SuiMover

Move 語法簡介

Justa Liang - Move Developer in bucketprotocol.io

suimover.org

SuiMover

suimover.org

專案架構

專案架構

Move 專案的架構

```
package 0x...
   module a
      public struct A
      fun hello()
      public fun say_hello()
   module b
      public struct B
      fun sorry()
      public fun echo()
```

```
sources/
   a.move
   b.move
   ...
tests/
   ...
examples/
   using_my_module.move
Move.toml
Move.lock
```

Move 專案的 CLI 操作

創建 Move 專案

```
sui move new package_name
```

編譯 Move 專案

```
sui move build
```

執行單元測試

```
sui move test
```

部署合約

```
sui client publish
```

專案架構

Move 專案的設定檔 (Move.toml)

```
[package]
name = 專案名字
edition = "2024.beta"

[dependencies]
專案名1 = { local = 相對路徑 }
專案名2 = { git = github連結, subdir = 子路徑, rev = 版本號 }

[addresses]
專案別名 = “0x0”
```

SuiMover

suimover.org

基本語法

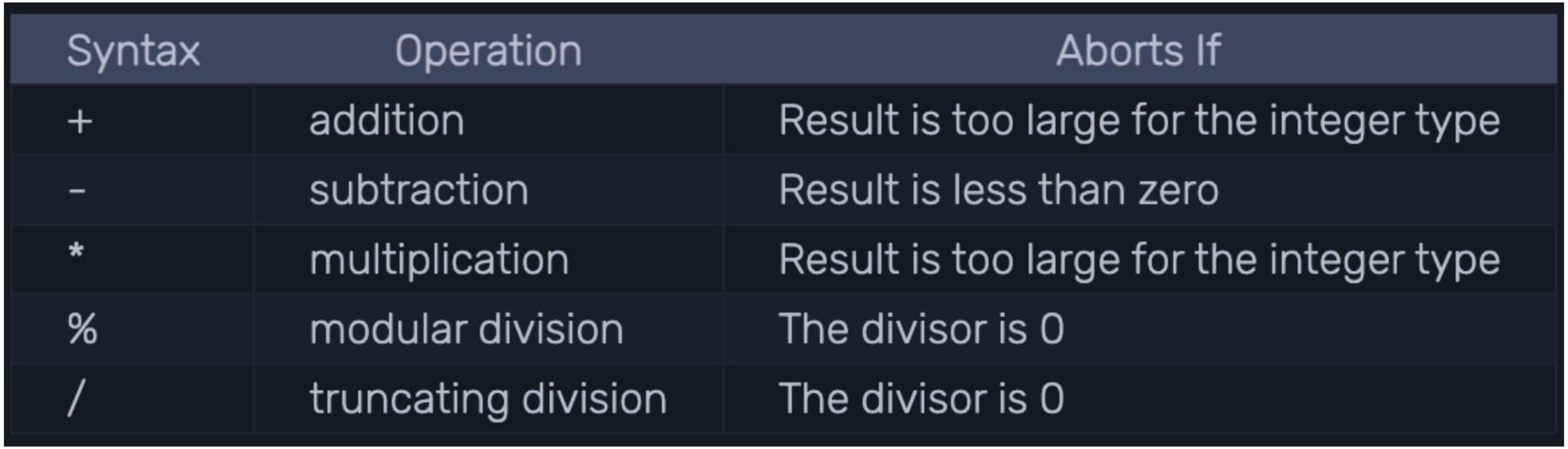
基本型別

- bool
- unsigned integers
  - u8, u16, u64, u128, u256
- address
- vector<T>

常見型別

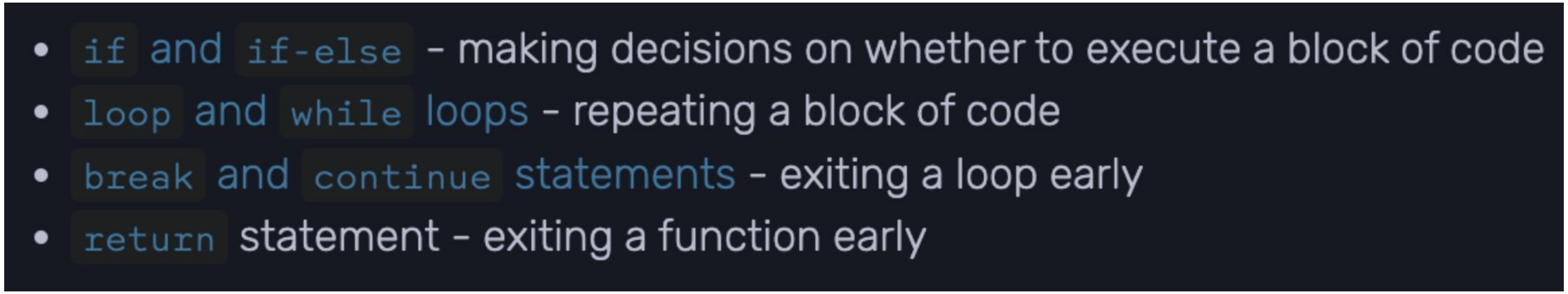
- ID: 用來表示Object 的 ID（可以與 address 做轉換）
- String: 字串
- Option<T>: 有兩種狀態 Some(T) or None
- Coin<T>: 同質化代幣（可被擁有）
- Balance<T>: 同質化代幣（不可被擁有）

運算符



Syntax	Operation	Aborts If
+	addition	Result is too large for the integer type
-	subtraction	Result is less than zero
*	multiplication	Result is too large for the integer type
%	modular division	The divisor is 0
/	truncating division	The divisor is 0

判斷式、迴圈、中斷



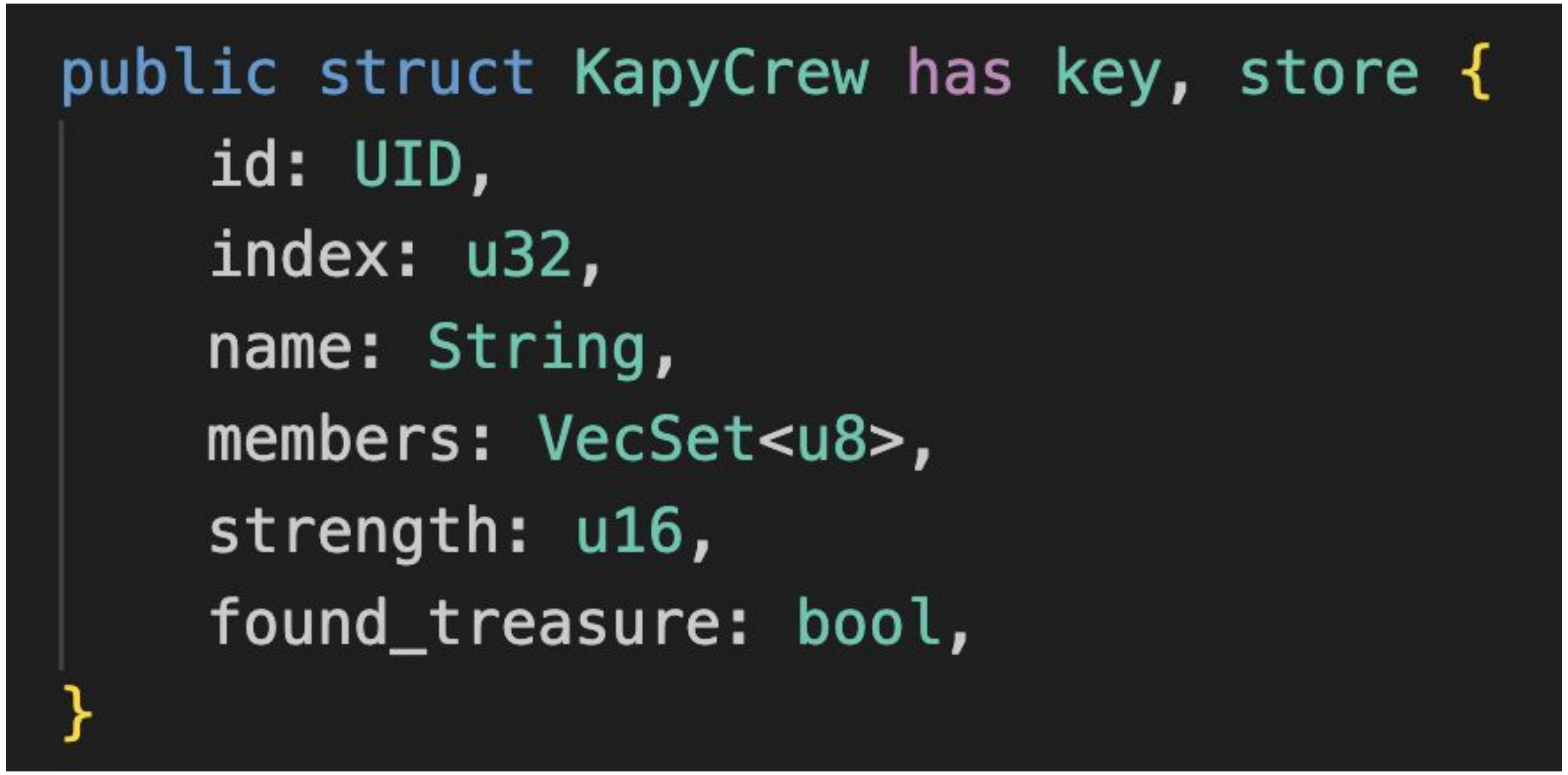
- `if` and `if-else` - making decisions on whether to execute a block of code
- `loop` and `while` loops - repeating a block of code
- `break` and `continue` statements - exiting a loop early
- `return` statement - exiting a function early

SuiMover

suimover.org

自定義型別與函式

型別（**type**）與行為（**ability**）



```
public struct KapyCrew has key, store {
    id: UID,
    index: u32,
    name: String,
    members: VecSet<u8>,
    strength: u16,
    found_treasure: bool,
}
```

行為分類

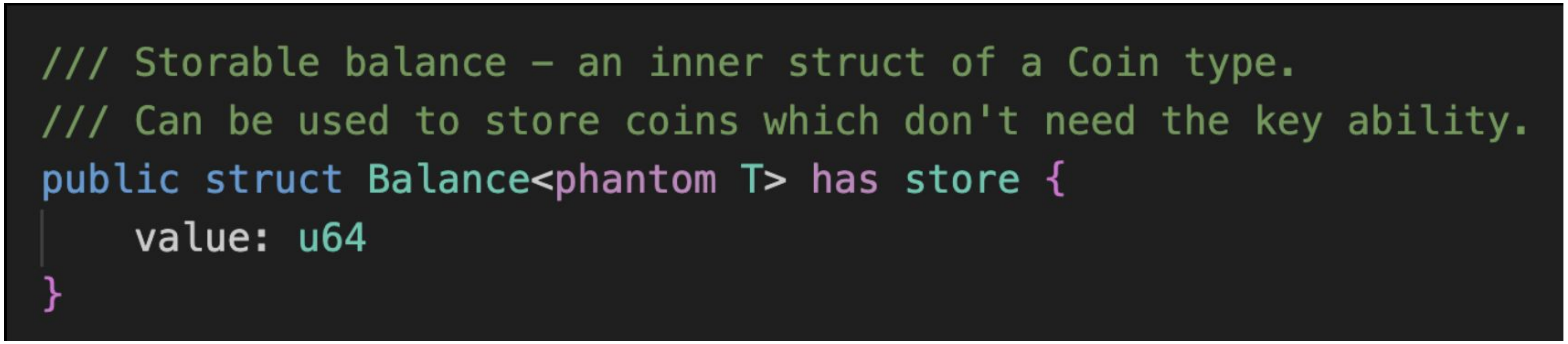
copy - 此型別可被複製

- drop - 此型別可被任意丟棄
- key - 此型別可被持有或分享
- store - 此型別可被儲存
- key (without store) - 此型別可被擁有或分享但不能被任意轉移
- key + store - 此行別可被擁有或分享且可被任意轉移

函式（**function**）與能見度（**visibility**）

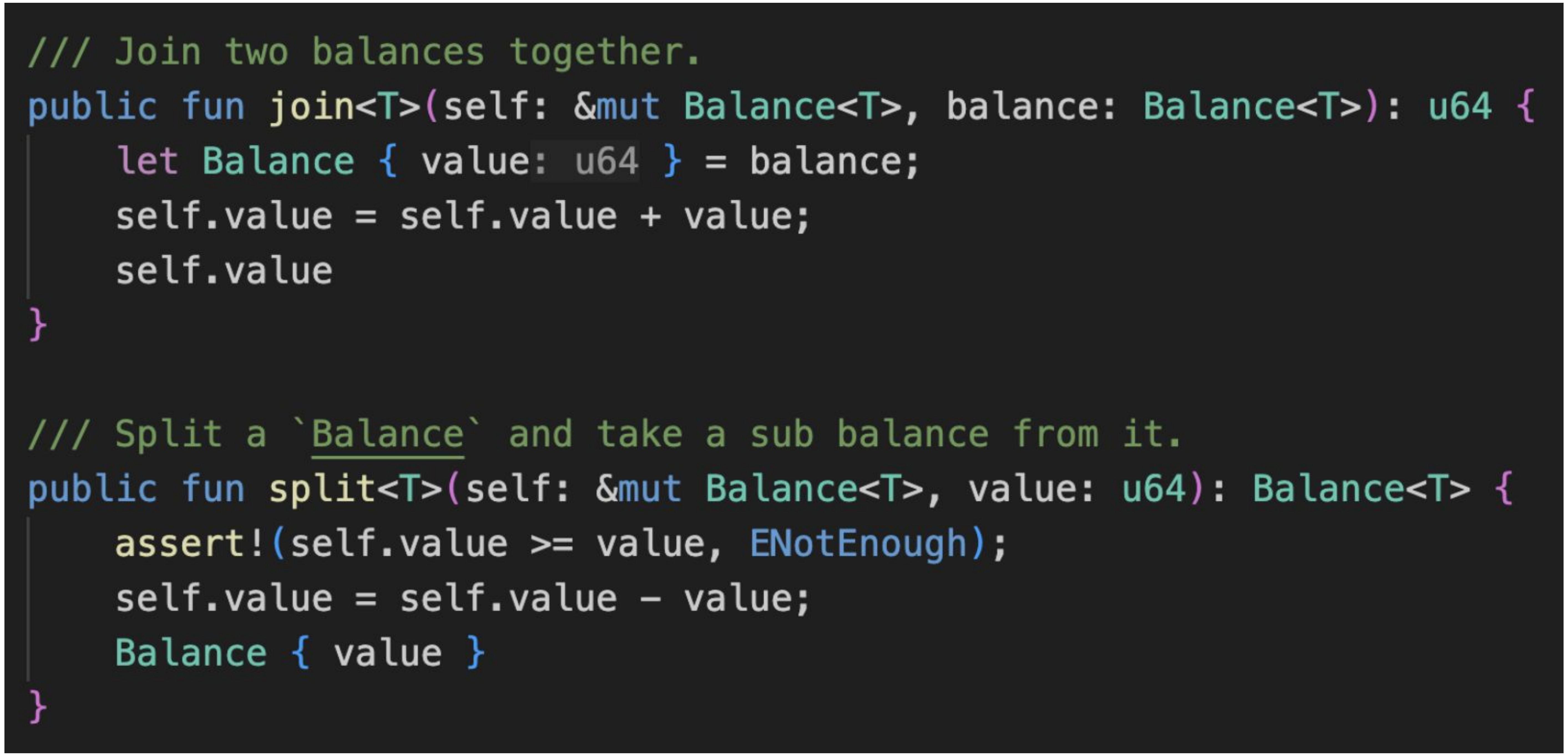
- fun - 只能在該模組內呼叫
- public fun - 可以被外部模組呼叫
- public(package) fun - 可以被同個套件下的模組呼叫
- entry fun - 可以被呼叫但不能包進 PTB

Balance（有 store 行為）



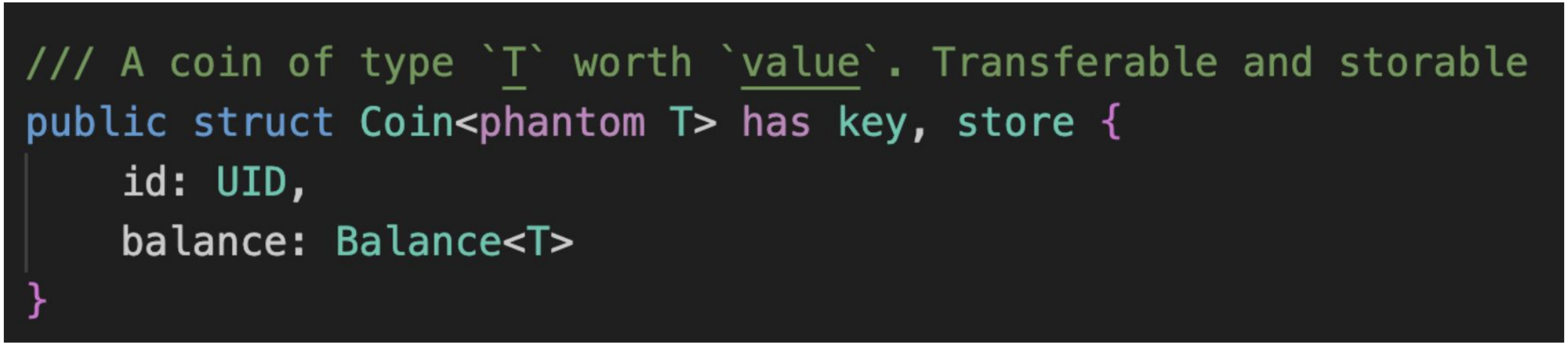
```
/// Storable balance - an inner struct of a Coin type.
/// Can be used to store coins which don't need the key ability.
public struct Balance<phantom T> has store {
    value: u64
}
```

Balance Methods



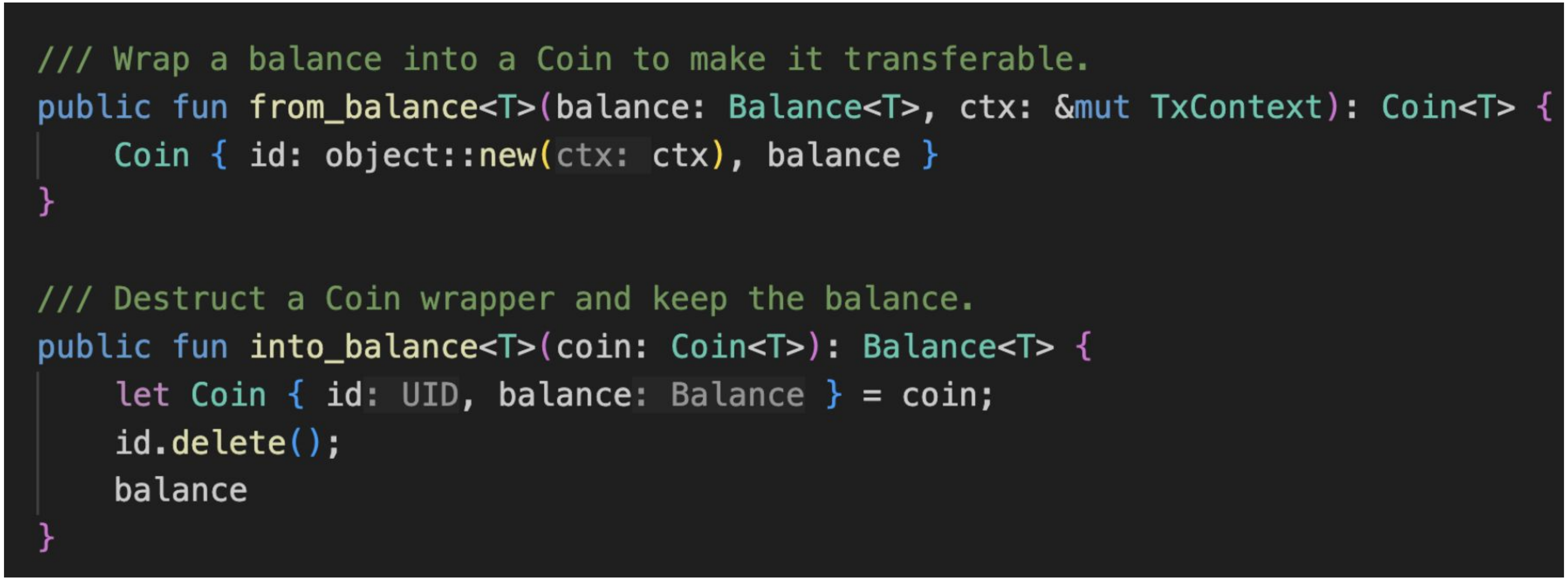
```
/// Join two balances together.
public fun join<T>(self: &mut Balance<T>, balance: Balance<T>): u64 {
    let Balance { value: u64 } = balance;
    self.value = self.value + value;
    self.value
}

/// Split a `Balance` and take a sub balance from it.
public fun split<T>(self: &mut Balance<T>, value: u64): Balance<T> {
    assert!(self.value >= value, ENotEnough);
    self.value = self.value - value;
    Balance { value }
}
```

Coin（有 key + store 行為）



```
/// A coin of type `T` worth `value`. Transferable and storable
public struct Coin<phantom T> has key, store {
    id: UID,
    balance: Balance<T>
}
```

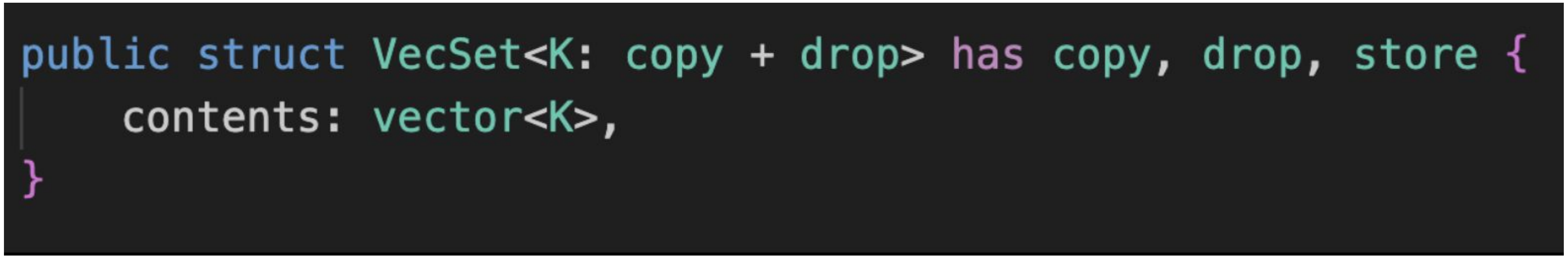
Coin Methods



```
/// Wrap a balance into a Coin to make it transferable.
public fun from_balance<T>(balance: Balance<T>, ctx: &mut TxContext): Coin<T> {
    Coin { id: object::new(ctx: ctx), balance }
}

/// Destruct a Coin wrapper and keep the balance.
public fun into_balance<T>(coin: Coin<T>): Balance<T> {
    let Coin { id: UID, balance: Balance } = coin;
    id.delete();
    balance
}
```

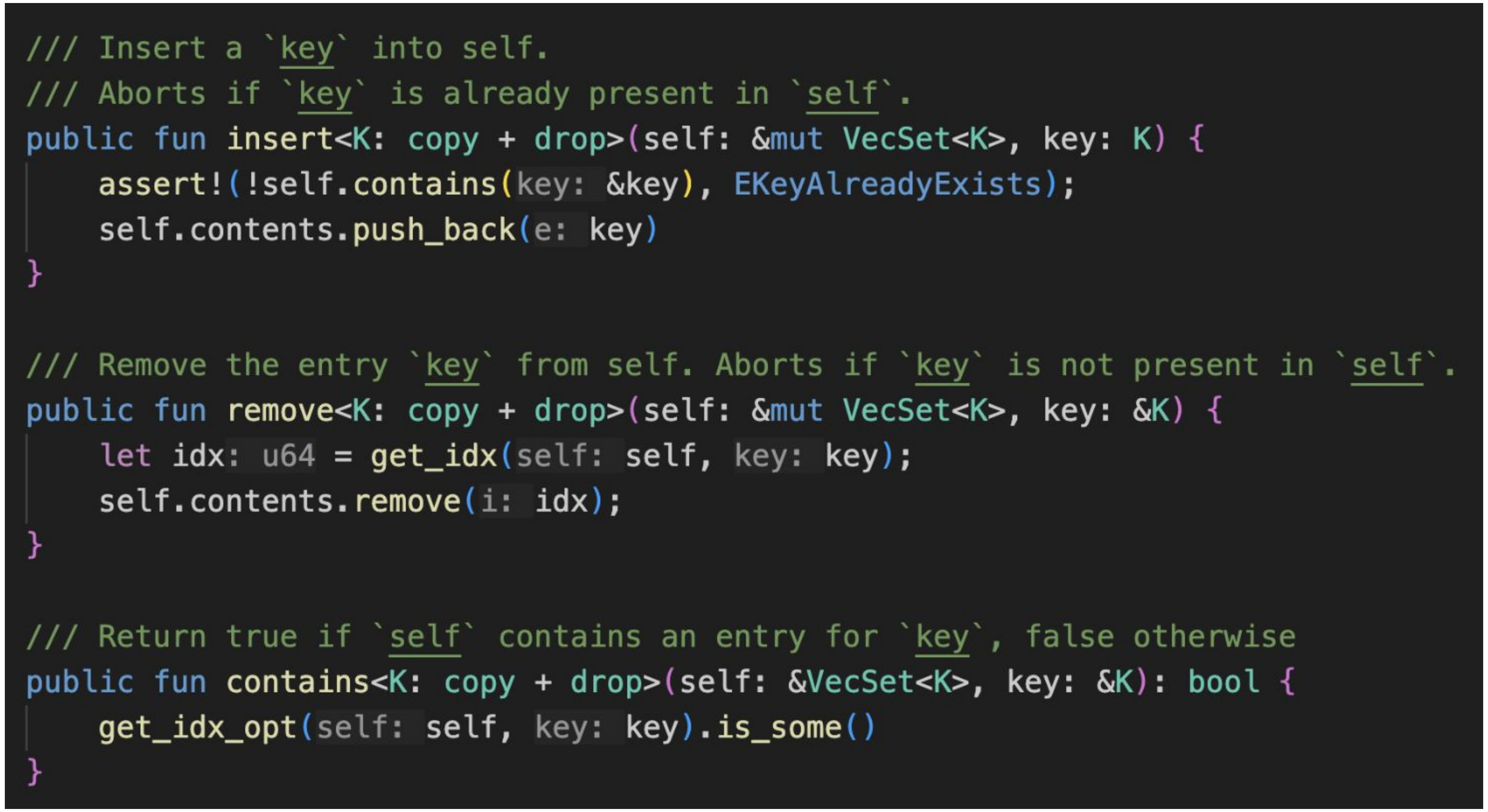
自定義型別與函式

VecSet



```
public struct VecSet<K: copy + drop> has copy, drop, store {
    contents: vector<K>,
}
```

VetSet Methods



```
/// Insert a `key` into self.
/// Aborts if `key` is already present in `self`.
public fun insert<K: copy + drop>(self: &mut VecSet<K>, key: K) {
    assert!(!self.contains(key: &key), EKeyAlreadyExists);
    self.contents.push_back(e: key)
}

/// Remove the entry `key` from self. Aborts if `key` is not present in `self`.
public fun remove<K: copy + drop>(self: &mut VecSet<K>, key: &K) {
    let idx: u64 = get_idx(self: self, key: key);
    self.contents.remove(i: idx);
}

/// Return true if `self` contains an entry for `key`, false otherwise
public fun contains<K: copy + drop>(self: &VecSet<K>, key: &K): bool {
    get_idx_opt(self: self, key: key).is_some()
}
```

傳入函式的方式

- T: 傳入實體，可能會在函式內銷毀
- &T: 唯讀，表示該函式只會讀取物件內容
- &mut T: 函式會更改物件內容但該物件依然存在

THANK YOU

suimover.org

End Slide