11–5 useState 源码解读

资源

1. useState

Hook 相关类型定义与初始值

JavaScript

```javascript
export type Update<S, A> = {
  lane: Lane,
  revertLane: Lane,
  action: A,
  hasEagerState: boolean,
  eagerState: S | null,
  next: Update<S, A>,
};

export type UpdateQueue<S, A> = {
  pending: Update<S, A> | null,
  lanes: Lanes,
  dispatch: (A => mixed) | null,
  lastRenderedReducer: ((S, A) => S) | null,
  lastRenderedState: S | null,
};

export type Hook = {
  memoizedState: any,
  baseState: any,
  baseQueue: Update<any, any> | null,
  queue: any,
  next: Hook | null,
};

type Dispatch<A> = A => void;

let currentHook: Hook | null = null;
let workInProgressHook: Hook | null = null;
```

函数组件挂载阶段

mountState

JavaScript

```javascript
function mountState<S>(
  initialState: (() => S) | S,
): [S, Dispatch<BasicStateAction<S>>] {
  const hook = mountStateImpl(initialState);
  const queue = hook.queue;
  const dispatch: Dispatch<BasicStateAction<S>> = (dispatchSetState.bi
    null,
    currentlyRenderingFiber,
    queue,
  ): any);
  queue.dispatch = dispatch;
  return [hook.memoizedState, dispatch];
}
```

dispatchSetState

JavaScript

```
function dispatchSetState<S, A>(
  fiber: Fiber,
  queue: UpdateQueue<S, A>,
  action: A,
): void {


  const lane = requestUpdateLane(fiber);
  // ! 1. 创建update
  const update: Update<S, A> = {
    lane,
    revertLane: NoLane,
    action,
    hasEagerState: false,
    eagerState: null,
    next: (null: any),
  };

  if (isRenderPhaseUpdate(fiber)) {
    enqueueRenderPhaseUpdate(queue, update);
  } else {
    const alternate = fiber.alternate;
    if (
      fiber.lanes === NoLanes &&
      (alternate === null || alternate.lanes === NoLanes)
    ) {
      const lastRenderedReducer = queue.lastRenderedReducer;
      if (lastRenderedReducer !== null) {
        let prevDispatcher;
        try {
          const currentState: S = (queue.lastRenderedState: any);
          const eagerState = lastRenderedReducer(currentState, action)
          update.hasEagerState = true;
          update.eagerState = eagerState;
          if (is(eagerState, currentState)) {
            // 如果state没变，组件不做更新。此处和useReducer对比下，useReduce
            enqueueConcurrentHookUpdateAndEagerlyBailout(fiber, queue,
            return;
          }
        } catch (error) {
          // Suppress the error. It will throw again in the render pha
        } finally {

        }
      }
    }
    // ! 2. 把update暂存到concurrentQueues数组中
    const root = enqueueConcurrentHookUpdate(fiber, queue, update, lan
    if (root !== null) {
      // ! 3. 调度更新
      scheduleUpdateOnFiber(root, fiber, lane);
      entangleTransitionUpdate(root, queue, lane);
    }
  }

}
```

函数组件更新阶段

JavaScript

```javascript
function updateState<S>(
  initialState: (() => S) | S,
): [S, Dispatch<BasicStateAction<S>>] {
  return updateReducer(basicStateReducer, initialState);
}
```