区块链集训营
二期

登链社区 - Tiny熊

练习题

- 编写合约Score， 用于记录学生（地址）分数：
  - 仅有老师（用`modifier`权限控制）可以添加和修改学生分数
  - 分数不可以大于 100；
- 编写合约 Teacher 作为老师，通过 IScore 接口调用修改学生分数。

习题解答

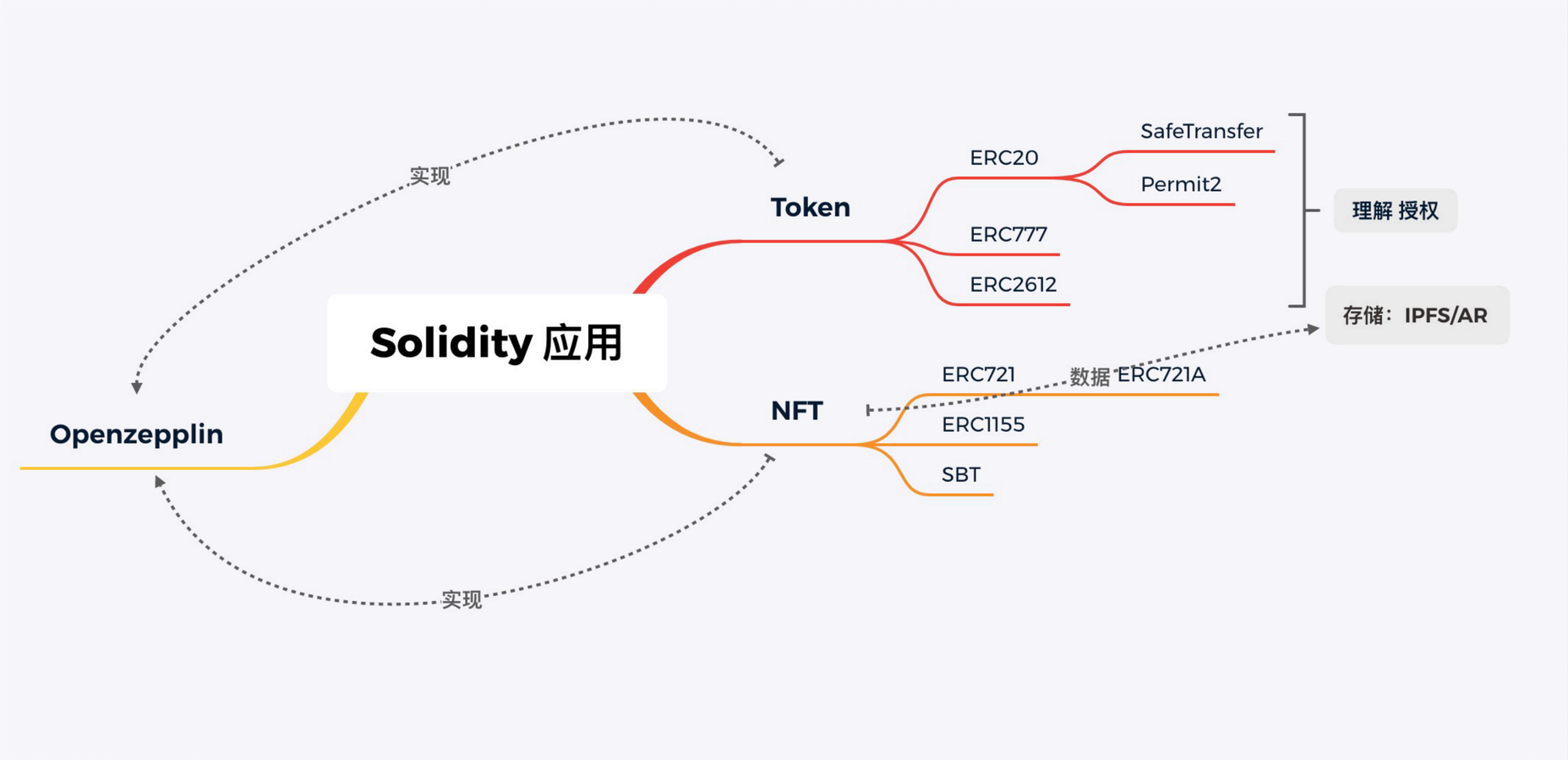
https://github.com/xilibi2003/training_camp_2

课件对应代码在 main 分支
习题解答在 Answer 分支

SOLIDITY – 开发进阶

- ERC 规范（EIP）
- Openzepplin 库 - 通用实现
- Token 标准：**ERC20、ERC777、EIP2612、ERC721、ERC1155**
  - 重点理解授权相关概念 **– approve/permit/permit2**
- 合约代理、合约升级、 Multicall
- Gas 技巧

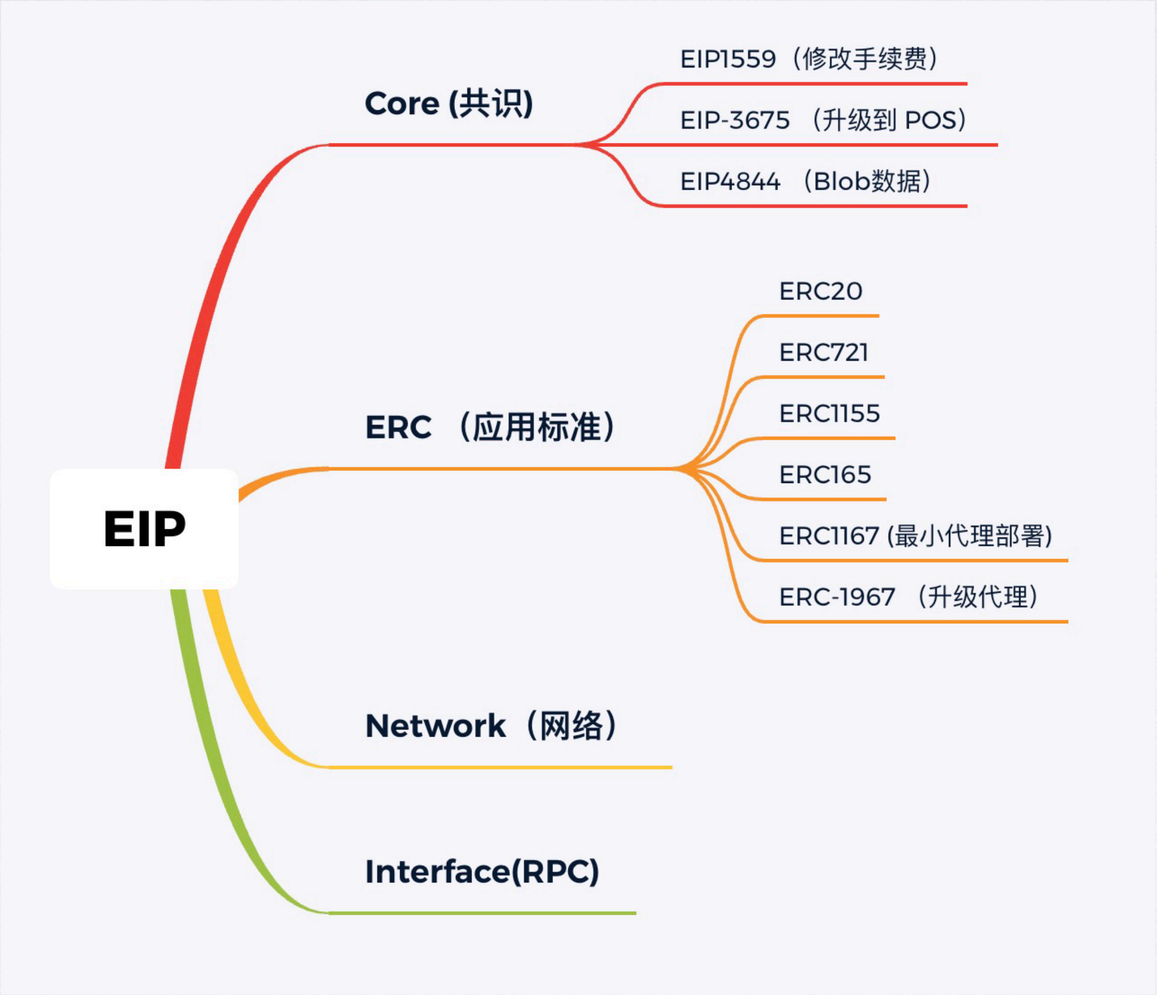
SOLIDITY – 开发进阶



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

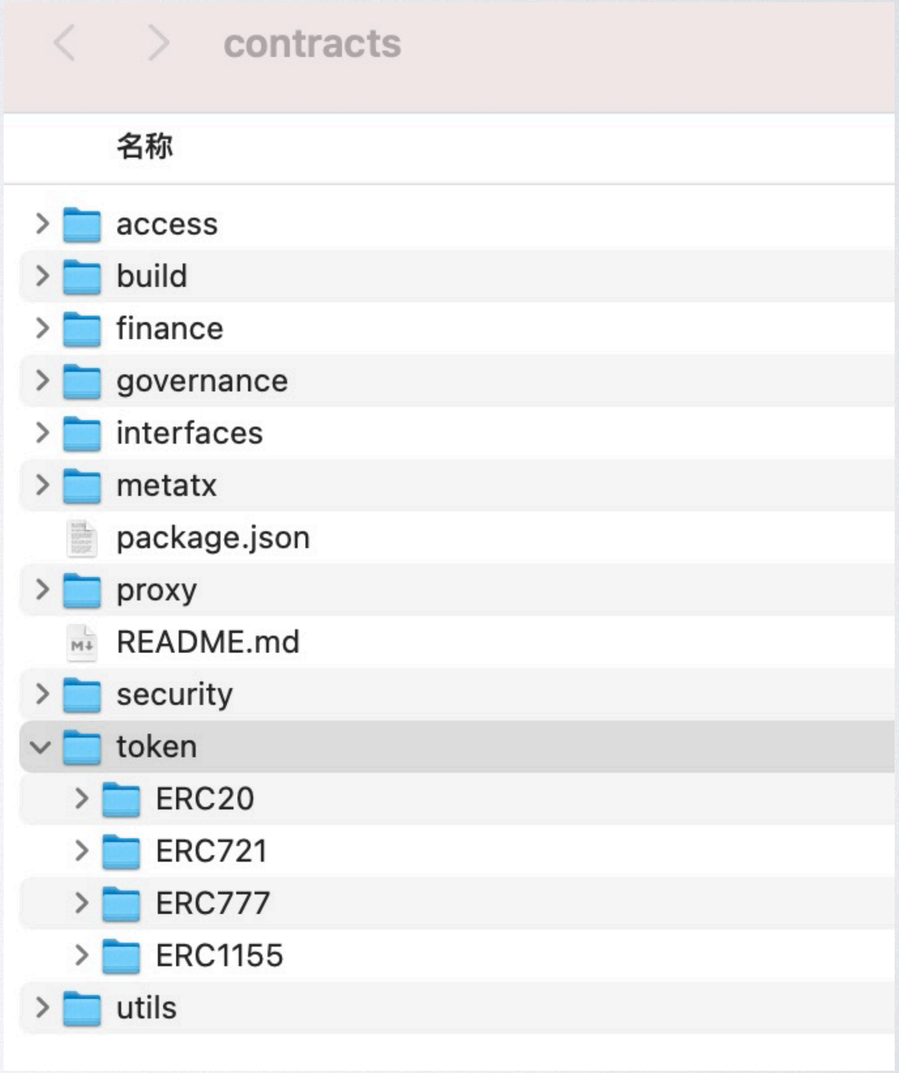

EIP
Core (共识)
EIP1559（修改手续费）
EIP-3675 （升级到 POS）
EIP4844 （Blob数据）
ERC （应用标准）
ERC20
ERC721
ERC1155
ERC165
ERC1167 (最小代理部署)
ERC-1967 （升级代理）
Network（网络）
Interface(RPC)

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

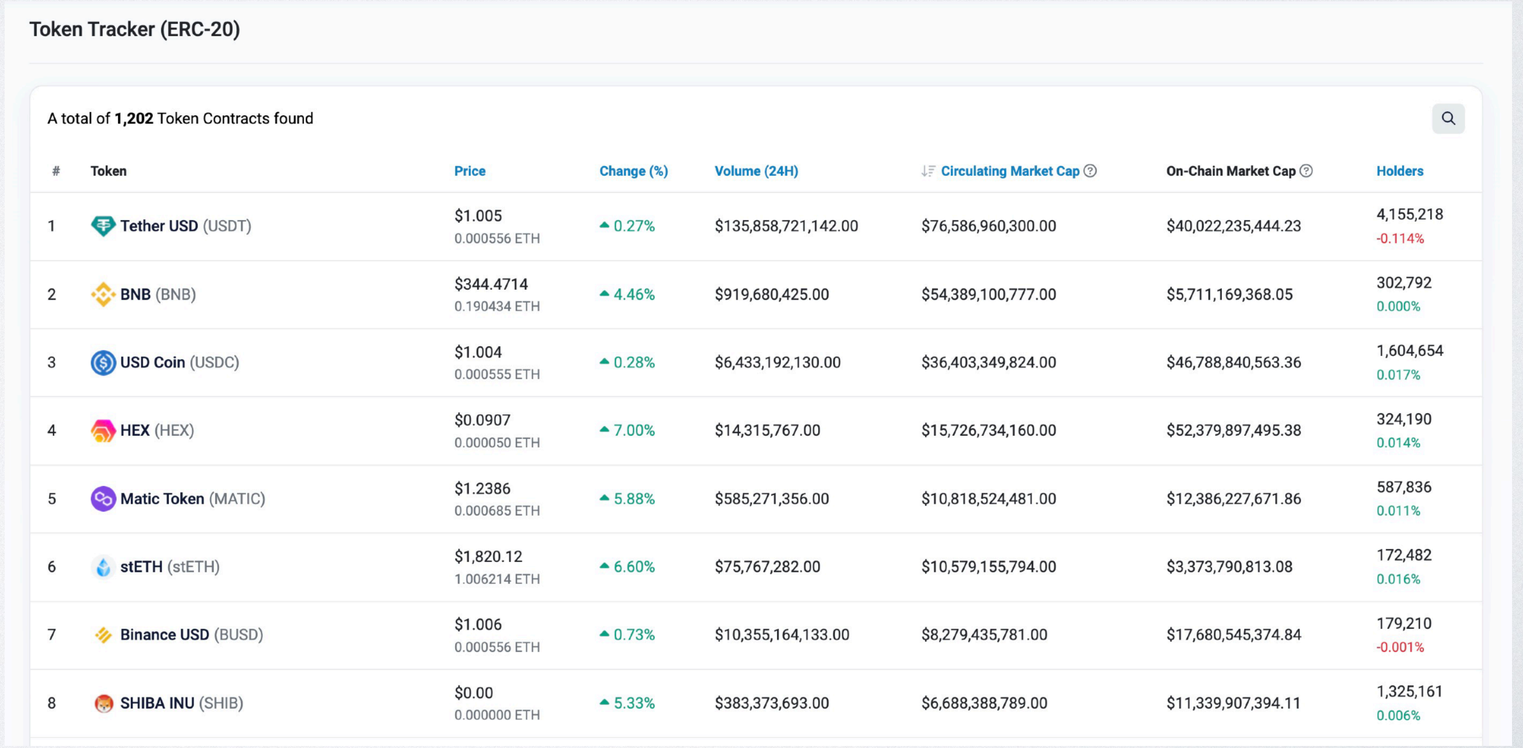


- token：包含 `ERC20`、`ERC721`、`ERC777`、`ERC1155` 代币
- access: 合约函数访问控制功能
- utils：实现一些工具库，如判断是否为合约地址、数学函数等。
- proxy：升级代理

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



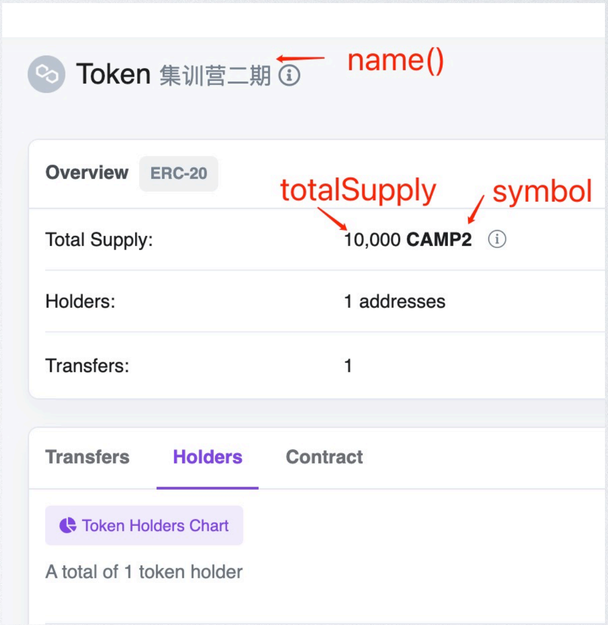
Token Tracker (ERC-20)

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 原理

- 如何表达每个人持有的数量?

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

ERC20

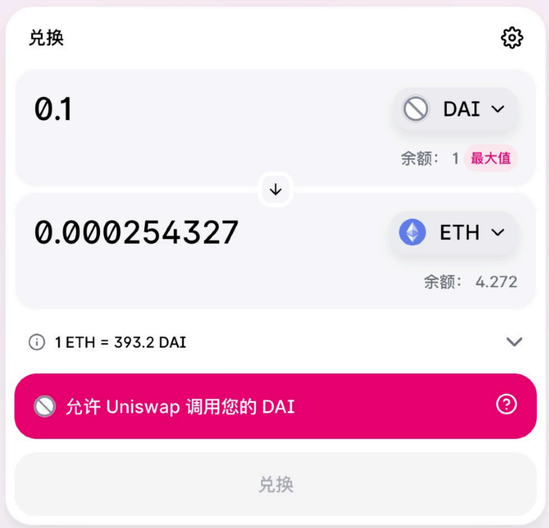
```
interface IERC20 {

      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(unicode"集训营二期", "CAMP2") {
        _mint(msg.sender, 10000*10**18);
    }
}
```

问题：如何用 uint 表示小数?

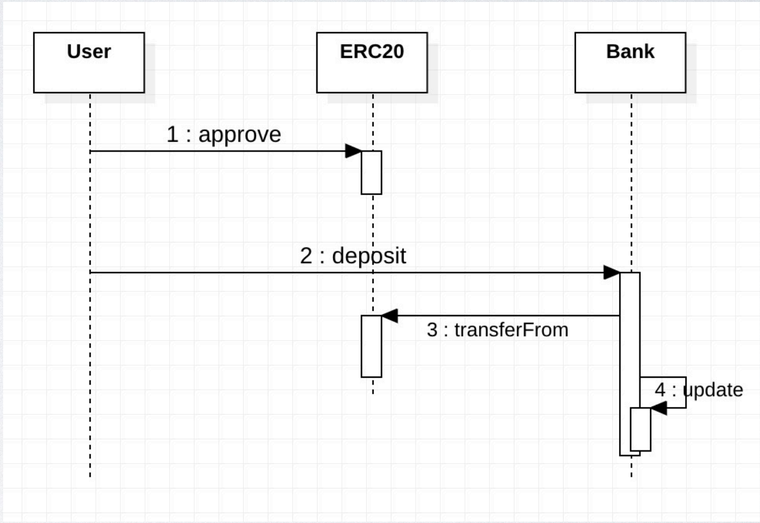
ERC20 – 授权



- 如何让合约使用 ERC20？
  - 例如，把 ERC20 存入合约？在合约里如何记录用户的存款量？
  - 用户直接向合约转账可以吗？

ERC20 – 授权

- 分两笔交易：
  - 用户 A 调用 ERC20 的 Approve(Bank，数量);
  - 用户 A 调用 合约 B 的 deposite(); 完成存款

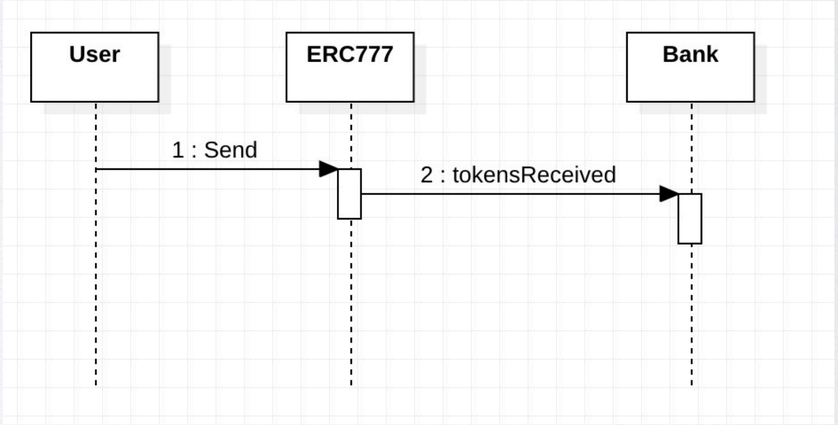



```
pragma solidity ^0.8.0;

contract Bank {
   mapping(address => uint) deposited;
   Function deposite(uint amount) {
     IERC20.transferFrom(msg.sender, address(this), amount);
     deposited[msg.sender] += amount;
   }
}
```

- 若要多次存款，如何减少授权次数?

ERC777

- ERC20 问题
  - 转账无法携带额外的信息。
  - 没有转账回调:
    - 依靠：授权、授权、授权
    - 误入合约被锁死。
- ERC777:
  - ERC777： send(dest, value, data)
  - 即便是普通地址也可以实现回调（如何实现的呢？）
  - 防止误入合约被锁死。
  - ERC777 通过全局注册表（ERC1820）的注册监听回调



ERC777

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC777/ERC777.sol";

contract MyERC777 is ERC777 {
    constructor() ERC777("MY777", "M777", new address[](0))
    {
        _mint(msg.sender, 1000 * 10 ** 18, "", "");
    }
}
```

https://learnblockchain.cn/docs/eips/eip-777.html

https://learnblockchain.cn/2019/09/27/erc777

ERC777

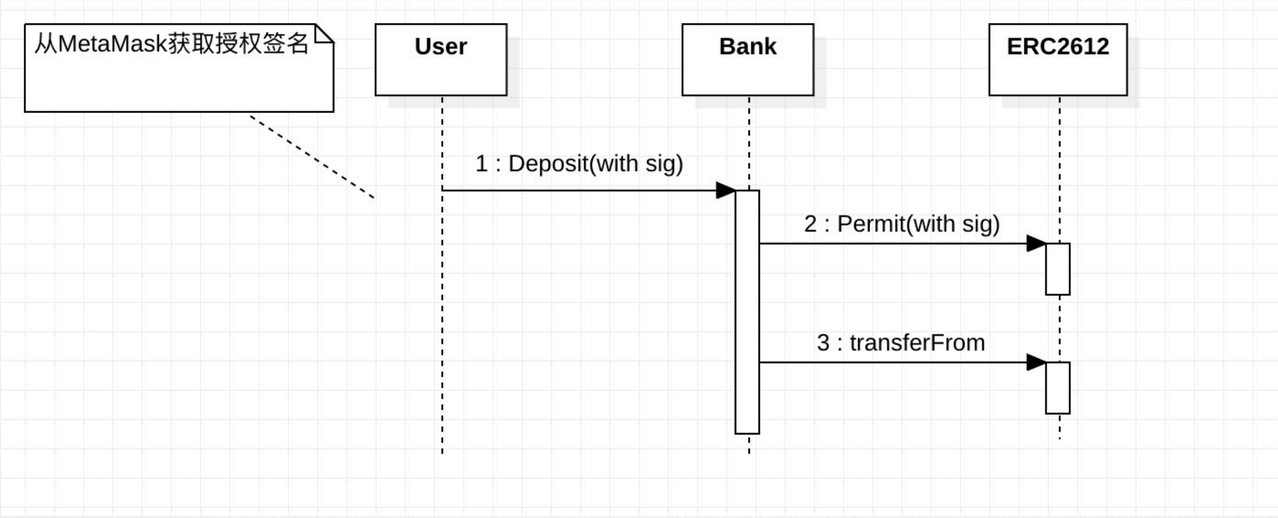
```
pragma solidity ^0.8.0;

function send(from, to, amount …) {
  _callTokensToSend(operator, from, to, amount,…);

  _move(operator, from, to, amount, userData, operatorData);

  _callTokensReceived(operator, from, to, amount, userData,…);

}

// 检查是否有注册回调，有的话就调用一下
function _callTokensToSend( operator,   from,   to, amount) private {
   address implementer = _ERC1820_REGISTRY.getInterfaceImplementer(from,
_TOKENS_SENDER_INTERFACE_HASH);
        if (implementer != address(0)) {
            IERC777Sender(implementer).tokensToSend(operator, from, to, amount,..);
        }
    }
```

ERC20-Callback

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/utils/Address.sol";

function transferWithCallback(address recipient, uint256 amount) external returns (bool) {
        _transfer(msg.sender, recipient, amount);

        if (recipient.isContract()) {
            bool rv = TokenRecipient(recipient).tokensReceived(msg.sender, amount);
            require(rv, "No tokensReceived");
        }

        return true;
    }
```

ERC20–Permit（EIP2612）

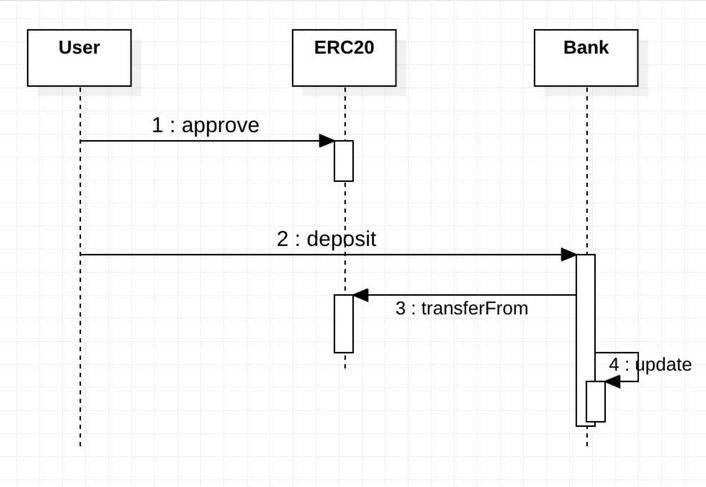
- 线下签名授权
  - （授权）可以在线下签名进行，签名信息可以在执行接收转账交易时提交到链上（permit），让授权和转账在一笔交易里完成。
  - 同时转账交易也可以由接收方（或其他第三方）来提交，也避免了用户（ERC20的拥有者）需要有 ETH的依赖。

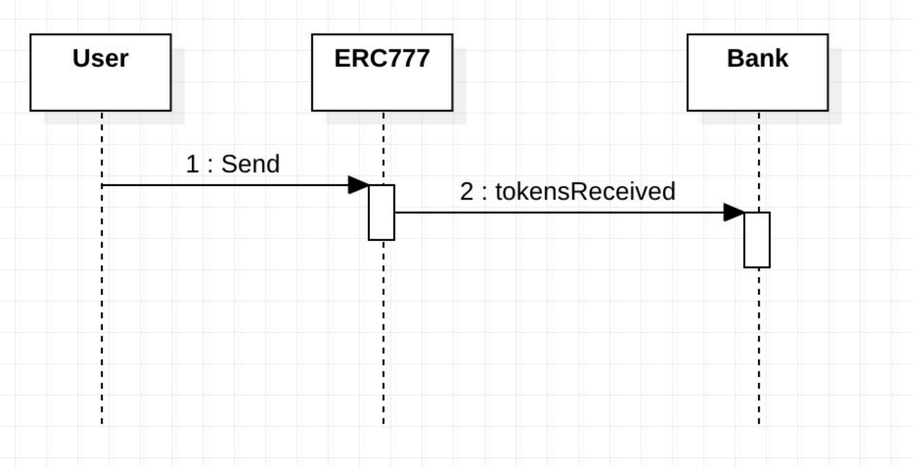


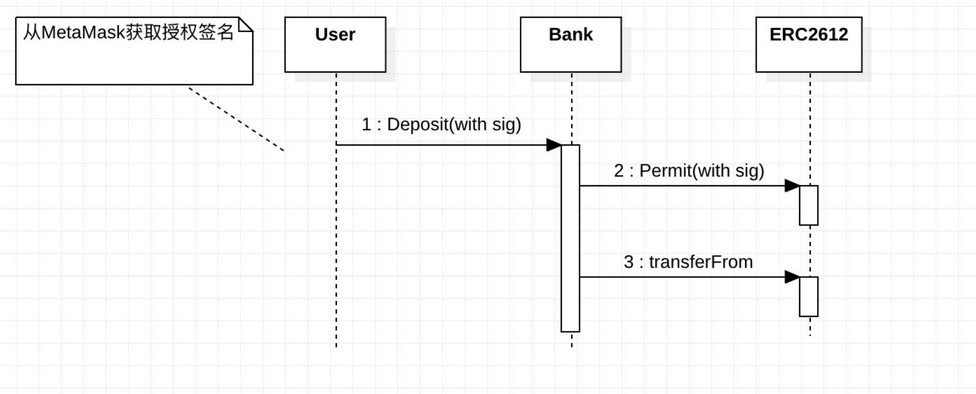
ERC20-Permit（EIP2612）

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract ERC2612 is ERC20, ERC20Permit {

    constructor() ERC20("ERC2612", "ERC2612") ERC20Permit("ERC2612") {
        _mint(msg.sender, 1000 * 10 ** 18);
    }
}
```

Token 总结

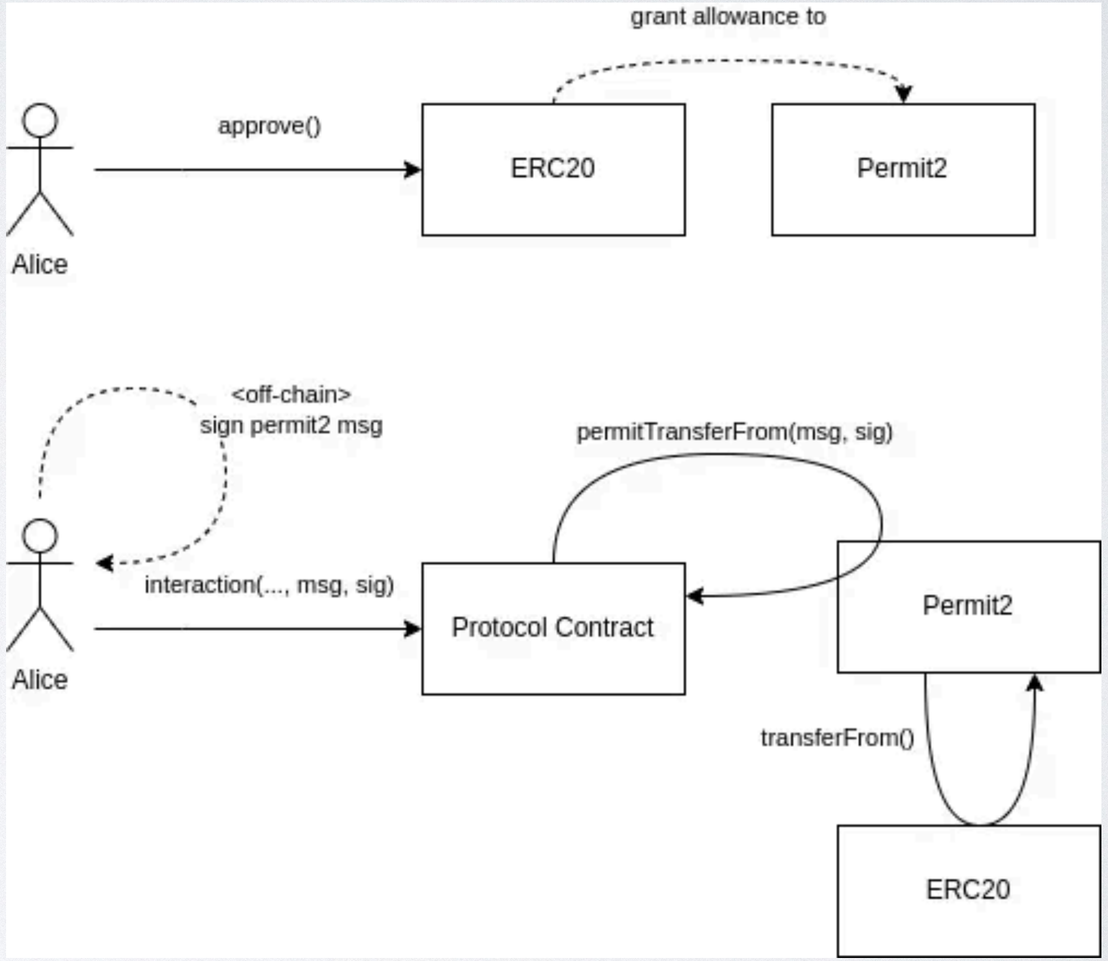

User
ERC20
Bank
1 : approve
2 : deposit
3 : transferFrom
4 : update


User
ERC777
Bank
1 : Send
2 : tokensReceived


从MetaMask获取授权签名
User
Bank
ERC2612
1 : Deposit(with sig)
2 : Permit(with sig)
3 : transferFrom

Permit2

- 有没有办法让为所有的Token 合约实现离线授权?
- Uniswap Permit2 结合了 approve 与 erc2612 - permit



https://github.com/Uniswap/permit2

ERC20 的一些坑

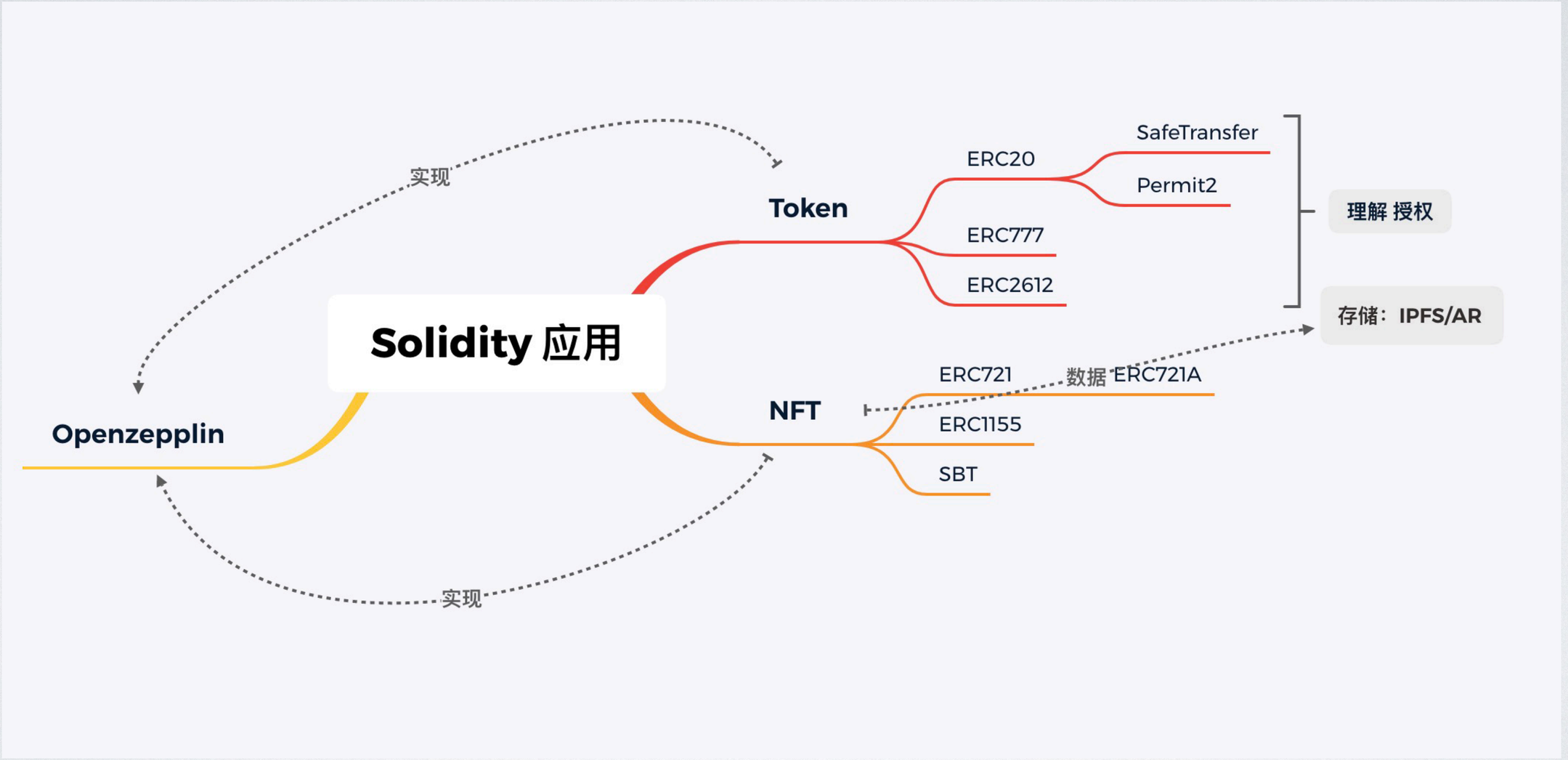
- 一些 Token 在实现时，转账失败没有回退，而是返回 false。
  - 导致，合约以为转账成功了，但却没有
  - 如：ZRX https://etherscan.io/address/0xe41d2489571d322189246dafa5ebde1f4699f498#code
- 怎么办？ 加一个返回值判断么，然而有些 Token 没有返回值。
  - 如： USDT： https://etherscan.io/address/0xdac17f958d2ee523a2206206994597c13d831ec7#code
- 那该怎么办?
  - ERC20 转账应该总是使用Openzeplin 的 SafeERC20 safeTransfer：

https://learnblockchain.cn/article/3074

练习题

- 发行一个 ERC20 Token（用自己的名字）， 发行 100000 token
- 编写一个金库 Vault 合约：
  - 编写 deposite 方法，实现 ERC20 存入 Vault，并记录每个用户存款金额（approve/transferFrom）
  - 编写 withdraw 方法，提取用户自己的存款
- 进阶练习：
  - 使用 ERC2612 标准 Token， 使用签名的方式 deposite

- https://learnblockchain.cn/video/play/274

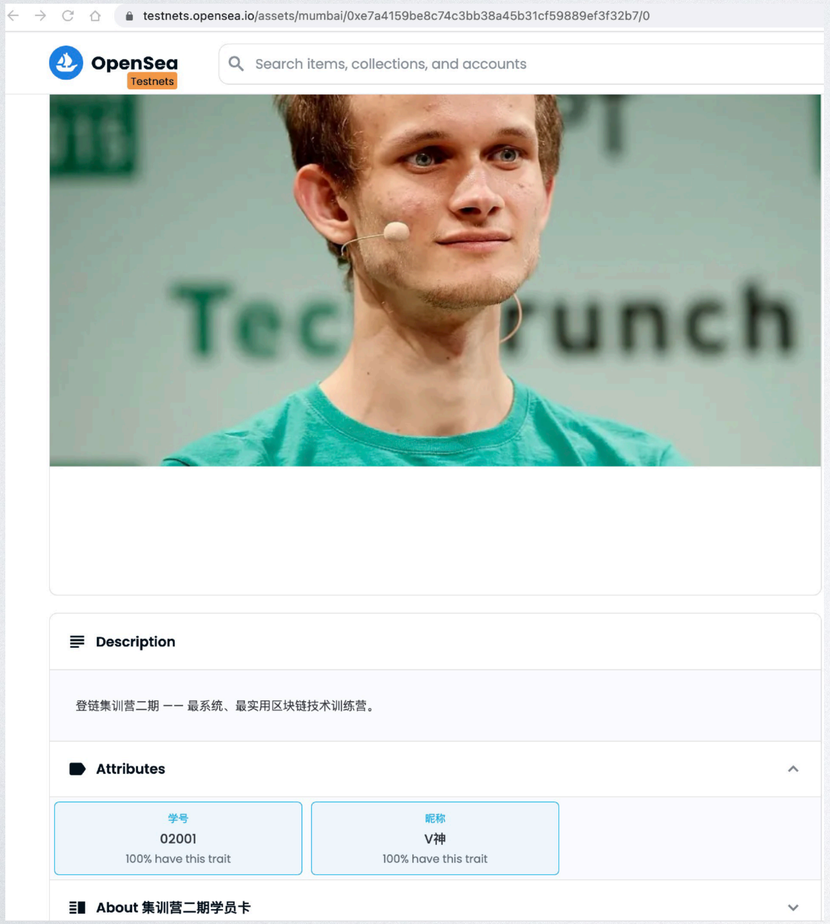

Solidity 应用
Token
ERC20
SafeTransfer
Permit2
ERC777
ERC2612
理解 授权
存储：IPFS/AR
NFT
ERC721
数据
ERC721A
ERC1155
SBT
Openzepplin
实现
实现

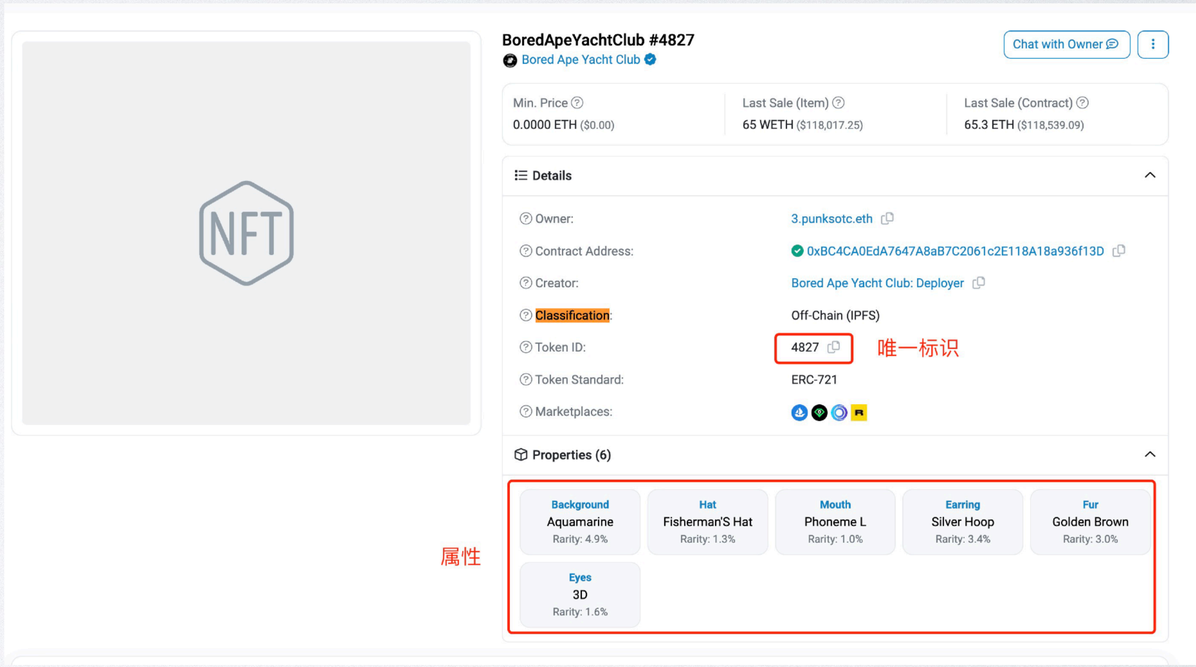
ERC721

- ERC20 合约中每个 token 与其他的 token 一样，称为同质化 Token（可置换的）
- ERC721 合约中每个 token 都是独一无二的，可用于表达如：
  - 艺术品画作、收藏品
  - 创作品：声音、影片、文章、一份档案
  - 游戏中的（限量）道具
  - 任何有特性的内容：一个交易记录

ERC721 长啥样?

- 集训营二期学员卡(Opensea/etherscan)





ERC721 如何实现

- 如何表达每个 token独一无二?

```
pragma solidity ^0.8.0;

contract ERC721 {
      // tokenId  => address
     mapping(uint256 => address) ownerOf;
     function tokenURI(uint256 tokenId) public view returns (string memory);

}
```

- ERC721 合约中每个 token 有一个 id
- 每个 Token 有一个对应 URI 来描述属性（JSON）

ERC721

```
pragma solidity ^0.8.0;

interface IERC721 /* is ERC165 */ {
    function name() external pure returns (string _name);
    function symbol() external pure returns (string _symbol);
    function tokenURI(uint256 _tokenId) external view returns (string);

    function balanceOf(address _owner) external view returns (uint256);
    function ownerOf(uint256 _tokenId) external view returns (address);

    function safeTransferFrom(address _from, address _to, uint256 _tokenId) external;
    function transferFrom(address _from, address _to, uint256 _tokenId) external;

    function approve(address _approved, uint256 _tokenId) external;
    function setApprovalForAll(address _operator, bool _approved) external;
    function getApproved(uint256 _tokenId) external view returns (address);
    function isApprovedForAll(address _owner, address _operator) external view returns
(bool);
}
```

ERC721

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract MyERC721 is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721(unicode"集训营二期学员卡", "CAMP2") {}

    function mint(address student, string memory tokenURI) public returns (uint256) {
        _tokenIds.increment();

        uint256 newItemId = _tokenIds.current();
        _mint(student, newItemId);
        _setTokenURI(newItemId, tokenURI);

        return newItemId;
    }
}
```

IPFS/AR

- 为什么需要去中心化存储?
  - 不可篡改、防单点故障
- IPFS： （取代HTTP）去中心化存储协议， 按内容寻址
- Arweave: 去中心化存储区块链，每个节点都是一个 HTTP 服务器

如何组织一个 NFT

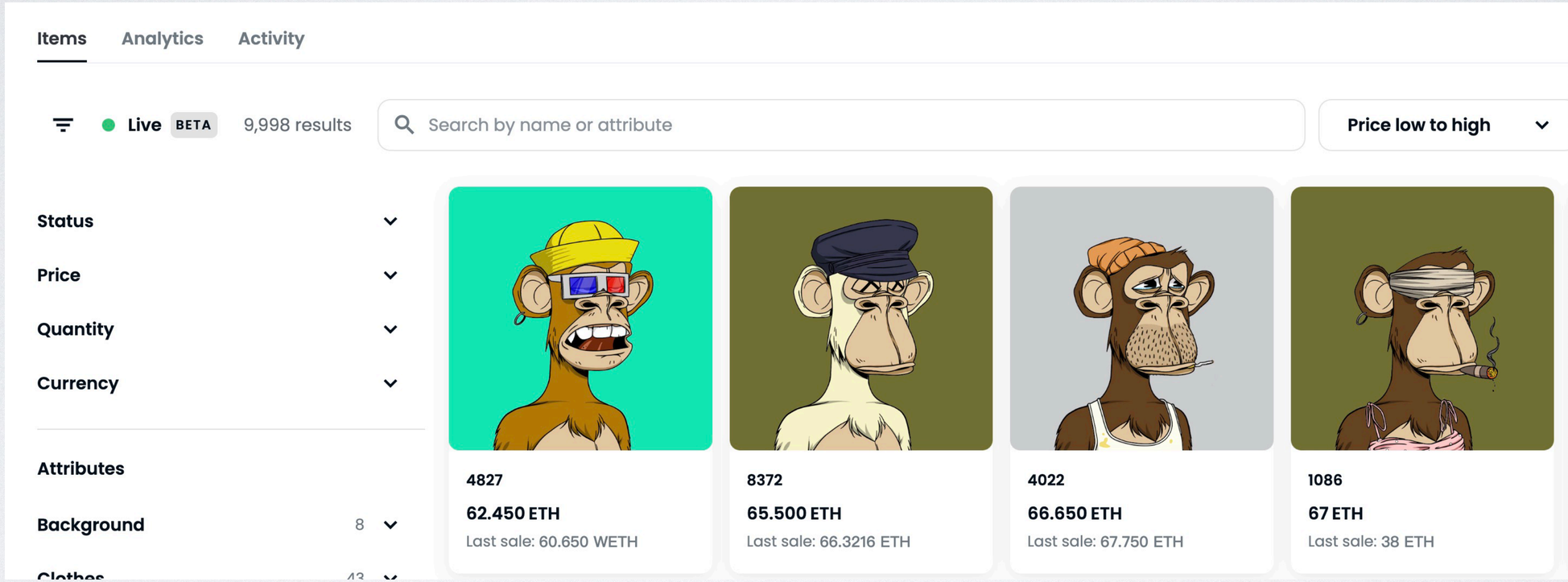
- 图片上传到如 IPFS （Pinata）
- 编写元数据文件(JSON)
- 元数据文件上传到 IPFS
- 调用mint() 方法
- Opensea 等 NFT 市场查看

ERC721 扩展

- ERC721A - 为批量铸造而生，优化Gas 成本
  - https://github.com/chiru-labs/ERC721A
- 用可升级合约单独展示元数据

```
contract YourNFTContract is ERC721 {
  address public metadataAddress;

  function setMetadataAddress(address addr) external onlyOwner {
    metadataAddress = addr;
  }

  function tokenURI(uint256 tokenId)
        public view override returns (string memory) {
    return MetadataContract(metadataAddress).tokenURI(tokenId);
  }
}


interface MetadataContract {
  function tokenURI(uint256 tokenId)
        external view returns (string memory);
}
```

NFT Market – Opensea



SBT

- 无法转让的 NFT = SBT （灵魂绑定 Token）
- 如何实现?

```
function transferFrom(address, address, uint256) public virtual override {
    revert("SBT:non-transferable");
}

function safeTransferFrom(
    address,
    address,
    uint256
) public virtual override {
    revert("SBT:non-transferable");
}
```

ERC1155

- ERC20 与 ERC721 的结合，兼顾独特性与数量
- 如：一份作品发行多个拷贝
- 一个限量的邮票、道具

```
mapping(uint256 => mapping(address => uint256)) private _balances;
```

ERC1155

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC1155/ERC1155.sol";

contract GameItems is ERC1155 {
    uint256 public constant GOLD = 0;
    uint256 public constant SILVER = 1;
    uint256 public constant THORS_HAMMER = 2;
    uint256 public constant SWORD = 3;
    uint256 public constant SHIELD = 4;

    constructor() ERC1155("https://game.example/api/item/{id}.json") {
        _mint(msg.sender, GOLD, 10**18, "");
        _mint(msg.sender, SILVER, 10**27, "");
        _mint(msg.sender, THORS_HAMMER, 1, "");
        _mint(msg.sender, SWORD, 10**9, "");
        _mint(msg.sender, SHIELD, 10**9, "");
    }
}
```

练习题

- 发行一个 ERC721 Token（用自己的名字）
  - 铸造 一个 NFT，在测试网上发行，在 Opensea 上查看
- 编写一个合约：使用自己发行的ERC20 Token 来买卖NFT：
  - NFT 持有者可上架 NFT（list 设置价格 多少个 TOKEN 购买 NFT ）
  - 编写购买NFT 方法，转入对应的TOKEN，获取对应的 NFT