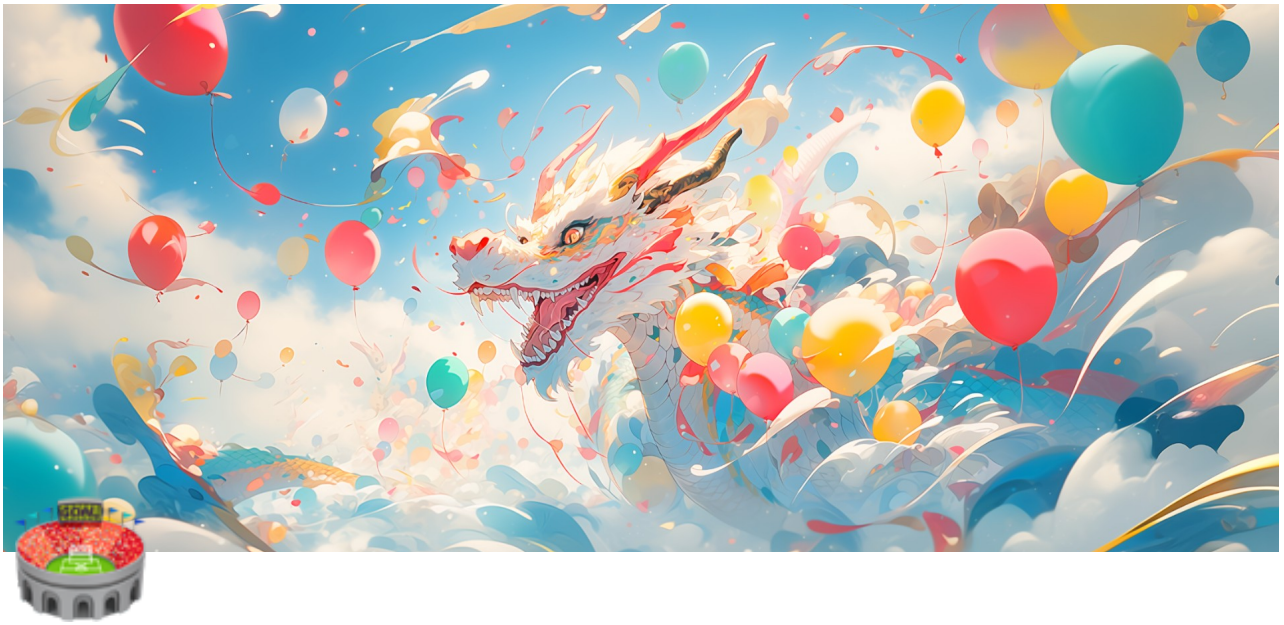

12–1 模拟事件，初步实现 useReducer

示例

```
TypeScript
```

```
function FunctionComponent() {
  const [count1, setCount1] = useReducer((x) => x + 1, 0);
  return (
    <div>
      <h3>函数组件</h3>
      <button
        onClick={() => {
          setCount1();
          // console.log("click"); //sy-log
        }}
      >
        {count1}
      </button>
    </div>
  );
}

ReactDOM.createRoot(document.getElementById("root") as HTMLElement).re
  (<FunctionComponent />) as any
);
```

react

packages/react/index.ts

TypeScript

```
export { useReducer } from "react-reconciler/src/ReactFiberHooks";
```

模拟事件

JavaScript

```
// 初始化属性
function finalizeInitialChildren(domElement: Element, props: any) {
  for (const propKey in props) {
    const nextProp = props[propKey];
    if (propKey === "children") {
      if (isStr(nextProp) || isNum(nextProp)) {
```

```
        // 属性
        domElement.textContent = nextProp;
      }
    } else {
      // 3. 设置属性
      (domElement as any)[propKey] = nextProp;
      if (propKey === "onClick") {
        domElement.addEventListener("click", nextProp);
      }
    }
  }
}
```

beginWork

ReactFiberBeginWork.ts

TypeScript

```
function updateFunctionComponent(current: Fiber | null, workInProgress
  const { type, pendingProps } = workInProgress;
  const children = renderWithHooks(current, workInProgress, type, pend
  reconcileChildren(current, workInProgress, children);
  return workInProgress.child;
}
```

定义 useReducer

TypeScript

```
// import { scheduleUpdateOnFiber } from "./ReactFiberWorkLoop";

type Hook = {
  memoizedState: any;
  next: null | Hook;
};

// 当前正在工作的函数组件的fiber
```

```ts
let currentlyRenderingFiber: Fiber | null = null;
let workInProgressHook: Hook | null = null;
let currentHook: Hook | null = null;

export function renderWithHooks<Props>(
  current: Fiber | null,
  workInProgress: Fiber,
  Component: any,
  props: Props
): any {
  currentlyRenderingFiber = workInProgress;
  workInProgress.memoizedState = null;

  let children = Component(props);

  finishRenderingHooks();

  return children;
}

function finishRenderingHooks() {
  currentlyRenderingFiber = null;
  currentHook = null;
  workInProgressHook = null;
}

export function useReducer<S, I, A>(
  reducer: (state: S, action: A) => S,
  initialArg: I,
  init?: (initialArg: I) => S
) {
  // ! 1.  构建hook链表
  const hook: Hook = { memoizedState: null, next: null };

  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
  hook.memoizedState = initialState;

  // ! 3. dispatch
  const dispatch = (action: A) => {
    const newValue = reducer(initialState, action);

    // scheduleUpdateOnFiber
    console.log(
      "%c [  ]-5",
      "font-size:13px; background:pink; color:#bf2c9f;",
      newValue
    );
  };

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