web3面试： 发送交易之后，如何确定交易的状态，以wagmi举例

在 Wagmi 框架中，发送交易后确定交易状态主要通过**组合多个钩子（Hooks）** 实现，覆盖从交易提交到最终确认的全生命周期。以下是基于 Wagmi 的完整流程和关键钩子解析：

1. 交易状态的生命周期与对应钩子

状态阶段	Wagmi 钩子	核心作用与返回值
Pending（待处理）	`useSendTransaction`	提交交易至网络，返回 `hash`（交易哈希）和 `isPending`（用户签名中）
打包确认中	`useWaitForTransactionReceipt`	监听交易是否被打包进区块，返回 `isLoading`（等待中）和 `isSuccess`（打包成功）
最终确认	`useTransactionConfirmations`	计算交易所在区块后的确认数（如6个区块），返回 `confirmations`（确认数） [1]
事件触发	`useWatchContractEvent`	监听合约事件（如 `Transfer`），实时响应链上日志 [20]

2. 核心钩子使用示例

(1) 发送交易并获取哈希

Tsx

```tsx
import { useSendTransaction, useWaitForTransactionReceipt } from 'wagmi';
import { parseEther } from 'viem';

function SendButton() {
  const { data: hash, sendTransaction, isPending } = useSendTransaction();

  const handleSend = () => {
      sendTransaction({
    to: '0x...',
    value: parseEther('0.001'),
      });
    };

  return (
    <button onClick={handleSend} disabled={isPending}>
     {isPending ? '签名中...' : '发送交易'}
    </button>
    );
}
```

- `hash`：交易唯一标识，用于后续跟踪。
- `isPending`：用户是否在钱包中确认交易（UI 禁用按钮的依据）。

(2) 监听交易打包状态

```tsx
const { isLoading: isConfirming, isSuccess: isMined } = useWaitForTransactionReceipt({
  hash, // 传入上一步的 hash
});

// 状态反馈
{isConfirming && <p>交易打包中...</p>}
{isMined && <p>交易已打包！</p>}
```

- `isMined` 为 `true` 表示交易成功上链（1次确认）。

(3) 计算区块确认数

```tsx
```

```
const { data: confirmations } = useTransactionConfirmations({
  hash,
});

// 6区块确认（防重组）
if (confirmations >= 6) {
 console.log('交易最终确认');
}
```

- `confirmations`：交易所在区块后的新增区块数，用于降低链重组风险 [1]。

(4) 监听合约事件（高级）

Tsx

```
useWatchContractEvent({
 address: '0x合约地址',
 abi: ERC20_ABI,
 eventName: 'Transfer',
 onLogs: (logs) => {
  logs.forEach(log => console.log('转账事件:', log));
  },
});
```

- 实时捕获与交易相关的链上事件（如代币转账） [20]。

⚠ 3. 失败处理与边界场景

场景	检测方式
用户拒绝签名	`useSendTransaction` 返回 `error`，类型为 `"UserRejectedRequestError"`
交易执行失败	`useWaitForTransactionReceipt` 返回的 `receipt` 中 `status === 0`（失败）
交易超时/被丢弃	设置超时阈值（如10分钟），若 `useWaitForTransactionReceipt` 长期无响应则重发

▮ 4. 最佳实践与面试要点

1. **状态管理组合**：
   - 基础流程：`useSendTransaction` → `useWaitForTransactionReceipt` → `useTransactionConfirmations`。
   - 事件驱动：`useWatchContractEvent` 替代轮询，降低 RPC 负载 [20]。

2. **性能优化**：
   - **轻量级监听**：对高频交易（如 DEX）优先用事件监听，减少 `getTransactionReceipt` 轮询 [20]。
   - **链重组处理**：通过 `confirmations >= N`（N=6）确保最终性，或监听区块哈希变化自动重试 [1]。
3. **用户体验**：
   - 使用 `useTransactor` 封装交易逻辑，提供完整 UI 反馈（按钮加载态、Toast 提示）：

```tsx
const writeTx = useTransactor();
await writeTx(sendTransaction, { blockConfirmations: 1 }); // 设置最小确认数
```

✅ 总结回答（面试精简版）

> 在 Wagmi 中确定交易状态需分三步：
>
> 1. **提交阶段**：通过 `useSendTransaction` 获取 `hash` 并监听 `isPending`（用户签名）；
> 2. **打包阶段**：通过 `useWaitForTransactionReceipt` 监听 `isSuccess`（交易上链），失败时检查 `status=0`；
> 3. **最终确认**：用 `useTransactionConfirmations` 计算区块确认数（≥6 防重组） [1]。
>
> **高级场景**：
>
> - 关键交易监听合约事件（`useWatchContractEvent`）实时响应；
> - 使用 `useTransactor` 封装可统一处理 UI 状态与重试逻辑 。

https://wagmi.sh/react/api/hooks/useTransactionConfirmations

https://wagmi.sh/react/guides/send-transaction

https://wagmi.sh/core/api/actions/getTransactionConfirmations

https://wagmi.sh/vue/guides/send-transaction

https://wagmi.sh/react/api/hooks/useTransactionReceipt

https://wagmi.sh/react/guides/write-to-contract

https://docs.polkadot.com/develop/smart-contracts/libraries/wagmi/

https://github.com/wevm/wagmi/discussions/4306

https://docs.reown.com/appkit/recipes/wagmi-send-transaction

https://wagmi.sh/vue/guides/write-to-contract

https://wagmi.sh/core/api/actions/getTransactionReceipt

https://wagmi.sh/react/api/hooks/useTransactionReceipt

https://wagmi.sh/core/api/actions

https://wagmi.sh/core/api/actions/getTransactionCount

https://wagmi.sh/core/api/actions/getTransaction

https://juejin.cn/post/7162597516326895629

https://wagmi.sh/core/api/actions/getWalletClient

https://ethereum.stackexchange.com/questions/155384/how-to-get-receipt-in-wagmi-viem-for-a-transaction-issued-with-safe-on-walletc

https://stackoverflow.com/questions/76691145/how-can-i-get-events-from-a-transaction-receipt-using-wagmi

https://wagmi.sh/react/api/hooks/useWatchContractEvent

https://wagmi.sh/core/api/actions/watchContractEvent

https://wagmi.sh/react/api/hooks/useAccountEffect

https://wagmi.sh/react/why

https://blog.softbinator.com/listen-blockchain-events-wagmi/

https://stackoverflow.com/questions/76691145/how-can-i-get-events-from-a-transaction-receipt-using-wagmi

https://ethereum.stackexchange.com/questions/144431/wagmi-usecontractwrite-response-not-including-event-parameters

https://www.tibyverse.xyz/articles/introduction-to-solidity-events-and-errors

https://wagmi.sh/core/api/actions/watchPendingTransactions

https://wagmi.sh/react/api/hooks/useWatchPendingTransactions

https://wagmi.sh/react/api/hooks/useWaitForTransactionReceipt

https://wagmi.sh/core/api/actions/waitForTransactionReceipt

https://wagmi.sh/react/api/hooks/useWaitForCallsStatus

https://docs.pimlico.io/permissionless/wagmi/reference/hooks/useWaitForTransactionReceipt

https://github.com/wevm/wagmi/discussions/3900

https://github.com/wagmi-dev/wagmi/issues/2461

https://ethereum.stackexchange.com/questions/165473/safe-connector-with-wagmi-library

https://wagmi.sh/react/guides/send-transaction

https://wagmi.sh/vue/guides/send-transaction

https://wagmi.sh/react/api/hooks/useTransactionReceipt

https://1.x.wagmi.sh/examples/send-transaction

https://docs.reown.com/appkit/recipes/wagmi-send-transaction

https://wagmi.sh/react/guides/write-to-contract

https://github.com/wevm/wagmi/discussions/3900

https://github.com/wagmi-dev/wagmi/discussions/2928

https://wagmi.sh/react/api/hooks

https://wagmi.sh/react/api/hooks/useTransaction

https://learnblockchain.cn/article/6913

https://1.x.wagmi.sh/react/hooks/useTransaction

https://github.com/wevm/wagmi/discussions/290

https://juejin.cn/post/7397030700455297065

https://wagmi.sh/react/guides/send-transaction

https://wagmi.sh/react/api/hooks/useTransactionConfirmations

https://wagmi.sh/react/api/hooks/useSendTransaction

https://web3auth.io/docs/sdk/web/react/ethereum-hooks

https://wagmi.sh/react/guides/send-transaction

https://wagmi.sh/react/api/hooks/useTransaction

https://github.com/wevm/wagmi/discussions/85

https://wagmi.sh/vue/guides/send-transaction

https://docs.polkadot.com/develop/smart-contracts/libraries/wagmi/

https://docs.reown.com/appkit/recipes/wagmi-send-transaction

https://ethereum.stackexchange.com/questions/155384/how-to-get-receipt-in-wagmi-viem-for-a-transaction-issued-with-safe-on-walletc

https://stackoverflow.com/questions/78796121/sendtransaction-vs-writecontract-with-functionname-transfer

https://wagmi.sh/react/api/hooks/useTransactionConfirmations

https://juejin.cn/post/7465730788177461289

https://github.com/wevm/wagmi/discussions/3989

https://docs.pimlico.io/permissionless/wagmi/reference/hooks/useWaitForTransactionReceipt

https://wagmi.sh/react/api/hooks/useSendTransaction

https://wagmi.sh/react/api/hooks/useWaitForTransactionReceipt

(注:文档部分内容可能由AI生成)