Blockchain Bootcamp Day 4

* All variables no matter the type it is defined with default value there is no undefined value.
* The difference between unit256, and uint. Underscore is best preference when dealing with function arguments.
* Integar Wrap Around
* 🡪uint8 – 0 to 255
* When decrementing a integar less than 0 it would wrap around to last value.
* Such as uint8 a = 0, a--; a(uint8) variable now equals 255(the max integar value it can go up to.);
* Address type used to send transactions or find balance.
* Address default value Is 0 20 hex value.
* Ether -> 10 to power of 18 wei.
* A public view that are considered reading function, which would return something
* Address type has some member function such as getting current balance or transferring specific amount of ether to another address.
* Strings are not simple types in solidity they are actually byte arrays.
* Whenever you have a reference type such as string(since it’s a byte array) when passing it as a function argument(or function variable) use the memory keyword so the argument is stored in memory not in storage variable therefore much cheaper.
* Not as easy to work with such as concatenate in other languages.
* Byte array, utf8. Solidity is not built to work with strings.
* Store strings outside of blockchain, or have it in a form of a hash.

Value Types in Solidity

* Simple variable types
* 🡪Boolean, Uint, Int, address, dynamically sized byte arrays(strings)
* All variables are initialized with default value.
* Public variable generate a getter function, it has the same name as variable itself.
* Can’t create any getter function or any function with a same name as variable
* Reference types need to be in memory
* Can negate Booleans
* Boolean or and and. || or &&
* Uint8 is from 0 to 255, 2 \*\* 8
* Int8 from -128 to +127
* Uint256, uint is a alias for uint256
* Automatic wrap around -> uint8 myuint, myUint --=> 256
* If you need a floating point number, assign a decimal point.
* Define amounts in cents similar in smart contracts
* Assign a decimal point in your smart contract such as uint numTokens = 10000, uint decimalPoints = 2, that vies 1000 tokens with divisibility like cents 53,01. 10000 – 5301 => 4699 => 46,99\
* Every interaction in Ethereum blockchain is address based. Each address holds 20 bytes of value.
* Addresses to send funds from one account to another account via a smart contract
* Difference between a payable address and a normal address
* A address always has a balance, wei stored in a ledger of that address.
* String as bytes are arrays.
* Bytes has a index and a fixed length.
* String does not has a index or fixed length
* Avoid use of strings in solidity, extremely expensive to work with in solidity .
* Bytes are used to store raw data such as hashes.
* Strings are used to hold utf8 data.
* You don’t have unlimited space in blockchain. Programming in low resource environment.
* Variable initialization and limited resources
* When using a setting function a transaction is set on the network, while a getter function does not do a transaction just a call. Since the smart contract variable in storage are not being changed.
* **NOTE:** In solidity version 0.8.0 and above, it will check if the a unsigned integar is zero before decrementing it. You need the **unchecked** block to allow it to decrement to max value.
* Address type can be stored in smart contracts and be used to transfer ether from a smart contract to an address stored in variable.
* Smart contract have their own address, or stored another address. A smart contract can transfer funds from it’s address to another address, BUT it cannot transfer funds from another address to it’s address.
* Transferring ether is fundamentally different than transferring erc20 token as you will see later.
* In solidity string comparison or string manipulation is not possible
* There are libraries to work with string, BUT strings are very expensive to work with.(GAS Costs), use **events instead.**