Solidity
CHEAT SHEET

Solidity

It is a contract-oriented high level language for implementing smart contracts. It was influenced by C++, JavaScript and Python and is designed to target the EVM

Contracts

Contracts in solidity are similar to classes in object oriented language.
A contract can be created using "new" keyword.

```
contract L {
function add(uint _p, uint _q) returns (uint) {
return _p + _q;
}
}
contract M {
address p;
function f(uint _p) {
p = new L();
}
}
```

Smart Contracts

It is a computer protocol which is used to streamline the process of contracts by digitally enforcing, verifying and managing them.

Remix, Solium& Doxity

- Remix: It is a browser based IDE with integrated compiler and solidity run-time environment without server side components.
- Solium: A linter to identify and fix style security issues in solidity.
- Doxity: Used in solidity as a documentation generator.

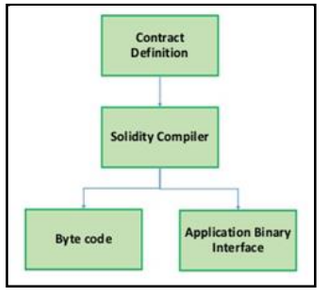
Solograph & Assembly

- Solograph: Used to visualize solidity control flow and highlight potential security vulnerabilities.
- Solidity Assembly: An assembly language which is used without solidity & inline assembly inside solidity source code

Gas & Block Gas Limit

- Gas: A measurement roughly equivalent to the computational steps
- Block Gas limit: It is used to control the amount of gas consumed during the transactions

Solidity Compiler



ABI & WEI

- ABI: A data encoding scheme called "Application Binary Interface" (ABI) is used in for working with smart contracts.
- WEI: The smallest denomination or base unit of Ether

Global Variables

block.blockhash(uint numberOfBlock) returns (bytes32): hash function of the given block which works for the 256 most recent blocks excluding the current block
block.coinbase (address): shows miner's address of current block
block.number (uint): number of current block
block.gaslimit (uint): gaslimit of the current block
block.timestamp (uint): timestamp of current block
msg.gas (uint): the gas that remains
msg.value (uint): the amount of WEI sent along with the message
msg.sender (address): address of the message sender (current call)
msg.sig (bytes4): first four bytes of the call data
now (uint): timestamp for the current block
tx.origin (address): the sender of the transaction (whole call chain)

Pragma

Used to specify certain conditions under which the source files can or cannot run.
Example:

- pragma solidity ^0.2.32
  This code compiles with a compiler function >=0.2.32
- A pseudocode example for pragma syntax
  'pragma' Identifier ([^;]+) ';'

Interface

Interface in solidity are defined as contracts, but the function bodies are omitted for the functions.
Example:

```
pragma solidity ^0.22;
interface Token
{
        function transfer(address recipient, uint amount);
}
```

Data Types

- Boolean:

true or false
Logical: ! (Logical negation), && (AND) , || (OR)
Comparisons: == (equality), != (inequality)

- Integer: Unsigned Integer, and Signed integer

Comparisons: <= (Less than or equal to), < (Less than), == (equal to), != (Not equal to), >= (Greater than or equal to), > (Greater than)
Arithmetic Operators: + (Addition), - (Subtraction), Unary –, Unary +, *(Multiplication), %(Division), ** (Exponential), << (Left shift), >>(Right shift)

- Address: Holds an Ethereum address (20 byte value).

Operators: Comparisons: <=, <, ==, !=, >= and >
Balance:

- <address>.balance (unit256): balance of address in WEI

Transfer and send:

- <address>.transfer(uint256 amount): send given amount of Wei to Address, throws on failure
- <address>.send(uint256 amount) returns (bool): send given amount of Wei to Address, returns false on failure

- Global functions:

- keccak256(…) returns (bytes32)- computes the ethereum SHA-3 hash associated with the arguments
- sha256(…) returns (bytes32) – Computes the SHA-256 argument hash
- mulmod(uint a, uint b, uint c) returns (unit) : It computes (a*b)%c, if the multiplication gets executed using arbitrary precision, thus not wrapping around 2**256
- addmod(uint p, uint q, uint m) returns (uint): Computes (p+q)%m
- ecrecover(bytes32 _hash, uint8 _v, bytes32 _r, bytes32 _s) returns (address): recovers the address linked with the public key and if an error occurs, zero is returned
- ripemd160(…) returns (bytes20): computes the RIPEMD-160 (tightly packed) argument hash
- <address>.balance (uint256) : Returns the balance of the specified address in WEI
- <address>.send(uint256 amount) returns (bool): It sends the specified number of WEI to the address given, on failure it returns false as an output
- <address>.transfer(uint256 amount): Sends specified number of WEI to the particular address, on failure throws

- Access Modifiers:

- Public: Accessible from the current contract, inherited contracts and externally
- Private: Accessible only from the current contract
- Internal: Accessible only from current contract and contracts inheriting from it
- External: Can be accessed externally only

Important Terms

- Voting: When the contract is quiet complex, it uses voting contract, it shows how delegated voting can be done so that vote counting is automatic and completely transparent at the same time
- Delegate call: It is a reusable library code that can be applied to a contracts storage in solidity.
- Logs: A feature called logs is used in solidity to implement events.
- NPM: It is a convenient and portable way to install Solidity compiler.
- Truffle: It is a test bed that will allow to easily test and compile the smart contract.
- Inheritance: In solidity inheritance is more syntactic. In the final compilation the compiler copies the parent class members to create the bytecode of the derived contract with the copied members.
- This: This is a keyword that refers to the instance of the contract where the call is made.
- msg.sender: This function refers to the address where the contract is being called from.
- Pure: It is a modifier which assures not to be read from or modified
- View: In solidity, view is a function that promises to not modify the state of the contract.
- Suicide: It is a alias for self destruct function, that is used to destroy the current contract.
- Delete: Delete is used to delete all elements in an array.
- Block.coinbase(address): It refers to the miner's address of the current block.
- Address(contractVar).send(amount): If a built-in send function needs to be used, it can be accessed using this function.
- Super: It is used to refer the contract that is higher by one level in the inheritance hierarchy.

Transaction Example