A gentle introduction to

gno.land

Who am I?

- Developer Relations Engineer @ gno.land
- ~3 yrs in Web3/blockchain
- github.com/leohhhn

Leon Hudak

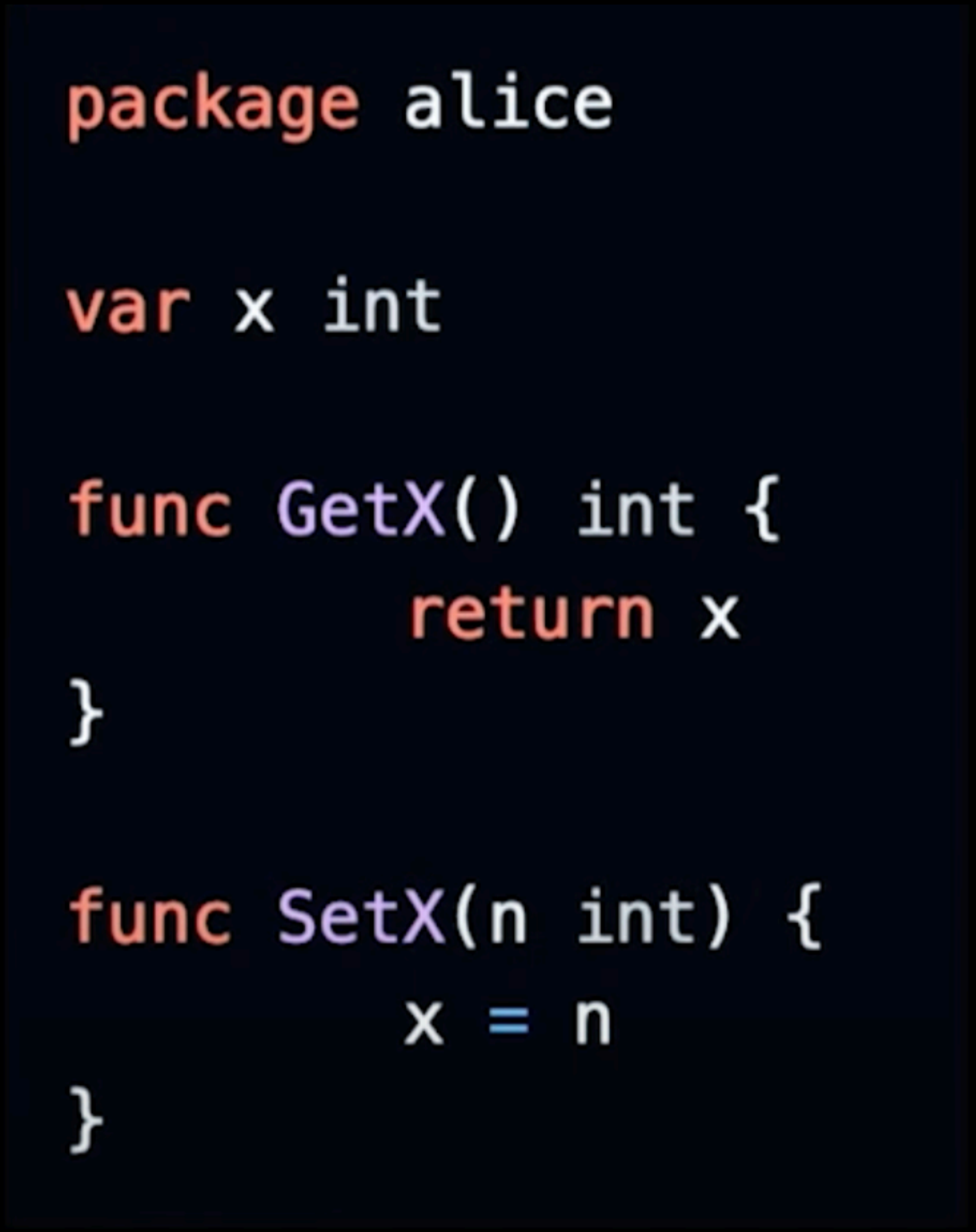
What is gno.land?

- Founded by Jae Kwon
- A new blockchain, running a custom virtual machine- the GnoVM
- Allows for writing smart contracts in Gno, an interpreted and fully deterministic version of Go

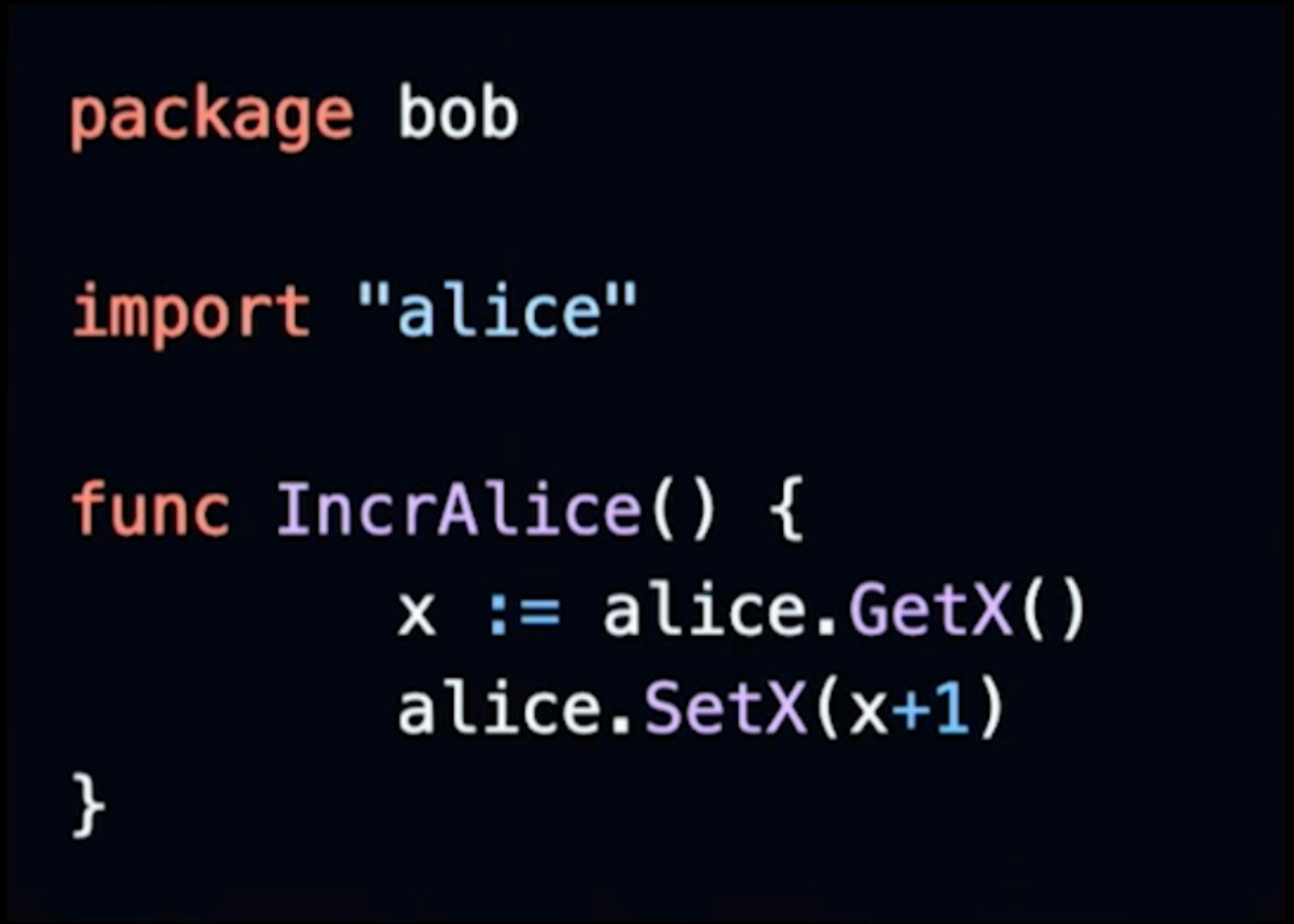
Why Go as a base?

- Go is a simple and straightforward language with a minimal learning curve
- Go has a large developer community, and lots of readily available resources, most of which can be used 1:1 for learning Gno
- Solid collection of performant, well-known standard libraries
- Gives lots of power to developers in spite of its simplicity

Gno



```
package alice

var x int

func GetX() int {
        return x
}

func SetX(n int) {
        x = n
}
```



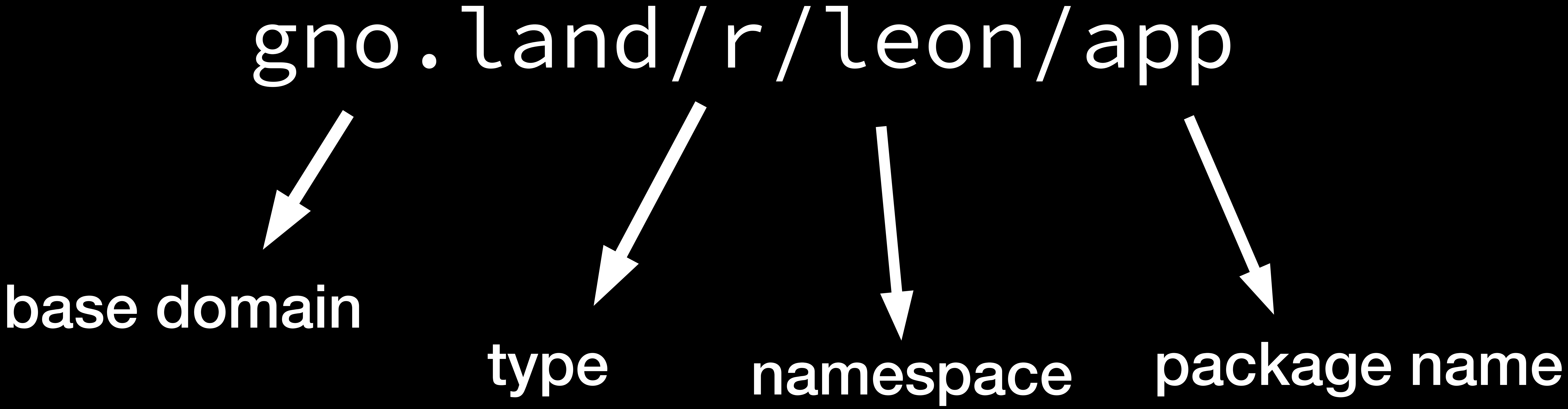
```
package bob

import "alice"

func IncrAlice() {
        x := alice.GetX()
        alice.SetX(x+1)
}
```

More on Gno…

- Modeled after Go 1.18
- Currently does not support generics & goroutines
- Gno is interpreted, allowing for on-chain composability
- All on-chain code lives on a specific package path, such as

  `gno.land/r/leon/app`

Anatomy of a Gno package path



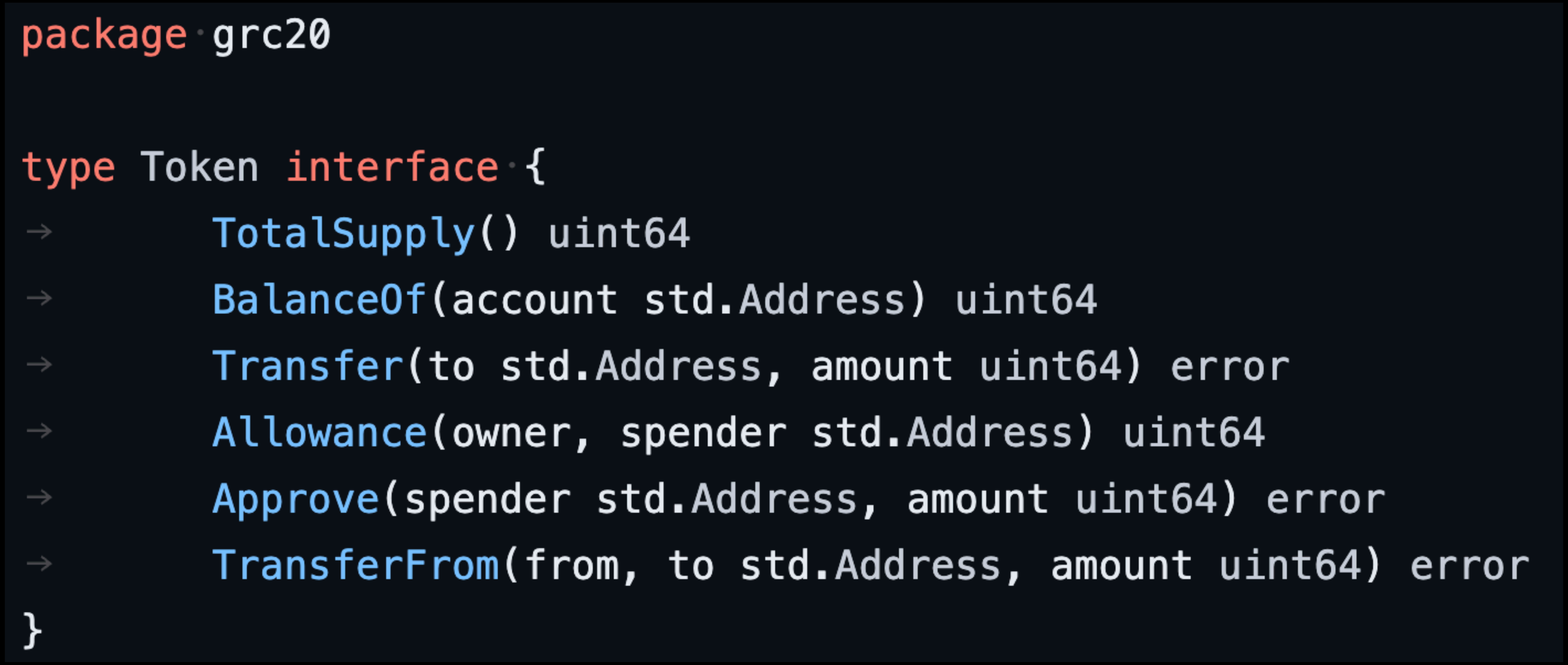
```
gno.land/r/leon/app
```

Gno code organization

- Realms, stateful code - “r/”
- On-chain libraries - “p/“
- Standard libraries
- Special “std” package

```go
package app

import (
	"std"
	"strconv"
	"strings"

	"gno.land/p/demo/avl"
	"gno.land/r/demo/users"
)
```

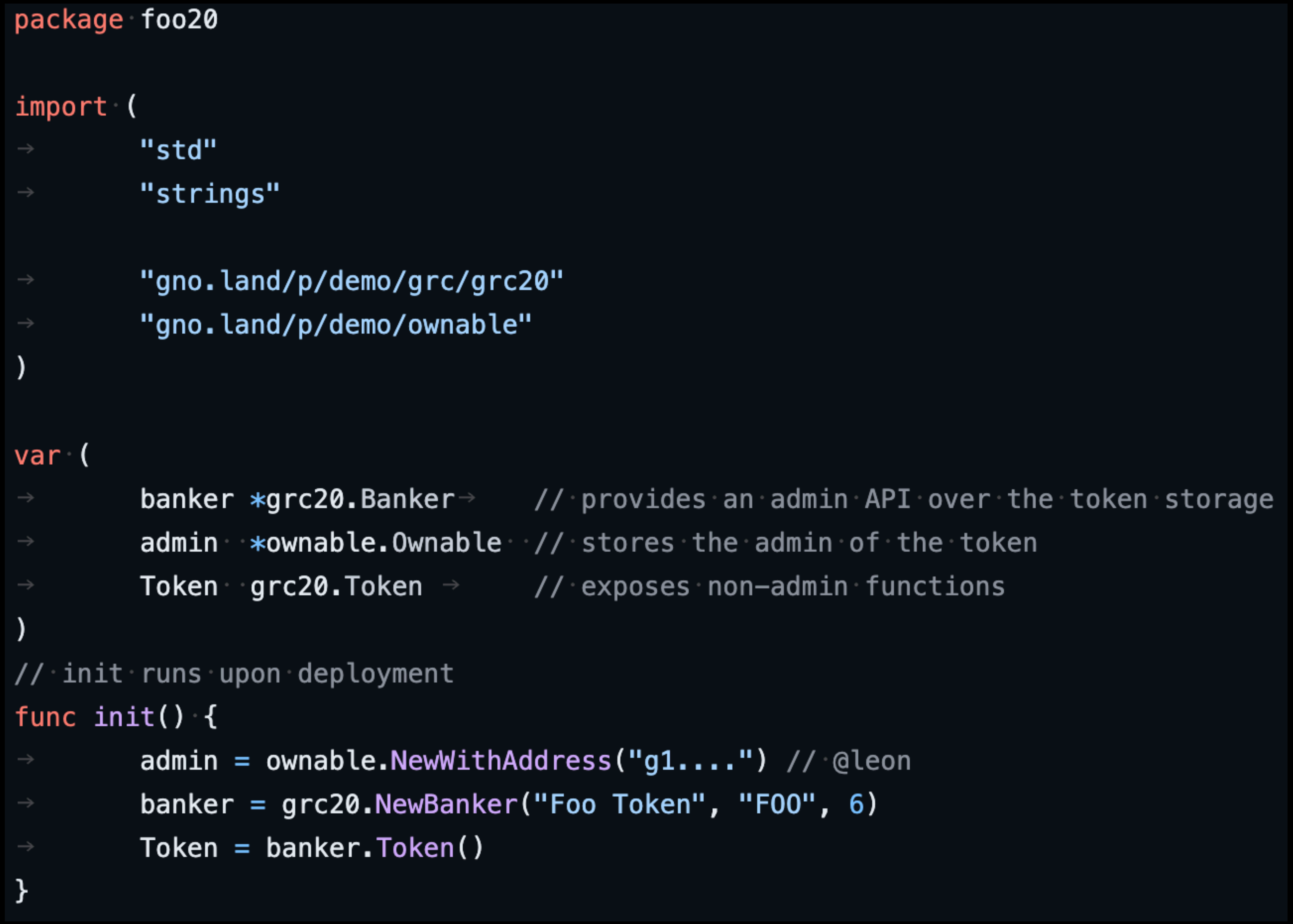
GRC20 Tokens



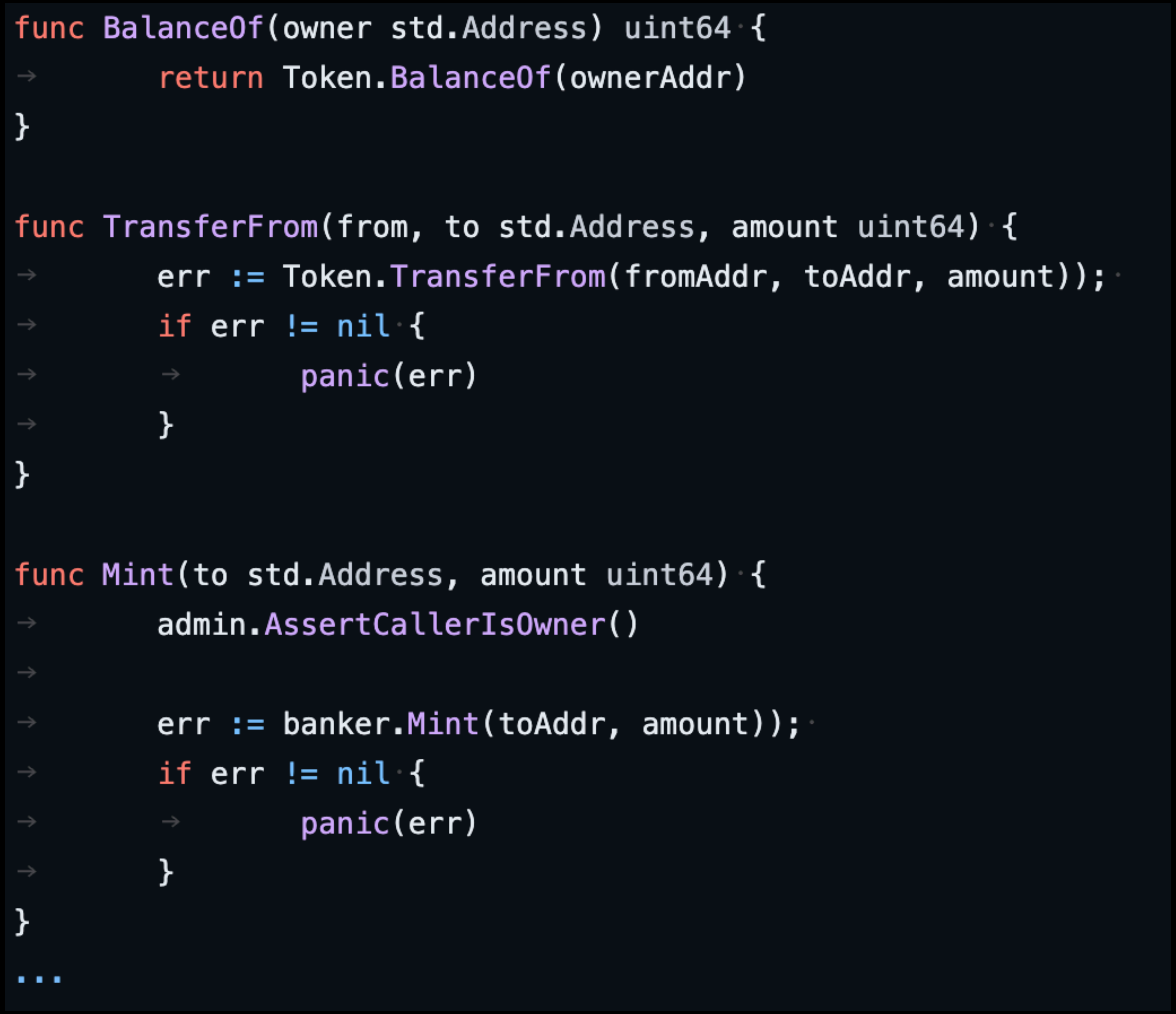
```go
package grc20

type Token interface {
	TotalSupply() uint64
	BalanceOf(account std.Address) uint64
	Transfer(to std.Address, amount uint64) error
	Allowance(owner, spender std.Address) uint64
	Approve(spender std.Address, amount uint64) error
	TransferFrom(from, to std.Address, amount uint64) error
}
```

gno.land/p/demo/grc/grc20

GRC20 Tokens

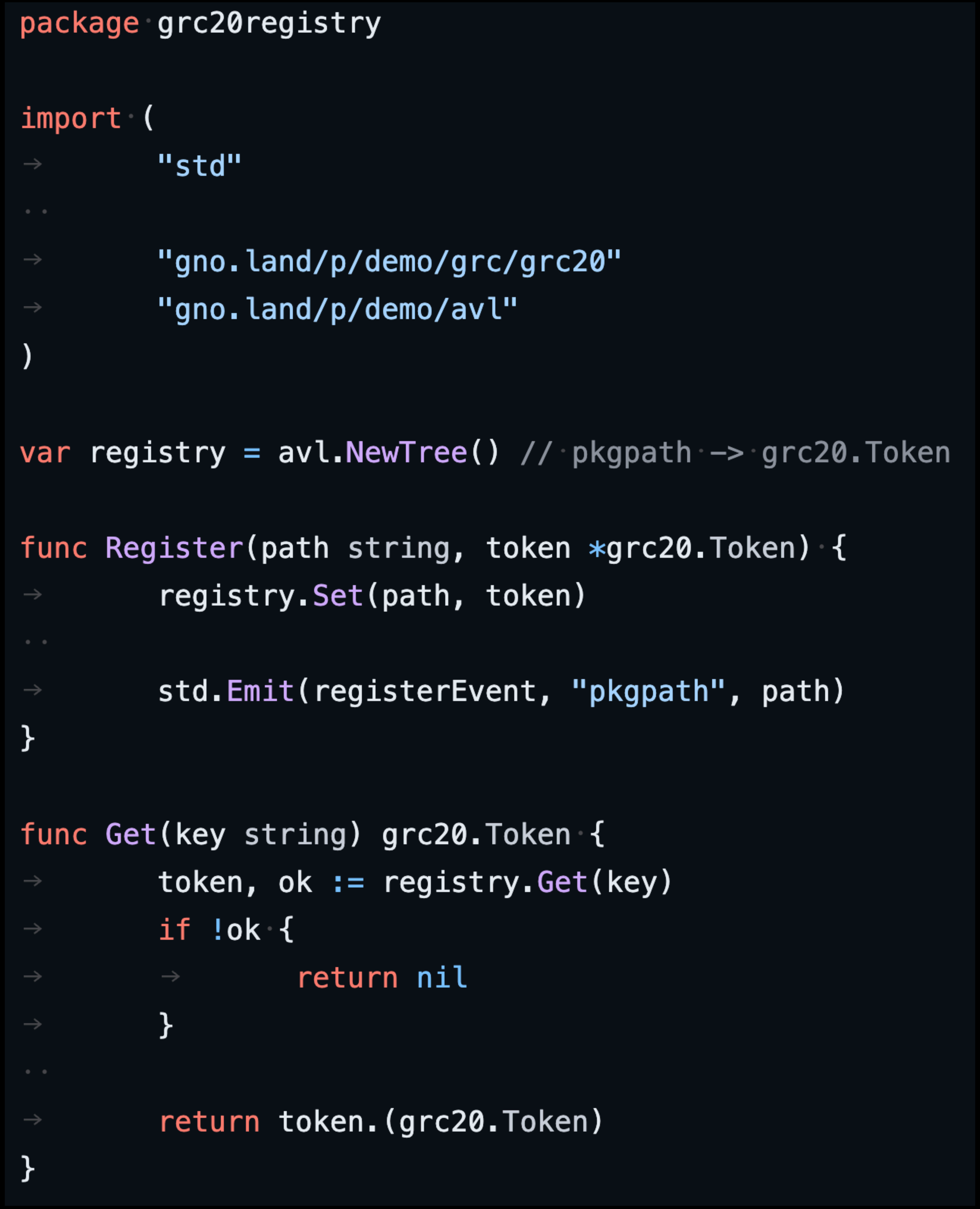


```go
package foo20

import (
	"std"
	"strings"

	"gno.land/p/demo/grc/grc20"
	"gno.land/p/demo/ownable"
)

var (
	banker *grc20.Banker	// provides an admin API over the token storage
	admin  *ownable.Ownable  // stores the admin of the token
	Token  grc20.Token	// exposes non-admin functions
)
// init runs upon deployment
func init() {
	admin = ownable.NewWithAddress("g1....") // @leon
	banker = grc20.NewBanker("Foo Token", "FOO", 6)
	Token = banker.Token()
}
```

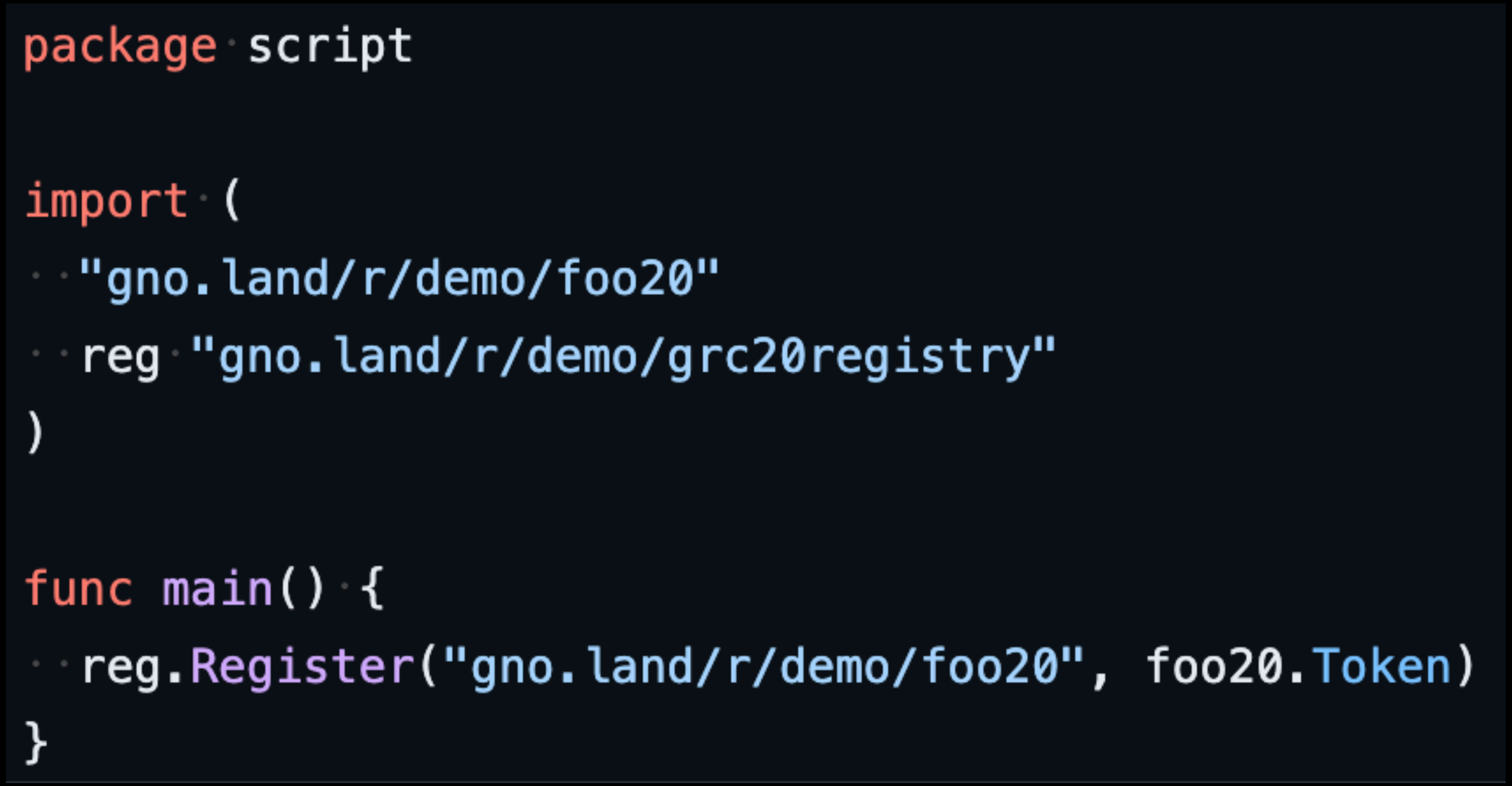


```go
func BalanceOf(owner std.Address) uint64 {
	return Token.BalanceOf(ownerAddr)
}

func TransferFrom(from, to std.Address, amount uint64) {
	err := Token.TransferFrom(fromAddr, toAddr, amount));
	if err != nil {
		panic(err)
	}
}

func Mint(to std.Address, amount uint64) {
	admin.AssertCallerIsOwner()

	err := banker.Mint(toAddr, amount));
	if err != nil {
		panic(err)
	}
}
...
```

GRC20 Registry



```go
package grc20registry

import (
	"std"

	"gno.land/p/demo/grc/grc20"
	"gno.land/p/demo/avl"
)

var registry = avl.NewTree() // pkgpath -> grc20.Token

func Register(path string, token *grc20.Token) {
	registry.Set(path, token)

	std.Emit(registerEvent, "pkgpath", path)
}

func Get(key string) grc20.Token {
	token, ok := registry.Get(key)
	if !ok {
		return nil
	}

	return token.(grc20.Token)
}
```



```go
package script

import (
  "gno.land/r/demo/foo20"
  reg "gno.land/r/demo/grc20registry"
)

func main() {
  reg.Register("gno.land/r/demo/foo20", foo20.Token)
}
```

“GitHub of the Ecosystem”

- All deployed code is open-source
- Once Gno code is deployed, anyone can import and use it

Displaying realm state with `Render()`

- gnoweb - a minimalistic, universal web server for gno.land
- gnoweb renders realm state formatted as a markdown string
- gno-js & tm2-js clients still available for custom frontends

```go
package app

var msg = "Hello Berlin!"

func UpdateMsg(newMsg string) {
	msg = newMsg
}

func Render(_ string) string {
	return msg
}
```

Tooling & Ecosystem

gnokey

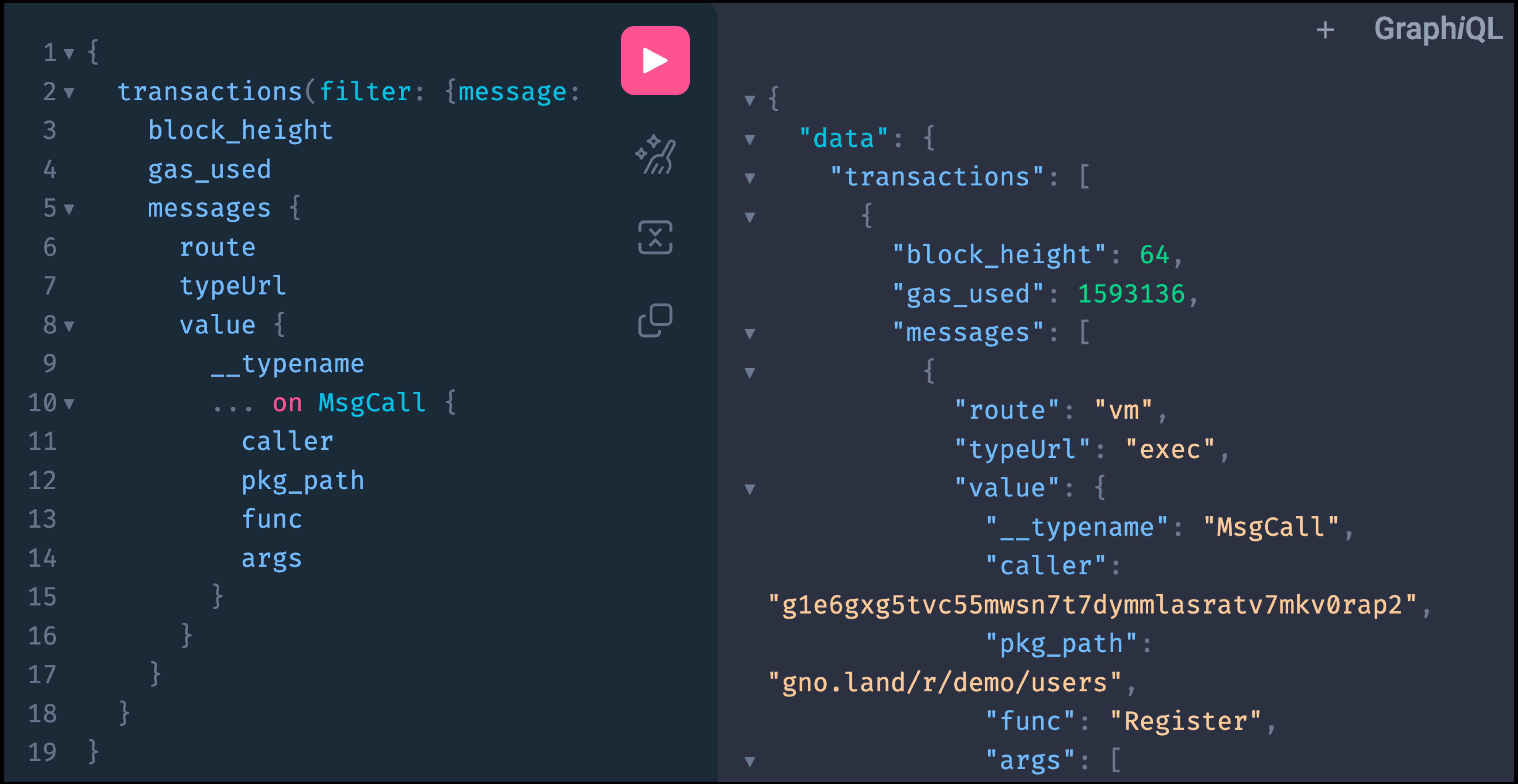
```
❯ gnokey --help
USAGE
  <subcommand> [flags] [<arg>...]

gno.land keychain & client

SUBCOMMANDS
  add        adds key to the keybase
  delete     deletes a key from the keybase
  generate   generates a bip39 mnemonic
  export     exports private key armor
  import     imports encrypted private key armor
  list       lists all keys in the keybase
  sign       signs the given tx document and saves it to disk
  verify     verifies the document signature
  query      makes an ABCI query
  broadcast  broadcasts a signed document
  maketx     composes a tx document to sign
```

gnodev

- All-in-one local Gno development environment
- Contains: local in-memory blockchain node, gnoweb, account premining & a directory watcher with automatic code reloading

```
❯ gnodev
Accounts       │ I default address imported name=test1 addr=g1jg8mtutu
9khhfwc4nxmuhcpftf0pajdhfvsqf5
               │ I pkgs loaded path="[{/Users/sasurai/Desktop/gno/gno/
examples g1jg8mtutu9khhfwc4nxmuhcpftf0pajdhfvsqf5 }]"
Node           │ I node started lisn=tcp://127.0.0.1:26657 chainID=dev
GnoWeb         │ I gnoweb started lisn=http://127.0.0.1:8888
--- READY      │ I for commands and help, press `h`
```

tx-indexer

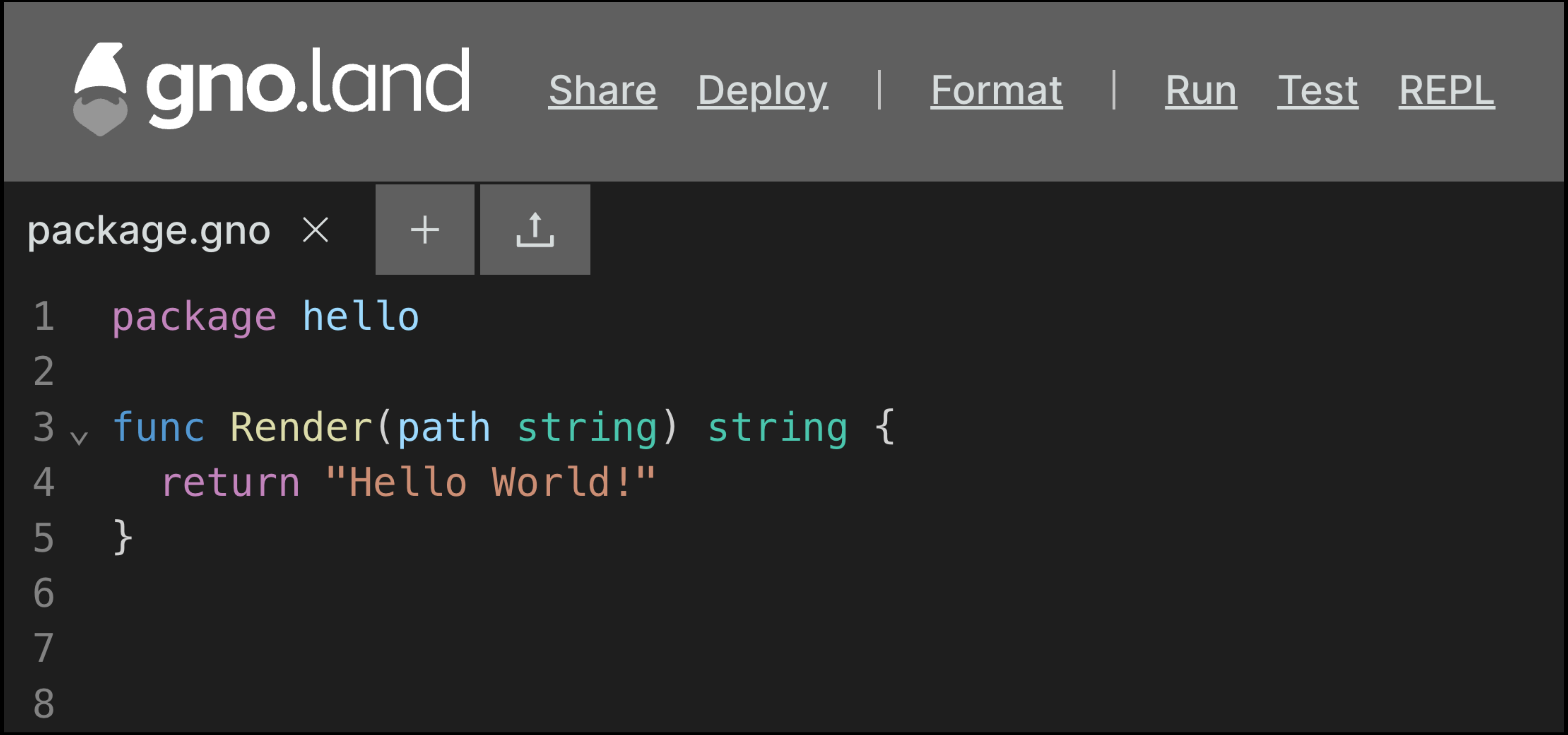


```
{
  transactions(filter: {message:
    block_height
    gas_used
    messages {
      route
      typeUrl
      value {
        __typename
        ... on MsgCall {
          caller
          pkg_path
          func
          args
        }
      }
    }
  }
}
```

```
{
  "data": {
    "transactions": [
      {
        "block_height": 64,
        "gas_used": 1593136,
        "messages": [
          {
            "route": "vm",
            "typeUrl": "exec",
            "value": {
              "__typename": "MsgCall",
              "caller":
"g1e6gxg5tvc55mwsn7t7dymmlasratv7mkv0rap2",
              "pkg_path":
"gno.land/r/demo/users",
              "func": "Register",
              "args": [
```

indexer.test4.gno.land

Gno Playground



```go
package hello

func Render(path string) string {
	return "Hello World!"
}
```

play.gno.land

Test4 launched July 2024!

Check it out at test4.gno.land!

Ecosystem

adena

gno scan

Gno Native Kit

Flippando

GnoSwap

GNO JS/TS Client

TERITORI

gno.studio

Tendermint2 JS/TS Client

Gno.land Go Client

GnoChess *BETA*

Chess: The Gnolang Way

dSocial

gnokey mobile

Consensus

- After working on PoS for years, we wanted to build something better, more secure
- PoS can be broken - with money
- gno.land will not be secured by PoS
- We are in the process of designing gno.land’s consensus mechanism
- More resistant to subversion, while rewarding project contributors fairly
- We want the chain to be owned by a DAO; and the code of the DAO to be shipped as an integral part of the chain
- Currently, Test4 is secured by 7 validator nodes, all of which are active and prominent contributors to gno.land

Call for Contributions

- We are looking for early adopters to contribute to gno.land.
- Be among the first to build useful dApps, libraries and work on protocol level problems and make an impact on the future of gno.land.

gno.land ecosystem grants program

CONTRIBUTING.md

Thanks!

gno.land

GitHub

Discord