

React基础知识速查表

知识点	涉及语法		备注
React引入	<pre><!-- 引入react核心库，需要按顺序引入 --> <script type="text/javascript" src="../js/react.development.js"></script> <!-- 引入react-dom，用于支持react操作DOM --> <script type="text/javascript" src="../js/react-dom.development.js"></script> <!-- 引入babel，用于将jsx转为js --> <script type="text/javascript" src="../js/babel.min.js"></script></pre>		
React基本使用	<pre><!-- 准备好一个容器 --> <div id="test"></div> <script type="text/babel"> // 此处必须写babel // 1. 创建一个虚拟DOM let VDOM = <h1>Hello, React! </h1> // 2. 让React 将虚拟DOM (VDOM) 插入页面，语法: ReactDOM.render(虚拟DOM, 目标容器) ReactDOM.render(VDOM,document.getElementById('test')) </script></pre>		
虚拟DOM	创建方式一：用jsx语法创造	<pre>let VDOM = (<h1 id="atguigu"> Hello, React! </h1>)</pre>	
	创建方式二：用js语法创造 createElement(标签名, 属性, 内容)	<pre>let VDOM = React.createElement('h1',{id:'atguigu'},'Hello, React!')</pre>	
JSX (JavaScript XML)	<ul style="list-style-type: none">* jsx是react定义的一种类似于XML的JS扩展语法：JS + XML* 本质是React.createElement(component, props, ...children)方法的语法糖		
	<p>jsx语法规则：</p> <ul style="list-style-type: none">* 1. 创建虚拟DOM时，不要写引号；* 2. 如果标签中要混入【js表达式】，要使用{}，注释也要先包起来再注释* 3. 标签中样式的类名要用className来指定* 4. 标签中的内联样式要用style={{color:'white'}}，属性名转为小驼峰* 5. VDOM每次创建只能有一个根标签* 6. 标签必须闭合（单标签加“/”自闭合）* 7. 关于标签首字母：<ul style="list-style-type: none">* 1) 若首字母小写，那么React就会去寻找与之同名的<html标签>* 2) 若首字母大写，那么React就会去寻找与之同名的组件（component），		
React定义组件	1 函数式组件 (简单组件)	<pre>function Demo(){ // console.log(this) return <h2>我是用函数定义的组件</h2> } * 函数里的this是谁？ undefined，因为经过babel翻译，开启了严格模式</pre>	<p>ReactDOM.render(<Demo/>...)后发生了什么？</p> <ul style="list-style-type: none">* 1. React解析组件标签，寻找Demo组件的定义位置* 2. React发现Demo组件是用函数定义的，随后React去直接调用Demo函数，将返回值渲染到页面
	2 类式组件 (复杂组件)	<pre>class Demo extends React.Component{ render(){ // console.log(this); return <h2>我是用类定义的组件</h2> } } * 1. render方法是放在哪里的？ Demo的原型对象上，供实例使用 * 2. render中的this是谁？ Demo类（Demo组件）的实例对象 => 也称“Demo组件对象”</pre>	<p>ReactDOM.render(<Demo/>...)后发生了什么？</p> <ol style="list-style-type: none">1. React解析组件标签，寻找Demo组件的定义2. React发现Demo组件是用类定义的，React实例创建一个Demo的实例对象D3. 通过D去调用到了render方法

React基础知识速查表

知识点	涉及语法		备注	
State	State 基本使用	<pre>class Weather extends React.Component{ constructor(props) { super(props) this.state = {isHot:false} // 初始化状态 this.changeWeather = this.changeWeather.bind(this) } render(){ const { isHot } = this.state return <h1 onClick={this.changeWeather}>今天天气很 {isHot ? '炎热' : '凉爽'}</h1> } changeWeather(){ const { isHot } = this.state this.setState({isHot:!isHot}) } }</pre>	<p>1.changeWeather函数是放在哪里的？ Weather的原型对象上，供实例使用</p> <p>2.changeWeather函数中的this是谁？ 如果changeWeather是通过Weather实例调用的，那么this就是Weather的实例对象。</p> <p>然而，此时的changeWeather是作为点击事件的回调函数，根本不是通过Weather的实例调用的，而且类中的方法自动开启了严格模式，所以this是undefined（this丢失）。</p> <p>VDOM里的changeWeather的问题：</p> <p>1. 为什么加this：因为changeWeather放在了原型对象上，所以这样指定回调函数。</p> <p>2. 为什么把changeWeather()放在类里：因为要构成一个完整的组件。</p> <p>3. 为什么里面不加小括号：因为JS解析的原理中，加小括号就直接运行了。</p>	
	State 简写方式	<pre>class Weather extends React.Component { state = { isHot: false } render() { ...内容同上 } changeWeather = ()=> { const { isHot } = this.state this.setState({isHot:!isHot}) } }</pre>	<p>* 类式组件中的构造器完全可以省略。若要写构造器，必须调用super，如果还需要在构造器中通过this.props取值，那么props要传给super</p> <p>* 组件类中，程序员定义的事件回调，必须写成赋值语句+箭头函数的形式，避免了this为undefined的问题</p> <p>* 注意：State不可以直接修改，要用专门的API——this.setState() 去修改</p>	
props	props 传递参数	1 分别传递	ReactDOM.render(<Person name="老刘" sex="female" age="23" />,document.getElementById('test'))	
		2 批量传递	<pre>const p1 = { name:'强哥', sex:'女', age: 19 } ReactDOM.render(<Person {...p1}/>, document.getElementById('test2'))</pre>	
	prop-types	<!-- 引入prop-types, 用于对标签属性进行限制--><script type="text/javascript" src="../../js/prop-types.js"></script>		
	props 使用	1 函数式组件中	<pre>function Person(props) { // 用参数传入props const { name, age, sex } = props return (姓名: {name} 性别: {sex} 年龄: {age}) }</pre>	<pre>// 限制只能写在函数外面 Person.propTypes = { name: PropTypes.string.isRequired, // .isRequired 要求必须传 age: PropTypes.number, asex: PropTypes.string } Person.defaultProps = { // 默认值 age: 18 }</pre>
		2 类式组件中	<pre>class Person extends React.Component { render(){ const { name, sex, age} = this.props return (NAME: {name} GENDER: {sex} AGE: {age}) } }</pre>	<pre>// 限制写在类中 static propTypes = { name: PropTypes.string.isRequired, age: PropTypes.number, sex: PropTypes.string } static defaultProps = { age: 18 }</pre>
ref	Ref 三种使用方式	语法		保存位置
		1 字符串形式	<input type="text" ref="input1" />	this.refs.input1
		2 回调形式	<input type="text" ref={ c => this.input1 = c}/>	this.input1
		3 createRef	组件中提前定义： container = React.createRef() <input type="text" ref={this.container} />	this.container.current
React事件处理	<p>1. 通过onXxxx属性指定事件处理函数（注意大小写）</p> <p>* 1) React使用的是自定义（合成）事件，而不是使用的原生DOM事件</p> <p>* 2) React中的事件是通过事件委托方式处理的（委托给组件最外层的元素，底层有冒泡，可用e.stopPropagation()阻止）</p> <p>2. 通过event.target得到事件发生的DOM元素对象</p> <p>event.currentTarget是绑定事件的对象</p>			

React基础知识速查表

知识点	涉及语法			备注	
受控组件和非受控组件	非受控组件	表单中的数据，在需要的时候，“现用现取”。（通过ref获得input元素节点，进而访问到value值）			
	受控组件	使用onChange属性指定一个函数，随着用户的实时输入，值将自动收集到State中，那么就称为受控组件			
函数的柯里化	通过函数调用继续返回函数的方式，实现多次接收参数最后统一处理的函数编码形式。（可用于简化模式重复的函数）				
React生命周期钩子(旧)	初始化阶段	触发条件： ReactDOM.render()	1 constructor()	一般做一些初始化的事情：开启定时器、发送ajax请求、消息订阅	
			2 componentWillMount()		
			3 render()		
			4 componentDidMount()		
	更新阶段	触发条件： this.setState()或父组件重新render	0 componentWillMountReceiveProps()	子组件被更新时触发，初次挂载不会触发	
			1 shouldComponentUpdate()	返回值设置为true时，允许更新； 返回值设置为false时，阻塞更新，	
			2 componentWillMountUpdate()	使用this.forceUpdate()可强制进入该阶段	
			3 render()		
			4 componentDidUpdate()		
	卸载组件	触发条件： ReactDOM.unmountComponentAtNode()	componentWillUnmount()	一般做一些收尾的事情：关闭定时器、取消订阅消息等等	
	React生命周期钩子(新)	挂载时	同上	1 constructor()	当组件的state仅取决于外部传来的prop时，就要用该钩子。返回一个setState()的参数即可生效。
				* static getDerivedStateFromProps(props, state)	
				2 render()	
3 componentDidMount()					
更新时		同上	* static getDerivedStateFromProps(props, state)		
			1 shouldComponentUpdate()		
			2 render()	使用this.forceUpdate()可强制进入该阶段	
			3 getSnapshotBeforeUpdate()	可以返回一个值，将作为下一个阶段的第3个参数	
			4 componentDidUpdate(prevProps, prevState, snapshotValue)	常用占位符占掉前两个参数： componentDidUpdate(____, ____, snapshotValue)	
卸载时		同上	componentWillUnmount()		
			新增新闻保持阅读位置案例	在更新前获取scrollHeight，更新后获取最新的scrollHeight，相减得到列表高度的变化值，将该值添加给元素的scrollTop，即可始终保持阅读位置相对不动。	
react/vue中的key有什么作用？	虚拟DOM中key的作用： 1) 简单的说： key是虚拟DOM对象的标识，在更新显示时key起着极其重要的作用。 2) 详细的说：当状态中的数据发生变化时，react会根据【新数据】生成【新的虚拟DOM】，随后React进行【新虚拟DOM】与【旧虚拟DOM】的diff比较，比较规则如下： a) 旧虚拟DOM中找到了与新虚拟DOM相同的key： 1.若虚拟DOM中的内容没变，直接使用之前的真实DOM 2.若虚拟DOM中的内容变了，则生成新的真实DOM，随后替换掉页面中之前的真实DOM b) 旧虚拟DOM中未找到与新虚拟DOM相同的key 根据数据创建新的真实DOM，随后渲染到页面				
为什么遍历列表时，key最好不要用index？	用index作为key可能会引发的问题： 1) 若对数据进行 逆序添加、逆序删除等破坏顺序操作 ： 会产生没有必要的真实DOM更新 ==> 界面效果没问题，但效率低 2) 如果结构中 还包含输入类的DOM ： 会产生 错误DOM更新 ==> 界面有问题 3) 注意！ 如果不存在对数据的逆序添加、逆序删除等破坏顺序的操作， 仅用于渲染列表、用于简单的展示，使用index作为key是没有问题的 。				
开发中如何选择key？	1) 最好使用每条数据的唯一标识作为key，比如id、手机号、身份证号、学号等唯一值。 2) 如果确定只是简单的展示数据，用index也是可以的。				