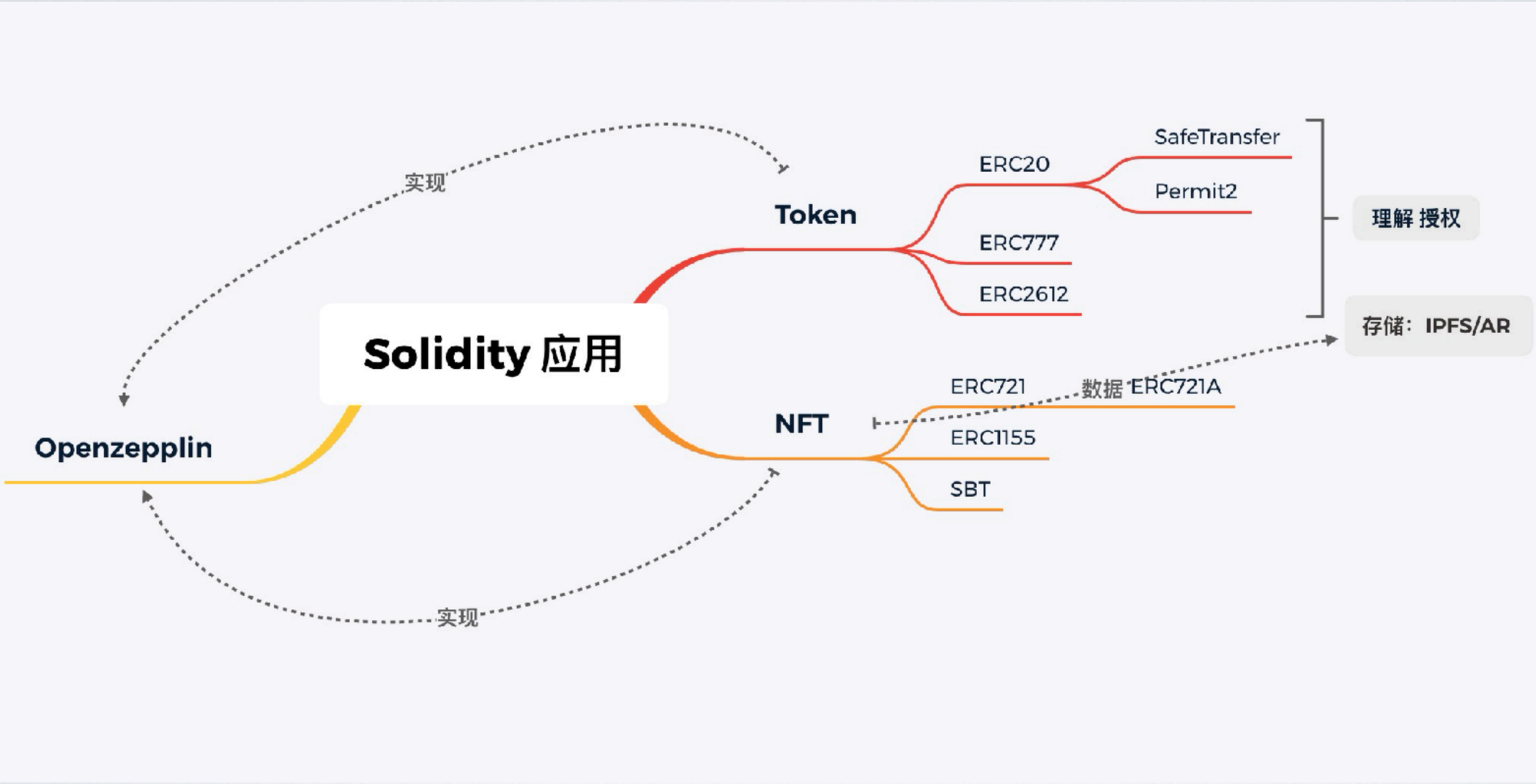
理解 NFT

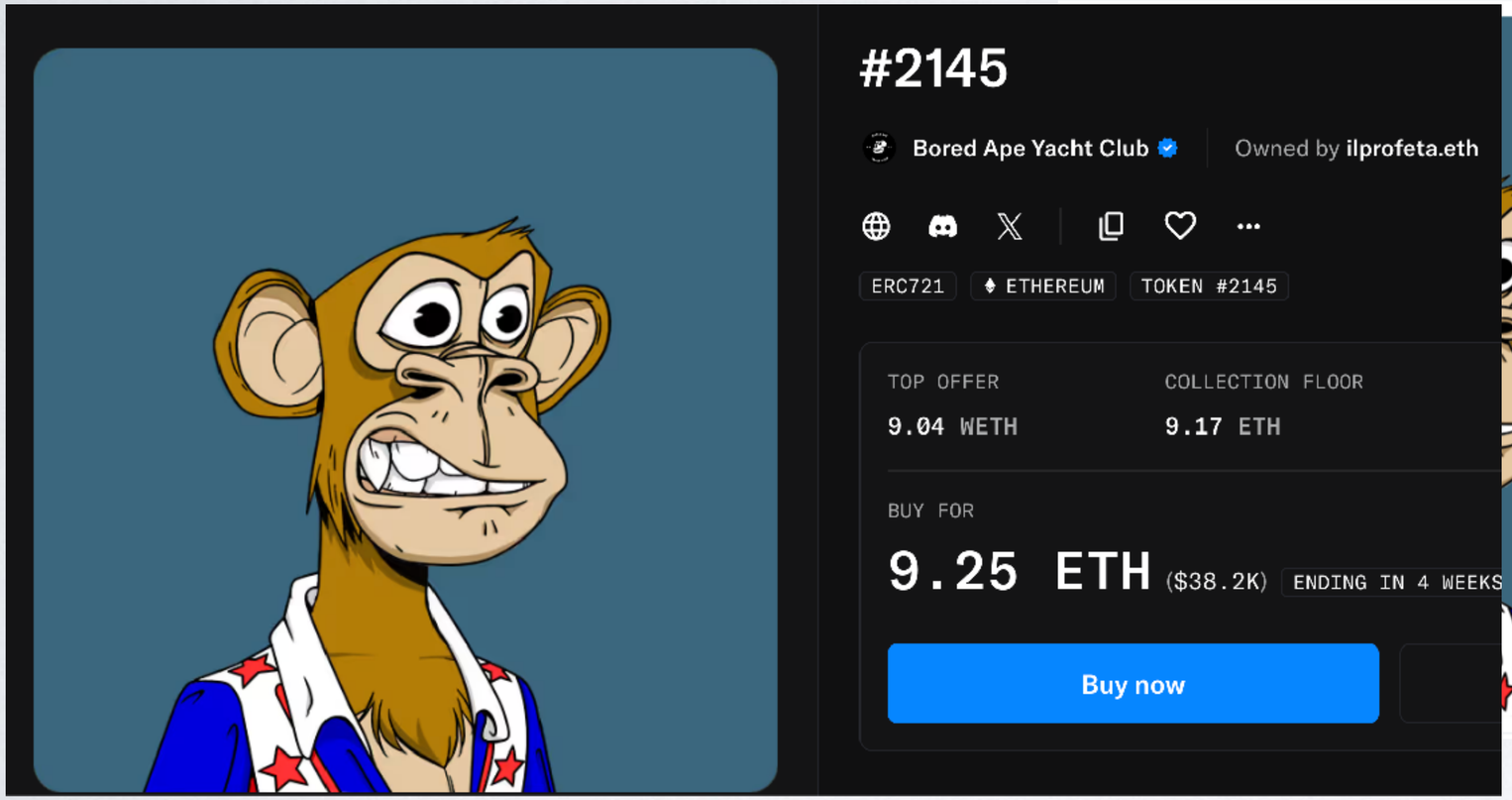

Solidity 应用
Token
ERC20
SafeTransfer
Permit2
ERC777
ERC2612
理解 授权
存储：IPFS/AR
NFT
ERC721
数据
ERC721A
ERC1155
SBT
Openzepplin
实现
实现

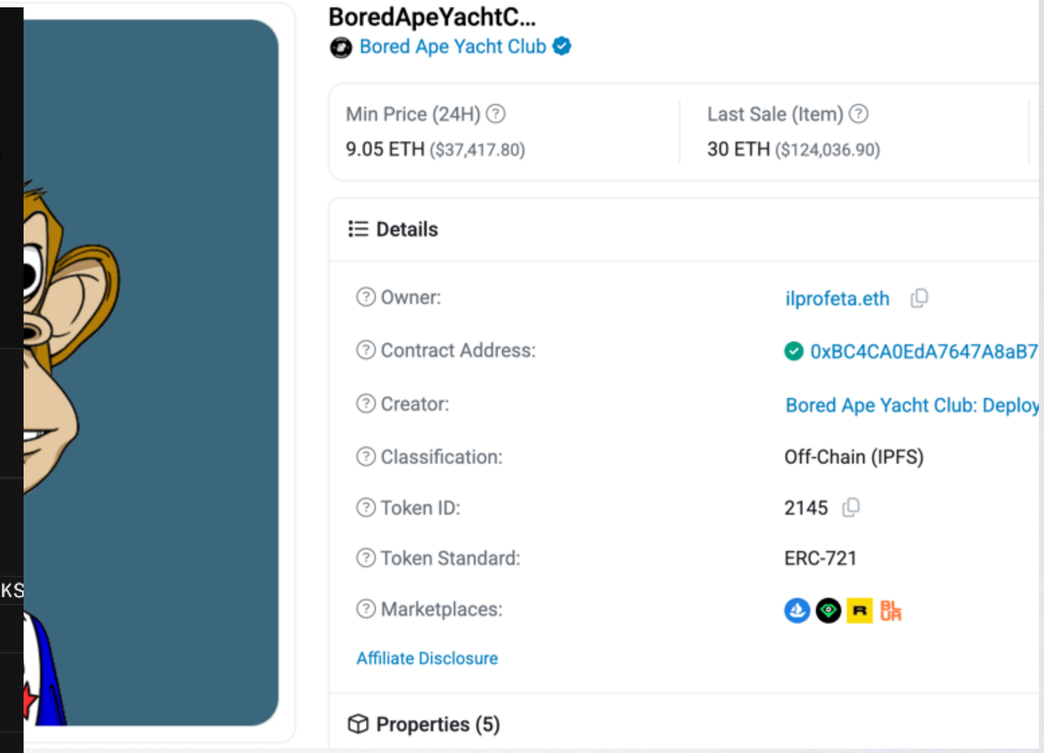
ERC721

- 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?

- Bored Ape Yacht(OpenSea / etherscan)





decert.me 挑战 NFT

如何表达独特性?

- 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 （序号）
- 每个 Token 有一个对应 URI 来描述属性（JSON）

https://docs.opensea.io/docs/metadata-standards

ERC721 MetaData

```
{
    "title": “集训营学员 - Vitalik",
    "description": “OpenSpace - 华语 Web3 黄埔军校”,
    "image": "ipfs://Qmdt6K59JBmz24iPh7FkU1Z1m3j8Uzbhc4kj97kBdfTJ8i",
    "attributes": [
        {
            "trait_type": "学号",
            "value": "02002"
        },
        {
            "trait_type": "昵称",
            "value": "Vitalik"
        }
    ],
    "version": “2"
}
```

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);
}
```

https://learnblockchain.cn/docs/eips/EIPS/eip-721/

ERC721

- ERC721 借鉴了 ERC20 标准经验教训
- 标准里引入 safeTransferFrom，加入对接收者 onERC721Received() 函数调用。
  - 通知接收者执行相应代码，同时避免 NFT 锁在合约地址里
- 加入了接口探测（ERC165），在合约中暴露自己实现了哪些方法
  - function supportsInterface(bytes4 interfaceID) external view returns (bool);

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol

ERC721 实现

- OpenZeppelin 包含了 ERC721 实现：
  - ERC721.sol ： 核心实现，所有权、转账等逻辑 （拼接 metadata url）
  - ERC721URIStorage.sol：记录 TokenID 对应的 Metadata url 的存储
  - ERC721Enumerable.sol：记录用户持有的 TokenId 及数量 （增加了转账成本）

ERC721 实现

```
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"集训营学员卡", "CAMP") {}

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

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

        return newItemId;
    }
}
```

铸造 NFT

NFT 合约准备好了

图片、属性等元数据该如何存储?

如何防止数据篡改或丢失？

IPFS/AR

- IPFS： （取代HTTP）去中心化存储协议， 按内容寻址
  - FileCoin： 去中心化存储区块链
- Arweave: 去中心化存储区块链，每个节点都是一个 HTTP 服务器

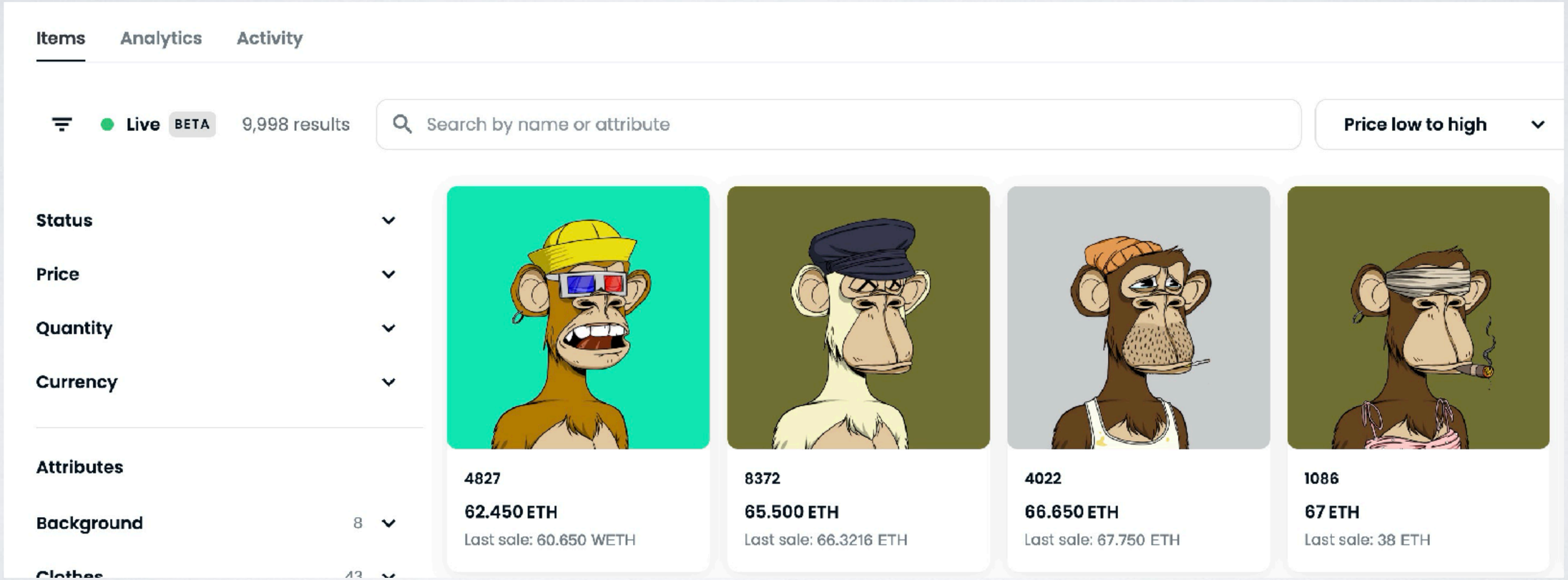
如何铸造一个 NFT

- 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 // hash
- 调用 mint() 方法
- Opensea 等 NFT 市场查看

https://www.pinata.cloud/
https://nft.storage/

NFT Market – Opensea

- NFT 交易市场通常会自动索引出链上所有的 NFT， 并展示出来。



https://opensea.io/collection/boredapeyachtclub

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

```
// token_id => who => amount
  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

- 练习编写 ERC721 NFT 合约
  - https://decert.me/challenge/852f5836-a03d-4483-a7e0-b0f6f8bda01c
- 发行一个 ERC721 Token（用自己的名字）
  - 铸造几个 NFT，在主网（Polygon、BNB 、OP Maintnet）上发行，在 Opensea 上查看
- 编写一个市场合约 NFTMarket：使用自己发行的 ERC20 Token 来买 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 buyNFT(uint tokenID, uint amount)，转入对应的TOKEN，获取对应的 NFT
  - 实现ERC20 扩展 Token 所要求的接收者方法 tokensReceived ，在 tokensReceived 中实现NFT 购买功能(注意扩展的转账需要添加一个额外数据参数)。

https://decert.me/quests/abdbc346-8314-4394-8f97-8732780602ed

练习题解答

- 熟悉 ERC721 实现
  - https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol
- safeTransferFrom 与合约如何接收 NFT
  - 实现 onERC721Received()