06 生命周期：如何理解 React 组件生命周期？

更新时间：2020-10-19 17:46:49

> 不经一翻彻骨寒，怎得梅花扑鼻香。——宋帆

如何理解 React 组件的生命周期

前言

很多面试官喜欢考察求职者对 React 组件生命周期的理解，如果求职者只是简单的把组件生命周期中的几个钩子函数说了出来，那么这样的回答并不能激起面试官的兴趣。那么，该如何回答有关 React 组件生命周期的理解呢？先来看一下代码示例 2.3.1。

```
class UpdateCounter extends React.Component {
 constructor(props) {
  super(props);
  this.state = {count: 0};
  this.handleClick = this.handleClick.bind(this);
 }

 componentWillMount() {
   // 组件首次渲染时会调用
  console.log('进入componentWillMount函数');
 }
 componentDidMount() {
   // 组件首次渲染时会调用
  console.log('进入componentDidMount函数');
 }

 componentWillReceiveProps(nextProps) {
   // 组件props更新时会调用，不稳定，父组件更新也可能会引起子组件执行该函数，需要判断nextProps === this.props
   console.log('进入componentWillReceiveProps函数');
 }

 shouldComponentUpdate(nextProps, nextState) {
   // 组件更新时会调用，react性能优化非常重要的一环，此处可阻止不必要的更新
   console.log('进入shouldComponentUpdate函数');
   return true;
 }

 componentWillUpdate(nextProps, nextState) {
   // 组件更新时会调用，此时不可以修改state
   console.log('进入componentWillUpdate函数');
 }

 componentDidUpdate(nextProps, nextState) {
   // 组件更新后会调用
   console.log('进入componentDidUpdate函数');
 }

 handleClick() {
  this.setState({
   count: this.state.count + 1
  });
 }

 render() {
  return (
   <div className="wrap-box">
      <div className="title-text">点击计数</div>
    <button key="1" onClick={this.handleClick}>Update counter</button>
    <span key="2" className="span-text">{this.state.count}</span>
   </div>
  )
 }
}

ReactDOM.render(<UpdateCounter />, document.getElementById('root'));
```

代码示例 2.3.1

代码示例 2.3.1 中定义了一个`UpdateCounter`组件，该组件的作用是统计按钮的点击次数。前面提到了 React 应用程序运行过程中组件会有两种渲染流程—**首次渲染（Mounting）** 和 **更新渲染（Updation）**。两种渲染流程会调用不同的生命周期函数，见图 2.3.1。

组件渲染流程与生命周期函数

React v16.3 之前的组件渲染流程与生命周期函数

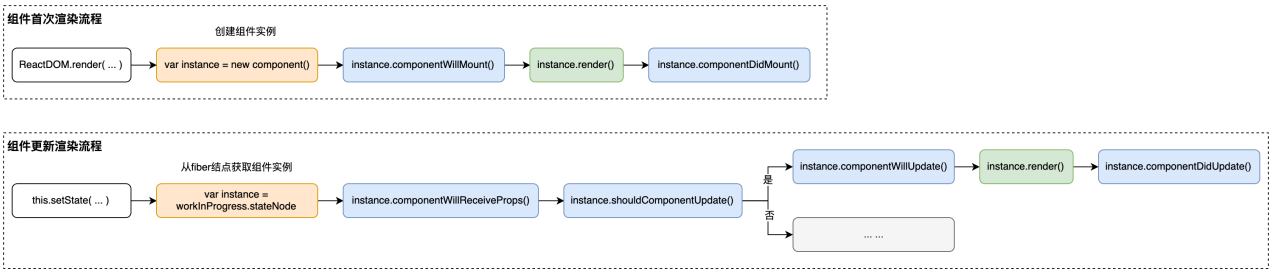

图 2.3.1 React v16.3 之前不同渲染流程组件调用生命周期函数的时机与方式

图 2.3.1分别以组件的首次渲染和更新渲染流程为主线描绘其生命周期函数的调用时机与方式，调用组件的生命周期函数前必须取得组件实例。首次渲染时以`instance = new component(...)`的方式创建并获得组件实例。更新渲染时以`instance = workInProgress.stateNode`的方式获得组件实例。

> 注：上面的 component 指的是对应组件的构造函数， workInProgress 指的是对应组件的 Fiber 结点。

React v15 版中组件的生命周期函数整体调用逻辑见图 2.3.2。

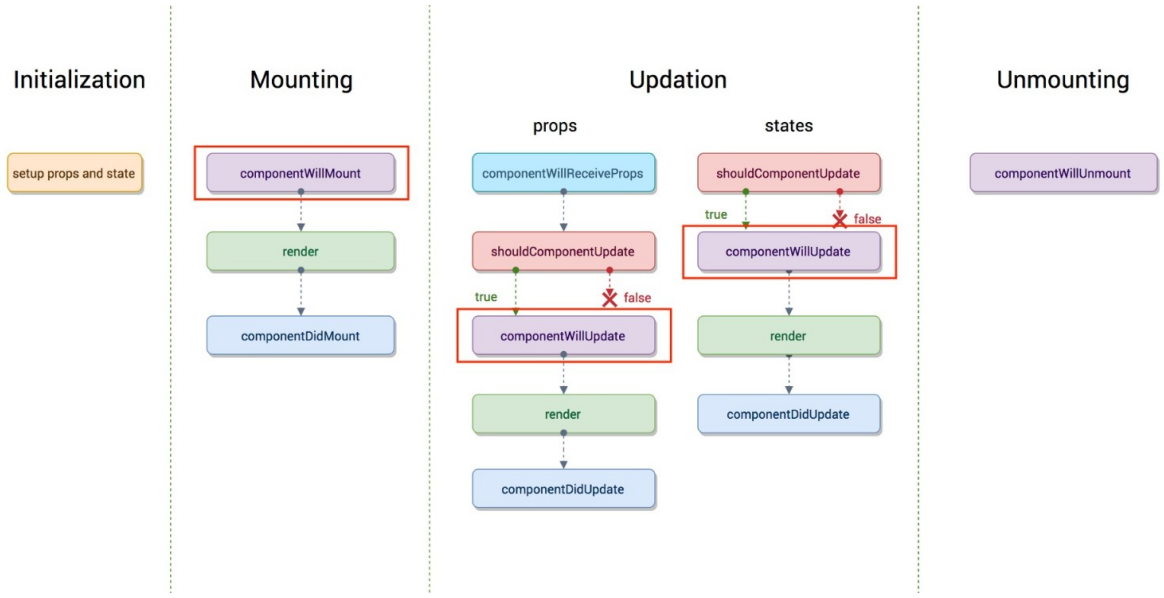

图 2.3.2 React v15 版生命周期函数调用逻辑

现在我们对生命周期的理解进行简单总结如下：

React 组件生命周期是指使用 class 定义的组件在整个应用程序运行过程中会经历**首次渲染（挂载）**，**更新渲染**和 **卸载**三个过程。在不同的过程可以调用不同的生命周期函数。组件在首次渲染时会被实例化，然后调用实例上面的 componentWillMount， render 和 componentDidMount 函数。组件在更新渲染时可以调用 componentWillReceiveProps， shouldComponentUpdate， componentWillUpdate， render 和 componentDidUpdate 函数。组件在卸载时可以调用 componentWillUnmount 函数。

React v16.3 之后的组件渲染流程与生命周期函数

从 React v16.3 版本开始，React 引入了两个新的生命周期函数 `getDerivedStateFromProps` 和 `getSnapshotBeforeUpdate`。同时 React 不再建议使用 `componentWillMount`，`componentWillReceiveProps` 和 `componentWillUpdate` 三个生命周期函数，如果使用的话会在控制台收到 Warning 信息，见图 2.3.3。

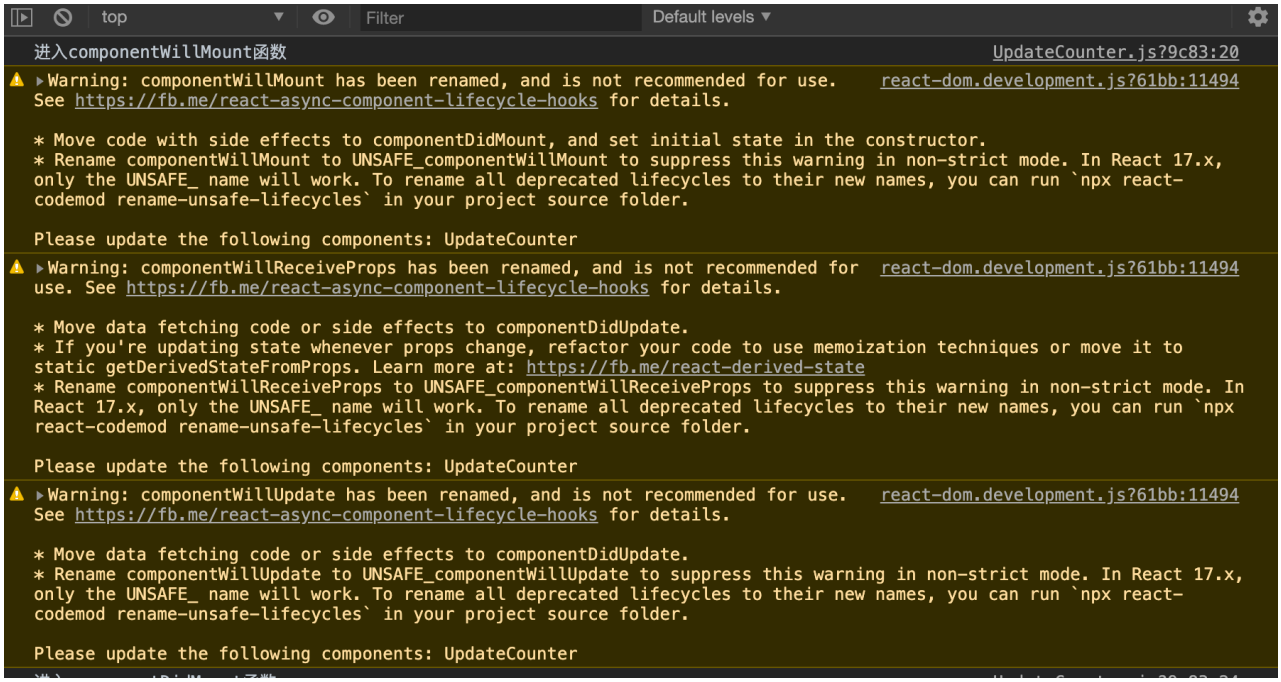

图 2.3.3 React v16.3 版以后对三个生命周期函数给出 Warning

React v16.3 版本中组件 **首次渲染（Mounting）** 和 ** 更新渲染（Updation）** 调用生命周期函数的时机与方式，见图 2.3.4。

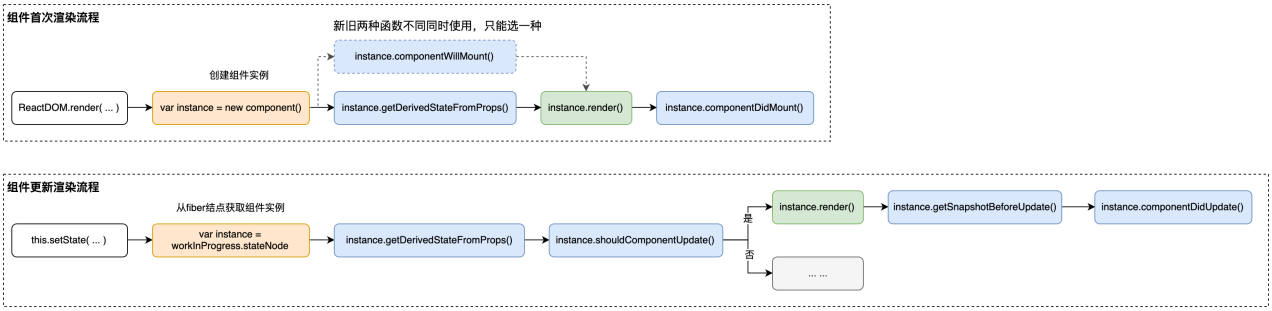

图 2.3.4 React v16.3 版不同渲染流程组件调用调用生命周期函数的时机与方式

从 React v16.3 版本开始，React 建议使用`getDerivedStateFromProps`和`getSnapshotBeforeUpdate`两个生命周期函数替代`componentWillMount`，`componentWillReceiveProps`和`componentWillUpdate`三个生命周期函数。这里需要注意的是 **新增的两个生命周期函数和原有的三个生命周期函数必须分开使用，不能混合使用**。React v16.3 版本的组件生命周期函数整体调用逻辑见图 2.3.5。

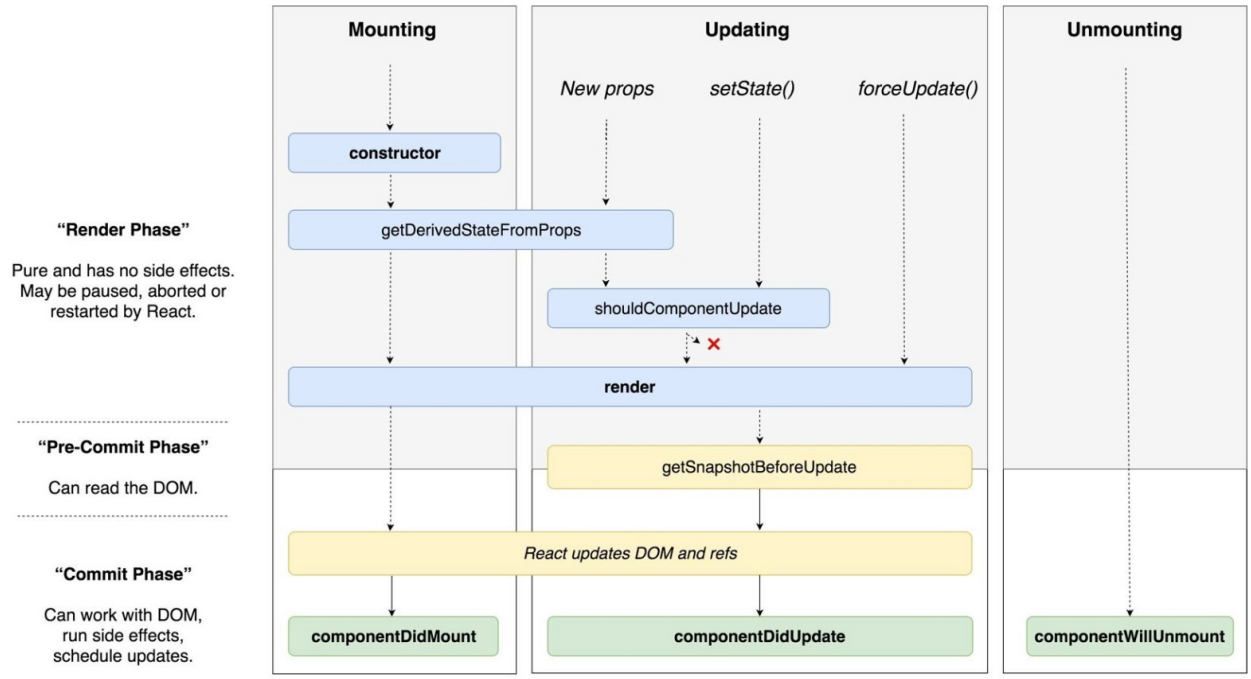

图 2.3.5 React v16.3 版组件生命周期函数整体调用逻辑

现在我们对文章开头那个问题—对生命周期的理解进一步总结。

React 组件生命周期是指使用 class 定义的组件在整个应用程序运行过程中会经历**首次渲染（挂载）**，**更新渲染**和 **卸载**三个过程。在不同的过程可以调用不同的生命周期函数。组件在首次渲染时会被实例化，然后调用实例上面的 componentWillMount， render 和 componentDidMount 函数。组件在更新渲染时可以调用 componentWillReceiveProps， shouldComponentUpdate， componentWillUpdate， render 和 componentDidUpdate 函数。组件在卸载时可以调用 componentWillUnmount 函数。React v16.3版本中将 componentWillMount， componentWillReceiveProps 和 componentWillUpdate 标记为不安全的生命周期函数并不推荐使用，同时新增了 getDerivedStateFromProps 和 getSnapshotBeforeUpdate 函数用于代替它们。

为什么 **React v16** 版的生命周期函数发生了变更呢？ 下一节进行详细介绍。

}

05 组件：从设计者的角度理解 React 组件

07 生命周期：为什么 React v16 产生了新的生命周期函数