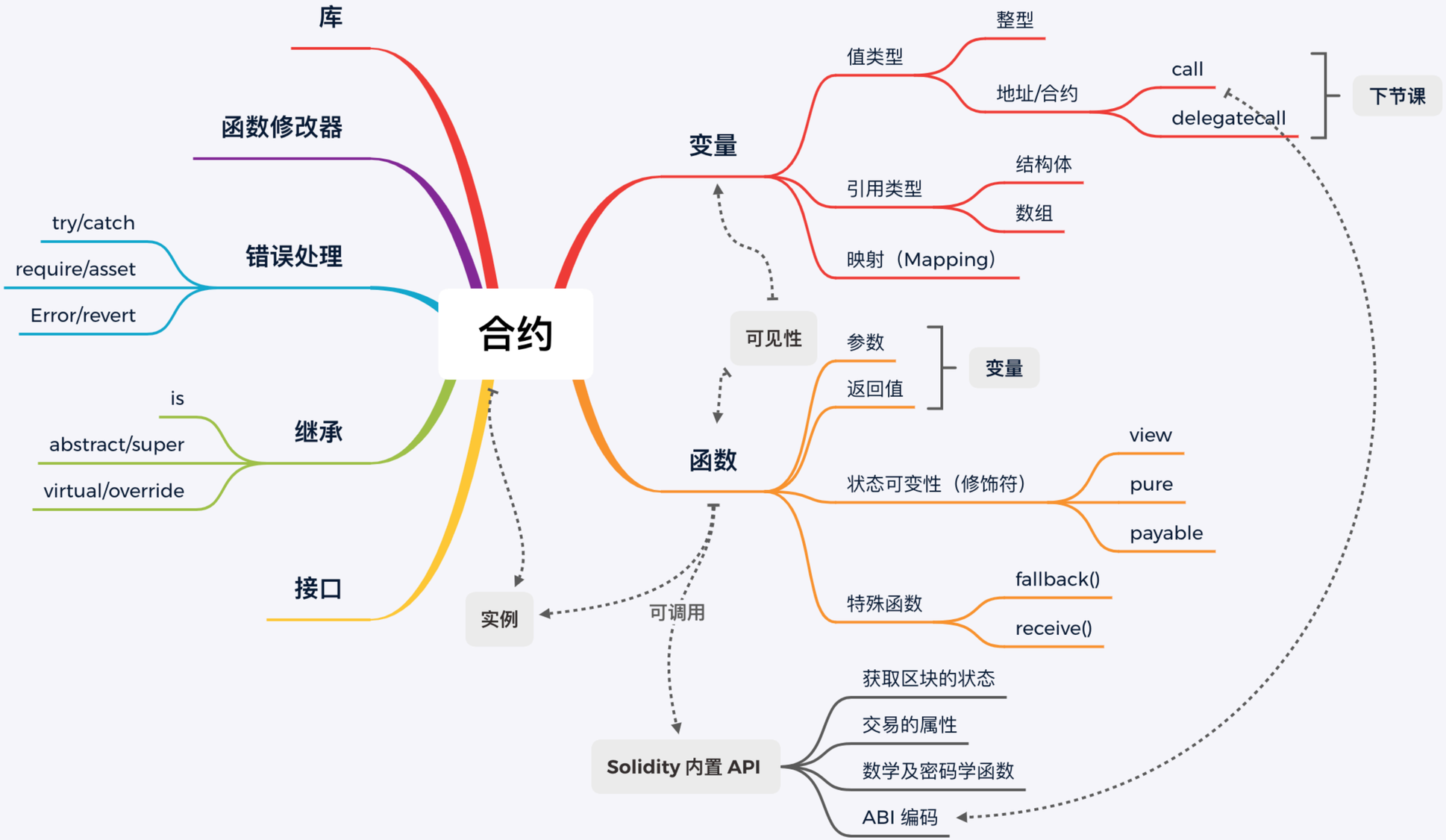

合约
库
函数修改器
错误处理
try/catch
require/asset
Error/revert
继承
is
abstract/super
virtual/override
接口
变量
值类型
整型
地址/合约
call
delegatecall
下节课
引用类型
结构体
数组
映射（Mapping）
可见性
参数
返回值
变量
函数
状态可变性（修饰符）
view
pure
payable
特殊函数
fallback()
receive()
实例
可调用
Solidity 内置 API
获取区块的状态
交易的属性
数学及密码学函数
ABI 编码

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().

地址 – CALL 转账

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

底层调用不受gas 限制

```
addr.call{value: 1 ether}(new bytes(0)) = addr.transfer(1 ether)
```

https://github.com/Uniswap/solidity-lib/blob/master/contracts/libraries/TransferHelper.sol

DAY5： SOLIDITY

登链社区 - Tiny熊

ABI

- ABI：Application Binary Interface 应用程序二进制接口
- ABI 接口描述： 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

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

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

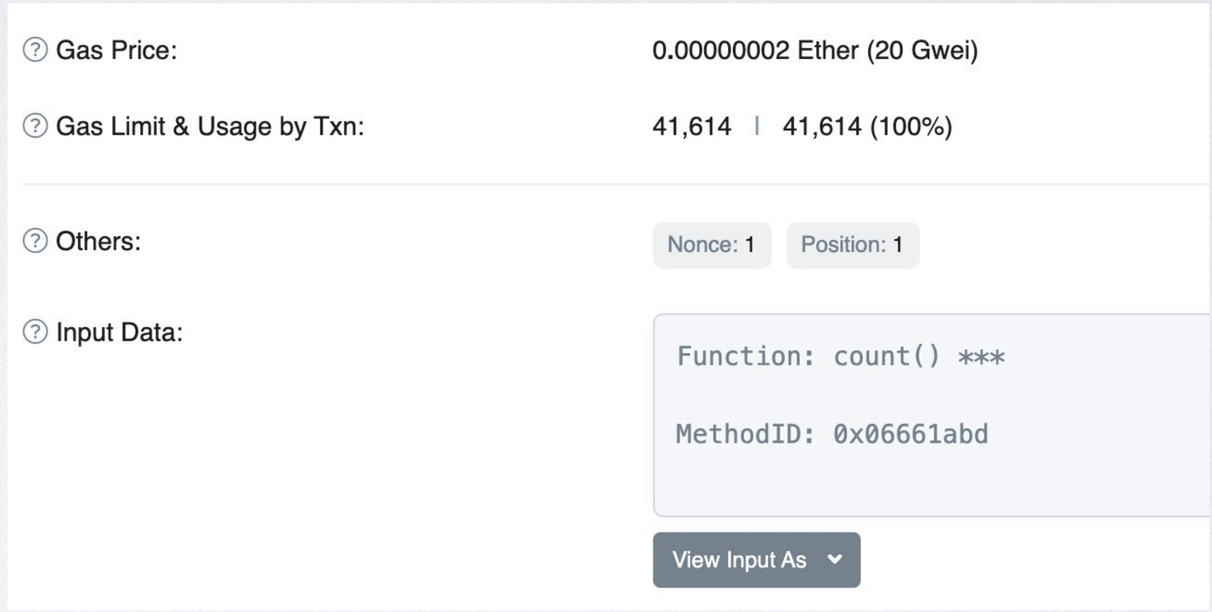
Ethereum

登链社区

ABI

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

Tx Hash：https://
mumbai.polygonscan.com/tx/
0x6d45ded0fa162200c9b589ded766978
30ea69652f5235d66ff671932d4aee51d



https://chaintool.tech/calldata

https://www.4byte.directory/

ABI 编码 API： abi.encodeWithSignature(string sigs)

地址类型 - 底层调用

- 底层函数：**call**，**delegatecall**，staticcall（不修改状态）
  - <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

直接abi编码的方式调用合约函数， 有什么作用?

地址 – 底层调用

- 底层调用失败不是发生异常（revert），而是用返回值表示
- call()：切换上下文
- delegatecall()：保持上下文

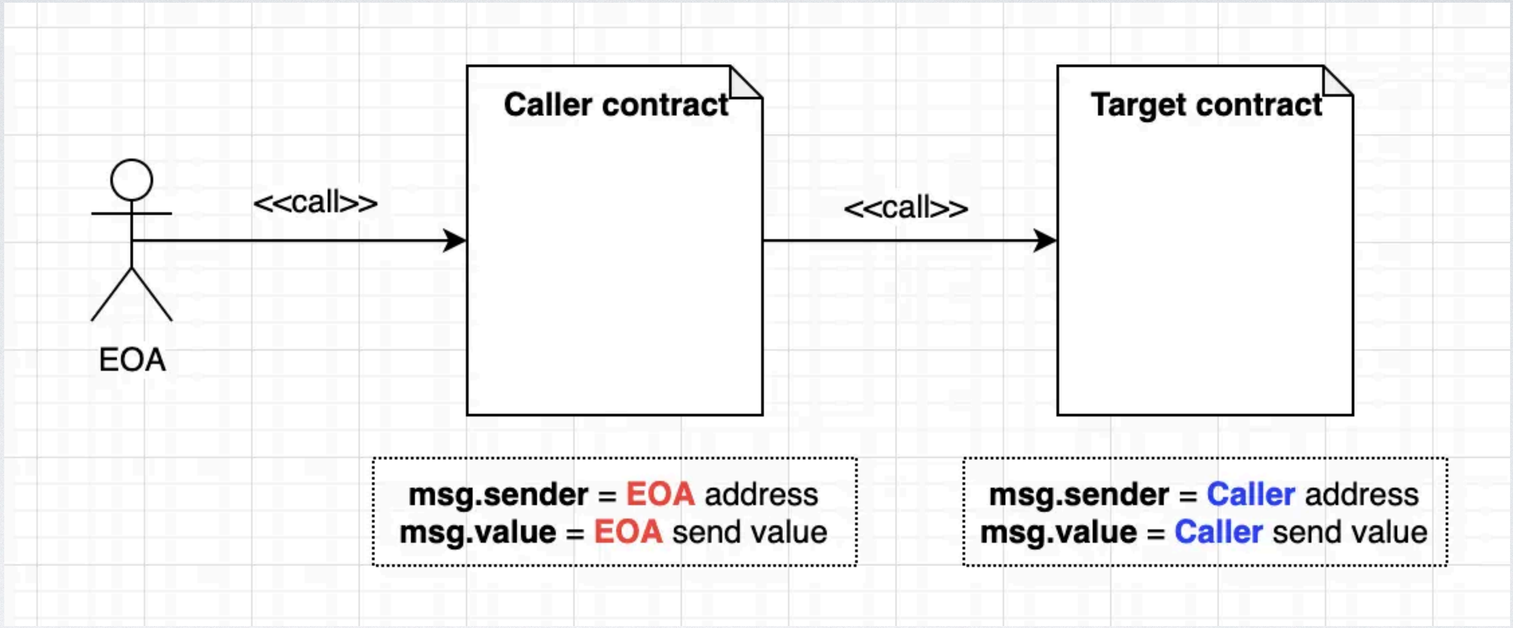
地址 – CALL 转账

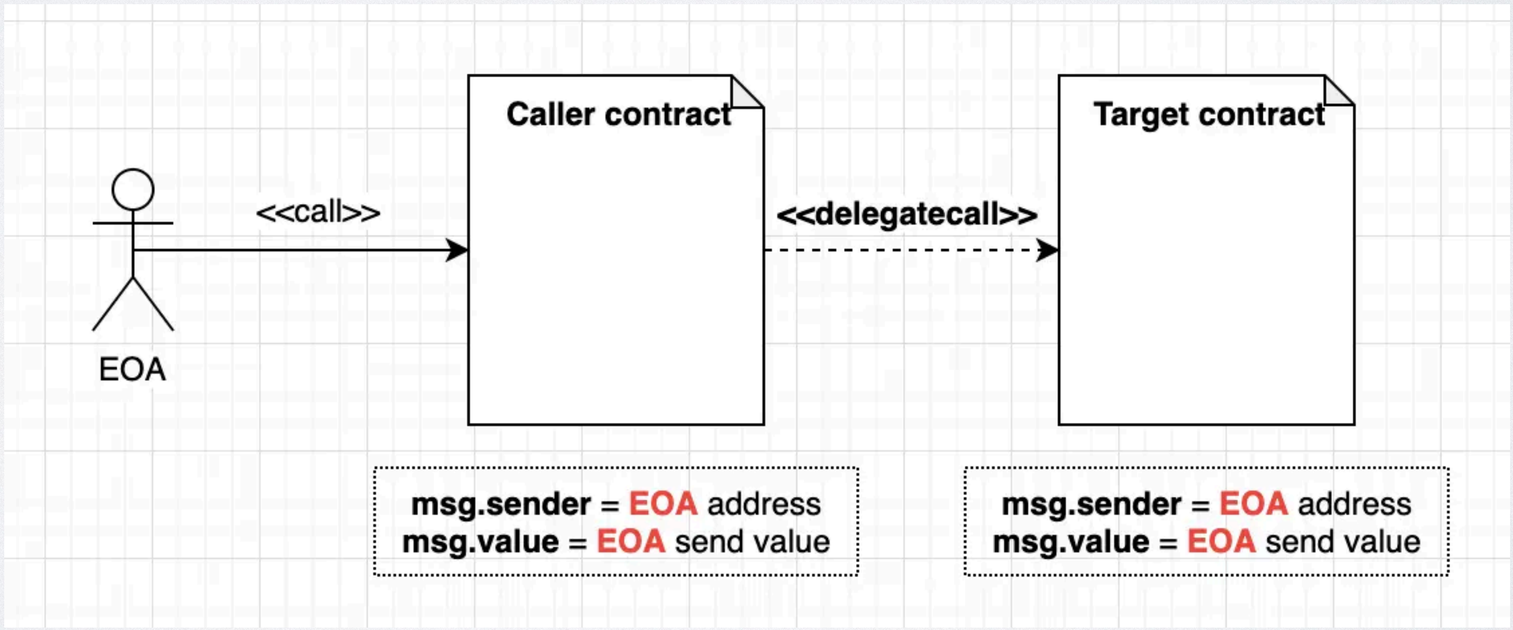
- `<addr>.transfer()`
  - gas 限制几乎无法向合约转账，怎么办?

```
addr.call{value: 1 ether}(new bytes(0))   =   addr.transfer(1 ether)
```

https://github.com/Uniswap/solidity-lib/blob/master/contracts/libraries/TransferHelper.sol

testCall_Delegate.sol

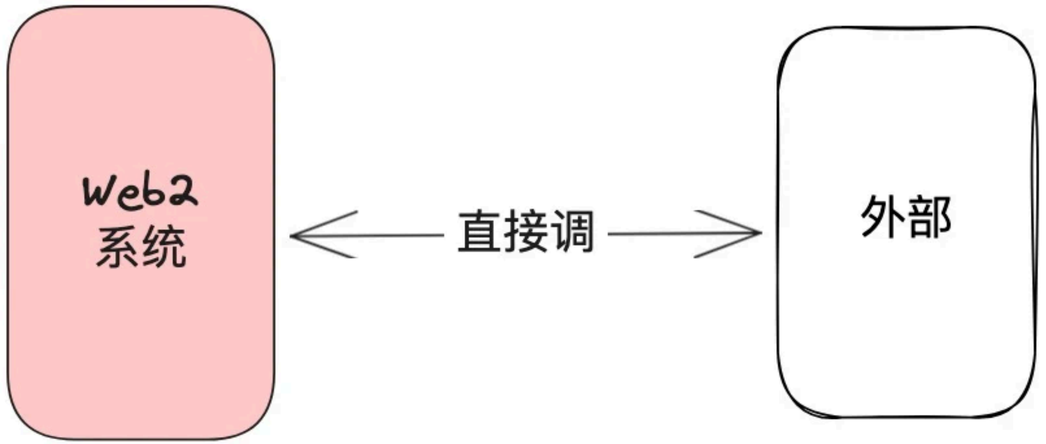


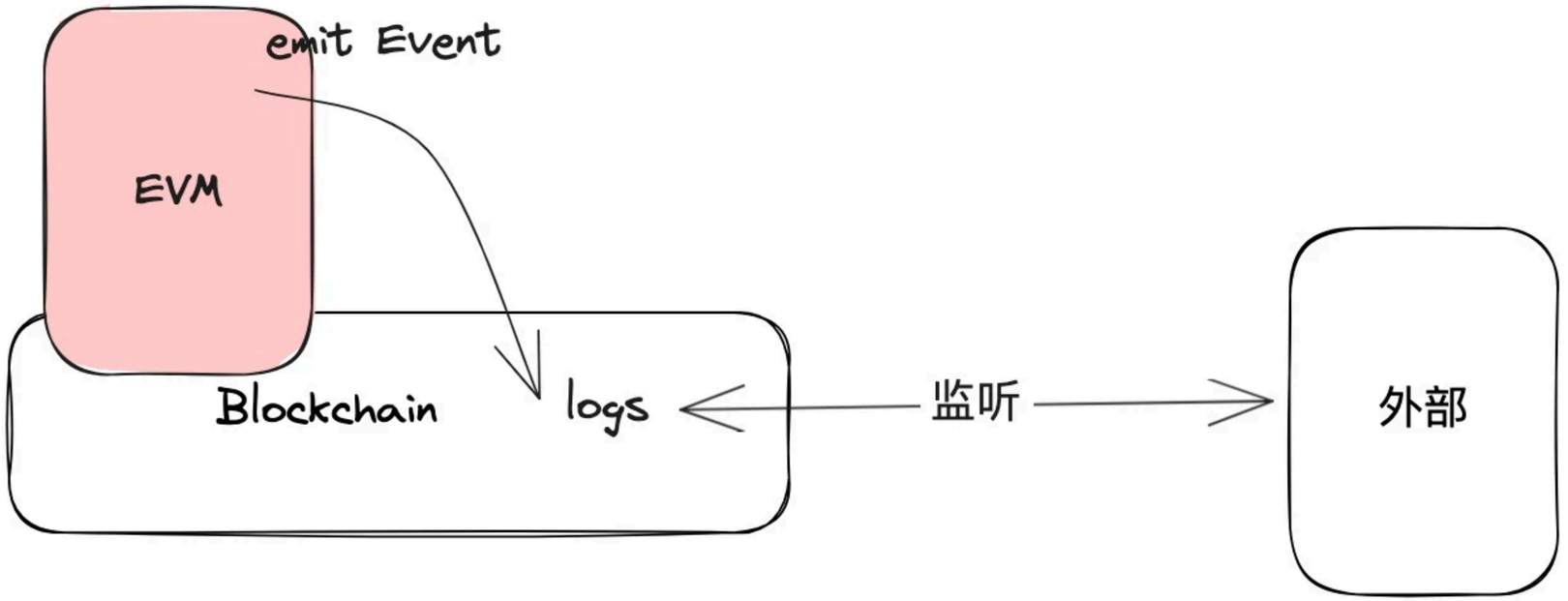


SOLIDITY – 事件

- 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
- 使用关键字 **event** 定义事件，事件不需要实现
- 使用关键字 **emit** 触发事件
- 事件中使用 **indexed** 修饰，表示对这个字段建立索引，方便外部对该字段过滤查找

testEvent.sol


Web2
系统
直接调
外部


emit Event
EVM
Blockchain
logs
监听
外部

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
- 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）
- 给类型扩展功能：Using lib for type; 如： using SafeMath for uint;

- abstract 代码库

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);

    }

}
```

SOLIDITY – 链接外部库

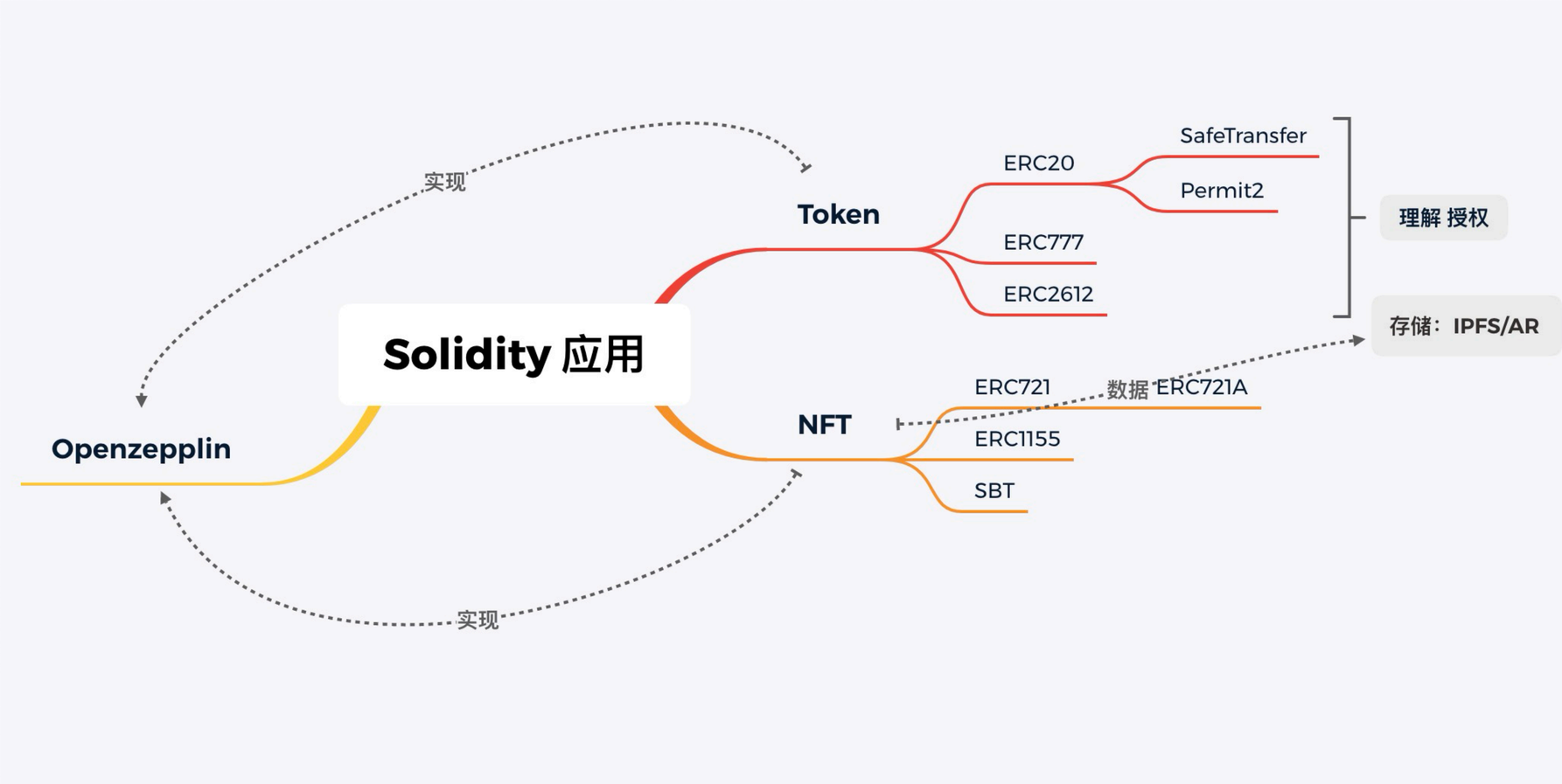
```
  const ExLib = await hre.ethers.getContractFactory("Library");
  const lib = await ExLib.deploy();
  await lib.deployed();


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

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/

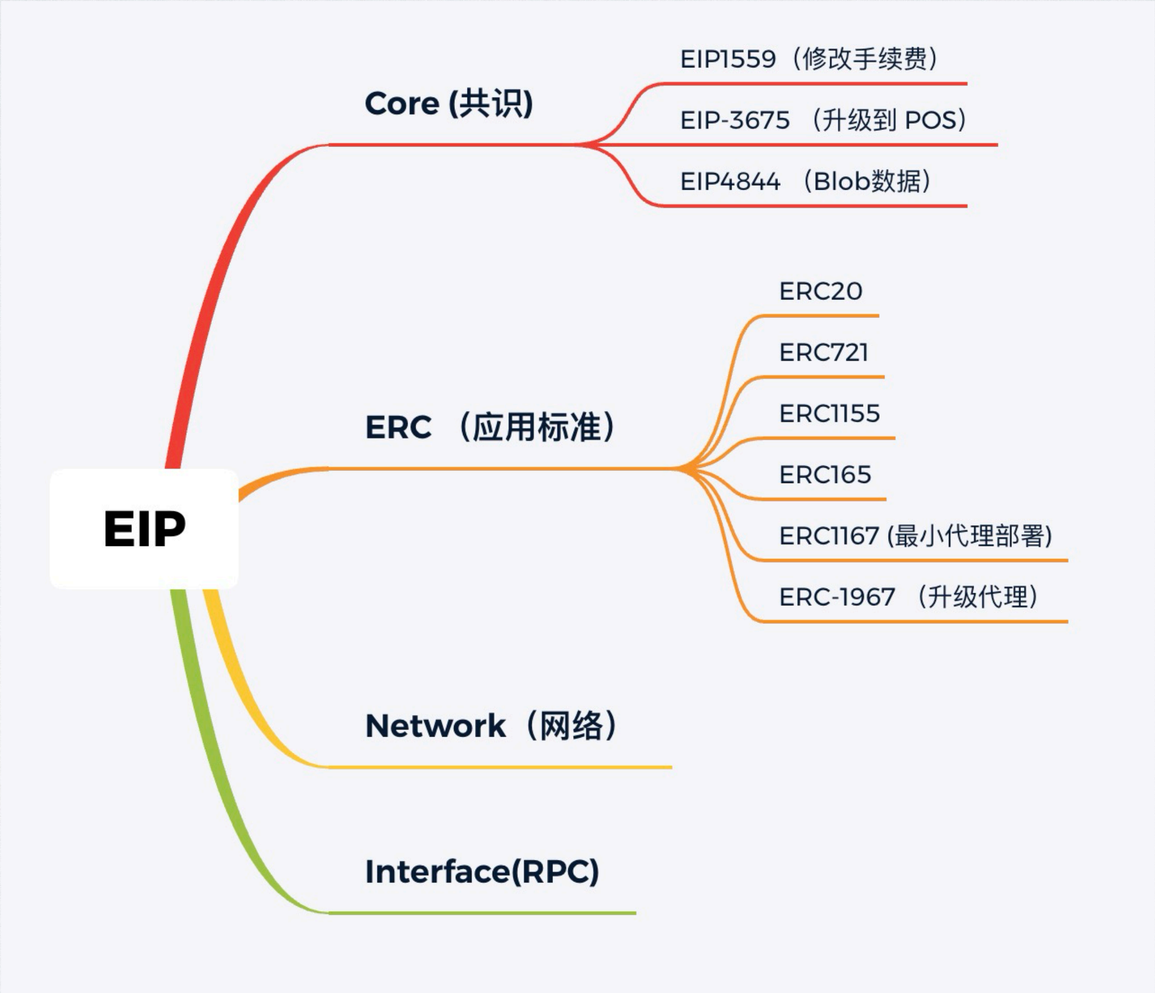
SOLIDITY应用


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

EIP / ERC 标准

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

EIP 分类

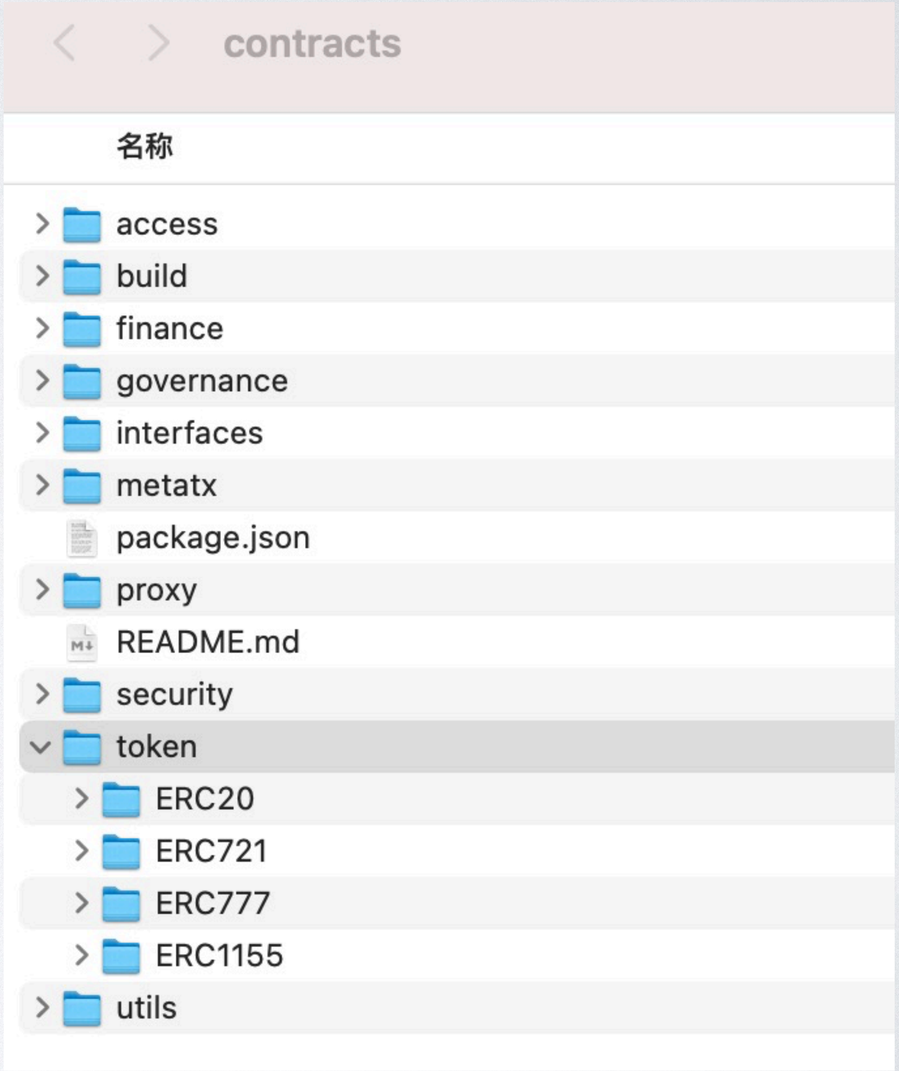

EIP
Core (共识)
EIP1559（修改手续费）
EIP-3675 （升级到 POS）
EIP4844 （Blob数据）
ERC （应用标准）
ERC20
ERC721
ERC1155
ERC165
ERC1167 (最小代理部署)
ERC-1967 （升级代理）
Network（网络）
Interface(RPC)

SOLIDITY – OpenZeppelin

- OpenZeppelin 功能丰富：实现了众多 ERC 标准。
  - 经过社区反复审计与验证
  - 最佳实践
- 善于复用库，不仅提高开发效率，还可以提高安全性
- 文档：https://docs.openzeppelin.com/contracts/4.x/

OpenZeppelin

- 代码库：https://github.com/OpenZeppelin/openzeppelin-contracts
- 安装(Hardhat)： npm install @openzeppelin/contracts —save-dev
- 安装(Foundry): forge install OpenZeppelin/openzeppelin-contracts

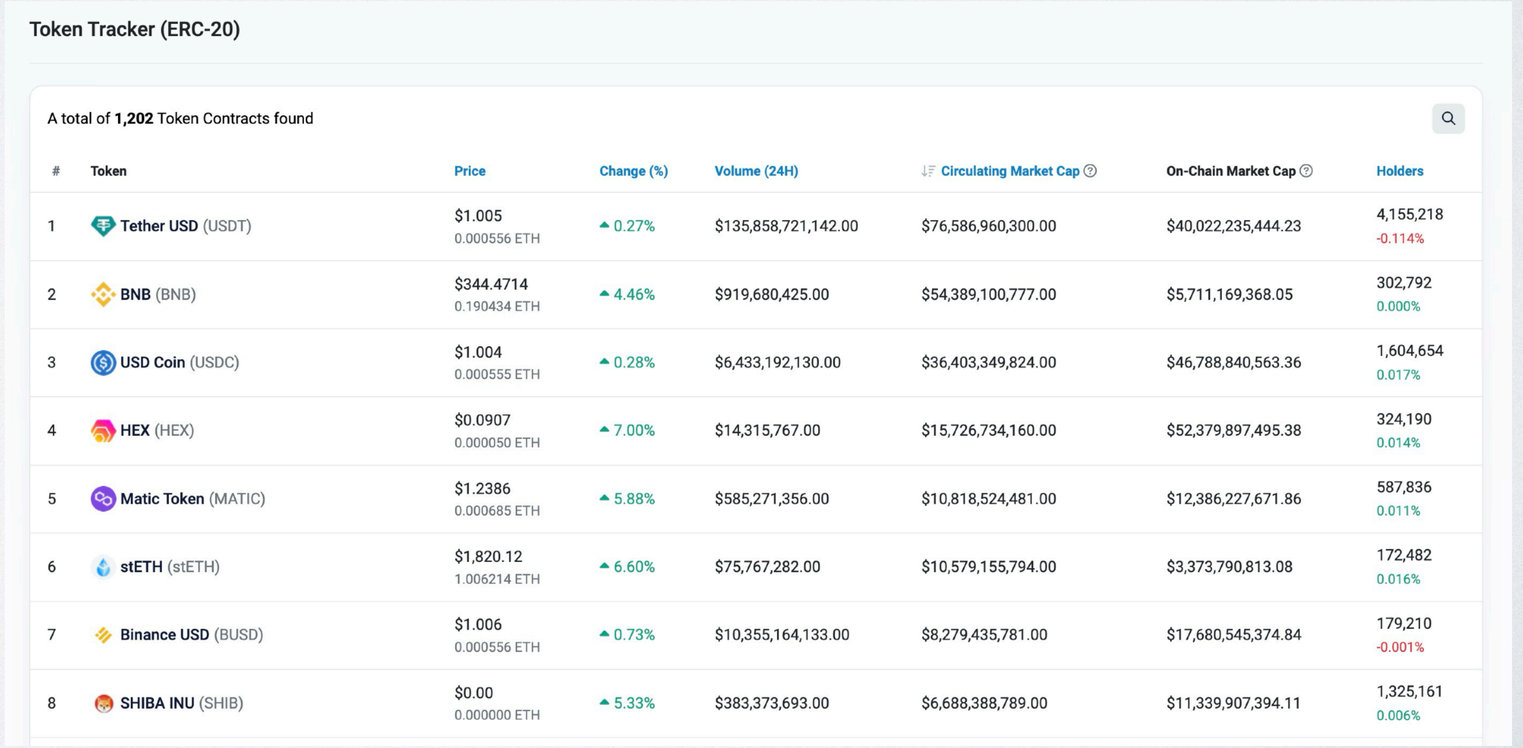


- token：包含 `ERC20`、`ERC721`、`ERC777`、`ERC1155` 代币
- access: 合约函数访问控制功能
- utils：实现一些工具库，如判断是否为合约地址、数学函数等。
- proxy：升级代理

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



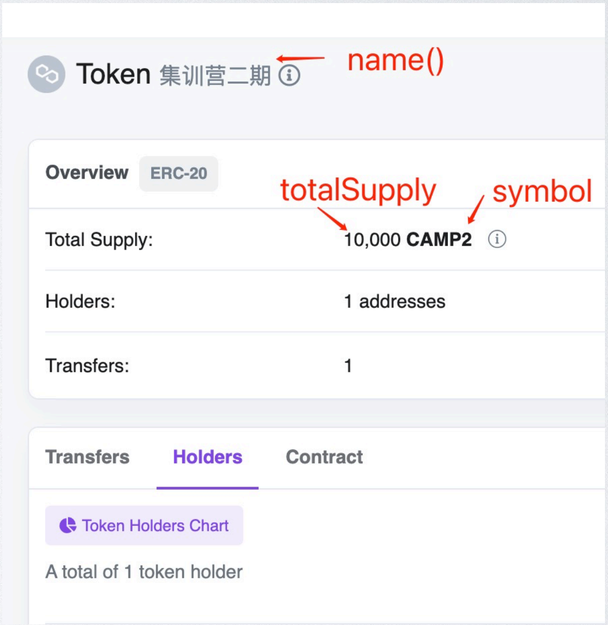
Token Tracker (ERC-20)

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	$5,711,169,368.05	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	1,604,654 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	$12,386,227,671.86	587,836 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
- 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



https://mumbai.polygonscan.com/token/0x21b4d1f6d42dc6083db848d42aa4b689537 1e1e7

ERC20 关键点

- 如何表达每个人持有的数量?

```
contract ERC20 {
    mapping(address => uint256) balanceOf;
}
```

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

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

contract MyERC20 is ERC20 {
    constructor() ERC20(“OpenSpace BootCamp”, "CAMP") {
        _mint(msg.sender, 10000*10**18);
    }
}
```

问题：如何用 uint 表示小数?

练习题

- 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