Chapter 5 - Solidity 101

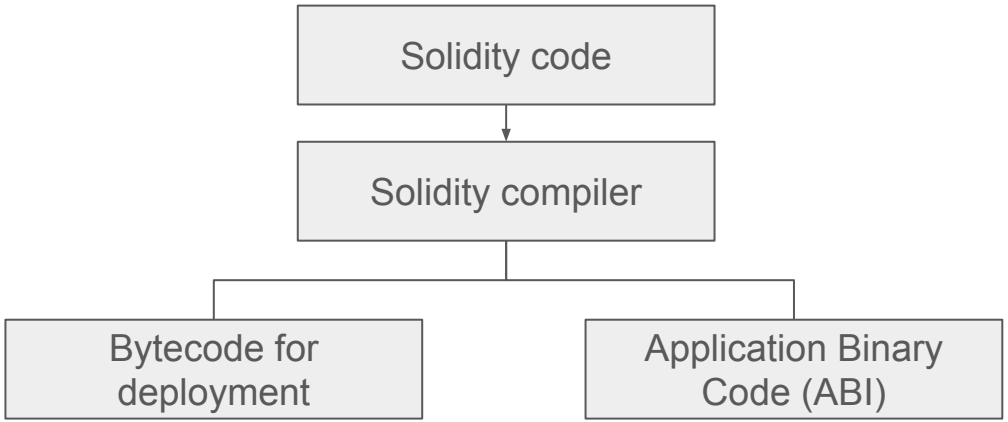
Learn Blockchain Technologies and be ready for the future.

What is the smart contract?

- An instruction is written in code which gets executed when under certain condition
- There are two types of accounts in Ethereum
  - Externally Owned Accounts (EOA) which is controlled by public-private keys
    - EOA is independent of any network
  - Contract Accounts or Smart Contract, controlled by code
    - Smart Contract is only accessible from specific network where it's deployed
- Properties inside Smart Contract transaction
  - Additional properties in smart contract transaction
    - Balance
      - Amount of ether this contract owns
    - Storage
      - Data store in this contract
    - Code
      - Machine level code for this contract

Solidity Programming Language

- The primary code is written in Ethereum smart contract
    - High-level language, influenced by C++, Python and JavaScript. Extension .sol
- Strongly typed language
    - Need to define the type for each variable
- If you know ECMAScript syntax, that would make you learn solidity faster



Solidity Compiler

- Remix (browser-based IDE with integrated compiler)
  - Recommend for quickly learn Solidity
  - https://remix.ethereum.org/
- Node.js
  - Using solcjs, a Solidity compiler
- Truffle framework
  - You can compile, deploy or test with Truffle suite

Solidity code example

Please note this contract is only for learning purpose, it doesn't fully implemented all security practices

INDUSTRY CONNECT
Connect to your Future

```
pragma solidity ^0.4.17;


contract Coin {
 // Visibility of the variable
 address public minter;
 // Mapping
 mapping (address => uint) public balances;

 // Events allow client to be notified if this event emitted
 event Sent(address from, address to, uint amount);

 // The constructor code will execute when this contract is deployed.
 constructor() public {
    minter = msg.sender;
 }

 function mint(address receiver, uint amount) public {
    require(msg.sender == minter);
    balances[receiver] += amount;
 }

 function send(address receiver, uint amount) public{
    require(balances[msg.sender] >= amount);
    balances[msg.sender] -= amount;
    balances[receiver] += amount;
    emit Sent(msg.sender, receiver, amount);
 }
}
```

Solidity code structure


```
pragma solidity ^0.4.17;
```

The version of the solidity code is written

```
contract Coin {
```

Declare a new contract name

```
address public minter;
// Mapping
mapping (address => uint) public balances;
```

Declare state permanent variables and its type

```
function mint(address receiver, uint amount) public {
    require(msg.sender == minter);
    balances[receiver] += amount;
}

function send(address receiver, uint amount) public{
    require(balances[msg.sender] >= amount);
    balances[msg.sender] -= amount;
    balances[receiver] += amount;
    emit Sent(msg.sender, receiver, amount);
}
```

Define all functions of this contract

```
// Events allow client to be notified if this event emitted
event Sent(address from, address to, uint amount);
```

Declare the event where client can be notified if this event emit

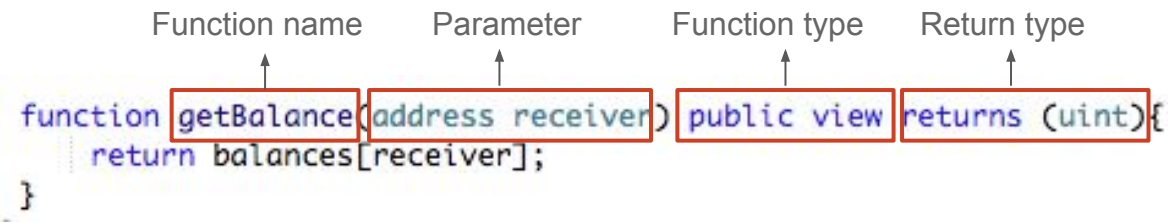
```
// The constructor code will execute when this contract is deployed.
constructor() public {
    minter = msg.sender;
}
```

The logic will be called when new contract is deployed

Solidity type

- **string**
  - Sequence of character
- **bool**
  - True or false value
- **int**
  - Integer, either positive or negative and no decimal
- **uint**
  - Unsigned integer, positive number only and no decimal
- **fixed/unfixed**
  - Fixed point number, number with a decimal after it
- **address**
  - For wallet address and has functions for checking balance or sending money

Function Declaration



- Public : anyone can access
- Private: only this contract can access
- View: this function returns data and doesn't modify data
- Pure: this function will not modify or read data
- Payable: Need to send ether to execute method

Gas and Transaction

- Ethereum use Proof of Work as consensus algorithm
  - Computation power is not free, user of blockchain have to pay for that computation
  - The payment is calculated in **Gas** and Gas is paid in **Ether**
- Gas computation
  - All transaction has a base gas cost 21,000.
  - It takes 21,000 gas if you only transferring ether and not making any state changes to the blockchain
  - If you make state changes to the blockchain, transaction takes 21,000 gas plus the addition gas associated with the change

Transaction cost = Gas usage (21,000) * Gas Price (in Wei)

That’s all the basics of Solidity

By far you can call yourself a crypto girl or boy!

Cheers.

Provided to you by Industry Connect
Education should be driven by Industry!