Standard Contracts (ERC-721)

Standard ERC-721 Contract

Reference https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md

Also I have discovered that EIP is an issue number in GIT - with "EIP" meaning Ethereum Improvement Proposal.

Method Name	Const	Params
Approval	event	(address _owner, address _approved, uint256 _tokenId)
ApprovalForAll	event	(address _owner, address _operator, bool _approved)
InterfaceId_ERC165	const	() returns (bytes4)
OwnershipRenounced	event	(address previousOwner)
OwnershipTransferred	event	(address previousOwner, address newOwner)
Transfer	event	(address _from, address _to, uint256 _tokenId)
approve	Tx	(address _to, uint256 _tokenId)
balanceOf	const	(address _owner) returns (uint256)
burn	Tx	(address _owner, uint256 _tokenId)
exists	const	(uint256 _tokenId) returns (bool)
getApproved	const	(uint256 _tokenId) returns (address)
isApprovedForAll	const	(address _owner, address _operator) returns (bool)
mint	Tx	(address _to, uint256 _tokenId, string _tokenURI)
name	const	() returns (string)
owner	const	() returns (address)
ownerOf	const	(uint256 _tokenId) returns (address)
renounceOwnership	Tx	()
safeTransferFrom	Tx	(address _from, address _to, uint256 _tokenId, bytes _data)
setApprovalForAll	Tx	(address _to, bool _approved)
supportsInterface	const	(bytes4 _interfaceId) returns (bool)
symbol	const	() returns (string)

Method Name	Const	Params
tokenByIndex	const	(uint256 _index) returns (uint256)
tokenOfOwnerByIndex	const	(address _owner, uint256 _index) returns (uint256)
tokenURI	const	(uint256 _tokenId) returns (string)
totalSupply	const	() returns (uint256)
transferFrom	Tx	(address _from, address _to, uint256 _tokenId)
transferOwnership	Tx	(address _newOwner)
constructor		()

```
pragma solidity >=0.4.25 <0.6.0;

import "openzeppelin-solidity/contracts/token/ERC721/ERC721Token.sol";
import "openzeppelin-solidity/contracts/ownership/Ownable.sol";

/// @title Demo721
/// @dev Very simple ERC-721 Token example.  This is what Crypto-Kitties is
///  built on.
contract Demo721 is Ownable, ERC721Token {

  string public constant name = "Demo ERC-721"; // solium-disable-line uppercas
  string public constant symbol = "D721"; // solium-disable-line uppercase

  constructor() public ERC721Token(name, symbol) {
  }

  function mint(address _to, uint256 _tokenId, string _tokenURI)
  public onlyOwner {
    super._mint(_to,_tokenId);
    super._setTokenURI(_tokenId,_tokenURI);
  }

  function burn(address _owner, uint256 _tokenId) public onlyOwner {
    super._burn(_owner,_tokenId);
    super._setTokenURI(_tokenId,"");
  }
}
```

```
pragma solidity >=0.4.25 <0.6.0;

/// @title ERC-721 Non-Fungible Token Standard
/// @dev See https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md
///  Note: the ERC-165 identifier for this interface is 0x80ac58cd.
```

```
interface ERC721 {
 /// @dev This emits when ownership of any NFT changes by any mechanism.
 ///  This event emits when NFTs are created (`from` == 0) and destroyed
 ///  (`to` == 0). Exception: during contract creation, any number of NFTs
 ///  may be created and assigned without emitting Transfer. At the time of
 ///  any transfer, the approved address for that NFT (if any) is reset to none
 event Transfer(address indexed _from, address indexed _to,
        uint256 indexed _tokenId);

 /// @dev This emits when the approved address for an NFT is changed or
 ///  reaffirmed. The zero address indicates there is no approved address.
 ///  When a Transfer event emits, this also indicates that the approved
 ///  address for that NFT (if any) is reset to none.
 event Approval(address indexed _owner, address indexed _approved,
        uint256 indexed _tokenId);

 /// @dev This emits when an operator is enabled or disabled for an owner.
 ///  The operator can manage all NFTs of the owner.
 event ApprovalForAll(address indexed _owner, address indexed _operator,
        bool _approved);

 /// @notice Count all NFTs assigned to an owner
 /// @dev NFTs assigned to the zero address are considered invalid, and this
 ///  function throws for queries about the zero address.
 /// @param _owner An address for whom to query the balance
 /// @return The number of NFTs owned by `_owner`, possibly zero
 function balanceOf(address _owner) external view returns (uint256);

 /// @notice Find the owner of an NFT
 /// @dev NFTs assigned to zero address are considered invalid, and queries
 ///  about them do throw.
 /// @param _tokenId The identifier for an NFT
 /// @return The address of the owner of the NFT
 function ownerOf(uint256 _tokenId) external view returns (address);

 /// @notice Transfers the ownership of an NFT from one address to another
 ///  address
 /// @dev Throws unless `msg.sender` is the current owner, an authorized
 ///  operator, or the approved address for this NFT. Throws if `_from` is
 ///  not the current owner. Throws if `_to` is the zero address. Throws if
 ///  `_tokenId` is not a valid NFT. When transfer is complete, this function
 ///  checks if `_to` is a smart contract (code size > 0). If so, it calls
 ///  `onERC721Received` on `_to` and throws if the return value is not
 ///  `bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"))`.
 /// @param _from The current owner of the NFT
 /// @param _to The new owner
 /// @param _tokenId The NFT to transfer
 /// @param data Additional data with no specified format, sent in call
 ///  to `_to`
 function safeTransferFrom(address _from, address _to, uint256 _tokenId,
        bytes data) external payable;
```

```
57
58      /// @notice Transfers the ownership of an NFT from one address to another
59      ///  address
60      /// @dev This works identically to the other function with an extra data
61      ///  parameter, except this function just sets data to "".
62      /// @param _from The current owner of the NFT
63      /// @param _to The new owner
64      /// @param _tokenId The NFT to transfer
65      function safeTransferFrom(address _from, address _to, uint256 _tokenId)
66            external payable;
67
68      /// @notice Transfer ownership of an NFT -- THE CALLER IS RESPONSIBLE
69      ///  TO CONFIRM THAT `_to` IS CAPABLE OF RECEIVING NFTS OR ELSE
70      ///  THEY MAY BE PERMANENTLY LOST
71      /// @dev Throws unless `msg.sender` is the current owner, an authorized
72      ///  operator, or the approved address for this NFT. Throws if `_from`
73      ///  is not the current owner. Throws if `_to` is the zero address.
74      ///  Throws if `_tokenId` is not a valid NFT.
75      /// @param _from The current owner of the NFT
76      /// @param _to The new owner
77      /// @param _tokenId The NFT to transfer
78      function transferFrom(address _from, address _to, uint256 _tokenId)
79            external payable;
80
81      /// @notice Change or reaffirm the approved address for an NFT
82      /// @dev The zero address indicates there is no approved address.
83      ///  Throws unless `msg.sender` is the current NFT owner, or an authorized
84      ///  operator of the current owner.
85      /// @param _approved The new approved NFT controller
86      /// @param _tokenId The NFT to approve
87      function approve(address _approved, uint256 _tokenId) external payable;
88
89      /// @notice Enable or disable approval for a third party ("operator")
90      ///  to manage all of `msg.sender`'s assets
91      /// @dev Emits the ApprovalForAll event. The contract MUST allow
92      ///  multiple operators per owner.
93      /// @param _operator Address to add to the set of authorized operators
94      /// @param _approved True if the operator is approved, false to revoke
95      ///  approval
96      function setApprovalForAll(address _operator, bool _approved) external;
97
98      /// @notice Get the approved address for a single NFT
99      /// @dev Throws if `_tokenId` is not a valid NFT.
100     /// @param _tokenId The NFT to find the approved address for
101     /// @return The approved address for this NFT, or the zero address
102     ///  if there is none
103     function getApproved(uint256 _tokenId) external view returns (address);
104
105     /// @notice Query if an address is an authorized operator for another
106     ///  address
107     /// @param _owner The address that owns the NFTs
```

```
108      /// @param _operator The address that acts on behalf of the owner
109      /// @return True if `_operator` is an approved operator for `_owner`,
110      ///  false otherwise
111      function isApprovedForAll(address _owner, address _operator)
112          external view returns (bool);
113      }
```