OST
Eastern Switzerland
University of Applied Sciences

Blockchain (BlCh)

NFTs

Thomas Bocek

14.10.2021

ERC-721

- Non-Fungible Token Standard
  - June 2018 - ERC721 accepted as final
  - “A Non-Fungible Token (NFT) is used to identify something or someone in a unique way.” [link]
  - “There is either one NFT or there are none”
  - Wikipedia
  - http://erc721.org (old)
  - Decrypt
  - https://eips.ethereum.org/EIPS/eip-721

- Pros
  - Trade items without middleman
  - Works 24/7
  - Only digital
- Cons
  - Technical complexity
  - NFTs on Ethereum not environment friendly
  - Only digital
  - Owning tokens does not necessarily mean owning copyright (unless it is explicitly transferred)

OST

Implementation

- `function balanceOf(address _owner) external view returns (uint256);`
  - Balance of NFTs of an owner
- `function ownerOf(uint256 _tokenId) external view returns (address);`
  - Queries the owner of a specific NFT
- `function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external payable;`
  `function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;`
  - Transfer token of owner, or of approved owner in a safe manner (calls onERC721Received)
- `function transferFrom(address _from, address _to, uint256 _tokenId) external payable;`
  - Make sure you have the right address

Implementation

- `function approve(address _approved, uint256 _tokenId) external payable;`
  `function setApprovalForAll(address _operator, bool _approved) external;`
  - Set approve that other address can transfer NFTs
- `function getApproved(uint256 _tokenId) external view returns (address);`
  `function isApprovedForAll(address _owner, address _operator) external view returns (bool);`
  - Check approval
- ERC165 - check here -mandatory! XOR of all function signatures
  - `function supportsInterface(bytes4 interfaceId) external view returns (bool);`
    -
```
return interfaceId == type(IERC721).interfaceId ||
       interfaceId == type(IERC721Metadata).interfaceId ||
       interfaceId == type(IERC165).interfaceId;
```


OST

Implementation

- Events
  - `event Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId);`
  - `event Approval(address indexed _owner, address indexed _approved, uint256 indexed _tokenId);`
  - `event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);`
- Optional ERC721Metadata
  - `function name() external view returns (string _name);`
  - `function symbol() external view returns (string _symbol);`
  - `function tokenURI(uint256 _tokenId) external view returns (string);`

OST

Implementation

- Optional Enumaration - discoverable
  - `function totalSupply() external view returns (uint256);`
  - `function tokenByIndex(uint256 _index) external view returns (uint256);`
  - `function tokenOfOwnerByIndex(address _owner, uint256 _index) external view returns (uint256);`

OST

Simple ERC-721

- Lets use the Imports this time
- Strings.sol
  - Helpers to convert uint256 to strings
  - Concatenate string with abi.encodePacked(…)
- Address.sol

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol";


contract BlChNFT is IERC721 {

}
```

```
import "@openzeppelin/contracts/utils/Strings.sol";
…
using Strings for uint256;
…
string(abi.encodePacked("https://dsl.hsr.ch/",_tokenId.toString()))
```


OST

Simple ERC-721

- Almost empty contract - set meta data
  - Only required functions

```
contract BlChNFT is IERC721 {
    using Strings for uint256;
    function name() external pure returns (string memory){return "Blockchain NFT";}
    function symbol() external pure returns (string memory){return "BlchNFT";}
    function tokenURI(uint256 _tokenId) external view returns (string memory){return
string(abi.encodePacked("https://dsl.hsr.ch/nft/", _tokenId.toString()));}
    function balanceOf(address _owner) external view returns (uint256) {return 1;}
    function ownerOf(uint256 _tokenId) external view returns (address) {return address(0);}
    function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes memory data) external override {}
    function safeTransferFrom(address _from, address _to, uint256 _tokenId) external override {}
    function transferFrom(address _from, address _to, uint256 _tokenId) external override{}
    function approve(address _approved, uint256 _tokenId) external override{}
    function setApprovalForAll(address _operator, bool _approved) external{}
    function getApproved(uint256 _tokenId) external view returns (address){return address(0);}
    function isApprovedForAll(address _owner, address _operator) external view returns (bool){return true;}
    function supportsInterface(bytes4 interfaceId) external view returns (bool){return true;}
}
```


OST

Simple ERC-721

- **No checks!**
- State variables
- Implement view

```
// Mapping owner address to token count
mapping(address => uint256) private _balances;
// Mapping from token ID to owner address
mapping(uint256 => address) private _owners;
// Mapping from token ID to approved address
mapping(uint256 => address) private _tokenApprovals;
// Mapping from owner to operator approvals
mapping(address => mapping(address => bool)) private _operatorApprovals;
```

```
function balanceOf(address _owner) external view returns (uint256) {
    return _balances[_owner];
}
function ownerOf(uint256 _tokenId) external view returns (address) {
    return _owners[_tokenId];
}
function getApproved(uint256 _tokenId) external view returns (address){
    return _tokenApprovals[_tokenId];
}
function isApprovedForAll(address _owner, address _operator) external view returns (bool){
    return _operatorApprovals[_owner][_operator];
}
```


OST

Simple ERC-721

- TransferFrom
- SafeTransferFrom

```
function transferFrom(address _from, address _to, uint256 _tokenId) public
override{
    address owner = ownerOf(_tokenId);
    require(msg.sender == owner || getApproved(_tokenId) == msg.sender ||
isApprovedForAll(owner, msg.sender), "ERC721: safeTransferFrom caller is not
owner nor approved for all");
    // Clear approvals from the previous owner
    _tokenApprovals[_tokenId] = address(0);
    emit Approval(owner, address(0), _tokenId);

    _balances[_from] -= 1;
    _balances[_to] += 1;
    _owners[_tokenId] = _to;

    emit Transfer(_from, _to, _tokenId);
}
```

```
function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes
memory _data) public override {
    transferFrom(_from, _to, _tokenId);
    if (_to.isContract()) {
        bytes4 retVal = IERC721Receiver(_to).onERC721Received(msg.sender,
_from, _tokenId, _data);
        require (retVal == IERC721Receiver.onERC721Received.selector, "ERC721:
transfer to non ERC721Receiver implementer");
    }
}
```


OST

Simple ERC-721

- Let’s mint!

```
address private ownerNFT;
constructor() {
    ownerNFT = msg.sender;
}
function _mint(address to, uint256 tokenId) external {
    require(ownerNFT == msg.sender, "only onwner");
    require(_owners[tokenId] == address(0), "ERC721: token already minted");

    _balances[to] += 1;
    _owners[tokenId] = to;
    emit Transfer(address(0), to, tokenId);
}
```


OST

ERC721

- Standard Token Deployment
- Extend our ERC-20 contract to receive tokens

```
function onERC721Received(
        address operator,
        address from,
        uint256 tokenId,
        bytes calldata data
    ) external returns (bytes4) {
        IERC721(msg.sender).transferFrom(address(this), address(0x0CbdF5B0c4E117619631bA4b97dC0d439ADAbD88), tokenId);
        return IERC721Receiver.onERC721Received.selector;
    }
```


OST

ERC721

- OpenSea
  - https://testnets.opensea.io/
  - More metadata here

```
contract MyCollectible is ERC721 {
    function contractURI() public view returns (string memory) {
        return "https://metadata-url.com/my-metadata";
    }
}
```

```
{
  "name": "OpenSea Creatures",
  "description": "OpenSea Creatures are adorable aquatic beings primarily for demonstrating what can be done using the
OpenSea platform. Adopt one today to try out all the OpenSea buying, selling, and bidding feature set.",
  "image": "https://openseacreatures.io/image.png",
  "external_link": "https://openseacreatures.io",
  "seller_fee_basis_points": 100, # Indicates a 1% seller fee.
  "fee_recipient": "0xA97F337c39cccE66adfeCB2BF99C1DdC54C2D721" # Where seller fees will be paid to.
}
```


OST