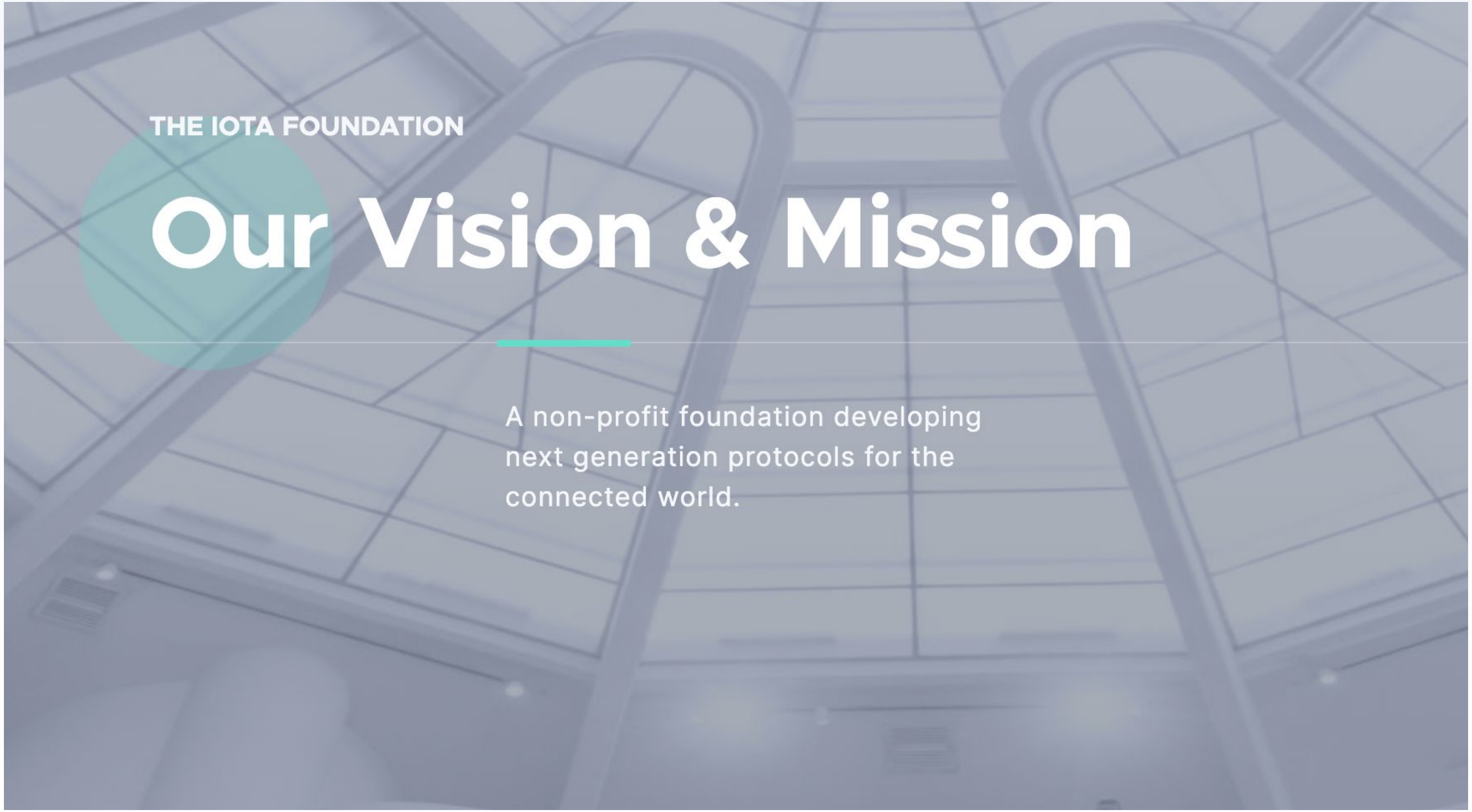
Move

the Language for Secure Next Gen Smart Contracts

Mirko Zichichi
mirko.zichichi@iota.org

IOTA

IOTA Foundation

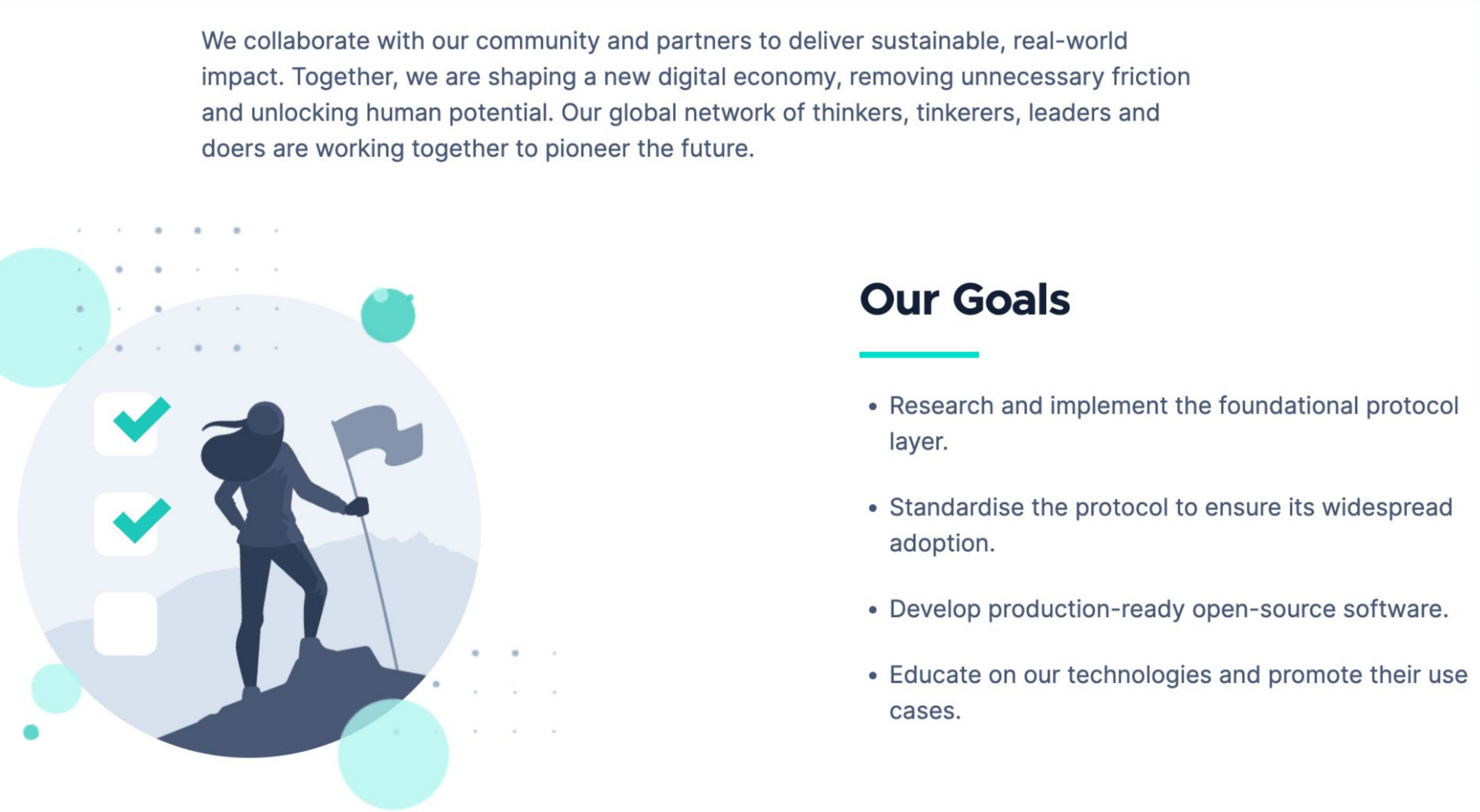


THE IOTA FOUNDATION

Our Vision & Mission

A non-profit foundation developing next generation protocols for the connected world.

IOTA Foundation



We collaborate with our community and partners to deliver sustainable, real-world impact. Together, we are shaping a new digital economy, removing unnecessary friction and unlocking human potential. Our global network of thinkers, tinkerers, leaders and doers are working together to pioneer the future.

Our Goals

- Research and implement the foundational protocol layer.
- Standardise the protocol to ensure its widespread adoption.
- Develop production-ready open-source software.
- Educate on our technologies and promote their use cases.

European blockchain regulatory sandbox for DLTs

European blockchain regulatory sandbox
for Distributed Ledger Technologies

“The sandbox establishes a pan-European framework for regulatory dialogues to increase legal certainty for innovative blockchain technology solutions […] across industry sectors such as energy & utilities, education, healthcare, mobility, finance & insurance, and logistics & supply chains.”

Web3 Identification Solution – A Decentralised and Secure Approach to User Authentication

The Web3 Identification Solution caters to the regulatory needs of Web3 and DeFi projects and enables them to interact seamlessly with verified users while excluding unverified addresses.

IOTA

IDnow.

walt.id

Bloom

HAVN

IOTA Stiftung (IDnow GmbH, walt.id GmbH, Bloom Labs Limited, Spyce5 GMBH)

https://ec.europa.eu/digital-building-blocks/sites/display/EBSISANDCOLLAB/European+Blockchain+Sandbox+announces+the+selected+projects+for+the+second+cohort#EuropeanBlockchainSandboxannouncestheselectedprojectsforthesecondcohort-IOTAStiftung

Move

the Language for Secure Next Gen Smart Contracts

Move

Resource

Move

Resource

Flexibility

Move

Resource

Flexibility

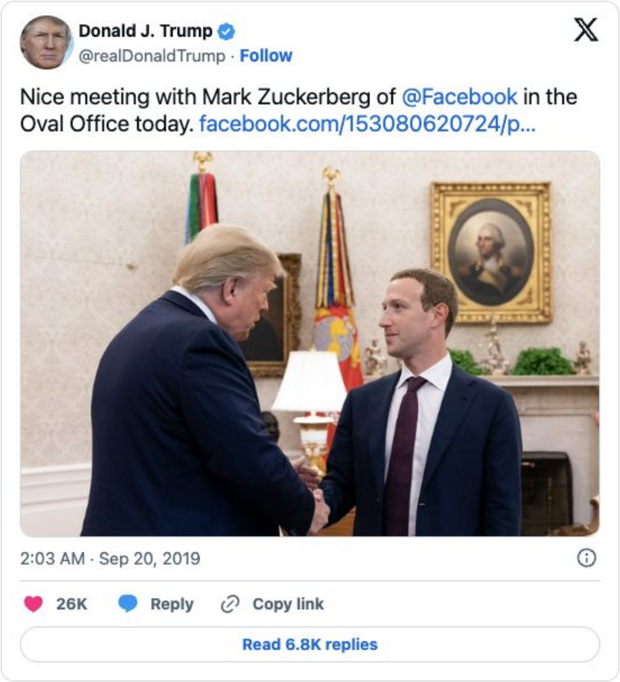
Security

Resource

Libra → Diem →

https://www.theverge.com/2019/6/26/18716326/facebook-libra-cryptocurrency-blockchain-irs-starbucks

Libra → Diem →


Donald J. Trump
@realDonaldTrump · Follow

Nice meeting with Mark Zuckerberg of @Facebook in the Oval Office today. facebook.com/153080620724/p...

2:03 AM · Sep 20, 2019

https://www.theverge.com/2019/6/26/18716326/facebook-libra-cryptocurrency-blockchain-irs-starbucks

*"Hey, **Libra** will have **smart contracts**, it is important to ensure that their programming on the blockchain is secure"*

Some Facebook chief, circa 2017.

Blackshear, Sam, et al. "Move: A Language With Programmable Resources"
https://diem-developers-components.netlify.app/papers/diem-move-a-language-with-programmable-resources/2020-05-26.pdf (2020).

"***The scarcest resource in the world is not time or money, but man's brain power.***

When these are used to develop software, if you can amplify brain capacity, i.e. do more per unit of time, this is one of the most impactful things you can achieve."

Blackshear, Sam, et al. "Move: A Language With Programmable Resources"
https://diem-developers-components.netlify.app/papers/diem-move-a-language-with-programmable-resources/2020-05-26.pdf (2020).

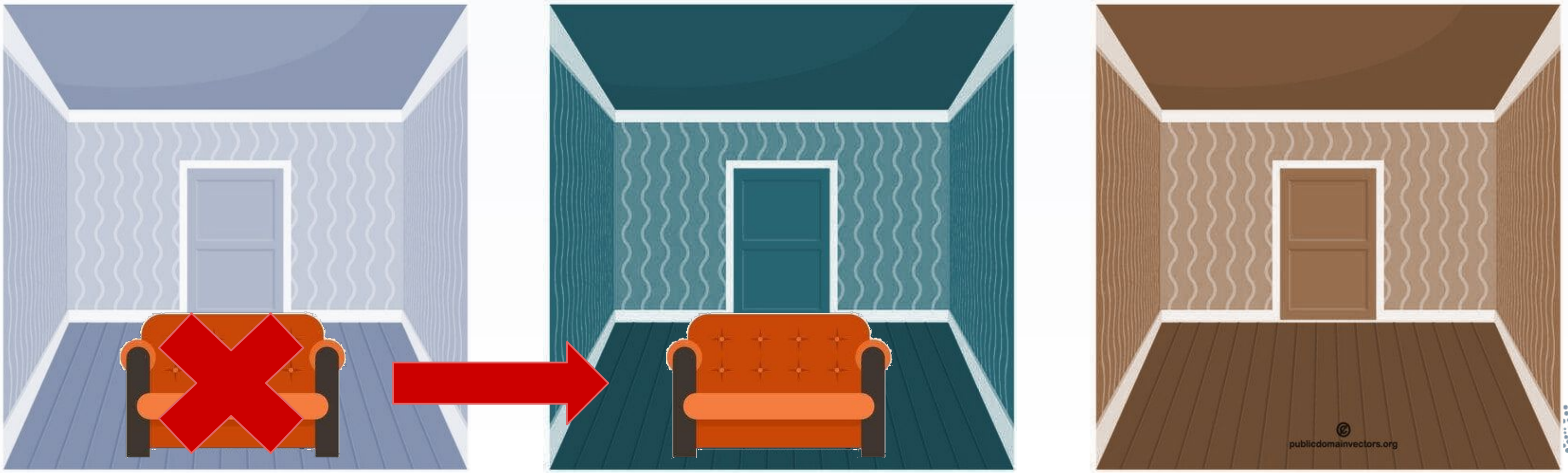
Move → Resource-oriented programming

It allows you to write software that behaves in line with *your physical intuition.*

publicdomainvectors.org
publicdomainvectors.org

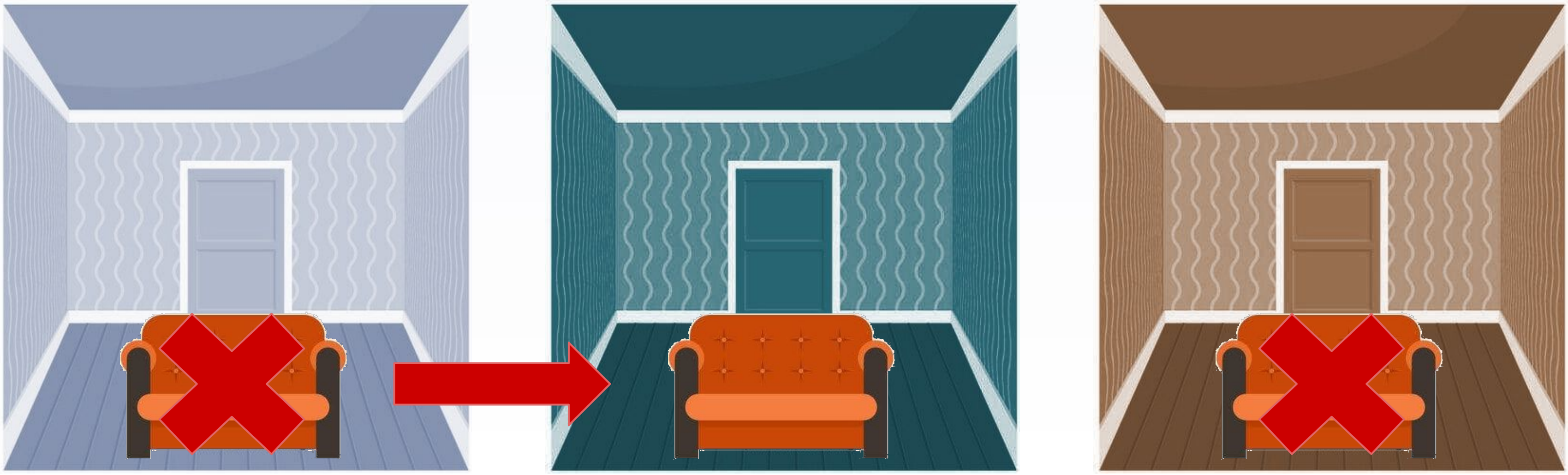
Move → Resource-oriented programming

It allows you to write software that behaves in line with *your physical intuition.*



Move → Resource-oriented programming

It allows you to write software that behaves in line with *your physical intuition.*



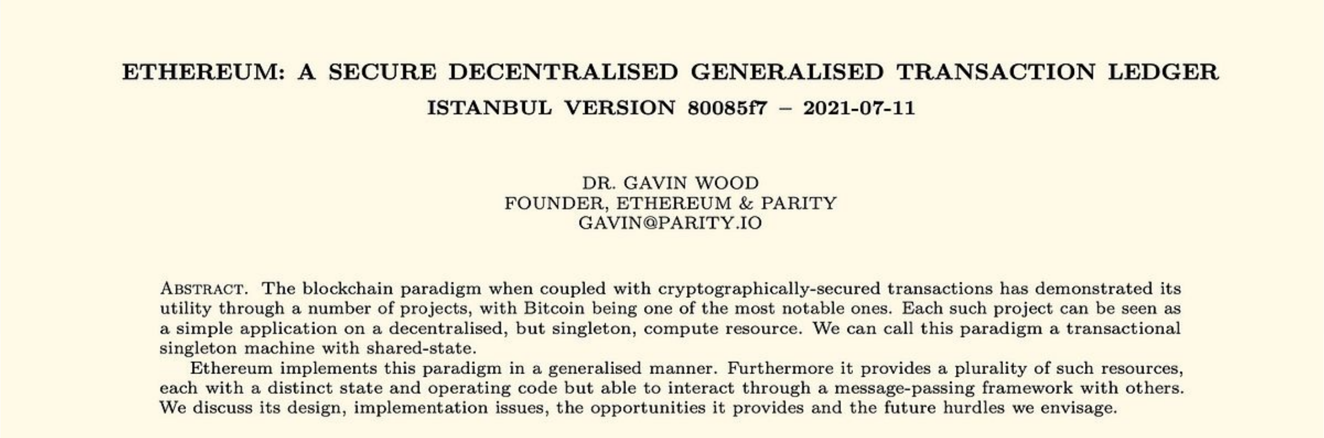
Move → Resource-oriented programming

- **Tangible** programming experience

- Linked to the physical intuitions of
  - **Exchange** → movement, transfer
  - **Ownership** → access control, possession

Criticism to existing blockchain languages

Criticism to existing blockchain languages → Ethereum Virtual Machine/Solidity



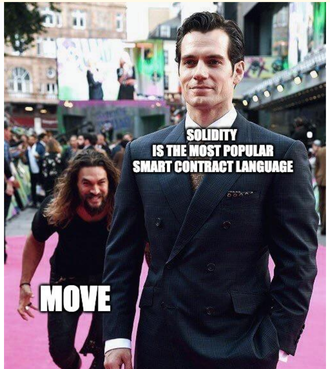
ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER

ISTANBUL VERSION 80085f7 – 2021-07-11

DR. GAVIN WOOD
FOUNDER, ETHEREUM & PARITY
GAVIN@PARITY.IO

Abstract. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, with Bitcoin being one of the most notable ones. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.



The Rise of Alternative Virtual Machines (AltVMs)

The Rise of Alternative Virtual Machines (AltVMs)

fluent
polygon Miden
sei
RADIX
SOLANA
FUEL
near
CARDANO
RISC ZERO
ARBITRUM
MONAD
cartesi

The Rise of Alternative Virtual Machines (AltVMs)

- Will **EVMs** continue to dominate or has the time come for a new leader to emerge?

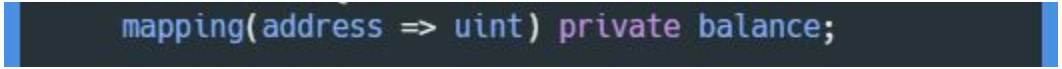
- Can **altVMs co-exist** with Ethereum or will they take over?

Criticism to existing blockchain languages

1. Indirect **asset representation**

Encoding assets using an integer number

→ but an integer is **not equivalent to an asset**.



```
mapping(address => uint) private balance;
```

Criticism to existing blockchain languages

2. Scarcity control of an asset is not built into the language

Criticism to existing blockchain languages

3. **Access control** not flexible

Criticism to existing blockchain languages

1. Indirect **asset representation**

2. Scarcity control of an asset is not built into the language

3. Access control not flexible

→

Move

*Representation of **state transitions** enabling ownership of **digital resources** to be encoded in an open source system*

First-class Resources

Move

First-class Resources

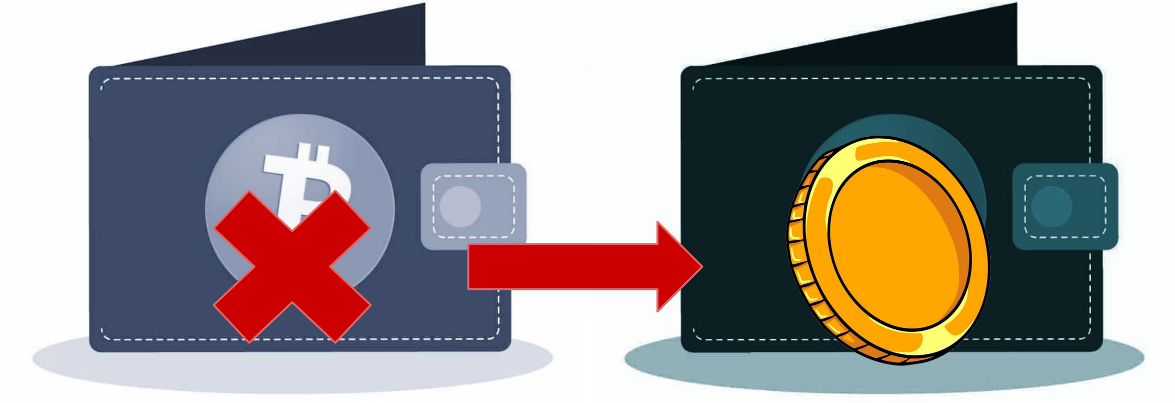
It provides the possibility of defining customized resource types with a semantics inspired by **linear logic**:

Move

First-class Resources

It provides the possibility of defining customized resource types with a semantics inspired by **linear logic**:

- a resource can never be implicitly copied or discarded



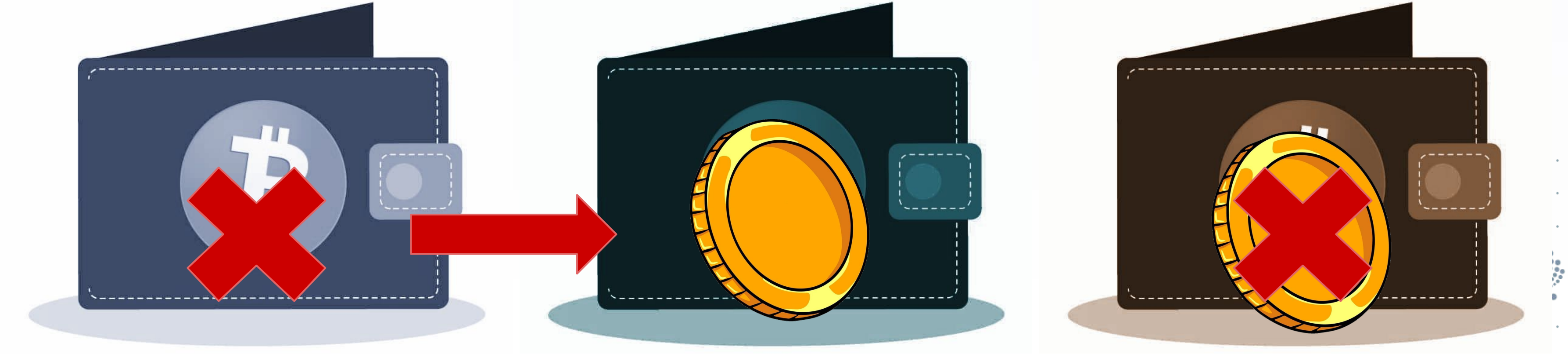
Move

First-class Resources

It provides the possibility of defining customized resource types with a semantics inspired by **linear logic**:

- a resource can never be implicitly copied or discarded

- only moved between the memory locations of the programme.



Move

First-class Resources

Move programmers can **protect access to critical operations** on resources through the

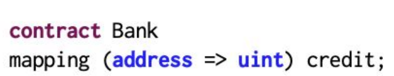
- **Modules**: contain resource types and procedures that encode rules for resources.

Move

First-class Resources

Solidity

Address	Ether Balance	Data

0x2	3.4	
		`contract Bank`
		`mapping (address => uint) credit;`

Move

Blackshear, Sam, et al. "Resources: A safe language abstraction for money." arXiv preprint https://arxiv.org/abs/2004.05106 (2020).

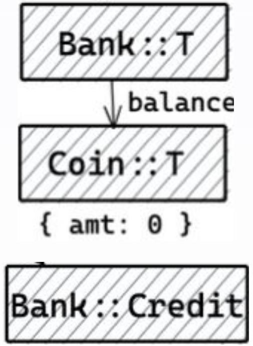
Move

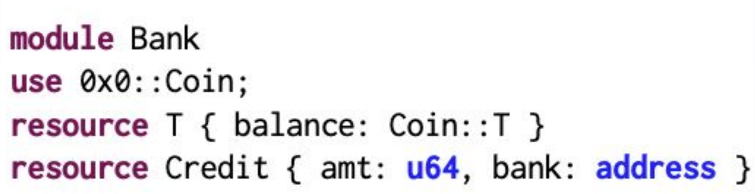
First-class Resources

Solidity

Address	Ether Balance	Data
0x2	3.4	contract Bank mapping (address => uint) credit;

Move





```
module Bank
use 0x0::Coin;
resource T { balance: Coin::T }
resource Credit { amt: u64, bank: address }
```

Blackshear, Sam, et al. "Resources: A safe language abstraction for money." arXiv preprint https://arxiv.org/abs/2004.05106 (2020).

Resource

Flexibility

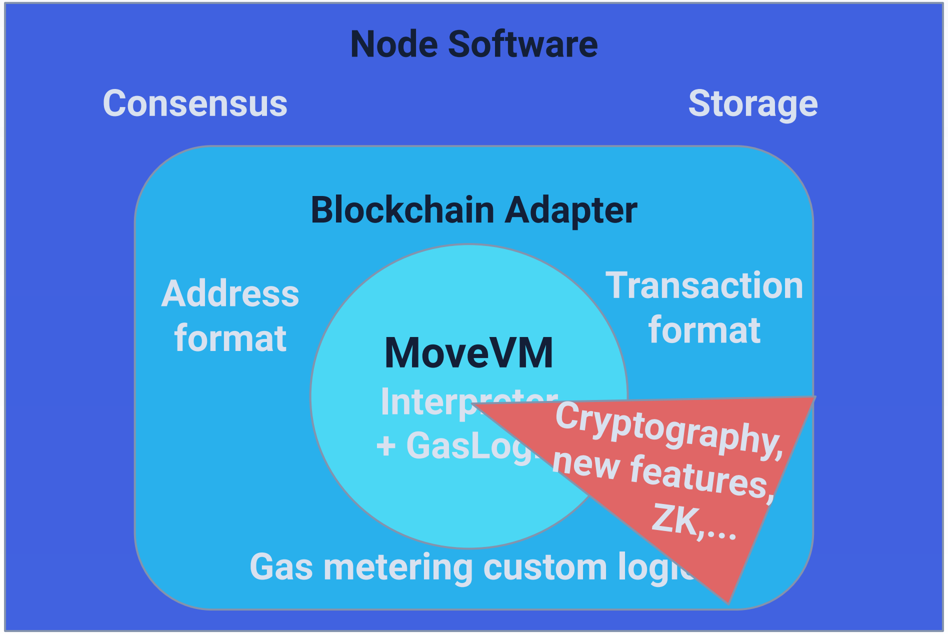
Move

MoveVM→ Easily extensible and blockchain-agnostic Virtual Machine

Flexibility

Move

Flexibility



Move

Flexibility

“Move Flavors”: two different models

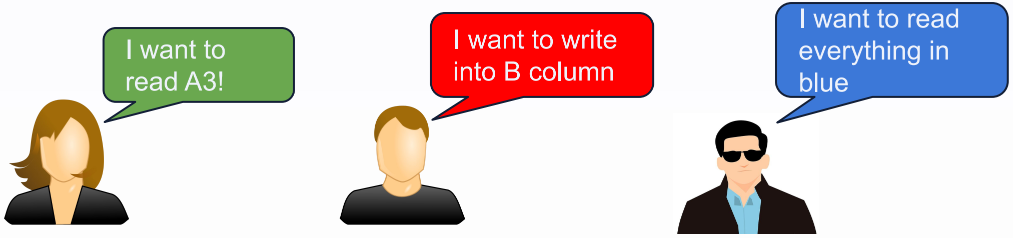
Unified Memory → Account-based Ledger
(EVM, WASM, ISC, Aptos, Diem, etc.)

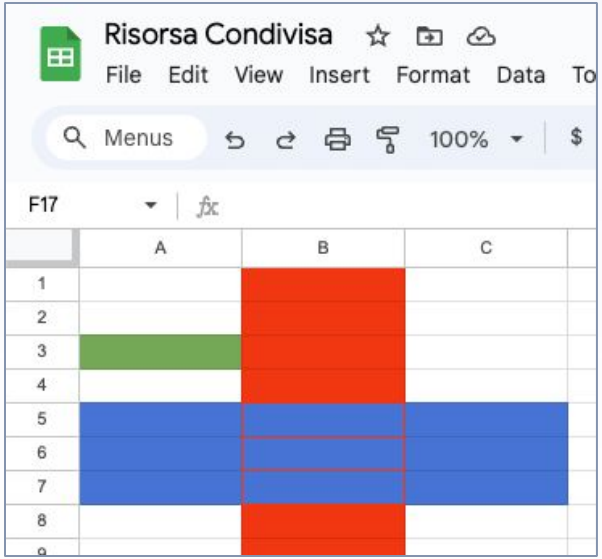
Partitioned Memory → Object-based Ledger (UTXO)
(Sui Move, Cardano, Radix, IOTA Stardust, etc.)

How to access a shared resource?

Scenario

- Everyone wants to edit the same Excel sheet
- One person needs 1 minute to update a cell

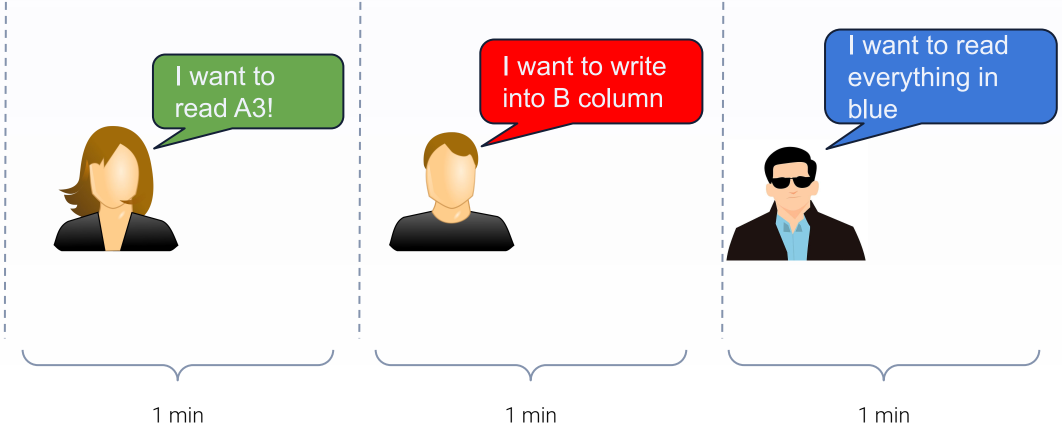

I want to read A3!
I want to write into B column
I want to read everything in blue

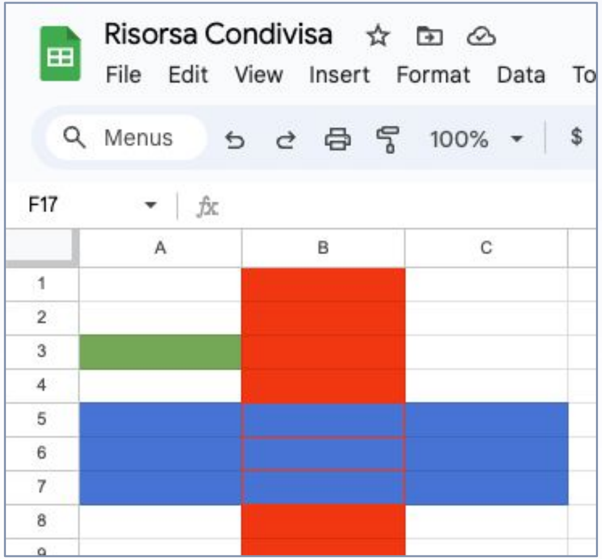


How to access a shared resource?

Unified Memory

- One person at a time can open the sheet, make changes and then save it.
- It takes **1 + 1 + 1 = 3 minutes** for everyone to finish.

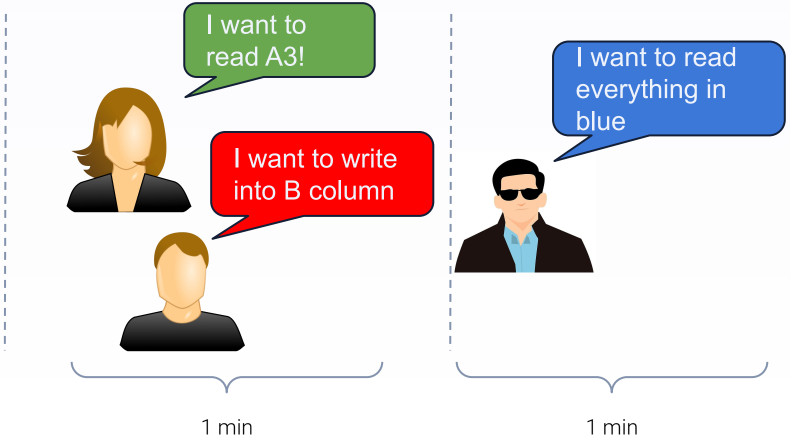

I want to read A3!
I want to write into B column
I want to read everything in blue
1 min
1 min
1 min

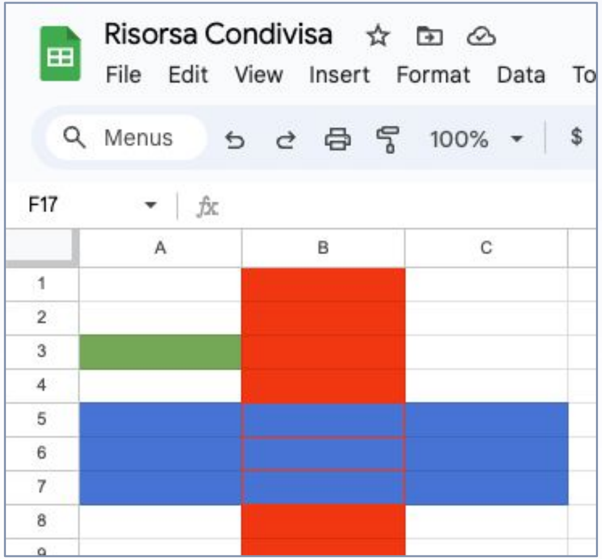


How to access a shared resource?

Partitioned Memory

- Declare the cells you will modify: if they are not in use, go ahead and modify them!
- It takes **1 + 1 = 2 minutes** before everyone is finished.


I want to read A3!
I want to write into B column
I want to read everything in blue
1 min
1 min

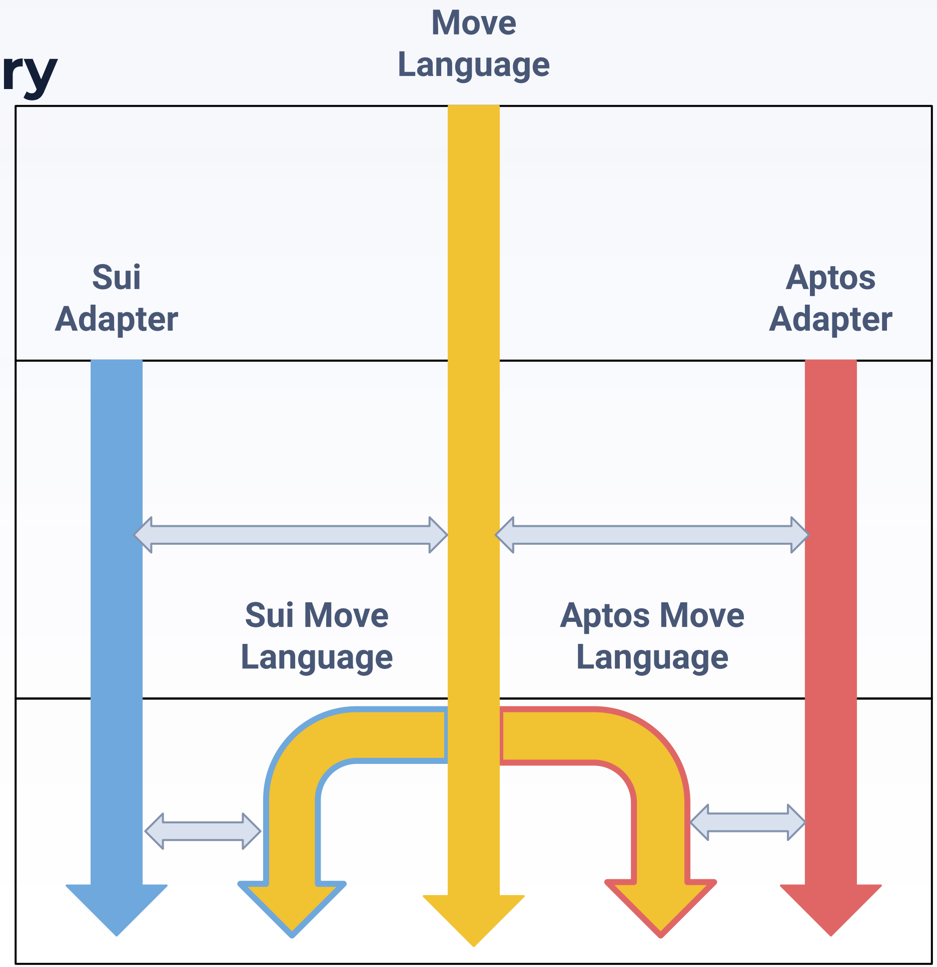


Move History

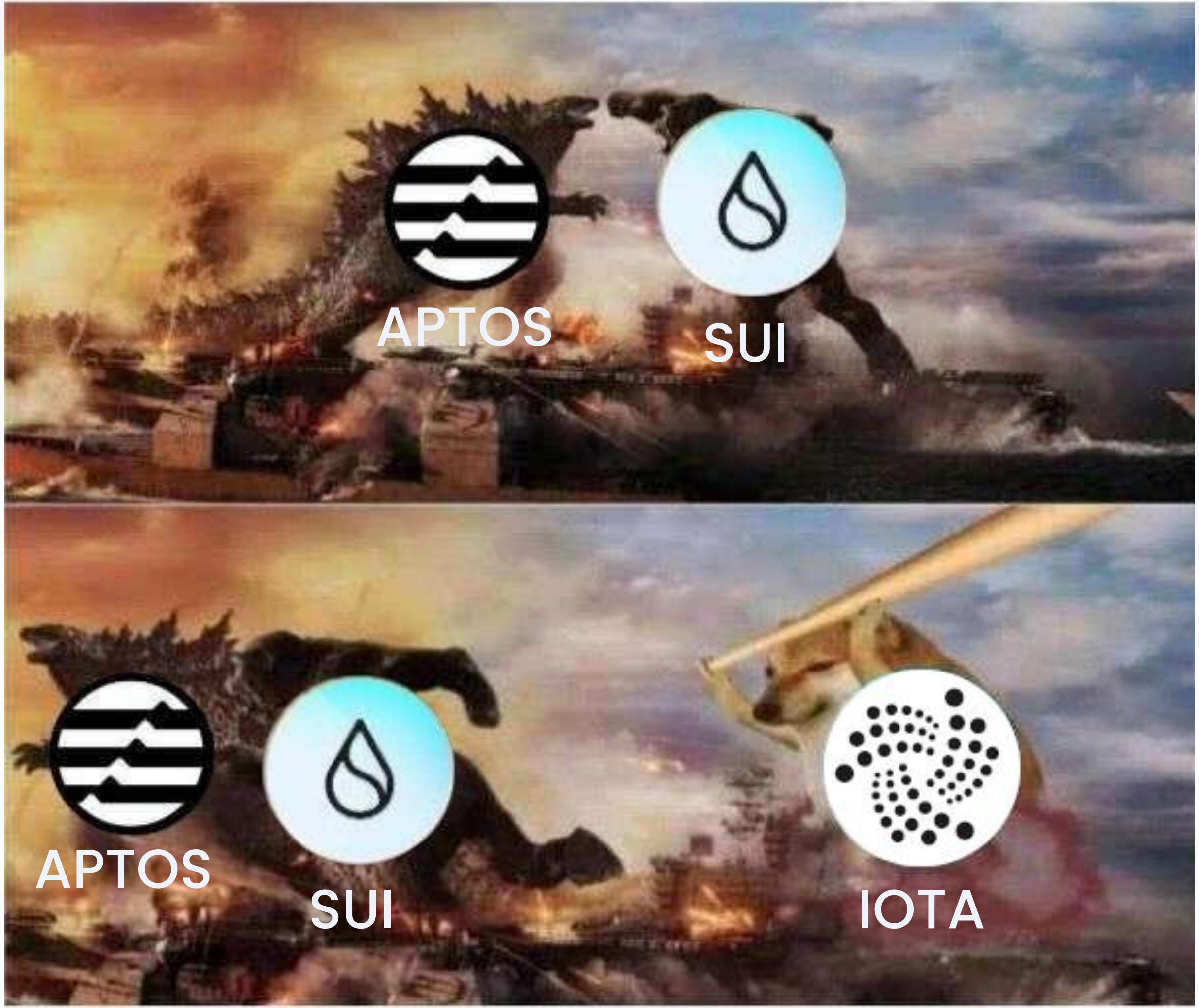
Early Move
Libra/Diem
2018-2021

Move Adapters
Sui & Aptos
2022-2023

Move 2024
Sui & Aptos Forks



APTOS
VS
sui


APTOS
SUI
APTOS
SUI
IOTA

Flexibility

Security

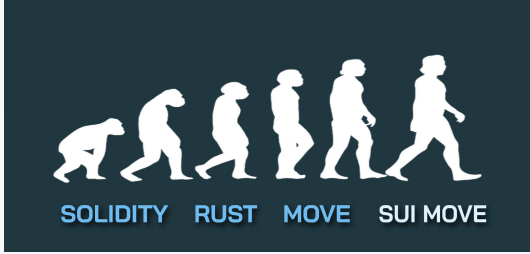
Move

"How can we write code that we are 100% sure will be safe, since we know it will have to handle money?"

Security

Move

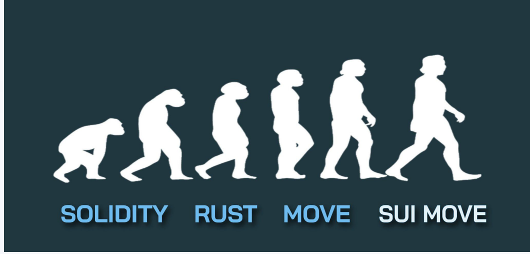
Security



- Inherits **memory and type safety** concepts from Rust
  - The compiler catches errors that would not normally be detected in other compilers (e.g. Solidity)

Move

Security



- Inherits **memory and type safety** concepts from Rust
  - The compiler catches errors that would not normally be detected in other compilers (e.g. Solidity)

- **Resource safety**
  - Simple types like integers and addresses → can be copied

  - resources → can only be moved.

  - use of **linear logic prevents 'double spending'** (moving a resource twice).

Move

Security

- **Access Control by default**
  - Forced by the language even though the programmer may forget to implement it.

Move

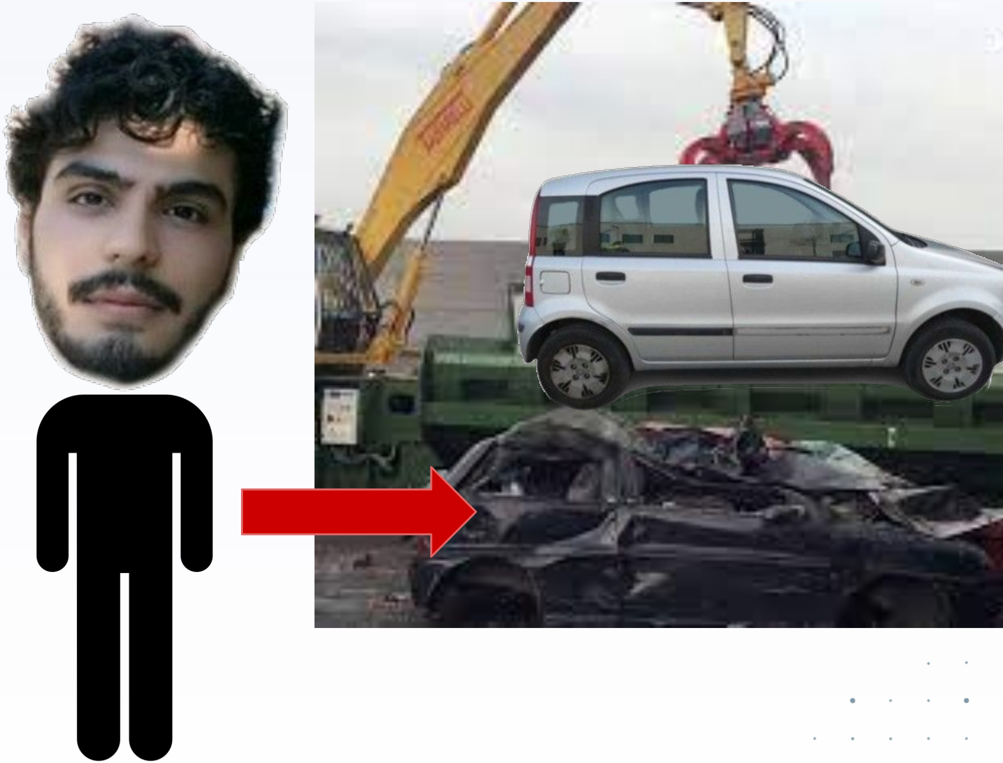
Security

- **Access Control by default**
  - Forced by the language even though the programmer may forget to implement it.
- **Limited mutability**
  - Any mutation of a value in Move occurs via a ***'reference'*** as in Rust.
    - **by-value** → value
    - **mutable** → &mut value
    - **read-only** → &value

Pass a value to a function *by-value*

Pass a value to a function *by-value*

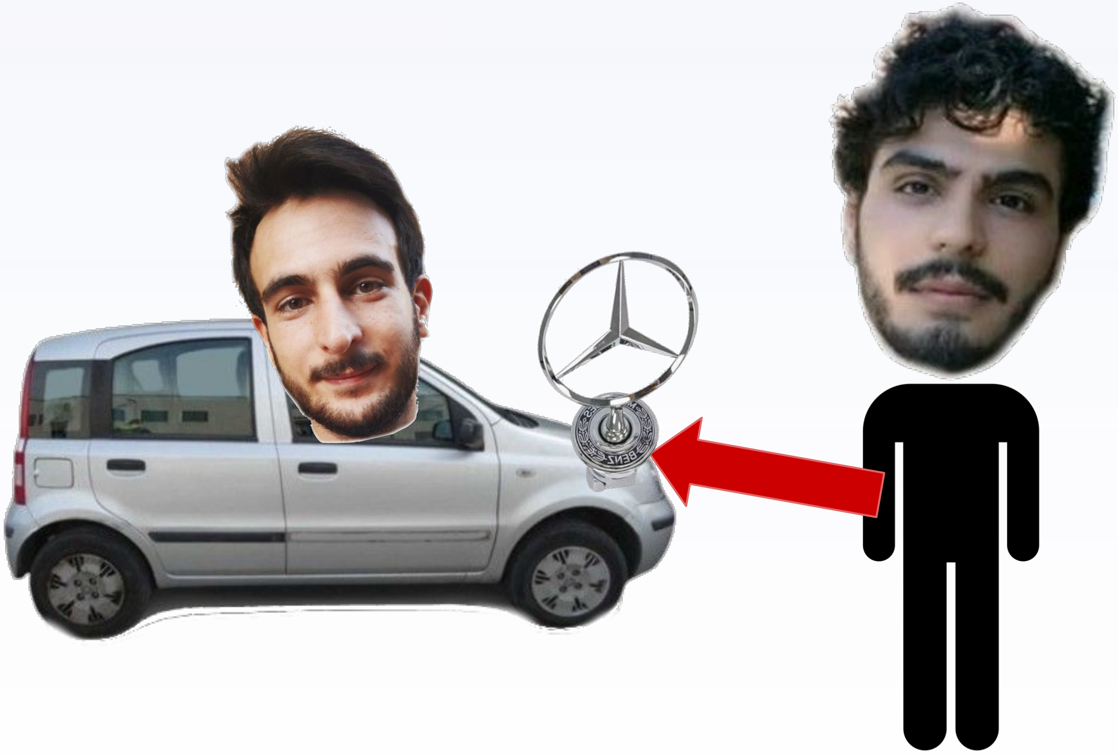
Pass a value to a function *by-value*



“Borrow” a value with *mutable ref (&mut)*

“Borrow” a value with *mutable ref (&mut)*

“Borrow” a value with *mutable ref (&mut)*



“Borrow” a value with *read-only ref* (&)

Move

Security

Double check:

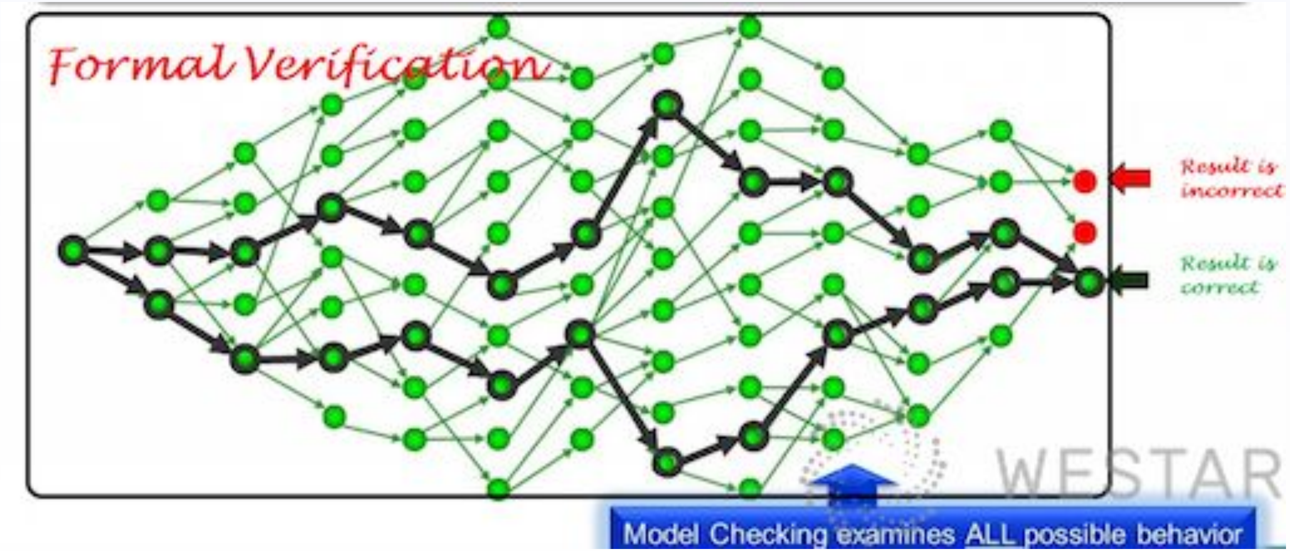
- the **high-level** programming language
  - is compiled using a **compiler that verifies security properties**

- the untyped **low-level** programming language
  - performs **security checks at runtime**

Move

Security

Move smart contracts are **Formally Verified**

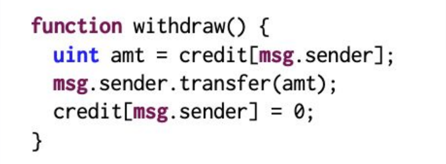


Move

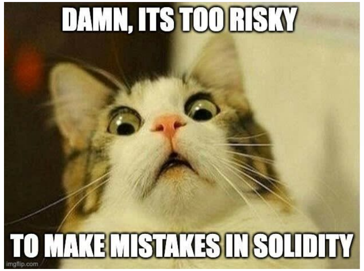
Security

NO reentrancy.

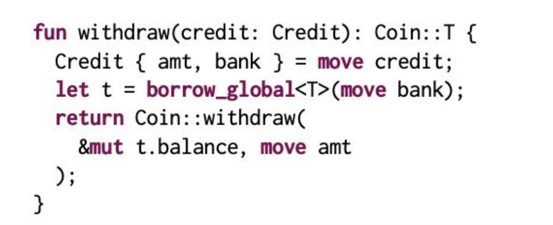
Solidity



```
function withdraw() {
  uint amt = credit[msg.sender];
  msg.sender.transfer(amt);
  credit[msg.sender] = 0;
}
```



Move



```
fun withdraw(credit: Credit): Coin::T {
  Credit { amt, bank } = move credit;
  let t = borrow_global<T>(move bank);
  return Coin::withdraw(
    &mut t.balance, move amt
  );
}
```

Blackshear, Sam, et al. "Resources: A safe language abstraction for money." arXiv preprint https://arxiv.org/abs/2004.05106 (2020).

Move

Security

NO reentrancy.

Main cause of reentrancy
→ **dynamic dispatch**:
within a smart contract you have a function whose definition is not known in advance to the developer.

In Move each time a function is called, the code that is called is statically known (static dispatch).

Security

Move

Resource

Flexibility

Security

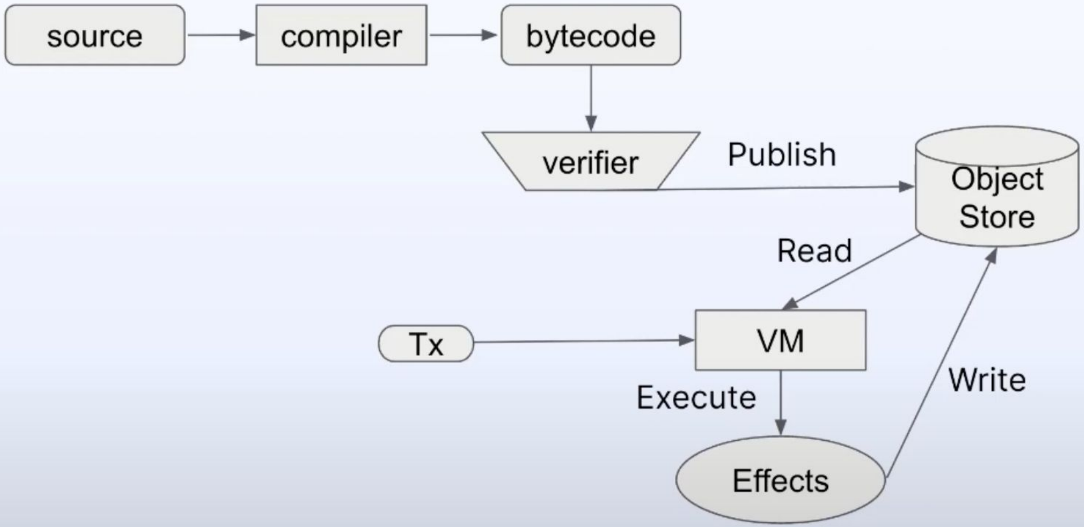
Questions?

Mirko Zichichi

mirko.zichichi@iota.org

IOTA

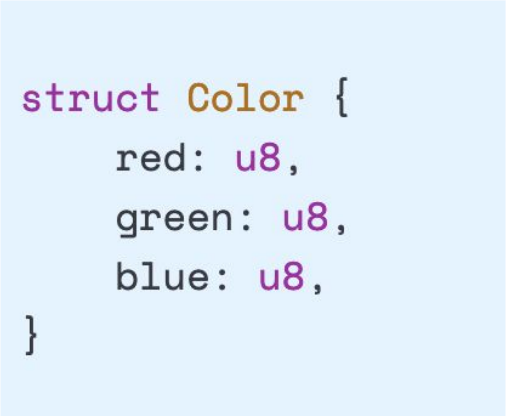
Move compile/publish/run toolchain



1. Object Basics

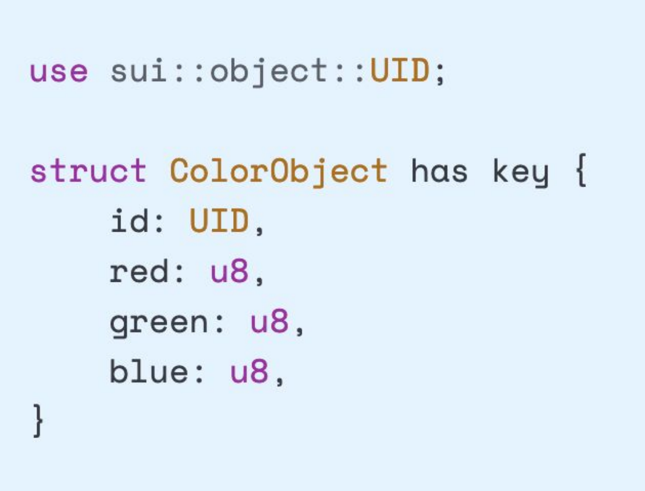
- The first field of the **struct** must be the id of the object with type **UID**

Struct



```
struct Color {
    red: u8,
    green: u8,
    blue: u8,
}
```

Object



```
use sui::object::UID;

struct ColorObject has key {
    id: UID,
    red: u8,
    green: u8,
    blue: u8,
}
```

https://docs.sui.io/build/programming-with-objects/ch1-object-basics

2. Owned, Shared and Immutable Objects

- Objects in Sui can have different types of **ownership**, with three categories:
  - **Owned mutable** object -> is owned by an address/object
  - **Shared mutable** object -> anyone can use it in a transaction
  - **Immutable** object -> an object that can't be mutated, transferred or deleted.

- In other blockchains, ***every object is shared***
  - In Sui Move programmers have the choice to implement a particular use-case using **shared objects, owned objects, or a combination**.

- In Sui, a transaction that touches a shared object needs to pass through the consensus mechanism. Whilst, a transaction that touches only owned objects does not need it.

https://docs.sui.io/learn/objects

3. Programmable Transaction Blocks

- The **inputs value** of a PTB is value is a vector of arguments, either *objects* or *pure values*
- The **commands value** of a PTB is a vector of commands using *inputs* or *results* to execute code
    - *TransferObjects* sends (one or more) objects to a specified address
    - *SplitCoins* splits off (one or more) coins from a single coin. It can be any sui::coin::Coin<_>
    - *MergeCoins* merges (one or more) coins into a single coin
    - *MakeMoveVec* creates a vector of Move values
    - ***MoveCall*** invokes either an *entry* or a *public* Move function in a published package.
    - *Publish* creates a new package and calls the init function of each module in the package.
    - *Upgrade* upgrades an existing package.
- The **result values** is a vector of values tha can be produced by each command; the type of the value can be any arbitrary Move type, not limited to objects or pure values.
- A PTB can perform up to 1,024 unique operations in a single execution.

https://docs.sui.io/concepts/transactions/prog-txn-blocks

3. Programmable Transaction Blocks

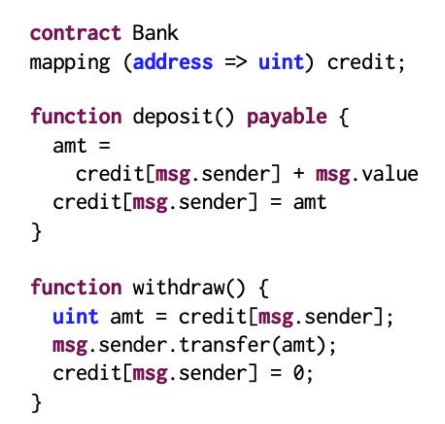
```
$ sui-ctf client ptb \
--move-call 0xd95b4510206e13fbe9413bc61183ac3b8375c8971adc54c81eeb9c96d61b5ff1::pkg::func
"<0xd95b4510206e13fbe9413bc61183ac3b8375c8971adc54c81eeb9c96d61b5ff1::pkg1::TYPE1,0xd95b451
0206e13fbe9413bc61183ac3b8375c8971adc54c81eeb9c96d61b5ff1::pkg2::TYPE2>"
@0x0b72fb4d8106699c773bf58fd0a49ffe3a08bdd58f245946d160ed5463f7ba47 99 true \
--assign result_variable \
--move-call sui::tx_context::sender \
--assign sender \
--transfer-objects "[result_variable.2]" sender \
--move-call 0xd95b4510206e13fbe9413bc61183ac3b8375c8971adc54c81eeb9c96d61b5ff1::pkg::func2
"<0xd95b4510206e13fbe9413bc61183ac3b8375c8971adc54c81eeb9c96d61b5ff1::pkg1::TYPE1"
@0x0b72fb4d8106699c773bf58fd0a49ffe3a08bdd58f245946d160ed5463f7ba47 result_variable.0 \
--gas-budget 50000000
```

https://docs.sui.io/references/cli/ptb

Move “Resource”

Solidity

Address	Ether Balance
0x2	3.4

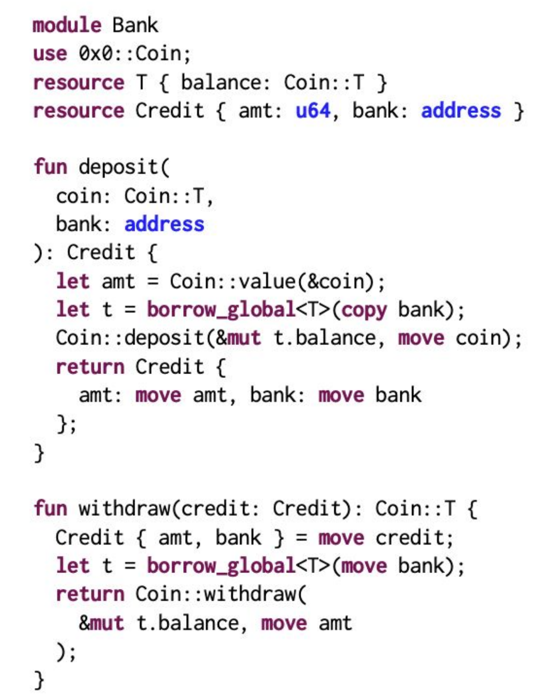


```
contract Bank
mapping (address => uint) credit;

function deposit() payable {
  amt =
    credit[msg.sender] + msg.value
  credit[msg.sender] = amt
}

function withdraw() {
  uint amt = credit[msg.sender];
  msg.sender.transfer(amt);
  credit[msg.sender] = 0;
}
```

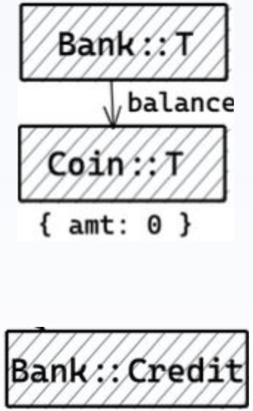
Move



```
module Bank
use 0x0::Coin;
resource T { balance: Coin::T }
resource Credit { amt: u64, bank: address }

fun deposit(
  coin: Coin::T,
  bank: address
): Credit {
  let amt = Coin::value(&coin);
  let t = borrow_global<T>(copy bank);
  Coin::deposit(&mut t.balance, move coin);
  return Credit {
    amt: move amt, bank: move bank
  };
}

fun withdraw(credit: Credit): Coin::T {
  Credit { amt, bank } = move credit;
  let t = borrow_global<T>(move bank);
  return Coin::withdraw(
    &mut t.balance, move amt
  );
}
```



Blackshear, Sam, et al. "Resources: A safe language abstraction for money." arXiv preprint https://arxiv.org/abs/2004.05106 (2020).