webpack

webpack就是使用模块化进行开发，打包需要使用的文件到服务器

webpack的执行依赖于node环境

node环境为了执行很多代码，必须其中包含各种依赖的包

npm是包管理工具 node packages manager

webpack局部安装

在本地安装一个webpack npm install [webpack@3.6.0](mailto:webpack@3.6.0) --save-dev

webpack的作用就是打包一个文件发送到服务器，完事之后他就可以去死了

是前端模块化打包工具

全局安装webpack 首先在node中直接

npm install [webpack@3.6.0](mailto:webpack@3.6.0) -g -g的意思就是全局安装 全局的就是在哪里都可以访问的

在写好文件后如（common js）用webpack打包

语法 进入那个父目录文件（包括打包文件和创建文件的那个父目录）

-----》 webpack 打包的文件 创建的文件

初始化包 ：npm init 生成一个package.json的文件这个里面有一些依赖

安装包的依赖： npm install 生成一个package-lock.json文件 就是上面json中有一些依赖的话，会生政一个依赖的文件

在使用node时候最好 初始化一个json的文件

loader

loader是一个转换器和一个加载器

loader就是转化各种webpack不能打包的文件如（图片，css，es6转化成为es5、、）

vue

在安装好webpack的局部安装后 需要结合vue来一起使用

在npm install vue –save 注意不是—save-dev 这个dev是开发时依赖只有开发的时候才会依赖的 运行的时候是不依赖的在安装的某个东西的时候

而这个vue那是巡行的时候也是依赖的，运行的就是vue的代码你说依赖不依赖而打包的项目里面dist也在用这个vue的文件啊，这个时候就不要加dev了

打包vue，使用vue

有两个版本

1. runtime-only🡪代码中，不可以有任何的template
2. runtime-compiler🡪代码中可以有template因为可以用于编译template

打包vue代码

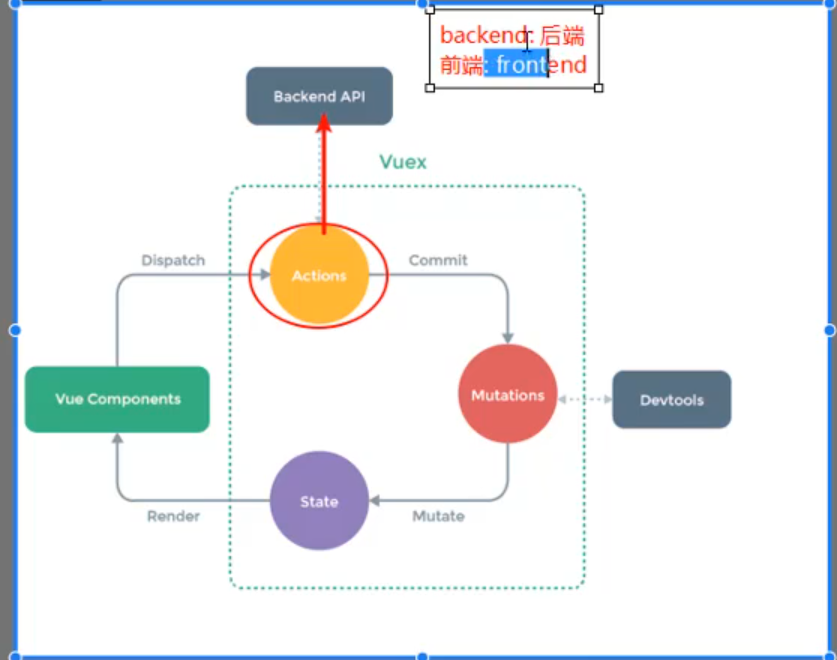
npm install vue-loader(加载程序) vue-template-compiler(编译) –save-dev

搭建本地服务器

npm install webpack-dev-server@2.9.3 --save -dev

vuex

vuex就是集中式和响应式的管理数据。把公共的数据放到一个文件里面去管理，使用的时候直接拿出来来使用就可以了。



Git安装

用户名字：yueshiwei666 [1467658005@qq.com](mailto:1467658005@qq.com)

密码： yue1467658005

1.add的需要操作的文件------

2.commit到暂存区文件------

3.push到git仓库文件

<https://www.git-scm.com/download/win> 地址

git的克隆文件到本地 git clone 地址‘为文件设置名字’

查看状态 git status

seem is取消那些暂存的文件 git restore

添加文件 git add . ----》上传到本地

初始化项目 git commit -m ‘初始化项目’ ---->上传到暂存区

提交到服务器 git push ----->上传到git仓库

恢复删除的文件 git checkout 文件名字

创建一个子分支 git checkout -b文件名字

查看当前目录的分支 git branch

查看服务器的所有分支以及自己当前所在的分支 git branch -a

创建本地分支 git branch 分支名字

切换到指定的分支 git checkout 分支的名称

合并分支 git merge 分支名称

删除分支 git branch -d 分支名称 -d是强制删除分支

创建一个忽略文件 touch gitignore 这个文件需要配置，就是上传一堆文件时，可以在这个文件中设置那个不需要上传的文件，文件在一开始的编辑器里面写上就生效了，就是写上你忽略的文件的名字，然后创建你忽略的文件的名字ok

具体的配置信息：

1. 以#开始的为注释的规则
2. \表示转义符 就是 ‘\ 就是’
3. !开始的表示否定
4. /开始的表示项目跟目录
5. /结尾的表示某个文件夹，就是忽略这个文件夹
6. \*.txt \*好开始的表示忽略的文件后缀or指定的文件的名字



axios

command line: npm install axios –save //保存之后直接运行使用

axios对网络上的数据进行请求

在main.js中导入axios import axios from ‘axios’

在文件的内部使用

设置axios的全局的配置

axios.defaults.timeout=5;

axios.defaults.baseURL=’’;

axios({

baseURL:’写入基础的url就行,下面的url连接上面的继续写就行’，

params:{ //参数的意思

type:’pop’,

page:1

} //如果url太长的话，可以在这里分开写，axios框架对自动的对路径进行拼接的

timeout:5, //设置超时的时间，超过这个时间数据请求失败

url(‘http://123.207.32.32:8000/home/multidata’)

//这里他整个会返回一个new Promise((resolve,reject) =>{resolve(‘数据’)})

//就是axios他内部会返回一个promise在在他的内部会执行的reslove方法，所以直接在后面调用then的方法就可以了

}).then(result =>{console.log(result)}) //输出结果

axios.all() 可以同时发送两个请求

发送并发请求

axios.all([axios({url:’’}), axios({url:’’})]).then( result =>{ 结果result是一个数组})