## Logistics and Welcome

- Welcome to the Q1 2025 cohort of the EV Boot Camp!
- Nan (Operations Manager at Anor Club) is your main logistical point of contact. Reach out with any questions or issues. Nan will provide contact details in the chat.
- Anor Club:

  An educational company that offers various educational programs, including boot camps, hackathons, and accelerators for small startups. They also create different kinds of educational content.

## Introduction of the Teacher

- Matos Pagani will be your primary teacher for the next six weeks. He has been teaching with Anor Club for 2-3 years.

## Sponsor

- Ron (formerly known as Wallet Connect) is the gold sponsor for this boot camp.
- They will host a guest lecture in week seven.

## Course Structure

| Week(s) | Focus |
|---|---|
| 1-6 | Teaching by Matos Pagani |
| 7 | Guest lectures from crypto industry |
| 8 | Final project building |

- You will be divided into groups of 5-7 people.
- Groups will be formed tomorrow morning and announced via Discord.
- These groups will work on weekly homework and the final project.

## Time Zone Change

- The UK switches from GMT to BST (British Summer Time) in the last week of March.
- If you're not in the UK, your boot camp timings might shift by an hour.
- Check your calendar invites for the correct time.

## House Rules

- Attend all sessions if possible.
- Recordings are for revision, not a replacement for live attendance.
- Join Zoom sessions with the name you used when applying.
- Complete all homework.

## Graduation Requirements

- Submit the final project with your group.
- Week 8 is dedicated to building your final project.
- A "show and tell" event will occur at the end where one group member will present the group's project.
- Graduates will either receive their deposit back (if applicable) or a certificate (PDF or NFT).
- Fill out a refund form and feedback form at the end of the course.

## Resources

- **GitHub organization (Even Boot Camp)**: You should have received an invite. Contact Nan if you don't have access.

- **Discord channel**: You have been added to a private channel only accessible to this cohort.## Communication Channels

- **Discord**: The main channel for communication with the instructor and fellow students.

- **Hacker Pack**: A Notion page containing basic information about the boot camp, resources, and session recordings.

    - Session recordings will be posted the morning after each session.

## Boot Camp Structure and Tone

- The boot camp is designed to be **interactive** and **hands-on**.
- Participants are encouraged to ask questions and engage with the material.
- The instructor will be looking for active participation and feedback from students.
- Use the **raise hand** reaction in Zoom to ask questions.
- Participants are encouraged to have their cameras on to simulate a classroom environment.
- The boot camp focuses on showing how technology works through practical application rather than just theoretical explanations.
- Participants are expected to actively install and run things on their computers to gain experience.

## Accessing Course Materials on GitHub

- Course materials are available in a GitHub repository called `code-club-solidity-bootcamp`.

- Each lesson has a dedicated folder with a README file outlining the topics covered.

- Participants should familiarize themselves with using GitHub to access lessons, clone repositories, and ask questions.

  **GitHub Issues**: A feature for asking questions and seeking clarity on course topics.

- Ensure you have access to the repository and resolve any access issues promptly.

## Weekly Schedule and Topics

- The boot camp consists of 24 lessons over six weeks, with four lessons per week.
- The structure includes weekly lessons, workshops, and a final project-building week.
- The first week focuses on wallets, transactions, and understanding Ethereum.
- Subsequent weeks cover smart contracts, Solidity syntax, EVM structure, and more advanced topics.
- Topics include toolings, building dApps, security practices, DeFi, scalability, and upgradability.
- The course follows a sequence of projects to build practical skills.

# Lesson 1: Getting Started

- Introduction to the concept of **wallets** and **transactions**.
- Brief understanding of how **Ethereum** works through exercises.
- Introduction to **coding smart contracts**.

# Understanding Blockchain

## Blockchain Definition

- **Blockchain** is a way to do **distributed computation** between parties that don't necessarily trust each other, especially in **permissionless blockchains** like Ethereum.
- An **immutable distributed ledger** with support for **smart contracts**.
- A system of **traceable and trackable transactions**, where each transaction has cryptographic hashes for verification.
- A technology to represent **decentralized data across borders**.

## Key Features and Innovations

- Ethereum moves beyond the limitations of existing blockchain technologies like Bitcoin by enabling the creation of **complex applications** and **customizable ownership** through **smart contracts**.
- Ethereum allows us to **change data in a deterministic way** across many computers through smart contracts.

## History

- Vitalik Buterin initially proposed Ethereum as an upgrade for Bitcoin.
- The Bitcoin community rejected the proposal due to concerns about cost.
- Vitalik Buterin, along with Joe Lubin, Gavin Wood, and others, created Ethereum.

### Traditional Blockchains vs. Ethereum

- Traditional blockchains are primarily for transacting assets.
- Ethereum innovates by supporting **smart contracts**.

## The Internet as a Metaphor

### Defining the Internet

- The Internet is a set of technologies that allows us to **interconnect different networks**, including local networks.

## Creating a Wallet

### Practical View of Blockchain

Instead of diving into technical explanations right away, we'll take a **practical approach** to understanding blockchain technology.

### Web3 Wallets

To start, we're going to create a wallet. While we'll be using **MetaMask**, there are many web3 wallets available, such as:

- QuickNode list of wallets
- Binance Wallet
- Coinbase Wallet
- Trust Wallet
- Rainbow Wallet
- Non-Rainbow Wallet

For this boot camp, we'll focus on MetaMask, which can be installed as a browser extension.

### MetaMask Installation

Installing the MetaMask extension creates a key pair, specifically an address key pair, which serves as your identity on the network. This setup provides an interface similar to a bank account, giving you control over your digital assets and information.

Go to metamask.io to install the extension.

### Account Creation

When creating a MetaMask wallet, no personal information like email, home address, or name is required. This raises a question:

> How does MetaMask know if you're creating a second account or if you've already created an account?

The answer is that it doesn't.

## Public and Private Keys

### Understanding Key Pairs

By creating a wallet, we'll gain insight into how blockchains work. Installing MetaMask provides a mnemonic phrase, from which multiple addresses can be derived, each with its own private key.

### Preventing Account Impersonation

A crucial question arises:

> What prevents someone from randomly creating an account that already holds funds and impersonating the owner?

While you can generate infinite wallets, the probability of collision (creating a wallet identical to one already in use) is extremely low.

One participant suggested the probability of collision is:

$1/2^{256}$

## Collision Resistance

Collision resistance is based on luck and avoiding hash collisions. It is important for the privacy of light clients. Metamask cannot map your ID or IP to count your accounts because of collision resistance.

When creating a public/private key pair or an account in Metamask, there is no check to see if the account already exists. Getting the same mnemonic as someone like Vitalik is possible, but the probability is extremely low. The Ethereum blockchain has more addresses than there are grains of sand on Earth, illustrating how unlikely it is to pick the same account as someone else. When creating an account, it is a random account that you pick and use for signing transactions later.

## Verifying Transactions

When depositing USDC for the course, you input the hash address of your transaction on Polygonscan to verify that the money has gone through.

## Testnets and Faucets

To interact with tokens without spending real money, testnets are used. Testnets operate the same way as the mainnet, but you can get free tokens using faucets.

> Faucets: Distribute tokens for testnets.

You can find faucets by searching on Google for "Goerli faucet" or any testnet faucet. Some faucets are even provided by Google. You can input your wallet address and request testnet tokens. These tokens are fake money used to test interactions with the blockchain.

Steps to use testnets:

1. Connect to a testnet like Goerli.
2. Fund your wallet using a faucet.
3. Send transactions.

## Example Transaction on a Testnet

An example transaction on a testnet can be used to explain how Ethereum works. A volunteer's address can be used to send some test assets.

Steps to send tokens:

1. Pick a destination address.
2. Specify the amount to send.
3. Approve the transaction by signing with your private key.
4. Broadcast the transaction to the network.

The transaction is broadcasted openly for the network. Eventually, someone will see the transaction and include it in a block for processing the new state.

## Block Explorers

Block explorers such as Etherscan allow you to view transactions. You can click on a transaction in Metamask to view it in the block explorer. Block explorers allow you to get information from the blockchain by querying a public node.

The block explorer shows:

- The transaction from your address to the recipient's address.
- The block in which the transaction was included.
- The amount of value transacted.

## Faucet Balances

Faucets provide small amounts of testnet tokens because they have limited balances. The testnet mirrors the mainnet, including the issuance and burn mechanisms of tokens like SepoliaETH.

## Gas Fees and Transactions

- When a transaction occurs, the block producer earns money, and someone pays a **transaction fee** (gas fee).
- Gas fees on testnets like Sepolia are paid in Sepolia ETH (SETH).
- These fees emulate real blockchain transactions and support the testnet ecosystem through donated computational power.

## Consensus Mechanisms: Proof of Stake

- The Ethereum blockchain uses a **Proof of Stake (PoS)** consensus mechanism.
- Block producers are assigned based on the amount of ETH they stake, not hash power.
- Transactions are included in a block, which is then produced by a user.

## EIP-1559 and ETH Burning

- **EIP-1559** is a mechanism that incentivizes block producers and creates savings in the blockchain.
- It involves burning a portion of the transaction fees.
- When the network is congested, the blockchain incentivizes block producers to validate transactions.
- When the blockchain is less utilized, it collects fees to create reserves.

## Transactions, Blocks, and Finality

- A transaction must be included in a block.
- Consensus about a block is needed to verify a transaction's correctness.
- Sufficient blocks must be produced to consider a transaction final.
- **Consensus** and **finality** are different concepts.
- Information on the blockchain may not always be final due to potential block reorganizations.
- Older blocks have a higher likelihood of being finalized, as more validators have vetted them.
- Newer blocks are less certain, and their information may be reverted.

## Blockchain as a Decentralized Bank Account

- The internet allows searching for and finding cat pictures online without storing them locally.
- The blockchain allows transacting control of assets.
- Users can receive assets and sign transactions to pass them on, similar to a decentralized bank account.

## Zero-Value Transactions

- Many transactions on the blockchain involve sending zero ETH but still incur gas fees.
- Possible reasons:
  - Sending encrypted messages.
  - Spamming the network.

## State Changes in Transactions

- Transactions cause state changes where one account loses money, and others gain.
- One account's funds decrease to pay another account, and the block producer is also compensated.

## Key Concepts

**Gas Fees:** Payments made by users to compensate for the computing energy required to process and validate transactions on a blockchain.

**Proof of Stake (PoS):** A consensus mechanism where block producers are chosen based on the amount of cryptocurrency they hold and "stake" as collateral.

**EIP-1559:** A proposal for the Ethereum network that changes the transaction fee mechanism to include a base fee that is burned and a tip to miners.

**Finality:** The guarantee that a transaction on the blockchain cannot be altered or reversed once it has reached a certain level of confirmation.

## Timeline for Understanding Blockchain Concepts

| Time | Topic |
|---|---|
| Now | First transaction, gas, and blocks |
| Later Lesson | Scalability |
| Lesson 20 | Consensus and finality |

# Ethereum Transactions

## Understanding Ethereum Transactions

- When a transaction occurs, it often involves calling a **function** within a **smart contract**.
- Transactions involve sending **zero Ether** from one address to another, including additional information or **bytes**.
- These bytes, appended to the transaction, enable **state changes** beyond just monetary value.
- Transactions facilitate **computation** on the blockchain, as programs can alter data.
- The **Ethereum Virtual Machine (EVM)** and **smart contract accounts** enable code execution.

## Tools for Experimentation

- **Browser-based tools** will be used for experimentation:
  - **Remix**
  - **MetaMask**
- The focus will be on understanding the **EVM** and **accounts**.

## Screen Sharing Caution

Be cautious when sharing your screen to avoid unwanted background noises or compromising your computer.

## Smart Contract Transaction with Uniswap

- A demonstration of a smart contract transaction using **Uniswap**
- To test:
  - Connect a wallet to **app.uniswap.org**.
  - Swap tokens on the **test network (Sepolia)**.

## Token Swap Example

1. Swapping 0.001 UNI tokens for USDC on the testnet.
2. Confirm the transaction in the wallet by approving the call.
3. Wait for the transaction to be included in the blockchain.
    - This is similar to a currency exchange order.

## Transaction Details

- The transaction involves sending zero Ether to a smart contract, triggering a series of operations.
- A small fee is paid to send the transaction.
- The smart contract execution results in a state change, modifying stored values.
- Each transaction modifies values stored in smart contracts
    - For example, balances of USDC.

## Potential Applications

- Smart contracts can support a variety of applications, including:
    - Games
    - Voting systems
    - Lotteries
    - DAOs (Decentralized Autonomous Organizations)
    - AI agents

## Focus on Technology

- The bootcamp is about EVM technology, not cryptocurrency trading or token launches.
- The goal is to explore applications of blockchain technology.

## Token Swap Demonstration

- A token swap was performed to demonstrate a state change inside a smart contract operation.

### Uniswap Errors

Errors encountered during token swaps are often due to low liquidity in the testnet, so swapping smaller quantities of tokens is better.

## Remix IDE and Solidity Introduction

### Remix: Web-Based IDE

- Remix is a web-based Integrated Development Environment (IDE) that can be used to write, compile, and deploy smart contracts.
- It can be accessed through browsers like Firefox, Chrome, or Brave. Safari is not supported.
- Remix can also be installed on your computer, but using it online is sufficient.

### Remix Interface

The Remix IDE interface consists of the following panels:

- Icon panel
- Side panel
- Main panel
- Terminal

The important tools for this boot camp are:

- File explorer
- Solidity compiler
- Deploy & run transactions

  Remix Documentation is an important resource to understand the tool

### Important Links

- Remix IDE documentation.
- Solidity documentation.

## Creating a New Workspace

1. Open Remix in your browser.
2. Click on the "workspace actions" button.
3. Create a blank workspace and give it a name.
4. Select the workspace from the drop-down menu.

## Creating a Hello World Contract

1. Create a new file named `hello_world.sol`.
2. Write Solidity code to create a smart contract that returns "Hello World."

## Solidity Overview

- Solidity is a high-level language for implementing smart contracts on the Ethereum Virtual Machine (EVM).

  > Solidity is a high-level language for implementing smart contracts that are programs designed to target the Ethereum Virtual Machine (EVM).

- Unlike languages like Rust, Python, Java, C, or JavaScript, Solidity is designed to target the EVM rather than operating systems or web interfaces.
- Solidity is influenced by C++, Python, and JavaScript but operates differently due to its focus on the EVM.
- The components of smart contracts in Solidity are designed to interact with the EVM, not to print things to a console.

## Initial Code

Start with the provided code snippet inside the `hello_world.sol` file. This code will be used to explain Solidity syntax, compilation to bytecode, deployment to a local environment or testnet blockchain, and interaction with smart contracts.

## Homework

- Read the introduction and topic tables from the Remix documentation to become familiar with the IDE.
- Read the introduction and topic tables from the Solidity documentation to understand its structure.
    - Introduction to Solidity
    - Language description
    - Compiler internals

## Issues

Create GitHub issues for any questions or problems encountered during the lesson.## Smart Contract Deployment Preparation

To prepare for deploying to a blockchain, focus on understanding the components of a smart contract, including the contract definition, the `constructor`, and the functions, denoted by green and blue text.

Before the next lesson, it is suggested to:

- Review today's content.
- Read the documentation.
- Ensure readiness for tomorrow's topics.

For any questions or issues, reach out via the issues tab or the Discord channel.