基于实例的react傻瓜课程

1-1

React是什么，特点

1 声明式的编程，强调结果，相对的是命令式的编程，强调过程

2 组件化，组件让代码的复用，测试，分离都更加简单

1-2 搭建环境

Create-react-app

需要node和npm

node版本要大于 6.5

Npm i create-react-app -g, 全局安装，就已经是指定了环境变量

1.5.2

Create-react-app my-project

1-4

class WelcomeRender extends Component {

    render(){

        return <h1>welcome render</h1>

    }

}

render方法return一个节点

import ReactDOM from 'react-dom';

ReactDOM.render

是处理react和dom的相关事宜的，接收两个参数，一个是渲染的html,一个是挂载的节点

ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);

2-1 jsx语法

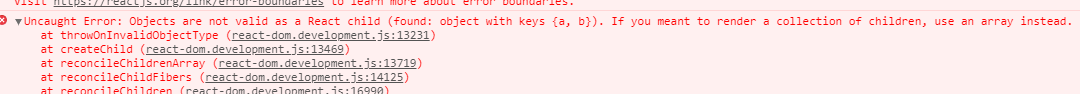
1：19 jsx语法只有一个花括号的概念, 花括号会返回一个值

，里面可以是javascript表达式,如： {‘abc’}, {5 + 5}

或者 dom节点 {<p>456<p>}，像里面只有一个dom节点，没有花括号也是一样的

但不能是对象{a:1,b:2},会报错

{/\* { obj } 如果是对象，会报错\*/}



jsx语法的属性跟html标签属性一致，除了两个，

一是 class, 因为和Class保留字一样，所以样式的类改成className

另一个是for,也是和保留字一致，要改成htmlFor



2-2 jsx第二部分

0220

jsx语法只是语法糖，是React.createElement()的语法糖，返回的是reactDom

简单明了，比用React.createElement简洁

返回的是React elements对象

使用babel在线编译工具

自定义类手写字母要大写，否则会被babel认为是普通的html标签

首字母小写

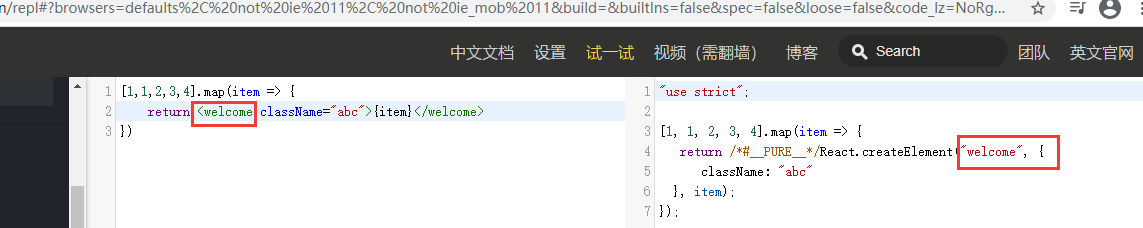
<welcome

被认为是

React.createElement("welcome", {

className: "abc"

}, item);



首字母大写，会被认为是一个类

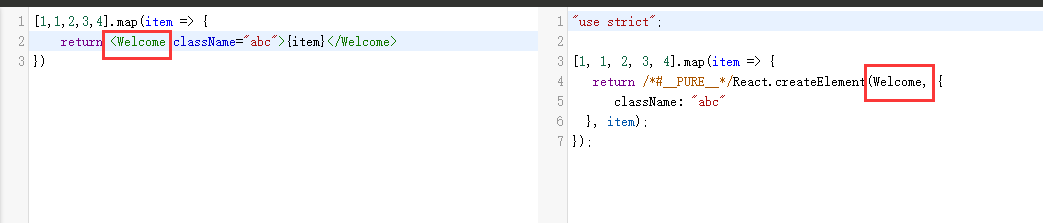
<Welcome

被认为

React.createElement(Welcome, {

className: "abc"

}, item);



使用连字符，会被认为是普通的html标签

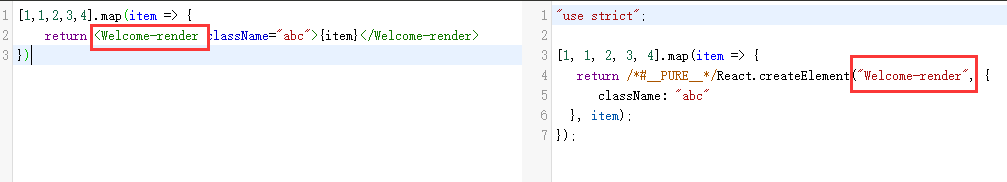
<Welcome-render

被编译成是

React.createElement("Welcome-render", {

className: "abc"

}, item);



不使用连字符，会被认为是个类

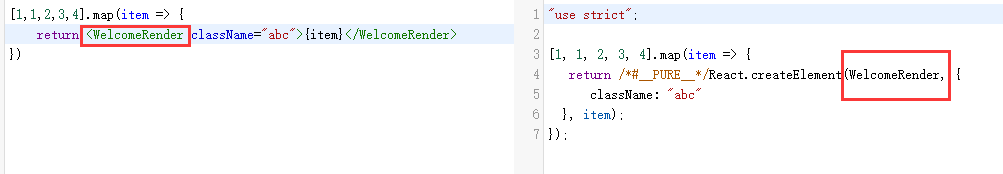
<WelcomeRender

被编译成

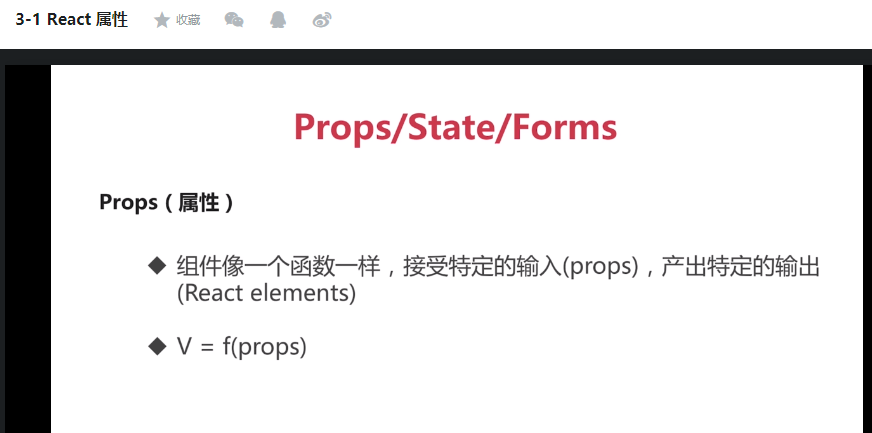
React.createElement(WelcomeRender, {

className: "abc"

}, item);



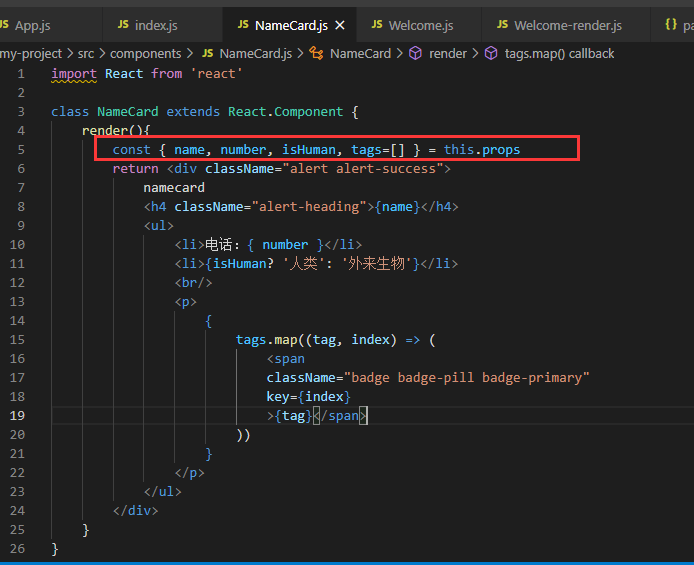
3-1 react组件属性



使用bootstrap 4.1.1

import 'bootstrap/dist/css/bootstrap.min.css'

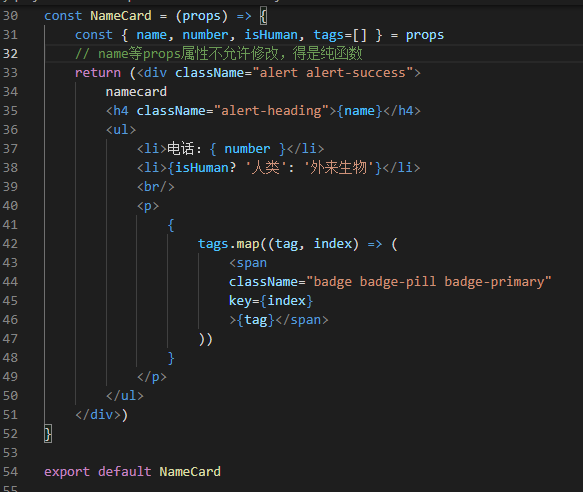
组件内



props传值



函数式写法



纯函数的概念：入参不允许改变

react组件里不允许改变 props上的属性

3-2 React 状态

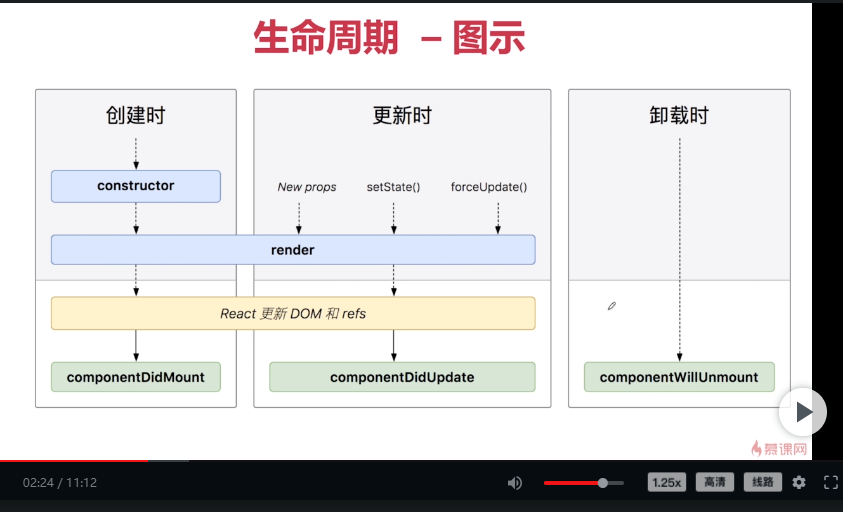
本来让组件重新渲染的话，用的是ReactDom.render方法，

还可以用state来控制，不改变props，但可以通过setState来改变state,从而更新视图



3-3 React生命周期





这个钩子函数接受两个参数，

componentDidUpdate(currentProps, currentState){

        // 接受两个参数，当前的props, 当前的state

        console.log('componentDidUpdate', currentProps, currentState)

    }

3-4 React表单

受控组件





受控组件，表单的所有状态都由组件控制，这可能变得非常繁琐

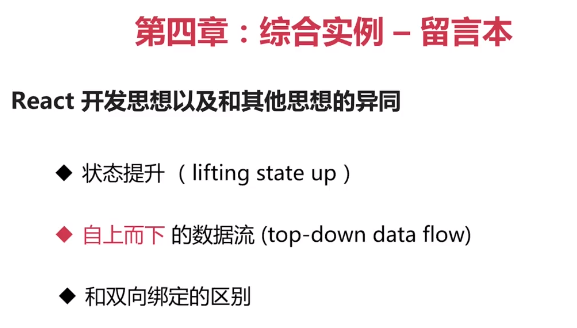
可以考虑使用非受控组件，使用ref,来操作dom节点

{/\* ref={(textInput) => {this.textInput = textInput}}

                    this.textInput可以拿到dom节点

                \*/}

4-1



状态提升

多个组件使用相同的数据，通常是放到父组件，称为状态提升

单向数据流

好处有2：

1缩小定位bug的范围，因为拥有状态的组件，也只有本身能修改状态，

2 可以使用自定义逻辑来更改用户的输入

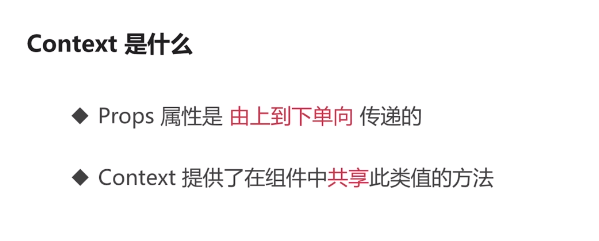
在react中对于可变数据，理应只有单一数据源，这里放到父组件app.js里

单向数据流，不同于双向数据绑定，父传给子

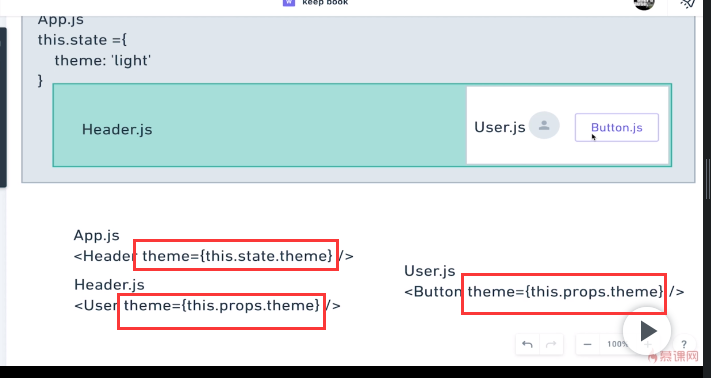
如果需要修改数据，比如修改comments，也是在父组件修改，只不过父组件传个方法给子组件，

子组件负责调用而已

5-1 Context 介绍



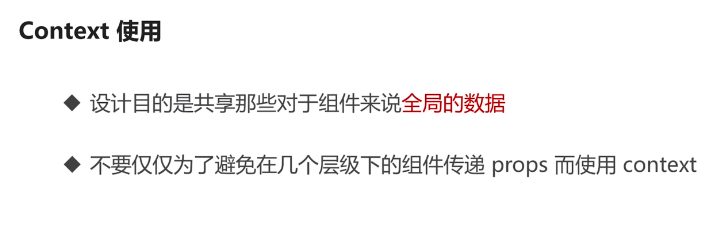
比如



Context是react 16.3 才有的，要使用得把react升级到16.3以上

全局的数据

比如当前用户，主题，首选语言





4-2 0129

当一个组件没有生命周期，也没有state的时候，就可以写成stateless（无状态）组件，即写成function 的形式

App.js两个子组件 CommentLists.js 和 CommentBox2.js

注意：

循环的生成 ReactDom要加key

<li key={index} className="list-group-item">{item}</li>

绑定方法要指定this

this.addComment = this.addComment.bind(this)

5-2 编码

Context 不属于某个组件，因此要放到src目录下

在App.js引入后，如何使用

const ThemeContext = React.createContext()

// 提供了两个组件ThemeContext.Provider ThemeContext.Consumer

//                            提供者     消费者

用这个包裹根节点

<ThemeContext.Provider value={themes.dark}>

</ThemeContext.Provider>

里面的组件就可以拿到了

function ThemeBar (){

    return <ThemeContext.Consumer>

        {

            (theme) => {

                console.log('value', theme)

            }

        }

    </ThemeContext.Consumer>

}

