Web前端面试题

二阶段面试题：.....13
三阶段面试题：.....35
63、封装原生promise函数.....97
65、Async/Await：比Promise更好的6个理由.....98
66、前端开发中的设计模式.....99
67.状态码.....100
68、盟迅面试.....102
69、面试重点.....103
1、什么是webpack和grunt和gulp有什么不同.....103
2、什么是bundle,什么是chunk，什么是module?.....103

一阶段面试题：

1. CSS浮动怎么理解的

浮动：float：left、right、none

作用：想要从上往下排列的元素变成从左到右或从右到左排列。

如果子元素有浮动，那么父元素一定要设置高度；浮动之后脱离文档流，不占据原来的位置；子元素浮动，如果子元素的宽度的和大于父元素的宽度，那么子元素就会往下去排列；

2. 绝对定位相对定位怎么理解

绝对定位：absolute

参照物：相对于浏览器额可视窗口或者最近的有定位的（非static）父级元素来定位

特点：给元素设置绝对定位后，元素会脱离文档流，不占据原来的位置，下面的元素可以往上走

相对定位：relative

参照物：元素本身的位置

特点：移动之后还占据着原来的位置，不脱离文档流

3. 块元素和行内元素什么区别

（1）块级元素：独占一行，可以设置宽高，块级元素可以作为所有内联元素的容器和部分块级元素的容器，常用的块级元素有：div,dl,dt,dd,ol,ul,fieldset,h1-h6,p,from,table,tr,td

（2）行内元素：不独占一整行，不可以设置宽高，宽高由内容撑开，行内元素也遵循盒子模型，但是margin-top、margin-bottom、padding-top、padding-bottom是没有效果的。常用的行内元素：a-超链接、span、img、i、b、em、br、input、sub、u

***4. Css如何实现盒子水平垂直居中

a,----定位方式（父元素宽高固定，子元素宽高固定）

```
    <div class="Father">
        <div class="children"></div>
    </div>
  <style lang="scss" scoped>
   .Father{
     position: relative;
   }
   .children{
     width: 50px;
     height: 50px;
     position: absolute;
     top: 0px;
     bottom: 0px;
     left: 0px;
     right: 0px;
     margin: auto;
   }
</style>
```

b, ----flex布局方式（父元素宽高不固定，子元素宽高不固定）

```
    <div class="Father">
        <div class="children"></div>
    </div>
  <style lang="scss" scoped>
  .Father {
    border: 1px solid red;
    height: 100px;
    display: flex;
    justify-content: center;
    align-items: center;
  }
  .children {
    width: 50px;
    height: 50px;
    border: 1px solid blue;
  }
</style>
```

c, ----transform方式（父元素宽高不固定，子元素宽高不固定）

```
    <div class="Father">
        <div class="children"></div>
    </div>
  <style lang="scss" scoped>
  .Father {
    border: 1px solid red;
    height: 100px;
    position: relative;
  }
  .children {
    width: 50px;
    height: 50px;
    border: 1px solid blue;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);  /* 使用css3的transform来实现 */
  }
</style>
```

5. Href 和 src的区别

href标识超文本引用，用在link和a等元素上，href是引用和页面关联，是在当前元素和引用资源之间建立联系

src表示引用资源，表示替换当前元素，用在img，script，iframe上，src是页面内容不可缺少的一部分。

6. Px和Rem的区别

px是固定长度单位，不随其它元素的变化而变化；rem是相对于根目录（HTML元素）的，所有它会随HTML元素的属性（font-size）变化而变化。

px和%用的比较广泛一些，可以充当更多属性的单位，而em和rem是字体大小的单位，用于充当font-size属性的单位。

7. 常见的浏览器内核和前缀有哪些？ 微信的浏览器内核是什么

Chrome(谷歌浏览器)： WebKit内核 -webkit-

Safari(苹果浏览器)： WebKit内核 -webkit-

Firefox(火狐浏览器)： Gecko内核 -moz-

IE(IE浏览器)：　　Trident内核　　-ms-

Opera(欧朋浏览器)：　Presto内核　　-o-

微信的浏览器内核：X5内核是腾讯基于开源Webkit优化的浏览器渲染引擎

8. html5中新增哪些api

拖拽释放(Drag and drop) API

音频、视频API(audio,video)

画布(Canvas) API

地理(Geolocation) API

9. html5新增哪些标签

Header、Nav、Article、Section、Aside、Footer

10. 移动端怎么做适配

<meta name="viewport" content="width=device-width, initial-scale=1.0">

11. 你们以前移动端是怎么写的

12. 你们以前移动端是怎么布局的

13. 你对语义化标签是怎么理解的？有什么好处？

理解：

html语义化就是让页面的内容结构化，便于对浏览器、搜索引擎解析。

好处：①去掉或者丢失样式的时候能够让页面呈现出清晰的结构

②有利于SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；

③方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；

④便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。

14. Css3动画有哪些

15. 使用 CSS 预处理器吗？喜欢那个？

CSS预处理器定义了一种新的语言，基本的思想是用一种专门的编程语言，开发者只需要使用这种语言进行编码工作，减少枯燥无味的CSS代码的编写过程的同时，它能让你的CSS具备更加简洁、适应性更强、可读性更加、层级关系更加明显、更易于代码的维护等诸多好处。

CSS预处理器种类繁多，本次就以Sass、Less、Stylus进行比较。

16. CSS优化、提高性能的方法有哪些？

加载性能：

1）css压缩：将写好的css进行打包压缩，可以减少很多的体积；

2） css单一样式：当需要下边距和左边距的时候，

很多时候选择：margin: top 0 bottom 0;

但 margin-bottom: bottom;margin-left: left;执行的效率更高；

3） 减少使用 @import，而建议使用link，因为后者在页面加载时一起加载，前者是等待页面加载完成之后再进行加载；

选择器性能：

CSS选择符是从右到左进行匹配的。当使用后代选择器的时候，浏览器会遍历所有子元素来确定是否是指定的元素等等；

避免使用通配规则

如*{} 计算次数惊人！只对需要用到的元素进行选择

尽量少的去对标签进行选择，而是用class

如：#nav li{},可以为li加上nav_item的类名，如下选择.nav_item{}

不要去用标签限定ID或者类选择符

如：ul#nav,应该简化为#nav

尽量少的去使用后代选择器，降低选择器的权重值

后代选择器的开销是最高的，尽量将选择器的深度降到最低，最高不要超过三层，更多的使用类来关联每一个标签元素

考虑继承

了解哪些属性是可以通过继承而来的，然后避免对这些属性重复指定规则

渲染性能：

1.慎重使用高性能属性：浮动、定位；

2.尽量减少页面重排、重绘；

重排按照css的书写顺序：

·位置：positon、top、left、z-index、float、dispay

·大小：width、height、margin、padding

·文字系列： font、line-height、color、letter-spacing

·背景边框：background、 border

·其它：anmation、transition

重绘:border、outline、background、box-shadow,能使用background-color,就尽量不要使用background;

3.去除空规则： {} ；

4.属性值为0时，不加单位；

5.属性值为浮动小数0.**，可以省略小数点之前的0；

6.标准化各种浏览器前缀：带浏览器前缀的在前。标准属性在后；

7.不使用@import前缀，它会影响css的加载速度；

8.充分利用css继承属性，减少代码量；

9.抽象提取公共样式，减少代码量；

10.选择器优化嵌套，尽量避免层级过深；

11.css雪碧图，同一页面相近部分的小图标，方便使用，减少页面的请求次数，但是同时图片本身会变大，使用时，优劣考虑清楚，再使用；

12.将css文件放在页面最上面

可维护性、健壮性：

1 将具有相同属性的样式抽离出来，整合并通过class在页面中进行使用,提高css的可维护性；

2 继上一条，oocss也是提高css性能的途径之一，通过定义可复用的、语义化良好的基础类，然后添加到html中，这也是很多ui框架都在使用的一种方法，例如：class="btn btn-active btn-blue"；

3 样式与内容分离：讲css代码定义到外部css中；

4 容器与样式分离；

17. 怎么让Chrome支持小于12px 的文字

针对谷歌浏览器内核，加webkit前缀，用transform:scale()这个属性进行缩放！

```
<style>
p span{font-size:10px;-webkit-transform:scale(0.8);display:block;}
</style>
<p><span>测试10px</span></p>
```

18. CSS3有哪些新特性?

1）选择器　2）CSS3 边框（Borders）（圆角边框，添加阴影框）3）CSS3 背景（background-clip、background-size） 4）CSS3 渐变（线性渐变、径向渐变） 5）CSS3 文本效果（text-overflow、text-shadow） 6）CSS3 字体 7）CSS3 转换和变形（transform） 8）CSS3 过渡（transition） 9）CSS3 动画（animation） 10）CSS3 多媒体查询 11）CSS3伸缩布局盒模型(弹性盒)

19. html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何 区分HTML和HTML5?

新特性:

1）拖拽释放(Drag and drop) API

2）语义化更好的内容标签（header,nav,footer,aside,article,section）

3）音频、视频API(audio,video)

4）画布(Canvas) API

5）地理(Geolocation) API

6）本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失;

7） sessionStorage 的数据在浏览器关闭后自动删除

8） 表单控件，calendar、date、time、email、url、search

9） 新的技术webworker, websocket, Geolocation

移除的元素：

1）纯表现的元素：basefont，big，center，font, s，strike，tt，u；

2）对可用性产生负面影响的元素：frame，frameset，noframes；

支持HTML5新标签：

1）IE8/IE7/IE6支持通过 document.createElement 方法产生的标签，可以利用这一特性让这些浏览器支持 HTML5 新标签，浏览器支持新标签后，还需要添加标签默认的样式（当然最好的方式是直接使用成熟的框架、使用最多的是html5shim框架）：

<!--[if lt IE 9]>

<script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>

<![endif]-->

如何区分：

DOCTYPE声明新增的结构元素、功能元素

20. ::before 和 :after中双冒号和单冒号 有什么区别?

单冒号(:)用于CSS3伪类，双冒号(::)用于CSS3伪元素。（伪元素由双冒号和伪元素名称组成）

双冒号是在当前规范中引入的，用于区分伪类和伪元素。不过浏览器需要同时支持旧的已经存在的伪元素写法，

比如:first-line、:first-letter、:before、:after等，

而新的在CSS3中引入的伪元素则不允许再支持旧的单冒号的写法。

想让插入的内容出现在其它内容前，使用::before，否者，使用::after；

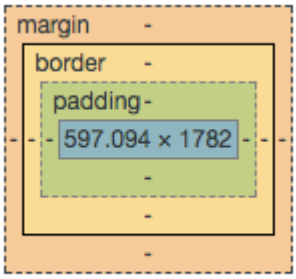
在代码顺序上，::after生成的内容也比::before生成的内容靠后。

如果按堆栈视角，::after生成的内容会在::before生成的内容之上

21. 说下遇到的各种兼容性的处理方法?

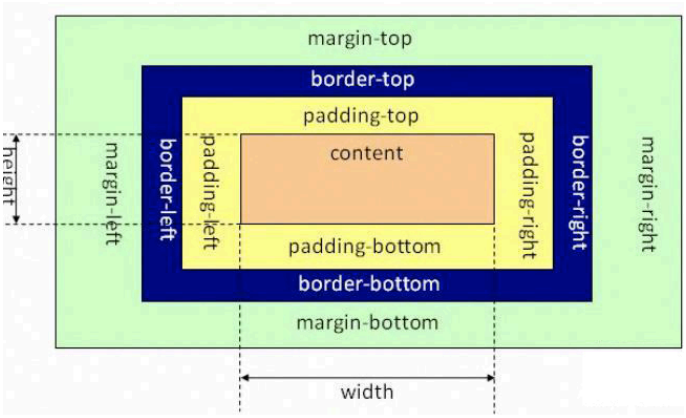
22. Html盒模型的不同?

①盒模型本质是一个盒子，封装周围的HTML元素，它包括：外边距、边框、内边距、实际内容。如图所示：



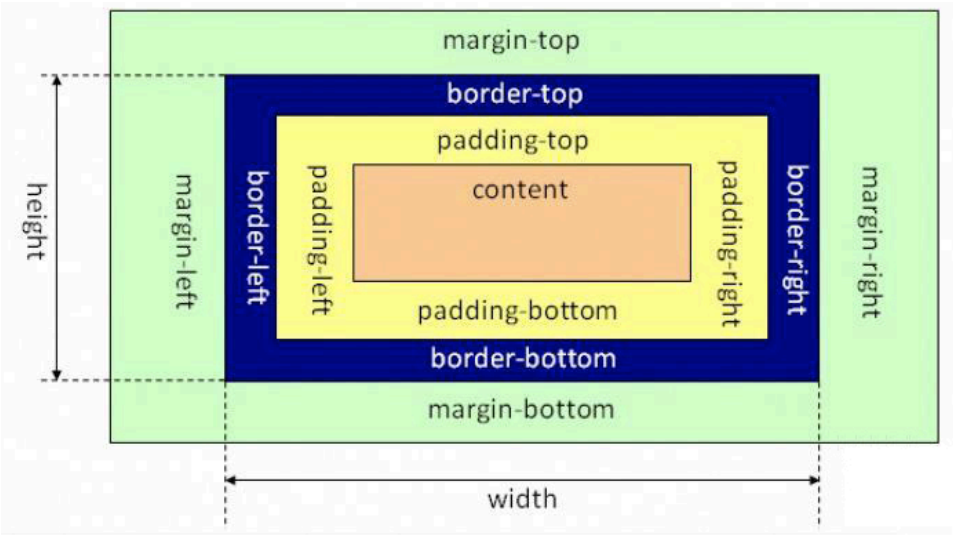
2.盒模型分为两类：标准盒子模型和怪异盒子模型。

标准盒子模型就不用说了，就是很规范的盒模型。如下图所示：



从上图可以看到标准盒子模型包括margin，border，padding和content，并且content部分不包含其他部分。

怪异盒模型又称为IE盒子模型。如图所示：



23. 响应式布局

① 设置 Meta 标签

在<head>标签里加入这个meta标签。<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">

②通过媒介查询来设置样式 Media Queries

Media Queries 是响应式设计的核心。·

它根据条件告诉浏览器如何为指定视图宽度渲染页面。假如一个终端的分辨率小于 980px，那么可以这样写：

```
@media screen and (max-width: 980px) {
  #head { … }
  #content { … }
  #footer { … }
}
```

③设置多种试图宽度

如我们要设定兼容 iPad 和 iphone 的视图，那么可以这样设置：

```
/** iPad **/
```

@media only screen and (min-width: 768px) and (max-width: 1024px) {}

/** iPhone **/

@media only screen and (min-width: 320px) and (max-width: 767px) {}

24. 透明度opacity和rgba的区别?

opacity 子元素颜色会被影响，即变相的被继承。rgba(r,g,b,a)不会

25. 兼容IE8以下的浏览器你是如何编写css样式的

26. 移动端项目开发过程中你遇到过哪些兼容行问题，

①圆角bug

某些Android手机圆角失效

background-clip: padding-box;

②安卓浏览器看背景图片，有些设备会模糊。

用同等比例的图片在PC机上很清楚，但是手机上很模糊，原因是什么呢?

经过研究，是devicePixelRatio作怪，因为手机分辨率太小，如果按照分辨率来显示网页，这样字会非常小，所以苹果当初就把iPhone 4的960*640分辨率，在网页里只显示了480*320，这样devicePixelRatio = 2。现在android比较乱，有1.5的，有2的也有3的。

想让图片在手机里显示更为清晰，必须使用2x的背景图来代替img标签（一般情况都是用2倍）。例如一个div的宽高是100*100，背景图必须得200*200，然后background-size:contain;，这样显示出来的

图片就比较清晰了。

代码可以如下：

background:url(../images/icon/all.png) no-repeat center center;
-webkit-background-size:50px 50px;
background-size: 50px 50px;display:inline-block; width:100%;
height:50px;

或者指定 background-size:contain;都可以，大家试试！

③iphone及ipad下输入框默认内阴影

Element{
-webkit-appearance:none;
}

二阶段面试题：

1、Ajax同步和异步怎么理解的

A、同步是指：发送方发出数据后，等接收方发回响应以后才发下一个数据包的通讯方式。

用户填写所有信息后，提交给服务器，等待服务器的回应（检验数据），是一次性的。信息错误又要重新填写！

B、异步是指：发送方发出数据后，不等接收方发回响应，接着发送下个数据包的通讯方式。

当用户填写完一条信息后，该信息会自动向服务器提交，然后服务器响应客户端，在此过程中，用户依然在填写表格的信息，即向服务器请求多次，节省了用户的时间，提高了用户的体验。

2、Js面相对象是怎么理解的

A、面向对象：就是一种编程思想，只关注其功能，不关注其内部细节面向对象特点：

①封装： 不考虑内部实现，只考虑功能

②继承： 从已有的对象上，继承出新的对象

③多态： 不同的操作，产生不同的结果

B、JS中的对象就是普通名值对的集合。使用对象时，只关心对象提供的功能，不用关注内部细节。

C、.创建对象的三种方式

a.工厂模式，使用简单的函数创建对象，为对象添加属性和方法，然后返回对象

b.构造函数模式，创建自定义引用类型，可以像创建内置对象实例一样使用new操作符，这种方法的缺点是，构造函数的每个成员都无法复用，每次创建出的对象都只有私有变量和私有方法，不能实现共用

c.原型模式，使用构造函数的prototype属性来指定共享的属性和方法，即使用构造函数定义实例属性，使用原型定义共享的属性和方法

d.继承

JS主要通过原型链实现继承，原型链是通过将一个类型的实例赋值给一个构造函数的原型实现的

3.Js写过原生插件吗?

4. Js闭包是如何理解的?

（一）

使用闭包主要是为了设计私有的方法和变量。闭包的优点是可以避免全局变量的污染，缺点是闭包会常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。

闭包有三个特性:

1.函数嵌套函数 2.函数内部可以引用外部的参数和变量 3.参数和变量不会被垃圾回收机制回收

（二）

A、定义和用法：当一个函数的返回值是另外一个函数，而返回的那个函数如果调用了其父函数内部的其它变量，如果返回的这个函数在外部被执行，就产生了闭包。

B、表现形式：使函数外部能够调用函数内部定义的变量。

C、实例如下:

(1)、根据作用域链的规则，底层作用域没有声明的变量，会向上一级找，找到就返回，没找到就一直找，直到window的变量，没有就返回undefined。这里明显count 是函数内部的flag2 的那个count 。

```
var count=10;    //全局作用域 标记为flag1
function add(){
  var count=0;     //函数全局作用域 标记为flag2
    return function(){
    count+=1;    //函数的内部作用域
    alert(count);
  }
}
var s = add()
s();//输出1
s();//输出2
```

D、变量的作用域

要理解闭包，首先必须理解Javascript特殊的变量作用域。

变量的作用域分类：全局变量和局部变量。

特点：

①、函数内部可以读取函数外部的全局变量；在函数外部无法读取函数内的局部变量。

②、函数内部声明变量的时候，一定要使用var命令。如果不用的话，你实际上声明了一个全局变量！

E、使用闭包的注意点

1）滥用闭包，会造成内存泄漏：由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

2）会改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把

内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

5、apply和call、bind的区别

希望使用某个上下文调用该函数，请使用 .bind() ，这在事件中很有用。 如果要立即调用函数，请使用.call() 或 .apply()，并修改上下文。

共同点：都可以改变函数this指向，都是函数的方法

区别：参数不同；call和apply会马上执行fn函数的代码，bind不会，bind只是改变this指向；

call和apply功能一样，区别在于参数不同（传参方式不同，call多个参数，apply只有两个参数，第二个参数为数组）

6.js中数组后面添加数据、数组前面添加数据、数组后面删除数据、数组前面删除数据用什么?

push（）、unshift（）、pop（）、shift（）

7. 请描述一下cookies，sessionStorage和localStorage的区别?

相同点：都存储在客户端

不同点：

a.存储大小

cookie数据大小不能超过4k。

sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。

b.有效时间

localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动

删除数据；

sessionStorage 数据在当前浏览器窗口关闭后自动删除。

cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭

c. 数据与服务器之间的交互方式

cookie的数据会自动的传递到服务器，服务器端也可以写cookie到客户端

sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

8. 你对localStorage是怎么理解的?

A、localStorage对象是HTML5的客户端存储持久化数据的方案。为了能访问到同一个localStorage对象，页面必须来自同一个域名（子域名无效），使用同一种协议，在同一个端口上。

过期策略：localstorage永久存储，不过期，除非手动删除，sessionstorage在重启浏览器、关闭页面或新开页面时失效。

B、localStorage是Storage类型的实例。有以下的几种方法：

①clear（）：删除所有值。

②getItem（name）：根据指定的名字name获取对应的值

③key(index)：在指定的数字位置获取该位置的名字。

④removeItem（name）：删除由name指定的名值对

⑤setItem(name,value)：为指定名字设置一个对应的值

C、例：使用方法来存储数据

localStorage.setItem("name","aaa");//这样就用localStorage存储

了一个名字为name的数据，数据的内容为 “aaa"

使用方法来读取数据

localStorage.getItem("name");//这样就读取了名字为“name”的数据的值。

9.跨域请求是怎么理解的?

10.Post和get的区别是怎么理解的?

GET和POST本质上两者没有任何区别。他们都是HTTP协议中的请求方法。底层实现都是基于TCP/IP协议。
传送方式：get通过地址栏传输，post通过报文传输。
get请求数据接在url后面，post请求数据通过send方法传递
get传递的数据会比较少，post没有限制
get传递的数据会暴露出来
GET产生一个TCP数据包；POST产生两个TCP数据包
长的说：
对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；
而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

11、什么情况下面存在跨域

① 不同域名属于跨域，如：www.a.com 和www.b.com，另外www.a.com 和www.a.com.cn也属于不同域名。
② 主域名和子域名（二级域名、三级域名等）跨域，如：www.a.com 和 bbs.a.com 跨域。
③ 不同协议属于跨域，如：http://www.a.com 和 https://www.a.com。
④ Ip和域名属于跨域，如：123.23.23.12 和 www.a.com。

如何解决跨域问题

1）JSONP

JSONP是JSON with Padding的略称。它是一个非官方的协议，它允许在服务器端集成Script tags返回至客户端，通过javascript

callback的形式实现跨域访问（这仅仅是JSONP简单的实现形式）。关于jsonp的使用方式，可以参考http://blog.csdn.net/alen1985/article/details/6365394，优缺点可以参考http://blog.csdn.net/z69183787/article/details/19191385

2）添加响应头，允许跨域

addHeader('Access-Control-Allow-Origin:*');//允许所有来源访问

addHeader('Access-Control-Allow-Method:POST,GET');//允许访问的方式

3）服务器代理的方式

服务器A的test01.html页面想访问服务器B的后台action，返回“test”字符串，此时就出现跨域请求，浏览器控制台会出现报错提示，由于跨域是浏览器的同源策略造成的，对于服务器后台不存在该问题，可以在服务器A中添加一个代理action，在该action中完成对服务器B中action数据的请求，然后在返回到test01.html页面。

4）CORS

12、你如何处理后台接口返回的数据

13开发过程中遇到的内存泄露情况，如何解决的?

解决：变量导致的内存泄露，将变量清除 a = null 即可。

事件监听导致的内存泄露，监听后移除即可。

出现内存泄露的的几种常见情况：

1.全局变量：如未声明变量

2.闭包：闭包可以读取函数内部的变量，然后让这些变量始终保存在内存中。如果在使用结束后没有将局部变量清除，就可能导致内

存泄露。

3）事件监听：对同一个事件重复监听，但是忘记移除，会导致内存泄露。

4）其他原因

console.log打印的对象不能被垃圾回收，可能会导致内存泄露。

setInterval也可能会导致内存泄露。

14、网站如何优化?

15、网站加载速度慢如何优化

16、 网站seo优化有哪些方法?

****17. 一次完整的HTTP事务是怎样的一个过程?**

基本流程:

a. 域名解析

b. 发起TCP的3次握手

c. 建立TCP连接后发起http请求

d. 服务器端响应http请求，浏览器得到html代码

e. 浏览器解析html代码，并请求html代码中的资源

f. 浏览器对页面进行渲染呈现给用户

18.你以前用过哪些js库和框架

库：库是更多是一个封装好的特定的集合，提供给开发者使用，而且是特定于某一方面的集合（方法和函数），库没有控制权，控制权在使用者手中，在库中查询需要的功能在自己的应用中使用，我们可以从封装的角度理解库；

框架：框架顾名思义就是一套架构，会基于自身的特点向用户提供一套相当于叫完整的解决方案，而且控制权的在框架本身，使用者要找框架所规定的某种规范进行开发。

19、ajax请求时，如何解释json数据

后台返回的是json字符串，使用JSON.parse()方法解析就可以了。

20. 如何阻止事件冒泡和默认事件

使用event.stopPropagation()起到阻止捕获和冒泡阶段中当前事件的进一步传播。使用event.preventDefault()可以取消默认事件。

21、解释jsonp的原理，以及为什么不是真正的ajax

①ajax的核心是： 通过XmlHttpRequest获取非本页内容，

jsonp的核心： 动态添加<script>标签来调用服务器提供的js脚本。

②ajax和jsonp的调用方式很像，目的一样，都是请求url，然后把服务器返回的数据进行处理，因此jquery和ext等框架都把jsonp作为ajax的一种形式进行了封装；

22、”==”和“===”的不同

前者会自动转换类型

后者不会

23、javascript的同源策略

所谓的 同源 是指域名、协议、端口号 相同。不同的客户端脚本（javascript，ActionScript）在没有授权的情况下，不能读取对方资源。简单来说，浏览器允许包含在页面A的脚本访问第二个页面B的数据资源，这一切是建立在A和B页面是同源的基础上。

概念:同源策略是客户端脚本（尤其是Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。

这里的同源策略指的是：协议，域名，端口相同，同源策略是一种安全协议。

指一段脚本只能读取来自同一来源的窗口和文档的属性。

24.你知道的jquery插件有哪些

25.你们公司前后端是如何配合工作的?

在前后端分离的情况下，后端同学的任务是把数据和文件做持久化存储，并提供操作数据数据的接口。前端同学负责请求数据，然后

对数据做处理和适配，最后再把数据呈现出来。其实就是以接口为界限，后端负责接口的制作，前端负责接口的调用。

在实际开发中，一般是前后端同学拿到需求后，一起来确定各个功能点需要的接口，然后再逐个细化出各个接口的输入输出字段，最后由后端同学生成一个接口文档。

26、Javascript如何实现继承?

1）原型链继承

核心： 将父类的实例作为子类的原型

特点:

a非常纯粹的继承关系，实例是子类的实例，也是父类的实例

b父类新增原型方法/原型属性，子类都能访问到

c简单，易于实现

缺点:

a要想为子类新增属性和方法，必须要在new Animal()这样的语句之后执行，不能放到构造器中

b无法实现多继承

c来自原型对象的所有属性被所有实例共享（d创建子类实例时，无法向父类构造函数传参

2）构造继承

核心：使用父类的构造函数来增强子类实例，等于是复制父类的实例属性给子类（没用到原型）

特点:

a解决了1中，子类实例共享父类引用属性的问题

b创建子类实例时，可以向父类传递参数

c可以实现多继承（call多个父类对象）

缺点：

a实例并不是父类的实例，只是子类的实例

b只能继承父类的实例属性和方法，不能继承原型属性/方法

c无法实现函数复用，每个子类都有父类实例函数的副本，影响性能

3）实例继承

核心：为父类实例添加新特性，作为子类实例返回

特点：

a不限制调用方式，不管是new 子类()还是子类(),返回的对象具有相同的效果

缺点：

a实例是父类的实例，不是子类的实例

b不支持多继承

4）拷贝继承

特点：

a支持多继承

缺点：

a效率较低，内存占用高（因为要拷贝父类的属性）

b无法获取父类不可枚举的方法（不可枚举方法，不能使用for in 访问到）

5）组合继承

核心：通过调用父类构造，继承父类的属性并保留传参的优点，然

后通过将父类实例作为子类原型，实现函数复用

特点：

a弥补了方式2的缺陷，可以继承实例属性/方法，也可以继承原型属性/方法

b既是子类的实例，也是父类的实例

c不存在引用属性共享问题

d可传参

e函数可复用

缺点：

a调用了两次父类构造函数，生成了两份实例（子类实例将子类原型上的那份屏蔽了）

6）寄生组合继承

核心：通过寄生方式，砍掉父类的实例属性，这样，在调用两次父类的构造的时候，就不会初始化两次实例方法/属性，避免的组合继承的缺点

特点：

a堪称完美

缺点：

a实现较为复杂

27. Javascript创建对象的几种方式?

第一种模式：工厂方式

```
var lev=function(){ return "脚本之家"; };
function Parent(){ var Child = new Object(); Child.name="脚本";
Child.age="4"; Child.lev=lev; return Child; };
```

```
var x = Parent();
alert(x.name);
alert(x.lev());
```

第二种模式：构造函数方式

```
var lev=function(){ return "脚本之家"; };
function Parent(){ this.name="脚本"; this.age="30";
this.lev=lev; };
var x =new Parent();
alert(x.name);
alert(x.lev());
```

第三种模式：原型模式

```
var lev=function(){ return "脚本之家"; };
function Parent(){ };
Parent.prototype.name="李小龙";
Parent.prototype.age="30";
Parent.prototype.lev=lev;
var x =new Parent();
alert(x.name);
alert(x.lev());
```

第四种模式：混合的构造函数，原型方式（推荐）

```
function Parent(){ this.name="脚本"; this.age=4; };
Parent.prototype.lev=function(){ return this.name; };
var x =new Parent();
alert(x.lev());
```

第五种模式：动态原型方式

```
function Parent(){ this.name="脚本"; this.age=4; if(typeof
```

```
Parent._lev=="undefined"){
                Parent.prototype.lev=function(){ return this.name; }
Parent._lev=true; }
};
var x =new Parent();
alert(x.lev());
```

28、JQuery的源码看过吗？能不能简单概况一下它的实现原理？

JQ本身就是一个类，在外面使用的$和jQuery是同一个东西，JQ中提供的方法分为两部分：

写在jQuery原型上的方法，专门给JQ的实例使用

写在jQuery私有属性上的，通过$.xxx可以获取进行操作等 jQuery() / $() =>创建JQ的实例，需要传递两个参数，第一个参数一般是选择器内容；第二个参数是获取的上下文，如果不传递默认是document；==>“此操作通俗的叫法：通过JQ选择器获取元素”

返回结果是一个类数组(它也是JQ的实例)，这个类数组是JQ自己去创建的，里面有一些自己特定的属性：length/context/selector/prevObject...

获取到的结果我们叫做JQ对象(JQ实例)，可以调取JQ原型上提供的方法，但是它不是原生的JS对象，不能调取浏览器提供的默认属性方法，当然原生JS也不能调用JQ上提供的属性方法；

29、jquery中如何将数组转化为json字符串，然后再转化回来？

JSON.stringify 把一个数组对象转换成json字符串，

JSON.parse 把一个json字符串解析成对象。

30、谈一下Jquery中的bind(),live(),delegate(),on()的区别？

相同点：

1.都支持单元素多事件的绑定；空格相隔方式或者大括号替代方式;

2.均是通过事件冒泡方式，将事件传递到document进行事件的响应;

区别:

.bind()是直接绑定在元素上

.live()则是通过冒泡的方式来绑定到元素上的。更适合列表类型的，绑定到document DOM节点上。和.bind()的优势是支持动态数据。

.delegate()则是更精确的小范围使用事件代理，性能优于.live()

.on()则是最新的1.9版本整合了之前的三种方式的新事件绑定机制

31、Jquery与jQuery UI有啥区别?

(1) jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。

(2) jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等。

(3) jQuery本身注重于后台，没有漂亮的界面，而jQuery UI则补充了前者的不足，他提供了华丽的展示界面，使人更容易接受。既有强大的后台，又有华丽的前台。jQuery UI是jQuery插件，只不过专指由jQuery官方维护的UI方向的插件。

32、Zepto如何理解的

33、requireJS是怎么理解的

①requireJs 属于异步非阻塞加载模块，AMD模式，原理类似队列

式进出。

这里只针对源码中的 “require”与 "define" 两个关键词做分析,require作为主模块，运行加载load函数，获取script标签载入js，分析当前模块加载状态（loading-文件正在加载，未载入完成,enabling-对该模块的依赖进行加载,defining-对处理的主程序模块进行加载，并运行模块中的callback）当加载完成后，再进行模块化分析。

其中define定义中有个queue队列，把需要加载的小模块放入队列中，并于require 中的全局变量defined中的模块进行对比，然后判断;

如果全局defined定义了该模块，就实例化Module，调用其中的工厂模式，来改变当前模块的加载状态，然后从全局defined中移除该模块，该过程算完成。

否则是全局defined没有定义，则会调用makeModuleMap去定义，然后再实现前一种逻辑，直到修改完成状态。然后再去load其他文件，通道一直是队列的进进出出，直到所有文件加载完成并实例化。

②使用requirejs好处：1、放置JS加载阻塞页面渲染。2、减少多行JS代码引入。

③require会定义三个变量：define,require,requirejs，其中require === requirejs，一般使用require更简短

define 定义一个模块

require 加载依赖模块，并执行加载完后的回调函数

④功能：声明不同js文件之间的依赖；可以按需、并行、延时载入js库；可以让我们的代码以模块化的方式组织

34、jQuery的优点

1）轻量级：JQuery非常轻巧，采用DeanEdwards编写的Packer压缩后，大小不到30KB,如果使用Min版并且在服务器端启用Gzip压缩后，大小只有18KB。

2）强大的选择器：JQuery允许开发者使用从CSS1到CSS3几乎所有的选择器，以及JQuery独创的高级而且复杂的选择器，另外还可以加入插件使其支持XPath选择器，甚至开发者可以编写属于自己的选择器。由于JQuery支持选择器这一特性，因此有一定CSS经验的开发人员可以很容易的切入到JQuery的学习中来。

3）出色的DOM操作的封装：JQuery封装了大量常用的DOM操作，使开发者在编写DOM操作相关程序的时候能够得心应手。JQuery轻松地完成各种原本非常复杂的操作，让JavaScript新手也能写出出色的程序。

4）完善的Ajax：JQuery将所有的Ajax操作封装到一个函数$.ajax()里，使得开发者处理Ajax的时候能够专心处理业务逻辑而无需关心复杂的浏览器兼容性和XMLHttpRequest对象的创建和使用的问题。

5）不污染顶级变量：JQuery只建立一个名为JQuery的对象，其所有的函数方法都在这个对象之下。其别名$也可以随时交流控制权，绝对不会污染其他的对象。该特性是JQuery可以与其他JavaScript库共存，在项目中放心地引用而不需要考虑到后期的冲突。

6）出色的浏览器兼容性：作为一个流行的JavaScript库，浏览器的兼容性是必须具备的条件之一。JQuery能够在IE6.0+,FF 2+,Safari2.+和Opera9.0+下正常运行。JQuery同时修复了一些浏览器之间的的差异，使开发者不必在开展项目前建立浏览器兼容库。

7）链式操作方式：JQuery中最有特色的莫过于它的链式操作方式

——即对发生在同一个JQuery对象上的一组动作，可以直接接连写无需要重复获取对象。这一特点使得JQuery的代码无比优雅。

8）隐式迭代：当用JQuery找到带有“.myClass”类的全部元素，然后隐藏他们时。无需循环遍历每一个返回的元素。相反，JQuery里的方法都被设计成自动操作的对象集合，而不是单独的对象，这使得大量的循环结构变得不再必要，从而大幅度地减少代码量。

9）行为层与结构层的分离：开发者可以使用选择器选中元素，然后直接给元素添加事件。这种将行为层与结构层完全分离的思想，可以使JQuery开发人员和HTML或其他页面开发人员各司其职，摆脱过去开发冲突或个人单干的开发模式。同时，后期维护也非常方便，不需要在HTML代码中寻找某些函数和重复修改HTML代码。

10）丰富的插件支持：JQuery的易扩展性，吸引了来自全球开发者来编写JQuery的扩展插件。目前已经有超过几百种官方插件支持，而且还不断有新插件面试。

11）完善的文档：JQuery的文档非常丰富，现阶段多位英文文档，中文文档相对较少。很多热爱JQuery的团队都在努力完善JQuery中文文档，例如JQuery的中文API。

12）开源：JQuery是一个开源的产品，任何人都可以自由地使用并提出修改意见。

35、懒加载的原理?

意义：懒加载的主要目的就是作为服务器前端的优化，减少请求次数或者延迟请求数。

实现原理:先加载一部分数据，当触发某个条件时利用异步

(async)加载剩余的数据，新得到的数据不会影响原有数据的显示，同时最大幅度的减少服务器端资源耗用。

实现方式：

1.第一种是纯粹的延迟加载，使用setTimeOut和setInterval进行加载延迟。

2.第二种是条件加载，符合某种条件，或是出发某些事件才开始异步加载。

3.第三种是可视区加载，仅记载用户的可视区域，这个主要监控滚动条来实现，一般会距用户看到某些图片前的一段距离时开始进行记载，这样就可保证用户拉下时正好可以看到加载完毕后的图片或是内容。

36、数组的所有遍历方法，包括es6

ES6（for-of）、ES5（forEach、for-in）、通用（for（i=0；i

37. 说下事件委托是什么？有什么好处？

定义：利用事件冒泡处理动态元素事件绑定的方法，专业术语叫事件委托，简单点来说就是给父元素绑定事件，用来监听子元素的冒泡事件，并找到是哪个子元素的事件，更加通俗一点那就是让别人来做，这个事件本来是加在某些元素上的，然而你却加到别人身上来做，完成这个事件。

好处：事件委托技术可以避免对每个子元素添加事件监听器，减少操作DOM节点的次数，从而减少浏览器的重绘和重排，提高代码的性能。

使用事件委托，只有父元素与DOM存在交互，其他的操作都是在

JS虚拟内存中完成的，这样就大大提高了性能。

38、开发过程中有什么编码规范?

（https://www.jianshu.com/p/97b0a70b5bd1）

html规范、CSS规范、class命名、CSS书写规范

39.实现多个标签页的通信?

（就是localStorage和cookie）

40. 在浏览器中输入url到整个页面显示在用户面前时，这个过程发生了什么?

浏览器根据请求的URL交给DNS域名解析，找到真实IP，向服务器发起请求;

服务器交给后台处理完成后返回数据，浏览器接收文件（HTML、JS、CSS、图象等）;

浏览器对加载到的资源（HTML、JS、CSS等）进行语法解析，建立相应的内部数据结构（如HTML的DOM）;

载入解析到的资源文件，渲染页面，完成。

41. 常见的http请求方式，以及区别?

1、opions　返回服务器针对特定资源所支持的HTML请求方法　或web服务器发送*测试服务器功能（允许客户端查看服务器性能）

2、Get　向特定资源发出请求（请求指定页面信息，并返回实体主体）

3、Post　向指定资源提交数据进行处理请求（提交表单、上传文件），又可能导致新的资源的建立或原有资源的修改

4、Put　向指定资源位置上上传其最新内容（从客户端向服务器传

送的数据取代指定文档的内容）

5、Head　与服务器索与get请求一致的相应，响应体不会返回，获取包含在小消息头中的原信息（与get请求类似，返回的响应中没有具体内容，用于获取报头）

6、Delete　请求服务器删除request-URL所标示的资源*（请求服务器删除页面）

7、Trace　回显服务器收到的请求，用于测试和诊断

8、Connect　HTTP/1.1协议中能够将连接改为管道方式的代理服务器

*****42. 记住几个常用的状态码(200，400,401,404,500)**

200 OK：客户端请求成功

400 Bad Request：客户端请求有语法错误，不能被服务器所理解

401　Unauthorized：请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

403 Forbidden：服务器收到请求，但是拒绝提供服务

404 Not Found：请求资源不存在

500 Internal Server Error：服务器发生不可预期的错误

503 Server Unavailable：服务器当前不能出来客户端的请求，一段时间后可能恢复正常。

43. Es6你用过什么？具体说下

44. 你对前端工程师的理解是什么？在代码方面有什么难度？

三阶段面试题：

1. *** 你对MVC、MVVM的理解？

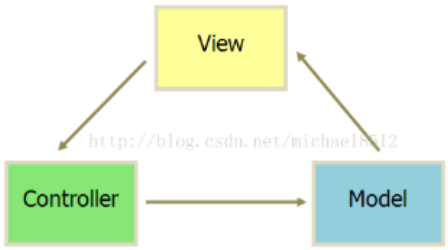
1、MVC

View 传送指令到 Controller

Controller 完成业务逻辑后，要求 Model 改变状态

Model 将新的数据发送到 View，用户得到反馈

所有的通信都是单向的。

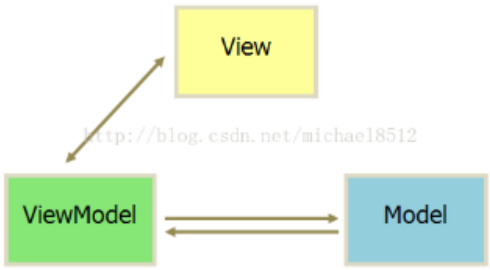


2、MVVM

View：UI界面

ViewModel：它是View的抽象，负责View与Model之间信息转换，将View的Command传送到Model；

Model：数据访问层



2. 你知道哪些htlm5的移动端框架。

Weui、SUI Mobile、Amaze UI、element、antd-moblie.....

3. 知道各种JS框架(Angular, Backbone, Ember, React, Meteor, Knockout...)么？能讲 出他们各自的优点和缺点么？

angular、backbone、knockout都是完整的MV*框架

angular是双向数据绑定的，backbone、knockout是单向数据绑定的

React只是单纯的View层

4. 常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？

①使用率较高的框架有jQuery、YUI、Prototype、Dojo、Ext.js、Mootools等。尤其是jQuery，超过91%。②开发工具HBuilder(H5）、WebStorm（JS)、Sublime、DW……③市选择插件，汽车型号选择插件、幻灯片插件。弹出层。（写过开源程序，加载器，js引擎更好）

5. 你怎么看待Web App 、hybrid App、Native App？

Web App 指采用Html5语言写出的App，不需要下载安装。类似于现在所说的轻应用。生存在浏览器中的应用，基本上可以说是触屏版的网页应用。

优点：（1）开发成本低，　（2）更新快，　（3）更新无需通知用户，不需要手动升级，　（4）能够跨多个平台和终端。

缺点：（1）临时性的入口　（2）无法获取系统级别的通知，提醒，动效等等　（3）用户留存率低　（4）设计受限制诸多　（5）体验较差

Hybrid APP指的是半原生半Web的混合类App。需要下载安装，看上去类似Native　App，但只有很少的UI　Web　View，访问的内容是Web 。

例如Store里的新闻类APP，视频类APP普遍采取的是Native的框架，Web的内容。

Hybrid App 极力去打造类似于Native App 的体验，但仍受限于技术，网速，等等很多因素。尚不完美。

Native APP 指的是原生程序，一般依托于操作系统，有很强的交互，是一个完整的App，可拓展性强。需要用户下载安装使用。

优点：（1）打造完美的用户体验 （2）性能稳定 （3）操作速度快，上手流畅 （4）访问本地资源（通讯录，相册） （5）设计出色的动效，转场， （6）拥有系统级别的贴心通知或提醒 （7）用户留存率高

缺点：（1）分发成本高（不同平台有不同的开发语言和界面适配） （2）维护成本高（例如一款App已更新至V5版本，但仍有用户在使用V2， V3， V4版本，需要更多的开发人员维护之前的版本） （3）更新缓慢，根据不同平台，提交-审核-上线 等等不同的流程，需要经过的流程较复杂

6. 你移动端前端开发的理解？（和 Web 前端开发的主要区别是什么？）

移动WEB前端开发属于HTML5开发，使用HTML+CSS+JavaScript三种语言编写，该职位偏向于移动端网页开发，包括H5专题页、移动前端网页，广告落地页、Hybrid App等；

Tips：由于原生APP开发需要开发两个客户端（Android和iOS），产品开发周期和成本相对较高，因此出现了Hybrid App。Hybrid App可以使用PhoneGap将网页直接打包成Android原生APP和iOS原生APP。

WEB前端开发也是属于HTML5开发，使用HTML+CSS+JavaScript三种

语言编写，主要偏向PC端的网页开发，包括PC端网页开发、PC端后台管理页面等，也会涉及部分的移动端网页开发。

7. 平时如何管理你的项目?

先期团队必须确定好全局样式（globe.css），编码模式(utf-8) 等；

编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；

标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；

页面进行标注（例如 页面 模块 开始和结束）；

CSS跟HTML 分文件夹并行存放，命名都得统一（例如style.css）；

JS 分文件夹存放 命名以该JS功能为准的英文翻译。

图片采用整合的 images.png png8 格式文件使用 尽量整合在一起使用方便将来的管理

8. 说说最近最流行的一些东西吧？常去哪些网站?

Node.js、Mongodb、npm、MVVM、MEAN、three.js、React；牛客网、Github、 CSDN

9. 简单描述一下你做过的移动APP项目研发流程?

一个完整的App 开发过程要经过项目启动→建立产品原型→形成效果图→进入研发阶段→研发成功后进入测试阶段→测试过后将问题反馈给研发人员进行调整-→多次测试确认没有bug。这就是一个完整的app开发过程。

10. 你认为怎样才是全端(全栈式)工程师（Full Stack developer）?

11. 介绍一个你最得意的作品吧?

12. 项目中遇到国哪些印象深刻的技术难题，具体是什么问题，怎么解决?

13. 最近在学什么？能谈谈你未来3，5年给自己的规划吗?

14. 模块化开发如何理解的。

所谓的模块化开发就是封装细节，提供使用接口，彼此之间互不影响，每个模块都是实现某一特定的功能。模块化开发的基础就是函数

模块化就是为了减少系统耦合度，提高高内聚，减少资源循环依赖，增强系统框架设计。

让开发者便于维护,同时也让逻辑相同的部分可复用。

模块化开发：针对js、css，以功能或业务为单元组织代码。js方面解决独立作用域、依赖管理、api暴露、按需加载与执行、安全合并等问题，css方面解决依赖管理、组件内部样式管理等问题。

15.* 如何提高网站性能**

【https://blog.csdn.net/liuyang755855737/article/details/78238987】

（一）Web前端性能优化

一般来说Web前端指网站业务逻辑之前的部分，包括浏览器加载，网站视图模型，图片服务，CDN服务等，主要优化手段有优化浏览器访问，使用反向代理，CDN等。

A、浏览器访问优化

(1)减少http请求(2)使用浏览器缓存(3)启用压缩(4)CSS放在页面最上面，JavaScript放在页面最下面(5)减少Cookie传输

B、CDN加速：CDN(Content Distribute Network,内存分发网络)的本质上仍然是一个缓存，而且将数据缓存在离用户最近的地方，是用户以最快速度获取数据，即所谓网络访问第一跳。CDN一般缓存的是静态资源，如图片，文件，CSS，Script脚本，静态网页等，但是这些文件访问频率很高，将其缓存在CDN可极大改善网页的打开速度。

C、反向代理：传统代理服务器位于浏览器一侧，代理浏览器将HTTP请求发送到互联网上，而反向代理服务器位于网站机房一侧，代理网站Web服务器接收HTTP请求。和传统代理服务器可以保护浏览器安全一样，反向代理服务器也具有保护网站安全的作用，来自互联网的访问请求必须经过代理服务器，相当于在Web服务器和可能的网络攻击之间建立了一个屏障。

除了安全功能，代理服务器也可以通过配置缓存功能加速Web请求，当用户第一次访问静态内容的时候，静态内容就被缓存在反向代理服务器上，这样当其他用户访问该静态内容的时候，就可以直接从反向代理服务器返回，加速Web请求响应速度，减轻服务器负载要

（一）应用服务器性能优化

应用服务器就是处理网站业务的服务器，网站的业务代码都部署在这里，是网站开发最复杂，变化最多的地方，优化手段主要有缓存，集群，异步等。

A、分布式缓存

(1)频繁修改的数据(2)没有热点的访问(3)数据不一致与脏读(4)缓存

可用性(5)缓存预热(6)缓存穿透

B、异步操作

C、使用集群

D、代码优化

1)使用多线程技术提升相应速度，目前主要的Web应用服务器都采用多线程的方式响应并发用户请求，因此网站开发天然就是多线程编程。有一个问题就是，在多线程任务中，如何设置合理的线程数呢？这个主要看我们的任务时什么类型，如果任务都是CPU计算型任务(我们称之为CPU密集型)，那么线程数最多不超过CPU内核数，因为启动再多线程，CPU也来不及调度；相反如果是任务需要等待磁盘操作，网络响应(我们称之为IO密集型)，那么多启动线程有助于提高任务并发度，调系统吞吐能力，改善系统性能。

(2)资源复用，常见的资源复用情况有单例对象和对象池，对象池包括常见的线程池和数据库连接池这种情况。

16. 怎么实现js的模块化编程?

17. 你有上线项目吗?

18.* 对React的理解**

react把用户界面抽象成一个个组件，如按钮组件button，对话框组件Dialog，日期组件Calendar，开发者通过组合这些组件，最终得到功能丰富、可交互的页面，通过引用jsx语法，复用组件变得非常容易，同时也能保证组件结构清晰，有了组件这层抽象，react把代码和真实渲染目标隔离开来，除了可以在浏览器端渲染到DOM来开发网页外，还能用于开发原生移动应用。

国外使用React的公司：Facebook，Flipboard，Airbnb，BBC，GitHub，Instagram，Reddit，Uber，WhatsApp，Yahoo

国内使用React的公司：支付宝，淘宝，大搜车，Teambition，豆瓣，豌豆荚

React 并不是完整的 MVC/MVVM 框架，它专注于提供清晰、简洁的 View（视图）层解决方案

（一）react的特点有：

1）声明式设计　　–React采用声明范式，可以轻松描述应用（自动dom操作）。

2）高效　–React通过对DOM的模拟(虚拟dom)，最大限度地减少与DOM的交互。

3）灵活 –React可以与已知的库或框架很好地配合。

4）JSX – JSX 是 JavaScript 语法的扩展。

5）组件 – 通过 React 构建组件，使得代码更加容易得到复用，能够很好的应用在大项目的开发中。

6）单向响应的数据流 – React 实现了单向响应的数据流，从而减少了重复代码，这也是它为什么比传统数据绑定更简单。

7）React的核心是组件，精髓是函数式编程，组件的设计目的是提高代码复用率、降低测试难度和代码复杂度。

（二）函数式编程的好处：

1）代码简洁，开发快速

2）接近自然语言，易于理解

3）更方便的代码管理

4）易于“并发编程 ”

5）代码的热升级

（三）生命周期

react生命周期函数:

A、初始化阶段:

getDefaultProps 取得默认属性：低版本

getInitialState 初始化状态：低版本

componentWillMount 即将进入dom

render 描画dom

componentDidMount 已经进入dom

B、运行中状态:

componentWillReceiveProps（组件将要接收新值）：已加载组件收到新的参数时调用，可以传参

shouldComponentUpdate(组件是否更新，唯一影响整个项目的功能，决定视图的更新)：可以传参，返回true后面的继续执行，返回false，后面不再执行

componentWillUpdate：组件即将更新

render：组件重新描绘

componentDidUpdate：组件更新完毕时运行的函数

C、销毁阶段:

componentWillUnmount:组件即将销毁

四.redux的优缺点

redux的优点:

规范了流程；减少了手动编码量，提高代码效率

redux的缺点：

一个组件所需要的数，必须由父组件传过来，而不能像flux中直接从store获取；当一个组件相关数据更新时，即时父组件不需要用到这个组件，父组件还是会重新render，可能会有效率影响，或者需要写复杂的shouldComponentUpdate进行判断。

19.* 对Vue的理解**

①vue是一个构建数据驱动的web界面的渐进式框架，它的目标是通过尽可能简单的API实现响应的数据绑定和组合的视图组件。Vue基于MVVM模式，M指的是数据，V指的是视图，VM是视图模型，将数据绑定视图上（双向绑定）；这个框架着重于VM部分。

官方文档中的一句话:Vue.js是一套构建用户界面的渐进式框架。

vue基础语法：

每个 Vue.js 应用都是通过构造函数 Vue 创建一个 Vue 的根实例 启动的

Var vm=new Vue({})

②Vue的特点：（1） 简洁 （2） 轻量 （3）快速 （4） 数据驱动 （5） 模块友好 （6） 组件化Vue

③Vue只提供一个Api就是Vue，它是类，我们要想获取Vue实例化对象，只能通过new关键字创建；通过el定义一个Vue视图的容器元素，可以传递css选择器，id选择，类选择器，元素名称选择器等等；页面中有多个符合条件选择器，vue只会捕获第一个符合条件的选择器对应的元素选择器；实现了将视图到视图模型的绑定

20. 对面向对象的理解

①定义：面向对象语言（Object-Oriented Language）是一类以对象作为基本程序结构单位的程序设计语言，指用于描述的设计是以对象为核心，而对象是程序运行时刻的基本成分。

②面向对象有三大特性，封装、继承和多态。

21. 你的项目，Redux怎么分层

22. socket断开后，如何重连

23. 如何实现vue无限级树型导航(2种及以上方法)

24. 嵌套路由和嵌套组件是否有区别：

组件是可以有数据交互的；路由嵌套 是没办法与父组件进行数据交互的

25. 表格封装是思路需要的注意思想

26.* v-model**

v-model是Vue用于表单元素上创建双向数据绑定，它本质是一个语法糖，在单向数据绑定的基础上，增加了监听用户输入事件并更新数据的功能。

v-model只不过是一个语法糖而已,真正的实现靠的还是v-bind:绑定响应式数据，触发 input 事件，并传递数据 (核心和重点)

27. vuex 组件A和组件B，组件A由渲染，B没有， 问题：组件A能不能拿到组件B的数据

28. 继承和原型链继承的优点

29. vue data和method里面的this指向是否一样

将事件写在methods中，这个时候this的指向指向Vue实例，但是当

将事件写到data中，this指向却指向了window

30. post收不到数据什么原因

原因就是这次的接口使用 java spring mvc ，并且在这个方法上使用了注解 @RequestParam

或者：首先是需要使用 URLSearchParams 处理 Post 参数。

第二个地方是，Content-Type 的值必须为 application/x-www-form-urlencoded，否则会出现问题。

31. vue-resource和拦截器

在vue项目使用vue-resource的过程中，临时增加了一个需求，需要在任何一个页面任何一次http请求，增加对token过期的判断，如果token已过期，需要跳转至登录页面。如果要在每个页面中的http请求操作中添加一次判断，那么会是一个非常大的修改工作量。那么vue-resource是否存在一个对于任何一次请求响应捕获的的公共回调函数呢？答案是有的!

vue-resource的interceptors拦截器的作用正是解决此需求的妙方。

在每次http的请求响应之后，如果设置了拦截器如下，会优先执行拦截器函数，获取响应体，然后才会决定是否把response返回给then进行接收。那么我们可以在这个拦截器里边添加对响应状态码的判断，来决定是跳转到登录页面还是留在当前页面继续获取数据。

```
下边代码添加在main.js中
Vue.http.interceptors.push((request, next) => {
  console.log(this)//此处this为请求所在页面的Vue实例
  // modify request
  request.method = 'POST';//在请求之前可以进行一些预处理和配置

  // continue to next interceptor
    next((response) => {//在响应之后传给then之前对response进行修改和逻辑判断。对于token时候已过期的判断，就添加在此处，页面中任何一次http请求都会先调用此处方法

        response.body = '...';
       return response;

  });
});
```

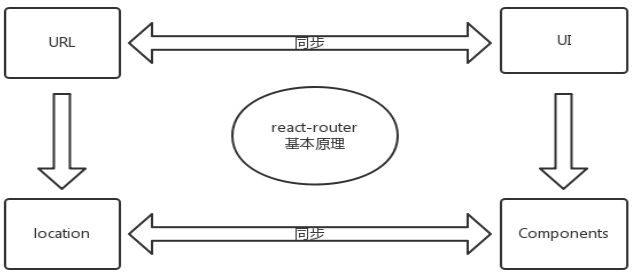
32. vue-route 路由拦截是?

beforeEach（to，from，next）

33.*** route的底层实现原理

vue-router： vue-router 是通过 Vue.use 的方法被注入进 Vue 实例中，在使用的时候我们需要全局用到 vue-router 的 router-view 和 router-link 组件，以及 this.$router/$route 这样的实例对象。

react-router：实现URL与UI界面的同步。其中在react-router中，URL对应Location对象，而UI是由react components来决定的，这样就转变成location与components之间的同步问题。



34.*** 数据驱动的原理

vuejs在实例化的过程中，会对实例化对象选项中的data 选项进行

遍历，遍历其所有属性并使用Object.defineProperty 把这些属性全部转为 getter/setter。同时每一个实例对象都有一个watcher实例对象，他会在模板编译的过程中,用getter去访问data的属性，watcher此时就会把用到的data属性记为依赖，这样就建立了视图与数据之间的联系。当之后我们渲染视图的数据依赖发生改变（即数据的setter被调用）的时候，watcher会对比前后两个的数值是否发生变化，然后确定是否通知视图进行重新渲染这样就实现了所谓的数据对于视图的驱动。通俗地讲，它意味着我们在普通 HTML 模板中使用特殊的语法将 DOM “绑定”到底层数据。一旦创建了绑定，DOM 将与数据保持同步。每当修改了数据，DOM 便相应地更新。这样我们应用中的逻辑就几乎都是直接修改数据了，不必与 DOM 更新搅在一起。这让我们的代码更容易撰写、理解与维护。

35. {{}}插值法的实现原理

主要作用是进行数据绑定，最常见的形式是使用“Mustache”语法（双大括号）的文本插值。例如：Message:{{msg}}Mustache标签将会被替代为对应数据对象上msg属性（msg定义在data对象中）的值。无论何时，绑定的数据对象上msg属性发生了改变，插值处的内容都会更新。

36. Cordova

37. canvas

<canvas> 标签用于绘制图像（通过脚本，通常是 JavaScript）。不过，<canvas> 元素本身并没有绘制能力（它仅仅是图形的容器），必须使用脚本来完成实际的绘图任务。

38. mongodb的了解

MongoDB 是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统；

档型类json格式存储数据，修改起来比传统的关系型数据库更方便

MongoDB(文档型数据库)：提供可扩展的高性能数据存储；特点：

1、基于分布式文件存储2、高负载情况下添加更多节点，可以保证服务器性能3、将数据存储为一个文档

39. mongodb和mysql的对比

MySQL是传统的关系型数据库，MongoDB则是非关系型数据库

mongodb以BSON结构（二进制）进行存储，对海量数据存储有着很明显的优势。

对比传统关系型数据库,NoSQL有着非常显著的性能和扩展性优势，

与关系型数据库相比，MongoDB的优点有：

①弱一致性（最终一致），更能保证用户的访问速度：

②文档结构的存储方式，能够更便捷的获取数据。

40. swiper插件

41.*** vue通信的几种方法

一、通过路由带参数进行传值

①两个组件 A和B,A组件通过query把orderId传递给B组件（触发事件可以是点击事件、钩子函数等）

```
this.$router.push({ path: '/conponentsB', query: { orderId: 123 } })
// 跳转到B
```

②在B组件中获取A组件传递过来的参数

二、通过设置 Session Storage缓存的形式进行传递

①两个组件A和B，在A组件中设置缓存orderData

```
const orderData = { 'orderId': 123, 'price': 88 }
sessionStorage.setItem('缓存名称', JSON.stringify(orderData))
```

②B组件就可以获取在A中设置的缓存了

```
const dataB = JSON.parse(sessionStorage.getItem('缓存名称'))
```

此时 dataB 就是数据 orderData

可以百度下 Session Storage（程序退出销毁） 和 Local Storage（长期保存） 的区别。

三、组件之间的传值

Vue的通信方式，也可以说是传参方式。主要分为四种：

父子传参：父控制子，通过子组件的props属性，抛出子组件自定义标签属性，来接收父组件的操作状态。

子父传参：子控制父，子组件绑定自定义事件，来处理父组件的方法函数，通过.$emit('自定义事件'，[参数])来触发属于自己的自定义事件。

不同组件之间传值：通过eventBus（小项目少页面用eventBus，大项目多页面使用 vuex）

42. 前端跨域解决方法

1）通过jsonp跨域

2）document.domain + iframe跨域

3）location.hash + iframe

4）window.name + iframe跨域

5）postMessage跨域

6）跨域资源共享（CORS）

7）nginx代理跨域

8）nodejs中间件代理跨域

9）WebSocket协议跨域

43. CORS 与 JSONP 的对比

（1） JSONP只能实现GET请求，而CORS支持所有类型的HTTP请求。

（2）使用CORS，开发者可以使用普通的XMLHttpRequest发起请求和获得数据，比起JSONP有更好的错误处理。

（3）JSONP主要被老的浏览器支持，它们往往不支持CORS，而绝大多数现代浏览器都已经支持了CORS

44. vue 混合

混合 (mixins) 是一种分发 Vue 组件中可复用功能的非常灵活的方式。混合对象可以包含任意组件选项。以组件使用混合对象时，所有混合对象的选项将被混入该组件本身的选项。
可复用性 混合是一种灵活的分布式复用 Vue 组件的方式。混合对象可以包含任意组件选项。以组件使用混合对象时，所有混合对象的选项将被混入该组件本身的选项。当组件和混合对象含有同名选项时，这些选项将以恰当的方式混合。比如，同名钩子函数将混合为一个数组，因此都将被调用

45. vue的生命周期

Vue生命周期是指vue实例对象从创建之初到销毁的过程，vue所有功能的实现都是围绕其生命周期进行的，在生命周期的不同阶段调用对应的钩子函数可以实现组件数据管理和DOM渲染两大重要功能。

e生命周期可以分为八个阶段，分别是：

beforeCreate（创建前）、created（创建后）、beforeMount(载入前)、mounted（载入后）、beforeUpdate（更新前）、updated（更新后）、beforeDestroy（销毁前）、destroyed（销毁后）

46. 前端的优化

1）减少HTTP请求

*CSS Sprites

2）使用CDN

CDN（内容发布网络）是一组分布在多个不同地理位置的Web服务器，用于更加有效地向用户发布内容。在优化性能时，向特定用户发布内容的服务器的选择基于对网络慕课拥堵的测量。例如，CDN可能选择网络阶跃数最小的服务器，或者具有最短响应时间的服务器。

CDN还可以进行数据备份、扩展存储能力，进行缓存，同时有助于缓和Web流量峰值压力。

CDN的缺点：

响应时间可能会受到其他网站流量的影响。CDN服务提供商在其所有客户之间共享Web服务器组。

如果CDN服务质量下降了，那么你的工作质量也将下降

无法直接控制组件服务器

3）避免页面跳转

4）缓存Ajax

5）延迟加载

6）提前加载

7）减少DOM元素数量

8）根据域名划分内容

9）减少iframe数量

优点：

可以用来加载速度较慢的内容，例如广告。

安全沙箱保护。浏览器会对iframe中的内容进行安全控制。

脚本可以并行下载

缺点：

即使iframe内容为空也消耗加载时间

会阻止页面加载

没有语义

10）避免404

47. vue页面缓存

keep-alive实现页面缓存（指定场景不刷新列表页）

keep-alive是vue内置的一个组件，可以使被它包含的组件处于保留状态，或避免被重新渲染

48. watch 监听不到变化

很早之前就遇到一个问题，父组件向子组件传值，子组件监听传入对象的某个属性。发现子组件使用deep watch都不能观察到对象中属性的变化，今天终于找到为啥出现这种问题和解决办法了。

为啥出现这种问题？受 ES5 的限制，Vue.js 不能检测到对象属性的添加或删除。请参照https://v1-cn.vuejs.org/guide/

reactivity.html

解决方式：

不过，有办法在实例创建之后**添加属性并且让它是响应的。**

对于 Vue 实例，可以使用 `$set(key, value)` 实例方法：

```
vm.$set('b', 2)
// `vm.b` 和 `data.b` 现在是响应的
```

对于普通数据对象，可以使用全局方法 `Vue.set(object, key, value)`：

```
Vue.set(data, 'c', 3)
// `vm.c` 和 `data.c` 现在是响应的
```

有时你想向已有对象上添加一些属性，例如使用 `Object.assign()` 或 `_.extend()` 添加属性。但是，添加到对象上的新属性不会触发更新。这时可以创建一个新的对象，包含原对象的属性和新的属性：

```
// 不使用 `Object.assign(this.someObject, { a: 1, b: 2 })`
this.someObject = Object.assign({}, this.someObject, { a: 1, b: 2 })
```

49.***vue的坑

1.1 prop值的改变--不是立即

如果父组件中给子组件传递了一个prop的值，然后调用子组件的方法去获取该值，会发现值没有立即改变。

解决方法：

1. 可以监听值的改变去调用相应子组件的方法
2. 将子组件相关方法的调用放在nextTick里面

1.2 对象中某属性值的监听

普通的`watch`中只能监听到某对象的变化才会调用，当想监听对象以及对象中属性的变化都调用函数时，可以使用`deep:true`

```
data() {
    return {
        bet: {
            pokerState: 53,
            pokerHistory: 'local'
        }
    }
},
watch: {
    bet: {
        handler(newValue, oldValue) {
            console.log(newValue)
        },
        deep: true
    }
}
```

49. 跨域是什么？解释下解决跨域的方法？关于vue的跨域有什么解决方法？

50. Vue插件的封装的一些思路，怎么做？

51. 在vue项目上遇到的一些比较难的问题，怎么解决的？

52. Vue组件之间的通信方式？

53. Vue和react的对比？

54. 虚拟dom的原理？为什么虚拟dom比操作dom节点好？

55. 双向绑定的原理，和react的单向数据流有什么区别？

56. Vue路由跳转方式？动态路由以及参数的传递

57. 关于vue的webpack的配置？

58. elementUI和ant Design有什么不同的地方？

59.* Keep-alive缓存**

<keep-alive></keep-alive> 包裹动态组件时，会缓存不活动的组件实例,主要用于保留组件状态，避免重新渲染。
用自己的理解说
比如有一个列表和一个详情，那么用户就会经常执行打开详情=>返回列表=>打开详情...这样的话列表和详情都是一个频率很高的页

面，那么就可以对列表组件使用<keep-alive></keep-alive>进行缓存，这样用户每次返回列表的时候，都能从缓存中快速渲染，而不是重新渲染，这样就会减轻服务器压力，提高性能。

61. 如果有一个数组对象生成的列表页，里面所有列表项都可以拖拽改变位置，同时如何得到位置改变后的数组?

60. 平时有封装过什么组件?

61. 项目中的优化方法有什么?

62. 有个搜索框，在搜索框中输入内容，找到当前页面的需要搜索的数据?

63. active-class 是哪个组件的属性?

vue-router模块的router-link组件。

64. 嵌套路由怎么定义?

在 VueRouter 的参数中使用 children 配置，这样就可以很好的实现路由嵌套。

65.* 懒加载（按需加载路由）**

webpack 中提供了 require.ensure()来实现按需加载。以前引入路由是通过 import 这样的方式引入，改为 const 定义的方式进行引入。

不进行页面按需加载引入方式:

```
import  home  from '../../common/home.vue'
```

进行页面按需加载的引入方式:

```
const  home = r => require.ensure( [], () => r (require('../../common/home.vue')))
```

66. vuex 是什么? 怎么使用? 哪种功能场景使用它?

vue框架中状态管理。在main.js引入store，注入。新建了一个目录store，….. export 。场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

67. vue-router 有哪几种导航钩子?

三种，一种是全局导航钩子：router.beforeEach(to,from,next)，作用：跳转前进行判断拦截。第二种：组件内的钩子；第三种：单独路由独享组件

70. 自定义指令(v-check, v-focus) 的方法有哪些? 它有哪些钩子函数? 还有哪些钩子函 数参数

全局定义指令：在vue对象的directive方法里面有两个参数，一个是指令名称，另外一个是函数。组件内定义指令：directives

钩子函数：bind（绑定事件触发）、inserted(节点插入的时候触发)、update（组件内相关更新）

钩子函数参数：el、binding

71. 说出至少 4 种 vue 当中的指令和它的用法

v-once: 只绑定一次　v-bind: 绑定数据　v-model: 绑定模型

v-on: 绑定事件　v-if v-show: 条件渲染

72. vuex 有哪几种属性

Vue有五个核心概念，state, getters, mutations, actions, modules。

73. vuex 的 state 特性是什么

stae就是存放数据的地方，类似一个仓库 ，特性就是当mutation修改了state的数据的时候，他会动态的去修改所有的调用这个变量的所有组件里面的值（ 若是store中的数据发生改变，依赖这个数据的组件也会发生更新 ）

74. vuex 的 getter 特性是什么

getter用来获取数据，mapgetter经常在计算属性中被使用

75. vuex 的 mutation 特性是什么

Action 类似于 mutation，不同在于：Action 提交的是 mutation，而不是直接变更状态。Action 可以包含任意异步操作

76. vue 中 ajax 请求代码应该写在组件的 methods 中还是 vuex 的 action 中

如果请求来的数据是不是要被其他组件公用，仅仅在请求的组件内使用，就不需要放入vuex 的state里。
如果被其他地方复用，这个很大几率上是需要的，如果需要，请将请求放入action里，方便复用，并包装成promise返回，在调用处用async await处理返回的数据。如果不要复用这个请求，那么直接写在vue文件里很方便

77. 不用 vuex 会带来什么问题

可维护性会下降，你要想修改数据，你得维护三个地方；可读性会下降，因为一个组件里的数据，你根本就看不出来是从哪来的；增加耦合，大量的上传派发，会让耦合性大大的增加，本来Vue用Component就是为了减少耦合，现在这么用，和组件化的初衷相背。

78. 使用 Vuex 只需执行 Vue.use(Vuex)，并在 Vue 的配置中传入一个 store 对象的示 例，store 是如何实现注入的?

Vue.use(Vuex) 方法执行的是 install 方法，它实现了 Vue 实例对象的 init 方法封装和注入，使传入的 store 对象被设置到 Vue 上下文环境的store中。因此在VueComponent任意地方都能够通过 this.store 访问到该 store

79. state 内部支持模块配置和模块嵌套，如何实现的?

在 store 构造方法中有 makeLocalContext 方法，所有 module 都会有一个 local context，根据配置时的 path 进行匹配。所以执行如

dispatch('submitOrder', payload)这类 action 时，默认的拿到都是 module 的 local state，如果要访问最外层或者是其他 module 的 state，只能从 rootState 按照 path 路径逐步进行访问。

80. 在执行 dispatch 触发 action(commit 同理)的时候，只需传入(type, payload)，action 执行函数中第一个参数 store 从哪里获取的?

store 初始化时，所有配置的 action 和 mutation 以及 getters 均被封装过。在执行如 dispatch('submitOrder', payload)的时候，actions 中 type 为 submitOrder 的所有处理方法都是被封装后的，其第一个参数为当前的 store 对象，所以能够获取到 { dispatch, commit, state, rootState } 等数据。

81. Vuex 如何区分 state 是外部直接修改，还是通过 mutation 方法修改的?

Vuex 中修改 state 的唯一渠道就是执行 commit('xx', payload) 方法，其底层通过执行 this._withCommit(fn) 设置_committing 标志变量为 true，然后才能修改 state，修改完毕还需要还原_committing 变量。外部修改虽然能够直接修改 state，但是并没有修改_committing 标志位，所以只要 watch 一下 state，state change 时判断是否_committing 值为 true，即可判断修改的合法性。

82. 调试时的"时空穿梭"功能是如何实现的?

devtoolPlugin 中提供了此功能。因为 dev 模式下所有的 state change 都会被记录下来，'时空穿梭' 功能其实就是将当前的 state

替换为记录中某个时刻的 state 状态，利用 store.replaceState(targetState) 方法将执行 this._vm.state = state 实现。

83. axios 是什么？怎么使用？描述使用它实现登录功能的流程

请求后台资源的模块。npm install axios -S装好，然后发送的是跨域，需在配置文件中config/index.js进行设置。后台如果是Tp5则定义一个资源路由。js中使用import进来，然后.get或.post。返回在.then函数中如果成功，失败则是在.catch函数中

84. 你最有成就感的项目是哪个

85. 如果让你负责一个项目，你是如何进行技术选型

86. 移动端项目需要考虑什么问题

87. 说说实现摇一摇功能的思路

88. 多人聊天系统中如何实现实时更新功能

89. 简述项目流程

项目需求分析：这个环节是由项目经理完成，项目经理首先和客户进行交流，了解客户的需求，然后分析项目的可行性，如果项目可以被实现，项目经理写出项目需求文档交给设计师完成后续的开发。

页面设计：这个环节主要是UI设计师参与，UI设计师根据产品需求分析文档，对产品的整体美术风格、交互设计、界面结构、操作流程等做出设计。负责项目中各种交互界面、图标、LOGO、按钮等相关元素的设计与制作。

编码：这个部分由程序员来实现。（程序员分为WEB前端开发工程

师和后台开发工程师。前端开发人员主要做我们可以在网页上看的见的页面，后台就做一些我们看不见的管理系统以及功能的实现。）

程序员根据UI设计师的设计，用编码来完成整个项目的各个功能。

测试：这部分由程序测试员来完成。程序测试员主要就是测试寻找程序还存在的bug，一般来说刚编码完成的程序都是存在问题的，就需要测试人员反复不断的测试并将存在问题的测试结果交给编码人员进行bug的修复。等到几乎所有bug修复完成，这个项目差不多就可以上线了。

维护：程序的维护是整个项目的最后一个阶段，但也是耗时最多，成本最高最高的的一个阶段。程序的维护包括程序上线后后续bug的修复和程序版本的更新。

90. 如何部分引入bootstrap的功能

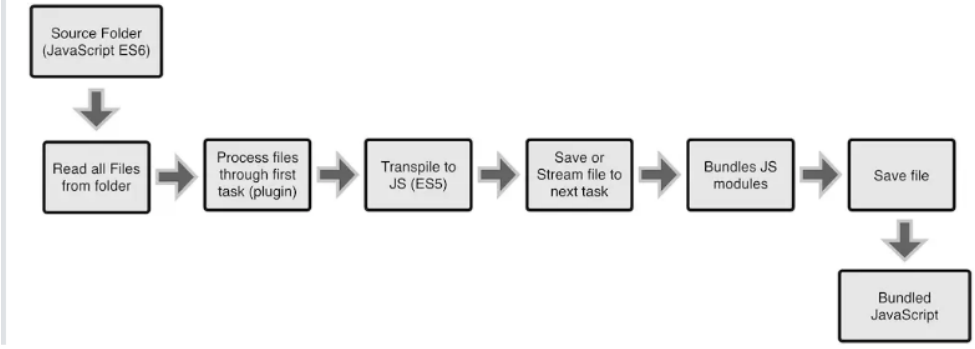
91. 点餐系统中如何实现实时显示点餐进度

92. webpack与gulp的区别

gulp是基于任务的打包工具，webpack是基于配置的打包工具

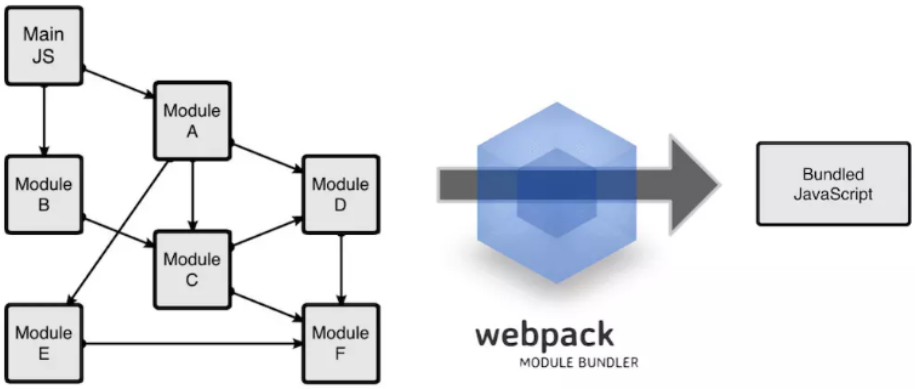
- gulp工作流程

Gulp的工作方式是：在一个配置文件中，指明对某些文件进行类似编译，组合，压缩等任务的具体步骤，工具之后可以自动替你完成这些任务



- webpack工作流程

Webpack的工作方式是：把你的项目当做一个整体，通过一个给定的主文件（如：index.js），Webpack将从这个文件开始找到你的项目的所有依赖文件，使用loaders处理它们，最后打包为一个（或多个）浏览器可识别的JavaScript文件。



93. vue中data与computed的区别

data属性的值，不会随赋值变量的改动而改动。如果要改变这个属性的值，则需要直接给data属性赋值，视图上对这个属性的显示才会变。

computed属性，属于持续变化跟踪。在computed属性定义的时候，这个computed属性就与给它赋值的变量绑定了。改变这个赋值变量，computed属性值会随之改变。

94. 如何设置对象属性为只读

95. 如何扩展原型对象的方法

96. webapp中如何实现下拉刷新

实现思路：

①后端提供分页接口②list页面打开默认显示第一页的列表

③js监听到页面顶部下拉时，清除所有list显示，请求第一页列表显示

④js监听到页面滚动到底部时，增量加载下一的列表进行列表显示

97. webapp中如何实现上拉加载更多

实现思路：

①后端提供分页接口②list页面打开默认显示第一页的列表

③js监听到页面顶部下拉时，清除所有list显示，请求第一页列表显示

④js监听到页面滚动到底部时，增量加载下一的列表进行列表显示

98. 移动端touch事件与click事件的区别

【http://www.softwhy.com/article-10108-1.html】

touch事件用于移动端，click事件专注于电脑端。touch事件的优先级高于click事件。

99. 移动端如何实现页面缓存

100. 移动端如何实现瀑布流效果

101. 现有浏览器不支持ES6，如何解决

102. 如何解决图片在不同分辨率下的显示问题，避免失真

103. 图片上传前如何实现缩略图功能

104. 你是如何解决git冲突问题

105. 乱码最根本的原因是什么

乱码的根本原因是字节和字符的问题。 字节由8个bit位表示，最早的编码是ASCII码，ASCII码是单字节的编码字符。因为单字节8个bit位对于中文字符和其他国家的字符来说根本不够用，需要更多的bit位来表示字符。我们现在常见的编码有GBK，BIG5，GB2312，UTF-8，通过编码映射表可以确定bit位和字符之间的映射关系。

一个应用从服务端把需要展现的文字换成一段字节流传输给浏览器，浏览器把字节流组装字符展现的过程一般是这样（应用的编码

是GBK）：

服务端：String text->getBytes("GBK")->byte[] bytes

浏览端：byte[] bytes->new String("bytes","GBK") ->String text

如果字节流处理的转换过程使用编码不一致就会出现乱码问题。

106. 移动端如何显示轮播图拖拽显示功能

107. nodejs如何罗列硬盘上的文件与文件夹

108. webapp中如何实现语音搜索功能

109. 项目开发中如何确定版本号

110. 简述viewport的作用，并说明如何实现字体在不同分辨率下的适配

111. hybrid App如何显示数据请求问题

112. 在项目中你们是如何实现支付功能

113. 移动端屏幕无法显示完全所有栏目时如何实现tap便签切换功能

114. 如何实现表格固定表头功能（最顶部与最左侧）

115. 什么场景下会用到路由嵌套

116. webpack中实现代码压缩如何避免某些关键字压缩，如define,require等

117. 在对象中说说你对值属性和存储属性的理解

118. 如何理解二次封装，它有什么好处

119. 如何实现在多个sass文件中不编译某个scss文件

120. 项目开发过程中如何解决安全性问题

121. nodejs如何实现一个服务器并解决html、图片等显示问题

==========（面试题）==========

1、jq的链式调用

① jq的链式调用实际上是通过return this的形式来实现的；通过对象上的方法，最后加上return this，把对象再返回来，对象就可以再继续调用方法，实现链式操作了；

② 链式调用的好处：节省代码量，代码看起来更优雅。

③ 链式调用的问题：所有对象的方法返回的都是对象本身，也就是说没有返回值，所以这种方法不一定在任何环境下都适合。

2、react和vue的区别：

- 两者本质的区别：模板和组件化的区别
  - Vue本质是MVVM框架，由MVC发展而来；
  - React是前端组件化框架，由后端组件化发展而来；
- Vue使用模板
- React使用JSX
- React本身就是组件化
- Vue是在MVVM上扩展的
- 共同点：
  - 都支持组件化，都是数据驱动视图

3、在不重新请求页面的情况下通过改变 URL，，更新视图的实现方法：

目前，主要有两种实现方式：hash模式和history模式（vue-router的两个方式）

hash模式背后的原理是onhashchange事件,可以在window对象上监听这个事件；history api可以分为两大部分，切换和修改，参考MDN，切换历史状态包括back、forward、go 三个方法，对应浏览器的前进，后退，跳转操作

修改历史状态包括了pushState,replaceState两个方法,这两个方法接收三个参数:stateObj,title,url

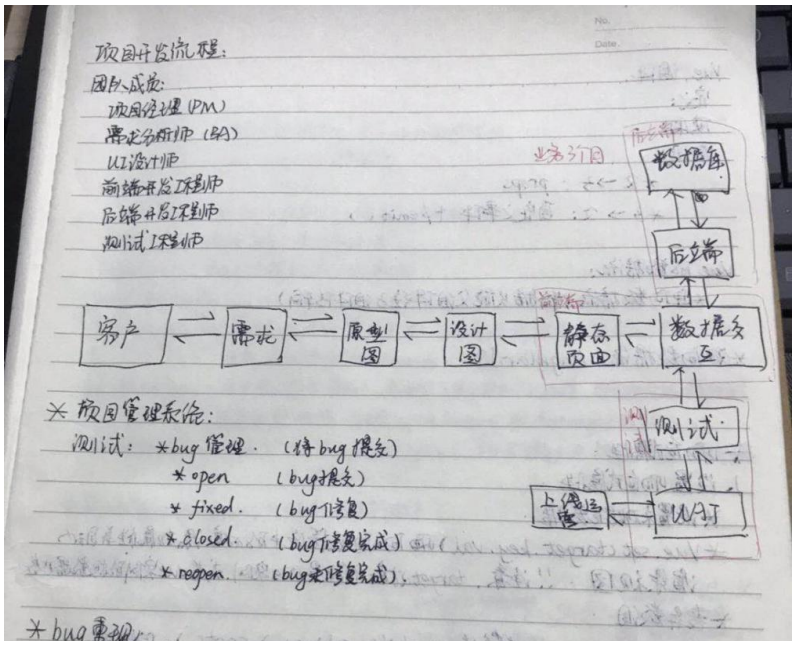
通过pushstate把页面的状态保存在state对象中，当页面的url再变回这个url时，可以通过event.state取到这个state对象，从而可以对页面状态进行还原，这里的页面状态就是页面字体颜色，其实滚

动条的位置，阅读进度，组件的开关的这些页面状态都可以存储到state的里面。

history模式的问题：

通过history api，我们丢掉了丑陋的#，但是它也有个问题：不怕前进，不怕后退，就怕刷新，f5，（如果后端没有准备的话），因为刷新是实实在在地去请求服务器的，不玩虚的。 在hash模式下，前端路由修改的是#中的信息，而浏览器请求时是不带它玩的，所以没有问题.但是在history下，你可以自由的修改path，当刷新时，如果服务器中没有相应的响应或者资源，会分分钟刷出一个404来。

4、项目开发流程



5、vue.js 中 data和computed 区别

（https://blog.csdn.net/formyqianduan/article/details/80791681）

①data属性的值，不会随赋值变量的改动而改动。如果要改变这个属性的值，则需要直接给data属性赋值，视图上对这个属性的显示才会变。

②computed属性，属于持续变化跟踪。在computed属性定义的时候，这个computed属性就与给它赋值的变量绑定了。改变这个赋值变量，computed属性值会随之改变。

6、订单模块

最后我讲一下订单模块，购物车完成后要生成订单，订单中有配送

信息，因此我们必须要求用户登录。我们使用的是单点登录，使用redis模拟Session，实现Session的统一管理,这样用户只需要登录一次就可以访问所有相互信任的应用。用户登录成功后，生成token，相当key， 把用户对象转换成json当value存入redis，模拟Session的过期时间。一般半个小时。最后把token写入cookie中（Cookie需要跨域）。

当点击生成订单时，先配置一个springmvc的拦截器，实现接口HandlerInterceptor,从cookie中取token,没有token，需要跳转到登录页面;有token,调用sso系统的服务，根据token查询用户信息,如果查不到用户信息,说明用户登录已经过期。需要跳转到登录页面，查询到用户信息，放行。

放行后，提交购物车OrderInfo，接收表单的数据，不全一些数据，生成订单id，向订单表插入数据。向订单明细表插入数据，向订单物流表插入数据。返回逻辑视图展示订单生成成功。在调用第三方接口，完成订单支付。

7.阐述JS中原型链及prototype和__proto__的区别

（https://blog.csdn.net/m0_38099607/article/details/72909806）

当实例对象本身不存在某个属性时，js会查找该实例的【构造函数】的【prototype】属性;

只有函数才有prototype属性，普通实例对象没有！！！.prototype里面查不到，能继续沿着这个链条往上查。

①概念:

prototype是函数的一个属性（每个函数都有一个prototype属性），这个属性是一个指针，指向一个对象。它是显示修改对象的原型的属性。

_ proto_ 是一个对象拥有的内置属性（请注意：prototype是函数的内置属性，_ proto_ 是对象的内置属性），_ proto_ 是JS内部使用寻找原型链的属性。简单来说，在 javascript 中每个对象都会有一个 _ proto _ 属性，当我们访问一个对象的属性时，如果这个对象内部不存在这个属性，那么他就会去 _ proto _ 里找这个属性，这个 _ proto _ 又会有自己的 _ proto _，于是就这样一直找下去，也就是我们平时所说的原型链的概念.

[用chrome和FF都可以访问到对象的_ proto _属性，IE不可以。]

8.Vue和React的区别

https://www.cnblogs.com/yunshangwuyou/p/9638112.html

相同点:

①都支持服务器端渲染

②.都有Virtual DOM,组件化开发,通过props参数进行父子组件数据的传递,都实现webComponent规范

③数据驱动视图

④都有支持native的方案,React的React native,Vue的weex

⑤都有管理状态，React有redux,Vue有自己的Vuex（自适应vue，量身定做）

不同点:

①React严格上只针对MVC的view层,Vue则是MVVM模式

②virtual DOM不一样,vue会跟踪每一个组件的依赖关系,不需要重新渲染整个组件树.

而对于React而言,每当应用的状态被改变时,全部组件都会重新渲染,所以react中会需要shouldComponentUpdate这个生命周期函数方法来进行控制

③.组件写法不一样, React推荐的做法是 JSX + inline style, 也就是把HTML和CSS全都写进JavaScript了,即'all in js';

Vue推荐的做法是webpack+vue-loader的单文件组件格式,即html,css,js写在同一个文件;

④数据绑定: vue实现了数据的双向绑定,react数据流动是单向的

⑤.state对象在react应用中不可变的,需要使用setState方法更新状态;

在vue中,state对象不是必须的,数据由data属性在vue对象中管理;

9、移动端开发的项目流程

一款应用的开发大体流程如下:

①项目立项：产品经理②需求确认：产品经理(业务逻辑说明文档)

③业务确认：产品经理，技术经理，架构师④业务架构：技术经理，架构师（业务流程文档）

⑤UI确认：产品经理，设计人员，开发人员全体

⑥UI交互确认：产品经理，移动端，前段开发人员⑦接口确认：架构师，接口开发人员，移动端、前端开发人员⑧UI工时评估：产品经理，设计人员

接口工时评估：架构师，接口开发人员

移动端、前端工时评估：相关开发人员，技术经理⑨工时确认：产品经理，技术经理，设计人员⑩项目开发①测试用例及流程设计：产品经理、测试组②测试用例及流程确认：产品经理、开发人员，测试组③测试及debug：产品经理，测试组，开发④产品定版，release

10、php、mysql查询语句

```
*：代表所有字段
1.添加内容
  (1)插入数据
    insert into 表名 (字段名1,字段名2,..字段名) values (值1,值2,..值n);
2.删除内容
  (1)删除数据
    delete from 表名 where 表达式
3.改
  (1)修改表数据
    update 表名 set 字段=值,字段=值,.. where 条件表达式;
      *应用：购物车
  (2)修改表名
    rename table 表名 to 新表名;
4.查询内容
  (1)查询表中特定数据
    select 字段名1,字段名2 from 表名 where 表达式;
  (2)查询表中某连续几行数据
    select * from 表名 order by id limit 0,10;
5.条件控制
  (1)where语句：条件控制语句 后面写条件表达式
  (2)相关条件控制符：
    等于、大于、小于、不等于   =、>、<、<>
    包含   IN(1,2,3.....)
    a至b之间    BETWEEN a AND b
    与、或、非   AND、OR、NOT
   (3)LIKE用法
      * % 匹配任意、
      * _ 匹配一个字符（可以是汉字,一个中文三个字节）
   (4)数量控制
      LIMIT idx,qty
        *idx 下标[从0开始]  *qty条数
      eg：select * from goods limit 2,5  [从下标2开始的5条]
   (5)空值检测
      IS NULL
   (6)排序
```

11、虚拟dom的理解

VirtualDOM是一个能够直接描述一段HTML DOM结构的JavaScript对象，浏览器可以根据其结构按照一定规则创建出确定唯一的HTML DOM结构。虚拟DOM最本质的就是使用JavaScript对象替代了DOM对象树，从而提升页面渲染性能。

12、对于MVVM的理解：

MVVM(Model-View-ViewModel)，源自于经典的 Model-View-Controller（MVC）模式。MVVM 的出现促进了 GUI 前端开发与后端业务逻辑的分离，极大地提高了前端开发效率。MVVM 的核心是 ViewModel 层，它就像是一个中转站（value converter），负责转换 Model 中的数据对象来让数据变得更容易管理和使用，该层向上

与视图层进行双向数据绑定，向下与 Model 层通过接口请求进行数据交互，起呈上启下作用。View 层展现的不是 Model 层。
如下图所示：

***13、ajax和axios的区别

①Ajax：

Ajax 即“Asynchronous Javascript And XML”（异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术。

Ajax = 异步 JavaScript 和 XML（标准通用标记语言的子集）。

Ajax 是一种用于创建快速动态网页的技术。

Ajax 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

通过在后台与服务器进行少量数据交换，Ajax 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 Ajax）如果需要更新内容，必须重载整个网页页面。

```
$.ajax({
        url: 'http://jsonplaceholder.typicode.com/users',
        type: 'get',
        dataType: 'json',
        data: {
            //'a': 1,
            //'b': 2,
        },
        success: function (response) {
            console.log(response);
        }
    })
```

②axios：

用于浏览器和node.js的基于Promise的HTTP客户端

从浏览器制作XMLHttpRequests
让HTTP从node.js的请求
支持Promise API
拦截请求和响应
转换请求和响应数据
取消请求
自动转换为JSON数据
客户端支持防止XSRF

```
axios({
        url: 'http://jsonplaceholder.typicode.com/users',
        method: 'get',
        responseType: 'json', // 默认的
        data: {
          //'a': 1,
          //'b': 2,
        }
    }).then(function (response) {
        console.log(response);
        console.log(response.data);
    }).catch(function (error) {
        console.log(error);
    })
```

14、angular、react、vue 三大前端框架适用场景

①angular

angular功能比较强大，适用在复杂、重大项目中。它有自己的一套规则，写出来的项目结构比较清晰，便于大型项目的维护迭代。

②react

react是一套跨平台的js框架，对比angular，react较轻量，同时，react有一个明显的特点，也是angular的“痛点”——跨平台。就移动端来说吧，angular当时出来的时候并没有把移动端考虑在内，只是后来才意识到移动端庞大的需求，所以出来了angular2.0（angular2.0说白了就是angular专门针对移动市场的一种解决方案）。react使用js去调用移动原生控件，性能比传统的hybrid app（webview加载html页面）高很多，体验也更接近于native。

③vue

对比其他两大框架，vue是最轻量的，所以上手比较简单。vue专注于 MVVM 模型的ViewModel层，它主要目的是想让大家可以尽可能

简单地达到数据和视图的绑定。如果你的产品需求是轻量级的、开发效率快的，那就选择它吧。

15、xml 与html的区别

①XML 被设计用来传输和存储数据；HTML 被设计用来显示数据。

②什么是 XML?

XML 指可扩展标记语言（EXtensible Markup Language）

XML 是一种标记语言，很类似 HTML

XML 的设计宗旨是传输数据，而非显示数据

XML 标签没有被预定义。您需要自行定义标签。

XML 被设计为具有自我描述性。

XML 是 W3C 的推荐标准

③XML 与 HTML 的主要差异

XML 不是 HTML 的替代。

XML 和 HTML 为不同的目的而设计：

XML 被设计为传输和存储数据，其焦点是数据的内容。

HTML 被设计用来显示数据，其焦点是数据的外观。

HTML 旨在显示信息，而 XML 旨在传输信息。

④XML 不是对 HTML 的替代

XML 是对 HTML 的补充。

XML 不会替代 HTML，理解这一点很重要。在大多数 web 应用程序中，XML 用于传输数据，而 HTML 用于格式化并显示数据。

对 XML 最好的描述是：

XML 是独立于软件和硬件的信息传输工具。

⑤XML 是 W3C 的推荐标准

可扩展标记语言 (XML) 于 1998 年 2 月 10 日成为 W3C 的推荐标准。

16、请列举至少3个Vue中的三个生命周期钩子函数及说明他们的作用?

生命周期钩子	组件状态	最佳实践
beforeCreate	实例初始化之后，this指向创建的实例，不能访问到data、computed、watch、methods上的方法和数据	常用于初始化非响应式变量
created	实例创建完成，可访问data、computed、watch、methods上的方法和数据，未挂载到DOM，不能访问到$el属性，$ref属性内容为空数组	常用于简单的ajax请求，页面的初始化
beforeMount	在挂载开始之前被调用，beforeMount之前，会找到对应的template，并编译成render函数	-
mounted	实例挂载到DOM上，此时可以通过DOM API获取到DOM节点，$ref属性可以访问	常用于获取VNode信息和操作，ajax请求
beforeupdate	响应式数据更新时调用，发生在虚拟DOM打补丁之前	适合在更新之前访问现有的DOM，比如手动移除已添加的事件监听器
updated	虚拟 DOM 重新渲染和打补丁之后调用，组件DOM已经更新，可执行依赖于DOM的操作	避免在这个钩子函数中操作数据，可能陷入死循环
beforeDestroy	实例销毁之前调用。这一步，实例仍然完全可用，this仍能获取到实例	常用于销毁定时器、解绑全局事件、销毁插件对象等操作
destroyed	实例销毁后调用，调用后，Vue 实例指示的所有东西都会解绑定，所有的事件监听器会被移除，所有的子实例也会被销毁	-

***18、编写一个数组去重的方法（不用set方法）

```
<script type="text/javascript">
var arr=[1,1,2,2,4,5,2,1,2];
var new_arr=[];
for(var i=0;i<arr.length;i++){
if(new_arr.indexOf(arr[i])==-1){
new_arr.push(arr[i]);
}
}
alert(new_arr)
</script>
```

3.利用对象的属性去重（推荐）

思路：每次取出原数组的元素，然后再对象中访问这个属性，如果存在就说明重复

```
function unique(arr){
    var res =[];
    var json = {};
    for(var i=0;i<arr.length;i++){
        if(!json[arr[i]]){
            res.push(arr[i]);
            json[arr[i]] = 1;
        }
    }
    return res;
}
```

或：

19、元素Position的常用属性值有哪些？说说他们的区别和用法。

Position属性四个值：static、fixed、absolute和relative的区别和用法

先看下各个属性值的定义：

①static（静态定位）：默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。

②relative（相对定位）：生成相对定位的元素，通过top,bottom,left,right的设置相对于其正常（原先本身）位置进行定位。可通过z-index进行层次分级。

③absolute（绝对定位）：生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。可通过z-index进行层次分级。

④fixed（固定定位）：生成绝对定位的元素，相对于浏览器窗口进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。可通过z-index进行层次分级。

20、css选择符有哪些？哪些属性可以继承？优先级算法如何计算？内联和important哪个优先级高？

1、CSS选择符有哪些?

??类型选择符（body）、群组选择符（h1，h2，h3，span）、包含选择符（h2 span）、ID选择符（#id）、Class选择符（.content）

2、哪些属性可以继承?

??CLASS属性，伪类A标签，列表ul、li、dl、dd、dt可以继承

3、优先级算法如何计算?

???优先级就近原则，

4、important的优先级比内联优先级更高;

21、apply和call的区别和作用?

apply：最多只能有两个参数——新this对象和一个数组argArray。如果给该方法传递多个参数，则把参数都写进这个数组里面，当然，即使只有一个参数，也要写进数组里。如果argArray不是一个有效的数组或arguments对象，那么将导致一个TypeError。如果没有提供argArray和thisObj任何一个参数，那么Global对象将被用作thisObj，并且无法被传递任何参数。

call：它可以接受多个参数，第一个参数与apply一样，后面则是一串参数列表。这个方法主要用在js对象各方法相互调用的时候，使当前this实例指针保持一致，或者在特殊情况下需要改变this指针。如果没有提供thisObj参数，那么 Global 对象被用作thisObj。

实际上，apply和call的功能是一样的，只是传入的参数列表形式不同。

https:https://www.jianshu.com/p/1a348132ba2a//www.jianshu.com/p/bc541afad6ee

22、行内元素和块级元素的区别?各有哪些?

(1)块级元素：独占一行，可以设置宽高，块级元素可以作为所有内联元素的容器和部分块级元素的容器，常用的块级元素有：div,dl,dt,dd,ol,ul,fieldset,h1-h6,p,from,table,tr,td（2）行内元素：不独占一整行，不可以设置宽高，宽高由内容撑开，行内元素也遵循盒子模型，但是margin-top、margin-bottom、padding-top、padding-bottom是没有效果的。常用的行内元素：a-超链接、span、img、i、b、em、br、input、sub、u

23、C/S架构的优缺点:

*优点:

①客户端因为是独立设计，所以可以实现个性化

②因为客户端是需要进行安装的，可以不需要重复安装和加载

③因为客户端是独立开发的，所以有能力对客户端进行安全设计
④如果遇到不同的操作系统，需要为不同的操作系统各开发一套客户端
*缺点：
①因为客户端是不需要重复安装，所以用户可以不更新与升级，增加了维护成本。
②因为需要开发客户端和服务器两套程序，所以开发成本会增加
B/S架构的优缺点：
*优点：
①因为B/S架构具备通用性，所以开发成本较低。
②因为不需要安装客户端，所以客户端不需要进行升级，只需要更新后台代码即可实现所有客户端的更新。
④因为B/S架构多用WEB网页进行开发，所以增、删功能也非常容易，只需要修改网页即可完成
*缺点：
①耗流量，每次都要加载全部的内容（不过有缓存可以降低流量损耗）
②因为没有独立的客户端，所以无法实现个性化（通过账号体系可以实现）
③因为没有独立设计客户端，所以客户端难以实现安全控制（HTTPS、控件）。
④难以实现特殊的操作（删本地文件），所以所有的杀毒软件都是C/S架构的。
B/S架构更多的时候是使用了HTTP协议、而C/S架构更多的时候使用的WinSocket协议（TCP、UDP）
*****24、什么是闭包（closure），为什么要用它？在开发项目时什么地方用闭包？以及优点和缺点？**
①闭包概念：即通过函数嵌套函数，内部函数引用局部变量实现变量不释放。广泛来讲，所有函数访问另一个函数内部变量的过程都可以称之为闭包
②优点：局部变量不释放
缺点：内存泄漏、内存占用
③闭包的实现基于以下三点：
函数可以创建独立作用域（因此闭包的实现依赖于函数）；

自由变量跨作用域取值，要去创建该函数的作用域中取值；
变量被函数引用时不会被释放；

④ 为什么要用它？

就是为了防止变量污染，但是用多了会内存泄漏

⑤在开发项目时什么地方用闭包？

闭包就是父函数给子函数传值，解决作用域问题。

在es6出现之后一般使用let

比如做一个随机抽题功能、选项卡、幻灯片等。一般用setInterval的地方用的比较多。

25.数组的方法：

map、forEach、filter、every、some、reduce、push、pop、shift、unshift、isArray、concat、toString、join、splice

1) Array.map()

此方法是将数组中的每个元素调用一个提供的函数，结果作为一个新的数组返回，并没有改变原来的数组

```
let arr = [1, 2, 3, 4, 5]
    let newArr = arr.map(x => x*2)
    //arr= [1, 2, 3, 4, 5]   原数组保持不变
    //newArr = [2, 4, 6, 8, 10] 返回新数组
```

2) Array.forEach()

此方法是将数组中的每个元素执行传进提供的函数，没有返回值，直接改变原数组，注意和map方法区分

```
let arr = [1, 2, 3, 4, 5]
    num.forEach(x => x*2)
    // arr = [2, 4, 6, 8, 10]  数组改变,注意和map区分
```

3) Array.filter()

此方法是将所有元素进行判断，将满足条件的元素作为一个新的数组返回

```
let arr = [1, 2, 3, 4, 5]
     const isBigEnough => value => value >= 3
     let newArr = arr.filter(isBigEnough )
     //newNum = [3, 4, 5] 满足条件的元素返回为一个新的数组
```

4) Array.every()

此方法是将所有元素进行判断返回一个布尔值，如果所有元素都满足判断条件，则返回true，否则为false:

```
let arr = [1, 2, 3, 4, 5]
    const isLessThan4 => value => value < 4
    const isLessThan6 => value => value < 6
    arr.every(isLessThan4 ) //false
    arr.every(isLessThan6 ) //true
```

5) Array.some()

此方法是将所有元素进行判断返回一个布尔值，如果存在元素都满足判断条件，则返回true，若所有元素都不满足判断条件，则返回false：

```
let arr= [1, 2, 3, 4, 5]
    const isLessThan4 => value => value < 4
    const isLessThan6 => value => value > 6
    arr.some(isLessThan4 ) //true
    arr.some(isLessThan6 ) //false
```

6）Array.reduce()

此方法是所有元素调用返回函数，返回值为最后结果，传入的值必须是函数类型：

```
let arr = [1, 2, 3, 4, 5]
   const add = (a, b) => a + b
   let sum = arr.reduce(add)
   //sum = 15  相当于累加的效果
   与之相对应的还有一个 Array.reduceRight() 方法，区别是这个是从右向左操作的
```

7）Array.push()

此方法是在数组的后面添加新加元素，此方法改变了数组的长度：

8）Array.pop()

此方法在数组后面删除最后一个元素，并返回数组，此方法改变了数组的长度：

```
let arr = [1, 2, 3, 4, 5]
    arr.pop()
    console.log(arr) //[1, 2, 3, 4]
    console.log(arr.length) //4
```

9）Array.shift()

此方法在数组后面删除第一个元素，并返回数组，此方法改变了数组的长度：

```
let arr = [1, 2, 3, 4, 5]
    arr.shift()
    console.log(arr) //[2, 3, 4, 5]
    console.log(arr.length) //4
```

10)Array.unshift()

此方法是将一个或多个元素添加到数组的开头，并返回新数组的长度：

```
let arr = [1, 2, 3, 4, 5]
    arr.unshift(6, 7)
    console.log(arr) //[6, 7, 2, 3, 4, 5]
    console.log(arr.length) //7
```

11)Array.isArray()

判断一个对象是不是数组，返回的是布尔值

12)Array.concat()

此方法是一个可以将多个数组拼接成一个数组：

```
let arr1 = [1, 2, 3]
      arr2 = [4, 5]
  let arr = arr1.concat(arr2)
  console.log(arr)//[1, 2, 3, 4, 5]
```

13)Array.toString()

此方法将数组转化为字符串：

```
let arr = [1, 2, 3, 4, 5];
   let str = arr.toString()
   console.log(str)// 1,2,3,4,5
```

14)Array.join()

此方法也是将数组转化为字符串：

```
let arr = [1, 2, 3, 4, 5];
   let str1 = arr.toString()
   let str2 = arr.toString(',')
   let str3 = arr.toString('##')
   console.log(str1)// 12345
   console.log(str2)// 1,2,3,4,5
   console.log(str3)// 1##2##3##4##5
```

通过例子可以看出和toString的区别，可以设置元素之间的间隔~

15)Array.splice(开始位置， 删除的个数，元素)

万能方法，可以实现增删改：

```
let arr = [1, 2, 3, 4, 5];
     let arr1 = arr.splice(2, 0 'haha')
     let arr2 = arr.splice(2, 3)
     let arr1 = arr.splice(2, 1 'haha')
     console.log(arr1) //[1, 2, 'haha', 3, 4, 5]新增一个元素
     console.log(arr2) //[1, 2] 删除三个元素
     console.log(arr3) //[1, 2, 'haha', 4, 5] 替换一个元素
```

26.字符串的方法：

先定义一个字符串 var str = "abbcalgldkjgldgehlgjldf";

1) **indexOf()**，返回指定字符创第一次出现的位置。

str.indexOf('a') //结果是0

2) **lastIndexOf()**，返回指定字符串最后一次出现的位置。

str.lastIndexOf('f') //结果是22

3) substring()，提取字符串中两个指定索引号之间的字符（两个索引不能为负值）

str.substring(0,5) //结果是abbca

4) slice()，提取字符串中两个指定索引号之间的字符（索引可以为负值，-1就是倒数第二位）

str.slice(0,-5) //结果是abbcalgldkjgldgehl

5) charAt()，返回指定索引的字符

str.charAt(1) //结果是b

6) concat()，连接两个或多个字符串，返回连接后的新字符串。

var str2 = "1";

str.concat(str2) //结果是abbcalgldkjgldgehlgjldf1

7) split()，把字符串分割为子字符串数组

str.split() //结果是["abbcalgldkjgldgehlgjldf"]

8) toString()，返回字符串对象，比如把数字转换成字符串对象。

9) trim()，移除字符串首位空格，经常在对input和textarea的值做判断时用到。

10)toLowerCase()，把字符串转换成小写的。

11)toUpperCase()，把字符串转换成大写的。

12)match()，找到一个或者多个子串或者正则表达式的匹配。

13)replace()，替换指定子串或者与正则表达式匹配的子串。

14)search()，检索指定子串或者与正则表达式匹配的值，返回的值是数字。

27.JS中的六大数据类型

①Number类型

②String类型

③Boolean类型

④Undefined类型

⑤Null类型

⑥Object类型

三大引用类型

Object类型、Array类型、Function类型

28.谈谈以前端角度出发做好SEO需要考虑什么?

【https://blog.csdn.net/jnshu_it/article/details/80029988】

①什么是SEO?

SEO由英文Search Engine Optimization缩写而来，中文意译为“搜索

引擎优化”。
其实叫做针对搜索引擎优化更容易理解。它是指从自然搜索结果获得网站流量的技术和过程，是在了解搜索引擎自然排名机制的基础上，对网站进行内部及外部的调整优化， 改进网站在搜索引擎中的关键词自然排名，获得更多流量，从而达成网站销售及品牌建设的目标。
②解决方法：
1) 维护网站，提高内容质量，保持更新
搜索引擎会考评网站的质量，保持网站的经常更新，使网站拥有大量的、有用的、可读性强的优质信息。网站的权重会相应的提升。为了保持更新频率而去抄袭并不可取。
2) 网站结构布局优化：尽量简单、开门见山，提倡扁平化结构。
一般而言，建立的网站结构层次越少，越容易被“蜘蛛”抓取，也就容易被收录。一般中小型网站目录结构超过三级，“蜘蛛”便不愿意往下爬，并且根据相关调查：访客如果经过跳转3次还没找到需要的信息，很可能离开。因此，三层目录结构也是体验的需要。
3) 控制首页链接数量
网站首页是权重最高的地方，如果首页链接太少，没有“桥”，“蜘蛛”不能继续往下爬到内页，直接影响网站收录数量。但是首页链接也不能太多，一旦太多，没有实质性的链接，很容易影响用户体验，也会降低网站首页的权重，收录效果也不好。注意链接要建立在用户的良好体验和引导用户获取信息的基础之上。
4) 导航优化
导航应该尽量采用文字方式，也可以搭配图片导航，但是图片代码一定要进行优化，img标签必须添加“alt”和“title”属性，告诉搜索引擎导航的定位，做到即使图片未能正常显示时，用户也能看到提示文字。其次，在每一个网页上应该加上面包屑导航，好处：从用户体验方面来说，可以让用户了解当前所处的位置以及当前页面在整个网站中的位置，帮助用户很快了解网站组织形式，从而形成更好的位置感，同时提供了返回各个页面的接口，方便用户操作；对“蜘蛛”而言，能够清楚的了解网站结构，同时还增加了大量的内部链接，方便抓取，降低跳出率。
5) 控制页面的大小
减少http请求，提高网站的加载速度。当速度很慢时，用户体验不

好，留不住访客，并且一旦超时，“蜘蛛”也会离开。

6) 适量的关键词和网页描述

关键词不宜太多也不宜太少，列举出几个页面的重要关键字即可，切记过分堆砌。网页描述要准确，精简地描述网页的内容。

(1) title标题：只强调重点即可，尽量把重要的关键词放在前面，关键词不要重复出现，尽量做到每个页面的title标题中不要设置相同的内容。

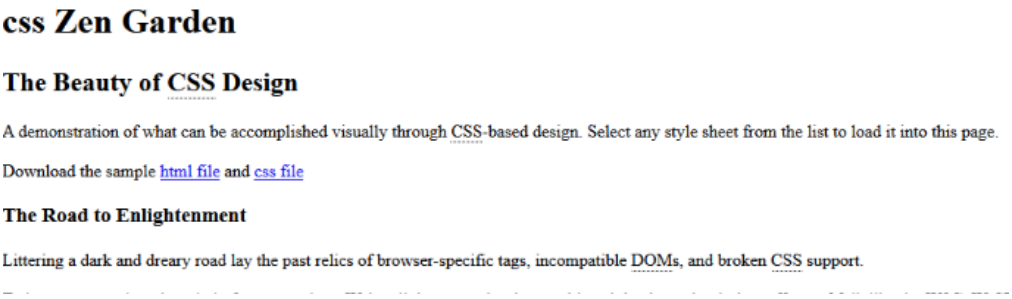
(2) meta keywords标签：关键词，列举出几个页面的重要关键字即可，切记过分堆砌。

(3) meta description标签：网页描述，需要高度概括网页内容，切记不能太长，过分堆砌关键词，每个页面也要有所不同。

(4) body中的标签：尽量让代码语义化，在适当的位置使用适当的标签，用正确的标签做正确的事。让阅读源码者和“蜘蛛”都一目了然。比如：h1-h6 是用于标题类的，nav标签是用来设置页面主导航的等。

对于搜索引擎来说，最直接面对的就是网页HTML代码，如果代码写的语义化，搜索引擎就会很容易的读懂该网页要表达的意思。例如文本模块要有大标题，合理利用h1-h6，列表形式的代码使用ul或ol，重要的文字使用strong等等。总之就是要充分利用各种HTML标签完成他们本职的工作，当然要兼容IE、火狐、Chrome等主流浏览器。我们来看看著名的禅意花园网站（http://www.csszengarden.com/），在没有样式的情况下，代码非常语义化，看起来很工整，加载不同的样式之后可以随心所欲的改变页面外观。

无样式表：



有样式表:

(5) a标签：页内链接，要加 “title”属性加以说明，让访客和 “蜘蛛” 知道。而外部链接，链接到其他网站的，则需要加上el="nofollow" 属性，告诉 “蜘蛛”不要爬，因为一旦“蜘蛛”爬了外部链接之后，就不会再回来了。

(6) img应使用 "alt" 属性加以说明。为图片加上alt属性,当网络速度很慢，或者图片地址失效的时候，就可以体现出alt属性的作用，他可以让用户在图片没有显示的时候知道这个图片的作用。

(7) strong、em标签 ：需要强调时使用。strong标签在搜索引擎中能够得到高度的重视，它能突出关键词，表现重要的内容，em标签强调效果仅次于strong标签。

(8) 巧妙利用CSS布局，将重要内容的HTML代码放在最前面，最前面的内容被认为是最重要的，优先让“蜘蛛”读取，进行内容关键词抓取。搜索引擎抓取HTML内容是从上到下，利用这一特点，可以让主要代码优先读取，广告等不重要代码放在下边。例如，在左栏和右栏的代码不变的情况下，只需改一下样式，利用float:left;和float:right;就可以随意让两栏在展现上位置互换，这样就可以保证重要代码在最前，让爬虫最先抓取。同样也适用于多栏的情况。

(9) 谨慎使用 display：none ： 对于不想显示的文字内容，应当设置z-index或设置到浏览器显示器之外。因为搜索引擎会过滤掉

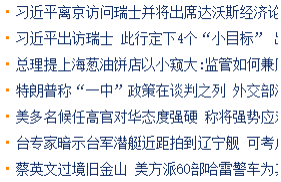
display:none其中的内容。

(10)保留文字效果

如果需要兼顾用户体验和SEO效果，在必须用图片的地方，例如个性字体的标题，我们可以利用样式控制，让文本文字不会出现在浏览器上，但在网页代码中是有该标题的。例如这里的“电视剧分类”，为了完美还原设计图，前端工程师可以把文字做成背景图，之后用样式让html中的文字的缩进设置成足够大的负数，偏离出浏览器之外，也可以利用设置行高的方法让文字隐藏。

(11)利用CSS截取字符

如果文字长度过长，可以用样式截取，设置高度，超出的部分隐藏即可。这样做的好处是让文字完整呈现给搜索引擎，同时在表现上也保证了美观。



29、订单模块

最后我讲一下订单模块，购物车完成后要生成订单，订单中有配送信息，因此我们必须要求用户登录。我们使用的是单点登录，使用redis模拟Session，实现Session的统一管理,这样用户只需要登录一次就可以访问所有相互信任的应用。用户登录成功后，生成token，相当key， 把用户对象转换成json当value存入redis，模拟Session的过期时间。一般半个小时。最后把token写入cookie中（Cookie需要跨域）。

当点击生成订单时，先配置一个springmvc的拦截器，实现接口HandlerInterceptor,从cookie中取token,没有token，需要跳转到登录页面;有token,调用sso系统的服务，根据token查询用户信息,如果查不到用户信息,说明用户登录已经过期。需要跳转到登录页面，查询到用户信息，放行。

放行后，提交购物车OrderInfo，接收表单的数据，不全一些数据，

生成订单id，向订单表插入数据。向订单明细表插入数据，向订单物流表插入数据。返回逻辑视图展示订单生成成功。在调用第三方接口，完成订单支付。

30、vue-cli脚手架修改端口号

https://blog.csdn.net/github_38708644/article/details/86608493

找到以下目录：项目路径下的\node_modules\@vue\cli-service\lib\commands

```
const defaults = {
  host: '0.0.0.0',
  port: 8080,
  https: false
}
```

31、Vue中带有__ob__: Observer的数组，无法取到里边的值

https://www.imooc.com/wenda/detail/414534

34、科腾项目公司面试总问题结：

科腾项目公司面试总问题结：
1. 说一下vue的生命周期
2. props传参最早可以在哪个生命周期获取
3. 说一下vue的组件通讯（父传子、子传父、兄弟组件）
4. 说一下mixin与mixins区别
5. 说一下vue插槽的slot的使用
6. v-if与v-show区别
7. vue的双向绑定原理
8. Vue-Router有什么模式、区别
9. computed与watch区别
10. 移动端布局方法
11. 如何减少webpack打包体积
12. 如何减少webpack打包时间
13. vue如何确保数据显示到html上
14. 原生js和vue分别怎么阻止事件冒泡
15. 移动端布局方法
16. 上家公司移动端ui设计图尺寸是多少，rem布局如何自适应不同屏幕大小的
17. 有了解过回调地狱吗，怎么解决
18. 安卓与ios接口调用区别

35、佳都新太面试

佳都新太面试情况：1.gulp跟wepback理解，gulp一般用来做什么，webpack的配置2.遇到过什么兼容性问题3.做过什么插件，现场讲怎么做4，插件一般怎么暴露引入5，如何用栅栏布局实现左右居中上下布满，大概就记得这些了，她都是想到什么问什么，以自己简历为主

36、科腾面试

科腾面试：v-if和v-show的区别，webpack的理解，webpack如何优化打包时间，transition和animate的属性，生命周期和钩子函数，react和vue的区别，vuex的理解，vuex获取数据this.$store.state.xxx怎么简写，vue中watch和computed的区别，💀简历怎么写，他就怎么问，问的问题很基础，但是要求理解透彻，简历吹得太过的同学，建议修改一下简历

37.vue之vue-cookies

安装:

```
npm install vue-cookies --save
```

使用:

```
import Vue from 'Vue'
import VueCookies from 'vue-cookies'
Vue.use(VueCookies)
```

Api:

设置 cookie:

```
this.$cookies.set(keyName, time)   //return this
```

获取cookie

```
this.$cookies.get(keyName)       // return value
```

删除 cookie

```
this.$cookies.remove(keyName)   // return  false or true , warning:  next version return this;  use isKey(keyname) return true/false,please
```

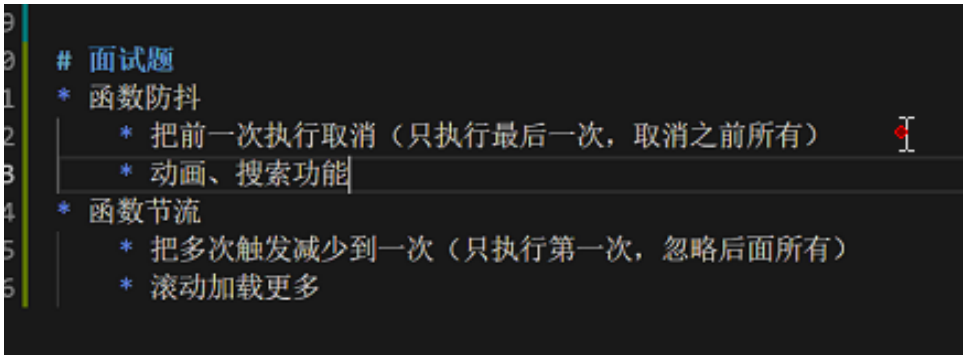
查看一个cookie是否存在（通过keyName）

```
this.$cookies.isKey(keyName)        // return false or true
```

获取所有cookie名称

```
this.$cookies.keys()  // return a array
```

41、函数防抖、函数节流



44、环球大爱面试题

环球大爱面试题：
1、h5有哪些新标签
2、css3的新特性
3、javascript上下文的理解
4、pormise的理解
5、pormise的语法糖的理解
6、vue的理解
7、vue的生命周期
8、vue的指令
9、vue的自定义指令
10、react的理解
11、有用过vuex吗
12、nodejs+express怎么实现页面渲染的
13、对闭包的了解
14、你自己有写过接口？怎么写的？
15、express的理解
（还有一些忘了、基本会看着你的简历去问，没有笔试、公司六月份会搬到佛山那边、去佛山的可以考虑一下）

45、美的总部hr后台项目 面试题总结：

(1) vuex是什么东西，一般用来做什么
(2) vue-router有哪些钩子函数，用来做什么功能
(3) vue组件通讯有哪些
(4) 有写过什么组件吗（业务通用的组件）
(5) 父组件如何知道子组件什么时候created完
(6) created与mounted区别
(7) watch与computed区别
(8) vue的组件优化有没有了解过（大量使用有特效的子组件会有卡顿，你怎么去优化你的组件）
(9) 有看过vue的文档吗，vue里面有哪些东西可以做到代码复用（从这题开始状态就开始崩了）

(10)有用过mixin吗
(11)ie浏览器的兼容
(12)（场景题）tab切换的时候怎么缓存数据不在重新请求数据
(13)（场景题）如果有多个tab来回切换，我希望我的数据及滚动条都在原来的位置，我需要怎么做
(14)keep-alive的生命钩子与普通组件的生命钩子有什么不同
(15)keep-alive的created和mounted在什么时候执行
(16)能说一下es5与es6里的关于数组的方法吗
(17)能说一下map方法时做什么的吗（因为上一题有说到map）
(18)能说一下map与forEach区别吗
(19)能说一下forEach与for循环的区别吗
(20)说一下数组去重的方法
(21)用new set()去重的话，怎么把set去重后的数组变成一个真正的课遍历的数组（用Array.from()）
(22)es6还用过其他的语法吗
(23)const与let一般在什么情况下会用到这2个关键字呢
(24)讲一下变量提升是怎么理解的
(25)函数提升有了解过吗，例如console.log(a);var a=1;function a(){}，console.log(a)会打印出来什么
(26)你在项目中怎么解决异步问题
(27)如果我存在一个多请求并发的一个状态，我想等所有请求完了之后再进行下一步操作，有什么办法吗（我回答了promise.all）
(28)promise.race与promise.all的区别
(29)有对改变过函数里的this指向吗(我回答了bind、call、apply)
(30)bind、call、apply三者的异同
(31)（场景题）那你在项目当中什么情况下会用到call、apply呢
(32)箭头函数与普通函数的this指向有什么区别
(33)new 操作一般做了什么（我答了自定义构造函数的过程）
(34)promise与setTimeout有什么区别
(35)平常的话会怎么学习
(36)看你简历上有写到express和koa，那你用这2个做了一些什么东西呢
(37)有没有搞过服务器部署
(38)一般你们怎么解决跨域的问题

(39)那你们后端是怎么代理的

(40)总结：问题从第9题的时候心态就开始崩了，后面有一些本来能回答的都没回答好，面试官的话会从你的简历上写的项目、技能来提问你，也会根据你的回答来扩展问题来让你回答，也会根据你的回答来提一些场景题，例如这个xxx在项目中要怎么使用等等，所以回答的时候要往自己会的方向引导，还有一些知识点的话也要准备一些场景的用法等等，以及面试过程中一定一定要保持好自己的心态，以及调整自己的心态，不然就像我一样心态崩了之后就会一崩到底了

46、卓骏科技

(1) 问了基本数据类型

(2) 引用类型中相互赋值会不会 其中一个改变另外一个会一个想吗

(3) 深浅复制

(4) es6知道哪些

(5) 对原生js的理解

(6) diff算法

(7) react是全局更新还是局部更新

(8) 继承

(9) protype原型链

(10)react遇到项目难点

(11)promise解决什么

(12)react用过什么插件

(13)react的redux说下理解

47、丽晶软件

(1) 做过什么优化 seo优化 前端页面优化

(2) 平时有看什么书吗 有看什么书的习惯吗

(3) 除了学习vue框架 react 还会去学什么东西

(4) 问了开发团队架构

(5) 假如请求回来1000条数据 全部请求回来 怎么做优化 使页面不会太卡顿

(6) 职业规划

(7) 会让你介绍公司的业务什么的 做什么东西

(8) 锁屏时间优化（一脸懵逼）

***48、什么叫原型以及原型链

4.什么叫原型以及原型链

原型

是一个可以被克隆的类，通过复制原型可以创建一个一模一样的新对象。

其实原型就相当于一个模板，它包含如下

1.它所有的引用类型都包括一个_ proto _(隐式原型)的属性，属性是一个普通的对象

2.所有构造函数都是有一个prototype（原型）属性，属性是一个普通的对象

3所有引用类型的_ proto _属性指向它的构造函数的prototype，即a为一个数组，a._ _ proto_ === Array.prototype

原型链

当访问一个对象的某个属性时，会先从该函数本身属性上找，如果没有找到，则会在他的 _ proto _隐式属性（即构造函数的prototype）上去找，如果还没有找到的话，则就会再去prototyep的 _ proto _ 中去找，这样一层一层向上查找就会形成一个链式结构，就称之为原型链。

构造器的实例"-proto-"属性指像的是构造器原型，所以构造原型上的属性和方法都能被实例访问到，加入A构造器的原型是B构造器的实例，B构造的原型是C构造器的实例，这样的话实例之间就形成一条由_"-proto-"_属性连接的原型链。

在原型链低端的实例可以使用原型链高端的属性和方法。

***49、promise可以解决回调地狱的底层原理是什么

7.promise可以解决回调地狱的底层原理是什么

回调地狱是因为异步回调函数嵌套太多。

Promise的内部也有一个 `defers` 队列存放事件，而 `.then` 方法的作用和发布订阅模式的on方法一样是用来订阅事件的，每次调用 `.then` 方法就会往defers队列中放入一个事件，当异步操作完成时， `resolve方法标示前一个异步过程完成并从defers队列中取出第一个事件执行并返回当前对象保证链式调用，以此类推，就完成了所有异步过程的队列执行。`

51、如何网站性能优化

Vue的性能优化

对路由组件进行懒加载const Login = () => import('@/pages/Login')

对第三方文件库引用CDN

对代码进行压缩打包处理

前端优化，页面优化JS优化 PC端 移动端使用

减少DOM节点操作，使用虚拟DOM vue react等

尽量减少ajiax 请求，多次请求同样数据可以多数据进行缓存处理

标签中尽量避免出现空的 href src 连接

减少cookie等 本地缓存

减少for循环的使用次数for循环很影响性能

图片的处理，使用精灵图，图片懒加载，预加载等
减少重定向
图片压缩处理
直接引用第三方资源库CDN
jq优化
jq版本问题，理论来说，版本越高，性能越好
选择器性能 id > tag > class > [attr] > :伪类
缓存对象
链式调用
避免频繁的操作DOM节点
事件代理
尽量用原生的方法
封装插件

52、vuex 的原理

55、vuex的state与props的区别

***56、前端性能优化

- step1，使用CDN加速，
- step2，静态资源的压缩合并，
- step3，减少HTTP请求，
- step4，尽可能的减少DOM操作，
- step5，由于浏览器渲染机制的影响，JS,CSS文件放在页面最底部
- step6，使用懒加载，
- step7，使用浏览器缓存,本地存储，减少请求数量
- step8，使用SSR(server side rendering)后端渲染，数据直接输出到HTML
- step9，项目中采用按需加载

渲染类的一些方法：

- step1，压缩图片，
- step2，尽量使用CSS动画，开启GPU加速
- step3，Iconfont代替图片
- step4，精简代码，压缩代码

***57、vue和react的区别

58、函数节流和函数防抖

59.vue项目打包后一片空白解决办法

网上很多说自己的VUE项目通过Webpack打包生成的dist文件，放到HBulider打包后，通过手机打开一片空白。这个主要原因是路径的问题。

①记得改一下config下面的index.js中bulid模块导出的路径。因为index.html里边的内容都是通过script标签引入的，而你的路径不对，打开肯定是空白的。先看一下默认的路径。

```
module.exports = {
  build: {
    env: require('./prod.env'),
    index: path.resolve(__dirname, '../dist/index.html'),
    assetsRoot: path.resolve(__dirname, '../dist'),
    assetsSubDirectory: 'static',
    assetsPublicPath: '/',
    productionSourceMap: true,
```

assetsPublicPath默认的是‘/’也就是根目录。而我们的index.html和static在同一级目录下面。 所以要改为‘./’

②另外还需要注意一点。src里边router/index.js路由配置里边默认模式是hash，如果你改成了history模式的话，打开也会是一片空白。所以改为hash或者直接把模式配置删除，让它默认的就行 。如果非要使用history模式的话，需要你在服务端加一个覆盖所有的情况的候选资源：如果URL匹配不到任何静态资源，则应该返回一个index.html,这个页面就是你app依赖页面。

*****60、vuejs项目性能优化总结**

【https://www.jianshu.com/p/41075f1f5297】

在使用elementUI构建公司管理系统时，发现首屏加载时间长，加载的网络资源比较多，对系统的体验性会差一点，而且用webpack打包的vuejs的vendor包会比较大。所以通过搜集网上所有对于vuejs项目的性能优化，做了有关3方面的优化建议，主要包括：上线代码包打包、源码编写优化、用户体验优化。(下面的优化建议只在vue-cli脚手架下做过测试，详情请参考)

①代码包优化

（1）屏蔽sourceMap

待下项目开发完成。进行打包源码上线环节，需要对项目开发环节的开发提示信息以及错误信息进行屏蔽，一方面可以减少上线代码包的大小；另一方面提高系统的安全性。在vuejs项目的config目录下有三个文件dev.env.js(开发环境配置文件)、prod.env.js(上线配置文件)、index.js(通用配置文件)。vue-cli脚手架在上线配置文件会自动设置允许sourceMap打包，所以在上线前可以屏蔽sourceMap。如下所示，index.js的配置如下,通用配置文件分别对开发环境和上线环境做了打包配置分类，在build对象中的配置信息中，productionSourceMap修改成false:

（2）对项目代码中的JS/CSS/SVG(*.ico)文件进行gzip压缩

在vue-cli脚手架的配置信息中，有对代码进行压缩的配置项，例如index.js的通用配置，productionGzip设置为true，但是首先需要对compress-webpack-plugin支持，所以需要通过 npm install --save-dev compression-webpack-plugin(如果npm install出错了，就使用cnpm install安装。可能网络比较差npm install会出现频率比较大)，gzip会对js、css文件进行压缩处理；对于图片进行压缩问题，

对于png，jpg，jpeg没有压缩效果，对于svg，ico文件以及bmp文件压缩效果达到50%，在productionGzipExtensions: ['js', 'css','svg']设置需要进行压缩的什么格式的文件。对项目文件进行压缩之后，需要浏览器客户端支持gzip以及后端支持gzip。下面可以查看成功支持gzip状态：

（3）对路由组件进行懒加载

在路由配置文件里，这里是router.js里面引用组件。如果使用同步的方式加载组件，在首屏加载时会对网络资源加载加载比较多，资源比较大，加载速度比较慢。所以设置路由懒加载，按需加载会加速首屏渲染。在没有对路由进行懒加载时，在Chrome里devtool查阅可以看到首屏网络资源加载情况（6requests 3.8MB transfferred Finish:4.67s DOMContentLoaded 2.61s Load 2.70s）。在对路由进行懒加载之后（7requests 800kb transffered Finish2.67s DOMContentLoaded 1.72s Load 800ms）,可以看见加载速度明显加快。但是进行懒加载之后，实现按需加载，那么项目打包不会把所有js打包进app.[hash].js里面，优点是可以减少app.[hash].js体积，缺点就是会把其它js分开打包，造成多个js文件，会有多次https请求。如果项目比较大，需要注意懒加载的效果。

②源码优化

（1）v-if 和 v-show选择调用

v-show和v-if的区别是：v-if是懒加载，当状态为true时才会加载，并且为false时不会占用布局空间；v-show是无论状态是true或者是false，都会进行渲染，并对布局占据空间对于在项目中，需要频繁调用，不需要权限的显示隐藏，可以选择使用v-show，可以减少系统的切换开销。

（2）为item设置唯一key值，

在列表数据进行遍历渲染时，需要为每一项item设置唯一key值，方便vuejs内部机制精准找到该条列表数据。当state更新时，新的状态值和旧的状态值对比，较快地定位到diff。

（3）细分vuejs组件

在项目开发过程之中，第一版本把所有的组件的布局写在一个组件中，当数据变更时，由于组件代码比较庞大，vuejs的数据驱动视图更新比较慢，造成渲染比较慢。造成比较差的体验效果。所以把组件细分，比如一个组件，可以把整个组件细分成轮播组件、列表组

件、分页组件等。

（4）减少watch的数据

当组件某个数据变更后需要对应的state进行变更，就需要对另外的组件进行state进行变更。可以使用watch监听相应的数据变更并绑定事件。当watch的数据比较小，性能消耗不明显。当数据变大，系统会出现卡顿，所以减少watch的数据。其它不同的组件的state双向绑定，可以采用事件中央总线或者vuex进行数据的变更操作。

（5）内容类系统的图片资源按需加载

对于内容类系统的图片按需加载，如果出现图片加载比较多，可以先使用v-lazy之类的懒加载库或者绑定鼠标的scroll事件，滚动到可视区域先再对数据进行加载显示，减少系统加载的数据。

（6）SSR(服务端渲染)

如果项目比较大，首屏无论怎么做优化，都出现闪屏或者一阵黑屏的情况。可以考虑使用SSR(服务端渲染)，vuejs官方文档提供next.js很好的服务端解决方案，但是局限性就是目前仅支持Koa、express等Nodejs的后台框架，需要webpack支持。目前自己了解的就是后端支持方面，vuejs的后端渲染支持php，其它的不太清楚。

③用户体验优化

（1）better-click防止iphone点击延迟

在开发移动端vuejs项目时，手指触摸时会出现300ms的延迟效果，可以采用better-click对ipone系列的兼容体验优化。

（2）loading

loading，在加载资源过程之中，可以提供loading。

（3）骨架屏加载

在首屏加载资源较多，可能会出现白屏和闪屏的情况。体验不好。盗图一波，小米商城使用骨架屏进行首屏在资源数据还没有加载完成时显示，给很好的体验效果。

61、如何实现一个组件封装?

①js前端组件的封装方法

定义一个类

类中增加一个方法

body中定义一个dom节点

脚本中把dom节点和类定义结合起来，实现特定的组件功能

②vue组件封装

建立组件的模板，先把架子搭起来，写写样式，考虑你的组件的基本逻辑

然后在引用得组件中 用import引入组件

通过component定义组件名称

在把组件以标签的形式写出来。

③react组件封装

创建一个react文件，搭建模板

把组件内的内容写清楚

使用export 把组件曝光

使用import把组件导入

62、react事件与原生的区别?

React合成事件一套机制：React并不是将click事件直接绑定在dom上面，而是采用事件冒泡的形式冒泡到document上面，然后React将事件封装给正式的函数处理运行和处理。

63、封装原生promise函数

【https://www.cnblogs.com/-roc/p/9985127.html】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script>
    /*
    *Promise实现思路
    * 1.构造函数
    * 2.回调函数的参数 resolve reject
    * 3.链式调用.then .catch
    */

    function PromiseM(cb){
        //初始状态
        this.status = "pending";
        this.msg = "";

        cb((data)=>{
```

```
            this.status = "resolve";
            this.msg = data;
        },()=>{
            this.status = "reject";
        })
        return this;
    }

    //链式调用.then
    PromiseM.prototype.then = function(){
        var cb = arguments;

        //轮询实现异步
        timer = setInterval(()=>{
            if(this.status == "resolve"){
                //成功状态的回调
                cb[0](this.msg);
                clearInterval(timer);
            }else if(this.status == "reject"){
                //失败状态的回调
                cb[1](this.msg);
                clearInterval(timer);
            }
        },3000)
    }

    new PromiseM(function (resolve,reject){
        setTimeout(function (){
            console.log(1111);
            resolve('11111');/*reject也就是失败时对应的函数，由于这个例子比较简单*/
        },1000)
    }).then(function (data){
        console.log("接收的值为"+data);
    })

</script>
</body>
</html>
```

65、Async/Await：比Promise更好的6个理由

原文链接：http://caibaojian.com/asyncawait.html
①什么是Async/Await?
async/await是写异步代码的新方式，以前的方法有回调函数和Promise。
async/await是基于Promise实现的，它不能用于普通的回调函数。
async/await与Promise一样，是非阻塞的。
async/await使得异步代码看起来像同步代码，这正是它的魔力所在。

②
（1）简洁
由示例可知，使用Async/Await明显节约了不少代码。我们不需要写.then，不需要写匿名函数处理Promise的resolve值，也不需要定义多余的data变量，还避免了嵌套代码。这些小的优点会迅速累计起来，这在之后的代码示例中会更加明显。·
（2）错误处理
Async/Await让try/catch可以同时处理同步和异步错误。在下面的promise示例中，try/catch不能处理JSON.parse的错误，因为它在Promise中。我们需要使用.catch，这样错误处理代码非常冗余。并且，在我们的实际生产代码会更加复杂。·
（3）条件语句
（4）中间值
（5）错误栈
（6）调试

66、前端开发中的设计模式

①单例模式：单例模式的定义是保证一个类只有一个实例，并且提供一个访问它的全局访问点。有些时候一些对象我们往往只需要一个，比如线程池、全局缓存、浏览器中的window对象等。
单例模式的优点是：
可以用来划分命名空间，减少全局变量的数量
使用单例模式可以使代码组织的更为一致，使代码容易阅读和维护
可以被实例化，且实例化一次
②观察者模式：观察者模式又叫做发布-订阅模式，它定义了对象间的一种一对多的依赖关系，当一个对象的状态发生变化时，所有依赖于他的对象都将得到通知，在javascript的开发中，一般用事件模

型来替代传统的发布 – 订阅模式。
③命令模式：命令模式中的命令（command）指的是一个执行某些特定事情的指令。
④职责链模式：使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系，将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理它为止。

67.状态码

①1xx 临时响应

100：客户端应当继续发送请求
101：客户端要求服务器根据请求转换HTTP协议版本
102：处理将继续被执行

②2xx 成功

200：请求成功
201：提示知道新文件的URL
202：接受和处理、但处理未完成
203：返回信息不确定或不完整
204：请求收到，但返回信息为空
205：服务器完成了请求，用户代理必须复位当前已经浏览过的文件
206：服务器已经完成了部分用户的GET请求
207：之后的消息体将是一个XML消息，并且包含一系列独立的响应代码

③3xx 重定向

300：请求的资源可在多处得到
301：客户请求的文档在其他地方，新的URL在Location头中给出，浏览器应该自动地访问新的URL。【被请求的资源已永久移动到新位置，并且将来任何对此资源的引用都应该使用本响应返回的若干个】
302：在其他地址发现了请求数据【请求的资源现在临时从不同的响应请求。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求】
303：建议客户访问其他URL或访问方式
304：客户端已经执行了GET，但文件未变化
305：请求的资源必须从服务器指定的地址得到

306：前一版本HTTP中使用的代码，现行版本中不再使用
307：请求的资源现在临时从不同的URI 响应请求。
④4xx 客户端错误
400：错误请求，如语法错误
401：请求授权失败
402：保留有效ChargeTo头响应
403：服务器已经理解请求，但是拒绝执行它
404：请求失败，请求所希望得到的资源未被在服务器上发现
405：请求行中指定的请求方法不能被用于请求相应的资源
406：请求的资源的内容特性无法满足请求头中的条件，因而无法生成响应实体
407：类似401，用户必须首先在代理服务器上得到授权
408：客户端没有在用户指定的时间内完成请求
409：由于和被请求的资源的当前状态之间存在冲突，请求无法完成
410：服务器上不再有此资源且无进一步的参考地址
411：服务器拒绝用户定义的Content-Length属性请求
412：服务器在验证在请求的头字段中给出先决条件时，没能满足其中的一个或多个
413：请求的资源大于服务器允许的大小
414：请求的资源URL长于服务器允许的长度
415：请求中提交的实体并不是服务器中所支持的格式，因此请求被拒绝
416：请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段
417：服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求
421：从当前客户端所在的IP地址到服务器的连接数超过了服务器许可的最大范围
424：由于之前的某个请求发生的错误，导致当前请求失败
426：客户端应当切换到TLS/1.0
440：客户端的会话已经过期，必须重新登录
449：客户端没有提供所需要的信息，服务器不能正确响应
495：客户端提供了一个无效的证书
496：需要客户端提供 SSL 证书，客户端却没有提供

497：HTTP 请求发送到了 HTTPS 端口
499：在服务器发送响应之前客户端就关闭了连接
⑤5xx 服务器错误
500：服务器产生内部错误【伺服器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。】
501：服务器不支持请求的函数
502：服务器暂时不可用，有时是为了防止发生系统过载
503：服务器过载或暂停维修
504：关口过载，服务器使用另一个关口或服务来响应用户，等待时间设定值较长
505：服务器不支持或拒绝支请求头中指定的HTTP版本
506：服务器存在内部配置错误
507：服务器无法存储完成请求所必须的内容
509：服务器达到带宽限制
510：获取资源所需要的策略并没有满足
520：未知错误
521：Web 服务器已关闭
522：没有和源服务器完成 TCP 握手
523：云服务器无法到达原始服务器，可能是源服务器的 DNS 记录不正确
524：云服务器可以完成 TCP 握手，但是没有及时收到 HTTP 响应
525：SSL 握手失败
526：无效的 SSL 证书

68、盟迅面试

(1) vue的理解
(2) react的理解
(3) vue与react的区别
(4) react比起vue有什么好处
(5) 对webpack的理解，他是怎么进行打包的
(6) vue的组件通讯
(7) react为什么能够兼容ie8
(8) vue双向数据绑定的原理
(9) vue组件通讯(注入vue实例的方法）的原理
(10)用过那些ui框架

(11)原型和原型链的理解
(12)vue项目的优化
(13)闭包的理解

69、面试重点

html5与css3新特性+ES6+继承+类+闭包+promise底层原理+vue相关知识 （vue+react的区别）+项目流程（公司介绍业务介绍+介绍项目+项目难点）+wepack一些东西（打包时间+打包体积减少）+性能优化+布局+兼容性问题

==========webpack面试题==========

1、什么是webpack和grunt和gulp有什么不同

Webpack是一个模块打包器，他可以递归的打包项目中的所有模块，最终生成几个打包后的文件。他和其他的工具最大的不同在于他支持code-splitting、模块化(AMD，ESM，CommonJs)、全局分析。

2、什么是bundle,什么是chunk，什么是module?

bundle是由webpack打包出来的文件，chunk是指webpack在进行模块的依赖分析的时候，代码分割出来的代码块。module是开发中的单个模块。

3、什么是Loader?什么是Plugin?

1）Loaders是用来告诉webpack如何转化处理某一类型的文件，并且引入到打包出的文件中

2）Plugin是用来自定义webpack打包过程的方式，一个插件是含有apply方法的一个对象，通过这个方法可以参与到整个webpack打包的各个流程(生命周期)。

4、配置问题:如何可以自动生成webpack配置?

答案： webpack-cli /vue-cli /etc ...脚手架工具

5、开发问题一：webpack-dev-server和http服务器如nginx有什么区别?

答案：webpack-dev-server使用内存来存储webpack开发环境下的打包文件，并且可以使用模块热更新，他比传统的http服务对开发更加简单高效。

6、开发问题二:什么 是模块热更新?

答案:模块热更新是webpack的一个功能，他可以使得代码修改过后

不用刷新浏览器就可以更新，是高级版的自动刷新浏览器。

7、优化问题一：什么是长缓存？在webpack中如何做到长缓存优化?

答案：浏览器在用户访问页面的时候，为了加快加载速度，会对用户访问的静态资源进行存储，但是每一次代码升级或是更新，都需要浏览器去下载新的代码，最方便和简单的更新方式就是引入新的文件名称。在webpack中可以在output纵输出的文件指定chunkhash,并且分离经常更新的代码和框架代码。通过NameModulesPlugin或是HashedModuleIdsPlugin使再次打包文件名不变。

8、优化问题二：什么是Tree-shaking?CSS可以Tree-shaking吗

答案：Tree-shaking是指在打包中去除那些引入了，但是在代码中没有被用到的那些死代码。在webpack中Tree-shaking是通过uglifySPlugin来Tree-shaking
JS。Css需要使用Purify-CSS。

9、几个常见的loader

file-loader：把文件输出到一个文件夹中，在代码中通过相对 URL 去引用输出的文件

url-loader：和 file-loader 类似，但是能在文件很小的情况下以 base64 的方式把文件内容注入到代码中去

source-map-loader：加载额外的 Source Map 文件，以方便断点调试

image-loader：加载并且压缩图片文件

babel-loader：把 ES6 转换成 ES5

css-loader：加载 CSS，支持模块化、压缩、文件导入等特性

style-loader：把 CSS 代码注入到 JavaScript 中，通过 DOM 操作去加载 CSS。

eslint-loader：通过 ESLint 检查 JavaScript 代码

***10、什么是webpack**

webpack是一个打包模块化javascript的工具，在webpack里一切文件皆模块，通过loader转换文件，通过plugin注入钩子，最后输出由多个模块组合成的文件，webpack专注构建模块化项目。

WebPack可以看做是模块打包机：它做的事情是，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展

语言（Scss，TypeScript等），并将其打包为合适的格式以供浏览器使用。

11、几个常见的plugin

define-plugin：定义环境变量

terser-webpack-plugin：通过TerserPlugin压缩ES6代码

html-webpack-plugin 为html文件中引入的外部资源，可以生成创建html入口文件

mini-css-extract-plugin：分离css文件

clean-webpack-plugin：删除打包文件

happypack：实现多线程加速编译

12、webpack与grunt、gulp的不同？

Webpack与Gulp、Grunt没有什么可比性，它可以看作模块打包机，通过分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其转换和打包为合适的格式供浏览器使用。Gulp/Grunt是一种能够优化前端的开发流程的工具，而WebPack是一种模块化的解决方案，不过Webpack的优点使得Webpack在很多场景下可以替代Gulp/Grunt类的工具。

他们的工作方式也有较大区别：

Grunt和Gulp的工作方式是：在一个配置文件中，指明对某些文件进行类似编译，组合，压缩等任务的具体步骤，工具之后可以自动替你完成这些任务。

Webpack的工作方式是：把你的项目当做一个整体，通过一个给定的主文件（如：index.js），Webpack将从这个文件开始找到你的项目的所有依赖文件，使用loaders处理它们，最后打包为一个（或多个）浏览器可识别的JavaScript文件。

三者都是前端构建工具，grunt和gulp在早期比较流行，现在webpack相对来说比较主流，不过一些轻量化的任务还是会用gulp来处理，比如单独打包CSS文件等。

grunt和gulp是基于任务和流（Task、Stream）的。类似jQuery，找到一个（或一类）文件，对其做一系列链式操作，更新流上的数据， 整条链式操作构成了一个任务，多个任务就构成了整个web的构建流程。

webpack是基于入口的。webpack会自动地递归解析入口所需要加

载的所有资源文件，然后用不同的Loader来处理不同的文件，用Plugin来扩展webpack功能。

*****13、webpack的优缺点**

webpack优点：

专注于处理模块化的项目，能做到开箱即用，一步到位

可通过plugin扩展，完整好用又不失灵活

使用场景不局限于web开发

社区庞大活跃，经常引入紧跟时代发展的新特性，能为大多数场景找到已有的开源扩展

良好的开发体验

webpack的缺点：

webpack的缺点是只能用于采用模块化开发的项目

14、分别介绍什么是loader?什么是plugin?

loader：模块转换器，用于将模块的原内容按照需要转成你想要的内容

plugin：在webpack构建流程中的特定时机注入扩展逻辑，来改变构建结果，是用来自定义webpack打包过程的方式，一个插件是含有apply方法的一个对象，通过这个方法可以参与到整个webpack打包的各个流程(生命周期)。

15、什么 是模块热更新?

模块热更新是webpack的一个功能，他可以使得代码修改过后不用刷新浏览器就可以更新，是高级版的自动刷新浏览器。

devServer中通过hot属性可以空时模块的热替换

16、通过webpack处理长缓存

浏览器在用户访问页面的时候，为了加快加载速度，会对用户访问的静态资源进行存储，但是每一次代码升级或是更新，都需要浏览器去下载新的代码，最方便和简单的更新方式就是引入新的文件名称。在webpack中可以在output纵输出的文件指定chunkhash,并且分离经常更新的代码和框架代码。通过NameModulesPlugin或是HashedModuleIdsPlugin使再次打包文件名不变。

17、如何提高webpack的构建速度

①通过externals配置来提取常用库

②利用DllPlugin和DllReferencePlugin预编译资源模块 通过DllPlugin来对那些我们引用但是绝对不会修改的npm包来进行预编译，再通

过DllReferencePlugin将预编译的模块加载进来。
③使用Happypack 实现多线程加速编译
要注意的第一点是，它对file-loader和url-loader支持不好，所以这两个loader就不需要换成happypack了，其他loader可以类似地换一下
④使用Tree-shaking和Scope Hoisting来剔除多余代码
⑤使用fast-sass-loader代替sass-loader
⑥babel-loader开启缓存
babel-loader在执行的时候，可能会产生一些运行期间重复的公共文件，造成代码体积大冗余，同时也会减慢编译效率
可以加上cacheDirectory参数或使用 transform-runtime 插件试试
⑦不需要打包编译的插件库换成全局"script"标签引入的方式
比如jQuery插件，react， react-dom等，代码量是很多的，打包起来可能会很耗时
可以直接用标签引入，然后在webpack配置里使用 expose-loader 或 externals 或 ProvidePlugin 提供给模块内部使用相应的变量
⑧优化构建时的搜索路径
在webpack打包时，会有各种各样的路径要去查询搜索，我们可以加上一些配置，让它搜索地更快
比如说，方便改成绝对路径的模块路径就改一下，以纯模块名来引入的可以加上一些目录路径，还可以善于用下resolve alias别名 这个字段来配置，还有exclude等的配置，避免多余查找的文件，比如使用babel别忘了剔除不需要遍历的。

18、 webpack的构建流程是什么

①初始化参数：从配置文件和 Shell 语句中读取与合并参数，得出最终的参数

②开始编译：用上一步得到的参数初始化 Compiler 对象，加载所有配置的插件，执行对象的 run 方法开始执行编译

Saber确定入口：根据配置中的 entry 找出所有的入口文件

④编译模块：从入口文件出发，调用所有配置的 Loader 对模块进行翻译，再找出该模块依赖的模块，再递归本步骤直到所有入口依赖的文件都经过了本步骤的处理

⑤完成模块编译：在经过第4步使用 Loader 翻译完所有模块后，得到了每个模块被翻译后的最终内容以及它们之间的依赖关系

⑥输出资源：根据入口和模块之间的依赖关系，组装成一个个包含多个模块的 Chunk，再把每个 Chunk 转换成一个单独的文件加入到输出列表，这步是可以修改输出内容的最后机会

⑦输出完成：在确定好输出内容后，根据配置确定输出的路径和文件名，把文件内容写入到文件系统

***19、如何利用webpack来优化前端性能

①压缩代码。删除多余的代码、注释、简化代码的写法等等方式

②利用 CDN 加速。在构建过程中，将引用的静态资源路径修改为 CDN 上对应的路径

③删除死代码 Tree Shaking）。将代码中永远不会走到的片段删除掉

④优化图片，对于小图可以使用 base64 的方式写入文件中

⑤按照路由拆分代码，实现按需加载，提取公共代码

⑥给打包出来的文件名添加哈希，实现浏览器缓存文件

20、怎么配置单页应用？怎么配置多页应用

单页应用可以理解为 webpack 的标准模式，直接在 entry 中指定单页应用的入口即可

多页应用的话，可以使用 webpack 的 AutoWebPlugin 来完成简单自动化的构建，但是前提是项目的目录结构必须遵守他预设的规范

21、css reset 和 normalize.css 有什么区别

两者都是通过重置样式，保持浏览器样式的一致性

前者几乎为所有标签添加了样式，后者保持了许多浏览器样式，保持尽可能的一致

后者修复了常见的桌面端和移动端浏览器的bug：包含了HTML5元素的显示设置、预格式化文字的font-size问题、在IE9中SVG的溢出、许多出现在各浏览器和操作系统中的与表单相关的bug。

前者中含有大段的继承链

后者模块化，文档较前者来说丰富

==========React面试题==========

1、React 的工作原理

React 会创建一个虚拟 DOM(virtual DOM)。当一个组件中的状态改变时，React 首先会通过 "diffing" 算法来标记虚拟 DOM 中的改变，第二步是调节(reconciliation)，会用 diff 的结果来更新 DOM。

2、使用 React 有何优点

只需查看 render 函数就会很容易知道一个组件是如何被渲染的

JSX 的引入，使得组件的代码更加可读，也更容易看懂组件的布局，或者组件之间是如何互相引用的

支持服务端渲染，这可以改进 SEO 和性能

易于测试

React 只关注 View 层，所以可以和其它任何框架(如Backbone.js, Angular.js)一起使用

3、展示组件(Presentational component)和容器组件(Container component)之间有何不同

展示组件关心组件看起来是什么。展示专门通过 props 接受数据和回调，并且几乎不会有自身的状态，但当展示组件拥有自身的状态时，通常也只关心 UI 状态而不是数据的状态。

容器组件则更关心组件是如何运作的。容器组件会为展示组件或者其它容器组件提供数据和行为(behavior)，它们会调用 Flux actions，并将其作为回调提供给展示组件。容器组件经常是有状态的，因为它们是(其它组件的)数据源。

4、类组件(Class component)和函数式组件(Functional component)之间有何不同

类组件不仅允许你使用更多额外的功能，如组件自身的状态和生命周期钩子，也能使组件直接访问 store 并维持状态

当组件仅是接收 props，并将组件自身渲染到页面时，该组件就是一个 '无状态组件(stateless component)'，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件

5、(组件的)状态(state)和属性(props)之间有何不同

State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作为用户事件行为的结果。

Props(properties 的简写)则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的(immutable)。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)。Props 也不仅仅是数据--回调函数也可以通过 props 传递。

6、指出(组件)生命周期方法的不同

componentWillMount -- 多用于根组件中的应用程序配置

componentDidMount -- 在这可以完成所有没有 DOM 就不能做的所有配置，并开始获取所有你需要的数据；如果需要设置事件监听，也可以在这完成

componentWillReceiveProps -- 这个周期函数作用于特定的 prop 改变导致的 state 转换

shouldComponentUpdate -- 如果你担心组件过度渲染，shouldComponentUpdate 是一个改善性能的地方，因为如果组件接收了新的 prop，它可以阻止(组件)重新渲染。shouldComponentUpdate 应该返回一个布尔值来决定组件是否要重新渲染

componentWillUpdate -- 很少使用。它可以用于代替组件的 componentWillReceiveProps 和 shouldComponentUpdate(但不能访问之前的 props)

componentDidUpdate -- 常用于更新 DOM，响应 prop 或 state 的改变

componentWillUnmount -- 在这你可以取消网络请求，或者移除所有与组件相关的事件监听器

7、应该在 React 组件的何处发起 Ajax 请求

在 React 组件中，应该在 componentDidMount 中发起网络请求。这个方法会在组件第一次“挂载”(被添加到 DOM)时执行，在组件的生命周期中仅会执行一次。更重要的是，你不能保证在组件挂载之前 Ajax 请求已经完成，如果是这样，也就意味着你将尝试在一个未挂载的组件上调用 setState，这将不起作用。在 componentDidMount 中发起网络请求将保证这有一个组件可以更新了。

8、何为受控组件(controlled component)

在 HTML 中，类似 <input>，<textarea> 和 <select> 这样的表单元素会维护自身的状态，并基于用户的输入来更新。当用户提交表单时，前面提到的元素的值将随表单一起被发送。但在 React 中会有些不同，包含表单元素的组件将会在 state 中追踪输入的值，并且每次调用回调函数时，如 onChange 会更新 state，重新渲染组件。一个输入表单元素，它的值通过 React 的这种方式来控制，这样的元素就被称为"受控元素"。

9、在 React 中，refs 的作用是什么

Refs 可以用于获取一个 DOM 节点或者 React 组件的引用。何时使用 refs 的好的示例有管理焦点/文本选择，触发命令动画，或者和第三方 DOM 库集成。你应该避免使用 String 类型的 Refs 和内联的 ref 回调。Refs 回调是 React 所推荐的。

10、何为高阶组件(higher order component)

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC 运行你重用代码、逻辑和引导抽象。最常见的可能是 Redux 的 connect 函数。除了简单分享工具库和简单的组合，HOC最好的方式是共享 React 组件之间的行为。如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的 HOC。

11、使用箭头函数(arrow functions)的优点是什么

作用域安全：在箭头函数之前，每一个新创建的函数都有定义自身的 this 值(在构造函数中是新对象；在严格模式下，函数调用中的 this 是未定义的；如果函数被称为“对象方法”，则为基础对象等)，但箭头函数不会，它会使用封闭执行上下文的 this 值。

简单：箭头函数易于阅读和书写

清晰：当一切都是一个箭头函数，任何常规函数都可以立即用于定义作用域。开发者总是可以查找 next-higher 函数语句，以查看 this 的值

12、为什么建议传递给 setState 的参数是一个 callback 而不是一个对象

因为 this.props 和 this.state 的更新可能是异步的，不能依赖它们的值去计算下一个 state。

13、除了在构造函数中绑定 this，还有其它方式吗

你可以使用属性初始值设定项(property initializers)来正确绑定回调，create-react-app 也是默认支持的。在回调中你可以使用箭头函数，但问题是每次组件渲染时都会创建一个新的回调。**14、怎么阻止组件的渲染**

在组件的 render 方法中返回 null 并不会影响触发组件的生命周期方法

15、当渲染一个列表时，何为 key？设置 key 的目的是什么

Keys 会有助于 React 识别哪些 items 改变了，被添加了或者被移除了。Keys 应该被赋予数组内的元素以赋予(DOM)元素一个稳定的标识，选择一个 key 的最佳方法是使用一个字符串，该字符串能惟一地标识一个列表项。很多时候你会使用数据中的 IDs 作为 keys，当你没有稳定的 IDs 用于被渲染的 items 时，可以使用项目索引作为渲染项的 key，但这种方式并不推荐，如果 items 可以重新排序，就会导致 re-render 变慢。

16、(在构造函数中)调用 super(props) 的目的是什么

在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。

17、何为 JSX

JSX 是 JavaScript 语法的一种语法扩展，并拥有 JavaScript 的全部功能。JSX 生产 React "元素"，你可以将任何的 JavaScript 表达式封装在花括号里，然后将其嵌入到 JSX 中。在编译完成之后，JSX 表达式就变成了常规的 JavaScript 对象，这意味着你可以在 if 语句和 for 循环内部使用 JSX，将它赋值给变量，接受它作为参数，并从函数中返回它。

18、在 React 中，何为 state

State 和 props 类似，但它是私有的，并且完全由组件自身控制。State 本质上是一个持有数据，并决定组件如何渲染的对象。

19、什么原因会促使你脱离 create-react-app 的依赖

当你想去配置 webpack 或 babel presets。

20、何为 redux

Redux 的基本思想是整个应用的 state 保持在一个单一的 store 中。store 就是一个简单的 javascript 对象，而改变应用 state 的唯一方式是在应用中触发 actions，然后为这些 actions 编写 reducers 来修改 state。整个 state 转化是在 reducers 中完成，并且不应该有任何副作用。

21、在 Redux 中，何为 store

Store 是一个 javascript 对象，它保存了整个应用的 state。与此同时，Store 也承担以下职责：

允许通过 getState() 访问 state

运行通过 dispatch(action) 改变 state

通过 subscribe(listener) 注册 listeners

通过 subscribe(listener) 返回的函数处理 listeners 的注销

22、何为 action

Actions 是一个纯 javascript 对象，它们必须有一个 type 属性表明正在执行的 action 的类型。实质上，action 是将数据从应用程序发送到 store 的有效载荷。

23、何为 reducer

一个 reducer 是一个纯函数，该函数以先前的 state 和一个 action 作为参数，并返回下一个 state。

24、Redux Thunk 的作用是什么

Redux thunk 是一个允许你编写返回一个函数而不是一个 action 的 actions creators 的中间件。如果满足某个条件，thunk 则可以用来延迟 action 的派发(dispatch)，这可以处理异步 action 的派发(dispatch)。

25、何为纯函数(pure function)

一个纯函数是一个不依赖于且不改变其作用域之外的变量状态的函数，这也意味着一个纯函数对于同样的参数总是返回同样的结果。

26、简述 flux 思想

Flux 的最大特点，就是数据的"单向流动"。

①用户访问 View

②View 发出用户的 Action

③Dispatcher 收到 Action，要求 Store 进行相应的更新

④Store 更新后，发出一个"change"事件

⑤View 收到"change"事件后，更新页面

27、了解 redux 么，说一下 redux 把

redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，主要有三个核心方法，action，store，reducer，工作流程是 view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据，flux 也是用来进行数据操作的，有四个组成部分 action，dispatch，view，store，工作流程是 view 发出一个 action，派发器接收 action，让 store 进行数据更新，更新完成以后 store 发出 change，view 接受 change 更新视图。Redux 和 Flux 很像。主要区别在于 Flux 有多个可以改变应用状态的 store，在 Flux 中 dispatcher 被用来传递数据到注册的回调事件，但是在 redux 中只能定义一个可更新状态的 store，redux 把 store 和 Dispatcher 合并,结构更加简单清晰

新增 state,对状态的管理更加明确，通过 redux，流程更加规范了，减少手动编码量，提高了编码效率，同时缺点时当数据更新时有时候组件不需要，但是也要重新绘制，有些影响效率。一般情况下，我们在构建多交互，多数据流的复杂项目应用时才会使用它们

28、React 中有三种构建组件的方式

React.createClass()、ES6 class 和无状态函数。

29、(在构造函数中)调用 super(props) 的目的是什么

在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。

30、描述事件在 React 中的处理方式。

为了解决跨浏览器兼容性问题，您的 React 中的事件处理程序将传递 SyntheticEvent 的实例，它是 React 的浏览器本机事件的跨浏览器包装器。

这些 SyntheticEvent 与您习惯的原生事件具有相同的接口，除了它们在所有浏览器中都兼容。有趣的是，React 实际上并没有将事件附加到子节点本身。React 将使用单个事件监听器监听顶层的所有

事件。这对于性能是有好处的，这也意味着在更新 DOM 时，React 不需要担心跟踪事件监听器。

31、react 组件的划分业务组件技术组件?

根据组件的职责通常把组件分为 UI 组件和容器组件。

UI 组件负责 UI 的呈现，容器组件负责管理数据和逻辑。

两者通过 React-Redux 提供 connect 方法联系起来。

32、createElement 和 cloneElement 有什么区别?

React.createElement():JSX 语法就是用 React.createElement()来构建 React 元素的。它接受三个参数，第一个参数可以是一个标签名。如 div、span，或者 React 组件。第二个参数为传入的属性。第三个以及之后的参数，皆作为组件的子组件。

```
React.createElement(
    type,
    [props],
    [...children]
)
```

React.cloneElement()与 React.createElement()相似，不同的是它传入的第一个参数是一个 React 元素，而不是标签名或组件。新添加的属性会并入原有的属性，传入到返回的新元素中，而就的子元素奖杯替换。

```
React.cloneElement(
  element,
  [props],
  [...children]
)
```

= （重点）Vue面试题 ==========

1、构建的 vue-cli 工程都到了哪些技术，它们的作用分别是什么?

①vue.js：vue-cli工程的核心，主要特点是 双向数据绑定 和 组件系统。

②vue-router：vue官方推荐使用的路由框架。

③vuex：专为 Vue.js 应用项目开发的状态管理器，主要用于维护vue组件间共用的一些 变量 和 方法。

④axios（或者 fetch 、ajax ）：用于发起 GET 、或 POST 等 http 请求，基于 Promise 设计。
⑤vux等：一个专为vue设计的移动端UI组件库。
⑥创建一个emit.js文件，用于vue事件机制的管理。
⑦webpack：模块加载和vue-cli工程打包器。

2、vue-cli 工程常用的 npm 命令有哪些?

下载 node_modules 资源包的命令：npm install

启动 vue-cli 开发环境的 npm命令：npm run dev

vue-cli 生成 生产环境部署资源 的 npm命令：npm run build

用于查看 vue-cli 生产环境部署资源文件大小的 npm命令：npm run build --report

3、scss是什么？在vue.cli中的安装使用步骤是？有哪几大特性?

答：css的预编译。

使用步骤:

第一步：用npm 下三个loader（sass-loader、css-loader、node-sass）

第二步：在build目录找到webpack.base.config.js，在那个extends属性中加一个拓展.scss

第三步：还是在同一个文件，配置一个module属性

第四步：然后在组件的style标签加上lang属性 ，例如：lang=”scss”

有哪几大特性:

①可以用变量，例如（$变量名称=值）；

②可以用混合器，例如（）

③可以嵌套

4、mint-ui是什么？怎么使用？说出至少三个组件使用方法?

答：基于vue的前端组件库。npm安装，然后import样式和js，vue.use（mintUi）全局引入。在单个组件局部引入：import {Toast} from ‘mint-ui’。组件一：Toast(‘登录成功’)；组件二：mint-header；组件三：mint-swiper

5、axios+tp5进阶中，调用axios.post(‘api/user’)是进行的什么操作？axios.put(‘api/user/8’)呢?

答：跨域，添加用户操作，更新操作。

6、什么是RESTful API？怎么使用？

答：是一个api的标准，无状态请求。请求的路由地址是固定的，如果是tp5则先路由配置中把资源路由配置好。标准有：.post .put .delete

7、自定义指令（v-check、v-focus）的方法有哪些？它有哪些钩子函数？还有哪些钩子函数参数？

答：全局定义指令：在vue对象的directive方法里面有两个参数，一个是指令名称，另外一个是函数。组件内定义指令：directives

钩子函数：bind（绑定事件触发）、inserted(节点插入的时候触发)、update（组件内相关更新）

钩子函数参数：el、binding

8、请说下封装 vue 组件的过程？

答：首先，组件可以提升整个项目的开发效率。能够把页面抽象成多个相对独立的模块，解决了我们传统项目开发：效率低、难维护、复用性等问题。

然后，使用Vue.extend方法创建一个组件，然后使用Vue.component方法注册组件。子组件需要数据，可以在props中接受定义。而子组件修改好数据后，想把数据传递给父组件。可以采用emit方法。

***9、你是怎么认识vuex的？**

答：vuex可以理解为一种开发模式或框架。比如PHP有thinkphp，java有spring等。

通过状态（数据源）集中管理驱动组件的变化（好比spring的IOC容器对bean进行集中管理）。

应用级的状态集中放在store中；　　改变状态的方式是提交mutations，这是个同步的事物；　异步逻辑应该封装在action中。

10、vue-loader是什么？使用它的用途有哪些？

答：解析.vue文件的一个加载器，跟template/js/style转换成js模块。

用途：js可以写es6、style样式可以scss或less、template可以加jade等

11、请说出vue.cli项目中src目录每个文件夹和文件的用法?

答：assets文件夹是放静态资源；components是放组件；router是定义路由相关的配置;view视图；app.vue是一个应用主组件；main.js是入口文件

12、vue.cli中怎样使用自定义的组件？有遇到过哪些问题吗?

答：第一步：在components目录新建你的组件文件（smithButton.vue），script一定要export default {

第二步：在需要用的页面（组件）中导入：import smithButton from '../components/smithButton.vue'

第三步：注入到vue的子组件的components属性上面,components:{smithButton}

第四步：在template视图view中使用，<smith-button> </smith-button>

问题有：smithButton命名，使用的时候则smith-button。

13、Vuex取值

This.$store.state.city

This.$store.commit('getData')

this.$store.dispath('getData')

This.$store.getters.getData

14、Vuex中actions和mutations的区别

Mutations的更改是同步更改，用于用户执行直接数据更改，this.$store.commit('名')触发

Actions的更改是异步操作，用于需要与后端交互的数据更改,this.$store.dispath("名")触发

注意:

1)：定义actions方法创建一个更改函数时，这个函数必须携带一个context参数，用于触发mutations方法，context.commit('修改函数名','异步请求值');

2)：mutations第一个参数必须传入state，第二个参数是新值

15、组件的设计原则

(1)页面上每个独立的可视/可交互区域视为一个组件(比如页面的头部，尾部，可复用的区块)

(2)每个组件对应一个工程目录，组件所需要的各种资源在这个目录

下就近维护(组件的就近维护思想体现了前端的工程化思想，为前端开发提供了很好的分治策略，在vue.js中，通过.vue文件将组件依赖的模板，js，样式写在一个文件中)
(每个开发者清楚开发维护的功能单元，它的代码必然存在在对应的组件目录中，在该目录下，可以找到功能单元所有的内部逻辑)
(3)页面不过是组件的容器，组件可以嵌套自由组合成完整的页面

16、Virtual DOM算法，简单总结下包括几个步骤：

①用JS对象描述出DOM树的结构，然后在初始化构建中，用这个描述树去构建真正的DOM，并实际展现到页面中

②当有数据状态变更时，重新构建一个新的JS的DOM树，通过新旧对比DOM数的变化diff，并记录两棵树差异

③把步骤2中对应的差异通过步骤1重新构建真正的DOM，并重新渲染到页面中，这样整个虚拟DOM的操作就完成了，视图也就更新了

= **面 试 题 补 充** ==========

1、线程与进程的区别

a. 一个程序至少有一个进程，一个进程至少有一个线程

b. 线程的划分尺度小于进程，使得多线程程序的并发性高

c. 进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率

d. 每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制

e. 多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配

2、列举IE与其他浏览器不一样的特性?

a. IE的排版引擎是Trident （又称为MSHTML）

b. Trident内核曾经几乎与W3C标准脱节（2005年）

c. Trident内核的大量 Bug等安全性问题没有得到及时解决

d. JS方面，有很多独立的方法，例如绑定事件的attachEvent、创建事件的createEventObject等

e. CSS方面，也有自己独有的处理方式，例如设置透明，低版本IE中使用滤镜的方式

3、什么叫优雅降级和渐进增强?

渐进增强 progressive enhancement：

针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级 graceful degradation：

一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：

a. 优雅降级是从复杂的现状开始，并试图减少用户体验的供给

b. 渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要

c. 降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带

4、WEB应用从服务器主动推送Data到客户端有那些方式?

a. html5 websoket

b. WebSocket 通过 Flash

c. XHR长时间连接

d. XHR Multipart Streaming

e. 不可见的Iframe

f. 标签的长时间连接(可跨域)

5、AMD和CMD 规范的区别?

AMD 提前执行依赖 - 尽早执行，requireJS 是它的实现

CMD 按需执行依赖 - 懒执行，seaJS 是它的实现

6、网站重构的理解

重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。

a. 使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)

b. 对于移动平台的优化，针对于SEO进行优化

c. 减少代码间的耦合，让代码保持弹性

d. 压缩或合并JS、CSS、image等前端资源

7、浏览器的内核分别是什么?

IE浏览器的内核Trident;

Mozilla的Gecko;

Chrome的Blink（WebKit的分支）;

Opera内核原为Presto，现为Blink;

8、请介绍下cache-control

每个资源都可以通过 Cache-Control HTTP 头来定义自己的缓存策略

Cache-Control 指令控制谁在什么条件下可以缓存响应以及可以缓存多久

Cache-Control 头在 HTTP/1.1 规范中定义，取代了之前用来定义响应缓存策略的头（例如 Expires）。

9、前端页面有哪三层构成，分别是什么？作用是什么?

a. 结构层：由 HTML 或 XHTML 之类的标记语言负责创建，仅负责语义的表达。解决了页面“内容是什么”的问题。

b. 表示层：由CSS负责创建，解决了页面“如何显示内容”的问题。

c. 行为层：由脚本负责。解决了页面上“内容应该如何对事件作出反应”的问题。

10、知道的网页制作会用到的图片格式有哪些?

png-8，png-24，jpeg，gif，svg

Webp：谷歌（google）开发的一种旨在加快图片加载速度的图片格式。图片压缩体积大约只有JPEG的2/3，并能节省大量的服务器带宽资源和数据空间。Facebook Ebay等知名网站已经开始测试并使用WebP格式。

Apng：全称是“Animated Portable Network Graphics”，是PNG的位图动画扩展，可以实现png格式的动态图片效果。04年诞生，但一

直得不到各大浏览器厂商的支持，直到日前得到 iOS safari 8的支持，有望代替GIF成为下一代动态图标准

11、一次js请求一般情况下有哪些地方会有缓存处理?

a. 浏览器端存储

b. 浏览器端文件缓存

c. HTTP缓存304

d. 服务器端文件类型缓存

e. 表现层&DOM缓存

12、一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验[性能优化]。

a. 图片懒加载，滚动到相应位置才加载图片。

b. 图片预加载，如果为幻灯片、相册等，将当前展示图片的前一张和后一张优先下载。

c. 使用CSSsprite，SVGsprite，Iconfont、Base64等技术，如果图片为css图片的话。

d. 如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提高用户体验。

13、一个200*200的div在不同分辨率屏幕上下左右居中，css实现

```
div{
position:absolute;
width:200px;
height:200px;
top:50%;
left:50%;
margin-left:-100px;
height:-100px;
z-index:1000;
}
```

14、写一个左中右布局占满屏幕，其中左右两块是固定宽度200 ，中间自适应宽，要求先加载中间块，请写出结构及样式:

```
<body>
    <h3>实现三列宽度自适应布局</h3>
     <div id = "left">我是左边</div>
```

```
        <div id = "right">我是右边</div>
        <div id = "center">我是中间</div>
</body>
html,body{ margin: 0px;width: 100%; }
h3{height: 100px;margin:20px 0 0;}
#left,#right{width: 200px;height: 200px; background-color: #ffe6b8;position: absolute;top:120px;}#left{left:0px;}
#right{right: 0px;}
#center{margin:2px 210px ;background-color: #eee;height: 200px; }
```

15、Javascript中的定时器有哪些？他们的区别及用法是什么？

setTimeout 只执行一次

setInterval 会一直重复执行

16、document.write和innerHTML的区别：

document.write是直接写入到页面的内容流，如果在写之前没有调用document.open，浏览器会自动调用open。每次写完关闭之后重新调用该函数，会导致页面被重写。

innerHTML则是DOM页面元素的一个属性，代表该元素的html内容。你可以精确到某一个具体的元素来进行更改。如果想修改document的内容，则需要修改document.documentElement.innerElement。

innerHTML将内容写入某个DOM节点，不会导致页面全部重绘

innerHTML很多情况下都优于document.write，其原因在于其允许更精确的控制要刷新页面的那一个部分。

17、xml和json的区别，请用四个词语来形容

· JSON相对于XML来讲，数据的体积小，传递的速度更快些

· JSON与JavaScript的交互更加方便，更容易解析处理，更好的数据交互

· XML对数据描述性比较好；

· JSON的速度要远远快于XML

18、http 和 https 有何区别？如何灵活使用？

http是HTTP协议运行在TCP之上。所有传输的内容都是明文，客户端和服务器端都无法验证对方的身份。

https是HTTP运行在SSL/TLS之上，SSL/TLS运行在TCP之上。所有传输的内容都经过加密，加密采用对称加密，但对称加密的密钥用服务器方的证书进行了非对称加密。此外客户端可以验证服务器端的身份，如果配置了客户端验证，服务器方也可以验证客户端的身份。

19、bootstrap响应式实现的原理

百分比布局+媒体查询

20、怎样添加、移除、移动、复制、创建和查找节点?

1）创建新节点

createDocumentFragment() //创建一个DOM片段

createElement() //创建一个具体的元素

createTextNode() //创建一个文本节点

2）添加、移除、替换、插入

appendChild() //添加

removeChild() //移除

replaceChild() //替换

insertBefore() //插入

3）查找

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的Name属性的值

getElementById() //通过元素Id，唯一性

***21、谈谈你对webpack的看法

WebPack 是一个模块打包工具，你可以使用WebPack管理你的模块依赖，并编绎输出模块们所需的静态文件。它能够很好地管理、打包Web开发中所用到的HTML、JavaScript、CSS以及各种静态文件（图片、字体等），让开发过程更加高效。对于不同类型的资源，webpack有对应的模块加载器。webpack模块打包器会分析模块间的依赖关系，最后 生成了优化且合并后的静态资源。

webpack的两大特色：

①code splitting（可以自动完成）

②loader 可以处理各种类型的静态文件，并且支持串联操作

webpack 是以commonJS的形式来书写脚本滴，但对 AMD/CMD 的支持也很全面，方便旧项目进行代码迁移。

webpack具有requireJs和browserify的功能，但仍有很多自己的新特性：

①对 CommonJS 、 AMD、ES6的语法做了兼容

②对js、css、图片等资源文件都支持打包

③串联式模块加载器以及插件机制，让其具有更好的灵活性和扩展性，例如提供对CoffeeScript、ES6的支持

④有独立的配置文件webpack.config.js

⑤ 可以将代码切割成不同的chunk，实现按需加载，降低了初始化时间

⑥支持 SourceUrls 和SourceMaps，易于调试

⑦ 具有强大的Plugin接口，大多是内部插件，使用起来比较灵活

⑧webpack 使用异步 IO 并具有多级缓存。这使得 webpack 很快且在增量编译上更加快

22、说说TCP传输的三次握手四次挥手策略

为了准确无误地把数据送达目标处，TCP协议采用了三次握手策略。用TCP协议把数据包送出去后，TCP不会对传送 后的情况置之不理，它一定会向对方确认是否成功送达。握手过程中使用了TCP的标志：SYN和ACK。

发送端首先发送一个带SYN标志的数据包给对方。接收端收到后，回传一个带有SYN/ACK标志的数据包以示传达确认信息。

最后，发送端再回传一个带ACK标志的数据包，代表“握手”结束。

若在握手过程中某个阶段莫名中断，TCP协议会再次以相同的顺序发送相同的数据包。

断开一个TCP连接则需要“四次握手”：

§ 第一次挥手：主动关闭方发送一个FIN，用来关闭主动方到被动关闭方的数据传送，也就是主动关闭方告诉被动关闭方：我已经不 会再给你发数据了(当然，在fin包之前发送出去的数据，如果没有收到对应的ack确认报文，主动关闭方依然会重发这些数据)，但是，此时主动关闭方还可 以接受数据。

§ 第二次挥手：被动关闭方收到FIN包后，发送一个ACK给对方，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号）。

§ 第三次挥手：被动关闭方发送一个FIN，用来关闭被动关闭方到主动关闭方的数据传送，也就是告诉主动关闭方，我的数据也发送完了，不会再给你发数据了。

§ 第四次挥手：主动关闭方收到FIN后，发送一个ACK给被动关闭方，确认序号为收到序号+1，至此，完成四次挥手。

23、TCP和UDP的区别

TCP（Transmission Control Protocol，传输控制协议）是基于连接的协议，也就是说，在正式收发数据前，必须和对方建立可靠的连接。一个TCP连接必须要经过三次“对话”才能建立起来

UDP（User Data Protocol，用户数据报协议）是与TCP相对应的协议。它是面向非连接的协议，它不与对方建立连接，而是直接就把数据包发送过去！

UDP适用于一次只传送少量数据、对可靠性要求不高的应用环境。

24、说说你对作用域链的理解

作用域链的作用是保证执行环境里有权访问的变量和函数是有序的，作用域链的变量只能向上访问，变量访问到window对象即被终止，作用域链向下访问变量是不被允许的。

25、常见web安全及防护原理

sql注入原理

就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

总的来说有以下几点：

1.永远不要信任用户的输入，要对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双"-"进行转换等。

2.永远不要使用动态拼装SQL，可以使用参数化的SQL或者直接使用存储过程进行数据查询存取。

3.永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

4.不要把机密信息明文存放，请加密或者hash掉密码和敏感的信息。

26、XSS原理及防范

Xss(cross-site scripting)攻击指的是攻击者往Web页面里插入恶意 html标签或者javascript代码。比如：攻击者在论坛中放一个

看似安全的链接，骗取用户点击后，窃取cookie中的用户私密信息；或者攻击者在论坛中加一个恶意表单，

当用户提交表单的时候，却把信息传送到攻击者的服务器中，而不是用户原本以为的信任站点。

XSS防范方法

首先代码里对用户输入的地方和变量都需要仔细检查长度和对”<”,”>”,”;”,”’”等字符做过滤；其次任何内容写到页面之前都必须加以encode，避免不小心把html tag 弄出来。这一个层面做好，至少可以堵住超过一半的XSS 攻击。

首先，避免直接在cookie 中泄露用户隐私，例如email、密码等等。

其次，通过使cookie 和系统ip 绑定来降低cookie 泄露后的危险。这样攻击者得到的cookie 没有实际价值，不可能拿来重放。

如果网站不需要再浏览器端对cookie 进行操作，可以在Set-Cookie 末尾加上HttpOnly 来防止javascript 代码直接获取cookie 。

尽量采用POST 而非GET 提交表单

27、XSS与CSRF有什么区别吗?

XSS是获取信息，不需要提前知道其他用户页面的代码和数据包。CSRF是代替用户完成指定的动作，需要知道其他用户页面的代码和数据包。

要完成一次CSRF攻击，受害者必须依次完成两个步骤:

登录受信任网站A，并在本地生成Cookie。

在不登出A的情况下，访问危险网站B。

CSRF的防御

§ 服务端的CSRF方式方法很多样，但总的思想都是一致的，就是在客户端页面增加伪随机数。

§ 通过验证码的方法

28、Web Worker 和webSocket

worker主线程:

1.通过 worker = new Worker(url) 加载一个JS文件来创建一个worker，同时返回一个worker实例。

2.通过worker.postMessage(data) 方法来向worker发送数据。

3.绑定worker.onmessage方法来接收worker发送过来的数据。

4.可以使用 worker.terminate() 来终止一个worker的执行。

WebSocket是Web应用程序的传输协议，它提供了双向的，按序到达的数据流。他是一个Html5协议，WebSocket的连接是持久的，他通过在客户端和服务器之间保持双工连接，服务器的更新可以被及时推送给客户端，而不需要客户端以一定时间间隔去轮询。

29.*对前端模块化的认识**

所谓的模块化开发就是封装细节，提供使用接口，彼此之间互不影响，每个模块都是实现某一特定的功能。模块化开发的基础就是函数

目前，通行的JavaScript模块规范共有两种：CommonJS和AMD。

模块化开发的4点好处：

1 避免变量污染，命名冲突

2 提高代码复用率

3 提高维护性

4 依赖关系的管理

AMD 是 RequireJS 在推广过程中对模块定义的规范化产出。

CMD 是 SeaJS 在推广过程中对模块定义的规范化产出。

AMD 是提前执行，CMD 是延迟执行。

AMD推荐的风格通过返回一个对象做为模块对象，CommonJS的风格通过对module.exports或exports的属性赋值来达到暴露模块对象的目的。

CMD模块方式

```
define(function(require, exports,module) {
  // 模块代码
});
```

30、栈和队列的区别?

栈的插入和删除操作都是在一端进行的，而队列的操作却是在两端进行的。

队列先进先出，栈先进后出。

栈只允许在表尾一端进行插入和删除，而队列只允许在表尾一端进行插入，在表头一端进行删除

***31、ES6的了解

新增模板字符串（为JavaScript提供了简单的字符串插值功能）、箭头函数（操作符左边为输入的参数，而右边则是进行的操作以及返回的值Inputs=>outputs。）、for-of（用来遍历数据—例如数组中的值。）arguments对象可被不定参数和默认参数完美代替。ES6将promise对象纳入规范，提供了原生的Promise对象。增加了let和const命令，用来声明变量。增加了块级作用域。let命令实际上就增加了块级作用域。ES6规定，var命令和function命令声明的全局变量，属于全局对象的属性；let命令、const命令、class命令声明的全局变量，不属于全局对象的属性。。还有就是引入module模块的概念

32、js继承方式及其优缺点

原型链继承的缺点

一是字面量重写原型会中断关系，使用引用类型的原型，并且子类型还无法给超类型传递参数。

借用构造函数（类式继承）

借用构造函数虽然解决了刚才两种问题，但没有原型，则复用无从谈起。所以我们需要原型链+借用构造函数的模式，这种模式称为组合继承

组合式继承

组合式继承是比较常用的一种继承方法，其背后的思路是 使用原型链实现对原型属性和方法的继承，而通过借用构造函数来实现对实例属性的继承。这样，既通过在原型上定义方法实现了函数复用，又保证每个实例都有它自己的属性。

33、defer和async

defer并行加载js文件，会按照页面上script标签的顺序执行
async并行加载js文件，下载完成立即执行，不会按照页面上script标签的顺序执行

34、对BFC规范的理解?

BFC，块级格式化上下文，一个创建了新的BFC的盒子是独立布局的，盒子里面的子元素的样式不会影响到外面的元素。在同一个BFC中的两个毗邻的块级盒在垂直方向（和布局方向有关系）的margin会发生折叠。

（W3C CSS 2.1 规范中的一个概念，它决定了元素如何对其内容进行布局，以及与其他元素的关系和相互作用。

35、Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

1）、<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器以何种模式来渲染文档。

2）、严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。

3）、在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

4）、DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

36、你知道多少种Doctype文档类型?

该标签可声明三种 DTD 类型，分别表示严格版本、过渡版本以及基于框架的 HTML 文档。

HTML 4.01 规定了三种文档类型：Strict、Transitional 以及 Frameset。

XHTML 1.0 规定了三种 XML 文档类型：Strict、Transitional 以及 Frameset。

Standards （标准）模式（也就是严格呈现模式）用于呈现遵循最新标准的网页，而 Quirks

（包容）模式（也就是松散呈现模式或者兼容模式）用于呈现为传统浏览器而设计的网页。

37、HTML与XHTML——二者有什么区别

区别：

①所有的标记都必须要有一个相应的结束标记

②所有标签的元素和属性的名字都必须使用小写

③所有的XML标记都必须合理嵌套

④所有的属性必须用引号""括起来

⑤把所有<和&特殊符号用编码表示

⑥给所有属性赋一个值

⑦不要在注释内容中使“--”

⑧图片必须有说明文字

38、new操作符具体干了什么呢?

①创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。

②属性和方法被加入到 this 引用的对象中。

③新创建的对象由 this 所引用，并且最后隐式的返回 this 。

```
var obj  = {};
obj.__proto__ = Base.prototype;
Base.call(obj);
```

39、js延迟加载的方式有哪些?

defer和async、动态创建DOM方式（创建script，插入到DOM中，加载完毕后callBack）、按需异步载入js

****40、avascript对象的几种创建方式**

一.工厂模式

二.构造函数模式

三.原型模式

四.混合构造函数和原型模式

五.动态原型模式

六.寄生构造函数模式

七.稳妥构造函数模式

****41、javascript继承的6种方法**

(1) 原型链继承

(2) 借用构造函数继承

(3) 组合继承(原型+借用构造)

(4) 原型式继承

(5) 寄生式继承

(6) 寄生组合式继承

42、异步加载和延迟加载

一.异步加载的方案： 动态插入script标签

二.通过ajax去获取js代码，然后通过eval执行

三.script标签上添加defer或者async属性

四.创建并插入iframe，让它异步执行js

五.延迟加载：有些 js 代码并不是页面初始化的时候就立刻需要的，而稍后的某些情况才需要的。

43、ie各版本和chrome可以并行下载多少个资源

IE6 两个并发，iE7升级之后的6个并发，之后版本也是6个

Firefox，chrome也是6个

44、说说网络分层里七层模型是哪七层

§ 应用层：应用层、表示层、会话层（从上往下）（HTTP、FTP、SMTP、DNS）

§ 传输层（TCP和UDP）

§ 网络层（IP）

§ 物理和数据链路层（以太网）

每一层的作用如下：

物理层：通过媒介传输比特,确定机械及电气规范（比特Bit）

数据链路层：将比特组装成帧和点到点的传递（帧Frame）

网络层：负责数据包从源到宿的传递和网际互连（包PackeT）

传输层：提供端到端的可靠报文传递和错误恢复（段Segment）

会话层：建立、管理和终止会话（会话协议数据单元SPDU）

表示层：对数据进行翻译、加密和压缩（表示协议数据单元PPDU）

应用层：允许访问OSI环境的手段（应用协议数据单元APDU）

***45、ajax和axios、fetch的区别

（一）jQuery ajax

传统 Ajax 指的是 XMLHttpRequest（XHR）， 最早出现的发送后端请求技术，隶属于原始js中，核心使用XMLHttpRequest对象，多个

请求之间如果有先后关系的话，就会出现回调地狱。
JQuery ajax 是对原生XHR的封装，除此以外还增添了对JSONP的支持。经过多年的更新维护，真的已经是非常的方便了，优点无需多言；如果是硬要举出几个缺点，那可能只有：
1.本身是针对MVC的编程,不符合现在前端MVVM的浪潮
2.基于原生的XHR开发，XHR本身的架构不清晰。
3.JQuery整个项目太大，单纯使用ajax却要引入整个JQuery非常的不合理（采取个性化打包的方案又不能享受CDN服务）
4.不符合关注分离（Separation of Concerns）的原则
5.配置和调用方式非常混乱，而且基于事件的异步模型不友好。

```
$.ajax({
 type: 'POST',
   url: url,
   data: data,
   dataType: dataType,
   success: function () {},
   error: function () {}
});
```

二.Axios

Vue2.0之后，尤雨溪推荐大家用axios替换JQuery ajax，想必让axios进入了很多人的目光中。

axios 是一个基于Promise 用于浏览器和 nodejs 的 HTTP 客户端，本质上也是对原生XHR的封装，只不过它是Promise的实现版本，符合最新的ES规范，它本身具有以下特征：
1.从浏览器中创建 XMLHttpRequest
2.支持 Promise API
3.客户端支持防止CSRF
4.提供了一些并发请求的接口（重要，方便了很多的操作）
5.从 node.js 创建 http 请求
6.拦截请求和响应
7.转换请求和响应数据
8.取消请求
9.自动转换JSON数据
PS:防止CSRF:就是让你的每个请求都带一个从cookie中拿到的key,

根据浏览器同源策略，假冒的网站是拿不到你cookie中得key的，这样，后台就可以轻松辨别出这个请求是否是用户在假冒网站上的误导输入，从而采取正确的策略。

```
axios({
    method: 'post',
    url: '/user/12345',
    data: {
        firstName: 'Fred',
        lastName: 'Flintstone'
    }
})
.then(function (response) {
    console.log(response);
})
.catch(function (error) {
    console.log(error);
});
```

三.Fetch

Fetch号称是AJAX的替代品，是在ES6出现的，使用了ES6中的promise对象。Fetch是基于promise设计的。Fetch的代码结构比起ajax简单多了，参数有点像jQuery ajax。但是，一定记住fetch不是ajax的进一步封装，而是原生js，没有使用XMLHttpRequest对象。

```
try {
  let response = await fetch(url);
  let data = response.json();
  console.log(data);
} catch(e) {
  console.log("Oops, error", e);
}
```

fetch的优点:

1.符合关注分离，没有将输入、输出和用事件来跟踪的状态混杂在一个对象里

2.更好更方便的写法

坦白说，上面的理由对我来说完全没有什么说服力，因为不管是Jquery还是Axios都已经帮我们把xhr封装的足够好，使用起来也足

够方便，为什么我们还要花费大力气去学习fetch?
我认为fetch的优势主要优势就是:

1. 语法简洁，更加语义化

2.基于标准 Promise 实现，支持 async/await

3.同构方便，使用 [isomorphic-fetch](https://github.com/matthew-andrews/isomorphic-fetch)

4.更加底层，提供的API丰富（request, response）

5.脱离了XHR，是ES规范里新的实现方式

最近在使用fetch的时候，也遇到了不少的问题:
fetch是一个低层次的API，你可以把它考虑成原生的XHR，所以使用起来并不是那么舒服，需要进行封装。
例如:

1）fetch只对网络请求报错，对400，500都当做成功的请求，服务器返回 400，500 错误码时并不会 reject，只有网络错误这些导致请求不能完成时，fetch 才会被 reject。

2）fetch默认不会带cookie，需要添加配置项： fetch(url, {credentials: 'include'})

3）fetch不支持abort，不支持超时控制，使用setTimeout及Promise.reject的实现的超时控制并不能阻止请求过程继续在后台运行，造成了流量的浪费

4）fetch没有办法原生监测请求的进度，而XHR可以

总结：axios既提供了并发的封装，也没有fetch的各种问题，而且体积也较小，当之无愧现在最应该选用的请求的方式。

***46、vue.Js项目的优化

【http://orangexc.xyz/】

基础优化

所谓的基础优化是任何 web 项目都要做的，并且是问题的根源。HTML，CSS，JS 是第一步要优化的点，分别对应到 vue 文件内的，<template>,<style>,<script>，下面逐个谈下 vue 项目里都有哪些值得优化的点。

（一）template

①注意语义化标签，避免乱嵌套，合理命名属性

②模板部分帮助我们展示结构化数据，vue 通过数据驱动视图，主要注意一下几点

A、v-show，v-if 用哪个？在我来看要分两个维度去思考问题，第一个维度是权限问题，只要涉及到权限相关的展示无疑要用 v-if，第二个维度在没有权限限制下根据用户点击的频次选择，频繁切换的使用 v-show，不频繁切换的使用 v-if，这里要说的优化点在于减少页面中 dom 总数，我比较倾向于**使用 v-if，因为减少了 dom 数量，加快首屏渲染**，至于性能方面我感觉肉眼看不出来切换的渲染过程，**也不会影响用户的体验**。

B、**不要在模板里面写过多的表达式与判断** v-if="isShow && isAdmin && (a || b)"，这种表达式虽说可以识别，但是不是长久之计，当看着不舒服时，适当的写到 methods 和 computed 里面封装成一个方法，这样的好处是方便我们在多处判断相同的表达式，其他权限相同的元素再判断展示的时候调用同一个方法即可。

C、循环调用子组件时添加 key，key 可以唯一标识一个循环个体，可以使用例如 item.id 作为 key，假如数组数据是这样的 ['a'，'b', 'c', 'a'],使用 :key="item" 显然没有意义，更好的办法就是在循环的时候 (item, index) in arr，然后 :key="index"来确保 key 的唯一性。

（二）style

①将样式文件放在 vue 文件内还是外？讨论起来没有意义，重点是按模块划分，我的习惯是放在 vue 文件内部，方便写代码是在同一个文件里跳转上下对照，无论内外建议加上 <style scoped> 将样式文件锁住，目的很简单，再好用的标准也避免不了多人开发的麻烦，约定命名规则也可能会冲突，锁定区域后尽量采用简短的命名规则，不需要 .header-title__text 之类的 class，直接 .title 搞定。

②为了和上一条作区分，说下全局的样式文件，全局的样式文件，尽量抽象化，既然不在每一个组件里重复写，就尽量通用，这部分抽象做的越好说明你的样式文件体积越小，复用率越高。建议将复写组件库如 Element 样式的代码也放到全局中去。

③不使用 float 布局，之前看到很多人封装了 .fl -- float: left 到全局文件里去，然后又要 .clear，现在的浏览器还不至于弱到非要用 float 去兼容，完全可以 flex，grid 兼容性一般，功能其实 flex

布局都可以实现，float 会带来布局上的麻烦，用过的都知道不相信解释坑了。

（三）script

①多人开发时尽量保持每个组件 export default {} 内的方法顺序一致，方便查找对应的方法。我个人习惯 data、props、钩子、watch、computed、components。

②data 里要说的就是初始化数据的结构尽量详细，命名清晰，简单易懂，避免无用的变量，isEditing 实际可以代表两个状态，true 或 false，不要再去定义 notEditing 来控制展示，完全可以在模板里 {{ isEditing ? 编辑中 : 保存 }}

③props 父子组件传值时尽量 :width="" :heigth="" 不要 :option={}，细化的好处是只传需要修改的参数，在子组件 props 里加数据类型，是否必传，以及默认值，便于排查错误，让传值更严谨。

④钩子理解好生命周期的含义就好，什么时间应该请求，什么时间注销方法，哪些方法需要注销。简单易懂，官网都有写。

⑤metheds 中每一个方法一定要简单，只做一件事，尽量封装可复用的简短的方法，参数不易过多。如果十分依赖 lodash 开发，methed 看着会简洁许多，代价就是整体的 bundle 体积会大，假如项目仅仅用到小部分方法可以局部引入 loadsh，不想用 lodash 的话可以自己封装一个 util.js 文件

⑥watch 和 computed 用哪个的问题看官网的例子，计算属性主要是做一层 filter 转换，切忌加一些调用方法进去，watch 的作用就是监听数据变化去改变数据或触发事件如 this.$store.dispatch('update', { ... })

组件优化

vue 的组件化深受大家喜爱，到底组件拆到什么程度算是合理，还要因项目大小而异，小型项目可以简单几个组件搞定，甚至不用 vuex，axios 等等，如果规模较大就要细分组件，越细越好，包括布局的封装，按钮，表单，提示框，轮播等，推荐看下 Element 组件库的代码，没时间写这么详细可以直接用 Element 库，分几点进行优化

组件有明确含义，只处理类似的业务。复用性越高越好，配置性越强越好。

自己封装组件还是遵循配置 props 细化的规则。

组件分类，我习惯性的按照三类划分，page、page-item 和 layout，page 是路由控制的部分，page-item 属于 page 里各个布局块如 banner、side 等等，layout 里放置多个页面至少出现两次的组件，如 icon, scrollTop 等

vue-router 和 vuex 优化

vue-router 除了切换路由，用的最多的是处理权限的逻辑，关于权限的控制这里不赘述，相关 demo 和文章有许多，那么说到优化，值得一提的就是**组件懒加载**

```
const Foo = r => require.ensure([], () =>
r(require('./Foo.vue')), 'group-foo')
const Bar = r => require.ensure([], () =>
r(require('./Bar.vue')), 'group-foo')
const Baz = r => require.ensure([], () =>
r(require('./Baz.vue')), 'group-foo')
```

这段代码将 Foo, Bar, Baz 三个组件打包进了名为 group-foo 的 chunk 文件，当然啦是 js 文件，其余部分正常写就可以，在网站加载时会自动解析需要加载哪个 chunk，虽然分别打包的总体积会变大，但是单看请求首屏速度的话，快了好多。

vuex 面临的问题和解决方案有几点：

当网站足够大时，一个状态树下，根的部分字段繁多，解决这个问题就要模块化 vuex，官①网提供了模块化方案，允许我们在初始化 vuex 的时候配置 modules。每一个 module 里面又分别包含 state 、action 等，看似是多个状态树，其实还是基于 rootState 的子树。细分后整个 state 结构就清晰了，管理起来也方便许多。

②由于 vuex 的灵活性，带来了编码不统一的情况，完整的闭环是 store.dispatch('action') -> action -> commit -> mutation -> getter -> computed，实际上中间的环节有的可以省略，因为 API 文档提供了以下几个方法 mapState、mapGetters、mapActions、mapMutations，然后在组件里可以直接调取任何一步，还是项目小想怎么调用都可以，项目大的时候，就要考虑 vuex 使用的统一性，我的建议是不论多简单的流程都跑完整个闭环，形成代码的统一，

方便后期管理，在我的组件里只允许出现 dispatch 和 mapGetters，其余的流程都在名为 store 的 vuex 文件夹里进行。

③基于上面一条，说下每个过程里面要做什么，前后端数据一定会有不一致的地方，或是数据结构，或是字段命名，那么究竟应该在哪一步处理数据转换的逻辑呢？有人会说其实哪一步都可以实现，其实不然，我的建议如下

A、在发 dispatch 之前就处理好组件内需要传的参数的数据结构和字段名

B、到了 action 允许我们做的事情很多，因为这部支持异步，支持 state, rootState, commit, dispatch, getters，由此可见责任重大，首先如果后端需要部分其他 module 里面的数据，要通过 rootState 取值再整合到原有数据上，下一步发出请求,建议（async await + axios），拿到数据后进行筛选转换，再发送 commit 到 mutation

C、这一步是将转换后的数据更新到 state 里，可能会有数据分发的过程（传进一个 object 改变多个 state 中 key 的 value），可以转换数据结构，但是尽量不做字段转换，在上一步做

D、此时的 store 已经更新，使用 getter 方法来取值，token: state => state.token，单单的取值，尽量不要做数据转换，需要转换的点在于多个地方用相同的字段，但是结构不同的情况（很少出现）。

E、在组件里用 mapGetters 拿到对应的 getter 值。

打包优化

前段时间打包的 vender bundle 足足 1.4M，app bundle 也有 270K，app bundle 可以通过组件懒加载解决，vender 包该怎么解决?

解决方法很简单，打包 vender 时不打包 vue、vuex、vue-router、axios 等，换用国内的 bootcdn 直接引入到根目录的 index.html 中。

例如：

```
<script src="//cdn.bootcss.com/vue/2.2.5/
vue.min.js"></script>
<script src="//cdn.bootcss.com/vue-router/2.3.0/
vue-router.min.js"></script>
<script src="//cdn.bootcss.com/vuex/2.2.1/
vuex.min.js"></script>
```

```
<script src="//cdn.bootcss.com/axios/0.15.3/
axios.min.js"></script>
```

在 webpack 里有个 externals，可以忽略不需要打包的库

```
externals: {
 'vue': 'Vue',
'vue-router': 'VueRouter',
'vuex': 'Vuex',
'axios': 'axios'
}
```

此时的 vender 包会非常小，如果不够小还可以拆分其他的库，此时增加了请求的数量，但是远比加载一个 1.4M 的 bundle 快的多。

***47、什么是对象

现实生活中：万物皆对象，对象是一个具体的事物，一个具体的事物就会有行为和特征。

举例： 一部车，一个手机

车是一类事物，门口停的那辆车才是对象 特征：红色、四个轮子 行为：驾驶、刹车

JavaScript中的对象

JavaScript中的对象其实就是生活中对象的一个抽象

JavaScript的对象是无序属性的集合。

其属性可以包含基本值、对象或函数。对象就是一组没有顺序的值。

我们可以把JavaScript中的对象想象成键值对，其中值可以是数据和函数。

对象的行为和特征 特征---属性 行为---方法

事物的特征在对象中用属性来表示。

事物的行为在对象中用方法来表示。

JavaScript 语言是通过一种叫做 原型（prototype）的方式来实现面向对象编程的

基于原型的面向对象方式中，对象（object）则是依靠 构造器（constructor）利用 原型（prototype）构造出来的。然而原型方式中的构造器 (constructor) 和原型 (prototype) 本身也是其他对象通过原型方式构造出来的对象。

***48、作用域链

在ECMAScript中，只有函数可以制造局部作用域结构。 那么只要是

代码，就至少有一个作用域，即全局作用域。
凡是代码中有函数，那么这个函数就构成另一个作用域。
如果函数中还有函数，那么在这个作用域中就又可以诞生一个作用域。
将这样的所有的作用域列出来，可以有一个结构： 函数内指向函数外的链式结构。就称作作用域链。
或：
当需要从局部函数查找某一属性或方法时，如果当前作用域没有找到，就会上溯到上层作用域查找，直至全局函数，这种组织形式就是作用域链。
全局函数无法查看局部函数的内部细节，但局部函数可以查看其上层的函数细节，直至全局细节。

49、Canvas 与 SVG 的比较

Canvas：

依赖分辨率
不支持事件处理器
弱的文本渲染能力
能够以 .png 或 .jpg 格式保存结果图像
最适合图像密集型的游戏，其中的许多对象会被频繁重绘

SVG：

不依赖分辨率
支持事件处理器
最适合带有大型渲染区域的应用程序（比如谷歌地图）
复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快）
不适合游戏应用

50、<keep-alive></keep-alive>的作用是什么?

<keep-alive></keep-alive> 包裹动态组件时，会缓存不活动的组件实例,主要用于保留组件状态，避免重新渲染。
用自己的理解说
比如有一个列表和一个详情，那么用户就会经常执行打开详情=>返回列表=>打开详情...这样的话列表和详情都是一个频率很高的页面，那么就可以对列表组件使用<keep-alive></keep-alive>进行缓存，这样用户每次返回列表的时候，都能从缓存中快速渲染，而不

是重新渲染，这样就会减轻服务器压力，提高性能。

*****51、vue项目遇到问题：如何解决在加载页面时出现的闪烁问题。**

在项目在开发中，经常会遇到当数据加载时vue的一些标签就会闪现，然后等数据加载完后消失，这时候就需要用到官网中提到的v-cloak来解决。

经常遇到的情况。

①加载时遇到{{value.name}}闪烁，是因为你在渲染时是这么写的 {{value.name}}

②加载时遇到一个空的盒子里边什么也没有，是因为你在渲染时是这么写的

解决：

```
<div class="#app" v-cloak>
    <p>{{value.name}}</p></div>
</div>
```

在css里面要添加

```
[v-cloak] {
   display: none;
}
```

但是有的时候会不起作用，可能的原因有二：

一，v-cloak的display属性被层级更高的给覆盖掉了，所以要提高层级

```
[v-cloak] {
    display: none !important;
}
```

二，样式放在了@import引入的css文件中（传统的开发方式）

v-cloak的这个样式放在@import 引入的css文件中不起作用，可以放在link引入的css文件里或者内联样式中

52、vue-cli 解决ie兼容性问题

首先兼容ie只需要安装一个插件就好了 `npm install babel-polyfill --save-dev`

- 使用方法,打开项目build文件夹下的 `webpack.base.conf.js` 文件，在里边配置如下

```
module.exports = {
  entry: {
    app: ['babel-polyfill', './src/main.js']
  },
}
```

然后再main.js中引入就好了

```
import 'babel-polyfill'
```

- 这样就可以解决兼容性问题了，但是这样并没有兼容axios的请求以及一些api,例如：-`Pormise`
  解决上述问题就需要安装其他的一些插件。

Step一，promise问题？axios不能直接兼容ie

```
npm install es6-promise
```

使用方法：

```
import promise from 'es6-promise';
promise.polyfill();
```

Step二，URLSearchParams未定义的问题，原来是IE9不支持URLSearchParams。

解决办法：下载qs

```
npm install qs
```

然后main.js全局引入 `import qs from 'qs';`

52、document.load和document.ready的区别。

document.load在所有的资源全部加载完成后，执行下一个函数（会有卡顿）。

document.ready只要加载完dom树就执行函数（加载比较快）。

53.事件捕获OR事件冒泡

事件捕获指的是从document到触发事件的那个节点，即自上而下的去触发事件。

相反的，事件冒泡是自下而上的去触发事件

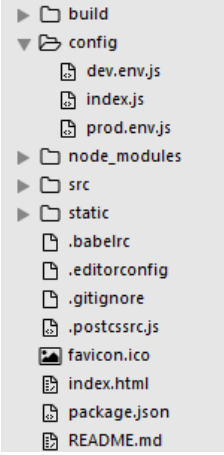
绑定事件方法的第三个参数，就是控制事件触发顺序是否为事件捕

获。
true,事件捕获;
false,事件冒泡。
默认false,即事件冒泡。

54、Vue-cli代理解决跨域问题

在vuejs-templates，也就是vue-cli的使用的模板插件里，有关于API proxy的说明，这个配置就是将localhost映射成访问的域名。

1.config目录找到index.js



2、在dev里添加proxyTable

2.在dev里添加proxyTable

```
dev: {
    env: require('./dev.env'),
    port: 8080,
    autoOpenBrowser: true,
    assetsSubDirectory: 'static',
    assetsPublicPath: '/',
    proxyTable: {
        '/api':{
            target:"http://47.93.166.112/BrainPcWeb",//设置你调用的接口域名和端口号 别忘了加http
            changeOrigin:true,
            pathRewrite:{
                '^/api':''//这里理解成用 '/api' 代替target里面的地址，后面组件中我们掉接口时直接用api代替
                          //比如我要调用'http://40.00.100.133:3002/user/login'，直接写 '/api/user/login' 即可
            }
        }
    }
```

这一步为止，你重新run一下vue已经不存在跨域问题了。

但是注意了

这只是开发环境（dev）中解决了跨域问题，生产环境中真正部署到服务器上如果是非同源还是存在跨域问题

54.JS创建函数的方法

答案：

（1）声明函数：function name(){}

（2）匿名函数：function(){}　首先创建一个变量，变量是一个没有名字的函数

（3）　具名函数：首先创建一个变量，变量时一个有名的函数，var a = function bb(){}

（4）构造函数 new function

（5）立即函数（自动执行函数）

有四种写法：

(function(){})()

(function(){} ())

!function(){}()

void function()

55、谈谈你对This对象的理解?

1、this总是指向函数的直接调用者（而非间接调用者）；

2、如果有new关键字，this指向new出来的那个对象；
3、在事件中，this指向触发这个事件的对象，特殊的是，IE中的attachEvent中的this总是指向全局对象Window

*****56、什么是垃圾回收机制?**

javascript具有自动垃圾回收机制，执行环境会负责管理代码执行过程中使用的内存。原理就是找出那些不再继续使用的变量，然后释放其占有内存。

找出程序中不在使用的变量，然后释放掉其占用的内存，垃圾回收机制会按照固定的时间周期来执行。不在使用的变量是局部变量，（生命周期结束的变量），全局变量的生命周期只有在浏览器页面关闭后才会结束。

垃圾回收机制的好处和坏处

好处：大幅简化程序的内存管理代码，减轻程序猿负担，并且减少因为长时间运转而带来的内存泄露问题。

坏处：自动回收意味着程序猿无法掌控内存。ECMAScript中没有暴露垃圾回收的借口，我们无法强迫其进行垃圾回收，更加无法干预内存管理。

*****57、Javascript原型链?**

答案：回答重点

在调用某个对象的属性或方法时，js引擎会优先在该对象自身上查找该属性或方法，如果没有便去该对象的构造函数中去查找，如此逐级递归的查找，直到原型链的顶端，（向上一级一级查找）

*****58、W3C标准及规范**

万维网联盟标准不是某一个标准，而是一些列标准的集合。网页主要有三部分组成：结构（Structure）、表现（Presentation）、行为（Behavior）。

对应的标准也有三方面：结构化标准主要包括XHTML和XML，表现标准语言主要包括CSS、行为标准主要包括（如W3C DOM）、ECMAScript等。这些标准大部分是W3C起草发布，也有一是其他标准组织制定的标准，比如ECMAScript（European Computer Manufacturers Association）的ECMAScript的标准。

标准规范：

1、 需要声明（DOCTYPE）

2、需要定义语言编码

<meta http-equiv="Content-Type" content="text/html;

charset=gb2312” />
注：如果忘记了定义语言编码，可能会出现页面乱码现象。
3、JavaScript定义
Js必须要用<script language="javascript" type="text/javascript">来开头定义，以保证在不支持js的浏览器上直接显示出来。
4、CSS定义
CSS必须要用<style type=“text/css”>开头来定义，为保证各浏览器的兼容性，在写CSS时请都写上数量单位，例如：错误：.space_10{padding-left:10} 正确：.space_10 {padding-left:10px}
5、使用注释
正确的应用等号或者空格替换内部的虚线。<!--这里是注释==========这里是注释-->
6、所有标签的元素和属性名字都必须使用小写
与HTML不一样，XHTML对大小写是敏感的，<title>和<TITLE>是不同的标签。XHTML要求所有的标签和属性的名字都必须使用小写。
7、所有属性值必须用引号括起来（"" "）双引号或单引号

8、把所有特殊符号用编码表示
空格为 、小于号（<）&lt、大于号（>）&gt、和号（&）&amp等。
9、所有属性必须有属性值
XHTML规定所有属性都必须有个值，没有值就是重复本身。
10、所有的标记都必须有相应的结束标记
双标记： 单标记：<img />
11、所有的标记都必须合理嵌套
必须修改为：
12、图片添加有意义的alt属性
图片加载失败时可以用alt属性表明图片内容。同理添加文字链接的title属性，帮助显示不完整的内容显示完整。
13、在form表单中增加label，以增加用户友好度

```
<form>
   <labelfor="firstname">first name: </label>
   <inputtype="text" id="firstname"/>
</form>
```

59、$route和$router的区别

答：$route是“路由信息对象”，包括path，params，hash，query，fullPath，matched，name等路由信息参数。而$router是“路由实例”对象包括了路由的跳转方法，钩子函数等。

60.vue.js的两个核心是什么?

答：数据驱动、组件系统

61、vue常用的修饰符?

prevent: 提交事件不再重载页面；.stop: 阻止单击事件冒泡；.self: 当事件发生在该元素本身而不是子元素的时候会触发；.capture: 事件侦听，事件发生的时候会调用