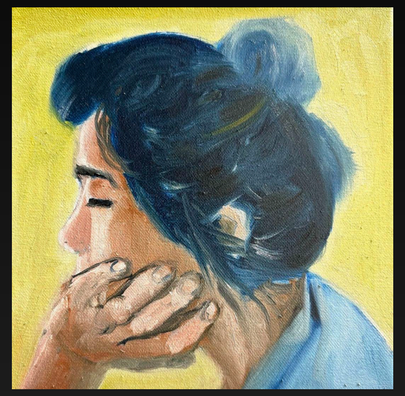
RSS3 Crossbell OpenBuild

Smart Contract Development

In Solidity

pseudoyu

About me



pseudoyu (x.com/pseudo_yu)

Blockchain developer | github.com/pseudoyu

- Tech writer/content creator | pseudoyu.com
- Core dev at RSS3 & Crossbell
- Solidity courses/tutorials contributor of OpenBuild

What's Smart Contract

Smart contracts are programs that run on the blockchain

Contract developers can interact with on-chain assets and data through smart contracts

Users can call contracts, access assets and data through their own on-chain accounts

Differences from the general programs

Native support for asset

Deployment and transactions require certain costs

The cost of storing data is higher

Unable to modify after deployment (Upgradable smart contracts?)

...

Solidity

A high-level programming language for the implementation of smart contracts

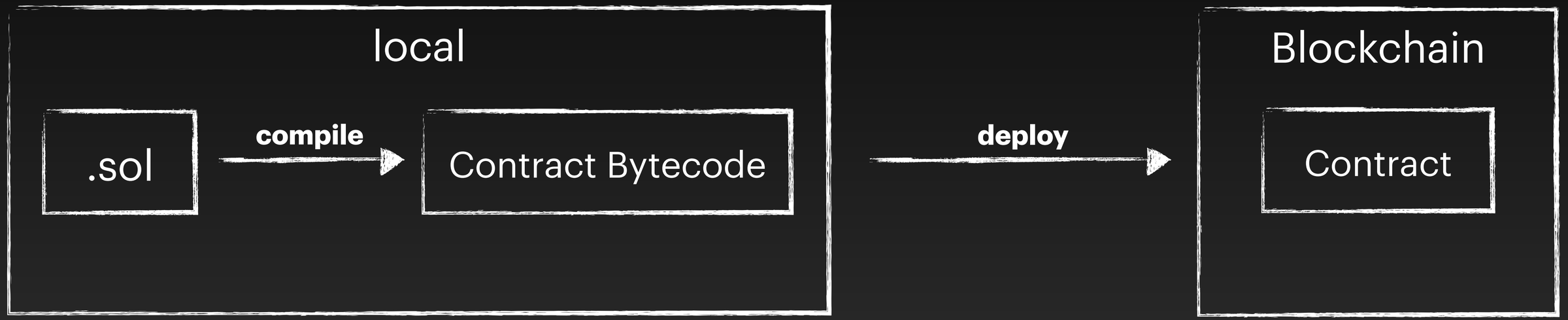
Run on the EVM

The syntax is similar to JavaScript

The most popular smart contract language

The language you need to know to get started with blockchain and web3

How to Deploy Smart Contracts



Solidity contracts are files with the .sol extension and cannot be executed directly. They need to be compiled into bytecode that can be recognized by the Ethereum Virtual Machine (EVM) in order to run on the blockchain.

Development Frameworks & Tools

Brownie

(python)

Simple python syntax

for local debugging and deployment testing

Hardhat

(javascript)

Rich plugins

Suitable for complex contract projects

Foundry

(solidity)

Writing contracts, tests and scripts all in solidity

Development Frameworks & Tools

Remix

IDE

→ Provides IDE, compile tools, deployment and debugging test node environment, accounts, etc., which can directly interact with the test network and main network through the MetaMask plugin

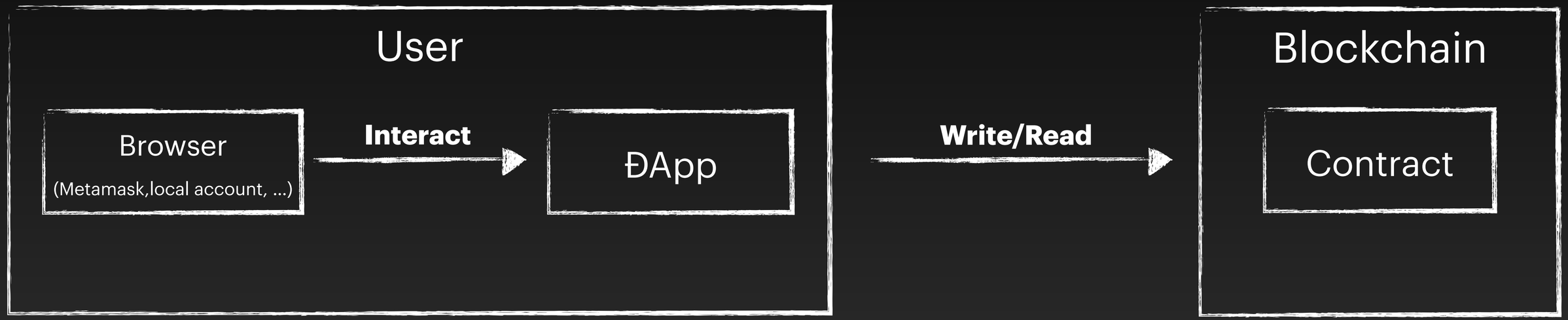
Ganache/Anvil

← Provides a virtual blockchain network that can interact with Remix IDE or some framework tools, suitable for local debugging and testing of projects with a certain scale

Others

→ MetaMask
Infura
OpenZeppelin
Chainlink
...

How to interact with smart contracts



Users typically connect wallets via browser extensions to interact with smart contracts, calling methods and reading on-chain data.

Viem

TypeScript Interface for Ethereum

More usable and efficient compared to ethers.js

Better Performance

...

Solidity Core

Data Type

Basic

boolean

int

uint

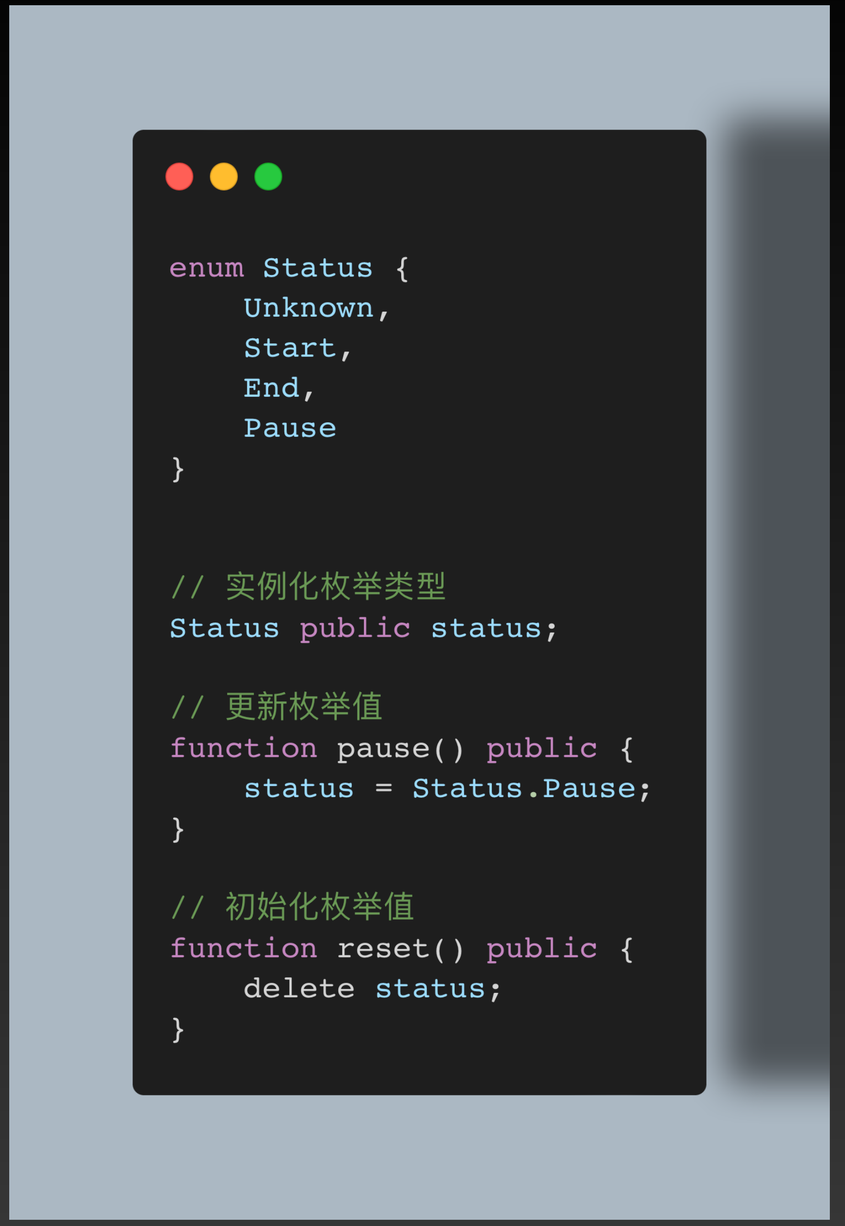
address

bytes

...

Data Type

enum



```
enum Status {
    Unknown,
    Start,
    End,
    Pause
}


// 实例化枚举类型
Status public status;

// 更新枚举值
function pause() public {
    status = Status.Pause;
}

// 初始化枚举值
function reset() public {
    delete status;
}
```

Data Type

array

```
// 定义数组类型
uint[7] public arr;

// 添加数据
arr.push(7);

// 删除最后一个数据
arr.pop();

// 删除某个索引值数据
delete arr[1];

// 获取数组长度
uint len = arr.length;
```

Data Type
mapping

```
// 定义嵌套 mapping 类型
mapping(string => mapping(string => string)) nestedMap;

// 设置值
nestedMap[id][key] = "0707";

// 读取值
string value = nestedMap[id][key];

// 删除值
delete nestedMap[id][key];
```

Data Type

struct

```
contract Struct {
    struct Data {
        string id;
        string hash;
    }

    Data public data;

    // 添加数据
    function create(string calldata _id) public {
        data = Data{id: _id, hash: "111222"};
    }

    // 更新数据
    function update(string _id) public {
        // 查询数据
        string id = data.id;

        // 更新
        data.hash = "222333"
    }
}
```

Variable

Type

local

state

global

Keywords

storage

memory

calldata

constant / immutable

constant

can’t be changed

reduce gas fee

immutable

can be initiated in constructor

but can’t be changed later

Function

Visibility and Keywords

Visibility

public

private

internal

external

Keywords

view

pure

Function
modifier

```
modifier onlyOwner() {
    require(msg.sender == owner, "Not owner");
    _;
}

modifier validAddress(address _addr) {
    require(_addr != address(0), "Not valid address");
    _;
}

modifier noReentrancy() {
    require(!locked, "No reentrancy");
    locked = true;
    _;
    locked = false;
}

function changeOwner(address _newOwner) public onlyOwner validAddress(_newOwner) {
    owner = _newOwner;
}

function decrement(uint i) public noReentrancy {
    x -= i;

    if (i > 1) {
        decrement(i - 1);
    }
}
```

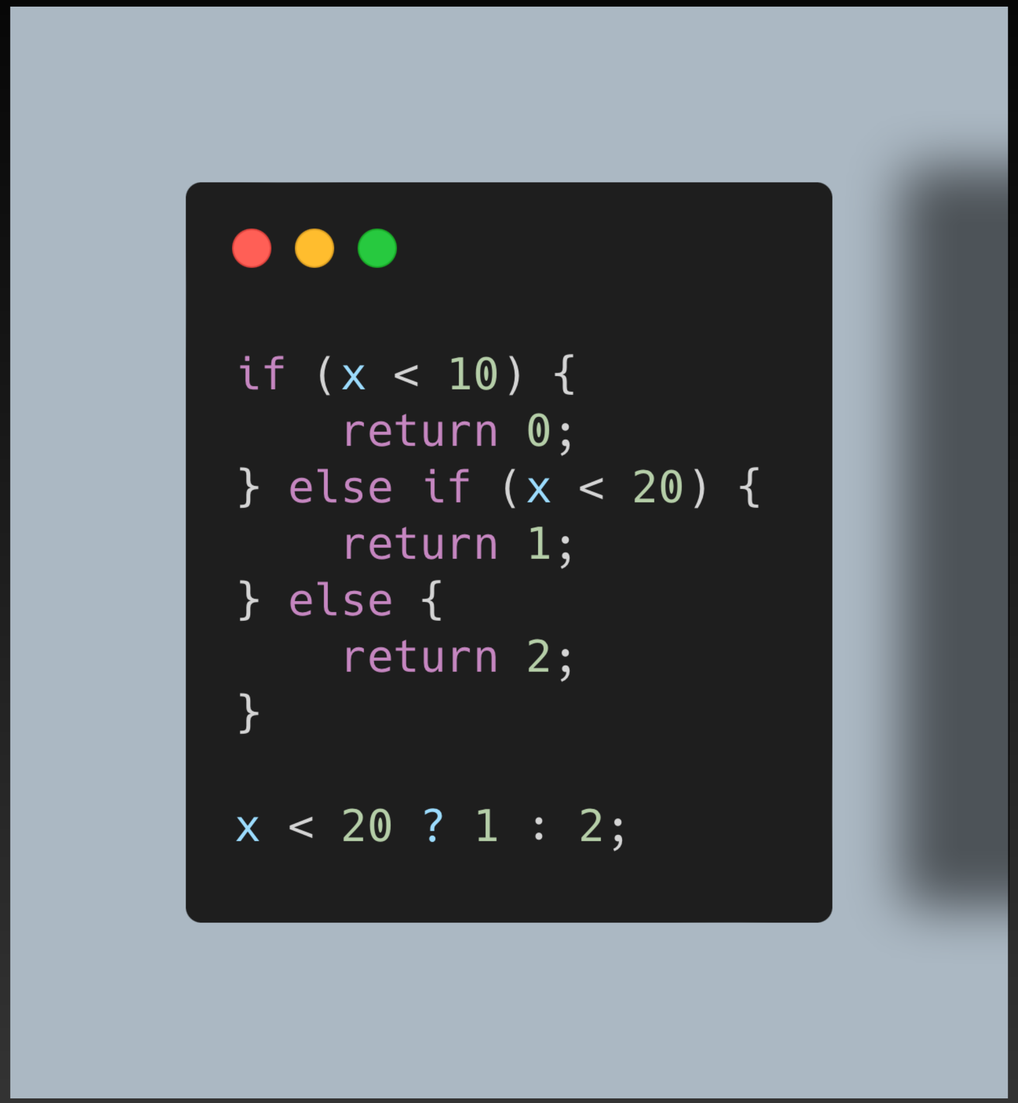
Function
selector

```
addr.call(abi.encodeWithSignature("transfer(address,uint256)", 0xSomeAddress, 123))

contract FunctionSelector {
    function getSelector(string calldata _func) external pure returns (bytes4) {
        return bytes4(keccak256(bytes(_func)));
    }
}
```

Condition

if / else if / else

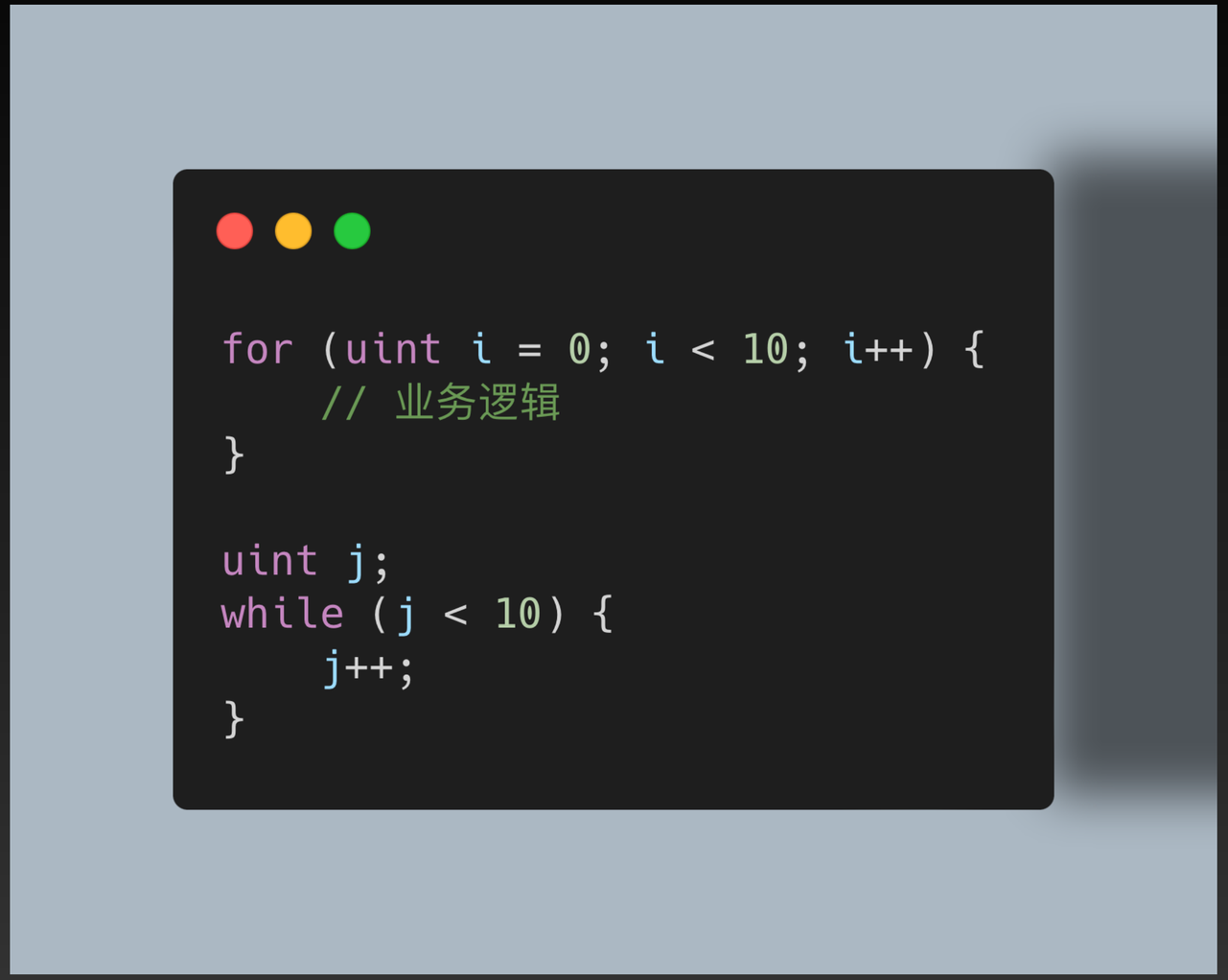


```
if (x < 10) {
    return 0;
} else if (x < 20) {
    return 1;
} else {
    return 2;
}

x < 20 ? 1 : 2;
```

Loop

for / while / do while

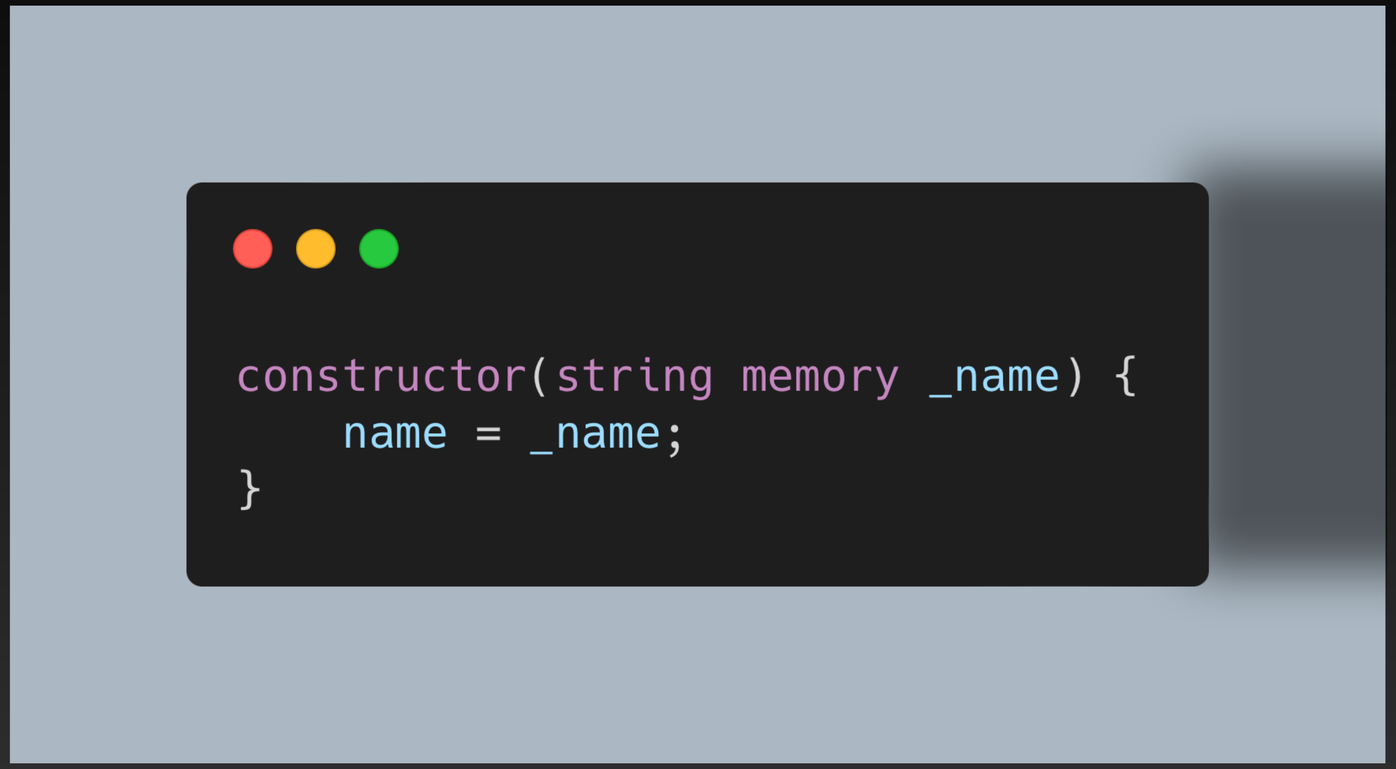


```
for (uint i = 0; i < 10; i++) {
    // 业务逻辑
}

uint j;
while (j < 10) {
    j++;
}
```

Contract

constructor



```
constructor(string memory _name) {
    name = _name;
}
```

Contract
interface

```
contract Counter {
    uint public count;

    function increment() external {
        count += 1;
    }
}

interface ICounter {
    function count() external view returns (uint);
    function increment() external;
}

contract MyContract {
    function incrementCounter(address _counter) external {
        ICounter(_counter).increment();
    }

    function getCount(address _counter) external view returns (uint) {
        return ICounter(_counter).count();
    }
}
```

Contract
Inheritance

```
// 定义父合约 A
contract A {
    function foo() public pure virtual returns (string memory) {
        return "A";
    }
}

// B 合约继承 A 合约并重写函数
contract B is A {
    function foo() public pure virtual override returns (string memory) {
        return "B";
    }
}

// D 合约继承 B、C 合约并重写函数
contract D is B, C {
    function foo() public pure override(B, C) returns (string memory) {
        return super.foo();
    }
}

contract B is A {
    function foo() public virtual override {
        // 直接调用
        A.foo();
    }

    function bar() public virtual override {
        // 通过 super 关键字调用
        super.bar();
    }
}
```

Contract

create

```
function create(address _owner, string memory _model) public {
    Car car = new Car(_owner, _model);
    cars.push(car);
}
```

```
function create2(address _owner, string memory _model, bytes32 _salt) public {
    Car car = (new Car){salt: _salt}(_owner, _model);
    cars.push(car);
}
```

Contract

import

```
// 本地导入
import "./Foo.sol";

// 外部导入
import "https://github.com/owner/repo/blob/branch/path/to/Contract.sol";
```

Contract

import library

```
library SafeMath {
    function add(uint x, uint y) internal pure returns (uint) {
        uint z = x + y;
        require(z >= x, "uint overflow");
        return z;
    }
}

contract TestSafeMath {
    using SafeMath for uint;
}
```

Contract

event

```
// 定义事件
event Log(address indexed sender, string message);
event AnotherLog();

// 抛出事件
emit Log(msg.sender, "Hello World!");
emit Log(msg.sender, "Hello EVM!");
emit AnotherLog();
```

Error Handling

require / revert / assert

```
function testRequire(uint _i) public pure {
    require(_i > 10, "Input must be greater than 10");
}

function testRevert(uint _i) public pure {
    if (_i <= 10) {
        revert("Input must be greater than 10");
    }
}

function testAssert() public view {
    assert(num == 0);
}
```

Error Handling

try / catch

```
event Log(string message);
event LogBytes(bytes data);

function tryCatchNewContract(address _owner) public {
    try new Foo(_owner) returns (Foo foo) {
        emit Log("Foo created");
    } catch Error(string memory reason) {
        emit Log(reason);
    } catch (bytes memory reason) {
        emit LogBytes(reason);
    }
}
```

Assets

payable

```
// 地址类型可以声明 payable
address payable public owner;

constructor() payable {
    owner = payable(msg.sender);
}

// 方法声明 payable 来接收 Ether
function deposit() public payable {}
```

Assets

send

```
contract SendEther {
  function sendViaCall(address payable _to) public payable {
    (bool sent, bytes memory data) = _to.call{value: msg.value}("");
    require(sent, "Failed to send Ether");
  }
}
```

Assets

receive

```
contract ReceiveEther {

    // 当 msg.data 为空时
    receive() external payable {}

    // 当 msg.data 非空时
    fallback() external payable {}

    function getBalance() public view returns (uint) {
        return address(this).balance;
    }
}
```

Assets

Gas Fee

params

gas spent

gas price

gas limit

block gas limit

tips

use calldata instead of memory

load state variables into memory

cache array elements

...

References

Demo code repository
https://github.com/pseudoyu/social-dApp-demo-contracts

Solidity Smart Contract Development - Basics
https://www.pseudoyu.com/en/2022/05/25/learn_solidity_from_scratch_basic/

foundry-starter-kit
https://github.com/pseudoyu/foundry-starter-kit

Other resources
Solidity by Example
Learn Solidity, Blockchain Development, & Smart Contracts

Thank you!

emit Log(msg.sender, “Thank You”);