

# Move on Sui

Justa Liang - Move Developer in bucketprotocol.io

# 以 **Object** 為中心的架構

什麼是 Object？什麼是所有權？為何 Sui 與其他鏈如此不同？

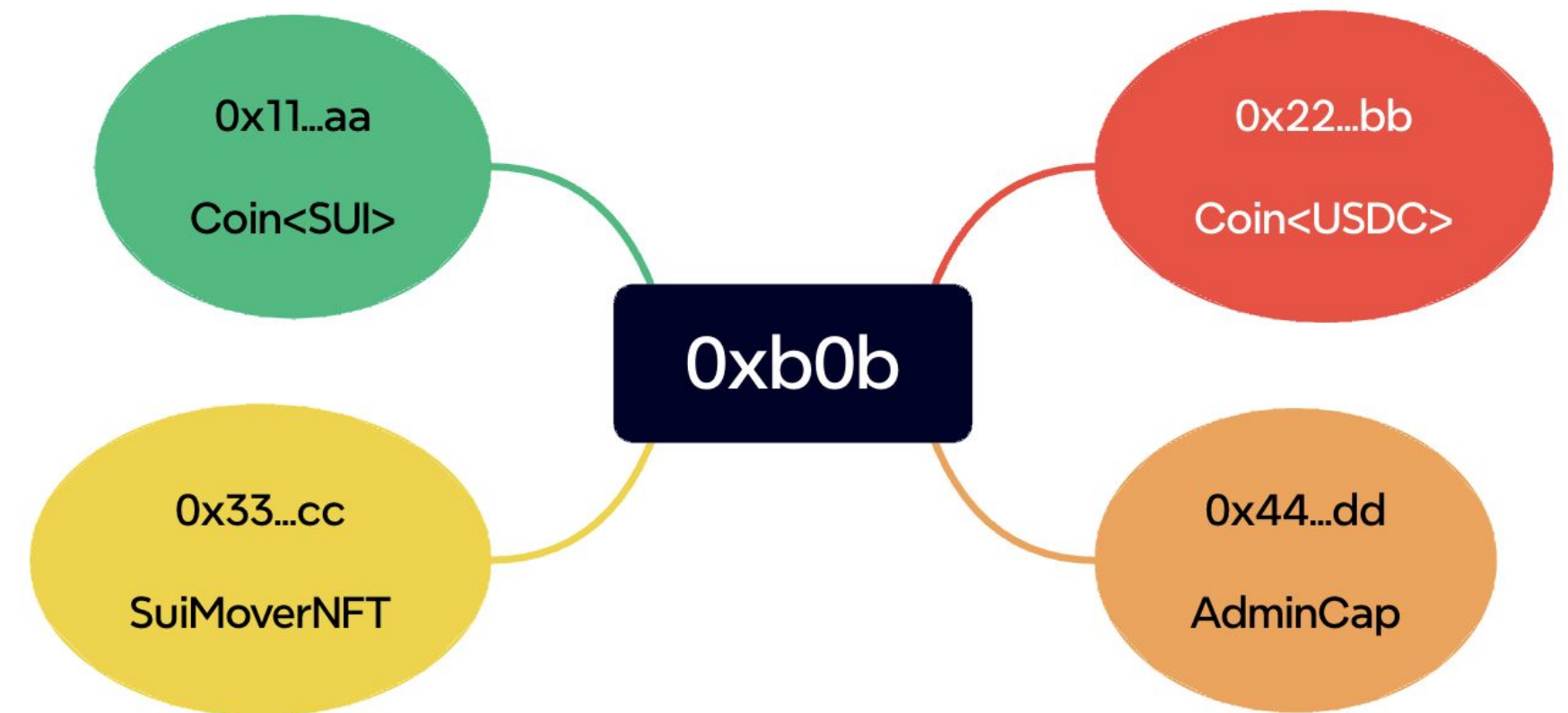
## 以 Object 為中心的架構

---

# 在 **Sui** 上 任何資產都是 **Object**

Object 的 Metadata 包含

1. 具唯一性的 ID (32-bytes)
2. 所有權 (32-bytes)
3. 型別 (package\_id::module\_name::TypeName)



## 以 Object 為中心的架構

## 用 Sui 的 Explorer 來查看

上 <https://suivision.xyz/>

查詢 sui-mover.sui

點 Portfolio 可以看 Coin 資產



**@sui-mover** sui-mover.sui

0xf905...f604 Sui Mover Admin

Portfolio \$159.16 Assets 18 items Transaction Blocks Activity

Coin Portfolio \$5.27

1 Verified 0 Unknown Hide Low Asset

#	NAME	PRICE	BALANCE	USD VALUE	OBJECT	ACTION
1	SUI   0x0...i::SUI	\$3.74 (-0.11%)	1.409093208 SUI	\$5.27	2	Swap
	SUI - 1		0.909093208 SUI	\$3.4	0x0750...edd6	
	SUI - 2		0.5 SUI	\$1.87	0x84a8...0475	

以 Object 為中心的架構

## 用 Sui 的 Explorer 來查看

上 <https://suivision.xyz/>

查詢 sui-mover.sui

點 Assets 可以看其他型別的資產

The screenshot shows the Sui Explorer interface for the user @sui-mover. The top navigation bar includes the user's profile, a search bar with 'sui-mover.sui', and tabs for Portfolio (\$159.51), Assets (18 items), Transaction Blocks, and Activity. The 'Assets' tab is active, showing a list of assets. The first asset is 'Sui Name Service' (1 item), and the second is 'Sui Mover' (1 item). The 'TYPE' and 'OBJECT ID' columns are highlighted with red boxes.

#	NAME	TYPE	OBJECT ID	LAST PRICE
1	@sui-mover	0xd22b...tion	0xda41...7a9e	-
1	Sui Mover: JustaLiang	0xcfb...Kapy	0x157a...1667	-

## 以 Object 為中心的架構

---

# 依 **Object** 所有權做分類

### Owned Object

1. 被某帳號所有
2. 只有所有者帳號能鏈上讀取或刪改

### Immutable Object

1. 不被任何帳號所有
2. 任何帳號都能讀取

### Shared Object

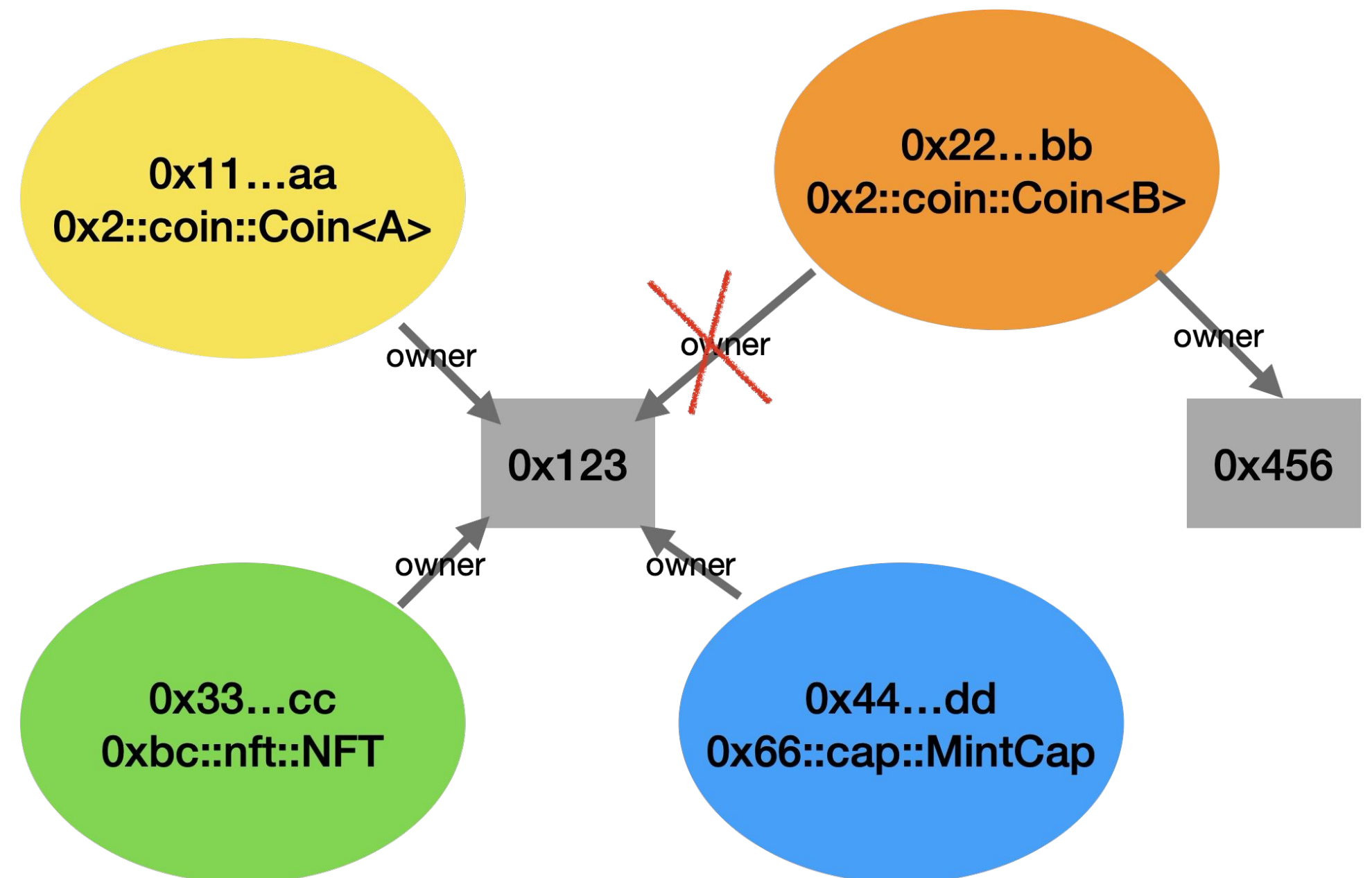
1. 不被任何帳號所有
2. 任何帳號都能鏈上讀取或根據型別規則去刪改

## 以 Object 為中心的架構

---

### 如何轉移資產？

1. 轉移資產 = 轉移 Object
2. 轉移的方式就是更改 Object 所有者
3. 只需更改一個記憶體空間
4. 資產轉移為底層邏輯，無需透過合約實作

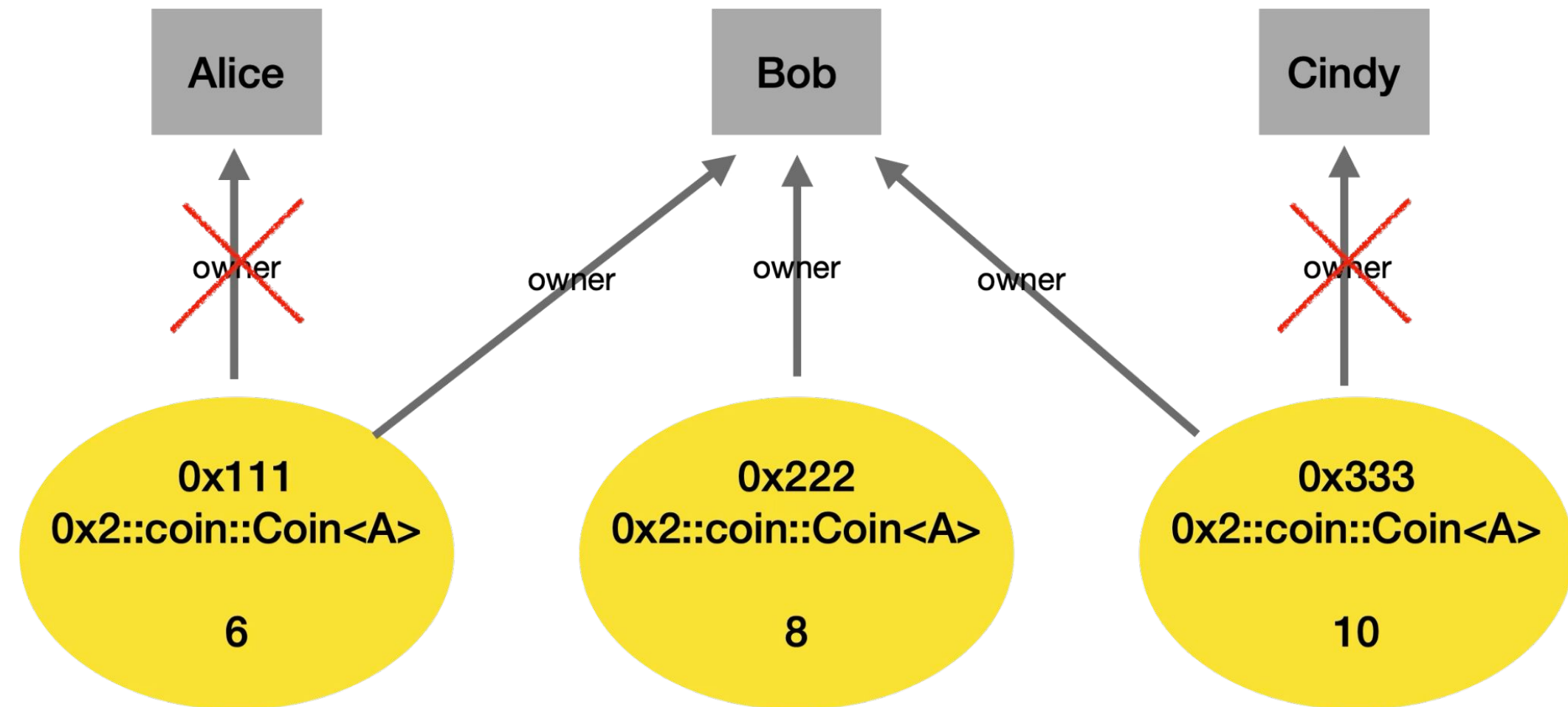


## 以 Object 為中心的架構

---

# 高度平行化

1. 轉移資產時沒有 Data Race 的問題
2. 無需排序的交易就無需經過共識層
3. 牽涉 Shared Object 的交易才需要排序
4. 簡單的交易不會造成雍塞

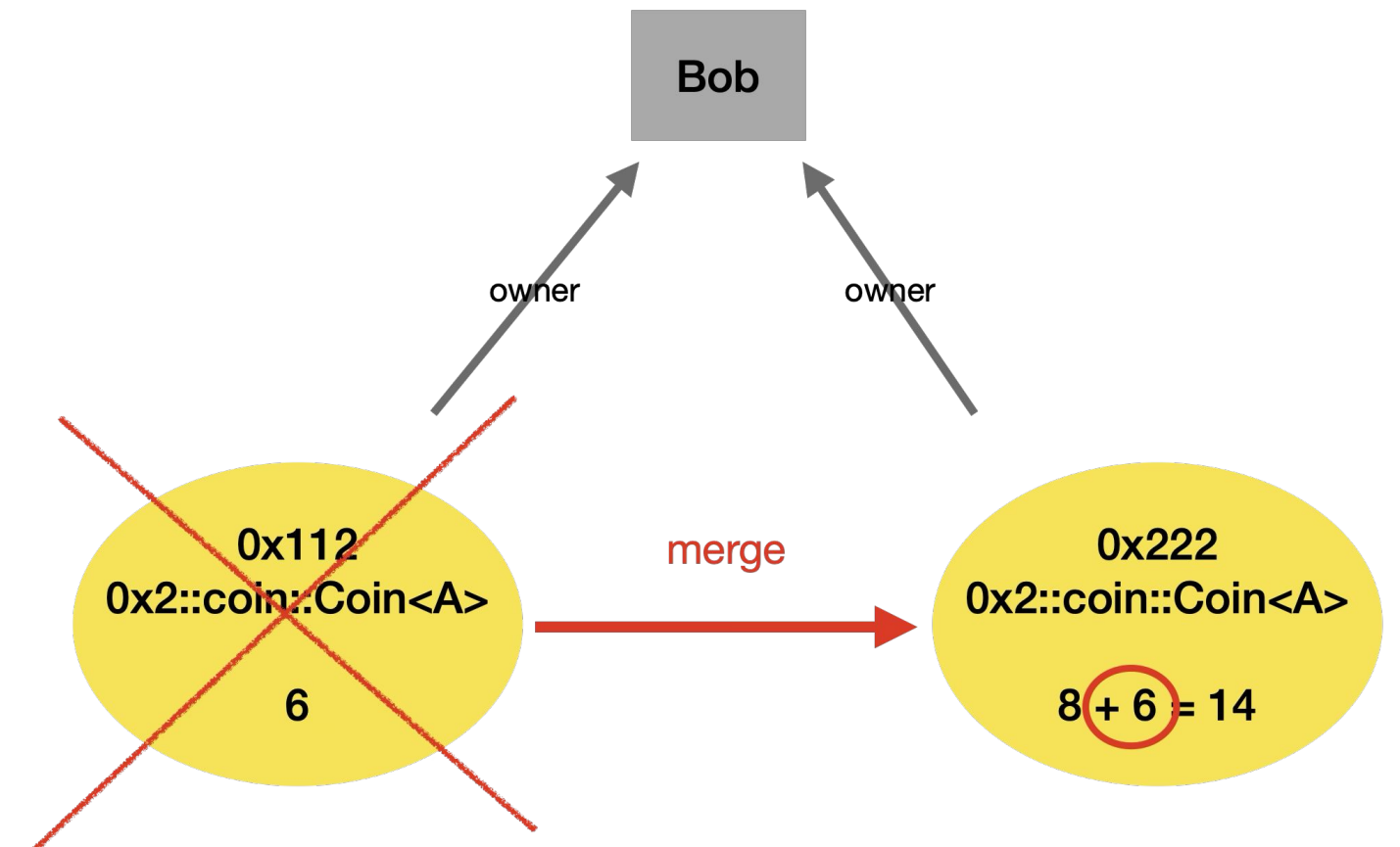
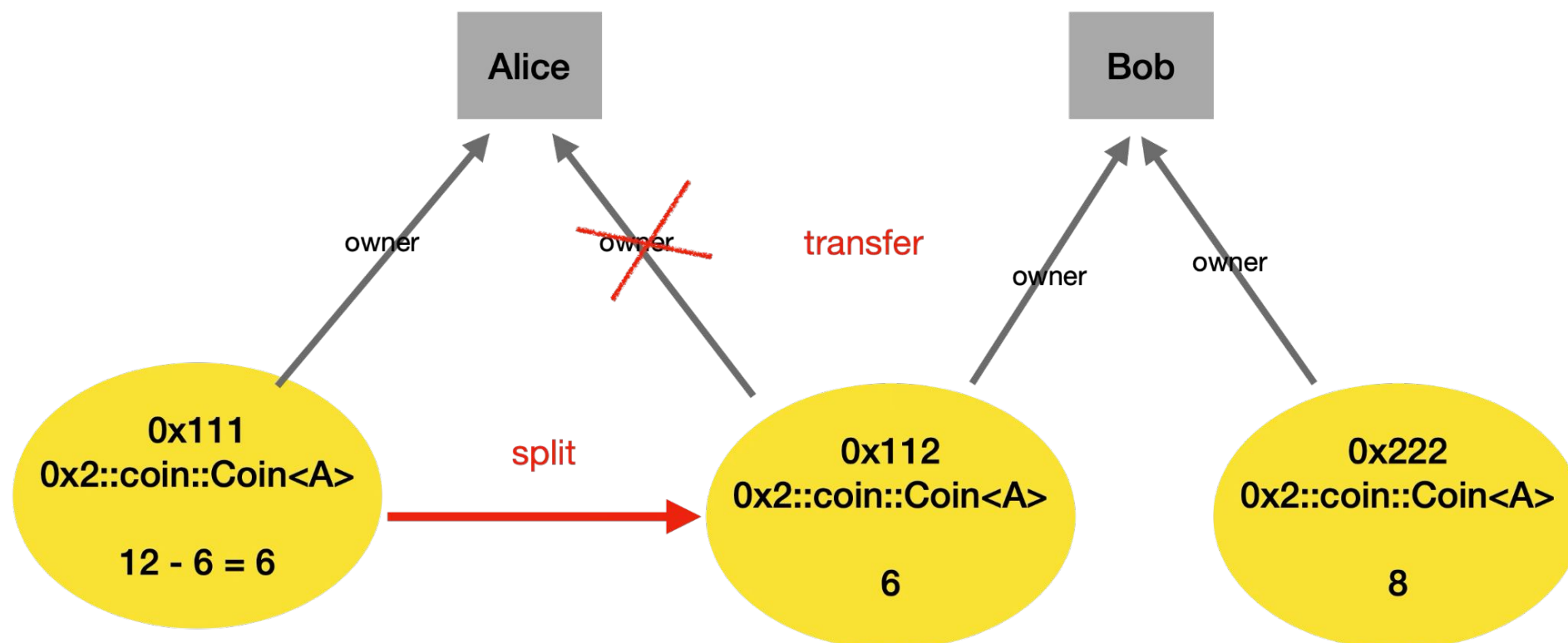




## 以 Object 為中心的架構

# 同質化代幣的轉移

1. 即使是同質化，也是以一個一個 Object 存在
2. 同質化體現在可以 Merge 和 Split

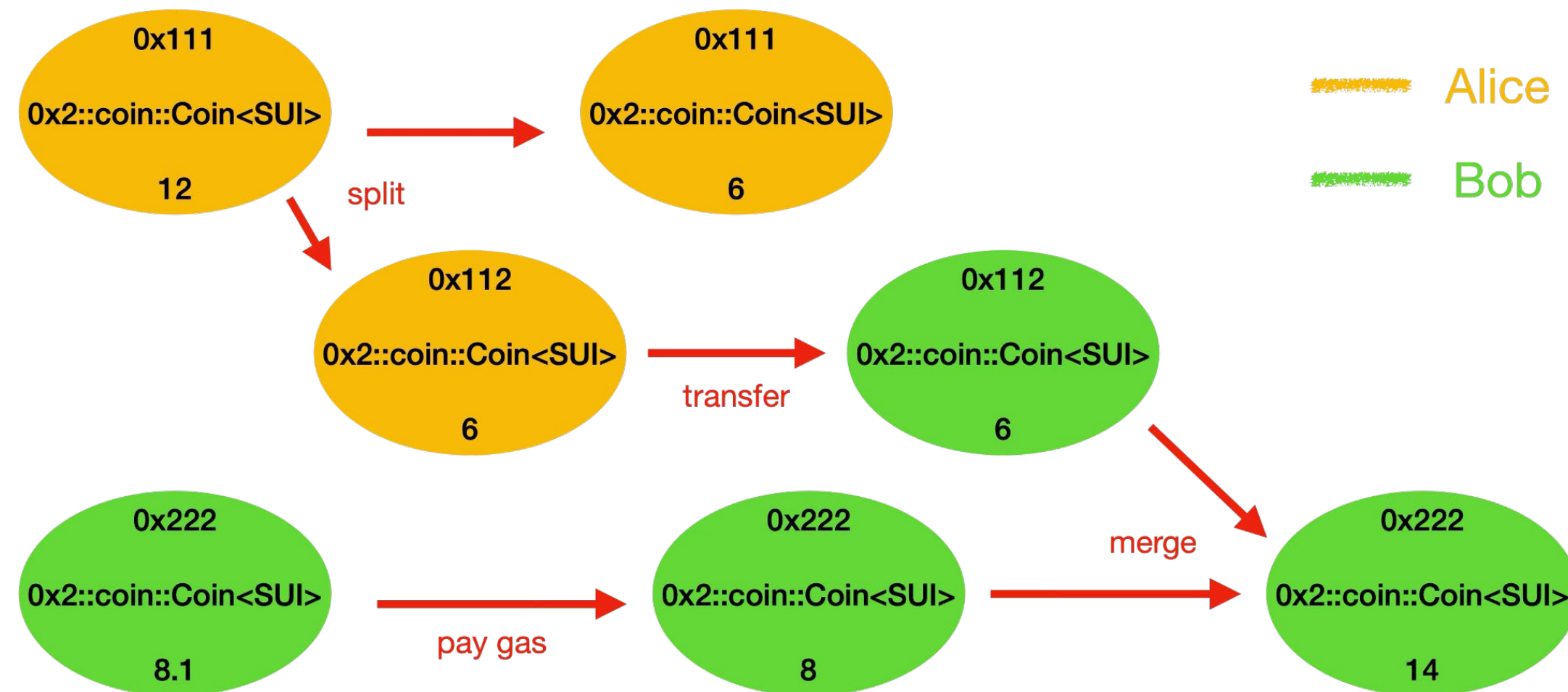


## 以 Object 為中心的架構

---

# Sui 本質是 DAG

嚴格來說 Sui 並非區塊鏈，而是 DAG (有向無環圖架構)



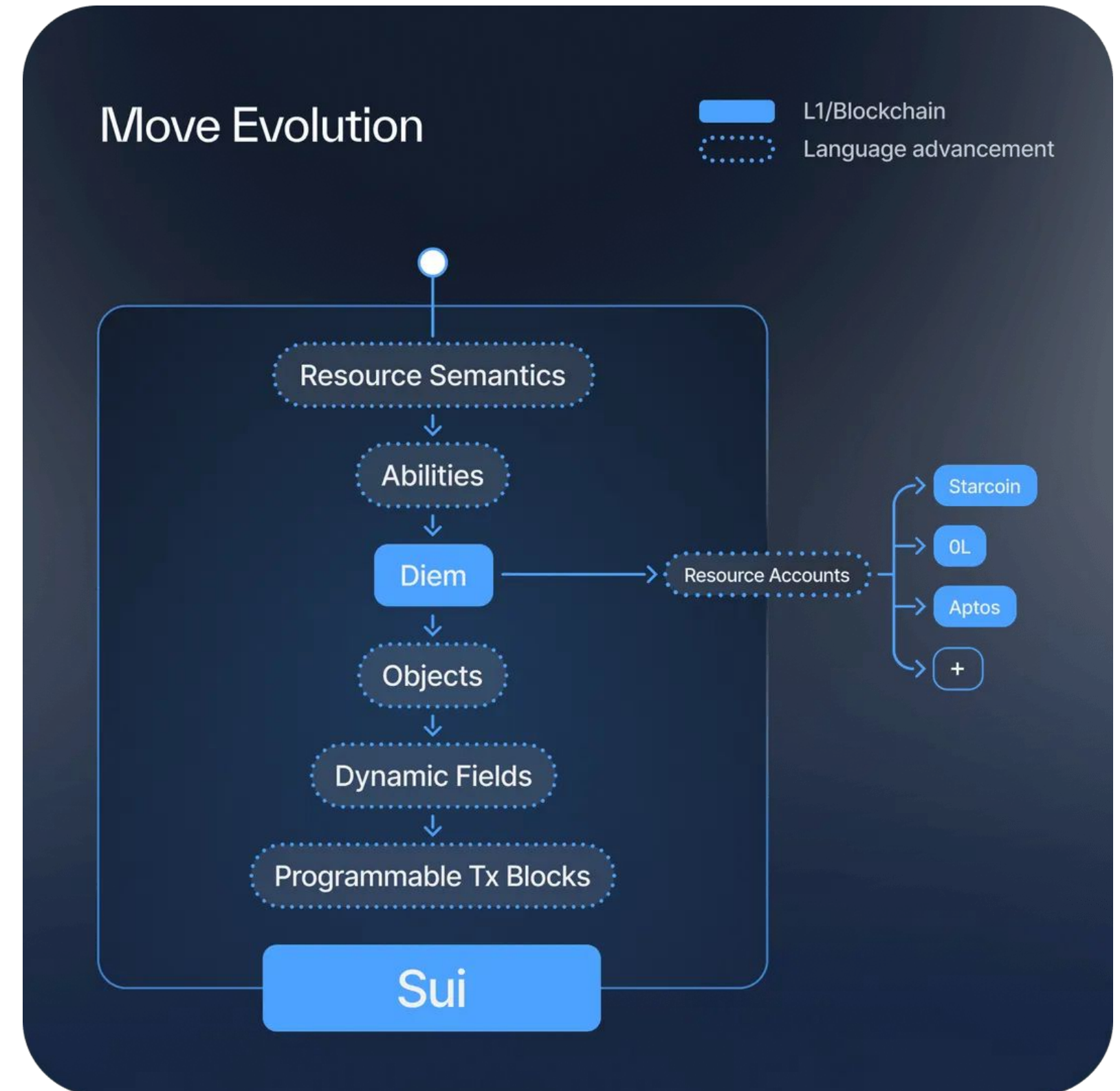
# Move 語言簡介

什麼是資產導向的程式語言？如何體現資產的安全性？

## Move 語言簡介

# 歷史發展與語言特性

1. Facebook 在開發 Diem (Libra) 時所發明的程式語言
2. 對資產管理友好的特性
  - a. 強型別 - 資產被封裝成型別而非單純的數字
  - b. 模組化 - 只有定義資產的模組能觸及資產內部資料
  - c. 隔離性 - 外部模組只能透過公開操作去處理資產
  - d. 組合性 - 資產能被封裝在另一個資產內
3. Sui 根據 Diem 版本的 Move 再結合 Object-centric 的架構, 推出特殊版本的 Move on Sui



## Move 語言簡介

---

# 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
```



# 定義型別及操作方法

1. public struct 可以定義型別
2. use 可以導入其他 package 的型別與方法
3. has 可以賦予型別特殊行為
  - a. key - 為一級資產, 即可成為 Object
  - b. store - 為次級資產, 能被封裝在 Object 內
  - c. copy - 能被複製
  - d. drop - 能隨意丟棄
  - e. key + store - 為一級資產, 且可以被封裝與轉移所有權
4. {} 內部可放入 fields
5. fun 可定義內部操作方法
6. public fun 可定義公開操作方法

```
1  module example_1::a;
2
3  use std::string::{Self, String};
4
5  public struct A has store, copy, drop {
6      |   last_words: String,
7      |
8  }
9
10 public fun say(a: &mut A, words: String) {
11     |   a.last_words = words;
12 }
13
14 public fun say_hello(a: &mut A) {
15     |   a.say(hello())
16 }
17
18 public fun last_words(a: &A): String {
19     |   a.last_words
20 }
21
22 fun hello(): String {
23     |   string::utf8(b"hello")
24 }
```

Sui 的強大來自於

**Object + DAG + Move**

# Sui Wallet 操作



# Sui Wallet 操作

# 官方錢包

 chrome 線上應用程式商店

 搜尋擴充功能和主題

探索

擴充功能

主題



Sui Wallet

4.9 ★ (5,424 個評分)

擴充功能

開發人員工具

1,000,000 使用者

 Mysten Labs

 Sui Wallet

zkLogin lets you easily create a self-custody wallet with just your SSO, including Google and Twitch.

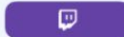
 Sui.

Welcome to Sui Wallet

Connecting you to the Sui network and the decentralized web.



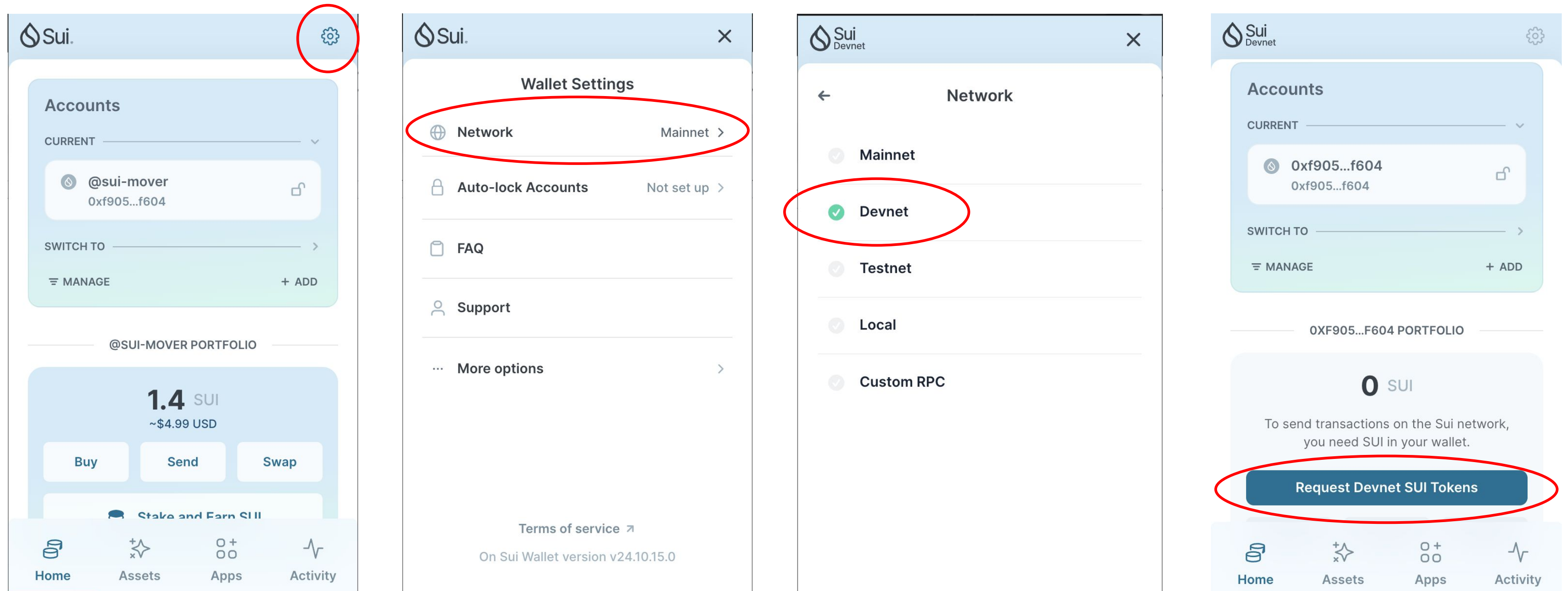
Sign In with your preferred service



More Options

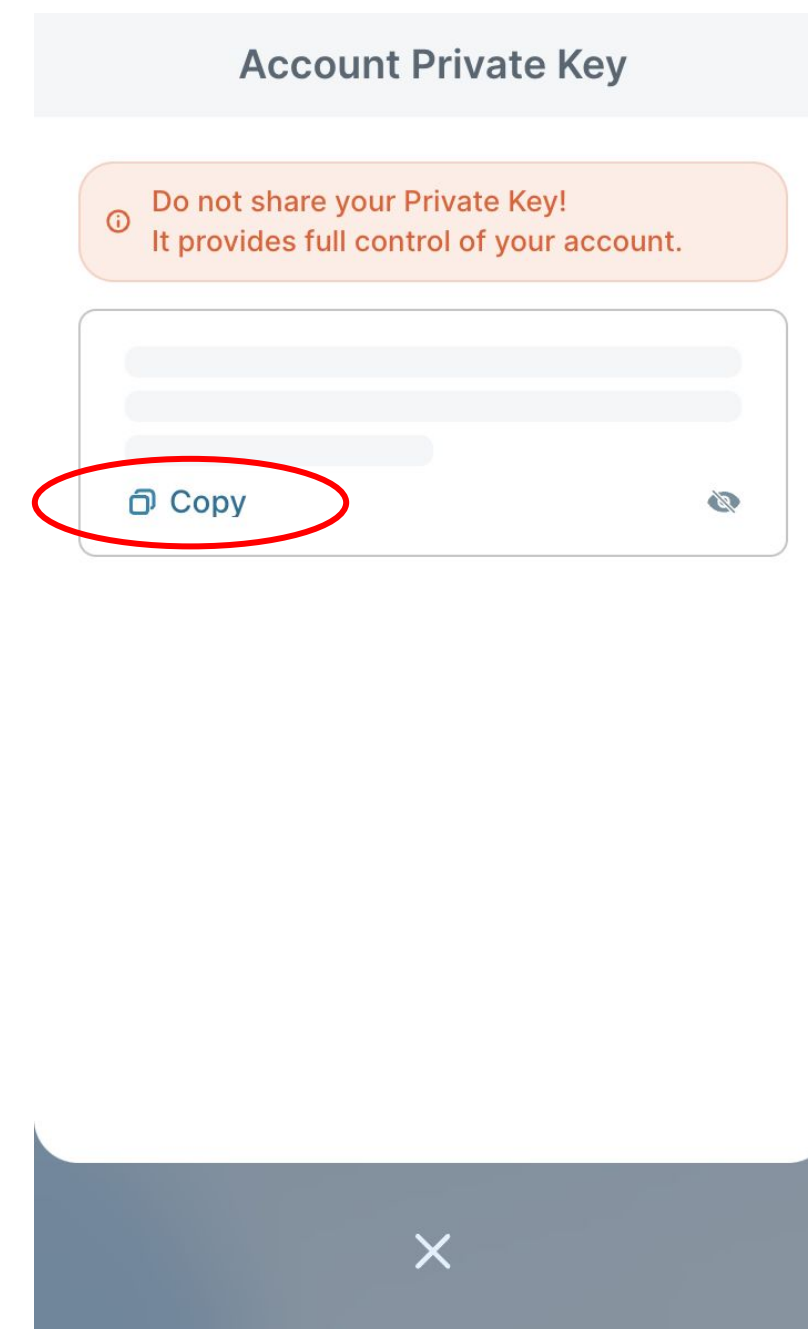
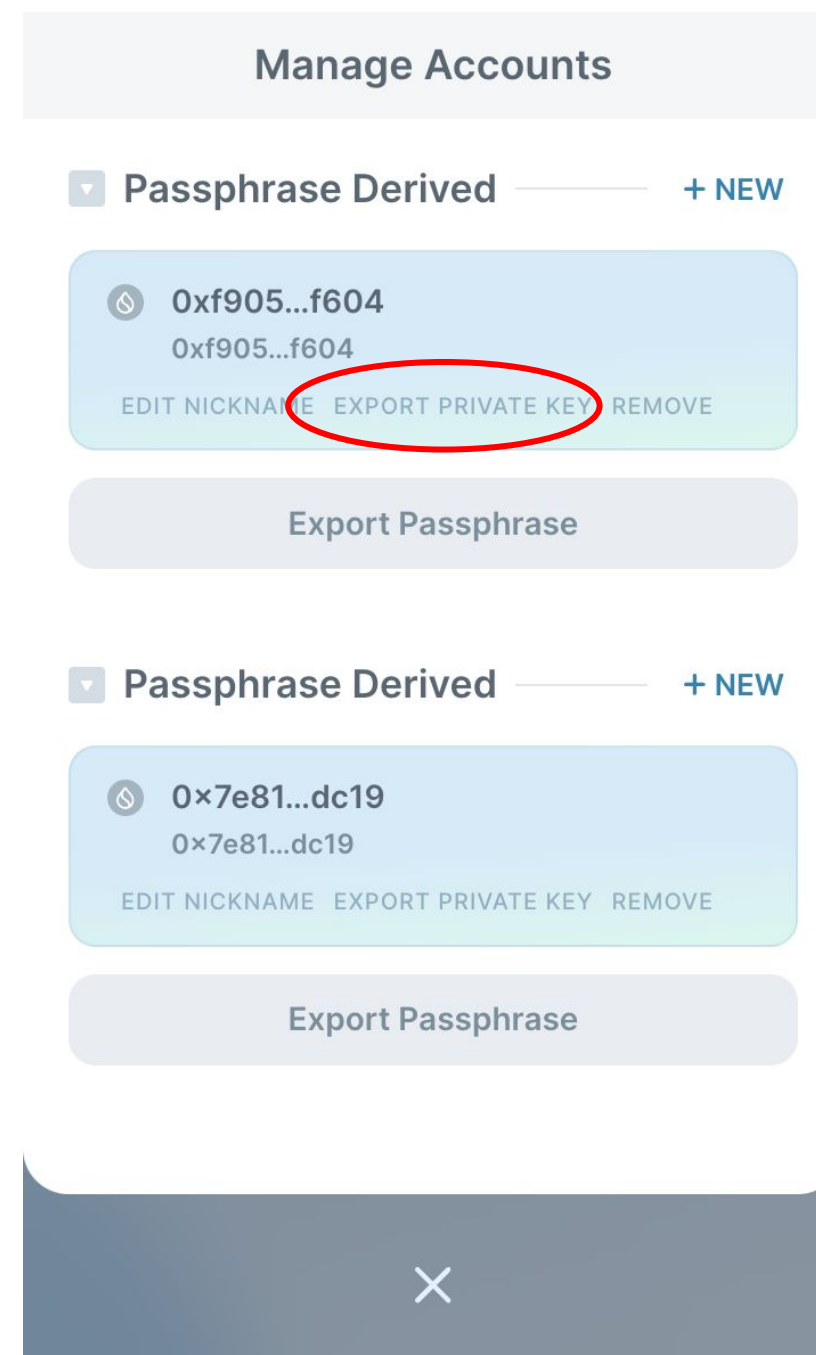
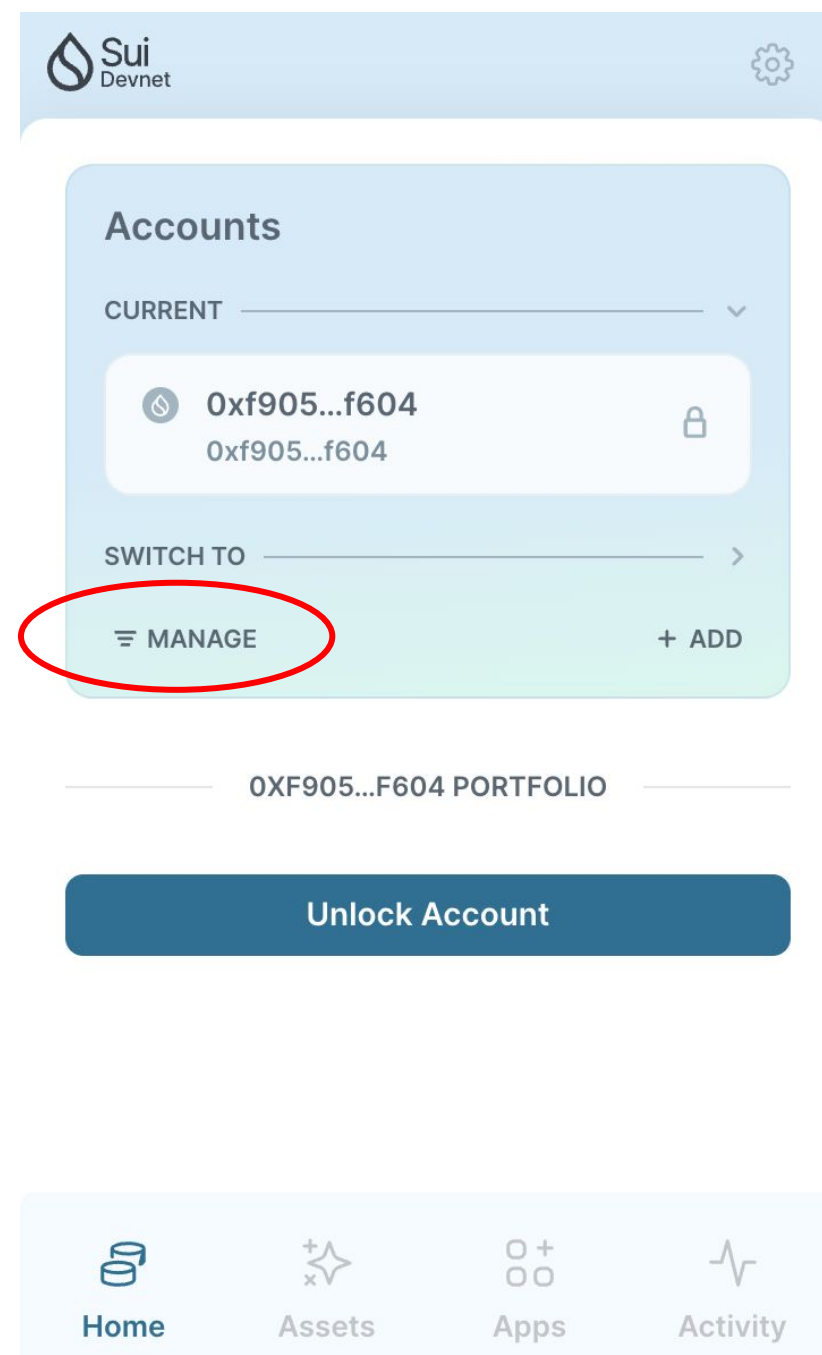
## Sui Wallet 操作

### 在開發網或測試網領水 (雙關?)



# Sui Wallet 操作

## 輸出私鑰



# Sui CLI 操作

## Sui CLI 操作

---

# 初步設置

查看版本

```
sui -V
```

輸入私鑰

```
sui keytool import private_key
```

創建帳號

```
sui keytool generate ed25519
```

查看帳號

```
sui client addresses
```

建立網路

```
sui client new-env --alias devnet --rpc https://fullnode.devnet.sui.io:443
```

切換帳號

```
sui client switch --address address --env alias
```

## Sui CLI 操作

---

# Coin 操作

### 領水

```
sui client faucet
```

### 查看Sui Coins

```
sui client gas
```

### 查看某個 Object

```
sui client object object_id
```

### Split

```
sui client split-coin --coin-id coin_id --amounts [n1 n2 ...]
```

### Merge

```
sui client merge --primary-coin coin_id1 --coin-to-merge coin_id2
```

## Sui CLI 操作

---

# 專案操作

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
```

## Sui CLI 操作

---

## 調用合約

sui client call --package package\_id --module module\_name --function function\_name  
--type-args [t1 t2 ...] --args [arg1 arg2 ...]

ex: 嘗試調用以下的合約方法

```
module sui::pay;

use sui::coin::Coin;

/// Split coin `self` to two coins, one with balance `split_amount`,
/// and the remaining balance is left is `self`.
public entry fun split<T>(
  coin: &mut Coin<T>,
  split_amount: u64,
  ctx: &mut TxContext
) {
  keep(coin.split(split_amount, ctx), ctx)
}
```



**THANK YOU**