**LearnWebpack**

Let's learn what is webpack and the basic of it.

**Tutorials**

**Getting-Started**

1. Install webpack global

npm i webpack -g

1. create a package json file

npm init -y

1. create index.js & index.html

<!-- index.html -->

<html>

<head>

<title>webpack 2 demo</title>

<script src="https://unpkg.com/lodash@4.16.6"></script>

</head>

<body>

<script src="app/index.js"></script>

</body>

</html>

// index.js

function component () {

var element = document.createElement('div');

/\* lodash is required for the next line to work \*/

element.innerHTML = \_.join(['Hello','webpack'], ' ');

return element;

}

document.body.appendChild(component());

1. install lodash library using this command

npm i lodash --save

1. add the contents below into the file

// index.js

// import & export is ES6 that doesn't work in the browser

// but webpack would replace them with compatible wrapper code

+ import \_ from 'lodash';

- <script src="https://unpkg.com/lodash@4.16.6"></script>

- <script src="app/index.js"></script>

+ <script src="dist/bundle.js"></script>

1. run this command below and start the index.html. You will see this result on the web page.

webpack app/index.js dist/bundle.js

|  |
| --- |
| \* Webpack 4 변경으로 인한 문제  webpack 명령어 시 에러 발생  $ webpack app/index.js dist/bundle.js  해결 방법 1 => **app** 폴더 이름 변경 -> **src**  해결 방법 2 => $ webpack app/index.js **--output** dist/bundle.js **--mode development** |

1. Let's add config file for more complex configuration

**추가** :

**$**  npm I webpack –save-dev

**$** npm i -D extract-text-webpack-plugin@next

// webpack.config.js

// `webpack` command will pick up this config setup by default

var path = require('path');

module.exports = {

entry: './**src**/index.js',

output: {

filename: 'bundle.js',

path: path.resolve(\_\_dirname, 'dist')**,**

**mode: 'production'**

}

};

**Example 1 - CSS Code Splitting**

* As for CSS files, use css-loaderfor default setting. The extra option ExtractTextWebpackPlugin is available for better performance

npm i css-loader style-loader --save-dev

npm i extract-text-webpack-plugin --save-dev

1. Create a new package.json

npm init -y

1. Install the necessary loaders and plugins using the commands above
2. Add index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>CSS & Libraries Code Splitting</title>

</head>

<body>

<header>

<h3>CSS Code Splitting</h3>

</header>

<div>

<p>

This text should be colored with blue after injecting CSS bundle

</p>

</div>

<script src="dist/bundle.js"></script>

</body>

</html>

1. Add base.css

p {

color : blue;

}

1. Add app/index.js

import '../base.css';

1. Add webpack.config.js

var path = require('path');

module.exports = {

entry: './app/index.js',

output: {

filename: 'bundle.js',

path: path.resolve(\_\_dirname, 'dist')

},

module: {

rules: [{

test: /**\.**css$/,

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

}]

},

}

1. Add ExtractPlugin to exract the bundled css filename

// webpack.config.js

// ...

var ExtractTextPlugin = require('extract-text-webpack-plugin');

// ...

{

// ...

module: {

rules: [{

test: /**\.**css$/,

// Comment this out to load ExtractTextPlugin

// use: ['style-loader', 'css-loader']

use: ExtractTextPlugin.extract({

fallback: "style-loader",

use: "css-loader"

})

}]

},

plugins: [

new ExtractTextPlugin('styles.css')

]

}

// index.html

// ...

<head>

<meta charset="utf-8">

<title>CSS & Libraries Code Splitting</title>

<link rel="stylesheet" href="dist/styles.css">

</head>

**Example 2 - Libraries Code Splitting**

* When using a couple of libraries, should you import them at the very beginning of bundling all files to avoid repetitively use them in every build.

npm install webpack --save-dev

npm install moment lodash --save

npm i webpack-manifest-plugin --save-dev

1. Create a new package.json

npm init -y

1. Install the necessary loaders and plugins using the commands above
2. Add index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Libraries Code Splitting</title>

</head>

<body>

<header>

<h3>Libraries Code Splitting</h3>

</header>

<div>

<label for="p1"><strong>Moment JS : </strong></label>

<p class="p1">

not yet loaded

</p>

<label for="p2"><strong>Lodash JS : </strong></label>

<p class="p2">

not yet loaded

</p>

</div>

<script src="dist/vendor.js"></script>

<script src="dist/main.js"></script>

</body>

</html>

1. Add app/index.js

var moment = require('moment');

var \_ = require('lodash');

var ele = document.querySelectorAll('p');

document.addEventListener("DOMContentLoaded", function(event) {

ele[0].innerText = moment().format();

ele[1].innerText = \_.drop([1, 2, 3], 0);

});

1. Add webpack.config.js

var webpack = require('webpack');

var path = require('path');

module.exports = {

entry: {

main: './app/index.js',

vendor: [

'moment',

'lodash'

]

},

output: {

filename: '[name].js',

path: path.resolve(\_\_dirname, 'dist')

}

}

optional

// 1

plugins: [

new webpack.optimize.CommonsChunkPlugin({

name: 'vendor' // Specify the common bundle's name.

}),

]

// 2

plugins: [

new webpack.optimize.CommonsChunkPlugin({

names: ['vendor', 'manifest'] // Extract the webpack bootstrap logic into manifest.js

}),

]

// 3

new ManifestPlugin({

fileName: 'manifest.json',

basePath: './dist/'

})

**Example 3 - Webpack Resolve & Plugins**

* Besides loader, plugins offer a wide variety of different features that Loaders don't provide

1. Create a new package.json and install plugins below

npm init -y

npm install webpack & jquery --save-dev

1. Add index.html

<html>

<head>

<title>Webpack Plugins</title>

</head>

<body>

<script src="dist/bundle.js"></script>

</body>

</html>

1. Add app/index.js

var $ = require('jquery');

console.log("loaded jQuery version is " + $.fn.jquery);

1. Add webpack.config.js

var path = require('path');

var webpack = require('webpack');

module.exports = {

entry: './app/index.js',

output: {

filename: 'bundle.js',

path: path.resolve(\_\_dirname, 'dist')

}

};

1. run webpack
2. uncomments #2 and #3 to see how Resolve alias & Provide Plugin works

// #2 - Using alias

// index.js

import $ from 'Vendor/jquery-2.2.4.min.js';

console.log("loaded jQuery version is " + $.fn.jquery);

// webpack.config.js

resolve: {

alias: {

Vendor: path.resolve(\_\_dirname, './app/vendor/')

}

}

// #3 - Using Provide Plugin

// index.js

console.log("loaded jQuery version is " + $.fn.jquery);

// webpack.config.js

plugins: [

new webpack.ProvidePlugin({

$: 'jquery'

})

]

**Example 4 - Webpack Dev Server Setting**

* Initial development setting to make the build process easier

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

webpack-dev-server --open

* or add this option to package.json to launch the dev server

"scripts": { "start": "webpack-dev-server" }

1. Create a new package.json and type the commands above
2. Add index.html

<html>

<head>

<title>Webpack Dev Server</title>

</head>

<body>

<div class="container">

hello

</div>

<script src="/dist/bundle.js"></script>

</body>

</html>

1. Add app/index.js

var ele = document.getElementsByClassName('container')[0];

ele.innerText = "Webpack loaded!!";

1. Add webpack.config.js

var path = require('path');

module.exports = {

entry: './app/index.js',

output: {

filename: 'bundle.js',

path: path.resolve(\_\_dirname, 'dist'),

publicPath: 'dist'

},

devtool: "cheap-eval-source-map",

devServer: {

publicPath: "/dist/",

port: 9000

},

};

1. Run npm start to launch the Webpack Dev Server

Please keep in mind that the **webpack devserver compiles in memory** not emits bundled file in output.path

**Example 5 - Webpack Dev Middleware**

* Have a full control over already installed Node.js by installing the commands below

npm install webpack --save

npm install express webpack-dev-middleware --save-dev

1. Create a new package.json and type the commands above
2. Add index.html

<html>

<head>

<title>Webpack Dev Middleware</title>

</head>

<body>

<div class="container">

hello

</div>

<script src="/dist/bundle.js"></script>

</body>

</html>

1. Create a new server.js file and add Express & EJS in it

var express = require('express');

var app = express();

var path = require('path');

app.use(express.static(\_\_dirname));

// view engine setup

// \_\_dirname : root folder

app.set('views', path.join(\_\_dirname));

app.set('view engine', 'ejs');

app.engine('html', require('ejs').renderFile);

app.get('/', function (req, res) {

res.send('index');

});

app.listen(3000);

console.log("Server running on port 3000");

1. Run server.js and make sure it doens't cause any errors
2. Add app/index.js

var ele = document.getElementsByClassName('container')[0];

ele.innerText = "Webpack loaded!!";

1. Add webpack.config.js

var path = require('path');

var webpack = require('webpack');

module.exports = {

entry: './app/index.js',

output: {

filename: 'bundle.js',

path: path.resolve(\_\_dirname, 'dist'),

publicPath: 'http://localhost:3000/dist'

},

};

1. Add the codes below to server.js

var webpackDevMiddleware = require("webpack-dev-middleware");

var webpack = require("webpack");

var webpackConfig = require("./webpack.config");

var compiler = webpack(webpackConfig);

app.use(webpackDevMiddleware(compiler, {

publicPath: webpackConfig.output.publicPath,

stats: {colors: true}

}));