**面试造火箭，工作拧螺丝！**

在技术圈毕竟只有百分之一的人能进入BAT，百分之九九的小伙伴只能在普通公司做这普通的事情，厌烦哪些标题党，我们抛开那些高大上的台词，回归到面试的本质。

本课程帮助小伙伴们快速梳理知识，不会设计到具体的很细节的知识点，关注面试本身。

公司一般会从以下5个方面考察一个人的能力，本课程的100问是总结了最近2-3年常问的面试题，适合初中级前端工程师。

# HTML(5)和CSS3方面

1. **前端与后端数据交互的格式有哪些，为什么大部分现在都用json而不用xml。**  
    答：XML：<person><name>知了堂</name><age>3岁</age></person>

JSON：{ name:”知了堂”,age:3}

JSON书写方便节省字节，更轻量，前后台都有直接解析JSON的方法（JSON.stringfity/parse）使用方便。

1. **Flex布局熟悉吗，说几个常用的属性。**

答：这个几乎每天都在用，还是挺熟悉的。

display：flex

align-items 多个

align-content：单个

justify-content

justify-items

flex-direction

flex-wrap：

flex-basic：初始盒子宽度 flex

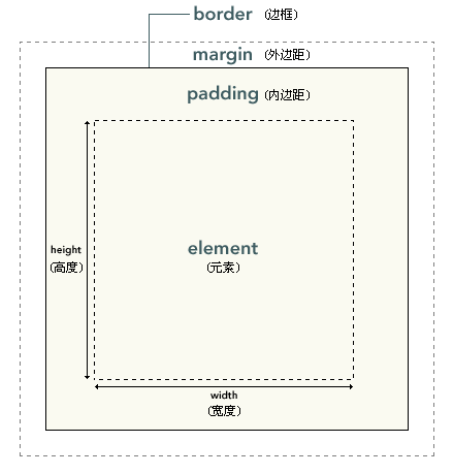
flex-grow：增长因子 200 4\*40=160 1,1,1,2 1/5\*40 flex

flex-shrink :缩减因子 200 60\*4=240

1. **说一下CSS盒模型**

答：CSS的盒模型包含了一下几个内容margin,padding,border,content。

在计算盒子宽高的时候，IE和Chrome会有一些区别，IE算到border，Chrome的宽度只包含content区域，因此CSS3提供了box-sizing这个属性来修改。

****

1. **CSS常用选择器，选择器权重问题。**

答：\*(has，not，target，root。。。。)通配符， ID，class，attr属性，element，子代（ > + ~ ）,

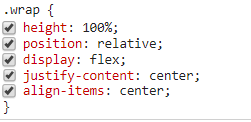
UI状态伪类选择器（hover,active,link,seceted..，checked），

结构性伪类选择器(nth-child,fist-child,last,nth-of-type...before,after....)

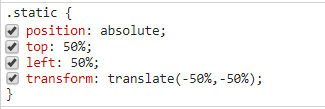
!important > style > id > class > elemnet > 伪类和属性

1. **请用5种方式实现元素垂直居中。**

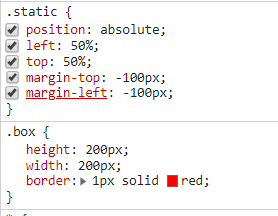
答：1、flex



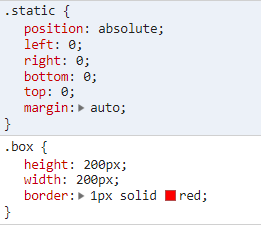
1. Tranform



1. 定位+margin负值（知道子节点宽高）



1. 定位+margin:auto



5、JS动态计算top、left值

1. **什么是BFC？垂直margin重叠是为什么？怎么解决这个问题？**

答：概念：BFC全称Block Formatting Context ，中文意思为块级格式上下文。

通俗的来说：BFC是一个独立的布局环境，我们可以理解为一个箱子（实际上是看不见摸不着的），箱子内部的元素无论如何翻江倒海，都不会影响到外部。转换为BFC的理解则是：BFC中的元素的布局是不受外界的影响（我们往往利用这个特性来消除浮动元素对其非浮动的兄弟元素和其子元素带来的影响。比如清除浮动）并且在一个BFC中，块元素与行元素都会垂直的沿着其父元素的边框排列。

如何触发 BFC

1.浮动元素，float 除 none 以外的值

2.position的值不为static或者relative

3.display不为none

4.overflow 除了 visible 以外的值

BFC的应用

1.解决浮动塌陷问题

2.自适应两栏布局（我们还可以运用BFC可以阻止元素被浮动元素覆盖的特性来实现自适应两栏布局。方法：给没有浮动的元素加overflow：hidden。）

3.解决设置margin值重叠问题。

总结：BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。它规定了元素与其他元素的关系和相互作用。

1. **什么是浮动，有什么作用，有何副作用，以及如何清除浮动？**

答：浮动是使用给节点添加float属性，最初的设计是用来实现文字环绕的。

添加了float的节点脱离文档流，同时触发节点的BFC，让节点往一个方向靠，并排成一行。

当一个父节点的子节点全部浮动，就会造成父节点高度塌陷，

解决的办法首先是可以给父节点主动添加高度值，再者利用只要有一个子节点不浮动原理来添加一个不浮动的节点（通常使用伪元素before，after），再者还可以触发父节点的bfc，常用的定位，或者overflow：hidden。

1. **CSS里面有哪些相对单位？都是相对什么的？**

答：REM,EM,VW,VH等

REM：相对于根节点html的font-size

EM：父节点的font-size

VW：视口的宽度为 100VW，相对于把视口分为100份。

VH：视口高度为100VH，同理

以上单位都可以在移动端做页面适配，但通常使用REM和VW

1. **fixed是相对于谁定位的？如果加上transform会出现问题吗？**

答：fixed定位相对于浏览器视口来定位的

添加上transform以后，fixed定位会失效。如果fixed元素的祖先有transform属性,则fixed元素会相对与这个祖先计算,而不是视口。

1. **为什么不推荐用style内联元素？内联元素有什么缺点？（css文件可以缓存）**

答：首先是style是节点的属性，不能被缓存；代码的可读性和可维护性相对弱一些，特别是多人协作开发的时候。但是如果一个页面的style样式足够少的时候，可以使用style元素，因为一次请求最多携带14K的数据，如果足够小，还可以节约一次请求。

1. **简单描述http与https协议，以及为什么要三次握手？什么是长链接**

答：http与https都是目前主流传输协议。

目前http协议已经发展到2.0阶段，支持长链接，断点续传，cache 缓存策略，多路复用，服务器推送等。

https相对于http更安全，增加了证书SSL加密，端口是433。

客户端SYN=1,Seq = x，服务端接受到后，服务端就知道了，有一个客户端要链接我

，然后服务器就会开启一个TCP socket的端口，然后返回数据给前端也是SYN=1,SEQ=Y,ACK = x+1，客户端接受到后，在发一个seq，和ACK+1.主要是为了防止开启无用的链接，或者网络延迟丢包，服务器无法确定到底客户端有没有收到消息

在使用长连接的情况下，当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，客户端再次访问这个服务器时，会继续使用这一条已经建立的连接。

Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。实现长连接需要客户端和服务端都支持长连接。

为何需要长链接?

长连接可以省去较多的TCP建立和关闭的操作，减少浪费，节约时间。

1. **http常见状态码有哪些？301和302的区别是什么？304是指什么？**

答： 404：找不到资源；

500：服务器内部错误；

200：请求成功，并返回数据；

301：永久重定向；

302： 临时移动，可使用原有URI；

304：资源未修改，可使用缓存；

400：请求语法错误（一般为参数错误）；

403：没有权限访问。。。等

1. **浏览器页面渲染的流程是什么（输入url后页面发生什么）？**

答： 首先dns解析IP，建立tcp链接下载资源，构建dom树，当遇到link标签，会下载并执行解析css（不会阻止dom树的构建）当遇到script标签的是，dom树构建会暂停，下载并执行完js才会继续（defer（下载延迟执行），async（异步下载并执行）） 然后再布局和绘制（layout,paint）最后在 render

1. **什么是reflow与repain？哪些操作会触发reflow，如何避免**

答：reflow：回流，当元素的尺寸、结构或触发某些属性时，浏览器会重新渲染页面，称为回流。

repain：重绘，当元素的样式（布局不发生）发生改变的时候。

以下常见操作都会触发：

浏览器窗口大小改变

元素尺寸、位置、内容发生改变

元素字体大小变化

添加或者删除可见的 dom 元素

激活 CSS 伪类（例如：:hover）等

尽量减少DOM操作。

1. **HTML5常用的API有哪些？你用过哪些？**

答：video、audio，获取dom的方式（queryselector），websocket

Canvas，svg，requestAnimationFrame，Geolocation，stroage ，notification，Orientation API用于检测手机的摆放方向等

1. **请列举出几个常见的浏览器兼容性问题？**

答：现在市面上IE678基本已经停止使用了，所以尽量不要说这方面的兼容性。

1、不同浏览器的默认margin和padding不一致

1. 图片的默认间距不一致
2. 获取视口的宽高window.innerheight/width
3. CSS3的动画，过渡，渐变，flex也有
4. Canvas，SVG
5. IE9以下不能的opacity，使用filter: alpha(opacity = 50);
6. event.offsetX/Y
7. 绑定事件IE9才支持
8. **什么是浏览器缓存（知道什么是 强缓存 和 协商缓存）？**

答：当浏览器访问过后的资源，会被浏览器缓存的本地，当下次在访问页面的时候，如果没有过期，直接读取缓存，加快浏览器的加载效率。

http缓存机制：1、Expires：通过设置最大缓存时间，当时间超过了就去服务器下载，

1. http1.1，cache-control：max-age = time ，当time过期后，检测etag 带上etag往服务器发请求，如果etag没变，直接告诉浏览器读本地缓存，如果没有etag 就会 检测 Last-Modified，判断 如果 上一次更改的时候，距离本次访问时间比较久，说明文件没有发生改变，返回304。

强缓存就是当前访问时间还在设置的最大时间范围内。

协商缓存就是时间过了，通过检查etag或者last-modifed来使用缓存的机制。

1. **说一下浏览器垃圾回收机制**

答： 老：标记清除算法，GC会检测当前对象有没有被变量所引用，如果没有就回收。

新： Scavenge ，把内存空间分为两部分，分别为 From 空间和 To 空间。当一个空间满了以后，会把空间中活动对象转移到另外一个空间，这样互换。

# 2、JS方面(ES6/ES7)

1. **在JS中什么是面向对象程序设计，面向对象设计优点?**

答：在JS中面向对象更多说的是通过构造函数或者class封装一个独立的功能，以达到代码的复用。

面向对象的三个特点：

封装：通过对象把属性和方法封装起来，相似对象的话采用构造函数或者类new得到。

继承：通过混合继承（构造函数和原型）的方式，可以达到属性和方法的复用。

多态：通过对象属性覆盖，可以让继承的对象拥有更多行为。

面向对象设计更多的是组织代码的方式，能提升开发效率与代码的可维护性。

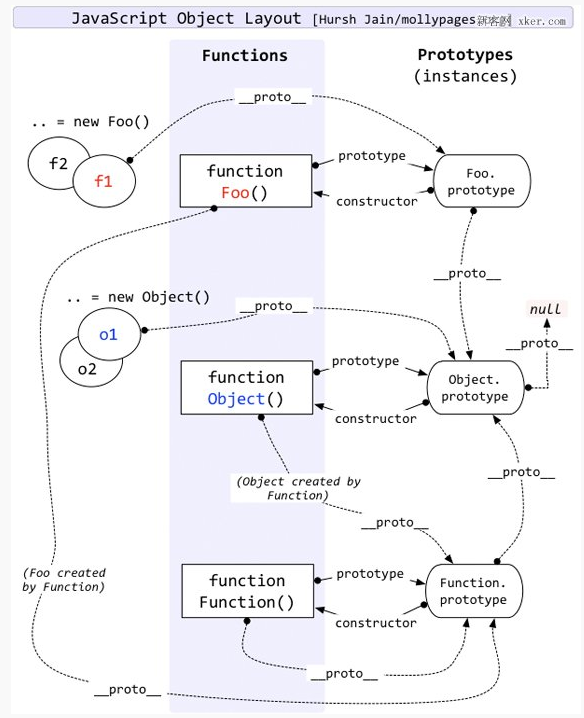
1. **什么是原型、原型链，有什么作用？**

答： 原型：每一个对象都与另一个对象相关联，那个关联的对象就称为原型。

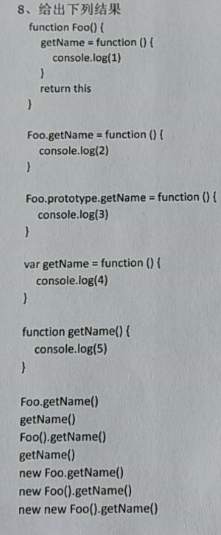
例如：函数Person有一个属性prototype，指向一个对象，对于普通函数来说没多意义，对于构造函数就有作用了，当使用new操作符时，会把Person.prototype（原型对象）赋值给实例的\_\_proto\_\_（原型实例）属性。

JS有一个原型查找机制，把原来定义在实例上的方法，放到原型对象上去，通过构造函数的new操作，会把原型对象赋值给实例的\_\_proto\_\_属性，那么当使用返回的实例去调用某一个方法的时候，如果实例本身上没有，就去自动去实例的\_\_proto\_\_上去查找，这样达到方法的复用，减少内存开销。

原型链：每一个对象，都有一个原型对象与之关联，这个原型对象它也是一个普通对象，这个普通对象也有自己的原型对象，这样层层递进，就形成了一个链条，这个链条就是原型链。通过原型链可以实现JS的继承，把父类的原型对象赋值给子类的原型，这样子类实例就可以访问父类原型上的方法了。

****

|  |
| --- |
| 1. Person.prototype.constructor == Person 2. Person.prototype.\_\_proto\_\_.constructor == Object 3. Person.prototype.\_\_proto\_\_ == Object.prototype 5. Object.prototype.\_\_proto\_\_ == null |



1. **如何实现继承（ES5/ES6）？**

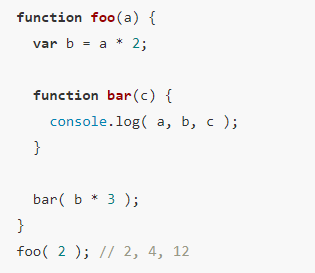
**答：**

|  |
| --- |
| 1. function Anima(name){ //父类 2. this.name = name; 3. } 4. Anima.prototype.sayName = function(){ 5. console.log(this.name) 6. } 7. function Person(name,age){ //子类 8. Anima.call(this,name); 9. this.age = age; 10. } 11. Person.prototype = Object.create(Anima.prototype, { 12. constructor: { 13. value: Person, 14. enumerable: false 15. } 16. }) 17. var p = new Persion("haha",12); 18. Extends //ES6 |
|  |

1. **什么是作用域以及作用域链？** 答：作用域是指程序源代码中定义变量的区域，限定一个变量可访问的范围的，作用域的本质是对象。在JS采用的词法作用域，在书写代码的时候（定义函数的时候），作用域就已经确定好了。

在ES6环境下，包含3个作用域，全局globel，函数作用域，快级作用域( {} ) （eval）

作用域链：由多级作用域对象，逐级引用的链式结构。本质为执行上下文的scope属性，存储所有的变量，包括局部与全局，控制变量的使用顺序。



|  |
| --- |
| var b = 10; //1：输出？2：改造代码输出10或20  (function b(){  b = 20;  console.log(b);  })();  var b = 10;  (function b(b) { 在这个函数b是一个常量，在函数b内部是可以使用的，但是不能修改，如果加上use strict 严格模式就会报错。  window.b = 20;  console.log(b) //输出10  })(b) |

|  |
| --- |
| var a = 10;  (function () {  console.log(a)  a = 5  console.log(window.a)  var a = 20;  console.log(a)  })() |

1. **什么是闭包，闭包的好处和坏处分别是?**

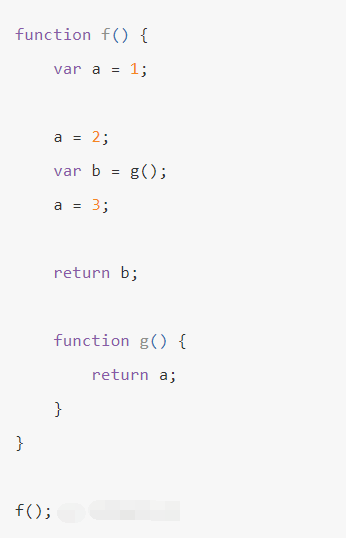
答：当函数可以记住并访问外部作用域时，就产生了闭包，那个外部作用域就称为闭包。

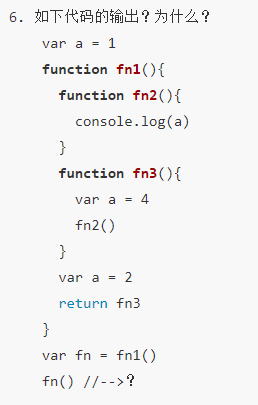
形成的原因：外层函数的作用域对象无法释放。

作用：保护一个变量，重用一个变量。坏处：使用不当，会造成内存泄漏。

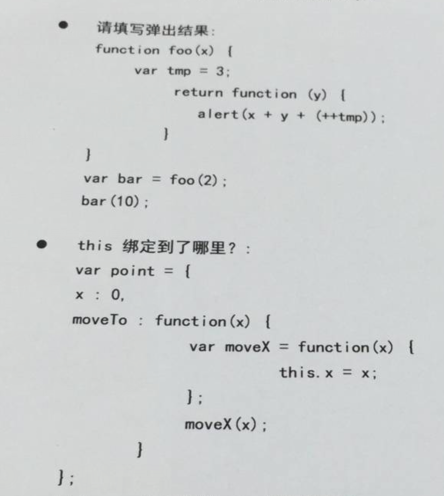
大白话来解释：函数A和函数B，当内部函数B引用了A 的局部变量时，函数A 称为闭包

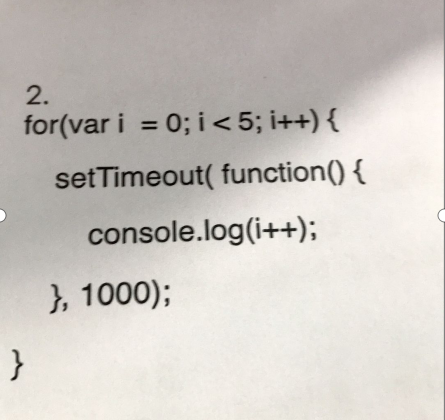
原因是：JS是词法作用域，B的作用域链上有对A执行环境的引用（这个执行环境用函数来表示），A的执行环境AO就不会回收。





|  |
| --- |
| for (var i = 0; i< 5; i++){ //改造代码，每间隔一秒 输出 0-4  setTimeout(() => {  console.log(i);  }, 1000)  } |





1. **什么是this，this的常用方式有哪些？如何改变this的指向？**

答：JS在运行过程中会产生执行上下文环境（context），context记录了包含函数在哪里被调用，作用域scope，this等信息。this就是context的其中一个属性，会在函数的执行过程中使用。This是在运行时候绑定的，它指代的上下文对象取决于函数调用的各种条件。

this提供了一种优雅的方式来隐式的传递一个对象的引用，所以在函数中使用this可以更加方便的复用函数。

1、全局使用 this === window 很少使用

2、函数当中 在全局调用这个fn() this === window

3、在方法当中使用 this === 调用当前这个函数的所在的对象啊

4、构造函数this执向的是 new 创建出来的实例对象啊

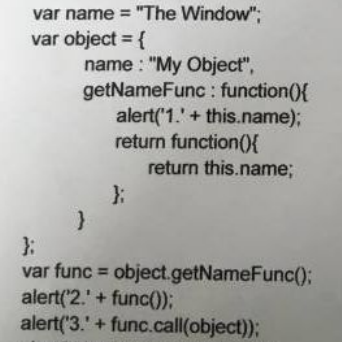
5、DOM事件处理函数中的this，指向当前的DOM节点

6、通过 bind，call，apply 操作符来显示的设置 this的指向

bind：绑定函数里面的this，返回新函数，

call，apply：绑定并执行这个函数，前者传参是“，”隔开，后者是数组

7、ES6的箭头函数 箭头函数没有自己的this，父作用域的this



|  |
| --- |
| var x = 3;    var foo = {       x: 2,       baz: {       x: 1,       bar: function() {       return this.x;       }       }     }    var go = foo.baz.bar;  go()?  foo.baz.bar()? |

|  |
| --- |
| //综合面试题  function Foo() {  Foo.a = function() {  console.log(1)  }  this.a = function() {  console.log(2)  }  }  Foo.prototype.a = function() {  console.log(3)  }  Foo.a = function() {  console.log(4)  }  Foo.a();  let obj = new Foo();  obj.a();  Foo.a(); |

1. **手写bind，call，apply函数**

答案：以上都是Function原型上的方法。

|  |
| --- |
| Function.prototype.myCall = function(context,...args){//绑定并执行  //执行函数  var fn = this;  context.fn = fn;  context.fn(...args);  delete context.fn  }  Function.prototype.myBind = function(context){//绑定返回新的执行函数  //判断调用者是不是函数  if(typeof this != 'function'){  throw new Error("Error")  }  //截取传递的参数  let args = [...arguments].slice(1);  var \_this = this;//保存一下当前的调用者  return function F() {  return \_this.apply(context,args.concat([...arguments]))  }  } |

1. **什么是深、浅拷贝，请写出代码**

答：浅拷贝，就是复制一个对象，当对象的属性值没有引用类型的时候。

Object.assign 迭代（for..in for..of object.enteries） 扩展运算符等

反之如果对象中还有引用类型，连着引用类型一并拷贝称为深拷贝。

JSON对象的方法（会忽略到值为函数和null的属性），递归

|  |
| --- |
| function deepCopy(obj){  //判断对象的类型  var newObj = Array.isArray(obj)?[]:{};  if(obj && typeof obj == "object"){  //迭代  for(var key in obj){  if(typeof obj[key] == 'obj'){  newObj[key] = deepCopy(obj[key])  }else{  newObj[key] = obj[key]  }  }  }  return newObj  } |

1. **什么是Ajax，如何封装一个Ajax？Get请求与 Post请求的区别？**

答**：**Ajax的全称是异步的js与xml技术，通过它与后台服务器进行数据交换，可以使网页实现异步更新，言外之意是在不重新加载整个页面的情况下，对网页进行局部更新。

1.nex xmlhttprequset对象

2.open(method，url，false)

3.绑定redaystatechange事件

4.调用send方法，如果是post请求，可以传递参数

前端的请求方式除了常用Get和Post，还有update，delete，put等（restful api设计）

GET 请求可被缓存，保留在浏览器历史记录中 ，请求的参数是直接跟在URL上，因此不应传递敏感数据。

GET 请求有长度限制(2048字符),IE和Safari浏览器限制2k;Opera限制4k;Firefox，Chrome限制8k 。

GET 请求通常只应当用于从后台获取数据。

POST 请求不会被缓存，不会保留在浏览器历史记录中

POST 请求对数据长度没有要求。

POST 请求通常用于往后台提交数据。

1. **说一些ES6、ES7新特性。**

答： let，count快级作用域；解构，从对象和数组中提取值；箭头函数；字符串模版；扩展运算符...；对象的简写；module；promise（async）；class；对原生对象的扩展（新增加了很多方法） ；for-of （Object.keys,values,entries等）；Symbal()；

不常用的proxy，reflect，generate函数

1. **什么是Promise对象，如何使用？**

答：Promise是解决异步回调的ES语法的标准，通常用来解决异步嵌套和多异步同时完成回调等问题。

Promise是一个构造函数，相当于一个容器，把异步代码包裹在里面，promise有三个状态（pending（进行中）、fulfilled（已成功）和rejected（已失败））初始化为pending，当异步请求成功后调用resolve函数，状态从pending--->fulfilled，失败的时候调用reject，状态从pending--->rejected。状态不可逆。

缺点：书写麻烦，不能实现异步代码，同步执行的需求（配合async函数使用即可）

1. **什么是跨域，解决跨域常用的方式有哪些？**

答：跨域是浏览器端行为，根据同源策略，当请求的协议、域名、端口只有一个不同，就会跨域，跨域是浏览器为了安全存在的机制，浏览器会把跨域请求的数据去掉，同行报错。

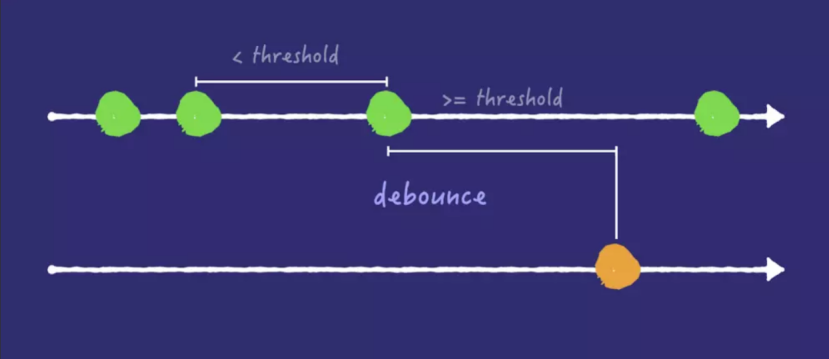
在实际开发难免会出现跨域的情况，解决方案通常有

1. JSONP技术，利用了script的src属性没有跨域限制
2. CORS，当在相应头信息中添加access-control-allow-origain属性，浏览器读取到就会允许返回数据。后台配置，或者下一个浏览器插件即可。
3. 后台代理（Node）
4. Iframe域的提升（很少）

1. **什么是函数的防抖、节流，并手写基本代码**

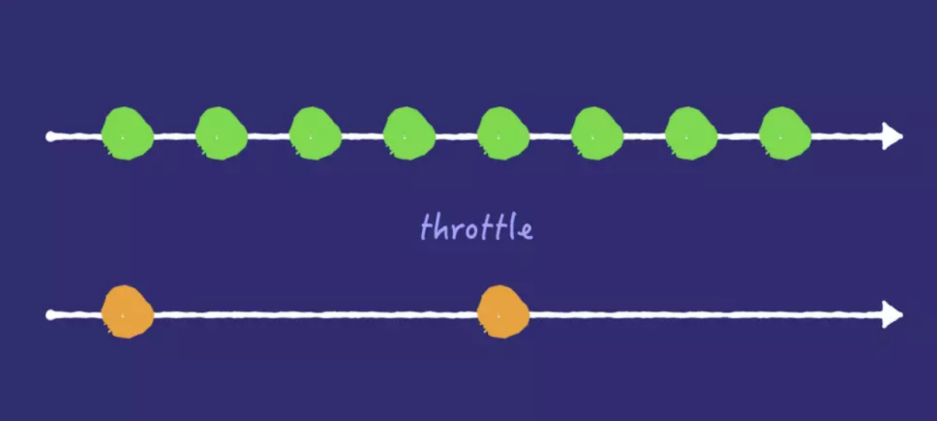
答：防抖和节流都是为了提升运行效率，减少函数的执行次数。

防抖:把多次函数执行，合并成一次执行

****

|  |
| --- |
| function debounce(fn,delay){ //fn真正执行的函数，delay间隔时间  var timer = null;  return function(){  var args = arguments;  var that = this;  if(timer) clearTimeout(timer)  timer = setTimeout(function(){  fn.apply(that,[...arguments])  },delay)  }  } |

节流 ：规定一个单位时间，在单位时间内触发一个事件回调,触发时超过间隔时间则执行，否则不执行。



|  |
| --- |
| function throttle(fn,gapTime){  let \_lastTime = null;  return function(){  var that = this;  var args = arguments;  let \_nowTime = + new Date();  if(\_nowTime-\_lastTime > gapTime || !\_lastTime){  fn.apply(that,args);  \_lastTime = \_nowTime;  }  }  } |

1. **什么事件循环（Event Loop）？**

答：js是非阻塞单线程语言，js在执行过程中会产生执行环境，执行环境会被顺序的加入到执行栈，当遇到异步任务，会添加到task队列当中，执行同步栈，当同步栈执行完以后，event loop 就会从异步task队列当中提取要执行的代码放到执行栈中。这个一个过程称为事件循环。所以JS的异步还是同步过程。

|  |
| --- |
| console.log('script start')  setTimeout(function() {  console.log('setTimeout')  }, 0)  new Promise(resolve => {  console.log('Promise')  resolve()  })  .then(function() {  console.log('promise1')  })  .then(function() {  console.log('promise2')  })  console.log('script end')  //输出结果？ |

1. **前端安全你有了解吗？什么是XSS攻击和CSRF 跨站请求伪造？怎么预防？**

答：前端的安全问题，在工作过程中会注意这些问题。

1、XSS 代码注入问题，通常不信任用户的输入，转义输入输出内容（）encodeURIComponent），括号 尖括号等。利用用户对站点的信任

2、CSRF 跨站请求伪造

是一种挟制用户在当前已登录的web应用中执行非本意的攻击

大白话：利用用户登录态发起恶意请求

网站对用户的信任

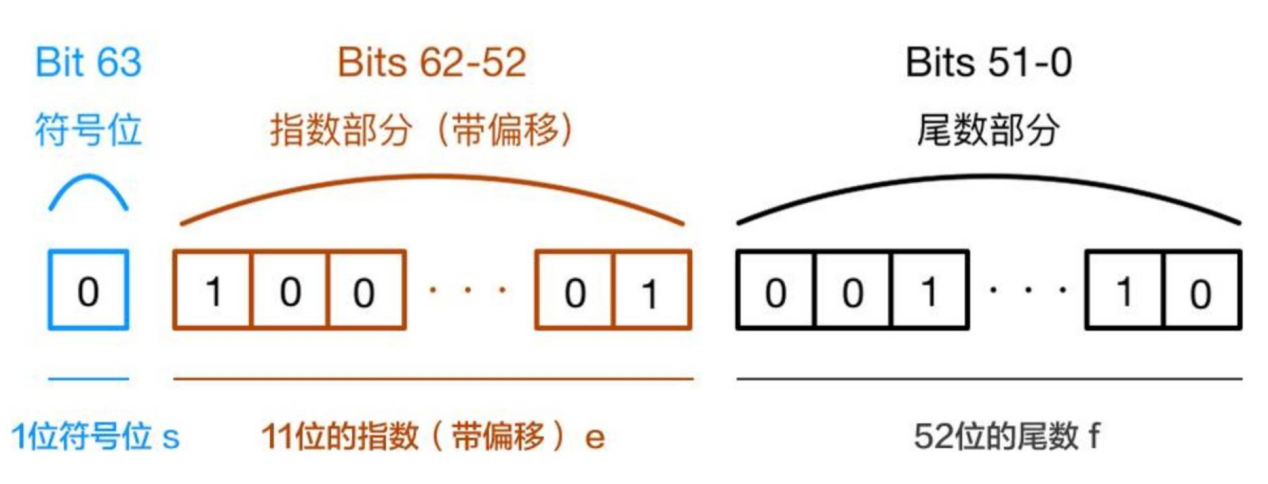
添加验证码（体验稍微差一些），不让第三方访问cookie 对cookie设置 samesite， 请求验证 加 token ，

1. 密码问题：对密码进行加密（MD5等）



1. **如何处理精度丢失问题？**

答：产生的原因：JS使用Number类型表示一个数字，使用64位。



计算机先把10进制转位2进制，0.1和0.2转成二进制的时候还会无限循环，由于有51位位数的限制，多余的会被裁掉，所以在进制之间转换就会丢掉精度。

通常可以把小数乘以倍数转成正整数，计算完后再转回去。

parseFloat((0.1 + 0.2).toFixed(10))

1. **前端常用请求数据的方式有哪些，并说出各自的区别。**

答：前端进行网络请求常用的form，Ajax，fetch

常用的库文件Jquery或者axios等。

Form提交数据页面会刷新，使用体验不佳。

Ajax是一个综合技术，可以实现异步更新页面内容。使用起来方便，通常使用JQ的封装，会额外的JS加载代价。

Fetch是新的API，比较底层。目前浏览器兼容性问题多一些，比如老版本不会携带cookie，不能直接传递JS对象作为参数，不支持JSONP等

1. **如何检测一个对象一定是数组？**

答：1、Object.prototype.toString.call()，前提是toString方式没有被重写。

2、instanceof 内部机制是通过判断对象的原型链中是不是能找到类型的 prototype。

[] **instanceof** Object; // true

3、Array.isArray() ES5新增的方法，兼容性稍微差一些。

1. **JS加载会阻塞页面渲染吗？会的话该怎么解决呢？**

答：会

defer：并行下载，在页面解析完后执行，会按照script的顺序执行。**（常用）**

async：异步下载代码，下载完毕后离职执行代码，不会按照页面的script顺序。

1. **常见前端性能优化有哪些？**

答：

1：加载优化

CDN，文件压缩混淆合并，按需加载（异步组件），前端缓存，浏览器缓存（cache-control,exprices）

DNS预先解析，服务器渲染 nuxt，next（SEO），图片懒加载

2：脚本优化

减少合并DOM操作（createDocumentFragment），CSS3代替js动画，缓存变量，用requestAnimationFrame代理setTimeout,减少reflow与repain，事件代理，开启GPU渲染tranfrom:translateX(0)

3：图片优化

雪碧图（减少请求），使用SVG和iconfont（字体图片）代替图片，图片压缩

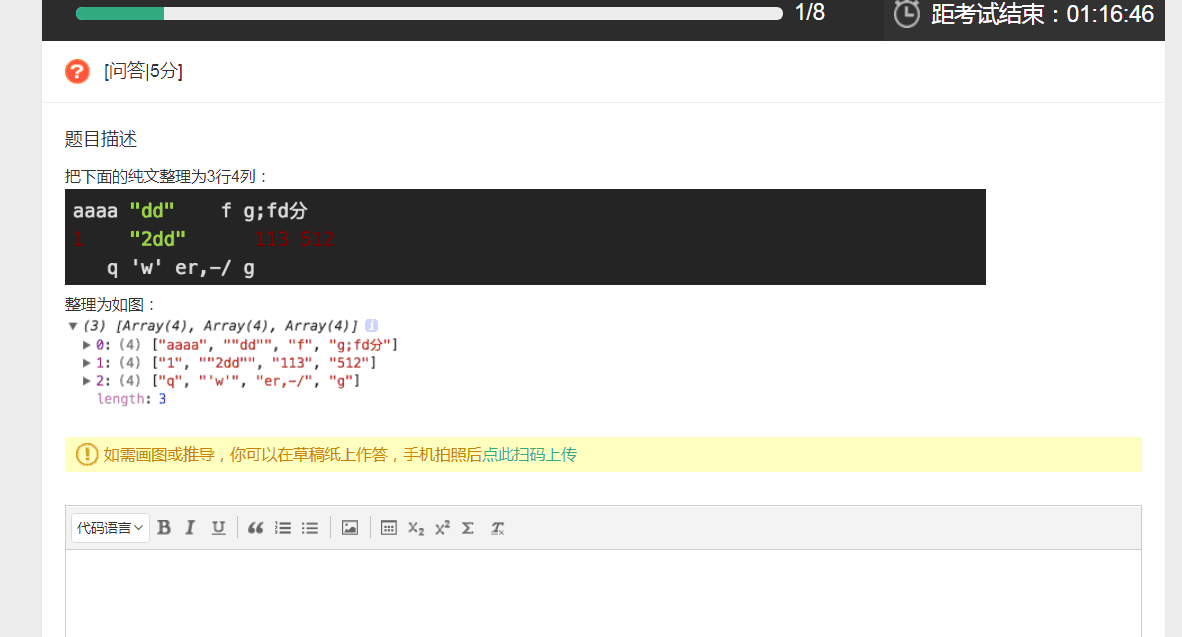
1. **正则表达式**

答：正则表达式在面试中出现的概率很高





|  |
| --- |
| String.prototype.trim = function(){  return this.replace(/(^\s\*)|(\s\*$)/g,'')  } |



|  |
| --- |
| **reg = /[^\s]+/g; //匹配非空格正则**  **let resultarr = [];**  **str.split(/\n/).forEach((item,index)=>{**  **return resultarr.push(item.match(reg));**  **})** |

1. **常用的前端缓存数据（不是浏览器缓存）方式有哪些？说出他们的优缺点。**

答： URL：存储有限，数据直接放到url上面，不能放敏感信息

Cookie：存储有限，每次请求都会带上cookie，会造成请求资源浪费

DOM节点：H5标准通过data-xx为节点添加自定义属性，存储有限，DOM操作的时候非常方便。

本地存储（storage）：存储5M左右，可以永久存储和会话存储，api方便，只能存字符串。

indexDB：前端数据库，键值对的方式存储，可以做离线应用，学习成本高，使用场景少。

1. **谈谈你对前端模块化理解？简单描述一下AMD,CMD,ES6（module）之间的区别**

答: 前端的模块化是把一个js文件看成一个模块，然后对外暴露方法和属性。

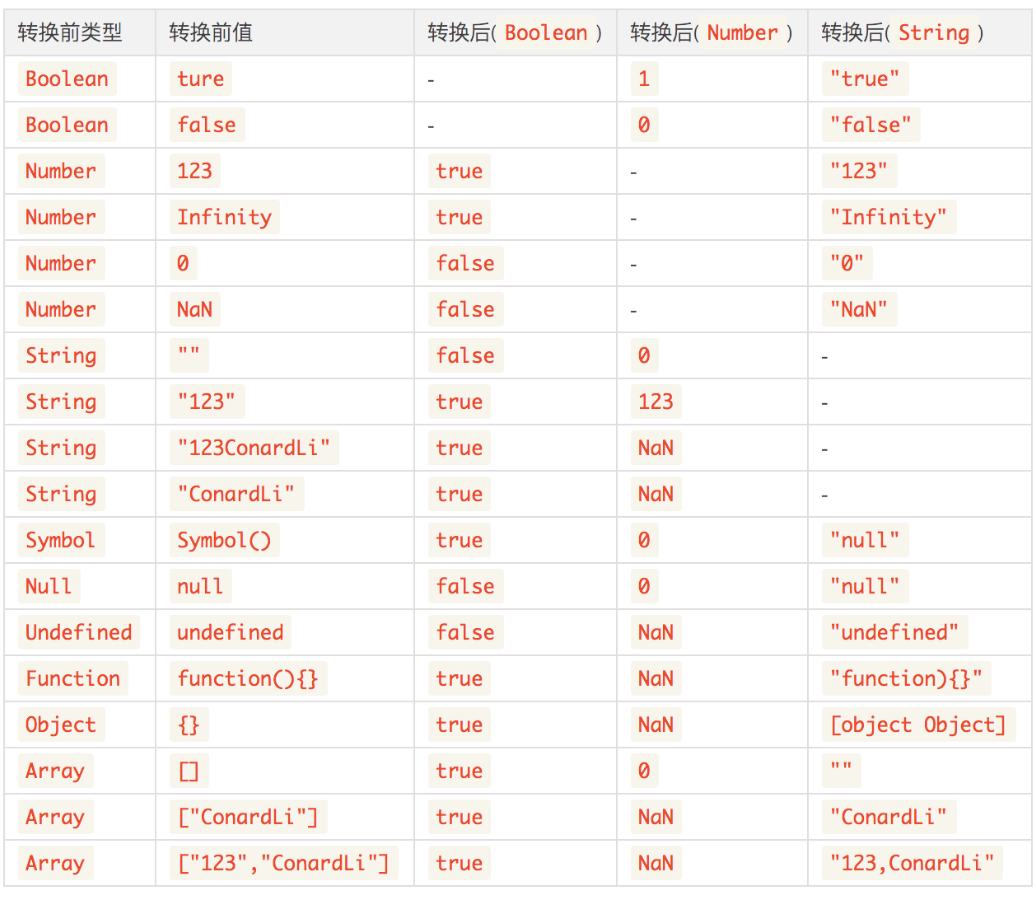
在早些年浏览器不支持模块化，社区有AMD，CMD两个标准，把代码封装在一个函数内部，通过请求的方式去动态的加载js文件。

AMD与CMD两者的区别为，前者为异步预先加载代码并执行；后者为异步加载，当使用到模块提供的函数的时候，在执行（延迟执行）。

ES6以后开始JS开始支持模块化，之前的模块化使用就很少了。

1. **==判断的转化规则？**

答: 当左右两边数据类型不相同的情况下，对象和字符串比较，是对象转字符串；undefined==null；其他情况都是把值转成数字，而对象转成数字要先经过toString转成字符串，再转成数字。



|  |
| --- |
| 1. if(a==1 && a==2 && a==3){ 2. console.log(ok) 3. }   //思路一，为把a定义为对象{}，当对象与数字比较的时候，会先调用toString方法，我们给当前的对象添加一个私有的tostring方法即可，就不会去调用原型上的方法了。  //思路二，使用属性劫持  Let n = 0  Object.defineProperty(window,a,{  get(){  return ++n  }  })  var a = {  n:0,  toString(){  return ++this.n  }  } |

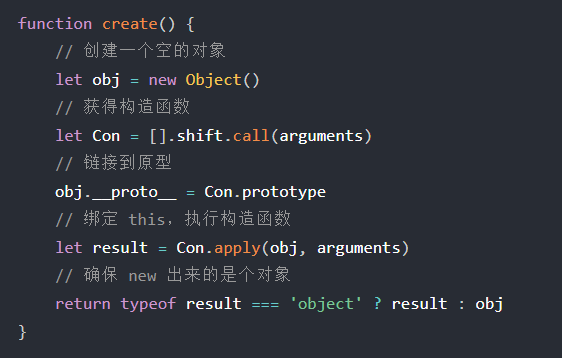
1. **通过new操作符调用构造函数，会经历哪些阶段？**

答：1、创建一个新的对象；

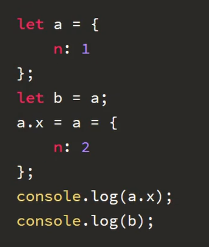
2、链接到原型；

3、将构造函数的this指向这个新对象，执行构造函数的代码，为这个对象添加属性，方法等；

4、返回新对象。



1. **请写出以下代码的输出，并解释**



# 3、框架方面

**1.什么是vdom（虚拟DOM）？为何要使用vdom？简单的描述一下什么是Diff算法？**

答：使用JS对象来模拟DOM结构，将DOM变化的对比放到JS层来做，提高效率。

Vdom是一类技术栈，（snabbdom）

例如：真实DOM结构



模拟结构



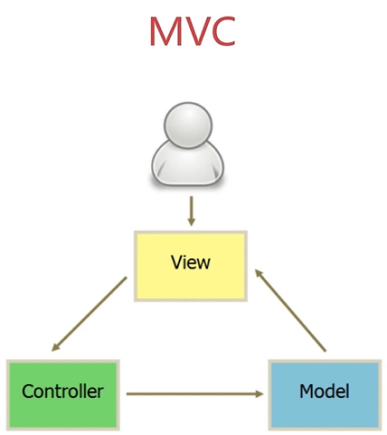
浏览器最费事操作就是DOM操作，DOM结构复杂，属性太多。节点与节点之间还相互关联。

Diff算法的存在是对比新旧两个虚拟DOM节点的区别，只修改变化的地方，其他节点不动。

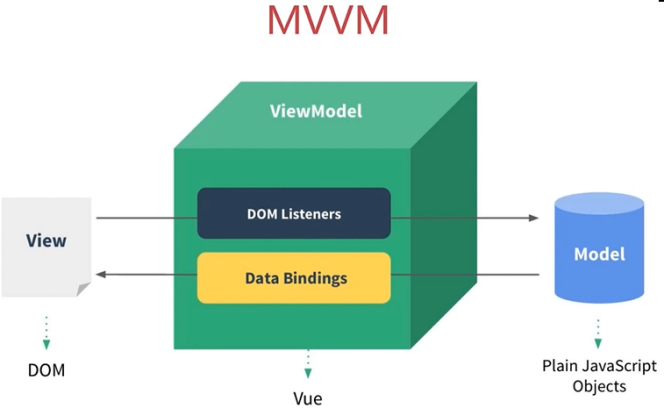
大概实现的思路为遍历新节点，与旧节点同级比较，如果节点相同，再比较属性值，再递归比较子节点，直到全部比较完，再进行下一次同级比较。当遇到不相同的地方记录下来，下一次事件循环的时候统一更新节点。

1. **简单说一下对MVC、MVVM的理解？**

答：1、M：model数据源，一个列表的数据、表单数据等；V：view视图，HTML+CSS；C：controllor控制器，控制视图或者数据的变化。



2、MVVM，在MVC模式上算是一个微创新，M和V和上面一样的，VM：viewmodel，相对于M/V之间的一个桥，view通过事件绑定影响到model，model可以通过数据监听来影响视图，这样就让View和Model分离了更加利于视图的复用。



1. **简单说一下对Vue模板的理解**

答：模版，字符串，有逻辑，可以嵌入JS变量。模版会被编译成render函数，执行render后返回的是vdom（vnode）





Vue的vdom是借鉴了snabbdom，在updateComponent中实现vdom的patch，首次渲染是会执行updateCompoent，在data每次修改的时候，执行updateComponent

1. **Vue如何实现数据的双向绑定的？**

答：Vue通过数据劫持（Object.defineProperty()）+ 观察者模式

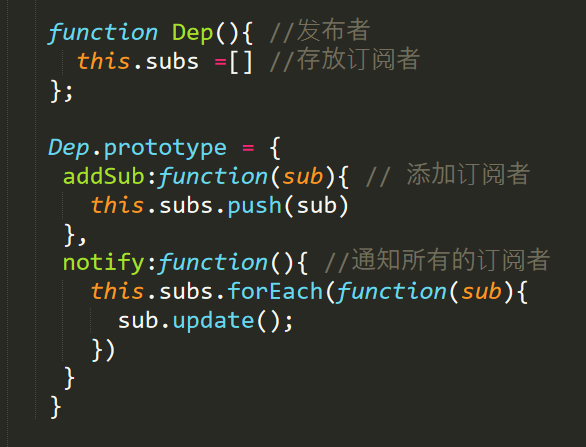
核心代码：



参数obj为当前的组件实例，key：val为data对象里面的每一个键值对。

每来监听一个属性就创建一个属于当前属性的发布者对象。







获取到页面表单上的v-model属性，拿到属性值val，为节点绑定input事件，同时把data中属性为val的属性值赋值给当前的表单的value，当前{{}}节点的时候，就创建一个观察者，在把这个观察者添加到当前val属性的发布者上。

1. **Vue，React子父组件如何通讯？**

答： 由于单项数据流的规范，Vue和react都遵循通讯机制。

vue和react都是使用prop传递数据给子组件，vue需要提前在子组件中提前使用props声明，react不用。

React中子组件传递数据到父组件（状态提升），同样是通过父组件给子组件传递属性，区别是属性值为一个函数函数引用（需要绑定函数的this），然后在子组件中调用这个函数即可，传递相应的参数。

Vue中子到父，是通过在使用子组件的时候绑定自定义事件，在子组件内部拿到事件名称，再通过this.$emit(‘事件名’)触发并传递参数。

其他方式：全局$bus，provide/inject，$refs/$parent/$children ，$attrs/$listeners ，

仓库，storage，url

1.x还有brodercast/dispatch 广播/派发已过时

1. **Vue和React的生命周期包含哪些？，并说出全部钩子函数与使用场景。**

答：详见文档

1. **简述Vue和React的框架的特点以及区别**

答： Vue（MVVM）和react都是组件化、数据驱动型的框架，现在基本都差不多了。

区别：

体积：vue独立一个 30K ，react 160K

学习成本：vue是面向模版编程易学，react函数式编程，需要提前学习ES6+。

效率：vue初始化渲染是比较慢的，是因为vue要把所有的属性都使用object.defineproperty劫持，react就不需要。但是在运行时，vue要快一点，只有数据发生变化就会执行set，再通知指令去更新视图，而react需要执行setState函数，进行diff的比较。

还可以加一些自己的理解:

模板语法上，我更加倾向于 JSX，因为它更接近js 语法（列如vue的循环用的是新指令v-for,而react用的是js中的map()函数）

模板分离上，我更加倾向于 vue（数据和视图分离的更彻底）

组件化上，我更加倾向于 React ，做的彻底

国内使用，首推 vue 。文档更易读、易学、社区够大 。 如果团队水平较高，有native的需求，推荐使用 React 。

1. **什么是单向数据流？谈谈对它的理解**

答：数据通常按照一个方向来传递，通常从父传递子组件，子组件要修改父组件数据通知父组件自己修改，目的便于状态的管理和解耦。

如果不加以限制，任何子组件内部直接修改父组件数据就会影响到其他引用相同数据的组件的显示，从而造成数据混乱。

大白话，数据是谁的，谁才有资格去修改。

1. **谈谈对VueX的理解（或者Redux）**

答：vuex是vue的一个插件，用来做状态管理的，

其实我们可以把 vuex（Redux）看做是一个仓库，我们把需要共享的数据统一存在仓库里，谁需要，直接取就可以。然后提供一个统一修改机制（提交mutation）来修改仓库里的数据。

仓库解决了数据传递层级和同级组件数据传递问题

State：存放状态（数据）的地方

Getters：派生状态的地方

Mutation：修改状态的地方

Action：做异步操作，并提交mutation修改状态的地方

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，主要包括三个核心方法，action，store，reducer

Store是整个仓库对象，存放了单一状态数（state）

action 是一个包含 { type, payload } 的对象

reducer 函数通过 store.dispatch(action) 触发

reducer 函数接受 (state, action) 两个参数，返回一个新的 state

reducer 函数判断 action.type 然后处理对应的 action.payload 数据来更新状态树

1. **Vue-router路由传参的方式有哪些？**

答： 1：js导航的xx.push({path:”/user”,query:{ }})

2：动态路由 ｛path：”/user/:id” name：”user”｝

3：mate

4：Storage/仓库

1. **不用vue-cli如何搭建一个vue项目？**

答：需要使用构建工具webpack（当然还有其他工具gulp，browserify），通过entry配置入口文件；output配置打包输出；module配置loader（loader是用来处理文件的），一般配置处理less/sass，二进制图片或字体等，babel-loader处理ES6/7等，还需要配置vue-loader处理.vue文件；通过plugins配置插件，常用的插件有（**见下面**）；还需要通过devServer（端口，代理，historyApiFallback，刷新浏览器等）配置开发服务器，配合使用webpack-dev-server。还有一些杂项可以通过resolve来配置（别名，去掉扩展名等）.

# 4、工具方面

**1.使用过哪些前端工具？**

答：了解过gulp，但是好像现在已经过时了。

目前在开发工程中都使用webpack.

1. **有了解过webpack吗？常用的配置项有哪些？**

答：在开发中会使用到webpack的

常见的配置包括：

mode：配置webpack环境，包含developer和production环境 （4+）

entry：配置项目的入口文件

output：打包后的文件名，存放的地址，chunk的名字。。。

module：loader test use指定loader

plugins：插件

devserver：配置webpack-dev-server

resolve：杂项，别名，文件的扩展名。。。

devtools：soure-map

**3.webpack常用的插件有哪些？分别的作用什么？**

答：

MiniCssExtractPlugin ：抽离css从js中

DefinePlugin：定义全局变量

htmlWebpackPlugin 设置模版 会自动帮我们打包好的文件路径注入模版

PurifyCSSPlugin：去掉没用的css样式

CopyWebpackPlugin：拷贝静态资源

HotModuleReplacementPlugin：组件热更新

# 5、常规算法方面

**1、JS中数据解构与算法**

**答：**

1. 栈 LIFO
2. 队列
3. 链表
4. 集合
5. 字典
6. 树

**2、排序（冒泡，插入，快排）**

答：冒泡，让数组中的当前项和后面的一项进行比较，如果当前项大于后一项，则两项交换位置。

|  |
| --- |
|  |

插入，准备一个新数组，然后依次去原数组中取值，先取一个放到数组里，再取第二个跟新数组里面的进行比较，如果大于就放到后面，小于就再往前找一个，直到找到比当前要小的数，并把它插入后面。

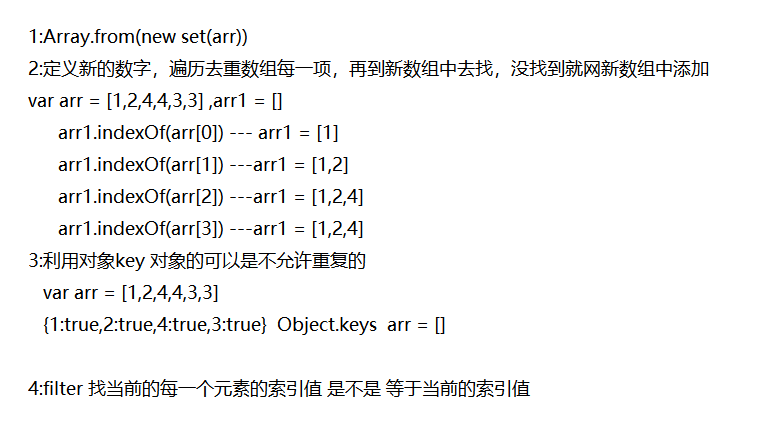
|  |
| --- |
|  |

快排，利用递归

|  |
| --- |
|  |

**3、数组去重**

|  |
| --- |
| function unique(array){ //filter同理  var n = []; //一个新的临时数组  for(var i = 0; i < array.length; i++){ //遍历当前数组  if (n.indexOf(array[i]) == -1){  n.push(array[i]);  }  return n;  }  const arr = [...new Set([arr])]; |



**4、实现一个方法，找出一个数组中重复的元素**

|  |
| --- |
| 一：Array.prototype.repeNum = function(){  let new\_arr = this.sort(); //先排序  let res = [] ;  for( let i = 0 ; i < new\_arr.length ; i++){  if(new\_arr[i] == new\_arr[i+1] && //判断是否重复,是否已经放入容器  new\_arr[i] !=new\_arr[i-1]){  res.push(new\_arr[i]);  }  }  return res  }  二：Array.prototype.repeNum = function(){//不排序，利用对象的key  function arrMore(arr){  if (!arr.length) return  let obj = {};  for (var i = 0,len=arr.length; i < len; i++) {  var val = obj[arr[i]];  if(!val){  obj[arr[i]] = 1  }else{  obj[arr[i]] += 1  }  }  var arr = [];  for (let [key, value] of Object.entries(obj)) {  if(value>1){  arr.push(key-0)  }  }  return arr  }  }  三：Array.prototype.repeNum = function(){  if (!arr.length) return  var ret = arr.filter((val, index) => arr.indexOf(val) != index);//先找到有重复的  return [...new Set(ret)];  } |

**5、数组的flat拉平**

|  |
| --- |
| function flat(arr){  var arr1 = [];//闭包缓存  function \_flat(arr){ //[2,3]  for (var i = arr.length - 1; i >= 0; i--) {  if (Array.isArray(arr[i])) {  \_flat(arr[i])  }else{  arr1.push(arr[i])  }  }  }  \_flat(arr)  return arr1  }  function flat(arr) {  arr.toString().split(',')  return arr  } |

# 6、其他问题

##### 6.1、技术方面

1. **最近做什么项目？在项目遇到什么问题？如何解决的呢？**

答：面试前自己一定要准备一波，把自己在简历上写的东西好好熟悉熟悉，不然面试的时候自己大脑中会一片空白。如果是应届生，把自己平时练习的项目总结好。

如果觉得自己解决的问题都没什么技术含量，也可以说项目中其他同事解决的问题，或者是自己在网上看到的问题。但前提是自己一定要搞清楚，以后自己遇到了也能解决。

1. **最近在学什么技术？**

答：此题主要是想考查你平时爱不爱学习，对新技术有没有一定的敏感度。不用你有多深的领悟了解即可，但千万不能说没有。如果真的没有，去面试之前自己谷歌了解一些。

1. **平时逛什么社区或者技术论坛？**

答：主要是想了解你平时都是通过哪些途径学习，哪些娱乐社区就别说了，说一些前端方面的社区，像掘金，阮一峰老师博客，github等

1. **你对加班怎么看？**

答：根据自己真实情况说就好了，如果你说自己绝对不能加班，那是不可能的。

1. **你还有什么想要了解的吗？**

答：技术面试的时候就别问工资、五险一金啥的，这不是技术项目经理回答的(hr),问团队正在做的项目类型，技术选型方面的问题，这对于你以后的工作很重要。同时如果你拿到offer也可以利用没有去报到的空闲时间弥补。

##### 6.2、HR方面

一般过了技术面试以后，收到offer的大多小伙伴不重视HR面试，但是HR也有否决权，死在这个上面太坑了，稍微准备一下即可，特别是刚毕业出来的小伙伴。

1. **为什么要换工作？**

答：换工作无非于那三个原因，钱给的不够、干的不开心（leader操蛋）、公司没有上升空间。只要大家的原因积极正向一点，别说前公司的坏话就好了，其它自由发挥。记住千万别说公司坏话，哪怕真的很操蛋。

1. **对薪水有什么期望？**

答：这个可以根据当前岗位给定的范围，比如10K-15K，要个12K就行了，不要超过，也不要低于最小值。

1. **对未来有什么规划？**

答：这个问题不要说的太空了，主要从两个方面出发，一个实近期规划（比如深入的学习前端面向对象OOP），还有一个是长远规划（未来5年都在前端技术方向，如何可以转产品或者管理等）

1. **你还有什么想了解的？**

答：现在可以充分了解公司的福利、待遇、公司环境等，这里就别羞射了，有什么想了解的尽管问。