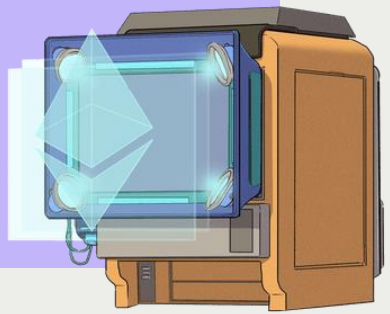
免责声明：

本课程所涉及案例仅为学习使用，不构成投资建议。
请谨慎辨别。

在中华人民共和国，区块链上的可转移数字资产（包括但不限于代币）与法币的直接或间接交换是违法行为，请遵守相关法律法规，避免参与任何非法活动。

第四讲 使用 ethers.js 和区块链交互（上）

主讲人：steven



第四讲 使用 ethers.js 和区块链交互（上）

库的选择

目前常用于此工作的工具有 *Web3.js*、*ethers.js*、*Web3-wrapper*、*Alchemyweb3* 和 *viem* 等。

注意：不同库之间有较大差异，请在具体编码过程中注意。ethers.js v6 和 v5 存在较大差异，如无特殊说明后续皆采用 v6 版本。

第四讲 使用 ethers.js 和区块链交互（上）

ethers.js 的几个类

- Provider 是一个提供了对区块链及其状态的只读访问的抽象类；
- Signer 是一个可以使用私钥对消息和交易进行签名，授权网络向您的账户收取以太币以执行操作的抽象类；
- Contract 是一个与以太坊网络上特定合约连接的抽象类；
- Transaction 是一个提供区块链交易信息查询的抽象类；
- 交易在区块链上记录将产生收据，Receipt 是一个访问收据对象的抽象类。

第四讲 使用 ethers.js 和区块链交互（上）

安装和引入方法（浏览器直接引入 ESM）

```
<script type="module">
  import { ethers } from
"https://cdnjs.cloudflare.com/ajax/libs/ethers/6.7.0/ethers.min.js";
  // Your code here...
</script>
```

第四讲 使用 ethers.js 和区块链交互（上）

安装和引入方法（包管理器安装）

项目根目录执行 npm install ethers（如果需要采用其他包管理器安装请自行选择合适的命令），再在需要的地方引入：

```
// 完整引入
import { ethers } from "ethers";
// 部分引入
import { BrowserProvider, parseUnits } from "ethers";
// 按需引入
import { HDNodeWallet } from "ethers/wallet";
```

第四讲 使用 ethers.js 和区块链交互（上）

构建一个 Provider 的实例（方法一）

使用 MetaMask 等浏览器插件挂载到 windows 对象上的 Eip1193Provider 实例：

```
import { ethers } from "ethers";

const provider = window.ethereum ? new
ethers.BrowserProvider(window.ethereum) : ethers.getDefaultProvider();
```

第四讲 使用 ethers.js 和区块链交互（上）

构建一个 Provider 的实例（方法二）

非浏览器环境下运行等情况，常常使用第三方服务（如 INFURA）申请的 url 来生成：

```
import { ethers } from "ethers";

const provider = new ethers.JsonRpcProvider(url);
```

第四讲 使用 ethers.js 和区块链交互（上）

Provider 实例的使用

注意：这上面两种方法中存在三种 provider，分别来自 new BrowserProvider()、getDefaultProvider() 和 new JsonRpcProvider()，它们有较大差异，但都能对区块链的基本信息进行查询。

第四讲 使用 ethers.js 和区块链交互（上）

Provider 实例的使用

```
// 获取当前区块区块高度
const currentBlockNumber = provider.getBlockNumber();
// 获取当前连接网络
const network = provider.getNetwork();
// 获取指定账户上的余额
const balance = await provider.getBalance("ethers.eth");
// 获取指定账户交易数量
const transactionCount = await provider.getTransactionCount("ethers.eth");
```

第四讲 使用 ethers.js 和区块链交互（上）

构建一个 Signer 的实例

```
const signer = await provider.getSigner();
```

注意：由 getDefaultProvider() 生成的 provider 只能执行只读处理，所以无法执行 getSigner() 方法。

第四讲 使用 ethers.js 和区块链交互（上）

Signer 实例的使用

Signer 实例可以直接查询当前账号的相关信息，试举一例：

```
// 获取当前账号的地址
const address = signer.getAddress();
```

Signer 实例也可以直接向指定地址转账，这里试向 ethers.eth 转 0.1 eth：

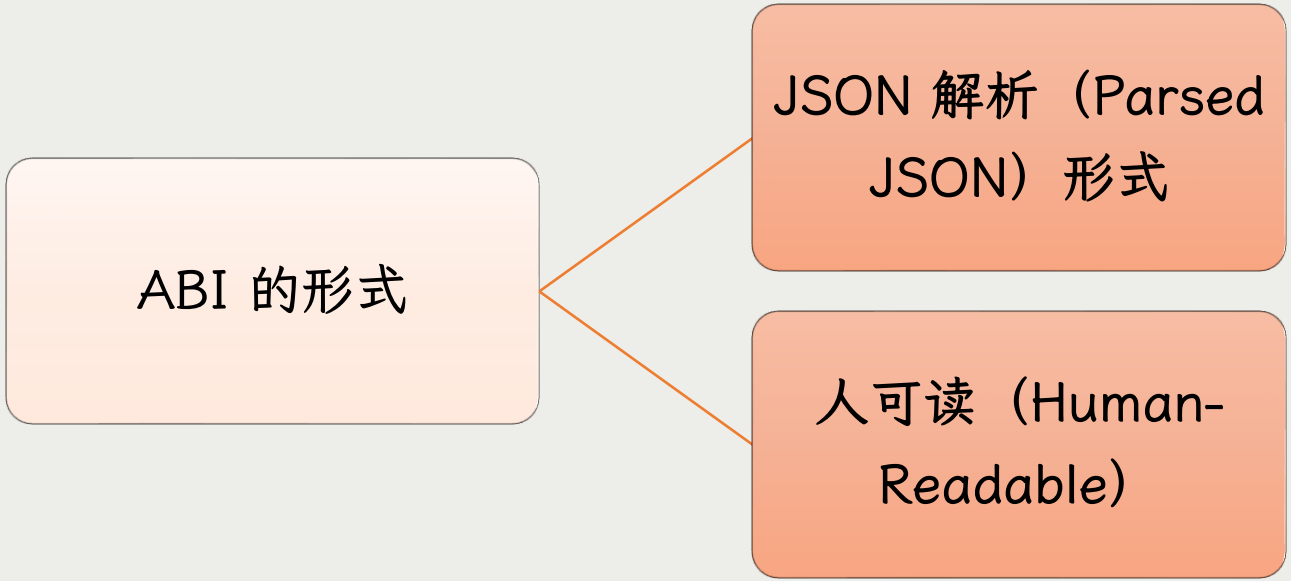
```
const tx = await signer.sendTransaction({
  to: "ethers.eth",
  value: parseEther("0.1")
});
const receipt = await tx.wait();
```

第四讲 使用 ethers.js 和区块链交互（上）

调用合约——ABI

应用程序二进制接口（Application Binary Interface，ABI）是两个程序模块之间的接口，通常这两个模块一个在机器代码（二进制）级别，另一个在程序级别。通过此接口，可以更方便地调用底层函数。

第四讲 使用 ethers.js 和区块链交互（上）



第四讲 使用 ethers.js 和区块链交互（上）

构造合约实例

```
const contract = new Contract(CONTRACT_ADDRESS, ABI, provider);
```

```
const contract = new Contract(CONTRACT_ADDRESS, ABI, signer);
```

第四讲 使用 ethers.js 和区块链交互（上）

一个相对完整的例子

```
const abi = [
  "event Approval(address indexed owner, address indexed spender, uint256 value)",
  "function decimals() view returns (uint8)",
  "function transfer(address recipient, uint256 amount) returns (bool success)"
]
// 构建 Contract 实例
// 注意此处传入的是 Provider 实例，可以执行写操作；反之如果传入 Signer 实例，则不能执行写操作；
const contract = new Contract(CONTRACT_ADDRESS, abi, provider);
// 执行只读函数，直接返回结果
const decimals = await contract.decimals();
// 换算获取 amount
const amount = parseUnits("1.0", 18);
// 执行写操作，返回交易对象
const tx = await contract.transfer(RECIPIENT_ADDRESS, amount);
// 等待交易执行完成
await tx.wait();
```

谢谢观看