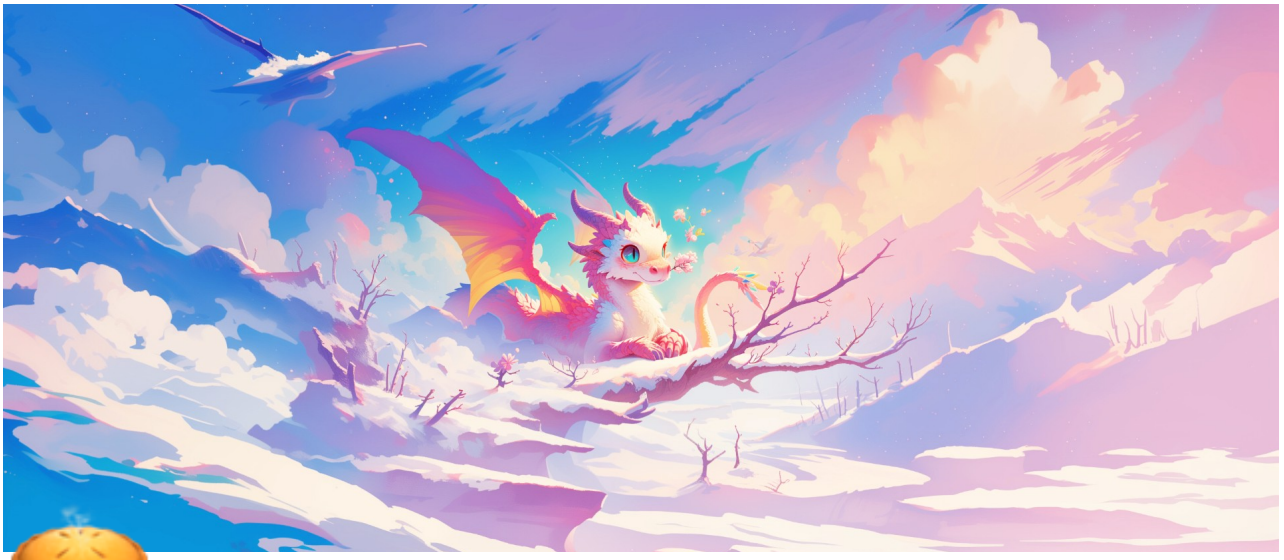

12-7 实现 useState

TypeScript

```
function FunctionComponent() {
  const [count1, setCount1] = useReducer((x) => x + 1, 1);
  const [count2, setCount2] = useState(1);

  // const arr = count1 % 2 === 0 ? [0, 1, 2, 3, 4] : [0, 1, 2, 3];
  // const arr = count1 % 2 === 0 ? [0, 1, 2, 3, 4] : [0, 1, 2, 4];

  const arr = count1 % 2 === 0 ? [0, 1, 2, 3, 4] : [3, 2, 0, 4, 1];

  // old 0, 1, 2, 4
  // new 0, 1, 2, 3, 4
  // 1个before 4

  // old 3, 2, 0, 4, 1
  // new 0, 1, 2, 3, 4
  // 3个before null

  const _cls = count1 % 2 === 0 ? "red green_bg" : "green red_bg";

  // 0 删除
  return (
    <div className="border">
```

```
      <h3 className={_cls}>函数组件</h3>
      <button
        onClick={() => {
          setCount1();
        }}
      >
        {count1}
      </button>
      <button
        onClick={() => {
          setCount2(count2 + 1);
        }}
      >
        {count2}
      </button>
      <ul>
        {arr.map((item) => (
          <li key={"li" + item}>{item}</li>
        ))}
      </ul>

      {/* {count1 % 2 === 0 ? (
        <button
          onClick={() => {
            setCount1();
          }}
        >
          {count1}
        </button>
      ) : (
        <span
          onClick={() => {
            setCount1();
          }}
        >
          react
        </span>
      )} */}
    </div>
  );
}
```

useState

```TypeScript
export function useState<S>(initialState: (() => S) | S) {
  const init = isFn(initialState) ? (initialState as any)() : initialS
  return useReducer(null, init);
}
```

useReducer

```TypeScript
export function useReducer<S, I, A>(
  reducer: ((state: S, action: A) => S) | null,
  initialArg: I,
  init?: (initialArg: I) => S
) {
  // ! 1.  构建hook链表(mount、update)
  const hook: Hook = updateWorkInProgressHook(); //{ memoizedState: nu

  let initialState: S;
  if (init !== undefined) {
    initialState = init(initialArg);
  } else {
    initialState = initialArg as any;
  }

  // ! 2. 区分函数组件是初次挂载还是更新
  if (!currentlyRenderingFiber?.alternate) {
    // mount
    hook.memoizedState = initialState;
  }

  // ! 3. dispatch
  const dispatch = dispatchReducerAction.bind(
```

```
    null,
    currentlyRenderingFiber!,
    hook,
    reducer as any
  );

  return [hook.memoizedState, dispatch];
}
```

调度更新

TypeScript

```
export function scheduleUpdateOnFiber(
  root: FiberRoot,
  fiber: Fiber,
  isSync?: boolean
) {
  workInProgressRoot = root;
  workInProgress = fiber;

  if (isSync) {
    queueMicrotask(() => performConcurrentWorkOnRoot(root));
  } else {
    ensureRootIsScheduled(root);
  }
}
```