## 【React】常见面试题总结

Alley alley [IT狗的成长日记](javascript:void(0);) \*20191018\*

001、请说下在react中如何处理异步的action

通过applyMiddleware来使用中间件来处理action

常用的中间件

​ reduxpromisemiddleware

​ reduxthunk

​ reduxsaga

\*\*002、render函数什么时候会执行？\*\*

当this.state 或者 this.props发生改变的时候render函数就会执行

\*\*003、react中如何对state中的数据进行修改？setState为什么是一个异步的\*\*

修改数据通过this.setState(参数1,参数2)

this.setState是一个异步函数

参数1 : 是需要修改的数据是一个对象

​ 参数2 : 是一个回调函数，可以用来验证数据是否修改成功，同时可以获取到数据更新后的DOM结构 等

​ 同于componentDidMount

this.setState中的第一个参数除了可以写成一个对象以外，还可以写成一个函数，函数中第一个值为prevState 第二个值为prePprops this.setState((prevState,prop)=({}))

为什么setState是一个异步的？

​ 当批量执行state的时候可以让DOM渲染的更快,也就是说多个setstate在执行的过程中还需要被合并

\*\*004、react中如何定义自定义属性，以及限制外部数据的类型\*\*

1、自定义属性

​ 组件名称.defaultProps= { key:val }

2、限制外部数据的类型

引入propTypes第三方模块

类型限制 组件名称.propTypes = { key:propTypes.类型 }

\*\*005、react路由常用的一些组件配置项有哪些？\*\*

BrowserRouter

HashRouter

withRouterRoute

Link

NavLink

Switch

Redirect

\*\*006、reatc路由中Route渲染组件的方法有哪几种？区别是什么？\*\*

渲染方式有2种

​ `<Route path="/home" component={组件名称}</Route` 通过component进行组件的渲染，这种方式的优点在于可以直接在组件的内部接受到history、location、match,缺点在于如果需要组件传参，或者渲染jsx语法的时候无法使用

​ `<Route path="/home" render={()={ return <Home/ }}</Route` 通过render进行渲染组件，优点在于可以进行组件传参，还可以渲染非组件的标签，缺点在于如果需要使用 histroy location match的话需要在函数中传递过去

\*\*007、如何控制路由的路径完全匹配\*\*

给NavLink 或者 Route设置 exact属性

008、react中路由传递参数的方法有哪些？

1、动态路由传参数

2、query传值

\*\*009、react中的编程式导航方法有哪些？\*\*

this.props.history.push("需要跳转的路径")；

this.props.history.back()

this.props.history.forward()

this.props.history.replace()

\*\*010、react中的生命周期有哪些？\*\*

(1)、组件创建的过程

constructor

componentWillMount

render

componentDidMount

(2)当props中的数据发生改变后会执行哪些生命周期函数

componentWillReceiveProps

shouldComponentUpdate

componentWillUpdate

render

componentDidUpdate

(3)组件销毁的时候 componentWillUnmount

\*\*011、reatc中如何强制更新DOM\*\*

this.foreUpdate()

\*\*012、谈谈你对react生命周期中shouldComponentUpdate的理解\*\*

shouldComponentUpdate是react性能优化非常重要的一环。组件接受新的state或者props时调用，我们可以设置在此对比前后两个props和state是否相同，如果相同则返回false阻止更新，因为相同的属性状态一定会生成相同的dom树，这样就不需要创造新的dom树和旧的dom树进行diff算法对比，节省大量性能，尤其是在dom结构复杂的时候

\*\*013、谈谈你对虚拟DOM的理解，以及好处？\*\*

虚拟DOM：虚拟DOM其实就是真实的js对象虚拟DOM提高了react的性能，每次数据更新后都会重新计算上虚拟DOM，并和上一次的虚拟DOM进行对比，对方法变化的部分进行批量更新。react中也提供了shouldComponentUpdate生命周期的回调，来减少数据变化后不必要的虚拟DOM对比过程。保证性能

\*\*014、谈谈你对flux的理解\*\*

利用单项数据流的方式来组合react的视图组件，它是一种模式，而不是一种框架简单来说flux可以帮我们解决非父子组件之间传值flux数据传递流程 https://www.cnblogs.com/nanianqiming/p/9870194.html

\*\*015、react17废除的生命周期函数，与新增的生命周期函数\*\*

由于未来采用异步渲染机制，所以即将在17版本中去掉的生命周期钩子函数

​ componentWillMount

​ componentWillRecieveProps

​ componentWIllUpdate

新增的生命周期

​ static getDerivedStateFromProps(nextProps, prevState) {} 用于替换componentWillReceiveProps，可以用来控制 props 更新 state 的过程；它返回一个对象表示新的 state；如果不需要更新，返回 null 即 可在每次渲染之前都会调用，不管初始挂载还是后面的更新都会调用，这一点 和componentWillReceiveProps不同（只有当父组件造成重新渲染时才调用） 每次都应该返回一个对象作为

​ getSnapshotBeforeUpdate() {} 用于替换 componentWillUpdate，该函数会在update后 DOM 更新前被调用， 用于读取最新的 DOM 数据，返回值将作为 componentDidUpdate 的第三个参 数,在最新的渲染数据提交给DOM前会立即调用，它让你在组件的数据可能要改 变之前获取他们

​ componendDidCatch(error, info){} 如果一个组件定义了componentDidCatch生命周期，则他将成为一个错误边 界(错误边界会捕捉渲染期间、在生命周期方法中和在它们之下整棵树的构造 函数中的错误，就像使用了try catch，不会将错误直接抛出了，保证应用的 可用性)

\*\*016、this.setState之后react做了哪些操作\*\*

shouldComponentUpdate

componentWillUpdate

render

componentDidUpdate

\*\*017、当组件的key值发生改变后会执行哪些生命周期函数？\*\*

componentWillUnmount

constructor

componentWillMount

render

componentDidMount

\*\*018、在react中是否封装过组件？封装组件需要注意的地方？\*\*

常用封装组件用到的东西1、propTypes 限制外部数据的类型2、defaultProps 设置默认的外部数据3、父传子 子传父 context 等传值方式4、高阶组件封装组件的时候一定要注意组件的复用性，灵活性

\*\*019、如何接收组件内部的标签/内容\*\*

通过this.props.children

\*\*020、请说下你对redux的理解，以及使用的方式\*\*

其实redux就是Flux的一种进阶实现。它是一个应用数据流框架,主要作用应用状态的管理redux的三大原则 1、单一的数据源 2、state是只读的 3、使用纯函数来进行修改

redux中有3个常用的方法

​ 1、createStore() 创建store

​ 2、combineReducers() 合并多个reducer

​ 3、applyMiddleware() 使用中间件，处理异步的actionredux数据传递的流程

当组件需要改变Store数据的时候。需要创建一个Action,然后通过 dispatch(action) 传递给Store，然后Store把Action转发给Reducers. Reducers会拿到previousState(以前的state数据) 和action。然后将 previousState和action进行结合做新的数据(store)修改。然后生成一个 新的数据传递给Store 。Store通过触发subscribe()这个方法来调用函数 执行setState使得view的视图发生改变

002、请说下redux中中间件的理解

中间件:请求和回复之间的一个应用。

在redux中中间件是dispatch与reducer之间的一个应用

003、请说下redux中你对reducers的理解,以及如何合并多个reducers

在redux中reducer是一个纯函数，其中这个纯函数会接收2个参数 一个是state 一个是action。state用来保存公共的状态，state有一个特点就是只允许读不允许进行修改 。

在实际开发中因为会涉及到多人协作开发所以每个模块都会有一个reducer。我们可以通过combineReducers来合并多个reducers

004、请说下你对高阶组件的理解，以及作用

高阶组件就是接受一个组件作为参数，返回一个相对增强性的组件的函数

作用：

​ 1、属性代理主要进行组件的复用

​ 2、反向集成主要进行渲染的劫持

005、在react中如何解决单页面开发首次加载白屏现象

1、通过路由懒加载的方式 reactloadable

2、首屏服务端渲染

006、请说下react中key值的理解

react利用key来识别组件，它是一种身份标识标识，相同的key react认为是同一个组件，这样后续相同的key对应组件都不会被创建

有了key属性后，就可以与组件建立了一种对应关系，react根据key来决定是销毁重新创建组件还是更新组件。

key相同，若组件属性有所变化，则react只更新组件对应的属性；没有变化则不更新。

key值不同，则react先销毁该组件(有状态组件的componentWillUnmount会执行)，然后重新创建该组件（有状态组件的constructor和componentWillUnmount都会执行）

007、组件第一次执行的时候会执行哪些生命周期函数

constructor

componentWillMount

render

componentDidMount

008、哪些生命周期会执行多次

render

componentWillReceiveProps

shouldComponentUpdate

componentWillUpdate

componentDidUpdate

009、当this.state 或者this.props发生改变的时候会执行哪些生命周期函数

this.props 发生改变

componentWillReceiveProps

shouldComponentUpdate

componentWillUpdate

render

componentDidUpdate

this.state发生改变

shouldComponentUpdate

componentWillUpdate

render

componentDidUpdate

010、谈谈你对context的理解

当你不想在组件树中通过逐层传递props或者state的方式来传递数据时，可以使用Context来实现跨层级的组件数据传递,使用context可以实现跨组件传递

新版context的基本使用

​ 1、根组件引入GlobalContext，并使用GlobalContext.Provider（生产者）

​ 2、组件引入GlobalContext并调用context，使用GlobalContext.Consumer

011、谈谈你对react中withRouter的理解

默认情况下必须是经过路由匹配渲染的组件在this.props上拥有路由参数，才能使用编程式导航的写法，withRouter是把不是通过路由切换过来的组件，将reactrouter 的 history、location、match 三个对象传入props对象上

简单的来说就是不是路由包裹的组件也可以使用路由的一些方法

012、说以下你对puerComoponent的理解

pureComponent表示一个纯组件，可以用来优化react程序。减少render函数渲染的次数。提高性能

pureComponent进行的是浅比较，也就是说如果是引用数据类型的数据，只会比较不是同一个地址，而不会比较这个地址里面的数据是否一致

好处:当组件更新时，如果组件的props或者state都没有改变，render函数就不 会触发。省去虚拟DOM的生成和对比过程，达到提升性能的目的。具体原因 是因为react自动帮我们做了一层浅比较

013、react请求接口数据是在componentDidMount 还是componentWillMount周期好

React异步渲染开启的时候，componentWillMount 就可能被中途打断，中断之后渲染又

要重做一遍，如果在componentWillMount 中做 AJAX 调用，代码里看到只有调用一次，但是实际上可能调用 N 多次，这明显不合适。相反，若把 AJAX 放在componentDidMount，因为 componentDidMount 在第二阶段，所以绝对不会多次重复调用，这才是 AJAX 合适的位置

\*\*持续更新！\*\*\*\*！\*\*\*\*！\*\*

​