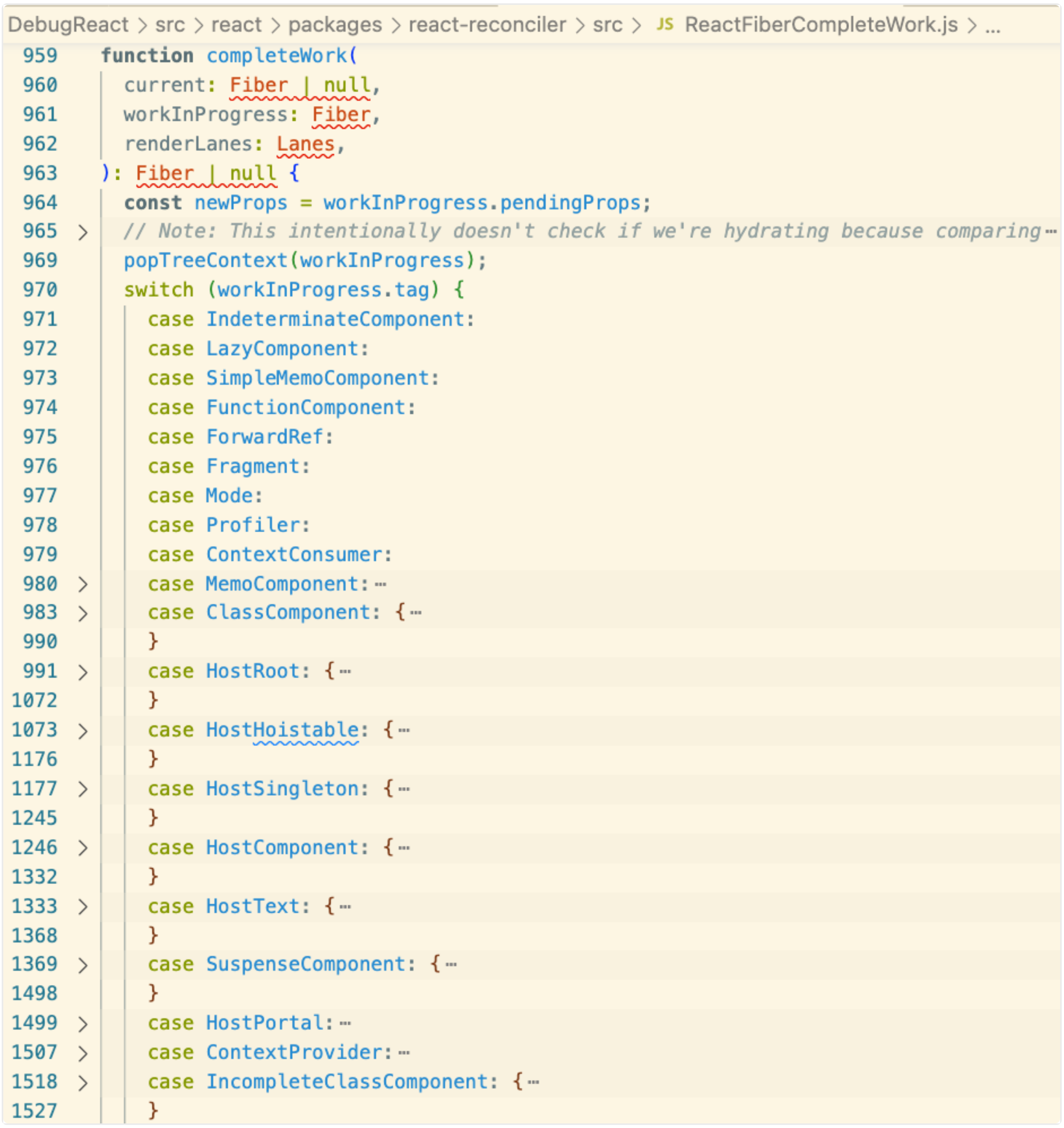
8-7 render 阶段-completeWork

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



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

```
function completeWork(
  current: Fiber | null,
  workInProgress: Fiber,
  renderLanes: Lanes,
): Fiber | null {
  const newProps = workInProgress.pendingProps;
  // Note: This intentionally doesn't check if we're hydrating because comparing⋯
  popTreeContext(workInProgress);
  switch (workInProgress.tag) {
    case IndeterminateComponent:
    case LazyComponent:
    case SimpleMemoComponent:
    case FunctionComponent:
    case ForwardRef:
    case Fragment:
    case Mode:
    case Profiler:
    case ContextConsumer:
    case MemoComponent:⋯
    case ClassComponent: {⋯
    }
    case HostRoot: {⋯
    }
    case HostHoistable: {⋯
    }
    case HostSingleton: {⋯
    }
    case HostComponent: {⋯
    }
    case HostText: {⋯
    }
    case SuspenseComponent: {⋯
    }
    case HostPortal:⋯
    case ContextProvider:⋯
    case IncompleteClassComponent: {⋯
    }
```

根据 tag 值区别不同 fiber：

`bubbleProperties`

向上冒泡 `subtreeFlags` 与 `childLanes`。

遍历 completedWork 的所有子节点：

JavaScript

```javascript
function bubbleProperties(completedWork: Fiber) {
  // 判断是否是bailout:
```

```
const didBailout =
  completedWork.alternate !== null &&
  completedWork.alternate.child === completedWork.child;

let newChildLanes = NoLanes;
let subtreeFlags = NoFlags;

if (!didBailout) {
  // ? sy
  // “向上冒泡”最早的过期时间
  // ! 遍历completedWork的所有子节点
  let child = completedWork.child;
  while (child !== null) {
    // ! 1. 将他们的lanes和childLanes合并到newChildLanes中
    newChildLanes = mergeLanes(
      newChildLanes,
      mergeLanes(child.lanes, child.childLanes),
    );
    // ! 2. 将他们的subtreeFlags和flags合并到subtreeFlags中
    subtreeFlags |= child.subtreeFlags;
    subtreeFlags |= child.flags;
    child = child.sibling;
  }
  // !
  completedWork.subtreeFlags |= subtreeFlags;
} else {
    // * bailout
    let child = completedWork.child;
    // ! 遍历completedWork的所有子节点
    while (child !== null) {
      // ! 1. 将他们的lanes和childLanes合并到newChildLanes中
      newChildLanes = mergeLanes(
        newChildLanes,
        mergeLanes(child.lanes, child.childLanes),
      );
      // “静态”标志 (Static flags) 与它们所属的 Fiber 或 Hook 共享生命周期
      // 而其他所有flags仅在单次render + commit 的生命周期内存在，因此我们应
      // ! 2. 将他们的 (subtreeFlags&StaticMask)和flags合并到subtreeFla
      subtreeFlags |= child.subtreeFlags & StaticMask;
      subtreeFlags |= child.flags & StaticMask;
```

```
        // 更新return pointer以保持树的一致性。
        child.return = completedWork;
        child = child.sibling;
      }
    completedWork.subtreeFlags |= subtreeFlags;
  }
  // !
  completedWork.childLanes = newChildLanes;
  return didBailout;
}
```

创建 DOM 节点，并添加到父节点中

如果组件是 `HostComponent`，即原生标签：

```JavaScript
const instance = createInstance(
  type,
  newProps,
  rootContainerInstance,
  currentHostContext,
  workInProgress,
);

// 把DOM子节点添加到父DOM节点中去
appendAllChildren(instance, workInProgress, false, false);
// 当节点是原生标签，比如div、span等，会在这里添加到fiber属性stateNode上
workInProgress.stateNode = instance;
```

`createInstance`

创建 DOM 节点。这里是核心代码，实际源码需要考虑特殊处理一些节点，如 SVG、select 等

react/packages/react-dom-bindings/src/client/ReactFiberConfigDOM.js

```JavaScript
```

```
let domElement = ownerDocument.createElement(type);
updateFiberProps(domElement, props);
return domElement;
```

`updateFiberProps`

把 fiber 上存储的属性更新到 DOM 节点上。

react/packages/react-dom-bindings/src/client/ReactDOMComponentTree.js

```JavaScript
export function updateFiberProps(
  node: Instance | TextInstance | SuspenseInstance,
  props: Props,
): void {
  (node: any)[internalPropsKey] = props;
}
```

`appendAllChildren`

把 DOM 节点添加到父 DOM 节点中。

`appendInitialChild`

```JavaScript
export function appendInitialChild(
  parentInstance: Instance,
  child: Instance | TextInstance,
): void {
  parentInstance.appendChild(child);
}
```