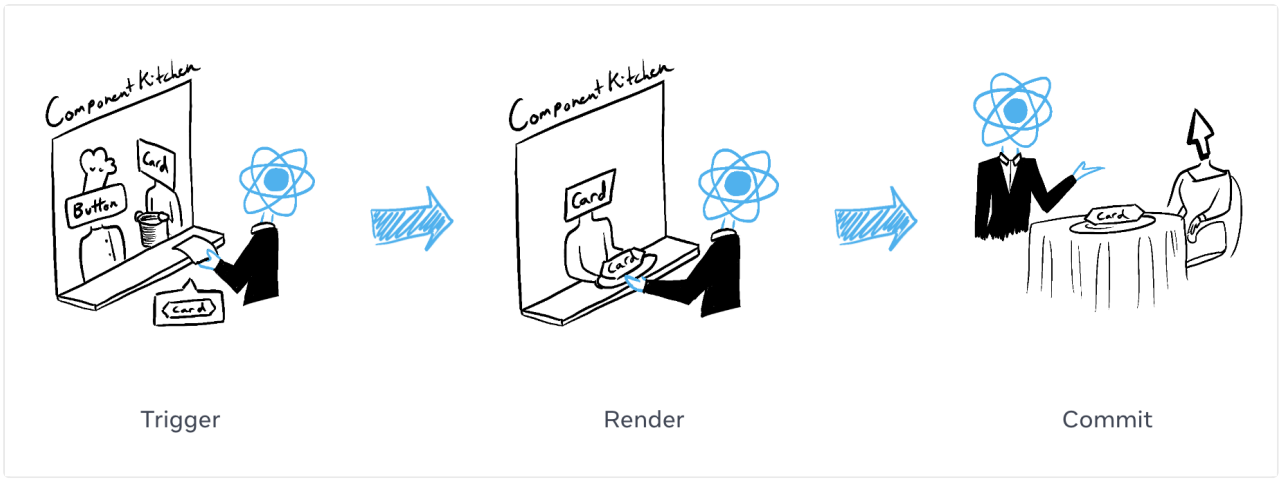
8-9 commit 阶段

如果以下代码没有特殊标记路径的源码，那么路径都是 src/react/packages/react-reconciler/src/ReactFiberWorkLoop.js。

React 中的阶段

1. **Triggering** a render (把客人的点单分发到厨房)
2. **Rendering** the component (在厨房准备订单) beginWork and completeWork
3. **Committing** to the DOM (将菜品放在桌子上)



render 阶段结束

`performConcurrentWorkOnRoot`

DebugReact > src > react > packages > react-reconciler > src > JS ReactFiberWorkLoop.js > performConcurrentWorkOnRoot

```
989   if (exitStatus !== RootInProgress) {
990     let renderWasConcurrent = shouldTimeSlice;
991     do {
992       if (exitStatus === RootDidNotComplete) {…
997       } else {
998         // ! 2. render结束，做一些检查…
1006        const finishedWork: Fiber = (root.current.alternate: any);
1007        if (
1008          renderWasConcurrent &&
1009          !isRenderConsistentWithExternalStores(finishedWork)
1010        ) {…
1019        }
1020
1021        // Check if something threw
1022        if (exitStatus === RootErrored) {…
1037        }
1038        if (exitStatus === RootFatalErrored) {…
1044        }
1045
1046        // !3. commit
1047        // 我们现在有了一个一致的树。下一步要么是commit，要么是，如果有什么被暂停了，就等待一段时间后再commit。
1048        root.finishedWork = finishedWork;
1049        root.finishedLanes = lanes;
1050        finishConcurrentRender(root, exitStatus, finishedWork, lanes);
1051      }
1052      break;
1053    } while (true);
1054  }
```

`finishConcurrentRender`

DebugReact > src > react > packages > react-reconciler > src > JS ReactFiberWorkLoop.js > ...

```
1138  function finishConcurrentRender(
1139    root: FiberRoot,
1140    exitStatus: RootExitStatus,
1141    finishedWork: Fiber,
1142    lanes: Lanes,
1143  ) {
1144    // TODO: The fact that most of these branches are identical suggests that some
1145    // of the exit statuses are not best modeled as exit statuses and should be
1146    // tracked orthogonally.
1147    switch (exitStatus) {…
1171    }
1172
1173    if (shouldForceFlushFallbacksInDEV()) {…
1182    } else {
1183      if (
1184        includesOnlyRetries(lanes) &&
1185        (alwaysThrottleRetries || exitStatus === RootSuspended)
1186      ) {…
1223      }
1224      commitRootWhenReady(
1225        root,
1226        finishedWork,
1227        workInProgressRootRecoverableErrors,
1228        workInProgressTransitions,
1229        workInProgressRootDidIncludeRecursiveRenderUpdate,
1230        lanes,
1231        workInProgressDeferredLane,
1232      );
1233    }
1234  }
```

commitRootWhenReady

JavaScript

```
commitRoot(
  root,
  recoverableErrors,
  transitions,
  didIncludeRenderPhaseUpdate,
  spawnedLane,
);
```

DebugReact > src > react > packages > react-reconciler > src > JS ReactFiberWorkLoop.js > ...

```
1236  function commitRootWhenReady(
1237    root: FiberRoot,
1238    finishedWork: Fiber,
1239    recoverableErrors: Array<CapturedValue<mixed>> | null,
1240    transitions: Array<Transition> | null,
1241    didIncludeRenderPhaseUpdate: boolean,
1242    lanes: Lanes,
1243    spawnedLane: Lane,
1244  ) {
1245    // TODO: Combine retry throttling with Suspensey commits. Right now they run
1246    // one after the other.
1247  > if (includesOnlyNonUrgentLanes(lanes)) {…
1280    }
1281
1282    // Otherwise, commit immediately.
1283    commitRoot(
1284      root,
1285      recoverableErrors,
1286      transitions,
1287      didIncludeRenderPhaseUpdate,
1288      spawnedLane,
1289    );
1290  }
```

commitRoot

commit 阶段。

commitRootImpl

1. 异步执行 passive effects

useEffect 的 effects。

2. 进入 commit 阶段

JavaScript

```
executionContext |= CommitContext;
```

3. mutation 阶段（包括 DOM 变更）

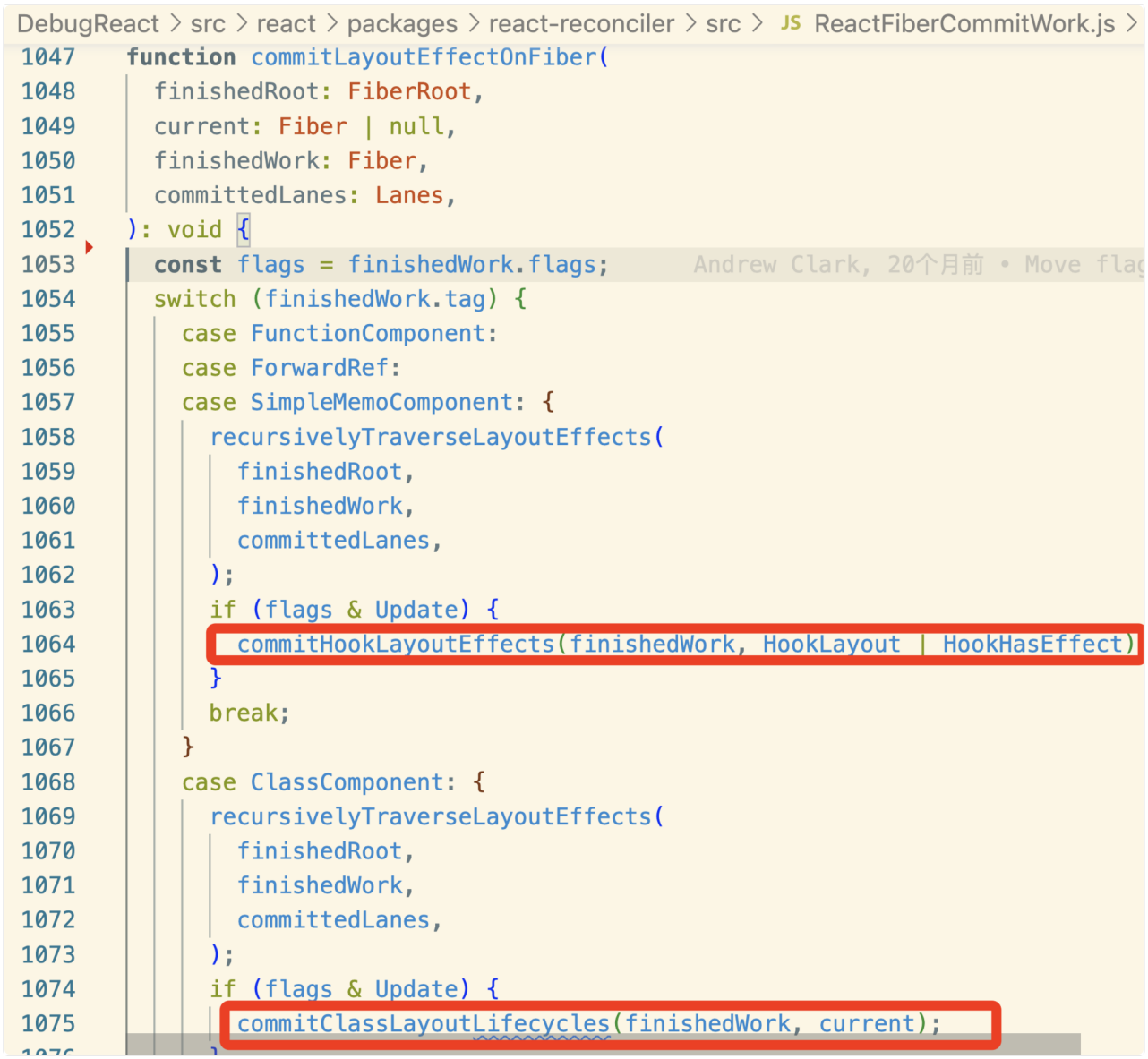
commitMutationEffects→commitMutationEffectsOnFiber

4. layout 阶段

commitLayoutEffects→commitLayoutEffectOnFiber

函数组件执行 Layout eEffects，类组件执行 componentDidMount 或者 componentDidUpdate。

由此可见，函数组件的 effects 和类组件中生命周期执行时机是不同的。


DebugReact > src > react > packages > react-reconciler > src > JS ReactFiberCommitWork.js >
1047 function commitLayoutEffectOnFiber(
1048 finishedRoot: FiberRoot,
1049 current: Fiber | null,
1050 finishedWork: Fiber,
1051 committedLanes: Lanes,
1052): void {
1053 const flags = finishedWork.flags; Andrew Clark, 20个月前 • Move flag
1054 switch (finishedWork.tag) {
1055 case FunctionComponent:
1056 case ForwardRef:
1057 case SimpleMemoComponent: {
1058 recursivelyTraverseLayoutEffects(
1059 finishedRoot,
1060 finishedWork,
1061 committedLanes,
1062);
1063 if (flags & Update) {
1064 commitHookLayoutEffects(finishedWork, HookLayout | HookHasEffect);
1065 }
1066 break;
1067 }
1068 case ClassComponent: {
1069 recursivelyTraverseLayoutEffects(
1070 finishedRoot,
1071 finishedWork,
1072 committedLanes,
1073);
1074 if (flags & Update) {
1075 commitClassLayoutLifecycles(finishedWork, current);


DebugReact > src > react > packages > react-reconciler > src > ReactFiberCommitWork.js > commitClassLayoutLifecycles
850 if (shouldProfile(finishedWork)) {…
858 } else {
859 try {
860 instance.componentDidMount();
861 } catch (error) {
862 captureCommitPhaseError(finishedWork, finishedWork.return, error);
863 }
864 }
865 } else {
866 const prevProps =
867 finishedWork.elementType === finishedWork.type
868 ? current.memoizedProps
869 : resolveDefaultProps(finishedWork.type, current.memoizedProps);
870 const prevState = current.memoizedState;
871 // We could update instance props and state here,
872 // but instead we rely on them being set during last render.
873 // TODO: revisit this when we implement resuming.
874 if (__DEV__) {…
900 }
901 if (shouldProfile(finishedWork)) {…
913 } else {
914 try {
915 instance.componentDidUpdate(
916 prevProps,
917 prevState,
918 instance.__reactInternalSnapshotBeforeUpdate,
919);