5-5 如何实现一个 requestIdleCallback

React 使用 MessageChannel 创建宏任务，来实现异步任务队列，以实现异步更新，确保 React 在执行更新时能够合并多个更新操作，并在下一个宏任务中一次性更新，以提高性能并减少不必要的重复渲染，从而提高页面性能和用户体验。

模拟 requestIdleCallback：MessageChannel

React 中的任务调度使用 MessageChannel 模拟 requestIdleCallback，而不是 setTimeout。

https://developer.mozilla.org/zh-CN/docs/Web/API/setTimeout#实际延时比设定值更久的原因：最小延迟时间

TypeScript

```
let isMessageLoopRunning = false;

function requestHostCallback() {
  if (!isMessageLoopRunning) {
    isMessageLoopRunning = true;
    schedulePerformWorkUntilDeadline();
  }
```

```
}

const performWorkUntilDeadline = () => {
  if (isMessageLoopRunning) {
    const currentTime = getCurrentTime();
    startTime = currentTime;
    let hasMoreWork = true;
    try {
      hasMoreWork = flushWork(currentTime);
    } finally {
      if (hasMoreWork) {
        schedulePerformWorkUntilDeadline();
      } else {
        isMessageLoopRunning = false;
      }
    }
  }
};

const channel = new MessageChannel();
const port = channel.port2;
channel.port1.onmessage = performWorkUntilDeadline;
function schedulePerformWorkUntilDeadline() {
  port.postMessage(null);
}

function flushWork(initialTime: number) {
  isHostCallbackScheduled = false;

  isPerformingWork = true;
  const previousPriorityLevel = currentPriorityLevel;
  try {
    return workLoop(initialTime);
  } finally {
    currentTask = null;
    currentPriorityLevel = previousPriorityLevel;
    isPerformingWork = false;
  }
}
```