dapp 前端面试题

前言

1. 本身的前端开发能力非常重要，也需要精心准备
2. 依然偏好react的技术栈，vue技术栈的有但是非常少

1. 钱包地址是什么?

钱包地址是区块链网络中用于标识用户账户的唯一标识符，类似于银行账户号码。它由公钥经过特定算法生成，用于接收和发送加密货币或数字资产。钱包地址通常是由一串字母和数字组成的字符串，确保了交易的准确性和安全性。

2. 怎么得到钱包地址的?

要获得钱包地址，需要生成一对密钥：

- **生成私钥**：首先，通过密码学算法生成一个随机的私钥。私钥是保密的，必须安全存储，不能泄露。
- **生成公钥**：使用加密算法（如椭圆曲线数字签名算法，ECDSA）从私钥推导出对应的公钥。
- **生成钱包地址**：对公钥进行哈希和编码处理，生成最终的钱包地址。

通常，用户可以使用钱包软件或应用（如MetaMask、Trust Wallet等）来自动完成上述过程。这些工具会在创建新账户时为用户生成密钥对和钱包地址，无需手动处理复杂的加密算法。

1. 什么是gas，如何计算要一笔交易花费多少gas

在以太坊网络中，当你发起一笔交易时，需要支付一定数量的 gas。Gas 是以太坊网络中衡量交易执行成本的单位，每一条指令的执行都需要一定数量的 gas。因此，需要在交易前计算出所需的 gas 数量，以确保交易能够被成功执行。

计算 gas 的公式是：**Gas Limit × Gas Price = Transaction Fee**。其中，Gas Limit 指交易所能够消耗的最大 gas 数量，Gas Price 指每单位 gas 的价格，而 Transaction Fee 则是交易的手续费。

2. 手写dapp广播流程?

- dapp对智能合约调用参数进行拼接
- 生成abi
- 调用三方钱包广播交易
- 拿到txHash
- 轮循rpc查询txHash对应交易状态

3. wallectConnet

WalletConnect 是一个开源协议，用于连接去中心化应用（DApps）与各种钱包之间。它允许用户通过扫描二维码或使用深度链接来安全地连接他们的移动钱包应用（如 Trust Wallet、MetaMask Mobile 等）到桌面或网页应用。

WalletConnect 的主要特点：

1. **跨钱包兼容性**：支持多种钱包应用，不需要用户下载特定的钱包。
2. **无缝体验**：用户可以通过扫描二维码来连接钱包，而不需要手动输入长串的地址。
3. **安全性**：WalletConnect 使用加密的通道来传输消息，确保交易和数据的安全性。
4. **支持多链**：不仅支持以太坊，还可以扩展到其他区块链网络。

工作原理：

1. **生成二维码**：DApp 生成一个 QR 码，二维码中包含了连接信息。
2. **扫描二维码**：用户在他们的移动钱包应用中扫描这个二维码。
3. **建立连接**：一旦二维码被扫描，钱包应用与 DApp 之间建立一个加密的 WebSocket 连接。
4. **交易签名**：用户可以在钱包应用中确认交易或其他操作，签名后，交易信息会通过 WalletConnect 传输回 DApp

4. 什么是erc20

阐述主要方法

5. 什么是erc721

阐述主要方法

6. EVM怎么知道dapp要调用某个方法?

EVM通过交易中的数据部分来知道dApp要调用哪个智能合约的方法。数据的前4个字节（方法选择器）用于识别 具体的方法，而ABI则提供了如何解码和执行这些方法的信息。

可以展开讲讲abi。

7. eip1193

EIP-1193 是以太坊改进提案（Ethereum Improvement Proposal）中的一项，旨在标准化以太坊钱包与去中心化应用（DApp）之间的通信接口。EIP-1193 提供了一种统一的方法，让 DApp 可以与不同的以太坊钱包进行交互。这个提案的核心目标是解决不同钱包之间接口不一致的问题，从而简化 DApp 的开发流程。

EIP-1193 的关键内容

1. **统一接口**：定义了 DApp 和钱包之间的标准通信接口，包括方法、事件和参数格式，使得 DApp 可以与任何遵循该标准的钱包进行交互。
2. **基本 API**：定义了一组基本的 API 方法，如 `eth_requestAccounts`、`eth_accounts` 和 `eth_chainId`，用于获取用户账户信息、链 ID 等。
3. **事件机制**：提供了 `accountsChanged` 和 `chainChanged` 事件，通知 DApp 用户账户或网络链发生变化。

EIP-1193 的主要方法和事件

方法

- **`eth_requestAccounts`**：请求用户授权 DApp 访问其账户。返回一个包含用户账户地址的数组。

```
ethereum.request({ method: 'eth_requestAccounts' })
  .then(accounts => console.log(accounts))
  .catch(error => console.error(error));
```

- **`eth_accounts`**：获取已授权的用户账户列表。

```
ethereum.request({ method: 'eth_accounts' })
  .then(accounts => console.log(accounts))
  .catch(error => console.error(error));
```

- **`eth_chainId`**：获取当前连接的链的 ID。

```
ethereum.request({ method: 'eth_chainId' })
  .then(chainId => console.log(chainId))
  .catch(error => console.error(error));
```

事件

- **`accountsChanged`**：当用户切换账户时触发。传递一个账户地址的数组作为参数。

```
ethereum.on('accountsChanged', accounts => {
  console.log('Accounts changed:', accounts);
});
```

- **`chainChanged`**：当用户切换网络时触发。传递一个链 ID 作为参数。

```
ethereum.on('chainChanged', chainId => {
  console.log('Chain changed:', chainId);
});
```

使用场景

EIP-1193 主要用于需要与以太坊钱包进行交互的 DApp，包括获取用户账户、获取网络信息、处理账户和网络变更等场景。它对钱包和 DApp 之间的交互进行标准化，从而使得开发者可以更加轻松地创建与不同钱包兼容的应用。

8. Wei，gwei，decimal, 以及相应的bignumber的处理

9. ethers 和 viem 保证相当程度的熟悉