**面试题总结**

**10月14日**

1. **写出下列代码的执行结果**

***Var num1 = 55;***

***Var num2 = 66;***

***Function f1(num,num1){***

***Num = 100;***

***Num1 = 100;***

***Num2 = 100;***

***Console.log(num);***

***Console.log(num1);***

***Console.log(num2);***

***}***

***F1(num1,num2)***

***Console.log(num);***

***Console.log(num1);***

***Console.log(num2);***

答：*Console.log(num); ------*100

*Console.log(num1);-----*100

*Console.log(num2);-----*100

*Console.log(num);------not defind*

*Console.log(num1);----*55

*Console.log(num2);----*100

1. **用两种以上的方法居中一个盒子内部的元素；**

答：①：给父盒子display：flex；

Align-items：center；

Justify-contont:center;

②:父盒子position：relative;

子元素position: absolute;

Left:0;

Right:0;

Top:0;

Bottom:0;

Margin:auto;

③:父盒子：position：relative;

子元素：position: absolute;

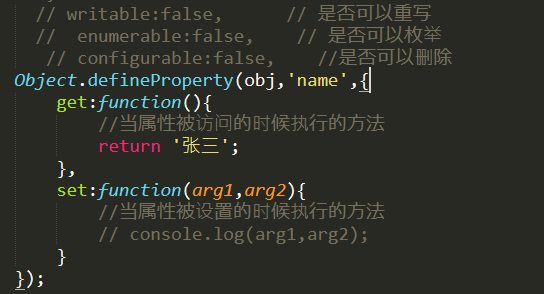
Left:50%;

Top:50%；

Transform:translate(-50%,-50%);

1. **在js中object.** **defineProperty有什么作用**

**答：**



1. **用css写一个三角；**

**答：**.box{

Width:0;

Height:0;

Border:20px solid red;

Border-left-color:transparent;

Border-right-color:transparent;

Border-top-color:transparent;

}

1. **用js实现斐波那契 1 1 2 3 5 8**

**答：**function fn(num){

If(num==1||num==2){

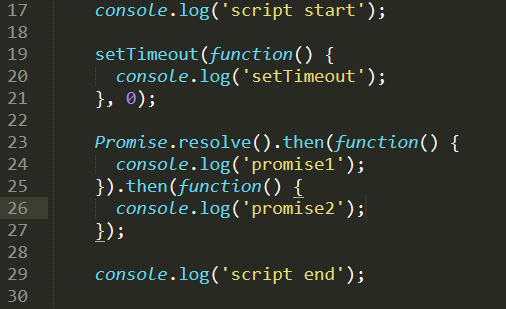
Return 1

}

Ret urn Fn(num-1)+fn(num-2)

}

1. **写出 代码运行效果**

****

**答：17,29,24,26,20**

1. **什么是宏任务，什么是微任务**

**答：**宏任务和微任务都是异步任务在同步任务队列之后

而宏任务一般是:包括整体代码script, setTimeout, setInterval、 setlmmediate。

微任务:原生Promise(有些实现的promise将then方法放到了宏任务中)、

process.nextTick、Object.observe(已 废弃)记住就行了new.promise是宏任务。

执行顺序

先执行宏任务,将宏任务放入event queue ,然后再执行微任务,将微

放

入eventqueue ,但是,这两个queue不是一个queue。当你往外拿的时候先

从微任务里拿这个回调函数,然后再从宏任务的queue拿宏任务的回调函数

1. **闭包是什么，闭包的作用是什么**

**答：**一个函数作为另一个函数的返回值，返回的这个函数可以调用父函

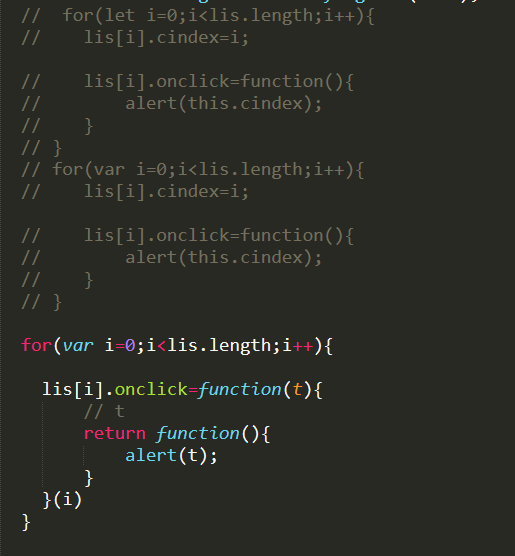
**.**数内部的其他变量

作用：防止全局污染，保存外部活动变量

1. **请写出闭包实际使用场景**

**答：**

1. **点击li弹出对应的索引下标**

**答：**

**10月15日（早上）**

1. **打印下列代码运行结果**

***Var text=（function（i）{***

***Return function(){***

***Alert(i\*2)***

***}***

***}(3)）***

***Text(5)***

**答：**结果为（6）

1. **写出下列代码执行结果：**

***（function（）{***

***Var val=1；***

***Var json={***

***val：10;***

***Col：21；***

***Dbl:function(){***

***Val\*=2***

***Console.log(this)***

***Alert(val)***

***}***

***}***

***Json.dbl();***

***Alert(json.val+val)***

***}（））***

**答：这是一个自调用函数**，第一个val是局部变量，json对象也是一个局部变量，dbl方法中的val\*=2是val=val\*2的简写，由于是json调用了dbl方法所以this指向的是json对象

1. **写出下列代码执行结果**

***function Person(name ,age, salary){***

***this. name name ;***

***this.age age ;***

***this. salary salary;***

***function f1(person){***

***person. name=' ls' ;***

***person =new Person("aa",18, 10);***

***var p=new Person(" zS" ,18, 1000);***

***console. log(p.name);***

***f1(p);***

***console. log(p. name);***

**答：**打印结果为（zs,ls）

单一类型是值传递会开辟新的内存空间，

复杂的数据类型是引用传递，不会开辟新的内存空间，赋值只是引用

当一个变量保存的对象被重新赋值时会开辟新的存储空间，

1. **把get-elmemt-bu-id转换成驼峰命名法**

答：var str =‘get-elment-bu-id’

Var str2=split(‘-’)

For(var i=0;i<str2;i++){

Str[i]=str[i].chartAt(0). toUpperCase()+str2[i].substr(2)

}

Console.log(str)

1. **substr  substring  slice 之间有什么区别？**

**答：***substring(1,5)参数一开始的位置，参数二结束的位置，只能用于截取字符串*

*Substr(1,5)参数一开始的位置，参数二截取的长度，只能用于截取字符串，可以传负数*

*Slice(1,5)参数一开始的位置，参数二结束的位置，可以截取数组，可以传递负数，*

1. **js实现数组的冒泡排序**

*var arr=[1,8,7,3,5,4,9,,40]*

*for(var i=0;i<arr.length;i++){*

*for(var a=a+1;a<arr.length;a++){*

*if(arr[i]>arr[a]){*

*[arr[i],arr[a]]=[arr[a],arr[i]]*

*}*

*}*

*}*

*Console.log(arr)*

1. **用js实现数组反转**

*Var arr=[1,2,3,4,5,6,7,8]*

*For(var i=0;i<arr.length/2;i++){*

*[arr[i],arr[arr.length-1-i]]=[arr[arr.length-1-i],arr[i]]*

*}*

1. **Js实现数组去重**

*Var arr=[1,4,5,54,4,8,32,1,4,5,2,3,54,1]*

*For(var i=0;i<arr.length;i++){*

*For(var c=c+1;c<arr.length;c++){*

*If(arr[i]===arr[c]){*

*Arr.splice(c,1)*

*}*

*}*

*}*

1. **用es6一行代码实现数组去重**

**答：***var arr=[1,2,4,5,7,4,1,4,2,21,4]*

Var set=[…new set(arr)]//转换数组

Var set=Array.from(new set(arr))//非数组转换数组

1. **怎样使用rem做适配**

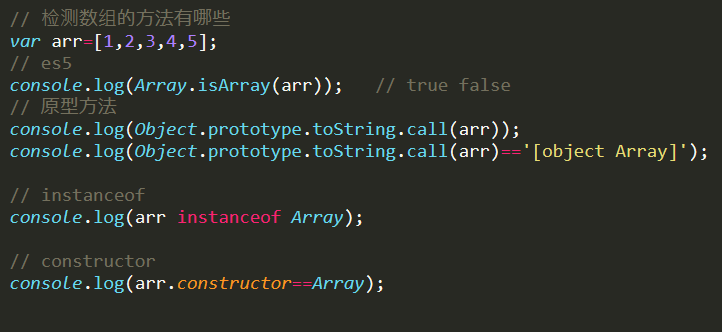
****

**10月15日（下午）**

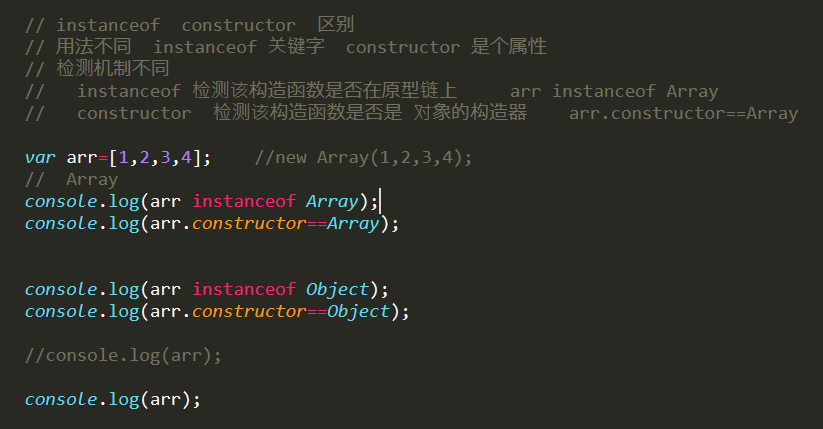
1. **type****of可以检测那些数组类型**

number Boolean string object undefined function

1. **检测是否是数组的方法？**

****

1. **Instanceof 和constructor 检测数据类型有什么区别**

****

1. **cookie  sessionStorage  localstorage 的区别？**

****

1. ***function Foo(){***

***getName function(){alert(1)};***

***return this;***

***}***

***Foo . getName= function(){***

***alert(2)***

***Foo . prototype . getName function( ){***

***alert(3)***

***}***

***var getName function(){***

***alert(4)***

***};***

***function getName(){***

***alert(5);***

***}***

***//写出下面代码的输出结果***

***Foo . getName( );***

***getName( );***

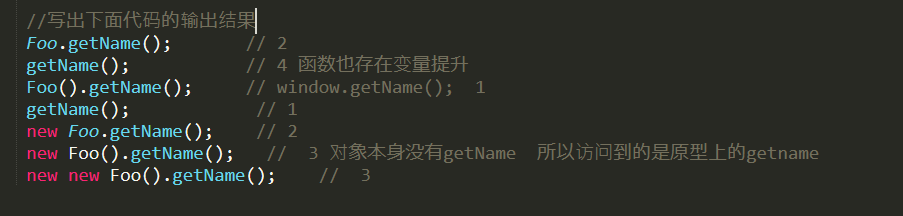
***Foo() . getName( );***

***getName();***

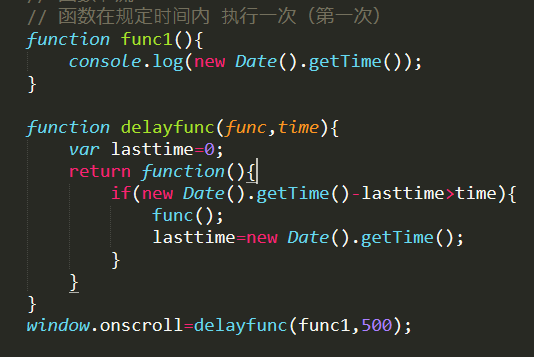
***new Foo . getName();***

***new Foo() . getName();***

***new new Foo() . getName();***



1. **什么函数节流 请写出demo**



1. **什么是函数防抖写出一个demo**



1. **跨域问题是怎样产生的？ 解决方案有哪些？**

同源策略（协议名，域名，端口号（出于安全考虑））

Jspon 代理 cors

1. **Jspon原理是什么**

jsonp开不是一种数据格式，通过动态创建script标签来读取其他域的动态资源

jsonp是用来解决跨域获取数据的一种解决方案， 具体是通过动态创建script标签，然后通过标签的src属性获取js文件中的is脚本，该脚本的内容是-一个函数调用，参数就是服务器返回的数据，为了处理这些返回的数据，需要事先在页面定义好回调函数，本质上使用

的并不是ajax 技术

1. **从一个 url地址输入 到页面呈现在用户面前 发什么了什么？**

**答：**DNS解析：将域名解析为IP地址

Tcp的三次连接：tcp的三次挥手{

浏览器是否可以发送请求

服务端浏览段可以发送请求

浏览器端向服务器发送请求

}

发送HTTP请求，请求头

服务器处理请求返回HTTP报文，响应头

浏览器解析渲染页面

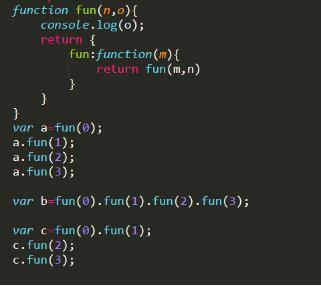
断开连接tcp四次挥手

**10月16日（早上）**

**31：闭包有什么缺陷？**

**答** 滥用闭包会造成内存泄漏，因为闭包中引用的的包裹函数中定义的变量永远不会被释放，所以我们在必要的时候机试释放这个闭包函数（赋值为null）

**32: 写出下列代码的运行结构？**



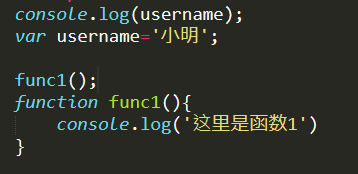
答：undefined 0 0 0

Undefined 0 1 2

Undefined 0 1 1

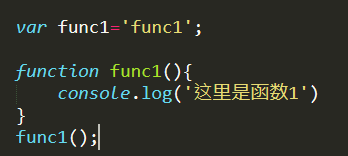
形参需要接受实参，如果没有实参，则为undefined

**33： 写出下列代码执行结构，**



答：undefined 这是函数1

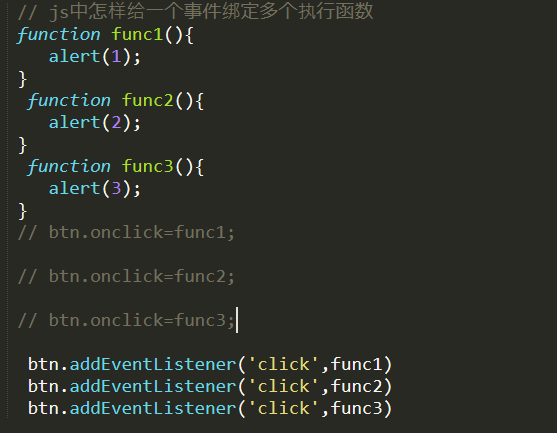
**34：写出下列代码运行结果**



答: 报错func1不是一个函数（因为变量提升的原因后面被赋值为字符串）

**35：给事件绑定多个执行函数**

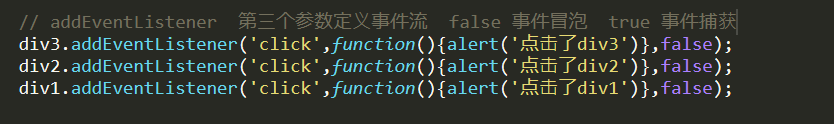
**答：**addEventlistener(w3c规定的标准绑定事件)



**36: js事件流模型有哪些怎样定义**

**答：**事件冒泡，事件捕获

addEventListener 第三个参数定义事件流false 事件冒泡true 事件捕获



**37:传统的事件绑定和符合w3c标准的事件绑定有何区别**

**答：**div1.onclick=function()};

<button onmouseover=” ”></button>

1、如果说给同一个元素绑定了两次或者多次相同类型的事件，那么后面的绑定会覆盖前面，

的绑定

2、不支持DOM事件流事件捕获阶段目标元素阶段=>事件冒泡阶段

addEventListener

1、如果说给同一个元素绑定了两次或者多次相同类型的事件，所以的绑定将会依次触发

2、支持DOM事件流的

3、进行事件绑定传参不需要on前端

addEventlistener(“click"”,function(f,tre);//比时的事件就是在事件冒泡阶段执行

ie9开始，ie11 edge: addEventListener

ie9以前: attachEvent/detachEvent

**38： js兼容有哪些写法**

**答：** var ev= evtI | window.event

//获取event对象

var target = ev.target|| ev.srcElement

//获取时间源

document.documentElementclientWidth |I document.bodyclientWidth // 获取屏幕的宽度

Dom.addEventListenerl ldomattatchEvent

// w3c标准的事件绑定

**39: call apply bind方法有什么区别**

**答：**都可以该表this指向

Bind 返回值this指向被改变后的函数

Apply call 改变this指向并执行函数

APPly 传递的参数必须是一个数组

Call 可以一个一个传递参数



**40： 使用call或者apply实现bind方法**

**答：**



**10月16日（下午）**

**41：** **es5 函数 this指向有几种情况？**

**答：**① 指向window对象

②指向调用对象

③指向new出来的对象

④call apply bind 可以改变this指向

**42: es6 箭头函数 this指向和普通函数this指向的区别？**

答：  箭头函数 this指向函数声明时的上下文环境

**43： 什么是原型继承 写一个实例.**

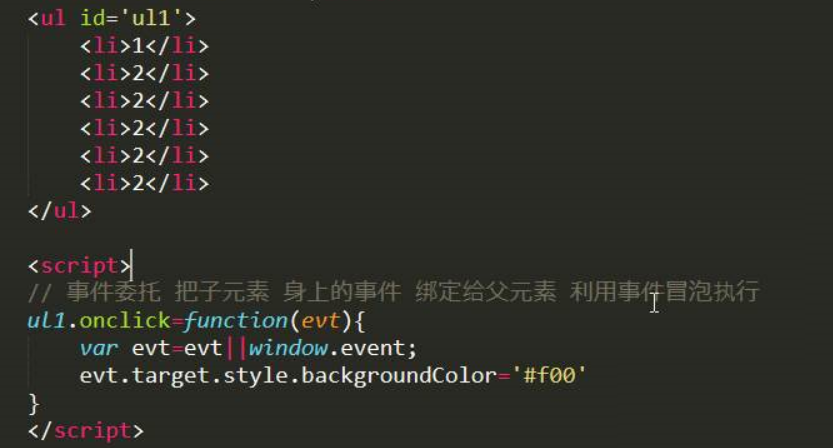


**44： 什么是构造函数继承  写一个实例**



**45：事件委托是什么**

**答：**把子元素身上的事件 绑定给父元素 利用事件冒泡执行



**46：如何阻止事件冒泡和默认事件**

**答：**e. stopPropagation()://标准浏览器

event.cancelBubble=true;//ie9之前

阻止默认事件:

为了不让a点击之后跳转，我们就要给他的点击事件进行阻止

retum false

e.preventDefault();

**47: 添加 删除 替换 插入到某个节点的方法**

**答：**Document.createELement('div'); //创建一个新的节点

Obj.appendChild()//插入一个新的节点

obj.insertBefore() //原生的js.中不提供insertAfter(); 插入一个新的节点

obj.replaceChild(//替换

obj.removeChild(//删除

**48、javascript 的本地对象，内置对象和宿主对象有哪些?**

本地对象为array object regexp等可以new实例化

内置对象为gload Math等不可以实例化的

宿主为浏览器自带的documentwindow等

**49：window.load和document ready的区别**

window.onload:不仅仅要在结构和样式加载完，还要执行完所有的样式、图片这些资源文

件，全部加载完才会触发window.onload事件

Document.ready内存中加载完成就触发速度快

window.onload: ?

只执行一次

Document.ready多 次使用

Document.ready原生种没有这个方法，jguery 中有$(document).ready(function)

可以多次执行|

**50：”==”和“===”的不同 a++ 和++a 有什么不同？**

**答：**==是只比较值是否相等

===比较数据类型和值 react只能使用===

A++ a先赋值，然后a自增1，最后返回赋值先进行表达式运算，再进行自增运算

++a 将A自增然后返回a 先进行自增运算，再进行表达式运算。

**10月17日（早上）**

**51： 输出今天的日期yyyy-mm-nn**

**答：**var timeobj =new Date()

var year-timeobj . getFullYear();

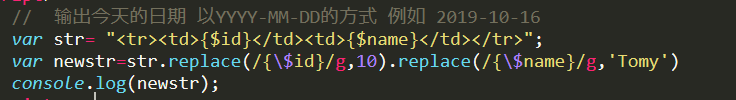
var month = timeobj . getMonth() +1>=10? t imeobj . getMonth() +1: '0'+ (timeobj . getMonth()+1);

var date t imeobj . getDate()>=10?t imeobj . getDate():'0'+timeobj .getDate();

console.log( `${year}- ${month}一${date}` );

**52:将字符串“<tr><td>{$id}</td><td>{$name}</td></tr>”中的${id}替换为10${name}替换成Tomy**

**答：**

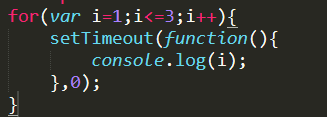


**53：js解析url的get参数例如：**[**www.baidu.com?name=1&age=2&like=3**](http://www.baidu.com?name=1&age=2&like=3)

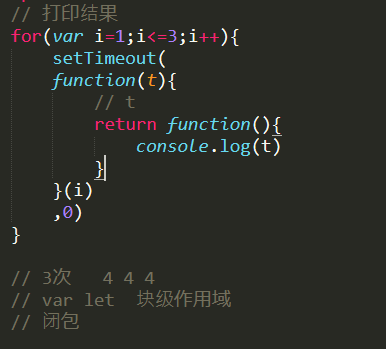
**答：**



**54：打印输出结果：并改变函数**



**答：**



**55：js 中caller 和callee的作用？**

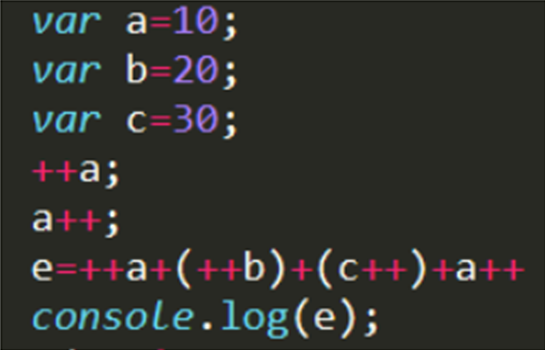
**答：**callee 当前函数的指引 始终指向函数本身

Caller 调用自身的函数 谁调用我 我指向谁

**56： 那一条语句会报错( A B C )**

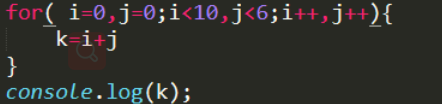
**A var a=() B var 0bj=[ ] C var a= // D var obj={ }**

**57: 打印结果**



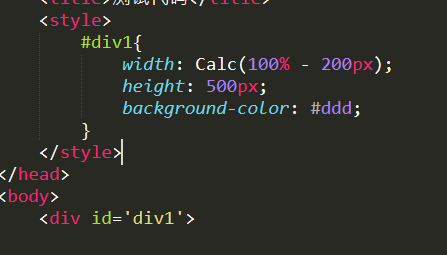
答：77

**58： 打印输出结果：**



**10**

**59： css计算属性 并且举例**



**60: Css3 有哪些新特性**

**答：** css3新增选择器圆角 图片边框 文字阴影 盒模型 弹性盒子过渡 动画

h5语义化标签 表单增强音视频canvas svg本地存储 websoket 地理位置定位

**10月17日（下午）**

**61：ajax 请求数据时get和post方式有什么区别**

答：GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。

(浏览器执行机制)

对参数的数据类型，GET只接受ASCII字符，而POST没有限制。

(数据类型)

GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信

息。

(安全性)

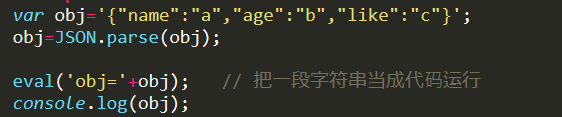
GET参数通过URL传递，POST放在Request body中。

(传参机制)

GET请求在URL中传送的参数是有长度限制的(256bit)，而POST没有。

(数据长度)

**62：** **ajax请求时，如何解析json数据**



**63： typeof null 返回值是多少  为什么？**

**答：**object null指向一个空指针对象

**64： 判断字符串以字母开头，后面可以是数字，下划线，字母，长度为6-30**

**答：**/^[a-zA-Z]\w{5,29}$/

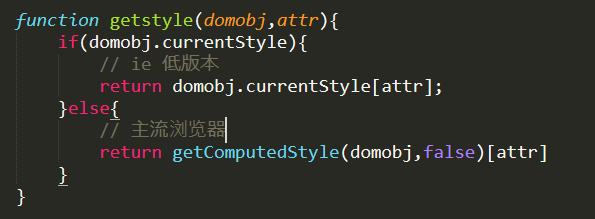
**65: new操作符具体干了什么**

**答：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。

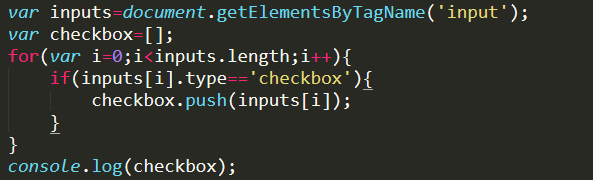
②属性和方法被加入到this 引用的对象中。

③新创建的对象由this 所引用，并且最后隐式的返回this **。**

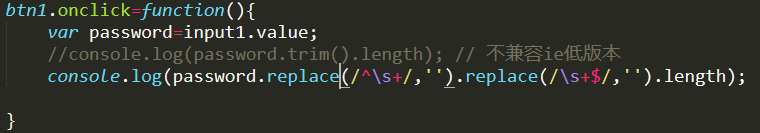
**66：Js 获取css样式**

****

**67： 不使用第三方框架 获取页面中所用的checkbox?**

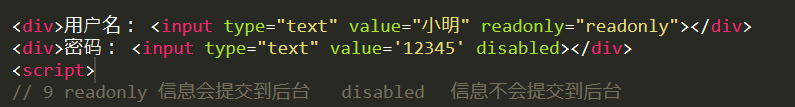


**68：写一个方法 清楚字符串 前后空格 兼容所用的浏览器**

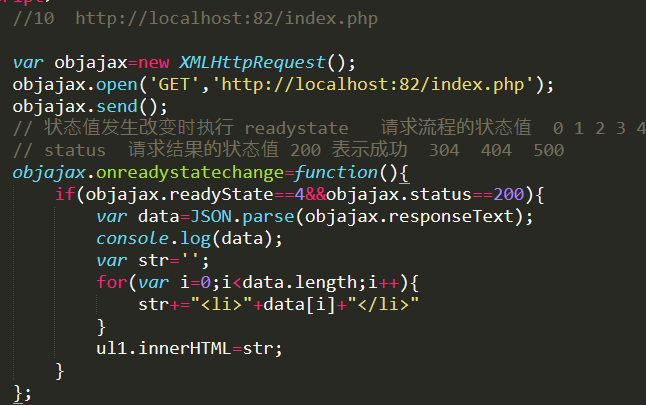


**69：input设置readyonly和disable 属性 有什么区别？**

**答：**



**70：ajax实现原理  或者 用原生js实现ajax**



**10月18日（早上）**

**71：找出字符串中出现次数最多的字符：例如：‘aaaccvffdddssss’**

**答：**var str=”aaaccvffdddssss”

Var obj={}

For（var i=0;i<str.length;i++）{

If(obj[str.charAt(i)]){

Obj[str.charAt(i)]++

}else{

Onj[str.charAt(i)]=1

}

}

Var max=0;

Var maxkey=””;

For(var k in obj ){

If(obj[k]>max){

Max=obj[k]

Maxkey=k

}

}

**72: 反转字符串或者排序字符串：例如：“abcdef”反转为“fedcba”排序“acdbfe”排序为：‘abcdef’:**

**答：**var str=”abcdef”

Console.log(str.split(‘’).reverse().join(‘’))

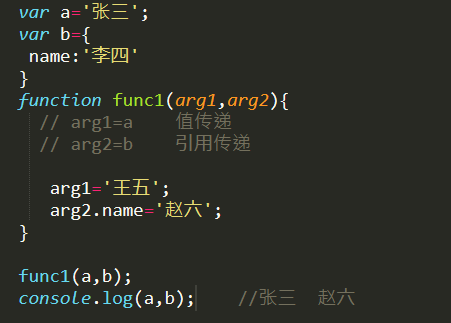
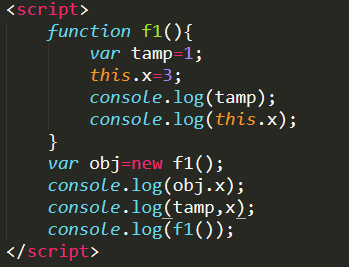
Var str2=”acdfsv”

Console.log(str.split(‘’).stort().join(‘’))

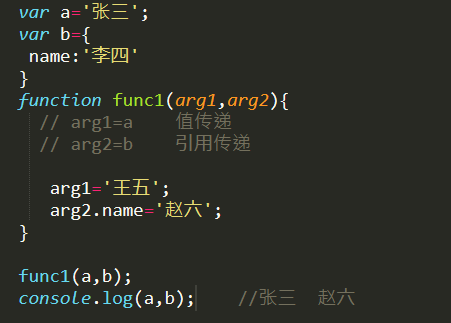
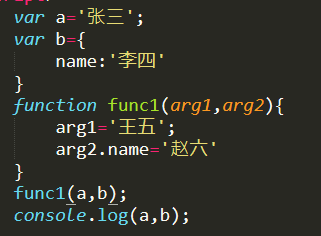
字符串的方法.length indexOf lastInde0f charat substr subString slice upperCase lowerCase split

数组的方法： unshift shift pop push join concat reverse sort splice slice map filter reducer some every

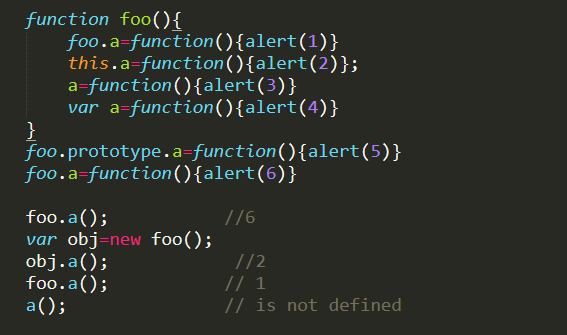
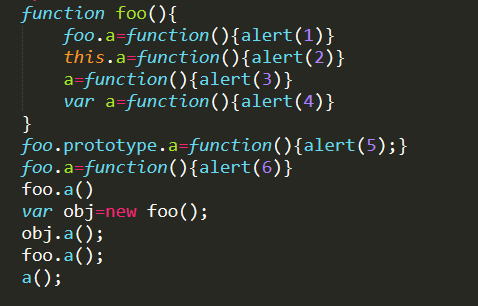
**73：打印输出结果；**



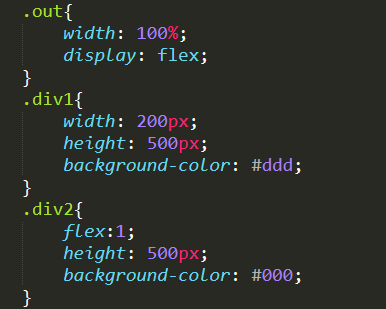
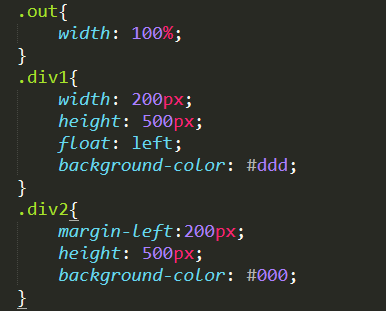
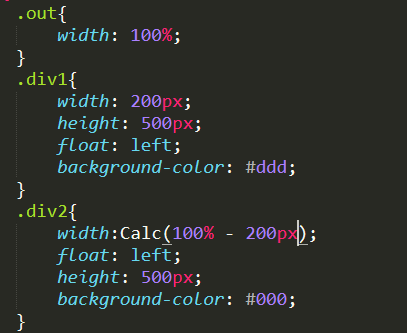
**74：打印出下列代码结过**



**75：打印出下列代码运行结果：**



**76：写出一个左侧固定宽度，200px右侧自适应的布局；**



**77：bind（）live（）delegate() .on的区别**

**答：**bind:绑定事件， 对新添加的事件不起作用，方法用于将一个处理程序附加到每个匹配元

素的事件上并返回jQuery对象。

live:

方法将一个事件处理程序附加到与当前选择器匹配的所有元素( 包含现有的或将

来添加的)的指定事件上并返回jQuery对象。.

delegate:方法基于-组特定的根元素将处理程序附加到匹配选择器的所有元素(现有的或

将来的)的一个或多个事件上。

.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。推荐使用on

**78：简述link和import的区别？**

答： 区别1: link 是HTML标签，除了加载CSS外，还可以定义RSS等其他事务; @import 属于CSS范畴，只能加载CSS。

区别2:link引用CSS时，在页面载入时同时加载;@import需要页面网页完全载入以后加

区别3: link 是HTML标签，无兼容问题，@import 是在CSS2.1提出的，低版本的浏览器不

支持。

区别4: link 支持使用Javascript 控制DOM去改变样式:而@import不支持。

**79：http状态码有哪些分别表示什么？**

**答：**200:请求已成功，请求所希望的响应头或数据体将随此响应返回。

302:请求的资源临时从不同的URI 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。只有在Cache-Control或Expires中进行了指定的情况下，这

个响应才是可缓存的

304:

: 如果客户端发送了一个带条件的GET请求且该请求已被允许，而文档的内容(自.上

次访问以来或者根据请求的条件)并没有改变，则服务器应当返回这个状态码。304响应禁

止包含消息体，因此始终以消息头后的第一一个空行结尾。

403:

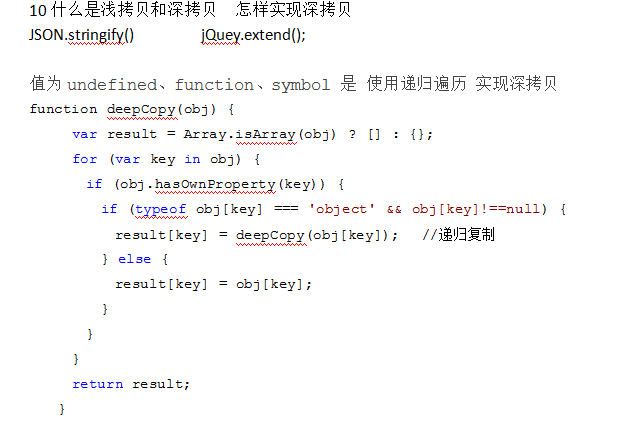
服务器已经理解请求，但是拒绝执行它。|

404:请求失败，请求所希望得到的资源未被在服务器上发现。

500:服务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。一般来说，这

个问题都会在服务器端的源代码出现错误时出现。

**80：什么是浅拷贝什么是深拷贝**



**10月18日（下午）**

**81  \_\_proto\_\_    prototype 区别和联系**

**答**



**82：什么情况下出发重排和重绘？**

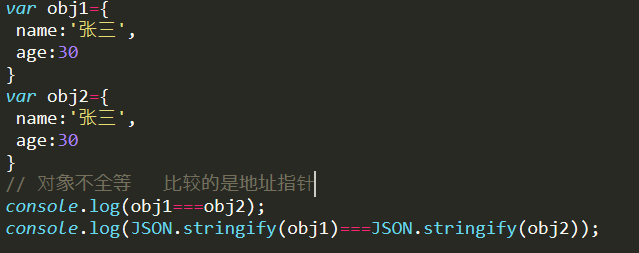
**答：**添加、删除、更新DOM节点

通过display:none 隐藏一个DOM节点\_触发重排和重绘

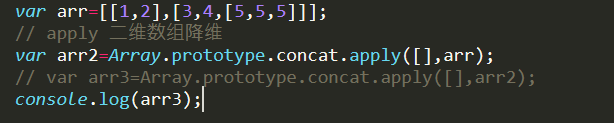
通过visibility: hidden隐藏-个DOM节点只触发重绘，因为没有几何变化

改变元素的颜色透明度仅仅会引发重绘不会引发重排

**83:怎样判断两个对象相等**



**84：怎样将一个二维数组转换为一维数组**



**85：offsetWidth offsetHeight和clientWidth clientHeight的区别**

答：(1)offsetWidth ( content宽度+ padding宽度+border宽度)

(2)offsetHeight ( content高度+ padding高度+border高度)

(3)clientWidth (content 宽度+padding宽度)

(4)clientHeight (content 高度+padding高度)

**86：JS中创建对象的多种方式**



**87：哪些操作会造成内存泄露？**

大量使用全局变量

大量使用闭包

清除DOM节点时，只清除节点，而没有删除事件

onclick=null addEventListener removeEventListener

定时器不清除

**88：defer和asynl是什么属性？**

答： 1默认引用script:<script type=" text/javascript" src="x.min.js"></script>

当浏览器遇到script标签时，文档的解析将停止,并立即下载并执行脚本，脚本执行完

毕后将继续解析文档。

2.async (异步)模式

<script type=' ' text/javascript" src=' x.min.js" async= 'async"></script>

当浏览器遇到script标签时，文档的解析不会停止，

其他线程将下载脚本，脚本下载完

成后开始执行脚本，脚本执行的过程中文档将停止解析，直到脚本执行完毕。

3.defer(延迟)模式

<script type=" text/javascript" src= ' x.min.js" defer=' 'defer"></script>

当浏览器遇到script 标签时，文档的解析不会停止，其他线程将下载脚本，待到文档解

析完成，脚本才会执行。

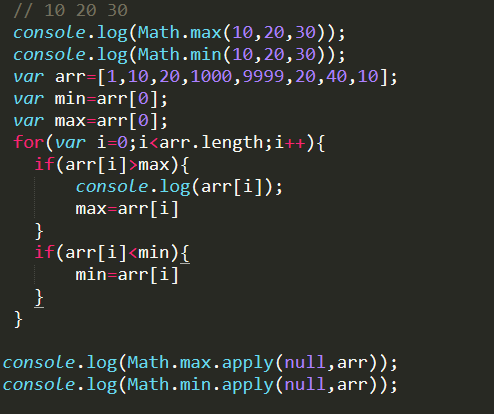
**89:javascript的两种变量范围有什么不同？**

全局变量:当前页面内有效

局部变量:函数方法内有效

Es6 let 块级 const块级

**90：如何获取JavaScript三个数中的最大值最小值和数组中的最大值和最小值**



**10月21日（早上）**

**91: Es6有哪些新特性**

**答：**1) let声明变量和const声明常量，两个都有块级作用域

2)模板字符串字符串方法扩展

3)数组扩展方法

4) ... 展开运算符;

结构赋值

5) class类和继承

6) Promise 解诀异步回调async await

7)新增数据类型map set symbol

8)模块化expot导出 import导入

**92：var let const 之间有什么区别？**

**答：**var 声明变量可以重复声明 而let不可以

Var 是不受限于块级的 而let 是受限于块级的

Var 会与window相映射（会挂载一个属性）， 而let不与window映射

Var 存在变量提升， let 不存在变量提升，但会有暂存死区，在申明的上面访问会变量报错

Cons声明之后必须赋值，否则会报错，

Const定义的不可变得量，改变了就会报错

Const和let一样不会与window映射，支持块级作用域，在声明变量上面访问变量会报错

**93： 使用箭头函数应该注意什么？（箭头函数和function的区别）**

**答：**（1）用了箭头函数，this就不是指向window 而是声明时的外部环境

（2）箭头函数不可以使用arguments（参数的集合）对象

（3）箭头函数不能使用构造函数，也就是说不能使用new命令，否则会抛出一个错误

（4）不可以使用yield 命令，因此箭头函数不能用作Generator

**94: es6怎么写class ， 为何会出现class**

**答：** es6新增关键字吗，新增语法糖，他的绝大部分功能Es5都可以实现，新的class写法只是让原型对象的写法更加清晰，更像面向对象编程语法，可以实例化对象类中申明属性和方法属性声明在constructor中，对象和对象之间通过extends关键字实现继承

**95：Promise构造函数是同步执行还是异步执行，那么 then 方法呢？**

**答：**promise构造函数是同步执行的，then 方法是异步执行的

**96：setTimeout、Promise、Async/Await 的区别**

**答：示例代码**



事件循环中分为宏任务队列和微任务队列

其中setTimeout的回调函数放到宏任务队列里，等到执行栈清空以后执行

promisethen里的回调函数会放到相应宏任务的微任务队列里,等宏任务里面的同步

代码执行完再执行

async函数表示函数里面可能会有异步方法，await 后面跟- -个表达式

async方法执行时，遇到await会立即执行表达式，然后把表达式后面的代码放到微

任务队列里，让出执行栈让同步代码先执行

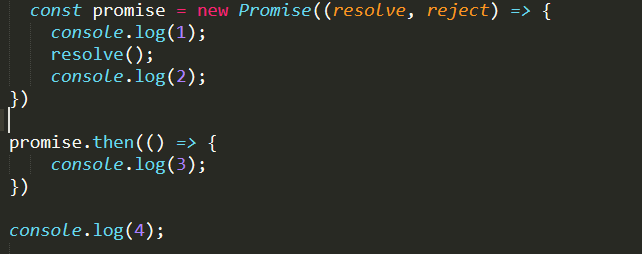
**97：promise 有几种状态 什么时候进入Cath**

**答：**promise有三种状态: pending/reslove/reject 。pending 就是未决，resolve 可以理解为成功，reject可以理解为拒绝。

两个过程: pendding ->resolve、penHding -> rejected

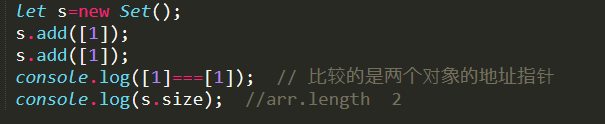
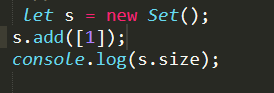
当pending为rejectd时，会进入catch

**98：打印出下列代码运行结果**



答：1，2，4，3状态改变之后的代码依然会执行

**99：下面的代码执行结果**



**100：Promise 中reject 和 catch 处理上有什么区别**

**答：**reject是用来抛出异常，catch 是用来处理异常

reject是Promise的方法，而catch 是Promise 实例的方法

reject后的东西，一定会进入then中的第二个回调，如果then中没有写第二个回调，则进入catch

网络异常(比如断网)，会直接进入catch而不会进入then的第二个回调

**10月21日（下午）**

**101：forEach、for in、for of三者区别**

**答:** forEach更多的用来遍历数组

for in一般常用来遍历对象

forof数组对象都可以遍历，遍历对象需要通过和Object.keys()

for in循环出的是key, for of循环出的是value

**102: 什么时候用到export import？export 和export default 有什么异同？**

**答:** 比如在项目中自定义一些公共的方法定义 组件

都可以导出模块区别:|

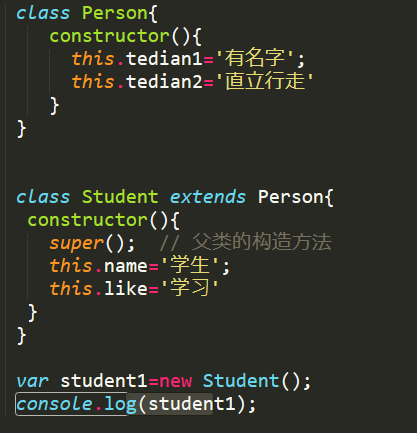
export可以多次使用导入要加{} 名 称对应导出时的名字

Export default只能使用 一次 导入不用加{}

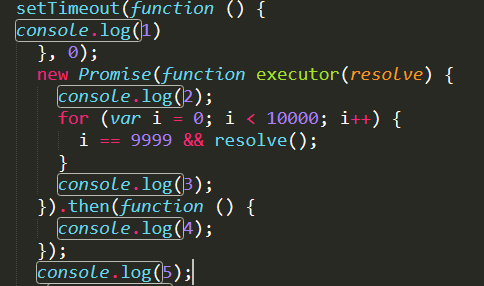
可以任意命名.

**103：使用class声明两个类并实现对象继承**

答：



**104：写出下列代码执行结果**



2 3 4 5 1

**105: promise.all方法执行流程？**

**答**：var P =Promise.all ([p1,p2,p3]);

P的状态由p1、p2、p3决定，分为两种情况。

当该数组里的所有Promise实例都进入resolve状态:Promise . all\*\*返回的实例才会变成resolve

状态。并将Promise实例数组的所有返回值组成一一个数组，传递给Promise . all返回实例的回调函数

当该数组里的某个Promise实例都进入Rejected状态: Promise. al1返回的实例会立即变成

Rejected状态。并将第一个rejected的实例返回值传递给Promise. all返回实例的回调函数。

**106：请用一行代码实现去重**

**答：**var arr =[1,2,4,5,7,8,5,4,1,1,]

Var set=[…new set(arr)]

Var set=Array.from(new set(arr))

**107: 不使用第三个变量交换两个变量的值**

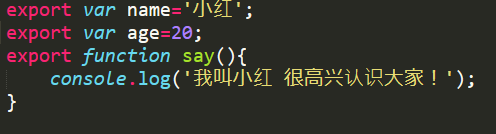
答：let a=1

Let b=2

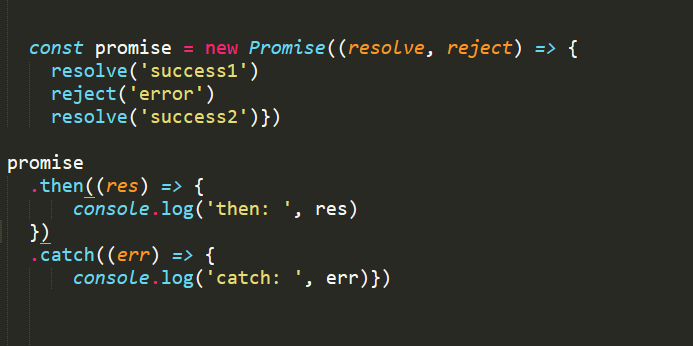
[a,b]=[b,a]

**108: 利用module模块，实现两个模块A和B，A模块导出变量name，age和say方法。B模块只导入变量name和say方法，并且重命名name为nickname。**

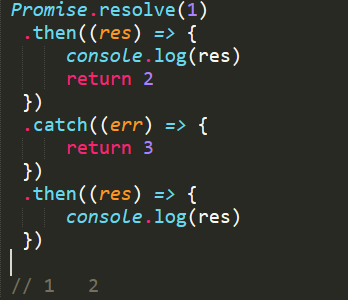
**答：**



**109：打印下列代码执行结果**



**110: 打印下列代码结果**



**10月22日（早上）**

**111: vue全家桶式开发？ 什么是vue渐进式开发？**

**答：**vue-router，vuex， vue-resource

渐进式：引入vue.js部分功能，追渐取代之前的原生js jquery

**112: 什么是mvvm?**

**答：**mvvm

m

model (模型层) axios data vuex(state)

V view (视图层) template < template>< /template>

Vm

viewmodel是双向绑定实现的机制

**113：数据渲染 有几种方式  有什么区别？**

**答：**{{}}

v-html:只有v-html可以渲染HTML代码

v-text

**114：** **v-if和v-show有什么区别，切换时组件触发那些生命周期钩子函数？**

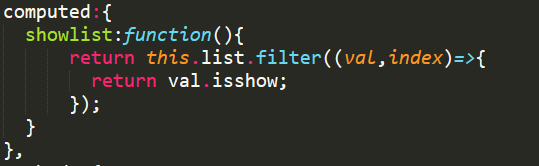
**答：**v-if控制的是dom的销毁和创建v-show只是控制dom的css中 display属性

触发的生命周期 v-if 显示：创建 挂载 隐藏：销毁

v-show显示隐藏不触发生命周期钩子函数

**115：v-if和v-for为什么不建议混合使用**

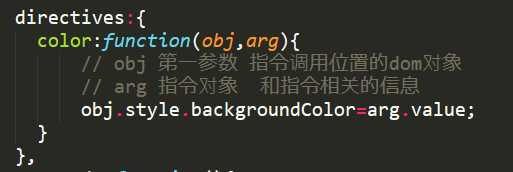
**答：**被渲染的dom会重复一个创建销毁的过程



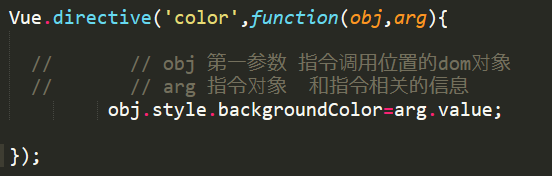
解决方法 使用计算属性 过滤数据

**116：vue中如何自定义指令（全局和局部）**

**答:**局部指令



全局指令



**117：vue中常用的指令有哪些**

**答：**V-text：主要用来更新textContent，可以等同于JS的text属性

V-html：双大括号的方式会将数据解释为纯文本，而非HTML。为了输出真正的HTML，可以用v-html指令。它等同于JS的innerHtml属性。

V-show：也是用于根据条件展示元素。和v-if不同的是，如果v-if的值是false，则这个元素被销毁，不在dom中。但是v-show的元素会始终被渲染并保存在dom中，它只是简单的切换css的dispaly属性。

V-if：可以实现条件渲染，Vue会根据表达式的值的真假条件来渲染元素。

V-else-if：充当v-if的else-if块，可以链式的使用多次。可以更加方便的实现switch语句。

V-else：是搭配v-if使用的，它必须紧跟在v-if或者v-else-if后面，否则不起作用。

V-on：主要用来监听dom事件，以便执行一些代码块。表达式可以是一个方法名。

V-bind：用来**动态的绑定一个或者多个特性**。没有参数时，可以绑定到一个包含键值对的对象。常用于动态绑定class和style。以及href等。

v-for：用v-for指令根据遍历数组来进行渲染

V-model：这个指令用于在表单上创建**双向数据绑定**

V-slot

V-pre：主要用来跳过这个元素和它的子元素编译过程。可以用来显示原始的Mustache标签。跳过大量没有指令的节点加快编译。

V-cloak：这个指令是用来保持在元素上直到关联实例结束时进行编译。

V-once：v-once关联的实例，只会渲染一次。之后的重新渲染，实例极其所有的子节点将被视为静态内容跳过，这可以用于优化更新性能。

**118: 自定义指令接受那些生命周期钩子函数**

**答：**



**119：vue循环用什么语法  key的指定起到什么作用？**

**答：** v-for

key给虚拟dom添加一个唯一 值提高虚拟dom的对比效率

transtion-group使用必须添加key

**120：什么是高阶函数你知道那些高阶函数？**

**答：**高阶函数 能接收一个回调函数作为参数的函数

arr . sort(()=>{});

arr. map()

arr. filter

arr . reduce(()=>{})

arr . some( )

arr. every()

arr . find()

**10月22日（下午）**

**121：计算属性和实例方法有什么区别？**

**答：**计算属性有缓存，实例方法没有缓存

计算属性调用时 total 不可以传参 实例方法调用时 total（）可以传参

**122：如何声明一个过滤器（全局的和局部的）**

**答：**vue.filter(‘gettime’,function(date){

Var time=new date(date)

Return time.getFullyear()+’/’+time.getMonth()+1’/’+getDate()

})

Filters:{

Gettime(date){

Var time=new date(date)

Return time.getFullyear()+’/’+time.getMonth()+1’/’+getDate()

}

}

**123:** **你用过那些v-model修饰符？**

答：.lazy文本框失去焦点后在更新数据

.trim清除前后空格的

.number从字符中获取数字

**124：你知道那些事件修饰符**

答：.stop阻止事件冒泡

.once只执行一次，一次性事件

.self阻止事件冒泡和事件捕获

.captrue事件捕获阶段触发

.prevent阻止浏览器默认行为

.native 触发原生事件

**125：怎样声明全局组件，怎样使用全局组件**

答：vue.component（‘myheader’,{

Template:`

<div>全局组件</div>

`

}）

<myheader></myheader>

**126:** **怎样声明局部组件，怎样使用局部组件**

答：声明成 \*\*\*.vue文件中模板在<template></template>中。Js模块化导出<script></script>css<style scoped></style>

定义 import引入 注册components 调用

**127：父子组件怎样实现通讯？**

答：父传子：父组件：<mydiv :list=’list’> </mydiv>

子组件：props[‘list’]

子传父：子组件：this.$emit(‘子组件事件名’，参数)

父组件：<mydiv @子组件事件名=‘父组件事件名’></mydiv>

**128：非父子组件之间如何传递参数？**

答：bus总线

兄弟组件可以父传子 子传父

Vuex也可以实现数据共享

**129：怎样使用props限定传递的数据，有哪些属性？**

答：type //限定传递的数据类型

Required:true false //是否必传

Default:false //指定默认值

Validator：function(){

//自定义规则

}

**130：非props属性有什么特点**

答：①会显示在解析之后的标签上

②子组件不能使用该属性

**10月23日（早上）**

**131：解释VUE中单向数据流的概念**

**答：**子组件不可以直接修改父组件传过来的参数，如果直接修改会报错，只能声明成自己的才可以修改

**132：什么是插槽，怎么使用，什么是具名插槽，如何使用**

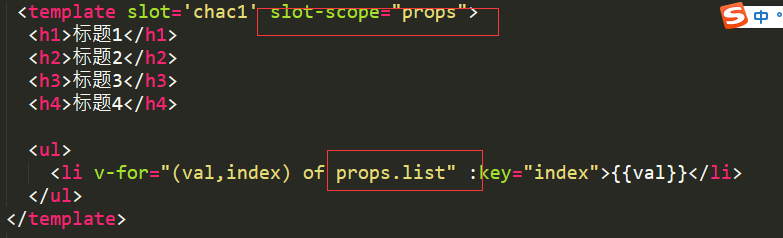
答：能够接受组件外部传给组件大段的HTML代码 <slot></slot>

具名插槽 具有name属性的插槽 可以结合template标签的slot属性，定向给插槽传递内容

**133：什么是作用域插槽？**

答：插槽获取子组件的数据 需要属性传递 结合作用域插槽语法，代码如下

Slot-scope=‘props’



**134：vue中怎样实现组件之间的嵌套**

**答：**组件之间的嵌套使用，可以借助插槽，让插槽接受子组件中的内容

**135：vue中怎样操作dom**

答：ref=’name’

This.$refs.name

**136: vue怎样监听数据的变化，怎样监听一个对象属性的变化**

答：watch



**137：vue中生命周期钩子函数有哪些，请写出语法**

答：创建前beforeCreate 创建后created

挂载前beforeMount 挂载后mounted

更新前beforeUpdate 更新后updated

销毁前beforeDestroy 销毁后destroyed

激活activated 未激活deactivated

**138：挂载和创建之间的区别**

答：创建之后只可以访问数据不可以操作dom

挂载之后可以访问数据可以操作dom

**139: keep-alive组件有什么作用？**

**答**：缓存组件缓存页面（router-view）

**140：什么时候触发激活和未激活生命周期钩子函数**

**答：**当页面 或者 组件被缓存 激活activated 未激活 deactivated 会取代 创建和销毁

**10月23日（下午）**

**141：keep-alive组件有哪些属性？**

**答：**include  声明需要被缓存的组件 页面  
       exclude  声明不需要被缓存的 组件 页面

**142：vue.nextTick方法有什么用？**

**答：**数据更新是一个 异步操作异步dom更新后使用vue.nextTick可以立即得到更新后的 数据

**143：vue实现双向数据绑定的原理是什么？**

**答：**当一个vue实例创建后vue会便利data选项的属性，用object.definproperty将他转化为getter或者setter，并且内部追踪相关依赖，在属性访问和修改时通知变化每一个组件实例化都有相应的watcher程序实例，他会在组件渲染的过程中把属性记录为依赖，之后当依赖项的setter被调用时会通知watcher重新计算，从而使得它相关的组件的以更新

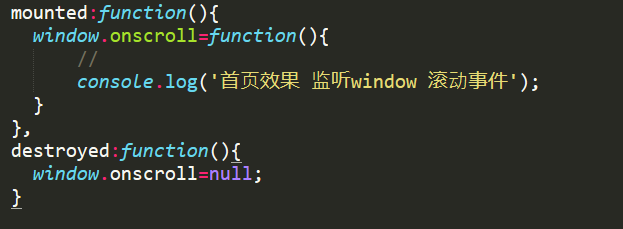
**144：用一段代码 模拟vue双向数据绑定？**

**答：**





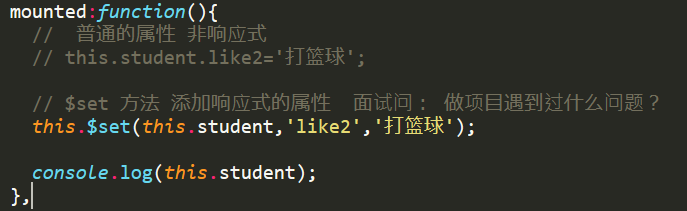
**145：什么是vue全局事件解绑？**



在vue中，写在methods里面的方法，在页面切换时一般就自动销毁了，但是写在window上的事件，需要我们手动解绑，也是为了提高代码执行效率，使用生命周期函数Destroy，在页面离开时执行解绑

**146：$set 方法有什么用？ 有什么特点？**

**答：**

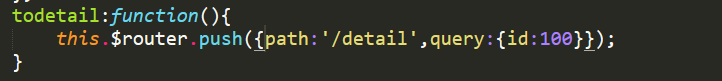
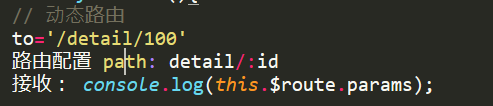


**147：路由显示区域和路由跳转 分别是哪个组件？**

答:显示：router-view 跳转 router-link

**148：路由中怎样传递和接收参数（单个 多个参数）**

**答：**



**149：怎样给路由添加 active-class ?**

**答：**配置文件中linkActiveClass

默认的是router- link- active

可以直接在标签中使用active-class

**150：vue-router有哪几种导航钩子 分别是什么？**

答全局

前置router . beforeEach

后置router. afterEach

路由独享的

前置beforeEnter.

组件内

beforeRouteEnter(to, from, next){

//在渲染该组件的对应路由被confirm 前调用

beforeRouteUpdate(to, from, next) {

//在当前路由改变，但是依然渲染该组件是调用

}，

beforeRouteLeave(to, from ,next) {

/ /导航离开该组件的对应路由时被调用

}

**10月24日（早上）**

**151：vue中子组件调用父组件的方法**

答：第一种方法是直接在子组件中通过this.$pareht.event来调用父组件的方法

第二种方法是在子组件里用$emit向父组件触发-一个事件，父组件监听这个事件。

第三种是父组件把方法传入子组件中，在子组件里直接调用这个方法

**152：路由懒加载怎么实现，有什么意义**

答：let Addclass = resolve =>require([' @/views/ Addclass' ] ，resolve )

**Vue项目性能优化?**

**代码减少对服务器请求次数 减小对服务器 请求代码的体积**

**V-if v-show**

**全局事件解绑函数节流函数防抖**

**Keep- alive缓存**

**css精灵图字体图标**

**spa单页应用图片 (数据)懒加载**

**路由懒加载**

**Webpack**

**打包压缩代码**

**153：vue路由模式有哪几种，怎样切换有什么区别**

答：(1 )在url显示: hash有# ; history 无#对seo比较友好

(2)回车刷新: hash可以加载到hash值对应页面; history 一般就是404掉了

(3 )支持版本: hash支持低版本浏览器和IE浏览器 ; historyHTML5新推出的API不支持ie低版本

**154：vue组件中的data为什么必须是一个函数**

答：对象为引用类型，当重用组件时，由于数据对象都指向同一个data对象，当在一个组件中修改data时，其他重用的组件中的data会同时被修改;而使用返回对象的函数，由于每次返回的都是一个新对象(Object的实例)，引用地址不同，则不会出现这个问题。

**155：$route和$router有什么区别**

$route是路由信息对象”，包括path, params， hash， query， matched, name 等路由信息参数。

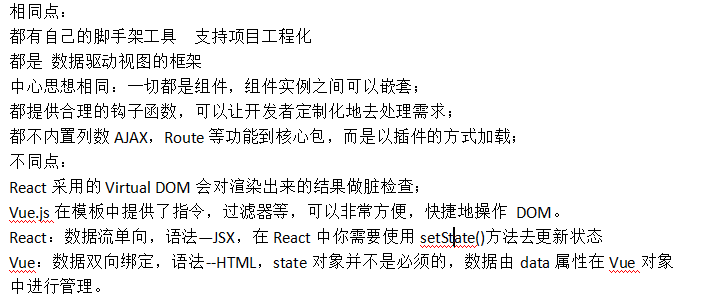
而$router是“路由实例"对象包括了路由的跳转方法，钩子函数等

**156：vue如何兼容ie**

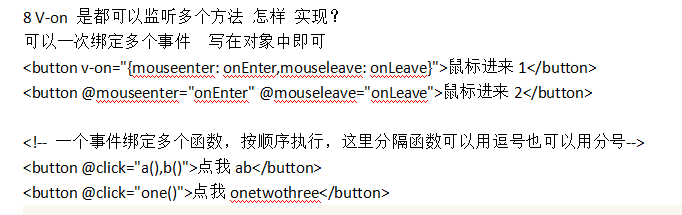
插件 最低兼容得到ie9  babel-polyfill

<https://blog.csdn.net/XuM222222/article/details/81561890>

**157：**.**vue与React的异同**



**158：V-on 是都可以监听多个方法 怎样 实现？**0



0

**159： Vue项目中如何区分开发环境和生产环境**

const urlDev = 'http://dev.womaoapp.com

const urlQa = htt://qa.oss.womaoapp.com

var urd =

process . env. NODE ENV === 'development' ? url = urlDev : url = urlQa

const excelUrl = url + '/fwas.message admin/sys/fileDown/excel?channelType=1' // 我茂消息发送:获取excel示例

export { excelUrl }

**160：什么情况会使用到vuex    vuex有哪些核心的属性？**

例如用户信息的存储.

购物车数据的存储

State

action mutation getters

module

**封装axios  请求超时时间  baseurl设置  请求头设置  （token 的设置） 错误的处理  （未登录）  token过期   404**

**https://www.cnblogs.com/ldlx-mars/p/7908950.html**

<https://www.cnblogs.com/chaoyuehedy/p/9931146.html>

**10月28日（早上）**

**161：简单描述一下微信小程序的相关文件类型**

答:微信小程序项目结构主要有四个文件类型,如下:

WXML (WeiXin Markup Language)是框架设计的一套标签语言,结合基础组件、

事件系统，可以构建出页面的结构。内部主要是微信自己定义的一套组件。

WXSS (WeiXin Style Sheets)是一套样式语言，用于描述WXML的组件样式，

js逻辑处理，网络请求

json

配饰文件小程序设置，如页面注册，页面标题及tabBar注册组件。

项目启动文件app.json

必须要有这个文件，如果没有这个文件，项目无法运行，因为微信框架把这个作

为配置文件入口，整个小程序的全局配置。包括页面注册，网络设置，以及小程.

序的window背景色，配置导航条样式，配置默认标题。

**162：页面内部有什么参数传值的方法？**

**答：**



**163：小程序两个页面之间怎样传递和接受数据？**

答：



**164：小程序和原生App的优势？**

**答:** 1.小程序除了拥有公众号的低开发成本、低获客成本低以及无需下载等优势

2.一次性开发跨平台使用

3.在服务请求延时与用户使用体验是都得到了较大幅度的提升，使得其能够承载跟复杂的服务功能以及使用户获得更好的用户体验。

4.有微信庞大的客户群作为依托利于项目的推广

**165：小程序是否可以做到长按识别二维码功能？**

答：1、小程序中长按识别二维码:这种方式目前不支持。至于其中原因,大家

可以猜得出,如果这个功能也开放出来, 从小程序向H5网页的调整入口就被打

刑，而目前小程序中唯一开放的嵌入 H5 ,只有Web-view一个组件,必须是

指定的安全域名下的网页,必须是HTTPS的安全连接,总之,束缚非常多。 但

换句话说,如果没有这些限制,小程序的概念也就空洞了。

2、长按识别小程序码:以上两个途径,分别在微信和小程序中都被支持的,

特别是在小程序中, 通过长按带有小程序码的图片,可以快速实现跳转到其他小

程序或者同一个小程序的另- -个界面,非常方便。

**166：小程序如何实现组件化？**

答：声明组件生成 js json wxml wss文件

注册组件

{

"navigat ionBarBackgroundColor": "#b3d4db" ，

"navigationBarTextStyle": "black",

"usingComponents": {

"cfooter": "/components/cfooter"

}

调用组件标签形 式调动

**167： 小程序组件和组件之间怎样实现通讯？**

组件的通信

父传子属性传递

属性必须在properties 里边进行声明

properties: {

pdata:{

type:String

|}

},

子传父发送事件

onTap: function () {

.

var myEventDetail = {} // detail 对象，提供给事件监听函数

var myEventOption = {} //触发事件的选项

this . triggerEvent( 'myevent, myEventDetail, myEventOpt ion)

**168：bindtap和catchtap有什么区别？**

Bind 会存在事件冒泡

catch阻止事件冒泡.

**169：简述小程序项目文件结构**

简叙小程序项目的文件结构?。

小程序: pages

index:

index.is页面逻辑)

/index.wxm!(页面结构)

/ndex.wss(顶面样式表)

/indexson(页面配置)

项目跟目录

App.js

小程序逻辑

App.json小程序公共设置

App.Wxss

小程序公共样式表

Project.config.ison

记录常用配置和安装包

**170:** **简述wx.navigateTo(), wx.redirectTo(), wx.switchTab(), wx.navigateBack(),**

**wx.reLaunch()的区别？**



**10月28日（下午）**

**171：小程序开发的优势**

1)容易上手，只要之前有HTML+CSS+JS基础知识，写小程序基本上没有大问题，当然如果

了解ES6+CSS3则完全可以编写出即精简又动感的小程序;

2)基本上不需要考虑兼容性问题，只要微信可以正常运行的机器，就可以运行小程序;

3 )基本组件库已经比较齐全:Toast,Loading框,Picker,定位及地

图,Image,Input,Checkbox,ext,TextAre-SrollView.等常用的组件都有，而且使用也挺简单、方

便;

4)发布、审核高效，基本上上午发布审核，下午就审核通过，升级简单，而且支持灰度发

布;

5)微信官方提供使用人数、频率等数据统计，小程序is脚本执行错误日志; .

6)开发文档比较完善，开发社区比较活跃;

7)支持组件化开发，

**172：如何分包加载？分包加载的优势在哪？**



**173：事件的绑定 有几种形式？**

Bind

会存在事件冒泡

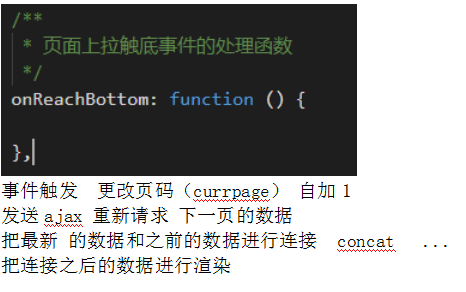
catch

阻止事件冒泡

capture-bind事件在捕获阶段触发

capture-catch捕获 只是执行最外层

**174： （上拉刷新）？**



**175：小程序怎样做到自适应？**

一般都以苹果6为基准

苹果6下1px=2rpx 用1:2去换算

Rpx是一个响应式的单位自动实现自适应

**176：请谈谈小程序的双向绑定和vue的异同？**

Vue . v-model

小程序直接this.data的属性是不可以同步到视图的，必须调用this.setData0方

法(数据改变同步到视图层)

Inputvalue视图改变 同步数据监听( onChange onInput)事件当数据改

变触发方法this.setData 把value值同步给数据

**177：请谈谈小程序项目的生命周期函数？**

onLoad()顶面加载时触发，只会调用一次。可获取当前页面路径中的参数。

Onshow页面显示/切入前台时触发，一般用来发送数据请求;

onReady() 页面初次渲染完成时触发，只会调用-次，代表页面已可和视图层进行交互

onHide()页面隐藏/切入后台时触发，如底部tab切换到其他页面或小程序切入后

台等。

onUnload()页面卸载时触发，如redirectTo 或navigateBack到其他页面时。

**178：** **请谈谈原生开发小程序、wepy、mpvue 的对比？**

个人认为，如果是新项目，且没有旧的h5项目迁移，则考虑用小程序原生开发，

好处是相比于第三方框架，坑少。而如果有老的h5项目是vue开发或者也有h5项目也需要小程序开发，则比较适合wepy或者mpyue 来做迁移或者开发，近期看wepy几乎不更新了，所以推荐美团的mpvue.

**179：请叙述 小程序中登录功能的实现流程**

前端wx.login() 获取code 码把code码发送给后台接口(939xx)

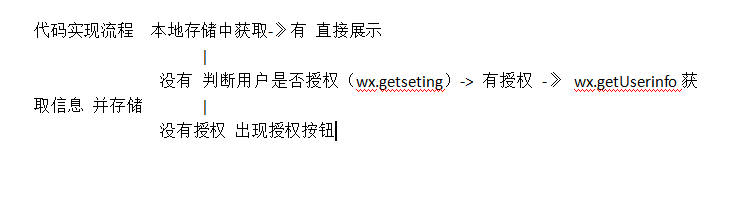
后台:拿到code+ epideappsecret>微信服务端返回(openid session\_ \_key)

给前端返回token|

前端:

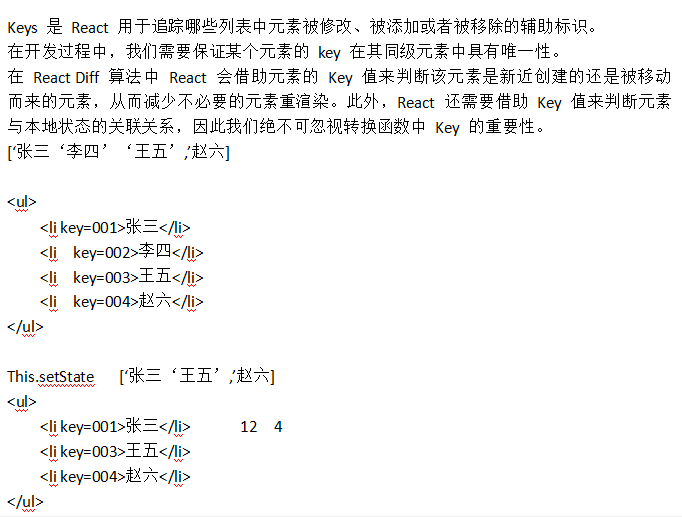
存储token每一 次请求都携带token

**180：小程序中怎样获取当前登录用户的信息  比如 头像 昵称**

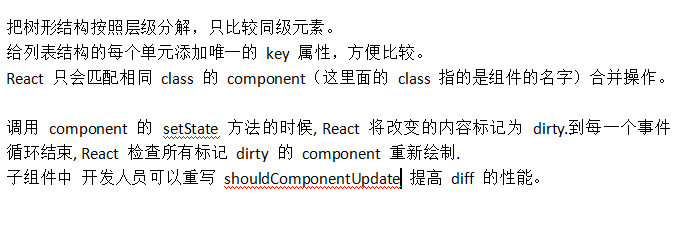


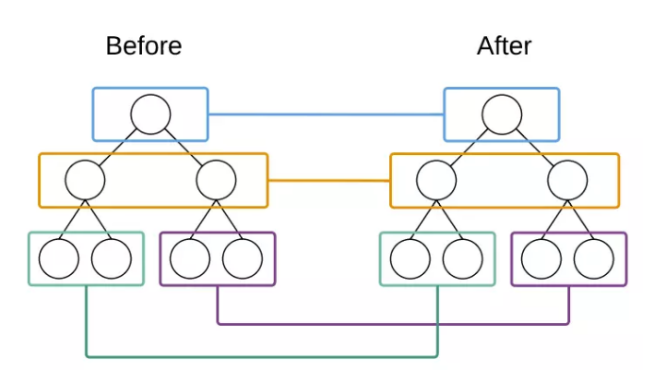
**10月29日（下午）**

**181:  React 中 keys 的作用是什么？**

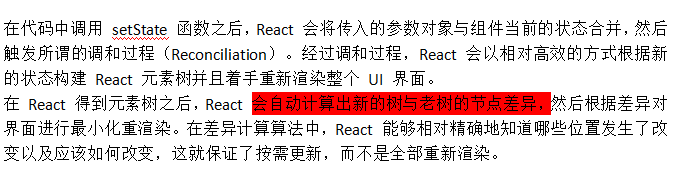


**182:** **react diff 原理（diff算法）？**

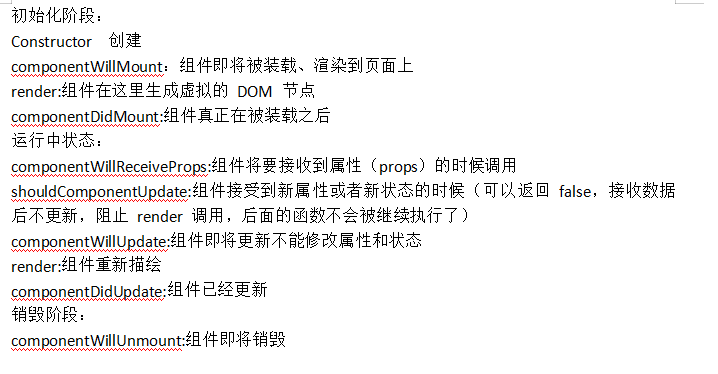




**183: 调用 setState 之后发生了什么？**



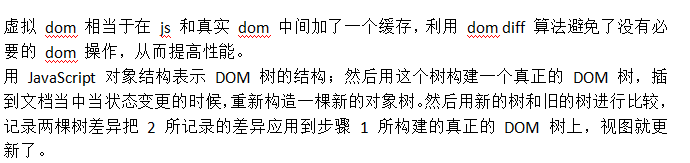
**184：react 生命周期函数**



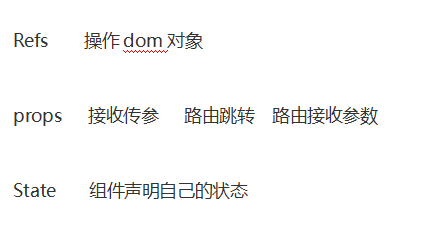
**185：shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？）**



**186：为什么虚拟 dom 会提高性能?(必考)**



**187：React组件的三大属性 及作用？**



**188：React  父子组件之间怎样传递参数？**

父传子 属性传递this. props.属性名接收

子传父 把父组件的方法传给子组件子组件给父组件的方法传值

**189：React 两个路由页面之间怎样传递参数？**

1动态路由配置传参

配置： path=’/detail/:id’

传参 ： to=’/detail/100’

接收： this.props.match.params.id

2 js跳转传参

  this.props.history.push({ pathname : '/user' ,query : { day: 'Friday'} })

  this.props.location.query.day

3   通过state同query差不多，只是属性不一样，而且state传的参数是加密的

this.props.history.push({ pathname:'/user',state:{day : 'Friday' } })

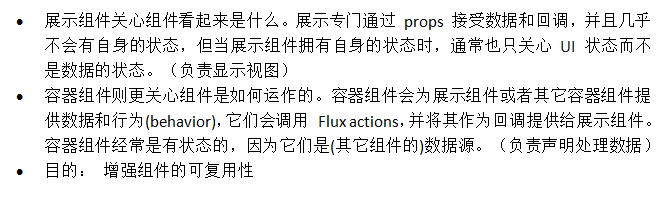
this.props.location.state.day

**190：redirect路由组件有什么作用？**

实现路由的跳转可以模拟实现路由守卫的功能

**10月29日（下午）**

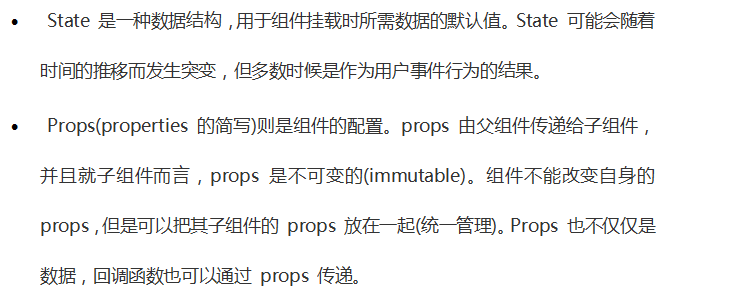
**191：展示组件(Presentational component)和容器组件(Container component)之间有何不同**



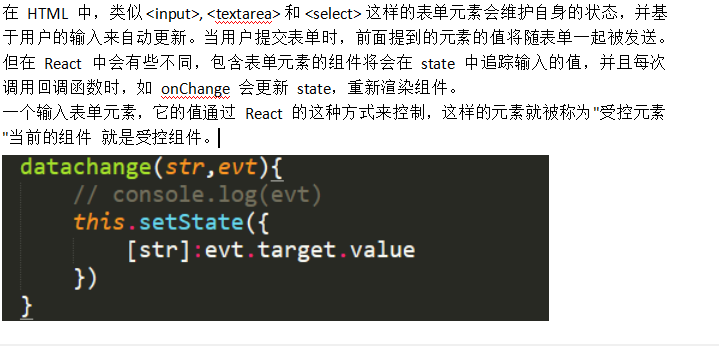
**192：类组件(Class component)和函数式组件(Functional component)之间有何不同**



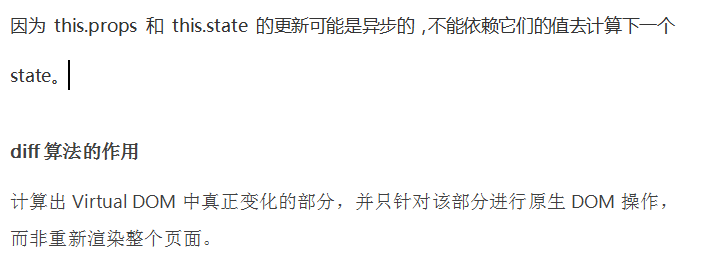
**193：(组件的)状态(state)和属性(props)之间有何不同**



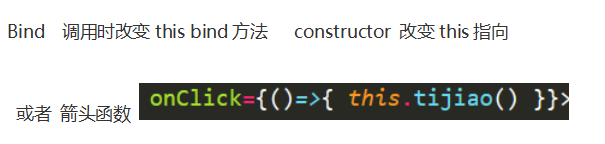
**194：何为受控组件(controlled component)**



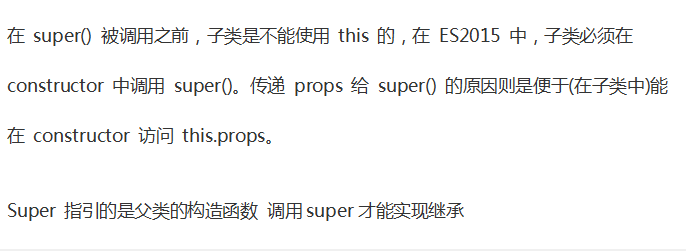
**195：为什么建议传递给 setState 的参数是一个 callback 而不是一个对象**



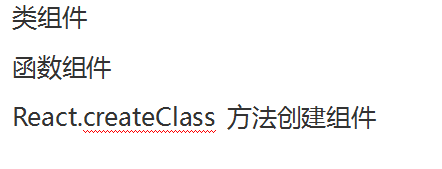
**196： 当我们调用方法时  this指向问题怎样修正？**



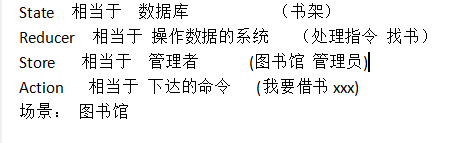
**197：(在构造函数中)调用 super(props) 的目的是什么**



**198：React 中有三种构建组件的方式**



**199：Redux有哪些核心特性？**



**200： React的一些主要优点？**

