Introduction to NFT

Web 3.0

Web 3.0 is an idea for a new iteration of the **World Wide Web based on blockchain** technology, which incorporates concepts including **decentralization** and **token-based** economics. Platforms and apps built on Web3 won't be **owned by** a central gatekeeper, but rather by **users**, who will earn their ownership stake by helping to **develop and maintain** those services. A good example of a web3 transaction would be sending Bitcoin directly to another person – not via an online exchange or wallet stored on a centralized server.

Web 1.0 refers roughly to the period from 1991 to 2004, where most websites were **static** web pages, and the vast majority of users were **consumers**, not producers, **of content**.

Web 2.0 is based around the idea of "**the web as platform**" and centers on **user-created content** uploaded to social media and networking services, blogs, and wikis, among other services. Web 2.0 is generally considered to have begun around 2004 and continues to the current day

Non-Fungible Token

“Non-fungible” more or less means that it’s **unique** and **can’t be replaced** with something else. For example, a bitcoin is **fungible** — trade one for another bitcoin, and you’ll have exactly the same thing. A one-of-a-kind trading card, however, is **non-fungible**. If you traded it for a different card, you’d have something completely different.

Most NFTs are part of the **Ethereum blockchain**. Other blockchains can implement their own versions of NFTs.

NFTs can really be anything **digital** (such as drawings, music), but a lot of the current excitement is around using the tech to sell digital art.

Dogecoin isn’t an NFT. But this GIF of a dogecoin is. | GIF: NyanCat on OpenSea

You can **copy** a digital **file** as many times as you want, including the art that’s included with an NFT. But NFTs are designed to give you something that **can’t** be **copied**: **ownership** of the work.

There are several **marketplaces** that have popped up around NFTs, such as OpenSea, which allow people to buy and sell.

Like cryptocurrencies, NFTs are stored in **digital wallets**.

Make an NFT

The **ERC-721** is the most common **NFT standard**. If your **Smart Contract implements** certain standardized **API** methods, it can be called an ERC-721 Non-Fungible Token Contract.

Open-source projects like **OpenZeppelin** have **simplified** the **development** process by implementing the most common ERC standards as a reusable library.

By **minting** an NFT, you **publish** a unique **token** on a blockchain. This token is an **instance of your Smart Contract**.

Each token has a **unique tokenURI**, which contains **metadata** of your asset in a **JSON** file that conforms to certain **schema**. The metadata is where you store information about your NFT, such as name, image, description, and other attributes.

An **example** of the JSON file for the "ERC721 Metadata Schema" looks like

```
{
	"attributes": [
		{
			"trait_type": "Shape",
			"value": "Circle"
		},
		{
			"trait_type": "Mood",
			"value": "Sad"
		}
	],
	"description": "A sad circle.",
	"image": "https://i.imgur.com/Qkw9N0A.jpeg",
	"name": "Sad Circle"
}
```

There are 3 ways to store the NFT's metadata:

- on the blockchain
- use IPFS (InterPlanetary File System)
- have your API return the JSON file

We need an **Ethereum address** to interact with our Smart Contract. We will need it to send and receive **transactions**. For example, minting an NFT is a transaction.

To interact with the Ethereum Network, you will need to be connected to an **Ethereum Node**. Running your own Node and maintaining the infrastructure is a project on its own. Luckily, there are **nodes-as-a-service providers** which host the infrastructure for you.

Practice

We follow https://www.freecodecamp.org/news/how-to-make-an-nft/ to make a NFT.

We will be using

- Alchemy as our **node provider**.
- Metamask as our **wallet**. It is a free virtual wallet that manages your **Ethereum addresses**.
- Ropsten **Test Network** for development purposes, and some Eth to cover the fees of deploying and minting your NFT
- Next.js, an open-source development **framework** built on top of Node.js enabling React based web applications functionalities such as server-side rendering and generating static websites
- Hardhat, a **development environment** to compile, deploy, test, and debug your **smart contracts**.
- **OpenZeppelin**
- **Solidity** language to program our Smart Contract
- ethers.js, a library aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem.

Step 1. Initialize the project

Step 2. Create .env file

Step 3. Create the smart contract (.sol)

Step 4. Metadata for NFT

For example, the metadata looks as below:

Step 5. Deploy Smart Contract on the ropsten test network

For example, the Smart Contract is deployed to
https://ropsten.etherscan.io/address/0x032988E42f8b422574645402Deb8E4dBCd3498Ad.

Step 6. Mint a NFT

For example, the transaction looks like this
https://ropsten.etherscan.io/tx/0x95740e35812e7a96d34a989d7ec2ed0a097ecb4bc2505f667f28f95ccd52a4c8

Step 7. View NFT in Metamask Wallet

*References

https://www.theverge.com/22310188/nft-explainer-what-is-blockchain-crypto-art-faq
https://en.wikipedia.org/wiki/Web3
https://www.wired.com/story/web3-gavin-wood-interview/