이더리움

https://github.com/ethereum/wiki/wiki/%5BKorean%5D-White-Paper

한글 백서

특징

- 이더리움의 목적은 분산 어플리케이션 제작을 위한 대체 프로토콜을 만드는 것
- 튜링 완전 언어를 내장하고 있는 블록체인 기반 기술임
- 네임 코인의 기본적인 형태는 두줄 정도의 코드로 작성 가능함

```
def register(name, value):
    if !self.storage[name]:
        self.storage[name] = value
```

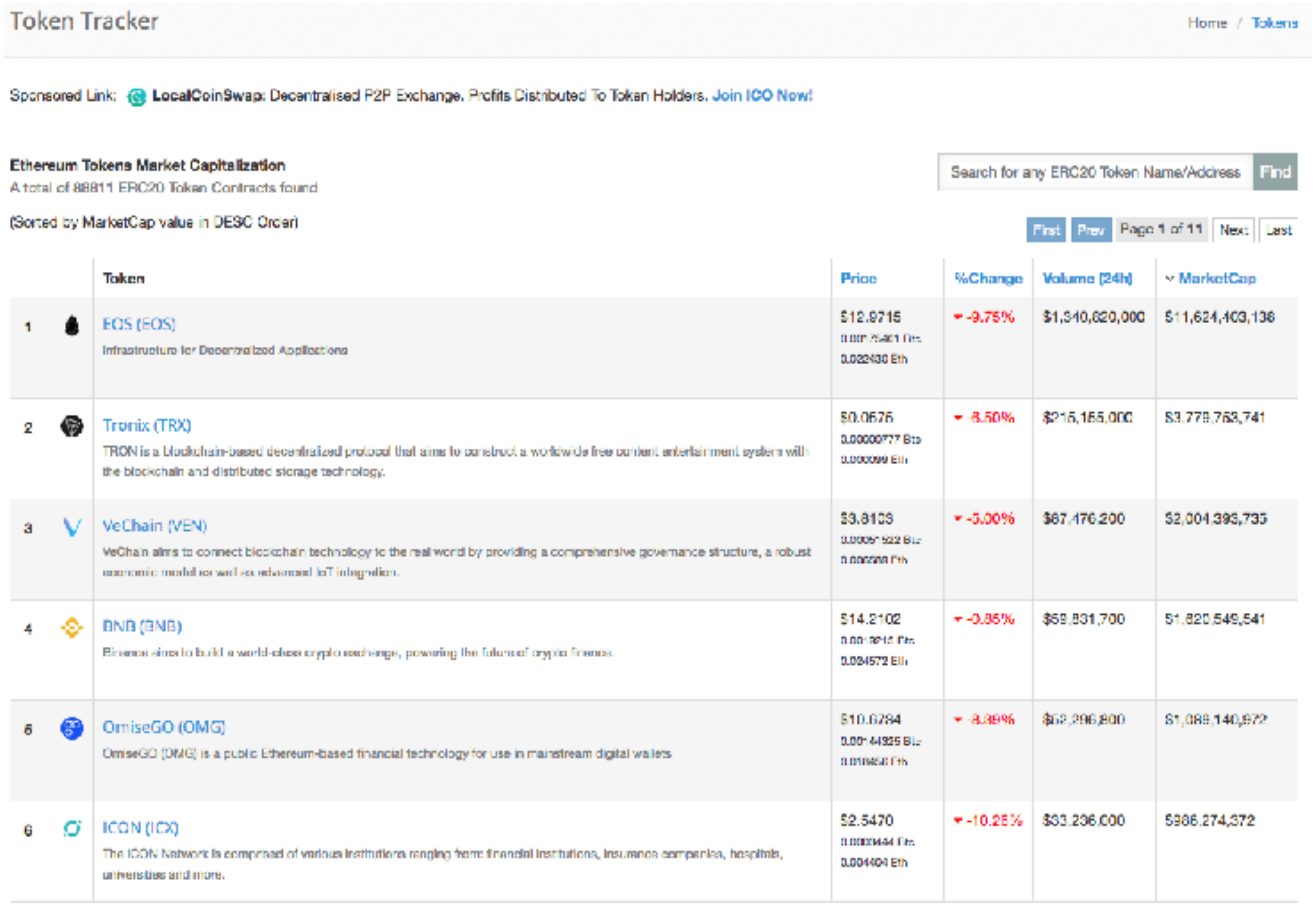
- 컬러드 코인을 지원하며 블록체인 위에 자신만의 고유한 디지털 화폐를 발행할 수 있음

```
def send(to, value):
    if self.storage[msg.sender] >= value:
        self.storage[msg.sender] = self.storage[msg.sender] - value
        self.storage[to] = self.storage[to] + value
```

ERC 20 표준 규약

(EIP - Ethereum Improvement Proposal 에서 제공된 ERC-20)

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md



Token Tracker

Home / Tokens

Sponsored Link: LocalCoinSwap: Decentralised P2P Exchange. Profits Distributed To Token Holders. Join ICO Now!

Ethereum Tokens Market Capitalization

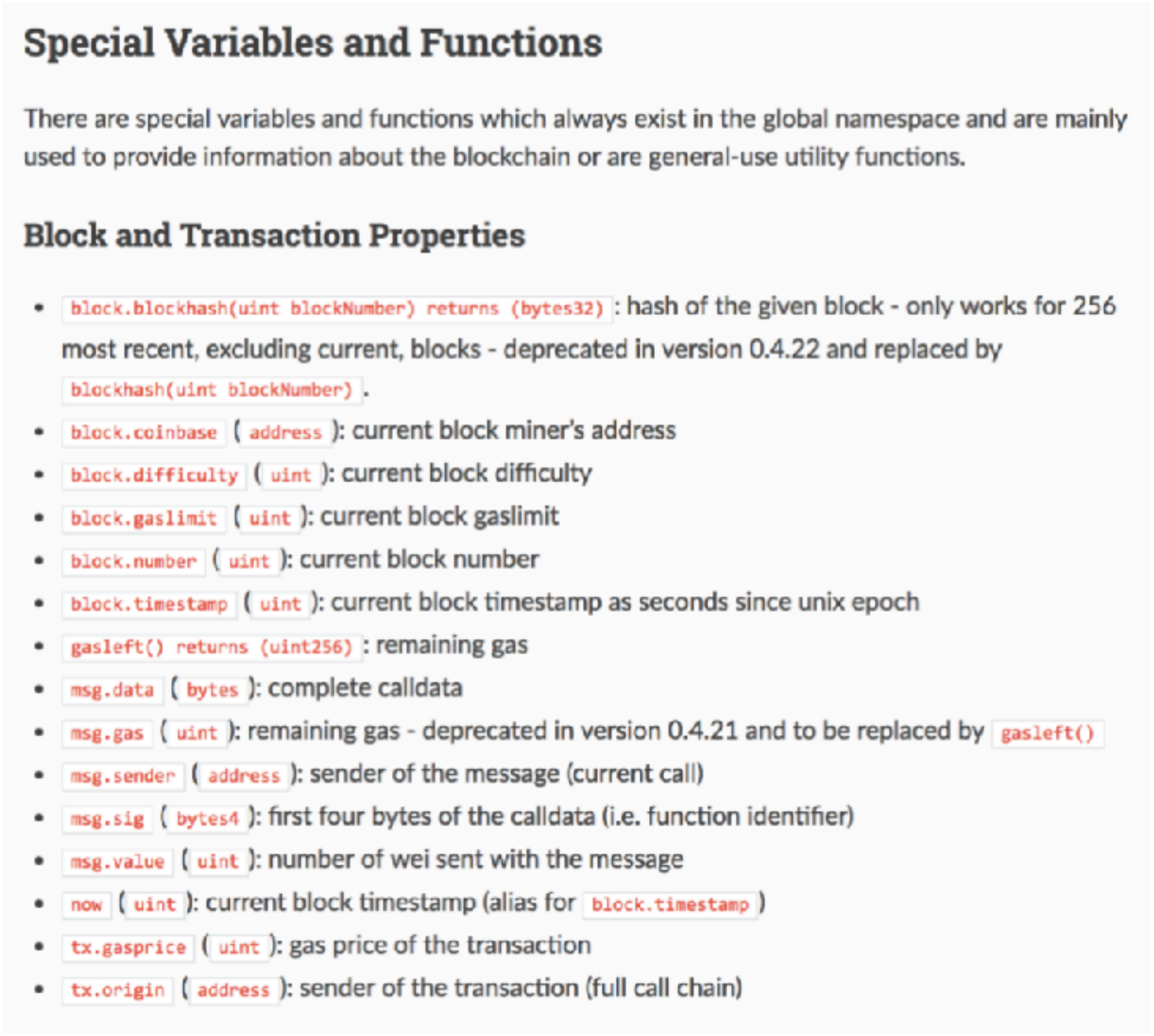
A total of 88811 ERC20 Token Contracts found

(Sorted by MarketCap value in DESC Order)

	Token	Price	%Change	Volume (24h)	MarketCap
1	EOS (EOS) Infrastructure for Decentralized Applications	$12.9715 [illegible] Btc 0.022430 Eth	-9.75%	$1,340,820,000	$11,624,403,136
2	Tronix (TRX) TRON is a blockchain-based decentralized protocol that aims to construct a worldwide free content entertainment system with the blockchain and distributed storage technology.	$0.0676 0.00000777 Btc 0.000099 Eth	-8.50%	$215,155,000	$3,779,763,741
3	VeChain (VEN) VeChain aims to connect blockchain technology to the real world by providing a comprehensive governance structure, a robust economic model as well as advanced IoT integration.	$3.8103 0.00051522 Btc 0.006588 Eth	-5.00%	$87,476,200	$2,004,393,735
4	BNB (BNB) Binance aims to build a world-class crypto exchange, powering the future of crypto finance.	$14.2102 0.0019215 Btc 0.024572 Eth	-0.85%	$59,831,700	$1,820,549,541
5	OmiseGO (OMG) OmiseGO (OMG) is a public Ethereum-based financial technology for use in mainstream digital wallets	$10.6784 0.00144325 Btc 0.018463 Eth	-8.39%	$52,296,800	$1,089,140,972
6	ICON (ICX) The ICON Network is comprised of various institutions ranging from financial institutions, insurance companies, hospitals, universities and more.	$2.5470 0.000344 Btc 0.004404 Eth	-10.25%	$33,236,000	$986,274,372

https://etherscan.io/tokens

이더리움은 블록체인 정보를 제공함



Special Variables and Functions

There are special variables and functions which always exist in the global namespace and are mainly used to provide information about the blockchain or are general-use utility functions.

Block and Transaction Properties

- `block.blockhash(uint blockNumber) returns (bytes32)` : hash of the given block - only works for 256 most recent, excluding current, blocks - deprecated in version 0.4.22 and replaced by `blockhash(uint blockNumber)`.
- `block.coinbase` (`address`): current block miner's address
- `block.difficulty` (`uint`): current block difficulty
- `block.gaslimit` (`uint`): current block gaslimit
- `block.number` (`uint`): current block number
- `block.timestamp` (`uint`): current block timestamp as seconds since unix epoch
- `gasleft() returns (uint256)` : remaining gas
- `msg.data` (`bytes`): complete calldata
- `msg.gas` (`uint`): remaining gas - deprecated in version 0.4.21 and to be replaced by `gasleft()`
- `msg.sender` (`address`): sender of the message (current call)
- `msg.sig` (`bytes4`): first four bytes of the calldata (i.e. function identifier)
- `msg.value` (`uint`): number of wei sent with the message
- `now` (`uint`): current block timestamp (alias for `block.timestamp`)
- `tx.gasprice` (`uint`): gas price of the transaction
- `tx.origin` (`address`): sender of the transaction (full call chain)

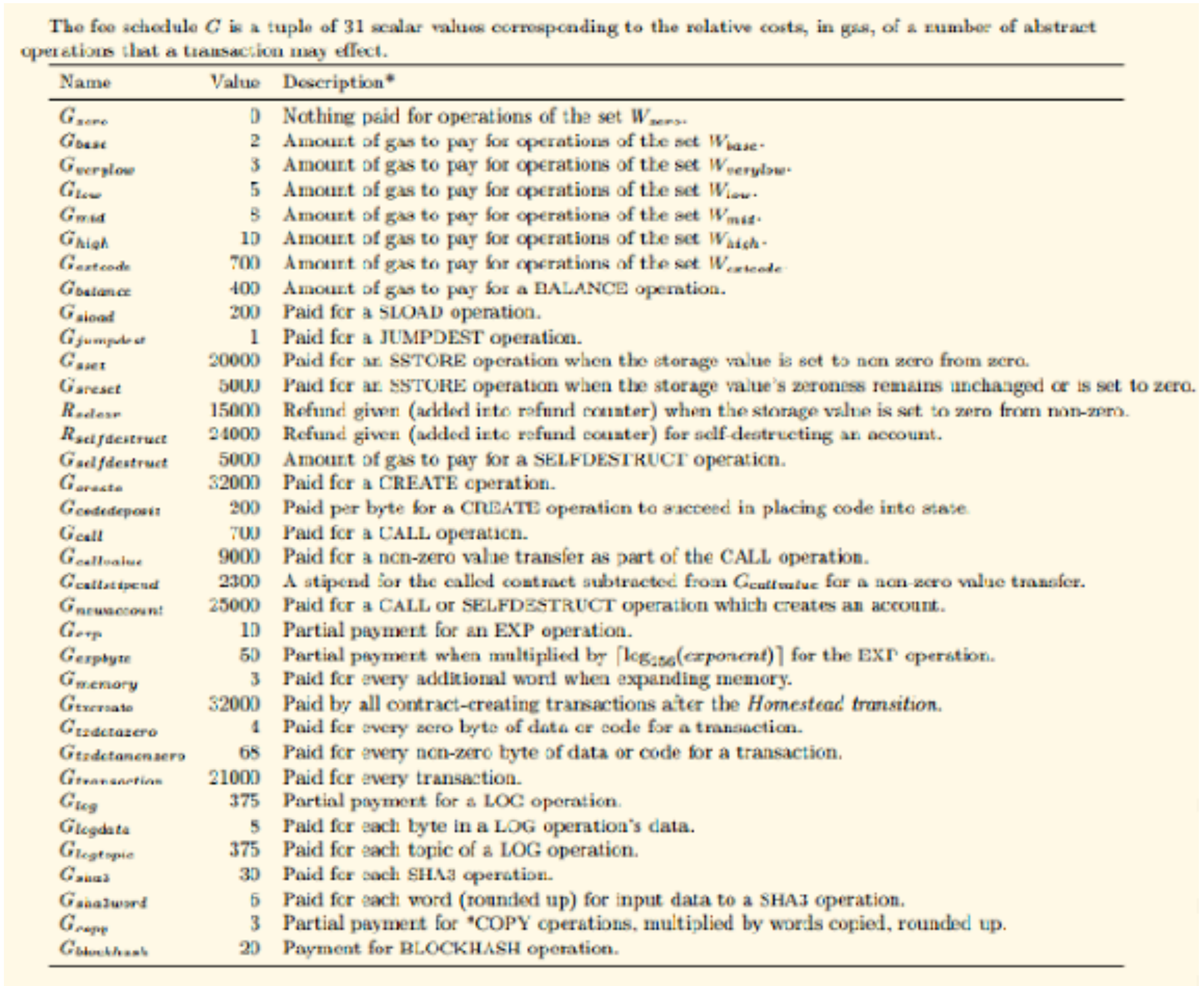
Gas (가스)

가스는 트랜잭션을 발생시켰을 때 내는 수수료의 개념 (마이너에 의하여 가스 가격이 결정 된다고 함)

가스는 수수료를 지불할 수 있는 하나의 단위이지 이더리움 네트워크의 또 다른 화폐가 아니다.

스마트 컨트랙트에는 가스의 한도가 설정되어 있음 (무한 실행 방지)

스마트 컨트랙트 송금 또는 하나의 코드를 실행하기 위해 Gas가 얼마나 소요 되는지 알아볼 수 있는 표

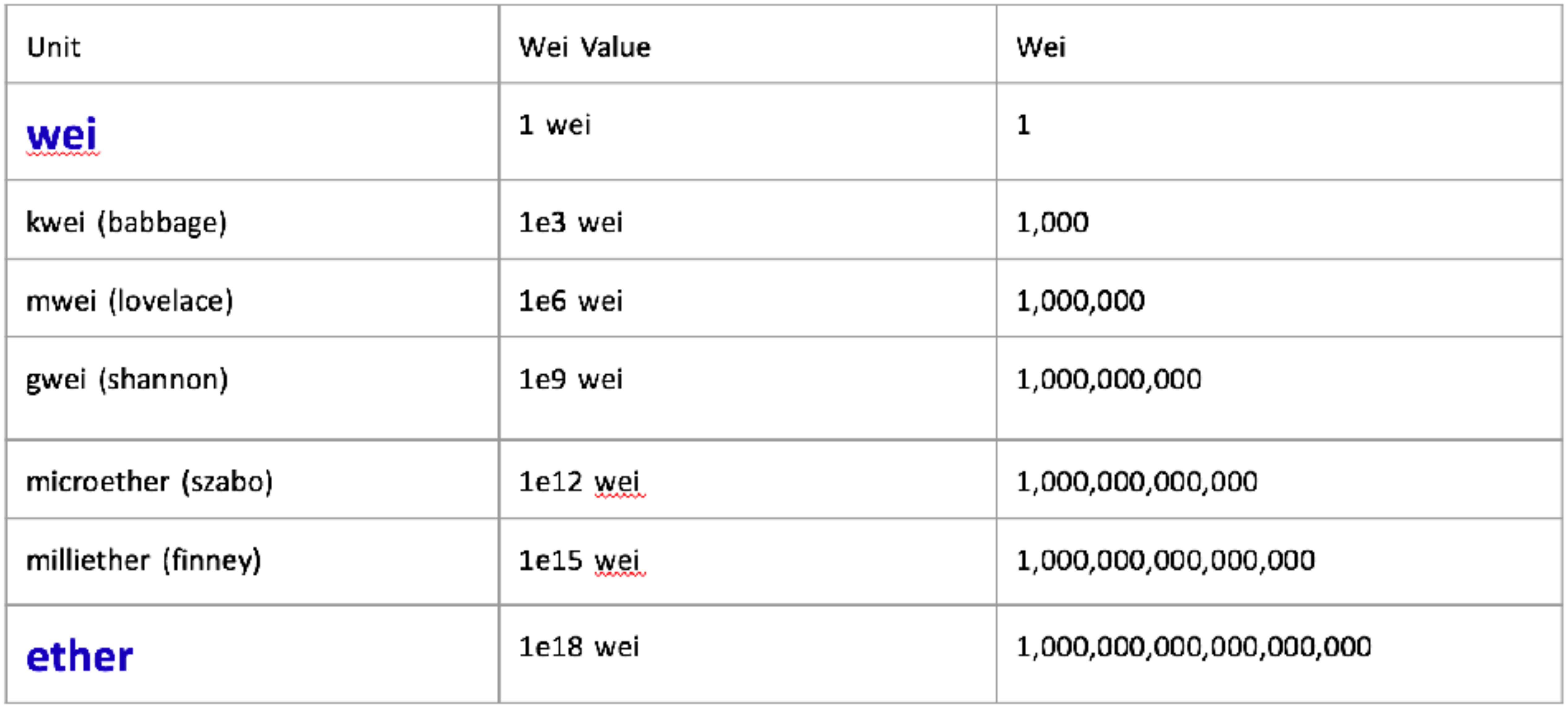


The fee schedule G is a tuple of 31 scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead transition*.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.

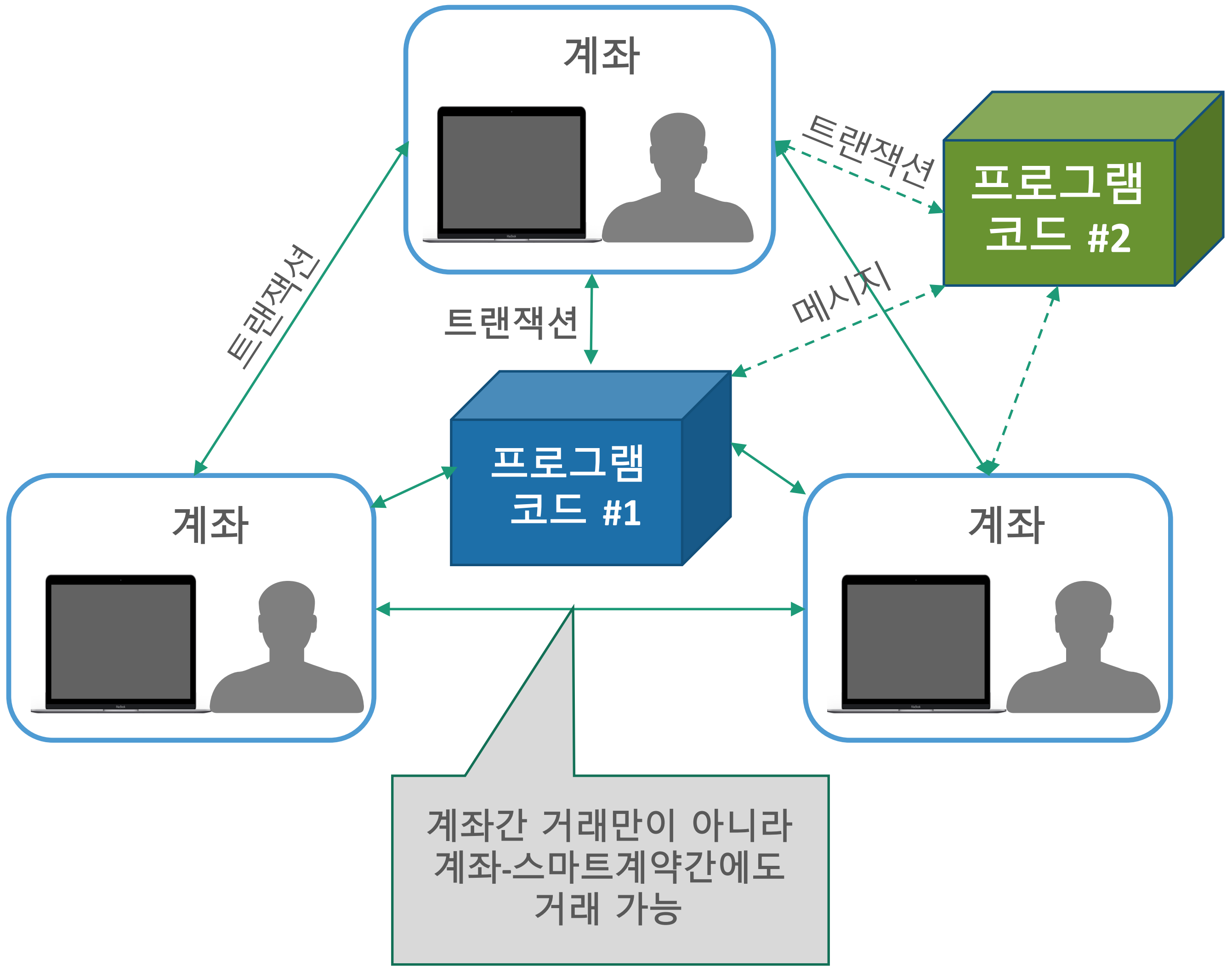
https://ethgasstation.info/

화폐 단위

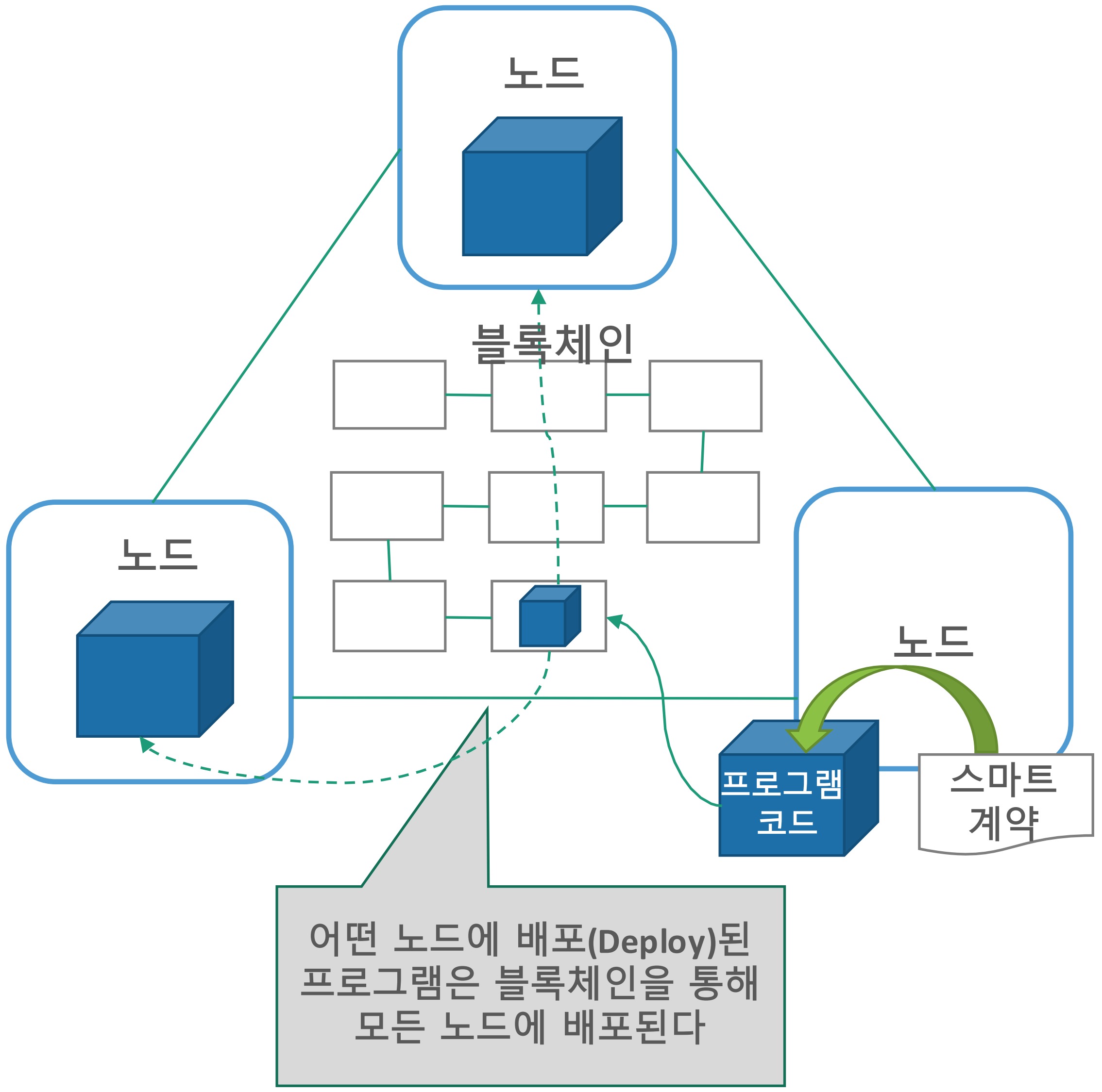


Unit	Wei Value	Wei
wei	1 wei	1
kwei (babbage)	1e3 wei	1,000
mwei (lovelace)	1e6 wei	1,000,000
gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

메시지와 트랜잭션

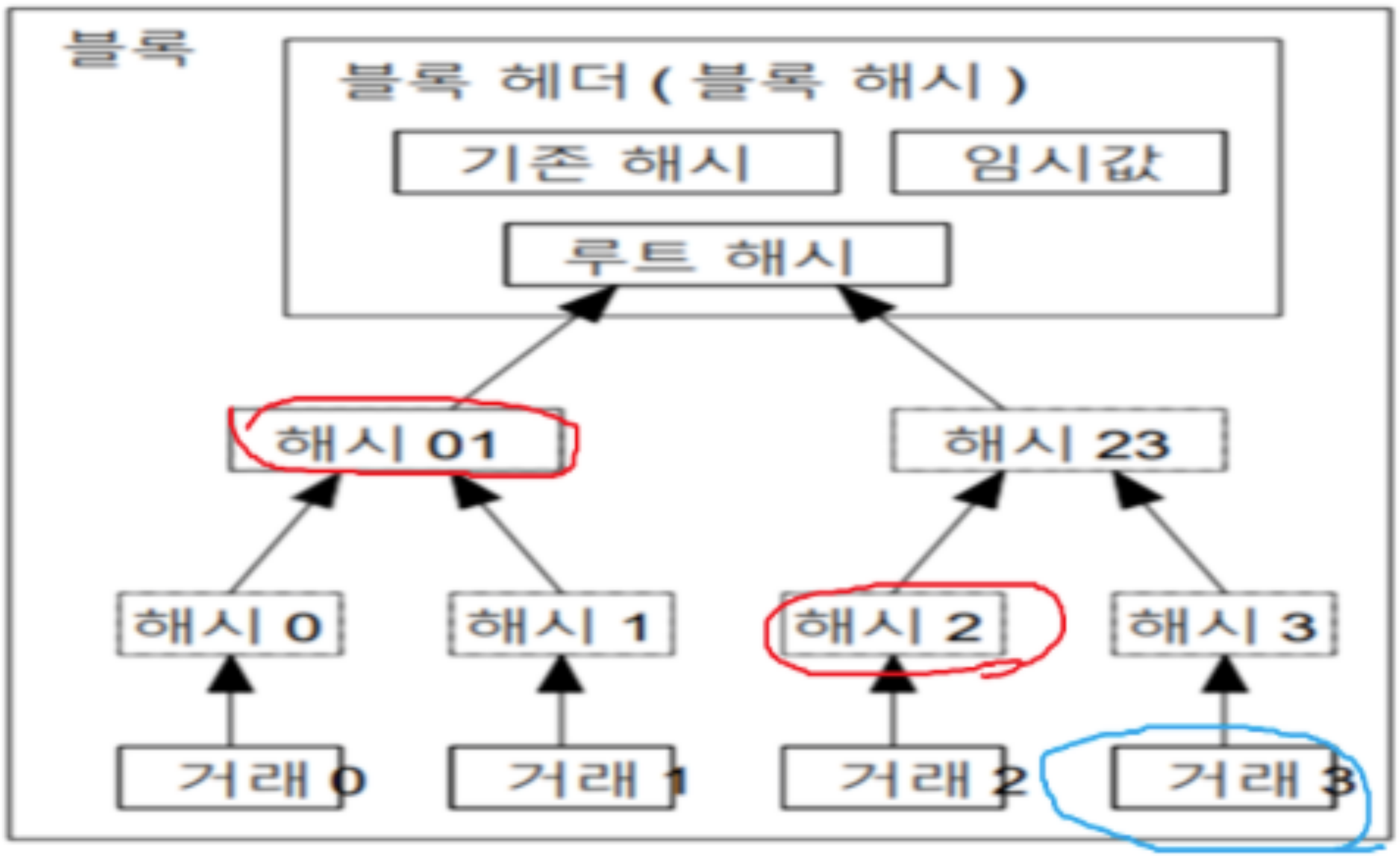


계좌간 거래만이 아니라 계좌-스마트계약간에도 거래 가능

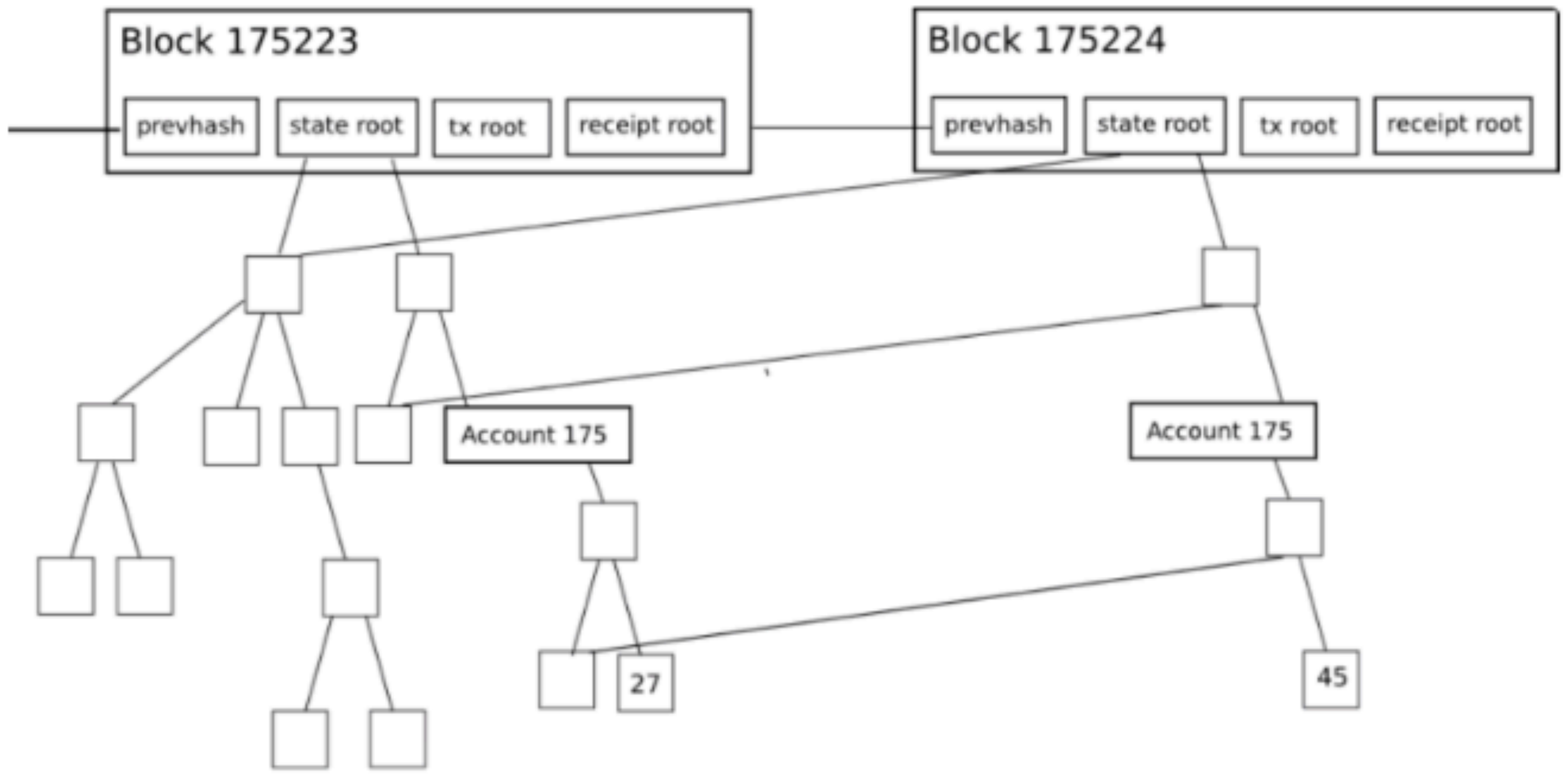


어떤 노드에 배포(Deploy)된 프로그램은 블록체인을 통해 모든 노드에 배포된다

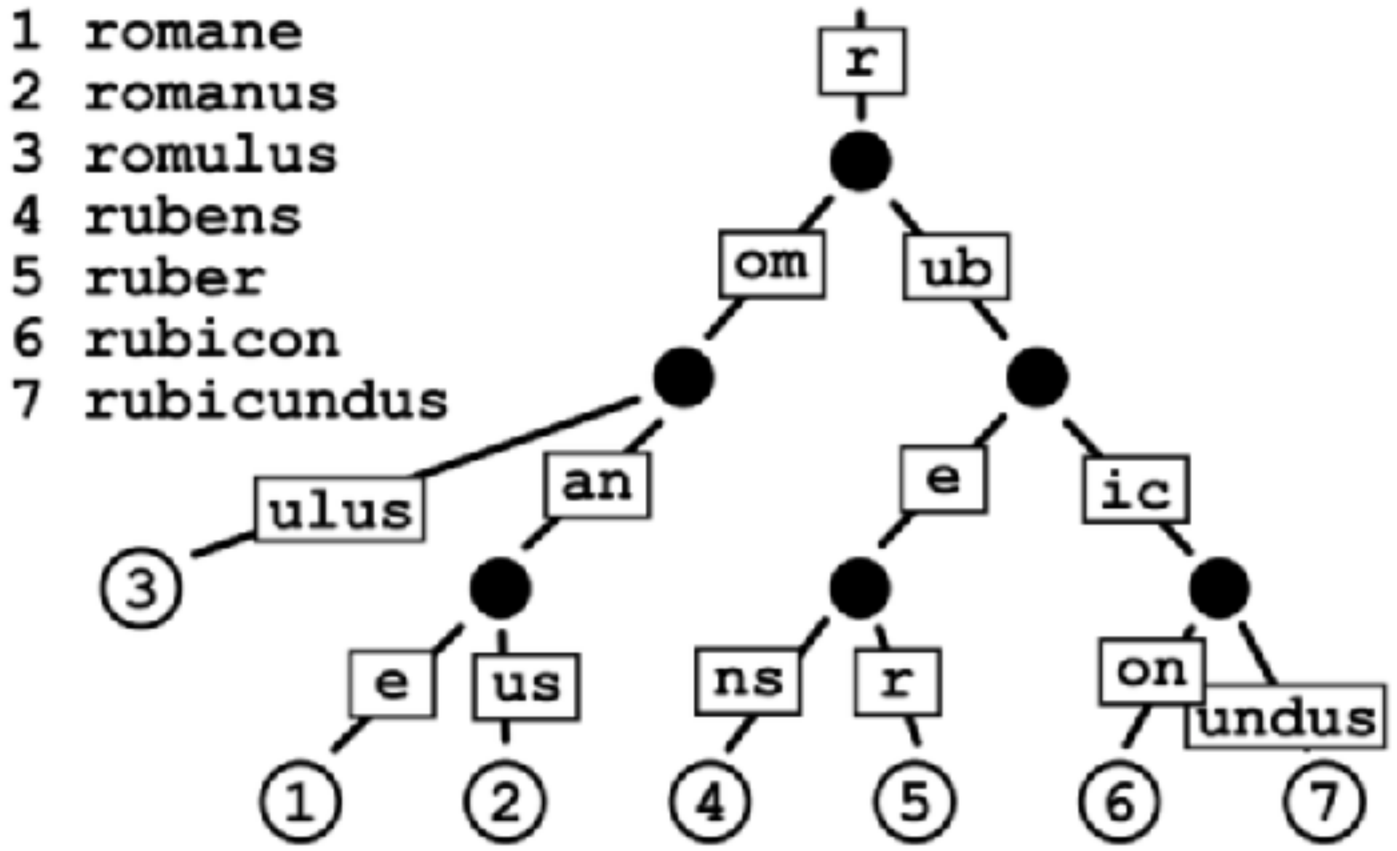
머클 바이너리 트리


블록
블록 헤더 (블록 해시)
기존 해시
임시값
루트 해시
해시 01
해시 23
해시 0
해시 1
해시 2
해시 3
거래 0
거래 1
거래 2
거래 3

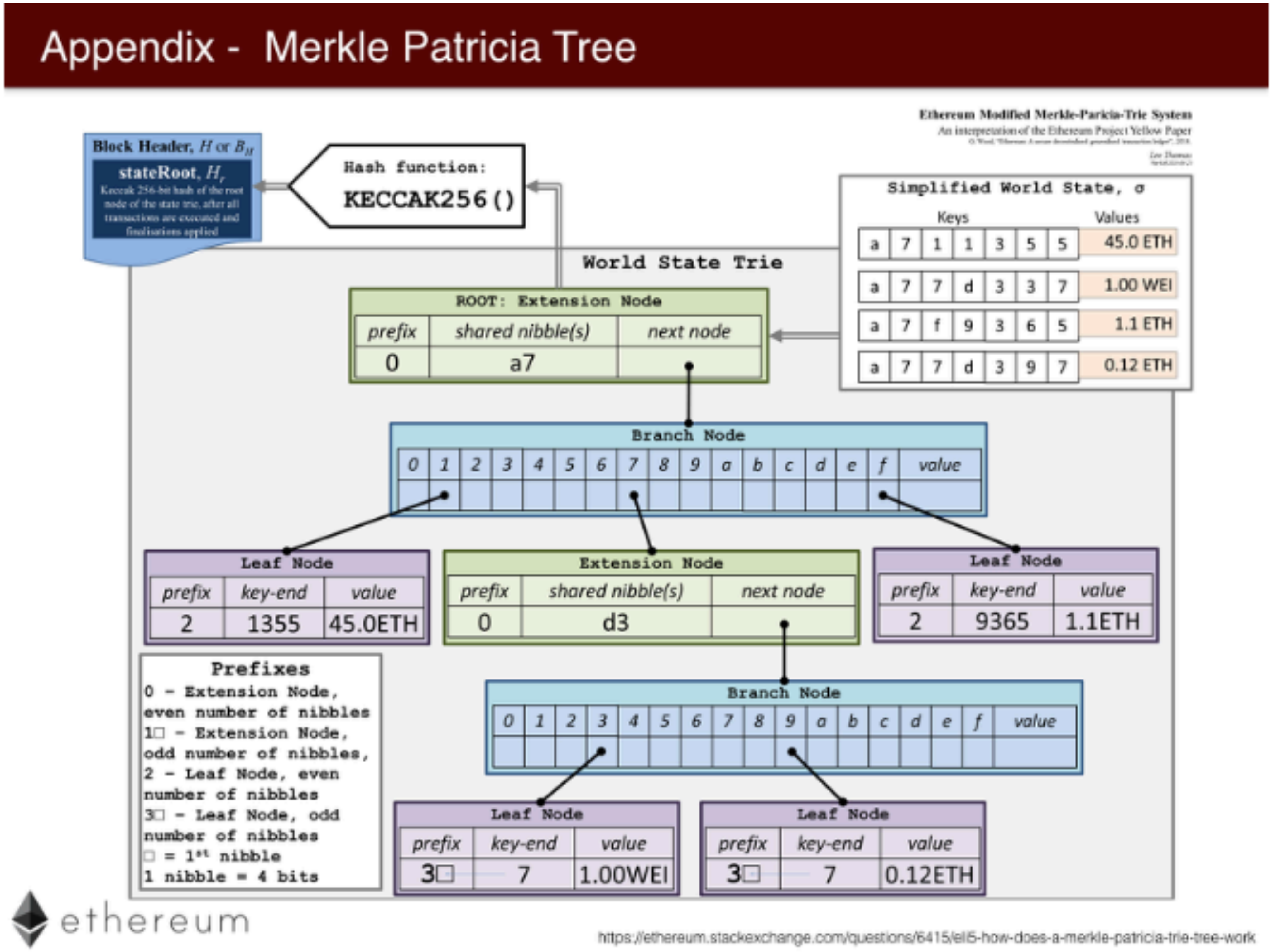
상태전이 머클트리


Block 175223
prevhash
state root
tx root
receipt root
Block 175224
prevhash
state root
tx root
receipt root
Account 175
Account 175
27
45

머클 패트리샤 트리

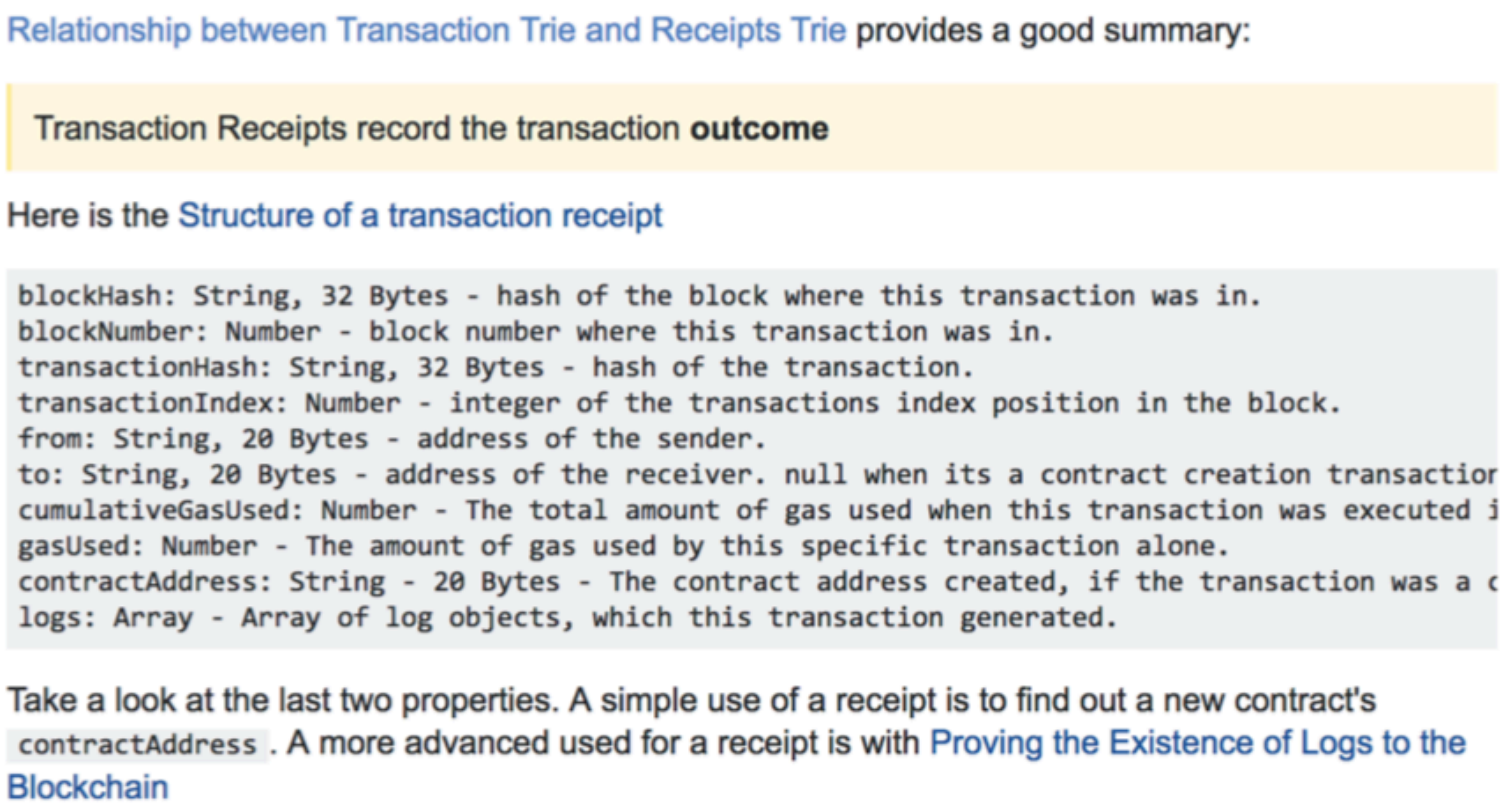

1 romane
2 romanus
3 romulus
4 rubens
5 ruber
6 rubicon
7 rubicundus
r
om
ub
ulus
an
e
ic
e
us
ns
r
on
undus
1
2
3
4
5
6
7

확장 머클 패트리샤 트리


Appendix - Merkle Patricia Tree
Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 — 45.0 ETH
a 7 7 d 3 3 7 — 1.00 WEI
a 7 f 9 3 6 5 — 1.1 ETH
a 7 7 d 3 9 7 — 0.12 ETH
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Prefixes
0 - Extension Node, even number of nibbles
1□ - Extension Node, odd number of nibbles,
2 - Leaf Node, even number of nibbles
3□ - Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH
ethereum
https://ethereum.stackexchange.com/questions/6415/eli5-how-does-a-merkle-patricia-trie-tree-work

상태전이, 머클트리, 머클 패트리샤 트리 http$/www.okjsp.pe.kr:8080/article/464145

Receipt



Relationship between Transaction Trie and Receipts Trie provides a good summary:

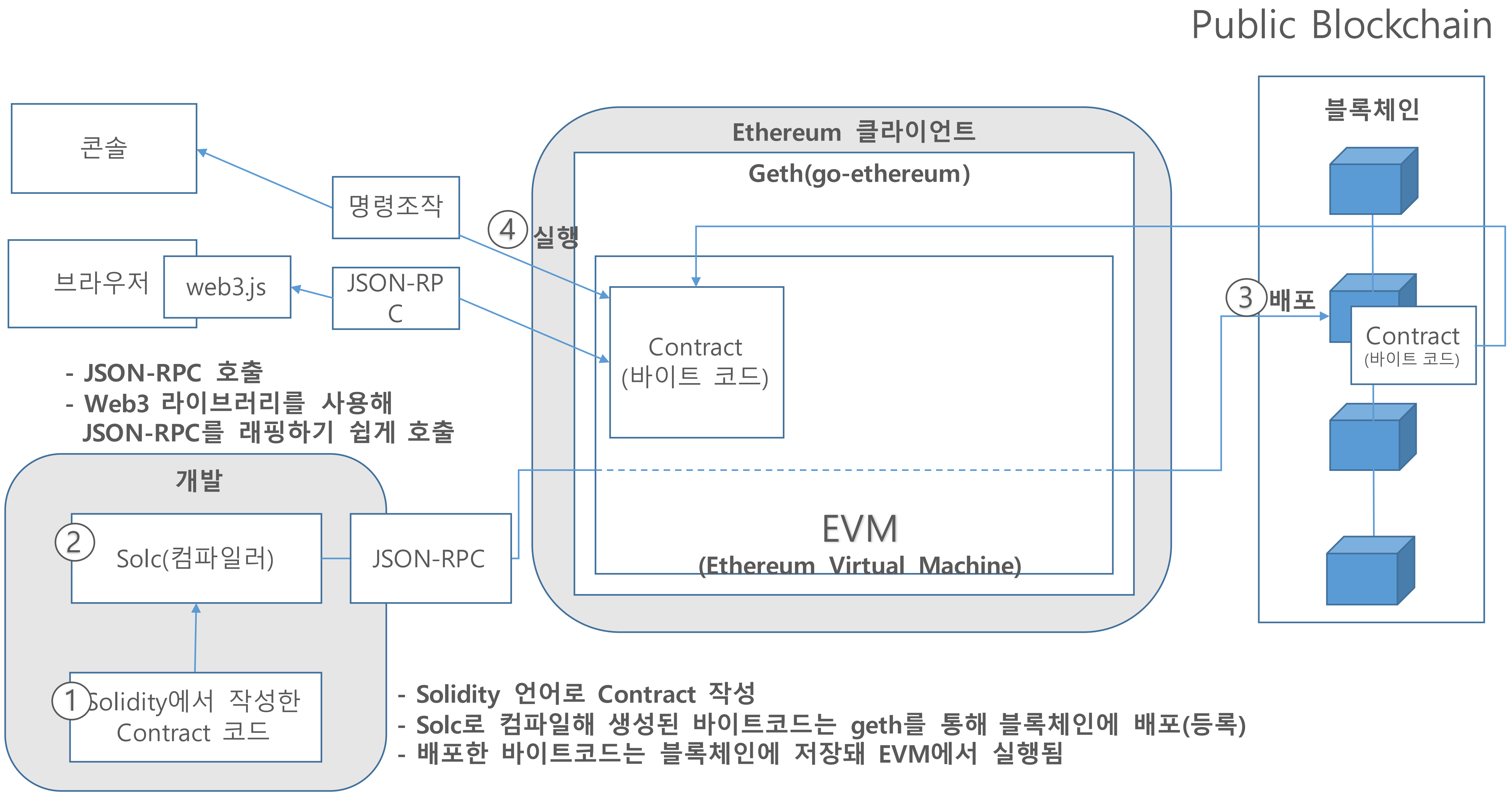
Transaction Receipts record the transaction **outcome**

Here is the Structure of a transaction receipt

```
blockHash: String, 32 Bytes - hash of the block where this transaction was in.
blockNumber: Number - block number where this transaction was in.
transactionHash: String, 32 Bytes - hash of the transaction.
transactionIndex: Number - integer of the transactions index position in the block.
from: String, 20 Bytes - address of the sender.
to: String, 20 Bytes - address of the receiver. null when its a contract creation transactior
cumulativeGasUsed: Number - The total amount of gas used when this transaction was executed i
gasUsed: Number - The amount of gas used by this specific transaction alone.
contractAddress: String - 20 Bytes - The contract address created, if the transaction was a c
logs: Array - Array of log objects, which this transaction generated.
```

Take a look at the last two properties. A simple use of a receipt is to find out a new contract's `contractAddress`. A more advanced used for a receipt is with Proving the Existence of Logs to the Blockchain

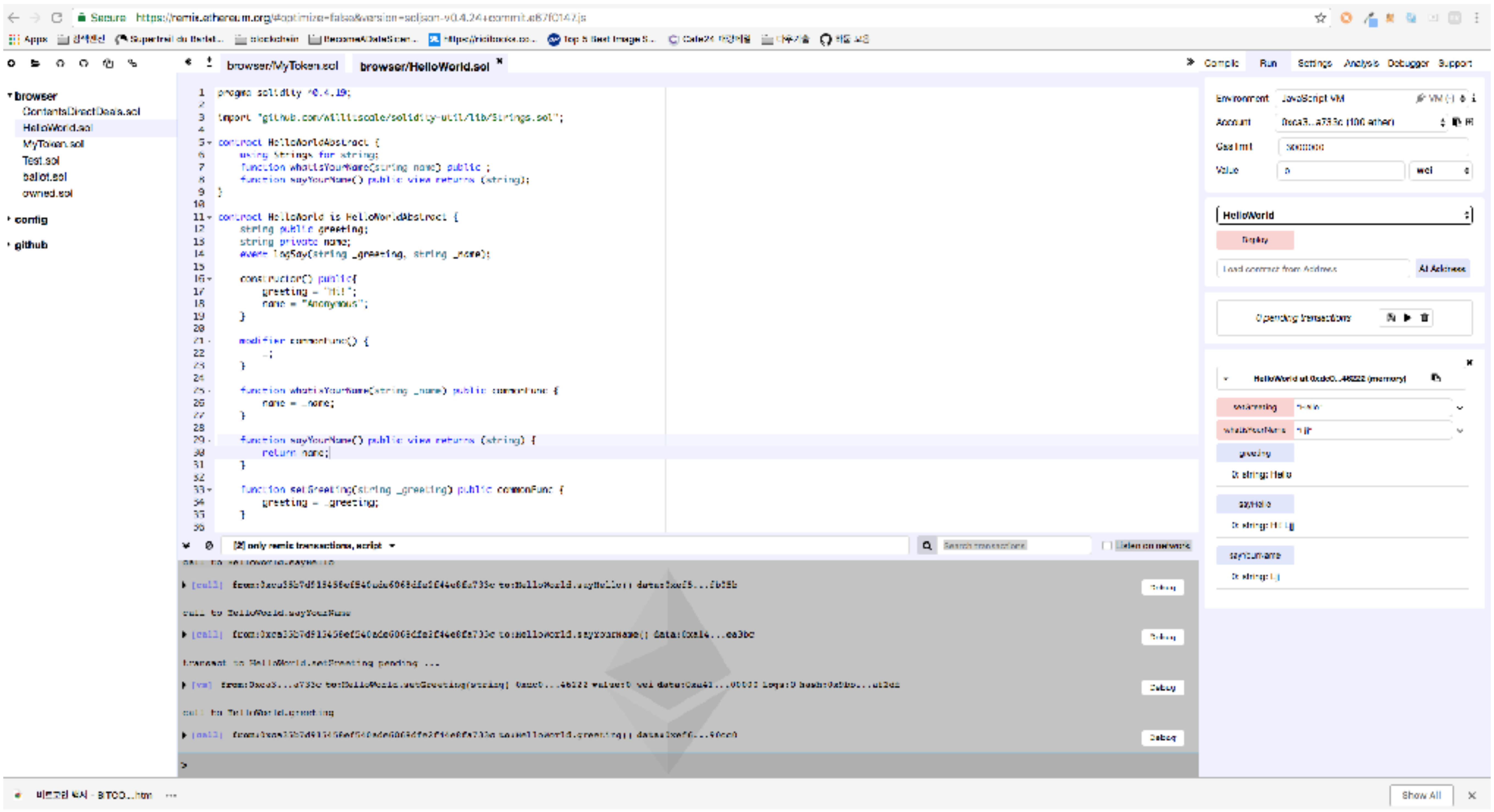
이더리움 개념도



- JSON-RPC 호출
- Web3 라이브러리를 사용해 JSON-RPC를 래핑하기 쉽게 호출

- Solidity 언어로 Contract 작성
- Solc로 컴파일해 생성된 바이트코드는 geth를 통해 블록체인에 배포(등록)
- 배포한 바이트코드는 블록체인에 저장돼 EVM에서 실행됨

솔리디티 (Solidity)



browser/MyToken.sol   browser/HelloWorld.sol

```
pragma solidity ^0.4.19;

import "github.com/willitscale/solidity-util/lib/Strings.sol";

contract HelloWorldAbstract {
    using Strings for string;
    function whatIsYourName(string name) public ;
    function sayYourName() public view returns (string);
}

contract HelloWorld is HelloWorldAbstract {
    string public greeting;
    string private name;
    event LogSay(string _greeting, string _name);

    constructor() public{
        greeting = "Hi!";
        name = "Anonymous";
    }

    modifier commonFunc() {
        _;
    }

    function whatIsYourName(string _name) public commonFunc {
        name = _name;
    }

    function sayYourName() public view returns (string) {
        return name;
    }

    function setGreeting(string _greeting) public commonFunc {
        greeting = _greeting;
    }

```

```
call to HelloWorld.sayHello
▸ [call]  from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c to:HelloWorld.sayHello() data:0xef5...fb05b
call to HelloWorld.sayYourName
▸ [call]  from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c to:HelloWorld.sayYourName() data:0xe14...ea3bc
transact to HelloWorld.setGreeting pending ...
▸ [vm]  from:0xca3...a733c to:HelloWorld.setGreeting(string) 0xde0...46122 value:0 wei data:0xa41...00000 logs:0 hash:0x5bb...a12d2
call to HelloWorld.greeting
▸ [call]  from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c to:HelloWorld.greeting() data:0xef6...90cc0
```

Environment: JavaScript VM
Account: 0xca3...a733c (100 ether)
Gas limit: 3000000
Value: 0 wei

HelloWorld at 0xde0...46122 (memory)
greeting — 0: string: Hello
sayHello — 0: string: Hi! Lj
sayYourName — 0: string: Lj

Remix 툴 활용