Wepack Cheat Sheet

1. Start

Npm init -y => Create json file.

Download webpack

Create webpack config file

Entry

Path

“start”: “webpack”

2. Loaders

Install babel loader, core, preset env

Infile config => module (search babel loader webpack) npm install -D babel-loader @babel/core @babel/preset-env

Module: {

Rules: [

{

Use: ‘babel-loader’,

Test: ‘/\.js$/’

}

]

}

.babelrc file (<https://babeljs.io/docs/en/babel-preset-env>)

{ "presets": ["@babel/preset-env"] }

"corejs": 3

Extra step: Convert module ES2015

Note: Use ‘-w’ in package.json file for auto build

(Core-js and Regenerator-runtime)

(Regenerator-runtime is not defined)

// yarn add @babel/plugin-transform-runtime --dev

// yarn add @babel/runtime

//.babelrc

"plugins": [

["@babel/plugin-transform-runtime",

{

"regenerator": true

}

]

],

3. CSS

Note: Sass loader: (https://github.com/webpack-contrib/sass-loader)

Link image for test purpose (https://www.sccnewsbyte.co.uk/wp-content/uploads/2020/08/securitythumbnail400x400.png

)

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

Create Css file and import Css file

Install Css loader and Style loader

Add rule to module rules (Remember right to left rule)

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

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

Extract-text-webpack-plugin

Import ExtractTextPlugin => MiniCssExtractPlugin

+ CSS loader

Change module (rule => loader) + Plugins

/////////////////////////////// Handling images

Install image-webpack-loader and url loader

Config file Use url-loader with limit 40000 and image-webpack-loader

Create assets

Big image, small image and import into image.js file

The difference between handling 2 images => out => publicPath => ‘dist/’

You need to declare publicPath because on some browsers this function does not support

{

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

use: [

{

loader: "url-loader",

options: {

limit: 40000,

},

},

"image-webpack-loader",

],

},

////////////////////////// Performance

=>> Code splitting

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

Code spliting for performance

1. =>> webpack.config.js =>> entry =>> object

const VENDOR\_LIBS = [

"faker",

"lodash",

"react",

"react-dom",

"react-input-range",

"react-redux",

"react-router",

"redux",

"redux-form",

"redux-thunk",

];

entry: {

bundle: "./src/index.js",

vendor: VENDOR\_LIBS,

},

output: {

path: path.join(\_\_dirname, "dist"),

filename: "[name].js",

},

=>> Create tow files bundle.js and vendor.js (but they are about the same size)

Note: Automatically VENDOR\_LIBS

const { dependencies } = require("./package.json");

const VENDOR\_LIBS = [];

*Object*.entries(dependencies).forEach((*entry*) => VENDOR\_LIBS.push(*entry*[0]));

Note: Another way

1. const { dependencies } = require('./package.json');
2. const libs = Object.keys(dependencies);

2. Solve the problem

<https://www.udemy.com/course/webpack-2-the-complete-developers-guide/learn/lecture/6296296#questions>

plugins: [

**new** webpack.*optimize*.CommonsChunkPlugin({

name: "vendor",

}),

],

3. Okay, bundle.js and vendor.js are now okay but need to add html-webpack-plugin

4. Manifest file

output: {

path: path.join(\_\_dirname, "dist"),

filename: "[name].[chunkhash].js",

},

plugins: [

**new** webpack.*optimize*.CommonsChunkPlugin({

names: ["vendor", "manifest"],

}),

],

Finally: npm install rimraf –save-dev

Add to Package.json file: “clean”: “rimraf dist”

=>> “build”: “npm run clean && webpack”

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

Webpack-dev-server

Node API

=>> Make webpack-dev-server work like a traditional sever (fetch data,…)

1. npm install webpack-dev-server –save-dev

2. “serve”: “webpack-dev-server”

3. npm run serve =>> ckeck

4. Change index.js file. See that webpack-dev-ser still watch changes in project but not rebuild project.

5. Webpack-dev-sever for development purpose cause it does not build project again.

=>> Check with hot module reloading at F8 tutorial page.

=>> Figure out webpack-dashboard

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

Webpack F8 => React (https://fullstack.edu.vn/blog/phan-1-tao-du-an-reactjs-voi-webpack-va-babel.html)

1. Npm init

2. Webpack and Webpack-cli => save dev

(--save-dev => This is called flag)

3. React and React-dom => dev

4. npm install @babel/core babel-loader @babel/preset-env @babel/preset-react --save-dev

5. public/index.html and add <div id=”root”></div> to index.html

6. src/index.js

import React from 'react' // nạp thư viện react

import ReactDOM from 'react-dom' // nạp thư viện react-dom

// Tạo component App

function App() {

return (

<div>

<h1>Xin chào anh em F8!</h1>

</div>

)

}

// Render component App vào #root element

ReactDOM.render(<App />, document.getElementById('root'))

7. npm install css-loader style-loader --save-dev

8. Create webpack.config.js

const path = require("path");

module.exports = {

entry: "./src/index.js", // Dẫn tới file index.js ta đã tạo

output: {

path: path.join(\_\_dirname, "/build"), // Thư mục chứa file được build ra

filename: "bundle.js" // Tên file được build ra

},

module: {

rules: [

{

test: /\.js$/, // Sẽ sử dụng babel-loader cho những file .js

exclude: /node\_modules/, // Loại trừ thư mục node\_modules

use: ["babel-loader"]

},

{

test: /\.css$/, // Sử dụng style-loader, css-loader cho file .css

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

}

]

},

// Chứa các plugins sẽ cài đặt trong tương lai

plugins: [

]

};

9. .babelrc

{

"presets": [

"@babel/preset-env",

"@babel/preset-react"

]

}

10. package.json

"scripts": {

...

"start": "webpack --mode development --watch",

"build": "webpack --mode production"

}

11. npm start

12. Link js to html => test purpose

Then, delete link

Plugin automatically link js to html

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

...

const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {

...

// Chứa các plugins sẽ cài đặt trong tương lai

plugins: [

new HtmlWebpackPlugin({

template: "./public/index.html"

})

]

};

14. npm install webpack-dev-server --save-dev

"scripts": {

...

"start": "webpack-dev-server --mode development --open --hot",

...

}

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

Core-js/stable and regenerator-runtime