Cryptocurrencies and Blockchain

Kambombo Mtonga

July 2, 2024

Outline

- Ethereum blockchain
  - Elements of Ethereum blockchain
  - Precompiled contracts
  - Ethereum accounts and contracts
  - Ethereum blocks
  - Mining
  - The Ethereum network
- Smart contracts
- Hyperledger

Reference materials

- Book on Ethereum Development with GO:https://goethereumbook.org/ethereum-development-with-go.pdf
- https://medium.com/haloblock/deploy-your-own-smart-contract-with-truffle-and-ganach
- https://github.com/ethereum/wiki/wiki/Design-Rationale
- https://blockgeeks.com/guides/solidity/
- https://truffleframework.com/tutorials/ethereum-overview

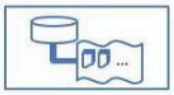
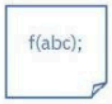
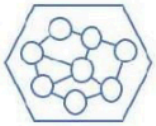
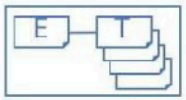
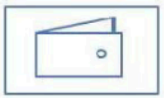
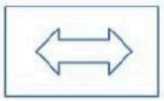
Why use Blockchain

- Blockchains are used when **multiple parties**, perhaps located across the world, **need to share data and transfer value** without **trusting** each other.
- The financial world describes this trust as the **counterparty risk:**
- **the risk that the other party won't hold up their end of the** bargain.
- Blockchains attempt remove the counterparty risk through a clever usage of mathematics, cryptography, and peer-to-peer networking.

How does Blockchain work?

- Nodes, Transactions, Blocks
- Mining through solving hard problems (solving Byzantine fault-tolerant consensus)
- Hashing for integrity
- Digital Signature for Authenticity and/or authorization
- Permanence (Tamper resistance)

Blockchain entities

Ledger		A ledger is a channel's chain and current state data which is maintained by each peer on the channel.
Smart Contract		Software running on a ledger, to encode assets and the transaction instructions (business logic) for modifying the assets.
Peer Network		A broader term overarching the entire transactional flow, which serves to generate an agreement on the order and to confirm the correctness of the set of transactions constituting a block.
Membership		Membership Services authenticates, authorizes, and manages identities on a permissioned blockchain network.
Events		Creates notifications of significant operations on the blockchain (e.g. a new block), as well as notifications related to smart contracts.
Systems Management		Provides the ability to create, change and monitor blockchain components
Wallet		Securely manages a user's security credentials
Systems Integration		Responsible for integrating Blockchain bi-directionally with external systems. Not part of blockchain, but used with it.

What is Ethereum?

- **Ethereum is a blockchain that allows you to run programs in its trusted environment.**
- contrasts with the Bitcoin blockchain, which only allows you to manage cryptocurrency.
- Ethereum has a virtual machine – Ethereum Virtual Machine (EVM).
  - A decentralized Turing-complete VM that serves as the runtime environment for smart contracts on Ethereum Blockchain
  - EVM very effective in preventing DOS attacks and confirms that the programs have no access to each other's state and ensures communication is established without any potential interference.
- This code is contained in "smart contracts"
- Ethereum maintains the state of the EVM on the blockchain.
- All nodes process smart contracts to verify the integrity of the contracts and their outputs.

Smart contracts

- **A smart contract is a self executing contract with terms of agreement directly written into the code.**
- Are deployed on the Ethereum network and execute on the EVM
- Smart contracts can accept and store ether, data, or a combination of both.
- Solidity programming language is used to develop smart contracts.
- Using the logic programmed into the contract,
- it can distribute that ether to other accounts or even other smart contracts.
- Example:
- Alice wants to hire Bob to build her a patio
- they are using an escrow contract (a place to store money until a condition is fulfilled) to store their ether before the final transaction.

Precompiled contracts

- A special type of contract that is pre-installed on the Ethereum network and can be used by other contracts without needing to be deployed separately.
- These contracts are typically used for operations that are computationally expensive, such as cryptographic operations, that would be too expensive to perform on-chain if they had to be executed within a regular contract.
- Are written in low-level programming languages and have already been compiled and optimized for execution on the EVM.
- Examples of precompiled contracts on Ethereum include, contracts that perform elliptic curve operations, hashing, and big integer arithmetic.
- Developers can save time and gas costs, as they don't have to deploy and compile their own version of the contract, and can rely on the optimized version that is already installed on the network.

Gas in Ethereum

- **Ethereum Gas** is the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network, e.g., to process transactions or use smart contracts
- Ethereum gas is denominated in gwei, short for gigawei, with one gwei equal to one billionth of an ETH (or $0.000000001 ETH or 10^{-9}$ ETH). Ethereum gas fees can only be paid in Ethereum's native token, Ether (ETH).

Determining Gas fees

Gas fee influenced by three variables

- **Complexity:** Specifically, the amount of gas required to process a transaction.
- **Base fee:** The reserve price below which a transaction is ineligible for processing on the blockchain.
- **Priority fee:** An optional incentive fee to expedite processing time. (It's like a theme park FastPass or a tip to the headwaiter.)

How Ethereum Gas fees work

1. **Initiate a transaction** - can use any platform or crypto exchange of choice to initiate a transaction—to purchase ETH.
2. **Approve an estimated gas fee** - The platform automatically evaluates current Ethereum gas fees and current network demand to suggest a gas fee that balances cost with transaction speed.

- **The transaction is sent to the Ethereum blockchain** - the ETH purchase order is sent to the Ethereum network. It includes an upper limit on the total gas fee that one is willing to pay.
- **A proof-of-stake validator creates a new block** - Ethereum network validators assemble new blocks for the Ethereum blockchain by selecting which transactions to validate.

How Ethereum Gas fees work (2)

- **Digital wallet balance updates** - once the transaction gets published to the blockchain, the Ethereum balance updates and the gas fee is withdrawn. *The transaction total is the ETH purchase amount plus the gas fee.*
- **The validator receives gas fees** - The validator that processed your transaction receives the tip portions of your gas fee and the gas fees from all the transactions in the block. The base fees are "burned" —removed from circulation—to prevent ETH currency inflation.

Gas fee schedule

Gas is charged in three scenarios as a prerequisite to the execution of an operation:

- The computation of an operation
- For contract creation or message call
- Increase in the usage of memory

Messages

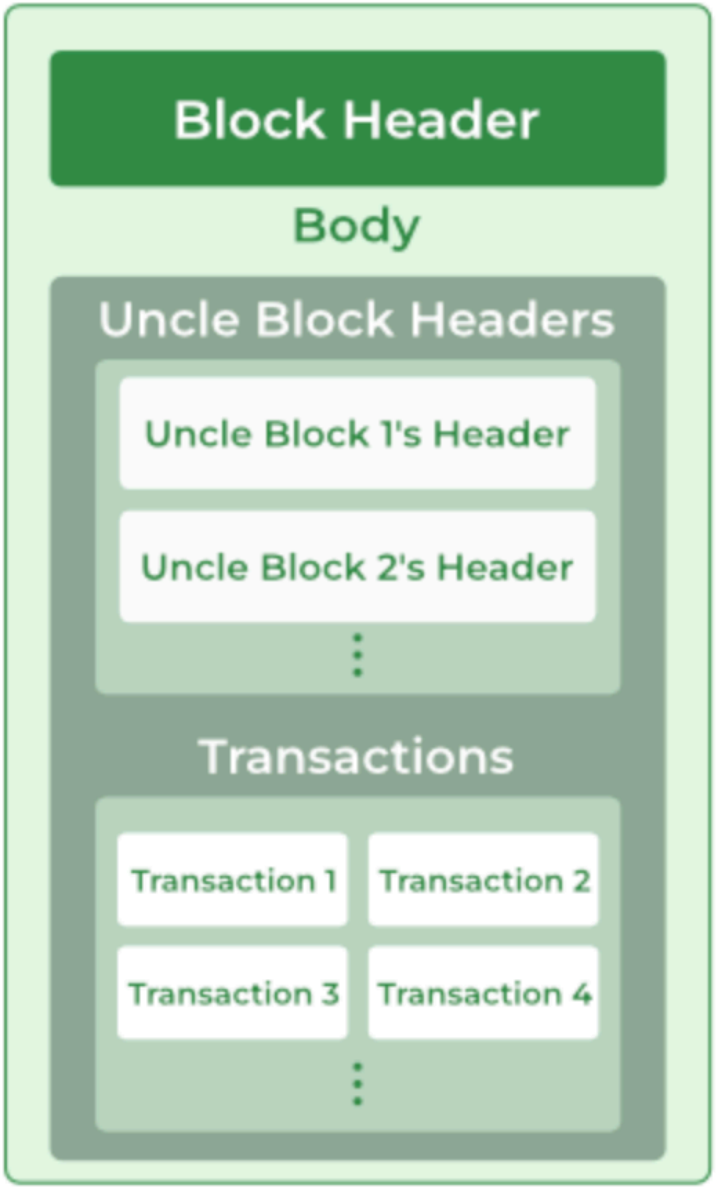
- A message is a data packet passed between two accounts. This data packet contains data and value (i.e, amount of ether).
- Contracts can send messages to other contracts.
- Messages only exist in the execution environment and are never stored.
- Messages are similar to transactions; however, the main difference is that they are produced by the contracts, whereas transactions are produced by entities external to the Ethereum environment.

Message components

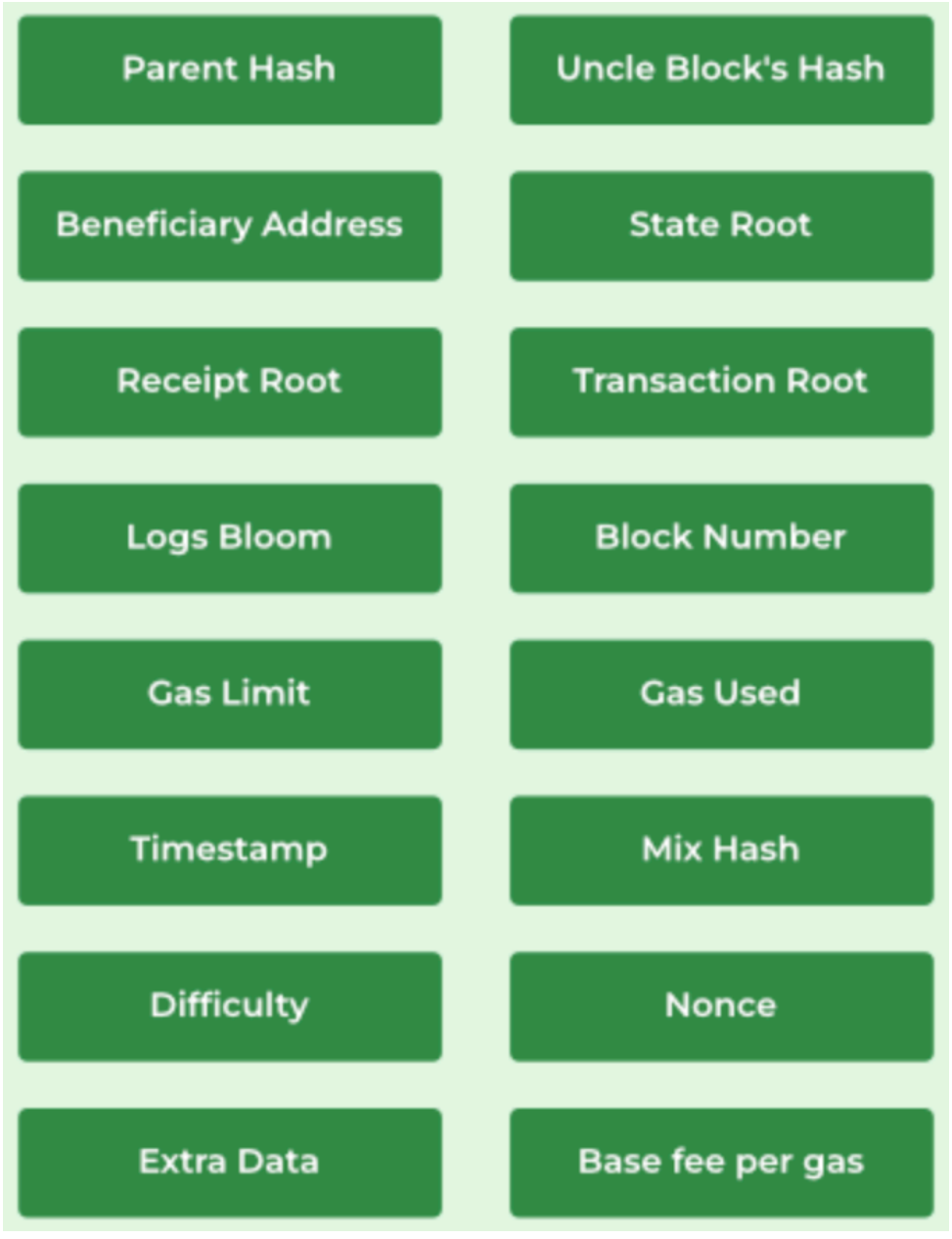
- Sender of the message
- Recipient of the message
- Amount of Wei to transfer and message to the contract address
- Optional data field (Input data for the contract)
- Maximum amount of gas that can be consumed

Ethereum block

- Every block in Ethereum consists of 2 main parts, Header, Body



Ethereum block - header



Ethereum Mining

- Need a crypto wallet that can hold the rewards once mined. Options, include; Trezor One, **MetaMask**, Ledger S Nano, Exodus, Mist
- Need a powerful computer system that can mine fast enough to harvest a profit. At least 256 GB of storage and RAM, 8 or 16 GB. Ethereum mining needs a minimum runtime memory of 4GB per GPU to perform optimally.
- Need to choose mining strategy e.g.,
  - **cloud mining:** cloud platforms include, ECOS and StormGain
  - **Pool mining:** Pool mining involves teaming up with other miners and clubbing together all of your resources.
  - **Solo mining:** Can mine alone. Have to bear all the costs
- Need to install necessary mining software. Options include, Go Ethereum, Minedollars, MinerGate, Cudo Miner, EasyMiner, Ethermine, Phoenix Miner, Kryptex, NBMiner, GMiner

Ethereum Mining - Cost

- Costs of the hardware necessary to build and maintain a mining rig (https://miningcave.com/product-category/rig-kit/) (https://buybitcoinworldwide.com/mining/hardware/)
- Electrical cost of powering the mining rig, i.e., Most newer model ASIC Ethereum mining machines require at least a 220-volt 20-amp electricity circuit. Also, ETH miners can be very loud with noise levels above 70 dB.
- If you were mining in a pool, these pools typically charged a flat % fee of each block generated by the pool
- Potential cost of equipment to support mining rig (ventilation, energy monitoring, electrical wiring, etc.)
- Ether is available at various exchanges for buying and selling: https://app.shapeshift.com/#/connect-wallet?returnUrl=/trade,

Ethereum networks

- On the **MainNet**, data on the chain—including account balances and transactions—are public, and anyone can create a node and begin verifying transactions.
- Ether on this network has a market value and can be exchanged for other cryptocurrency or fiat currencies like US Dollars.
- But there are other Ethereum networks as well.

Other Ethereum networks

- The Ethereum blockchain can be simulated locally for development.
- Local test networks process transactions instantly and Ether can be distributed as desired.
- An array of Ethereum simulators exist; https://trufflframework.com/ganache
- Developers use public test networks (or testnets) to test Ethereum applications before final deployment to the main network.

Private/Enterprise networks

- Private Ethereum networks allow parties to share data without making it publicly accessible.
- A private blockchain is a good choice for:
  - Sharing of sensitive data, such as health care records
  - Scaling to handle higher read/write throughput, due to the smaller network size
- An example of a private enterprise blockchain is Quorum, `https://consensys.io/`, originally written by J.P. Morgan.

Supporting protocols

Several protocols are in development to support the complete decentralized ecosystem.

- **Whisper:** provides decentralized peer-to-peer messaging capabilities to the ethereum network.
- **SWARM:**Swarm is being developed as a distributed file storage platform. It is a decentralized, distributed, and peer-to-peer storage network. Files in this network are addressed by the hash of their content

Distributed Applications (Daaps)

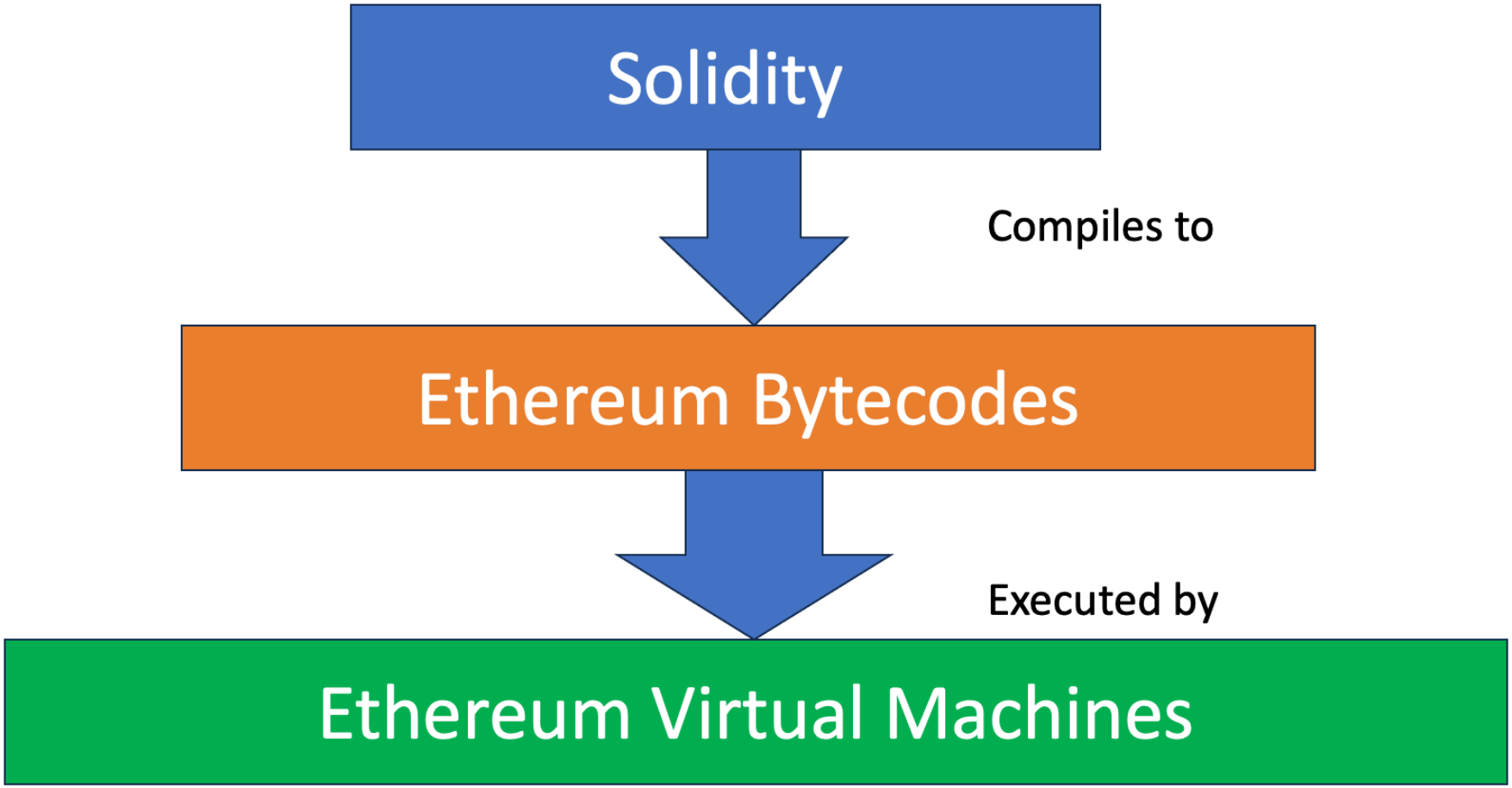
- Applications using smart contracts for their processing are called "distributed applications", or "dapps".
- The user interfaces for these dapps consist of familiar languages such as HTML, CSS, and JavaScript.
- The application itself can be hosted on a traditional web server or on a decentralized file service such as IPFS (`https://ipfs.tech/`), Swarm (`http://swarm-gateways.net/bzz:/theswarm.eth/`).
- Dapps based solution available for:
  - Record keeping
  - Finance
- Supply chains
  - Real estate
  - Marketplaces
- Examples of Daaps, BitTorrent, Uniswap, Melonport

Solidity - Langauage of smart contracts in Ethereum

What is Solidity?

- High-level object-oriented language for smart contracts
- Solidity was initially proposed in August 2014 by Gavin Wood
- Solidity lets you program on Ethereum, a blockchain-based virtual machine
- Solidity is a statically typed programming language
- A Contract programming language that has similarities to Javascript and C
- Contract-specific features include modifier (guard) clauses, event notifiers for listeners, and custom global variables.
- Solidity is compiled to bytecode that is executable on the EVM

Solidity


Solidity
Compiles to
Ethereum Bytecodes
Executed by
Ethereum Virtual Machines

Solidity (2)

Advantages

- Object oriented language support classes, inheritance, libraries, and complex user-defined types
- High-level language, and human readable code
- Large community support
- A lot of developer tools availabe, e.g., Remix, Truffle Suit, Hardhat, Brownie, Etherlime, Solhint, etc.
- Static type language, means every data contains a type before storing it
- Everything is contracted oriented

Disadvantages

- Once contract is created, adding features to the existing contract is not possible
- Solidity is the latest and young, so some bugs still exists
- It does not have a floating type

Solidity (3)

File extension

- File extension tells the type of the file. Smart contracts created with a file name and .sol extension
- For example, RP_BTech_Class.sol is a RP BTech Class smart contract

Setting up Solidity compiler

- Remix: Remix IDE provides plugins and development environment for smart contracts
- Remix can be used online without installing any software for the environment.

Data types in Solidity

- **uint:** Unsigned integer types
- **int:** Signed integer types
- **bool:** Boolean values - variable is true or false
- **address:** It's used to store the addresses of users or other contracts
- **enum:** User-defined types for creating a set of named constants
- **bytes and bytes32:** All other types are stored in form of bytes
- **strings:** A collection of one or more characters
- **arrays:** A collection of data of same type, like unit[] or bool[];
- **mappings:** key-value stores where data is organized in a mapping from keys to values
- **structs:** User-defined data structures that contain a combination of different data types

Solidity variables: visibility

- Variables can be declared with different visibility modifiers, e.g., public, private, internal, which determines who can access the variable.
- Solidity supports two types of visibility, State variable visibility and Function visibility

- Public: accessible by everyone, including the contract itself
- Private: only accessible by defined contracts
- Internal: Only accessible by defined / derived contracts
- External: can be called from other contracts and via transactions. **Note** - external visibility only usable in Functions.

Solidity syntax

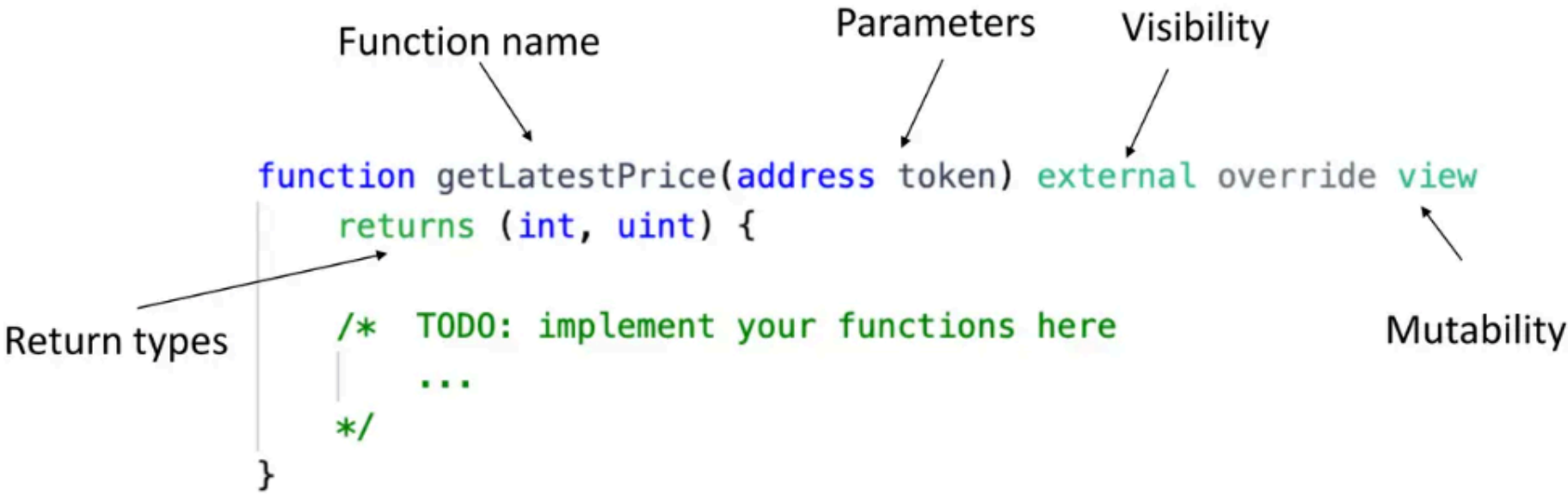
```
pragma solidity >=0.4.0 <0.6.0;

Contract Name{

        //variable declaration
        //function
}
```

Compiler Version

Solidity Contract

Codes / functions

Solidity syntax (2)



Solidity syntax (3)

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

Compiler version

Storage / state

Codes / functions

Solidity syntax (4)

Definition (State Mutability)

Refers to the ability of a function to modify the state of a contract, i.e., contract variables that can be read or modified by its functions

Definition (Pure functions)

A function that does not read or modify the state of a contract. Typically used for mathematical or string operations and executed locally.

Definition (View functions)

A function that can read the state of a contract but cannot modify it. typically used to retrieve data from contracts.

Definition (Payable functions)

A function that can receive Ether in a contract. Typically used for financial transactions.

Solidity syntax (5)

Definition (Non-payable functions)

A function that cannot receive Ether in a contract. Typically used to modify the state of a contract.

- Solidity is statically typed `https://docs.soliditylang.org/en/v0.8.26/`
- supports inheritance, libraries, and complex user-defined types
- Similar to Javascript syntactically

- To learn solidity go to `https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.25+commit.b61c2a91.js` and you can start programming smart contracts without having to create your own Ethereum network
- Can also check: `https://www.dappuniversity.com/articles/solidity`

Program execution

- When an Ethereum block is "mined",
- the smart-contract deployments and function calls within that block get executed on the node that mines the block
- the new state changes to any storage spaces or transactions within that smart-contract actually occur on that miner node.
- As the new block gets propagated to all the other nodes
- Each node tries to independently verify the block,
- Verifying includes doing those same state changes to their local copy of the blockchain
- it will fail if the smart-contract acts non-deterministically.
- If the other nodes cannot come to a consensus about the state of blockchain after the new block and its contracts get executed, the network could literally halt.

Program execution (2)

- EVM smart-contracts cannot access data outside the "memory" , and "storage"
  - (we don' t want the smart-contract to be able to read or delete the hard-drives of the nodes it runs on)
- Cannot query outside resources like with a JQuery.
- Do not have access to many library functions like for parsing JSON structures or doing floating-point arithmetic,
  - it' s actually cost-prohibitive to do those sub-routines or store much data in the Ethereum blockchain itself.

Program execution (3)

- When you call a smart-contract that does some state-changing work or computation, you will incur a **gas "cost"** for the work done by the smart contract
  - this gas cost is related to the amount of computational work required to execute your function.
  - sort of a "micropayment for microcomputing" system, where you can expect to pay a set amount of gas for a set amount of computation, forever.
- The price of gas itself is meant to stay generally constant, meaning that when Ether goes up on the global markets, the price of gas against Ether should go down.

Program execution (4)

- When you execute a function call to a smart-contract, you can get an estimation of the amount of gas you must pay beforehand, but you must also specify the price (in ether per gas) that you are willing to pay
- the mining nodes can decide if that' s a good enough rate for them to pick up your smart-contract function call in their next block.

Addresses of Solidity Smart contracts

- Smart-contracts have their own address, from which they can receive and send Ether.
- Smart contracts can track the "caller" of the function in a verifiable way,
  - it can determine if one of its functions is being called by a privileged "owner" or "admin" account, and act accordingly for administrative functions.
- They have the ability to read data from the Ethereum blockchain, and access info on transactions in older blocks.

Getting data outside the Blockchain

- But are smart-contracts "locked in" into their own little deterministic world, only able to be aware of data stored in the Ethereum blockchain itself?
- We can make a call to an oracle that will tell us something about the outside world in a trustable way, and act on that data within the smart contract.
- even though real-world events themselves are not deterministic, the **Oracle** can be trusted to always answer every node' s request about what happened in a deterministic way
- so that all nodes can still come to a consensus.
- An "oracle" will take some data, say a ticker price feed about a real-world stock price, and record that data into "storage" in a simple Oracle smart-contract,

Ethereum native currency vs tokens

- Ethereum technology supports hosting of other cryptocurrencies known as 'Tokens' directly on the blockchain.
- A token is a digital asset on top of a cryptocurrency or blockchain, often as a programmable asset managed by a smart contract, for use within a project or dApp
- Anyone can build a token on the Ethereum blockchain. No permissions or licenses are required
- Common types of tokens are,

ERC20 Tokens

- **Ethereum Request for Comment - 20 (ERC20) standard**: covers tokens such as, USDT, DAI, LINK, UNI, SHIBA INU, etc.
- ERC20 tokens are **fungible**, e.g. like a dollar bill, two ERC20 tokens have the same value regardless of who uses it and where it is used
- Crypto projects use tokens to attract users and build communities to supply liquidity and to create governance structures.

Ethereum native currency vs tokens

NFTs (ERC721)

Non-Fungible Tokens (NFTs): are not fungible liek ERC2o tokens.

No two NFTs are alike, as such if you hold one, it absolute proof that you are the owner of whatever the NFTs presents.

NFTs can be applied to both digital objects and real-world asserts

Security Alert - Approvals to DApps in your wallet

- Some DApps can request unlimited permissions - implying you could end up spending much more than you intend
- Whenever a DApps requests that you sign a message, be careful to consider whether you trust the DApp
- Avois interacting with NFTs platforms or exchanges that you are not familiar with - Always use reputable secure wallet, never give anyone your keys - double check urls and certificates of websites

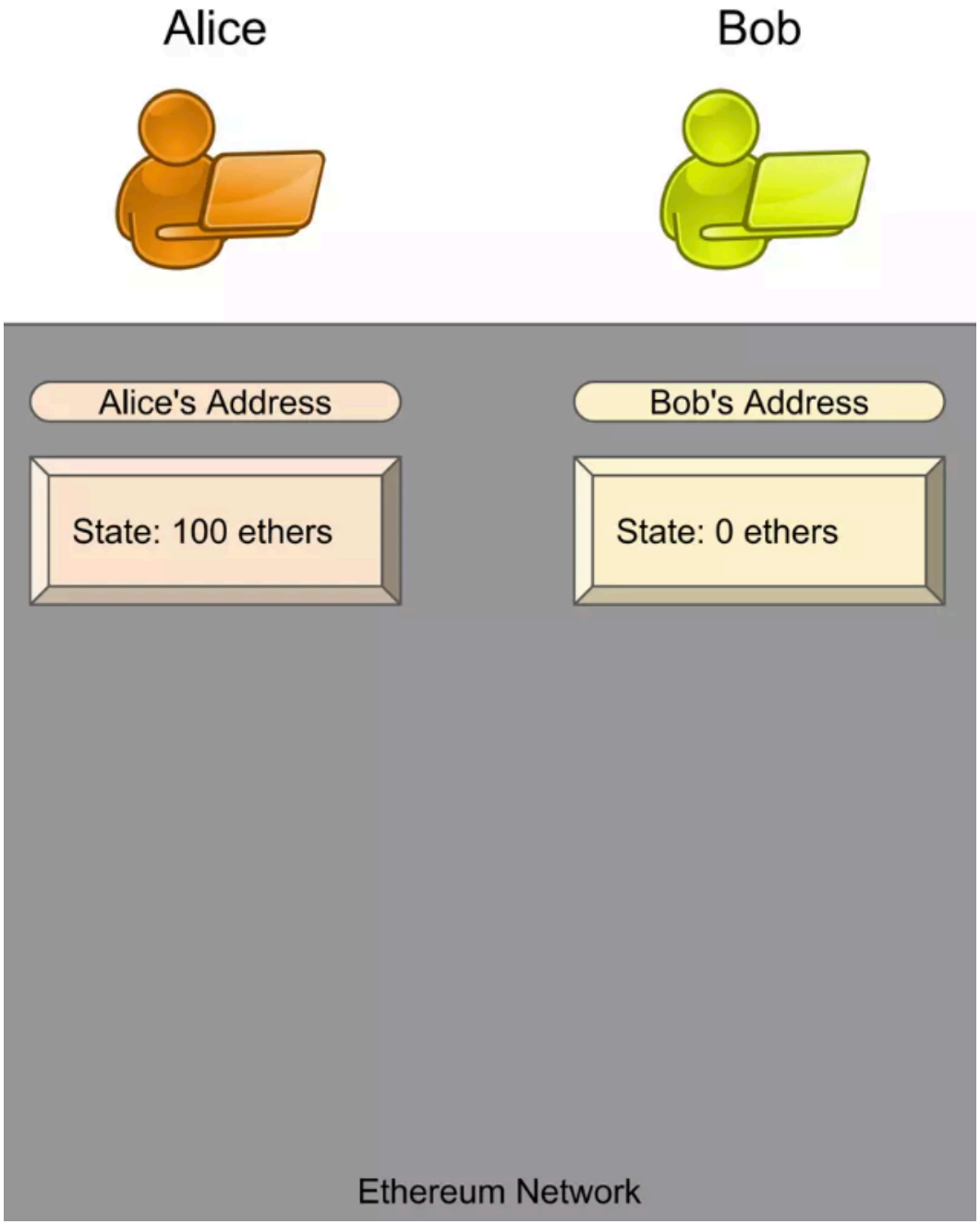
Ethereum native currency vs tokens

Native Currency	Token
Built-in currency inside a cryptocurrency system, which has its own infrastructure holding the ledger.	A contract keeping a balance table and running on top of a contract platform.
The ledger is handled by a Distributed Ledger Technology (DLT). The most common DLT nowadays are blockchain. But there are others such as Directed Acyclic Graph (DAG).	Not related to any DLT.
Being "distributed" and "decentralized" because the native currency is a "state", which is agreed across the whole infrastructure (every node agrees on this state).	Being "distributed" and "decentralized" only because the underlying contract platform is distributed and decentralized. The contract is still held by a contract owner, and from this point, Token is not completely decentralized.
Examples: Bitcoin, Ethereum, Litecoin, Zcash, IOTA.	Examples: EOS, TRON, OmiseGO, which are tokens on top of Ethereum smart contract platform.

Native currency

- In Ethereum, native currency (ethers) is a state associated to an address.
- For example, when we say Alice has 100 ethers and Bob has none, these are the states of the Ethereum network
- These states are agreed across the Ethereum network
- The state is part of the blockchain: the ethers will be there as long as Alice has the private key to access it

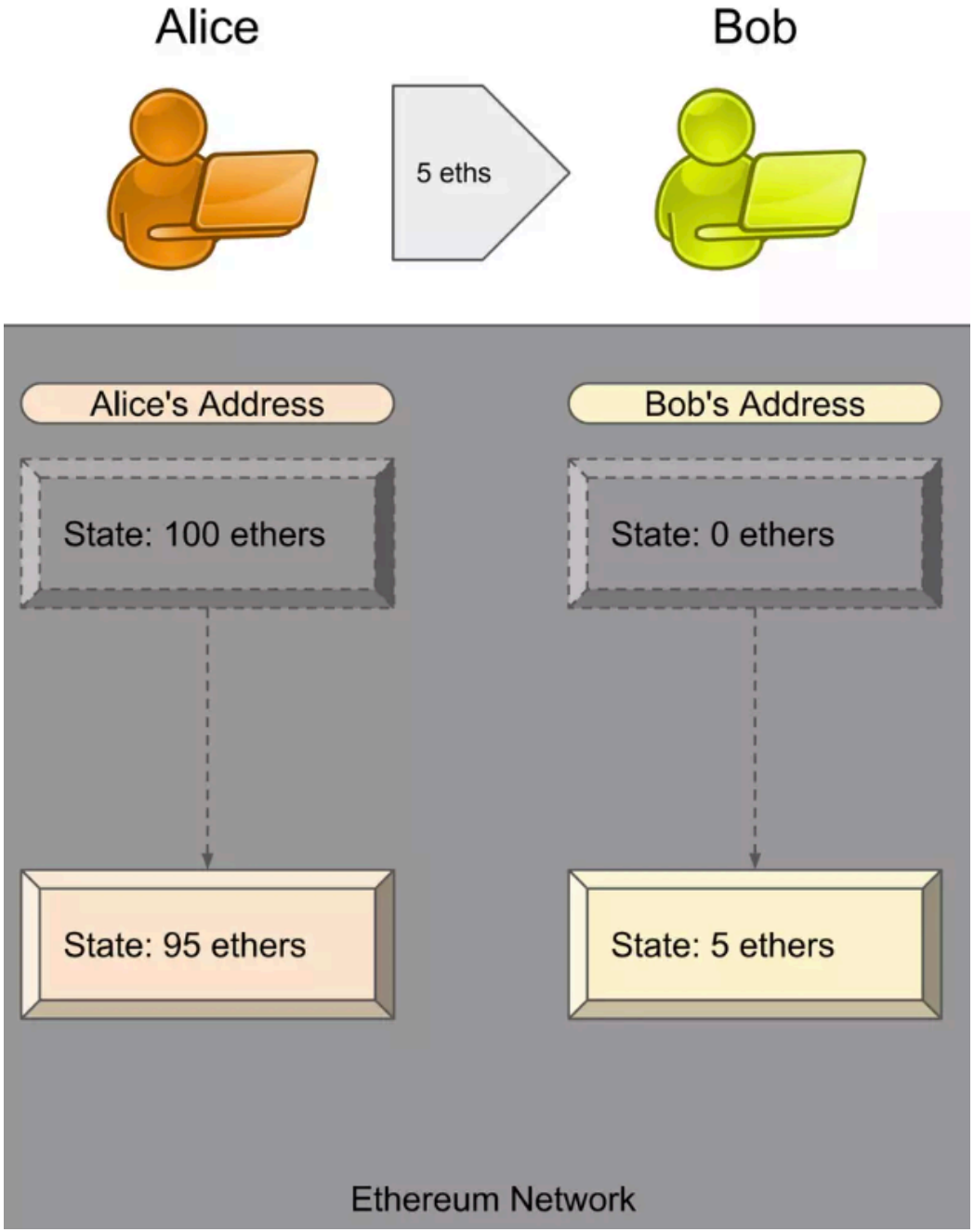
Native currency (2)



Native currency (3)

- Transfer of Ether is a transaction inside Ethereum network, and triggers a state change
- Say Alice is sending 5 ethers to Bob. It is done natively in Ethereum network
- The transaction is confirmed by miners and the state will be updated and the whole Ethereum network agrees to this change

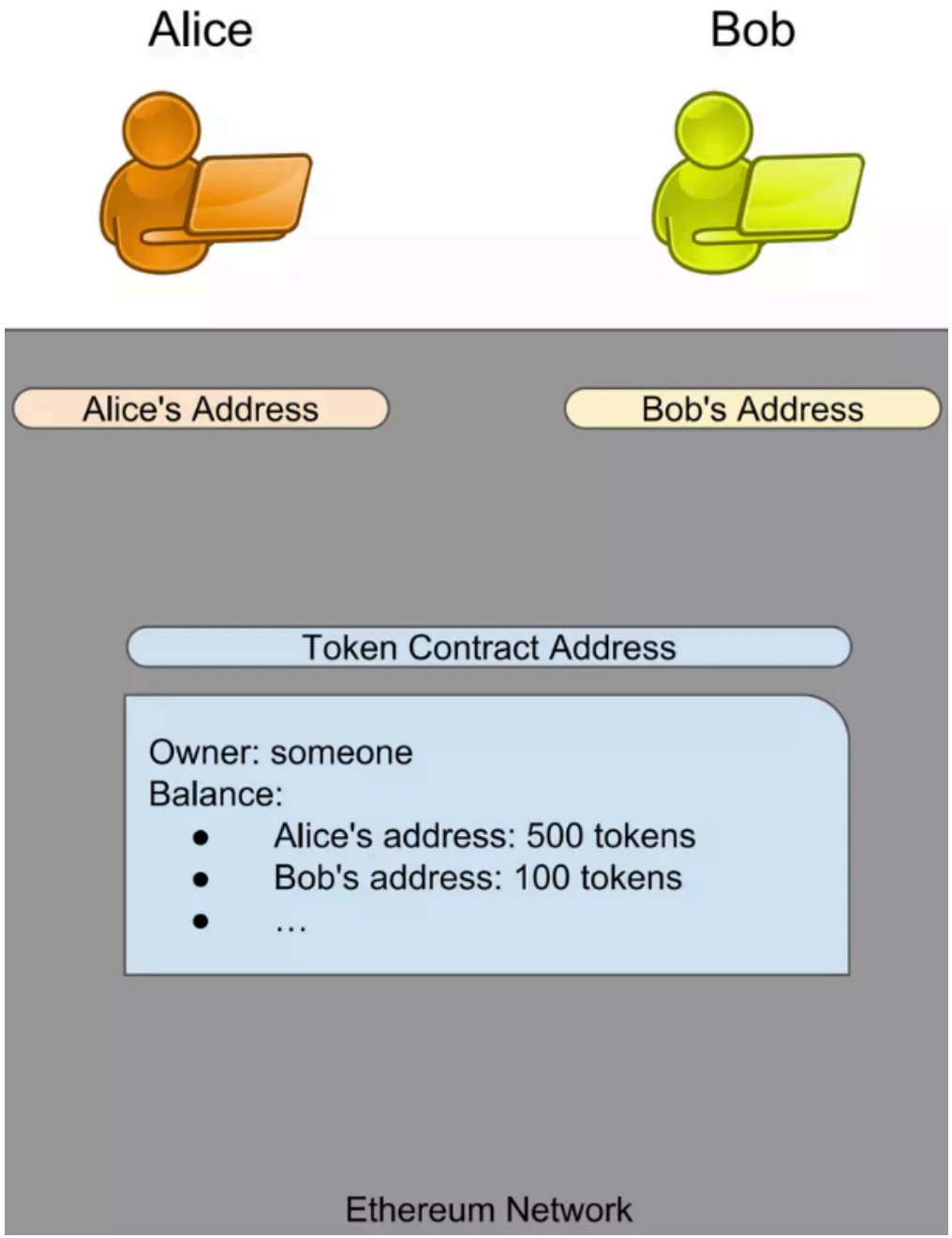
Native currency (4)



Tokens contract

- Assume a token contract is deployed on the Ethereum network. This contract is represented by a Contract Address
- If someone holds some tokens, it means, it is an entry of record stored inside this deployed contract
- In the figure (next slide) we see that Alice has 500 tokens and Bob has 100 tokens. However, they do not represent the state of the Ethereum network, rather its just a record inside the Token Contract.

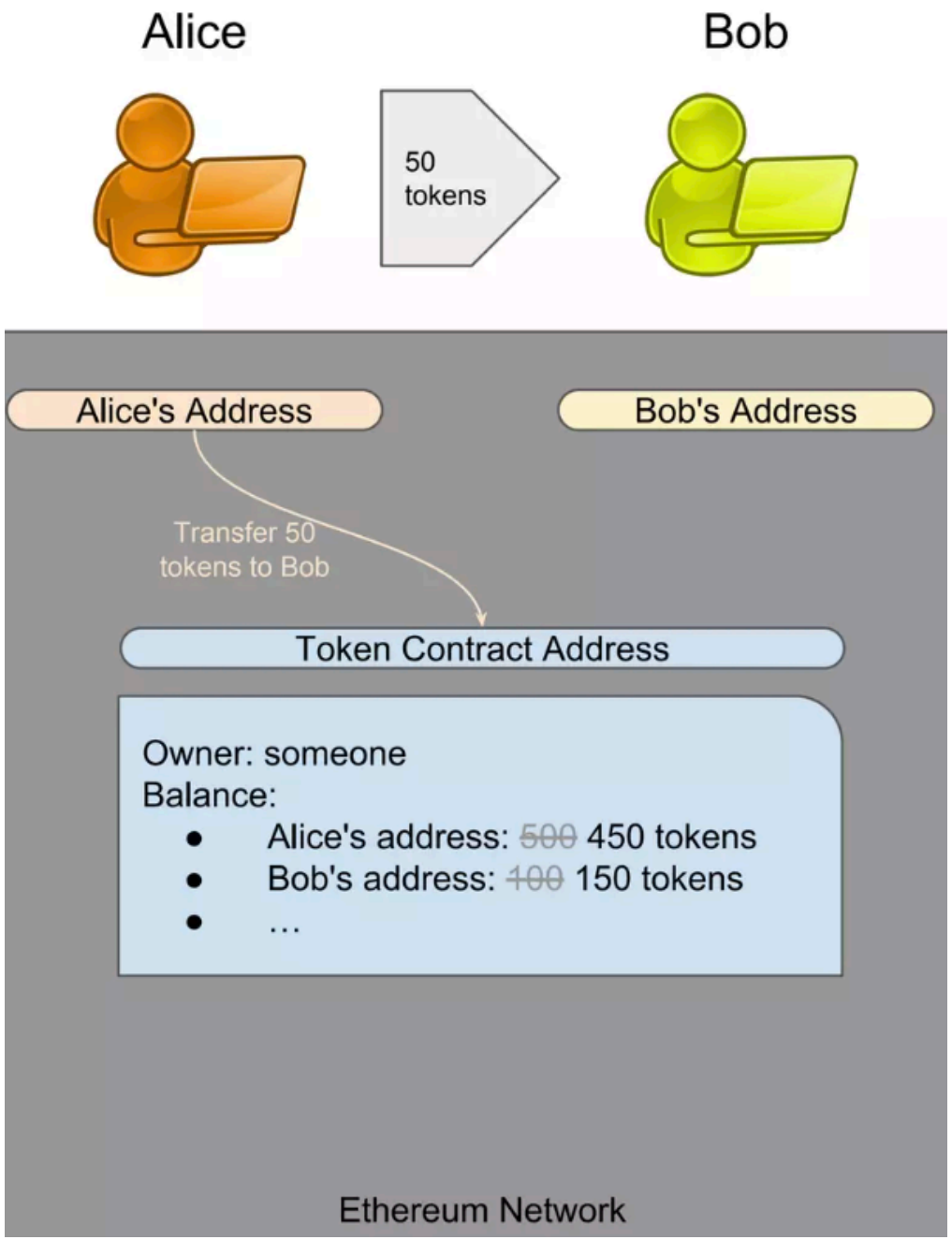
Token contract (2)



Token contract (3)

- Transfer of tokens is a function executed on the Contract
- Say, if Alice transfers 50 tokens to Bob, it is not done natively on Ethereum. Rather, Alice is executing a "transfer" function onto the Contract
- Upon receiving the transfer request, Contract updates the entries of both Alice and Bob, and the result is reflected on the updated balance record.

Token contract (4)

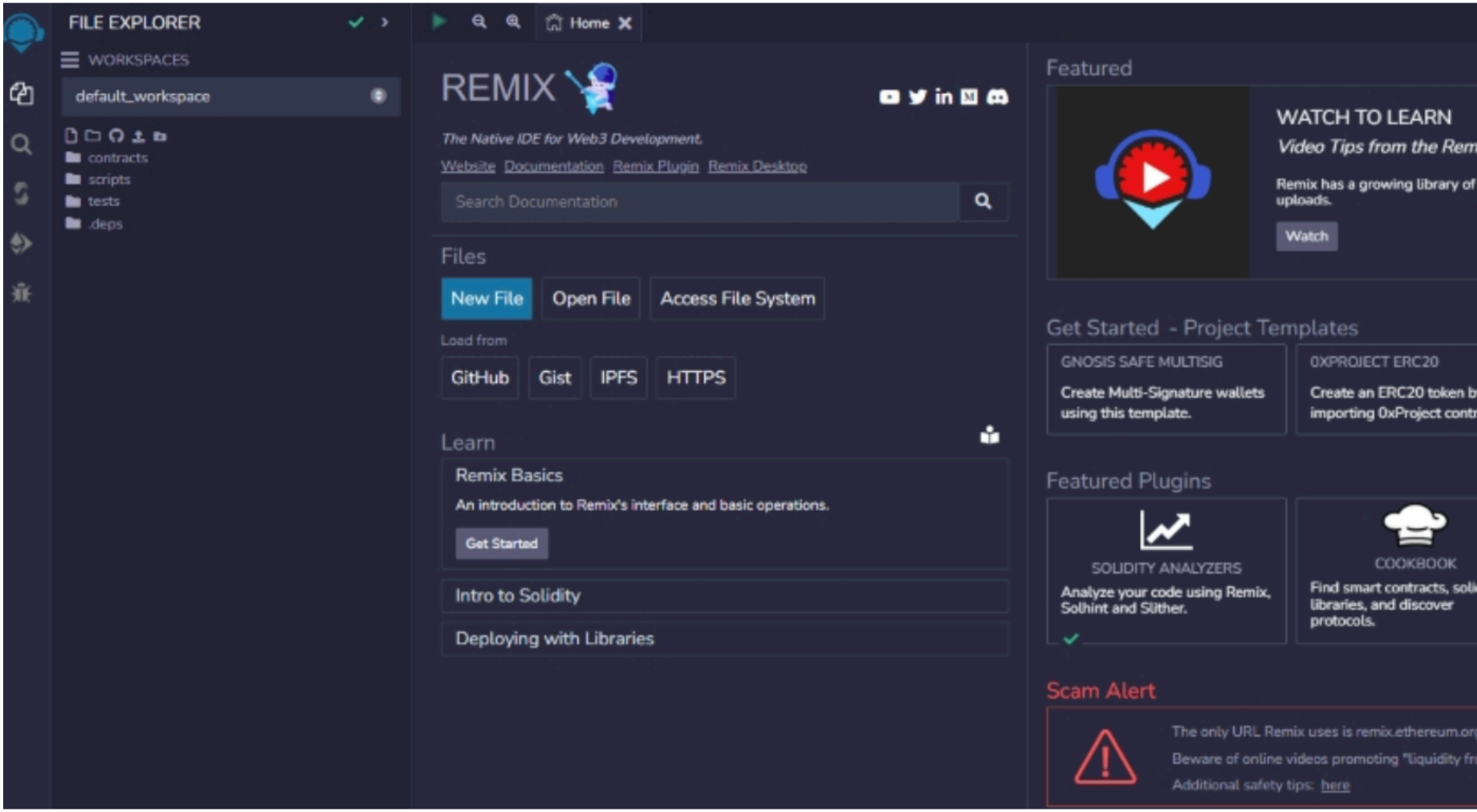


Some differences - Bitcoin Vs Ethereum

- Accounts and not UTXOs
- Merkle Patricia Trees
- RLP -Recursive Length Prefix -main serialization format
- Compression Algorithm
- Trie Usage
- Uncle Incentivization
- Difficulty Update Algorithm
- Gas and Fees
- EVM

Remix IDE

- Click: `https://remix.ethereum.org/`

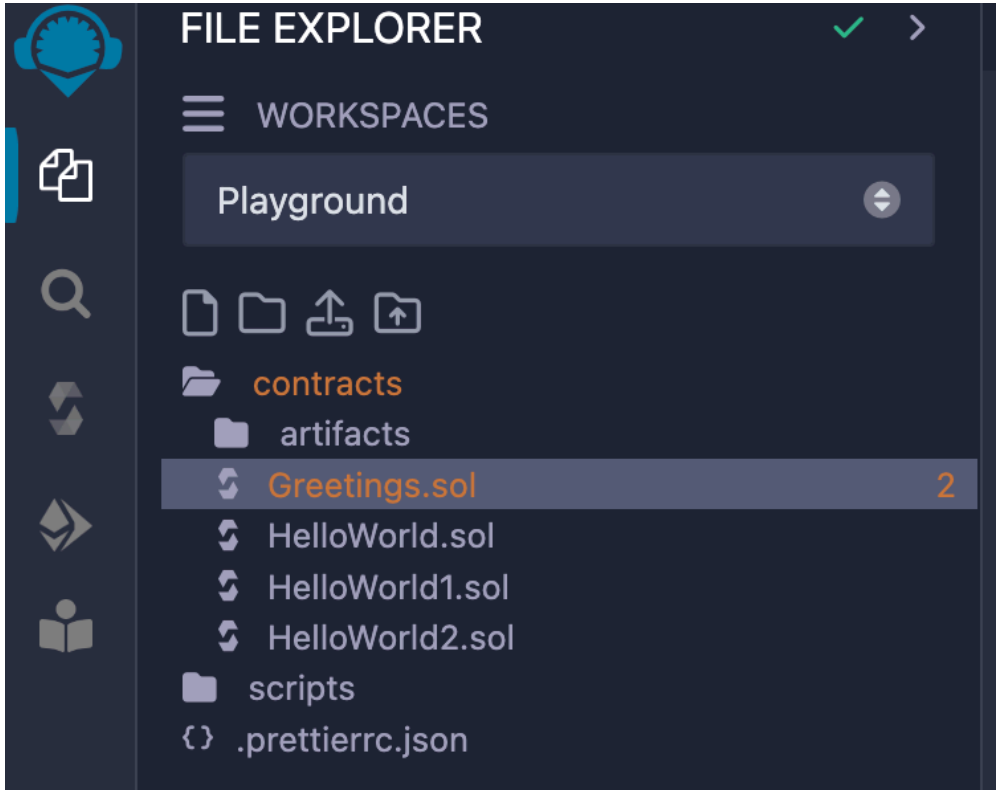


Remix IDE - Overview

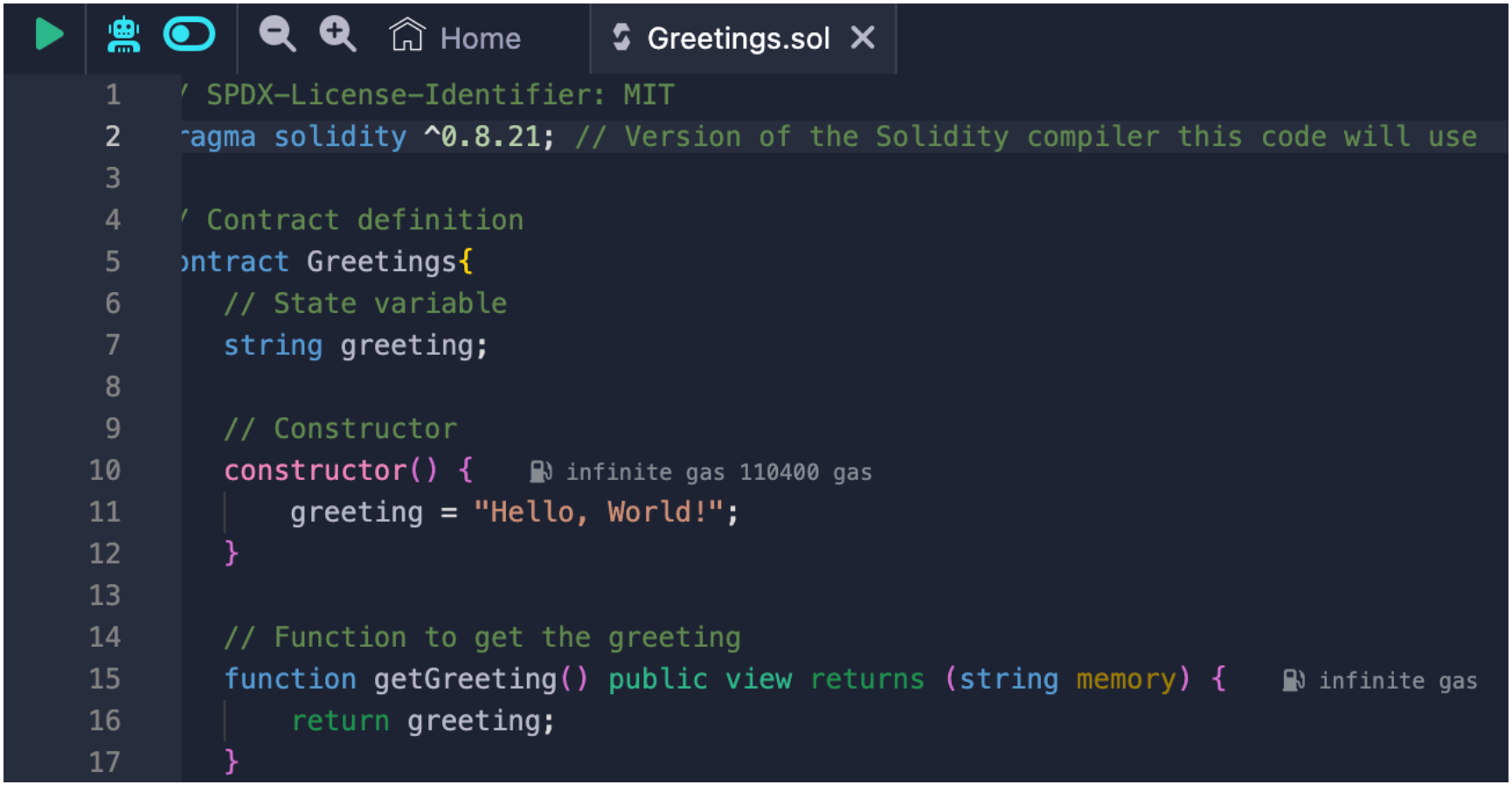
- **File Explorer:** This panel on the left allows you to manage your project files. You can create new contracts, import existing ones, and organize your project structure.
- **Editor:** The central area of the IDE is the code editor. This is where you write and edit your Solidity contracts. Remix provides syntax highlighting, auto-completion, and other helpful features to enhance your coding experience.
- **Compiler:** The Compiler panel allows you to configure the Solidity compiler version and compile your contracts. It displays the compilation results, including any warnings or errors.
- **Deploy & Run Transactions:** This panel is used for deploying and interacting with your smart contracts. You can select the contract you want to deploy, set constructor arguments, and execute functions within the contract.
- **Note:** Remix has its testing network where we can test our smart contracts as much as we want without jeopardizing our real money.

Remix IDE - Crate new file

- On the file explorer panel **on the left,** click on the contracts folder. Now to create a new file inside the contracts folder, **click on the Create New File button.** Name the file with something like Greetings.sol.



Remix IDE - write a contract



Remix IDE

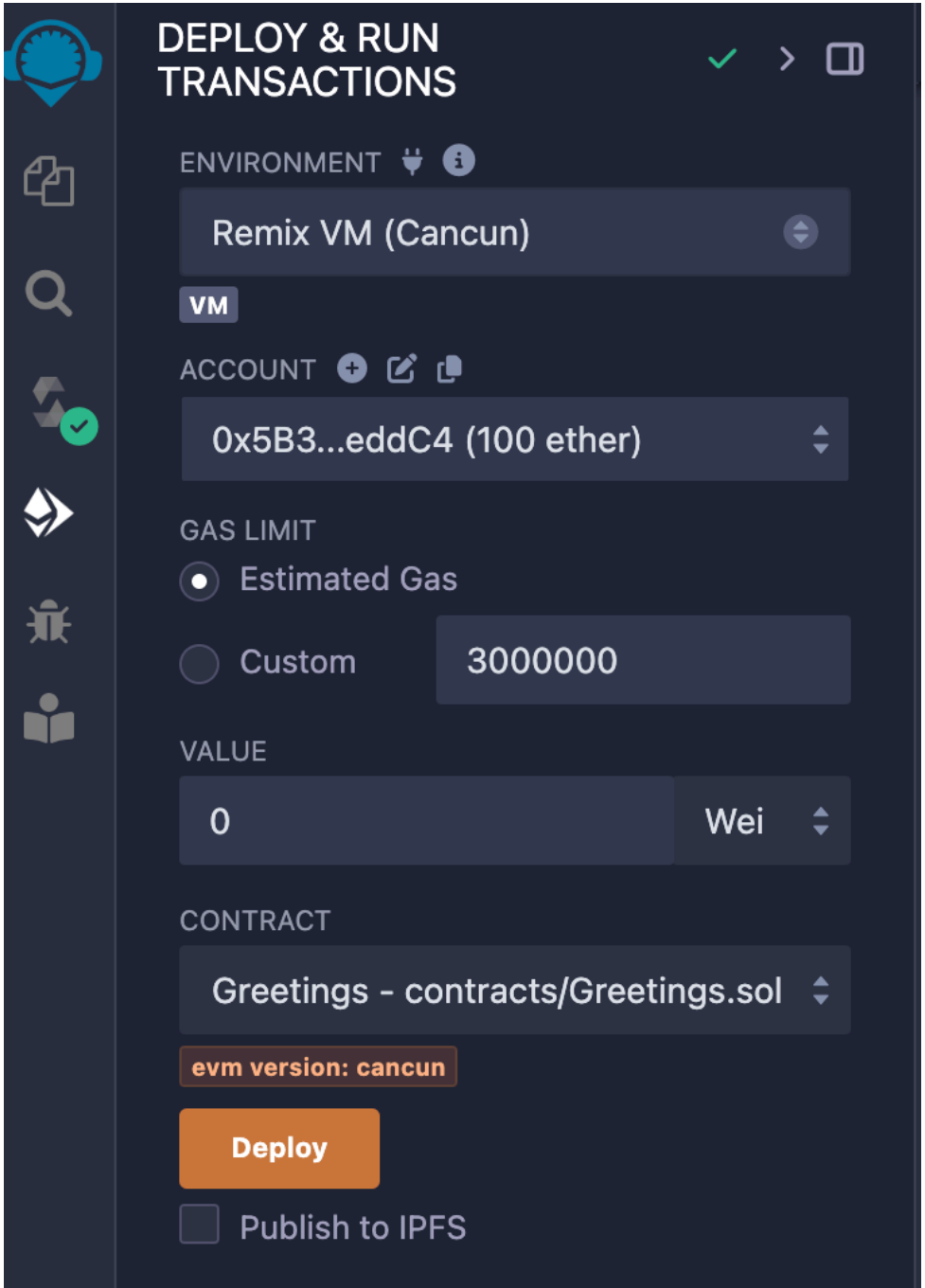
Code compilation

- If your contract compiles successfully, Remix IDE will generate a set of compiled artifacts. These include the bytecode (which gets deployed to the blockchain) and the Application Binary Interface (ABI), which defines how to interact with the contract.

Deploy contract

- Before deploying, decide which blockchain network you want to use. For beginners, starting with a testnet is recommended. This allows us to deploy and test our contract without using real cryptocurrency. There are many testnet available. You can select one of the available test networks and add it to your wallet, like Metamask. After adding the testnet to your wallet, you can get some faucets to it. These faucets will be required during the deployment of the smart contract.

Remix IDE


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Cancun)
VM
ACCOUNT
0x5B3...eddC4 (100 ether)
GAS LIMIT
Estimated Gas
Custom
3000000
VALUE
0
Wei
CONTRACT
Greetings - contracts/Greetings.sol
evm version: cancun
Deploy
Publish to IPFS

Connect Remix IDE to network

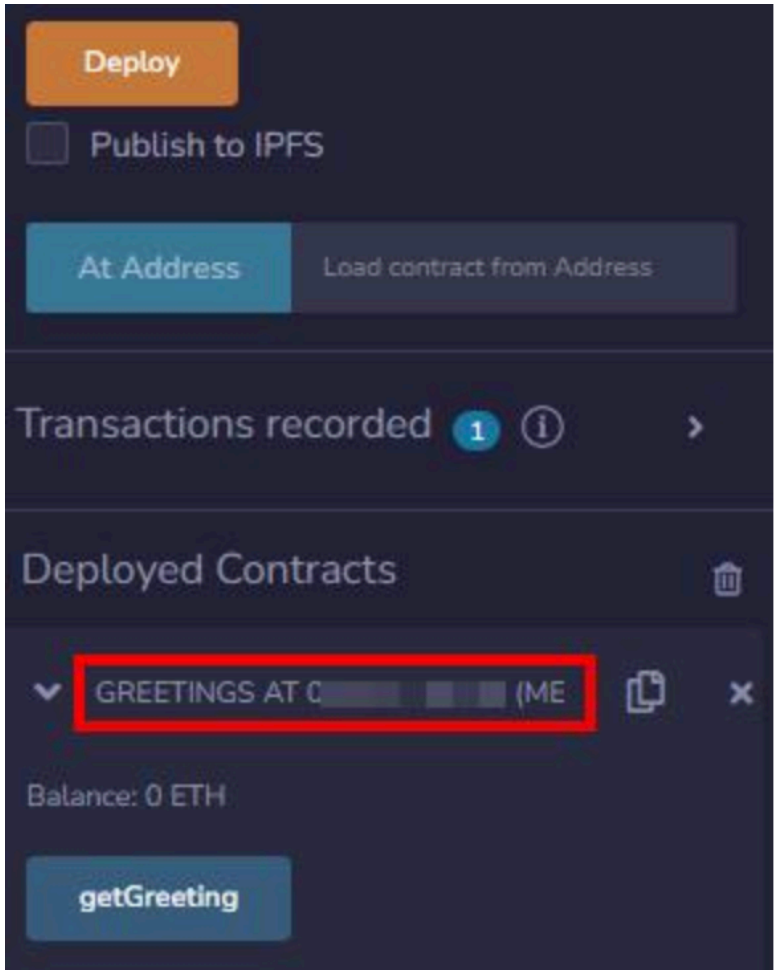
- By default, the environment selected by the Remix IDE is its own test environment. Remix IDE provides the user with its own test environment to test the smart contract before deploying it to the main network.
- But when deployed on the Remix test environment, the smart contract is not visible outside the Remix Ide. You will not be able to see the contract outside the Remix environment.
- To prevent this, we can deploy the smart contract on a publically available testnet like the Polygon Mumbai testnet, Goerli Testnet, or any other testnet.
- To do so, connect to the wallet on which we have connected with these networks. For example, to connect to Metamask, we can select **Injected Provider - MetaMask.**
- By selecting this, it will try to connect to your metamask wallet. Allow the connection, and you are ready to deploy on your desired network.

Remix IDE - deploy the contract

- Click the Deploy button. This will open up your metamask transaction builder if you are using any other network rather than the default environment provided by the Remix.
- Sign the transaction, and the Remix IDE will do the rest of the work automatically.
- **Please remember that for signing a transaction, you should have some faucet in your account if you are using a testnet.**
- After the transaction is successful, you can see it on the block explorer using its transaction hash.

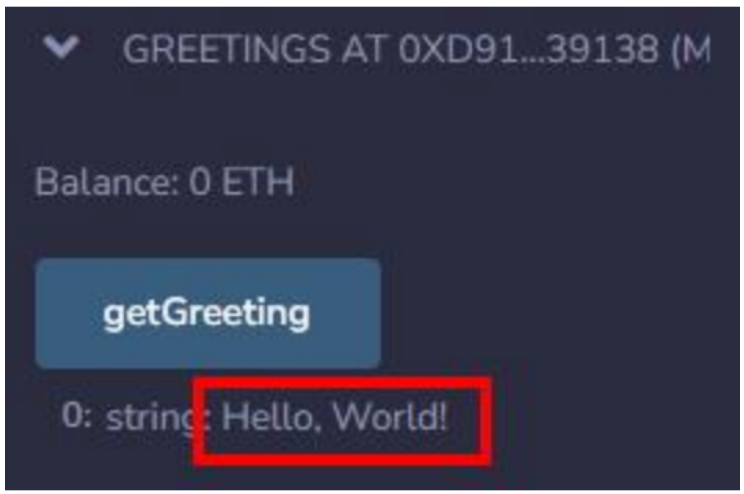
Remix IDE - deployed contract

- Once the smart contract is successfully deployed, you will be able to interact with its function in the deploy and run transaction panel.
- The deployed contract will be visible under the Deployed Contracts panel.



Remix IDE - Interacting with the contract

- Expand the deployed contract view to see all the functions available inside it. In this tutorial, the Greetings contract has only one function `getGreeting`, which you can see in the picture above.
- To call the function, click on the button with the function name. In the above example, when we click on the `getGreeting` function button, the message Hello World is reflected on the screen.



Solidity programming in Remix IDE

- Check: `https://www.dappuniversity.com/articles/solidity-tutorial`
- A good example on: `https://docs.horizen.io/horizen_eon/tutorials/todolist/`

Hyperledger

- **Hyperledger is not a blockchain**, but a project initiated by the Linux Foundation in December 2015 to advance blockchain technology.
- Project collaborators seek to build an open source distributed ledger framework that can be used to develop and implement cross-industry blockchain applications and systems.
- The project also focuses on improving the reliability and performance of blockchain systems.

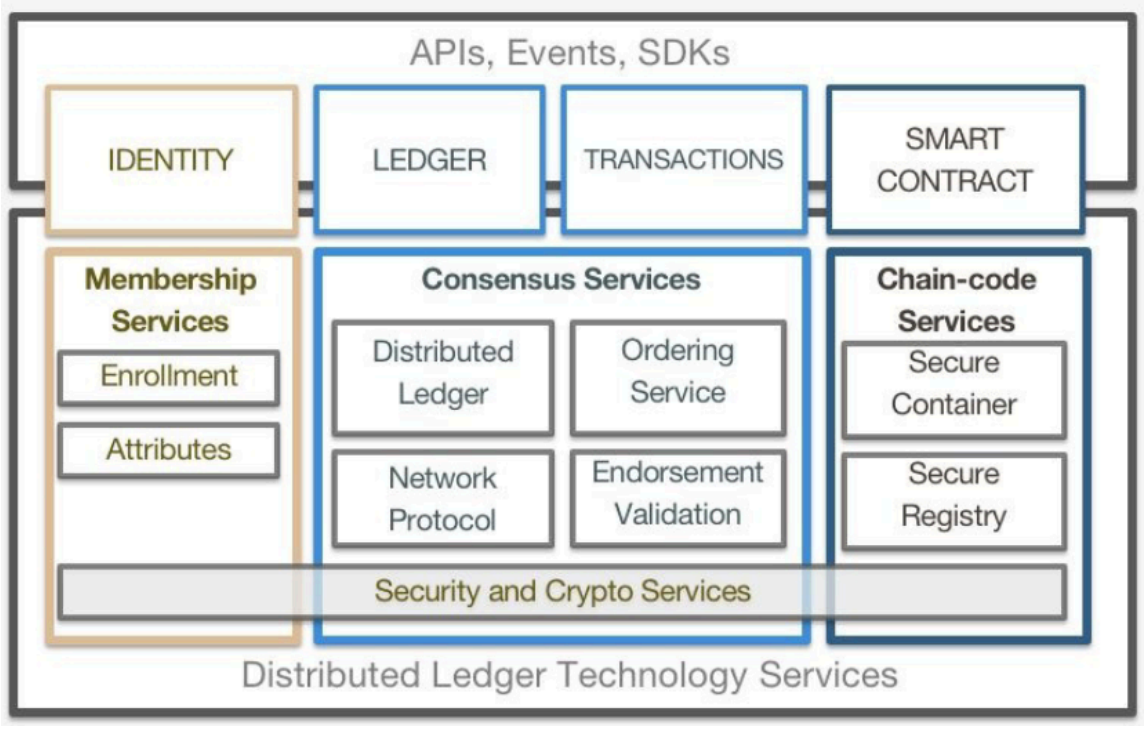
Projects under Hyperledger

- Four categories of projects exists with multiple projects under each category.
  - Distributed ledgers
  - Libraries
  - Tools
  - Domain-specific
- Currently, are six distributed ledger projects under the Hyperledger umbrella: **Fabric, Sawtooth, Iroha, Indy, Besu, and Burrow.**
- Under libraries, there are the **Aries, Transact, Quilt, and Ursa projects.**
- The tools category of Hyperledger includes projects such as **Avalon, Cello, Caliper, and Explorer.**
- There are also domain-specific projects such as Hyperledger **Grid** and Hyperledger **Labs.**
- Checkout:
  `https://www.hyperledger.org/projects#All-Projects`

Hyperledger projects

Hyperledger Fabric and its architecure

- contribution made initially by IBM and Digital Assets to the Hyperledger project
- Hyperledger Fabric is intended as a foundation for developing applications and solutions with modular architecture. It provides many benefits like permissioned networks, confidential transactions, etc.



Hyperledger Fabric components

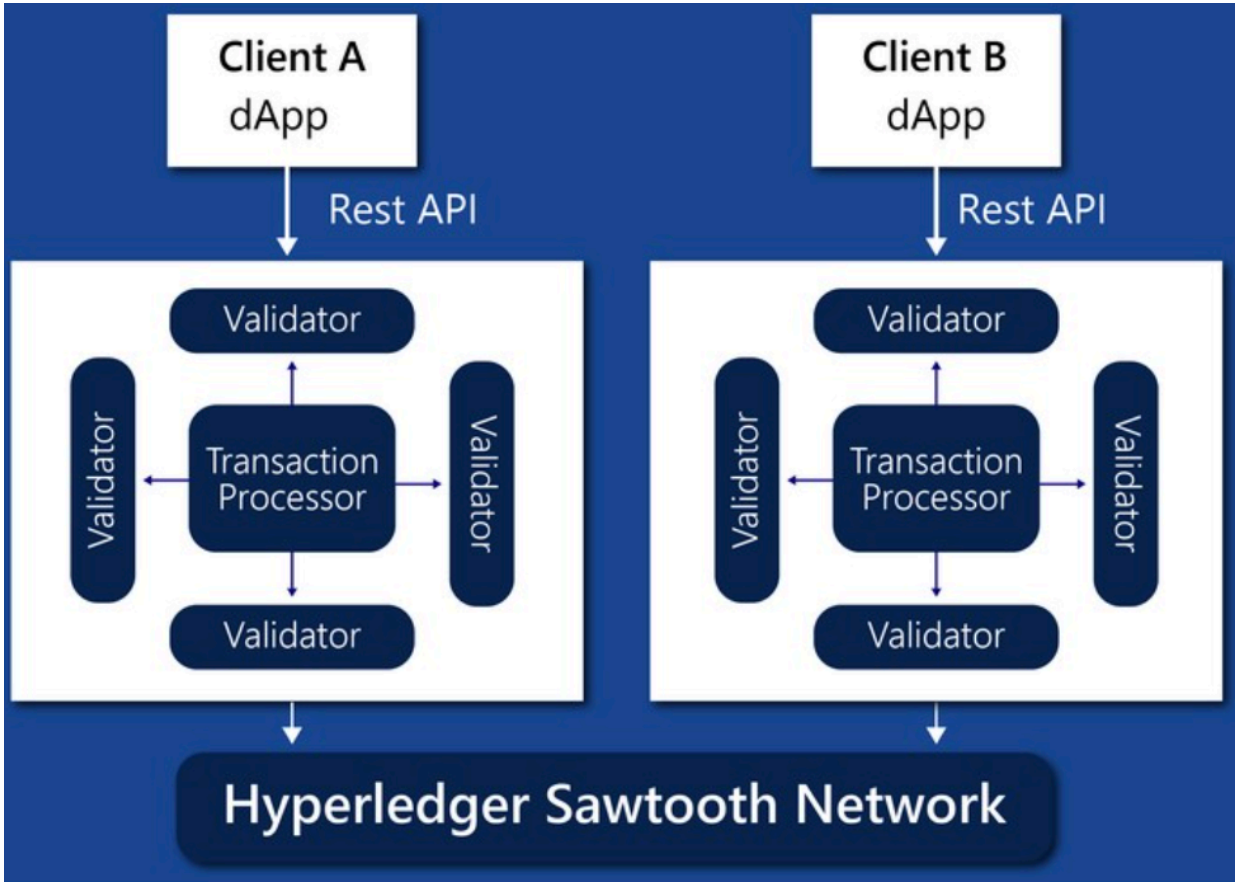
- **Membership services:** Provides a specialized digital certificate authority for issuing certificates to members of the blockchain network, and it leverages cryptographic functions provided by Hyperledger Fabric.
- **Transactions:** A request to the blockchain to execute a function on the ledger. The function is implemented by a chaincode.
- **Smart contract or chaincode services:** An application-level code stored on the ledger as a part of a transaction. Its logic is written as chaincode (in the Go or JavaScript languages), and executes in secure Docker containers.
  - Chaincode is installed on peers, which require access to the asset states to perform reads and writes.
  - The chaincode is then instantiated on specific channels for specific peers. Ledgers within a channel can be shared across entire networks of peers or include only a specific set of participants.

Hyperledger Fabric components (2)

- **Ledger:** The distributed encrypted ledger, includes an append-only data store. Provides the ability to query and write data across distributed ledgers.
  - *Level DB (default embedded KV DB)* supports keyed queries, composite key queries
  - *Couch DB (external option)* supports keyed queries, composite key queries, key range queries, plus full data rich queries
- **Client SDK:** A client SDK enables the creation of applications that deploy and invoke transactions atop a shared ledger.
  - The Hyperledger Fabric Reference Architecture supports both Node.js and Java SDK.

Hyperledger Sawtooth and architecture

- Proposed by Intel, designed to provide more, Modularity, Scalability, and support for Permissionless & Permissioned networks.
- Supports consensus algorithms, e.g., PBFT, and PoET.



Sawtooth network components

Rest API: medium for the interaction of the client with the validator. Sawtooth has a pragmatic RESTful API which provides language independence interface for submitting transactions and reading blocks.

Transaction Processor: smart contracts which act as the administrative head for the data. By controlling all the business logic and validations, it submits the transactions with some payload.

Validator: validates all the transaction, communicates between the client and the other validators, and the transaction processors.

Consensus Engine: consensus Engine enables more consensus options for Sawtooth node to executes all the transactions within the smart contracts.

Developing with Sawtooth Recommended read:

`https://learnthings.online/course/2020/03/06/hyperledger-sawtooth-for-application-developers`

Blockchain developers

Developer interaction with Hyperledger Fabric constructs can include the following

- The developer creates an application and a smart contract
- The application can invoke calls within the smart contract through an SDK
- The calls are processed by the business logic built into the smart contract through various commands and protocols:
  - A `put` or `delete` command will go through the selected consensus protocol and will be added to the blockchain
  - A `get` command can only read from the world state but is not recorded on the blockchain
- An application can access block information using rest APIs such as `get block height`

Blockchain developers (2)