Introduction to

gno.land

Writing smart contracts in Go

April 10th, 2025
Belgrade, Serbia

Who am I?

- Lav Leon Hudak
- DevRel Engineer @ gno.land
- 3.5 yrs in Web3, started with Ethereum

LinkedIn

GitHub

Today’s agenda

- What is gno.land & Gno?
- Short intro to Golang
- Transitioning to Gno
- Live Coding Session
- Networking & drinks!

What is gno.land ?

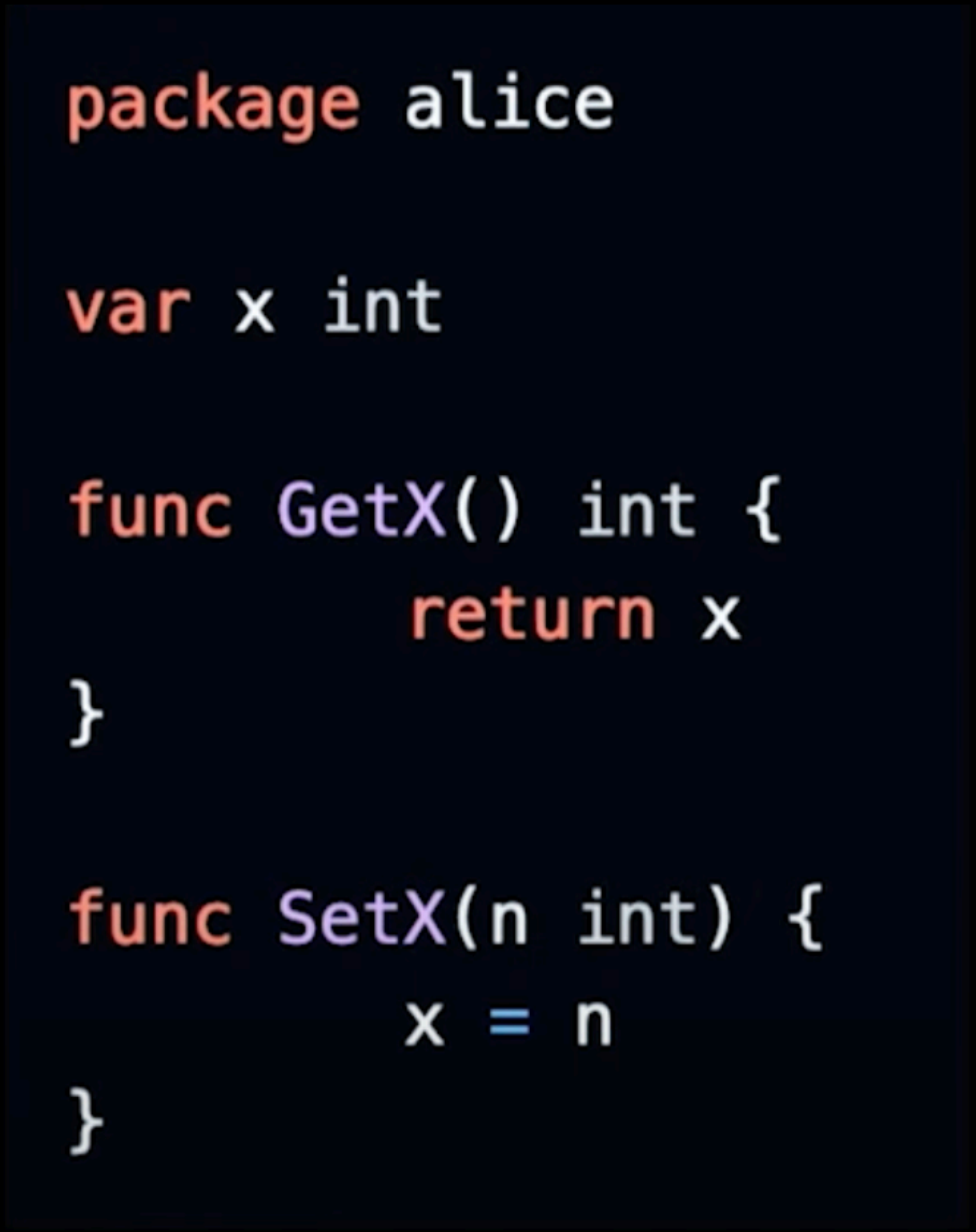
- Founded by Jae Kwon
- A new blockchain, running a custom virtual machine- the GnoVM
- Allows for writing smart contracts in Gno, an interpreted and fully deterministic version of Go

Why Go as a base?

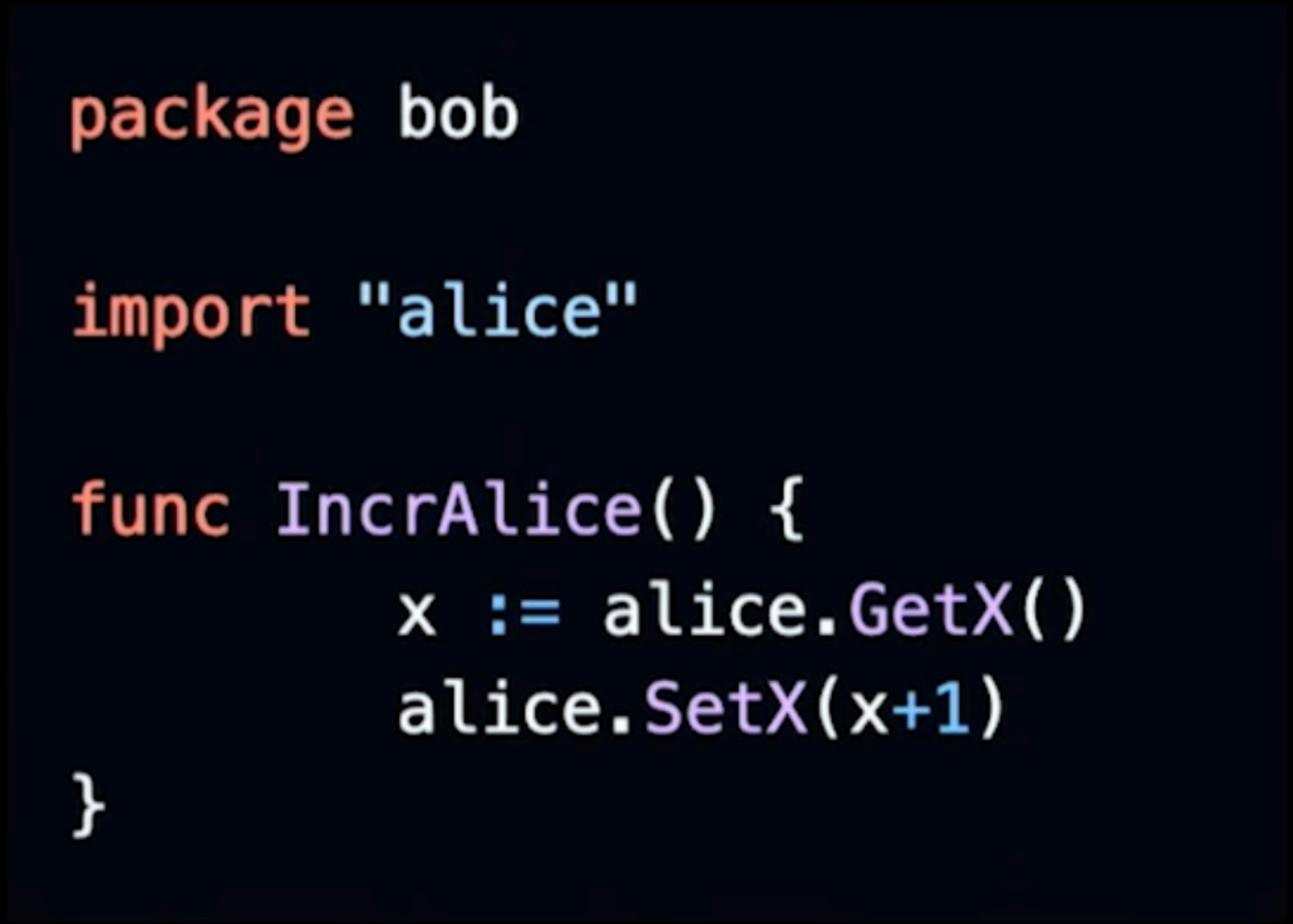
- Go is a simple and straightforward language with a minimal learning curve
- Go has a large developer community, and lots of readily available resources, most of which can be used 1:1 for learning Gno
- Solid collection of performant, well-known standard libraries
- Gives lots of power to developers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

Let’s try Go

Gno



```
package alice

var x int

func GetX() int {
        return x
}

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



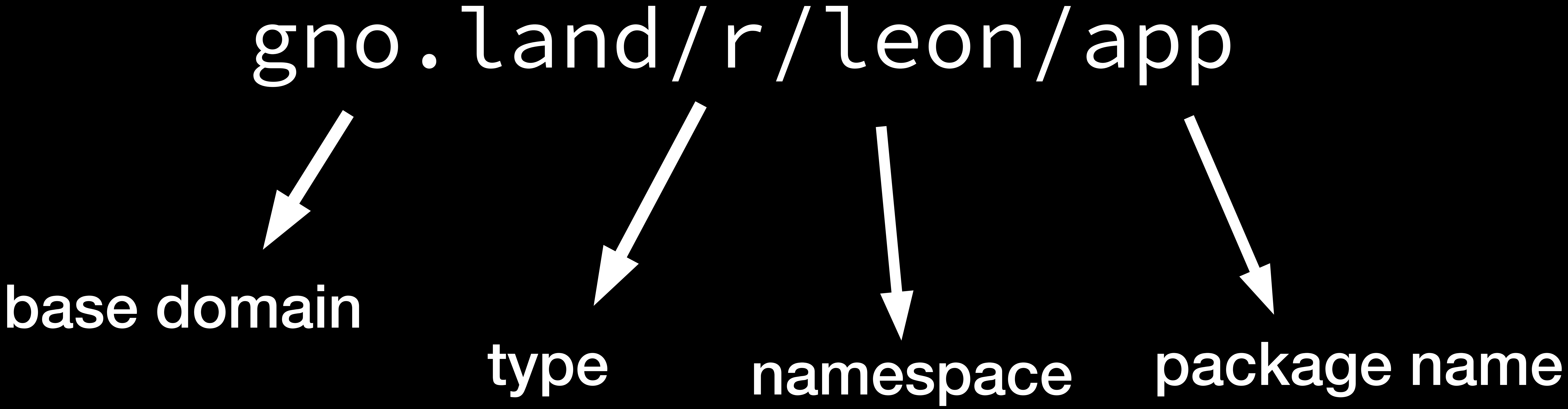
```
package bob

import "alice"

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

More on Gno…

- Modeled after Go 1.18
- Interpreted instead of compiled
- Currently does not support all go features, such as generics & goroutines
- Unlike Ethereum, 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
```

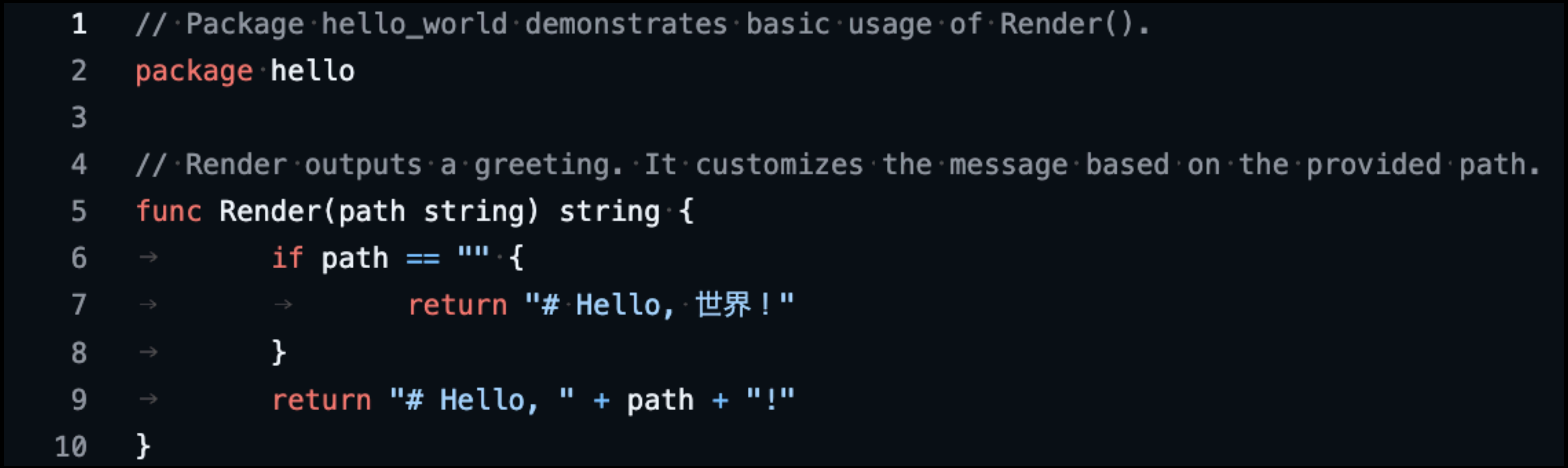
base domain

type

namespace

package name

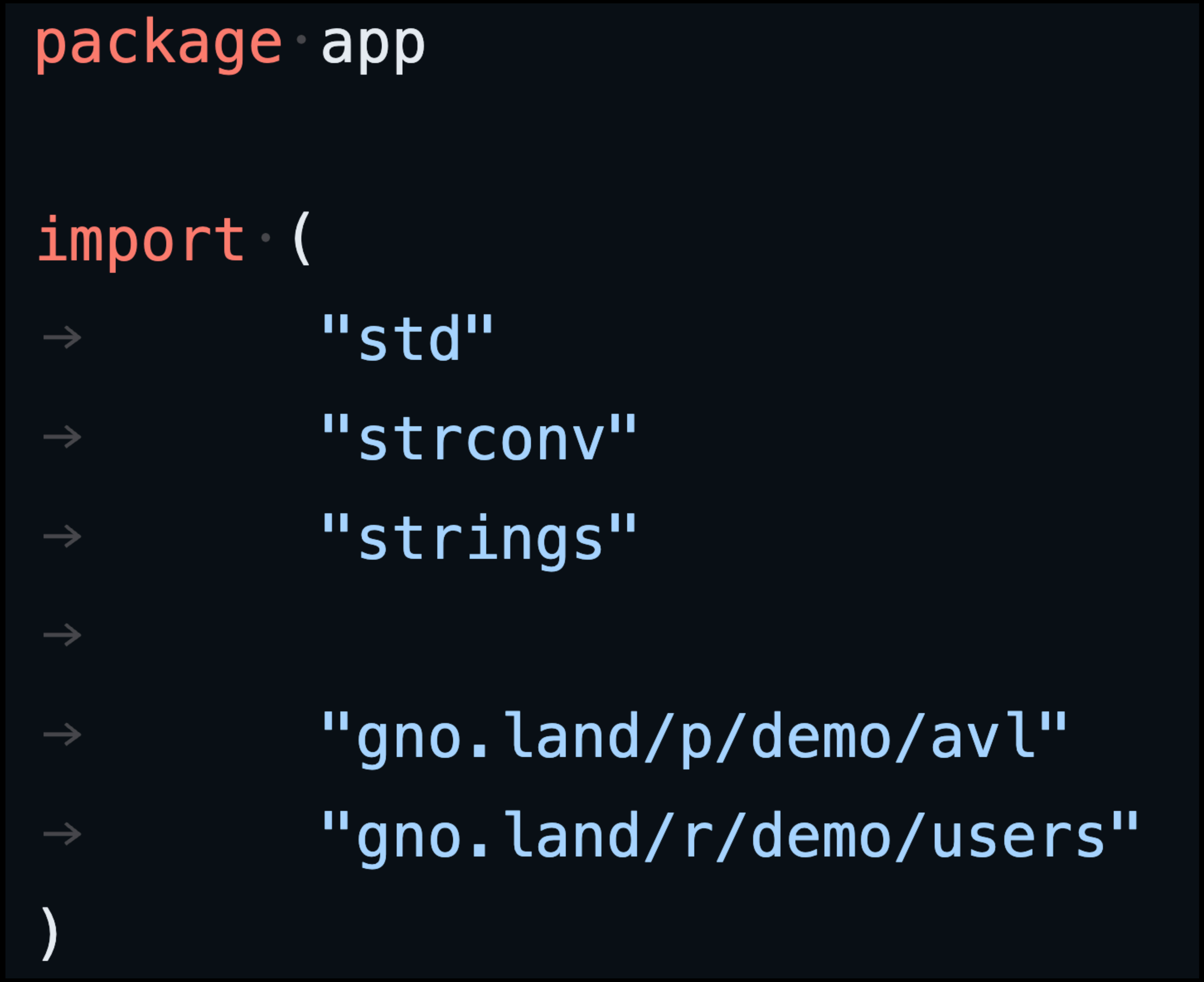
hello_world.gno



```go
// Package hello_world demonstrates basic usage of Render().
package hello

// Render outputs a greeting. It customizes the message based on the provided path.
func Render(path string) string {
	if path == "" {
		return "# Hello, 世界！"
	}
	return "# Hello, " + path + "!"
}
```

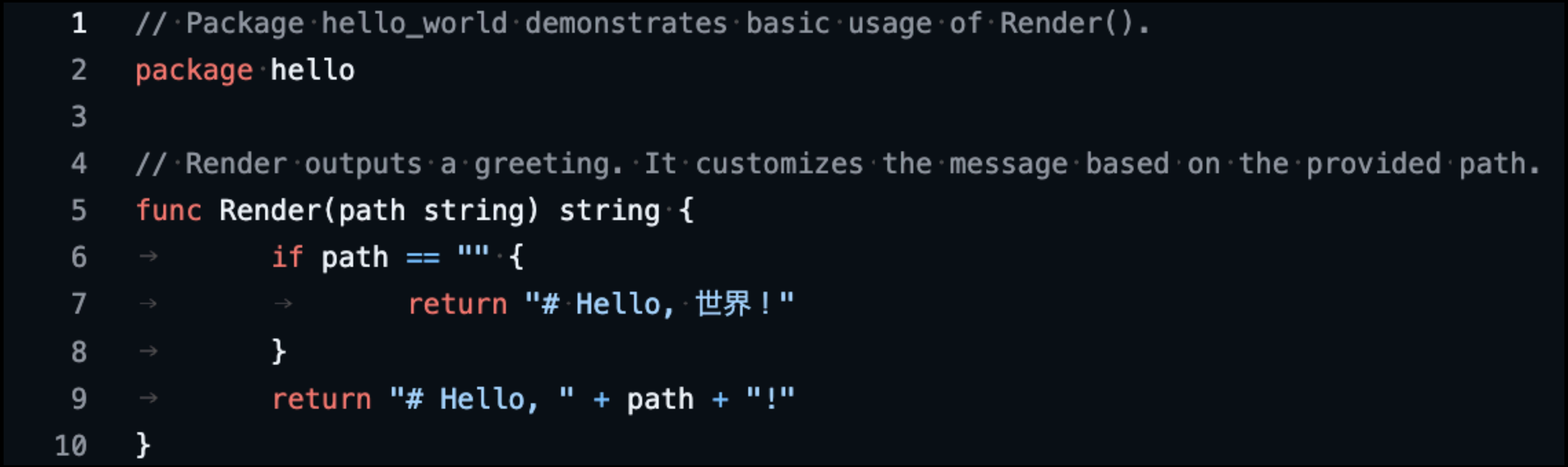
Gno code organization

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



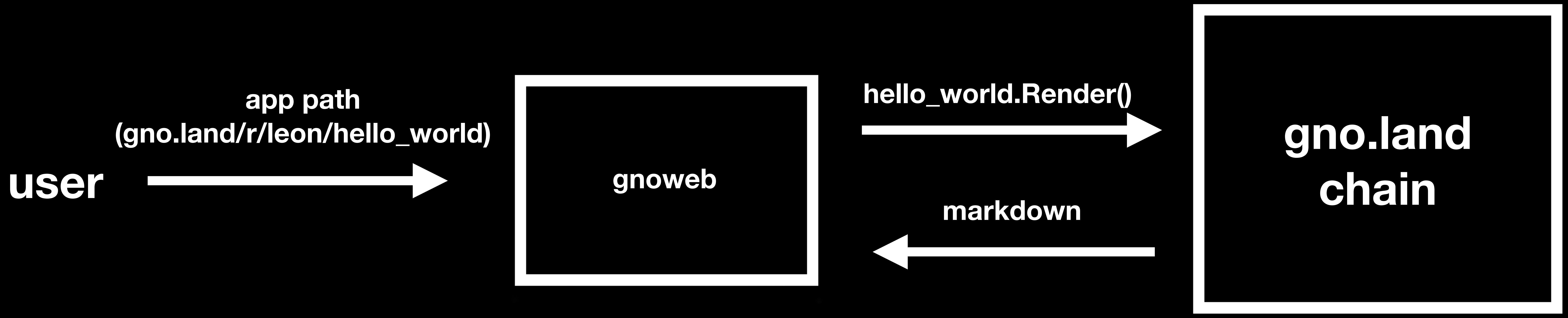
```
package app

import (
	"std"
	"strconv"
	"strings"

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

hello_world.gno



```go
// Package hello_world demonstrates basic usage of Render().
package hello

// Render outputs a greeting. It customizes the message based on the provided path.
func Render(path string) string {
	if path == "" {
		return "# Hello, 世界！"
	}
	return "# Hello, " + path + "!"
}
```

gnoweb & Render



user

app path (gno.land/r/leon/hello_world)

gnoweb

hello_world.Render()

markdown

gno.land chain

Let’s code!

Please take 2 minutes to provide feedback about today’s workshop!

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.

Linktree (Student program, Grants, etc.)

Thanks!

gno.land

GitHub

Discord