Move Programming Language

Dang Quang Vu

February 17, 2023

Contents

1 Introduction

- Chào mừng bạn đến với thế giới của ngôn ngữ lập trình Move! Trong khóa học này, chúng tôi sẽ đưa bạn đi từ con số 0 đến trở thành chuyên gia và chỉ cho bạn cách sử dụng kiến thức mới để tạo ra các smart contract mạnh mẽ và an toàn trên giao thức blockchain Aptos & Sui.

- Move là một ngôn ngữ lập trình mới được Facebook tạo ra cho dự án blockchain Libra của họ. Đây là một ngôn ngữ được thiết kế đặc biệt để phát triển các smart contract an toàn và đáng tin cậy, và đã thu hút rất nhiều sự chú ý trong thế giới blockchain.

- Vậy Move có gì đặc biệt? Khác với các ngôn ngữ lập trình khác, Move có các tính năng tích hợp giúp đảm bảo an toàn của smart contract. Move sử dụng hệ thống dựa trên tài nguyên để thiết lập sự sở hữu của dữ liệu và cho phép kiểm soát tài nguyên chi tiết. Điều này có nghĩa là các nhà phát triển có thể tạo ra các smart contract với sự tự tin, biết rằng chúng sẽ hoạt động một cách dự đoán và không có hậu quả không mong muốn.

- Nhưng đừng lo lắng, bạn không cần phải là chuyên gia blockchain để học Move! Trong khóa học này, chúng tôi sẽ bắt đầu từ những điều cơ bản và hướng dẫn bạn qua từng bước trong quá trình học tập. Chúng tôi sẽ đề cập đến mọi thứ bạn cần biết, từ biến và kiểu dữ liệu đến hàm và cấu trúc điều khiển. Chúng tôi cũng sẽ cung cấp cho bạn nhiều ví dụ và bài tập để giúp bạn có được kinh nghiệm thực tế.

- Khi chúng ta tiến bộ qua khóa học, chúng tôi sẽ giới thiệu bạn đến giao thức blockchain Aptos & Sui, nơi bạn sẽ đặt kỹ năng Move của mình vào thử thách bằng cách tạo ra các smart contract thực tế. Bạn sẽ học cách viết mã tương tác với blockchain, cách thiết kế smart contract an toàn và hiệu quả, và cách triển khai chúng lên blockchain Aptos & Sui.

- Khi kết thúc khóa học, bạn sẽ có đủ kiến thức và kinh nghiệm để bắt đầu phát triển các smart contract của riêng mình trên blockchain Aptos & Sui. Bạn sẽ biết cách sử dụng Move để tạo ra các smart contract an toàn và đáng tin cậy, và bạn sẽ hiểu rõ hơn về cách mà blockchain và smart contract hoạt động.

- Khóa học này không chỉ giúp bạn hiểu về Move và blockchain, mà còn giúp bạn trở thành một nhà phát triển blockchain chuyên nghiệp. Với khả năng tạo ra các smart contract mạnh mẽ và an toàn, bạn sẽ có thể thực hiện các ý tưởng và dự án của mình trên blockchain và tham gia vào cộng đồng blockchain đang phát triển nhanh chóng.
- Hãy cùng chúng tôi khám phá thế giới của Move và blockchain và trở thành một nhà phát triển blockchain chuyên nghiệp!

2 What Smart Contract Do?

- SmartContracts chỉ thực hiện 3 điều sau:
  - Định nghĩa một loại tài sản mới.
  - Đọc, Viết và chuyển tài sản.
  - Check được quyền sở hữu, quyền truy cập.
- Need languages support for:
  - Safe abstractions for custom assets, ownership, access control.
  - Khả năng an toàn khi dự án open-source.

3 Why use Move for SmartContract?

- `Chuyển tài sản dưới dạng 1 argument của một function, hoặc trả về một tài khoảan ở function đó.`
- Lưu trữ tài sản trong một cấu trúc dữ liệu. `struct`
- Có thể cho function mượn tài sản tạm thời để xử lý logic.
- Khởi tạo một tài sản ở Contract này nhưng có thể call và sử dụng ở contract khác.
- Lấy một tài sản bên ngoài của contract đó để khởi tạo.

4 Assets & Onwership

- if you `give` me a lot of tokens, i will `give` you a superCar.

```
fun buy(t: Token): SuperCar {...};
```

- If you `show` me your SuperCar & fee, i will `give` you a CarRegistration

```
fun register(s: &SuperCar, fee: Token): CarRegistration {...}
```

5 Types System

- Duplication

```
fun dup(t: Token) {
    let c = copy t; // error

    let y = &t;
    let copied = *y; // error
}
```

- Double-spending

```
fun double_spending(t: Token) {
    pay(move t);
    pay(move t); // error
}
```

- Destruction

```
fun destruction(t: Token) {
    t = ...; // error
}
```

6 Module

```
module token::Token {}
module nft::Marketplace {
    use token::Token;
}

module defi::DEX {
    use token::Token;
```

```
}

script {
    fun transfer_to(){}
    fun main() {
        // logic
        transfer_to();
    }
}
```

- Có thể tái sử dụng bằng cách import vào các module khác.
- Modules có thể chứa các struct types, functions, constants.

7 Data Types

- bool: true/false
- unsigned integer: u8, u16, u32, u64, u128, u256.
- asset owner/tx sender: address
- Strings: std::string::String (UTF-8), std::ascii::String (ASCII)
- vector<T>
- Table<K,V>

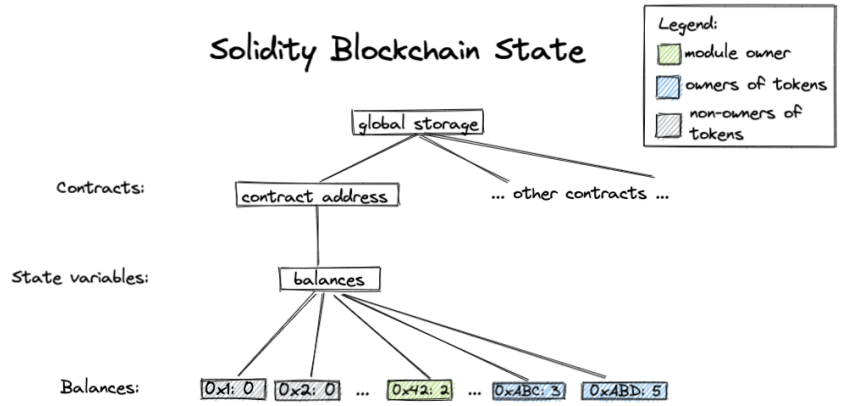
```
module 0xCAFE::MyModule {
    struct AnotherStruct {
        boolean: bool
    }

    struct MyStruct {
        num: u64,
        another_struct: AnotherStruct
    }
}
```

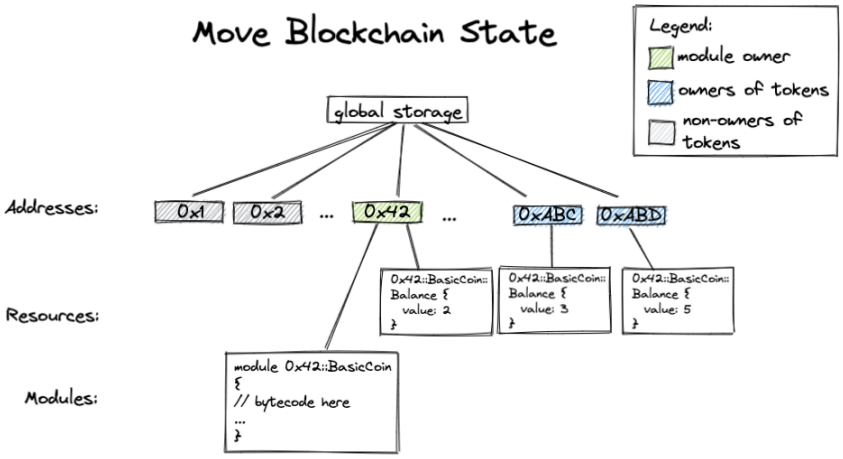
8 References & Ownership

- Rust-like
  - &T - Chỉ có thể đọc dữ liệu
  - &mut T - có thể đọc và ghi dữ liệu.

  - T - có thể đọc, ghi, chuyển dữ liệu.
- Structs và Data structures không được chứa tham chiếu (references).
  - No Lifetimes, no reborrowing (different with rust)
  - Much simpler borrow checker as a result
- No references to references (&&T)

9 What Different with Ethereum





10 Create a First Contract

```
move new <name>
```

```
module 0xCAFE::BasicToken {
    struct Token has key {
        value: u64,
    }

    public fun mint(account: signer, value: u64) {
        move_to(&account, Token { value })
    }
}
```

11 Abilities

Structures của Move có thể có nhiều khả năng khác nhau được mô tả sau:

- `copy`: Cho phép dữ liệu có khả năng copy.
- `drop`: Cho phép dữ liệu có khả năng popped/dropped
- `store`: Cho phép giá trị của kiểu dữ liệu đó có được lưu trữ và tồn tại trên global storage trong một struct.
- `key`: Cho chép kiểu có thể sử dụng như một key-value cho bộ nhớ toàn cầu.

12 Generics constain struct and function defs

```
module 0x1::MyModule {
    struct A<T: store> {t: T}

    fun copies<T: copy>(t: &T): T {
        *t
    }

    struct Coin<phantom T> { value: u64 }
}
```

13 Move Function visibility

```
// có thể call được từ một transaction và từ một module khác.
public entry fun a()

// chỉ có thể call từ một transaction hoặc từ chính module đó.
entry fun b()

// Private-only chỉ call được từ chính module đó.
fun c()

// có thể call bằng những modules khác.
public fun d()

// có thể call được từ module "friend" khác ở trong cùng một package.
public(friend) fun e()
// tx
public(script) fun f()
```

- public(friend)

14 Install

```
git clone https://github.com/move-language/move.git
```

```
cd move
./scripts/dev_setup.sh -ypt
```

```
source ~/.profile
```

```
cargo install --path language/tools/move-cli
```

```
move --help
```

> move-cli 0.1.0 Diem Association <opensource@diem.com> Move-CLI is the CLI that will be executed by the ‘move-cli‘ command The ‘cmd‘ argument is added here rather than in ‘Move‘ to make it easier for other crates to extend ‘move-cli‘
>
> USAGE: move [OPTIONS] <SUBCOMMAND>
>
> OPTIONS: –abi Generate ABIs for packages –arch <ARCHITECTURE> -d, –dev Compile in ’dev’ mode. The ’dev-addresses’ and ’dev-dependencies’ fields will be used if this flag is set. This

flag is useful for development of packages that expose named addresses that are not set to a specific value –doc Generate documentation for packages –fetch-deps-only Only fetch dependency repos to $\text{MOVE}_{\text{HOME}}$ –force Force recompilation of all packages -h, –help Print help information –install-dir <$\text{INSTALL}_{\text{DIR}}$> Installation directory for compiled artifacts. Defaults to current directory -p, –path <$\text{PACKAGE}_{\text{PATH}}$> Path to a package which the command should be run with respect to –skip-fetch-latest-git-deps Skip fetching latest git dependencies –test Compile in 'test' mode. The 'dev-addresses' and 'dev-dependencies' fields will be used along with any code in the 'tests' directory -v Print additional diagnostics if available -V, –version Print version information

SUBCOMMANDS: build Build the package at ‘path‘. If no path is provided defaults to current directory coverage Inspect test coverage for this package. A previous test run with the ‘–coverage‘ flag must have previously been run disassemble Disassemble the Move bytecode pointed to docgen Generate javadoc style documentation for Move packages errmap Generate error map for the package and its dependencies at ‘path‘ for use by the Move explanation tool experimental (Experimental) Run static analyses on Move source or bytecode help Print this message or the help of the given subcommand(s) info Print address information new Create a new Move package with name ‘name‘ at ‘path‘. If ‘path‘ is not provided the package will be created in the directory ‘name‘ prove Run the Move Prover on the package at ‘path‘. If no path is provided defaults to current directory. Use ‘.. prove .. – <options>‘ to pass on options to the prover sandbox Execute a sandbox command test Run Move unit tests in this package

15 Global Storage - Operators

Operation	Description
move$_{\text{to}}$<T>(&signer, T)	Publish T under signer.address
move$_{\text{from}}$<T>(address): T	Remove T from address and return it
borrow$_{\text{globalmut}}$<T>(address): &mut T	Return a mutable reference to the T stored under address
borrow$_{\text{global}}$<T>(address): &T	Return an immutable reference to the T stored under addr
exists<T>(address): bool	Return true if a T is stored under address

16 Testing the Contract

```
module 0xCAFE::BasicToken {
    #[only_test]
    use std::signer;

    struct Token has key {
        value: u64,
    }

    public fun mint(account: signer, value: u64) {
        move_to(&account, Token { value });
    }

    #[test(account = @0xC0FFEE)]
    fun test_mint(account: signer) acquires Token {
        let addr = signer::address_of(&account);
        mint(account, 10);
        assert!(borrow_global<Token>(addr).value == 10, 0);
    }
}
```