Learn Ethereum concepts

Solidity *Powered by Zastrin*	One of the popular object oriented programming languages used to build smart contracts on the Ethereum blockchain platform. It is statically typed, supports inheritance and user defined types. Solc is the compiler used to compile Solidity code.

EVM *Powered by Zastrin*	Ethereum Virtual Machine (EVM) is a runtime environment with in which smart contracts are executed. There is a clearly defined spec for what constitutes the EVM. There are a number of clients like Geth and Parity that implement the EVM per the spec.

Vyper *Powered by Zastrin*	Vyper is a programming language to write smarts contracts for the Ethereum platform. It is heavily inspired by the Python programming language.

Gas *Powered by Zastrin*	Gas is the unit of work in Ethereum. Any computation that needs to be executed in the EVM consumes the resources on the host computer. The measure of this resource is gas. Ex: To add 2 integers, it takes 3 gas. To multiply, it takes 5 gas.

Gas Price *Powered by Zastrin*	Gas price is the cost of one unit of gas. The price is paid in Wei, the smallest denomination of the Ethereum's native currency Ether. User's set the gas price to whatever they like. The higher the gas price, faster their transaction is mined by the miners.

Gas Limit *Powered by Zastrin*	Every transaction in Ethereum takes a certain amount of gas. As a user, you can set the maximum amount of gas you are willing to pay the miners to execute your transaction and include it in the block.

Account Nonce *Powered by Zastrin*	Account nonce is a counter associated with each Ethereum account. Every time an account initiates a transaction, the nonce is incremented. The miners make sure to execute transactions in the order they were generated by the account. This helps prevent double spending.

Ommer/Uncle Block *Powered by Zastrin*	An Ommer (also called Uncle) block is a stale block that was correctly mined by a miner but did not end up being part of the longest chain. They still get paid a small amount for their work and the block header is included 2 blocks later. This occurs because of the relatively shorter block times in Ethereum (15 seconds).

ABI *Powered by Zastrin*	ABI stands for Application Binary Interface. It's is a data encoding scheme used in Ethereum for working with smart contracts. It is generated when contract is compiled and is in JSON format. It is used by client code to see contract details and interact with the contract.

Ether *Powered by Zastrin*	Ether is the native currency of the Ethereum blockchain. It is created as a reward to the miner who successfully mines a block and includes it in the chain. Ether is used by network users to transact and also pay for their transactions.

Wei *Powered by Zastrin*	Wei is the smallest denomination of Ether. 1 Ether is equal to 10 ^18. EVM only understands Wei. So, any monetary value in the smart contracts are always represented in terms of Wei.

Full Node *Powered by Zastrin*	A full node is an Ethereum node/server that connects to the network, receives and validates all the transactions and blocks. It stores all the blockchain data locally and can be queried. It does not contain state information for old blocks but only for blocks synced going forward. However, any state can be derived from the data on disk.

Light Node *Powered by Zastrin*	As the name indicates, light node is an Ethereum node/server that just downloads the headers of the blocks. To obtain any other information, it has to connect to a full node and query. This node is ideal to run on devices that do not have lot of disk/processing power such as IoT devices or phones.

Archive Node *Powered by Zastrin*	An archive node is an Ethereum node/server that connects to the network, receives and validates all the transactions and blocks. It stores all the data including every historical state information of the blockchain. Archive nodes consume lot of disk space and is not required for majority of the use cases.

codeHash *Powered by Zastrin*	An Ethereum account is composed of multiple fields. codeHash is one such field which is the hash of the EVM code of this account. If this address receives a message call, this is the code that gets executed. It is immutable and can not be changed after construction.

Externally Owned Accounts *Powered by Zastrin*	Externally Owned Account (EOA) is one of 2 different types of accounts in Ethereum. EOA always has a corresponding private key associated with it. This account can be used to send/receive Ether and also execute transactions on a smart contract.

Contract Accounts *Powered by Zastrin*	Contract Account is one of 2 different types of accounts in Ethereum. They are created when a contract is deployed to the network. That means, they have code associated with them but they do not have a corresponding private key. This account can be used to send/receive Ether and also call other smart contracts.

EIP *Powered by Zastrin*	EIP stands for Ethereum Improvement Proposal. EIPs are proposals anyone can propose to improve the Ethereum platform. It can be a proposal to make changes to the core protocol, networking, application level standards and so on. EIPs go through a standard process before they are accepted or rejected.

ERC *Powered by Zastrin*	ERC stands for Ethereum Request for Comments. ERC is a type of EIP that is specifically used for application-level standards and conventions. They are usually contract standards such as token standards (ERC20, ERC721), name registries (ERC137), URI schemes (ERC681), and wallet formats (EIP85).

Uncle Rate *Powered by Zastrin*	Uncle rate is the rate at which uncle blocks are created in the network. The more transactions a block contains, the longer it takes for the block to propagate in the network. This is one of the factors affecting the uncle rate.

Ethash *Powered by Zastrin*	Ethash is the algorithm used to perform the Proof of Work in the Ethereum network. It was derived from the Dagger Hashimoto algorithm. However, the two algorithms have diverged significantly now. You can find more details at https://github.com/ethereum/wiki/wiki/Dagger-Hashimoto and https://github.com/ethereum/wiki/wiki/Ethash

Events *Powered by Zastrin*	Events are dispatched signals the smart contracts can fire. Dapp frontend can listen to these events and act accordingly. It should be noted that the event data is not accessible from within contracts but they can be indexed, so that the event history is searchable later. Up to 3 parameters in an event can be indexed.

Storage Trie *Powered by Zastrin*	A storage trie is where all of the contract data lives. Each Ethereum account has its own storage trie. A 256-bit hash of the storage trie's root node is stored as the storageRoot value in the global state trie.

EVM Bytecode *Powered by Zastrin*	Smart contract code is usually written in a high level programming language such as Solidity or Vyper. When this code is compiled, you get a hexadecimal representation of the contract and is called the bytecode. This bytecode is what is deployed to the blockchain and EVM executes this bytecode.

State Trie *Powered by Zastrin*	State Trie is the Trie that holds the global state of Ethereum. There is only one state trie and is constantly updated. The state trie contains a key and value pair for every account which exists on the Ethereum network. The state trie's root node is used as a secure and unique identifier for the state trie,

Transaction Trie *Powered by Zastrin*	Each Ethereum block has its own separate transaction trie. A block contains many transactions. The order of the transactions in a block are decided by the miner who assembles the block. The path to a specific transaction in the transaction trie, is via (the RLP encoding of) the index of where the transaction sits in the block.