# react的面试题

## 怎么阻止组件的渲染

在组件的 render 方法中返回 null 并不会影响触发组件的生命周期方法

## react的工作原理是什么

React 会创建一个虚拟 DOM(virtual DOM)。当一个组件中的状态改变时，React 首先会通过 "diffing" 算法来标记虚拟 DOM 中的改变，第二步是调节(reconciliation)，会用 diff 的结果来更新 DOM。

## 简单介绍下react hooks hooks产生的背景 hooks的优点

hooks是针对在使用react时存在以下问题而产生的：

组件之间复用状态逻辑很难，在hooks之前，实现组件复用，一般采用高阶组件和 Render Props，它们本质是将复用逻辑提升到父组件中，很容易产生很多包装组件，带来嵌套地域。

组件逻辑变得越来越复杂，尤其是生命周期函数中常常包含一些不相关的逻辑，完全不相关的代码却在同一个方法中组合在一起。如此很容易产生 bug，并且导致逻辑不一致。

复杂的class组件，使用class组件，需要理解 JavaScript 中 this 的工作方式，不能忘记绑定事件处理器等操作，代码复杂且冗余。除此之外，class组件也会让一些react优化措施失效。

针对上面提到的问题，react团队研发了hooks，它主要有两方面作用：

用于在函数组件中引入状态管理和生命周期方法

取代高阶组件和render props来实现抽象和可重用性

优点也很明显：

避免在被广泛使用的函数组件在后期迭代过程中，需要承担一些副作用，而必须重构成类组件，它帮助函数组件引入状态管理和生命周期方法。

Hooks 出现之后，我们将复用逻辑提取到组件顶层，而不是强行提升到父组件中。这样就能够避免 HOC 和 Render Props 带来的「嵌套地域」

避免上面陈述的class组件带来的那些问题

## 知道hoc和render props吗，它们有什么作用？

render props模式是一种非常灵活复用性非常高的模式，它可以把特定行为或功能封装成一个组件，提供给其他组件使用让其他组件拥有这样的能力。他把组件可以动态渲染的地方暴露给外部，你不用再关注组件的内部实现，只要把数据通过函数传出去就好。

Render Props 组件和高阶组件主要用来实现抽象和可重用性。

## useContext 的作用是什么

useContext()，共享钩子。该钩子的作用是，在组件之间共享状态。 可以解决react逐层通过Porps传递数据，它接受React.createContext()的返回结果作为参数，使用useContext将不再需要Provider 和 Consumer。

## React hooks 怎么模拟componentDidMount

// componentDidMount，必须加[],不然会默认每次渲染都执行

useEffect(()=>{

}, [])

## 说说在react hooks 中的坑 以及为什么？？？

不要在循环，条件或嵌套函数中调用Hook，必须始终在React函数的顶层使用Hook。这是因为React需要利用调用顺序来正确更新相应的状态，以及调用相应的钩子函数。一旦在循环或条件分支语句中调用Hook，就容易导致调用顺序的不一致性，从而产生难以预料到的后果。

## 当渲染一个列表时，何为 key？设置 key 的目的是什么

Keys 会有助于 React 识别哪些 items 改变了，被添加了或者被移除了。Keys 应该被赋予数组内的元素以赋予(DOM)元素一个稳定的标识，选择一个 key 的最佳方法是使用一个字符串，该字符串能惟一地标识一个列表项。很多时候你会使用数据中的 IDs 作为 keys，当你没有稳定的 IDs 用于被渲染的 items 时，可以使用项目索引作为渲染项的 key，但这种方式并不推荐，如果 items 可以重新排序，就会导致 re-render 变慢。

## 什么原因会促使你脱离 create-react-app 的依赖

当你想去配置 webpack 或 babel presets。

## 在 Redux 中，何为 store

Store 是一个 javascript 对象，它保存了整个应用的 state。与此同时，Store 也承担以下职责：

* 允许通过 getState() 访问 state
* 运行通过 dispatch(action) 改变 state
* 通过 subscribe(listener) 注册 listeners
* 通过 subscribe(listener) 返回的函数处理 listeners 的注销

## 何为 action

Actions 是一个纯 javascript 对象，它们必须有一个 type 属性表明正在执行的 action 的类型。

## 何为 reducer

一个 reducer 是一个纯函数，该函数以先前的 state 和一个 action 作为参数，并返回下一个 state。

## 在 React 中，refs 的作用是什么

Refs 可以用于获取一个 DOM 节点或者 React 组件的引用。何时使用 refs 的好的示例有管理焦点/文本选择，触发命令动画，或者和第三方 DOM 库集成。你应该避免使用 String 类型的 Refs 和内联的 ref 回调。Refs 回调是 React 所推荐的。

## 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。

## 使用箭头函数(arrow functions)的优点是什么

* 作用域安全：在箭头函数之前，每一个新创建的函数都有定义自身的 this 值(在构造函数中是新对象；在严格模式下，函数调用中的 this 是未定义的；如果函数被称为“对象方法”，则为基础对象等)，但箭头函数不会，它会使用封闭执行上下文的 this 值。
* 简单：箭头函数易于阅读和书写
* 清晰：当一切都是一个箭头函数，任何常规函数都可以立即用于定义作用域。开发者总是可以查找 next-higher 函数语句，以查看 this 的值

## 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。

## (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。

## 何为 JSX

JSX 是 JavaScript 语法的一种语法扩展，并拥有 JavaScript 的全部功能。JSX 生产 React "元素"，你可以将任何的 JavaScript 表达式封装在花括号里，然后将其嵌入到 JSX 中。在编译完成之后，JSX 表达式就变成了常规的 JavaScript 对象，这意味着你可以在 if 语句和 for 循环内部使用 JSX，将它赋值给变量，接受它作为参数，并从函数中返回它。

## React 的工作原理

React 会创建一个虚拟 DOM(virtual DOM)。当一个组件中的状态改变时，React 首先会通过 "diffing" 算法来标记虚拟 DOM 中的改变，第二步是调节(reconciliation)，会用 diff 的结果来更新 DOM。

## 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。

## 何为 Children

在JSX表达式中，一个开始标签(比如<a>)和一个关闭标签(比如</a>)之间的内容会作为一个特殊的属性props.children被自动传递给包含着它的组件。

## 什么是属性代理

属性代理组件继承自React.Component，通过传递给被包装的组件props得名