1. 项目运行起来的时候，让浏览器自动打开

就是在package.json文件的运行属性“dev”或者”serve”代码末端加上--open

“serve”:’vue-cli-service serve --open’

1. eslint校验功能关闭(比如避免声明变量但是没有使用而造成的报错)

在文件的根目录下创建vue.config.js文件

module.exports={

lintOnSave:false

}

1. src文件夹简写方法。配置别名[@components]

在文件的根目录下创建jsconfig.json文件

{

“compilerOptions”:{

“baseUrl”:”./”,

“paths ”:{

“@/\*”:[“src/\*”]

}

},

“exclude”:[“node\_modules”,”dist”]

}

1. 编程式导航在传参时（相同路由），多次执行会报错？

声明式导航<router-view>不会因为多次传参（相同路由）而报错，因为vur-router的底层已经处理好了

而编程式导航this.$router.push/replace会报错

1. this的指向问题？

this指向的时当前组件实例，

this.$router属性：当前的这个属性，属性值VueRouter类的一个实例，当在入口文件注册路由的时候，给组件实例添加$router|$route属性

push:Router类的一个实例

1. 全局组件怎么设置
2. 创建组件（.vue文件）
3. 在maIn.js文件中导入全局组件(import...)
4. 在main.js文件中注册全局组件（Vue.component(TypeNav.name,TypeNav)）
5. 如何利用Postman测试接口？

需要将项目的服务器地址与请求地址拼接起来，并且选择正确的接口方式

1. 为什么需要进行二次封装axios？

请求拦截器：可以在发送请求之前处理一些业务

响应拦截器：当服务器数据返回以后，可以处理一些事情

1. API接口统一管理（注意跨域问题）

在api文件夹下创建index.js文件

引入request.js=> import request from ‘./request’

发请求：axios发送请求返回结果为Promise对象：export const reqCategoryList=()=>request({url:’/product/getBaseCategoryList’,method:’get’})

在main.js文件中进行测试调用

Import {reqCategoryList} from ‘@/api’

reqCategoryList();

1. 如何实现跨域？

JSONP,CORS,代理

1. VueX是什么？

VueX是官方提供的一个插件，状态管理库，集中式管理项目中组件共用的数据

12为什么要使用Vuex?

并不是所有的项目都需要Vuex，如果项目很小，完全不需要Vuex

当项目很大，组件很多，组件之间的关系很复杂的时候，数据很多且复杂，就需要Vuex来管理这些组件共用的数据

1. Vuex的几个核心概念是？

State:仓库存储数据的地方

Mutations:修改state的唯一手段

Action:处理action，可以书写自己的业务逻辑，也可以处理异步

Getters:理解为计算属性，用于简化仓库数据，让组件获取仓库的数据更加方便

Modules:模块化，将大仓库分为小仓库，逻辑清晰

1. Vux可以实现模块化开发

把大仓库的数据拆分成很多小仓库，按照模块去存储数据

1. 如何实现动态获取数据？

首先需要在store文件夹下创建主仓库文件index.js，然后重新创建小仓库home文件夹，同时里面也创建index.js文件，在index,js文件里

// home仓库

//引入reqCategoryList方法并调用

import { reqCategoryList } from "@/api"

// 创建对象

const state = {

    // state中的数据默认初始值很关键，不能乱写

    // 若服务器返回的时对象，则初始值是对象

    // 若服务器返回的是数组，则初始值是数组

    categoryList: []

};

const mutations = {

 // 修改仓库中过的数据唯一手段

    // categoryList是服务器返回的数据

    CATEGORYLIST(state, categoryList) {

        state.categoryList = categoryList;

    }

};

const actions = {

    // 通知API里面的接口函数调用，向服务器发送请求，获取服务器的数据

    // async和await是同时存在的，返回成功的回调

    async categoryList({ commit }) {

        let result = await reqCategoryList();

        // console.log(result);

        if (result.code == 200) {

            commit("CATEGORYLIST", result.data)

        }

    }

};

const getters = {};

// 对外暴露Store类的一个实例

export default ({

    state,

    mutations,

    actions,

    getters

})

接着，就需要在引用数据的TypeNav组件里

导入mapState ;Import {mapState} from “vuex”,

然后

 computed: {

    // mapState是state的辅助函数，也可以说是state的语法糖

    ...mapState({

      // 意思是将vuex中的数据映射到组件的computed属性中

      // 右侧需要的是一个函数，当使用这个计算属性的时候，右侧的函数会立即执行一次

      // 注入一个参数state，其实即为大仓库中的数据

      categoryList: (state) => {

        // console.log(state);

        return state.home.categoryList;

      },

    }),

  },

最后就是在标签内部引用数据

<div class="item" v-for="(c1,index) in categoryList" :key="c1.categoryId">

          <h3>

              <a href="">{{c1.categoryName}}</a>

            </h3>

1. 通过JS控制二三级分类

最开始的时候，是通过CSS样式display:block|none显示与隐藏二三级商品分类

通过JS手段为二三级目录绑定style样式:<div class=”item-list clearfix” :style=”{display:currentIndex==index?’block’:’none’;}”>

16防抖与节流

用户的操作过快，当回调函数中有大量业务时，可能会出现卡顿现象

节流：在规定的间隔时间范围内不会重复触发回调，只有大于这个时间间隔才会触发回调，把频繁触发变为少量触发

防抖：前面的所有触发都被取消，最后一次执行在规定的时间之后才会触发，也就是说如果连续快速的触发，只会执行一次（更合理一些）

lodash插件：里面封装函数的防抖与节流的业务【核心：闭包+延迟器】

防抖：input.oninput=\_debounce(function(){

//在触发后延迟一秒再执行，并且只执行一次

Console.log(‘ajax请求)

}，1000)

节流：button.onclick=\_throttle(function(){

Count++;

//在点击后，延迟五秒才执行下一次，五秒之内不会再执行，五秒之后再执行下一次

//在业务中，可以给浏览器更多的时间去解析

Console.log(“执行”)

},5000)

总结：

防抖：用户操作很频繁，但是只执行一次

节流：用户操作很频繁，但是把频繁的此操作变为少量操作，可以给浏览器有充裕的时间解析代码

1. 如何使用mock模拟数据？

首先安装mockjs：cnpm install mockjs --save

(1)需要在src文件夹下创建mock文件夹，紧接着需要准备JSON数据

创建banner.json文件，将JSON数据写出进去（代码需要格式化，避免有空格而导致失败）

创建floor.json文件,将JSON数据写出进去（webpack默认对外暴露：图片，JSON数据格式）

(2)把mock数据需要的图片放置到public文件夹中【public文件夹在打包的时候，会把相应的资源原封不动打包到dist文件夹中】

(3)开始mock（虚拟的数据）,通过mockjs模块实现，创建mockServe.js文件

编写mockServe.js文件

1. 在main.js引入mockServe.js

Import Mock from “mockjs”

Import banner from “./banner.json”

Import floor from “./floor.json”

//mock数据：第一个参数：请求地址 第二个参数：请求数据

Mock.mock(“/mock/banner”,{code:200,data:banner})

Mock.mock(“/mock/floor”,{code:200,data:floor})

1. 轮播图的最完美的解决方案

Watch + nextTick数据监听：监听已有数据变化

nextTick:在下次DOM更新循环结束之后执行延迟回调，在修改数据之后立即使用这个方法，获取更新后的DOm

This.$nextTIck(()=>{})

1. 组件通信的方式有哪些

props:用户父子组件通信

自定义时间:@on,@emit

全局事件总线:$bus(全能)

插槽:

Vux

Pubsub-js：vue当中几乎不同(全能)

1. vue中计算属性和方法的不同

在vue中，计算属性和方法达到的效果是一样的，

但是计算属性是基于依赖进行缓存的，只有相关依赖发生改变时他们才会重新求值

而方法是，只要每次触发渲染，总会调用方法再次执行函数

1. 分页器的封装原理？

知道当前在第几页:pageNo

知道分页器一共需要展示多少条数据：total

知道每一页需要展示数据的个数：pageSize

知道连续的页码数--奇数（5|7）

1. vue中style和class的区别

Vue中style是一个对象，内联样式

class是一个类,不会对已有的class产生影响

1. 什么是路由的懒加载？

懒加载简单来说就是延迟加载或者按需加载，即在需要的时候进行加载

目的是为了给用户更好的体验，首屏组件加载速度更快一些，解决白屏问题

常用的懒加载方式有两种：Vue异步组件和ES中的import(最常用)

1.Vue异步组件实现懒加载

routes: [

{

path: '/',

name: 'HelloWorld',

component: resolve=>(require(["@/components/HelloWorld"],resolve))

}

2.ES中的import

const HelloWorld = ()=>import("@/components/HelloWorld")用来代替传统的写法

1. HTML5中新增特性

语义标签,增强型表单,视频和音频,Canvas绘图,SVG绘图,地理定位,拖放API,Web Worker

Web Storage,WebSocket

1. 是否自己封装过组件

封装过:分页器以及日历

1. 已经学过的组件通信有那些？

总共学了六种组件通信方式

1. Pros

适用于的场景：父子组件通信

注意事项：

如果父组件给子组件传递数据（函数）：本质其实是子组件给组组件传递数据

如果父组件给子组件传递数据（非函数）：本质就是父组件给子组件传递数据

书写方式：3种

[‘todos’] [type:Array] {type:Array,default:[]}

小提示:路由的props

书写形式：布尔值，对象，函数形式

1. 自定义事件

适用于场景：子组件给父组件传递数据

$on与$emit

1. 全局事件总线$bus

适用于场景：万能

Vue.prototype.$bus=this;

1. pubsub-js,在React框架中使用比较多（发布与订阅）

适用于场景：万能

1. Vuex

适用于场景：万能

1. 插槽

适用于场景：父子组件通信------（一般结构）

默认插槽

具名插槽

作用域插槽

1. 事件注意事项？

事件：系统事件：click,双击,鼠标系列等

自定义事件

1. 什么是后台管理系统项目？

注意：前端领域中，开发后台管理系统项目，并非是java,php等后台语言项目

需要知道的是，之前开发的项目【尚品会电商平台项目】针对的是用户（游客）---可以让用户在平台当中购买产品，但是需要知道的是，用户购买的产品信息从何而来呀？

比如：前台项目中的数据来源于卖家，但是需要注意，卖家不会操作数据库操作，对于卖家而言，需要把产品的信息上传于服务器，写入数据库

卖家并非程序员，不会数据库操作（增删改查），导致卖家，找了一个程序员，开发一个产品，可以让我进行可视化操作数据库（增删改查）

卖家（公司）组成：老板，员工

老板：开发这个项目，对于老板而言，什么都可以操作【产品的上架，产品的下架，查看员工个人的业绩】

员工：查看个人业绩，不能对产品进行上下架

后台管理系统：可以让用户通过一个可视化工具，可以实现对数据库进行增删改查的操作

需要注意，根据不同的角色（老板，员工），看到的，操作内容是不同的

对于后台管理项目，一般而言是不需要注册的

1. 后台管理系统模板？

简洁版:<https://github.com/PanJiaChen/vue-admin-template>

加强版:<https://github.com/PanJiaChen/vue-element-admin>

1. 后台管理系统API接口在线文档

<http://39.98.123.211:8170/swagger-ui.html>

<http://39.98.123.211:8216/swagger-ui.html>