

# LESSON 15

NEXTJS NFT MARKETPLACE

---

Hikmah Nisya - 1103184094

Radzis Araaf Jaya Jamaludin - 1103184234

Raudhatul Rafiqah Assyahiddini - 1103180225

```
> node_modules
{...} .solhint.json
📄 .solhintignore
📄 hardhat-config.js
📄 helper-hardhat-config.js
📄 package-lock.json
```

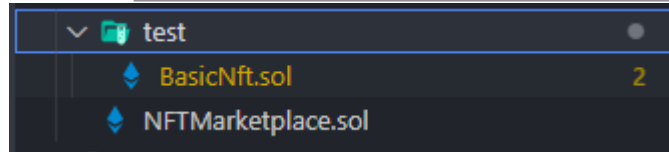
```
+ chai@4.3.6
+ ethereum-waffle@3.4.4
+ ethers@5.6.9
+ hardhat-gas-reporter@1.0.8
+ hardhat-deploy@0.11.11
+ prettier@2.7.1
+ dotenv@16.0.1
+ solidity-coverage@0.7.21
+ prettier-plugin-solidity@1.0.0-beta.19
+ solhint@3.3.7
+ hardhat-deploy-ethers@0.3.0-beta.13 (as @nomiclabs/hardhat-ethers)
+ hardhat@2.9.9
+ @nomiclabs/hardhat-waffle@2.0.3
+ @nomiclabs/hardhat-etherscan@3.1.0
+ hardhat-contract-sizer@2.6.1
added 1979 packages from 878 contributors in 455.988s

163 packages are looking for funding
```

# Part I: NFT Marketplace Contracts (Setup)

---

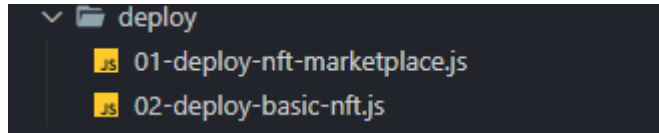
# Part I: NFT Marketplace Contracts



```
NFTMarketplace.sol x
contracts > NFTMarketplace.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
3
4 import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
5 import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
6
7 error NFTMarketplace__PriceMustBeAboveZero();
8 error NFTMarketplace__NotApprovedForMarketplace();
9 error NFTMarketplace__AlreadyListed(address nftAddress, uint256 tokenId);
10 error NFTMarketplace__NotOwner();
11 error NFTMarketplace__NotListed(address nftAddress, uint256 tokenId);
12 error NFTMarketplace__PriceNotMet(address nftAddress, uint256 tokenId, uint256 price);
13 error NFTMarketplace__NoProceeds();
14
15 contract NFTMarketplace is ReentrancyGuard {
16     struct Listing {
17         uint256 price;
18         address seller;
19     }
20
21     mapping(address => mapping(uint256 => Listing)) private s_listings;
22     mapping(address => uint256) private s_proceeds;
23
24     modifier notListed(
25         address nftAddress,
26         uint256 tokenId,
27         address owner
28     ) {
29         Listing memory listing = s_listings[nftAddress][tokenId];
```

```
BasicNft.sol 2 x
contracts > test > BasicNft.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
3
4 import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
5
6 contract BasicNft is ERC721 {
7     string public constant TOKEN_URI = "ipfs://bafybeig37ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?";
8     uint256 private s_tokenCounter;
9
10    event DogMinted(uint256 indexed tokenId);
11
12    constructor() ERC721("Dogie", "DOG") {
13        s_tokenCounter = 0;
14    }
15
16    function mintNft() public {
17        _safeMint(msg.sender, s_tokenCounter);
18        emit DogMinted(s_tokenCounter);
19        s_tokenCounter = s_tokenCounter + 1;
20    }
21
22    function tokenURI(uint256 tokenId) public view override returns (string memory) {
23        require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");
24        return TOKEN_URI;
25    }
26
27    function getTokenCounter() public view returns (uint256) {
28        return s_tokenCounter;
29    }
}
```

# Part I: NFT Marketplace Contracts (deploy)



```
01-deploy-nft-marketplace.js X
deploy > 01-deploy-nft-marketplace.js > ...
1  const { network } = require("hardhat");
2  const { developmentChains } = require("../helper-hardhat-config");
3  const { verify } = require("../utils/verify");
4
5  module.exports = async ({ getNamedAccounts, deployments }) => {
6    const { deploy, log } = deployments;
7    const { deployer } = await getNamedAccounts();
8
9    args = [];
10
11    const nftMarketplace = await deploy("NFTMarketplace", {
12      from: deployer,
13      args: args,
14      log: true,
15      waitConfirmations: network.config.blockConfirmation || 1,
16    });
17
18    if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_K
19      console.log("verifying");
20      await verify(nftMarketplace.address, args);
21    }
22  };
23
24  module.exports.tags = ["all", "nftmarketplace"];
25

02-deploy-basic-nft.js X
deploy > 02-deploy-basic-nft.js > <unknown> > exports
1  const { network } = require("hardhat");
2  const { developmentChains } = require("../helper-hardhat-config");
3  const { verify } = require("../utils/verify");
4
5  module.exports = async ({ getNamedAccounts, deployments }) => {
6    const { deploy, log } = deployments;
7    const { deployer } = await getNamedAccounts();
8
9    const args = [];
10    const basicNft = await deploy("BasicNft", {
11      from: deployer,
12      args: args,
13      log: true,
14      waitConfirmations: network.config.blockConfirmation || 1,
15    });
16
17    if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_K
18      console.log("verifying");
19      await verify(nftMarketplace.address, args);
20    }
21  };
22
23  module.exports.tags = ["all", "basicNft"];
24
```

```
1  const { assert, expect } = require('chai');
2  const { network, deployments, ethers } = require("hardhat");
3  const { developmentChains } = require("../helper-hardhat-config");
4
5  !developmentChains.includes(network.name)
6    ? describe.skip
7    : describe("Nft Marketplace Unit Tests", function () {
8      let nftMarketplace, nftMarketplaceContract, basicNft, basicNftContract;
9      const PRICE = ethers.utils.parseEther("0.1");
10     const TOKEN_ID = 0;
11
12     beforeEach(async () => {
13       accounts = await ethers.getSigners(); // could also do with getNamedAccounts
14       deployer = accounts[0];
15       user = accounts[1];
16       await deployments.fixture(["all"]);
17       nftMarketplaceContract = await ethers.getContract("NftMarketplace");
18       nftMarketplace = nftMarketplaceContract.connect(deployer);
19       basicNftContract = await ethers.getContract("BasicNft");
20       basicNft = await basicNftContract.connect(deployer);
21       await basicNft.mintNft();
22       await basicNft.approve(nftMarketplaceContract.address, TOKEN_ID);
23     });
24
```

# Part I: NFT Marketplace Contracts (unit test)

---