(5) 移动端开发1

1 介绍一下 React？？

React 是一个用于构建用户界面的 JAVASCRIPT 库。React主要用于构建UI，很多人认为 React 是 MVC 中的 V（视图）

React特点有：

1.声明式设计 −React采用声明范式，可以轻松描述应用。

2.高效 −React通过对DOM的模拟，最大限度地减少与DOM的交互。

3.灵活 −React可以与已知的库或框架很好地配合。

4.JSX − JSX 是 JavaScript 语法的扩展。React 开发不一定使用 JSX ，但我们建议使用它。

5.组件 − 通过 React 构建组件，使得代码更加容易得到复用，能够很好的应用在大项目的开发中。

6.单向响应的数据流 − React 实现了单向响应的数据流，从而减少了重复代码，这也是它为什么比传统数据绑定更简单。

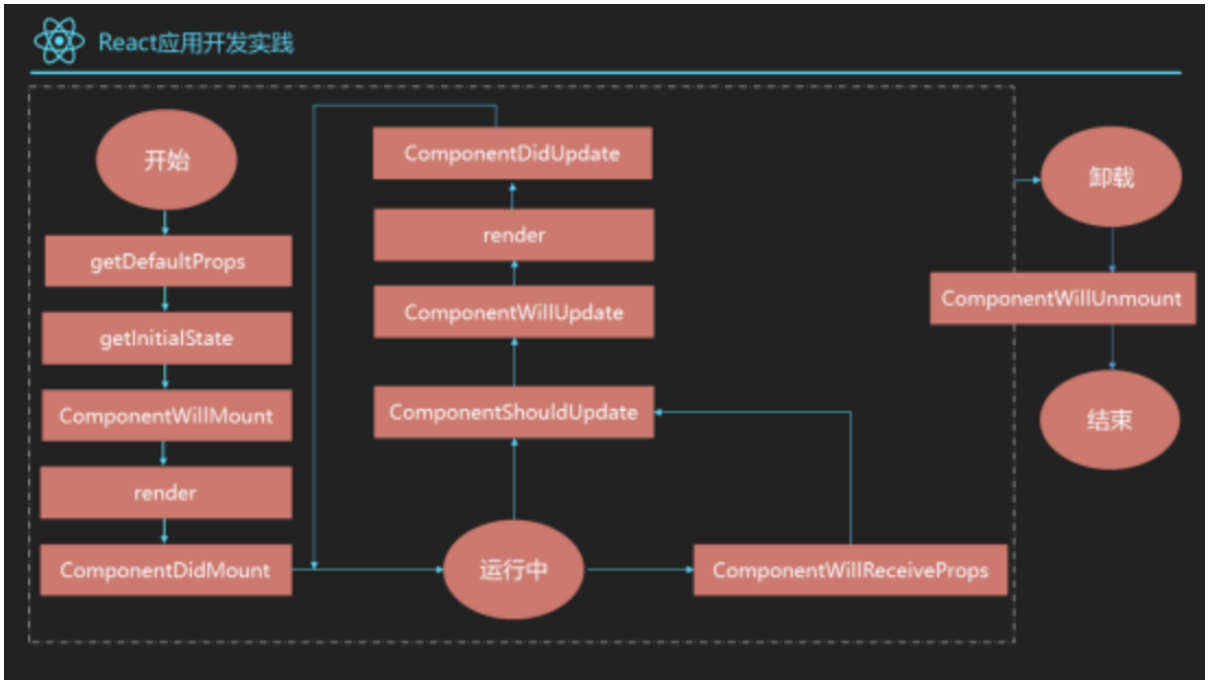
2 React的单向数据流【数据是单向流动的，是从上到下的方向，即从父组件到子组件的方向！！】

在React中，数据是单向流动的，是从上向下的方向，即从父组件到子组件的方向。
state和props是其中重要的概念，如果顶层组件初始化props，那么React会向下遍历整颗组件树，重新渲染相关的子组件。其中state表示的是每个组件中内部的的状态，这些状态只在组件内部改变。
把组件看成是一个函数，那么他接受props作为参数，内部由state作为函数的内部参数，返回一个虚拟dom的实现。

注意：
把组件视作一个函数，那么他接受 props 作为参数，内部由 state 作为函数的内部参数，返回一个一个虚拟 DOM的实现。

3 react 相关的生命周期？？【需要系统的学习、巩固一下！！】

React的组件在第一次挂在的时候首先获取父组件传递的props，接着获取初始的state值，接着经历挂载阶段的三个生命周期函数，也就是ComponentWillMount，render，ComponentDidMount，这三个函数分别代表组件将会挂载，组件渲染，组件挂载完毕三个阶段，在组件挂载完成后，组件的props和state的任意改变都会导致组建进入更新状态，在组件更新阶段，如果是props改变，则进入ComponentWillReceiveProps函数，接着进入ComponentShouldUpdate进行判断是否需要更新，如果是state改变则直接进入ComponentShouldUpdate判定，这个默认是true，当判定不需要更新的话，组件继续运行，需要更新的话则依次进入ComponentWillMount，render，ComponentDidMount三个函数，当组件卸载时，会首先进入生命周期函数ComponentWillUnmount,之后才进行卸载，如图



React的生命周期函数：

初始化阶段：getDefaultProps获取实例的默认属性，getInitialState获取每个实例的初始化状态，ComponentWillMount：组件将被装载，渲染到页面上，render：组件在这里生成虚拟的DOM节点，ComponentDidMount:组件真正被装载之后

运行中状态： componentWillReceiveProps:组件将要接收到属性的时候调用 shouldComponentUpdate:组件接受到新属性或者新状态的时候（可以返回 false，接收数据后不更新，阻止 render 调用，后面的函数不会被继续执行了） 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。 componentWillUpdate:组件即将更新不能修改属性和状态 render:组件重新描绘 componentDidUpdate:组件已经更新 销毁阶段：componentWillUnmount:组件即将销毁

4 React组件的交流主要分为 3种！！

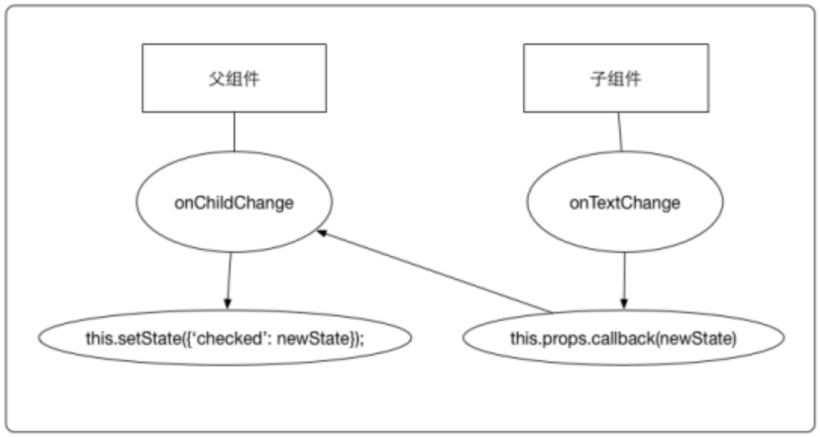
父传子；

子传父；

无相关联的2组件。

1、父组件向子组件传值:主要是利用props来进行交流

2、子组件向父组件传值：子组件通过控制自己的state然后告诉父组件的点击状态。然后在父组件中展示出来，如图：



3、没有任何嵌套关系的组件之间传值：如果组件之间没有任何关系，组件嵌套层次比较深（个人认为 2 层以上已经算深了），或者你为了一些组件能够订阅、写入一些信号，不想让组件之间插入一个组件，让两个组件处于独立的关系。对于事件系统，这里有 2 个基本操作步骤：订阅（subscribe）/监听（listen）一个事件通知，并发送（send）/触发（trigger）/发布（publish）/发送（dispatch）一个事件通知那些想要的组件。

5 有了解React的虚拟DOM吗？ 虚拟DOM怎么进行对比、从而更新？？ 【diff算法如何从n^3 ---> n^1？？？】

当然是使用的diff算法，diff算法有三种优化形式：

tree diff：将新旧两颗DOM树按照层级遍历，只对同级的DOM节点进行比较，即同一父节点下的所有子节点，当发现节点已经不存在，则该节点及其子节点会被完全删除，不会进一步比较

component diff：不同组件之间的对比，如果组件类型相同，暂不更新，否则删除旧的组件，再创建一个新的组件，插入到删除组件的位置

element diff:在类型相同的组件内，再继续对比组件内部的元素，

参考：https://juejin.im/post/5a3200fe51882554bd5111a0

6 React的优点、优势？？ 【那缺点呢？？】

（1）声明式设计

（2）高效：通过对DOM的模拟，最大限度的减少与DOM的交互。

（3）灵活：可以与已知的框架或库很好的配合。

（4）JSX：是js语法的扩展，不一定使用，但建议用。

（5）组件：构建组件，使代码更容易得到复用，能够很好地应用在大项目的开发中。

（6）单向响应的数据流：React实现了单向响应的数据流，从而减少了重复代码，这也是解释了它为什么比传统数据绑定更简单。

7 怎么样获取真正的DOM？？

ReactDOM.findDOMNode() 或者 “引用” this.refs

8 React的生命周期函数【比较符合自己所学的！！】

8.1 初始化

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。

8.2 更新时【“亦称为 运行时？？”】

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。

8.3 卸载时

卸载

11、componentWillUnmount()

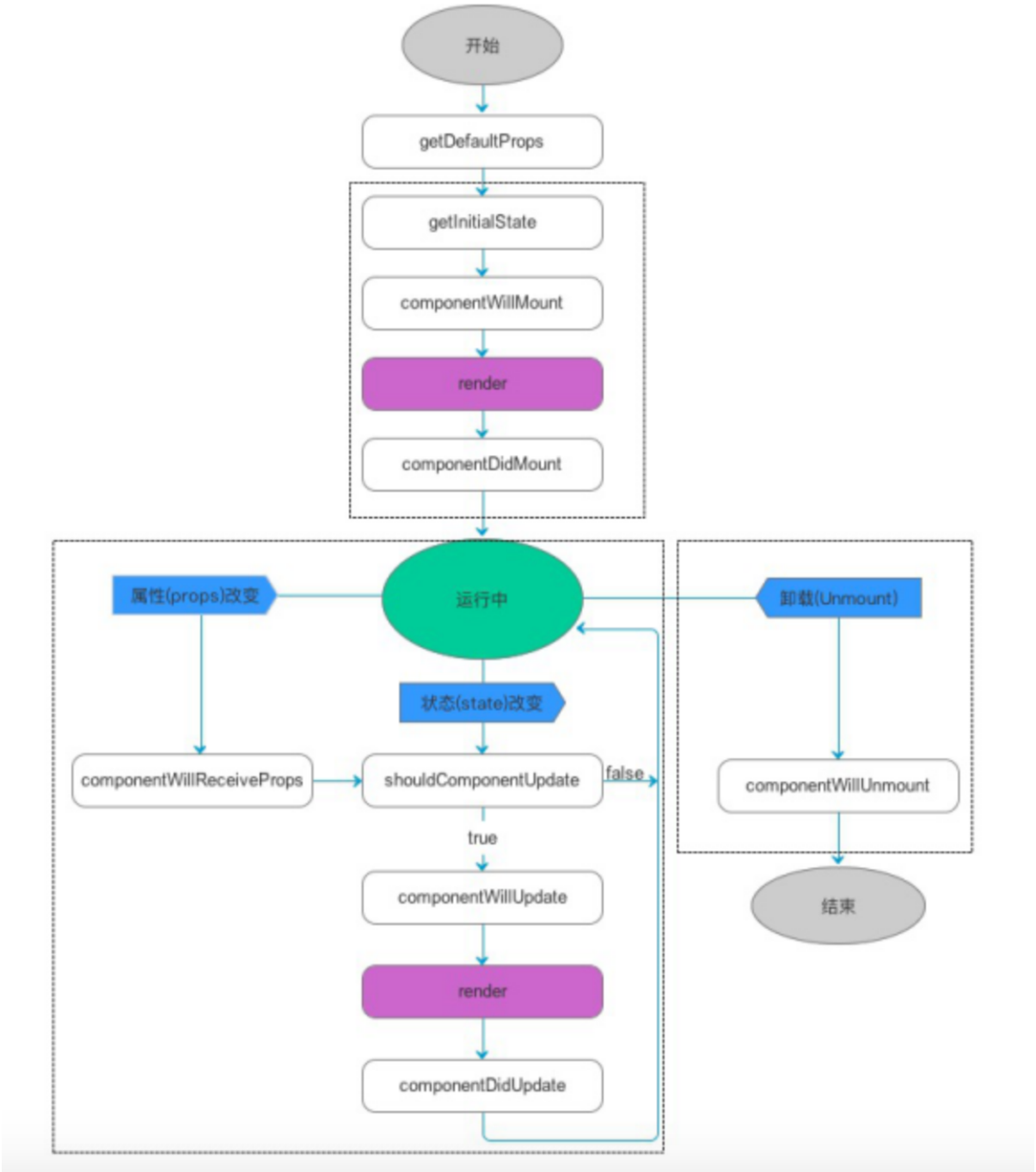
组件将要卸载时调用，一些事件监听和定时器需要在此时清除。

9 setState之后的流程？？【根据新、老树的差异 对页面进行最小化重渲染。
保证 按需更新、而不是全部重新渲染！！】

在代码中调用setState函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。 经过调和过程，React 会以相对高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个UI界面。 在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染。 在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

10 react高阶组件的定义、作用、使用场景？？
高阶组件【本质是一个函数、不是组件！！ 注意其与高阶函数的关系？?!】接受一个 React 组件作为参数，并返回 一个新的React组件。

11 React的生命周期图【3状态：初始化、更新、销毁】：



12 react组件是否刷新的依据【其 state 是否发生改变！！】：
state是否发生改变！！

13

- **React Native**

- **其他移动APP开发框架（PhoneGap，AppCan，HTML5+，Framework7）**

完