1. Vue params 和 query 传参的 区别
2. 使用query传参的时候，name，path都可以引入，但使用params传参的时候只能使用name进行引入。
3. 接收参数的时候使用this.$route.params.id或者this.$ route.query.id

3)进行路由跳转的时候，使用this.$ router.push('路径')

4)如果index.js配置路由时，我们能看到，params的参数是URL不可或缺的一部分，但是query的参数是拼接起来的，没有也不影响

this.$router.push({

path: '/cart',

query:{ goodsid: 123456 } });

this.goodsid = this.$route.query.goodsid; //123456

2.$route和$router的区别

$ router是路由操作对象，只编写要跳转的路由

$ route激活路由信息对象，只读对象（参数的获取）

3.谈谈Cookie在客户端和服务的使用

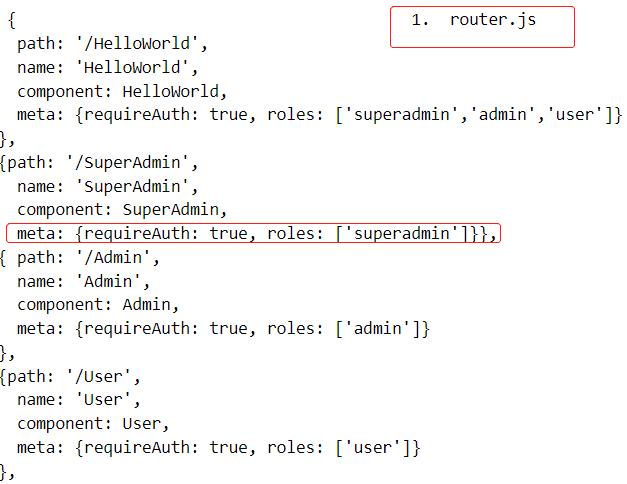
定义:用来给无状态的http请求增添状态标记，供服务端辨识,也称客户端缓存

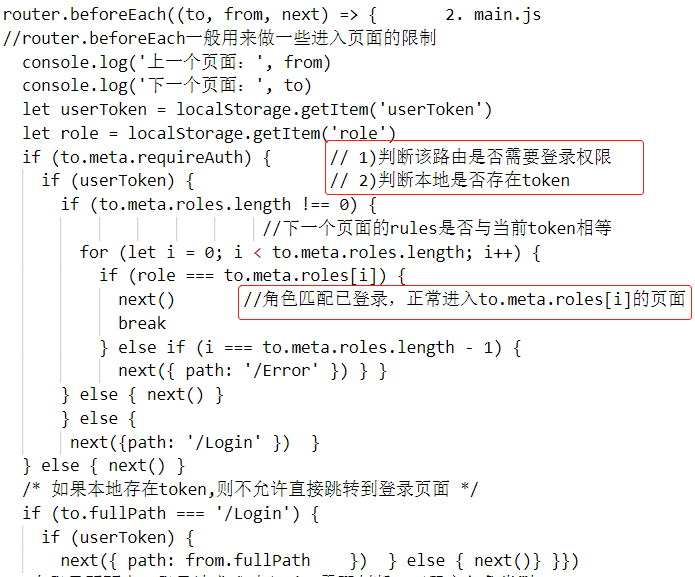
客户端:可以设置、更改、删除cookie

cookie的使用 :客户端发送http请求是会携带cookie（凭证）到服务端。

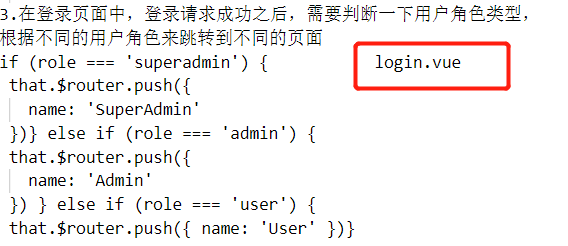
默认携带:ajax请求（不跨域） jsonp请求

for in 中数组的键值为下标

1. 使用 vue-router 之导航守卫 + meta 实现不同的用户角色具有不同的页面访问权限的功能
2. meta字段（元数据） 直接在路由配置的时候，给每个路由添加一个自定义的**meta对象**，在meta对象中可以设置一些状态，来进行一些操作,适合做登录校验
3. 



有token 说明登录过就有rules,没有就去登录



5.解决HTML加载时，外部js文件引用较多，影响页面打开速度问题

通常HTML文件在浏览器中加载时，浏览器都会按照<script>元素在页面中出现的先后顺序，对它们依次加载，一旦加载的js文件数量过多，就会导致页面展示延迟。传统中js加载的做法，所有的<script>元素都应该放在页面的<head>元素中，  这种做法的目的就是把所用的外部文件（包括CSS文件和js文件）的引用都放在相同的地方。可是，在文档的<head>元素中包含所有JavaScript文件，意味着必须等到全部JavaScript代码都被下载、解析和执行完成以后，才能开始呈现页面的内容。而对于需要很多JavaScript文件的页面来说，无疑会出现页面展示延迟现象。下面我给出3种解决方案：

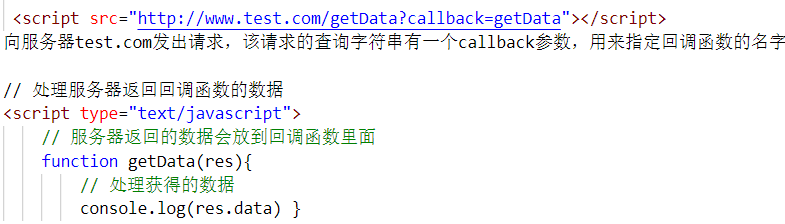
1. 改变标签的位置   把<script>元素引用放在<body>元素中
2. 在<script>元素中加延迟脚本:  HTML4.01为<script>标签定义了defer属性。这个属性的用途是表明**脚本在执行时不会影响页面的构造**。也就是说，**脚本会被延迟到整个页面解析完成后再运行**。因此，在<script>元素中设置defer属性，相当于告诉浏览器立即下载，但延迟执行。



1. 在<script>元素中加异步脚本:HTML5为<script>元素定义了async属性，这个属性与defer属性类似，都用于改变处理脚本的行为。async与defer类似，**只适用于外部脚本文件**，并告诉浏览器立即下载文件。但与defer不同的是，标记为async的脚本并**不保证按照指定它们的先后顺序执行。**

6.[vue解决跨域问题](https://www.cnblogs.com/lihaohua/p/12372267.html)

1)使用jsonp实现，网页通过**script标签向服务器请求json数据**，服务器受到请求后，将数据放在一个**前端指定名字的回调函数**的参数里面传给前端。



2)使用Jquery ajax实现

3)在vue开发中实现跨域：在vue项目根目录下找到vue.config.js文件（如果没有该文件则自己创建），在proxy中设置跨域

在vue中使用proxy进行跨域的原理是：将**域名**发送给**本地的服务器**（启动vue项目的服务,loclahost:8080），再由本地的服务器去请求**真正的服务器**。

7.谈一谈你对vue中MVVM的理解

M就是：模型（Model）：数据模型；负责数据存储。泛指后端进行的各种业务逻辑处理和数据操控，主要围绕数据库系统展开。

V就是：View 视图，负责页面展示，也就是用户界面。主要由 HTML 和 CSS 来构建

VM就是：视图模型（View-Model）：负责业务逻辑处理（比如Ajax请求等），对数据进行加工后交给视图展示

通过vue类创建的对象叫Vue实例化对象，这个对象就是MVVM模式中的VM层，模型通过它可以将数据绑定到页面上，视图可以通过它将数据映射到模型上

ViewModel 是由前端开发人员组织生成和维护的视图数据层。在这一层，前端开发者对从后端获取的 Model 数据进行转换处理，做二次封装，以生成符合 View 层使用预期的视图数据模型。

优点:低耦合。视图（View）可以独立于Model变化和修改，

可重用性。可以把一些视图逻辑放在一个ViewModel里面，让很多view重用这段视图逻辑

前后端分离，开发人员可以专注于业务逻辑（ViewModel）和数据的开发，设计人员可以专注于页面设计

1. 虚拟DOM(virtual DOM)

除了字符串模板，还可以使用**render渲染函数**来**创建HTML**，它更接近编译器

vue在使用render渲染函数在**内存中**生成与真实DOM相对应的**数据结构**,称之为虚拟DOM.

render函数语法格式：render(createElement){ **return** createElement(标签名,[数据对象],子元素) }

createElement()函数返回一个**对象**， 就是虚拟DOM => vnode

虚拟dom优点:

大量的操作DOM会影响性能，在更新数据后会重新渲染页面，这样造成在没有改变数据的地方也重新渲染DOM节点，这样就造成了很大程度上的资源浪费.当数据发生变化时,能够智能地计算出重新渲染组件的最小代价并应用到DOM上.



10.scoped的作用 实现组件的私有化，不对全局造成样式污染，表示当前style属性只属于当前模块

10、说一下你用过的几个UI(框架)库的区别

bootstrap: **响应式**（PC,平板，移动三合一）；缺点：性能，代码**冗余**

layui： 相比较bootstrap来说，**组件**更丰富，兼容比较旧的浏览器（IE）

Elementui: 基于 Vue 的桌面端组件库，快速开发 Vant：移动端UI库，分为vue版和小程序版

11.Object.defineProperty() 方法用于在一个对象上定义一个新属性，或者修改一个对象的现有属性，并返回此对象。注意：只能在Object 构造器对象上调用此方法，而不能在任意一个 Object 类型的实例上调用。

Object.defineProperty(obj, prop, descriptor)  
obj：目标对象，就是要在这个对象上面定义属性 prop：要定义或修改的**属性**的名称或 Symbol 。

descriptor：要定义或修改的属性描述符。是一个对象(数据描述符(value .writable)和存取描述符  
(getter 函数和 setter 函数所描述的属性)

通过赋值操作添加的普通属性是可枚举的，（for...in 或 Object.keys 方法），既可以改变这些属性的值，也可以删除这些属性。

Object.defineProperty()方法可以通过对属性的配置，实现更精确的控制

1. for…in和for…of的用法与区别

for...in遍历键名 for..of是遍历键值

for...in 遍历当前对象**可枚举属性**的键名（数组的index/对象的属性(**自身属性和从原型继承的**属性)）

for of是ES6提供的新方法**,遍历**当前可遍历对象拥有的**可迭代元素**(lterator)

如果**部署**了lterator遍历**接口**,就可以for..of遍历

[Array.Map.Set.String.函数的arguments对象.Nodelist对象,不包括Object]

for of不支持普通对象，想遍历对象的属性，可以用for in循环, 或内建的Object.keys()方法：

Object.keys(myObject)获取对象的实例属性组成的数组，不包括原型方法和属性

