css单独分离打包需安装插件extract-text-webpack-plugin

安装 cnpm install extract-text-webpack-plugin --save-dev

修改webpack.config.js

增加：

**const** ExtractPlugin=require(**"extract-text-webpack-plugin”**);

**if**(isDev){

*// 区分环境,在开发环境下使用*

config.**module**.**rules**.push({

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

*// "loader":"style-loader!css-loader",*

**use**:[

**"style-loader"**,

**"css-loader"**

]

    });

}**else**{

*//在生产环境使用*

*config.entry={*

*app:path.join(\_\_dirname,"src/main.js"),*

*vendor:["vue","jquery","bootstrap"]*

*};*

config.**output**.**filename**=**'[name].[chunkhash:8].js'**;

    config.**module**.**rules**.push({

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

*// "loader":"style-loader!css-loader",*

**use**:ExtractPlugin.extract({

**fallback**:**"style-loader"**,

**use**:[

**"css-loader"**

]

        })

    });

    config.**plugins**.push(

*//在这里指定输出css文件的名字*

**new** ExtractPlugin(**"styles.[contenthash:8].css”**)，

**new** webpack.[optimize.CommonsChunkPlugin](http://optimize.commonschunkplugin/)({

**name**:**"vendor"**

}),

*//把生成在app.js中的相关的webpack代码打包到runtime.js中*

*//生成runtime.[XXX].js*

**new** webpack.[optimize.CommonsChunkPlugin](http://optimize.commonschunkplugin/)({

**name**:**"runtime"**

})

    )

}

这种带哈希值的文件名，可以帮助实现静态资源的长期缓存，在生产环境中非常有用。

* [hash]。hash值是特定于整个构建过程的。
* [chunkhash]。hash值是特定于每一个文件的内容的。

完整的配置文件

**const** path=require(**"path"**);

**const** HTMLPlugin=require(**"html-webpack-plugin"**);

**const** webpack=require(**"webpack"**);

*//打包单独css文件需用到ExtractPlugin插件*

**const** ExtractPlugin=require(**"extract-text-webpack-plugin"**);

**const** isDev=process.**env**.**NODE\_ENV**==**"development"**;*//在package.json里定义的环境变量，都可以通过process.env这个对象来得到*

**const** config={

**target**:**"web"**,*//webpack定义的目标是web平台*

**entry**:path.join(\_\_dirname,**"src/main.js"**),

**output**:{

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

**filename**:**"bundle.js"**

},

**module**:{

**rules**:[

            {

**"test"**:/\.vue$/,

**"loader"**:**"vue-loader"**

},

            {

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

*// "loader":"url-loader",*

**use**:[

                    {

**loader**:**"url-loader"**,

**options**:{

**limit**:1024,

*//name为图片的名称,ext为后缀*

**name**:**'[name].[ext]'**

}

                    }

                ]

            },

            {

**"test"**:/\.js$/,

**"exclude"**: /node\_modules/,

**"loader"**:**"babel-loader"**,

**"options"**:{

**"presets"**: [**"es2015"**],

**"plugins"**: [**'transform-runtime'**]

                }

            }

        ]

    },

**plugins**:[

*//在使用vue框架的时候需要用一个webpack插件*

*//vue项目在打包时会区分不同的环境打包*

*//在这里定义了,在写的js代码中也可以引用process.evn这个对象*

**new** webpack.DefinePlugin({

**"process.env"**:{

**NODE\_ENV**:isDev?**'"development"'**:**'"production"'**

}

        }),

**new** HTMLPlugin()

    ]

};

**if**(isDev){

*// 区分环境,在开发环境下使用*

config.**module**.**rules**.push({

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

*// "loader":"style-loader!css-loader",*

**use**:[

**"style-loader"**,

**"css-loader"**

]

    });

*//帮助在页面上调试代码,在项目里使用的是.vue的文件开发模式,使用的也是es6代码*

*//这些代码在浏览器里是不能直接运行的,如果直接去调试浏览器代码是经过编译的*

*//所以需要使用source-map去进行代码的映射,当在浏览器调试的时候还是自己编写的原码*

*//这样更容易定位到错误,webpack提供了devtool*

config.**devtool**=**"#cheap-module-eval-source-map"**;

    config.**devServer**={

**port**:**"9900"**,

**host**:**"0.0.0.0"**,*//设置成0.0.0.0的好处是除了可以通过localhost和127.0.0.1访问,还能通过内网ip地址访问*

**overlay**:{

**errors**:**true***//当webpack编译过程中出现任何错误,让它显示在网页上面*

},

**hot**:**true***//只刷新当前组件的代码不刷新整个页面*

};

*//设置hot后需要加相关插件*

config.**plugins**.push(

*//启动hot功能的插件*

**new** webpack.HotModuleReplacementPlugin(),

**new** webpack.NoEmitOnErrorsPlugin()

    )

}**else**{

*//在生产环境使用*

*//把main.[chunkhash:8].js中业务代码和类库代码分开打包*

*//生成app.[XXX].js和vendor[XXX].js*

config.**entry**={

**app**:path.join(\_\_dirname,**"src/main.js"**),

**vendor**:[**"vue"**,**"jquery"**,**"bootstrap"**]

    };

*//生成main.[chunkhash:8].js*

config.**output**.**filename**=**'[name].[chunkhash:8].js'**;

*//打包css*

*//生成styles.[contenthash:8].css*

config.**module**.**rules**.push({

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

*// "loader":"style-loader!css-loader",*

**use**:ExtractPlugin.extract({

**fallback**:**"style-loader"**,

**use**:[

**"css-loader"**

]

        })

    });

    config.**plugins**.push(

*//在这里指定输出css文件的名字*

**new** ExtractPlugin(**"styles.[contenthash:8].css"**),

**new** webpack.[optimize.CommonsChunkPlugin](http://optimize.commonschunkplugin/)({

**name**:**"vendor"**

}),

*//把生成在app.js中的相关的webpack代码打包到runtime.js中*

*//生成runtime.[XXX].js*

**new** webpack.[optimize.CommonsChunkPlugin](http://optimize.commonschunkplugin/)({

**name**:**"runtime"**

})

    )

}

module.exports=config;