14–3 useMemo 与 useCallback

`useMemo` 经常与 `useCallback` 一同出现。当尝试优化子组件时，它们都很有用。他们会 记住（或者说，缓存）正在传递的东西：

```typescript
// 在 React 内部的简化实现
function useCallback(fn, dependencies) {
  return useMemo(() => fn, dependencies);
}
```

- **`useMemo` 缓存函数调用的结果**。在这里，它缓存了调用 `computeRequirements(product)` 的结果。除非 `product` 发生改变，否则它将不会发生变化。这让你向下传递 `requirements` 时而无需不必要地重新渲染 `ShippingForm`。必要时，React 将会调用传入的函数重新计算结果。
- **`useCallback` 缓存函数本身**。不像 `useMemo`，它不会调用你传入的函数。相反，它缓存此函数。从而除非 `productId` 或 `referrer` 发生改变，`handleSubmit` 自己将不会发生改变。这让你向下传递 `handleSubmit` 函数而无需不必要地重新渲染 `ShippingForm`。直至用户提交表单，你的代码都将不会运行。

useMemo

react/packages/react-reconciler/src/ReactFiberHooks.js

TypeScript

```
function mountMemo<T>(
  nextCreate: () => T,
  deps: Array<mixed> | void | null,
): T {
  const hook = mountWorkInProgressHook();
  const nextDeps = deps === undefined ? null : deps;
  const nextValue = nextCreate();
  hook.memoizedState = [nextValue, nextDeps];
  return nextValue;
}

function updateMemo<T>(
  nextCreate: () => T,
  deps: Array<mixed> | void | null,
): T {
  const hook = updateWorkInProgressHook();
  const nextDeps = deps === undefined ? null : deps;
  const prevState = hook.memoizedState;
  if (nextDeps !== null) {
    const prevDeps: Array<mixed> | null = prevState[1];
    if (areHookInputsEqual(nextDeps, prevDeps)) {
      return prevState[0];
    }
  }
  const nextValue = nextCreate();
  hook.memoizedState = [nextValue, nextDeps];
  return nextValue;
}
```

useCallback

react/packages/react-reconciler/src/ReactFiberHooks.js

JavaScript

```
function mountCallback<T>(callback: T, deps: Array<mixed> | void | nul
  const hook = mountWorkInProgressHook();
  const nextDeps = deps === undefined ? null : deps;
  hook.memoizedState = [callback, nextDeps];
  return callback;
}

function updateCallback<T>(callback: T, deps: Array<mixed> | void | nu
  const hook = updateWorkInProgressHook();
  const nextDeps = deps === undefined ? null : deps;
  const prevState = hook.memoizedState;
  if (nextDeps !== null) {
    const prevDeps: Array<mixed> | null = prevState[1];
    if (areHookInputsEqual(nextDeps, prevDeps)) {
      return prevState[0];
    }
  }
  hook.memoizedState = [callback, nextDeps];
  return callback;
}
```