**1.什么是跨域,你是怎么解决跨域的?**

跨域：域名，端口，协议三者中只要有一个不同就是跨域。对于端口和协议的不同，只能通过后台来解决。我们要解决的是域名不同的问题。

解决：

1.下面是用 php 进行的设置，“\*”号表示允许任何域向我们的服务端提交请求 header{"Access-Control-Allow-Origin： \*"}

2.JSONP，JSONP由回调函数和数据两部分组成。回调函数是当响应到来的时候应该在页面中调用的函数。数据就是传入回调函数中的JSON数据；第一个script标签定义了一个处理数据的函数，第二个script标签载入一个js文件

优点：它的兼容性好，在古老的浏览器中都可以运行，不需要XMLHtttpRequest或ActiveX的支持；它能直接访问响应文本，支持浏览器与服务器之间双向通信

缺点：JSONP 是从其他域中加载代码执行。如果其他域不安全，很可能会在响应中夹带一些恶意代码，而此时除了完全放弃 JSONP 调用之外，没有办法追究。因此在使用不是你自己运维的 Web 服务时，一定得保证它安全可靠；它只支持 GET 请求而不支持 POST 等其它类型的 HTTP 请求；它只支持跨域 HTTP 请求这种情况，不能解决不同域的两个页面之间如何进行 JavaScript 调用的问题。

**2.什么是闭包**

闭包：简单说就是函数嵌套函数，函数内部可以访问外部变量，外部不能直接访问该变量闭包保存了自己的私有变量，通过提供的接口给外部使用

缺点：垃圾回收机制无法回收其作用域影响内存，长时间闭包容易造成内存泄露

闭包的理及应用场景：<https://blog.csdn.net/qq_21132509/article/details/80694517>

**3.原型链**

Javascript 是面向对象的，每个实例对象都有一个\_\_proto\_属性，该属性指向它原型对象，这个实例对象的构造函数有一个原型属性 prototype，与实例的\_\_proto\_\_属性指向同一个对象。当一个对象在查找一个属性的时，自身没有就会根据\_\_proto\_\_ 向它的原型进行查找，如果都没有，则向它的原型的原型继续查找，直到查到Object.prototype.\_proto\_为 null，这样也就形成了原型链。

**4.继承**

1.原型继承：将父类的实例作为子类的原型，非常纯粹的关系，实例是子类的实例，也是父类的实例，父类中新增的方法和属性，子类都能访问到b.prototype=new a

2. 拷贝继承：将对象中的属性或者方法直接复制到另一个对象中

**5.盒子在页面水平方向和垂直方向居中**

1.定位加top和left各50%,margin-left和margin-top各-50%  
2.定位加transform:translateX/Y(-50%)  
3.felx布局 justify-content:center;arign-items:center

**6.ajax原理**

原理:  
1.创建异步对象  
 var xhr = new XMLHttpRequest();  
2.设置请求头 open('请求方式','请求地址',同步/异步);  
 xhr.opend('get','get.php?name='+name)//get如果有参数需要在地址栏中拼接  
 xhr.opend('post','post.php')//post 不许要传递参数需要设置请求头  
   xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded")  
3.设置请求体 send()  
 xhr.send(null)//get不用传参数  
 xhr.send('uname='+name)//post需要传参数  
4.接受服务器返回的参数 onreadystatechange并判断是否响应成功  
    xhr.onreadystatechange = function(){  
        if(xhr.readyState ==4 && xhr.status ==200){  
          var res = xhr.responseText;  
           document.getElementById('info').innerHTML = res;  
          }  
  }  
 jquery封装后的ajax  
   $.ajax({  
 type:'get/post',  
       url:'请求地址',  
       data:{  
           id :id //设置请求需要传递的参数  
      },  
       dataType:'json',//设置响应数据的格式  
       beforeSend:function(){  
           //发送请求之前的回调  
      }  
       success:function(){  
      //请求成功后的回调  
  }  
  })

**7.清除浮动的方式**

1.父级div定义height  
优点:解决了父级div无法获取高度的问题,代码少,易掌握.  
缺点:只适合宽高固定的布局  
2.结尾处添加空div标签,设置clear:both  
优点:利用css提供clear:both清除浮动  
缺点:如果浮动布局多就需要添加很多空div,让人感觉布局不友好  
3.父级div定义伪类:after和zoom  
缺点:有兼容性问题  
4.父级div定义overflow:hidden  
缺点:超出盒子部分会被隐藏  
5.双伪元素 clearfix:before,clearfix:after{}

**8.js执行机制**

1.JavaScript语言的一大特点就是单线程，也就是说，同一个时间只能做一件事,单线程就意味着，所有任务需要排队，前一个任务结束，才会执行后一个任务。如果前一个任务耗时很长，后一个任务就不得不一直等着。将所有任务分成两种，一种是同步任务（synchronous），另一种是异步任务（asynchronous）。同步任务指的是，在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务；异步任务指的是，不进入主线程、而进入"任务队列"（task queue）的任务，只有"任务队列"通知主线程，某个异步任务可以执行了，该任务才会进入主线程执行。

2.预解析 把所有的函数定义提前，所有的变量声明提前，变量的赋值不提前

**9.垃圾回收机制**

垃圾回收器会周期性的找出那些不再继续使用的变量，然后释放其内存。最常用的方式是标记清除和引用计数。为确保占用最少的内存让页面获得更好的性能，优化内存占用的最佳方式就是为执行中的代码只保存必要的数据。一旦数据不再有用，最好通过将其值设置为null来释放其引用--这个方法也叫做解除引用。

1.变量有生命周期  
2.标记清除（mark and sweep）  
这是JavaScript最常见的垃圾回收方式，当变量进入执行环境的时候，比如函数中声明一个变量，垃圾回收器将其标记为“进入环境”，当变量离开环境的时候（函数执行结束）将其标记为“离开环境”。  
3.引用计数(reference counting)

**10. cookie、sessionStorage、localStorage**

Cookie的作用是与服务器进行交互，作为HTTP规范的一部分而存在，大小只有4KB左右；有有效期，过期后将会删除;而Web Storage仅仅是为了在本地“存储”数据而生。

localStorage: 只能存储字符串，可以将对象JSON.stringify() 编码后存储  
1.存储的内容大概是20mb  
2.不同浏览器不能共享数据.但是在同一个浏览器下使用相同域名的不同窗口中可以共享数据  
3.永久存储,它的数据是存储在硬盘,可以手动清除  
sessionStorage:  
1.存储数据到本地,存储的容量为5mb  
2.这个数据本质是存储在当前页面的内存中,其它页面和浏览器无法获取  
3.它的生命周期为关闭当前页面,关闭你页面时,数据会自动清除  
使用方法:  
SetItem(key,value):设置数据，以键值对的方式  
getItem(key):通过指定的键获取对应的值内容  
removeItem(key):删除指定的key及对应的值内容  
clear():清空所有存储内容

**11.get与post**

1.GET参数通过URL传递;POST放在Request body中 2.GET存储内容小,不能超过2kb;POST原则没有长度的限制  
3.GET因为显示地址栏;POST需要加密,比get更安全  
4.GET请求没有请求主体,使用xhr.send(null);POST通过设置xhr.send('name='+name)  
5.GET一般用查询数据,POST一般用于数据的添加,修改和删除

6.GET在浏览器回退时不会重新请求；POST会重新请求

7.GET请求会被浏览器主动缓存，POST不会被缓存

**12.如何做好前端优化**

1.尽量减少http请求

2.使用浏览器缓存

3.使用压缩组件

4.图片、JS的预载入

5.将脚本放在底部

**13.vue生命周期函数**

1.beforeCreat：在beforCreate生命周期函数开始执行时，data和methods中的数据还没初始化

2.created：在created函数执行时，data和methdos都已经初始化好了，如果要调用methdos中的方法或者操作data中的数据，最早只能在created中操作

3.beforeMount：beforeMount函数执行时，模板已经在内存中编译好了，但是尚未挂载到页面中去，此时页面还是旧的

4.mounted：只要执行完mounted函数，表示整个vue实例已经初始化完毕了，.此时组件已经脱离了创建阶段，进入执行阶段.如果要用插件操作DOM中的节点,最早在mouted中进行

5.beforeUpdate：当执行beforUpdate函数式，页面中的显示数据还是旧的，此时date中数据是最新的，页面尚未和最新的数据保持同步

6.updated：当updated函数执行时，页面和date数据已经保持同步了，都是最新的

7.beforeDestroy：当执行beforeDestory钩子函数时，Vue实例就已经从运行阶段进入销毁阶段，实例身上的data和所有methdos以及指令，过滤器都处于可用状态，此时还没有真正的执行销毁过程

8.destroyed：当执行destroy函数时，组件已经完全销毁，此时实例中的所有数据，指令方，方法都已经不可用了

**14.less和sass区别**

1.sass 是基于 Ruby 的，然后是在服务器端处理的；less是基于javascript，在客户端处理。

2.关于变量在 LESS 和 Sass 中的唯一区别就是，LESS 用@，Sass 用$。

Less：

1. //less中的注释,不会被编译,/\*\*/会被编译
2. 申明变量:@变量名:变量值,多次出现的值,需要修改的值设为变量

Sass:

1. 申明变量:$变量名:变量值
2. Sass中运算,单位也会进行运算,使用时注意最终单位

**15.http协议**

1.首先客户机与服务器需要建立连接。只要单击某个超级链接，HTTP 的工作就开始了。

2.建立连接后，客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是 MIME 信息包括请求修饰符、客户机信息和可能的内容。

3.服务器接到请求后，给予相应的响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是 MIME 信息包括服务器信息、实体信息和可能的内容。

4.客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户机与服务器断开连接。

5.请求报文格式:请求行 － 请求头 － 空行 – 请求体

6.响应报文格式: 状态行 － 响应头 － 空行 – 响应体

7.持久链接/http长链接: 通过使用Connection:keep-alive进行长连接，。客户端只请求一次，但是服务器会将继续保持连接，当再次请求时，避免了重新建立连接。

8.长链接中的管线化:把现在的请求打包,一次性发过去,你也给我一次响应回来

**16.TCP三次握手**

1.第一次握手：建立连接时，客户端发送 syn 包（syn=j）到服务器，并进入 SYN\_SENT状态，等待服务器确认；SYN：同步序列编号（ Synchronize Sequence Numbers ）。

2.第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN\_RECV 状态；

3. 第三次握手：客户端收到服务器的 SYN+ACK 包，向服务器发送确认包ACK(ack=k+1），此包发送完毕，客户端和服务器进入 ESTABLISHED（TCP 连接成功）状态，完成三次握手。

4. TCP 协议的优缺点：

优点：TCP 发送的包有序号，对方收到包后要给一个反馈，如果超过一定时间还没收到反馈就自动执行超时重发，因此 TCP 最大的优点是可靠。

缺点：很简单，就是麻烦，如果数据量比较小的话建立连接的过程反而占了大头，不断地重发也会造成网络延迟，因此比如视频聊天通常就使用 UDP，因为丢失一些包也没关系，速度流畅才是重要的。

**17.主流浏览器的内核**

1.Trident 内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称 MSHTML]Gecko 内核：Netscape6 及以上版本，FF,MozillaSuite/SeaMonkey 等。

2.Presto 内核：Opera7 及以上。 [Opera 内核原为：Presto，现为、Blink;]Webkit 内核：Safari,Chrome 等。 [ Chrome 的：Blink（WebKit 的分支）]

3. EdgeHTML 内核：Microsoft Edge。 [此内核其实是从 MSHTML fork 而来，删掉了几乎所有的 IE 私有特性]

**18.this关键字**

1.this 在函数中指向window,因为window调用了这个函数.

2.this在对象中指向的是Obj,指向调用者

3. this在构造函数中指向调用者

4. 一般情况下,this指向的是它的调用者

5. apply 调用 this 指向 apply 方法的第一个参数

**19.Promise**

1.promise是es6中新增的一个构造函数，是为了解决异步操作中数据调用嵌套(回调地狱)的问题。

2. Promise 是一个 构造函数，既然是构造函数， 那么，我们就可以 new Promise() 得到一个 Promise 的实例；

3. 构造函数的参数是一个函数，这个函数会立即执行

4.在 Promise 构造函数的 Prototype 属性上，有一个 .then() 方法，也就说，只要是 Promise 构造函数创建的实例，都可以访问到 .then() 方法

5. 在 Promise 上，有两个函数，分别叫做 resolve（成功之后的回调函数） 和 reject（失败之后的回调函数

6. Promise 表示一个 异步操作；每当我们 new 一个 Promise 的实例，这个实例，就表示一个具体的异步操作；这个异步操作的结果，有两种状态：

异步执行成功了，需要在内部调用 成功的回调函数 resolve 把结果返回给调用者；异步执行失败了，需要在内部调用 失败的回调函数 reject 把结果返回给调用者；

7.由于 Promise 的实例，是一个异步操作，所以，内部拿到 操作的结果后，无法使用 return 把操作的结果返回给调用者； 这时候，只能使用回调函数的形式，来把 成功 或 失败的结果，返回给调用者；

8. 我们可以在 new 出来的 Promise 实例上，调用 .then() 方法，【预先】 为 这个 Promise 异步操作，指定 成功（resolve） 和 失败（reject） 回调函数；

9. 后续的Promise 执行，依赖于 前面 Promise 执行的结果，如果前面的失败了，则后面的就没有继续执行下去的意义了，此时，我们想要实现，一旦有报错，则立即终止所有 Promise的执行；

**20. Vue中调试工具**

安装插件: 安装一个谷歌插件Ready.Detected Vue

**21.在vue中怎么取消地址栏里的#号**

const router = new VueRouter({

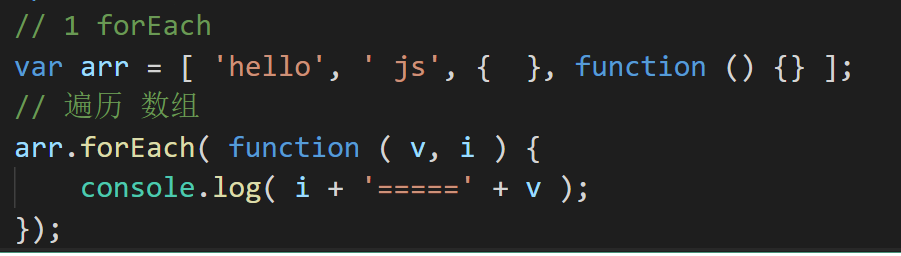
mode: ‘history’,//history去掉#，hash是有#的

routes: […]

}）

**22.ES5中数组新增方法**

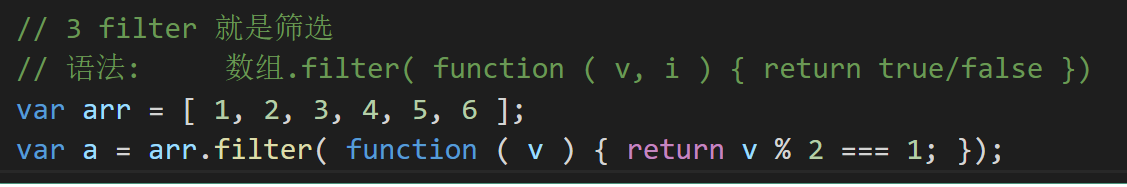
1. forEach（） 不能中止 break



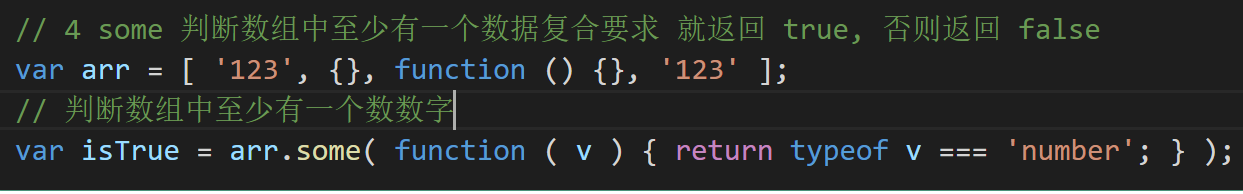
2. map() 映射



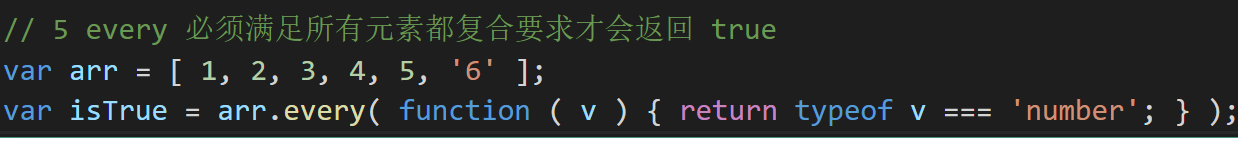
3. filter() 筛选



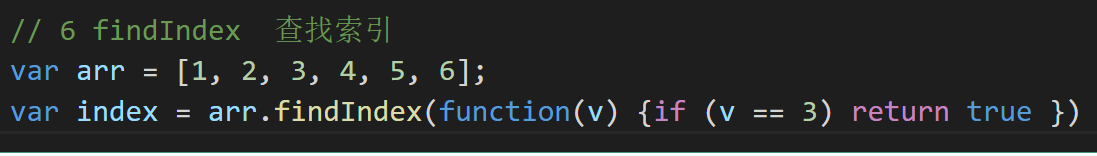
4. some()



5.every()



6.findIndex()



**23.let和var的区别**

1.for循环中用var申明的变量

for (var i = 0; i <10; i++) {

setTimeout(function() { // 同步注册回调函数到 异步的 宏任务队列。

console.log(i); // 执行此代码时，同步代码for循环已经执行完成

}, 0);

}

// 输出结果 10 共10个

// 这里面的知识点： JS的事件循环机制，setTimeout的机制等

2.for循环中用let声明的变量

// i虽然在全局作用域声明，但是在for循环体局部作用域中使用的时候，变量会被固定，不受外界干扰。

for (let i = 0; i < 10; i++) {

setTimeout(function() {

console.log(i); // i 是循环体内局部作用域，不受外界影响。

}, 0);

}

// 输出结果：0 1 2 3 4 5 6 7 8 9

**24.Vue中常用的钩子函数有哪些**

1. bind：只调用一次，指令第一次绑定到元素时调用。在这里可以进行一次性的初始化设置。

2. inserted：被绑定元素插入父节点时调用 (仅保证父节点存在，但不一定已被插入文档中)。

3. update：所在组件的 VNode 更新时调用，但是可能发生在其子 VNode 更新之前。指令的值可能发生了改变，也可能没有。但是你可以通过比较更新前后的值来忽略不必要的模板更新

4. componentUpdated：指令所在组件的 VNode 及其子 VNode 全部更新后调用。

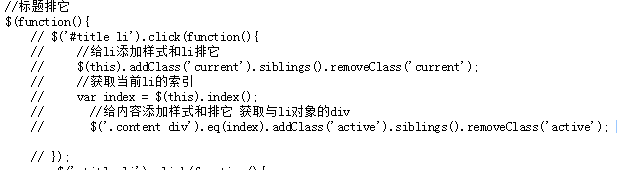
5. unbind：只调用一次，指令与元素解绑时调用。

钩子函数的参数 (即 el、binding、vnode 和 oldVnode)

**25. css实现倒三角**



**26.tab栏切换**

****

**27.前端实现登录拦截(路由拦截和http拦截)**

1.路由拦截: 首先在定义路由的时候就需要多添加一个自定义字段requireAuth，用于判断该路由的访问是否需要登录。如果用户已经登录，则顺利进入路由， 否则就进入登录页面。

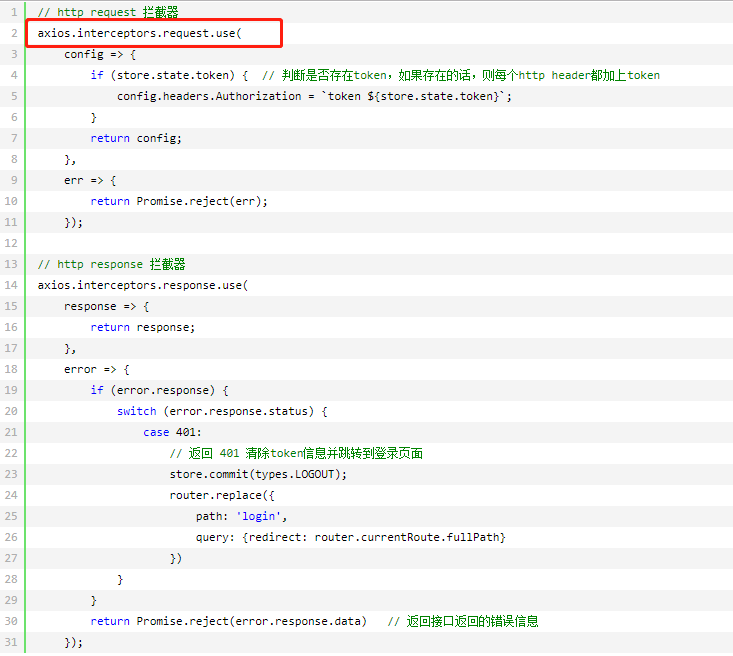


定义完路由后，我们主要是利用vue-router提供的钩子函数beforeEach()对路由进行判断。



每个钩子方法接收三个参数：   
\* to: Route: 即将要进入的目标 路由对象   
\* from: Route: 当前导航正要离开的路由   
\* next: Function: 一定要调用该方法来 resolve 这个钩子。执行效果依赖 next 方法的调用参数。   
\* next(): 进行管道中的下一个钩子。如果全部钩子执行完了，则导航的状态就是 confirmed （确认的）。   
\* next(false): 中断当前的导航。如果浏览器的 URL 改变了（可能是用户手动或者浏览器后退按钮），那么 URL 地址会重置到 from 路由对应的地址。   
\* next(‘/’) 或者 next({ path: ‘/’ }): 跳转到一个不同的地址。当前的导航被中断，然后进行一个新的导航。

2.拦截器: 要想统一处理所有http请求和响应，就得用上 axios 的拦截器。通过配置http response inteceptor，当后端接口返回401 Unauthorized（未授权），让用户重新登录。



3.http拦截 见链接<https://www.cnblogs.com/guoxianglei/p/7084506.html>

28.vue中怎么实现跨域

1. 在App.vue中使用created方法创建fetch，将域名及方法等创建，如下图



2. 在config配置文件中的index.js中的跨域区域中写入如下代码：



**29.ES6中class关键字**

ES6引入了Class（类）这个概念，通过class关键字可以定义类。该关键字的出现使得其在对象写法上更加清晰，更像是一种面向对象的语言。

class Person{//定义了一个名字为Person的类

constructor(name,age){//constructor是一个构造方法，用来接收参数

this.name = name;//this代表的是实例对象

this.age=age;

}

say(){//这是一个类的方法，注意千万不要加上function

return "我的名字叫" + this.name+"今年"+this.age+"岁了";

}

}

var obj=new Person("laotie",88);

console.log(obj.say());//我的名字叫laotie今年88岁了

注意项:

1. 在类中申明方法的时候,千万不要给方法加上function关键字
2. 方法之间不能用逗号分割,否则会报错
3. Class不存在变量提升,所以需要先定义再使用

**30.数组的常用方法**

toString() //把数组转换成字符串，逗号分隔每一项  
push() // 向数组最后添加一个或多个元素    
pop() //删除数组的最后一个元素  
shift() //删除数组的第一个元素  
unshift() //向数组的开头添加一个或多个元素  
reverse() //颠到数组中元素的顺序  
sort() //对数组的元素进行排序  
concat() //连接两个或多个数组  
slice() //数组截取[start,end)  
splice() //删除数组元素(index,number)  
indexOf() //获取数组中某个字符的位置  
lastIndexOf()//获得某个字符的倒数位置  
join('分隔符')//将数组转为字符串

数组迭代方法(过滤)arr.filter(匿名函数);  
var arr = [1500, 1200, 2000, 2100, 1800];  
var resu = arr.filter(function(item) {  
   // item 是个形参   包含了 每个 数组值  
   return item < 2000  
})  
console.log(resu);

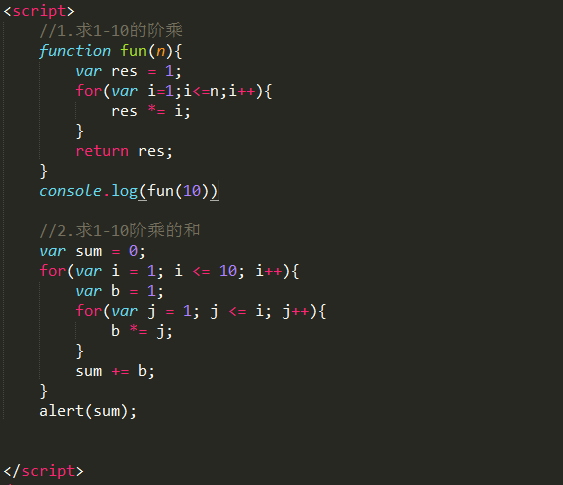
**31.字符串常用方法**

charAt() //获取指定位置字符  
charCodeAt() //获取指定位置处字符的ASCII码  
concat() //拼接字符串，等效于+，+更常用  
slice() //从start位置开始截取,截取到end位置,end取不到  
substring() //从start位置开始截取,街渠道end位置,end取不到  
substr() //从start位置开始,截取length个字符  
indexof() //返回指定内容在原字符串中的位置  
lastIndexOf() //从后往前找,只找第一个匹配额  
trim() //只能除去字符串前后的空白  
toUpperCase() //转换大写  
toLowerCase() //转换小写  
replace() //替换  
split() //切割

**32.字符串练习**



**33.求1-10的阶乘**



**34.Vue中内置组件**

1.component组件：有两个属性---is inline-template渲染一个‘元组件’为动态组件，按照'is'特性的值来渲染成那个组件

2.transition组件：为组件的载入和切换提供动画效果，具有非常强的可定制性，支持16个属性和12个事件

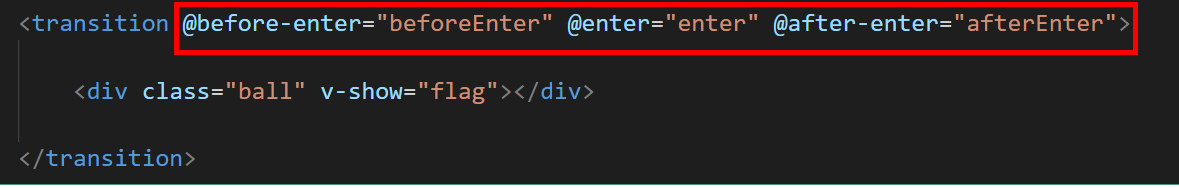
3.transition-group：作为多个元素/组件的过渡效果

4.keep-alive：包裹动态组件时，会缓存不活动的组件实例，而不是销毁它们

5.slot：作为组件模板之中的内容分发插槽，slot元素自身将被替换

**35.Vue中的小球动画钩子函数**

****





**36.jquery中的事件绑定**

1.bind:

绑定多个事件

参数:{"事件的类型":事件处理函数,....}

2.delegate

参数:父级元素.delegate("子级元素","事件类型",事件处理函数)

3.on

参数:父级元素.on("事件类型","子级元素",事件处理函数);

建议：在手机上或者动态添加的事件推荐使用on来添加

标签.on('click',function(){}) //给自己添加事件

标签.on('click','p',function(){}) //给子元素添加事件

**37.webpack和gulp的区别**

gulp强调的是前端开发的工作流程，我们可以通过配置一系列的task，定义task处理的事务（例如文件压缩合并、雪碧图、启动server、版本控制等），然后定义执行顺序，来让gulp执行这些task，从而构建项目的整个前端开发流程。

webpack是一个前端模块化方案，更侧重模块打包，我们可以把开发中的所有资源（图片、js文件、css文件等）都看成模块，通过loader（加载器）和plugins（插件）对资源进行处理，打包成符合生产环境部署的前端资源。

**38.内存泄露** <http://www.ruanyifeng.com/blog/2017/04/memory-leak.html>

1.内存泄露:不再用到的内存，没有及时释放，就叫做内存泄漏（memory leak）。

2. 如果连续五次垃圾回收之后，内存占用一次比一次大，就有内存泄漏。这就要求实时查看内存占用。

3.chrome中查看内存占用：

1.打开开发者工具，选择 Timeline 面板

2.在顶部的Capture字段里面勾选 Memory

3.点击左上角的录制按钮。

4.在页面上进行各种操作，模拟用户的使用情况。

5.一段时间后，点击对话框的 stop 按钮，面板上就会显示这段时间的内存占用情况。

4.引发内存泄露的原因及解决方法

1.闭包：把element变量设置微null来删除对dom的引用。这样就可以减少对dom的引用数，从而正常回收其占用的内存

2.一个为申明的全局变量：在JavaScript 文件开头添加 'use strict'; 语句

3.定时器setTimeout和setInterval：及时清除定时器

4. 给DOM对象添加的属性是一个对象的引用：在window.Onunload添加doucment.getElement.By.Id(‘idname).property=null

5. echarts配合循环计时器等出现的内存泄漏

**38.怎么优化DOM节点**

1.减少DOM访问次数,尽量使用javascript处理

2.使用局部变量存储DOM引用

3.在内存中操作元素(文档片段DocumentFragment)

4.减少iframe,样式放在head中,脚本放在关闭标签</body>之前

5.重绘与重排浏览器会使渲染树中收到影响的部分消失，重新构建渲染树，这个过程称为“**重排(reflow)**”。重排完成之后，浏览器会重新将受到影响的部分绘制到浏览器中，这个过程称之为“**重绘(repaint)**

**39.父组件调用子组件中的方法**



**40.Vue和React的区别**

**1.** 监听数据变化的实现原理不同:

Vue 通过 getter/setter 以及一些函数的劫持，能精确知道数据变化，不需要特别的优化就能达到很好的性能

React 默认是通过比较引用的方式进行的，如果不优化（PureComponent/shouldComponentUpdate）可能导致大量不必要的VDOM的重新渲染

2.数据流的不同

Vue中实现组价和DOM之间用v-model实现双向数据绑定

React一直提倡的是单向数据流，他称之为 onChange/setState()模式。

3.hoc和mixins

Vue 中我们组合不同功能的方式是通过 mixin

React中我们通过 HoC (高阶组件）

4.模板渲染方式不同

Vue是通过一种拓展的HTML语法进行渲染

React 是通过JSX渲染模板

5. Vuex 和 Redux 的区别

Vuex 中，$store 被直接注入到了组件实例中，因此可以比较灵活的使用, 使用 dispatch 和 commit 提交更新, 通过 mapState 或者直接通过 this.$store 来读取数据.

Redux 中，我们每一个组件都需要显示的用 connect 把需要的 props 和 dispatch 连接起来。

**41.怎样保证vue中组件样式的唯一性**

在组件的样式style标签内添加scoped

**42.在vuezhong怎样引入css的公共类样式**

在main.js中引入,不会冲突,会覆盖

**43.登录拦截的的token是通过什么发送的**

**44.平时工作中你是怎样封装函数的,要注意什么**

**45.怎样设置cookie**

1.通过document.cookie,可以设置cookie的值,document.cookie=”user=wang”

2.设置持久cookie有效时间: Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2019 07:28:00 GMT;

**46.图片压缩**

在移动端压缩图片并且上传主要用到filereader、canvas 以及 formdata 这三个h5的api。逻辑并不难。整个过程就是：

　　（1）用户使用input file上传图片的时候，用filereader读取用户上传的图片数据（base64格式）

　　（2）把图片数据传入img对象，然后将img绘制到canvas上，再调用canvas.toDataURL对图片进行压缩

（3）获取到压缩后的base64格式图片数据，转成二进制塞入formdata，再通过XmlHttpRequest提交formdata。