2018跳槽面试经历

创人信息科技有限公司

> 首先自我介绍，然后问了两个问题；
> `如何画一个圆角（canvas）`
> `用户上传了一张800*400的图片，用户截取400*400，怎么截取？`
> 本人面试react方向的，但是相关问题没有问，（他说他也不懂react）

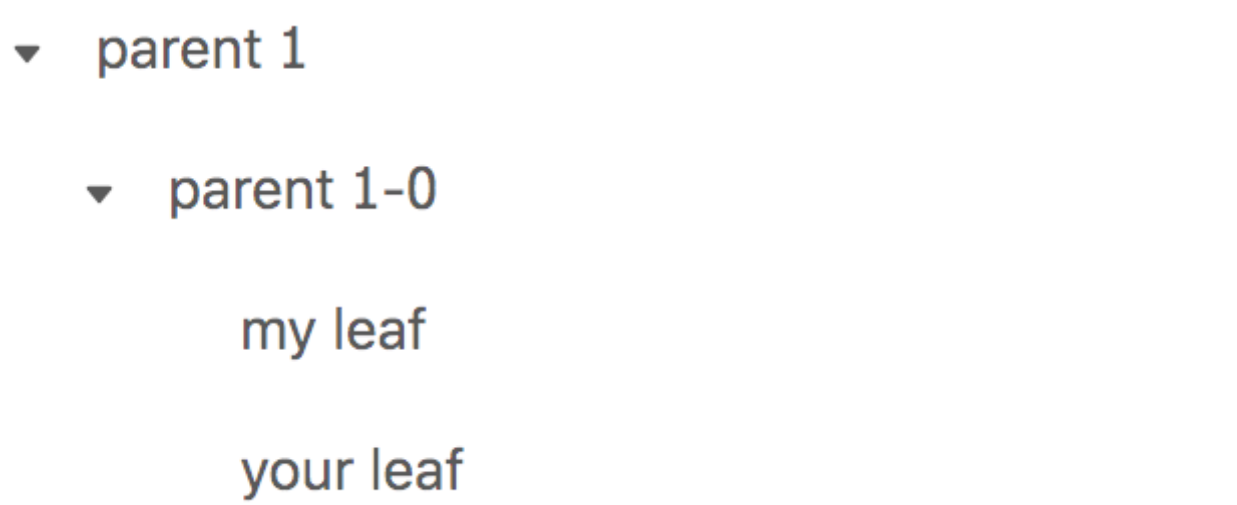
上机两道题

> `画一个Switch开关`

> `一个json字符串做一个tree` json字符串的格式如下

```
//需求是根据pid来判断一级还是二级tree（例如pid: 1-0 是父级 pid:1-1是子集，数据是无限的没有规定好是多少级的tree）面试官说用闭包可以做
    let tree = [
    {
        id:20,
        pid:1-0,
        name:menu1
    },
```

```
    {
        id:21,
        pid:1-1,
        name:menu1
    },
    {
        id:22,
        pid:2-0,
        name:menu1
    },
    {
        id:23,
        pid:2-1,
        name:menu1
    }
    ]
```



> 面试评价 :公司规模不算大，前端团队还没成立，面试官对前端技术不懂（省略好多字 ，前端人能多体会吧不都说）。

久誉软件

> 首先自我介绍（印象蛮深的，因为一下来了三个人）
> 问了一下项目的背景，项目流程，
> 看了一下项目地址，
> 项目的页面总共多少?
> 花费的时间多少和开发团队人数，人员如何分配的?
> 技术相关问题几乎也是没怎么问，（三个人冲人数吓人呀 ）
> 聊天聊得还行，也是等人事消息

撷（xie）知教育

> 首先自我介绍
> 这次直接问了react技术问题但是被虐的一塌糊涂，下面一一道来

为什么选择react，如何看待react的

> 个人思想是：
>
> - [ ] 框架是否成熟,社区大小以及社区支持是否充足，解决问题上，迭代后有没有大的区别，文档是否健全
> - [ ] 你是否喜欢API的风格，代码执行速度性能如何
> - [ ] 支持跨原生的性能难度多少，是否依赖性
>
> `我们说VirtualDOM提升了React的性能，但这并不是React的唯一亮点。此外，VirtualDOM 的渲染方式也比传统 DOM 操作好一些，但并不明显，因为对比 DOM 节点也是需要计算资源的。它最大的好处其实还在于方便和其他平台集成，比如 react-native 是基于 Virtual DOM 渲染出 原生控件，因为 React 组件可以映射为对应的原生控件。在输出的时候，是输出 Web DOM，还 是 Android 控件，还是 iOS 控件，就由平台本身决定了`。
> **JSX语法-json书写**

```
const Button = ({ color, text }) => { return {
          type: 'button',
          props: {
          className: `btn btn-${color}`,
           children: {
                     type: 'em', props: {
                      children: text,
                      },
             }, },
         }; }
```

jsx特别属性

> 关键字 class -className for-htmlFor
> **组件属性一般都是自定义属性，指就是props；也可以理解是组件所需要的参数**

Flux对比redux区别

> 一种是思想，redux是基于flux实现的实例（作者Dan说react解决不了的问题就用redux，建议不复杂的组件不建议用，一个组件依赖于另一个组件的数据，简单说就是不能滥用）

redux中有几个store

只能有一个store唯一的性。

redux中的基本原理

> 看看阮一峰博客深入redux

画一个圆怎么画

> canvas 简单说了原理 （最后说深入canvas不会）

before after 作用

> 是伪元素，用来实现一些小的icon ， 之前用来实现清除浮动的，当块元素使用

5个人做了5个零件花费了5分钟，100人做100个零件花费了多久时间
这几个把我直接问的头蒙蒙的，我算是结结巴巴都回答了。

> 面试官语速特别快，问他问题后，回答模糊，看不起人一样，`就一直问,还有啥问题,啥问题`

恒格信息科技有限公司

第一次面试:

> 三个人面试的（个人评价这组面试比较好）

第二次复试（4天后）

> 总经理面试 从学历问起，学的什么？哪些专业主修什么？栈内存和堆内存区别、人生规划。薪资要求

提供GitHub账号

> 提供

项目搭建流程

> 架构和客户沟通后画原型后，交接主要项目的思想和功能，和后台商议制定接口，编写开发文档

props 和state 区别

props和state之间是紧密相关的。父组件的state常常转变子组件的props
props：一般用于组件向组件通信，在组件之间通信使用。
state: 一般用于组件内部的状态维护，更新组建内部的数据，状态，更新子组件的props等。
React 的单向数据流，主要的流动管 道就是 props。props 本身是不可变的。
简单理解：父组件用props给子组件传递数据，而子组件的不能改变父组件的Props，如何改变子组件的Props，只有一种方法就是改变父组件的state。

除了改变状态和属性，还可以用什么方法，触发render

promise

首先介绍作用：解决的是异步的问题，比如ajax请求数据嵌套太深，维护太难。
说明三个状态 等待 => 成功 ;等待 => 失败
Promise是一个构造函数，自己身上有all、reject、resolve这几个眼熟的方法，原型上有then、catch等同样很眼熟的方法

```
> console.dir(Promise)
  ▼ function Promise()
    ▶ accept: function accept()
    ▶ all: function all()
      arguments: (...)
      caller: (...)
    ▶ defer: function defer()
      length: 1
      name: "Promise"
    ▼ prototype: Promise
      ▶ catch: function catch()
      ▶ chain: function chain()
      ▶ constructor: function Promise()
      ▶ then: function then()
        Symbol(Symbol.toStringTag): "Promise"
      ▶   proto  : Object
    ▶ race: function race()
    ▶ reject: function reject()
    ▶ resolve: function resolve()
    ▶   proto  : function ()
    ▶ <function scope>
```

```
var p = new Promise(function(resolve, reject){
    //做一些异步操作
    setTimeout(function(){
        console.log('执行完成');
        resolve('随便什么数据');
    }, 2000);
});
p.then(reason => {
//成功后操作
}).then(reject => {
//失败
}).catch(reason => {
//捕获
    console.log('rejected');
    console.log(reason);
```

```
16. });
```

ES6更新哪些
箭头函数
this 指向问题 对比es5区别

> **普通函数中的this:**
>
> 1. this总是代表它的直接调用者, 例如 obj.func ,那么func中的this就是obj;
> 2. 在默认情况(非严格模式下,未使用 ‘use strict’),没找到直接调用者,则this指的是window
> 3. 在严格模式下,没有直接调用者的函数中的this是 undefined
> 4. 使用call,apply,bind(ES5新增)绑定的,this指的是 绑定的对象
>
> **箭头函数有几个使用注意点。**
> （1）函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象。
> （2）不可以当作构造函数，也就是说，不可以使用new命令，否则会抛出一个错误。
> （3）不可以使用arguments对象，该对象在函数体内不存在。如果要用，可以用 rest 参数代替。
> （4）不可以使用yield命令，因此箭头函数不能用作 Generator 函数。

数组深度克隆
两种方式

> JSON 的两个方法 先转字符串，在转换成对象
> 使用jq的$.extend(true,target,obj)

改变this的方法，call，bind 区别

> call是直接执行改变this， 不用通过事件的触发
> bind 需要事件触发在进行改变this（有种预处理的思想）

跨域问题

> JSONP CORS NODE设置反向代理
> **router遇到的问题**
> 整体都回答ok了感觉回答的没有任何毛病，三个人都问了，最后就说等人事这块。技术过关了。

上海斯干网络科技（第一弹）

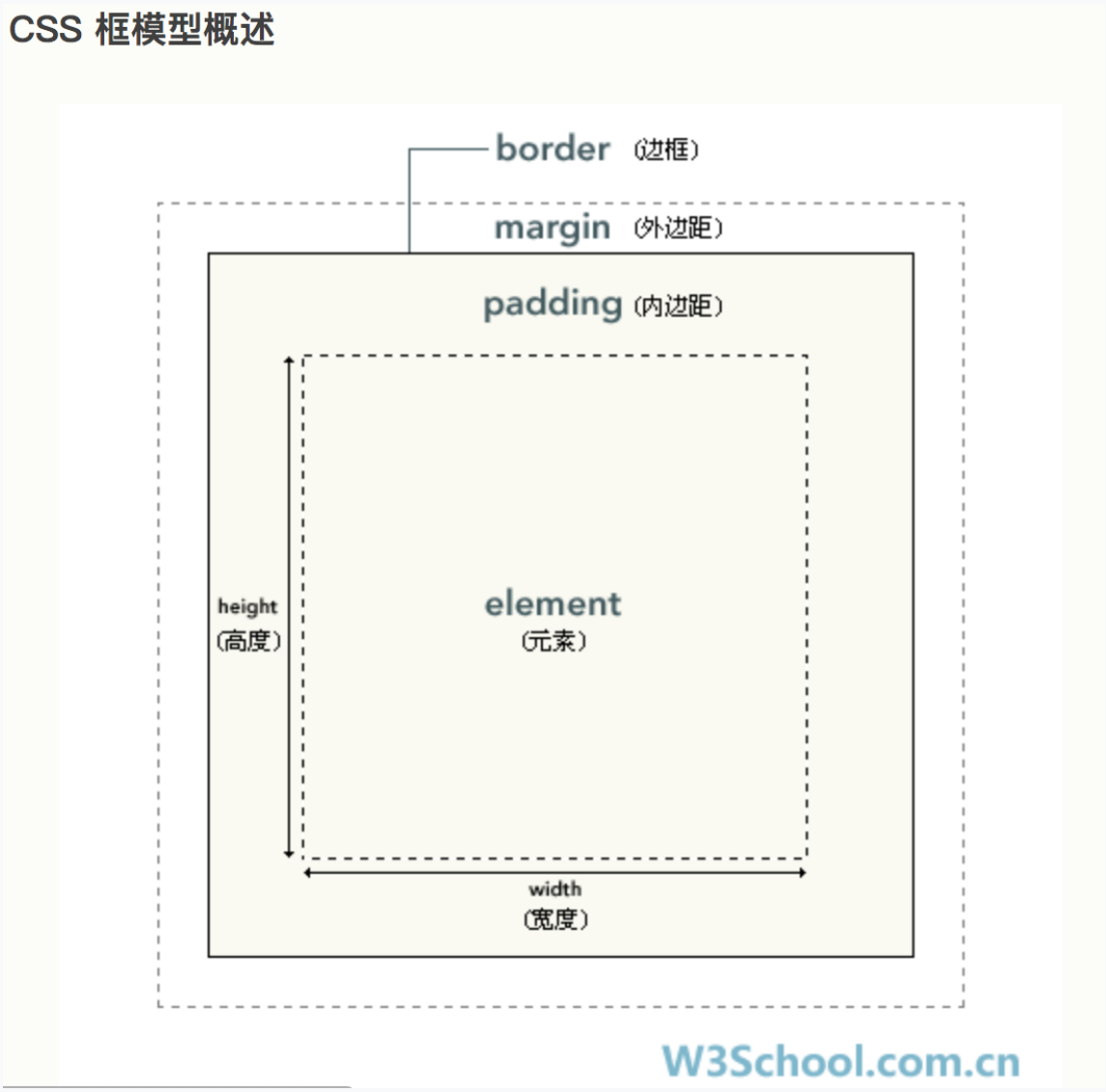
一面 js基础和react基础 。
二面CEO面试谈优化网站思想，感觉那个网站最好好在哪里（我说了鹅厂视频原因是客户进去无卡顿，并发做得好）谈开发和维护那个比较重要，谈理想，人生规划
三面 人事谈薪资 查学历（省略。。。）

BFC

中文常译为块级格式化上下文。是 W3C CSS 2.1 规范中的一个概念，它决定了元素如何对其内容进行定位，以及与其他元素的关系和相互作用。
1、浮动元素，float 除 none 以外的值；
2、绝对定位元素，position（absolute，fixed）；
3、display 为以下其中之一的值 inline-blocks，table-cells，table-captions；
4、overflow 除了 visible 以外的值（hidden，auto，scroll）
http://blog.csdn.net/riddle1981/article/details/52126522

css盒子模型

CSS 框模型 (Box Model) 规定了元素框处理元素内容、内边距、边框 和 外边距 的方式。



事件区别（冒泡阶段）event对象

> 1.事件池的处理机制不同
> 2.语法糖不同，相同事件绑定方法个数不同
> 3.注销方式不同 （dom0 直接赋值null，dom2要元素，事件类型，绑定方法三者相同才可以移除）

`DOM0事件绑定的原理`

> 1、给当前元素对象的 `某一私有属性` （onxxx这样的私有属性） `赋值的过程` （之前属性默认值是null，如果我们给 `赋值一个函数` ，相当于绑定了一个方法）
>
> 2、当我们赋值成功（ `赋值一个函数` ），此时浏览器会把DOM元素和赋值的函数 `建立关联` ，以及 `建立DOM元素行为操作的` 监听，当某一个行为被用户触发，浏览器会把相关行为赋值的函数执行

`DOM0特点` :

> DOM元素天生拥有这个私有属性（onxxx事件私有属性），我们赋值的方法才叫做 `事件绑定` ，否则属于给当前元素设置一个 `自定义属性而已`

`DOM2事件绑定的原理`
1、我们DOM2事件绑定使用的addEventListener/attachEvent都是在 `EventTarget` 这个内置类的原型上定义的.

> 我们调取使用的时候，
> 首先通过原型链找到这个方法，
> 然后执行完成事件绑定的效果

2、浏览器首先会给 `当前元素` 的 `某一个事件行为` 开辟一个事件池（事件队列）

> [其实是浏览器有一个统一的事件池，我们每个元素的某个行为绑定的方法都放在这个事件池中，只是通过相关的标识来区分的];
> 当我们通过addEventListener做事件监听的时候，会把绑定的方法存放在事件池中

3、当元素的某一个行为触发，

> 浏览器会到对应的事件池中，
> 把当前存放在事件池中的所有方法，
> 依次按照存放的先后顺序执行

4.dom2特点：

> 1、所有DOM0支持的`事件行为`，DOM2都可以使用，不仅如此，DOM2还支持一些DOM0没有的事件行为：`DOMContentLoaded`

前端关注安全问题

> “安全”是个很大的话题，各种安全问题的类型也是种类繁多。如果我们把安全问题按照所发生的区域来进行分类的话，那么`所有发生在后端服务器、应用、服务当中的安全问题就是“后端安全问题”`，所有`发生在浏览器、单页面应用、Web页面当中的安全问题`则算是“前端安全问题”。比如说，`SQL注入漏洞发生在后端应用中`，是后端安全问题，`跨站脚本攻击（XSS）则是前端安全问题`，因为它发生在用户的浏览器里。
>
> ## 8大前端安全问题
>
> 按照上面的分类办法，我们总结出了8大典型的前端安全问题，它们分别是：
>
> - 老生常谈的XSS
> - 警惕iframe带来的风险
> - 别被点击劫持了
> - 错误的内容推断
> - 防火防盗防猪队友：不安全的第三方依赖包
> - 用了HTTPS也可能掉坑里
> - 本地存储数据泄露
> - 缺失静态资源完整性校验
>
> - XSS是跨站脚本攻击。如何防范例如script标签中不能存放数据，css数据存放等等
>   如今的前端开发框架基本上都默认提供了前端输出编码，这大大减轻了前端开发小伙伴们的工作负担。
> - 有些时候我们的前端页面需要用到第三方提供的页面组件，通常会以iframe的方式引入。典型的例子是使用iframe在页面上添加第三方提供的广告、天气预报、社交分享插件等等。
>   解决方案 iframe有了一个叫做sandbox的安全属性，通过它可以对iframe的行为进行各种限制，充分实现“最小权限“原则。
> - CSRF是Cross Site Request Forgery，翻译为跨站请求伪造。CSRF的概念很容易和XSS混淆。CSRF和XSS攻击都是发起各种请求，但对CSRF来说，请求是来源于其他网站的，即为跨站的请求。

假设网站a有个页面是通过GET请求来删除数据的，使用的URL如下：

```
http://www.a.com/del?id=21
```

攻击者就可以利用这一点，构建一个页面并创建一个指向此链接的iframe、img或者script等标签。相当于伪造了一个GET请求。

此后，攻击者把新构建页面的地址发布出去，添加一些吸引眼球的消息，诱骗目标用户打开此页面。用户打开此页面就相当于间接地完成了删除数据的操作。

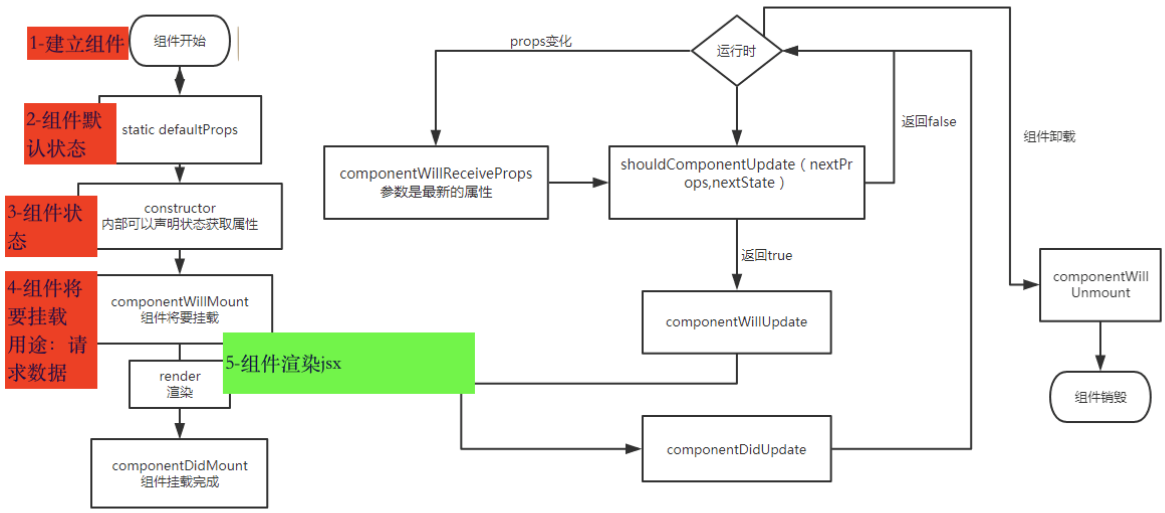
可以看到这个CSRF攻击的过程明显不同于XSS攻击，这个攻击可以没有任何的JavaScript参与。当然，如果想要利用JavaScript脚本代码也是可以的，比如利用JavaScript代码来动态构建form表单，并发起一个针对目标网站的POST请求，从而达到攻击目标网站的目的。

1. 攻击者精心构造一个诱导用户点击的内容，比如Web页面小游戏
2. 将我们的页面放入到iframe当中
3. 利用z-index等CSS样式将这个iframe叠加到小游戏的垂直方向的正上方
4. 把iframe设置为100%透明度
5. 受害者访问到这个页面后，肉眼看到的是一个小游戏，如果受到诱导进行了点击的话，实际上点击到的却是iframe中的我们的页面

点击劫持的危害在于，攻击利用了受害者的用户身份，在其不知情的情况下进行一些操作。如果只是迫使用户关注某个微博账号的话，看上去仿佛还可以承受，但是如果是删除某个重要文件记录，或者窃取敏感信息，那么造成的危害可就难以承受了。

解决方案:有多种防御措施都可以防止页面遭到点击劫持攻击，例如Frame Breaking方案。一个推荐的防御方案是，使用X-Frame-Options：DENY这个HTTP Header来明确的告知浏览器，不要把当前HTTP响应中的内容在HTML Frame中显示出来

react生命周期



react的redux实现原理

react-redux是一个轻量级的封装库，核心方法只有两个：

- Provider:
  在原应用组件上包裹一层，使原来整个应用成为Provider的子组件
- connect:连接 React 组件与 Redux store。

- connect([mapStateToProps], [mapDispatchToProps], [mergeProps], [options])（组件本身）

前端优化react方向

> 代码分割（通过webpack 配置 再有路由分配相关组件引入）
> 拆解无用js 减少打包大小（运用插件 webpack-bundle-analyzer 在浏览器查看）；减少打包时间（插件 Webpack.DDLPlugin， HappyPack 来提高构建速度）
> 避免更新 DOM 就是减少render重复渲染（shouldComponentUpdate生命周期函数编写中间件）

react请求后台数据在那个生命周期中触发

> 我在项目中是把数据请求单独定义一个函数，在进入页面是didMount中加载这个函数，如果有数据更新（也就是你说的刷新列表触发事件）时，再执行这个行数即可

组件如何通信

> 分为三类
>
> - [x] 父传子
> - [x] 子传父
> - [x] 嵌套深层
> - [x] 组件之间通讯

React 数据流

> 从顶向下的单向数据流，即从父组件到子组件；

红倍心（标注把面试官骂了）

> 面试问了一些基础题 css jq

马上办公（要全站开发要使用vue先把页面搭建起来，在用node写后台）

> 做了一套面试题
> require.js 作用、 获取dom 操作dom元素 、增加元素、删除元素、克隆元素、数组的常用方法、字符串常用方法、谈谈 this、link和import 区别、谈谈html5新特性
> react 组件通信有哪些

上海捷信医药科技股份有限公司

> 架构师面试（竟然不懂打包如何优化，也不知道掘金是什么，让我去翻译英文文档（一万个草泥马））出了一道找出数组中最大的和出现的次数，如果出现一次返回0

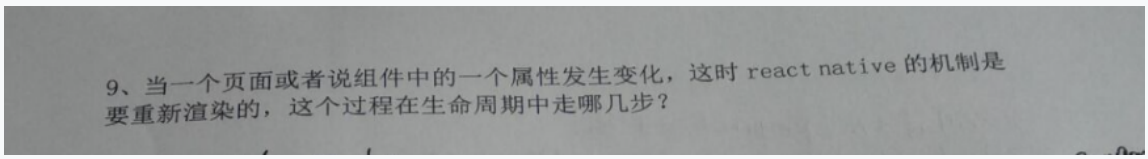
掌门一对一

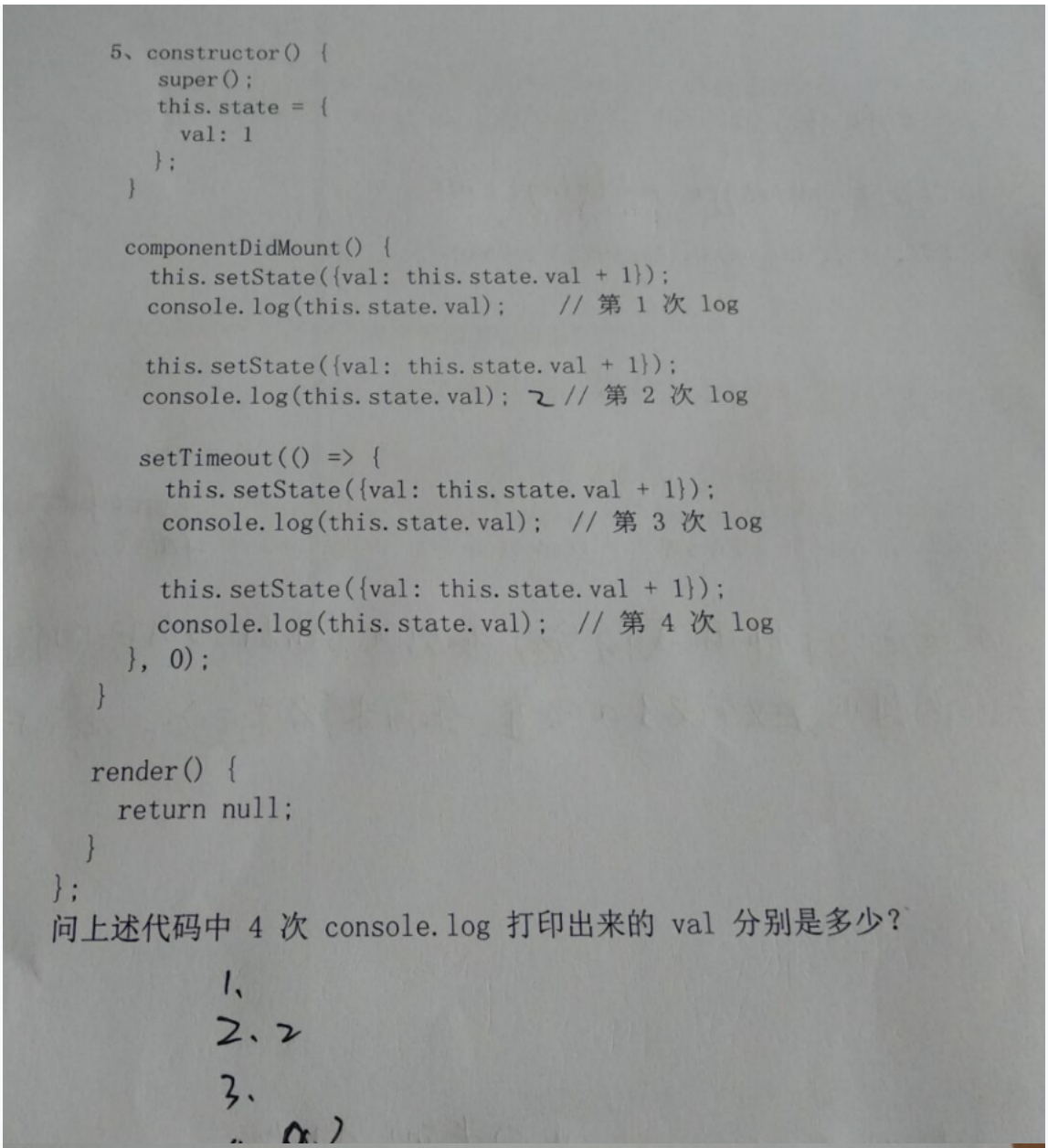
> 面试了三次 一面（最基础css动画，渐变，动画的实现），二面（webpack如何优化、canvas 使用，如何保证组件的加载顺序，react组件通信，代码拆分）三面 CEO 谈谈mvc思想，mvvm区别，框架对比，现场做了一道题目

```
1. //写一个函数，找出相同的字符并且返回得到
2.  var str1 = "139abc",str2 = "123adc";
```

海星淘（上海南站薪资在20-30）

> 面试问了如何优化react、打包太大怎么办、项目周期、redux 实现原理、组件通信、es6用过哪些、#### 上海集德健康管理（安心肿瘤）
> 做了一套面试题





5、constructor() {
super();
this.state = {
val: 1
};
}

componentDidMount() {
this.setState({val: this.state.val + 1});
console.log(this.state.val); // 第 1 次 log

this.setState({val: this.state.val + 1});
console.log(this.state.val); 2 // 第 2 次 log

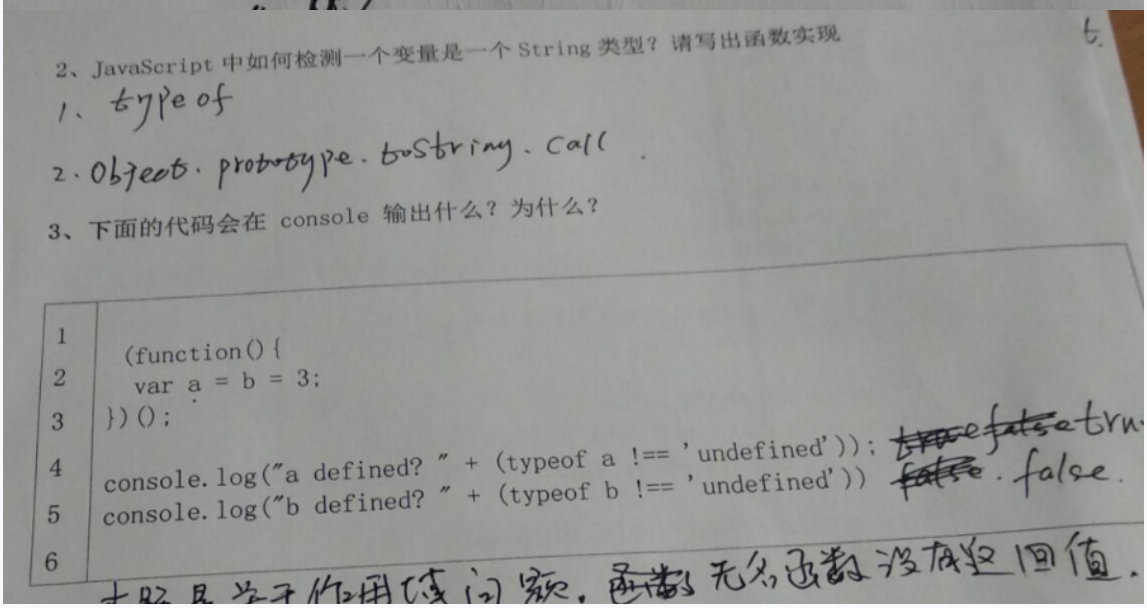
setTimeout(() => {
this.setState({val: this.state.val + 1});
console.log(this.state.val); // 第 3 次 log

this.setState({val: this.state.val + 1});
console.log(this.state.val); // 第 4 次 log
}, 0);
}

render() {
return null;
}
};

问上述代码中 4 次 console.log 打印出来的 val 分别是多少？

1、
2、2
3、
0 2



2、JavaScript 中如何检测一个变量是一个 String 类型？请写出函数实现

1、typeof

2、Object.prototype.toString.call

3、下面的代码会在 console 输出什么？为什么？

```
1
2   (function(){
    var a = b = 3;
3   })();
4   console.log("a defined? " + (typeof a !== 'undefined'));
5   console.log("b defined? " + (typeof b !== 'undefined'))
6
```

false true false. false.

无名函数没有返回值.

RN 的常用组件最少5个、

“”和“=”区别

问了为啥开发这个项目，项目的主要功能, RN了解没、会不会小程序、谈谈组件通信，职业规划。

上海复深蓝软件股份有限公司

> 做了一套面试题
> 作用域问题、js继承 方式有哪些、改变this方法区别，数组去重编写函数、react生命周期、组件通信有哪些、redux中交互（reducer是做什么的，以及实现的原理）
> 一面 技术过了，二面技术总监，技术过了，三面成功（offer）

投递过（自己推了没有去面试）

- [x] 文思海辉的HR 梁辰，现邀请您参加文思海辉百度项目的面试(原因是外包)
- [x] 北京深蓝海域信息科技有限公司（上海子公司）（原因也和软通一样跟着项目走的）
- [x] 慧筑科技hr（俞君杰）（原因人员太少距离太远）
- [x] 华筹电子（创业型公司）
- [x] 上海微企信息技术股份有限公司(路程太远)

state 和props 区别

> props：一般用于组件向组件通信，在组件之间通信使用。
> state: 一般用于组件内部的状态维护，更新组建内部的数据，状态，更新子组件的props等。
> React 的单向数据流，主要的流动管 道就是 props。props 本身是不可变的。
> 简单理解：父组件用props给子组件传递数据，而子组件的不能改变父组件的Props，如何改变子组件的Props，只有一种方法就是改变父组件的state。
> **React生命周期**
> React 生命周期根据广义定义描述，分为挂载，渲染，卸载这几个阶段，当渲染后的组件需要更新时，我们会从新渲染组件，直至卸载。
> 以此分为两大类
>
> - [x] 当组件挂载，或卸载时；
> - [x] 组件更新时（组件接收新的数据时）

> 初始化过程没什么特别的，包括读取初始 state 和 props 以及两个组件生命周期方法 componentWillMount 和 componentDidMount，这些都只会在组件初始化时运行一次。
> 如果我们在 componentWillMount 中执行 setState 方法，组件会更新 state， 但组件只渲染一次。因此，这是无意义的执行，初始化时的 state 都可以放在 this.state。
> 如果我们在 componentDidMount 中执行 setState 方法，又会发生什么呢?组件当然会再次更新，不过在初始化过程就渲染了两次组件，这并不是一件好事。但实际情况是，有一些场景不得 不需要 setState，比如计算组件的位置或宽高时，就不得不让组件先渲染，更新必要的信息后， 再次渲染。

声明周期特别又重要的生命周期 shouldComponentUpdate

> shouldComponentUpdate 是一个特别的方法，它接收需要更新的 props 和 state，让开发者增加 必要的条件判断，让其在需要时更新，不需要时不更新。因此，当方法返回 false 的时候，组件 不再向下执行生命周期方法。
> 值得注意的是，无状态组件是没有生命周期方法的，这也意味着它没有 shouldComponent-Update。渲染到该类组件时，每次都会重新渲染。当然，不少开发者在使用无状态组件时会纠结 这一点。为了更放心地使用，我们可以选择引用 Recompose 库的 pure 方法:

```
const OptimizedComponent = pure(ExpensiveComponent);
```

获取React-Dom

> React 提供的获取 DOM 元素的方法有两种，其中一种就是 ReactDOM 提供的 findDOMNode:
> refs 回调方式 是操作我们利用 render 方法得到了 App 组件的实例，然后就可以对它做一些操作。但在组件内， JSX 是不会返回一个组件的实例的!它只是一个 ReactElement

```
1. import React, { Component } from 'react';
2. import ReactDOM from 'react-dom';
3. class App extends Component {
4. componentDidMount() {
5. const dom = ReactDOM.findDOMNode(this);
6. }
7. render(){}
8. }
9. // 如果在 render 中返回 null，那么 findDOMNode 也返回 null。findDOMNode 只对已经挂载的组 件有效。
10. //涉及复杂操作时，还有非常多的原生 DOM API 可以用。但是需要严格限制场景，在使用之前多问自己为什么要操作 DOM。
11. // 注意的是，findDOMNode 和 refs 都无法用于无状态组件中，原因在前面已经说过。无状
12. //态组件挂载时只是方法调用，没有新建实例。
```

React事件注意事项

> 箭头函数可以不用bind改变this， 不写箭头函数需要手动写bind改变this指向

表单受控组件和非受控组件

>

受控组件

React 官方推荐受控组件 在表单必须有一个onChange事件

每当表单的状态发生变化时，都会被写入到组件的 state 中，这种组件在 React 中被称为受控组件(controlled component)。在受控组件中，组件渲染出的状态与它的 value 或 checked prop 相对应。React 通过这种方式消除了组件的局部状态，使得应用的整个状态更加可控；

非受控组件

在 React 中，非受控组件是一种反模式，它的值不受组件自身的 state 或 props 控制。通常， 需要通过为其添加 ref prop 来访问渲染后的底层 DOM 元素。

区别

如果不对受控组件绑定 change 事件，我们在文本框中输入任何值都不会起作用。多数情 况下，对于非受控组件，我们并不需要提供 change 事件。通过上面的示例可以看出，受控组件 和非受控组件的最大区别是:非受控组件的状态并不会受应用状态的控制，应用中也多了局部组 件状态，而受控组件的值来自于组件的 state。

React样式问题

当设置 width 和 height 这类与大小有关的样式时，react会自动添加上