React 面试题解析

序号	1
题目	简述 ReactNative 与原生 Android 常用的通信方式有几种？
解析	常用的通信方式如下。 （1）通过 RCTDeviceEventEmitter 事件通信。 （2）通过回调函数异步通信。 （3）通过 Promise 规范实现通信。 （4）通过原生 Android 直接向 ReactNative 传递常量数据

序号	2
题目	简述 Node.js 中的 Reactor Pattern 有什么理解？
解析	Node.js 中的 Reactor Pattern 基本上是一个非阻塞 I/O 操作的概念。该模式提供了一个与每个 I/O 操作相关联的处理程序，一旦生成 I/O 请求，它就会提交给多 路分解器。该解复用器是一个通知接口，能够在非阻塞 I/O 模式下处理并发。它还有助于以事件的形式收集每个请求，然后将每个事件放入队列中。从而导致事件队列的产生。同时，我们有事件循环，它迭代事件队列中存在的事件。

序号	3
题目	简述 React 有什么特点？
解析	1.声明式设计：React 使创建交互式 UI 变得轻而易举。为你应用的每一个状态设计简洁的视图，当数据变动时 React 能高效更新并渲染合适的组件。 2.组件化：构建管理自身状态的封装组件，然后对其组合以构成复杂的 UI。 3.高效：React 通过对 DOM 的模拟，最大限度地减少与 DOM 的交互。 4.灵活：无论你现在使用什么技术栈，在无需重写现有代码的前提下，通过引入 React 来开发新功能。 React 是一个声明式，高效且灵活的用于构建用户界面的 JavaScript 库。使用 React 可以将一些简短、独立的代码片段组合成复杂的 UI 界面，这些代码片段被称作“组件”。 由于 React 的设计思想极其独特，属于革命性创新，性能出众，代码逻辑却非常简单。所以，越来越多的人开始关注和使用，认为它可能是将来 Web 开发的主流工具。 这个项目本身也越滚越大，从最早的 UI 引擎变成了一整套前后端通吃的 Web App 解决方案。衍生的 React Native 项目，目标更是宏伟，希望用写 Web App 的方式去写 Native App。如果能够实现，整个互联网行业都会被颠覆，因为同一组人只需要写一次 UI ，就能同时运行在服务器、浏览器和手机。

序号	4
题目	简述 React 有什么优缺点？
解析	React 优点： 1、React 速度快、性能好 它并不直接对 DOM 进行操作，引入了一个叫做虚拟 DOM 的概念，安插在 javascript 逻辑和实际的 DOM 之间，性能好

	2、跨浏览器兼容 虚拟 DOM 的原因帮助我们解决了跨浏览器问题，它为我们提供了标准化的 API 3、单向数据流 Flux 随着 React 视图库的开发而被 Facebook 概念化，是一个用于在 JavaScript 应用中创建单向数据层的架构 4、React 兼容性好 使用 RequireJS 来加载和打包，而 Browserify 和 Webpack 适用于构建大型应用。 React 缺点 1.React 并不是一个单独完整的框架，React 是目标是 UI 组件，通常可以和其它框架组合使用，目前并不适合单独做一个完整的框架

序号	5
题目	简述 React store 的概念 ？
解析	React Store 就是把它们联系到一起的对象。Store 有以下职责： const store = createStore(reducer) 1:维持应用的 state； 2:提供 getState() 方法获取 state； 3:提供 dispatch(action) 方法更新 state； 4:通过 subscribe(listener)注册监听器； 5:通过 subscribe(listener)返回的函数注销监听器

序号	6
题目	简述 React 中引入 css 的方式？
解析	组件中引入 .module.css 文件 使用.module.css 文件来为组件引入样式，这种方式也被称为 CSS 模块化。 在.module.css 文件中定义的样式只能作用于当前组件内部，不会影响到其他组件或全局样式，这样可以避免样式冲突的问题。 CSS in JS CSS in JS 是一种前端开发技术，它将 CSS 样式表的定义和 JS 代码紧密结合在一起，以实现更高效、更灵活的样式控制。在 CSS in JS 中，开发者可以使用 JS 来编写 CSS 样式，可以在代码中通过变量或函数等方式来动态生成样式。这种方式可以避免传统 CSS 中的一些问题，如全局作用域、选择器嵌套、命名冲突等，同时也提供了更高的可重用性和可维护性。 在 React 中，有多种支持 CSS in JS 的第三方库，比较常用的有 styled-components、Emotion、JSS 等。这些库都提供了方便的 API 来定义和应用样式，并且可以自动管理 CSS 的引入和组件的封装。使用 CSS in JS 可以更好地与组件化开发思想结合，提高代码的可复用性和可维护性

序号	7
题目	请介绍 React 中的 key 有什么作用？
解析	在 React 中，key 是用来给每个组件的元素（Element）做一个唯一的标识。当 React 更新组件的时候，它会对比新旧两个组件的 key 是否一致，如果一致，则说明是同一个组件，直接更新它的内容即可。如果不一致，则说明是不同的组件，需要先删除旧组件，再新建一个新的组件并插入到 DOM 树中。

	因此，key 的作用是帮助 React 快速判断出哪些元素发生了变化，从而提高性能，避免不必要的 DOM 操作。同时，key 也可以用来保证数组渲染时每个元素的稳定性，避免出现类似于数组元素位置发生变化但是内容没变的情况

序号	8
题目	请列举常用的 React Hooks ？
解析	useState()：允许在函数组件中使用状态。使用 useState() 声明一个状态变量，并使用它来存储组件的状态。每次更改状态时，组件将重新渲染。 useEffect():用于处理副作用。副作用指在 React 组件之外进行的操作，例如从服务器获取数据，处理 DOM 元素等。使用 useEffect() hook，您可以执行此类操作，而无需在类组件中编写生命周期方法。点击去学习 useContext()：允许您在 React 中使用上下文。上下文是一种在组件树中传递数据的方法，可以避免通过 Props 一层层传递数据。使用 useContext() hook，您可以访问整个应用程序中定义的上下文对象。点击去学习 useReducer()：是 useState() hook 的替代品，用于管理更复杂的状态。它使用 Reducer 函数来管理组件状态，Reducer 函数接收当前状态和要进行的操作，然后返回新状态。详细使用方式见此文章。点击去学习 useCallback()：用于避免在每次渲染时重新创建回调函数。当您需要将回调函数传递给子组件时，这非常有用，因为它可以避免子组件不必要地重新渲染。点击去学习 useMemo()：用于缓存计算结果，以避免在每次渲染时重新计算。这非常有用，特别是当计算成本很高时。点击去学习 useRef()：用于创建对 DOM 元素的引用。它还可以用于存储组件之间共享的变量，这些变量不会在组件重新渲染时发生更改

序号	9
题目	请列举 React 和 vue.js 的相似性和差异性 ？
解析	相似性如下 （1）都是用于创建 UI 的 JavaScript 库。 （2）都是快速和轻量级的代码库（这里指 React 核心库）。 （3）都有基于组件的架构。 （4）都使用虚拟 DOM。 （5）都可以放在单独的 HTML 文件中，或者放在 Webpack 设置的一个更复杂的模块中。 （6）都有独立但常用的路由器和状态管理库。 它们最大的区别在于 Vue. js 通常使用 HTML 模板文件，而 React 完全使用 JavaScript 创建虚拟 DOM。 Vue. js 还具有对于“可变状态”的“ reactivity”的重新渲染的自动化检测系统

序号	10
题目	React Hook 的使用限制有哪些?
解析	React Hooks 的限制主要有两条: 不要在循环、条件或嵌套函数中调用 Hook; 在 React 的函数组件中调用 Hook。 那为什么会有这样的限制呢? Hooks 的设计初衷是为了改进 React 组件的开发模式。在旧有的开发模式下遇到了三个问题。 组件之间难以复用状态逻辑。过去常见的解决方案是高阶组件、render props 及状态管理框架。 复杂的组件变得难以理解。生命周期函数与业务逻辑耦合太深,导致关联部分难以拆分。 人和机器都很容易混淆类。常见的有 this 的问题,但在 React 团队中还有类难以优化的问题,希望在编译优化层面做出一些改进。 这三个问题在一定程度上阻碍了 React 的后续发展,所以为了解决这三个问题, Hooks 基于函数组件开始设计。然而第三个问题决定了 Hooks 只支持函数组件。 那为什么不要在循环、条件或嵌套函数中调用 Hook 呢? 因为 Hooks 的设计是基于数组实现。在调用时按顺序加入数组中,如果使用循环、条件或嵌套函数很有可能导致数组取值错位,执行错误的 Hook。当然,实质上 React 的源码里不是数组,是链表。 这些限制会在编码上造成一定程度的心智负担,新手可能会写错,为了避免这样的情况,可以引入 ESLint 的 Hooks 检查插件进行预防

序号	11
题目	使用 React Router 时,如何获取当前页面的路由或浏览器中地址栏中的地址?
解析	在当前组件的 props 中,包含 location 属性对象,包含当前页面路由地址信息,在 match 中存储当前路由的参数等数据信息。可以直接通过 this .props 使用它们

序号	12
题目	React.Children.map 和 js 的 map 有什么区别?
解析	JavaScript 中的 map 不会对为 null 或者 undefined 的数据进行处理,而 React.Children.map 中的 map 可以处理 React.Children 为 null 或者 undefined 的情况。

序号	13
题目	请简述 react-router 和 react-router-dom 的有什么区别?
解析	api 方面 React-router: 提供了路由的核心 api。如 Router、Route、Switch 等,但没有提供有关 dom 操作进行路由跳转的 api; React-router-dom: 提供了 BrowserRouter、Route、Link 等 api,可以通过 dom 操作触发事

	件控制路由。 Link 组件，会渲染一个 a 标签；BrowserRouter 和 HashRouter 组件，前者使用 pushState 和 popState 事件构建路由，后者使用 hash 和 hashchange 事件构建路由。 使用区别 react-router-dom 在 react-router 的基础上扩展了可操作 dom 的 api。Swtich 和 Route 都是从 react-router 中导入了相应的组件并重新导出，没做什么特殊处理。 react-router-dom 中 package.json 依赖中存在对 react-router 的依赖，故此，不需要额外安装 react-router

序号	14
题目	React 当调用 setState 的时候，发生了什么操作？
解析	当调用 setState 时， React 做的第一件事是将传递给 setState 的对象合并到组件的当前状态，这将启动一个称为和解（ reconciliation）的过程。 和解的最终目标是，根据这个新的状态以最有效的方式更新 DOM。 为此，React 将构建一个新的 React 虚拟 DOM 树(可以将其视为页面 DOM 元素的对象表示方式)。一旦有了这个 DOM 树，为了弄清 DOM 是如何响应新的状态而改变的， React 会将这个新树与上一个虚拟 DOM 树比较。 这样做， React 会知道发生的确切变化，并且通过了解发生的变化后，在绝对必要的情况下进行更新 DOM，即可将因操作 DOM 而占用的空间最小化

序号	15
题目	在 React 中元素（ element）和组件（ component）有什么区别？
解析	在 React 中元素（虚拟 DOM）描述了你在屏幕上看到的 DOM 元素。 换个说法就是，在 React 中元素是页面中 DOM 元素的对象表示方式。在 React 中组件是一个函数或一个类，它可以接受输入并返回一个元素。 注意：工作中，为了提高开发效率，通常使用 JSX 语法表示 React 元素（虚拟 DOM）。在编译的时候，把它转化成一个 React. createElement 调用方法

序号	16
题目	React 在哪个生命周期中你会发出 Ajax 请求？为什么？
解析	Ajax 请求应该写在组件创建期的第五个阶段，即 componentDidMount 生命周期方法中。原因如下。 在创建期的其他阶段，组件尚未渲染完成。而在存在期的 5 个阶段，又不能确保生命周期方法一定会执行（如通过 shouldComponentUpdate 方法优化更新等）。在销毁期，组件即将被销毁，请求数据变得无意义。因此在这些阶段发出 Ajax 请求显然不是最好的选择。 在组件尚未挂载之前，Ajax 请求将无法执行完毕，如果此时发出请求，将意味着在组件挂载之前更新状态（如执行 setState），这通常是不起作用的。 在 componentDidMount 方法中，执行 Ajax 即可保证组件已经挂载，并且能够正常更新组件

序号	17
题目	React shouldComponentUpdate 有什么用？为什么它很重要？
解析	组件状态数据或者属性数据发生更新的时候，组件会进入存在期，视图会渲染更新。在生命周期方法 should ComponentUpdate 中，允许选择退出某些组件（和它们的子组件）的和解过程。 和解的最终目标是根据新的状态，以最有效的方式更新用户界面。如果我们知道用户界面的某一部分不会改变，那么没有理由让 React 弄清楚它是否应该更新渲染。通过在 shouldComponentUpdate 方法中返回 false, React 将让当前组件及其所有子组件保持与当前组件状态相同

序号	18
题目	如何用 React 构建（ build）生产模式?
解析	通常，使用 Webpack 的 DefinePlugin 方法将 NODE ENV 设置为 production。这将剥离 propType 验证和额外的警告。除此之外，还可以减少代码，因为 React 使用 Uglify 的 dead-code 来消除开发代码和注释，这将大大减少包占用的空间

序号	19
题目	请简述 React 生命周期调用方法的顺序 ？
解析	React 生命周期分为三大周期，11 个阶段，生命周期方法调用顺序分别如下。 （1）在创建期的五大阶段，调用方法的顺序如下。 getDetaultProps：定义默认属性数据。 getInitialState：初始化默认状态数据。 component WillMount：组件即将被构建。 render：渲染组件。 componentDidMount：组件构建完成 （2）在存在期的五大阶段，调用方法的顺序如下。 componentWillReceiveProps：组件即将接收新的属性数据。 shouldComponentUpdate：判断组件是否应该更新。 componnent WillUpdate：组件即将更新。 render：渲染组件。 componentDidUpdate：组件更新完成。 （3）在销毁期的一个阶段，调用方法 componentWillUnmount，表示组件即将被销毁。

序号	20
题目	简述 React 组件开发中关于作用域的常见问题 ？
解析	在 EMAScript5 语法规范中，关于作用域的常见问题如下。 （1）在 map 等方法的回调函数中，要绑定作用域 this（通过 bind 方法）。 （2）父组件传递给子组件方法的作用域是父组件实例化对象，无法改变。 （3）组件事件回调函数方法的作用域是组件实例化对象（绑定父组件提供的方法就是父组件实例化对象），无法改变。 在 EMAScript6 语法规范中，关于作用域的常见问题如下。 （1)当使用箭头函数作为 map 等方法的回调函数时，箭头函数的作用域是当前组件的实例化对象(即箭头函数的作用域是定义时的作用域），无须绑定作用域。 （2）事件回调函数要绑定组件作用域。 （3）父组件传递方法要绑定父组件作用域。 总之，在 EMAScript6 语法规范中，组件方法的作用域是可以改变的。

序号	21
题目	ReactNative 中，如何解决 adb devices 找不到连接设备的问题？
解析	在使用 Genymotion 时，首先需要在 SDK 的 platform-tools 中加入环境变量，然后在 Genymotion 中单击 Setting，选择 ADB 选项卡，单击 Use custom Android SDK tools，浏览本地 SDK 的位置，单击 OK 按钮就可以了。启动虚拟机后，在 cmd 中输入 adb devices 可以查看设备

序号	22
题目	在使用 React Router 时，如何获取当前页面的路由或浏览器中地址栏中的地址？
解析	在当前组件的 props 中，包含 location 属性对象，包含当前页面路由地址信息，在 match 中存储当前路由的参数等数据信息。可以直接通过 this .props 使用它们

序号	23
题目	简述 React 中什么是纯组件？
解析	纯组件是可以编写的最简单和最快的组件。它们可以替换任何只有 render() 的组件。这些组件增强了代码的简单性和应用程序的性能。

序号	24
题目	解释 React Reducer 的作用？
解析	Reducers 是纯函数，它指定应用程序的状态如何响应 ACTION 变化。Reducers 通过接收之前的状态和动作来工作，然后它返回一个新状态。它根据操作的类型确定需要进行何种类型的更新，然后返回新值。 如果不需要做任何工作，它会按原样返回先前的状态

序号	25
题目	React 中的箭头函数是什么？它是如何使用的？

解析	箭头函数更多是用于编写函数表达式的简短语法。它们也被称为“胖箭头”（=>）函数。这些函数允许正确绑定组件的上下文，因为在 ES6 中自动绑定默认不可用。箭头函数在处理高阶函数时最有用。
	
	登录后复制
	//General way
	render() {
	return(
	
	);
	}
	//With Arrow Function
	render() {
	return(
	this.handleOnChange(e) } />
	);
	}

序号	26
题目	解释为什么在 React Router v4 中使用 switch 关键字?
解析	switch 是用来封装 Router 内部的多条路由的。当您只想显示要在多个定义的路由中呈现的单个路由时，使用 “switch”关键字 。 使用中的 标记将键入的 URL 与已定义的路由按顺序匹配。 当找到第一个匹配项时，它会呈现指定的路由。从而绕过其余 路线。

序号	27
题目	简述什么是 React Context?
解析	Context 通过组件树提供了一个传递数据的方法，从而避免了在每一个层级手动的传递 props 属性

序号	28
题目	React 中如何避免不必要的 render?
解析	React 基于虚拟 DOM 和高效 Diff 算法的完美配合，实现了对 DOM 最小粒度的更新。大多数情况下，React 对 DOM 的渲染效率足以业务日常。但在个别复杂业务场景下，性能问题依然会困扰我们。此时需要采取一些措施来提升运行性能，其很重要的一个方向，就是避免不必要的渲染（Render）。这里提下优化的点： shouldComponentUpdate 和 PureComponent 在 React 类组件中，可以利用 shouldComponentUpdate 或者 PureComponent 来减少因父组件更新而触发子组件的 render，从而达到目的。shouldComponentUpdate 来决定是否组件是否重新渲染，如果不希望组件重新渲染，返回 false 即可。 利用高阶组件 在函数组件中，并没有 shouldComponentUpdate 这个生命周期，可以利用高阶组件，封装一个类似 PureComponet 的功能 使用 React.memo React.memo 是 React 16.6 新的一个 API，用来缓存组件的渲染，避免不必要的更新，其实也是一个高阶组件，与 PureComponent 十分类似，但不同的是， React.memo 只能用于函数组件

序号	29
题目	简述 React- Router 有几种形式?
解析	有以下几种形式。 HashRouter，通过散列实现，路由要带#。 BrowerRouter，利用 HTML5 中 history API 实现，需要服务器端支持，兼容性不是很好

序号	30
题目	简述 React 中的 Portal 是什么?
解析	Portals 提供了一种很好的将子节点渲染到父组件以外的 DOM 节点的方式。 第一个参数（child）是任何可渲染的 React 子元素，例如一个元素，字符串或碎片。 第二个参数（container）则是一个 DOM 元素。 ReactDOM.createPortal(child, container)