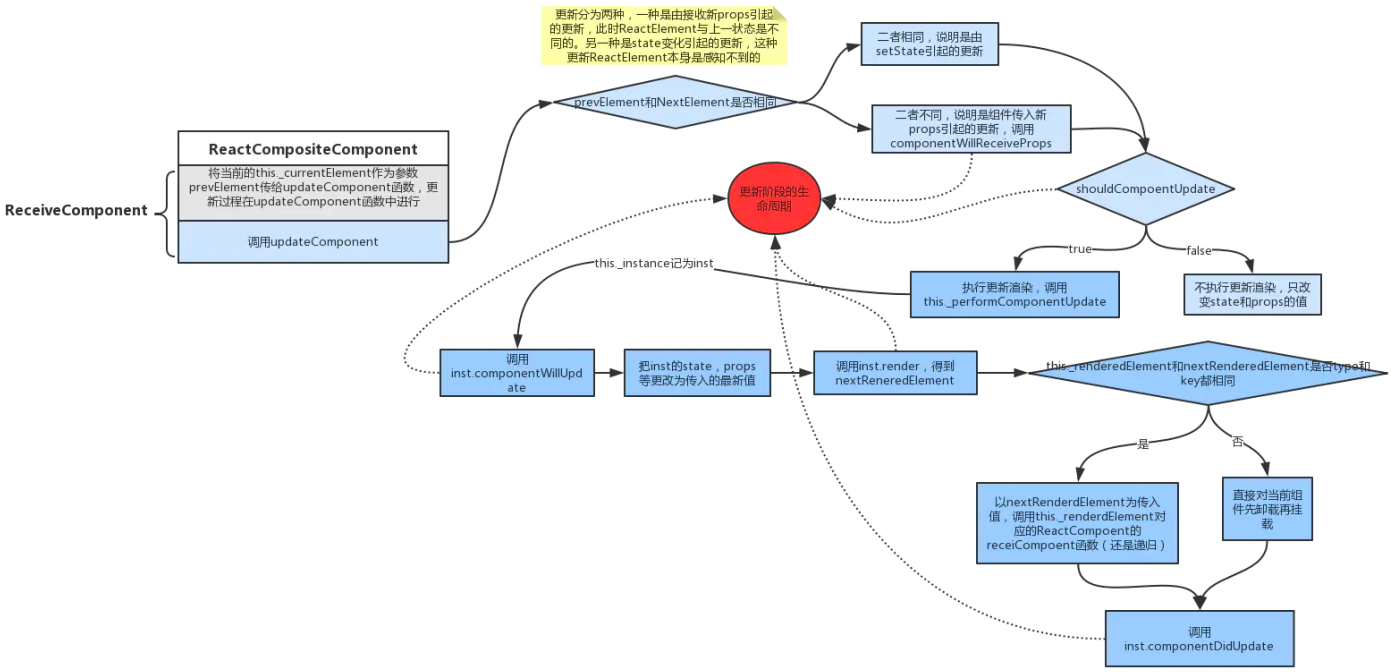
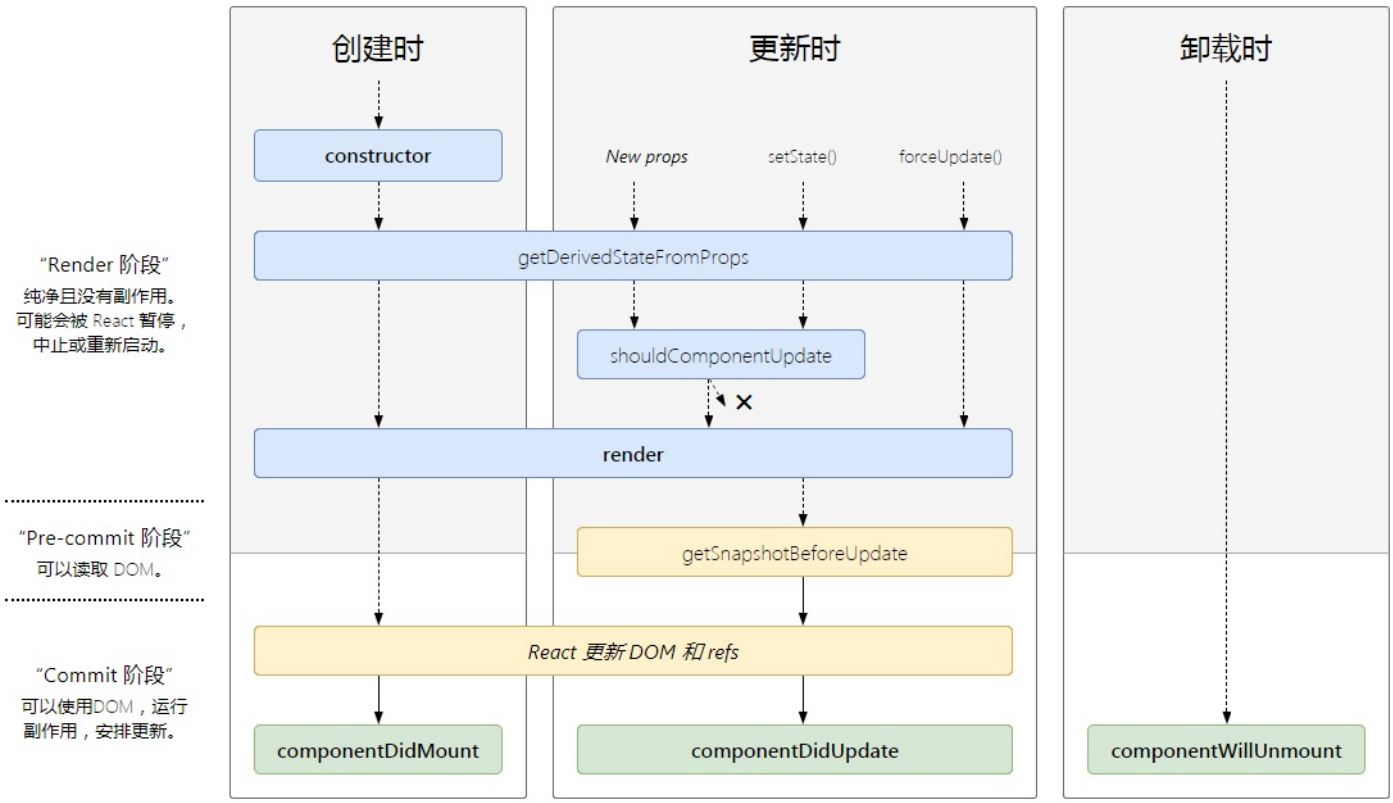
React15生命周期



React^16.4生命周期



See project on GitHub

生命周期方法讲解

- **shouldComponentUpdate: (nextProps, nextState) => boolean**
  - 当 props 或 state 发生变化时，shouldComponentUpdate() 会在渲染执行之前被调用,根据 shouldComponentUpdate() 的返回值，判断 React 组件的输出是否受当前 state 或 props 更改的影响。当返回值为true(默认)时, state和props每次发生变化组件都会重新渲染。大部分情况下，你应该遵循默认行为。返回值默认为 true。
  - 首次渲染或使用 forceUpdate() 时不会调用该方法。
  - 尽量不使用该方法，应该考虑使用内置的 PureComponent 组件或React.memo，而不是手动编写 shouldComponentUpdate()。
  - 如果你一定要手动编写此函数，可以将 this.props 与 nextProps 以及 this.state 与nextState 进行比较，并返回 false 以告知 React 可以跳过更新。返回 false 并不会阻止子组件在 state 更改时重新渲染。我们不建议在 shouldComponentUpdate() 中进行深层比较或使用 JSON.stringify()。这样非常影响效率，且会损害性能。
- **static getDerivedStateFromProps: (nextProps, prevState) => nextState**
  - 这是一个静态方法，无法在该生命周期方法内部使用this指针
  - 该方法根据最新的Props和组件的上一次state，返回一个新的state对象用于更新组件状态
  - 该函数应该是个纯函数，它的唯一作用是根据props生成派生state，不要在该方法内部进行任何副作用操作
- **getSnapshotBeforeUpdate: (prevProps, prevState) => any**
  - 该方法应该是一个纯函数，内部不能有任何副作用
  - 该方法的返回值会作为componentDidUpdate方法的最后一个参数传入
  - 该方法在render方法被调用后，React更新DOM之前执行，可以在该方法内部获取更新前的DOM信息，传递给componentDidUpdate方法。例子参照官网实例
- **componentDidUpdate(): (prevProps, prevState, snapshot) => void**
  - 该方法在DOM被更新之后调用, 该方法的第三个参数是getSnapshotBeforeUpdate的返回值
  - 该方法的前两个参数是上一次的props和state，可以在该方法中调用 setState()，但务必将这个逻辑包裹在一个条件语句里，否则会导致组件无限次重复渲染