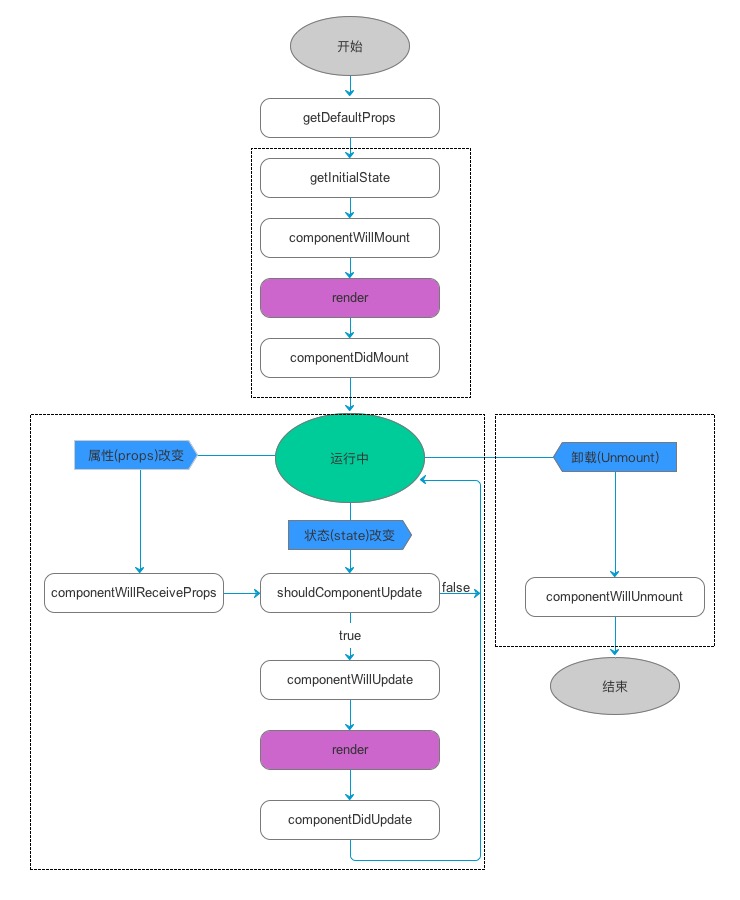
**1、React组件的生命周期**



**初始化：**

1、getDefaultProps()

设置默认的props，也可以用dufaultProps设置组件的默认属性.

2、getInitialState()

在使用es6的class语法时是没有这个钩子函数的，可以直接在constructor中定义this.state。此时可以访问this.props

3、componentWillMount()

组件初始化时只调用，以后组件更新不调用，整个生命周期只调用一次，此时可以修改state。

4、 render()

react最重要的步骤，创建虚拟dom，进行diff算法，更新dom树都在此进行。此时就不能更改state了。

5、componentDidMount()

组件渲染之后调用，只调用一次。

**运行中状态（更新）：**

6、componentWillReceiveProps(nextProps)

组件初始化时不调用，组件接受新的props时调用。

7、shouldComponentUpdate(nextProps, nextState)

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

8、componentWillUpdata(nextProps, nextState)

组件初始化时不调用，只有在组件将要更新时才调用，此时可以修改state

9、render()

组件渲染

10、componentDidUpdate()

组件初始化时不调用，组件更新完成后调用，此时可以获取dom节点。

**卸载：**

11、componentWillUnmount()

组件将要卸载时调用，一些事件监听和定时器需要在此时清除。

**2、shouldComponentUpdate 是做什么的？（react 性能优化是在哪个周期函数？）**

shouldComponentUpdate 这个方法用来判断是否需要调用 render 方法重新描绘 dom。因为 dom 的描绘非常消耗性能，如果我们能在 shouldComponentUpdate 方法中能够写出更优化的 dom diff 算法，可以极大的提高性能。

**3、为什么虚拟 dom 会提高性能?(必考)**

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档中，当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异把所记录的差异应用到所构建的真正的 DOM 树上，视图就更新了。

**4、react diff 原理（常考，大厂必考）**

（1）把树形结构按照层级分解，只比较同级元素。

（2）给列表结构的每个单元添加唯一的 key 属性，方便比较。

（3）React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）

（4）合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制.

（5）选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。

**5、React 中 refs 的作用是什么？**

Refs 是 React 提供给我们的安全访问 DOM 元素或者某个组件实例的句柄。我们可以为元素添加 ref 属性然后在回调函数中接受该元素在 DOM 树中的句柄，该值会作为回调函数的第一个参数返回：

class CustomForm extends Component {

handleSubmit = () => {

console.log("Input Value: ", this.input.value)

}

render () {

return (

<form onSubmit={this.handleSubmit}>

<input type='text'

ref={(input) => this.input = input} />

<button type='submit'>Submit</button>

</form>

)

}

}

上述代码中的 input 域包含了一个 ref 属性，该属性声明的回调函数会接收 input 对应的 DOM 元素，我们将其绑定到 this 指针以便在其他的类函数中使用。

**6、展示组件(Presentational component)和容器组件(Container component)之间有何不同？**

展示组件关心组件看起来是什么。展示专门通过 props 接受数据和回调，并且几乎不会有自身的状态，但当展示组件拥有自身的状态时，通常也只关心 UI 状态而不是数据的状态。

容器组件则更关心组件是如何运作的。容器组件会为展示组件或者其它容器组件提供数据和行为(behavior)，它们会调用 Flux actions，并将其作为回调提供给展示组件。容器组件经常是有状态的，因为它们是(其它组件的)数据源。

**7、类组件(Class component)和函数式组件(Functional component)之间有何不同？**

类组件不仅允许你使用更多额外的功能，如组件自身的状态和生命周期钩子，也能使组件直接访问 store 并维持状态

当组件仅是接收 props，并将组件自身渲染到页面时，该组件就是一个 '无状态组件(stateless component)'，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件。

**8、(组件的)状态(state)和属性(props)之间有何不同？**

State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作为用户事件行为的结果。

Props(properties 的简写)则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的(immutable)。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)。Props 也不仅仅是数据--回调函数也可以通过 props 传递。

**9、什么是受控组件(controlled component)？**

在HTML中，像<input>、<textarea>和<select>这类表单元素会维持自身状态，并根据用户输入进行更新。但在React中，可变状态通常保存在组件的状态属性中，并且只能使用 setState() 方法进行更新，而呈现表单的React组件也控制着在后续用户输入时该表单中发生的变化。相应的，其值由React控制的输入表单元素称为“受控组件”。

**10、何为高阶组件(higher order component)？**

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC 运行你重用代码、逻辑和引导抽象。最常见的可能是 Redux 的 connect 函数。除了简单分享工具库和简单的组合，HOC 最好的方式是共享 React 组件之间的行为。如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的 HOC。

**11、为什么建议传递给 setState 的参数是一个 callback 而不是一个对象？**

因为 this.props 和 this.state 的更新可能是异步的，不能依赖它们的值去计算下一个 state。

**12、应该在 React 组件的何处发起 Ajax 请求？**

在 React 组件中，应该在 componentDidMount 中发起网络请求。这个方法会在组件第一次“挂载”(被添加到 DOM)时执行，在组件的生命周期中仅会执行一次。更重要的是，你不能保证在组件挂载之前 Ajax 请求已经完成，如果是这样，也就意味着你将尝试在一个未挂载的组件上调用 setState，这将不起作用。在 componentDidMount 中发起网络请求将保证这有一个组件可以更新了。

**13、createElement 和 cloneElement 有什么区别？**

React.createElement():JSX 语法就是用 React.createElement()来构建 React 元素的。它接受三个参数，第一个参数可以是一个标签名。如 div、span，或者 React 组件。第二个参数为传入的属性。第三个以及之后的参数，皆作为组件的子组件。

React.createElement(

type,

[props],

[...children]

)

React.cloneElement()与 React.createElement()相似，不同的是它传入的第一个参数是一个 React 元素，而不是标签名或组件。新添加的属性会并入原有的属性，传入到返回的新元素中，而旧的子元素将被替换。

React.cloneElement(

element,

[props],

[...children]

)

**14、React 中有三种构建组件的方式？**

React.createClass()、ES6 class 和无状态函数。

**15、React 项目用过什么脚手架（本题是开放性题目）**

create-react-app、Yeoman等.

**16、简述 flux 思想**

Flux 的最大特点，就是数据的"单向流动"。

（1）用户访问 View

（2）View 发出用户的 Action

（3）Dispatcher 收到 Action，要求 Store 进行相应的更新

（4）Store 更新后，发出一个"change"事件

（5）View 收到"change"事件后，更新页面

**17、了解 redux吗？说一下 redux吧**

（1）redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，主要有三个核心方法，action，store，reducer，工作流程是 view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据，flux 也是用来进行数据操作的，有四个组成部分 action，dispatch，view，store，工作流程是 view 发出一个 action，派发器接收 action，让 store 进行数据更新，更新完成以后 store 发出 change，view 接受 change 更新视图。Redux 和 Flux 很像。主要区别在于 Flux 有多个可以改变应用状态的 store，在 Flux 中 dispatcher 被用来传递数据到注册的回调事件，但是在 redux 中只能定义一个可更新状态的 store，redux 把 store 和 Dispatcher 合并,结构更加简单清晰

（2）新增 state,对状态的管理更加明确，通过 redux，流程更加规范了，减少手动编码量，提高了编码效率，同时缺点时当数据更新时有时候组件不需要，但是也要重新绘制，有些影响效率。一般情况下，我们在构建多交互、多数据流的复杂项目应用时才会使用它们。

**18、redux 有什么缺点？**

（1）一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。

（2）当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新render，可能会影响效率，或者需要写复杂的 shouldComponentUpdate 进行判断。

**19、使用箭头函数(arrow functions)的优点是什么？**

（1）作用域安全：在箭头函数之前，每一个新创建的函数都有定义自身的 this 值(在构造函数中是新对象；在严格模式下，函数调用中的 this 是未定义的；如果函数被称为“对象方法”，则为基础对象等)，但箭头函数不会，它会使用封闭执行上下文的 this 值。

（2）简单：箭头函数易于阅读和书写

（3）清晰：当一切都是一个箭头函数，任何常规函数都可以立即用于定义作用域。开发者总是可以查找 next-higher 函数语句，以查看 this 的值。

**20、怎么阻止组件的渲染？**

在组件的 render 方法中返回 null 并不会影响触发组件的生命周期方法。

**21、当渲染一个列表时，设置 key 的目的是什么？**

Keys 会有助于 React 识别哪些 items 改变了，被添加了或者被移除了。Keys 应该被赋予数组内的元素以赋予(DOM)元素一个稳定的标识，选择一个 key 的最佳方法是使用一个字符串，该字符串能惟一地标识一个列表项。很多时候你会使用数据中的 IDs 作为 keys，当你没有稳定的 IDs 用于被渲染的 items 时，可以使用项目索引作为渲染项的 key，但这种方式并不推荐，如果 items 可以重新排序，就会导致 re-render 变慢。

**22、怎么用 React.createElement 重写下面的代码？**

Question：

const element = (

<h1 className="greeting">

Hello, rdhub.cn!

</h1>

);

Answer：

const element = React.createElement(

'h1',

{className: 'greeting'},

'Hello, rdhub.cn!'

);

**23、何为 JSX？**

JSX 是 JavaScript 语法的一种语法扩展，并拥有 JavaScript 的全部功能。JSX 生产 React "元素"，你可以将任何的 JavaScript 表达式封装在花括号里，然后将其嵌入到 JSX 中。在编译完成之后，JSX 表达式就变成了常规的 JavaScript 对象，这意味着你可以在 if 语句和 for 循环内部使用 JSX，将它赋值给变量，接受它作为参数，并从函数中返回它。

**24、何为 redux？**

Redux 的基本思想是整个应用的 state 保持在一个单一的 store 中。store 就是一个简单的 javascript 对象，而改变应用 state 的唯一方式是在应用中触发 actions，然后为这些 actions 编写 reducers 来修改 state。整个 state 转化是在 reducers 中完成，并且不应该有任何副作用。

**25、在 Redux 中，何为 store？**

Store 是一个 javascript 对象，它保存了整个应用的 state。与此同时，Store 也承担以下职责：

（1）允许通过 getState() 访问 state

（2）运行通过 dispatch(action) 改变 state

（3）通过 subscribe(listener) 注册 listeners

（4）通过 subscribe(listener) 返回的函数处理 listeners 的注销

**26、何为 action？**

Actions 是一个纯 javascript 对象，它们必须有一个 type 属性表明正在执行的 action 的类型。实质上，action 是将数据从应用程序发送到 store 的有效载荷。

**27、何为 reducer？**

一个 reducer 是一个纯函数，该函数以先前的 state 和一个 action 作为参数，并返回下一个 state。

**28、Redux Thunk 的作用是什么？**

Redux thunk 是一个允许你编写返回一个函数而不是一个 action 的 actions creators 的中间件。如果满足某个条件，thunk 则可以用来延迟 action 的派发(dispatch)，这可以处理异步 action 的派发(dispatch)。

**29、何为纯函数(pure function)？**

一个纯函数是一个不依赖于且不改变其作用域之外的变量状态的函数，这也意味着一个纯函数对于同样的参数总是返回同样的结果。

**30、(在构造函数中)调用 super(props) 的目的是什么？**

在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。