Vue-cli安装教程

node：https://www.runoob.com/nodejs/nodejs-install-setup.html

vue-cli：https://blog.csdn.net/qq_36711388/article/details/79405402

项目创建

创建一个基于webpack的新项目：vue init webpack projectName;

安装依赖：npm install;

运行： npm run dev;

java专题学习

任务自动化 [gulp工具]减少人工操作，让电脑自动监听并相应。

编译工具（babel、webpack）处理项目中的依赖关系。

使ES6变成浏览器可以运行的ES版本如（ES5）。

目录解析

build：构建脚本目录，配置了webpack的基本配置、开发环境配置、生产环境配置等。

config：构建配置目录，配置了路径端口值。

nodes_modules：依赖的node工具包目录，安装的axios，stylus等等需要的模块。

src：放置组件和项目入口文件。

assets：项目中的一些css，font，img等资源都存在这里

components：存放项目中的组件。

router：路由的相关配置，路径指示。

store：使用vuex时才建立这个文件夹，存放vuex相关文件。

App.vue：页面级vue组件。

main.js：页面入口js文件（很多项目必须的资源需要在这里引入）。

static：静态文件目录，没有接口数据时，也可以把模拟数据存在这里，进行测试。

index.html：入口页面。

package.json：项目描述文件。

build

build.js 将项目打包成静态文件。存放在项目根目录的dist文件夹中。

check-version.js 检查一些所依赖的工具的版本是否适用，如noejs、npm，若版本太低则会显示出来。

dev-client.js 本地开发热部署。

dev-server.js 是一个用作服务器端的东西，涵盖了express和它的一些模块，为了在本地服务器上把我们的项目跑起来的一个文件，引入了反向代理的模块，我们可以用来发起跨域

请求。

utils.js 文件引入了css-loader，以便于解析各种格式的css如less，sass什么的。

vue-loader.conf.js 引入了utils.js，用于切换开发模式和生产模式。

config

```
// see http://vuejs-templates.github.io/webpack for documentation.
// path是node.js的路径模块，用来处理路径统一的问题
var path = require('path')

module.exports = {
    // 下面是build也就是生产编译环境下的一些配置
    build: {
        // 导入prod.env.js配置文件，只要用来指定当前环境，详细见(1)
        env: require('./prod.env'),
        // 下面是相对路径的拼接，假如当前跟目录是config，那么下面配置的index属性的属性值就是dist/index.html
        index: path.resolve(__dirname, '../dist/index.html'),
        // 下面定义的是静态资源的根目录 也就是dist目录
        assetsRoot: path.resolve(__dirname, '../dist'),
        // 下面定义的是静态资源根目录的子目录static，也就是dist目录下面的static
        assetsSubDirectory: 'static',
        // 下面定义的是静态资源的公开路径，也就是真正的引用路径
        assetsPublicPath: '/',
        // 下面定义是否生成生产环境的sourcmap，sourcmap是用来debug编译后文件的，通过映射到编译前文件来实现
        productionSourceMap: true,
        // Gzip off by default as many popular static hosts such as
        // Surge or Netlify already gzip all static assets for you.
        // Before setting to `true`, make sure to:
        // npm install --save-dev compression-webpack-plugin
        // 下面是是否在生产环境中压缩代码，如果要压缩必须安装compression-webpack-plugin
        productionGzip: false,
        // 下面定义要压缩哪些类型的文件
        productionGzipExtensions: ['js', 'css'],
```

```
        // Run the build command with an extra argument to
        // View the bundle analyzer report after build finishes:
        // `npm run build --report`
        // Set to `true` or `false` to always turn it on or off
        // 下面是用来开启编译完成后的报告，可以通过设置值为true和false来开启或关闭
        // 下面的process.env.npm_config_report表示定义的一个npm_config_report环境变量，可以自行设置
        bundleAnalyzerReport: process.env.npm_config_report
    },
    dev: {
        // 引入当前目录下的dev.env.js，用来指明开发环境，详见(2)
        env: require('./dev.env'),
        // 下面是dev-server的端口号，可以自行更改
        port: 8080,
        // 下面表示是否自定代开浏览器
        autoOpenBrowser: true,
        assetsSubDirectory: 'static',
        assetsPublicPath: '/',
        // 下面是代理表，作用是用来，建一个虚拟api服务器用来代理本机的请求，只能用于开发模式
        // 详见(3)
        proxyTable: {},
        // CSS Sourcemaps off by default because relative paths are "buggy"
        // with this option, according to the CSS-Loader README
        // (https://github.com/webpack/css-loader#sourcemaps)
        // In our experience, they generally work as expected,
        // just be aware of this issue when enabling this option.
        // 是否生成css，map文件，上面这段英文就是说使用这个cssmap可能存在问题，但是按照经验，问题不大，可以使用
        // 给人觉得没必要用这个，css出了问题，直接控制台不就完事了
        cssSourceMap: false
    }
}
```

(1)下面是prod.env.js的配置内容

```
module.exports = {
    // 作用很明显，就是导出一个对象，NODE_ENV是一个环境变量，指定production环境
    NODE_ENV: '"production"'
}
```

(2)下面是dev.env.js的配置内容

```
// 首先引入的是webpack的merge插件，该插件是用来合并对象，也就是配置文件用的，相同的选项会被覆盖，至于这里为什么多次一举，可能另有他图吧
var merge = require('webpack-merge')
// 导入prod.env.js配置文件
var prodEnv = require('./prod.env')
// 将两个配置对象合并，最终结果是 NODE_ENV: '"development"'
module.exports = merge(prodEnv, {
    NODE_ENV: '"development"'
})
```

(3)下面是proxyTable的一般用法

vue-cli使用这个功能是借助http-proxy-middleware插件，一般解决跨域请求api

```
proxyTable: {
    '/list': {
        target: 'http://api.xxxxxxxx.com', -> 目标url地址
        changeOrigin: true, -> 指示是否跨域
        pathRewrite: {
        '^/list': '/list' -> 可以使用 /list 等价于 api.xxxxxxxx.com/list
        }
    }
}
```