Fundamentals of Decentralized Applications

FinTech

Lesson 22.1

Class Objectives

By the end of this lesson, you will be able to:

Build non-fungible token (NFT) contracts by using ERC-721 standards and the OpenZeppelin library.

Deploy the NFT to a local blockchain by using Ganache, MetaMask, and Remix.

Build a decentralized application for an NFT by using Streamlit and Web3.py.

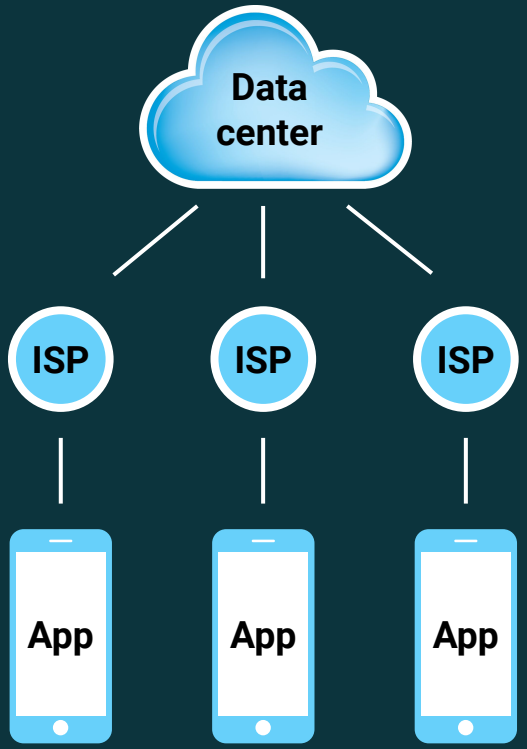
Introduction to Decentralized Applications

Remember dApps?

A **dApp** is a software application that has a decentralized operation and uses decentralized storage or a database.

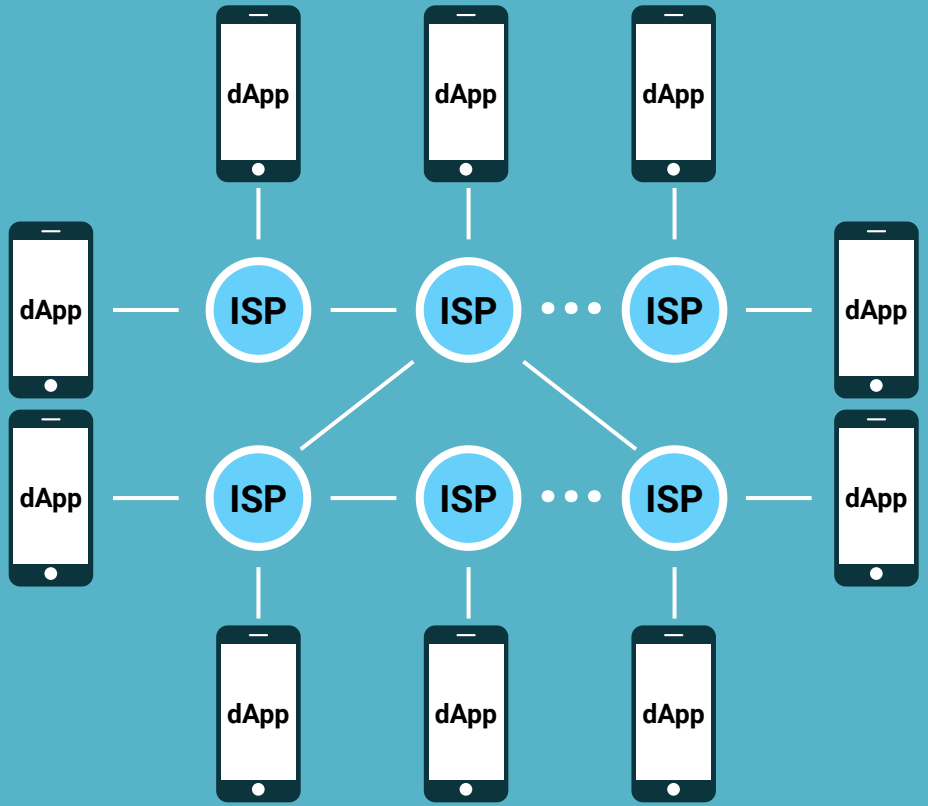
Apps

Run on a centralized server
and use centralized storage.



dApps

Run in a decentralized environment
that the blockchain nodes provide.



dApps...

- Use financial programming and blockchain technology to build new financial capabilities for fintech.
- Can even use machine learning, although that's not required for a dApp to be considered a dApp.

We Have an Exciting Day Ahead!

You'll learn...

...about the structure and components of a dApp and how dApps use blockchain technologies to achieve decentralization.

You'll create...

...a smart contract for NFTs that uses the Ethereum Request for Comments (ERC)-721 standard.

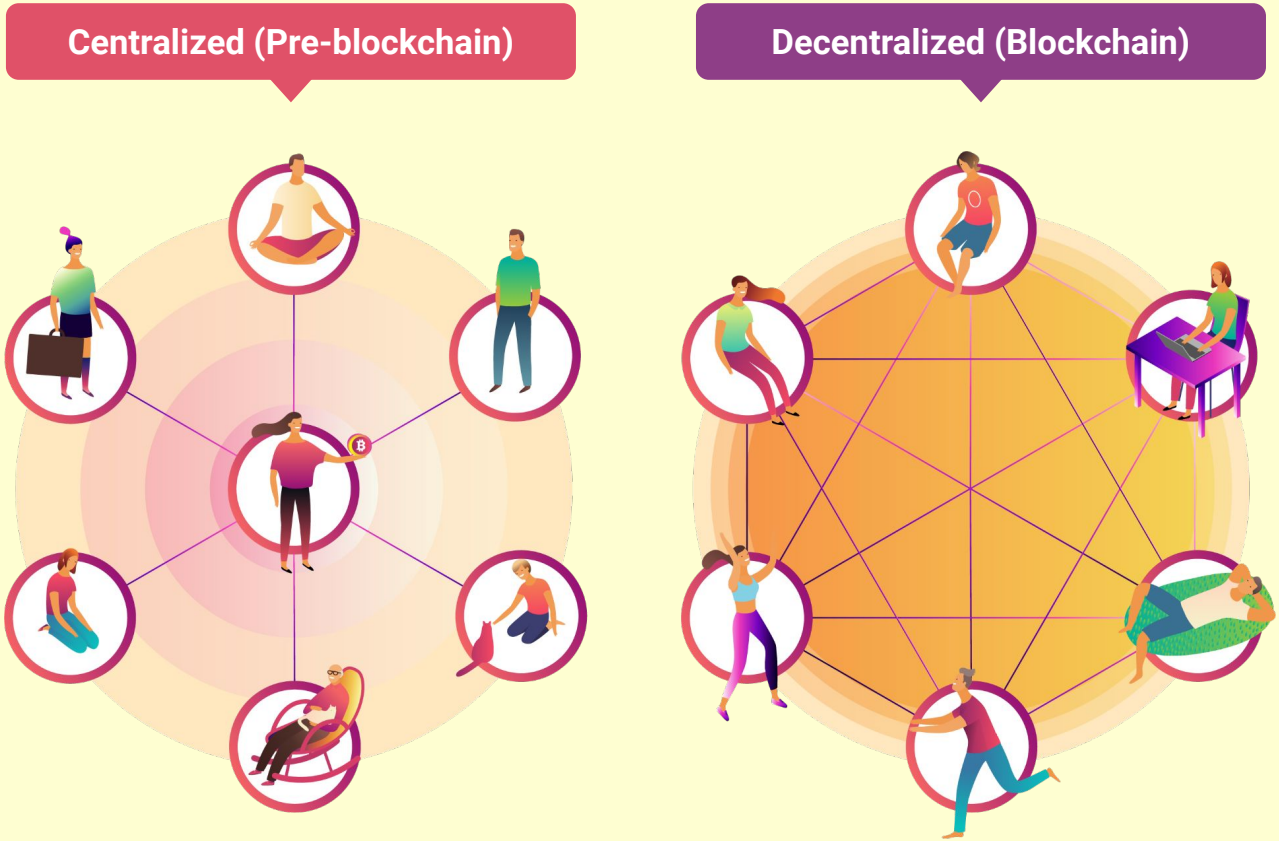
You'll build...

...a dApp that can register new digital artwork on a blockchain by minting the NFTs.

Introduction to Decentralized Applications

From the perspective of a user, a dApp is just like any other application that they'd use on the internet.

What makes a dApp different is that it's built on a decentralized network, such as a blockchain.

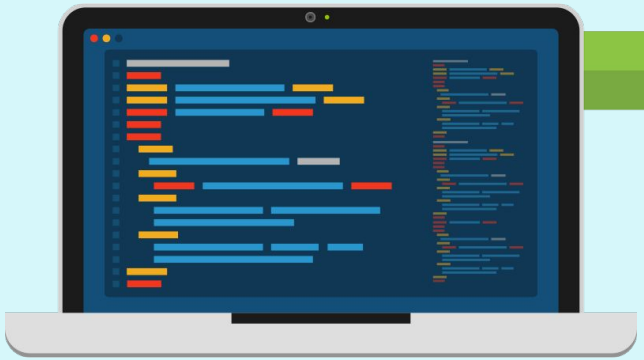


Introduction to Decentralized Applications

In this course, we've created many financial applications by using Python.
The structure of each application typically consists of a back end and a front end.

Typical Application

Back End
The core logic of the application



Data passes between the front end and the back end

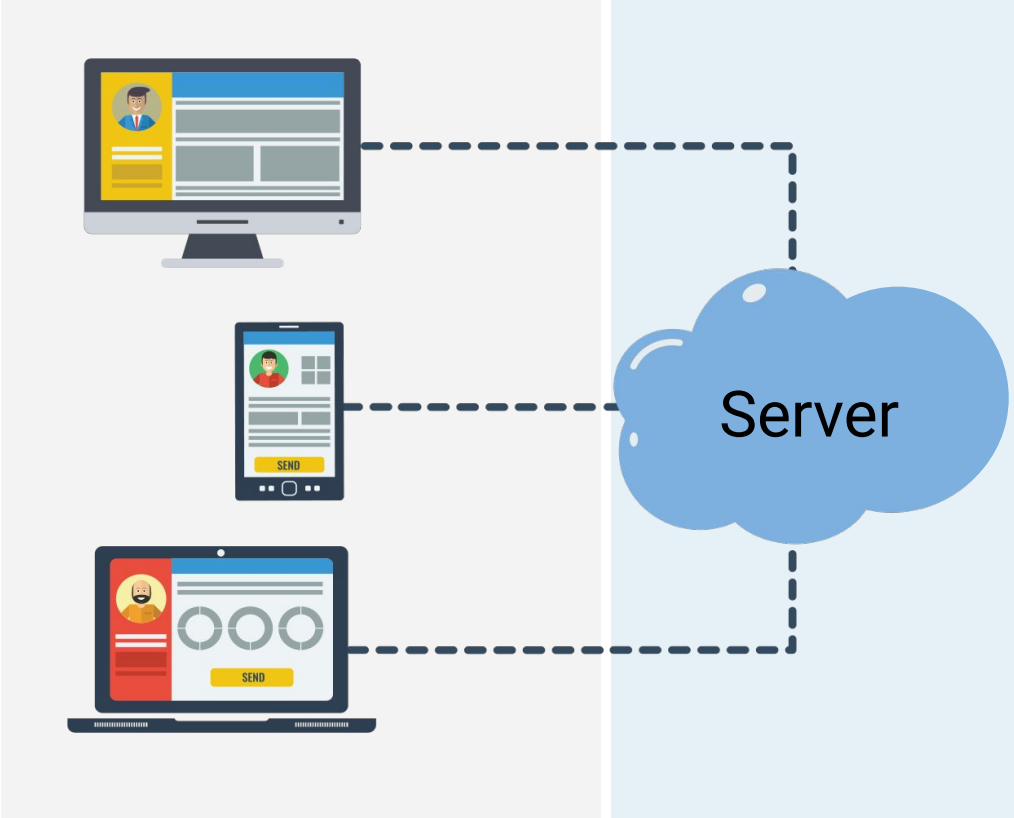
Front End
The user interface of the application

Introduction to Decentralized Applications

With a traditional application, the entire application might be centralized.

For example:

- A server in the cloud might run the code and manage the data.
- Users then interact with this server through a webpage interface.
- While this approach works, these apps suffer from many of the same centralization limitations that we previously discussed regarding blockchain.



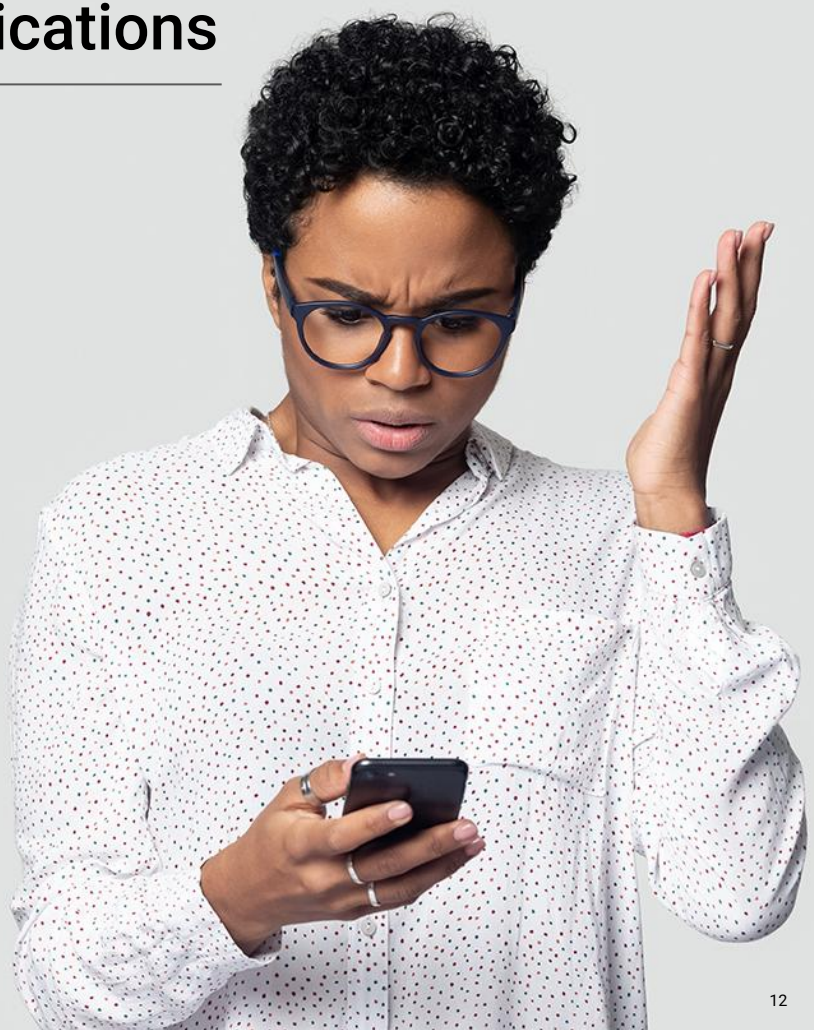
Introduction to Decentralized Applications

Centralized applications are widely used. But, they often suffer from centralization issues, such as:

- having a single point of failure
- being susceptible to attack
- encountering access problems

dApps embrace decentralization by using new blockchain technologies, such as smart contracts, so they become robust against these issues.

The back end of these dApps live on the blockchain and persist despite anything that happens to a single copy (an instance).



Introduction to Decentralized Applications

Tokens are an essential part of dApps, so we'll begin by reviewing fungible and non-fungible tokens.


RECAP

Review of Fungible and Non-fungible Tokens

Last Class

You previously implemented a basic token as well as the ERC-20 fungible token standard by using OpenZeppelin libraries.

This Class

Today, you'll implement your first ERC-721 non-fungible token by using the OpenZeppelin libraries.

An **ERC**, or **Ethereum Request for Comment**, is a common standard for creating tokens on the Ethereum blockchain. There are several ERCs defined by the Ethereum community.

What are some differences between fungible and non-fungible tokens?

Review of Fungible and Non-fungible Tokens

Some differences between fungible and non-fungible tokens:

Fungible tokens	Non-fungible tokens
Not unique	Unique
Interchangeable with one another	Not interchangeable with one another
Fungible tokens use the ERC-20 standard	Non-fungible tokens use ERC-721

What are some examples of fungible assets?

Examples of Fungible Assets

United States Dollars (USD)

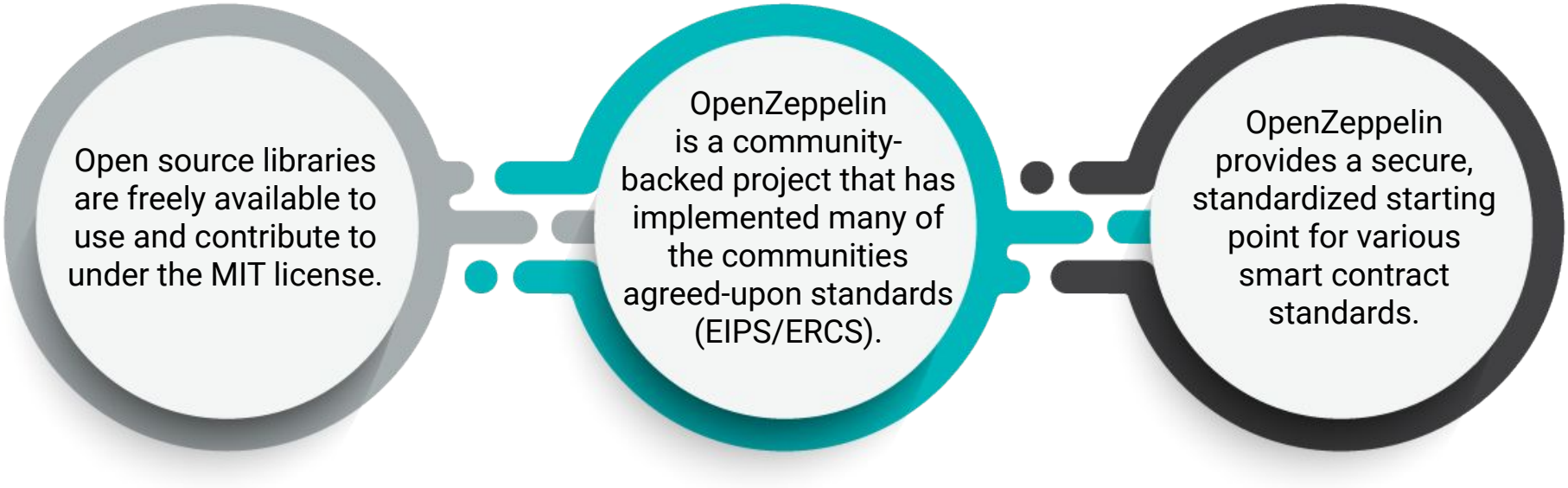
Ether (ETH)

Bitcoin (BTC), gold

What are some benefits of using open source libraries such as OpenZeppelin?

Recap

Some benefits of using the OpenZeppelin libraries:



Questions?

Build a dApp Back End

NFTs and ERC-721

Many of the applications that you have built over the last few units have been decentralized applications.

For example, if the Arcade and XP tokens were launched on the live Ethereum mainnet, they would act as decentralized applications.

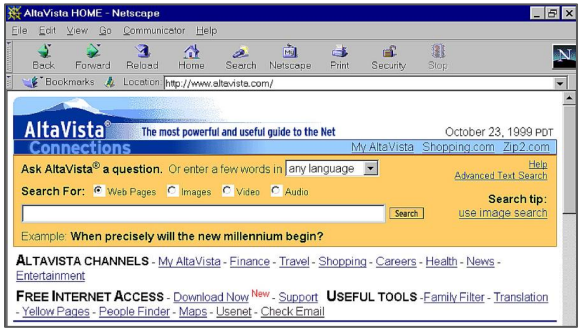
NFTs and ERC-721

The `ArcadeToken` and `XP_Token` applications are fungible tokens, but some of the most popular dApps are created for **non-fungible tokens**, or **NFT**s.

NFTs and ERC-721

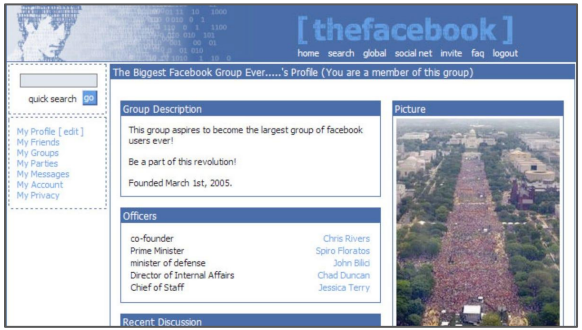
By developing applications for a blockchain, you automatically develop dApps for the Next Generation Web, also known as Web 3.0.

Web 1.0



The beginning of the Internet. Websites were very simple with limited interactivity, and the basic tools and protocols were defined.

Web 2.0



Marked by the rise of social networks and cloud computers.

Web 3.0

The evolution of the Internet to a decentralized environment powered by blockchain technologies.

https://www.techradar.com/ /www.businessinsider.com

Examples of NFTs

NFTs represent unique assets, such as land, art, or other items with no interchangeable counterpart.

Pieces of land

Diamonds

Collectibles

https://www.funko.com

NFTs

With dApps specifically designed for NFTs, users can buy or sell artwork NFTs in a decentralized marketplace.

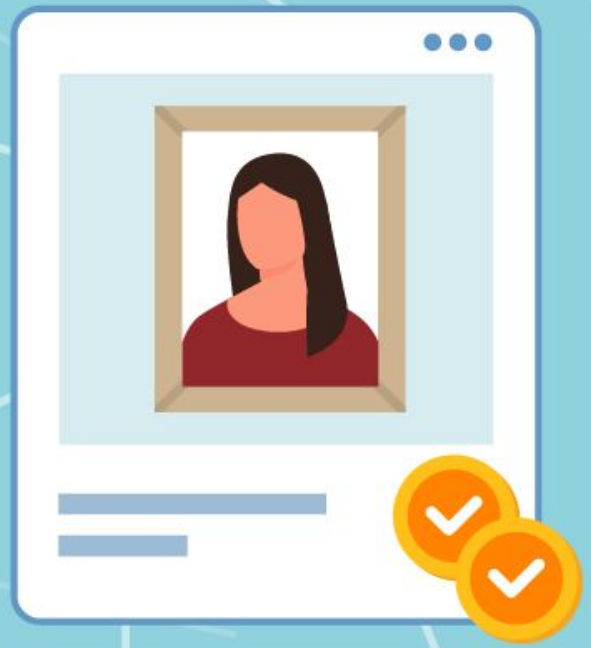
They can also play games for which the avatars and game items are NFTs on a blockchain.

NFTs and ERC-721

In this lesson, we'll further explore dApps by learning how to build a new type of smart contract specifically designed for NFTs.

To build this new type of contract, we'll use the ERC-721 Non-Fungible Token Standard defined by the Ethereum Improvement Proposal (EIP)-721.

ERC-721 is considered the default standard for implementing most NFTs.



We'll apply the ERC-721 standard to build an art token contract. We will use ERC-721 tokens to register new pieces of artwork.

Instructor Demonstration

Build a dApp Back End

Uniform Resource Identifier (URI)

Uniform Resource Identifier (URI) originated from a World Wide Web initiative to standardize the way that files or objects get encoded for the web.

The URI is an identifier of a specific resource, like a page, a document, or a book.

This is in contrast to the more familiar URL, which is referred to as a locator, as it includes how to access a resource like HTTPs or FTP.

The URL is a subset of the URI.

URI (Uniform Resource Identifier) = URL + URN

URL (Universal Resource Locator)

https://docs.openzeppelin.com/contracts/2.x/api/token/erc721#IERC721

URN (Universal Resource Name)

https://docs.openzeppelin.com/contracts/2.x/api/token/erc721#IERC721

Method that defines how to access resource

Location where resources reside

https://docs.openzeppelin.com/contracts/2.x/api/token/erc721#IERC721

Resource

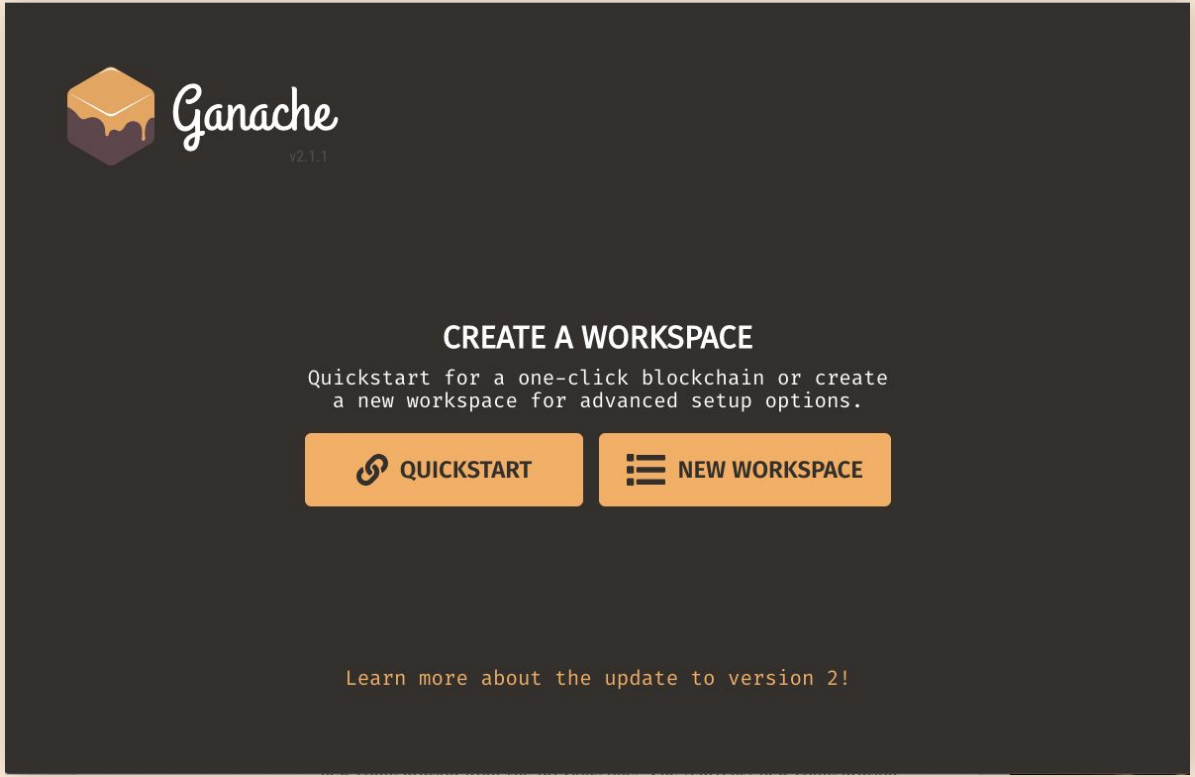
Compile and Deploy the Contract

As with the fungible tokens, the next step involves compiling and deploying the ArtToken contract by using the Remix IDE, MetaMask, and Ganache.

Compile and Deploy the Contract

Workspaces in Ganache are a configuration for a development blockchain.

Ganache's local test blockchain will provide the accounts and the ether that will be needed later to deploy and test the ArtToken dApp.

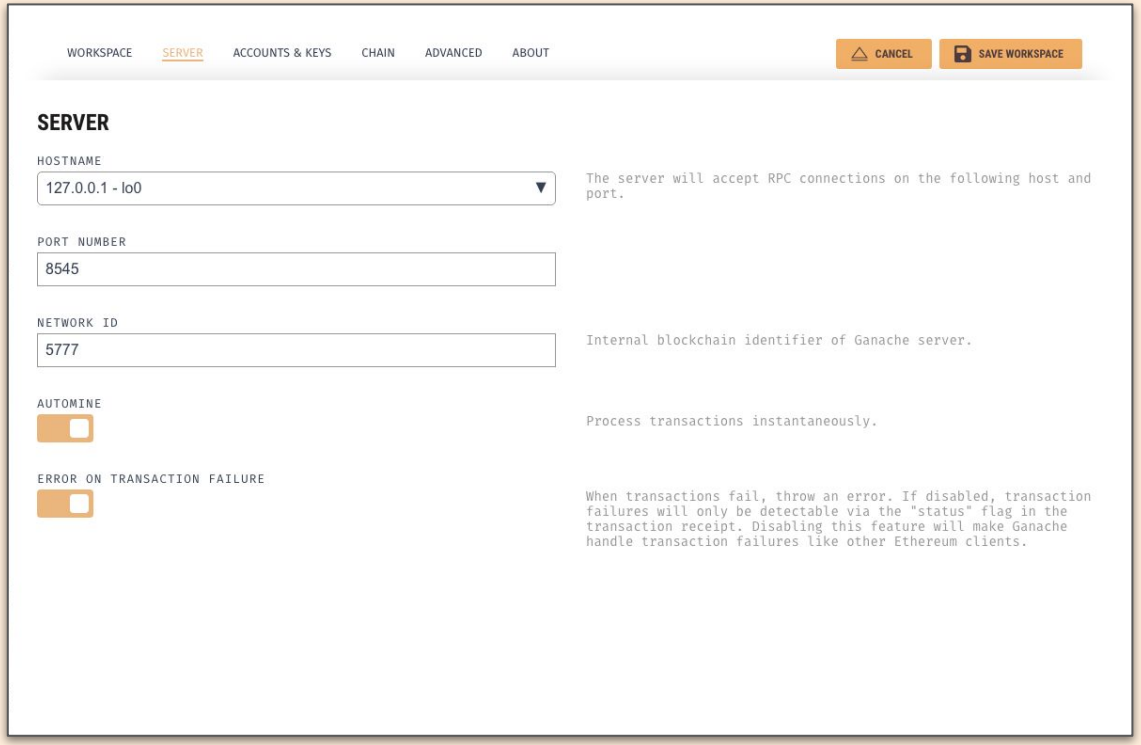


Compile and Deploy the Contract

Workspaces allow you to customize many things about your chain:

An initial pre-mine of any size deposited into any number of generated addresses.

Specific ports for talking to your blockchain node.



Compile and Deploy the Contract

MetaMask is the wallet that serves as the link between the Remix IDE and the Ganache blockchain.

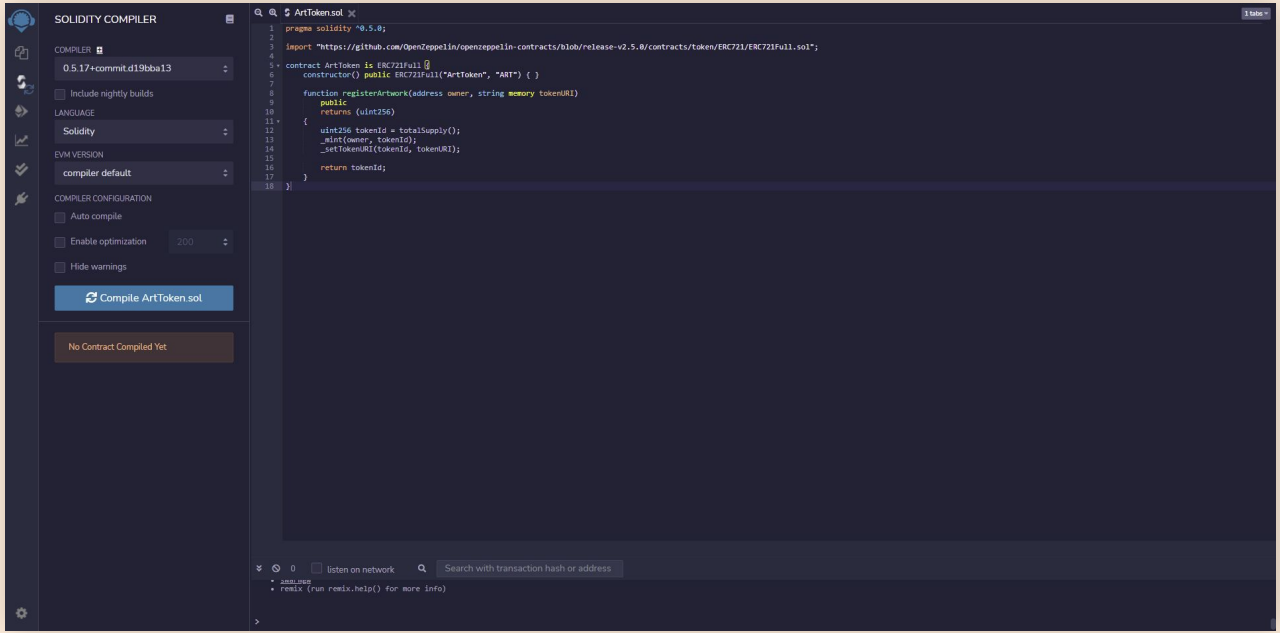
Accounts from the new Ganache instance need to be imported into MetaMask in order to be accessed in Remix.

Compile and Deploy the Contract

There is a new step we must undertake as part of the process of creating our dApp.

The details of the contract will need to be saved for use by the front end of the application.

This is accomplished by copying its application binary interface (ABI) file, found at the bottom of the Compiler page, into a local JSON file.



Questions?

Time to Code

Build the Mint-Condition Artwork Back End

Suggested Time:

25 minutes

What are some differences between fungible and non-fungible tokens?

Recap

Some differences between fungible and non-fungible tokens:

Fungible tokens	Non-fungible tokens
Not unique	Unique
Interchangeable with one another	Not interchangeable with one another
Fungible tokens use the ERC-20 standard	Non-fungible tokens use ERC-721

What are the four processes that the `registerArtwork` function will perform?

Build the Mint-Condition Artwork Back End

The four processes that the `registerArtwork` function will perform are:

Create a new `tokenId`.

Mint a new token for that new `tokenId`.

Set the `tokenURI`, which associates the digital artwork with the token.

Return the `tokenId`.

What back-end processes are performed by the `ArtToken` smart contract?

Build the Mint-Condition Artwork Back End

The back-end processes performed by the ArtToken smart contract are :

The `ArtToken` contract controls all of the logic associated with the `ArtToken` NFT.

It keeps track of each new token, catalogued by tokenId.

Each `tokenId` contains references to the owner's address and the URI associated with the asset.

All of this information will be stored on the blockchain and can be accessed by making a call to the blockchain.

Questions?


Break

Instructor Demonstration

Build a dApp Front End

Questions?

Time to Code

Build a dApp for Mint-Condition Artwork

Suggested Time:

25 minutes

Questions?

Capstone Project Requirements

Capstone Project Requirements

Capstone Project Guidelines: Technical Requirements

This project will be graded according to the following rubric:

Proficiency	(≥ 90% of the points)
Approaching proficiency	(≥ 80% of the points)
Progressing	(≥ 70% of the points)
Emerging	(< 70% of the points)

Capstone Project Guidelines

Software Version Control (10 points)

Repository created on GitHub.	(2 points)
Files frequently committed to repository.	(3 points)
Repository organized, and relevant information and project files included.	(2 points)
Software Version Control	(3 points)

Data Collection and Preparation (10 points)

Data collected, cleaned, and prepared for the application or analysis.	(10 points)

Capstone Project Guidelines

Development (40 points)

Jupyter notebook, Google Colab notebook, Amazon SageMaker Studio notebook, or Streamlit application created.	(10 points)
One or more Python modules, machine learning models, or Solidity smart contracts created.	(10 points)
Calculations, metrics, visualizations, or video needed to demonstrate the application included.	(10 points)
One new technology or library used that the class hasn't covered.	(10 points)

Documentation (15 points)

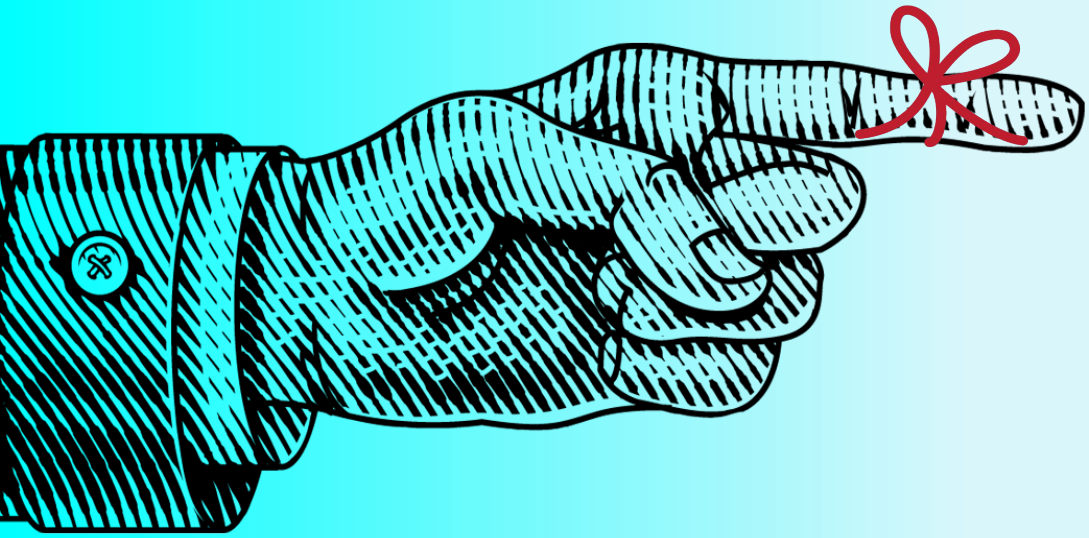
Code is well commented with concise, relevant notes.	(5 points)
GitHub `README.md` file includes a concise project overview.	(2 points)
GitHub `README.md` file includes detailed usage and installation instructions.	(3 points)
GitHub `README.md` file includes either examples of the application or the results and summary of the analysis.	(5 points)

Capstone Project Guidelines

Each group will prepare a formal 10-minute presentation that should include the following:

Presentation (25 points)

An executive summary of the project and project goals.	• Explain how this project relates to fintech.	(15 points)
The approach that your group took to achieve the project goals.	• Include any relevant code or demonstrations of the analysis or application. • Describe the techniques that you used to test or evaluate the code. • Discuss any unanticipated insights or problems that arose and how you resolved them.	(10 points)
The results and conclusions from the analysis or application	• Include relevant images or examples to support your work. • If the project goal wasn't achieved, share the issues and what the group tried for resolving them.	(5 points)
Next steps	• Briefly discuss the potential next steps for the project. • Discuss any additional questions that you'd explore if you had more time. Specifically, if you had additional weeks to work on your project, what would you research next?	(5 points)



Remember:

Projects are a professional opportunity.

Projects are portfolio pieces, so use projects as an opportunity to challenge yourself and showcase your knowledge for potential employers.

Minimum Viable Product

You can develop your project as a **minimum viable product**, also known as **MVP**.

An **MVP** is a version of a product that implements enough features to be usable by early customers or users who can then provide feedback for product enhancement or future product development.

Feature Creep

Keep your MVP simple and add features as you go to avoid feature creep.

Remember the Project Metrics

Projects will be evaluated on:

- Concept
- Design
- Functionality
- Collaboration
- Presentation

Questions?

Time to Code

Project Time

Suggested Time:

30 minutes

Project Time

Implement best practices that you learned from your previous two projects:

01

Use office hours to ask questions about your project.

02

Communication is key when it comes to successfully completing a group project. Be sure to meet with your group regularly.

03

Plan to work with your group outside of class.


The End