1. 简介：

Webpack 是当下最热门的前端资源模块化管理和打包工具。它可以将许多松散的模块按照依赖和规则打包成符合生产环境部署的前端资源。还可以将按需加载的模块进行代码分隔，等到实际需要的时候再异步加载。

webpack已经是大部分前端项目打包工具的首选，grunt、glup、browserify、fis等逐渐沦为辅助甚至完全被替代。

1. 安装：

注册cnpm：

npm install -g cnpm --registry=https://registry.npm.taobao.org

全局安装npm install -g webpack

项目安装

cnpm init

cnpm install –save-dev webpack

1. 使用：
2. 无配置文件使用：

webpack 源文件 目标文件

例：webpack index.js bound.js

1. 使用配置文件：

* 默认配置文件:

webpack.config.js

* 可以配置多个config文件

1. webpack.config.js文件的配置

需要采用common.js规范进行文件的抛出一个webpack的配置对象：

module.exports={}

1. 入口文件属性：
2. 字符串：

entry：\_\_dirname+文件路径 //\_\_dirname是node的全局文件根目录，推荐采用此种写法；

entry：./文件路径 //此种写法,貌似在3.0版本中不再支持，不推荐此种写法

1. 对象语法：
2. 单入口文件：

entry：{

main：\_\_dirname+文件路径

}

1. 多入口文件：

entry：{

app1：\_\_dirname+文件路径，

app2：\_\_dirname+文件路径，

}

1. 输出文件属性：

output：{

path: \_\_dirname+文件路径 //\_\_dirname是node的全局文件根目录，采用此种写法；

path：./文件路径 //此种写法不支持

filename：输出文件名 注：如果多入口文件，不能指定文件名，需要使用[name]动态获取；

chunkFilename:[name][hash] name为模块的名字，hash加入hash值，防止缓存，使用此属性是，模块的引入必须结合require.ensure方法，否则此方法无效；

chunkFilename: "[id][name][hash].js"

publicPath:设置服务路径

}

1. 加载器：
2. 语法：

module: {

rules: [

{test: /\.css$/, use: ['css-loader'](/loaders/css-loader)},

]

}

1. rules数组中每个规则的属性：

test：要匹配的文件类型；

use:使用何种加载器进行加载处理文件，可以同时使用多个loader，数组的形式；数组的值，可以为字符串，或者对象；

loader：使用何种加载器加载处理文件，只能加载一个loader。字符串形式；

options：加载器的配置项;

1. 常用的loader：

cnpm install babel babel-core babel-loader babel-preset-es2015 css-loader eslint eslint-loader html-loader html-webpack-plugin style-loader url-loader webpack webpack-dev-server --save-dev

1. 插件：
2. 语法：

plugins：[ ]

1. 常用插件：
2. 内置插件

UglifyJsPlugin

HotModuleReplacementPlugin

CommonsChunkPlugin：

配置项：

new webpack.optimize.CommonsChunkPlugin({

name: "commons",

// ( 公共chunk(commnons chunk) 的名称)

filename: "commons",

// ( 公共chunk 的文件名)

})

1. 三方插件

cnpm install –save-dev html-webpack-plugin

HtmlWebpackPlugin

配置项：

{

template：”” //入口html的路径

}

1. devtool:

**生产环境主要使用：（推荐使用**cheap-source-map**）**

1. source-map - 生成完整的 SourceMap，输出为独立文件。由于在 bundle 中添加了引用注释，所以开发工具知道在哪里去找到 SourceMap
2. hidden-source-map - 和 source-map 相同，但是没有在 bundle 中添加引用注释。如果你只想要 SourceMap 映射错误报告中的错误堆栈跟踪信息，但不希望将 SourceMap 暴露给浏览器开发工具。
3. cheap-source-map - 不带列映射(column-map)的 SourceMap，忽略加载的 Source Map。
4. cheap-module-source-map - 不带列映射(column-map)的 SourceMap，将加载的 Source Map 简化为每行单独映射。
5. nosources-source-map - 创建一个没有 sourcesContent 的 SourceMap。它可以用来映射客户端（译者注：指浏览器）上的堆栈跟踪，而不会暴露所有的源码。

**开发环境主要使用：（推荐使用**cheap-module-eval-source-map**）**

1. eval - 每个模块都使用 eval() 执行，并且都有 //@ sourceURL。此选项会相当快地构建。主要缺点是，由于会映射到转换后的代码，而不是映射到原始代码，所以不能正确的显示显示行数。
2. inline-source-map - SourceMap 转换为 DataUrl 后添加到 bundle 中。
3. eval-source-map - 每个模块使用 eval() 执行，并且 SourceMap 转换为 DataUrl 后添加到 eval() 中。初始化 SourceMap 时比较慢，但是会在重构建时提供很快的速度，并且生成实际的文件。行数能够正确映射，因为会映射到原始代码中。
4. cheap-module-eval-source-map - 和 cheap-eval-source-map 类似，然而，在这种情况下，loader 能够处理映射以获得更好的结果。
5. devServer:
6. 基于webpack的静态服务：
7. 全局安装webpack-dev-server（实现热更新，必要时需要webpack-dev-server的项目安装）
8. 配置项：

{

//告诉服务器从哪里提供内容。这只有在您想要提供静态文件时才需要。例如图片？？

contentBase：\_\_dirname + 'dist',

// --告诉服务器观看由devServer.contentBase选项提供的文件。文件更改将触发整个页面重新加载。

watchContentBase: true,

// --随所有内容启用gzip压缩

compress: true,

// 模块的热加载，必须结合 HotModuleReplacementPlugin使用

hot: true,

port: 9999,

host: '127.0.0.1',

//相当于gulp的middleware中间件拦截请求；

setup(app) {

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

console.log(11)

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

});

},

proxy: {

// "/api": "http://localhost:3000/"

"/": "http://localhost:3000/"

// "/api": {

// target: "http://localhost:3000",

// pathRewrite: {"^/api" : "/backapi"},

// secure: false

// }

},

}

1. webpack手动搭建vue项目
2. 需要的loader以及插件的安装：

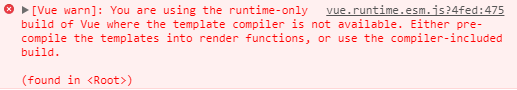
cnpm install babel-core babel-loader babel-preset-es2015 eslint eslint-loader html-webpack-plugin url-loader webpack webpack-dev-server vue vue-loader vue-template-compiler css-loader --save-dev

babel-plugin-transform-runtime（对于一些全局的方法进行es6的转码）

1. 组件式开发的发展：

早 期进行组件化编写时，我们往往会把一个组件的 html、css、js 放在三个不同的文件中，然后 利用编译工具再合到一起，vue-loader 的出现，使得我们能将一个组件的 html、css、js 放在一个文件中；

1. template 标签中的即为该组件的DOM 结构，默认采用 HTML 形式，每个.vue 文件中 最多只能包含一个 template 标签，可以通过配置 template 的 lang 属性，使用不同的模板语法；
2. script 标签中即为该组件的 js 代码，且同 template 一样，一个 .vue 文件中最多只能包 含一个script 标签，而且最终必须输出（export）一个符合Vue.extend() 参数规范的对象， 用于建立 Vue 组件构建器；
3. style 标签即为该组件的CSS代码，同个vue 文件中可以包含多个style 标签，除了直 接使用CSS写法外，还可以通过配置loader，支持sass、less 等样式写法。另外还有一个 scoped 属性，添加之后，vue-loader 会把当前同 .vue 文件 template 中的 DOM 都添加一个 \_v….. 的属性，并把 style 中的样式也加上对应的属性选择器，使得这部分样式仅在当前 vue 的 DOM 中生效。这个方式使得组件间的样式不会互相冲突，也不需要过长的命名来维护
4. 注意事项：
5. 在引用vue这个包时，注意需要引用编译时的文件(vue.js)，而不能引用运行的文件（vue.common.js）,vue作为模块引入时，默认引入的就是运行的文件，此文件不支持template模板编译，只支持render运行时构建；报错如下



解决办法：

1. 在webpack中指定vue的别名，进行引用；配置如下：

resolve: {

alias: {

"vue": "vue/dist/vue.js"

}

},

1. 直接在引入vue包的时候指定对应的路径：

import Vue from " vue/dist/vue.js ";

1. 其他工具

webpack

js：处理

babel-core

babel-loader

babel-preset-es2015

vue处理：

vue-template-compiler

vue-loader

css处理：

本地服务器：

webpack-dev-server