

## Graph-nft-marketplace-fcc

- src/mapping.ts

```
graph-nft-marketplace-fcc > src > TS mapping.ts > ...
1 import { BigInt, Address } from "@graphprotocol/graph-ts"
2 import {
3   NftMarketplace,
4   ItemBought as ItemBoughtEvent,
5   ItemCanceled as ItemCanceledEvent,
6   ItemListed as ItemListedEvent,
7 } from "../generated/NftMarketplace/NftMarketplace"
8 import { ItemListed, ActiveItem, ItemBought, ItemCanceled } from "../generated/schema"
9
10 export function handleItemListed(event: ItemListedEvent): void {
11   let itemListed = ItemListed.load(
12     getIdFromEventParams(event.params.tokenId, event.params.nftAddress)
13   )
14   let activeItem = ActiveItem.load(
15     getIdFromEventParams(event.params.tokenId, event.params.nftAddress)
16   )
17   if (!itemListed) {
18     itemListed = new ItemListed(
19       getIdFromEventParams(event.params.tokenId, event.params.nftAddress)
20     )
21   }
22   if (!activeItem) {
23     activeItem = new ActiveItem(
24       getIdFromEventParams(event.params.tokenId, event.params.nftAddress)
25     )
26   }
27   itemListed.seller = event.params.seller
28   activeItem.seller = event.params.seller
29
30   itemListed.nftAddress = event.params.nftAddress
31   activeItem.nftAddress = event.params.nftAddress
32
33   itemListed.tokenId = event.params.tokenId
```

- generated/schema.ts

```
1 // THIS IS AN AUTOGENERATED FILE. DO NOT EDIT THIS FILE DIRECTLY.
2
3 import {
4   TypedMap,
5   Entity,
6   Value,
7   ValueKind,
8   store,
9   Bytes,
10  BigInt,
11  BigDecimal
12 } from "@graphprotocol/graph-ts";
13
14 export class ActiveItem extends Entity {
15   constructor(id: string) {
16     super();
17     this.set("id", Value.fromString(id));
18
19     this.set("buyer", Value.fromBytes(Bytes.empty()));
20     this.set("seller", Value.fromBytes(Bytes.empty()));
21     this.set("nftAddress", Value.fromBytes(Bytes.empty()));
22     this.set("tokenId", Value.fromBigInt(BigInt.zero()));
23   }
24
25   save(): void {
26     let id = this.get("id");
27     assert(id != null, "Cannot save ActiveItem entity without an ID");
28     if (id) {
29       assert(
30         id.kind == ValueKind.STRING,
31         `Entities of type ActiveItem must have an ID of type String but the id '${id.displayData()}' is of type ${id.kind}`
32       );
33       store.set("ActiveItem", id.toString(), this);
34     }
35   }
36 }
```

- NftMarketplace/NftMarketplace.ts

```
graph-nft-marketplace-fcc > generated > NftMarketplace > TS NftMarketplace.ts > ...
1  // THIS IS AN AUTOGENERATED FILE. DO NOT EDIT THIS FILE DIRECTLY.
2
3  import {
4    ethereum,
5    JSONValue,
6    TypedMap,
7    Entity,
8    Bytes,
9    Address,
10   BigInt
11 } from "@graphprotocol/graph-ts";
12
13 export class ItemBought extends ethereum.Event {
14   get params(): ItemBought__Params {
15     return new ItemBought__Params(this);
16   }
17 }
18
19 export class ItemBought__Params {
20   _event: ItemBought;
21
22   constructor(event: ItemBought) {
23     this._event = event;
24   }
25
26   get buyer(): Address {
27     return this._event.parameters[0].value.toAddress();
28   }
29
30   get nftAddress(): Address {
31     return this._event.parameters[1].value.toAddress();
32   }
33 }
```

Activate Windows  
Go to Settings to activate Windows.

## Hardhat-nft-marketplace-fcc

- contract/sublessor/ReentrantVulnerable.sol

```
ReentrantVulnerable.sol X
hardhat-nft-marketplace-fcc > contracts > sublessor > ReentrantVulnerable.sol
32
33 contract ReentrantVulnerable {
34     mapping(address => uint256) public balances;
35
36     function deposit() public payable {
37         balances[msg.sender] += msg.value;
38     }
39
40     function withdraw() public {
41         uint256 bal = balances[msg.sender];
42         require(bal > 0);
43
44         (bool sent, ) = msg.sender.call{value: bal}("");
45         require(sent, "Failed to send Ether");
46
47         balances[msg.sender] = 0;
48     }
49
50     // Helper function to check the balance of this contract
51     function getBalance() public view returns (uint256) {
52         return address(this).balance;
53     }
54 }
55
56 contract Attack {
57     ReentrantVulnerable public reentrantVulnerable;
58
59     constructor(address _reentrantVulnerableAddress) {
60         reentrantVulnerable = ReentrantVulnerable(_reentrantVulnerableAddress);
61     }
62
63     // Fallback is called when EtherStore sends Ether to this contract.
64     fallback() external payable {
```

- contract/test/BasicNft.sol

```
BasicNft.sol X
hardhat-nft-marketplace-fcc > 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 =
8         "ipfs://bafybeig37ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?filename=0-PUG.json";
9     uint256 private s_tokenCounter;
10
11     event DogMinted(uint256 indexed tokenId);
12
13     constructor() ERC721("Dogie", "DOG") {
14         s_tokenCounter = 0;
15     }
16
17     function mintNft() public {
18         _safeMint(msg.sender, s_tokenCounter);
19         emit DogMinted(s_tokenCounter);
20         s_tokenCounter = s_tokenCounter + 1;
21     }
22
23     function tokenURI(uint256 tokenId) public view override returns (string memory) {
24         require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");
25         return TOKEN_URI;
26     }
27
28     function getTokenCounter() public view returns (uint256) {
29         return s_tokenCounter;
30     }
31 }
```

- contract/test/BasicNftTwo.sol

```
hardhat-nft-marketplace-fcc > contracts > test > BasicNftTwo.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
3
4 import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
5
6 contract BasicNftTwo is ERC721 {
7     string public constant TOKEN_URI = "ipfs://QmdryoExpqEQQgJPoruwGJyZmz6SqV4FRTX1i73CT3iXn";
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    }
30 }
```

- contract/NftMarketplace.sol

```
10 error PriceNotMet(address nftAddress, uint256 tokenId, uint256 price);
11 error ItemNotForSale(address nftAddress, uint256 tokenId);
12 error NotListed(address nftAddress, uint256 tokenId);
13 error AlreadyListed(address nftAddress, uint256 tokenId);
14 error NoProceeds();
15 error NotOwner();
16 error NotApprovedForMarketplace();
17 error PriceMustBeAboveZero();
18
19 contract NftMarketplace is ReentrancyGuard {
20     struct Listing {
21         uint256 price;
22         address seller;
23     }
24
25     event ItemListed(
26         address indexed seller,
27         address indexed nftAddress,
28         uint256 indexed tokenId,
29         uint256 price
30     );
31
32     event ItemCanceled(
33         address indexed seller,
34         address indexed nftAddress,
35         uint256 indexed tokenId
36     );
37
38     event ItemBought(
39         address indexed buyer,
40         address indexed nftAddress,
```

- deploy/01-deploy-nft-marketplace.js

```
JS 01-deploy-nft-marketplace.js X
hardhat-nft-marketplace-fcc > deploy > JS 01-deploy-nft-marketplace.js > ...
1  const { network } = require("hardhat")
2  const { developmentChains, VERIFICATION_BLOCK_CONFIRMATIONS } = 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    const waitBlockConfirmations = developmentChains.includes(network.name)
9      ? 1
10     : VERIFICATION_BLOCK_CONFIRMATIONS
11
12    log("-----")
13    const arguments = []
14    const nftMarketplace = await deploy("NftMarketplace", {
15      from: deployer,
16      args: arguments,
17      log: true,
18      waitConfirmations: waitBlockConfirmations,
19    })
20
21    // Verify the deployment
22    if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_KEY) {
23      log("Verifying...")
24      await verify(nftMarketplace.address, arguments)
25    }
26    log("-----")
27  }
28
29  module.exports.tags = ["all", "nftmarketplace"]
```

- deploy/deploy-basic-nft.js

```
JS 02-deploy-basic-nft.js X
hardhat-nft-marketplace-fcc > deploy > JS 02-deploy-basic-nft.js > ...
1  const { network } = require("hardhat")
2  const { developmentChains, VERIFICATION_BLOCK_CONFIRMATIONS } = 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    const waitBlockConfirmations = developmentChains.includes(network.name)
9      ? 1
10     : VERIFICATION_BLOCK_CONFIRMATIONS
11
12    log("-----")
13    const args = []
14    const basicNft = await deploy("BasicNft", {
15      from: deployer,
16      args: args,
17      log: true,
18      waitConfirmations: waitBlockConfirmations,
19    })
20
21    const basicNftTwo = await deploy("BasicNftTwo", {
22      from: deployer,
23      args: args,
24      log: true,
25      waitConfirmations: waitBlockConfirmations,
26    })
27
28    // Verify the deployment
29    if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_KEY) {
30      log("Verifying...")
31      await verify(basicNft.address, args)
32      await verify(basicNftTwo.address, args)
33    }
34  }
```

Activate Windows  
Go to Settings to activate Windows.

- deploy/update-front-end.js

```
JS 03-update-front-end.js X
hardhat-nft-marketplace-fcc > deploy > JS 03-update-front-end.js > ...

1  const {
2    frontEndContractsFile,
3    frontEndContractsFile2,
4    frontEndAbiLocation,
5    frontEndAbiLocation2,
6  } = require("../helper-hardhat-config")
7  require("dotenv").config()
8  const fs = require("fs")
9  const { network } = require("hardhat")
10
11  module.exports = async () => {
12    if (process.env.UPDATE_FRONT_END) {
13      console.log("Writing to front end...")
14      await updateContractAddresses()
15      await updateAbi()
16      console.log("Front end written!")
17    }
18  }
19
20  async function updateAbi() {
21    const nftMarketplace = await ethers.getContract("NftMarketplace")
22    fs.writeFileSync(
23      `${frontEndAbiLocation}NftMarketplace.json`,
24      nftMarketplace.interface.format(ethers.utils.FormatTypes.json)
25    )
26    fs.writeFileSync(
27      `${frontEndAbiLocation2}NftMarketplace.json`,
28      nftMarketplace.interface.format(ethers.utils.FormatTypes.json)
29    )
30
31    const basicNft = await ethers.getContract("BasicNft")
32    fs.writeFileSync(
33      `${frontEndAbiLocation}BasicNft.json`,
34      basicNft.interface.format(ethers.utils.FormatTypes.json)
35    )
36  }
37}
```

- script/buy-item.js

```
JS buy-item.js X
hardhat-nft-marketplace-fcc > scripts > JS buy-item.js > ...
1  const { ethers, network } = require("hardhat")
2  const { moveBlocks } = require("../utils/move-blocks")
3
4  const TOKEN_ID = 1
5
6  async function buyItem() {
7    const nftMarketplace = await ethers.getContract("NftMarketplace")
8    const basicNft = await ethers.getContract("BasicNft")
9    const listing = await nftMarketplace.getListing(basicNft.address, TOKEN_ID)
10   const price = listing.price.toString()
11   const tx = await nftMarketplace.buyItem(basicNft.address, TOKEN_ID, { value: price })
12   await tx.wait(1)
13   console.log("NFT Bought!")
14   if ((network.config.chainId === "31337")) {
15     await moveBlocks(2, {sleepAmount: 1000})
16   }
17 }
18
19 buyItem()
20   .then(() => process.exit(0))
21   .catch((error) => {
22     console.error(error)
23     process.exit(1)
24   })
```

- scripts/cancel-item.js

```
JS cancel-item.js X
hardhat-nft-marketplace-fcc > scripts > JS cancel-item.js > ...
1  const { ethers, network } = require("hardhat")
2  const { moveBlocks } = require("../utils/move-blocks")
3
4  const TOKEN_ID = 0
5
6  async function mintAndList() {
7      const nftMarketplace = await ethers.getContract("NftMarketplace")
8      const basicNft = await ethers.getContract("BasicNft")
9      const tx = await nftMarketplace.cancellisting(basicNft.address, TOKEN_ID)
10     await tx.wait(1)
11     console.log("NFT Canceled!")
12     if ((network.config.chainId = "31337")) {
13         await moveBlocks(2, {sleepAmount = 1000})
14     }
15 }
16
17 mintAndList()
18     .then(() => process.exit(0))
19     .catch((error) => {
20         console.error(error)
21         process.exit(1)
22     })
```

- scripts/mine.js

```
JS mine.js X
hardhat-nft-marketplace-fcc > scripts > JS mine.js > ...
1  const { moveBlocks } = require("../utils/move-blocks")
2
3  const BLOCKS = 5
4
5  async function mine() {
6      await moveBlocks(BLOCKS)
7  }
8
9  mine()
10     .then(() => process.exit(0))
11     .catch((error) => {
12         console.error(error)
13         process.exit(1)
14     })
```

- scripts/mint-and-list-item.js

```
JS mint-and-list-item.js X
hardhat-nft-marketplace-fcc > scripts > JS mint-and-list-item.js > ...
1  const { ethers, network } = require("hardhat")
2  const { moveBlocks } = require("../utils/move-blocks")
3
4  const PRICE = ethers.utils.parseEther("0.1")
5
6  async function mintAndList() {
7    const nftMarketplace = await ethers.getContract("NftMarketplace")
8    const randomNumber = Math.floor(Math.random() * 2)
9    let basicNft
10   if (randomNumber == 1) {
11     basicNft = await ethers.getContract("BasicNftTwo")
12   } else {
13     basicNft = await ethers.getContract("BasicNft")
14   }
15   console.log("Minting NFT...")
16   const mintTx = await basicNft.mintNft()
17   const mintTxReceipt = await mintTx.wait(1)
18   const tokenId = mintTxReceipt.events[0].args.tokenId
19   console.log("Approving NFT...")
20   const approvalTx = await basicNft.approve(nftMarketplace.address, tokenId)
21   await approvalTx.wait(1)
22   console.log("Listing NFT...")
23   const tx = await nftMarketplace.listItem(basicNft.address, tokenId, PRICE)
24   await tx.wait(1)
25   console.log("NFT Listed!")
26   if (network.config.chainId == 31337) {
27     // Moralis has a hard time if you move more than 1 at once!
28     await moveBlocks(1, (sleepAmount = 1000))
29   }
30 }
31
32 mintAndList()
33   .then(() => process.exit(0))
34   .catch((error) => {
35     console.error(error)
36     process.exit(1)
37   })
38 }
```

Activate Windows  
Go to Settings to activate Windows.

-scripts/mint.js

```
JS mint.js X
hardhat-nft-marketplace-fcc > scripts > JS mint.js > ...
1  const { ethers, network } = require("hardhat")
2  const { moveBlocks } = require("../utils/move-blocks")
3
4  const PRICE = ethers.utils.parseEther("0.1")
5
6  async function mintAndList() {
7    const basicNft = await ethers.getContract("BasicNftTwo")
8    console.log("Minting NFT...")
9    const mintTx = await basicNft.mintNft()
10   const mintTxReceipt = await mintTx.wait(1)
11   console.log(
12     `Minted tokenId ${mintTxReceipt.events[0].args.tokenId.toString()} from contract: ${
13       basicNft.address
14     }`
15   )
16   if (network.config.chainId == 31337) {
17     // Moralis has a hard time if you move more than 1 block!
18     await moveBlocks(2, (sleepAmount = 1000))
19   }
20 }
21
22 mintAndList()
23   .then(() => process.exit(0))
24   .catch((error) => {
25     console.error(error)
26     process.exit(1)
27   })
28 }
```



- test/unit/NftMarketplace.test.js

```
JS NftMarketplace.test.js X
hardhat-nft-marketplace-fcc > test > unit > JS NftMarketplace.test.js > ...
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
25     describe("listItem", function () {
26       it("emits an event after listing an item", async function () {
27         expect(await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)).to.emit(
28           "ItemListed"
29         )
30       })
31       it("exclusively items that haven't been listed", async function () {
32         await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
33         const error = `Already listed("${basicNft.address}", ${TOKEN_ID})`
```

- utils/move-blocks.js

```
JS move-blocks.js X
hardhat-nft-marketplace-fcc > utils > JS move-blocks.js > ...
1  const { network } = require("hardhat")
2
3  function sleep(timeInMs) {
4    return new Promise((resolve) => setTimeout(resolve, timeInMs))
5  }
6
7  async function moveBlocks(amount, sleepAmount = 0) {
8    console.log("Moving blocks...")
9    for (let index = 0; index < amount; index++) {
10     await network.provider.request({
11       method: "evm_mine",
12       params: [],
13     })
14     if (sleepAmount) {
15       console.log(`Sleeping for ${sleepAmount}`)
16       await sleep(sleepAmount)
17     }
18   }
19   console.log(`Moved ${amount} blocks`)
20 }
21
22 module.exports = {
23   moveBlocks,
24   sleep,
25 }
```

- utils/verify.js

```
JS verify.js X
hardhat-nft-marketplace-fcc > utils > JS verify.js > ...
1  const { run, network } = require("hardhat")
2  const { networkConfig } = require("../helper-hardhat-config")
3
4  const verify = async (contractAddress, args) => {
5      console.log("Verifying contract...")
6      try {
7          await run("verify:verify", {
8              address: contractAddress,
9              constructorArguments: args,
10         })
11     } catch (e) {
12         if (e.message.toLowerCase().includes("already verified")) {
13             console.log("Already verified!")
14         } else {
15             console.log(e)
16         }
17     }
18 }
19
20 module.exports = {
21     verify,
22 }
```

## nextjs-nft-marketplace-moralis-fcc

- pages/\_app.js

```
JS _app.js X
nextjs-nft-marketplace-moralis-fcc > pages > JS _app.js > ...
1  import "../styles/globals.css"
2  import { MoralisProvider } from "react-moralis"
3  import Header from "../components/Header"
4  import Head from "next/head"
5  import { NotificationProvider } from "web3uikit"
6
7  const APP_ID = process.env.NEXT_PUBLIC_APP_ID
8  const SERVER_URL = process.env.NEXT_PUBLIC_SERVER_URL
9
10 function MyApp({ Component, pageProps }) {
11   return (
12     <div>
13       <Head>
14         <title>NFT Marketplace</title>
15         <meta name="description" content="NFT Marketplace" />
16         <link rel="icon" href="/favicon.ico" />
17       </Head>
18       <MoralisProvider appId={APP_ID} serverUrl={SERVER_URL}>
19         <NotificationProvider>
20           <Header />
21           <Component {...pageProps} />
22         </NotificationProvider>
23       </MoralisProvider>
24     </div>
25   )
26 }
27
28 export default MyApp
```

- pages/index.js

```
JS index.js X
nextjs-nft-marketplace-moralis-fcc > pages > JS index.js > ...
1  import Image from "next/image"
2  import styles from "../styles/Home.module.css"
3  import { useMoralisQuery, useMoralis } from "react-moralis"
4  import NFTBox from "../components/NFTBox"
5
6  export default function Home() {
7    const { isWeb3Enabled } = useMoralis()
8    const { data: listedNfts, isFetching: fetchingListedNfts } = useMoralisQuery(
9      // TableName
10     // Function for the query
11     "ActiveItem",
12     (query) => query.limit(10).descending("tokenId")
13   )
14   console.log(listedNfts)
15
16   return (
17     <div className="container mx-auto">
18       <h1 className="py-4 px-4 font-bold text-2xl">Recently Listed</h1>
19       <div className="flex flex-wrap">
20         {isWeb3Enabled ? (
21           fetchingListedNfts ? (
22             <div>Loading...</div>
23           ) : (
24             listedNfts.map((nft) => {
25               console.log(nft.attributes)
26               const { price, nftAddress, tokenId, marketplaceAddress, seller } =
27                 nft.attributes
28               return (
29                 <div>
30                   <NFTBox
31                     price={price}
32                     nftAddress={nftAddress}
33                     tokenId={tokenId}

```

- pages/sell-nft.js

```
JS sell-nft.js X
nextjs-nft-marketplace-moralis-fcc > pages > JS sell-nft.js > ...
1 import styles from "../styles/Home.module.css"
2 import { Form, useNotification, Button } from "web3uiikit"
3 import { useMoralis, useWeb3Contract } from "react-moralis"
4 import { ethers } from "ethers"
5 import nftAbi from "../constants/BasicNft.json"
6 import nftMarketplaceAbi from "../constants/NftMarketplace.json"
7 import networkMapping from "../constants/networkMapping.json"
8 import { useEffect, useState } from "react"
9
10 export default function Home() {
11   const { chainId, account, isWeb3Enabled } = useMoralis()
12   const chainString = chainId ? parseInt(chainId).toString() : "31337"
13   const marketplaceAddress = networkMapping[chainString].NftMarketplace[0]
14   const dispatch = useNotification()
15   const [proceeds, setProceeds] = useState("0")
16
17   const { runContractFunction } = useWeb3Contract()
18
19   async function approveAndList(data) {
20     console.log("Approving...")
21     const nftAddress = data.data[0].inputResult
22     const tokenId = data.data[1].inputResult
23     const price = ethers.utils.parseUnits(data.data[2].inputResult, "ether").toString()
24
25     const approveOptions = {
26       abi: nftAbi,
27       contractAddress: nftAddress,
28       functionName: "approve",
29       params: {
30         to: marketplaceAddress,
31         tokenId: tokenId,
32       },
33     },
34   }
35 }
```

Activate Windows  
Go to Settings to activate Windows.

- cloudFunctions/updateActiveItems.js

```
JS updateActiveItems.js X
nextjs-nft-marketplace-moralis-fcc > cloudFunctions > JS updateActiveItems.js > ...
1 Moralis.Cloud.afterSave("Item Listed", async (request) => {
2   const confirmed = request.object.get("confirmed")
3   const logger = Moralis.Cloud.getLogger()
4   logger.info("Looking for confirmed TX...")
5   if (confirmed) {
6     logger.info("Found item!")
7     const ActiveItem = Moralis.Object.extend("ActiveItem")
8
9     // In case of listing update, search for already listed ActiveItem and delete
10    const query = new Moralis.Query(ActiveItem)
11    query.equalTo("nftAddress", request.object.get("nftAddress"))
12    query.equalTo("tokenId", request.object.get("tokenId"))
13    query.equalTo("marketplaceAddress", request.object.get("address"))
14    query.equalTo("seller", request.object.get("seller"))
15    logger.info(`Marketplace | Query: ${JSON.stringify(query)}`)
16    const alreadyListedItem = await query.first()
17    console.log(`alreadyListedItem ${JSON.stringify(alreadyListedItem)}`)
18    if (alreadyListedItem) {
19      logger.info(`Deleting ${alreadyListedItem.id}`)
20      await alreadyListedItem.destroy()
21      logger.info(
22        `Deleted item with tokenId ${request.object.get(
23          "tokenId"
24        )} at address ${request.object.get(
25          "address"
26        )} since the listing is being updated.`
27      )
28    }
29  }
30
31  // Add new ActiveItem
32  const activeItem = new ActiveItem()
33  activeItem.set("marketplaceAddress", request.object.get("address"))
34  activeItem.set("nftAddress", request.object.get("nftAddress"))
35 }
```

Activate Windows  
Go to Settings to activate Windows.

- components/Header.js

```
JS Header.js X
nextjs-nft-marketplace-moralis-fcc > components > JS Header.js > ...
1 import { ConnectButton } from "web3uikit"
2 import Link from "next/link"
3
4 export default function Header() {
5   return (
6     <nav className="p-5 border-b-2 flex flex-row justify-between items-center">
7       <h1 className="py-4 px-4 font-bold text-3xl">NFT Marketplace</h1>
8       <div className="flex flex-row items-center">
9         <Link href="/">
10           <a className="mr-4 p-6">Home</a>
11         </Link>
12         <Link href="/sell-nft">
13           <a className="mr-4 p-6">Sell NFT</a>
14         </Link>
15         <ConnectButton moralisAuth={false} />
16       </div>
17     </nav>
18   )
19 }
```

- component/NFTBox.js

```
JS NFTBox.js X
nextjs-nft-marketplace-moralis-fcc > components > JS NFTBox.js > ...
1 import { useState, useEffect } from "react"
2 import { useWeb3Contract, useMoralis } from "react-moralis"
3 import nftMarketplaceAbi from "../constants/NftMarketplace.json"
4 import nftAbi from "../constants/BasicNft.json"
5 import Image from "next/image"
6 import { Card, useNotification } from "web3uikit"
7 import { ethers } from "ethers"
8 import UpdateListingModal from "../UpdateListingModal"
9
10 const truncateStr = (fullStr, strLen) => {
11   if (fullStr.length <= strLen) return fullStr
12
13   const separator = "..."
14   const separatorLength = separator.length
15   const charsToShow = strLen - separatorLength
16   const frontChars = Math.ceil(charsToShow / 2)
17   const backChars = Math.floor(charsToShow / 2)
18   return (
19     fullStr.substring(0, frontChars) +
20     separator +
21     fullStr.substring(fullStr.length - backChars)
22   )
23 }
24
25 export default function NFTBox({ price, nftAddress, tokenId, marketplaceAddress, seller }) {
26   const { isWeb3Enabled, account } = useMoralis()
27   const [imageURI, setImageURI] = useState("")
28   const [tokenName, setTokenName] = useState("")
29   const [tokenDescription, setTokenDescription] = useState("")
30   const [showModal, setShowModal] = useState(false)
31   const hideModal = () => setShowModal(false)
32   const dispatch = useNotification()
```

- components/UpdateListingModal.js

```
JS UpdateListingModal.js X
nextjs-nft-marketplace-moralis-fcc > components > JS UpdateListingModal.js > ...
1 import { Modal, Input, useNotification } from "web3uikit"
2 import { useState } from "react"
3 import { useWeb3Contract } from "react-moralis"
4 import nftMarketplaceAbi from "../constants/NftMarketplace.json"
5 import { ethers } from "ethers"
6
7 export default function UpdateListingModal({
8   nftAddress,
9   tokenId,
10  isVisible,
11  marketplaceAddress,
12  onClose,
13 }) {
14   const dispatch = useNotification()
15
16   const [priceToUpdateListingWith, setPriceToUpdateListingWith] = useState(0)
17
18   const handleUpdateListingSuccess = async (tx) => {
19     await tx.wait(1)
20     dispatch({
21       type: "success",
22       message: "listing updated",
23       title: "Listing updated - please refresh (and move blocks)",
24       position: "topR",
25     })
26     onClose && onClose()
27     setPriceToUpdateListingWith("0")
28   }
29
30   const { runContractFunction: updateListing } = useWeb3Contract({
31     abi: nftMarketplaceAbi,
32     contractAddress: marketplaceAddress,
33     functionName: "updateListing"
```

-components/UpdateListingModal.tsx

```
nextjs-nft-marketplace-moralis-fcc > components > TS UpdateListingModal.tsx > ...
1 import { Modal, useNotification, Input, Illustration, Button } from "web3uikit"
2 import { useState } from "react"
3 import { useWeb3Contract } from "react-moralis"
4 import { ethers } from "ethers"
5 import Image from "next/image"
6
7 export interface UpdateListingModalProps {
8   isVisible: boolean
9   onClose: () => void
10  nftMarketplaceAbi: object
11  marketplaceAddress: string
12  nftAddress: string
13  tokenId: string
14  imageURI: string | undefined
15  currentPrice: number | undefined
16 }
17
18 export const UpdateListingModal = ({
19   isVisible,
20   onClose,
21   nftMarketplaceAbi,
22   marketplaceAddress,
23   nftAddress,
24   tokenId,
25   imageURI,
26   currentPrice,
27 }: UpdateListingModalProps) => {
28   const dispatch = useNotification()
29
30   const [priceToUpdateListingWith, setPriceToUpdateListingWith] = useState<string | undefined>()
31
32   const handleUpdateListingSuccess = async (tx) => {
33     await tx.wait(1)
```

