阿里

测评题：

```

//实现mergePromise函数，把传进去的数组顺序先后执行，

//并且把返回的数据先后放到数组data中

const timeout = ms => new Promise((resolve, reject) => {

setTimeout(() => {

resolve();

}, ms);

});

const ajax1 = () => timeout(2000).then(() => {

console.log('1');

return 1;

});

const ajax2 = () => timeout(1000).then(() => {

console.log('2');

return 2;

});

const ajax3 = () => timeout(2000).then(() => {

console.log('3');

return 3;

});

const mergePromise = ajaxArray => {

// 在这里实现你的代码

var data = [];

var p = Promise.resolve();

ajaxArray.forEach(function(item){

p = p.then(item).then(function(res){

data.push(res);

return data;

});

})

return p;

};

mergePromise([ajax1, ajax2, ajax3]).then(data => {

console.log('done');

console.log(data); // data 为 [1, 2, 3]

});

```

// 分别输出

// 1

// 2

// 3

// done

// [1, 2, 3]

一面：

1.mvvm实现双向绑定

见github： 单项绑定，双向绑定介绍

2.vue获取dom

3.生命周期

4.父子间传递数据

5.虚拟dom

用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中

当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异

把2所记录的差异应用到步骤1所构建的真正的DOM树上，视图就更新了

6.如何实现单页应用

（路由，懒加载）

1.路由中配置异步组件:

component: resolve => require(['../page/my/my.vue'], resolve),//懒加载

2.在webpack配置文件中的output路径配置chunkFilename属性。

3.plugin:[new ExtractTextPlugin(...)]配置css

https://www.cnblogs.com/zhanyishu/p/6587571.html

7.ajax和jsonp区别，状态，能否截断

xhr.readyState

值 状态 描述

0 未初始化 尚未调用 open() 方法。

1 启动 已经调用open() 方法，但未调用send()方法。

2 发送 已经被调用send() 方法，但未接收到响应。

3 接受 已经接收到部分响应数据。

4 完成 已经接受到全部响应数据。

xhr.abort()取消异步数据

8.跨域

9.es6

1.const let

2.模板字面量（多行字符串，字符串占位符，标签模板）

3.函数（默认参数值，剩余参数，展开运算符，箭头函数，尾调用优化）

4.对象（Object.is,assign,对象字面量（属性初始值简写，方法简写，可计算属性名））

5.解构赋值

6.Symbol

7.Set,Map,WeakSet,WeakMap

8.迭代器，生成器

9 class

10.数组（Array.of,from,fill,find,findIndex,CopyWithin）

11.promise

12.代理与反射

13.模块

10.箭头函数：

没有this,super,arguments,new.target绑定

不能作为构造函数（不能new,没有原型）

不改变this,

不支持arguemnts参数

不支持重复参数命名

11.数组arr.map迭代截断

不能breack,continue,(return没用，forEach(return false或者true都是结束本次循环执行下一次循环)）

用for of就可以用break或者some,every(return true,return false)

12.现在如何学前端

13.为学前端做的准备

多益（offer)

1.最自豪的项目

2.web国际站点优化

不同语言展示不同页面

用谷歌翻译插件，

根据不同的local，返回不同的json数据，

用i18n插件

3.string判断

4.css优化

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： 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，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(scss）

5.首屏加载优化

缓存，图片懒加载，按需加载，压缩合并css,js,图片，首页静态化

6,不用第三方交换变量

7.内存泄漏

不再用到的内存，没有及时释放，就叫做内存泄漏

ie有bug，比如说闭包的时候不会垃圾回收，造成内存泄漏，（其实闭包不会造成内存泄漏）

8.看过的书

9.看待加班

10.了解多益。。

11.地区

百度(offer)

一面：

1.移动端适配,rem ,em

2.padding-top，magin-top的100%相对于最近父级块级元素的width

3.flex

flex-direction

flex-wrap

jusitify-content

align-items

align-content

flex-grow

flex-shrink

flex-basis

order

align-self

4除了flex实现垂直居中

5懒加载

img的src是loading图，data-src是真实地址，当图片进入视口，则把data-src的网址给src，并杀掉data-src

判断视口：

```

clientHeight = document.documentElement.clientHeight

元素Top = 元素.getBoundingClientRect().top;

```

6.domAPI

document.querySelector/All

第一个元素(对象)，所有的元素NodeList（对象）

伪数组变对象

Array.prototype.slice.call(obj)

[...obj] Array.from(obj)

判断数组

Array.isArray

Object.prototype.toString.call(arr) === [object Array]

value instanceof Array

7.说一下Array.filter和reduce(累计)

8.var let

(var可以重复声明，let 不可以会报错)

9.箭头函数

this,构造函数，arguments(以命名参数和不定参数访问)

10.promise解决了什么问题

11.讲一下promise，promise.all参数是什么，只要一个成功就行吗，Promise.resolve()

12.ajax和fetch区别

fetch和ajax 的主要区别

1、fetch()返回的promise将不会拒绝http的错误状态，即使响应是一个HTTP 404或者500

2、在默认情况下 fetch不会接受或者发送cookies(credentials: 'include')

fetch其实就是封装了ajax和promise（带详细解读promise实现原理）

13.event loop

promise.Trick()>promise的回调>setTimeout>setImmediate

```

async function async1() {

console.log("async1 start");

await async2();

console.log("async1 end");

}

async function async2() {

console.log( 'async2');

}

console.log("script start");

setTimeout(function () {

console.log("settimeout");

},0);

async1();

new Promise(function (resolve) {

console.log("promise1");

resolve();

}).then(function () {

console.log("promise2");

});

console.log('script end')

```

```

script start

async1 start

async2

promise1

script end

promise2

async1 end

setTimeout

```

await 后面若返回的是promise,则会将它放入promise队列之后再执行

14.模块化

CommonJS模块的重要特性是

1.加载时执行，即脚本代码在require的时候，就会全部执行。

2.输出是值的拷贝，即原来模块中的值改变不会影响已经加载的该值。

3.一旦出现某个模块被”循环加载”，就只输出已经执行的部分，还未执行的部分不会输出。（再开头就循环引用会报错）

ES6模块是

1.静态分析，动态引用。运行前判断得出模块之间的依赖关系，如果使用import从一个模块加载变量，那些变量不会被缓存，而是成为一个指向被加载模块的引用

2.输出的是值的引用，值改变，引用也改变，即原来模块中的值改变则该加载的值也改变。

3.ES6根本不会关心是否发生了"循环加载"，只是生成一个指向被加载模块的引用，需要开发者自己保证，真正取值的时候能够取到值

14.安全

xss ,csrf(scrf\_token,reffer)

15.webpack （tree shake：通常用于描述移除 JavaScript 上下文中的未引用代码(dead-code)）

16.vue a.b组件数据方法处理

17.vuex

二面：

1.http状态码 (301与302区别，301变成200会咋样)

2.200 from cache,304

3.捕获和冒泡，捕获冒泡都监听处理可以吗？

4.除ajax外异步请求有哪些

jsonp

5.es6 await和async和promise

6.两道编程题

1.

promise.all([p1(),p2()]).then(p3)

2.

```

function test(num) {

// TODO

return num.toLocaleString()

}

console.log(test(1234)); // 1,234

console.log(test(1234.567)); // 1,234.567

console.log(test(-123456789)); // -123,456,789

```

或者

```

function test(num) {

// TODO

if(typeof num !== "number"){

return

}

let arr = num.toString().split("")

let str = ""

let count = 0

for(let i=arr.length-1;i>=0;i--){

let val = arr[i]

if(count >= 3){

if(val!=='.'&&val !=='-'){

str=val+','+str

count = 1

}

if(val!=='.'&&val ==='-'){

str = val + str

}

if(val=== '.'){

str = val + str

count = 0

}

continue

}

if(count<3){

str = val+ str

count++

}

}

return str;

}

```

三面：

1.算法题

Given an array nums of n integers, are there elements a, b, c in nums such that a + b + c = 0? Find all unique triplets in the array which gives the sum of zero.

Note:

The solution set must not contain duplicate triplets.

Example:

Given array nums = [-1, 0, 1, 2, -1, -4],

A solution set is:

[

[-1, 0, 1],

[-1, -1, 2]

]

const fn = (nums) =>{

if(!(nums instanceOf Array) ){

return false

}

let len = nums.length

let res = []

let sum

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

let newArr = []

newArr.push(arr[i])

for(let j=0;j<len;j++){

newArr .push(arr[j])

for(let q=0;q<len;q++){

newArr .push(arr[q])

let sum = newArr.reduce((pre,cur)=>{

return pre+cur

},0)

if(sum===0){

res.push(newArr)

}

}

}

}

return res

}

1.编程题：3sum

添加判断条件

1.固定的值一样，跳过到下一个

2.第一个指针一样，跳过到下一个

（https://blog.csdn.net/zzxboy1/article/details/53763797）

2.web的缺点

（兼容性）css，js.babel

3.什么东西感觉让你感觉入前端门了

4.web开发中遇到的最大的困难

5.缓存

6.event loop （浏览器，node）

7.模块化的理解

CommonJS模块的重要特性是

1.加载时执行，即脚本代码在require的时候，就会全部执行。

2.输出是值的拷贝，即原来模块中的值改变不会影响已经加载的该值。

3.一旦出现某个模块被”循环加载”，就只输出已经执行的部分，还未执行的部分不会输出。（再开头就循环引用会报错）

ES6模块是

1.静态分析，动态引用。运行前判断得出模块之间的依赖关系，如果使用import从一个模块加载变量，那些变量不会被缓存，而是成为一个指向被加载模块的引用

2.输出的是值的引用，值改变，引用也改变，即原来模块中的值改变则该加载的值也改变。

3.ES6根本不会关心是否发生了"循环加载"，只是生成一个指向被加载模块的引用，需要开发者自己保证，真正取值的时候能够取到值

(EsModule的缺点) CommonJS（同步）EsModule可以动态加载import().then()

注：import 同步，import().then()异步

8.新的东西，如pwa，service worker之类的

1.Manifest实现添加至主屏幕

2.service worker实现离线缓存

3.serice worker实现消息推送

网易云音乐

网易面试有些记不得了，记得的写下来了。

一面：

1. 三个笔试题：

1.共有属性相关

2.忘了

3.setTimeout

2.vue 的get干了什么

（解析html的时候会生成观察者，watcher 会将自己添加到相应属性的 dep 中）

执行getter 判断当前有没有watcher(观察者)要订阅它（data:name）,有的话就把watcher（观察者）加到这个属性对应的订阅列表里，然后返回这个值

就是看看这个应用的data中有哪些是实际会被用到的，避免性能白白消耗。

(observer ,dep ,watcher,compile,view)

![](https://upload-images.jianshu.io/upload\_images/5722076-ac9faceb2ba5dd3f.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

何时订阅?（难点不是优点，优点是上面的依赖收集避免性能浪费） 设定一个currentobsever 。在需要看这个值的时候，我给他设置currentobsever = this，我再去调用它的属性，他会调用get，就会进行订阅。普通情况下。当用户去获取值的时候（下次）已经绑定了，就不会有currentobsever（每用一次设currentobsever = null）,只有再new时会有。

异步更新实现

Watcher实例调用update又会把自身放入一个队列中，在下一次事件循环中执行flushSchedulerQueue来一起更新。这一块是用自定义的nextTick函数来做的

3.commonjs和ES Module的区别

CommonJS模块的重要特性是

1.加载时执行，即脚本代码在require的时候，就会全部执行。

2.输出是值的拷贝，即原来模块中的值改变不会影响已经加载的该值。

3.一旦出现某个模块被”循环加载”，就只输出已经执行的部分，还未执行的部分不会输出。（再开头就循环引用会报错）

ES6模块是

1.静态分析，动态引用。运行前判断得出模块之间的依赖关系，如果使用import从一个模块加载变量，那些变量不会被缓存，而是成为一个指向被加载模块的引用

2.输出的是值的引用，值改变，引用也改变，即原来模块中的值改变则该加载的值也改变。

3.ES6根本不会关心是否发生了"循环加载"，只是生成一个指向被加载模块的引用，需要开发者自己保证，真正取值的时候能够取到值

(EsModule的缺点) CommonJS（同步）EsModule可以动态加载import().then()

注：import 同步，import().then()异步

4.promise.all(),中间拒绝会咋样(被catch到，只返回错误的那个值)

二面：

1.根据博客async和await实现并行操纵

await promise.all([p1(),p2(),p3()])

2.实现一个promise.all

```

Promise.all = arr => {

let aResult = []; //用于存放每次执行后返回结果

return new \_Promise(function (resolve, reject) {

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

arr[i].then(function (res) {

aResult.push(res); //执行后返回的结果放入数组

if (aResult.length == arr.length) {

resolve(aResult);

}

})

}

}

```

对比着阿里的mergePromise看

3.事件监听，捕获，冒泡，事件代理，及改进

DOM 0:onclick,DOM 2:addEventListener,IE事件：attchEvent

4.vue的缺点

数据写在data,props外不会双向绑定

写组件要在import和components两处写

v-model双向数据流不易追踪

5.根据博客写this

见你不知道的js上p86

6.博客缓存每种缓存中的两个的区别

分布式系统尽量关闭掉ETag(每台机器生成的ETag都会不一样）使用Last-Modified

为了解决一个缓存文件在短时间内被多次修改的问题, 因为Last-Modified只能精确到秒.使用ETag

3面 hr闲聊

猿辅导

一面：

1.vue双向绑定，解析过程，data中name更改是如何影响到{{name}}

2.再次登陆是登陆后的页面，原理。说一下cookie,session

说一下：cookie参数

path：表示 cookie 影响到的路径，匹配该路径才发送这个 cookie。

secure; https

httpOnly:浏览器不允许脚本操作 document.cookie 去读取更改 cookie；

expires 和 maxAge

3.fetch

Fetch 是浏览器提供的原生 AJAX 接口，使用了promise.

1.Fetch 请求默认是不带 cookie 的，需要设置 fetch(url, {credentials: 'include'})

2.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。

3.使用 fetch 无法取消一个请求

这是因为 Fetch API 基于 Promise，而 Promise 无法做到这一点

4。webpack

5.跨域,jsonp.cors,jsonp和cors场景区别。浏览器对于跨域的处理(options)

（自定义的头部）headers:NCZ

6.学习方法：书籍之类

7.es6

8.编程题：广度遍历

二面：

1.自我介绍

2.实习干了哪些事情

3.rebase（rebase指令，为啥会冲突）

交互式

1.git rebase -i HEAD ~3

2.p

s

3.修改为一个commit

4.wq保存并推出

不交互

git checkout test

git rebase master

while(存在冲突) {

git status

找到当前冲突文件，编辑解决冲突

git add -u

git rebase --continue

if( git rebase --abort )

break;

}

4.本科会计，跨专业。考研学了哪些

4.hash实现，时间复杂度，解决冲突

5.手写代码实现拓扑排序

依图科技

1.一个元素左红右绿（3种）

border 伪元素 渐变

background: linear-gradient(to right, #ff3232 0%,#ff2828 20%,#3eff30 20%,#3eff30 20%,#3eff30 100%);

2.说一下项目

3.call,apply,bind区别，实现bind,call

(https://blog.csdn.net/smallsun\_229/article/details/80721758)

```

function func() {

console.log(this.value)

}

var obj = {

value: 233

}

Function.prototype.mycall = function(obj,...res) {

obj = obj || window;

obj.fn = this; //这里的this就是调用mycall的func函数

var result = obj.fn(...res);

delete obj..fn;

return result

}

Function.prototype.myapply = function(obj,arr) {

obj = obj || window;

obj.fn = this; //这里的this就是调用mycall的func函数

var result

if(!arr) {

result = obj.func();

}else{

result = obj.fn(...arr);

}

delete obj..fn;

return result

}

func.mycall(obj); //233

```

4vue路由相关api

导航守卫：（全局的, 单个路由独享的, 或者组件级的）

全局的

router.beforeEach((to, from, next)

router.beforeResolve

router.afterEach((to, from)

单个路由独享的

beforeEnter: (to, from, next)

组件级

beforeRouteEnter(to, from, next)

beforeRouteUpdate(to, from, next)

beforeRouteLeave(to, from, next)

5\_\_proto\_\_和prototype区别

6.子数组最大

```

function FindGreatestSumOfSubArray(array)

{

if(array.length===0||!Array.isArray(array)){

return 0

}

var cursum = array[0]

var maxsum = array[0]

for(var i=1;i<array.length;i++){

if(cursum>0){

cursum = cursum+array[i]

}else{

cursum = array[i]

}

if(cursum>maxsum){

maxsum = cursum

}

}

return maxsum

}

```

7.click outside(wrapper,window监听，wrapper冒泡时阻止)

第四范式

一面：

1.自我介绍

2.重构，逻辑，node

3.\_proto\_,prototype区别（自己说的正确）

4.基本类型，复杂类型（说一下symbol）

Symbol()

不能new Symbol()

Symbol()返回值唯一

可作为对象的属性（用于创造一个隐藏属性（私有属性））

Symbol.iterator定义对象的默认迭代器（数组，map,set,,字符串）

5.闭包

6.get,post的区别

7.输入网址

（说了，还不错）

8判断类型（基本，复杂）typeof 特殊：null ,函数

补充：判断数组（3种）

9.内存泄漏(http://www.ruanyifeng.com/blog/2017/04/memory-leak.html)

不再用到的内存，没有及时释放，就叫做内存泄漏（memory leak）。

那些很占空间的值，一旦不再用到，你必须检查是否还存在对它们的引用。如果是的话，就必须手动解除引用。

及时清除引用非常重要

[WeakSet]和 [WeakMap]们对于值的引用都是不计入垃圾回收机制的，所以名字里面才会有一个"Weak"，表示这是弱引用。

新建引用的时候就声明，哪些引用必须手动清除，哪些引用可以忽略不计，当其他引用消失以后，垃圾回收机制就可以释放内存。这样就能大大减轻程序员的负担，你只要清除主要引用就可以了。

10.深拷贝和浅拷贝的区别（https://www.zhihu.com/question/23031215）

浅拷贝对于复杂类型只是拷贝了引用

11.事件模型（dom0/2 ie事件模型）

12.选择器

13.三角形

14.说一下行高

15.轮播图（3个循环）

说了循环队列。

增加切换左右按钮（反向shift,push,逆序播放）

（与我的类似https://segmentfault.com/a/1190000014953286）

还有一种方法（前后增加一个拷贝

https://juejin.im/post/5b31e703e51d4558842ac127

）

二面：

1.自我介绍

2.作用域

3.说一下作用域链

对象：

「读」属性时会沿着原型链搜索。

「新增」属性时不会去看原型链（但是如果你给这个属性加了一些配置，则不一样）？

4.判断构造函数是否在实例的原型链上

instanceof

isPrototypeOf（）

Object.getPrototypeOf（）

constructor

5.css选择器

6.说一下cookie的作用

![image.png](https://upload-images.jianshu.io/upload\_images/5722076-43ba21adbe714fe9.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

3 3 1

![image.png](https://upload-images.jianshu.io/upload\_images/5722076-20a9a7af3aaa36b9.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

1 3

1 1

(从构造函数再到原型链找)

![image.png](https://upload-images.jianshu.io/upload\_images/5722076-6fcf51820eabc4f2.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

1 1

1 3

（数组内obj1指向{x:1}一直没变）

![image.png](https://upload-images.jianshu.io/upload\_images/5722076-0b68b4538ded67ca.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

1 3 4

1. 找出一段英文文章中所有的重复出现的单词并统计重复单词出现的次数

hasOwnPropertyf防止类似constructor单词误记录，直接+1。

（空白用split()）

3面技术相关聊天

携程

一面：

聊天：基础面

二面：

1.混合开发原生和js的通信

同步还异步（为啥）

2.reactnative和混合开发有什么区别

3.前后端分离是怎么做的

4.逻辑处理会放在前端吗？如果是，前端哪里逻辑会复杂

5.一道js内存题

Keep（记得不多，就记了自己没回答好的）

1.js如何判断安卓还是苹果

var u = navigator.userAgent;

var isAndroid = u.indexOf('Android') > -1 || u.indexOf('Adr') > -1; //android终端

var isiOS = !!u.match(/\(i[^;]+;( U;)? CPU.+Mac OS X/); //ios终端

alert('是否是Android：'+isAndroid);

alert('是否是iOS：'+isiOS);

2.虚拟dom转化成真实dom

https://www.cnblogs.com/giggle/p/7538533.html

3.初始化一个链表

4.用户绑定

没做到绑定

5.localStorage溢出（ 5M）

告诉浏览器本地存储不够了，让他删除一些无用的商品

6.apicloud自定义loader

7.判断环的入口

8.css

动画·：

div {

background: #000;

width: 100px;

height: 100px;

animation: spin 3s linear infinite;

}

@keyframes spin {

0% {

transform: translate(0px,0px);

}

50% {

transform: translate(100px,0px);

}

100% {

transform: translate(100px ,100px);

}

}

第三个元素

nth-child(3)

每隔三个

nth-child(3n+1)

兄弟间

p:nth-of-type(2n+1)

li:first-child { ... }

li:last-child { ... }

body :not(p) { ... }

p:not(.warning) { ... }

5.从哪些方面提高网站的渲染速度？

缓存，css,js，图片的合并压缩，服务端开启gzip压缩，懒加载，预加载，函数防抖

服务端渲染

6.js脚本的异步加载

async defer

7.如何确保页面中的js加载完成呢

```

function loadScript（url , callback）{

var script = document.createElement("script");

script.type="text/javascript";

if(script.readyState){

script.onreadystatechange = function(){

if(script.readyState=="loaded"||script.readyState=="complete"){

script.onreadystatechange=null;

callback();

}

}

}else{

script.onload = function(){

callback();

}

}

script.src = url;

document.getElementsByName("head")[0].appendChild(script);

}

```

美团 （记录了一些有印象的）（offer)

一面：

xss后台如何处理，httponly,xssfilter

require如何寻扎模块

正则实现要求

二面:

fn(1)(2) === 3

除了promise还有哪些微任务

react function 和class区别

无状态函数式组件:是为了创建纯展示组件，这种组件只负责根据传入的props来展示，不涉及到要state状态的操作。

实现工厂模式

创建有状态组件的方式

```

function createPerson(name,age,job){

let o = new Object()

o.name = name

o.age= age

o.job= job

o.sayName= function(){

console.log(this.name)

}

}

```

co,generator模块原理

apicloud如何获取摄像头

diff算法：key的实现，手写

之字形打印：

```

function Print(pRoot)

{

// write code here

if(!pRoot){

return []

}

var queue = []

var res = []

var flag = true

queue.push(pRoot)

while(queue.length){

var arr = []

var len = queue.length

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

var node = queue.shift()

arr.push(node.val)

if(node.left){

queue.push(node.left)

}

if(node.right){

queue.push(node.right)

}

}

if(!flag){

arr.reverse()

}

flag = !flag

res.push(arr)

}

return res

}

```

3面

1.缓存

2.意见（实习太短，apicloud别用）

3.知道apicloud如何调用摄像头吗

去哪儿网（offer）

(主要手写了一些代码)

1.优化代码

2.promise任务

3.实现延迟更改

宜信（offer）

一面：

图的遍历

（DFS,BFS）

state变化触发的生命周期（vue,react）

shouldComponentUpdate()

componentWillUpdate()

render()

componentDidUpdate()

,

breforeupdate

updated

二面：

1.二叉树查找左子节点的和

2.数组局部最小数 复杂度o（n）以下

二分查找思想：

局部最小可以使用二分法完成，隐藏了一个在定义局部最小时就隐含的规律，即：一个不存在相同元素的数组一定存在局部最小值，因为只需要找到一个局部最小值，这样就可以使用二分查找将数组规模逐渐减小。具体过程如下：1. 如果 arr[mid]<arr[mid+1] && arr[mid]<arr[mid-1], arr[mid]为局部最小，返回2. 否则 arr[mid]不是最小，则必有 arr[mid+1]>arr[mid] 或是 arr[mid-1]>arr[mid],这时我们选择min(arr[mid+1],arr[mid-1])的一边形成一个新数组，这个数组也是不存在相同元素，而且边界条件也符合，所以肯定含有局部最小值。3.对新数组重复进行 1,2，直到找到局部最小即可

```

function findPeakElement(nums) {

int N = nums.length;

if (N == 1) {

return 0;

}

if (nums[0] > nums[1]) {

return 0;

}

if (nums[N - 1] > nums[N - 2]) {

return N - 1;

}

let low = 0, high = N - 1;

while (low <=high) {

int mid = (low + high) / 2;

if (nums[mid - 1] >nums[mid] && nums[mid] < nums[mid + 1]) {

return mid;

} else if (nums[mid - 1] < nums[mid] && nums[mid] < nums[mid + 1]) {

high = mid-1;

} else if (nums[mid - 1] < nums[mid] && nums[mid] > nums[mid + 1]) {

high = mid-1;

} else{

low = mid+1;

}

}

return -1;

}

三面：

1.灯泡找规律

2.为何要有同源策略，举例子

A网站是一家银行，用户登录以后，又去浏览其他网站。如果其他网站可以读取A网站的 Cookie，会发生什么？

很显然，如果 Cookie 包含隐私（比如存款总额），这些信息就会泄漏。更可怕的是，Cookie 往往用来保存用户的登录状态，如果用户没有退出登录，其他网站就可以冒充用户，为所欲为。

小米

一面：

1.懒加载，预加载如何做

<link rel="prefetch" href="https://i.loli.net/2017/08/08/5989307b6c87b.gif">

element.getBoundingClientRect().top;

2.动画 transition animation

3.闭包 内存泄漏 weakset

```

let set = new WeakSet()

key = {}

set.add(key)

console.log(set.has(key))

key = null

```

4.说一下cookie，

cookie的一些参数（domain，path.httponly,secure,expires,maxAge,）

5.vue生命周期，父子间通信

二面：

1.eventloop

为啥，setImmediate与setTimeout(fn,0)各自添加了一个回调函数A和timeout，都是在下一次Event Loop触发。那么，哪个回调函数先执行呢？答案是不确定

2.nodejs

timer..?

3.闭包

定义时确定作用域还是执行时确定作用域（定义）

静态作用域和动态作用域

4.promise内部如何实现的

5.async,await是如何实现的

generator co模块原理

哪些可迭代

如何使对象可迭代

6.rem

7.懒加载

8.函数节流和函数防抖

（节流说白了就是每ms执行一次函数，防抖就是 我最后一次触发后ms后执行一次回调函数。)

七牛云

1.hash链表太长如何解决

2.快排优化（有相同的）[1,1,2,2,4,6]，为啥要取中间值

3.数组和链表实现栈

4.上亿数据找出前10个

5.环的判断

6.udp能实现可靠传输吗，可以

7.LRU如何实现（链表和hash）

8.put和post区别（改，增）

头条

1.typeof NaN（number）

2.字符串len,传出

3.比较数组最大值[1,2,345,642,4341]

4.link与import区别

本质上，这两种方式都是为了加载CSS文件，但还是存在着细微的差别。

\* 差别1：老祖宗的差别。link属于XHTML标签，而@import完全是CSS提供的一种方式。

link标签除了可以加载CSS外，还可以做很多其它的事情，比如定义RSS，定义rel连接属性等，@import就只能加载CSS了。

\* 差别2：加载顺序的差别。当一个页面被加载的时候（就是被浏览者浏览的时候），link引用的CSS会同时被加载，而@import引用的CSS 会等到页面全部被下载完再被加载。所以有时候浏览@import加载CSS的页面时开始会没有样式（就是闪烁），网速慢的时候还挺明显（梦之都加载CSS 的方式就是使用@import，我一边下载一边浏览梦之都网页时，就会出现上述问题）。

\* 差别3：兼容性的差别。由于@import是CSS2.1提出的所以老的浏览器不支持，@import只有在IE5以上的才能识别，而link标签无此问题。

\* 差别4：使用dom控制样式时的差别。当使用javascript控制dom去改变样式的时候，只能使用link标签，因为@import不是dom可以控制的。

（@import会阻塞加载吧）

5.data数据不变(object.freeze)

```

const randomStr = (len) => {

//...

}

randomStr(15) //

typeof NaN

let arr = [1, 2, 90, 789, 110]

const maxNum = (arr) => {

//...

}

maxNum(arr) // 9078921110

link @import

new Vue({

//...

data: {

//.....

a: ,

}

})

```

```

const randomStr = (len) => {

let arr = [A-Z|a-z|0-9]

let i = Math.random(Math.floor(len))

let str = ''

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

str +=arr[i]

}

}

randomStr(15) //

let arr = [1, 2, 90, 789, 110]

const maxNum = (arr) => {

let len =arr.length

let str = ''

if(len === 1){

return arr[0]

}

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

let val = ''+arr[i]

for(let j=0;j<val.length;j++){

val[j]

}

}

}

maxNum(arr) // 9078921110

```

商汤（offer）

一面：

问了下项目

1.Dotype的作用，如何向前兼容

2.如何保存select选中的值，下次还是它。

localStorage.

说一下localStorage和cookie的区别（都是浏览器端存储）

大小，时间，携带，事件通知

3.说一下ajax,异步同步怎么做（true,false）

不刷新页面而局部更新内容

4.进度条

<div id="line"></div>

div{

border:1px solid black;

width:0;

}

let widthChange=document.body.clientWidth/10

let $div = document.getElementById('line')

setInterval(()=>{

$div.style.width = $div.style.width + widthChange +'px'

},1000)

5.数据嵌入

<ul>

<li></li>

<li></li>

<li></li>

</ul>

for(let i=0;i<user.length;i++){

user[i].name

user[i].age

}

二面

自我介绍

1.let var const

let 不可重复声明

2.post有哪些类型

（https://imququ.com/post/four-ways-to-post-data-in-http.html）

application/x-www-form-urlencoded

multipart/form-data

application/json

text/xml

3.js如何实现异步

4.三次握手，为啥三次

了防止已失效的连接请求报文段突然又传送到了服务端，因而产生错误。

主要目的防止server端一直等待，浪费资源

5.四次挥手，为啥四次

只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。

6.整一个http连接过程，立即断开吗

7.canvas和svg的区别

8.实现一个oading进度条

```

<div class="wrapper">

<div class="inner" id="inner"></div>

</div>

.wrapper {

width:300px

hight:40px;

}

.inner{

width:0

hight:40px;

background:black;

}

let $inner = document.getElementById('inner')

const fetchData = () =>{

fetch('http://baidu.com')

.then((data)=>{

return data.json()

}).then((json)=>{

$inner.style.width = $inner.style.width+json+'px'

if($inner.style.width >= '100px'){

clearTimeout(timer)

}

}).catch((err)=>{

console.log(err)

})

}

let timer = setInterval(()=>{

fetchData()

},300)

```

三面

1.骄傲的项目

2.技术项目

3.公司技术栈

四面经理

聊天，了解我的情况

Shopee（offer）

一面

1.es6

2.值拷贝，址引用

3.promise,eventloop

4.渲染流程（dom=>cssom=>render tree => 布局 => 渲染 => 合成（composite））display:none;在哪一步？

tab是线程还是进程，每个tan有哪些线程（js引擎，js渲染，setTimeout线程等）

流程优化有哪些（css书写规范，片段，dom读写不在一块）

Gpu加速，越多越好吗

5.数组内输出数字个数。arr.sort()用了那些排序算法

V8 引擎 sort 函数只给出了两种排序 InsertionSort 和 QuickSort，数量小于10的数组使用 InsertionSort，比10大的数组则使用 QuickSort。

快排 时间复杂度 最坏呢 稳定性（不稳）如何选中间值（越混乱越好）

6.框架异步更新原理 nextTick如何实现的 setState

7.js模块方案

AMD require.js异步

CMD

Common.js同步

ESModule

babel

区别：

Common.js模块输出的是一个值得复制，ES6模块输出的是值的引用

Common.js模块是运行时加载，ES6模块是编译时输出接口

8 插入数字实现回文，问所需最少个数

123 =》12321 2个

12231

9.动态规划 汉罗塔实现

```

main()

{

int n;

printf("请输入数字n以解决n阶汉诺塔问题：\n");

scanf("%d",&n);

hanoi(n,'A','B','C');

}

void hanoi(char A,char B,char C,int n)

{

if(n==1)

{

printf("Move disk %d from %c to %c\n",A,C,n);

}

else

{

hanoi(A,C,B, n-1);

printf("Move disk %d from %c to %c\n",n,A,C);

hanoi(B,A,C,n-1,);

}

}

```

10.防抖（throttle） 节流（debounce）

11.100元，空盒，空盒。愿意吗

二面

12.url输入网址发生什么

13.websocket（WS WWS）

comet

轮询

14.node提升性能

15.MVC

16.绘图性能优化（遍历，异步）

页面性能优化

17.CORS JSONP

18. css阻塞渲染，css阻塞js。js会阻塞解析和渲染

hr面

闲聊