### **https://segmentfault.com/a/1190000016885832?utm\_source=tag-newest**

### **React 中 keys 的作用是什么？**

Keys 是 React 用于追踪哪些列表中元素被修改、被添加或者被移除的辅助标识。

render () {

**return** (

<ul>

{this.state.todoItems.**map**(({**item**, key}) => {

**return** <li key={key}>{**item**}</li>

})}

</ul>

)

}

在开发过程中，我们需要保证某个元素的 key 在其同级元素中具有唯一性。在 React Diff 算法中 React 会借助元素的 Key 值来判断该元素是新近创建的还是被移动而来的元素，从而减少不必要的元素重渲染。此外，React 还需要借助 Key 值来判断元素与本地状态的关联关系，因此我们绝不可忽视转换函数中 Key 的重要性。

### **调用 setState 之后发生了什么？**

在代码中调用 setState 函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。经过调和过程，React 会以相对高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个 UI 界面。在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染。在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

### **react 生命周期函数**

初始化阶段：

* + getDefaultProps:获取实例的默认属性
  + getInitialState:获取每个实例的初始化状态
  + componentWillMount：组件即将被装载、渲染到页面上
  + render:组件在这里生成虚拟的 DOM 节点
  + componentDidMount:组件真正在被装载之后

运行中状态：

* + componentWillReceiveProps:组件将要接收到属性的时候调用
  + shouldComponentUpdate:组件接受到新属性或者新状态的时候（可以返回 false，接收数据后不更新，阻止 render 调用，后面的函数不会被继续执行了）
  + componentWillUpdate:组件即将更新不能修改属性和状态
  + render:组件重新描绘
  + componentDidUpdate:组件已经更新

销毁阶段：

* + componentWillUnmount:组件即将销毁

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

shouldComponentUpdate 这个方法用来判断是否需要调用 render 方法重新描绘 dom。因为 dom 的描绘非常消耗性能，如果我们能在 shouldComponentUpdate 方法中能够写出更优化的 dom diff 算法，可以极大的提高性能。

参考react 性能优化-sf

### **为什么虚拟 dom 会提高性能?(必考)**

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

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。

参考 [如何理解虚拟 DOM?-zhihu](https://www.zhihu.com/question/29504639?sort=created)

### **react diff 原理（常考，大厂必考）**

* 把树形结构按照层级分解，只比较同级元素。
* 给列表结构的每个单元添加唯一的 key 属性，方便比较。
* React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）
* 合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制.
* 选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。

参考：React 的 diff 算法

### **React 中 refs 的作用是什么？**

Refs 是 React 提供给我们的安全访问 DOM 元素或者某个组件实例的句柄。我们可以为元素添加 ref 属性然后在回调函数中接受该元素在 DOM 树中的句柄，该值会作为回调函数的第一个参数返回：

**class** **CustomForm** **extends** **Component** {

handleSubmit = () => {

console.log("Input Value: ", **this**.input.value)

}

render () {

**return** (

<form onSubmit={**this**.handleSubmit}>

<input

**type**='text'

ref={(input) => **this**.input = input} />

<button **type**='submit'>**Submit**</button>

</form>

)

}

}

上述代码中的 input 域包含了一个 ref 属性，该属性声明的回调函数会接收 input 对应的 DOM 元素，我们将其绑定到 this 指针以便在其他的类函数中使用。另外值得一提的是，refs 并不是类组件的专属，函数式组件同样能够利用闭包暂存其值：

**function** **CustomForm** ({handleSubmit}) {

**let** inputElement

**return** (

<form onSubmit={() => handleSubmit(inputElement.value)}>

<input

type='text'

ref={(input) => inputElement = input} />

<button type='submit'>Submit</button>

</form>

)

}

### **如果你创建了类似于下面的 Twitter 元素，那么它相关的类定义是啥样子的？**

<Twitter username='tylermcginnis33'>

{(user) => user === null

? <Loading />

: <Badge info={user} />}

</Twitter>

**import** **React**, { **Component**, **PropTypes** } from 'react'**import** fetchUser from 'twitter'*// fetchUser take in a username returns a promise// which will resolve with that username's data.***class** **Twitter** **extends** **Component** {

*// finish this*

}

如果你还不熟悉回调渲染模式（Render Callback Pattern），这个代码可能看起来有点怪。这种模式中，组件会接收某个函数作为其子组件，然后在渲染函数中以 props.children 进行调用：

**import** **React**, { **Component**, **PropTypes** } from 'react'**import** fetchUser from 'twitter'**class** **Twitter** **extends** **Component** {

state = {

user: null,

}

static propTypes = {

username: **PropTypes**.string.isRequired,

}

componentDidMount () {

fetchUser(**this**.props.username)

.then((user) => **this**.setState({user}))

}

render () {

**return** **this**.props.children(**this**.state.user)

}

}

这种模式的优势在于将父组件与子组件解耦和，父组件可以直接访问子组件的内部状态而不需要再通过 Props 传递，这样父组件能够更为方便地控制子组件展示的 UI 界面。譬如产品经理让我们将原本展示的 Badge 替换为 Profile，我们可以轻易地修改下回调函数即可：

<Twitter username='tylermcginnis33'>

{(user) => user === null

? <Loading />

: <Profile info={user} />}

</Twitter>

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

* 展示组件关心组件看起来是什么。展示专门通过 props 接受数据和回调，并且几乎不会有自身的状态，但当展示组件拥有自身的状态时，通常也只关心 UI 状态而不是数据的状态。
* 容器组件则更关心组件是如何运作的。容器组件会为展示组件或者其它容器组件提供数据和行为(behavior)，它们会调用 Flux actions，并将其作为回调提供给展示组件。容器组件经常是有状态的，因为它们是(其它组件的)数据源。

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

* 类组件不仅允许你使用更多额外的功能，如组件自身的状态和生命周期钩子，也能使组件直接访问 store 并维持状态
* 当组件仅是接收 props，并将组件自身渲染到页面时，该组件就是一个 '无状态组件(stateless component)'，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件

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

* State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作为用户事件行为的结果。
* Props(properties 的简写)则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的(immutable)。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)。Props 也不仅仅是数据--回调函数也可以通过 props 传递。

### **何为受控组件(controlled component)**

在 HTML 中，类似 <input>, <textarea> 和 <select> 这样的表单元素会维护自身的状态，并基于用户的输入来更新。当用户提交表单时，前面提到的元素的值将随表单一起被发送。但在 React 中会有些不同，包含表单元素的组件将会在 state 中追踪输入的值，并且每次调用回调函数时，如 onChange 会更新 state，重新渲染组件。一个输入表单元素，它的值通过 React 的这种方式来控制，这样的元素就被称为"受控元素"。

### **何为高阶组件(higher order component)**

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC 运行你重用代码、逻辑和引导抽象。最常见的可能是 Redux 的 connect 函数。除了简单分享工具库和简单的组合，HOC 最好的方式是共享 React 组件之间的行为。如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的 HOC。

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

因为 this.props 和 this.state 的更新可能是异步的，不能依赖它们的值去计算下一个 state。

### **除了在构造函数中绑定 this，还有其它方式吗**

你可以使用属性初始值设定项(property initializers)来正确绑定回调，create-react-app 也是默认支持的。在回调中你可以使用箭头函数，但问题是每次组件渲染时都会创建一个新的回调。

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

在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。

### **应该在 React 组件的何处发起 Ajax 请求**

在 React 组件中，应该在 componentDidMount 中发起网络请求。这个方法会在组件第一次“挂载”(被添加到 DOM)时执行，在组件的生命周期中仅会执行一次。更重要的是，你不能保证在组件挂载之前 Ajax 请求已经完成，如果是这样，也就意味着你将尝试在一个未挂载的组件上调用 setState，这将不起作用。在 componentDidMount 中发起网络请求将保证这有一个组件可以更新了。

### **描述事件在 React 中的处理方式。**

为了解决跨浏览器兼容性问题，您的 React 中的事件处理程序将传递 SyntheticEvent 的实例，它是 React 的浏览器本机事件的跨浏览器包装器。

这些 SyntheticEvent 与您习惯的原生事件具有相同的接口，除了它们在所有浏览器中都兼容。有趣的是，React 实际上并没有将事件附加到子节点本身。React 将使用单个事件监听器监听顶层的所有事件。这对于性能是有好处的，这也意味着在更新 DOM 时，React 不需要担心跟踪事件监听器。

### **createElement 和 cloneElement 有什么区别？**

React.createElement():JSX 语法就是用 React.createElement()来构建 React 元素的。它接受三个参数，第一个参数可以是一个标签名。如 div、span，或者 React 组件。第二个参数为传入的属性。第三个以及之后的参数，皆作为组件的子组件。

**React**.createElement(

**type**,

[props],

[...children]

)

React.cloneElement()与 React.createElement()相似，不同的是它传入的第一个参数是一个 React 元素，而不是标签名或组件。新添加的属性会并入原有的属性，传入到返回的新元素中，而就的子元素奖杯替换。

**React**.cloneElement(

**element**,

[props],

[...children]

)

### **React 中有三种构建组件的方式**

React.createClass()、ES6 class 和无状态函数。

### **react 组件的划分业务组件技术组件？**

* 根据组件的职责通常把组件分为 UI 组件和容器组件。
* UI 组件负责 UI 的呈现，容器组件负责管理数据和逻辑。
* 两者通过 React-Redux 提供 connect 方法联系起来。

### **简述 flux 思想**

Flux 的最大特点，就是数据的"单向流动"。

1. 用户访问 View
2. View 发出用户的 Action
3. Dispatcher 收到 Action，要求 Store 进行相应的更新
4. Store 更新后，发出一个"change"事件
5. View 收到"change"事件后，更新页面

### **React 项目用过什么脚手架（本题是开放性题目）**

creat-react-app Yeoman 等

### **了解 redux 么，说一下 redux 把**

* redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，主要有三个核心方法，action，store，reducer，工作流程是 view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据，flux 也是用来进行数据操作的，有四个组成部分 action，dispatch，view，store，工作流程是 view 发出一个 action，派发器接收 action，让 store 进行数据更新，更新完成以后 store 发出 change，view 接受 change 更新视图。Redux 和 Flux 很像。主要区别在于 Flux 有多个可以改变应用状态的 store，在 Flux 中 dispatcher 被用来传递数据到注册的回调事件，但是在 redux 中只能定义一个可更新状态的 store，redux 把 store 和 Dispatcher 合并,结构更加简单清晰
* 新增 state,对状态的管理更加明确，通过 redux，流程更加规范了，减少手动编码量，提高了编码效率，同时缺点时当数据更新时有时候组件不需要，但是也要重新绘制，有些影响效率。一般情况下，我们在构建多交互，多数据流的复杂项目应用时才会使用它们

### **redux 有什么缺点**

* 一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。
* 当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断。

**<https://blog.csdn.net/eyeofangel/article/details/88797314>**

### **React**

* 什么时候使用状态管理器？
* render函数中return如果没有使用()会有什么问题？
* componentWillUpdate可以直接修改state的值吗？
* 说说你对React的渲染原理的理解
* 什么渲染劫持？
* React Intl是什么原理？
* 你有使用过React Intl吗？
* 怎么实现React组件的国际化呢？
* 说说Context有哪些属性？
* 怎么使用Context开发组件？
* 为什么React并不推荐我们优先考虑使用Context？
* 除了实例的属性可以获取Context外哪些地方还能直接获取Context呢？
* childContextTypes是什么？它有什么用？
* contextType是什么？它有什么用？
* Consumer向上找不到Provider的时候怎么办？
* 有使用过Consumer吗？
* 在React怎么使用Context？
* React15和16别支持IE几以上？
* 说说你对windowing的了解
* 举例说明React的插槽有哪些运用场景？
* 你有用过React的插槽(Portals)吗？怎么用？
* React的严格模式有什么用处？
* React如何进行代码拆分？拆分的原则是什么？
* React组件的构造函数有什么作用？
* React组件的构造函数是必须的吗？
* React中在哪捕获错误？
* React怎样引入svg的文件？
* 说说你对Relay的理解
* 在React中你有经常使用常量吗？
* 为什么说React中的props是只读的？
* 你有使用过formik库吗？说说它的优缺点
* 你有用过哪些React的表单库吗？说说它们的优缺点
* 如果组件的属性没有传值，那么它的默认值是什么？
* 可以使用TypeScript写React应用吗？怎么操作？
* super()和super(props)有什么区别？
* 你有使用过loadable组件吗？它帮我们解决了什么问题？
* 你有使用过suspense组件吗？它帮我们解决了什么问题？
* 怎样动态导入组件？
* 如何给非控组件设置默认的值？
* 怎么在React中引入其它的UI库，例如Bootstrap
* 怎样将事件传递给子组件？
* 怎样使用Hooks获取服务端数据？
* 使用Hooks要遵守哪些原则？
* render方法的原理你有了解吗？它返回的数据类型是什么？
* useEffect和useLayoutEffect有什么区别？
* 在React项目中你用过哪些动画的包？
* React必须使用JSX吗？
* 自定义组件时render是可选的吗？为什么？
* 需要把keys设置为全局唯一吗？
* 怎么定时更新一个组件？
* React根据不同的环境打包不同的域名？
* 使用webpack打包React项目，怎么减小生成的js大小？
* 在React中怎么使用async/await？
* 你阅读了几遍React的源码？都有哪些收获？你是怎么阅读的？
* 什么是React.forwardRef？它有什么作用？
* 写个例子说明什么是JSX的内联条件渲染
* 在React中怎么将参数传递给事件？
* React的事件和普通的HTML事件有什么不同？
* 在React中怎么阻止事件的默认行为？
* 你最喜欢React的哪一个特性（说一个就好）？
* 在React中什么时候使用箭头函数更方便呢？
* 你最不喜欢React的哪一个特性（说一个就好）？
* 说说你对React的reconciliation（一致化算法）的理解
* 使用PropTypes和Flow有什么区别？
* 怎样有条件地渲染组件？
* 在JSX中如何写注释？
* constructor和getInitialState有不同？
* 写例子说明React如何在JSX中实现for循环
* 为什么建议Fragment包裹元素？它的简写是什么？
* 你有用过React.Fragment吗？说说它有什么用途？
* 在React中你有遇到过安全问题吗？怎么解决？
* React中如何监听state的变化？
* React什么是有状态组件？
* React v15中怎么处理错误边界？
* React Fiber它的目的是解决什么问题？
* React为什么不要直接修改state？如果想修改怎么做？
* create-react-app有什么好处？
* 装饰器(Decorator)在React中有什么应用？
* 使用高阶组件(HOC)实现一个loading组件
* 如何用React实现滚动动画？
* 说出几点你认为的React最佳实践
* 你是如何划分React组件的？
* 举例说明如何在React创建一个事件
* 如何更新组件的状态？
* 怎样将多个组件嵌入到一个组件中？
* React的render中可以写{if else}这样的判断吗？
* React为什么要搞一个Hooks？
* React Hooks帮我们解决了哪些问题？
* 使用React的memo和forwardRef包装的组件为什么提示children类型不对？
* 有在项目中使用过Antd吗？说说它的好处
* 在React中如果去除生产环境上的sourcemap？
* 在React中怎么引用sass或less？
* 组件卸载前，加在DOM元素的监听事件和定时器要不要手动清除？为什么？
* 为什么标签里的for要写成htmlFor呢？
* 状态管理器解决了什么问题？什么时候用状态管理器？
* 状态管理器它精髓是什么？
* 函数式组件有没有生命周期？为什么？
* 在React中怎么引用第三方插件？比如说jQuery等
* React的触摸事件有哪几种？
* 路由切换时同一组件无法重新渲染的有什么方法可以解决？
* React16新特性有哪些？
* 你有用过哪些React的UI库？它们的优缺点分别是什么？
* <div onClick={handlerClick}>单击</div>和<div onClick={handlerClick(1)}>单击</div>有什么区别？
* 在React中如何引入图片？哪种方式更好？
* 在React中怎么使用字体图标？
* React的应用如何打包发布？它的步骤是什么？
* ES6的语法'...'在React中有哪些应用？
* 如何封装一个React的全局公共组件？
* 在React中组件的props改变时更新组件的有哪些方法？
* immutable的原理是什么？
* 你对immutable有了解吗？它有什么作用？
* 如何提高组件的渲染效率呢？
* 在React中如何避免不必要的render？
* render在什么时候会被触发？
* 写出React动态改变class切换组件样式
* React中怎么操作虚拟DOM的Class属性？
* 为什么属性使用className而不是class呢？
* 请说下react组件更新的机制是什么？
* 怎么在JSX里属性可以被覆盖吗？覆盖的原则是什么？
* 怎么在JSX里使用自定义属性？
* 怎么防止HTML被转义？
* 经常用React，你知道React的核心思想是什么吗？
* 在React中我们怎么做静态类型检测？都有哪些方法可以做到？
* 在React中组件的state和setState有什么区别？
* React怎样跳过重新渲染？
* React怎么判断什么时候重新渲染组件呢？
* 什么是React的实例？函数式组件有没有实例？
* 在React中如何判断点击元素属于哪一个组件？
* 在React中组件和元素有什么区别？
* 在React中声明组件时组件名的第一个字母必须是大写吗？为什么？
* 举例说明什么是高阶组件(HOC)的反向继承？
* 有用过React Devtools吗？说说它的优缺点分别是什么？
* 举例说明什么是高阶组件(HOC)的属性代理？
* React的isMounted有什么作用？
* React组件命名推荐的方式是哪个？为什么不推荐使用displayName？
* React的displayName有什么作用？
* 说说你对React的组件命名规范的理解
* 说说你对React的项目结构的理解
* React16废弃了哪些生命周期？为什么？
* 怎样在React中开启生产模式？
* React中getInitialState方法的作用是什么？
* React中你知道creatClass的原理吗？
* React中验证props的目的是什么？
* React中你有使用过getDefaultProps吗？它有什么作用？
* React中你有使用过propType吗？它有什么作用？
* React中怎么检验props？
* React.createClass和extends Component的区别有哪些？
* 高阶组件(HOC)有哪些优点和缺点？
* 给组件设置很多属性时不想一个个去设置有什么办法可以解决这问题呢？
* React16跟之前的版本生命周期有哪些变化？
* 怎样实现React组件的记忆？原理是什么？
* 创建React动画有哪些方式？
* 为什么建议不要过渡使用Refs？
* 在React使用高阶组件(HOC)有遇到过哪些问题？如何解决？
* 在使用React过程中什么时候用高阶组件(HOC)？
* 说说React diff的原理是什么？
* React怎么提高列表渲染的性能？
* 使用ES6的class定义的组件不支持mixins了，那用什么可以替代呢？
* 为何说虚拟DOM会提高性能？
* React的性能优化在哪个生命周期？它优化的原理是什么？
* 你知道的React性能优化有哪些方法？
* 举例说明在React中怎么使用样式？
* React有哪几种方法来处理表单输入？
* 什么是浅层渲染？
* 你有做过React的单元测试吗？如果有，用的是哪些工具？怎么做的？
* 在React中什么是合成事件？有什么用？
* 使用React写一个todo应用，说说你的思路
* React16的reconciliation和commit分别是什么？
* React的函数式组件有没有生命周期？
* useState和this.state的区别是什么？
* 请说说什么是useImperativeHandle？
* 请说说什么是useReducer？
* 请说说什么是useRef？
* 请说说什么是useEffect？
* 举例说明useState
* 请说说什么是useState？为什么要使用useState？
* 请描述下你对React的新特性Hooks的理解？它有哪些应用场景？
* 说说你对Error Boundaries的理解
* 说说你对Fiber架构的理解
* 说说你是怎么理解React的业务组件和技术组件的？
* 为什么建议setState的第一个参数是callback而不是一个对象呢？
* 展示组件和容器组件有什么区别？
* Mern和Yeoman脚手架有什么区别？
* 你有在项目中使用过Yeoman脚手架吗？
* 你有在项目中使用过Mern脚手架吗？
* shouldComponentUpdate方法是做什么的？
* 怎样在React中使用innerHTML？
* 你有写过React的中间件插件吗？
* React的中间件机制是怎么样的？这种机制有什么作用？
* React中你用过哪些第三方的中间件？
* 不用脚手架，你会手动搭建React项目吗？
* 请说说React中Portal是什么？
* React中修改prop引发的生命周期有哪几个？
* React多个setState调用的原理是什么？
* React中调用setState会更新的生命周期有哪几个？
* React中setState的第二个参数作用是什么呢？
* React中的setState是同步还是异步的呢？为什么state并不一定会同步更新？
* React中的setState批量更新的过程是什么？
* React中的setState执行机制是什么呢？
* 在React中遍历的方法有哪些？它们有什么区别呢？
* 请说说你对React的render方法的理解
* props.children.map和js的map有什么区别？为什么优先选择React的？
* 有用过React的严格模式吗？
* React中的setState和replaceState的区别是什么？
* React中的setState缺点是什么呢？
* 有用过React的Fragment吗？它的运用场景是什么？
* React组件间共享数据方法有哪些？
* React的状态提升是什么？使用场景有哪些？
* 简单描述下你有做过哪些React项目？
* 在构造函数中调用super(props)的目的是什么？
* 你是如何学习React的？
* 从旧版本的React升级到新版本的React有做过吗？有遇到过什么坑？
* 你用过React版本有哪些？
* 有用过React的服务端渲染吗？怎么做的？
* React的mixins有什么作用？适用于什么场景？
* React怎么拿到组件对应的DOM元素？
* 请描述下事件在React中的处理方式是什么？
* JSX和HTML有什么区别？
* React的书写规范有哪些？
* create-react-app创建新运用怎么解决卡的问题？
* 使用React的方式有哪几种？
* 说说你对reader的context的理解
* 同时引用这三个库React.js、React-dom.js和babel.js它们都有什么作用？
* 你知道Virtual DOM的工作原理吗？
* 你阅读过React的源码吗？简要说下它的执行流程
* React中怎样阻止组件渲染？
* React非兄弟组件如何通信？
* React兄弟组件如何通信？
* React非父子组件如何通信？
* React父子组件如何通信？
* React组件间的通信有哪些？
* 类组件和函数式组件有什么区别？
* React自定义组件你写过吗？说说看都写过哪些？
* React组件的state和props两者有什么区别？
* React有几种构建组件的方式？可以写出来吗？
* React中遍历时为什么不用索引作为唯一的key值？
* React中的key有什么作用？
* React中除了在构造函数中绑定this,还有别的方式吗？
* 在React中页面重新加载时怎样保留数据？
* 请描述下React的事件机制
* 怎样在React中创建一个事件？
* 在React中无状态组件有什么运用场景？
* 描述下在React中无状态组件和有状态组件的区别是什么？
* 写一个React的高阶组件(HOC)并说明你对它的理解
* React中可以在render访问refs吗？为什么？
* React中refs的作用是什么？有哪些应用场景？
* 请描述你对纯函数的理解？
* 受控组件和非受控组件有什么区别？
* React中什么是非控组件？
* React中什么是受控组件？
* React中发起网络请求应该在哪个生命周期中进行？为什么？
* 说说React的生命周期有哪些？
* 说说你对“在React中，一切都是组件”的理解
* 写React你是用es6还是es5的语法？有什么区别？
* 浏览器为什么无法直接JSX？怎么解决呢？
* 在使用React过程中你都踩过哪些坑？你是怎么填坑的？
* 说说你喜欢React的原因是什么？它有什么优缺点？
* 如何解决引用类型在pureComponent下修改值的时候，页面不渲染的问题？
* createElement与cloneElement两者有什么区别？
* 解释下React中Element 和Component两者的区别是什么？
* 解释下React中component和pureComponent两者的区别是什么？
* React的虚拟DOM和vue的虚拟DOM有什么区别？
* 你觉得React上手快不快？它有哪些限制？
* 说说你对声明式编程的理解？
* React与angular、vue有什么区别？
* React是哪个公司开发的？
* React是什么？它的主要特点是什么？
* 简要描述下你知道的React工作原理是什么？
* 在React中怎样改变组件状态，以及状态改变的过程是什么？
* 在React中你是怎么进行状态管理的？
* React声明组件有哪几种方法，各有什么不同？

### **ReactNative**

* 如何在React Native中设置环境变量？
* 请描述下Code Push的原理是什么？
* React Native怎样查看日记？
* React Native怎样测试？
* React Native怎样调试？
* React Native和React有什么区别？
* 有做过React Native项目吗？

### **React-Router**

* React-Router怎么获取历史对象？
* React-Router怎么获取URL的参数？
* 在history模式中push和replace有什么区别？
* React-Router怎么设置重定向？
* React-Router 4中<Router>组件有几种类型？
* React-Router 3和React-Router 4有什么变化？添加了什么好的特性？
* React-Router的实现原理是什么？
* React-Router 4的switch有什么用？
* React-Router的路由有几种模式？
* React-Router 4怎样在路由变化时重新渲染同一个组件？
* React-Router的<Link>标签和<a>标签有什么区别？
* React的路由和普通路由有什么区别？
* 请你说说React的路由的优缺点？
* 请你说说React的路由是什么？

### **Redux/Mobox**

* 你有了解Rxjs是什么吗？它是做什么的？
* 在Redux中怎么发起网络请求？
* Redux怎样重置状态？
* Redux怎样设置初始状态？
* Context api可以取代Redux吗？为什么？
* 推荐在reducer中触发Action吗？为什么？
* Redux怎么添加新的中间件？
* redux-saga和redux-thunk有什么本质的区别？
* 在React中你是怎么对异步方案进行选型的？
* 你知道redux-saga的原理吗？
* 你有使用过redux-saga中间件吗？它是干什么的？
* Redux中异步action和同步action最大的区别是什么？
* Redux和vuex有什么区别？
* Redux的中间件是什么？你有用过哪些Redux的中间件？
* 说说Redux的实现流程
* Mobx的设计思想是什么？
* Redux由哪些组件构成？
* Mobx和Redux有什么区别？
* 在React项目中你是如何选择Redux和Mobx的？说说你的理解
* 你有在React中使用过Mobx吗？它的运用场景有哪些？
* Redux的thunk作用是什么？
* Redux的数据存储和本地储存有什么区别？
* 在Redux中，什么是reducer？它有什么作用？
* 举例说明怎么在Redux中定义action？
* 在Redux中，什么是action？
* 在Redux中，什么是store？
* 为什么Redux能做到局部渲染呢？
* 说说Redux的优缺点分别是什么？
* Redux和Flux的区别是什么？
* Redux它的三个原则是什么？
* 什么是单一数据源？
* 什么是Redux？说说你对Redux的理解？有哪些运用场景？

### **Flux**

* 请说说点击按钮触发到状态更改，数据的流向？
* 请描述下Flux的思想
* 什么是Flux？说说你对Flux的理解？有哪些运用场景？