Chapter 6: Transactions.

Transactions are signed messages originated by an externally owned account, transmitted by the Ethereum network, and recorded on the Ethereum blockchain.

A transaction contains a nonce, gas price, gas limit, recipient, value, data, vrs.

That is because the EOA's public key can be derived from the v,r,s components of the ECDSA signature.

The nonce is one of the most important and least understood components of a transaction. It is a scalar value equal to the number of transactions sent from this address or, in the case of accounts with associated code, the number of contract-creations made by this account.

Transactions are processed in order of the nonces, i.e a transaction with nonce of 4 will have to wait first until transactions with nonces 1-3 are confirmed.

With nonces available, it removes the risks of enemies repasting the transaction data to drain our ether, because every nonce is unique to every transaction. A new transaction generates a new nonce.

```
> web3.eth.getTransactionCount("address");
>> 5
```

This means that nonces from 0-4 have been processed and the next one will have a nonce of 5. Nonces start from 0.

Gas is the fuel of Ethereum. It pays for computation, memory and storage. The price is measured in wei per gas unit. The higher the price, the faster the transaction is likely to be confirmed. The minimum gas price is 0.

The transaction recipient is specified in the `to` field. It contains a 20-byte Ethereum address which can be a contract address or a wallet address.

Ethereum cannot confirm that an address exists or has a private key. Sending ether to an unknown address renders it U N S P E N D A B L E.

The main payloads of a transaction is in the value and data. A transaction with value only is a payment. A transaction with data only is an invocation. A transaction with value and data is a payment and an invocation. A transaction without both, is a waste of gas, but still possible. Remember the msg.value and msg.data in Solidity, well, here we are.

When you construct an Ethereum transaction that contains a value, it is the equivalent of a payment. Such transactions behave differently depending on whether the destination address is a contract or not.

Contracts are created by sending a transaction to a special destination address called the 0 address. They are solely used for contract creation, and burning ether, because of the reason stated before.

If you want to do an intentional burn, there is a dead address used:
>> 0x000000000000000000000000000000000000dEaD

The payload for contract creation is the compiled bytecode that will create the contract.

The signature proves 3 things:
> Authorization.
> Non-repudiation (Authorizer cannot deny).
> Immutability.

To verify the signature, one must have the signature (r and s), the serialized transaction, and the public key that corresponds to the private key used to create the signature.

The signature verification algorithm takes the message (i.e., a hash of the transaction for our usage), the signer's public key, and the signature (r and s values), and returns true if the signature is valid for this message and public key.

sigVerify(hash, pub_k, signature)