全局安装

npm install webpack-dev-server －g

项目安装

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

这里建议进行项目安装，这样别人开发的时候只需要执行npm install 命令就能安装了，没有必须单独进行 全局安装了。

--content-base //设定webpack-dev-server的director根目录。如果不进行设定的话，默认是在当前目录下。

--quiet: //控制台中不输出打包的信息，开发中一般设置为false，进行 打印，这样查看错误比较方面

--no-info: // 不显示任何信息

--colors: //对信息进行颜色输出

--no-colors: //对信息不进行颜色输出

--compress:  //开启gzip压缩

--host <hostname/ip>: //设置ip

--port <number>: //设置端口号，默认是:8080

--inline: //webpack-dev-server会在你的webpack.config.js的入口配置文件中再添加一个入口,

--hot: //开发热替换

--open: //启动命令，自动打开浏览器

--history-api-fallback: //查看历史url

1、content-base

设定webpack-dev-server的根目录。如果不进行设定的话，默认是在当前目录下。

webpack-dev-server --content-base ./assets  
这个时候还要注意的一点就是在webpack.config.js文件里面，如果配置了output的publicPath这个字段的值的话，在index.html文件里面也应该做出调整。**因为**webpack-dev-server的根目录**是相对publicPath这个路径的**。因此，如果你的webpack.config.js配置成这样的：

 module.exports = {

        entry: './src/js/index.js',

        output: {

            path: './dist/js',

            filename: 'bundle.js'，

            publicPath: '/assets/'

        }

    }  
那么，在index.html文件当中引入的路径也发生相应的变化:

    <!DOCTYPE html>

    <html lang="en">

    <head>

        <meta charset="UTF-8">

        <title>Demo</title>

    </head>

    <body>

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

    </body>

    </html>  
如果在webpack.config.js里面没有配置output的publicPath的话，那么index.html最后引入的文件js文件路径应该是下面这样的。

    <!DOCTYPE html>

    <html lang="en">

    <head>

        <meta charset="UTF-8">

        <title>Demo</title>

    </head>

    <body>

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

    </body>

    </html>

直接在命令行执行一下命令，就可以启动服务器：

webpack-dev-server --content-base ./assets

四、webpack-dev-server 的热替换

inline model

devServer: {

historyApiFallback: true,

contentBase: "./",

quiet: false, //控制台中不输出打包的信息

noInfo: false,

hot: true,

inline: true,

lazy: false,

progress: true, //显示打包的进度

watchOptions: {

aggregateTimeout: 300

},

port: '8088'

}

二、plugins插件

**常用Plugins介绍**

* 代码热替换, HotModuleReplacementPlugin
* 生成html文件，HtmlWebpackPlugin
* 将css成生文件，而非内联，ExtractTextPlugin
* 报错但不退出webpack进程，NoErrorsPlugin
* 代码丑化，UglifyJsPlugin，开发过程中不建议打开
* 多个 html共用一个js文件(chunk)，可用CommonsChunkPlugin
* 清理文件夹，Clean
* 调用模块的别名ProvidePlugin，例如想在js中用$，如果通过webpack加载，需要将$与jQuery对应起来

module.exports = {

// ...

devServer: {

hot: true

}

}

## API

**var** WebpackDevServer = require("webpack-dev-server");

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

**var** compiler = webpack({

*// configuration*

});

**var** server = **new** WebpackDevServer(compiler, {

*// webpack-dev-server options*

contentBase: "/path/to/directory",

*// Can also be an array, or: contentBase: "http://localhost/",*

hot: true,

*// Enable special support for Hot Module Replacement*

*// Page is no longer updated, but a "webpackHotUpdate" message is sent to the content*

*// Use "webpack/hot/dev-server" as additional module in your entry point*

*// Note: this does \_not\_ add the `HotModuleReplacementPlugin` like the CLI option does.*

historyApiFallback: false,

*// Set this as true if you want to access dev server from arbitrary url.*

*// This is handy if you are using a html5 router.*

compress: true,

*// Set this if you want to enable gzip compression for assets*

proxy: {

"\*\*": "http://localhost:9090"

},

*// Set this if you want webpack-dev-server to delegate a single path to an arbitrary server.*

*// Use "\*\*" to proxy all paths to the specified server.*

*// This is useful if you want to get rid of 'http://localhost:8080/' in script[src],*

*// and has many other use cases (see https://github.com/webpack/webpack-dev-server/pull/127 ).*

setup: **function**(app) {

*// Here you can access the Express app object and add your own custom middleware to it.*

*// For example, to define custom handlers for some paths:*

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

*// res.json({ custom: 'response' });*

*// });*

},

*// pass [static options](http://expressjs.com/en/4x/api.html#express.static) to inner express server*

staticOptions: {

},

clientLogLevel: "info",

*// Control the console log messages shown in the browser when using inline mode. Can be `error`, `warning`, `info` or `none`.*

*// webpack-dev-middleware options*

quiet: false,

noInfo: false,

lazy: true,

filename: "bundle.js",

watchOptions: {

aggregateTimeout: 300,

poll: 1000

},

*// It's a required option.*

publicPath: "/assets/",

headers: { "X-Custom-Header": "yes" },

stats: { colors: true }

});

server.listen(8080, "localhost", **function**() {});

*// server.close();*