ETHEREUM AND SMART CONTRACTS

Created: Sept 2022
Last Edited: Sept 2022

UofT: CSCD71F22
-David Liu, Founder of dApp Technology Inc.

Blockchain Development

SMART CONTRACT PLANNING

Smart contracts cannot be changed after it has been deployed!

They should be thoroughly planned out before coding.

SMART CONTRACT UPGRADABLE

Actually, there's a work around. We can have a smart contract factory the maintains code versions via another smart contract.

SMART CONTRACT PROPERTIES

Property	Description
Native Token Balance	ETH owned by the smart contract
Data State	Data stored in the variables via transactions
Code	Low Level compiled bytecode that could be executed
...	...

EXAMPLE SMART CONTRACT

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity ^0.8.9;
3
4  // Import this file to use console.log
5  import "hardhat/console.sol";
6
7  contract Lock {
8      uint public unlockTime;
9      address payable public owner;
10
11     event Withdrawal(uint amount, uint when);
12
13     constructor(uint _unlockTime) payable {
14         require(
15             block.timestamp < _unlockTime,
16             "Unlock time should be in the future"
17         );
18
19         unlockTime = _unlockTime;
20         owner = payable(msg.sender);
21     }
22
23     function withdraw() public {
24         console.log("Unlock time is %o and block timestamp is %o", unlockTime, block.timestamp);
25
26         require(block.timestamp >= unlockTime, "You can't withdraw yet");
27         require(msg.sender == owner, "You aren't the owner");
28
29         emit Withdrawal(address(this).balance, block.timestamp);
30
31         owner.transfer(address(this).balance);
32     }
33 }
```

SOLIDITY

- Strongly Typed
- Object Oriented
- Similar to JavaScript and Java
- filename.sol
- Current Version (Sept 2022): 0.8.17

FUNCTION VISIBILITY AND TYPES

external	Function can only be called from other contracts
internal	Function can only be called from the current contract
private	Same as internal but additional not visible to derived contracts
public	Function can be called internally or externally
view	Additional Type. Function does not write data and only returns data
pure	Additional Type. Function does not read or write data, only returns data
payable	Additional Type. Function call may also have Ether attached
virtual	Additional Type. Function overrides an inherited function.

MODIFIERS

```
contract Mutex {
    bool locked;
    modifier noReentrancy() {
        require(
            !locked,
            "Reentrant call."
        );
        locked = true;
        _;
        locked = false;
    }

    function f() public noReentrancy returns (uint) {
        (bool success,) = msg.sender.call("");
        require(success);
        return 7;
    }
}
```

When a function uses a modifier, the functions code will me modified, as if the code is moved to where the _ is

VARIABLE VISIBILITY AND TYPES

public	Data can be accessed externally and internally. Also, getters are auto generated
internal	Data can only be accessed within the contract. NOTE: Data is still visible.
private	Same as internal but not visible in derived contracts. NOTE: Data is still visible.
string	A list of characters
bool	true or false
int	Positive or negative number. No Decimal
uint	Positive number. No Decimal
address	Unique identifier for accounts and contracts
enum	User defined type

REF TYPES

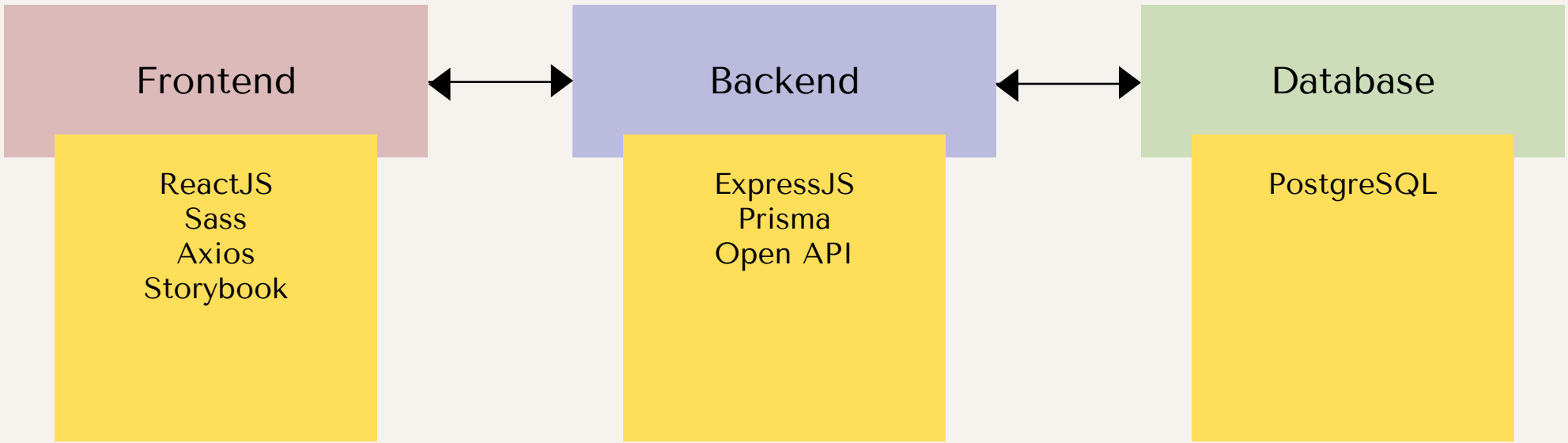
arrays	Fixed or dynamic sized lists
structs	User defined type
mapping	Hashmap with bytes, strings or enum as keys to any value
uint	Positive number. No Decimal
address	Unique identifier for accounts and contracts
enum	User defined type

DATA LOCATION

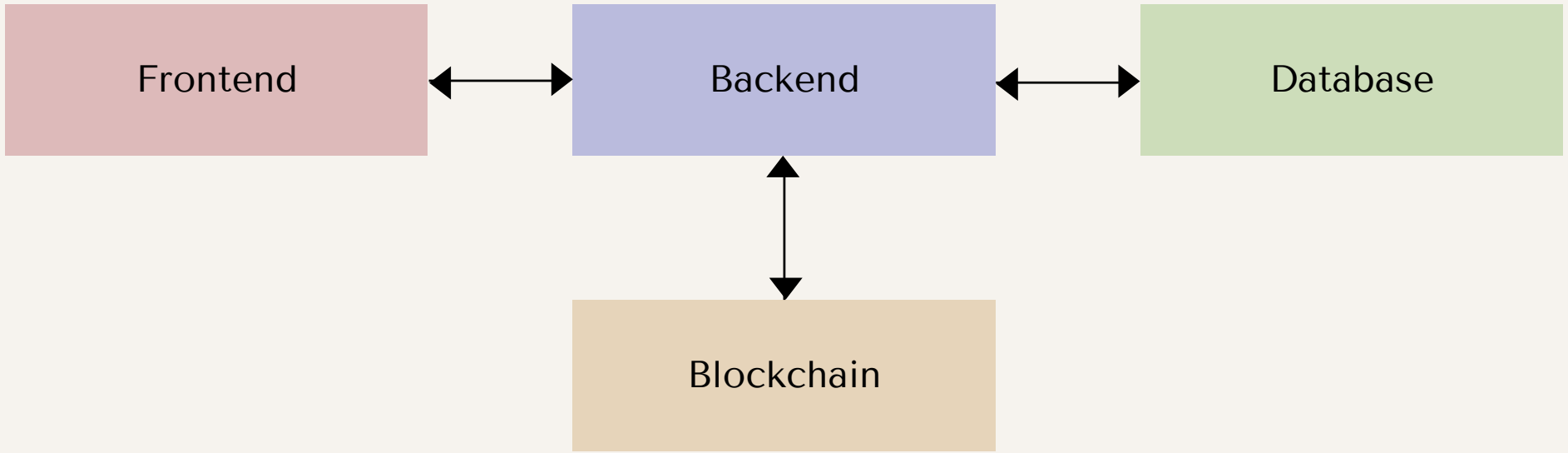
Memory	Temporary store, lifetime is limited to a function call
Storage	Global data store
Calldata	Same as Memory but can only be used in function parameter declaration

FULL STACK

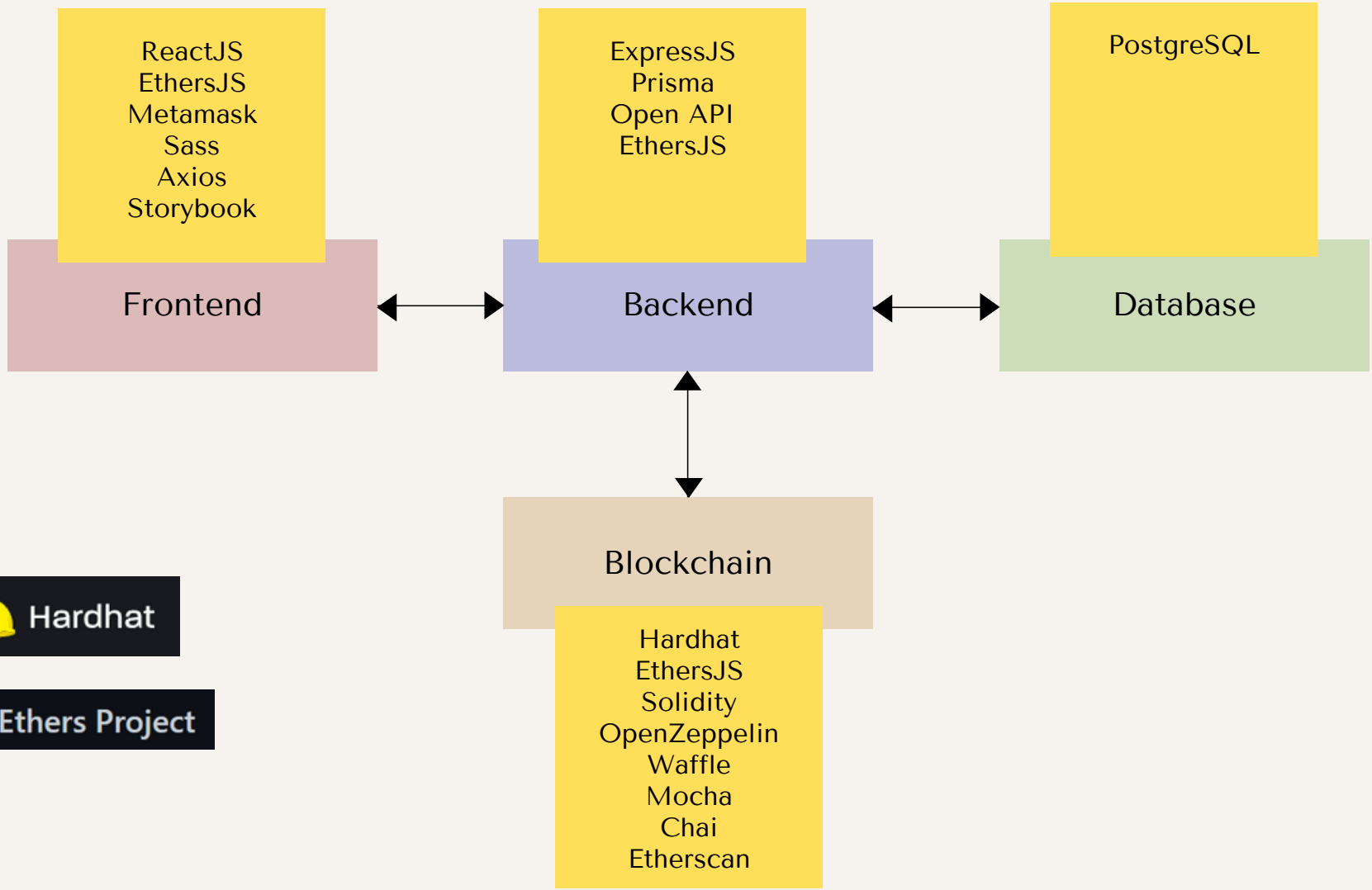
FULL STACK EXAMPLE



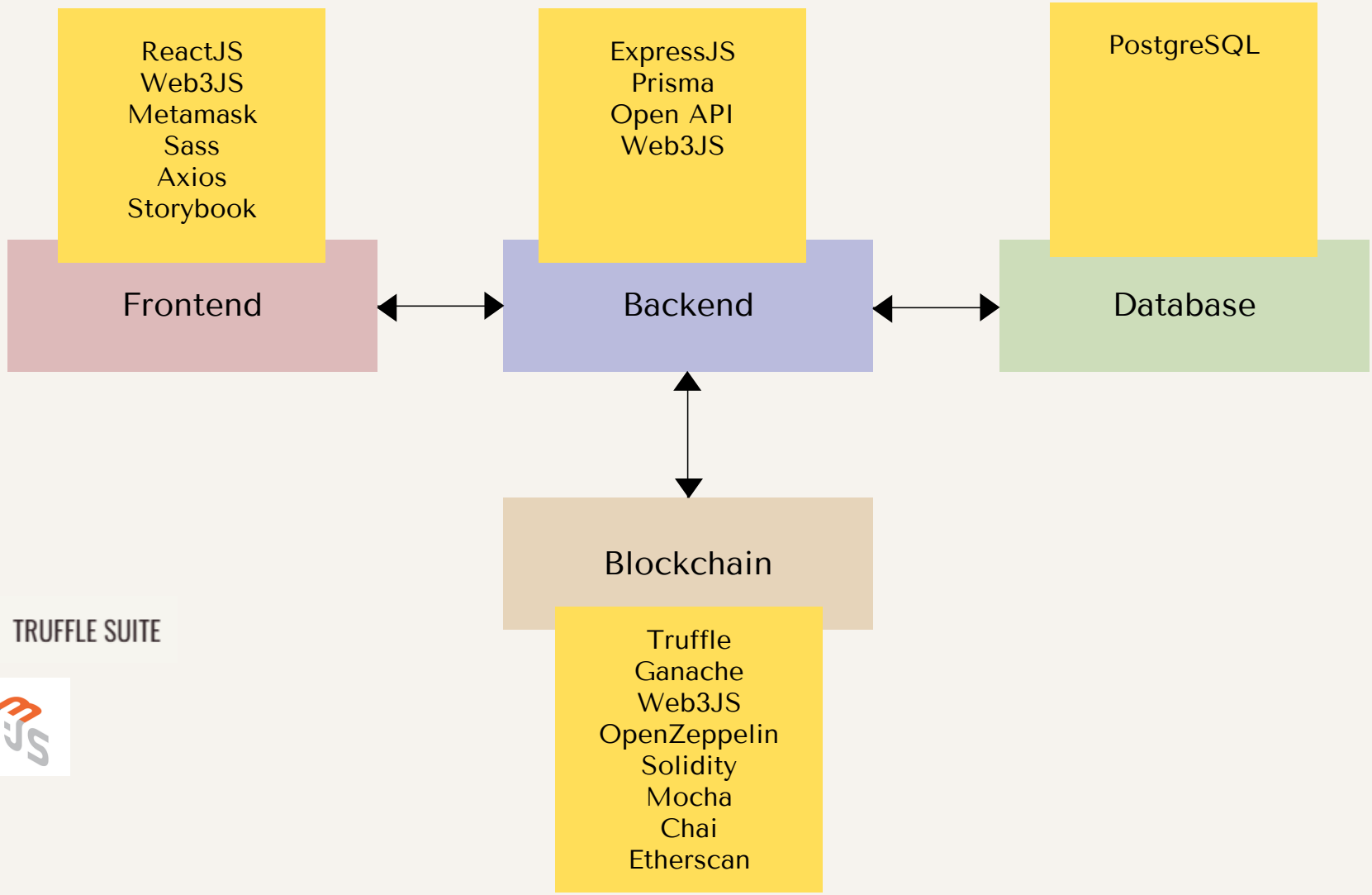
BLOCKCHAIN FULL STACK



BLOCKCHAIN FULL STACK ETHERSJS EXAMPLE



BLOCKCHAIN FULL STACK WEB3JS EXAMPLE



BLOCKCHAIN INTERACTION TECH COMPARISON

FEATURES	ETHERSJS	WEB3JS
Community		●
Code readability	●	
Package Size	●	
Well tested	●	
Tutorial Materials		●

ADVANCED TOPICS

IPFS

Interplanetary File System is a form of Decentralized File Sharing.

Web2 uses location based addressing.
Example: https://images.com/dog.png
Example: 142.127.240.100/dog.png

If the hosting server is down, users cannot retrieve their files.

IPFS

Web3 IPFS uses content based addressing.

Example:
ipfs://Qmf3xGUcdwzynagoTjZkKdWpxuo5kRVBdv38rdH9VfQ47j?filename=dog.png

Example:
https://ipfs.io/ipfs/Qmf3xGUcdwzynagoTjZkKdWpxuo5kRVBdv38rdH9VfQ47j?filename=dog.png

Qmf3xGUcdwzynagoTjZkKdWpxuo5kRVBdv38rdH9VfQ47j is the content id (CID), derived from the hash of the file data.

IPFS

How does it work?

IPFS data is organized IPFS Objects.

Each Object contains data up to 256kb and links to other IPFS Objects.

Data larger than 256kb can be split up into several objects, with each object linking to each other.

IPFS

Advantages:

1. Data Availability
2. Efficient storage (no duplicates)
3. Speed of download

Pinning Services:

1. Pinata
2. Filecoin

IPFS

Challenge: Privacy
Solution: Encrypted Content Hash

Challenge: Immutable data
Solution: Directed Acyclic Graph

Challenge: Malicious or inaccurate data served
Solution: Verification by rehashing content

Challenge: Slow download speed due to long distance
Solution: Serve files P2P with closest node

IPFS USE CASE: NFT

ERC721 live coding on Remix

ENS

Web2 uses DNS (Domain Named Service).
Purpose is to change machine address to human readable address.
Example: www.dogs.com -> 192.256.220.91

Web3 uses ENS (Ethereum Named Service).
Purpose is to change machine address to human readable address.
Example: david.eth -> 0x9e9809988185b0ab70a992f0aaf9e057806c0f92
Example: dogs.eth -> ipfs://QmccqhJg5wm5kNjAP4k4HrYxoqaXUGNuotDUqfvYBx8jrR/qr#enter%252520text%252520here

ENS

How does it work?

There are 2 categories of smart contracts that makes up ENS. The Registry and the Resolver.

Registry: stores the owner and the resolver contract address. Also registers subdomains. Example: corgi.dog.eth

Resolver: stores the actual address of the .eth name

Lookup users will interact with the Registry and then the Resolver to get the actual address.

ENS

Additional top level domains:
.crypto
.xyz
.club

Proof of ownership represented as an NFT.

ENS NFTs are rented and need to pay a yearly fee in ETH.

ENS is decentralized and open source.

ORACLES

Trusted Data injection into the blockchain.

Blockchain is a state machine, and it has no way of getting data off chain on its own.

Three types of oracles:

1. Hardware Oracle
2. Software Oracle
3. Human Oracle

ORACLES

Oracles can be decentralized too.

Chainlink is the largest decentralized oracle service.

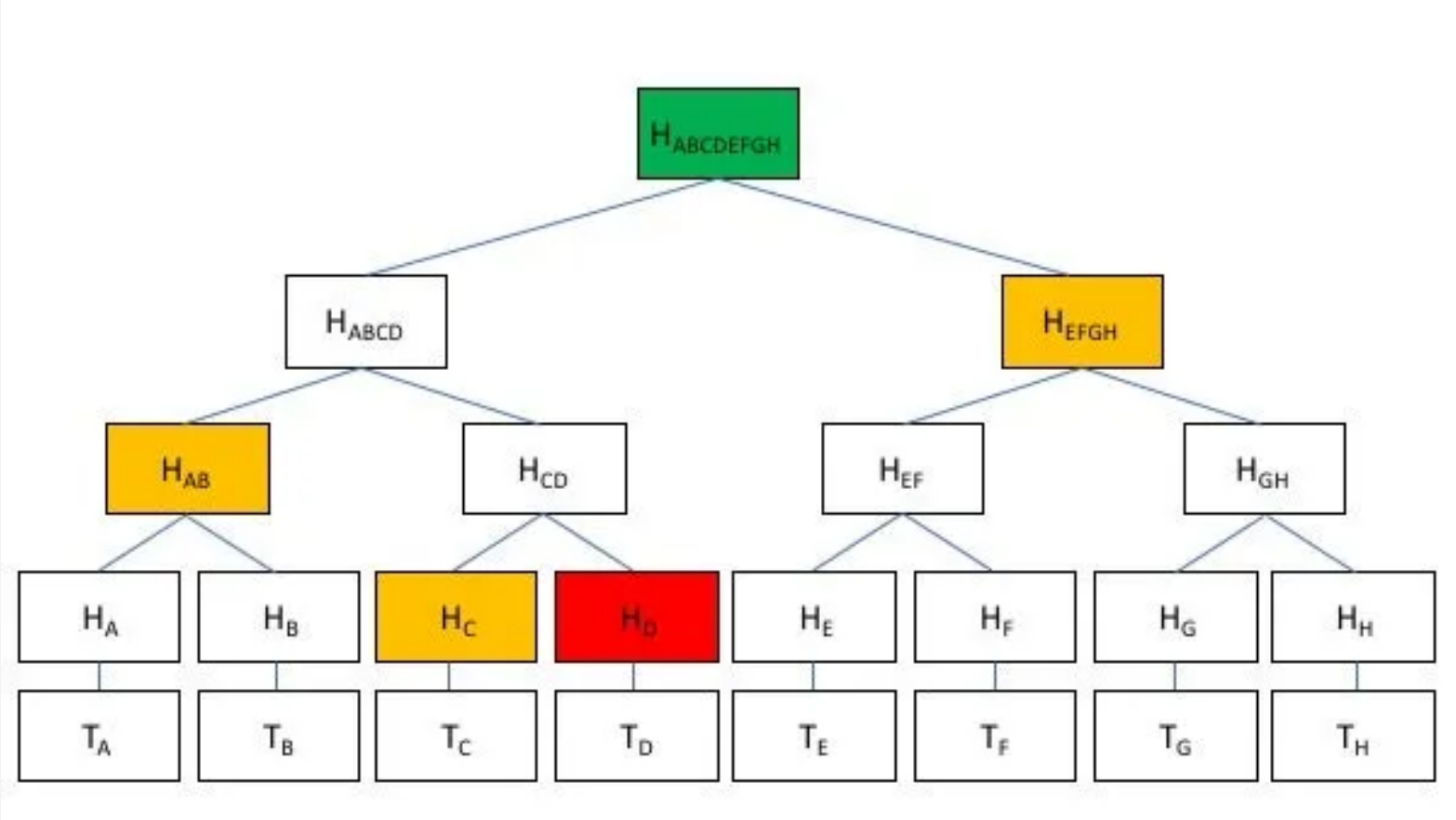
Chainlink is an EVM blockchain that uses POS.

Chainlink operates with 3 categories of smart contracts:

1. Reputation Contract: Payment to add good nodes to the network
2. Order-Matching Contract: Request to fetch some data
3. Aggregating Contract: Answering nodes come to consensus on whose data are to be accepted.

MERKLE TREE

A quick way to validate data.



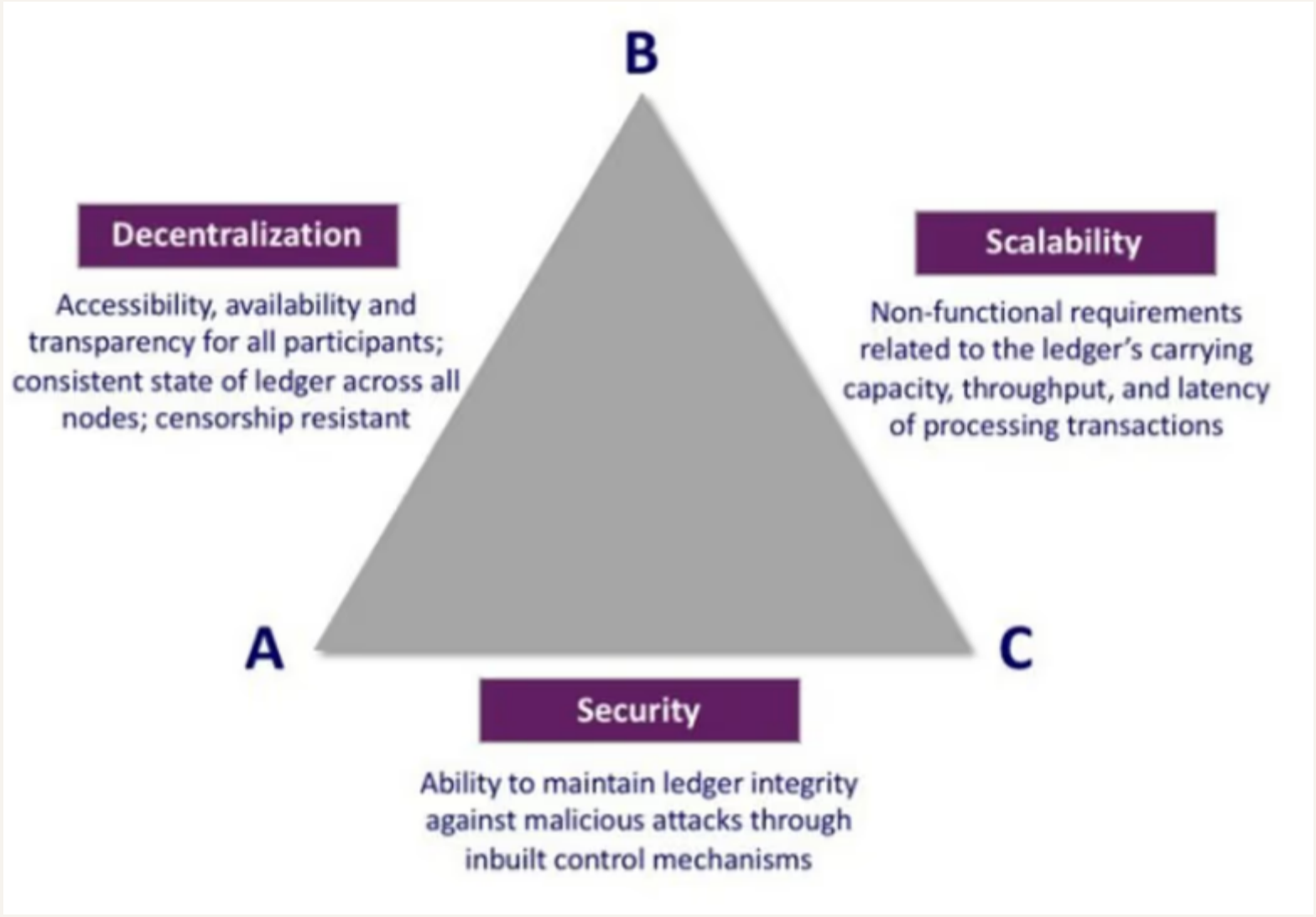
ZK PROOF

A mathematical way to prove data requested, without revealing the data itself.

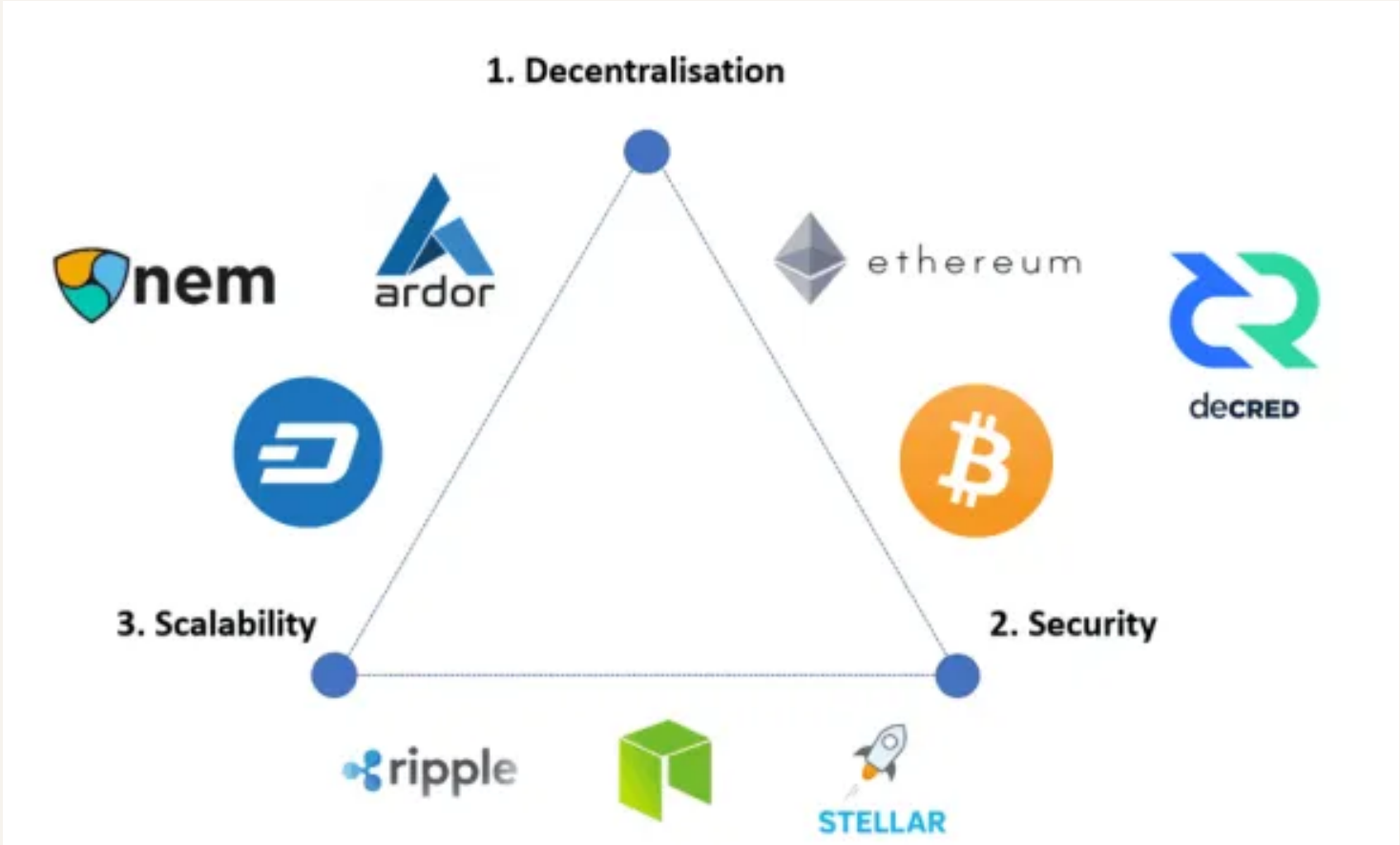
Two types:
Interactive
Non Interactive (ZK-SNARK)

Disadvantage:
Requires much more computational power

BLOCKCHAIN TRILEMA



BLOCKCHAIN SCALING TRILEMA



Resources Used:

https://coinsbench.com/about-evm-opcode-gas-ethereum-accounts-9f0896f09d04

https://ethereum.org/

https://hardhat.org/

https://docs.ethers.io/v5/

https://www.openzeppelin.com/

https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf

https://www.skillsoft.com/