复习

**nextTick**：

例如我现在的一个dom节点是依赖于某个数据的，而在数据更新之后，由于vue的dom更新是异步的，此时无法立即操作这个dom。因此现在可以使用这个nextTick的方法，把逻辑代码写到其内部，这样就可以操作数据变化之后的dom了

**为什么要把token同时存在于vuex和localstorage中呢？**

**首先vuex可以看作是管理数据的仓库，一旦发生数据变化，那么依赖这个数据的相应组件就会重新加载，这时候会导致某些数据丢失，所以此时需要在localstorage中也得存一份，这样一来的话就降低了页面刷新数据丢失的可能性。再一个vuex存在于内存当中，localstorage存在于磁盘中，相对于磁盘，电脑从内存中读取的速度更快，可以有效的提升效率，两者优势可以互补。**

**前端组件化与模块化的理解**

**组件化主要是针对ul设计界面来说的，比如一个登录页面是一个组件，一个注册界面又是一个组件，它是为了完成某些特定的功能。而模块化是针对于代码来说的，一个方法如果代码量很大，可以拆分成不同的子模块，分别放到不同的js文件中，再搭配导入导出链式调用，模块化最主要的优点就是避免了命名冲突，可以按需加载，便于维护，提高复用**

**登录的流程**

**登录的时候会发送axios请求，后端拿到这个请求之后会在自己的数据库内看看相关的信息是否匹配，如果不匹配，则返回相应的错误信息；如果匹配，后端会返回一个token值，这个token值后，会把这个token存放于vuex和localstorage里面，开始页面跳转。如果现在要跳转至某个具有登录功能的页面，需要判断当前的token是否存在，有的话就正常跳转，没有的话就重新跳转回登录页。一般把这个token放在路由守卫的模块里面。最后这个token一般也会放到登录跳转相关的请求拦截器当中，后端会判断这个token是否过期，如果过期，则会返回相应的错误码**

**十万条数据怎么渲染**

**可视化区域渲染，非可视化区域暂不渲染，一，准备一个div容器，固定高度，看看这个区域可以实现多少数据展示，用div高度除以单个数据的高度，二，再监听滚动，滚动条变化的时候，计算一下被卷去的数据高度，三，计算当前可视区域第一条索引，用被卷去的高度除以单条数据高度，在计算可视区域最后一条索引，再把来这两个索引中间的数据渲染到上面就行了**

**浏览器渲染机制**

**解析html生成dom树，解析css生成css规则树，合并dom树和css规则树生成render渲染树，那这个render渲染树只包括要渲染的节点和要渲染的样式信息，如果有display:none，就不会渲染在页面中。接着就要渲染页面，有两个概念，重绘和回流，回流表示操作dom的宽高等等，影响到页面布局，必然会重绘，而重绘不一定会回流。**

**Null和undefined区别**

**Undefined代表未定义，应该有值，但是却没给它赋值。而null代表空值，空引用。区别在于null转为数值是0，而undefined转为数值是NAN，而typeof null的时候是object，而undefined就是undefined。场景: 1- 比如声明了一个变量，但是没有赋值，相当于给这个变量留了一个位置 2- 函数的形参若没有值传递，则打印就是undefined，函数没有返回值，则函数返回undefined，3访问一个对象中没有的属性也是undefined**

**4- 原型链的终点是null， 以及操作一个不存在的dom也返回null**

**Data和props的区别**

**1.data里的数据我在当前组件中就可以定义，拿了就可以直接用，而props里的数据必须依靠来自父组件的传递，如果想要修改这个props数据，有时候会报错，因为props的传递过程是一个单选数据流，传递过来的如果是基本类型数据，那么在子组件内部修改就会报错，如果是引用类型，改变它其中的某一个属性不会报错。如果想对这个整体props数据作为一个比较大的改动又不影响父组件，则可以把这个对象深拷贝之后放到data里就行了**

**子元素继承父元素的行高需要注意什么**

**如果父元素的行高是具体某一个值，比如20px，那么子元素行高就是20px。如果父元素行高不加单位，那么子元素会把它理解成倍数，自己的font-size会乘这个倍数。百分比也是一样的**

**V-for 和 v-if谁更优先**

**V-for的优先级更高，同时每次v-for时都会执行v-if，这样非常消耗性能，那怎么办呢，分为两种情况，如果v-if的判断条件不依赖于外面的v-for，那可以在v-for外层包裹一个template标签，让v-if优先；如果依赖，就在computed计算属性里面把这些数据依赖全部过滤出来，生成数组，再循环这个数组就行了。此时就不需要v-if了，computed里的过滤成本远小于这个v-if的**

**如何判断一个对象为空对象**

**1.JSON.stringfy( )可以把一个js对象转换成 “ { }“，如果等于，那么就是空对象**

**2.Object.keys( ) 这个方法可以把对象里的属性名取出来作为数组返回，在判断这个数组长度就行了**

**3.for in 把每次遍历的属性或属性值push数组里面，如果这个数组为空，那么就是空对象**

**白屏渲染时间和首屏渲染时间？**

**白屏渲染时间等于用户输入第一个字符按回车，到浏览器出现第一个内容之间的时间，因此白屏时间等于页面展示第一个字符的时间节点，减去开始发送请求的那个时间节点；**

**而首屏渲染时间等于页面全部渲染的那个时间节点，减去开始发送请求的那个时间节点。**

**首屏时间等于白屏时间加上全部渲染完毕的时间**

**怎么样随机取出数组内的元素**

**调用random函数，因为random函数的返回值是0-1，所以要让这个返回值乘数组的长度，再向下取整就行了，得出来的返回值就是数组元素的索引值**

**UDP和TCP的区别**

**两者都是通向协议，也是传输层协议，但是两者的通信机制和使用场景不同**

**区别：**

**UDP是无连接的，发送数据之前不需要建立连接；而TCP是面向连接的，需要连接后传输数据，这里就涉及到三次握手。**

**UDP是不可靠传输，TCP是可靠传输。因为TCP用拥塞控制等机制，能够保证数据无差错传递，而没有类似这种的机制，所以不能保证数据的可靠交付**

**UDP是面向报文传输 TCP将数据看成是一串无结构的字节流，它是面向字节流传输的。**

**UDP支持一对多，多对多的交互通信；而TCP是一对一的两点服务**

**由此可见，TCP是可靠的传输协议，但效率慢；而UDP是不可靠的传输协议，但效率快。**

**TCP更重视通信数据的完整性和准确性**

**TCP的场景：重要文件的传输和邮件的发送；而UDP的适用场景是通信速度比较高，对数据的安全性和完整性要求较低的情况，例如视频电话，直播实时通信等等**

**Css隐藏元素的方法**

**display：none 它会让元素在页面上彻底消失，它原来占据的空间会被其他元素填补。因此会导致浏览器的重绘回流**

**Visibility: hidden 它消失后元素占据的空间依然保留，因此只会导致浏览器的重绘**

**Opacity: 0 透明度为0，元素完全隐藏**

**transform:scale( 0 )**

**什么时候z-index会失效**

**1.元素没有设置position属性，或者只设置为static时会失效。因为z-index只会用于于定位了的元素上。**

**2.如果元素同时拥有z-index和float浮动，此时z-index不生效。这时候只要清除浮动，z-index就能生效了。**

**3.如果父级元素z-index太小，不给力，那它子孙z-index再大也没用**

**Find和filter方法区别：**

**Find遍历数组返回第一个符合条件的具体内容，后面就不管了。filter筛选出所有符合条件的元素，返回一个新数组。两者都不会改变原数组。Findindex会查找到数组中符合条件的第一个索引值**

**计算属性Computed和侦听属性watch的区别：**

**Computed不支持异步，异步要放在watch或者methods当中；**

**computed有缓存，读取值的时候直接从缓存内读取，而watch没有**

**Watch有一些属性，例如deeply，immediately，新旧值**

**箭头和普通函数的区别：**

**1.箭头函数都是匿名函数 普通函数可匿名可具名**

**2.箭头函数不能new**

**3.this指向问题**

**4.箭头函数无argument伪数组、**

**5.箭头函数没有自己的显示原型**

**请求应该放在created里还是mounted里？**

**Created期间Vue实例当中的data和methods已经可以使用，但还没挂载到页面上。而mounted是挂载完成，这时候初始化页面已经完成，此时页面已经渲染出来了，可以进行dom操作。**

**放在哪都行，因为created和mounted是同步的，而请求是异步的，因此不会阻塞页面渲染的主线程，还是得看个人习惯。如果涉及到dom的操作，就要放在mounted，因为这时候页面才渲染完成。如果涉及到依赖数据的dom操作，则需要放到nextTick里面。**

**如判断一个元素是不是数组？**

**Object.prototype.toString.call( )**

**Array.isArray( )**

**arr . instanceof . Array**

**link和@import区别**

**1. Link是html标签，所以是HTML方式引入，而@import属于css的范畴，是css方式**

**2. @import对于一些老版本浏览器来说有局限性，而html不存在这个兼容性问题**

**3. 加载页面时，link标签引入的css会被同时加载，而import只会在页面加载完毕后加载**

**4. link支持js去控制dom来改变样式，，主要是js操作dom灵活插入标签来改变样式。而import不支持**

**标准盒模型与怪异盒模型区别？**

**标准盒子模型总宽度等于width+内边距+边框+外边距**

**怪异盒子模型总宽度等于内容+内边距+边框，无视外边框**

**可通过box-sizing来指定盒子模型，当其值为border-box时，就是怪异盒子**

**如果值为content-box，就是标准盒模型**

**em和rem区别**

**两者都是css中的相对长度单位**

**区别**

**rem是针对于根元素字体大小进行计算的，默认浏览器根元素字体大小都是16px，所以16px就等于1rem**

**但em是针对于当前字体或父元素大小来进行计算的，如果父元素设置了字体大小，那么子元素会默认继承。当当前元素或父元素大小改变时，又得重新计算，层级越多换算越复杂。而rem针对于根元素，有了rem 只调整html的根元素字体大小就可以达到所有元素的动态适配**

**Vue懒加载**

**Webpack打包的时候会把所有的js文件都给包起来，造成更新页面慢，耗时长，不利于用户的体验。路由懒加载就是我们在需要展示哪个页面就引入对应的组件，动态灵活的展示界面。页面白屏怎么办？首次加载js包太大了，导致白屏，路由懒加载就是一种优化方案**

**方式：**

**1.Vue的resolve异步组件，vue会把每个resolve组件单独打包成一个js组件**

**2.import，按需加载**

**3.webpack的require.ensure方法，只要被我们指定了的块名称的路由，那么这些相同的也会被合并单独的js文件**

**伪类，伪元素**

**Scoped原理**

**Scoped可以实现样式隔离，组件之间样式各自私有化互不影响。添加scoped之后就会给页面上的每个标签元素添加一个唯一标识，也就是自定义属性，data-v-xxx。同时对应的样式选择器也会添加上这个唯一的属性选择器**

**Vue中刷新页面的方式**

**1.原生js中的location.reload( )**

**2.借助vue中页面跳转的方式 this.$router.go( 0 )**

**3.provide/inject**

**Data为什么是一个函数？**

**如果data是一个对象的话，那么每个用到这个data的vue组件实例都会受到data改变的影响；如果data是一个函数的话，那么每次new一个vue实例时就能占用不同的堆内地址，改变其中一个data的值，其他组件内的data都不会受影响，提高了组件的独立性和可复用性**

**如何实现图片懒加载**

**如果请求数据返回图片地址，瞬间加载所有图片势必会增大服务器的压力。此时我们可以给图片的src属性赋一个默认的本地图片，等到要展示的图片进入可视区域，再来发送请求获取图片的具体地址，替换掉原来的默认本地图片即可**

**http和https区别**

**http明文传输，容易被外部劫持，安全性低。而https是http的加强版，加入了ssl协议，**

**这一协议放在了传输层和应用层，可以对通信数据进行加密，还能验证身份，使得传输变的更为安全**

**区别：**

**http端口是80，https端口是443**

**http传输是明文，而https是加密的**

**https需要申请CA证书，而http不需要**

**http协议速度比https更快 因为https要经过复杂的ssl握手，也就是安全相应的过程**

**For in 和 for of区别？**

**For in 可以遍历出对象身上的属性，而for of只能遍历带有literable接口的对象。For in 能遍历出数组的索引，而for of能遍历出数组的元素值**

**Literable：map ，set等等。可以通过object.keys( )==>数组，方法来搭配for of使用**

**For of内部都可以使用break来终止循环 。 for in可以使用contiue。**

**HTTP状态码**

**见桌面**

**获取js数组最后一个元素**

**数组的Pop( )，删除并返回该元素**

**数组的length-1方法**

**arr.slice( -1 ) -n负数代表从数组最后一个元素截取n个元素**

**arr.at( -1 ) 获取数组最后一个**

**reverse翻转数组，再获取arr[0]**

**Ref的作用**

**获取当前页面的dom元素，做一些操作，this.$refs.test**

**可以获取子组件身上的数据和方法 vueComponent对象**

**如何判断js数字是小数**

**1.先把这个数字通过toString方法转化为字符串，再利用indexof方法查看里面是否有小数点，如果有就会返回相应的索引，如果没有就会返回 -1**

**2.如果想要获取小数点的具体长度，可以这样做，利用split方法把小数点前后的部分作为元素存放在数组，再利用arr[ 1 ]就可以拿到了**

**Vue中如何实现echart自适应**

**监听窗口的变化，在调用echarts提供的resize方法即可**

**.伪类伪元素**

**伪类**

**a:hover, 这个 :hover就是一个伪类，:visit，代表如果访问这个链接了，那么再回来看时，原链接就变颜色了。:nth也是伪类，相当于设置一个元素的状态**

**伪元素**

**p::before{**

**content: '干嘛';**

**border: solid 10px red;**

**}**

**p::after{**

**content: '几级';**

**}**

**Bfc：**

**来说就是，BFC是一个完全独立的空间（布局环境），让空间里的子元素不会影响到外面的布局。**

**怎样触发BFC\n这里简单列举几个触发BFC使用的CSS属性**

**overflow: hidden**

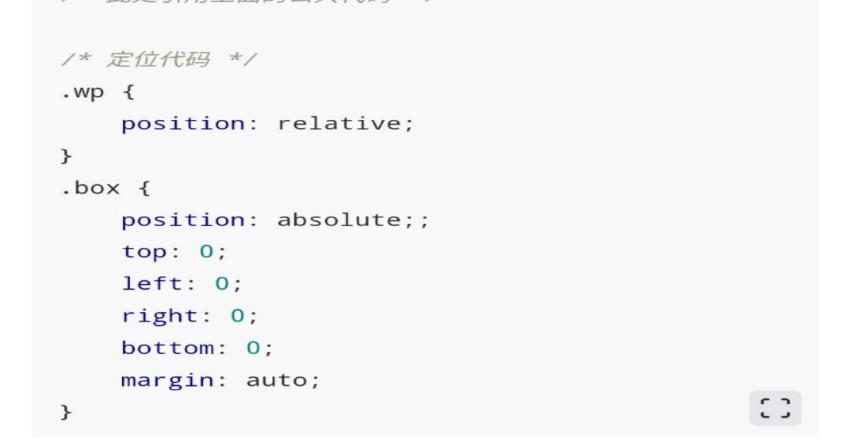
**display: inline-block、 flex**

**position: absolute、 fixed**

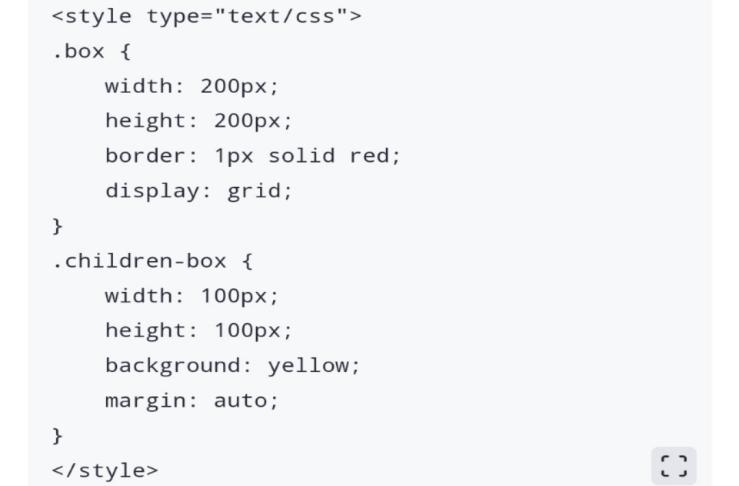
**居中**

**父position : relative 子position : absolute left :50% top:50% transform:translate(-50%,-50%)**

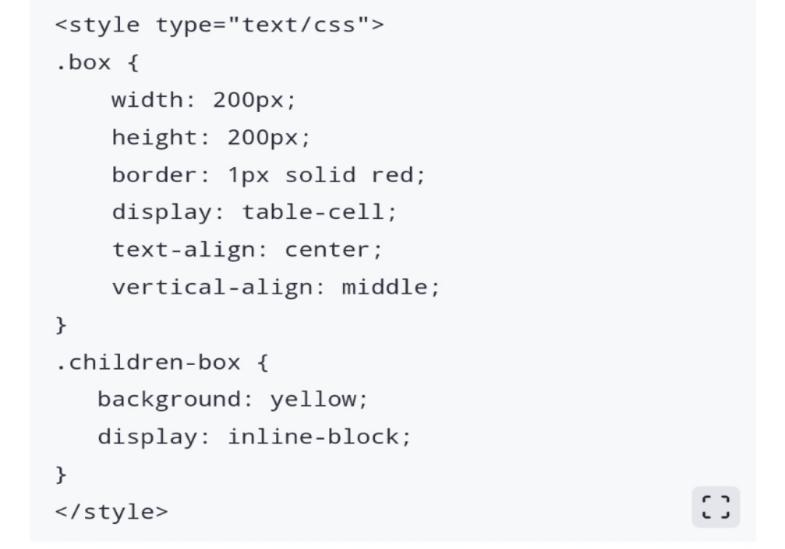
**Flex display : flex justify-content:center align-item : center**

****

**grid**

****

**Table-cell**

****

**合并数组**

**[...arr1, ...arr2]**

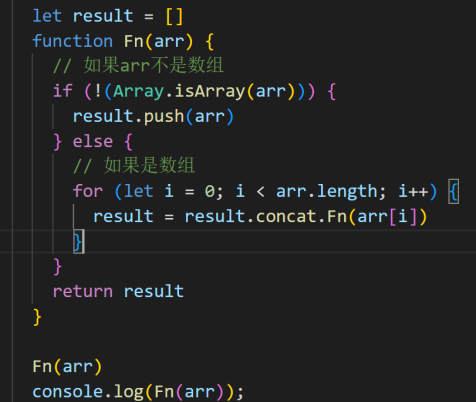
**array1.concat( array2 ) 或者 [ ].concat(arr1,arr2)**

**arr2.push(...arr1)**

**for循环逐个push**

**数组扁平化**

**1. console.log( arr.flat(3) ); 3是第三层 不知道具体层数，可以用Infinity**



**延迟加载JS的方式有哪些？**

**defer和async**



区别：

Defer：等html全部解析完，才会执行js代码，顺次执行

Async：async和html解析是一起的，不是顺次执行js脚本（谁先加载完毕谁执行）

Slice

<script>

let arr = ['a', 'b', 'c', 'd','e']

console.log(arr.slice(1,3)); //b c 从b开始截取，但d截取不到

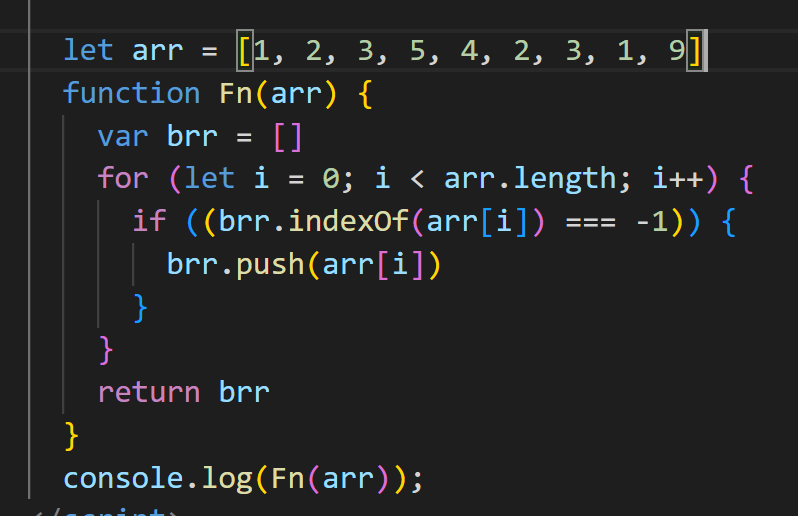
</script>

Slice不会改变原数组 splice会

数组去重

1

1

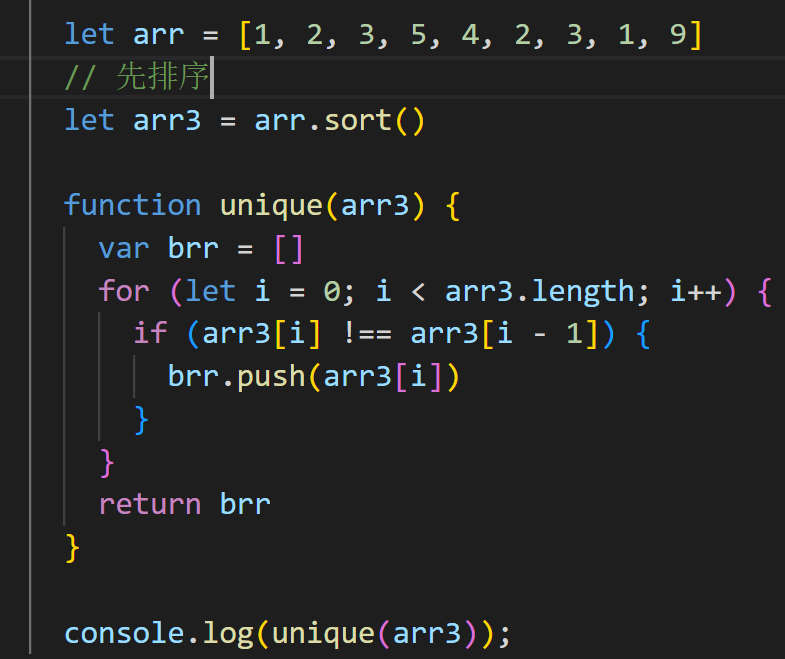




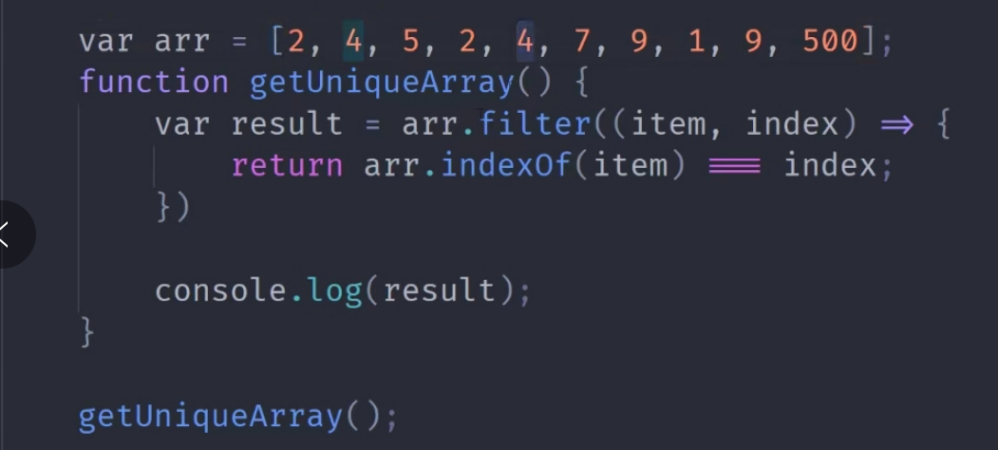
2

new Set([...{ }] )返回对象，再在数组里面展开就行了

3



4



new操作符具体做了什么

1. 创建了一个空的对象

2. 将空对象的原型，指向于构造函数的原型

3. 将空对象作为构造函数的上下文（改变this指向）

4. 对构造函数有返回值的处理判断 如果构造函数执行返回基本类型数据，那么打印的时候会忽略掉；如果返回的是对象，那么打印的就是一个对象

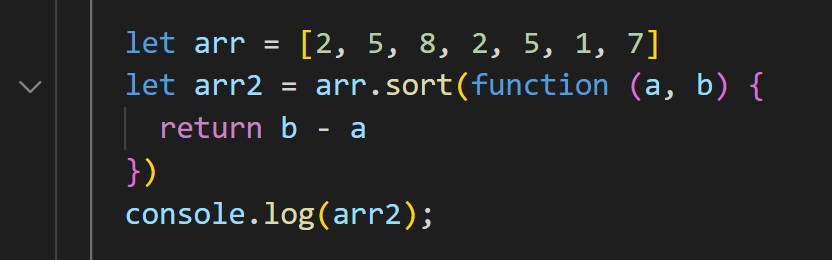
创建对象的方式 let obj = { } Object.create( ) new Object( ) 构造函数constructor

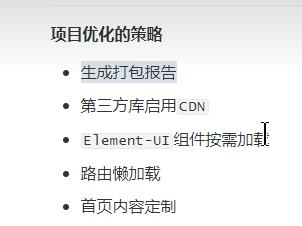
Keep-alive

有include和exclude属性，前者是告诉哪些要被缓存，后者是哪些不被缓存

排序

Sort





1



2



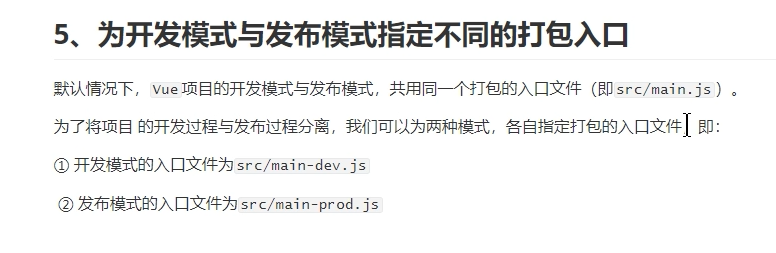
3



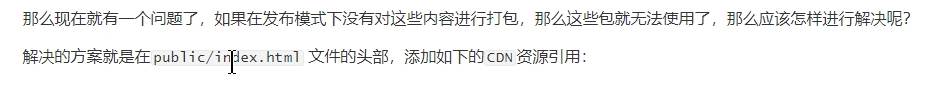
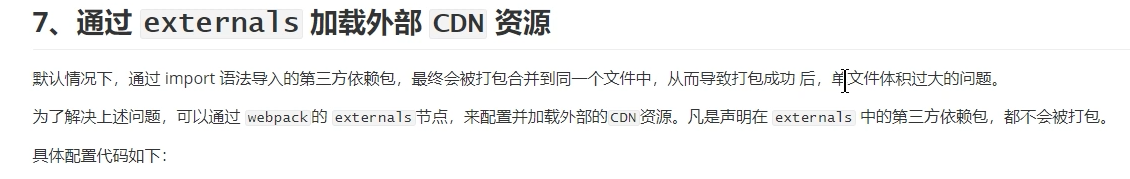
4



5



6



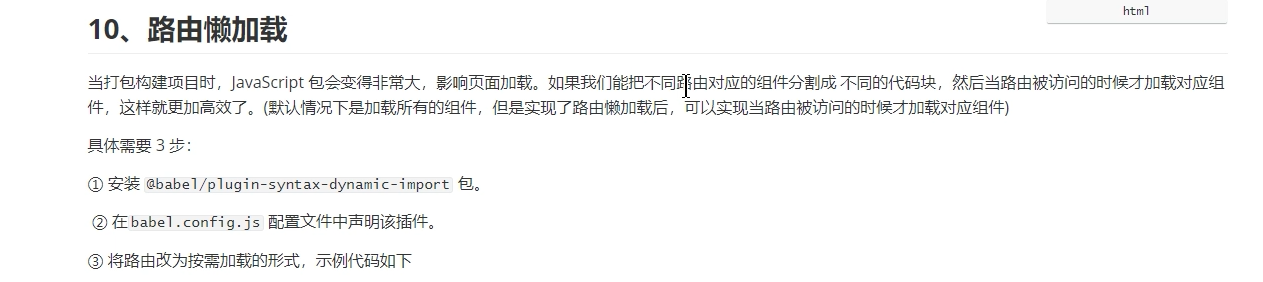
7



8

首页内容定制

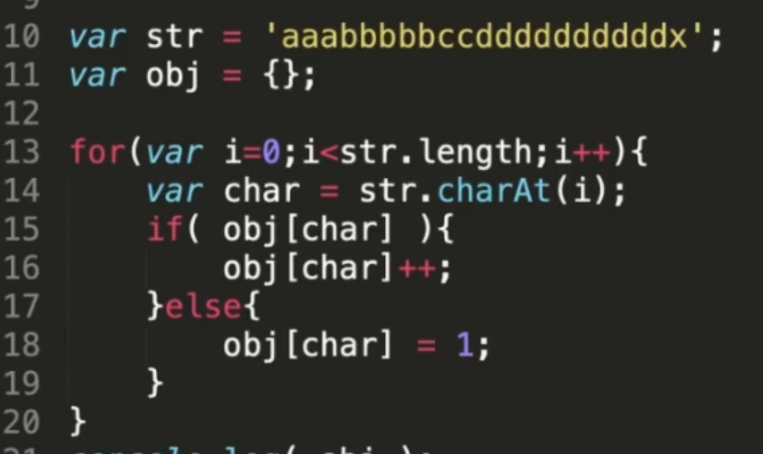
9

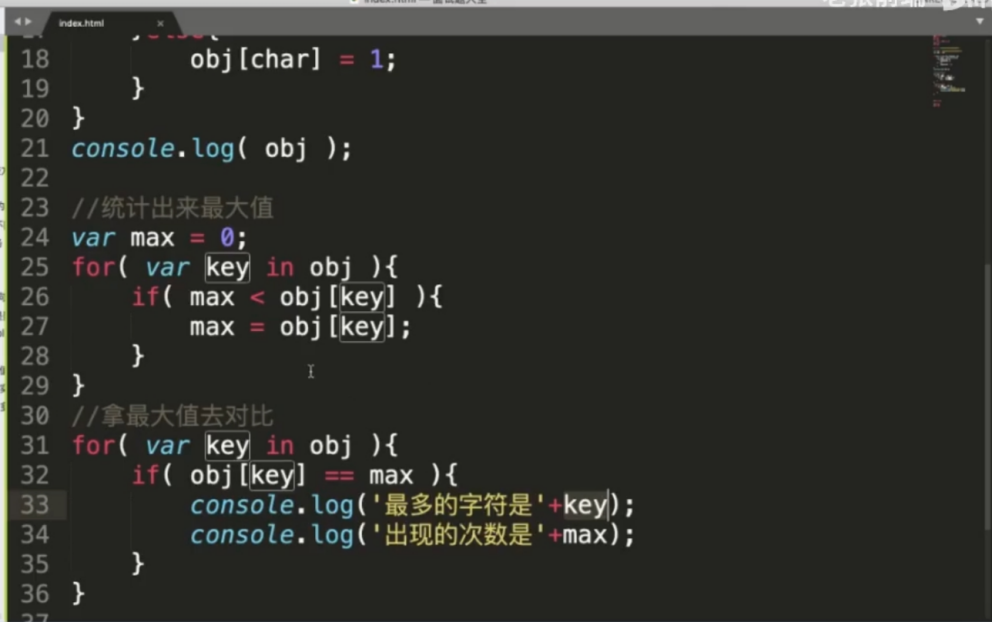


10



找出字符串中出现次数最多的字符





Vue常用指令

v-on:对节点进行方法绑定，函数必须写在methods里面 简写成@

v-bind动态绑定:及时对页面的数据进行更改，简写成 :

V-slot:组件插槽

v-for:根据数组的元素的个数循环生成相应的标签

v-show：显示内容

v-if：显示与隐藏 v-else

v-text：解析文本

v-html：解析html标签

v-model：数据双向绑定

v-for为什么要用到key

key可以提高虚拟dom的更新效率，在vue中 默认“就地复用”策略，在dom操作的时候，如果没有key，会造成选项错乱，key只能是字符串或者number，说的再详细一些，就是key会根据当前dom是否已被修改，如果没被修改，那就直接复用当前这个旧元素。要注意尽量不要在数组中使用添加删除的操作，因为这样会让vue认为后面所有的元素都发生了变化，导致多余的对比和原地复用，影响效率。

路由的几种模式：

History模式和hash模式

区别：1.关于找不到当前页面发送请求的问题

history会给后端发送一次请求，而hash不会

2.关于项目打包前端自测问题

Mode为history的 项目打包生成dist的时候打开对应的页面是打开之后无法显示的，而hash可以正常显示

3.关于表象不同

History有/ hash有#

判断图片是否渲染完

图片节点添加一个onload属性，当触发完后会执行里面你自定义的function函数

调用图片节点的complete的属性，当触发完后会执行里面你自定义的function函数

Onreadystatechange事件也可以

输入网址后发生了什么？

**Url输入：**

浏览器会对输入的信息进行判断，检查输入的内容是否是一个合法的URL链接

如果合法，则判断输入的URL是否完整，如果不完整，浏览器会对输入的内容进行猜测，补全前缀或者后缀；如果不合法，则把输入的内容作为搜索条件，来进行搜索

把域名域名转换为ip地址 ------DNS解析过程

**DNS解析**：用户输入域名后操作系统会检查浏览器缓存和本地的host文件中是否有这个网址记录，有的话就从记录里面找到对应的IP地址，完成域名的解析；没有的话使用tcp/ip参数中设置的DNS服务器进行查询，如果查询的域名包含在本地配置区域资源中，有则返回解析结果，完成域名的解析；再没有的话会检查本地DNS服务器是否缓存有该网址记录，有则返回解析结果，完成域名的解析；还没有的话，权威的dns服务器就会把这个url转换为ip地址，完成域名解析

**发送请求简历TCP连接**：三次握手 四次挥手

**通过HTTP或https发送数据请求**

**如果使用https，会在TCP和HTTP之间多添加一层协议，作为加密及认证的服务；https使用ssl和tls协议，保障了信息的安全**

**服务器端响应请求**

**当浏览器到web服务器的连接建立之后，服务器端响应请求，发送相应的资源**

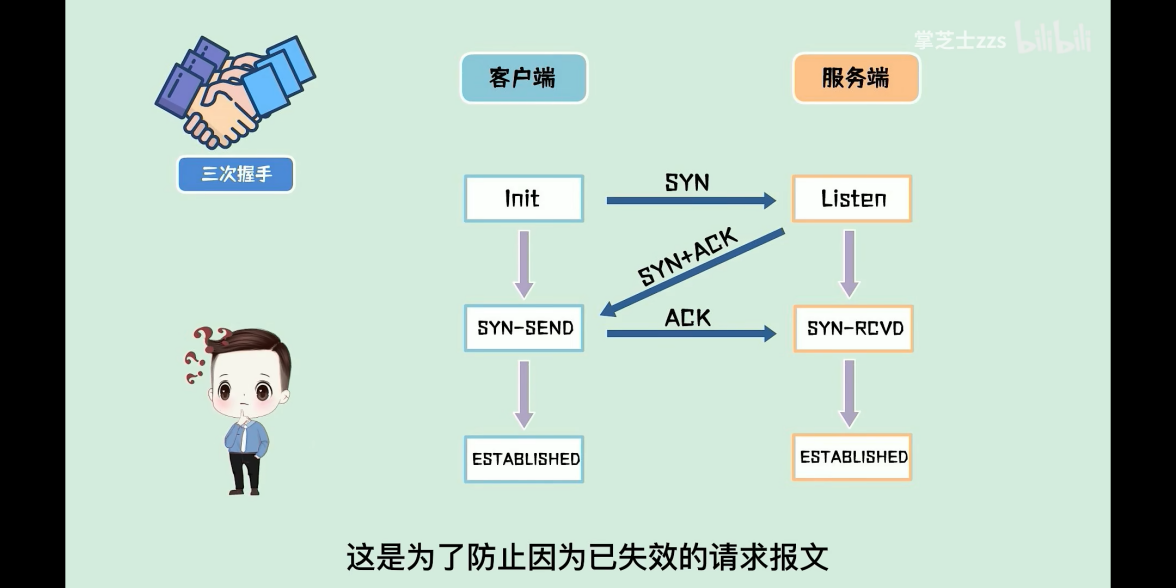
浏览器解析渲染页面

处理html标记并构建dom数，再处理css构建cssom数，将dom树和cssom树合并为render树，根据渲染树来布局，计算每个节点中的几何信息，再将内容渲染到屏幕上

http请求结束，断开tcp连接

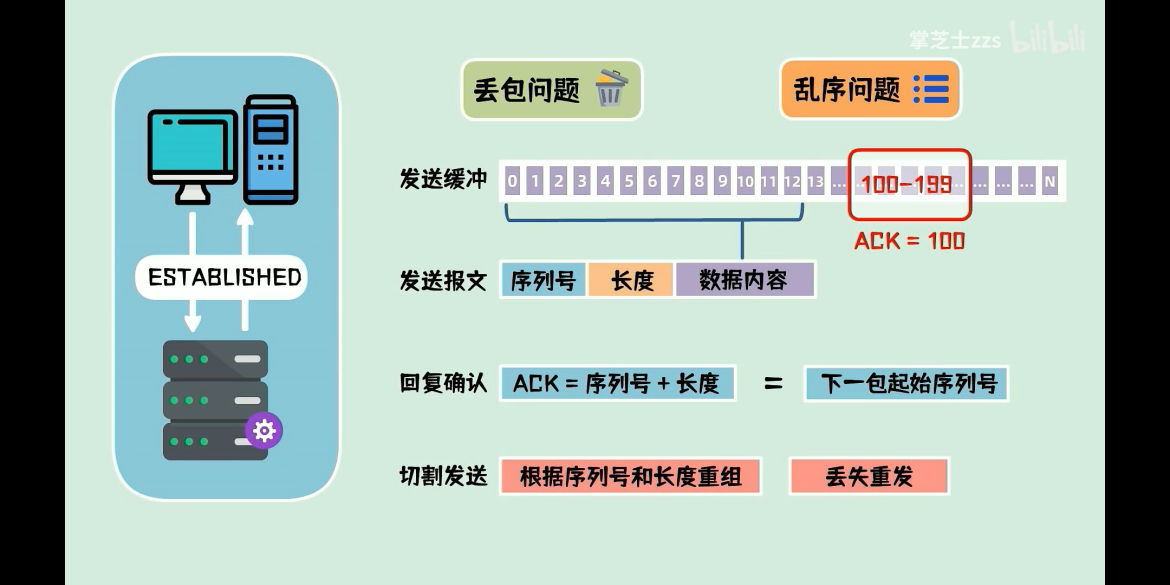
三次握手四次挥手

客户端向服务端发送SYN包，服务端同意连接后，向客户端发送SYN和ACK包，客户端收到后，又向服务端发送ACK包，至此连接建立，在这个过程中发送了三包数据，所以称之为三次握手

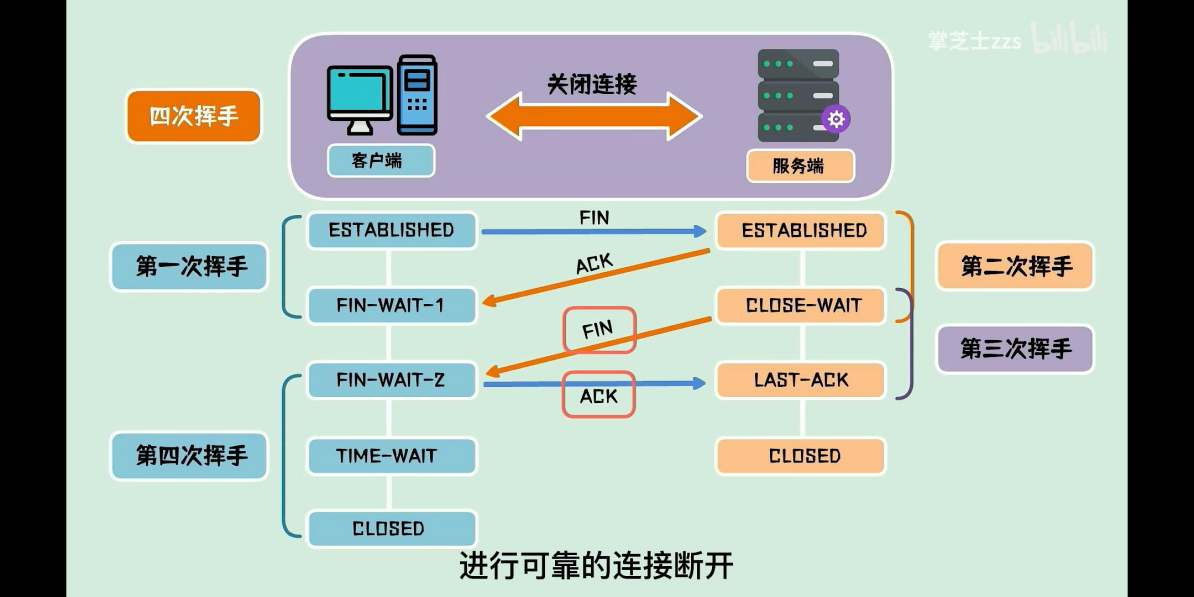


那么为什么不能两次握手呢？这是为了防止已失效的报文突然又传到服务器引起错误。

三次握手之后客户端和服务端就进入了数据传入状态



四次挥手：（tcp）



Udp：udp协议非连接，并只把数据简单封装一下从网卡发出去就可以了，并不能保证是否丢包

Get和post区别

最常见的数据请求方法就是get和post

1.get请求参数是放在url里面，安全性差；而post的请求参数是放在请求体里面，安全性好

2.get请求可以被浏览器缓存，而post请求不会被浏览器缓存

3.因为get请求是放在url里面的，url长度受限，最大就是2048个字符，而post不会受限

4.get请求可以通过浏览器直接访问，支持刷新和后退；post不能被浏览器直接访问，刷新后数据需要重新传递

垃圾回收机制

垃圾回收机制主要针对一些字符串，对象等等，只要这些变量不再使用，那就可以对它进行回收。

首先，js环境中的分配的内存，一般有如下的生命周期：  
1.内存分配：当我们声明变量、函数、对象的时候，系统会帮他们分配内存

2.内存使用：即读写内存，也就是使用变量，函数等

3.内存回收：使用完毕，由垃圾回收自动回收掉不再使用的内存

因此判断一个变量或函数会不会被回收，只看他会不会被使用就好了

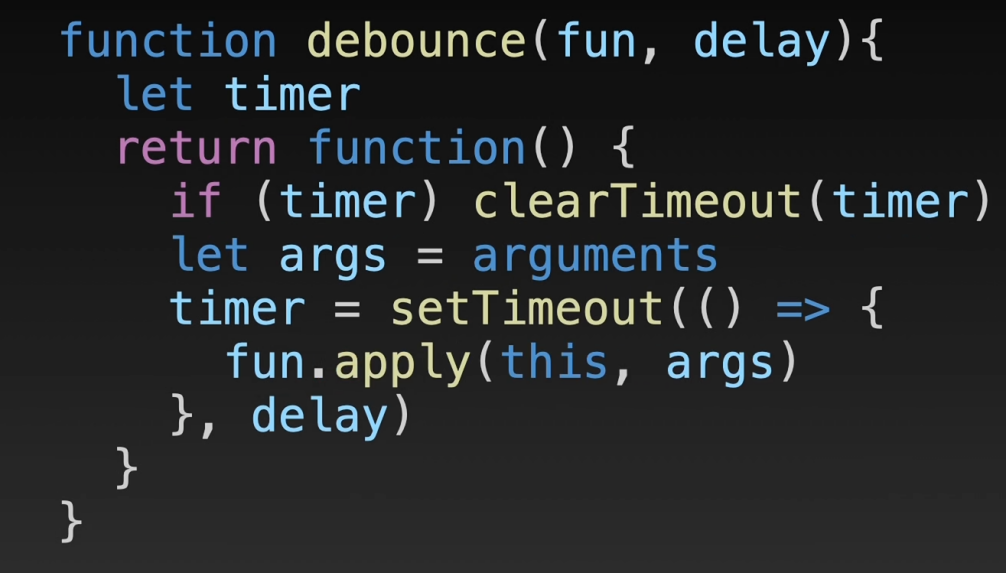
垃圾回收机制的两个算法

引用计数法：

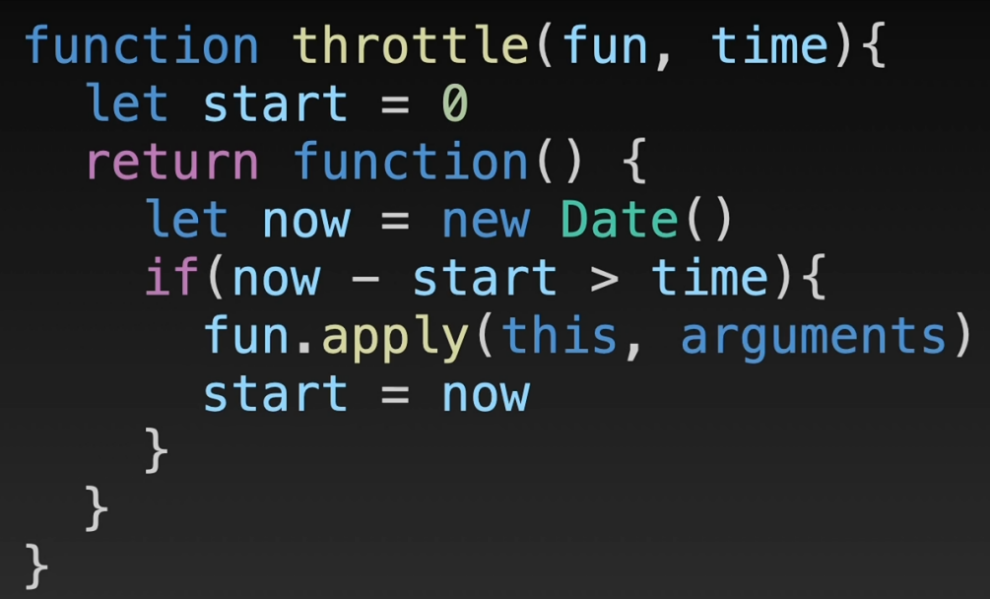
看看一个变量到底被引用了多少次，引用一次加一，多次引用那就进行累加；减少一次引用就减一，直至减为零，此时被回收。Bug：多个对象相互指引的情况下变量无法被回收，解决这个问题要靠标记清除法

标记清除法：

从根部出发，如果能找到某个变量，说明它会被使用，此时会被标记上，不回回收；如果找不到某个变量，那执行完毕之后就会进行回收

防抖：  


节流：



父组件给子组件传值，为什么子组件接收不到？

这是因为子组件在初始化的时候可以正常接收来自父组件传递过来的值，但当父组件内的某个值改变的时候，并没有同步更新到子组件内部，此时给子组件内部添加一个watch监听器就可以了；第二个可以在父组件内子组件的值改变的地方，绑定一个唯一标识key，当key发生变化时，子组件就重新渲染，就可以与父组件中的相应的值保持同步更新了

Async就是异步操作的语法糖

Async将函数标记为异步函数，这里的异步函数就是指返回值为Promise对象的函数，在这个函数内部我们可以使用await来等待异步函数并返回最终的结果值，简化了原来的then的链式调用，最后就是await必须和async同时存在

分页器

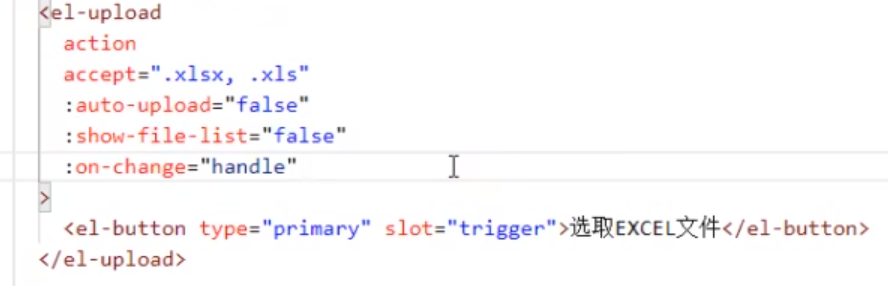
分页器的几个基本要素：当前所处页，每一页展示的数据，数据总条数 以及有多余条数要新增页数 和连续页码数（一般是5或7）

要计算总共多少页，并向上取整

计算出连续的页码的起始数字和结束数字

里面还有很多细节，比如v-for来循环渲染总页数；以及鼠标点击第一页的时候就不显示上‘上一页’按钮，点击最后一页就不显示‘下一页’操作

VueExcel表格上传解析和导出



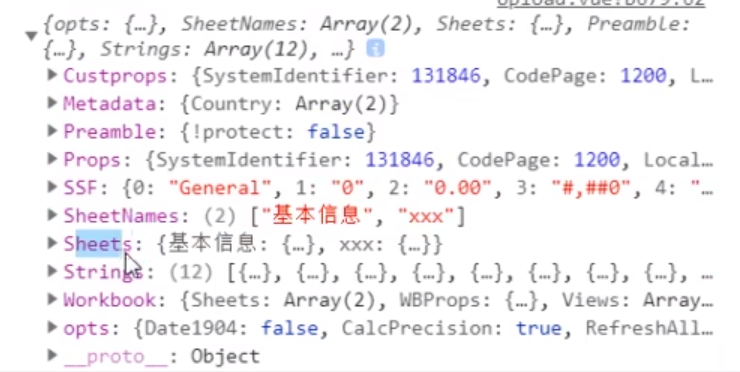
这里的upload组件里面必须要有action方法

通过upload方法中的一个ev.raw方法，拿到这个文件，把上传的文件先变为二进制数据data

引入xlsx组件，通过xlsx.read( )方法读取这个data，并创建相应的一个工作表



接着打印这个方法的返回值



里面有一些重要的信息，比如SheetNames页卡名，和表格详细信息sheets

接着在这个对象里面要拿到第一张分页表里的内容，调用xlsx里的sheet\_to\_json方法把这些内容变为JSON格式数据

再把里面的一些关键信息通过字段对应表做一个映射关系，主要是通过循环遍历的方式，比如把汉字的‘姓名’替换成name，把这些处理过的对象方在数组里面再给data中的变量供table渲染

搜索功能：根据用户输入的关键字检索到相应的内容做到懒加载展示，运用防抖限制用户频繁操作

此时可以做一个判断，判断我们这个数据到底有没有table组件拿到，如果没拿到的话就让它隐藏；拿到的话再显示

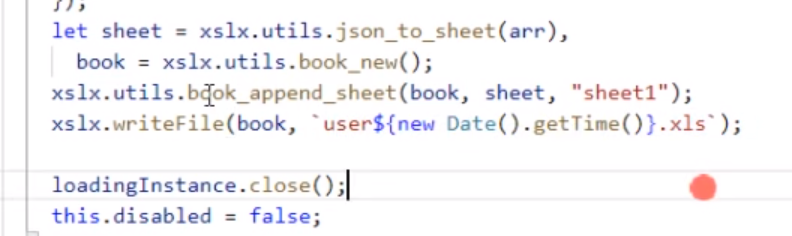
接着点击按钮，把这个data数组传递给服务器，可以先判断这个数组长度为不为空，如果为空，那么直接返回；否则直接传给服务器就可以了。此时再做一个判断，看看服务器接不接受一次性全部上传，如果接收直接传递即可，否则的话要一条条上传，也就是把这个数组内的元素从索引值为0的那一项上传就可以了，上传成功就让这个索引值+1。

如果这个索引值比数组长度还大的话，直接返回就行了。这里也可以做一个防抖，比如用户不停的点击上传按钮时，做一个操作的限制。

针对不同角色实现动态路由的生成，以及实现按钮权限相关的限制性操作

接着导出excel

把原先data数组中的JSON元素全部变成sheet格式数据，再利用xslx中的一个方法新创建一个表格，再把这个sheet格式数据给插入到这个表里就可以了，最后再利用xslx中的writeFile方法把这个book内容写入到指定文件就行了



大文件分片上传

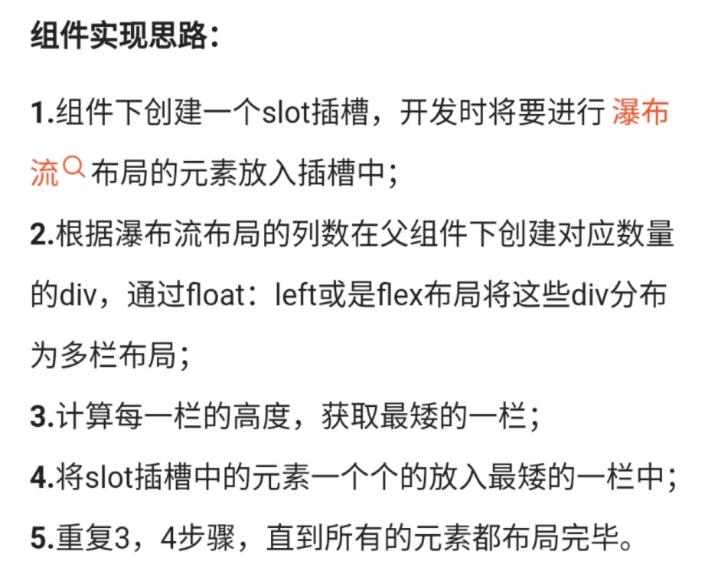
对类型文件类型做判断，打印并获取到文件的name和size大小 和type类型

为什么要拿到name？因为有name才能拼接到新的name，如果两份上传的文件名相同的话，就会出问题 type类型是后端要对这个文件类型做判断

**微信小程序支付业务：**

在小程序官网完成注册，需要执照，并把小程序关联到商户号，授权JSAPI权限

Vue中流式瀑布 npm i vue-waterfall2@latest --save



防抖节流

懒加载：先可视区域图片，等到鼠标滚动到某一个区域再次发送请求获取数据加载图片，期间要判断图片是否已经获取，没获取的话用本地图片就行了

父子组件渲染生周期顺序

父created 父beforecreated 子created 子mounted 父beforeMount 父mounted

只有在子组件挂载完毕后，父组件才算是真正完成mounted

混入mixin：

把不同组件内的相同配置提取成一个混入项，谁需要谁去引入加载就行了，格式是mixin[ ] 要注意如果这个混入项内的数据跟当前组件数据产生冲突的时候，以当前组件数据为准；但如果两边都有生命周期函数，那都会执行

v-model实现：

父子组件通过props传参，子组件内接收到数据利用:value=“xxx”显示，并且子组件再使用自定义事件，向父组件内传递数据修改数据，实现数据双向绑定

Sync修饰符

子组件内不能修改修改来自于父组件传递过来的props属性，如果想修改，可以使用自定义事件来修改。或者用在父组件内的添加sync例如：:title.sync=xxx ，也可以修改内容

作用域插槽

子组件插槽内部的动态数据没办法直接给父组件的自己使用，这时候需要在插槽标签内部动态绑定这个数据，父组件内使用template v-slot=“xxx”标签使用就可以了

Vue.use( )

某一个在main.js引入的组件可以被挂载到全局，当页面生成时会自动执行这个use里的内容。这样一来就不需要每个页面都引入注册相同的组件。深层原理就是vue会检索到并调用一个install方法，在这个方法内部我们利用Vue.component方法来循环遍历并注册相应组件，这样一来就能全局直接用，不需要引入了

Nginx反向代理

异步组件懒加载：

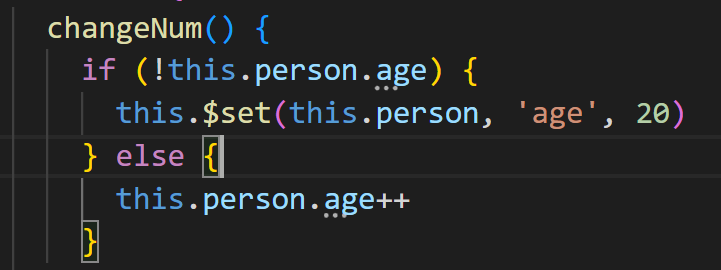
1.可以在父组件内的component里面懒加载相应组件



或者

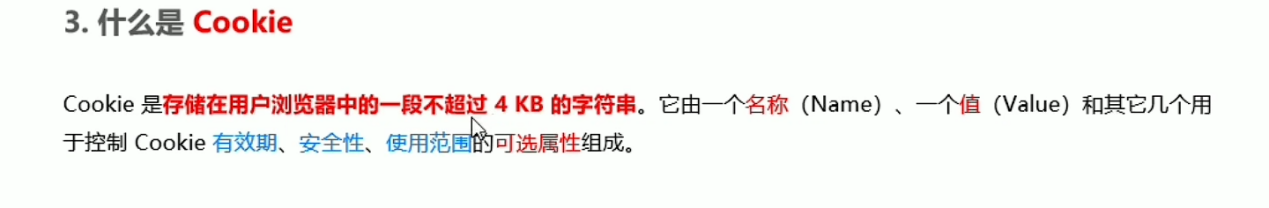


this.$set()



判断data数据中有没有age这个属性，没有的话就用this.$set( )动态生成一个就行了

Cookie



不同域名下的Cookie各自独立，每当客户端发起请求时，会自动把当前域名下的未过期的Cookie一同发送到服务器

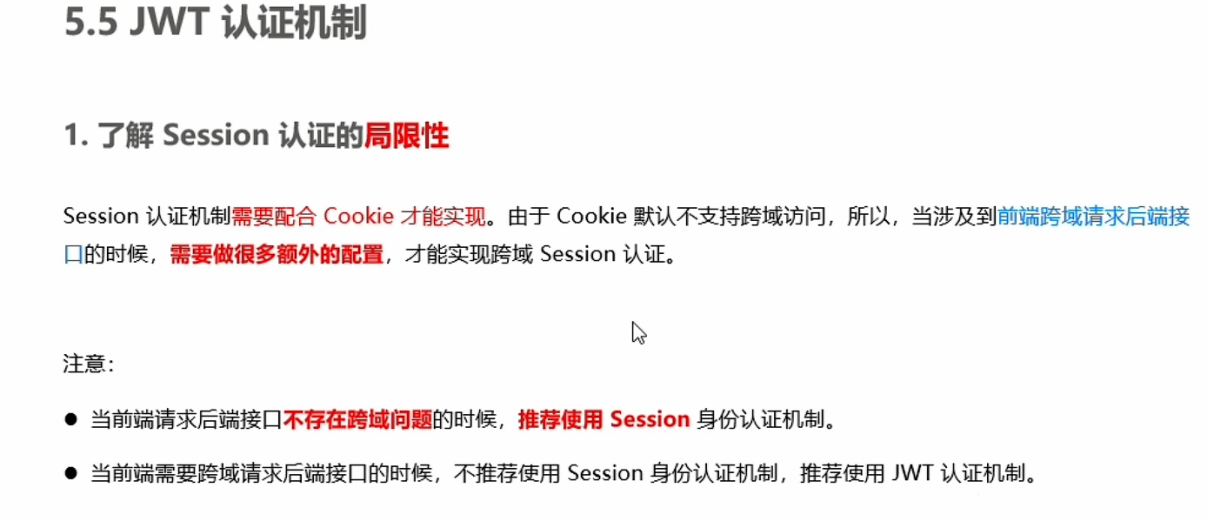
Cookie在身份认证机制中的作用：

客户端第一次请求服务器的时候，服务器通过响应头的形式向客户端发送一个身份认证的cookie，客户端会自动把cookie保存在浏览器中。

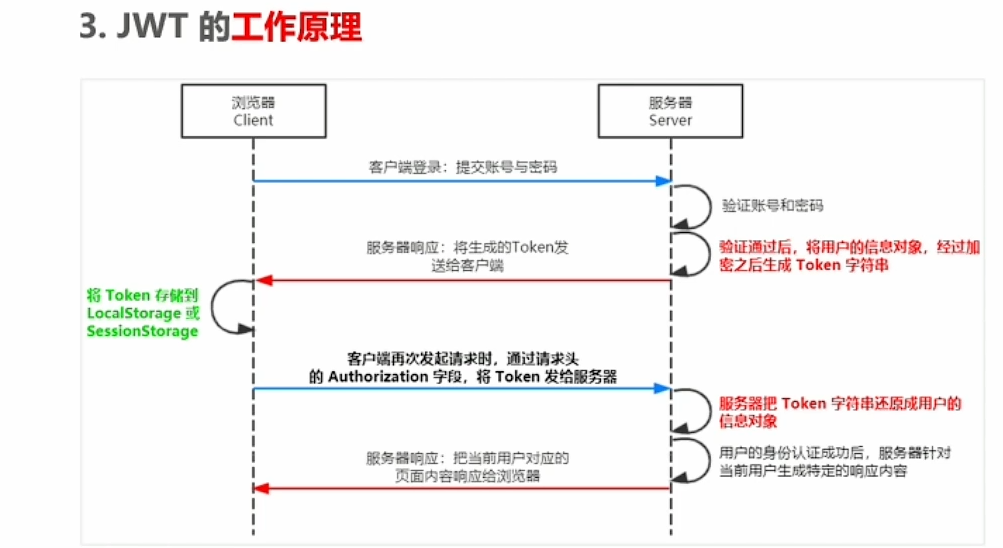
然后再当浏览器向服务器发送请求的时候，浏览器会通过请求头自动的将这个cookie字符串发送给服务器，通过验证后服务器就能返回给浏览器相应的内容。



这里注意，服务器端存储用户名信息，并且生成cookie，让客户端存放这个cookie



Jwt是目前最流行的跨域解决方案



JWT的三个组成部分，从前到后分别是Header，Payload，Signature

Payload部分才是真正的用户信息，它是用户信息经过加密之后生成的字符串

Header和Signature是安全性相关的部分，只是为了保证Token的安全性

Felx属性：

flex-direction：决定主轴方向

flex-wrap：定义是否换行

Flex-flow

Justify-content：定义项目在主轴上的对齐方式

Align-items:定义项目在交叉轴的对齐方式

省略号

white-space: nowrap; （ 默认 normal 自动换行）

overflow: hidden;

text-overflow: ellipsis;

cookie怎么样去存储

Document.cookie = ‘name=value’

下载cookie.js插件 接着可以使用

Cookie.set( ) //存值

Cookie.get( ) //取值

Cookie.remove( ) //删除

Session可以共享吗？

Session可以共享

数据通信

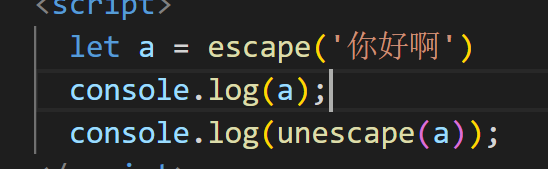


前端加密

Base64加密

Md5加密

escape加密 unescape解密



RSA加密，非对称加密算法

移动端适配解决方案：

1.百分比布局

2.响应式布局：通过@media实现一套html配合多套css实现适配

3.通过rem或者vw，vh，vmin, vmax等实现不同设备有相应的比例来实现适配

4.淘宝的无限适配flexible.js

5.postcss-pxtrem

Vue2和vue3的区别

1.Vue2 是ES5的API，ObjectdefinePropoty对数据进行劫持并结合发布订阅者模式来实现的

Vue3是ES6的API，proxy 对数据进行代理

2.vue3支持碎片化，不用像vue2一样必须有个盒子装着

3.vue3出现了组合式API，而vue2是选项型API

4.建立data的方式 vue2-是数据放在data属性中 vue3需要使用setup方法，

5.生命周期钩子不同

6.vue3新增了teleport组件

7.vue3使用TS写的，未来会更好地拥抱TS

v-xxx 自定义指令

directives：{

bind(newValue, Oldvalue){

...

}

}

beforeMounted:

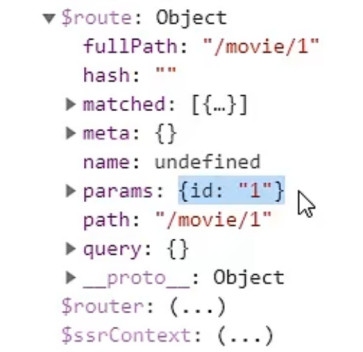
在这个函数中正在创建虚拟dom，即将渲染为真实的dom，此时也可以操作数据。

mounted函数中所有的节点以及全部加载完毕，所以在这个节点我们可以操作dom，或者操作数据。

Updated：

当页面数据改变后并且页面上渲染完相应的区域后，会执行这个updated，所有数据发生变化的时候都会执行它

this.$route是路由的参数对象



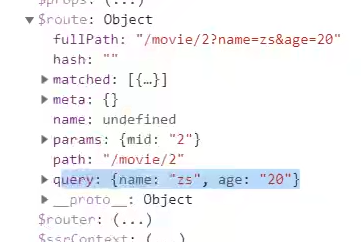
路由里的props : true

要想在组件里直接拿到路由里的动态参数，直接在路由后面加个 props : true 就行了

查询参数



打印：



this.$router声明式导航：

像 <a> 或者 <router-link> 是声明式导航

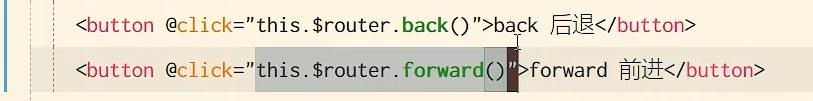
像this.$router.push/replace/go是编程式导航

this.$router是导航对象

this.$router.push( ‘hash地址’ ) 跳转到指定的hash地址，并增加一条历史记录

this.$router.replace( ‘hash地址’ ) 跳转到指定的hash地址，并替换当前的历史记录

this.$router.go( 数值 ) 前进后退，如果后退的层数超过上限，则原地不动 可以直接在按钮上使用



require与import的区别

1，require是CommonJS规范的模块化语法，import是ECMAScript 6规范的模块化语法；

2，require是运行时加载，import是编译时加载；

3，require可以写在代码的任意位置，import只能写在文件的最顶端且不可在条件语句或函数作用域中使用；

4，require通过module.exports导出的值就不能再变化，import通过export导出的值可以改变；

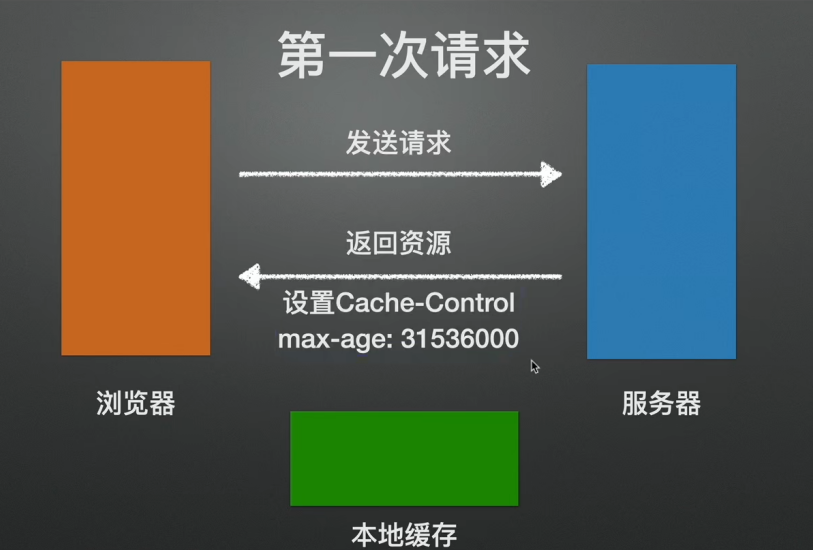
协商缓存



协商缓存是一种服务端的缓存策略，浏览器向服务器发送请求获取数据，如果不是最新资源的话，那么服务器就会正常返回数据并放置在本地缓存；

如果是最新资源，那么服务器就会返回304状态码，并且直接从本地缓存内部读取数据，耗时更低。

强制缓存



服务器响应数据式，如果觉得某些css，js文件可以缓存下来的，那它就会在响应头增加一个cache-control，并设置时长max-age，那么就会在本地缓存中存下相对应的文件资源。那下次如果再次发送请求，如果max-age没过期的话，就不会向服务器发送请求获取相应的资源了。如果过期了，就会查看这个缓存里面是否有Last-Modified和ETag，如果有，那就会把Last-Modified和ETag的值放在请求头当中。接着服务器会判断是否可以继续使用本地缓存，如果返回304，那就可以使用。如果是200，那就还是服务器返回数据。

宏微任务有哪些

宏务：setTimeout、setInterval，文件读取

微任务: promise.then、nextTick

ES8的async和await就是来简化Promise的操作

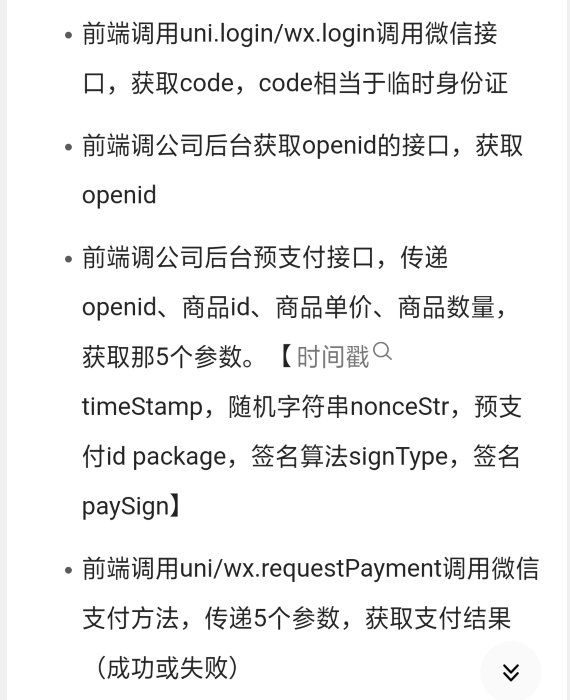
微信小程序定位

经纬度：在方法内部调用wx.getLocation( )这个API，接着打印返回值res内部的经纬度就可以了

地图位置：外面包一个form表单，再绑定一个事件，然后再wx.openLocation(Object object)

微信小程序支付

用户需要注册商户号，需要有营业执照、法人信息等



Vue3响应式实现原理:

通过 Proxy（代理）: 拦截对象中任意属性的变化, 包括：属性值的读写、属性的添加、属性的删除等。

通过Reflect（反射）: 对源对象的属性进行操作。

从定义数据角度对比

ref用来定义：基本类型数据。

reactive用来定义：对象（或数组）类型数据。

备注：ref也可以用来定义对象（或数组）类型数据, 它内部会自动通过reactive转为代理对象。

Promise:

Promise.all( ) 里面接收一个都是promise的数组，只有所有元素都成功，了这个方法才会成功，并且结果值就是五个Promise元素结果值组成的数组

Promise.race( )里面接收一个都是Promise的数组，当其中一个发生改变时，那race方法会立即返回这个优先改变的状态，结果值就是这个优先改变的值

当then里的箭头函数内部返回的不是一个promise对象时，那这个then返回是一个成功的promise对象；如果返回是一个失败的promise，就then就失败，失败值就是这个promise的失败值

Vue2数据响应式原理

对象

首先申明一个对象，接着把这个对象传到observe函数中

observe函数中会判断这个对象身上有没有\_\_ob\_\_这个属性，有就返回；没有的话会new一个Observer函数

Observer函数其实就是一个类，在这个类里面我们会调用def文件为这个大对象添加一个\_\_ob\_\_属性，属性值是一个对象

接着调用this.walk方法，开始为obj对象第一层的属性绑定数据响应式，此时调用了defineReactive文件，这个文件中会做一层判断，如果这个obj对象的子对象还是对象的话，那么直接再重复observe函数，直到都不是对象位置，然后依次跳出递归并执行下面的绑定数据响应式

数组：

此时应该调用 Object.setPrototypeOf(value, arrayMethods)方法，循环遍历Array身上原型的所有方法，把增加和删除的方法全部提取出来进行重写，内容主要是给数组实例身上添加一个\_\_ob\_\_属性，接着也要给这个数组包括其内部的子元素都绑定数据响应式。

收集依赖：

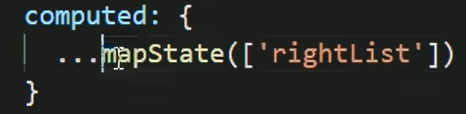
当编译器解析到{{ }}后，会执行new Watcher( )这一操作，Watcher函数中最重要的就是把当前Watcher实例赋给了全局Dep.target ，并且在我们试图访问这个属性的时候，它会把当前的watcher实例推入到dep的subs数组里面，再然后将 Dep.target 的值重新赋值为 null，表示退出依赖收集阶段。最后一旦我们调用了set方法，它会迅速调用dep.notify方法，循环遍历这个数组，触发很多连锁函数，来通知每个实例此时要发生变化了。

引入vuex的东西



调用vuex的mutation方法，并往state里面存放数据，供其他页面使用





v-if懒加载：

组件创建时v-show可以被创建，但不能显示

v-if 的初始化较快，但切换代价高；v-show 初始化慢，但切换成本低

后台管理系统权限方面（菜单，路由，按钮，控制台）

首先涉及到三个概念，用户，角色，权限。一个用户可以拥有多个角色，而一个角色可以拥有多个权限，他们之间关系就是多对多的。而如果跳过角色这个卡点之间给用户分配权限的话，会加大我们开发人员的压力，因此我们把所有权限里面相应的权限分配给不同的角色，管理用户，让用户拥有其中某个角色，也就有了对应的权限。

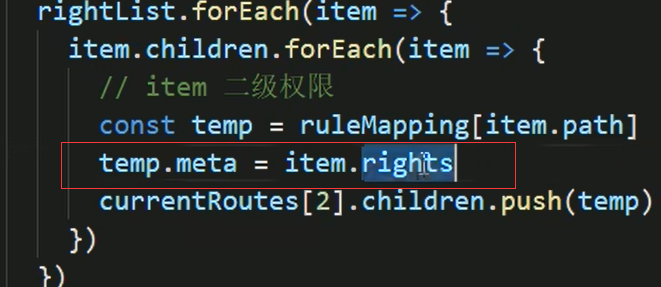
步骤：

1 首先点击登录之后获取数据，数据要立即存放在vuex的state中，供我们跳转界面的菜单渲染使用，但因为此时刷新页面必然会导致vuex中的数据清空，造成白屏时间，所以我们要提前把这个数据放到sessionstorage中，并且或上一个 [ ]，不然会报错。

2.登录进去之后，我们在主界面引入并映射一下vuex中的数据，供上面的menu组件渲染菜单，此时点击退出时，我们要把vuex中的state数据和sessionstorage中的数据全部清空，window.location.reload( )清空的是vuex， sessionstorage.clear( )清空的是sessionstorage

3.防止用户地址栏随机输入并跳转：在登录之后拿到token，并存储于sessionstorage中，此时如果要通过地址栏跳转，首先会判断sessionstorage中有没有token，如果有那就正常跳转；如果没有，那就要返回至登录页。这一步操作要在前置路由守卫中完成，至少要判断三种情况，跳转至登录刚好放行，登录了跳转至别的页面要看有没有token，有就放行，没有就跳转至登录页

4.如果把路由写死的话，用户跳转至本不具备权限的页面就会很麻烦，此时可根据data返回来的数据，来动态的展示路由。先做个路由表，循环遍历出store里data数组中的路由权限字符串，根据这个字符串来把相应的路由从路由表里拿出来，动态的追加到当前路由的children子路由里面



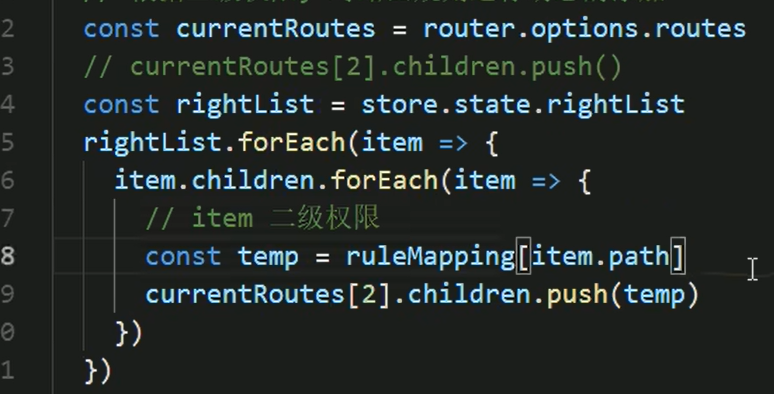
5.接着会遇到一个问题，就是如果此时地址栏输入新地址跳转成功，但点击刷新时显示空白页，是因为我们刷新没有触发跳转地址的这个方法，因此要在App.vue里的created周期里把这个方法调用一下就行了

6. 接着要做关于按钮的权限，可以使用自定义指令，这个v-x，x是一个direactive函数，它的第一个参数就是这个自定义x的值，这个函数里面有一个inserted函数。接**着**在之前追加动态路由上我们可以添加一个meta源信息，这个meta的值就等于返回的data中相应控制展示的字符w，接着在这个 inserted函数里利用indexof判断当前路由中是否也有el自定指令中的字符w，如果没有，则直接remove移除当前节点。如果不想移除，想直接禁用，那么可以在前面indexof判断为-1的里面，接着判断当前节点的自定义指令中有没有字符stop，如果有那么就el.disable=false，否则还是删除。

7.如果想要渲染当前列表，可以在请求拦截器中获取到当前的token并配置到header请求头中，如果当前所处的不是登录页，那么就能正常的获取数据并渲染。

8. 接着如果用户在调试器中做某些操作，可能会改变当前控件的属性，此时需要对这种行为做一个限制。在请求拦截器中，可以获得当前路由的meta，与config配置中的method对应的属性值进行对比，看看这个meta是否用这个属性值，如果没有就说明当前用户没有修改的权限，直接promise报个错就行了

9.如果用户长时间停留在页面，会造成token过期，此时刷新页面获取数据时所携带的token就会造成异常，根据并判断这个异常所对应的状态码，我们清空vuex中的state和sessionstorage中的值，并强制跳转至登录页就好了



flex布局，怎么样实现一行排三个，多的换行



Flex-wrap：自动换行

Flex-flow：是direction和wrap的集合 例如：flex-flow：row wrap

Transform 、translate

Transform属性：平移，旋转，缩放，渐变背景，3d旋转

渐变背景：background-image: linear-gradient(transparent, rgba(0,0,0,.6));

平移：translate(8px, 8px)

兄弟组件传值三种方式

1.eventbus



2.两个子组件注册到父组件，父组件再通过props来修改当前子组件的值

3.vuex

自行了解以下：

V8引擎

Sessionstorage的局限性

Sessionstorage和localstorage的区别，以及是否遵循同源策略

虚拟dom和diff算法(四种命中机制)

Css动画，来回的钟摆