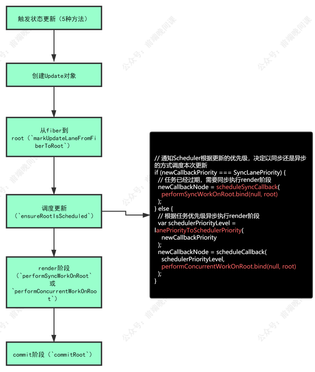
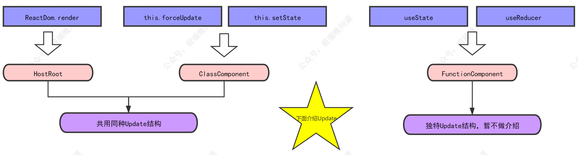
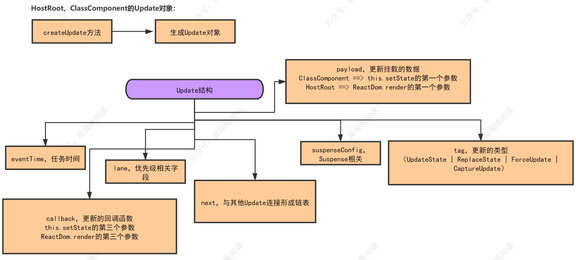
触发状态更新关键节点：



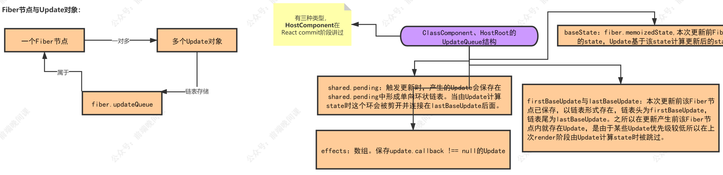
触发状态更新的方法（5种方法、3种组件类型）：



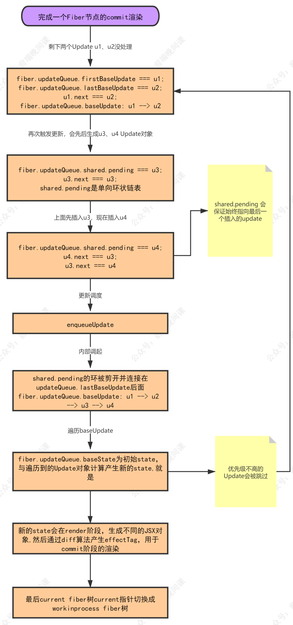
HostRoot、ClassComponent的Update对象：



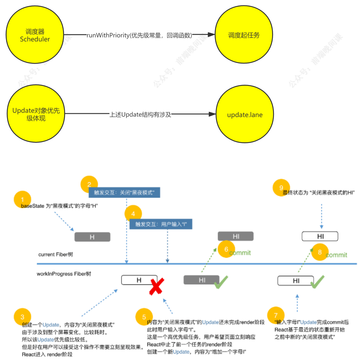
Fiber节点与Update对象：

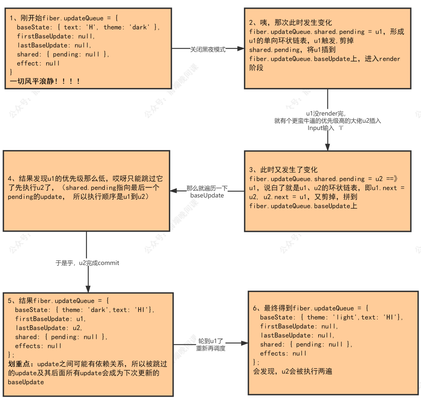


updateQueue的工作流程：

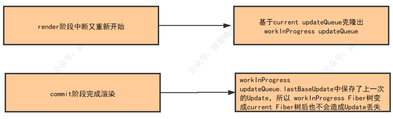


调度优先级：

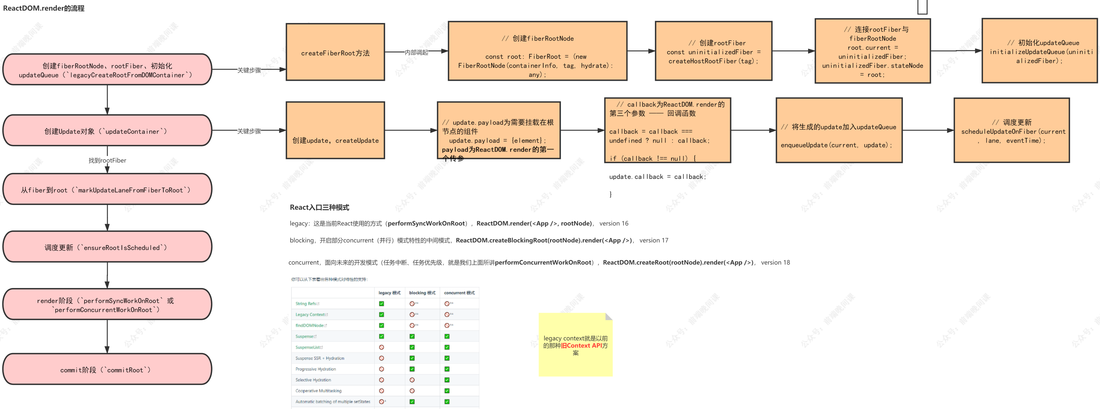




保证Update不丢失（两种情况）



ReactDOM.render的流程



React入口三种模式

legacy：这是当前React使用的方式（**performSyncWorkOnRoot**），**ReactDOM.render(<App />, rootNode)**，version 16

blocking，开启部分concurrent（开行）模式特性的中间模式，**ReactDOM.createBlockingRoot(rootNode).render(<App />)**，version 17

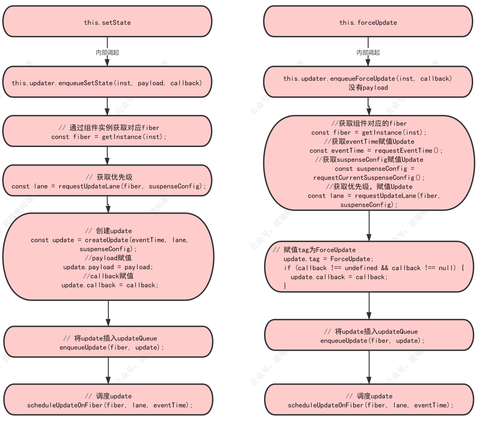
concurrent，面向未来的开发模式（任务中断、任务优先级，就是我们上面所讲**performConcurrentWorkOnRoot**），**ReactDOM.createRoot(rootNode).render(<App />)**，version 18

你可以以下表格中的特性来比较各自的优势：

	legacy 模式	blocking 模式	concurrent 模式
String Refs.[1]	✅	⭕ [2]	⭕ [2]
Legacy Context[1]	✅	⭕ [2]	⭕ [2]
findDOMNode[1]	✅	⭕ [2]	⭕ [2]
Suspense[1]	✅	✅	✅
SuspenseList[1]	⭕	✅	✅
Suspense SSR + Hydration	⭕	✅	✅
Progressive Hydration	⭕	✅	✅
Selective Hydration	⭕	⭕	✅
Cooperative Multitasking	⭕	⭕	✅
Automatic batching of multiple setStates	⭕ [3]	✅	✅
Priority-based Rendering[1]	⭕	⭕	✅
Interruptible Prerendering[1]	⭕	⭕	✅
useTransition[1]	⭕	⭕	✅
usePrevious[1]	⭕	⭕	✅
Suspense Reveal "Train"[1]	⭕	⭕	✅

legacy context就是以前的那种旧Context API方案

this.setState、this.forceUpdate的流程



为啥会赋值update.tag = ForceUpdate？

因为ClassComponent判断是否更新决定于shouldComponentUpdate，是否为this.forceUpdate触发更新，而update.tag = ForceUpdate作用就是会直接强制触发更新

```
const shouldUpdate =
  checkHasForceUpdateAfterProcessing() ||
  checkShouldComponentUpdate(
    workInProgress,
    ctor,
    oldProps,
    newProps,
    oldState,
    newState,
    nextContext,
  );
```

checkShouldComponentUpdate针对state、props浅比较

checkHasForceUpdateAfterProcessing，内部会判断本次更新的Update是否为ForceUpdate