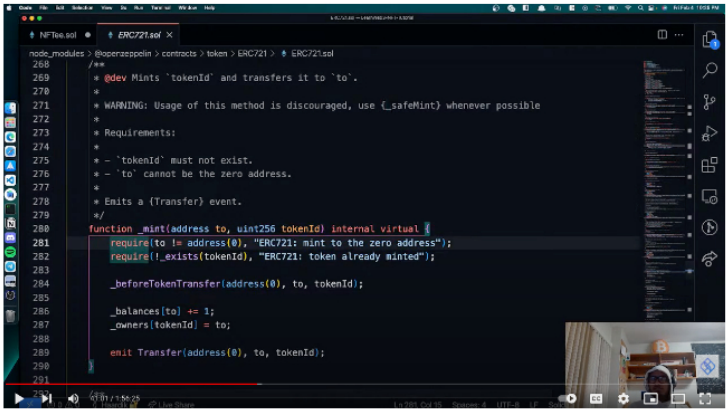
Lesson Type: Practical | Estimated Time: 15-20 minutes | Current Score: 0%

Build your own NFT using ERC-721

Prefer a Video?

If you would rather learn from a video, we have a recording available of this tutorial on our YouTube. Watch the video by clicking on the screenshot below, or go ahead and read the tutorial!



Prerequisites

- Set up a Metamask (Beginner Track - Level-4)
- Check if your computer has Node.js. If not download from here

Build

Smart Contract

To build the smart contract we would be using Hardhat. Hardhat is an Ethereum development environment and framework designed for full stack development. In simple words you can write your smart contract, deploy them, run tests, and debug your code.

To setup a Hardhat project, Open up a terminal and execute these commands

```
mkdir NFT-Tutorial
cd  NFT-Tutorial
npm init --yes
npm install --save-dev hardhat
```

In the same directory where you installed Hardhat run:

```
npx hardhat
```

Select `Create a Javascript Project` and follow the steps. At the end, you will have a new Hardhat project ready to go!

Write NFT Contract Code

Lets install some OpenZeppelin contracts so we can get access to the ERC-721 contracts. In your terminal, execute the following command:

```
npm install @openzeppelin/contracts
```

- In the contracts folder, create a new solidity file called NFTee.sol
- Now we would write some code in the NFTee.sol file. We would be importing **Openzeppelin's ERC721 Contract**. ERC721 is the most common standard for creating NFT's. In the freshman track, we would only be using ERC721. In the sophomore track, you would learn more about ERC721's in detail. So dont worry, if you dont understand everything :)

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

// Import the openzepplin contracts
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

// GameItem is  ERC721 signifies that the contract we are creating imports ERC721 and follows ERC721 contract from op
contract GameItem is ERC721 {

    constructor() ERC721("GameItem", "ITM") {
        // mint an NFT to yourself
        _mint(msg.sender, 1);
    }
}
```

Compile the contract, open up a terminal and execute these commands

```
npx hardhat compile
```

If there are no errors, you are good to go :)

Configuring Deployment

Lets deploy the contract to `goerli` test network. To do this, we'll write a deployment script and then configure the network. First, create a new file/replace the default file named `deploy.js` under the `scripts` folder, and write the following code there:

```
// Import ethers from Hardhat package
const { ethers } = require("hardhat");

async function main() {
  /*
A ContractFactory in ethers.js is an abstraction used to deploy new smart contracts,
so nftContract here is a factory for instances of our GameItem contract.
*/
  const nftContract = await ethers.getContractFactory("GameItem");

  // here we deploy the contract
  const deployedNFTContract = await nftContract.deploy();

  // wait for the contract to deploy
  await deployedNFTContract.deployed();

  // print the address of the deployed contract
  console.log("NFT Contract Address:", deployedNFTContract.address);
}

// Call the main function and catch if there is any error
main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });
```

Now create a `.env` file in the `NFT-Tutorial` folder and add the following lines. Follow the instructions below.

Go to **Quicknode** and sign up for an account. Quicknode is a node provider that lets you connect to various different blockchains. We will be using it to deploy our contract through Hardhat. After creating an account, `Create an endpoint` on Quicknode, select `Ethereum`, and then select the `Goerli` network. Click `Continue` in the bottom right and then click on `Create Endpoint`. Copy the link given to you in `HTTP Provider` and add it to the `.env` file below for `QUICKNODE_HTTP_URL`

To get your private key, you need to export it from Metamask. Open Metamask, click on the three dots, click on `Account Details` and then `Export Private Key`. MAKE SURE YOU ARE USING A TEST ACCOUNT THAT DOES NOT HAVE MAINNET FUNDS FOR THIS. Add this Private Key below in your `.env` file for `PRIVATE_KEY` variable.

```
QUICKNODE_HTTP_URL="add-quicknode-http-provider-url-here"

PRIVATE_KEY="add-the-private-key-here"
```

You can think of Quicknode as AWS EC2 for blockchain. It is a node provider. They run Ethereum nodes (among other blockchains) and give us access to them. It helps us to connect with the blockchain by providing us with nodes so that we can read and write to the blockchain. Your Hardhat application will take your smart contract and send it to the Ethereum node given to us by Quicknode so it can be deployed on the Goerli network.

- Now we would install `dotenv` package to be able to import the env file and use it in our config. In your terminal, execute these commands.

```
npm install dotenv
```

- Now open the hardhat.config.js file, we would add the `goerli` network here so that we can deploy our contract to the Goerli network. Replace all the lines in the `hardhat.config.js` file with the given below lines

```
require("@nomicfoundation/hardhat-toolbox");
require("dotenv").config({ path: ".env" });

const QUICKNODE_HTTP_URL = process.env.QUICKNODE_HTTP_URL;
const PRIVATE_KEY = process.env.PRIVATE_KEY;

module.exports = {
  solidity: "0.8.9",
  networks: {
    goerli: {
      url: QUICKNODE_HTTP_URL,
      accounts: [PRIVATE_KEY],
    },
  },
};
```

- To deploy in your terminal type:

```
npx hardhat run scripts/deploy.js --network goerli
```

- Save the NFT Contract Address that was printed on your terminal in your notepad, you would need it.

Verify on Etherscan

- Go to **Goerli Etherscan** and search for the address that was printed.
- If the `address` opens up on etherscan, you have deployed your first NFT 🎉
- Go to the transaction details by clicking on the transaction hash, check that there was a token transfered to your address

Submit Practical

Verify your smart contract address to pass the assessment for this level.

Ethereum Rinkeby | 0x... | Submit

LearnWeb3

Dashboard
Courses
Community
Blog

About
Work With Us
We're Hiring
Buy us a coffee

Your email address