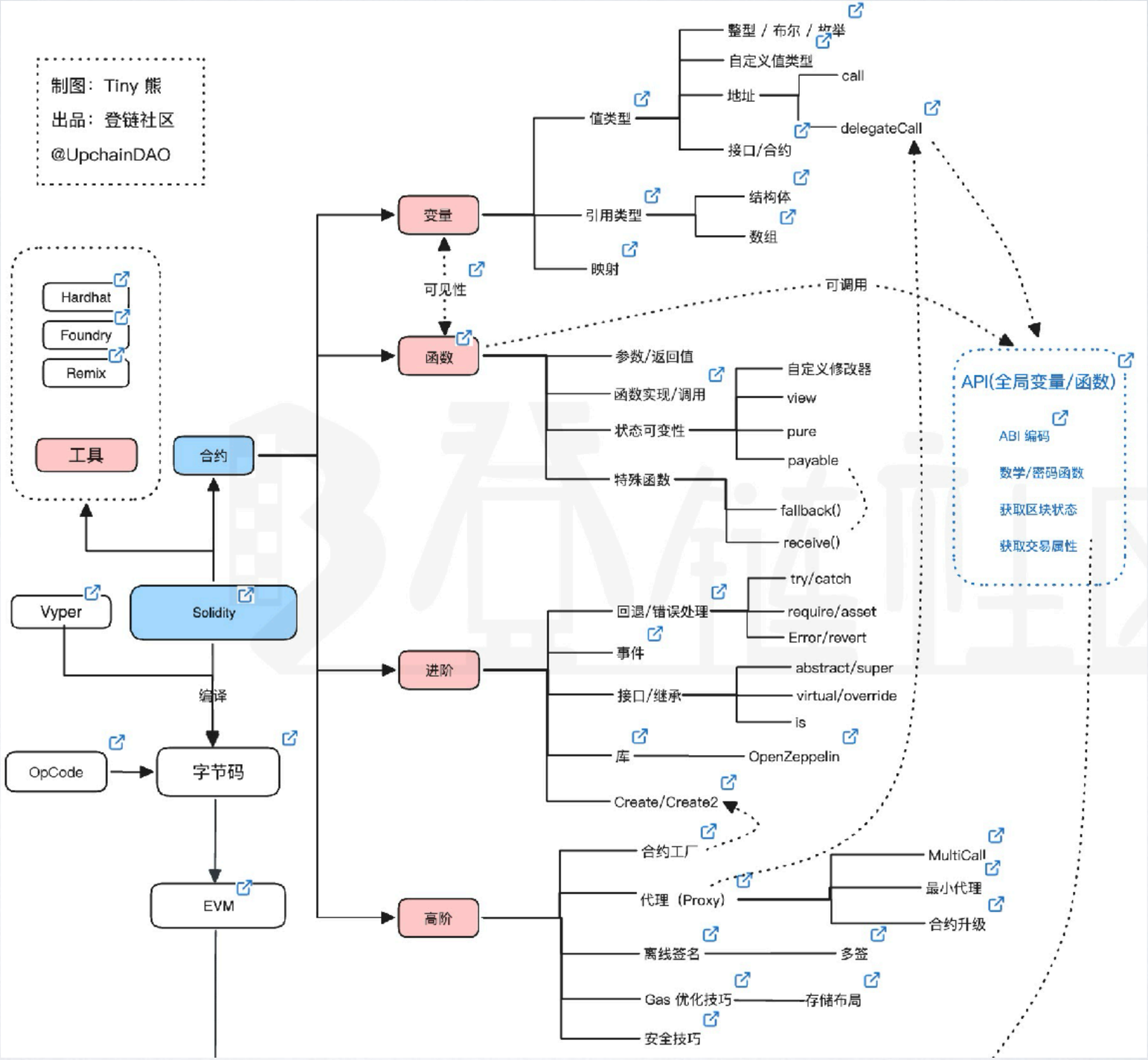
练习题

- 编写一个 BigBank 合约继承自Bank，要求：
  - 仅 > 0.001 ether（用`modifier`权限控制）可以存款
- 把管理员转移给 Ownable 合约，只有 Ownable 可以调用 BigBank 的withdraw().

Solidity 图谱


制图：Tiny 熊
出品：登链社区
@UpchainDAO
Hardhat
Foundry
Remix
工具
合约
Vyper
Solidity
编译
OpCode
字节码
EVM
变量
函数
进阶
高阶
值类型
整型 / 布尔 / 枚举
自定义值类型
地址
call
delegateCall
接口/合约
引用类型
结构体
数组
映射
可见性
可调用
参数/返回值
函数实现/调用
状态可变性
自定义修改器
view
pure
payable
特殊函数
fallback()
receive()
API(全局变量/函数)
ABI 编码
数学/密码函数
获取区块状态
获取交易属性
回退/错误处理
try/catch
require/asset
Error/revert
事件
接口/继承
abstract/super
virtual/override
is
库
OpenZeppelin
Create/Create2
合约工厂
代理（Proxy）
MultiCall
最小代理
合约升级
离线签名
多签
Gas 优化技巧
存储布局
安全技巧

SOLIDITY - ABI 及底层调用

以太坊 三种交易

- 普通转账交易

```
{
   to: ‘0x687422…’,
   value: 0.0005,
   data: "0x" // 也可以附加消息
}
```

- 创建合约

```
{
  to: '',   //  零地址
  value: 0.0,
  data:"0x6060604052341561000 57x106……"
}
```

- 调用合约

```
{
  to: ‘0x687422eEA2cB73..’, //合约地址
  value: 0.3,
  data: “0x06661abd"
}
```

ABI

调用一个合约函数 = 向合约地址发送一个交易

交易的内容就是 ABI 编码数据

ABI

- ABI：Application Binary Interface 应用程序二进制接口
- ABI 接口描述（JSON）： 定义如何与合约交互
- ABI 编码
  - 函数选择器：对函数签名计算 Keccak-256 哈希，取前 4 个字节
  - 参数编码

ABI 描述

ABI编码

Calling a Smart Contract

Address: 0x0123456.....

1 Convert function call to HEX

myFunction(parameters) → → 0xabcdef0123456789.....

2 Put the Information into Transaction object

```
{
  "to": "0x0123456.....",
  "value": 0, // No need to send money here
  "data": "0xabcdef0123456789......"
}
```

3 Sign the Transaction with your Private key

{ ... } + Private Key → → 0xfedcba9876...

Signed Transaction
Can only be decrypted with YOUR public key
Only you can have sent this transaction

4 Send the transaction to the Ethereum Network

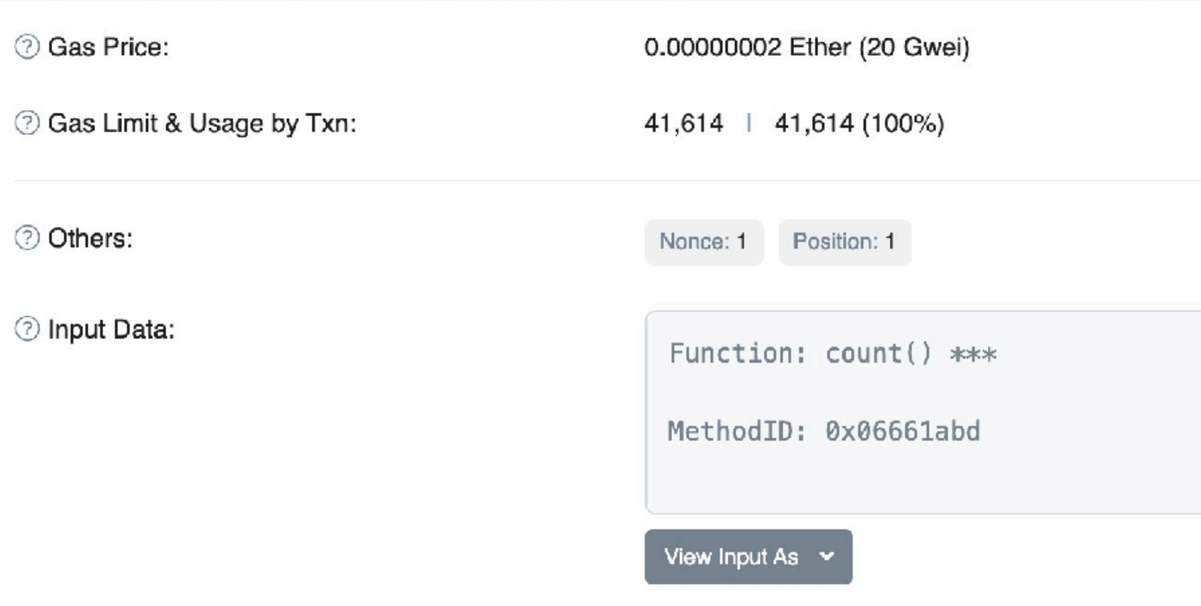
Ethereum

登链社区

ABI

bytes4(keccak256("count()")) = 0x06661abd

Tx Hash：https://sepolia.etherscan.io/tx/0x70d77a2fa95c612d35cd1ca36de797b219b64b5a96ea97ab556c26f1b7259ab8



https://chaintool.tech/calldata

https://www.4byte.directory/

ABI 编码 API： abi.encodeWithSignature(string sigs)

在 EVM 中，最终执行都是字节码，
可否在 Solidity 直接通过调用字节码来调用函数呢?

ADDRESS – CALL

- 底层函数：**call**
  - `<address>.call(bytes memory) returns (bool, bytes memory)`

调用合约的count() 方法：

```
function callCount(Counter c) public {
    c.count();
}
```

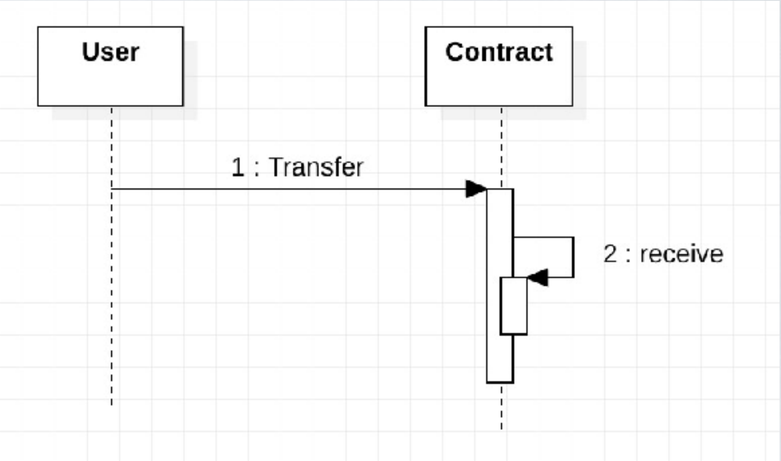
```
function lowCallCount(address c) public {
    bytes memory methodData = abi.encodeWithSignature("count()");
    c.call(methodData);   // 0x06661abd
}
```

testABI.sol

思考：直接 abi 编码的方式调用合约函数， 有什么作用?

ADDRESS – CALL 转账

- `<addr>.transfer()  // 2300`
  - 由于 gas 限制几乎无法向合约转账，怎么办?




```
addr.call{gas: 1000000}(new bytes(0))
```

```
addr.call{value: 1 ether}(new bytes(0))
addr.call{value: 1 ether, gas: 1000000}(new bytes(0))
```

无 gas 限制，或手动指定 gas 限制

功能等价

```
addr.transfer(1 ether)
```

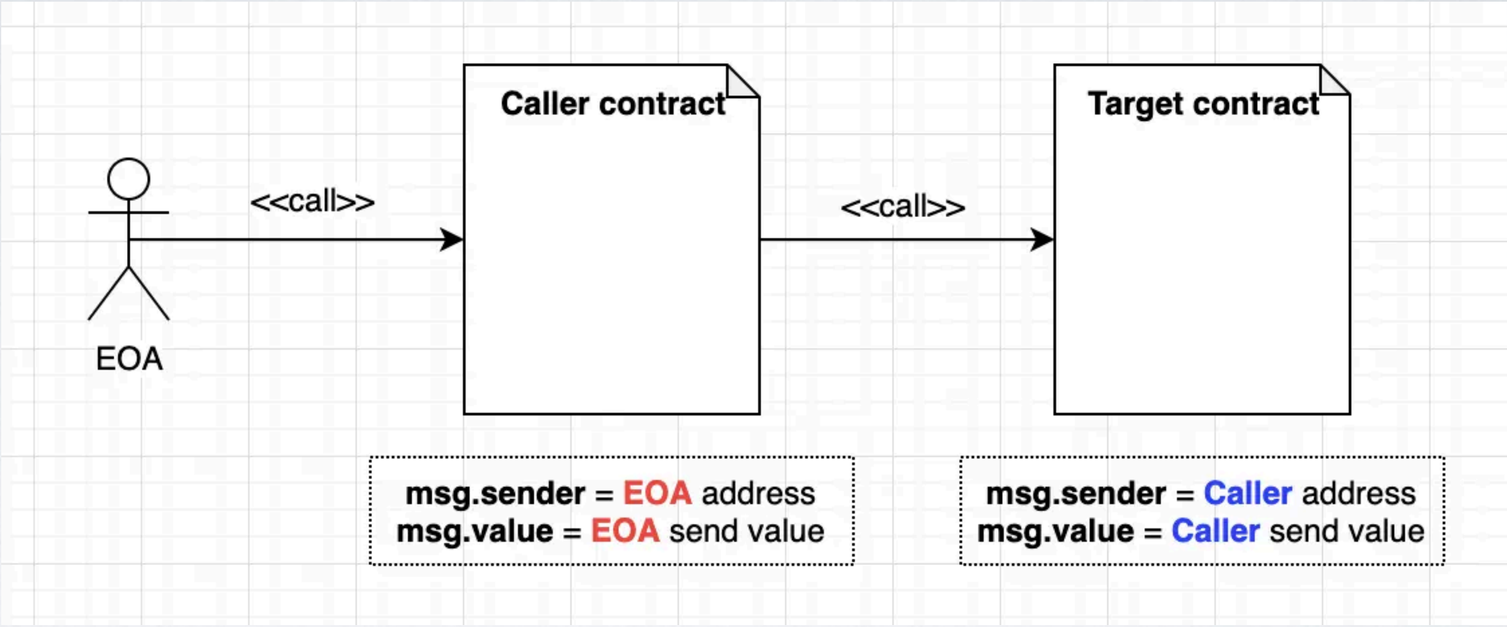
2300 gas 限制

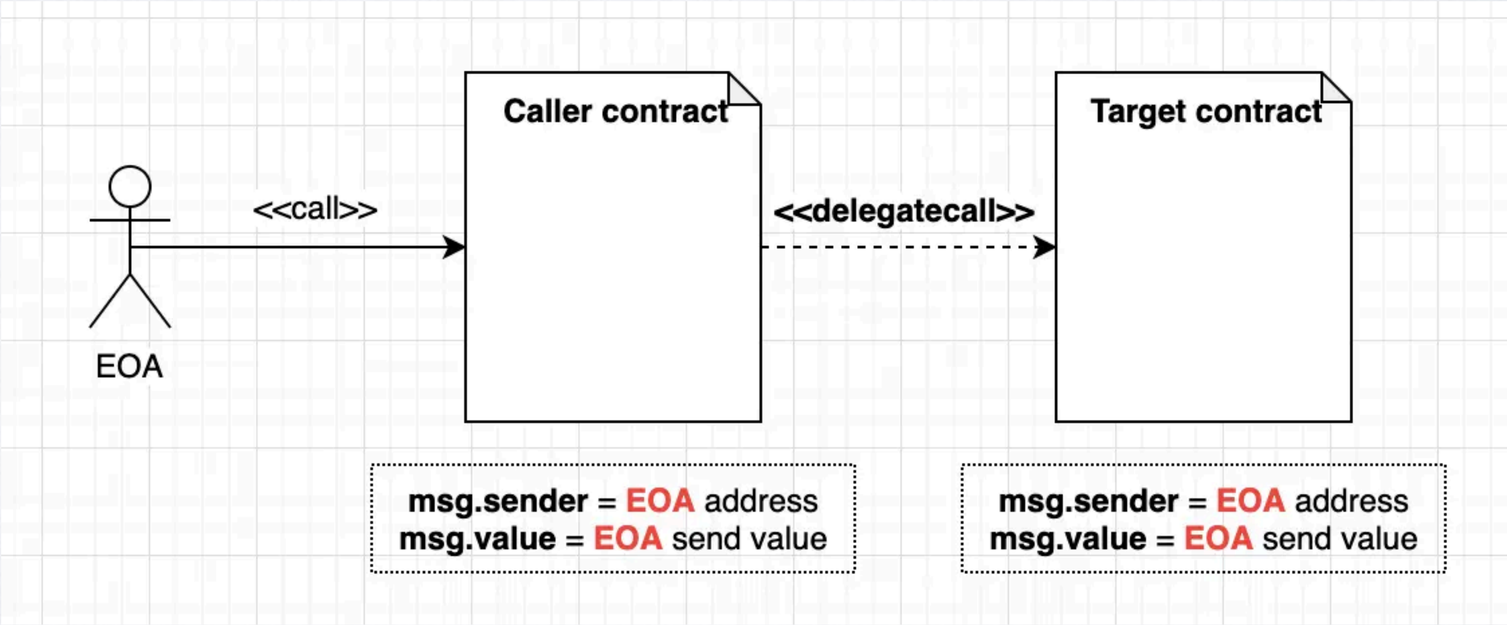
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v5.3.0/contracts/utils/Address.sol

delegatecall

- 另一个底层函数还有：**`delegatecall`**
- call()： 切换上下文
- delegatecall()：保持上下文，借用目标函数的代码在当前的合约

testCall_Delegate.sol





地址 – 小结

- 属性：.balance （ether） / .code / .codehash
- 成员 x.transfer() / send()
  - 2300 gas 限制
- 底层调用：`call, delegatecall, staticcall`
  - 底层调用失败不会冒泡异常（ revert），而是用返回值表示
  - call()：切换上下文
  - delegatecall()：保持上下文
  - staticcall(): 被调用的函数不能修改状态（view）

SOLIDITY – 库（LIBRARY）

- 与合约类似（一个特殊合约），是函数的封装，用于代码复用。 library 定义库
  - abstract 抽象合约是另一个代码复用的方式
- 如果库函数都是 internal 的，库代码会嵌入到合约。
- 如果库函数有 external 或 public ，库需要单独部署，并在部署合约时进行链接，使用委托调用
- 没有状态变量（library）
- 不能给库发送 Ether （library）
- （语法糖）给类型扩展功能：Using lib for type; 如： using SafeMath for uint;

testLib.sol

```
pragma solidity ^0.8.0;

library SafeMath {
    function add(uint x, uint y) internal pure returns (uint) {
        uint z = x + y;
        require(z >= x, "uint overflow");

        return z;
    }
}

contract TestLib {
    using SafeMath for uint;

    function testAdd(uint x, uint y) public pure returns (uint) {

        return x.add(y);  // SafeMath.add(x, y);

    }

}
```

HARDHAT – 链接外部库

```
  const ExLib = await hre.ethers.getContractFactory("Library");
  const lib = await ExLib.deploy();
  await lib.deployed();


await hre.ethers.getContractFactory("TestExLib", {
    libraries: {
      Library: lib.address,
    },
  });
```

FOUNDRY – 链接外部库

```
// 方法 1
> forge create src/MyCT.sol:MyCT –libraries LibXX.sol:LibXX:0x...


// 方法 2
// # foundry.toml
 [profile.default]
     libraries. = “src/libraries/MyLibrary.sol:MyLibrary:0x..."]
```

import

- import 导入代码，有助于代码模块化。
- import 导入第三方的库（lib）及抽象合约，也可提高开发效率，提高安全性
- 相对路径导入， 路径以 ./ ../ 开头
- 直接导入：指定完整的路径 或 HTTP IPFS URL

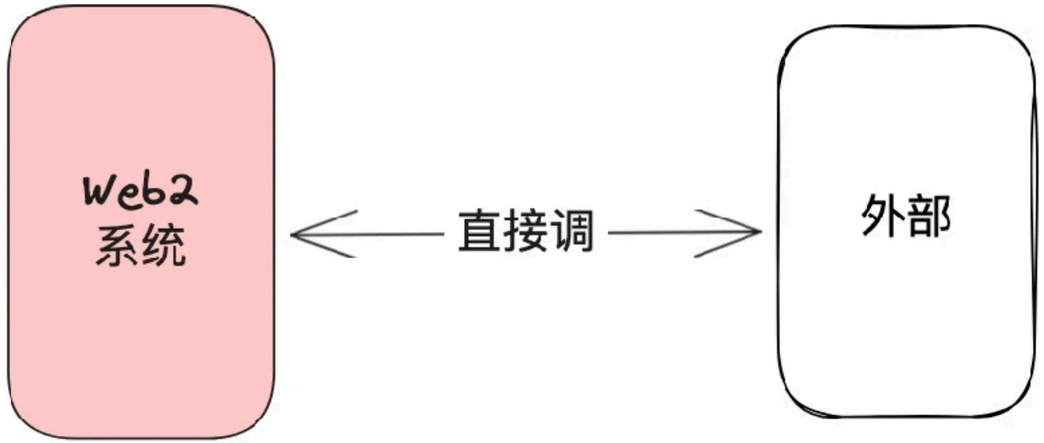
```
import “./filename.sol”; // 将 “filename” 中的所有全局符号导入到当前文件中

// import { Token as T1} from "./token.sol";
import { Token } from "./token.sol";

import { Address } from "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/
master/contracts/utils/Address.sol";
```

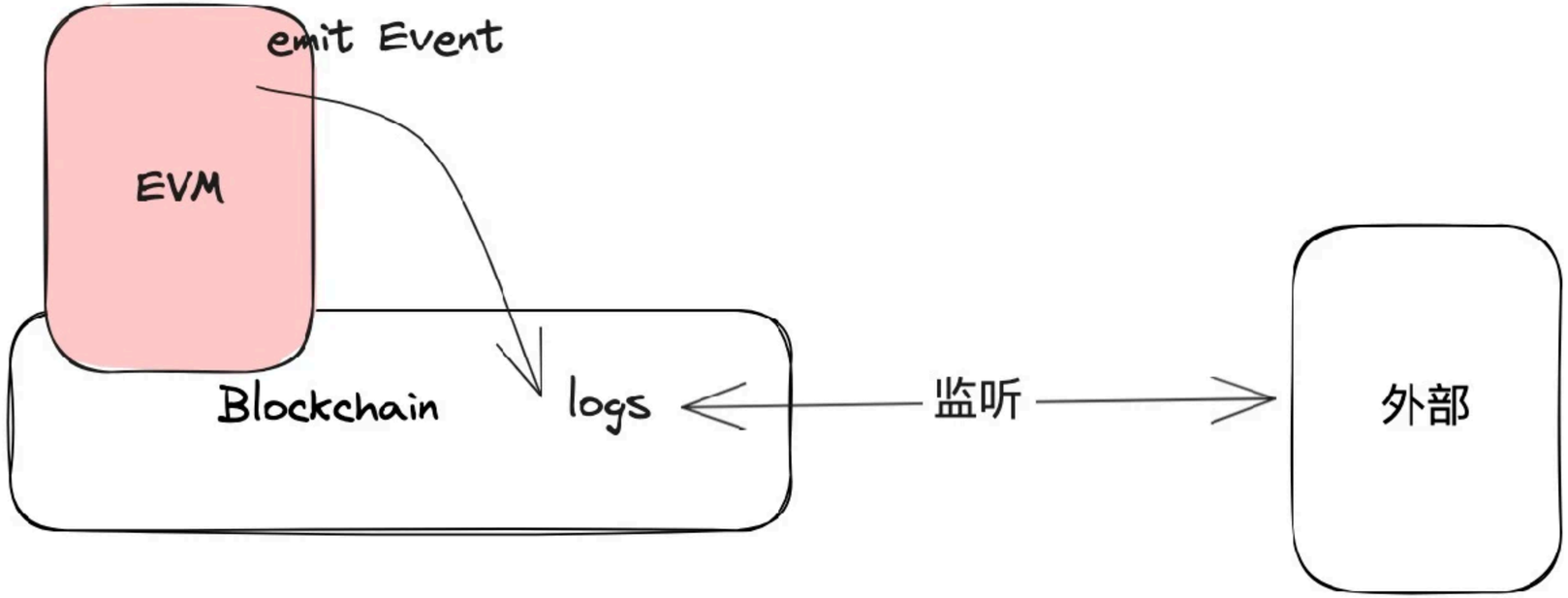
testImport.sol

SOLIDITY – 事件

- 合约与外部世界的重要接口，通知外部世界链上状态的变化
- 事件有时也作为便宜的存储，触发事情会在链上记录一个日志。
- 使用关键字 **event** 定义事件，事件不需要实现
  - 事件中使用 **indexed** 修饰，表示对这个字段建立索引(也称为 Topic)，方便外部对该字段过滤查找
- 使用关键字 **emit** 触发事件

testEvent.sol





DeCert.me

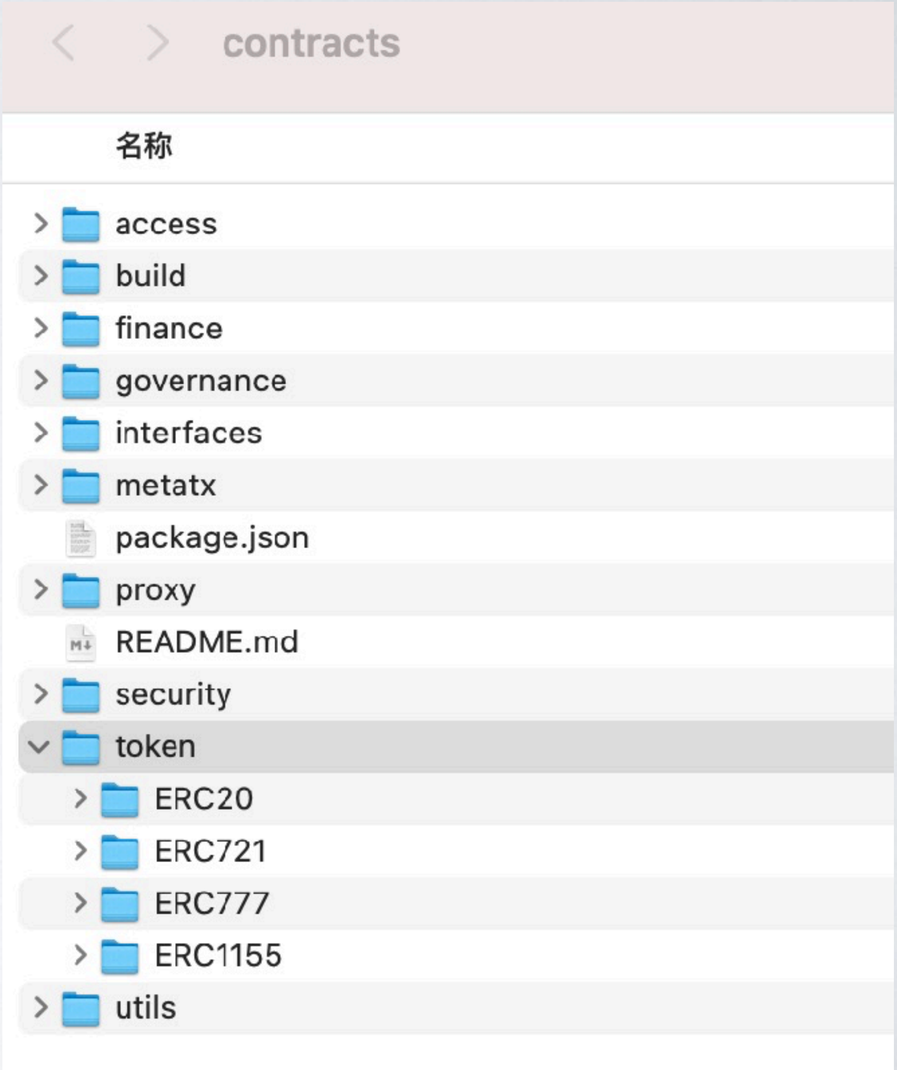
第三方代码库

- 善于复用库（不要重复造轮子），不仅提高开发效率，还可以提高安全性
  - 通常遵循最佳实践编写的、经过社区反复审计与验证
- 两个优秀的代码库：OpenZeppelin 、Solady （前 Solmate）
- OpenZeppelin 采用率高、功能丰富
  - 代码：https://github.com/OpenZeppelin/openzeppelin-contracts
- solady： 高度 gas 优化
  - https://github.com/vectorized/Solady

OpenZeppelin

- Utils：实现一些工具库，如判断是否为合约地址、数学函数等。
- access: 合约函数访问控制功能
- token：包含 ERC20、ERC721、ERC1155 代币
- proxy：升级代理
- ...

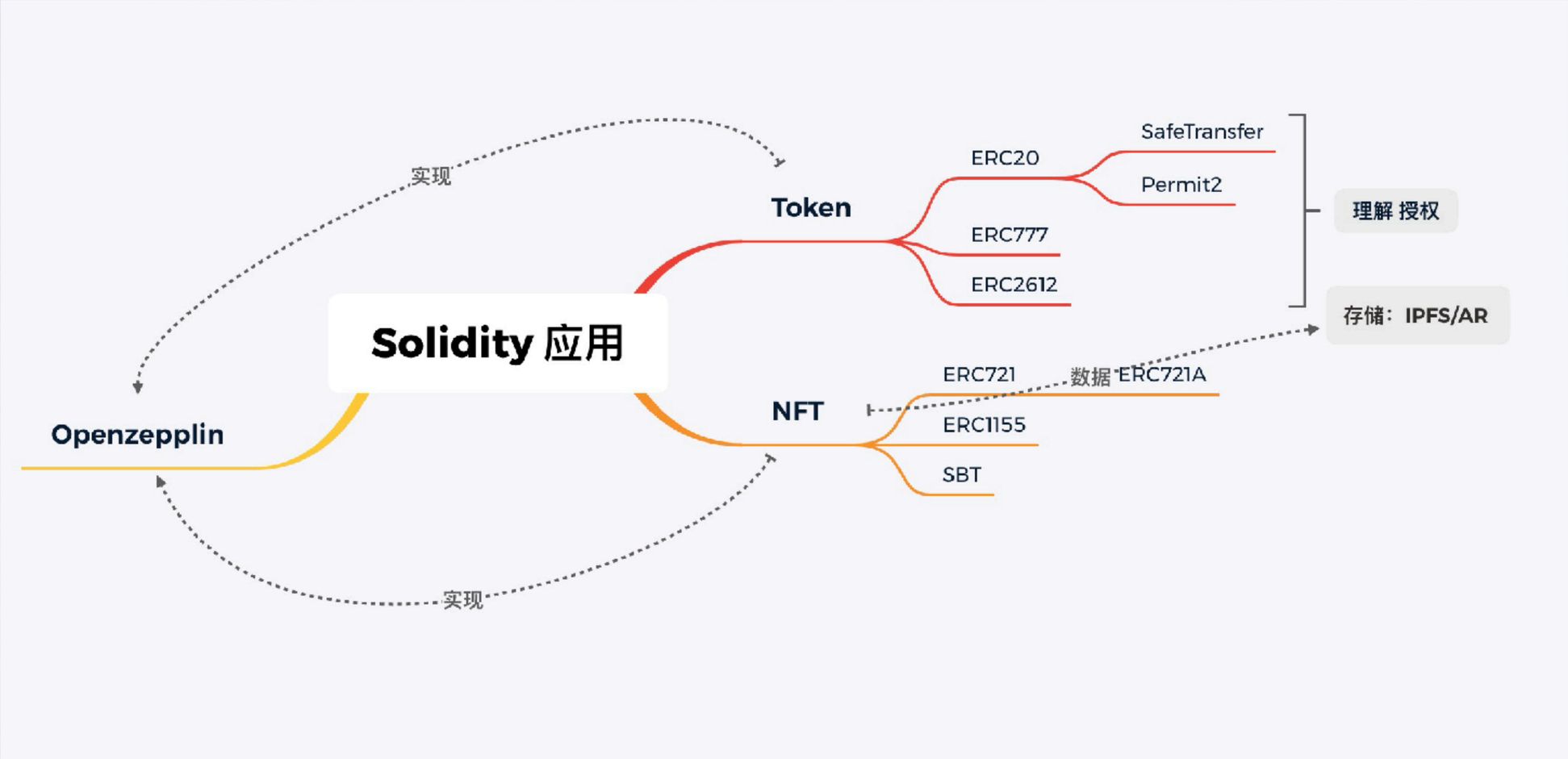
https://github.com/OpenZeppelin/openzeppelin-contracts



推荐SOLIDITY学习资料

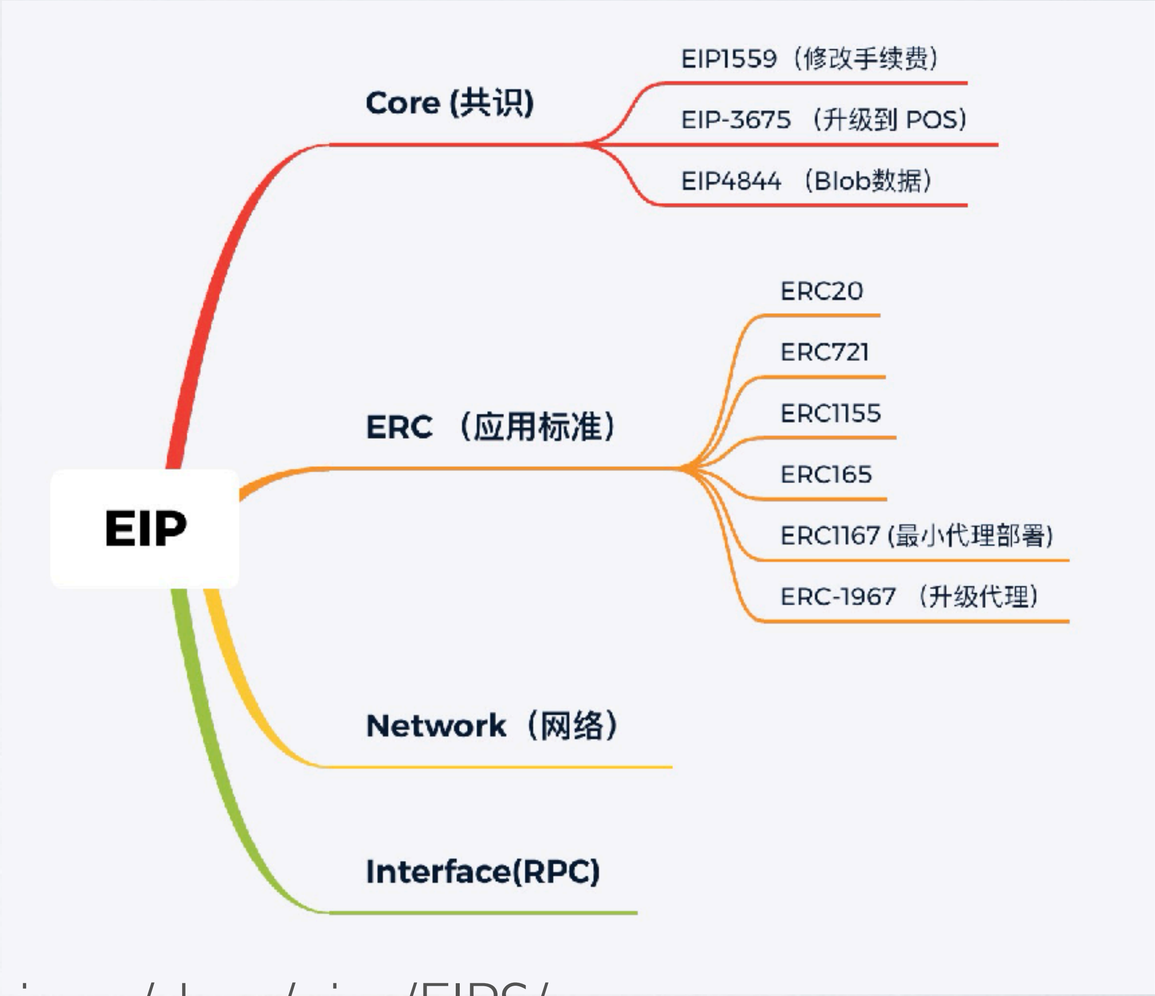
- https://ethernaut.openzeppelin.com/
- https://cryptozombies.io/en/course/
- https://solidity-by-example.org/
- https://learnblockchain.cn/column/1
- https://decert.me/tutorial/solidity/intro/

SOLIDITY 应用


Solidity 应用
Token
ERC20
SafeTransfer
Permit2
ERC777
ERC2612
理解 授权
存储：IPFS/AR
NFT
ERC721
数据 ERC721A
ERC1155
SBT
Openzepplin
实现
实现

EIP 提案 / ERC 标准

- EIP: Ethereum Improvement Proposal 以太坊改进提案
  - 提交改进 Issue、社区的讨论（实现及验证）、形成共识
    - ERC20：https://github.com/ethereum/EIPs/issues/20
    - EIP1599：https://github.com/ethereum/EIPs/issues/1599
- 应用标准：
  - 降低沟通协作成本、增强互操作性



所有提案：https://learnblockchain.cn/docs/eips/EIPS/

ERC20

- 什么是 ERC20 代币
  - 最常用的代币标准，使去中心化交易所、钱包、支付系统无缝对接，繁荣了生态
  - 注意与以太币不同：以太币（Coin）原生币， ERC20 （Token） 智能合约代币 (WETH)

https://etherscan.io/tokens

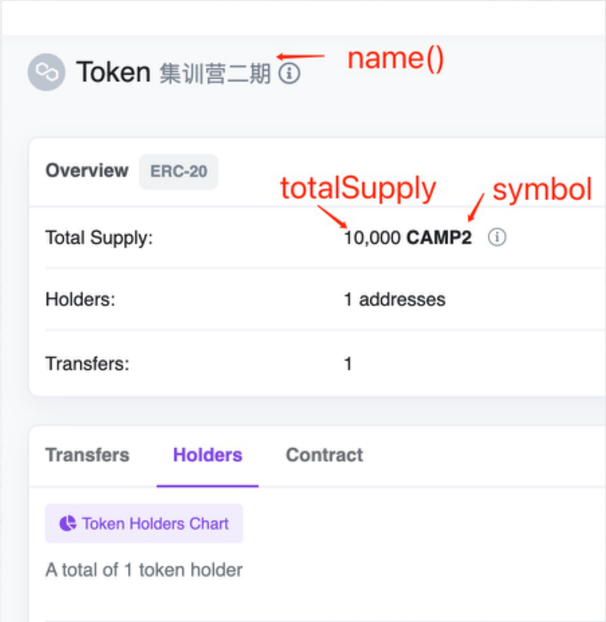
Token Tracker (ERC-20)

A total of **1,202** Token Contracts found

#	Token	Price	Change (%)	Volume (24H)	Circulating Market Cap	On-Chain Market Cap	Holders
1	Tether USD (USDT)	$1.005 0.000556 ETH	▲ 0.27%	$135,858,721,142.00	$76,586,960,300.00	$40,022,235,444.23	4,155,218 -0.114%
2	BNB (BNB)	$344.4714 0.190434 ETH	▲ 4.46%	$919,680,425.00	$54,389,100,777.00	$5,711,169,358.05	302,792 0.000%
3	USD Coin (USDC)	$1.004 0.000555 ETH	▲ 0.28%	$6,433,192,130.00	$36,403,349,824.00	$46,788,840,563.36	1,604,554 0.017%
4	HEX (HEX)	$0.0907 0.000050 ETH	▲ 7.00%	$14,315,767.00	$15,726,734,160.00	$52,379,897,495.38	324,190 0.014%
5	Matic Token (MATIC)	$1.2386 0.000685 ETH	▲ 5.88%	$585,271,356.00	$10,818,524,481.00	$12,386,227,571.86	587,835 0.011%
6	stETH (stETH)	$1,820.12 1.006214 ETH	▲ 6.60%	$75,767,282.00	$10,579,155,794.00	$3,373,790,813.08	172,482 0.016%
7	Binance USD (BUSD)	$1.006 0.000556 ETH	▲ 0.73%	$10,355,164,133.00	$8,279,435,781.00	$17,680,545,374.84	179,210 -0.001%
8	SHIBA INU (SHIB)	$0.00 0.000000 ETH	▲ 5.33%	$383,373,693.00	$6,688,388,789.00	$11,339,907,394.11	1,325,161 0.006%

ERC20

- ERC20 可以表示任何同质的可以交易的内容：
  - 货币、股票、积分、债券、利息…
  - 可以用数量来表示的内容 基本上可以 ERC20 表示
- ERC20 标准包含哪些内容
  - 定义统一的函数名：名称、发行量、转账函数、转账事件等
  - 以便交易所、钱包进行集成
- 所有实现了这些函数的合约都是 ERC20 Token



https://etherscan.io/token/0xdac17f958d2ee523a2206206994597c13d831ec7

代币关键点

- 如何表达每个人持有的数量?

```
contract Token {
    mapping(address => uint256) holds;

    constructor(uint total)  {
        holds[msg.sender] = total;
    }

    function sendTo(address to, uint x) public  {
        holds[msg.sender] -= x;
        holds[to] += x;
    }

}
```

token.sol

ERC20

```
interface IERC20 {

    function name() external view returns (string memory);
    function symbol() external view returns (string memory);
    function decimals() external view returns (uint8);  // 小数位数

    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);

    function transfer(address to, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address from, address to, uint256 amount) external returns (bool);

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}
```

ERC20 实现

- OpenZeppelin 实现了众多 ERC 标准，通常只需要扩展一下。
  - https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

发行 10000 token，名称为 CAMP

```
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract MyERC20 is ERC20 {

    constructor() ERC20(“OpenSpace BootCamp”, "CAMP") {
        _mint(msg.sender, 10000*10**18);
    }
}
```

合约不支持浮点数，如何用 uint 表示小数?

对比 ETH / Token 转账

向 0xabcd… 转移 10 ETH， Tx:

```
{
   to: '0xabcd…',
   value: 10 * 10^18,
   data: "0x" // 也可以附加消息
}
```

如果 ETH 接收者是合约地址，receive() 会被调用

向 0xabcd… 转移 10 Token， Tx:

```
{
  to: '0x6874..', // Token 合约地址
  value: 0,
  data: "transfer(0xabcd…,10*10^18) ABI编码"
}
```

不管接收者是谁，都只有 Token 合约代码被执行

持有的余额是记录在 Token 合约内，不记录在用户地址下

练习

- ABI 编码：
  - https://decert.me/quests/10c11aa7-2ccd-4bcc-8ccd-56b51f0c12b8
- 底层调用：
  - https://decert.me/quests/5849ac2d-7a6f-4c94-978c-73c582a575dd

实践练习

- 自己写一个Token，并部署（ERC20）
- 编写一个 TokenBank ， 可以将 Token 存入 TokenBank:
  - 记录每个用户存入的 token 数量
  - 管理员可以提取所有的Token （withdraw 方法）。

https://decert.me/challenge/aa45f136-27a3-4bc9-b4f7-15308e1e0daa

https://decert.me/quests/eeb9f7d8-6fd0-4c38-b09c-75a29bd53af3

合约地址如何接收 Token ？

- 在 ERC20 转账到合约地址（假设为 TokenBank ）时，TokenBank合约代码无法执行，TokenBank 如何记录用户存了多少 Token ？

练习题

Foundry 环境安装