’

mode : ‘development’, //실서비스면 product

devtool : ‘eval’,

resolve : {

extension :[‘.js’, ‘.jsx’],

},

//1~3번 중요

//1. 합칠파일 입력

entry:{ // client에서 WordRelay 불러오니까 client만 불러와도 됨,

app :[‘./client’],

//확장자는 안써도 되는데 안쓰려면 위에 resolve안에 extension 쓰고

//확장자 해주면 웹팩이 찾아서 연결함.

},

//2. 어느 모듈에 적용해줄지 정함

module : {

rules : [{ //규칙적용

test: /\.jsx?/,

loader : ‘babel-loader’,

options : {

presets :[‘@bable/preset-env’, ‘@babel/preset-react’],

}

}]

},

//3. 합칠 파일 출력

output : {

//첫번째 인자, \_\_dirname은 현재 디렉토리, 두번째는 그안에 폴더를 말함

path : path.join(\_\_dirname, ‘dist’),

finame : ‘app.js’

}

}

//JSX 파일을 JS로 돌아가게 해주는 것은 바벨이고, 바벨에서도 JSX를 설정해주어야함.

//바벨 설치

// @babel/core : 바벨코어, @babel/preset-env : 우리 브라우저의 최신문법지원

// @babel/preset-react : JSX 같은 걸 지원할 수 잇음, babel-loader : 바벨과 웹팩 연결

npm i -D @babel/core @babel/preset-env @babel/preset-react babel-loader

//설치하면 package.json의 devDependencies에 내용추가됨.

//빌드의 두가지 방법 : npm run build / npx webpack

npx webpack을 하면 에러가 가는데

npm i -D @babel/plugin-proposal-class-properties 을 하고 하자

//webpack.config.js 파일의 module안에 presets를 다음처럼 옵션 설정 가능함

presets : [

[‘@babel/preset-end’, {

targets : {

browsers: [‘last 2 chrome versions’].

},

}],

// <https://github.com/browserslist/browserslist> 에서 세부옵션 볼 수 있다.

바벨

-ES6 문법을 ES5로 변환해주는 트렌스 파일러. React를 구형 브라우저에서 실행시킬 수 있다.