Blockchain Bootcamp Day 6(First Part)

* You can completely remove smart contracts due to blocks
* What happens when you deploy a smart contract?
* 1st block is called genesis block(block 0) -> block 1 -> block 2
* Deploy smart contract into blockchain that is compiled into evm byte code.
* Why is it impossible to delete data, all the blocks are connected.
* Block 3 reference block 2, block 2 references block 1
* To delete smart contract, undo transaction and remove the block from blockchain, impossible you would have to link the preceding and seceding block.
* It will go through these blocks, and create blockchain nodes.
* Remove one block in every network, it’s impossible to do this. Once code is deployed, it can’t be changed.
* Data privacy is a issue, can be used for specific specs, setup own corsetum blockchain instead of using mainnet.
* Smart contract lifecycle
* Start -> Compile, send transaction’
* Running -> Interact, Send transactions to it.
* Stop -> selfdestruct(), remove smart contract from state.
* The compile smart contract is in data field., This will indicate if a smart contract has data.
* Selfdestruct(eoa address) will receive funds from destroyed smart contract.
* Smart contracts get compiled and sent to the blockchain as a transaction. Smart contract are transaction based.
* Transactions successfully done, cannot be doned. Atomic operations, if failed it seems like nothing happened.
* Immutable can only change state of block not change data.
* Once a smart contract is destroyed, you can’t interact with it anymore

Introduction to Mappings

* Mappings are defined via mapping keyword, pass lambda with key and value. Can be any type.
* Mapping(integar key => Boolean value)
* Mapping is similar to array or hashmap
* They can be accessed via a key or index.
* Can group multiple variables into a mapping type.
* The index or key of a mapping can be a address.
* If typed correctly only owner of smart contract can edit mapping
* Can be used for whitelisting addresses.
* Can be used for administrator privileges, and stored balance.

Mapping and Structs

* Use a mapping to store addresses
* Can use mapping to store multiple address balances.
* Structs in solidity make you create your own data type.
* Struct can hold each address balance.
* Each address balance would have the number of payments and amount left on account.
* Defining structs outside contract, if you want to detach storage from smart contract to eoa.
* Array in solidity can have fixed or dynamic length
* Structs outside contract definition costs gas
* Array costs gas, mapping are a better alternative. Since it’s a lower computational engine
* Enum are used to create user-defined types in solidity.
* Solidity will assigned unsigned integars
* Mappings are like hash maps
* Structs define their own datatype.
* To prevent re-entrancy, follow the checks and effects pattern.
* Setup variable to the state you would like
* You would access property of a struct using dot notation, while you would access members in a mapping by their key and bracket notation.