**webpack3.6最新用法总结**

2017年10月08日 14:43:34

阅读数：470

**webpack的几大模块**

      如果说，一个webpack配置文件，我们到底需要配置的东西有什么？我觉得有这么几个重要的模块：entry（入口）、output（输出）、loaders、以及 plugins（插件）。

entry：代表入口。表示webpack从哪里开始解析你的项目依赖关系。

output：代表输出。表示webpack最终把你的项目打包到哪个文件中。

loaders：代表你用什么东西来解析你的各种文件。从原理上来说，webpack只认识js文件，其他文件需要通过各种loaders来实现解析。

plugins：代表你需要的一些其他的特殊功能。如js文件压缩插件-UglifyJsPlugin。

**各个模块的使用语法以及配置方式**

1、关于entry的使用   
entry可以有字符串的配置方式，也可以由对象的配置方式,其中对象的配置方式可以由多个入口。

字符串的配置方式：

module.exports = {

entry:'./src/index.js'

}

* 1
* 2
* 3

对象的配置方式（单入口）：

module.exports = {

entry:{

app:'./src/index.js'

}

}

* 1
* 2
* 3
* 4
* 5

对象的配置方式（多入口）：

module.exports = {

entry:{

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

vendor:'./src/vendor.js'

}

}

* 1
* 2
* 3
* 4
* 5
* 6

2、output的使用   
      output的使用和entry的使用相对应。有单个入口的写法，也有多个入口的写法（其中多个入口的写法兼容单个入口的写法，个人推荐使用第二种方式）。

单个入口的写法：

module.exports = {

output:{

filename:'bundle.js',

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

}

}

* 1
* 2
* 3
* 4
* 5
* 6

多个入口的写法：

module.exports = {

output:{

filename:'[name].js',

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

}

}

* 1
* 2
* 3
* 4
* 5
* 6

3、loaders写法   
test代表一个正则匹配。   
use代表使用什么样的loader解析。   
include代表在哪个目录下工作。   
exclude代表一定不解析某个目录和文件。

module:{

rules:[{

test:/\.(js|jsx)$/,

use:['babel-loader'],

include: path.resolve(\_\_dirname, "src"),

exclude: /node\_modules/

},{

test:/\.css$/,

include: path.resolve(\_\_dirname, "src"),

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

},{

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

include: path.resolve(\_\_dirname, "src"),

use:['file-loader']

},{

test:/\.xml$/,

include: path.resolve(\_\_dirname, "src"),

use:['xml-loader']

},{

test:/\.(csv|tsv)$/,

include: path.resolve(\_\_dirname, "src"),

use:['csv-loader']

},{

test:/\.less$/,

include: path.resolve(\_\_dirname, "src"),

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

}]

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28

**你不得不知的用法：devServer**

devServer是在你本地开启一个简单功能的服务器。用于开发调试。   
基本的使用方式：   
npm install –save-dev webpack-dev-server

devServer: {

contentBase: path.join(\_\_dirname,'dist'),

compress: true,

port: 8080,

hot: true

},

* 1
* 2
* 3
* 4
* 5
* 6

contentBase：表示在哪一个目录下作为服务的启动目录。   
port：表示本地服务的端口号是8080。   
hot：代表是否进行热替换。   
如果hot为true的话，需要配合插件 “uglify-webpack-plugin”。   
npm install –save-dev uglify-webpack-plugin

const UglifyJSPlugin = require('uglifyjs-webpack-plugin');

const Plugin\_UglifyJSPlugin = new UglifyJSPlugin();

module.exports = {

plugins:[

Plugin\_UglifyJSPlugin,

]

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

**你不得不知的用法：精简输出**

      精简输出主要是用于生产环境，在上线的时候，尽量让你的代码包更小，代码没有重复等等。

涉及到的插件有：UglifyJSPlugin 、DefinePlugin、CommonChunkPlugin 等。

1、UglifyJSPlugin：上面说到了这个插件，主要进行js文件的压缩。

//plugin5:UglifyJSPlugin

const UglifyJSPlugin = require('uglifyjs-webpack-plugin');

const Plugin\_UglifyJSPlugin = new UglifyJSPlugin({

sourceMap: true

});

* 1
* 2
* 3
* 4
* 5
* 6

2、DefinePlugin：这个插件主要让你分辨是开发环境还是准备上线环境。它在不同环境下加载的不同代码，生产环境下node自动加载针对用户的代码优化，开发环境下下自动加载额外log和日志。（细节下面会说到）

//plugin6:DefinePlugin

const Plugin\_DefinePlugin = new webpack.DefinePlugin({

'process.env':{

'NODE\_ENV':JSON.stringify('production')

}

});

* 1
* 2
* 3
* 4
* 5
* 6

3、CommonChunkPlugin：这个插件主要是针对公共代码提取。（细节下面会说到）

//plugin3:CommonChunkPlugin

const Plugin\_CommonChunkPlugin = new webpack.optimize.CommonsChunkPlugin({

name:'common'

});

* 1
* 2
* 3
* 4

**你不得不知的用法：devtool**

      devtool的使用，个人觉得它最大的方便在于错误定位。

开发环境中，给这个属性设置 inline-source-map，能够把错误定位到你的编码的地方。如果不设置，它只会在生成的bundle.js中某一行报错。

devtool: 'inline-source-map'

* 1

**你不得不知的用法：生产环境和v开发环境相分离**

      一般而言，生产环境和开发环境是有很大的区别的。人们总是期望，在生产环境中，我的代码是被压缩的，并且是有hash生成的。在开发环境中，我们需要代码及时的跟新，希望有报错提示等等。

      所以，我们需要配置两套不同的webpack环境，来适应不同的需求。而这些需求，一般是用不同的插件来实现的。根据需求，开发环境一般需要这几个插件：HotModuleReplacementPlugin、DefinePlugin。而生产环境需要的插件是：UglifyJSPlugin、HashedModuleIdsPlugin。

      生产环境和开发环境使用的不同的webpack配置示例，本文最后给出。

**你不得不知的用法：公用代码提取**

      公共代码的提取，可以极大的提高代码质量，很多情况下，这也是提升编译速度的一种方式。其使用到的插件是：CommonChunkPlugin。

output:{

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

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

publicPath: "/",

chunkFilename: '[name].js'//公共部分提取的文件名

},

plugins:[

Plugin\_CommonChunkPlugin,//公共部分提取

],

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

**关于自己配置的webpack以及package.json**

**1、webpack公共配置部分**

webpack.common.js

* 1
* 2

/\*\*

\* Created by mapbar\_front on 2017/10/6.

\*/

const path = require('path');

const webpack = require('webpack');

//plugin1:HtmlWebpackPlugin

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

const Plugin\_HtmlWebpackPlugin = new HtmlWebpackPlugin({

title: 'My App',

filename: 'index.html',

template: './src/assets/index.html'

});

//plugin2:CleanWebpackPlugin

const CleanWebpackPlugin = require('clean-webpack-plugin');

const Plugin\_CleanWebpackPlugin = new CleanWebpackPlugin(['dist']);

//plugin3:CommonChunkPlugin

const Plugin\_CommonChunkPlugin = new webpack.optimize.CommonsChunkPlugin({

name:'common'

});

//plugin3:CommonChunkPlugin

const Plugin\_CommonChunkPlugin2 = new webpack.optimize.CommonsChunkPlugin({

name:'vendor'

});

//config

const config = {

entry:{

app:\_\_dirname + '/src/index.js',

vendor: ['react']

},

output:{

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

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

publicPath: "/",

chunkFilename: '[name].js'

},

plugins:[

Plugin\_HtmlWebpackPlugin,//html模板插件

Plugin\_CleanWebpackPlugin,//清除dist的插件

Plugin\_CommonChunkPlugin2,//vendor提取

Plugin\_CommonChunkPlugin,//公共部分提取

],

module:{

rules:[{

test:/\.(js|jsx)$/,

use:['babel-loader'],

include: path.resolve(\_\_dirname, "src"),

exclude: /node\_modules/

},{

test:/\.css$/,

include: path.resolve(\_\_dirname, "src"),

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

},{

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

include: path.resolve(\_\_dirname, "src"),

use:['file-loader']

},{

test:/\.xml$/,

include: path.resolve(\_\_dirname, "src"),

use:['xml-loader']

},{

test:/\.(csv|tsv)$/,

include: path.resolve(\_\_dirname, "src"),

use:['csv-loader']

},{

test:/\.less$/,

include: path.resolve(\_\_dirname, "src"),

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

}]

}

};

module.exports = config;

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82

**2、webpack生产环境配置部分**

webpack.prod.js

* 1
* 2

/\*\*

\* Created by mapbar\_front on 2017/10/6.

\*/

const merge = require('webpack-merge');

const common = require('./webpack.common');

const webpack = require('webpack');

const path = require('path');

//plugin5:UglifyJSPlugin

const UglifyJSPlugin = require('uglifyjs-webpack-plugin');

const Plugin\_UglifyJSPlugin = new UglifyJSPlugin({

sourceMap: true

});

//plugin6:DefinePlugin

const Plugin\_DefinePlugin = new webpack.DefinePlugin({

'process.env':{

'NODE\_ENV':JSON.stringify('production')

}

});

//plugin7:

const Plugin\_HashedModuleIdsPlugin = new webpack.HashedModuleIdsPlugin()

module.exports = merge(common,{

devtool: 'source-map',

plugins: [

Plugin\_UglifyJSPlugin,//代码精简插件，未引用代码精简，代码压缩等。

Plugin\_DefinePlugin,//不同环境下加载的不同代码，生产环境下node自动加载针对用户的代码优化，开发环境下下自动加载额外log和日志

Plugin\_HashedModuleIdsPlugin,//基于模块id的hash解析

]

});

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34

**3、webpack开发环境配置部分**

webpack.dev.js

* 1
* 2

/\*\*

\* Created by mapbar\_front on 2017/10/6.

\*/

const merge = require('webpack-merge');

const common = require('./webpack.common');

const webpack = require('webpack');

const path = require('path');

//plugin4:Hot Module Replacement

const Plugin\_HMR = new webpack.HotModuleReplacementPlugin();

//plugin6:DefinePlugin

const Plugin\_DefinePlugin = new webpack.DefinePlugin({

'process.env':{

'NODE\_ENV':JSON.stringify('development')

}

});

module.exports = merge(common,{

devtool: 'inline-source-map',

devServer: {

contentBase: [path.join(\_\_dirname,'dist'),path.join(\_\_dirname,'public')],

compress: true,

port: 8080,

hot: true

},

plugins: [

Plugin\_HMR,//热替换插件

Plugin\_DefinePlugin,

]

});

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31

**4、关于webpack的package.json文件**

package.json

* 1
* 2

{

"name": "webpack3\_study",

"version": "1.0.0",

"description": "学习webpack3",

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

"scripts": {

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

"build": "webpack --config webpack.prod.js",

"watch": "webpack --watch --config webpack.dev.js",

"start": "webpack-dev-server --open --config webpack.dev.js",

"server": "node server.js"

},

"author": "mapbar\_front",

"license": "ISC",

"devDependencies": {

"babel-core": "^6.26.0",

"babel-loader": "^7.1.2",

"clean-webpack-plugin": "^0.1.17",

"css-loader": "^0.28.7",

"csv-loader": "^2.1.1",

"express": "^4.16.1",

"file-loader": "^1.1.5",

"html-webpack-plugin": "^2.30.1",

"inline-source-map": "^0.6.2",

"less-loader": "^4.0.5",

"style-loader": "^0.19.0",

"ts-loader": "^2.3.7",

"uglifyjs-webpack-plugin": "^0.4.6",

"url-loader": "^0.6.2",

"webpack": "^3.6.0",

"webpack-dev-middleware": "^1.12.0",

"webpack-dev-server": "^2.9.1",

"webpack-merge": "^4.1.0",

"xml-loader": "^1.2.1"

},

"dependencies": {

"lodash": "^4.17.4",

"react": "^16.0.0"

}

}