前端面试题-react专项：共计35道4817字

react篇

1、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

2、React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'，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或展示组件

3、React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。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。

4、什么是高阶组件

高阶组件是一个以组件为参数并返回一个新组件的函数。最常见的就是是 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。如果发现在不同的地方写了大量代码来做同一件事时，就可以用 HOC

5、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

6、(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(在子类中能在 constructor 访问 this.props。

7、React事件处理

React中的事件处理程序将传递SyntheticEvent实例，该实例是React跨浏览器本机事件的跨浏览器包装器。这些综合事件具有与您惯用的本机事件相同的界面，除了它们在所有浏览器中的工作方式相同.

> React实际上并未将事件附加到子节点本身。React将使用单个事件侦听器在顶层侦听所有事件

8、React如何创建refs

> Refs 是使用 **React.createRef()** 方法创建的，并通过 `ref` 属性添加到 React 元素上

9、什么是JSX

> JSX即JavaScript XML。一种在React组件内部构建标签的类XML语法。JSX为react.js开发的一套语法糖，也是react.js的使用基础。React在不使用JSX的情况下一样可以工作，然而使用JSX可以提高组件的可读性，因此推荐使用JSX

10、为什么不直接更新state状态

> 如果直接更新state状态，那么它将不会重新渲染组件，而是使用 `setState()` 方法。它计划对组件状态对象的更新。状态改变时，组件通过重新渲染做出响应

11、React中的这三个点（...）是做什么的

> 扩展传值符号，是把对象或数组里的每一项展开，是属于ES6的语法

12、简单介绍下react hooks 产生的背景及 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

13、 React hooks 怎么模拟生命周期

1、模拟componentDidMount

useEffect(()=>{console.**log**('第一次渲染时调用')},[])

2、模拟componentDidUpdate

没有第二个参数代表监听所有的属性更新

useEffect(()=>{console.**log**('任意属性该改变')})

同时监听多个属性的变化需要将属性作为数组传入第二个参数。

useEffect(()=>{console.**log**('n变了')},[n,m])

3、模拟componentWillUnmount

```
useEffect(()=>{
    const timer = setTimeout(()=>{
        ...
    },1000)
    return()=>{
        console.log('组件销毁')
        clearTimerout(timer)
    }
})
```

14、React 中的 `useState()` 是什么?

useState是react hooks中最常用且用法最简单的一个hook。`useState(0)` 返回一个元组，其中第一个参数`count`是计数器的当前状态，`setCounter` 提供更新计数器状态的方法。

15、当调用setState时，React render 是如何工作的

虚拟 DOM 渲染:当render方法被调用时，它返回一个新的组件的虚拟 DOM 结构。当调用setState()时，render会被再次调用，因为默认情况下shouldComponentUpdate总是返回true，所以默认情况下React 是没有优化的。
原生 DOM 渲染:React 只会在虚拟DOM中修改真实DOM节点，而且修改的次数非常少——这是很棒的React特性，它优化了真实DOM的变化，使React变得更快。

16、React 中 key 的重要性是什么?

key 用于识别唯一的 Virtual DOM 元素及其驱动 UI 的相应数据。它们通过回收 DOM 中当前所有的元素来帮助 React 优化渲染。这些 key 必须是唯一的数字或字符串，React 只是重新排序元素而不是重新渲染它们。这可以提高应用程序的性能

17、什么是Redux?

Redux 是当今最热门的前端开发库之一。它是 JavaScript 程序的可预测状态容器，用于整个应用的状态管理。使用 Redux 开发的应用易于测试，可以在不同环境中运行，并显示一致的行为

18、列出 Redux 的组件

1. **Action** – 这是一个用来描述发生了什么事情的对象。
2. **Reducer** – 这是一个确定状态将如何变化的地方。
3. **Store** – 整个程序的状态/对象树保存在Store中。
4. **View** – 只显示 Store 提供的数据

19、Redux 有哪些优点?

Redux 的优点如下：

结果的可预测性 - 由于总是存在一个真实来源，即 store ，因此不存在如何将当前状态与动作和应用的其他部分同步的问题。
可维护性 - 代码变得更容易维护，具有可预测的结果和严格的结构。
服务器端渲染 - 你只需将服务器上创建的 store 传到客户端即可。这对初始渲染非常有用，并且可以优化应用性能，从而提供更好的用户体验。
开发人员工具 - 从操作到状态更改，开发人员可以实时跟踪应用中发生的所有事情。
社区和生态系统 - Redux 背后有一个巨大的社区，这使得它更加迷人。一个由才华横溢的人组成的大型社区为库的改进做出了贡献，并开发了各种应用。
易于测试 - Redux 的代码主要是小巧、纯粹和独立的功能。这使代码可测试且独立。
组织 - Redux 准确地说明了代码的组织方式，这使得代码在团队使用时更加一致和简单

20、常用的hooks

useState：定义state的数据，参数是初始化的数据，返回值两个值1. 初始化值，2. 修改的方法

useEffect：副作用函数，顾名思义，副作用即只有使用过后才会产生副作用

当作生命周期来使用： 第二个参数如果没写的话，页面一更新触发，compoentDidMount compoentDidUpdate

第二个参数如果空数组的话，只执行一次，compoentDidMount

数组中跟某些变量，当作监听器来使用，监听数据的变化，

useEffect是一个副作用函数，组件更新完成后触发的函数

如果我们在useEffect 返回一个函数的，组件被销毁的时候触发

useMemo：用来计算数据，返回一个结果，监听数据的变化，第二个参数就是监听的数据，具有缓存性

useMemo和useEffect 相比较来说，useMemo 是组件更新的时候触发生命周期

useMemo是怎么做性能优化的?

当父组件向子组件组件通信的时候，父组件中数据发生改变，更新父组件导致子组件的更新渲染，但是如果修改的数据跟子组件无关的话，更新子组件会导致子组件不必要的DOM渲染，是比较消耗性能的，这个时候我们可以使用useMemo或者memo做组件的缓存，减少子组件不必要的DOM渲染

useCallback：当父组件向子组件传递函数的时候，父组件的改变会导致函数的重新调用产生新的作用域，所以还是会导致子组件的更新渲染，这个时候我们可以使用useCallback来缓存组件

useRef：相当于createRef的使用，创建组件的属性信息

useContext：相当在函数组件中获取context状态数的内容信息

useReducer：useReducer是用来弥补useState的补不足， 可以把数据进行集中式的管理，单独处理数据的逻辑信息

21、为什么浏览器无法阅读JSX?

浏览器只能读取JavaScript对象，而不能读取普通JavaScript对象中的JSX。因此，要使浏览器能够读取JSX，首先，我们需要使用Babel之类的JSX转换器将JSX文件转换为JavaScript对象，然后将其传递给浏览器。

22、什么是高阶成分（HOC）?

高阶组件是重用组件逻辑的高级方法。基本上，这是从React的组成性质衍生的模式。HOC是自定义组件，在其中包裹了另一个组件。他们可以接受任何动态提供的子组件，但不会修改或复制其输入组件中的任何行为。您可以说HOC是“纯”组件。

23、React的严格模式如何使用，有什么用处?

`StrictMode` 是一个用来突出显示应用程序中潜在问题的工具。与 `Fragment` 一样，`StrictMode` 不会渲染任何可见的 UI。它为其后代元素触发额外的检查和警告。

24、React中什么是受控组件和非控组件?

（1）受控组件 在使用表单来收集用户输入时，例如