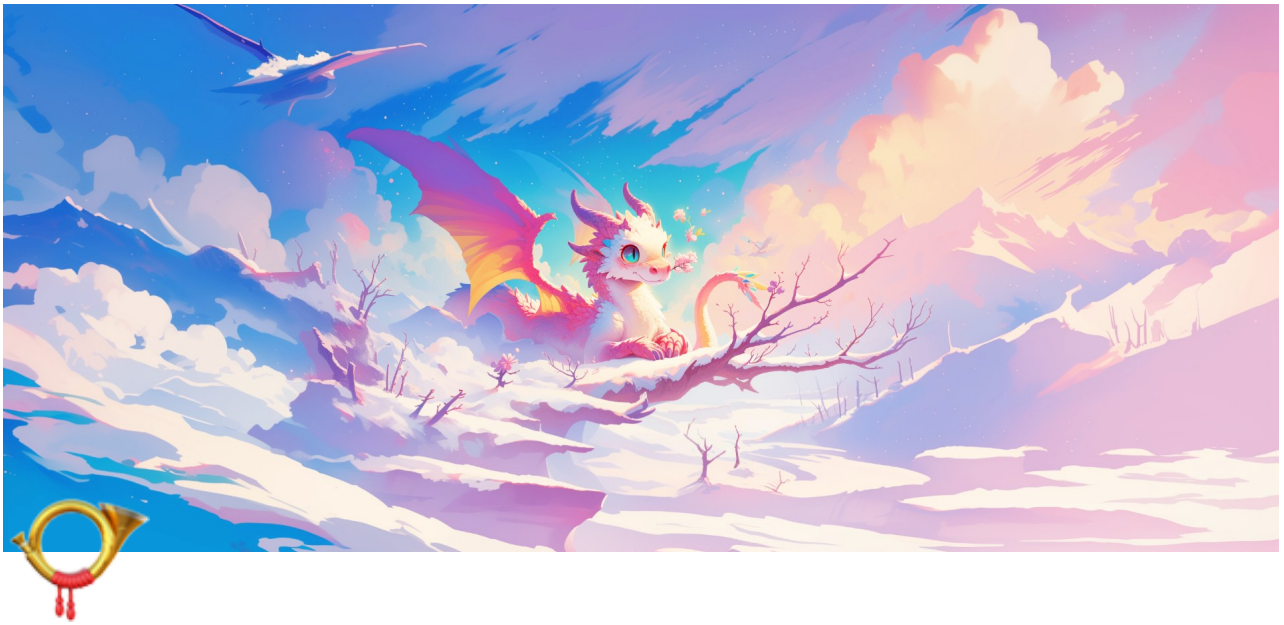


13–1 分析不同子节点类型，React VDOM DIFF 的处理

协调函数

协调子节点的核心函数是 `reconcileChildFibersImpl`，这里根据不同的子节点 newChild 的类型进行协调：

DebugReact > src > react > packages > react-reconciler > src > JS ReactChildFiber.js > createChildReconciler

```
1610  function reconcileChildFibersImpl(
1611    returnFiber: Fiber,
1612    currentFirstChild: Fiber | null,
1613    newChild: any,
1614    lanes: Lanes,
1615    debugInfo: ReactDebugInfo | null,
1616  ): Fiber | null {
1617    // This function is not recursive.…
1626    const isUnkeyedTopLevelFragment =
1627      typeof newChild === 'object' &&
1628      newChild !== null &&
1629      newChild.type === REACT_FRAGMENT_TYPE &&
1630      newChild.key === null;
1631      // 如果newChild是Fragment类型，且没有key，则直接协调其子节点
1632    if (isUnkeyedTopLevelFragment) {
1633      newChild = newChild.props.children;
1634    }
1635
1636    // Handle object types
1637    // 单个节点、数组、迭代器、promise、context
1638    if (typeof newChild === 'object' && newChild !== null) {…
1731    }
1732
1733    if (
1734      (typeof newChild === 'string' && newChild !== '') ||
1735      typeof newChild === 'number'
1736    ) {…
1745    }
1746
1747    if (__DEV__) {…
1754    }
1755
1756    // Remaining cases are all treated as empty.
1757    return deleteRemainingChildren(returnFiber, currentFirstChild);
1758  }
```

协调 Fragment

如果 newChild 是 Fragment 类型，且没有 key，则直接协调其子节点。

JavaScript

```
const isUnkeyedTopLevelFragment =
  typeof newChild === 'object' &&
  newChild !== null &&
  newChild.type === REACT_FRAGMENT_TYPE &&
  newChild.key === null;
  // 如果newChild是Fragment类型，且没有key，则直接协调其子节点
if (isUnkeyedTopLevelFragment) {
  newChild = newChild.props.children;
}
```

协调单个节点

JavaScript

```
return placeSingleChild(
    reconcileSingleElement(
      returnFiber,
      currentFirstChild,
      newChild,
      lanes,
      mergeDebugInfo(debugInfo, newChild._debugInfo),
    ),
  );
```

协调多个子节点

JavaScript

```
return reconcileChildrenArray(
  returnFiber,
  currentFirstChild,
  newChild,
  lanes,
  mergeDebugInfo(debugInfo, newChild._debugInfo),
);
```

文本节点

JavaScript

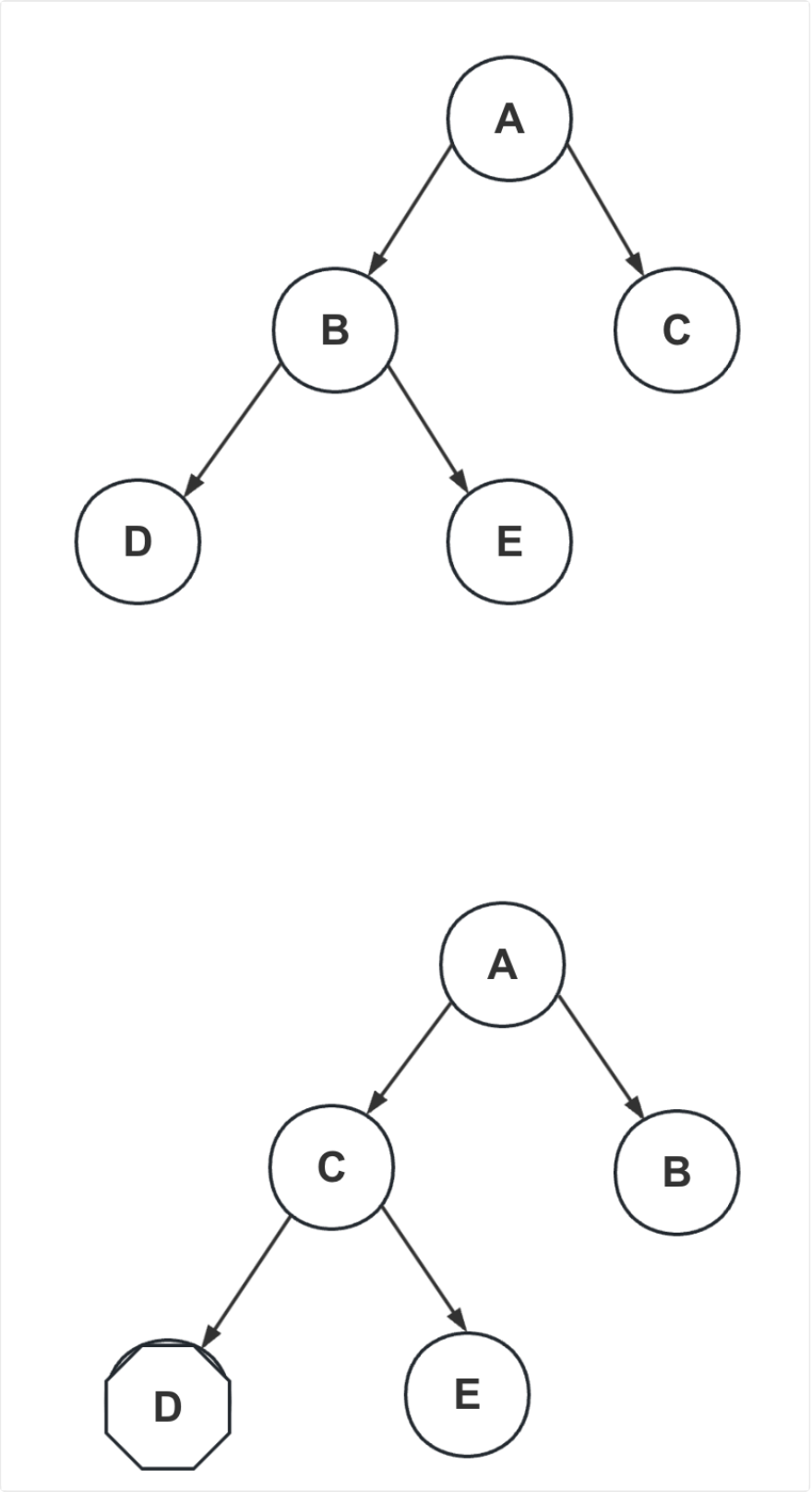
```
return placeSingleChild(
  reconcileSingleTextNode(
    returnFiber,
    currentFirstChild,
    '' + newChild,
    lanes,
```

```
  ),
);
```

协调规则

目的：构建新的子 fiber 结构。

检查是否有老节点，如果有，则检查是否可以**复用老节点**。否则，**创建新节点**。



复用老节点规则

1. 同一层级下
2. 相同 key

3. 相同 type

工具函数

删除单个节点

JavaScript

```
function deleteChild(returnFiber: Fiber, childToDelete: Fiber): void
  if (!shouldTrackSideEffects) {
    // Noop.
    return;
  }
  const deletions = returnFiber.deletions;
  if (deletions === null) {
    returnFiber.deletions = [childToDelete];
    returnFiber.flags |= ChildDeletion;
  } else {
    deletions.push(childToDelete);
  }
}
```

删除节点链表

JavaScript

```
function deleteRemainingChildren(
  returnFiber: Fiber,
  currentFirstChild: Fiber | null,
): null {
  if (!shouldTrackSideEffects) {
    return null;
  }

  let childToDelete = currentFirstChild;
  while (childToDelete !== null) {
    deleteChild(returnFiber, childToDelete);
    childToDelete = childToDelete.sibling;
```

```
    }
    return null;
  }
```