L18: ERC-721 and ERC-1155 Standards

Module B: Ethereum & Smart Contracts

Blockchain & Cryptocurrency Course

December 2025

Learning Objectives

By the end of this lesson, you will be able to:

- Explain the difference between fungible and non-fungible tokens
- Describe the ERC-721 interface for NFTs
- Understand token metadata and IPFS storage
- Implement safe transfer mechanisms to prevent token loss
- Explain the ERC-1155 multi-token standard
- Compare batch operations and gas efficiency across standards
- Analyze real-world NFT projects (CryptoPunks, BAYC, ENS)

Fungible vs Non-Fungible Tokens

Fungible (ERC-20):

- Interchangeable (1 USDC = 1 USDC)
- Divisible (can send 0.5 tokens)
- Uniform value
- Examples: Currencies, commodities
- Use case: Payments, DeFi

Analogy:

- Dollar bills
- Bitcoin
- Gold bars (fungible)

Non-Fungible (ERC-721):

- Unique (token ID 1 $\neq$ token ID 2)
- Indivisible (cannot send 0.5 NFT)
- Individual value
- Examples: Art, collectibles, domain names
- Use case: Ownership proof, identity

Analogy:

- Real estate deeds
- Baseball cards
- Concert tickets (numbered seats)

Key Insight: ERC-721 tracks ownership of individual items, not aggregate balances

ERC-721 Interface: Core Functions

```
interface IERC721 {
    // Returns number of NFTs owned by owner
    function balanceOf(address owner) external view returns (uint256);

    // Returns owner of token ID
    function ownerOf(uint256 tokenId) external view returns (address);

    // Transfers token from 'from' to 'to' (unsafe, checks if recipient is contract)
    function transferFrom(address from, address to, uint256 tokenId) external;

    // Safe transfer with data (checks if recipient can receive NFTs)
    function safeTransferFrom(address from, address to, uint256 tokenId, bytes data)
        external;

    // Approve address to transfer specific token
    function approve(address to, uint256 tokenId) external;

    // Approve operator to transfer all tokens owned by caller
    function setApprovalForAll(address operator, bool approved) external;

    // Get approved address for specific token
    function getApproved(uint256 tokenId) external view returns (address);

    // Check if operator is approved to manage all tokens of owner
    function isApprovedForAll(address owner, address operator)
        external view returns (bool);
}
```

ERC-721 Interface: Events

```
interface IERC721 {
    // Emitted when token is transferred (including minting and burning)
    event Transfer(address indexed from, address indexed to, uint256 indexed tokenId);

    // Emitted when approval is set for specific token
    event Approval(address indexed owner, address indexed approved,
                   uint256 indexed tokenId);

    // Emitted when operator approval is set for all tokens
    event ApprovalForAll(address indexed owner, address indexed operator, bool approved);
}
```

Key Differences from ERC-20:

- `tokenId` is indexed (can filter by specific NFT)
- `Transfer` uses `from` address (ERC-20 uses `from` only in `transferFrom`)
- Minting: `from = address(0)`, Burning: `to = address(0)`
- `ApprovalForAll` enables operators (e.g., OpenSea) to manage all NFTs

Safe Transfer Mechanism

Problem: Transferring NFT to contract that cannot handle it = permanent loss

Solution: `safeTransferFrom()` checks if recipient is contract

```
interface IERC721Receiver {
    function onERC721Received(address operator, address from, uint256 tokenId,
                              bytes calldata data) external returns (bytes4);
}

// Safe transfer checks if recipient is contract
function safeTransferFrom(address from, address to, uint256 tokenId) public {
    transferFrom(from, to, tokenId);

    if (to.code.length > 0) {  // Recipient is contract
        require(
            IERC721Receiver(to).onERC721Received(msg.sender, from, tokenId, "")
                == IERC721Receiver.onERC721Received.selector,
            "Recipient cannot receive NFT"
        );
    }
}
```

Recipient must implement `onERC721Received()` to accept NFTs

ERC-721 Metadata Extension

Optional but widely used:

```
interface IERC721Metadata {
    // Token collection name (e.g., "CryptoPunks")
    function name() external view returns (string memory);

    // Token collection symbol (e.g., "PUNK")
    function symbol() external view returns (string memory);

    // URI for token metadata (JSON file)
    function tokenURI(uint256 tokenId) external view returns (string memory);
}
```

Metadata JSON Structure:

```
{
  "name": "Bored Ape #1234",
  "description": "A unique Bored Ape Yacht Club NFT",
  "image": "ipfs://QmX7H9K2pZ3.../1234.png",
  "attributes": [
    {"trait_type": "Background", "value": "Blue"},
    {"trait_type": "Eyes", "value": "Laser"}
  ]
}
```

Metadata Storage: On-Chain vs IPFS

On-Chain Storage:

- Store metadata in contract storage
- Expensive (20,000 gas per 32 bytes)
- Permanent and immutable
- Example: Loot (full text stored on-chain)

Example:

```
mapping(uint => string) private _uris;

function tokenURI(uint tokenId)
    returns (string memory) {
    return _uris[tokenId];
}
```

Cost: Storing 1 KB = approximately 640,000 gas (approximately $20-100 depending on gas price)

IPFS Storage:

- Store only IPFS hash on-chain
- Cheap (approximately 20,000 gas per token)
- Content-addressable (hash = content)
- Requires IPFS node for retrieval
- Example: Bored Ape Yacht Club

Example:

```
string private _baseURI =
    "ipfs://QmX7H9K2pZ3.../";

function tokenURI(uint tokenId)
    returns (string memory) {
    return string(abi.encodePacked(
        _baseURI,
        Strings.toString(tokenId)
    ));
}
```

Cost: Storing hash = approximately 20,000 gas (approximately $1-5)

IPFS: InterPlanetary File System

Decentralized content-addressable storage:

Key Concepts:

- **Content-Addressed:** File hash = file identifier (CID)
- **Immutable:** Changing content changes hash
- **Distributed:** Files stored across peer network
- **Permanent (with pinning):** Keep file alive by pinning to IPFS node

IPFS URIs:

- Format: `ipfs://QmX7H9K2pZ3.../metadata.json`
- CID: `QmX7H9K2pZ3...` (content identifier, approximately 46 characters)
- Gateway access: `https://ipfs.io/ipfs/QmX7H9K2pZ3.../metadata.json`

Tradeoffs:

- Pro: Cheap, decentralized, immutable
- Con: Requires IPFS node to retrieve, not guaranteed available (unless pinned)
- Mitigation: Pin files to Pinata, Infura IPFS, or self-hosted node

Basic ERC-721 Implementation

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract MyNFT is ERC721, Ownable {
    uint256 private _tokenIdCounter;
    string private _baseTokenURI;

    constructor(string memory baseURI) ERC721("My NFT Collection", "MNFT") {
        _baseTokenURI = baseURI;
    }

    function mint(address to) public onlyOwner {
        _tokenIdCounter++;
        _safeMint(to, _tokenIdCounter);
    }

    function _baseURI() internal view override returns (string memory) {
        return _baseTokenURI;
    }

    function tokenURI(uint256 tokenId) public view override returns (string memory) {
        require(_exists(tokenId), "Token does not exist");
        return string(abi.encodePacked(_baseURI(), Strings.toString(tokenId), ".json"));
    }
}
```

Real-World Example: CryptoPunks

One of the first NFT projects (2017):

Unique Characteristics:

- Pre-dates ERC-721 standard (custom implementation)
- 10,000 unique 24x24 pixel art characters
- Images stored on-chain (composite image in contract)
- No royalties (pre-dates EIP-2981)
- Wrapped version (Wrapped Punks) for ERC-721 compatibility

Key Functions:

- `getPunk(uint punkIndex)`: Claim free punk (original launch)
- `transferPunk(address to, uint punkIndex)`: Transfer ownership
- `offerPunkForSale(uint punkIndex, uint minPrice)`: List for sale
- `buyPunk(uint punkIndex)`: Purchase listed punk

Address: 0xb47e3cd837dDF8e4c57F05d70Ab865de6e193BBB

Real-World Example: Bored Ape Yacht Club (BAYC)

Premier NFT collection (launched 2021):

Technical Details:

- Standard ERC-721 implementation (OpenZeppelin)
- 10,000 unique apes with programmatically generated traits
- Metadata stored on IPFS (cost-efficient)
- Base URI: `ipfs://QmeSjSinHpPnmXmspMjwiXyN6zS4E9zccariGR3jxcaWtq/`
- Provenance hash: Single hash proving pre-reveal image order

Smart Contract Features:

- Fixed supply (10,000, all minted)
- Mint price: 0.08 ETH (2021)
- No owner mint (fair launch)
- Commercial rights granted to NFT holders

Address: 0xBC4CA0EdA7647A8aB7C2061c2E118A18a936f13D

Real-World Example: Ethereum Name Service (ENS)

Decentralized domain name system:

Technical Implementation:

- ENS domains are ERC-721 NFTs (e.g., `vitalik.eth`)
- Each domain has unique token ID (hash of domain name)
- Ownership = control of domain resolution
- Renewable (not permanent, annual fees)

Key Contracts:

- **ENS Registry:** Core mapping of names to owners
- **BaseRegistrar:** ERC-721 wrapper for .eth domains
- **Resolver:** Maps domain to address/content hash/text records

Unique Features:

- Subdomains: Owner of `alice.eth` can create `pay.alice.eth`
- Reverse resolution: Address $\rightarrow$ primary ENS name
- NFT marketplace integration (OpenSea, LooksRare)

ERC-1155: Multi-Token Standard

Unified standard for fungible and non-fungible tokens:

Key Innovation:

- Single contract can manage multiple token types
- Each token type has unique ID
- Token ID can be fungible (multiple units) or non-fungible (single unit)
- Batch operations: Transfer multiple token types in one transaction

Use Cases:

- Gaming: Swords (fungible), unique armor (non-fungible) in same contract
- Collectibles: Common cards (fungible), legendary cards (non-fungible)
- Event tickets: General admission (fungible), VIP seats (non-fungible)

Advantages over ERC-20 + ERC-721:

- Reduced deployment costs (one contract vs many)
- Gas-efficient batch transfers
- Atomic swaps (trade multiple token types in single transaction)

ERC-1155 Interface

```
interface IERC1155 {
    // Returns balance of token type id for account
    function balanceOf(address account, uint256 id) external view returns (uint256);

    // Returns balances of multiple account/id pairs
    function balanceOfBatch(address[] accounts, uint256[] ids)
        external view returns (uint256[]);

    // Transfer amount of token type id from 'from' to 'to'
    function safeTransferFrom(address from, address to, uint256 id, uint256 amount,
                              bytes data) external;

    // Batch transfer multiple token types
    function safeBatchTransferFrom(address from, address to, uint256[] ids,
                                   uint256[] amounts, bytes data) external;

    // Approve operator to manage all tokens for caller
    function setApprovalForAll(address operator, bool approved) external;

    // Check if operator is approved for owner
    function isApprovedForAll(address owner, address operator)
        external view returns (bool);
}
```

Note: No per-token approval (only operator approval)

ERC-1155 Events

```
interface IERC1155 {
    // Emitted on single transfer
    event TransferSingle(address indexed operator, address indexed from,
                         address indexed to, uint256 id, uint256 value);

    // Emitted on batch transfer
    event TransferBatch(address indexed operator, address indexed from,
                        address indexed to, uint256[] ids, uint256[] values);

    // Emitted when operator approval changes
    event ApprovalForAll(address indexed owner, address indexed operator, bool approved);

    // Emitted when token URI changes (optional)
    event URI(string value, uint256 indexed id);
}
```

Key Differences:

- `operator` field: Who initiated transfer (may differ from owner)
- `TransferBatch`: Single event for multiple token types
- `URI`: Allows dynamic metadata updates

Batch Operations: Gas Efficiency

Example: Transfer 5 different items

ERC-721 (5 separate transactions):

- Transaction 1: $21{,}000 + 50{,}000 = 71{,}000$ gas
- Transaction 2: $21{,}000 + 50{,}000 = 71{,}000$ gas
- Transaction 3: $21{,}000 + 50{,}000 = 71{,}000$ gas
- Transaction 4: $21{,}000 + 50{,}000 = 71{,}000$ gas
- Transaction 5: $21{,}000 + 50{,}000 = 71{,}000$ gas

Total: 355,000 gas

At 30 Gwei: 0.01065 ETH (approximately $21 at $2000/ETH)

ERC-1155 (1 batch transaction):

- Transaction: 21,000 (base)
- + 5,000 per token (cold SLOAD)
- + 2,900 per token (storage update)
- Total: $21{,}000 + 5 \times 7{,}900$
- = 60,500 gas

Total: 60,500 gas

At 30 Gwei: 0.001815 ETH (approximately $3.63 at $2000/ETH)

Savings: 83% gas reduction

Royalties: EIP-2981

Standard for NFT royalties on secondary sales:

```
interface IERC2981 {
    // Returns royalty receiver and amount for given sale price
    function royaltyInfo(uint256 tokenId, uint256 salePrice)
        external view returns (address receiver, uint256 royaltyAmount);
}

contract MyNFTWithRoyalties is ERC721, ERC2981 {
    constructor() ERC721("My NFT", "MNFT") {
        _setDefaultRoyalty(msg.sender, 500);  // 5% royalty to deployer
    }

    function supportsInterface(bytes4 interfaceId)
        public view override(ERC721, ERC2981) returns (bool) {
        return super.supportsInterface(interfaceId);
    }
}
```

Note: Marketplaces (OpenSea, Blur) voluntarily honor royalties
Challenge: No on-chain enforcement, purely social/marketplace-enforced

Key Takeaways

1. **Non-Fungible Tokens:** ERC-721 standard for unique, indivisible assets with individual ownership
2. **Safe Transfers:** `safeTransferFrom()` prevents token loss by checking if recipient can receive NFTs
3. **Metadata:** Token URIs point to JSON metadata, typically stored on IPFS for cost efficiency
4. **IPFS:** Content-addressable decentralized storage, requires pinning for availability
5. **ERC-1155:** Multi-token standard supporting both fungible and non-fungible tokens in single contract
6. **Batch Operations:** ERC-1155 batch transfers reduce gas costs by up to 83%
7. **Real-World:** CryptoPunks (pre-ERC-721), BAYC (standard ERC-721), ENS (renewable NFT domains)

Discussion Questions

1. Why is the `onERC721Received()` callback necessary for safe NFT transfers?
2. What are the tradeoffs between storing NFT metadata on-chain vs IPFS vs centralized servers?
3. How does ERC-1155 achieve gas savings compared to ERC-721 for batch operations?
4. Should NFT royalties (EIP-2981) be enforceable on-chain, or remain voluntary for marketplaces?
5. What are the implications of ENS domains being NFTs for domain squatting and trademark issues?

Next Lesson Preview: L19 - Token Lifecycle Management

Coming up next:

- Minting strategies: owner-controlled, public mint, allowlist
- Burning mechanisms and deflationary tokenomics
- Pausing contracts for emergency stops
- Access control patterns: Ownable, AccessControl, multi-sig
- Upgradeability patterns: Transparent proxy, UUPS
- Governance and decentralized control

Preparation:

- Review OpenZeppelin's Ownable and Pausable contracts
- Understand proxy patterns (delegatecall mechanism)
- Explore Gnosis Safe multi-sig wallet