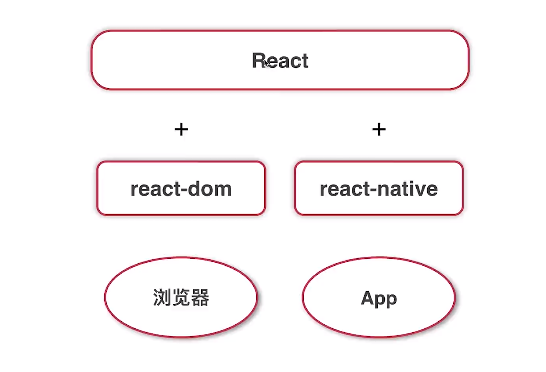
1 浏览器与移动端的区别

用react写出来的代码，既适合浏览器又适合手机app运行

react-dom 封装了很多跟浏览器相关的东西, 比如获取浏览器的dom元素

React-native 封装了跟手机相关的api,比如打开手机相机等



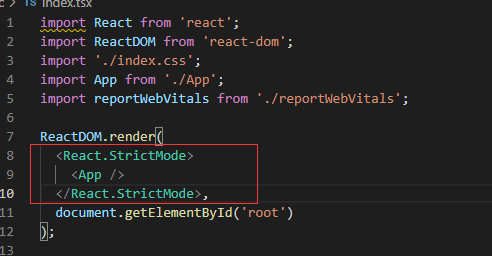


2 生成支持typescript的 react项目

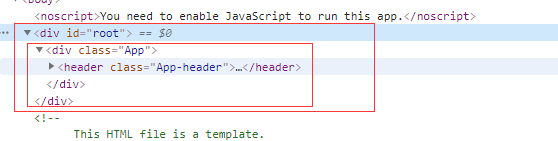
npx create-react-app react-antd-ts-again --template typescript

3

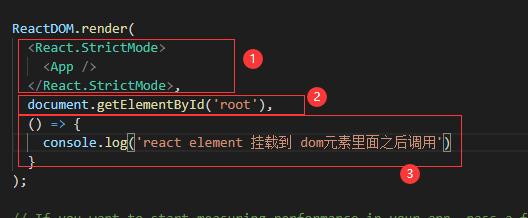
ReactDOM.render, 把react element塞到后面的dom元素里面，



不是替换。



ReactDOM.render是异步的，有三个参数



4-1

02：44

ReactDom.render的渲染机制，有两种



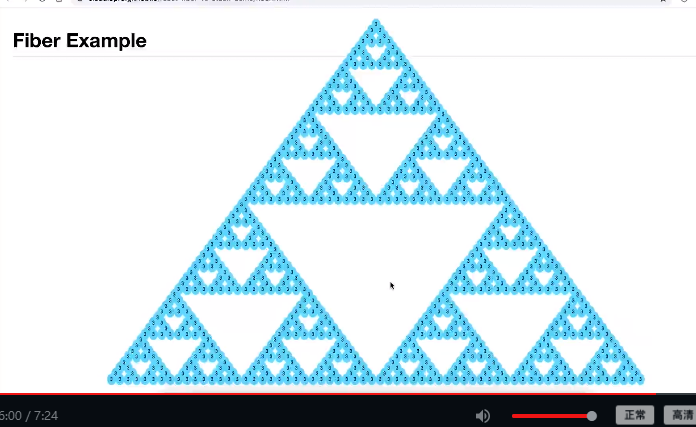
Fiber react16之后使用

Stack react16之前，后来发现有点问题。

两者区别  
react 新dom跟老dom进行diff算法，比对哪些需要更新。

Stack 等所有的比对结束后，再更新页面，比对的时候是不更新的，但比对是个很好性能的工作，导致出现一卡一卡的感觉。

Fiber 把比对的过程拆成了多个部分，一部分比对结束了，就更新这一部分，不是等全部比对后再更新，这样页面看起来就比较顺滑，做到至少60帧的频率，也就是16ms/帧，这个时间把需要比对的部分比对玩并且渲染到页面上。



reactDom.unmountComponentAtNode

卸载的两种方式

1

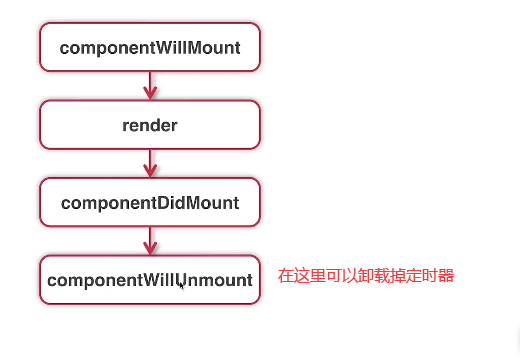
((document.getElementById('root') as HTMLElement).firstChild as HTMLElement).remove()

2

ReactDOM.unmountComponentAtNode(document.getElementById('root') as HTMLElement)

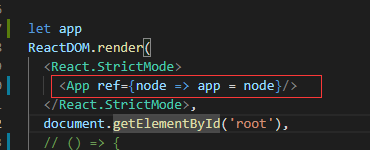
区别，第一种不会调用， componentWillUnMount

第二种会调用componentWillUnMount

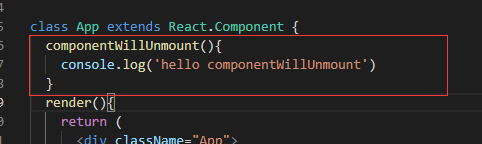


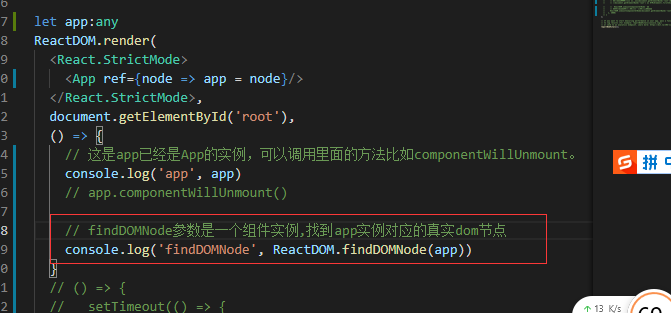
3 reactDOM.findDomNode 官方不推荐，但是有时操作dom的场景让我们不得不使用它。

ref={node => app = node}， 当App这个组件被挂载到dom节点之后，就调用这个方法，方法的node参数，就是这个App实例。然后app存储的就是这个实例了。

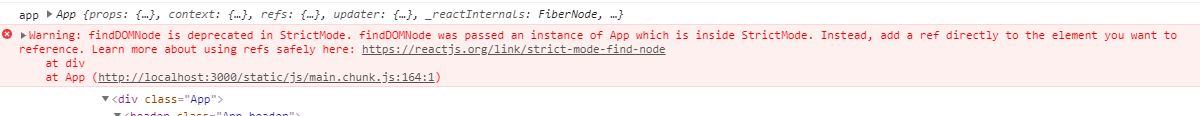








但是会有个warning

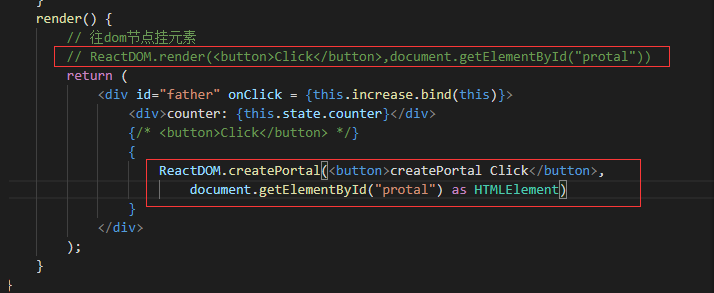


4-5

ReactDOM.createPortal，把react element 渲染到dom节点<div id="protal"></div>之后，会把生成的真实dom节点的冒泡机制，计算到<div id="father"></div>上，认为还是在写

ReactDOM.createPortal(<but。。。的地方

使用场景：弹框的遮罩在组件外面，但是要算是组件里的元素，正常冒泡。



效果

