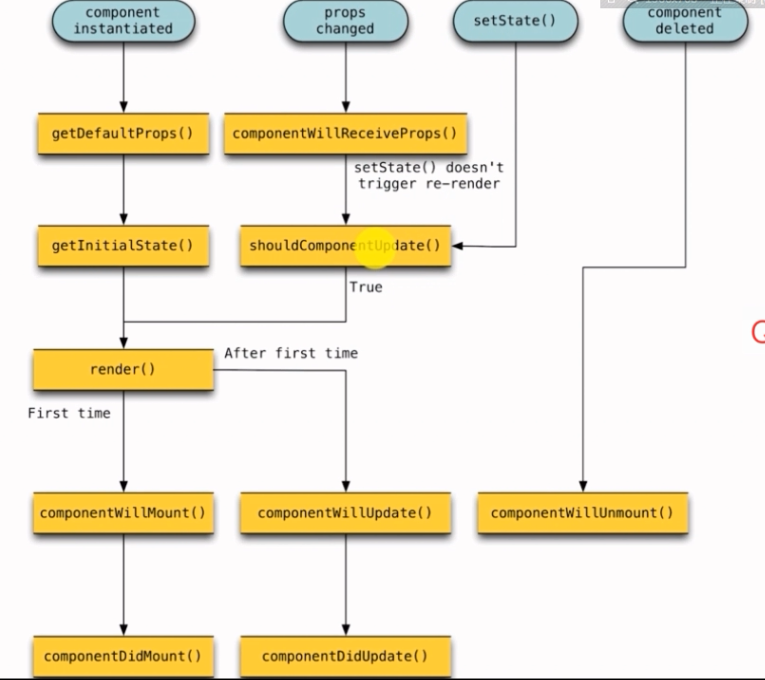
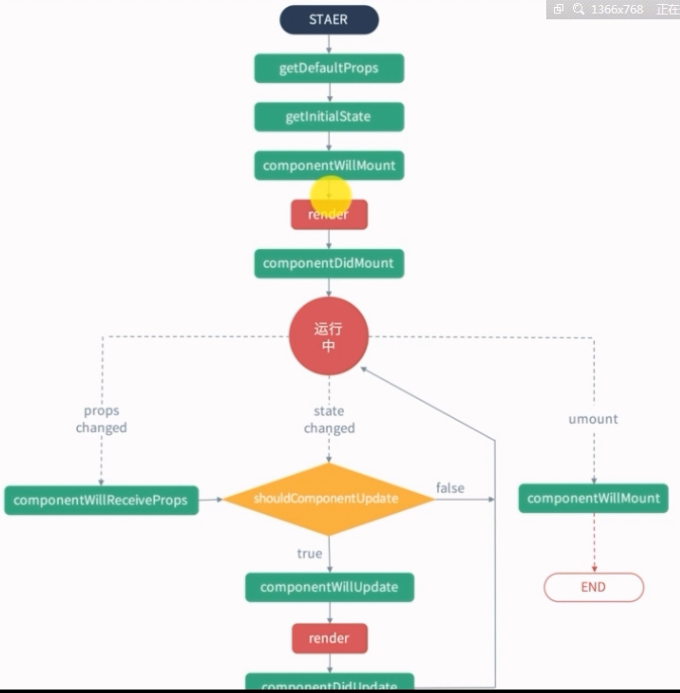
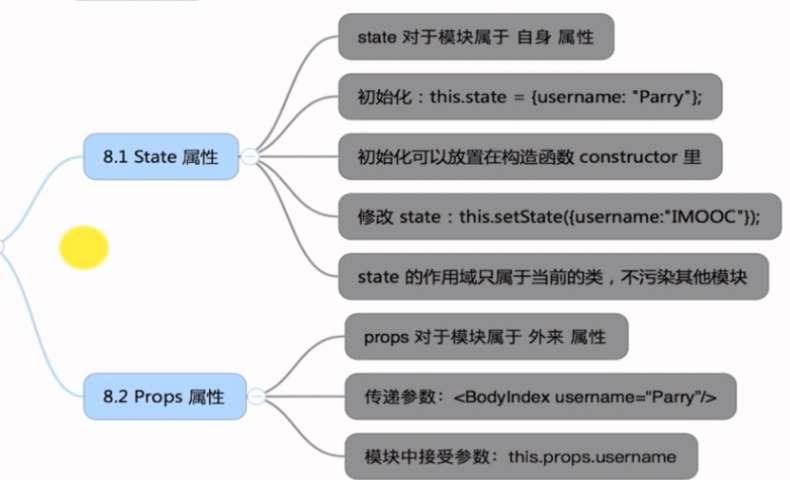
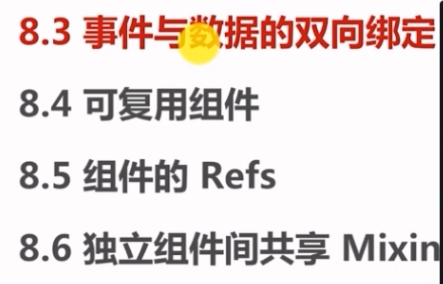
# React





## 属性和事件





### 1、事件和数据的双向绑定

import React, { Component } from 'react'

import BodyChild from './bodychild';

export default class BodyIndex extends Component {

constructor() {

super();

this.state = { age: 20 };

}

changeAge(event) {

this.setState({

age: event.target.value

});

}

render() {

return (

<div>

age {this.state.age}

<BodyChild changeAge={this.changeAge.bind(this)} />

</div>

)

}

}

import React, { Component } from 'react'

export default class BodyChild extends Component {

render() {

return (

<div>

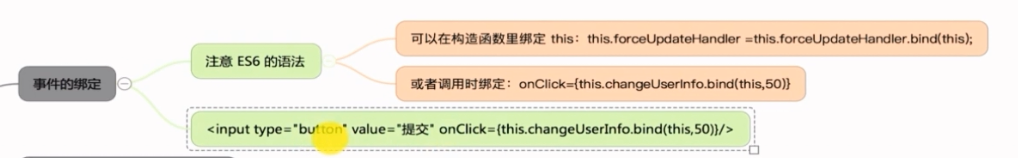
<input type="text" onChange={this.props.changeAge}/>

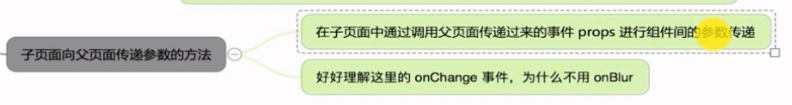
</div>

)

}

}





### 2、可复用组件

#### 属性的类型检查和默认值设置

import React, { Component } from 'react';

import PropTypes from 'prop-types';

const defaultProps={

greet:'这是一个默认的祝福语'

};

export default class BodyChild extends Component {

render() {

return (

<div>

<h1>{this.props.greet}</h1>

<input type="text" onChange={this.props.changeAge} />

</div>

)

}

}

BodyChild.propTypes = {

greet: PropTypes.string.isRequired

};

BodyChild.defaultProps=defaultProps;

#### 当需要从根节点传递参数到最下面的子节点时

App.js

<BodyIndex userId={123} username={'liumingyu'}/>

Bodyindex.js

<BodyChild {...this.props} id={'213a'} changeAge={this.changeAge.bind(this)} />

Bodychild.js

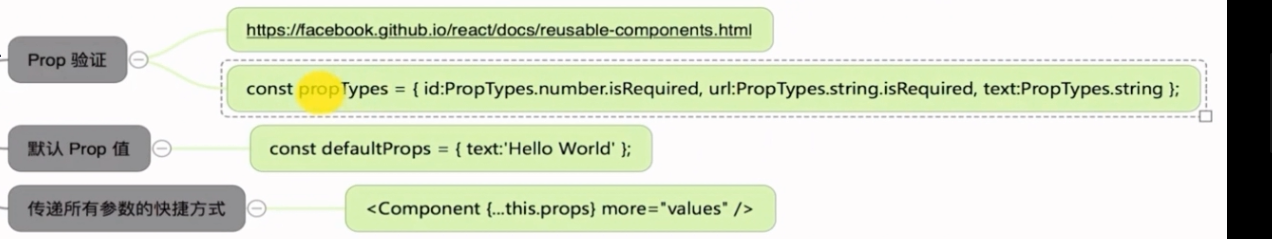
<p>{this.props.userId}

{this.props.username}

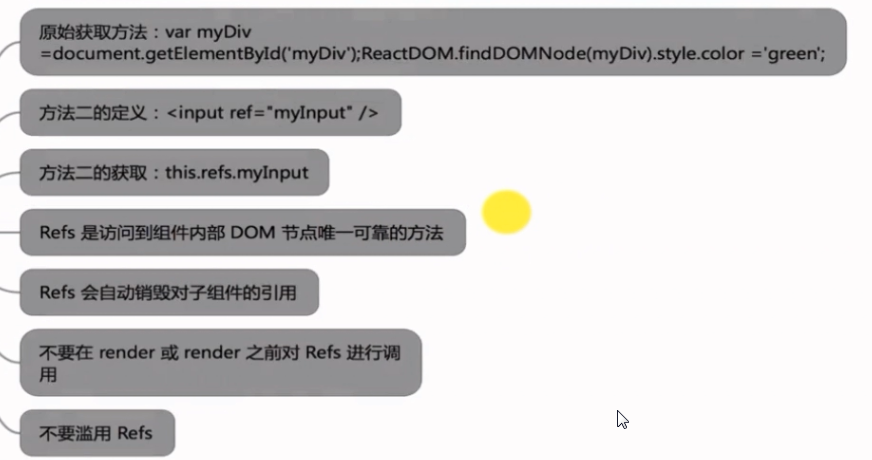
{this.props.id}

</p>

主要是用es6的语法，对象的扩展运算符。



### 3、组件的refs 获取dom节点的方法



不要在render或render之前对refs进行调用。

因为在两个时间点，组件还没有加载好。

<button id="btn" ref="btn" onClick={this.changeAge.bind(this)}>Hello world</button>

changeAge(event) {

this.setState({

age: event.target.value

});

// 第一种方式

// var btn=document.getElementById("btn");

// ReactDOM.findDOMNode(btn).style.color='red';

// 第二种方式

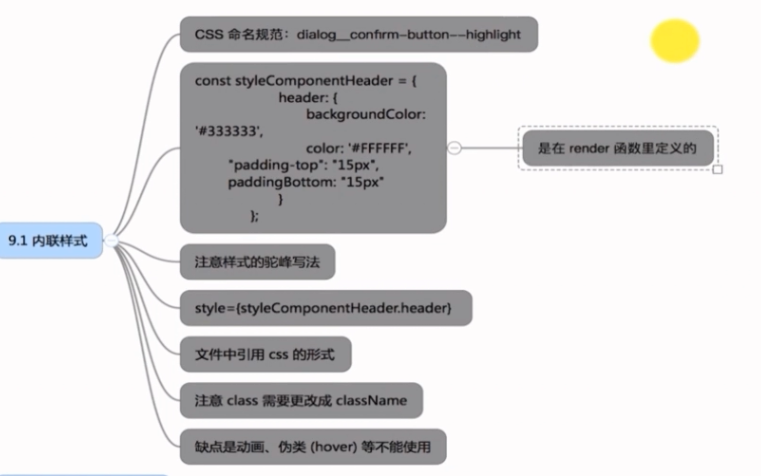
this.refs.btn.style.color='red';

}

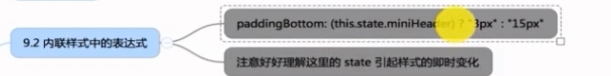
### 4、独立组件间的共享mixins

## react样式

### 1、内联样式

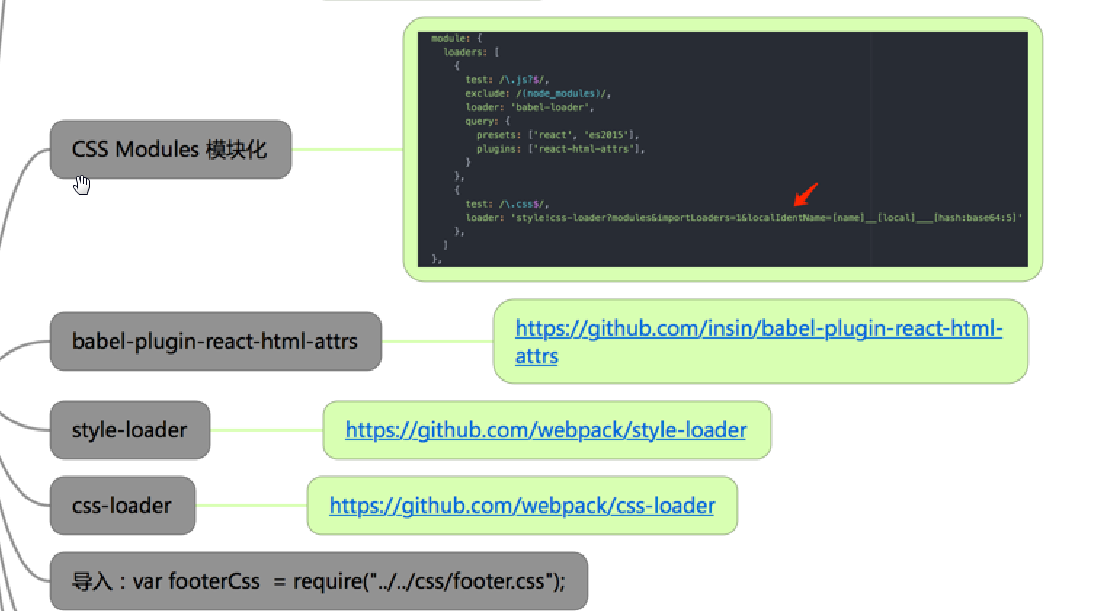


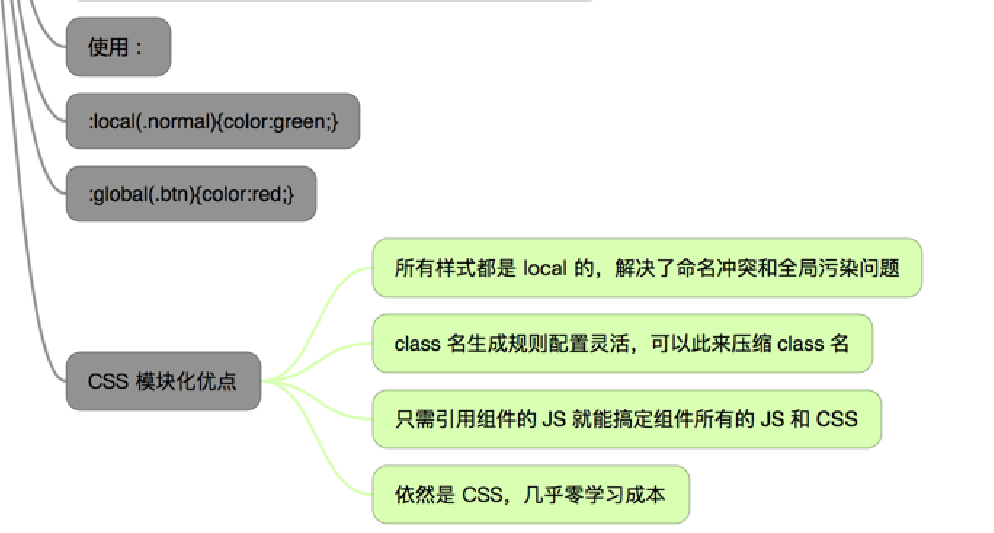
### 2、内联样式中的表达式



### 3、Css模块化









## React router

## 启动命令



Webpack-dev-server –contentbase src –inline –hot

## 使用ant design

### 1. 安装和初始化

npm install -g create-react-app

npm install –save antd

修改 src/App.js，引入 antd 的按钮组件。

import React, { Component } from 'react';

import Button from 'antd/lib/button';

import './App.css';

class App extends Component {

render() {

return (

<div className="App">

<Button type="primary">Button</Button>

</div>

);

}

}

export default App;

修改 src/App.css，在文件顶部引入 antd/dist/antd.css。

@import '~antd/dist/antd.css';

.App {

text-align: center;

}

### 2.高级配置

## Iconfinder 网站

<https://www.iconfinder.com>

## 登录模块



## 安装的插件

1、react-responsive

2、fetch

## 问题：

### 1.为什么在这里用onChange事件，为什么不用onBlur事件

### 2.为什么修改state的时候用setState 而不用等于号

### 3.为什么用refs获取dom节点是可靠的

### 4.图片地址怎么用。非得用import语句吗

### 5.路由嵌套多层的话，发现在子组件中的子组件，点击title，浏览器url变化

一直加 deails

<http://localhost:3000/details/details/details/details/161028192308012>

### 6.另一个问题是页面的样式统一写在一个css文件中不好用。

### 7.样式到底是写在组件里面还是写在统一的一个文件中

### 8.实战中mobile端路由嵌套

MobileHeader ,MobileContainer中的导航菜单，

怎么把导航菜单的mobileList列表和详情做成路由。

# 注意

## 1. 创建provider 需要注意几点

1.provider 中需要定义getChildContext方法和 指定 childContextTypes

import React, { Component } from 'react'

import PropTypes from 'prop-types';

export default class Provider extends Component {

getChildContext() {

return {

store: this.props.store

};

}

render() {

return this.props.children;

}

}

Provider.propTypes = {

store: PropTypes.object.isRequired

}

Provider.childContextTypes = {

store: PropTypes.object

};

2.在需要用到context对象的组件中需要定义contextTypes

CounterContainer.contextTypes ={

store:PropTypes.object

}

# 看深入浅出react和redux

## 问题

### 1.ref的应用

<input className="new-todo" ref={this.refInput}/>

constructor(){

super(...arguments);

this.onSubmit=this.onSubmit.bind(this);

this.refInput=this.refInput.bind(this);

}

refInput(node) {

this.input = node;

}