Blockchain Bootcamp Day 5

* Payable keyword indicates a function will receive funds or money.
* This keyword to indicate instance of smart contract.
* Cast this(reference to smart contract instance) to a address type to get balance.
* Msg object can be used to transfer.
* Msg.value is the amount of wei passed to smart contract during the transaction.
* When withdrawing funds or money doesn’t need to be payable.
* You can withdraw funds using msg sender property. When withdrawing funds to your account create a payable address variable. Which will hold the address of address that called smart contract
* Then call transfer function pass the amount in wei to that specific address.
* Then passing a address to get paid, pass a payable address argument. The function DOES NOT have to be payable since the smart contract is not accepting funds.
* So when transferring funds, the smart contract that initiated the transaction pays the gas fees
* Msg is a global existing object that will contain some information about the transaction.
* Msg has two important properties about the transaction, it has the **value** property that contains the amount of wei sent in transaction, and the **sender** property that contains the address that sent the transaction or called the smart contract.
* The function that return information about the smart contract, and doesn’t add to storage or need to be mined so they are virtually free of charge.
* In the JavaScript vm accounts or wallets. The funds from the wallet are sent to the smart contract in the blockchain. That is why a wallet loses money because the amount of ether or wei they send as a value in a form of a msg object during a transaction is removed from that wallet.
* Now if we want to pay another wallet from the smart contract we would use or lose those funds that we received from another wallet or account and send it to that wallet. In this case we would send it to the msg's object sender property. Therefore, we would pay transaction fees (gas fees) since are doing a transaction from that account.
* In cases when you pass a address to send money to, and you are using a different wallet then that address and you sending funds from the smart contract.
* The address receiving the funds if it is not initiating the transaction, IT WILL NOT PAY THE GAS FEES

Accounts – How they work together with smart contracts

* Externally owned account, contract accounts.
* Externally owned account -> private key -> outside of blockchain, used to send funds.
* Contract Account -> Not private key -> Code is managing account -> Is in Ethereum blockchain.
* Use a externally owned account to send funds to smart contract.
* Send funds from eoa(externally owned account), the smart contract HAS to be initiated by a another eoa(externally owned account) It’s a atomic transaction can’t trigger parellal transaction. The origin is always from eoa(externally owned account)
* Blockchain -> ledger shared, transactions are NOT stored on eoa or contract account.
* Address type has a send function that returns a Boolean.
* Exceptions will cascade and will cause the transaction to be cancelled.
* Send can send to another smart contract, if there is a exception, will return false, you or the contract account would have to handle it itself.
* Be aware of possible re-entrancy dangers.
* Following the checks patterns when doing low-level calls
* Can only send 2300 gas using send or transfer function.(Gas stipend)
* Global objects
* Msg.value, msg.sender, now(current timestamp) note: can be altered by miners don’t use it for odd/even numbers.
* Have to mark functions as payable to receive ether, if the function is not marked is payable it will just throw an exception and fail.
* All information is public, they are stored in blockchain. You don’t need permission from permission to send or receive ether.
* Global object tell you what’s inside a transaction.
* The constructor is called during the smart contract deployment, this is where you would set the owner address to the smart contract and this function can never be called again.
* Anyone that deployed the smart contract can withdraw, and anyone can send funds or money to that smart contract.
* When destroying a smart contract use the selfdestruct function, so no one can interact with it in the blockchain.
* When destroyed you can send funds to a destroyed smart contract.
* Be careful with smart contract lifecycle.