Page Date

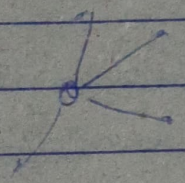
Blockchain

Unit 1

Networks

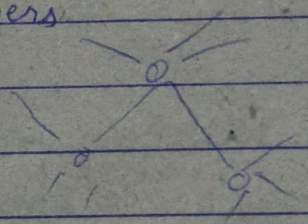
① Centralised → All systems are connected to central computer

- Single owner
- Full control
- More prone to attack
- Third party involvment is present
- Easy to implement
- Vertical scaling
- Client server architecture



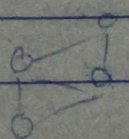
② Decentralized → There are multiple central servers

- No single owner
- No single point of faliure
- Data is replicated on different central servers
- User can access data from other server if one fails



③ Distributed → All nodes are masters

- All have equal rights
- Compution is distributed across components
- Each node interact by message passing
- P2P network



Page Date

Distributed Ledger → It is like a register whose copy is with everyone and it is being updated at everyone's end.

* Block chain is like data structure which has block of data connected together.
* Blockchain is append only database (permissioned)

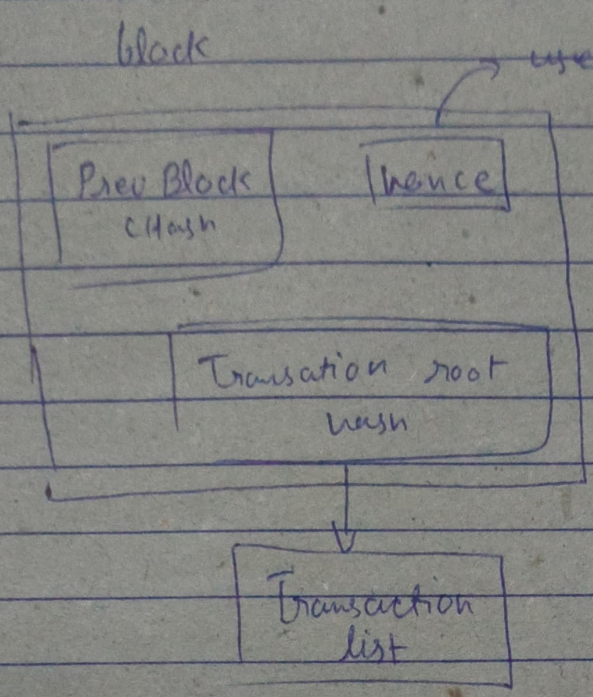
cryptocurrency :- digital currency

A blockchain is like a database, where information is stored in a continuously growing chain of data blocks, implemented in a decentralized network in a ~~way~~ way that create data integrity, trust and security for the nodes, without the need of central authority.

or

Blockchain is cryptographically secured record of tranctions stored on decentralized network. where each block contain various transactions which are approved after complex consensus algorithm.

Transaction :- Any ~~to~~ operation that is to be recorded is transaction



Consensus mechanism ⇒ It is a complex algorithm which when solved then only block is added.

Smart contract ⇒ It is sely executable code that is executed on every transaction.
⇒ It checks address, signature and account of senders and receiver.

Blockchain platforms

① Public	② Private	③ Consortium
→ permissionless	→ Permissioned	→ partially decentralized

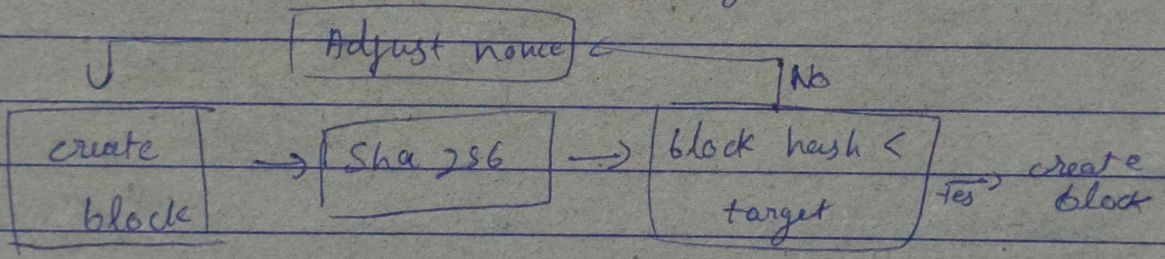
Page Date

Unit 2

Consensus → mutual ~~agg~~ agreement

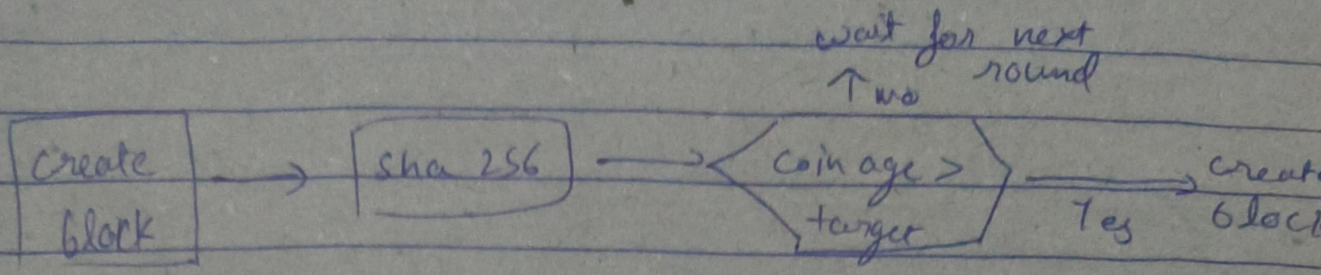
Proof of work

- Hash value depend on nonce
- Miners compute the nonce
- They generate hash which is less than target block hash
- They compute the hash and transmit the message, which are checked and validated by other miners



Proof of stake

- Miners have to put their coin age at stake
- If they do ~~fee~~ wrong calculation their coin age became 0
- If coinage of miner is greater than the computed hash than only he becomes the miner



Proof of Activity

Combination of POW + POS

To create block miner use POW, he has to compute the nonce and hash.

After creation of block, validators are choosen to validate block

Proof of Burn time

- Miners have to burn their coin, they send their coin to variable unspendable address (Eater address), by burning their coin they get virtual mining rig.
- coin regularly decay
- Burned coin = amount of power miner has

Proof of capacity

It is storage oriented (largest capacity - miner), It stores the posibility result of hash value.

plotting : Store nonce and hash in hard disk

mining : use those base value to mine

Page Date

Concensus algorithm for permissioned blockchain

① Proof of Elapsed time

- developed by intel and it works on lottery system.
- An algorithm is used to generate random wait time for each node. Node sleep during wait time.
- Node with shortest wait time wins the race.

② Proof of authority

- Has high fault tolerance
- certain prevalidated ~~blocks~~ nodes can create blocks
- Blockchain networ based on nodes reputation choose multiple validity node.
- Out of multiple validity node one node is selected to create block

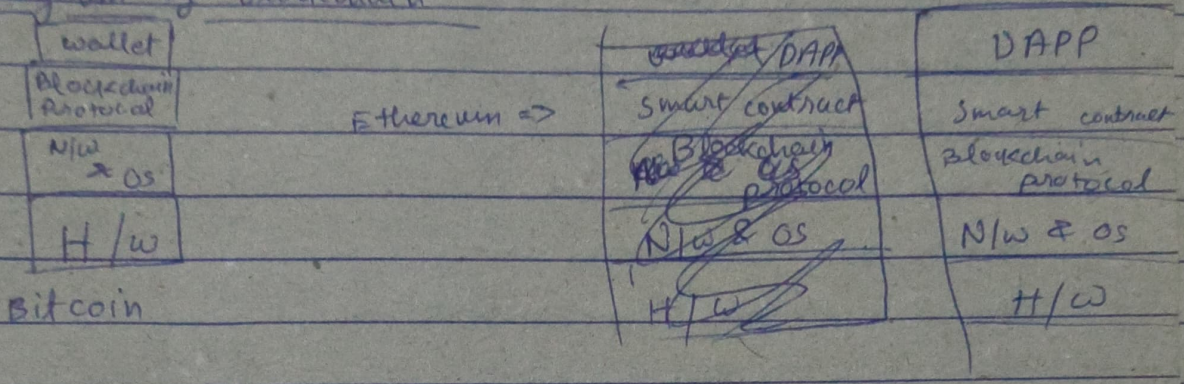
③ Proof of importance

- It is modification of POS
- It adds more variable to POS like size of transaction, no. of transaction, etc.

Page Date

Unit 3

Basic diagram of blockchain



Wallet ⇒ It is same as bank account, it keeps the record of amount of cryptocurrency you own.

Type:
- ↳ Desktop
- ↳ Mobile app
- ↳ Browser

Ethereum has 2 account ⇒ Externally owned account
Smart contract account

EOA control smart contract
EOA is controlled by private key

Testing network

- ↳ Ropsten
- ↳ Kovan
- ↳ Rinkeby
- ↳ Goerli

Unit 4

Remix ⇒. It is a ide that allows us to write, compile and deploy smart contract.

- Smart contract are written in solidity (.sol)
- On compiling contract address and abi is generated which is used to connect smart contract to app.

Get age and name

```
pragma solidity ^0.4.0;
contract person{
    string private name;
    uint private age;

    function setName (string newName) public {
        name = newName;}
    function getName () public constant returns (string) {
        return name }
    function setAge (uint newAge) public {
        age = newAge; }
    function getAge () public constant returns (uint) {
        return age; }
}
```

Page Date

Counter

```
pragma solidity ^0.6.0;

contract counter {
    uint value;

function initialize(uint x) public {
        value = x;
    }
function get() view public return(uint) {
            return value; }
function increment() public {
            value = value + 1; }
function decrement() public {
            value = value - 1; }
}
```

Page Date

Unit 5

Hyperledger is a collaborative effort to advance blockchain technology for business. It is trademark of linix foundation. One of the mayor contributer in hyperledger fabric is IBM

It is :-

↳ Open source ⇒ In open source the source code of software can be accessed by anyone.

↳ Open standard ⇒ openly accessable by anyone

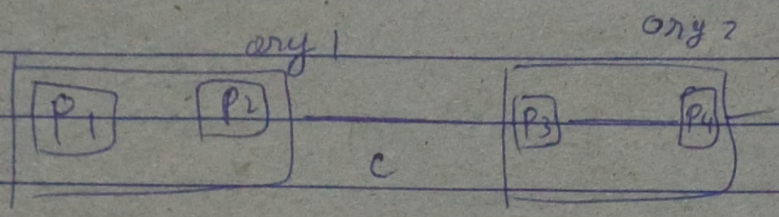
↳ Open governance ⇒ No single organization is controling it

- It create enturprise grade distributed ledger and a code base upon which user can build industry software.
- It is designed for industry
- It is permissioned and non anonymous

Page Date

- Hyperledger is built on idea of selective transaction
  - ↳ Multi transaction (both party sign document)

How fabric work



The network is governed using policy

↳ policy is agreed upon
↳ encode each organization rights
↳ All organization / majority of organization must aggree to the change

↳ It comprises → Blockchain, world state
↳ hold current state of asset

↳ Endorsing node create Rw set and sign → Sdk → ordering service
↳ Block created by Rw set

Page Date

Use case

Blockchain application

- Automative
- Banking & finance
- Government
- Healthcare
- Insurance
- Media and entertainment
- Retail / consumer good
- Travel and transport

When blockchain

- Multiple company
- certain level of trust

Blockchain

Centralized banks	Blockchain
↳ Centralized power	↳ Decentralized powers
↳ Private ledgers	↳ Public ledger
↳ Prone to hack	↳ Immutable to hacks
↳ Double spending	↳ No double spending

Blockchain → public distributed database that holds the public ledger.
↳ Block → collection of all the recent transaction

Bitcoin → first decentralized digital currency. (only 21 million)
↳ use cryptography to control its creation and management.
↳ created and held electronically in a peer to peer open ledger called the blockchain.

Block →

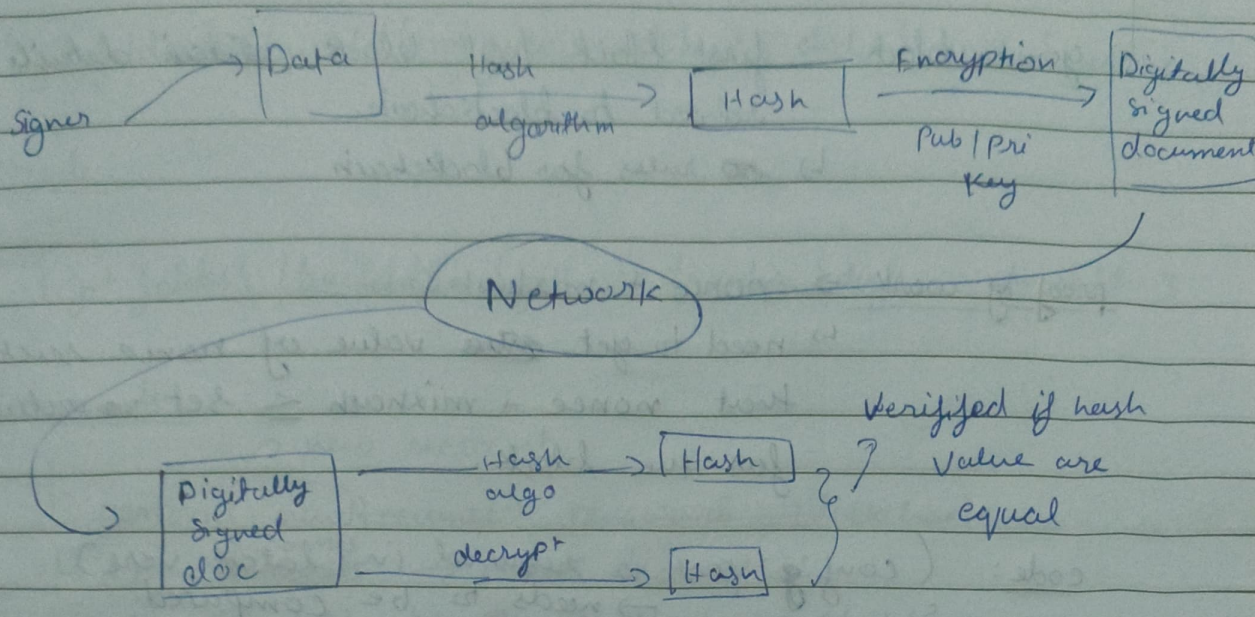
Previous hash code
transaction
Proof of work

↳ ledger is produced by people using software that solves mathematical problems.

Private key Cryptography

↳ involves two different keys private and public
↳ one key is kept private and other is provided to third party.
↳ if private key is used to encrypt then public key can decrypt and vice versa.
↳ asymmetric encryption.

P2P network - Digital Signature



Blockchain transactions

- All transactions are logged including information on the time, date, participants and amount.
- Each node in network holds a full copy of the blockchain.
- Transactions are verified by the Miners after solving complex maths puzzles.
- The mathematical principle ensures that the nodes automatically and continuously agree to the current state of the ledger and every transaction in it.
- if anyone attempts to corrupt a transaction the nodes will not arrive at a consensus and hence will refuse to incorporate the transaction in the blockchain.

Installing blockchain

- clone go ethereum
- make a branch (git checkout tags/(ver) -b name

↳ create a genesis block

↳ genesis block → first block / 0th block, details related to blockchain.
↳ rules for blockchain

proof of work → nonce + mixhash
↳ need to get value of nonce such that nonce + mixhash ≤ set value for the block.

code: (config part is required in latest vers)

```
{
    "nonce": "0x3",          ← needs to be computed
    "timestamp": "0x0",
    "parentHash": "hash",    ← hash of parent
    "extraData": "0x0",      ← extra 32 bit memory to pass message
    "gasLimit": "0x4c4b40",  ← limit of resource you can spent
    "difficulty": "0x400",   ← fixed
    "mixhash": "hash",       ← how difficult it is to be to mine a block
    "coinbase": "0x coinbase",   ← coins has to be sent to
    "alloc": {
    }                        ↳ list of prefilled wallet
}
```

↳ ~~initializing the blockchain~~

↳ ~~geth --datadir genesis.json~~
~~init genesis.json~~

↳ ~~geth console~~

↳ ~~geth --datadir --networkid 33 console~~

↳ initializing the blockchain

{ git folder } /build/bin/geth --datadir {dir} init {genesis.json}

↳ console

{ git folder } /build/bin/geth --datadir {dir} --networkid }

↳ commands

↳ personal.newAccount(), password : blockchain
↳ eth.accounts → check accounts
↳ eth.blockNumber → total no. of block.
↳ miner.start() & miner.stop / start
↳ miner.stop()
↳ eth.getbalance (Hash) → check balance

↳ Decentralized Application

↳ user invokes a smart contract
↳ smart contract contain all the rules abiding for the service being provided and it contain state information which contain data for the smart contract.

↳ resource are paid by you.

↳ Technology stack

Frontend		Backend
HTML		Smart contracts
CSS	← web3 →	Solidity
Javascript		

↳ Advantages of DAPPS

↳ Autonomy :- No need of third party

↳ Trust :- encrypted on shared ledger

↳ Backup :- documents are ~~dddd~~ ~~dddd~~ duplicated many times

↳ Accuracy

Some applications ⇒ blockverify, ripple, storj, augur, ~~ethlance~~ ethlance

↳ Etherium

↳ Etherium is an open-source, publicly distributed platform featuring smart contract functionality to build decentralized apps on top of this platform

Bitcoin	Ethereum
→ Digital Money	World Computer
→ BTC	Ether
→ Turing incomplete Clanguage	Turing complete
→ SHA 256	~~Eth~~ Ethash
→ Early mining	Through ICO
→ ~10 min	~12-15 sec

Etherium Account

↓ Externally owned account

→ Owned by people or organization

→ controlled by private keys

↓ Contract (smart contra Account

→ Autonomous account

→ controlled by code

→ triggered by ~~transa~~ transaction.

↳ smart contract → immutable piece of code that perform certain task
→ they automatically execute when certain condition are met.

↳ Solidity → it is a contract oriented, high level language for implementing smart contracts on Ethereum Virtual Machine (EVM).

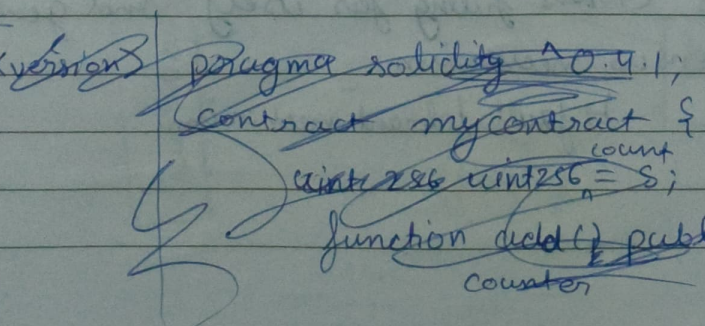
↳ Ethereum Virtual Machine - Contracts written in a smart contract specific programming language are compiled into 'bytecode' which an EVM can read and execute

↳ Ethereum gas → execution fee payed by senders for a transaction
↳ it is the unit for the amount of computational work done by the computer for one cycle of the contract.
↳ gas limit is the maximum amount of gas the contract can use for it's computation.

↳ Deploying Smart Contracts

↳ ganache
↳ npm install -g ganache-cli and ethereumjs-testrpc and truffle

↳ Solidity



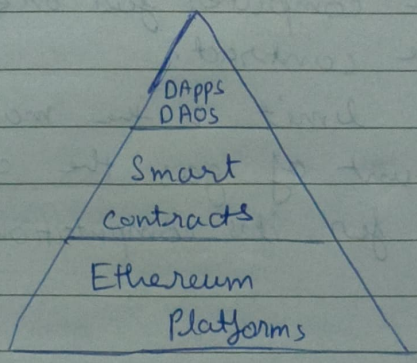
↳ solidity (example)

```
pragma solidity ^0.4.1; //version

contract my_contract {
    uint256 counter = 5;
    function add() public {
        counter++;
    }
    function sub() public {
        counter--;
    }
    function gc() public constant returns (uint256) {
        return counter;
    }
}
// no gas is required for returning a constant function.
```

View

View



↳ DAPP → set of smart contracts that operates on the data stored in those contracts

- ↳ open source
- ↳ decentralized
- ↳ tokens (tokens ruling for itself and generates tokens
- ↳ consensus mechanism
  - ↳ fault tolerant mechanism

consensus mechanisms

↳ it ensures records are true and honest.

↳ **proof of work** : ↳ transaction data is stored in block
↳ validated by solving a complicated math problem attached to it.
↳ done by powerful computer and is known as mining.

eg : bitcoin, ethereum

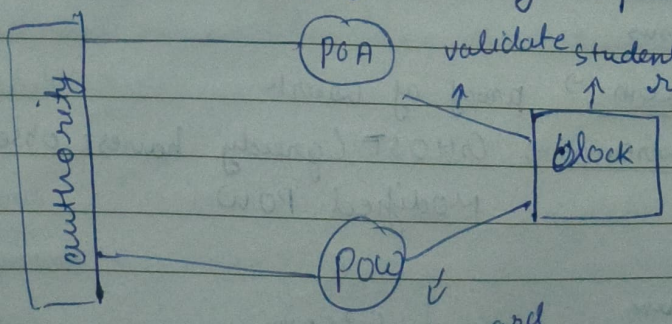
↳ **proof of stake** : creator of new block also known as validator is randomly choosen based on how much stake they commit to network.
↳ the higher the stake placed, higher the chance to be selected.

eg :- EOS, cardano ouraboros

↳ **proof of authority** : modified form of proof of stake, validated party are selected based on their reputation are selected as validator.

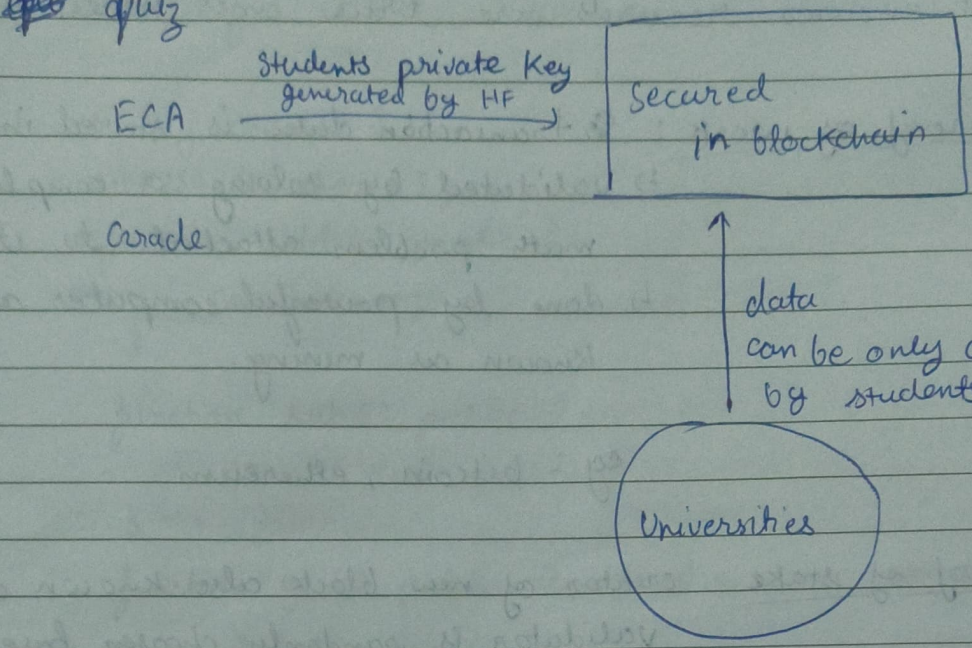
eg :- ethereum kovan testnet, IBM hyperledger

test case (opet foundation) (proof of Authority + proof of work



↳ POA is done by hyperledger fabric

quiz



↳ ethereum vs Bitcoin

crypto currency
- ↳ both uses proof of work
- ↳ bitcoin → cryptocurrency
- ↳ ethereum → to develop decentralized applications

account
- ↳ bitcoin → no distinct accounts.
- ↳ ethereum → proper accounts

Smart contract
- ↳ bitcoin → limited functionality scripting language
- ↳ ethereum → full general purpose language (solidity)

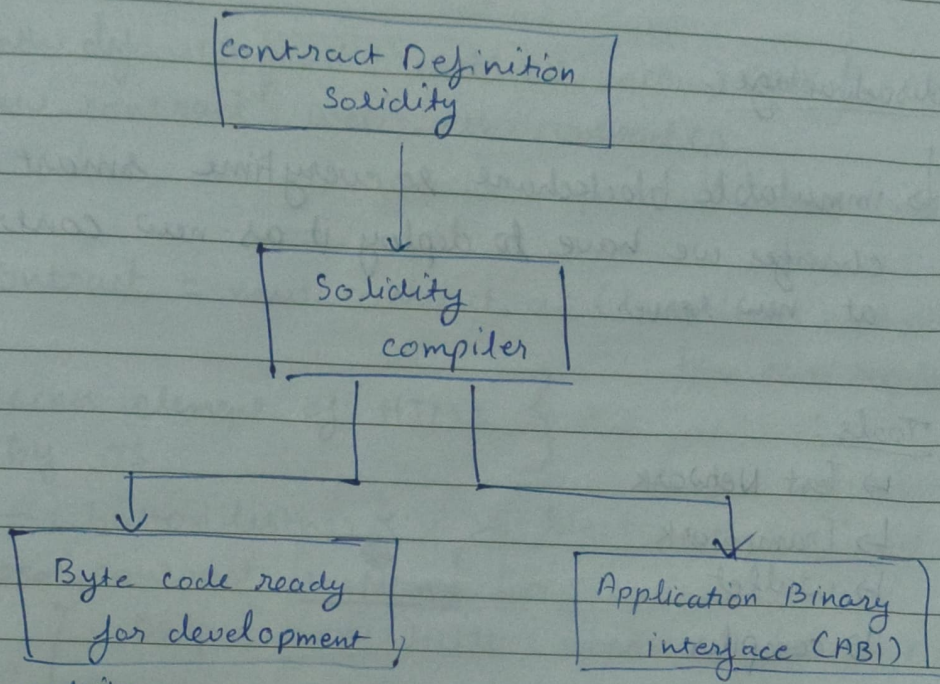
transaction
- ↳ bitcoin → 1747.7 transactions / block or 2.917 transaction / s
- ↳ ethereum → 224.4 transaction / block or 14.96 transaction / s

→ consensus
- ↳ bitcoin → proof of work
- ethereum → GHOST (greedy heaviest observed sub tree)
  Modified POW

↳ ethereum	Bitcoin
POW: ethash	POW: SHA256
ASIC - resistance	ASIC - receptive
5 ETH / block	12.5 BTC / BLOCK

Ethereum Development Tools

↳ web3 → ethereum provider
↳ smart contract → remix, pragma
↳ solidity (6 language use to create smart contract)
    ↳ Solidity compiler : SOLC



↳ 'Solium' fixes issue in solidity code
↳ to interact with ethereum blockchain we use ethereum client : Kethclient, parity, metamask
↳ wallet : my ether wallet
↳ cli tools : truffle, embark, dapple
↳ scalable : infura
↳ Analytics : DAppBoard

Smart Contracts

↳ smart contract are executable logic that runs on a blockchain network.
↳ features:
    ↳ autonomous
    ↳ Auto-sufficient
    ↳ Decentralized

↳ language: Viper, solidity, Vy-Lang, Rust

↳ advantages:

- ↳ secure
- ↳ Netwrality
- ↳ precision
- ↳ saves time

↳ disadvantage:

↳ immutable blockchane so everytime smart contract changes we have to deploy it as new contract at new server.

↳ Tools:

- ↳ Test Network
- ↳ Framework
- ↳ wallet
- ↳ compiler

Example:

↳ npm init // initialize npm

↳ index.html

```
:
<script type="..." src="..web3.js"></script>
:
:
<body>
:
<declare layout>
<script>
var web3;
if (typeof web !== "undefined"){
    web3 = new web3(web3.currentProvider);
}
```

else {

web3 = new Web3(web3.providers.HttpProvider(
"http://localhost:8545"));
↳ wallet

}

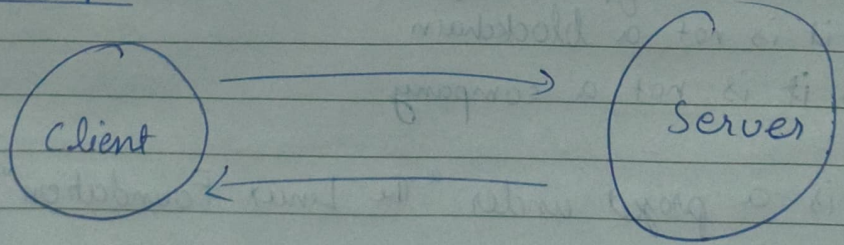
web3.eth.defaultAccount = web3.eth.account[0];
var new_contract = web3.eth.contract( )
↳ Smart contract ABI

var contract = new_contract.at("address contract has been deployed")

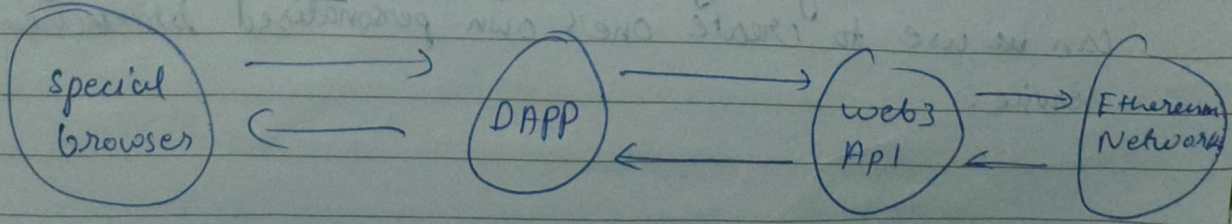
{ access element of HTML by JS
{ button listner } → function in contract
contract.setName(input.value);
output.innerHTML = contract.getName( );
}
function →

Ethereum Dapp

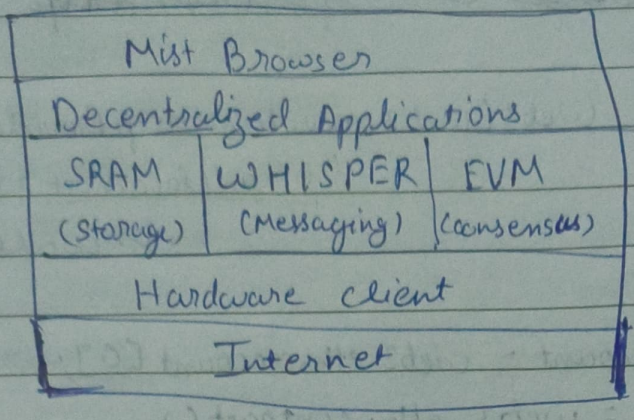
↳ Traditional App



↳ Decentralized App



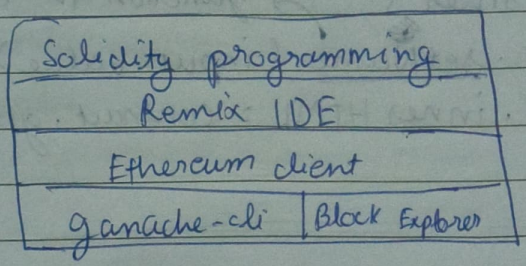
↳ Web3.0 Tech Stack



↳ smart contract language : solidity, serpent, LLL
↳ preffered

↳ metamask (↗ addon), mist (↳ website)

↳ web3.js



↳ Hyperledger

↳ it is not cryptocurrency
↳ it is not a blockchain
↳ it is not a company

↳ it is a project under "The Linux Foundation"

↳ it can be thought of as a software which everyone can use to create one's own personalised blockchain service.

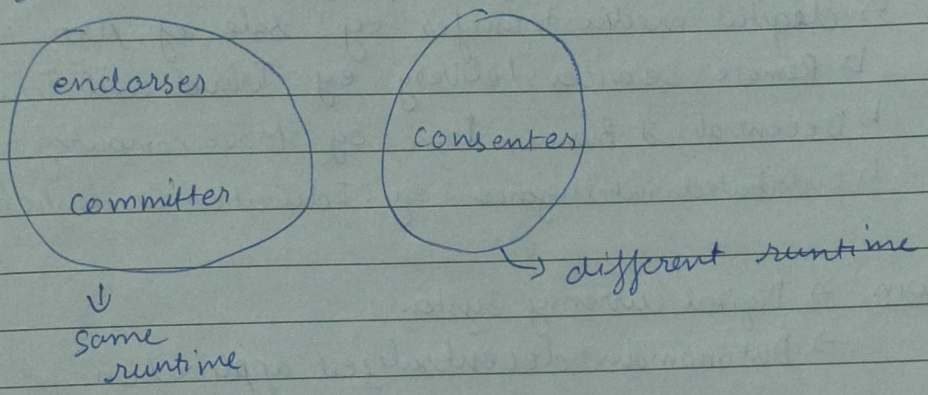
→ Public blockchain:

↳ are not scalable

↳ do not support private and confidential transactions

→ on hyperledger network, only parties directly affiliated with the deal are updated on the leadger and notified. Thus maintaining privacy and confidientially.

↳ changes



↳ committes → Append validated transactions to their specific ledger

↳ endorses: → simulating transactions
→ prevent unstable and non deterministic transactions

↳ consentes: → Networks Consensus Service
→ it will order transactions into blocks according to the network's choosen ordering implementation.

↳ hyperledger
→ no coin
→ permissioned network
→ smart contract: chaincode
→ consensus: PBFT
↳ practical byzantine fault tolerance
→ language: golang, Java

↳ Projects
↳ Sawtooth
↳ Fabric
↳ Indy
↳ Iroha