Patrick Collins

- Foundry Course with AI

- Blockchain Basics.
→ What is Blockchain.
  - Bitcoin
  - Etherium

- Smart contract :- set of instruction betⁿ parties.
- Oracle → device that interacts with the offchain world to provide external data or computation to smart contract
- we need decentralized oracle network.
- Hybrid smart contracts :- On-chain + off chain Agreements
- Chainlink : Decentralized Oracle network.
Hybrid smart contract → have some offchain component
overview
- L2 → Layer 2 solves scalability issues.
Two types of L2 → optimistic Rollup & zero knowledge Rollups
Optimistic Rollups → Arbitrum, Optimism
zeroknowledge → zk sync, Polygon zksync Evm.
Dapp = Decentralized Application = Decentralized Protocol
= Smart Contract.
web1 : Permissionless open source web with static content
web2 : Permissioned web, with dynamic content. where companies run your agreements on their servers.
web3 : The permissionless web, with dynamic content. where decentralized censorship resistant network run your agreement and code. It generally is accompanied by the idea of user owned ecosystems, where the protocols you interact with you also own a portion of instead of solely using the product
- Value of Smart Contract. [Video Explanation]
- Unbreakable Promises.
Purpose of Smart contract.
- Immutable, Decentralized, Transparent [Blockchain]
[- Hybrid smart contracts
→ combines onchain logic with offchain decentralized data and decentralized computation]

2] Transparency & Flexibility
3) Speed & Efficiency [Finance]
4) Security & Immutability.
5] Counterparty Risk Removal
6) Trust minimized agreements.
• What Smart Contracts have done so far?

1] Defi = Decentralized Finance.
2] DAOs = Decentralized Autonomous Organization.
3] NFTs = Non Fungible Token.

• First Transaction.
- Metamask Wallet [Extension]
• Introduction to Gas
Node -> miner or validator get paid to process the transaction
Gas -> unit of computational measurement.
More complex the transaction more gas required.
• Transaction price = Gas Price × Gas Used
Fee

• How Blockchain works.
-> what is Hash
-> A unique fixed length string meant to identify a piece of data. They are created by placing said data into a "hash function".
SHA-256 or Keccak256
-> Genesis Block => First Block in a Blockchain
Nonce -> A number used once to find the solution to the blockchain problem.
Used to define the transaction number for an account/address.
Private Key -> Used to sign transactions
Used -> Elliptic Curve Digital Algorithm.
(ECDSA)

Secret Phrase >> Private Key ||| Public address

↳ Gives Access to all accounts

↳ Gives access to single account

↳ Anyone can see and use.

Gas2 => Block Reward and EIP 1559.

* The more people use a chain the more expensive it is to send transaction.

1 Ether = 1,000,000,000 Gwei = 1,000,000,000,000,000,000 wei
= 10^9 Gwei = 10^{18} wei

- Remix - simple storage
- Basic Variables
  - bool = true / false
  - uint256 = unsigned integer
  - int256 = signed integer
  - string =
  - address =
  - bytes32 =

function store( ) { }.
↓
attributes.

Arrays Structs

Public -> visible externally and internally
Private -> only visible in current contract
external -> only visible externally (only for functions)
internal -> only visible internally.

view -> read states from the blockchain
pure -> ↑ updating states

AI website for developers

phind.com

Memory -> data can be modified
calldata -> temporary so data cannot be modified
storage -> It is a permanent variable that can be modified

Mapping:

• Remix Storage Factory.

• Deploying Contract from Contract
1st way → Using both contract in a single contract
2nd way → Importing using filepath

• How to call other contract functions from the another one.
→ For that we will require
1] Address of that Contract
2] ABI of that contract

Inheritance

override
↳ Virtual / override
↓
If we want to override any function we have to mention virtual in the base contract of that function

override
→ when we were adding something new to that function we will use override.

Using new keyword we can deploy contracts from another contract.

1 eth = 1e18

Payable functions
Revert → Undo any actions that have been done and send the remaining gas back.

Blockchain Oracle
→ Any device that interacts with the off chain world to provide external ~~d~~ data or computation to smart contracts.

- IF we are declaring a variable and it is called up only once we can use 'CONSTANT' while declaring that variable.

- Constant & Immutable
  ↳ If we assign a variable once outside of function and they never change it so if it's assigned at compile time we will write <u>constant</u>.

- After 0.8.7 or 0.8.8
  instead of require we can declare general error
  syntax [error NotOwner() ;]

Two special functions
receive()
Fallback()

Receive() → when we are transacting without any data receive function gets triggered if other function is not properly defined

fallback() → when there is data with transaction

. When calling a contract without any equivalent function if there is data included with that transaction fallback function is called if there is no data in that transaction receive function gets called.

Suppose Ether is being sent to Contract

$$\begin{tikzcd}
 & & \text{If msg-data is empty?} \arrow[dl] \arrow[dr] & \\
 & \text{yes} \arrow[dl] \arrow[dr, "\text{If}"] & & \text{no} \arrow[d] \\
\text{yes} \arrow[d] & & \text{no} \arrow[d] & \text{fallback()} \\
\text{receive() is called} & & \text{fallback() is called.} &
\end{tikzcd}$$

Check for receive function

If yes

Here directly fallback gets called.

Website -> SpeedRunEthereum.com.

Foundry simple storage.

=> Smart Contract Development Framework
It is solidity base.

Foundry Installation

First install wsl if we are on windows
wsl --install [CMD]

Install foundry (commands)

① curl -L https://foundry.paradigm.xyz | bash
② source command mentioned when run the above command
③ foundryup.

• SRC File contains contract.
For compiling => forge compile
To start local node => anvil
Deploy.
=> Command
① forge create ContractName --interactive.
OR. Filename.
② forge create Filename --rpc-url link --
Private-key [Private key]

• How to Deploy with real scripts
Create a file with Eg "DeploySimpleStorage.s.sol"
=> write script
Command to compile script
=> forge script script/filename

Previous command will store the deployed contract temporarily
To put it onchain we have to define rpc-url
command => forge script script/filename --rpc-url
https://127.0.0.1:8545
Now we will have the broadcast folder where we can saw our previous deployment
Now we will deploy it again.

Command
↳ forge script script/filename --rpc-url ----.
-- broadcast --private-key $\boxed{0x-}$

• Converting Hex to decimal
Command
↳ cast --to-base $\boxed{hex}$ dec
↳ cast --help.

$\boxed{7:40}$
• we can use .env for development purpose only.
use keystore with password for live projects.
Create .env file -> Enter details
Now run command -> source .env ↵
It will add the details to the project.
Now deploy using => forge script command
but instead of writing url write $RPC_URL
$PRIVATE_KEY
↳ names which are declared in env file.

To send transaction Calling a function.
- cast send "contract address" "store(uint256)" ↓ function
123 --rpc-url $RPC_URL --private-key $PRIVATE_KEY.
↳ value we want to send