Discover Gnoland

The Smart Contract Platform to Improve Our Understanding of the World

pwnh4 & moul

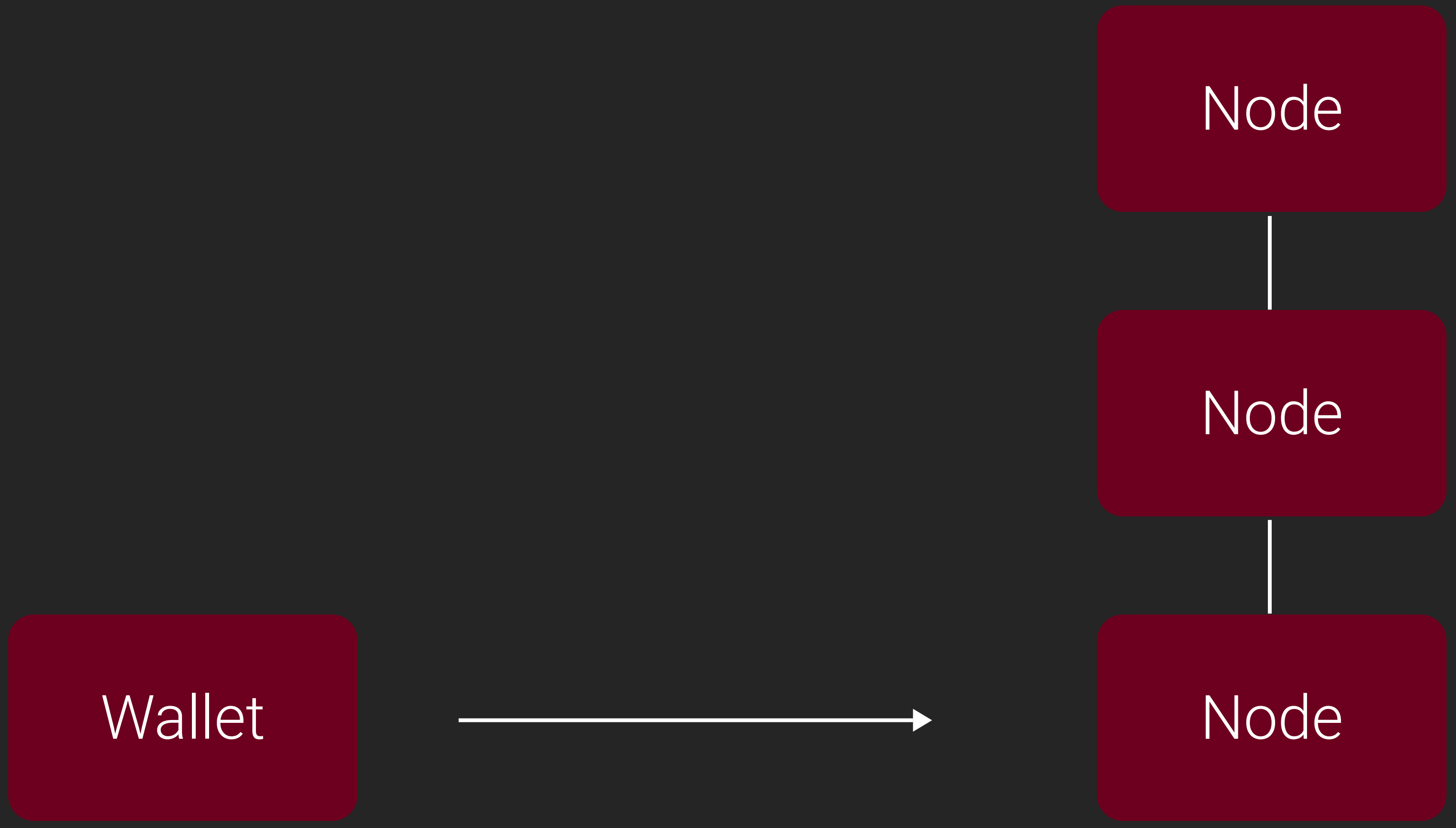
https://gnoland.space/docs/what-is-gno

[REMINDER] Smart-Contract Blockchains 1/3

GOALS

- Allow anyone to execute code via a common interface
- Code execution is verified by a P2P network
- Execution and verification are opened and verifiable by everyone

[REMINDER] Smart-Contract Blockchains 2/3



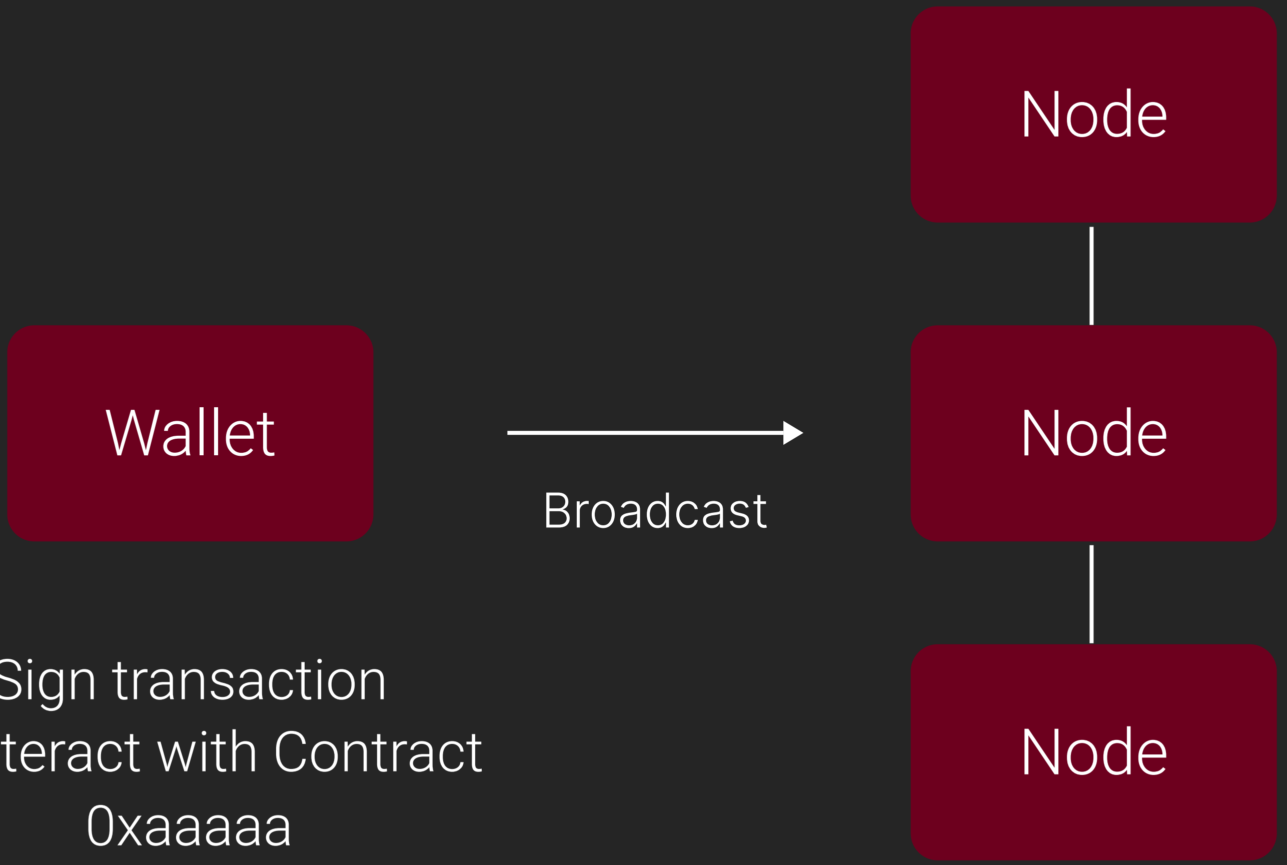
Sign transaction
to create Smart Contract

Create the Smart Contract at an
address in the network state

0xaaaaa

[REMINDER] Smart-Contract Blockchains 3/3

1. Broadcasting a transaction to the network



Wallet

Broadcast

Node

Node

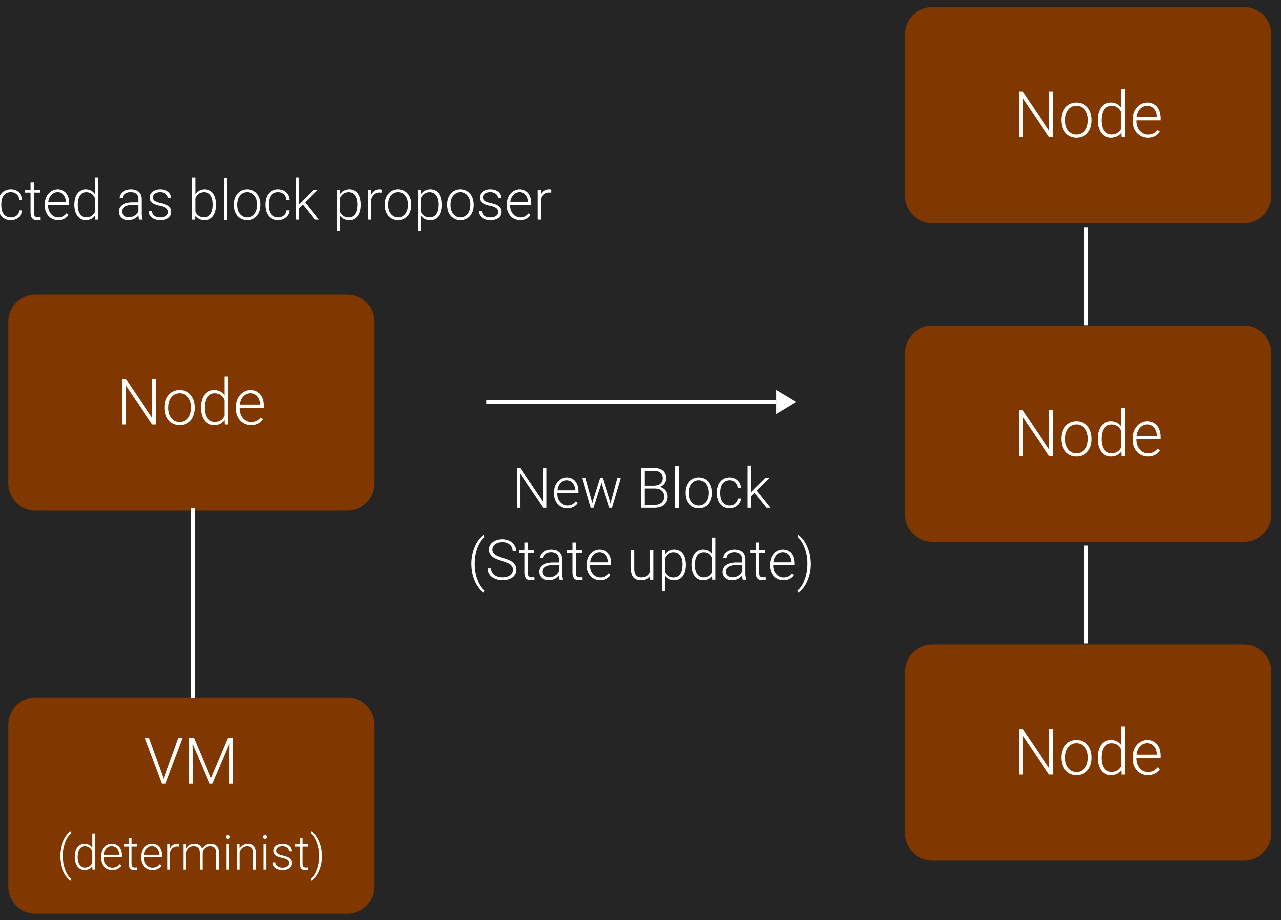
Node

Sign transaction
to interact with Contract
0xaaaaa

Share pending transactions

2. Executing transaction and storing it in the network state

1. Selected as block proposer



Node

VM
(determinist)

New Block
(State update)

Node

Node

Node

2. Execute pending transactions

Verify state

Current frictions

USERS

- Transparency of smart contracts sources (developers upload bytecode)
- Transparency of smart contracts tests

DEVELOPERS

- Have to learn blockchain specific skills (ex: solidity)
- Hard to use existing development tools
- No royalties

What Is Gno.land?

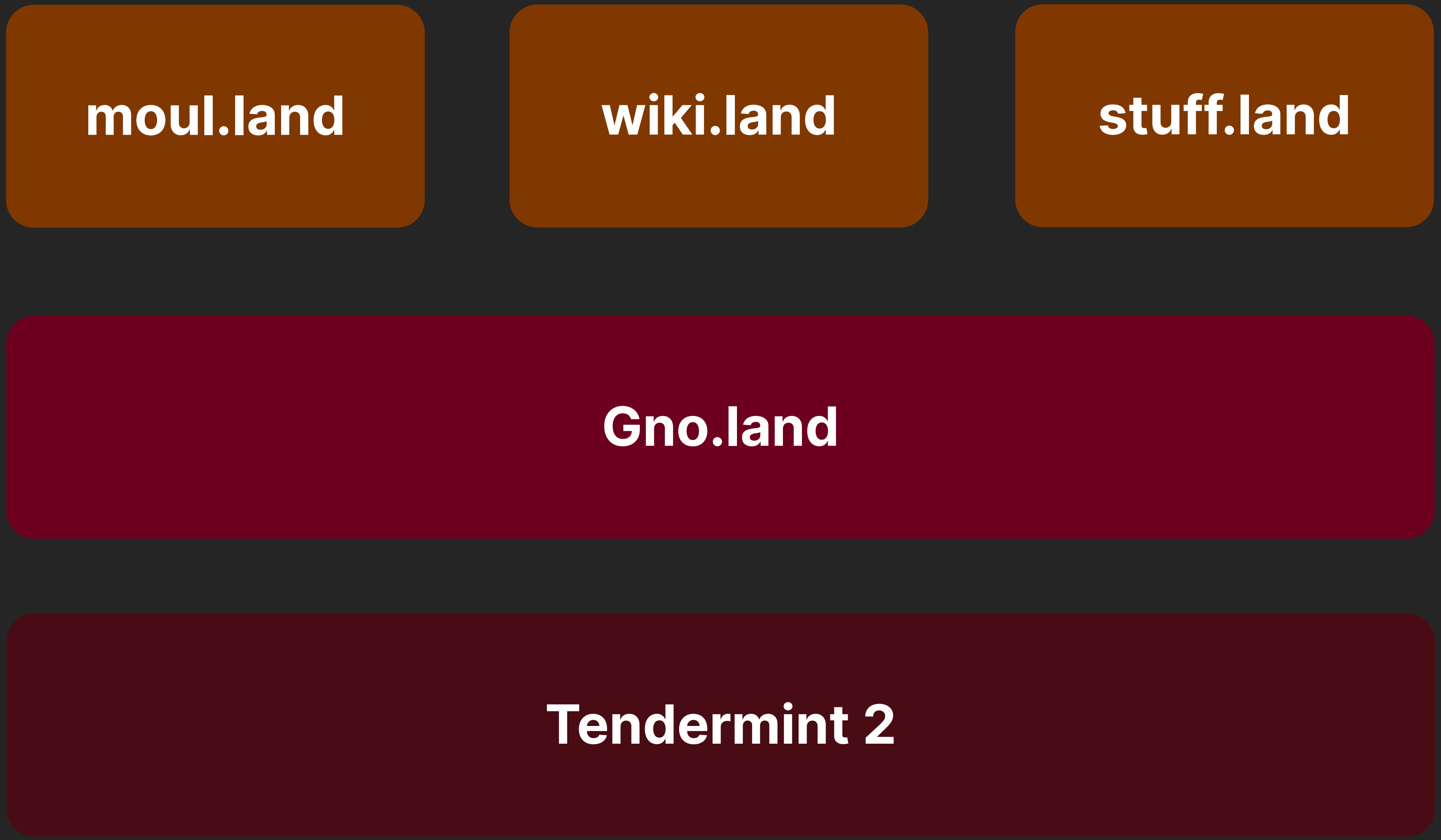
A L1 BLOCKCHAIN

- A new layer 0: Tendermint 2
- L1 Blockchain
- Code Transparency and Safety by Design

FOR DEVELOPERS

- All G(n)olang
- Proof of Contribution
- A blockchain Hub for Smart Contracts

Gno.land Ecosystem



Community chains

moul.land

wiki.land

stuff.land

Use Gno.land libraries and realms, focused chains

Hub for Gno chains

Gno.land

Libraries, Governance, Licensing fees for developers

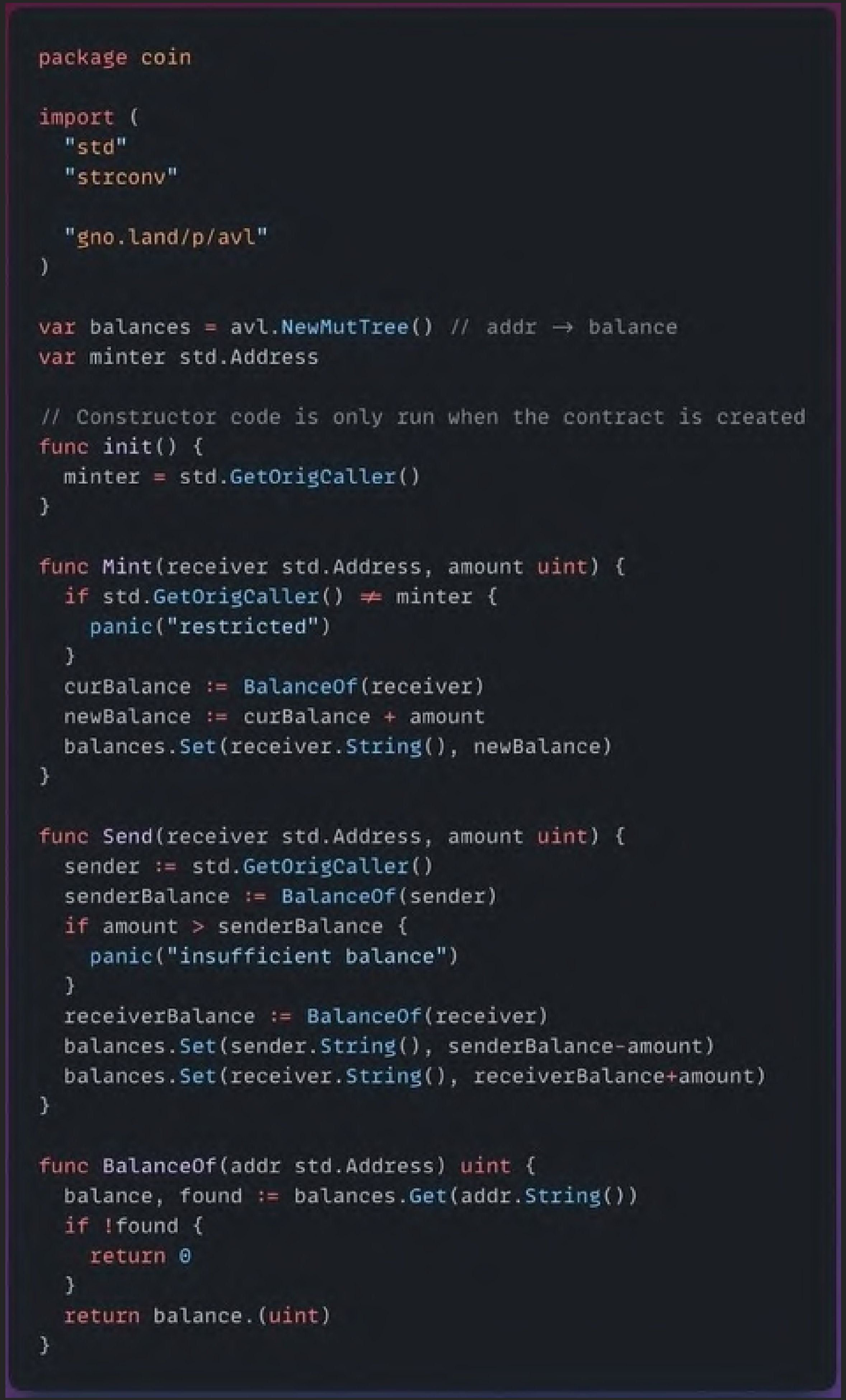
Minimal

Tendermint 2

Framework to make scalable blockchains, forked from Tendermint

Gnolang

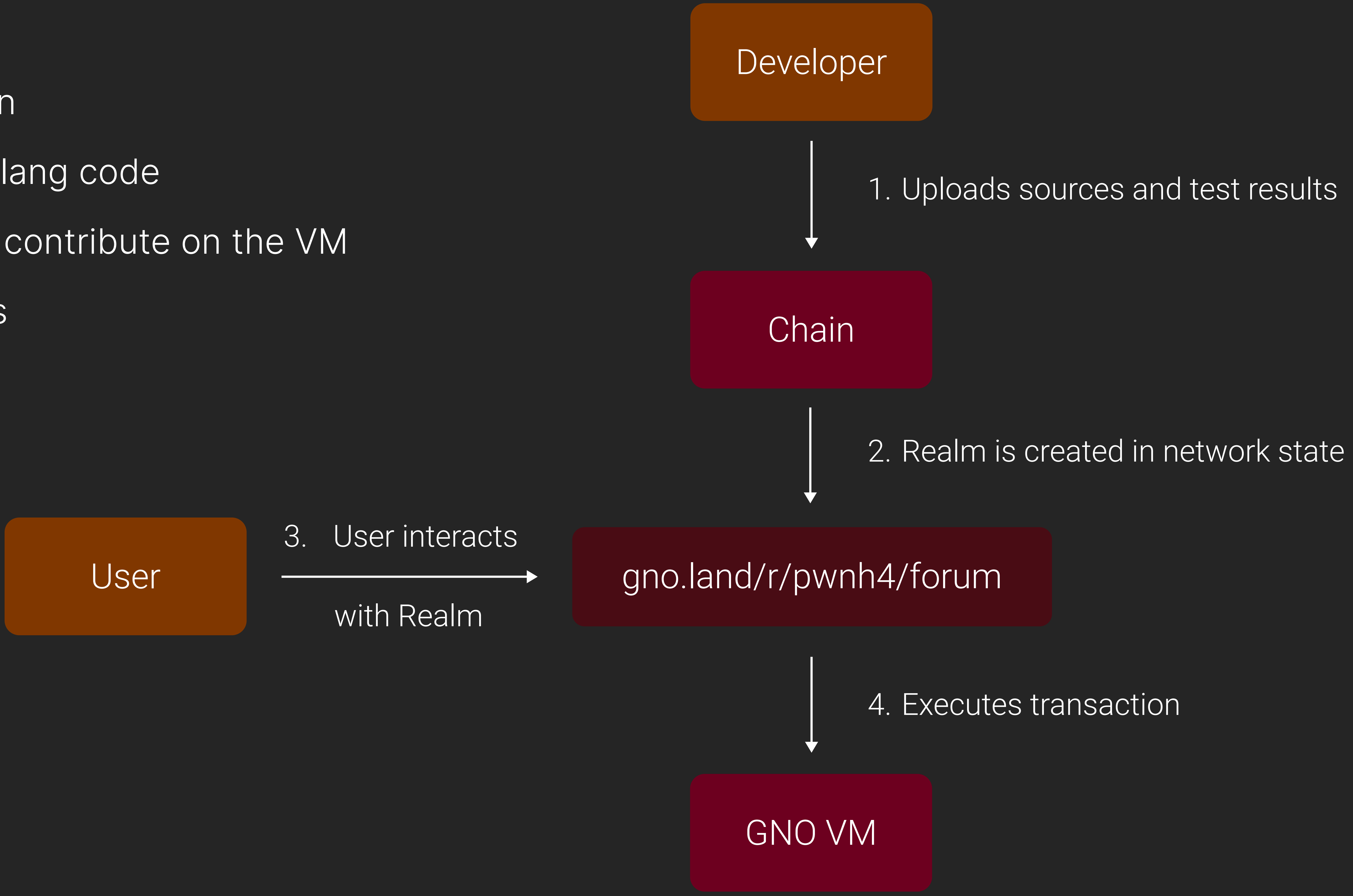
- Like Go
- Create Packages (**gno.land/p**) or Realms (**gno.land/r**)
- Robust standard library
- Unit tests



```go
package coin

import (
  "std"
  "strconv"

  "gno.land/p/avl"
)

var balances = avl.NewMutTree() // addr -> balance
var minter std.Address

// Constructor code is only run when the contract is created
func init() {
  minter = std.GetOrigCaller()
}

func Mint(receiver std.Address, amount uint) {
  if std.GetOrigCaller() != minter {
    panic("restricted")
  }
  curBalance := BalanceOf(receiver)
  newBalance := curBalance + amount
  balances.Set(receiver.String(), newBalance)
}

func Send(receiver std.Address, amount uint) {
  sender := std.GetOrigCaller()
  senderBalance := BalanceOf(sender)
  if amount > senderBalance {
    panic("insufficient balance")
  }
  receiverBalance := BalanceOf(receiver)
  balances.Set(sender.String(), senderBalance-amount)
  balances.Set(receiver.String(), receiverBalance+amount)
}

func BalanceOf(addr std.Address) uint {
  balance, found := balances.Get(addr.String())
  if !found {
    return 0
  }
  return balance.(uint)
}
```

Gnolang VM

- Upload sources to the chain
- Gnolang VM interprets Gnolang code
- AST interpretation, easy to contribute on the VM
- Typed contract interactions



Developer

1. Uploads sources and test results

Chain

2. Realm is created in network state

User

3. User interacts with Realm

gno.land/r/pwnh4/forum

4. Executes transaction

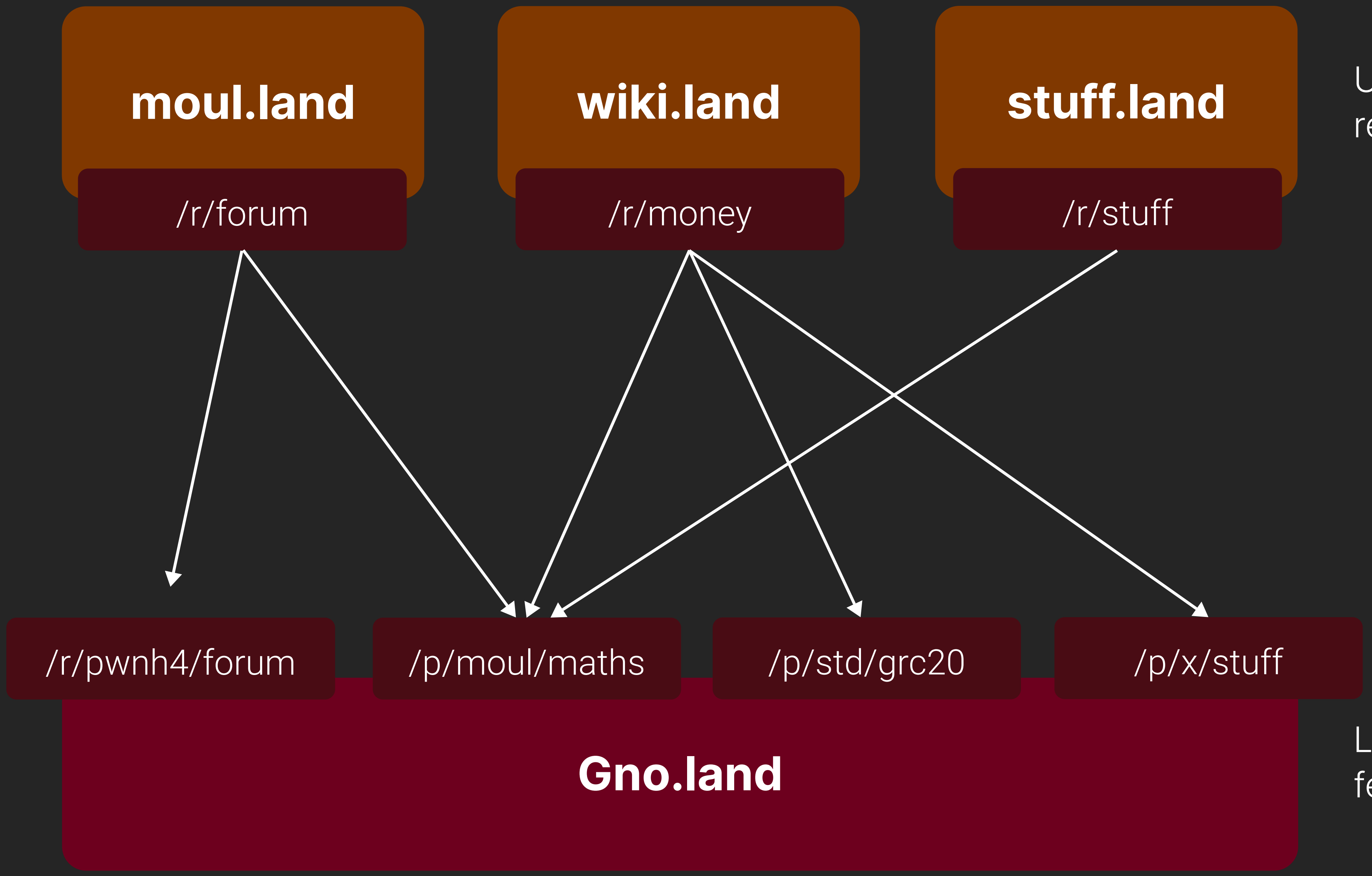
GNO VM

Gnolang Ecosystem

Community chains

IBC2

Hub for Gno chains



Use Gno.land libraries and realms, focused chains

Libraries, Governance, Licensing fees for developers

Tokenomics

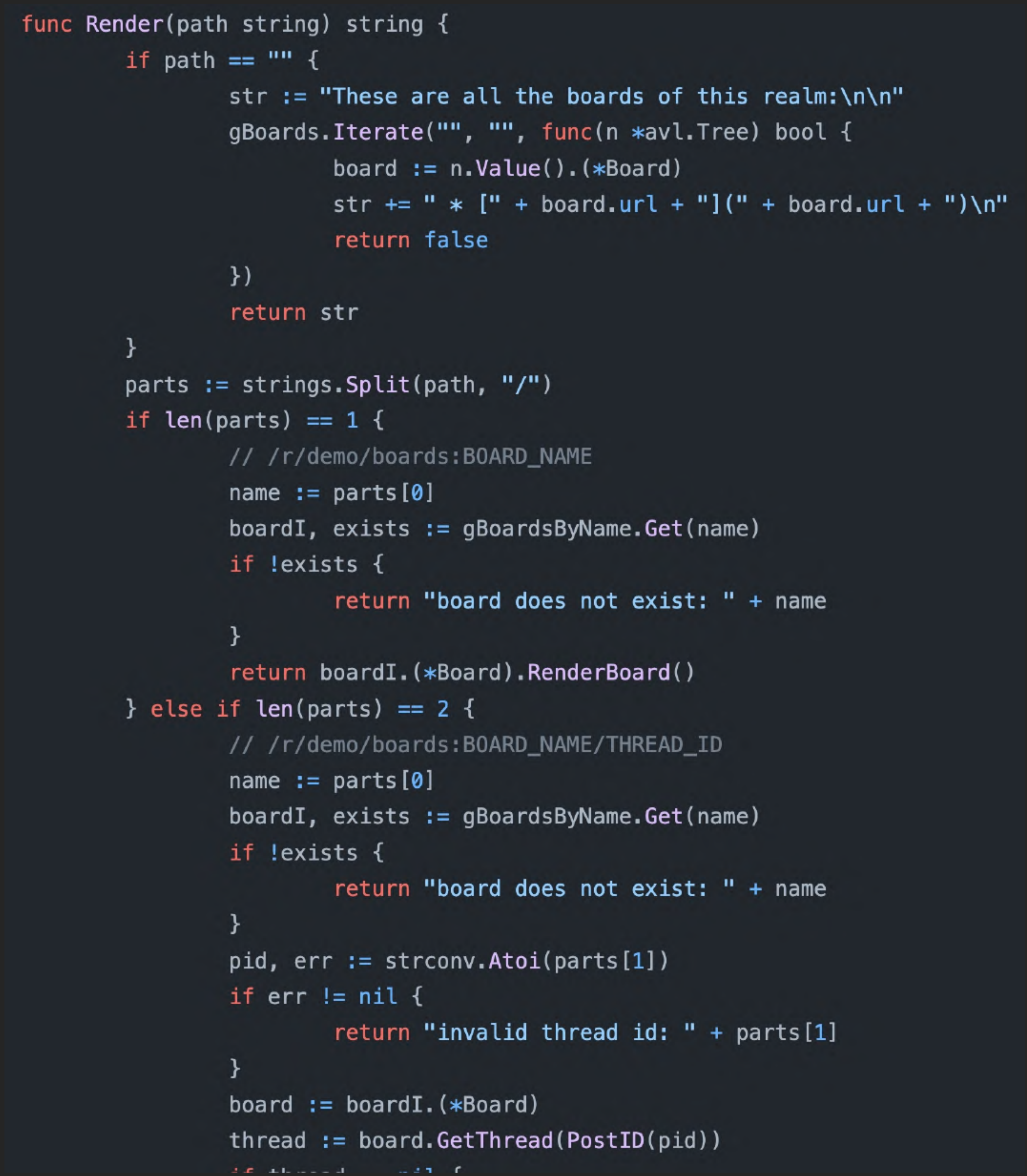
$GNOT

- "Default" token
- Used to pay transactions fees etc.
- 75% of the supply will be airdropped to ATOM stakers 1:1

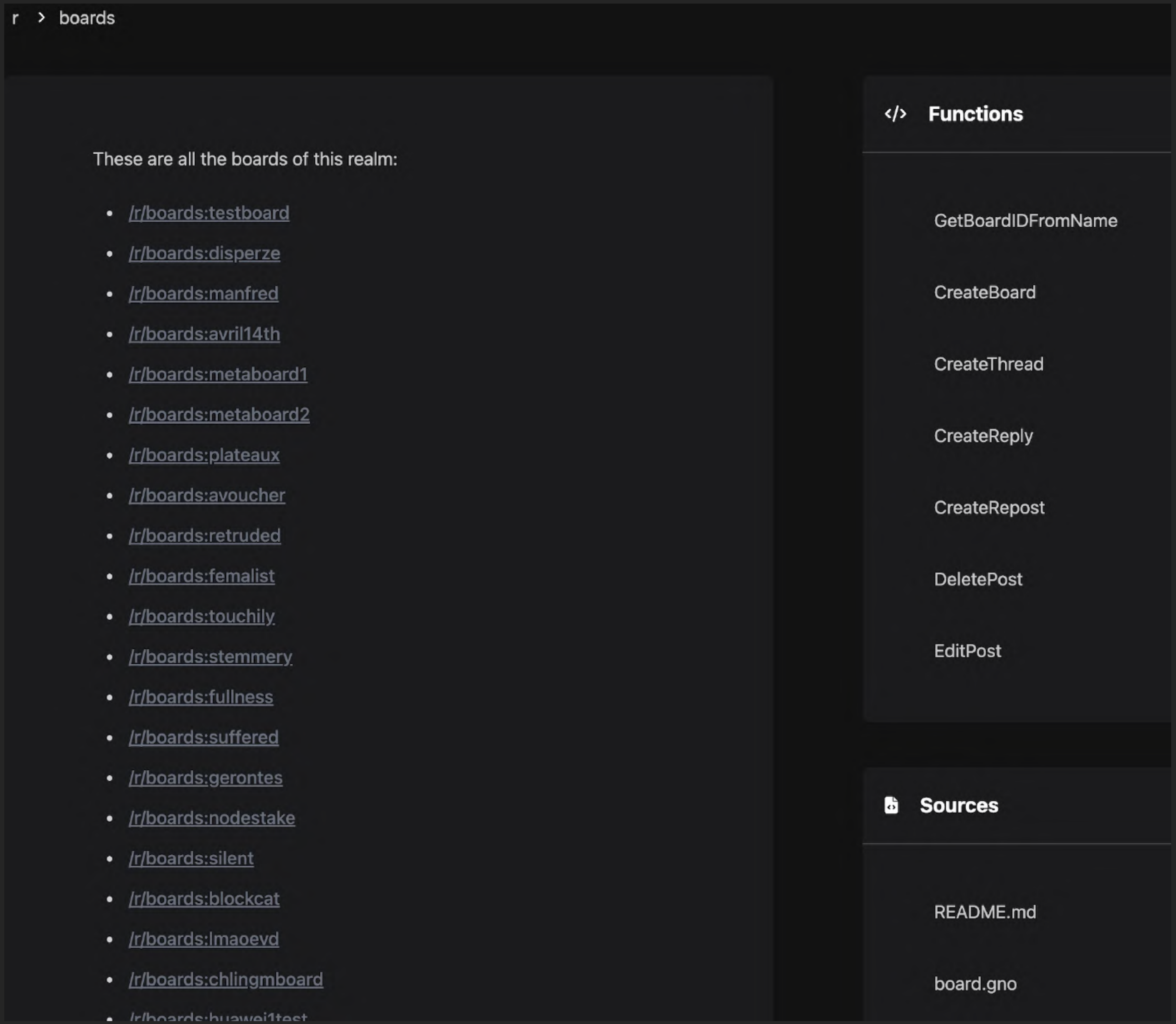
$GNOSH

- "Share" token
- Distributed to contributors
- Receive part of transaction fees

Realms renderability



```go
func Render(path string) string {
	if path == "" {
		str := "These are all the boards of this realm:\n\n"
		gBoards.Iterate("", "", func(n *avl.Tree) bool {
			board := n.Value().(*Board)
			str += " * [" + board.url + "](" + board.url + ")\n"
			return false
		})
		return str
	}
	parts := strings.Split(path, "/")
	if len(parts) == 1 {
		// /r/demo/boards:BOARD_NAME
		name := parts[0]
		boardI, exists := gBoardsByName.Get(name)
		if !exists {
			return "board does not exist: " + name
		}
		return boardI.(*Board).RenderBoard()
	} else if len(parts) == 2 {
		// /r/demo/boards:BOARD_NAME/THREAD_ID
		name := parts[0]
		boardI, exists := gBoardsByName.Get(name)
		if !exists {
			return "board does not exist: " + name
		}
		pid, err := strconv.Atoi(parts[1])
		if err != nil {
			return "invalid thread id: " + parts[1]
		}
		board := boardI.(*Board)
		thread := board.GetThread(PostID(pid))
		if thread == nil {
```



These are all the boards of this realm:

- /r/boards:testboard
- /r/boards:disperze
- /r/boards:manfred
- /r/boards:avril14th
- /r/boards:metaboard1
- /r/boards:metaboard2
- /r/boards:plateaux
- /r/boards:avoucher
- /r/boards:retruded
- /r/boards:femalist
- /r/boards:touchily
- /r/boards:stemmery
- /r/boards:fullness
- /r/boards:suffered
- /r/boards:gerontes
- /r/boards:nodestake
- /r/boards:silent
- /r/boards:blockcat
- /r/boards:lmaoevd
- /r/boards:chlingmboard
- /r/boards:huawei1test

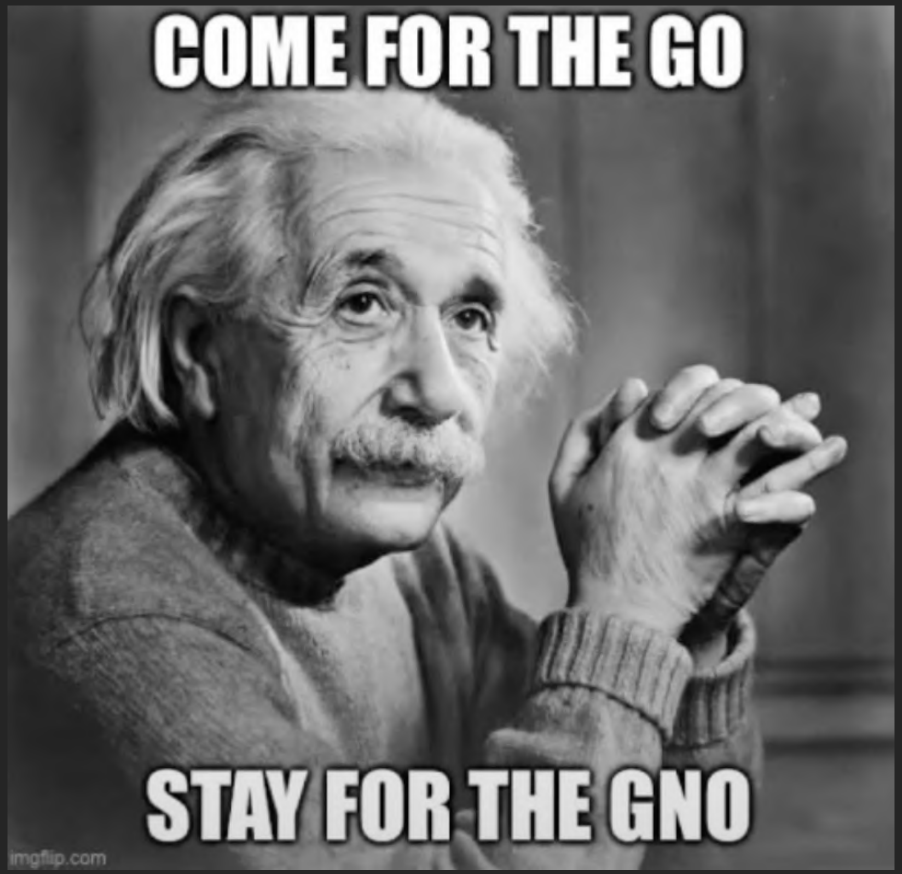
EXPOSE RENDER FUNCTION

CAN BE RENDERED IN ANY WEBAPP

LIVE CODING

Let's make a Meetup Realm on gnoland

Next challenges



Game of Realms Q1 — The best Realms will win

Core Team / Hiring — See www.ignite.com/careers

Contributors / Bounties — See /r/bounties

Documentation / Onboarding — Contribute now!

THANKS

- https://twitter.com/_gnoland
- https://github.com/gnolang/gno