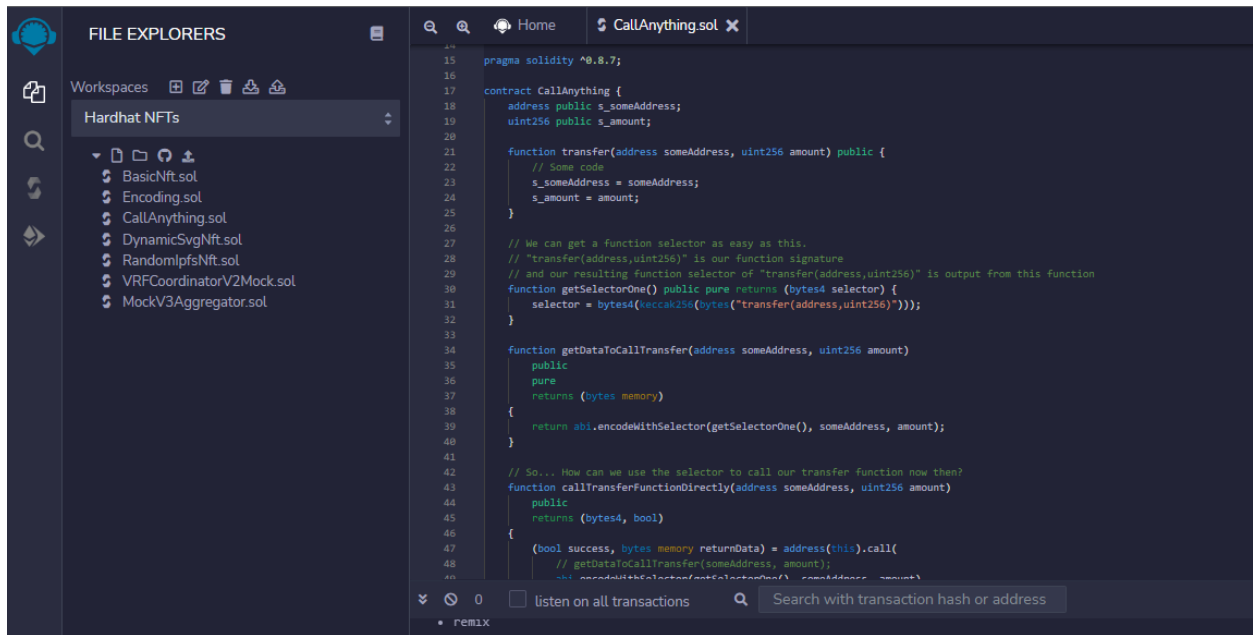
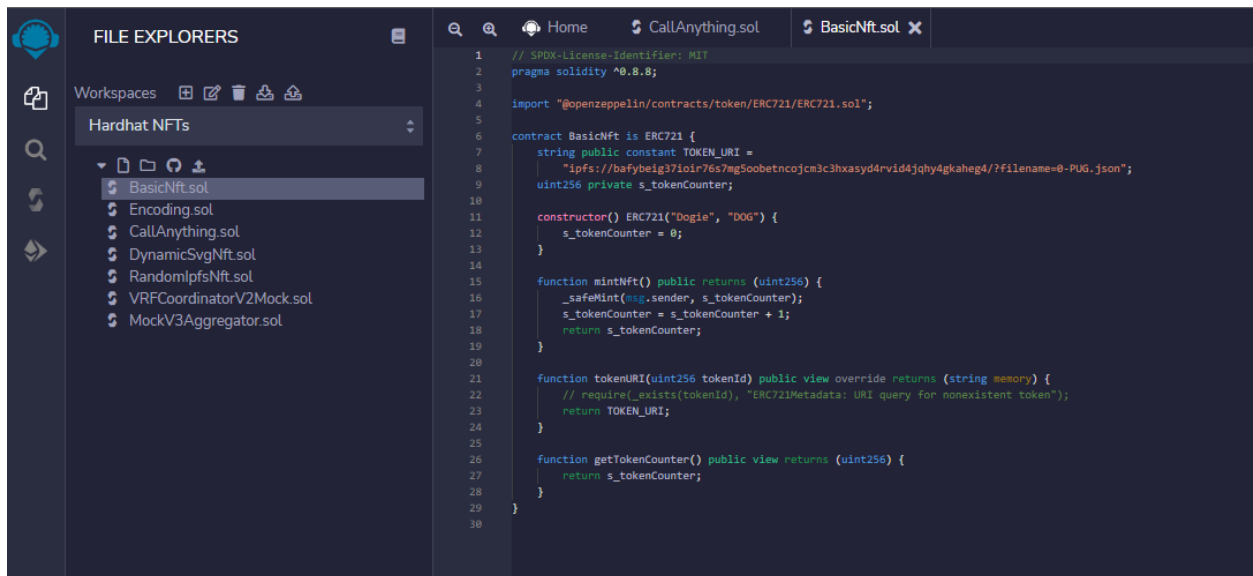


# Hardhat NFTs



The screenshot shows the Hardhat IDE interface. On the left, the 'FILE EXPLORERS' panel displays a list of files under the 'Hardhat NFTs' workspace, including BasicNft.sol, Encoding.sol, CallAnything.sol, DynamicSvgNft.sol, RandomlpfsNft.sol, VRFCoordinatorV2Mock.sol, and MockV3Aggregator.sol. The main editor area shows the code for CallAnything.sol. The code defines a contract with a transfer function, a function to get a selector, and a function to call the transfer function directly. The bottom status bar shows '0' and 'listen on all transactions'.

```
15 pragma solidity ^0.8.7;
16
17 contract CallAnything {
18     address public s_someAddress;
19     uint256 public s_amount;
20
21     function transfer(address someAddress, uint256 amount) public {
22         // Some code
23         s_someAddress = someAddress;
24         s_amount = amount;
25     }
26
27     // We can get a function selector as easy as this.
28     // "transfer(address,uint256)" is our function signature
29     // and our resulting function selector of "transfer(address,uint256)" is output from this function
30     function getSelectorOne() public pure returns (bytes4 selector) {
31         selector = bytes4(keccak256(bytes("transfer(address,uint256)")));
32     }
33
34     function getDataToCallTransfer(address someAddress, uint256 amount)
35     public
36     pure
37     returns (bytes memory)
38     {
39         return abi.encodeWithSelector(getSelectorOne(), someAddress, amount);
40     }
41
42     // So... How can we use the selector to call our transfer function now then?
43     function callTransferFunctionDirectly(address someAddress, uint256 amount)
44     public
45     returns (bytes4, bool)
46     {
47         (bool success, bytes memory returnData) = address(this).call(
48             // getDataToCallTransfer(someAddress, amount);
49             abi.encodeWithSelector(getSelectorOne(), someAddress, amount)
50         );
51     }
52 }
```



The screenshot shows the Hardhat IDE interface. On the left, the 'FILE EXPLORERS' panel displays a list of files under the 'Hardhat NFTs' workspace, including BasicNft.sol, Encoding.sol, CallAnything.sol, DynamicSvgNft.sol, RandomlpfsNft.sol, VRFCoordinatorV2Mock.sol, and MockV3Aggregator.sol. The main editor area shows the code for BasicNft.sol. The code defines a contract that inherits from ERC721, with a constructor, a mintNft function, a tokenURI function, and a getTokenCounter function.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.8;
3
4 import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
5
6 contract BasicNft is ERC721 {
7     string public constant TOKEN_URI =
8         "ipfs://bafybeig37i0lr76s7mg5oobetncojcm3c3hxasydd4rvid4jqhy4gkaheg4/?filename=0-PUG.json";
9     uint256 private s_tokenCounter;
10
11     constructor() ERC721("Dogie", "DOG") {
12         s_tokenCounter = 0;
13     }
14
15     function mintNft() public returns (uint256) {
16         _safeMint(msg.sender, s_tokenCounter);
17         s_tokenCounter = s_tokenCounter + 1;
18         return s_tokenCounter;
19     }
20
21     function tokenURI(uint256 tokenId) public view override returns (string memory) {
22         // require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");
23         return TOKEN_URI;
24     }
25
26     function getTokenCounter() public view returns (uint256) {
27         return s_tokenCounter;
28     }
29 }
30
```

FILE EXPLORERS

Workspaces

Hardhat NFTs

BasicNft.sol

Encoding.sol

CallAnything.sol

DynamicSvgNft.sol

RandomlpfsNft.sol

VRFCoordinatorV2Mock.sol

MockV3Aggregator.sol

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.8;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";
import "base64.sol/base64.sol";
import "hardhat/console.sol";

contract DynamicSvgNft is ERC721, Ownable {
    uint256 private s_tokenCounter;
    string private s_lowImageURI;
    string private s_highImageURI;

    mapping(uint256 => int256) private s_tokenIdToHighValues;
    AggregatorV3Interface internal immutable i_priceFeed;
    event CreatedNFT(uint256 indexed tokenId, int256 highValue);

    constructor(
        address priceFeedAddress,
        string memory lowSvg,
        string memory highSvg
    ) ERC721("Dynamic SVG NFT", "DSN") {
        s_tokenCounter = 0;
        i_priceFeed = AggregatorV3Interface(priceFeedAddress);
        // setLowSVG(lowSvg);
        // setHighSVG(highSvg);
        s_lowImageURI = svgToImageURI(lowSvg);
        s_highImageURI = svgToImageURI(highSvg);
    }

    // function setLowURI(string memory svgLowURI) public onlyOwner {
    //     s_lowImageURI = svgLowURI;
    // }
}
```

FILE EXPLORERS

Workspaces

Hardhat NFTs

BasicNft.sol

Encoding.sol

CallAnything.sol

DynamicSvgNft.sol

RandomlpfsNft.sol

VRFCoordinatorV2Mock.sol

MockV3Aggregator.sol

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

```
// Or put another way, we ABI encode it.
function encodeNumber() public pure returns (bytes memory) {
    bytes memory number = abi.encode(1);
    return number;
}

// You'd use this to make calls to contracts
function encodeString() public pure returns (bytes memory) {
    bytes memory someString = abi.encode("some string");
    return someString;
}

// https://forum.openzeppelin.com/t/difference-between-abi-encodepacked-string-and-bytes-string/11837
// encodePacked
// This is great if you want to save space, not good for calling functions.
// You can sort of think of it as a compressor for the massive bytes object above.
function encodeStringPacked() public pure returns (bytes memory) {
    bytes memory someString = abi.encodePacked("some string");
    return someString;
}

// This is just type casting to string
// It's slightly different from below, and they have different gas costs
function encodeStringBytes() public pure returns (bytes memory) {
    bytes memory someString = bytes("some string");
    return someString;
}

function decodeString() public pure returns (string memory) {
    string memory someString = abi.decode(encodeString(), (string));
    return someString;
}

function multiEncode() public pure returns (bytes memory) {
    bytes memory someString = abi.encode("some string", "it's bigger!");
}
```

0

listen on all transactions

Search with transaction hash or address

remix