关于算法，三面都给了编程题，印象比较深的题罗列如下∶

第一题∶给出一个网址如∶www.baidu.com，请反转成如下输出∶ com.baidu.www

再例如∶mail.toutiao.byte/action/hello 输出为∶ hello/action/byte.toutiao.mail

第二题∶两个超长的全是由数字组成的字符串

a="23423412323364562343….", b="34982988237987271653723."求和并输出。

第三题∶找出字符串中的全是字母构成的字串，并将该字串反转。例如∶ 3abdh34eer输出∶ 3hdba34ree

第四题∶ 找出数组中各个元素的后面比他大的第一个数字，例如∶ 3，4，7，1，2，8输出为∶4，7，8，2，8

算法题∶

1. 打印一个二叉树中，所有根节点到叶子节点的路径

2. 提供根节点到叶子节点的二维List数据，如何还原出对应的二叉树?

3. 在递增数组中，找到第一个值得key的元素，返回其位置信息，如在【1，1，2，3，4，4，5.6，6} 找到第一个4，应返回4。

4. 线程和进程的区别，为什么要设计进程、为什么要设计线程?

5. http1.0和http1.2的区别， http1.2和http2.0的区别?

6. https的原理，ssl运行在哪一层?

7. tcp和udp的区别

8. tcp如何保证对方接收数据成功

9. 冷启动优化是怎么做的

10. 找出数组中与给定数值最接近的数

11. findNext(num, rr)

12. css 实现两层边框

13. 实现事件监听和触发function

14. EventEmit(){}

15. 简要实现ASi的解析

算法题

1. 基础值、引用值

2. null和undefined区别

3. 深拷贝、浅拷贝区别

4.实现深拷贝

5.实现【a，b，c】 和 【b，d，e】 两个数组合并再去重

6.defineProperty实现简单的双向绑定

7.vue2.0和vue3.0双向绑定实现区别

8.Event Loop∶setTimeout和promise执行时机

9.最近的技术项目和业务项目

10.项目中难点等…

11.是否考虑考虑换工作，为什么.…

内容：

一面

1. 基本数据类型

2. 数组类型的检验

3. Promise的串行执行∶ 三个异步函数abc顺序执行。

4. map数据类型的使用

5.盒模型∶①含义"②宽度包含哪些属性。

6. cookies怎么使用的。

7. cookies跨域共享。

二面∶

1. 浏览器渲染机制

2. js和css资源下载和执行的顺序，是并行还是串行? 为什么?

3. 算法题∶ 根据两个数组【1，2，5，6】 和 【3，4，5，6】 ，得出一个结果【【1，2】，【3，4】】

4. vdom的算法实现∶

给你一个字符串"<div><div><p></p><div><span></span></div>"，写一个函数得出一个object的结果即ftag∶ diy. children∶ T】

5、最有挑战的项目∶遇到什么问题，怎么解决的，总结得到了什么。

三面∶

1. 项目架构怎么设计的

2. 印象最深的项目∶ 做了什么事情，得到了什么结果。

3. 团队情况，你的位置，对团队做了什么。

（注：基本上都是基于项目问的，问项目中某个方案具体怎么实现，为什么这么实现，有什么挑战，如何解决之类的（star模型）。

其他的包括∶

1. 如何实现一个自定义时钟组件

2、https的原理

3、网络安全（xss、csrf））

4、算法题∶实现快排算法，时间复杂度

5、算法题∶ n阶台阶，每次只能跨一个台阶或者两个台阶，求到达n阶台阶的所有可能性，进一步提问，如何优化

6、项目中如何选择react和vue

7、项目中如何做兼容性

8、服务端渲染以及同构

1面∶ http2;浏览器缓存策略;vue双向绑定; 实现Promise.all ; 实现debouncel

2面∶ 首屏时间优化策略;链表去重;打印二叉树层数

1、http与https

2、get、post的区别

3、请求头、响应头有哪些数据

4、eventloop相关的知识

5、/\*

给定⼀个只包括 '('，')'，'{'，'}'，'['，']' 的字符串，判断字符串是否有效

有效字符串需满⾜：

1. 左括号必须⽤相同类型的右括号闭合。

2. 左括号必须以正确的顺序闭合。

注意空字符串可被认为是有效字符串。

示例 1：

输⼊: "()[]{}"

输出: true

示例 2:

输⼊: "([)]"

输出: false

示例 3:

输⼊: "{[]}"

输出: true

\*/

1先说一下项目；结合项目问了一些实现方法

2、spa和多页面应用优缺点；

3、http2比http1新增哪些属性；

4、301，302，304状态码；

5、事件循环；

6、webpack配置；

7、keep-alive

8、js继承，原型，继承有几种方式

css：

页面布局

实现字符串的replace方法 - \_replace

例如 let str = ‘xiaomi jd com’

输出：xiaomi com

输入是：str.\_replace()

给定一个整数数组 nums 和一个整数目标值 target，请你在该数组中找出「和」为目标值 target 的两个整数，并返回它们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素在答案里不能重复出现。

你可以按任意顺序返回答案。

示例 1：

输入：nums = [2,7,11,15], target = 9

输出：[0,1]

解释：因为 nums[0] + nums[1] == 9 ，返回 [0, 1] 。

示例 2：

输入：nums = [3,2,4], target = 6

输出：[1,2]

示例 3：

输入：nums = [3,3], target = 6

输出：[0,1]

提示：

2 <= nums.length <= 104

-109 <= nums[i] <= 109

-109 <= target <= 109

只会存在一个有效答案

获取两个数组中不相同的元素

const arr1 = [0,1,2,3,4,5]

const arr2 = [0,4,6,1,3,9]

result: [2, 5, 6, 9]

A王康:

var x = 1,

y = 0,

z = 0;

var add = function (x) {

return x = x + 1;

}

y = add(x);

function add(x) {

return x = x + 3;

}

z = add(x);

console.log(x, y, z);

A王康:

给你一个字符串 time ，格式为 hh:mm（小时：分钟），其中某几位数字被隐藏（用 ? 表示）。

有效的时间为 00:00 到 23:59 之间的所有时间，包括 00:00 和 23:59 。

替换 time 中隐藏的数字，返回你可以得到的最晚有效时间。

输入 2?:?9

输出 23:59

??:??

?7:?8

[20,18,16,10,13,14,15] 输出 3

[3,2,1,2,3,4,5] 输出2

由大到小 再有小到大，找到中心点， 本来用for写的，说要优化一下时间复杂度

# 第一二轮

0. 自我介绍

1. TS里面Record类型是怎么实现的

2. 如何使用TS获取函数的返回类型

3. 当一个CSS文件很大的时候，如果文件没下载完成，js会不会执行

4. 算法：实现一个二叉树深度优先遍历

5. 说一个你熟悉的数据流管理方案，你对它的理解

6. mobx和redux的异同

7. immerJs的作用

8. 说一个你比较擅长的领域，做了哪些事情

9. 介绍一下浏览器性能优化怎么做

# 第三轮

1. 自我介绍

2. 说一个你参与的比较重要的项目，说一下你承担了什么角色，做了哪些事情，有什么结果

3. 设计题：使用MVP模式设计一个系统，包含首页feed流推送、用户发帖、内容详情展示，精细到class

4. 介绍了一下flutter，和RN相比的异同

5. 和外面的机会相比为什么会选择小红书

6. 对团队的期望

7. 有什么想问的

1. web/RN 如何做渲染优化？性能优化。

2. React Native interpolate 是否用过。

3. 如何在React Native 中实现 sticky。

4. 状态管理知道哪些？

5. redux的实现。

6. immer如何实现的？

7. webpack的处理流程？tapable的作用是什么？有什么设计意义

8. 以下代码 调用setCount的时候，child是否会渲染，如何做到让其不渲染。

```javascript

const Child = ()=> <div>I'm Child</div

let Parent = (props)=> <div>{props.children}</div>

Parent = React.memo(Parent)

const App = () => {

const [count, setCount] = useState(0)

return(

<div>

<button onClick={()=>setCount(count+1)}></button>

<Parent>

<Child></Child>

</Parent>

</div>

)

}

```

9. 实现 Promise.allSettled

10. 实现 PromiseBatchLoad

面试时间半小时

先问的项目 每个都问了 项目出发点 用到的技术栈 问了20分钟

题方面：

1、手写promiseAll 支持设置同时最大并发n个请求

2、Mep Set对象

3、vue3 新增技术点

4、vite

如何进行性能调优，比如当前页面滚动卡不卡，根据哪些指标来衡量· mpvue如何将vue代码变成小程序代码的。随便问了几个常见的问题，jsonp，原型，new，bfs遍历，判断是否是二叉搜索树，最大连续和。

- css position, bfc

- js变量提升，let声明，死区

- 输出结果

```

Function.prototype.a = () => alert(1)

Object.prototype.b = () => alert(2)

function A() {}

const a = new A()

a.a()

a.b()

```

- 输出结果

```

console.log(1)

setTimeout(() => {

console.log(2)

})

process.nextTick(() => {

console.log(3)

})

setImmediate(() => {

console.log(4)

})

new Promise(resolve => {

console.log(5)

resolve()

console.log(6)

})

.then(() => {

console.log(7)

})

Promise.resolve()

.then(() => {

console.log(8)

process.nextTick(() => {

console.log(9)

})

})

```

- 代码题

- 二叉树层序遍历

- promisify

- typescript

- Required

```

type Required<T> = {

[K in keyof T]-?: T[K]

}

```

- Pick

```

type Pick<T, K extends keyof T> = {

[P in K]: T[P]

}

```

- Webpack,Vite

现场编程题

题一和题四要求候选人独立阅读完成，候选人可以提问题，面试官回答。但面试官不讲解整个题目。

1.Tournament

Tally the results of a small football competition.

Tally the results of a small football competition. Based on an input file containing which team played against which and what the outcome was create a file with a table like this:

Team | MP | W | D | L | P

Devastating Donkeys | 3 | 2 | 1 | 0 | 7

Allegoric Alaskans | 3 | 2 | 0 | 1 | 6

Blithering Badgers | 3 | 1 | 0 | 2 | 3

Courageous Californians | 3 | 0 | 1 | 2 | 1

What do those abbreviations mean?

\* • MP: Matches Played

\* • W: Matches Won

\* • D: Matches Drawn (Tied)

\* • L: Matches Lost

\* • P: Points

A win earns a team 3 points. A draw earns 1. A loss earns 0.

The outcome should be ordered by points, descending. In case of a tie, teams are ordered alphabetically.

Input

Your tallying program will receive input that looks like:

Allegoric Alaskans;Blithering Badgers;win

Devastating Donkeys;Courageous Californians;draw

Devastating Donkeys;Allegoric Alaskans;win

Courageous Californians;Blithering Badgers;loss

Blithering Badgers;Devastating Donkeys;loss

Allegoric Alaskans;Courageous Californians;win

The result of the match refers to the first team listed. So this line

Allegoric Alaskans;Blithering Badgers;win

Means that the Allegoric Alaskans beat the Blithering Badgers.

This line:

Courageous Californians;Blithering Badgers;loss

Means that the Blithering Badgers beat the Courageous Californians.

And this line:

Devastating Donkeys;Courageous Californians;draw

Means that the Devastating Donkeys and Courageous Californians tied.

Your program should only process input lines that follow this format. All other lines should be ignored: If an input contains both valid and invalid input lines, output a table that contains just the results from the valid lines.

2 Ordered Link List

根据输入的数组中每项的 before/after/first/last 规则，输出一个新排好序的数组或者链表。要求，多解的情况可以只求一解，如果无解要求程序能检测出来。注意输入数组是无序的，before 和 after 规则不需要紧邻着指定的元素，只要满足是 before/after 即可。

示例 Input:

[

{id: 1},

{id: 2, before: 1}, // 这里 before 的意思是自己要排在 id 为 1 的元素前面

{id: 3, after: 1}, // 这里 after 的意思是自己要排在 id 为 1 元素后面

{id: 5, first: true},

{id: 6, last: true},

{id: 7, after: 8}, // 这里 after 的意思是自己要排在 id 为 8 元素后面

{id: 8},

{id: 9},

]

3 Create Tree from flat data

将输入的数组组装成一颗树状的数据结构，时间复杂度越小越好。要求程序具有侦测错误输入的能力。注意输入数组是无序的。

示例 Input:

[

{id:1, name: 'i1'},

{id:2, name:'i2', parentId: 1},

{id:4, name:'i4', parentId: 3},

{id:3, name:'i3', parentId: 2},

{id:8, name:'i8', parentId: 7}

]

4 Dominoes

Make a chain of dominoes.

Compute a way to order a given set of dominoes in such a way that they form a correct domino chain (the dots on one half of a stone match the dots on the neighbouring half of an adjacent stone) and that dots on the halfs of the stones which don't have a neighbour (the first and last stone) match each other.

For example given the stones 21, 23 and 13 you should compute something like 12 23 31 or 32 21 13 or 13 32 21 etc, where the first and last numbers are the same.

For stones 12, 41 and 23 the resulting chain is not valid: 41 12 23's first and last numbers are not the same. 4 != 3

Some test cases may use duplicate stones in a chain solution, assume that multiple Domino sets are being used.

Input example: (1, 2), (5, 3), (3, 1), (1, 2), (2, 4), (1, 6), (2, 3), (3, 4), (5, 6)

5 设计事件总线

请设计一个事件总线工具，要求满足以下条件，请尽量完成能完成的最高难度。

难度一：可以是 class 或者 非 class 的形式。有基本的监听、卸载监听、多重监听、触发等基本功能，可以传递参数。以下只是示例代码，可以自行设计 api。

const bus = new Bus()bus.listen('testEvent', (...argv) => {

console.log('event callback')})bus.trigger('testEvent', 1, 2)

难度二：在 listener 中可以继续触发事件，要求在总线对象内部要保持正确的事件调用栈(树形)，并能提供接口打印出来，例如:

bus.listen('testEvent', function callback1(){

// do something

this.trigger('testEvent2')})

bus.listen('testEvent2', function callback2(){

// do something})

bus.trigger('testEvent')/\*\* 设计 api 和数据结构并打印出这次 trigger 内部所有发生的事件和监听信息 \* 期望得到的结果是一个树形结构，描述着触发的事件、事件的响应函数以及响应函数中再触发的事件。例如： \* event: testEvent \* |-callback: callback1 \* |-event: testEvent2 \* |--callback: callback2 \* \* 注意，bus.trigger 应该可以执行多次，每一次trigger 都应该得到一个独立的事件栈。 \*/

难度三：增加对 async callback 的支持，并要求仍然能够正确打印出调用栈。增加对无限循环调用事件的判断。 注意要求是调用 callback 时，如果碰到 async callback，正常调用，不要阻塞。在 async callback 中再触发的 event 要在事件栈中的正确位置。

SEO是什么，React如何做的？

event-loop事件流是如何工作的?

node全局异常捕获?

CSS-BFC是什么?

https为什么更安全?

webpack HMR?

HTTP状态码

自己最大的优势是什么?

/\*\*

\* 1. 请实现一个 querystring 的编码函数

\* 参考文档：https://nodejs.org/dist/latest-v14.x/docs/api/querystring.html

\*/

/\*\*

\* 2. 实现一个 lodash.get 函数

\* 参考文档：https://lodash.com/docs/4.17.15#get

\*/

/\*\*

\* 3. 请实现一个 Node.js 中的事件模型（EventEmitter）,要求包含 on、off、once、emit 四个方法

\* 参考文档：https://nodejs.org/dist/latest-v14.x/docs/api/events.html

\*/

## vue响应式原理

## $nexttick 和 setTimeout 哪个先执行？为什么？

## computed 实现原理

## 前端性能优化做过哪些

## 浏览器缓存机制？cache-control 设置 no-cache 和 no-store 的区别

## vue生命周期 beforeMounted 和 created 区别是什么？

## v-model实现原理

## 编程题

```js

// var repeat = makeRepeat(console.log, 4, 3000)

// repeat("hello world")

// // 请实现一个makeRepeat方法领上面repeat之后可以每隔3秒打印"hello world"一次，一共要打印4次。

// function makeRepeat(fn, times, waits) {

// // 实现方法

// }

```

```js

function person(str) {

// 实现方法

}

person('abc').execute()

// 打印'abc'

person('abc').do('df').execute()

// 打印'abc'

// 打印'df'

person('abc').wait(2000).do('df').execute()

// 打印'abc'

// 等2秒

// 打印'df'

person('abc').wait(2000).do('df').wait(1000).do('gh').waitFirst(3000).execute()

// 等3秒

// 打印'abc'

// 等2秒

// 打印'df'

// 等1秒

// 打印'gh'

```

1、// 样例：obj = {a:{b:{c: 1}}}

// findValue(obj, 'a.b.c') // 1

// findValue(obj, 'a.c.b') // undefind

function findValue(obj, path) {

// 待写

}

2、输出结果

var a = 0, b = 0;

function A(a) {

A = function (b){

console.log(a+b++);

}

console.log(a++);

}

A(1);

A(2);

3、输出结果

function Foo() {

getName = function () {

alert(1);

};

return this;

}

var getName;

function getName() {

alert(5);

}

Foo.getName = function () {

alert(2);

};

Foo.prototype.getName = function () {

alert(3);

};

getName = function () {

alert(4);

};

Foo.getName();

getName();

Foo().getName();

getName();

new Foo.getName();

new Foo().getName();

new new Foo().getName();

4、防抖和截流，手写截流

5、邮箱的正则表达式

6、项目：

1）微前端选型、子应用更新的版本控制

2）监控系统，怎样埋点