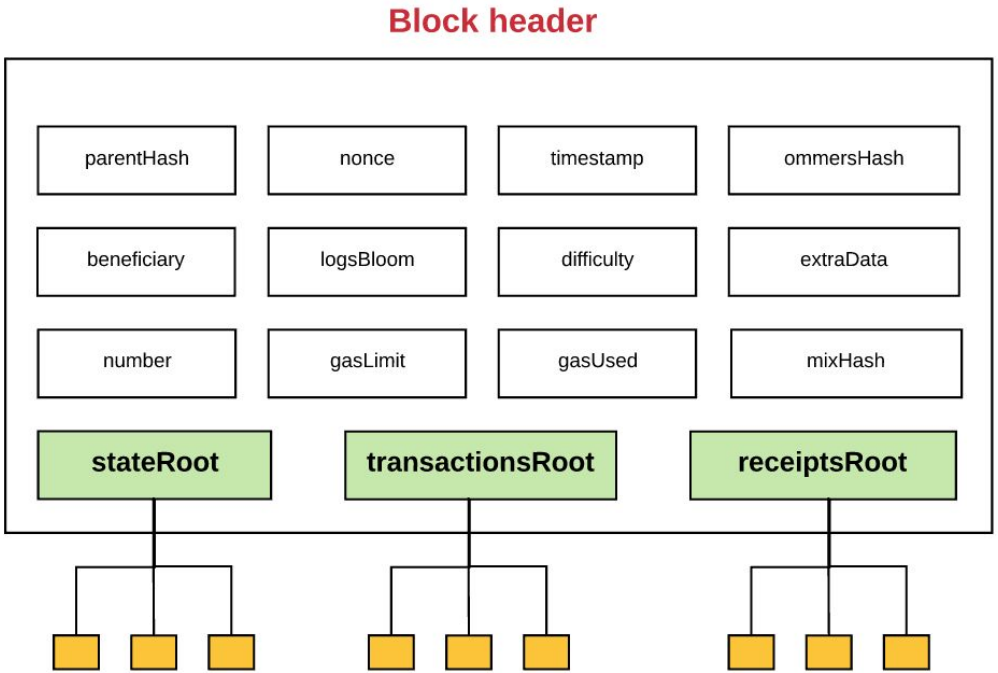
Desarrollo Blockchain Ethereum con Solidity

Módulo 2 - Variables globales y unidades

Block Properties & Transaction Properties

Block Properties & Transaction Properties

Con toda transacción que es enviada a la Ethereum Blockchain, se envía además, información extra adicional que permite realizar ciertos manejos y validaciones sumamente útiles.



Block Properties & Transaction Properties

Block

- block.**blockhash**(uint blockNumber) returns (bytes32): hash del bloque.
- block.**coinbase**(address): dirección del minero que cerró el bloque.
- block.**difficulty**(uint): dificultad del bloque.
- block.**gaslimit**(uint): limite de gas del bloque.
- block.**number**(uint): numero del bloque actual.
- block.**timestamp**(uint): Timestamp del bloque actual.

Message

- msg.**data**(bytes): llamada en bytecode
- msg.**gas**(uint): gas sobrante
- msg.**sender**(address): quien ha hecho esta llamada
- msg.**sig**(bytes4): identificador de la función (son los 4 primeros bytes de msg.data).
- msg.**value**(uint): cantidad en wei enviada
- now(uint): timestamp del bloque actual (lo mismo que block.timestamp)

Transacción

- tx.**gasprice**(uint): precio del gas para esta transacción
- tx.**origin**(address): dirección del origen de la transacción.

Block Properties & Transaction Properties

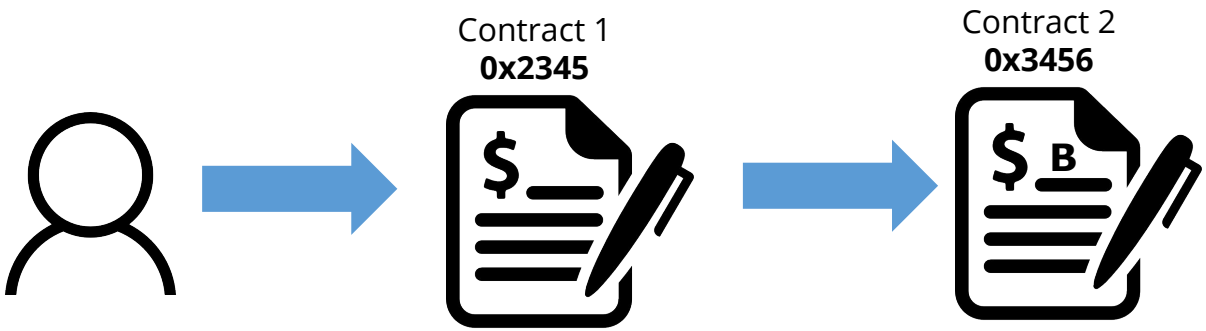
Los valores retornados como el timestamp son escritos por los mineros a la hora de confirmar.

Existe una remota posibilidad de que ese dato sea alterado por los mismos, por lo tanto se recomienda no utilizar esos valores para operaciones críticas.

También aplica para el alias “now”.

Block Properties & Transaction Properties

Una de las utilidades que nos brinda tener Block & Transaction Properties es poder hacer cosas como:



Si **tx.origin == msg.sender** quien llama al método no es un contrato.

Unidades y variables globales

Unidades y variables globales

Al desarrollar con Solidity para Ethereum, contamos con múltiples unidades y variables disponibles de manera global para su utilización en nuestros contratos inteligentes.

Estas variables pueden ser clasificadas en:

- Unidades de tiempo
- Unidades de Ether
- Propiedades del Bloque*
- Propiedades de la Transacción*
- Funciones matemáticas / criptográficas
- Funciones de dirección (address functions)
- Funciones de contrato

Unidades y variables globales

Unidades de tiempo

Sufijos como **seconds**, **minutes**, **hours**, **days**, **weeks** y **years** utilizados después de numeros literales pueden usarse para convertir unidades de tiempo donde los segundos son la unidad de base.

Tabla de equivalencias

1	== 1 seconds
1 minutes	== 60 seconds
1 hours	== 60 minutes
1 days	== 24 hours
1 weeks	== 7 days
1 year	== 265 days

Unidades y variables globales

Unidades de Ether

Un número literal puede tomar un sufijo como el **wei**, el **gwei** o el **ether** para convertirlo entre las subdenominaciones del Ether. Se asume que un número sin sufijo para representar la moneda Ether está expresado en Wei

Tabla de equivalencias

Wei	1000000000000000000
Gwei	1000000000
Ether	1

Unidades y variables globales

Funciones matemáticas / criptográficas

- **assert(bool condition)**: lanza excepción si la condición no está satisfecha.
- **addmod(uint x, uint y, uint k) returns (uint)**: computa (x + y) % k donde la suma se realiza con una precisión arbitraria y no se desborda en 2**256.
- **mulmod(uint x, uint y, uint k) returns (uint)**: computa (x * y) % k donde la multiplicación se realiza con una precisión arbitraria y no se desborda en 2**256.
- **keccak256(...) returns (bytes32)**: computa el hash de Ethereum-SHA-3 (Keccak-256) de la unión (compactada) de los argumentos.
- **sha3(...) returns (bytes32)**: equivalente a keccak256(). sha256(...) returns (bytes32): computa el hash de SHA-256 de la unión (compactada) de los argumentos.
- **ripemd160(...) returns (bytes20)**: computa el hash de RIPEMD-160 de la unión (compactada) de los argumentos.
- **ecrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address)**: recupera la dirección asociada a la clave pública de la firma de tipo curva elíptica o devuelve cero si hay un error (ejemplo de uso).
- **revert()**: aborta la ejecución y revierte los cambios de estado a como estaban.

Unidades y variables globales

Funciones de dirección (address functions)

- **<address>.balance (uint256):** balance en Wei de la dirección.
- **<address>.transfer(uint256 amount):** envía el importe deseado en Wei a la dirección o lanza excepción si falla.
- **<address>.send(uint256 amount) returns (bool):** envía el importe deseado en Wei a la dirección o devuelve false si falla.
- **<address>.call(...) returns (bool):** crea una instrucción de tipo CALL a bajo nivel o devuelve false si falla. Usa el contexto del contrato a llamar.
- **<address>.staticcall(...) returns (bool):** similar al call pero revierte si se intenta modificar algún estado.
- **<address>.delegatecall(...) returns (bool):** crea una instrucción de tipo DELEGATECALL a bajo nivel o devuelve false si falla. Usa el contexto del llamador.

Unidades y variables globales

Funciones de contrato

- **this (el tipo del contrato actual)**: el contrato actual, explícitamente convertible en Address.
- **selfdestruct(address recipient)**: destruye el contrato actual y envía los fondos que tiene a una dirección especificada.
- **Además**, todas las funciones del contrato actual se pueden llamar directamente, incluida la función actual.

¡Muchas gracias!

¡Sigamos trabajando!