

Building Apps that work Anywhere

Andrew Williams - 26 March 2025

About Me

- Software Engineer, Author, Entrepreneur
- Open source contributor
- Founder of Fyne project
- Go developer since 2018
- CEO Fyne Labs



fyne
labs

How Go makes GUI simple

- Write once, run anywhere
- Apps that just work, do not require libraries or setup
- Native performance, without duplicating code
- Lower barrier of entry to building GUI apps
- Modern language standards and techniques
- Promote good engineering principles too

Fyne Project

"

Fyne aims to be the simplest toolkit for
developing beautiful and usable
native graphical applications
for desktop, mobile and beyond

"

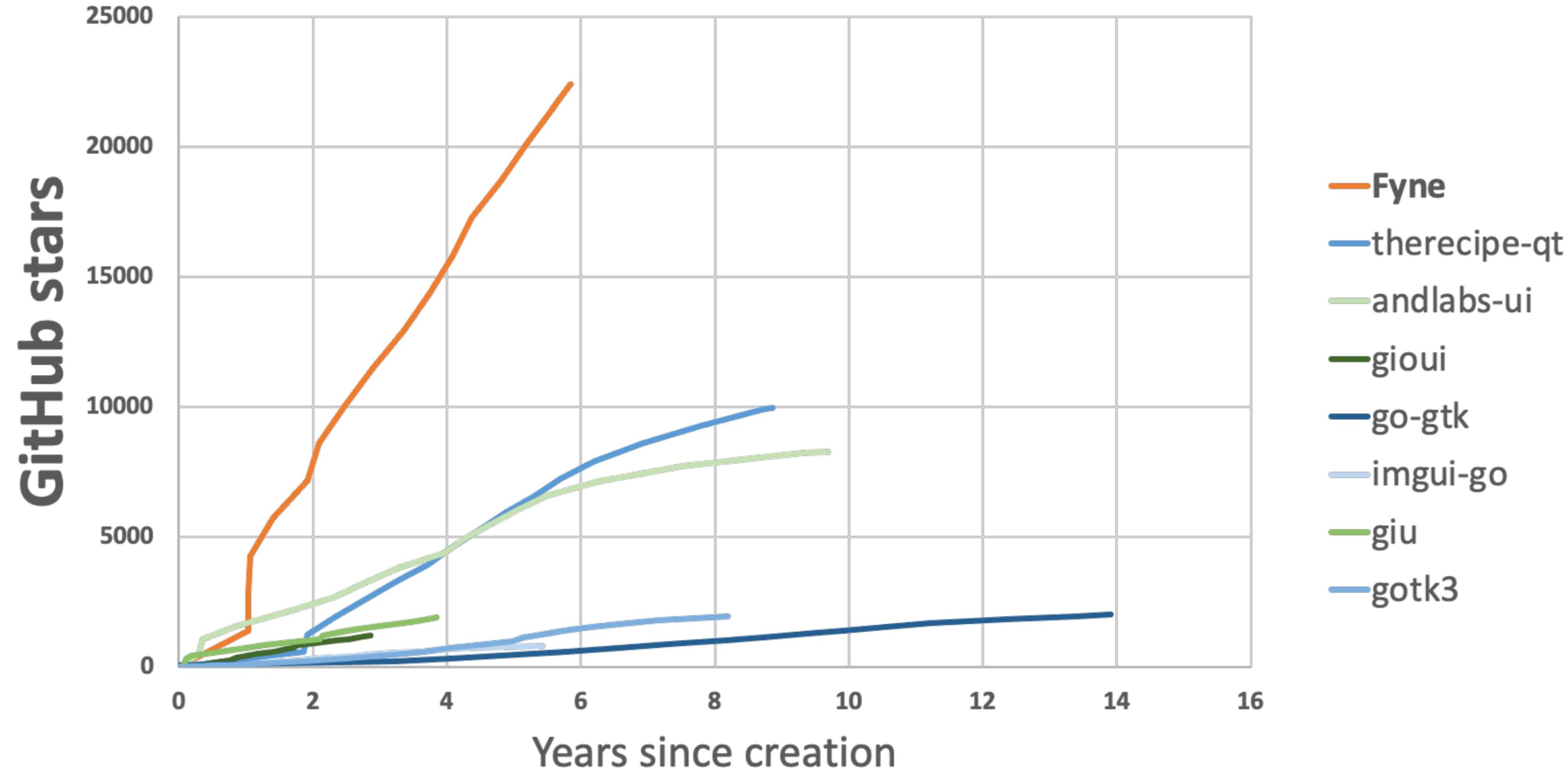


fyne

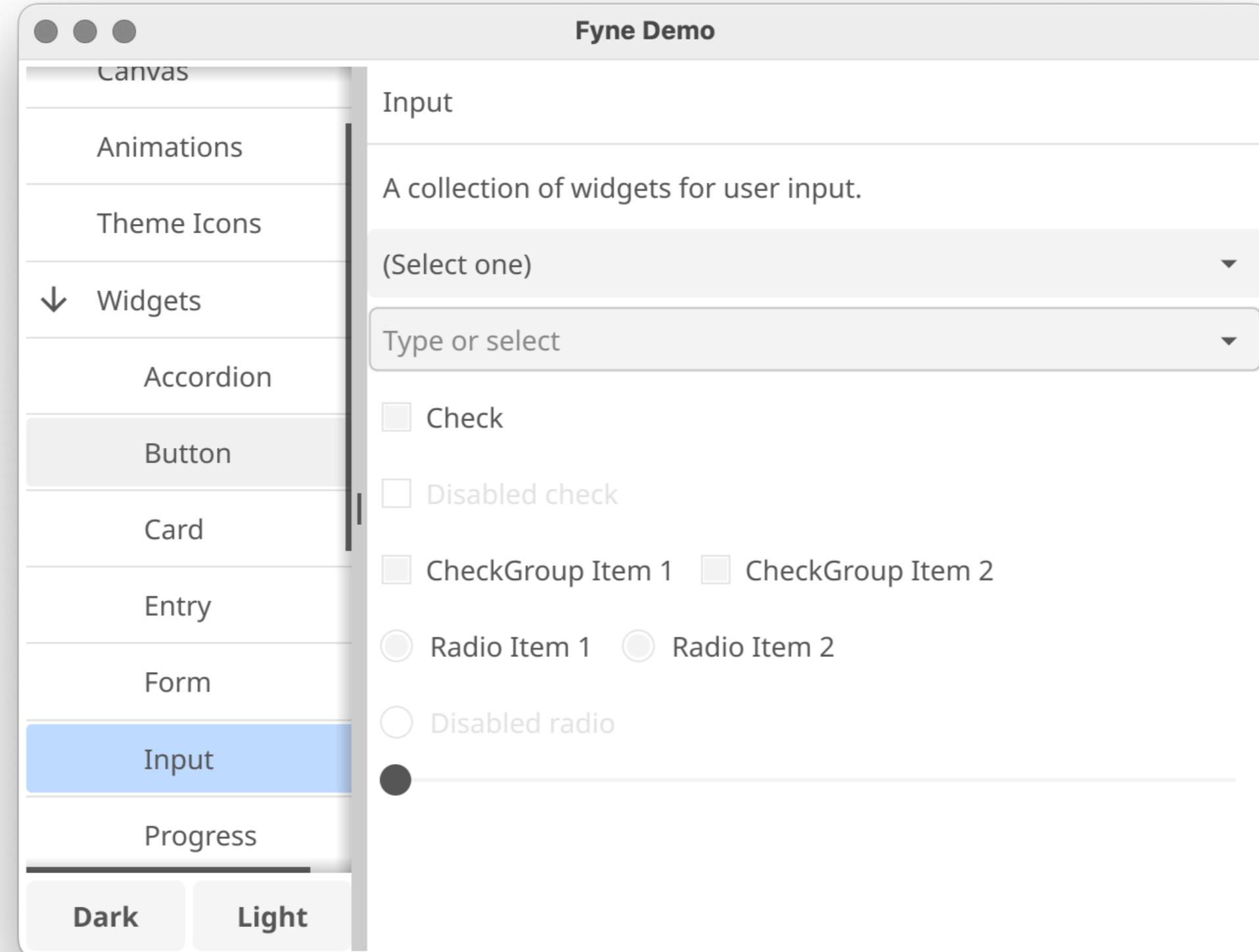
Fyne Stats

- 7 years old
- Most popular GUI toolkit for Go
- Ranked 5th of all GUI tools by @OSSInsight
- Over 26'000 GitHub stars!
- Community of >2500 on Slack, Discord, Matrix
- 15% popularity of Flutter, 20% of React Native

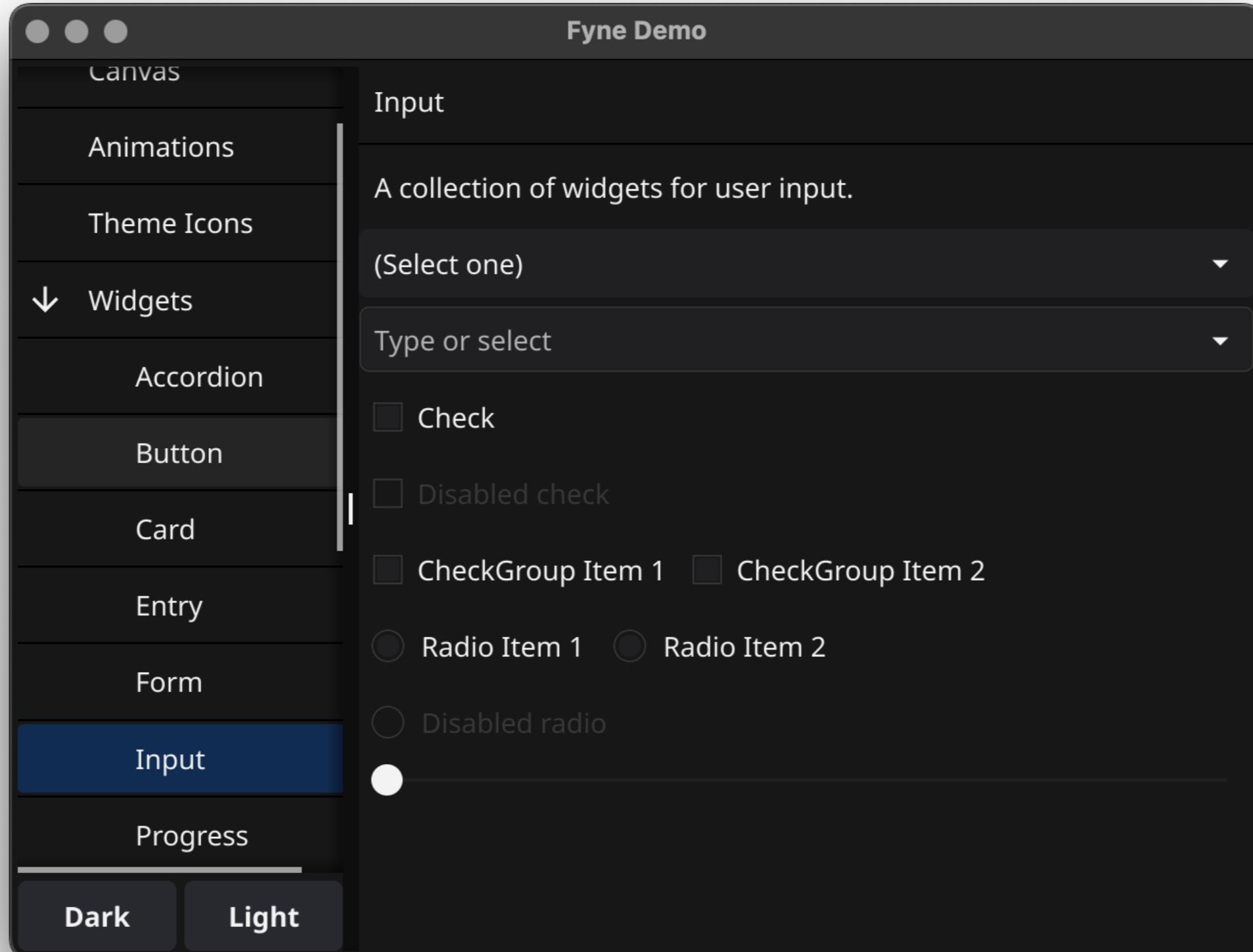
Fyne Stats



Screenshot



Screenshot (dark)



Prerequisites

- Install Go (≥ 1.19)
- Set up gcc/clang

<https://docs.fyne.io/started/>

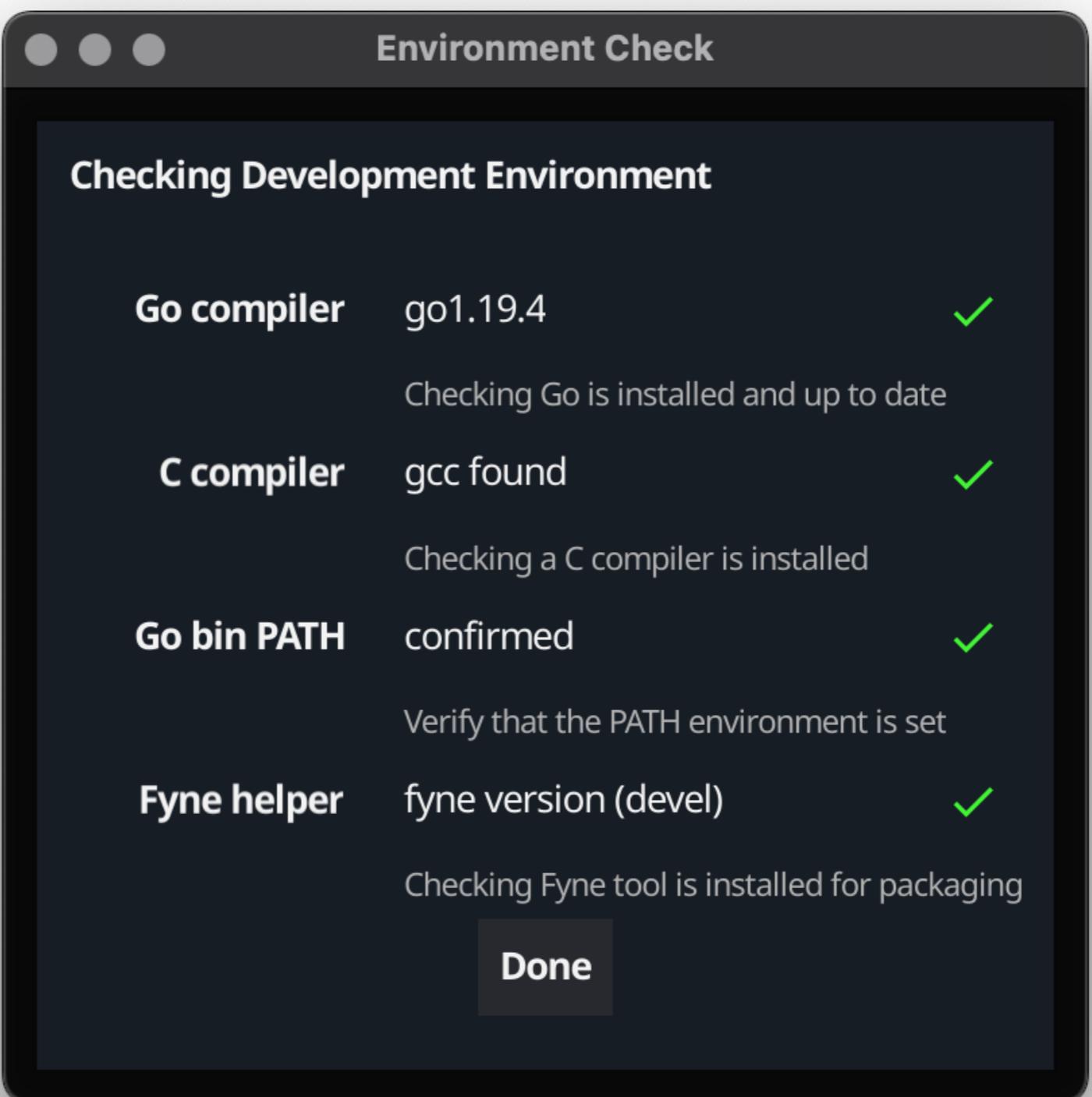
Prerequisites

- Install Go (≥ 1.19)

- Set up gcc/clang

<https://docs.fyne.io/started/>

- Double check with "Fyne Setup"



Build our first app

```
$ mkdir myproject; cd myproject
```

```
$ go mod init myproject
```

```
$ go get fyne.io/fyne/v2@latest
```

```
$ vim hello.go
```

```
$ go run .
```

Code: Hello World

```
1 package main
2
3 import (
4     "fyne.io/fyne/v2/app"
5     "fyne.io/fyne/v2/widget"
6 )
7
8 func main() {
9     a := app.New()
10    w := a.NewWindow("Hello")
11
12    w.SetContent(widgetNewLabel("Hello Bristol!"))
13    w.ShowAndRun()
14 }
```

Compiling for other targets

```
$ go install fyne.io/fyne/v2/cmd/fyne@latest
```

```
$ fyne install
```

```
$ fyne package -os windows
```

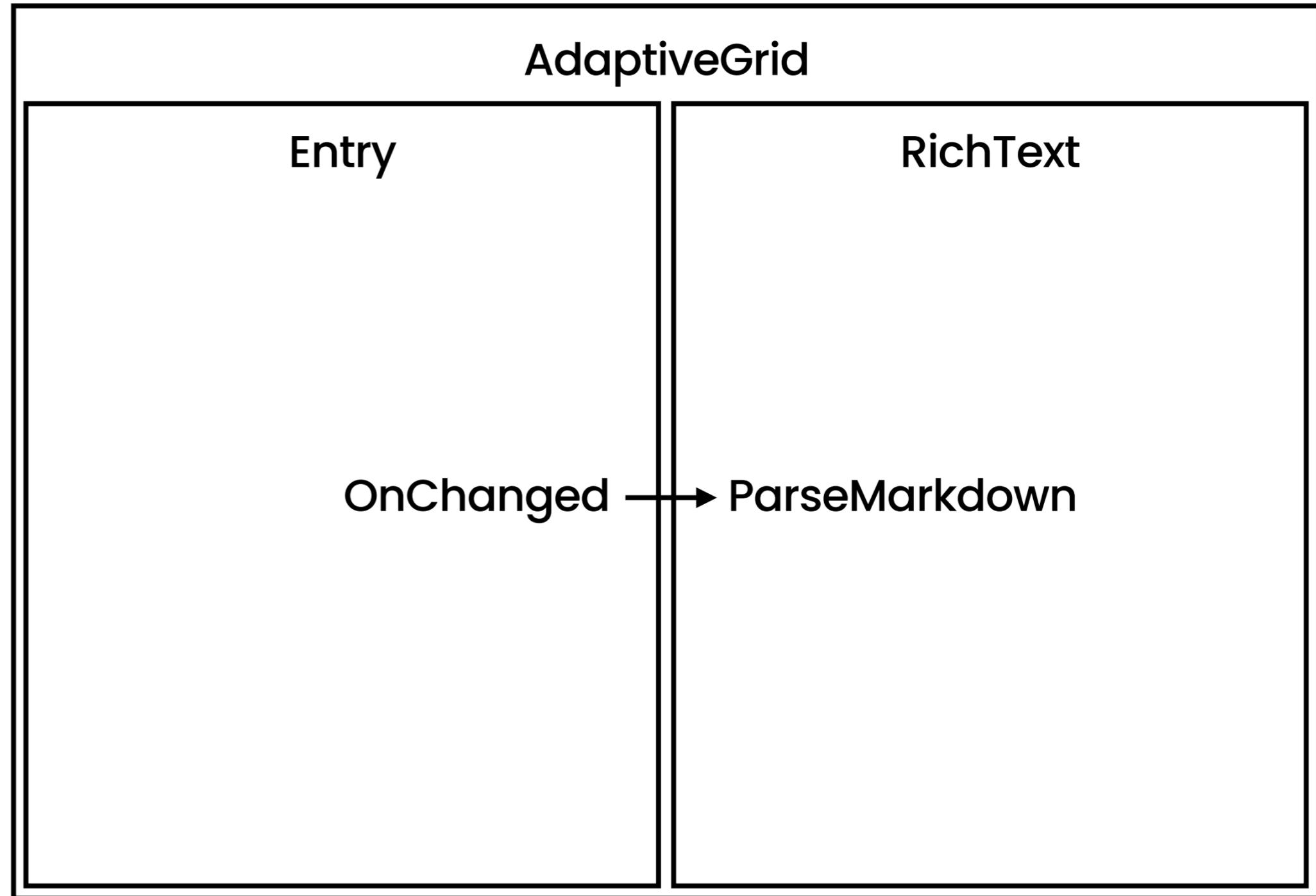
```
$ fyne package -os android -applD com.company.myapp
```

--

Also see fyne-cross tool <https://github.com/fyne-io/fyne-cross>

Let's make a Markdown editor!

- Editor widget for input
- RichText widget for output
- AdaptiveGrid container
- Update through OnChanged

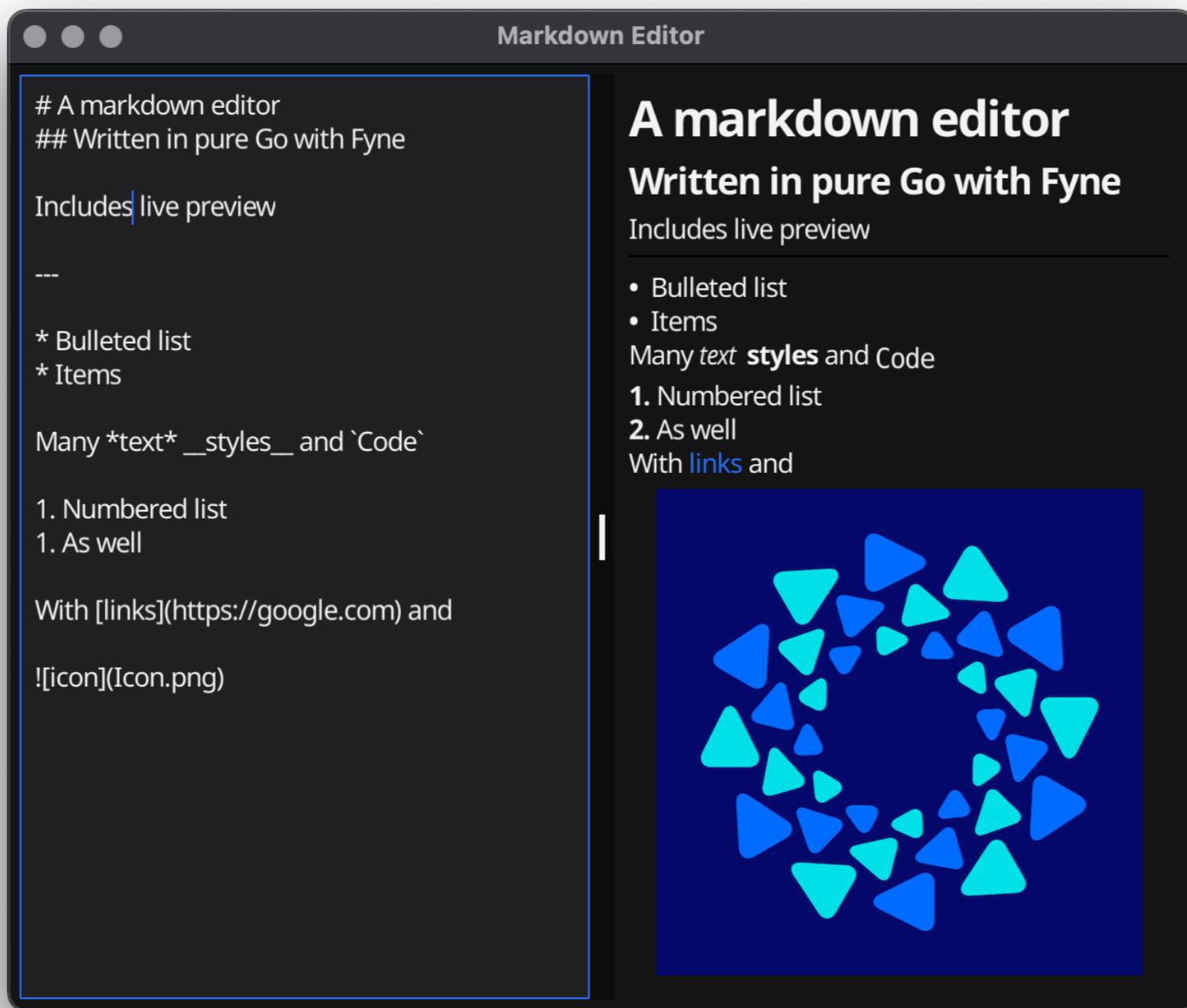


Code: Markdown Editor

```
1 package main
2
3 import (
4     "fyne.io/fyne/v2"
5     "fyne.io/fyne/v2/app"
6     "fyne.io/fyne/v2/container"
7     "fyne.io/fyne/v2/widget"
8 )
9
10 func main() {
11     a := app.New()
12     w := a.NewWindow("Markdown")
13
14     input := widget.NewMultiLineEntry()
15     output := widget.NewRichTextFromMarkdown("")
16     content := container.NewAdaptiveGrid(2, input, output)
17
18     input.OnChanged = output.ParseMarkdown
19
20     w.SetContent(content)
21     w.Resize(fyne.NewSize(320, 240))
22     w.ShowAndRun()
23 }
```

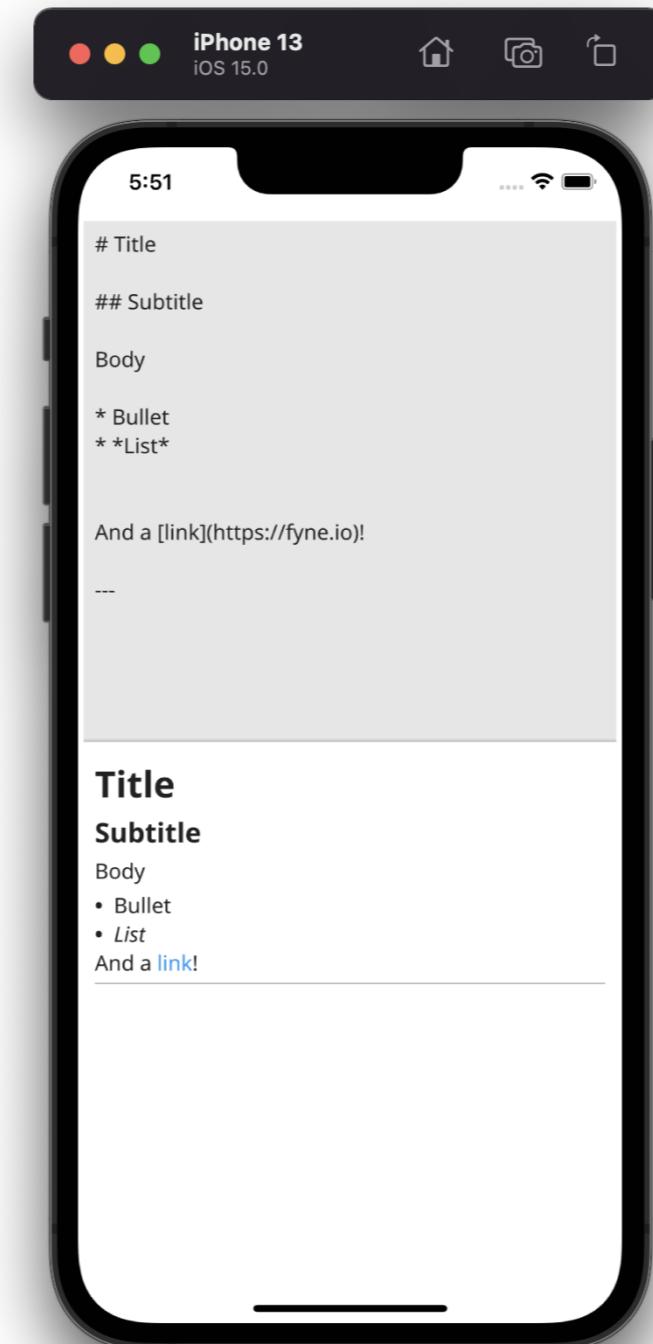
Let's make a Markdown editor!

\$ go run (or fyne package)



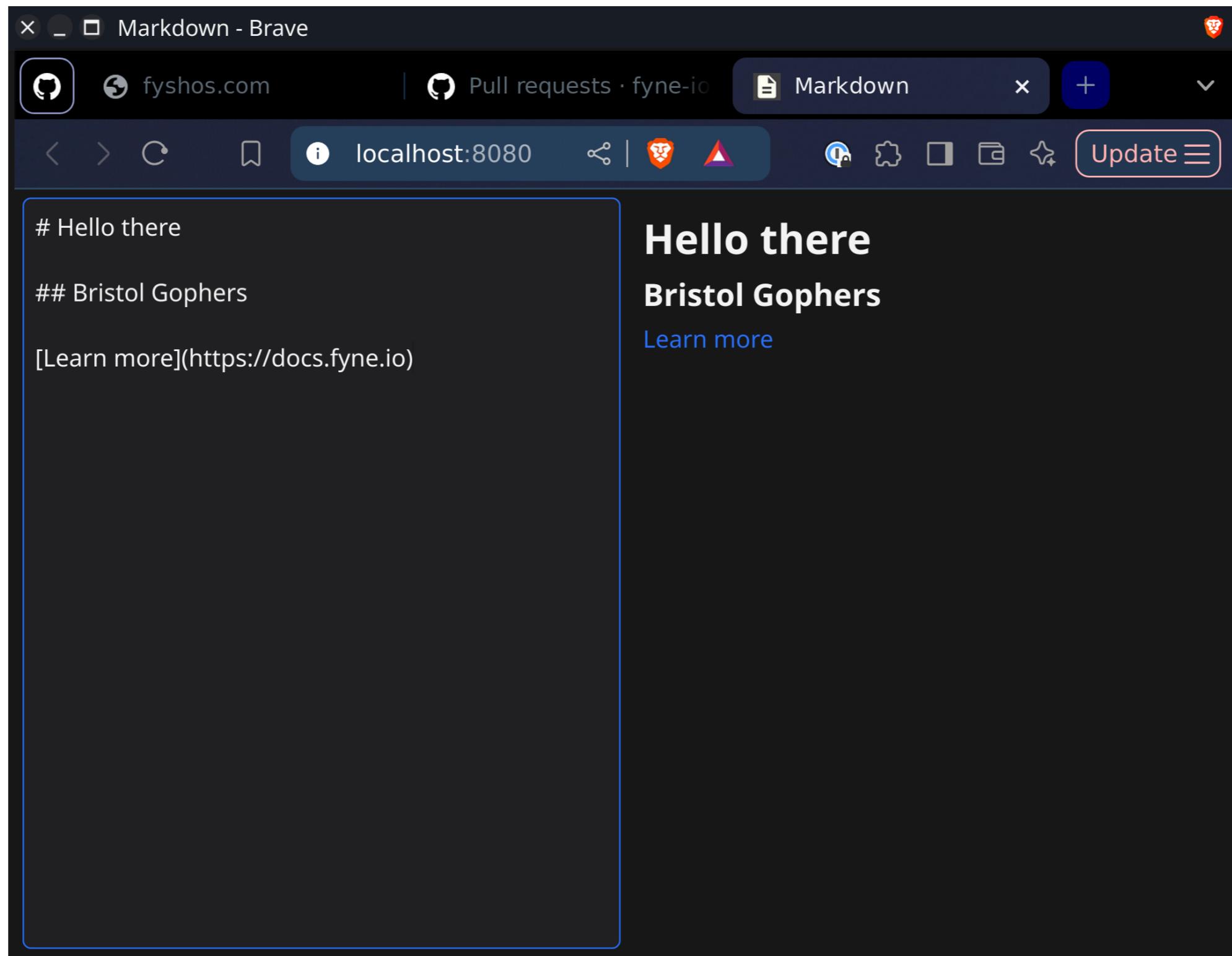
Let's make a Markdown editor!

\$ fyne install -os ios



Let's make a Markdown editor!

\$ fyne serve



Testing

```
1 package main
2
3 import (
4     "testing"
5
6     "github.com/stretchr/testify/assert"
7
8     "fyne.io/fyne/v2/test"
9     "fyne.io/fyne/v2/widget"
10 )
11
12 func TestText_Selected(t *testing.T) {
13     e := widget.NewEntry()
14     test.Type(e, "Hello")
15     assert.Equal(t, "Hello", e.Text)
16
17     test.DoubleTap(e)
18     assert.Equal(t, "Hello", e.SelectedText())
19     assert.Equal(t, 5, e.CursorColumn)
20 }
```

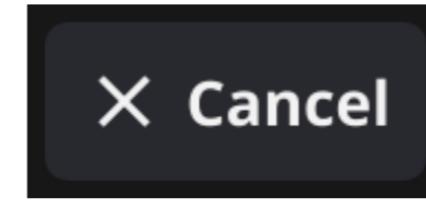
Exploring further...

Basic Widgets

```
widgetNewLabel("Text label")  
  
widget.NewButtonWithIcon("Cancel"  
    theme.CancelIcon, func() {})  
  
widget.NewProgressBar()  
  
widget.NewCard("Card Title",  
    "subtitle", content)  
  
widget.NewRichTextFromMarkdown(  
    `# RichText Heading ...`)
```

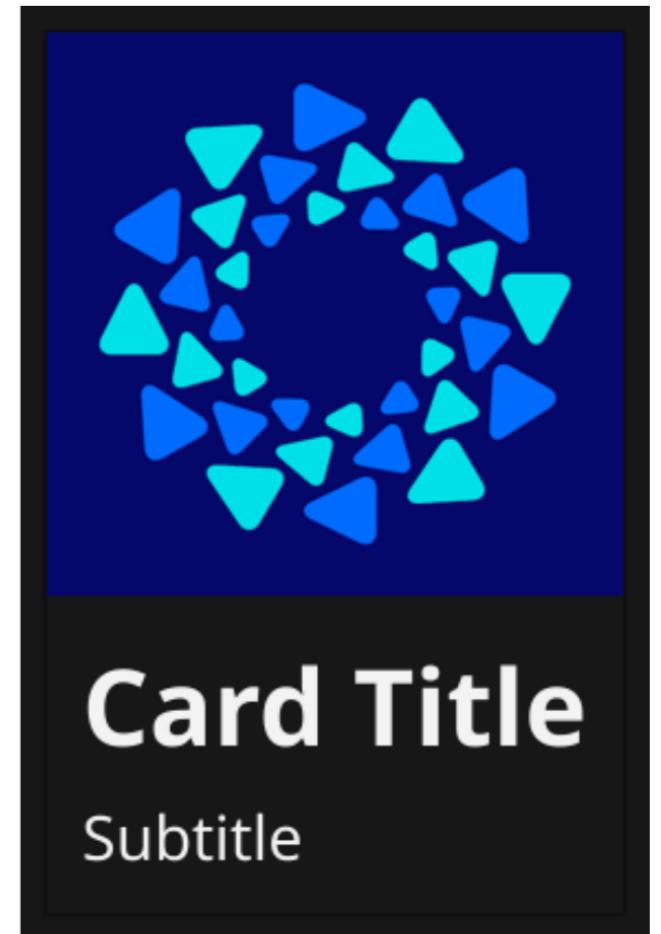
1 TextGrid
2 Content

Text label



74%

RichText Heading
A Sub Heading



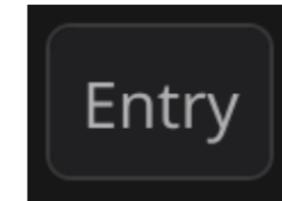
▼ A

▲ B

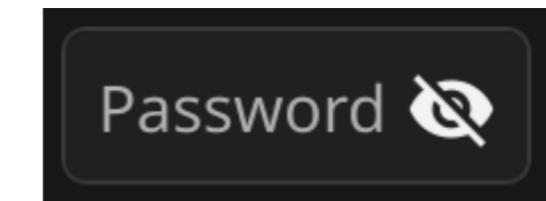
Shown item

Input Widgets

```
widget.NewEntry()
```



```
widget.NewPasswordEntry()
```



```
widget.NewSlider(0, 100)
```



```
widget.NewCheck("Check", func(bool) {})
```



Check



Item 1



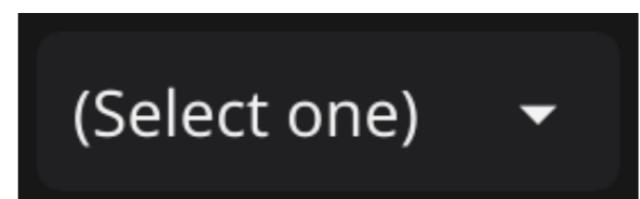
Item 2

```
widget.NewRadioGroup([]string{
```

```
    "Option 1", "Option 2"}, func(string) {})
```

```
widget.NewSelect([]string{"Option A", "Option B"},
```

```
    func(string) {})
```

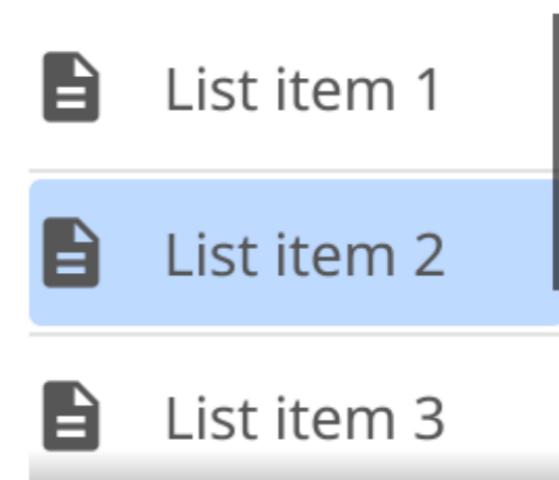
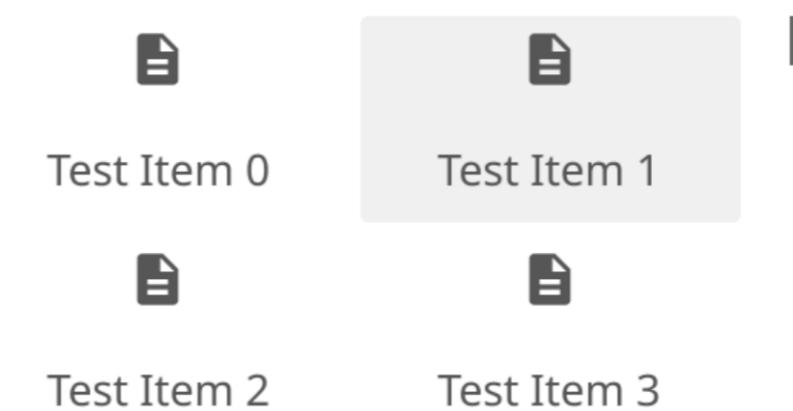
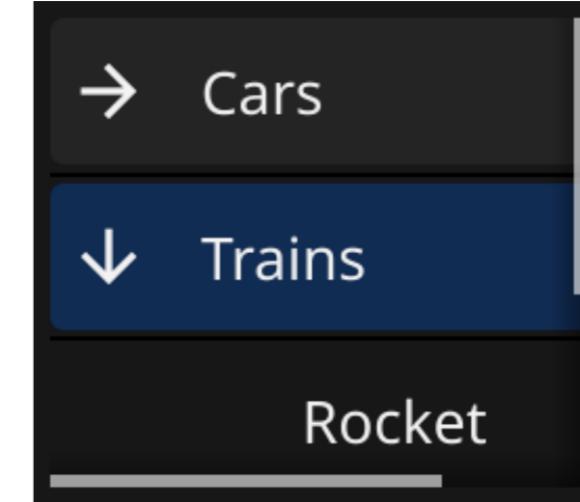


Select one or type ▾

Collection Widgets

```
widget.NewList(  
    func() int {  
        return len(data)  
    },  
    func() fyne.CanvasObject {  
        return widgetNewLabel("Template Object")  
    },  
    func(id widget.ListItemID, o fyne.CanvasObject) {  
        o.(*widget.Label).SetText(data[id])  
    },  
)
```

A	B
1 A longer cell	Cell 1, 2
2 A longer cell	Cell 2, 2



Containers

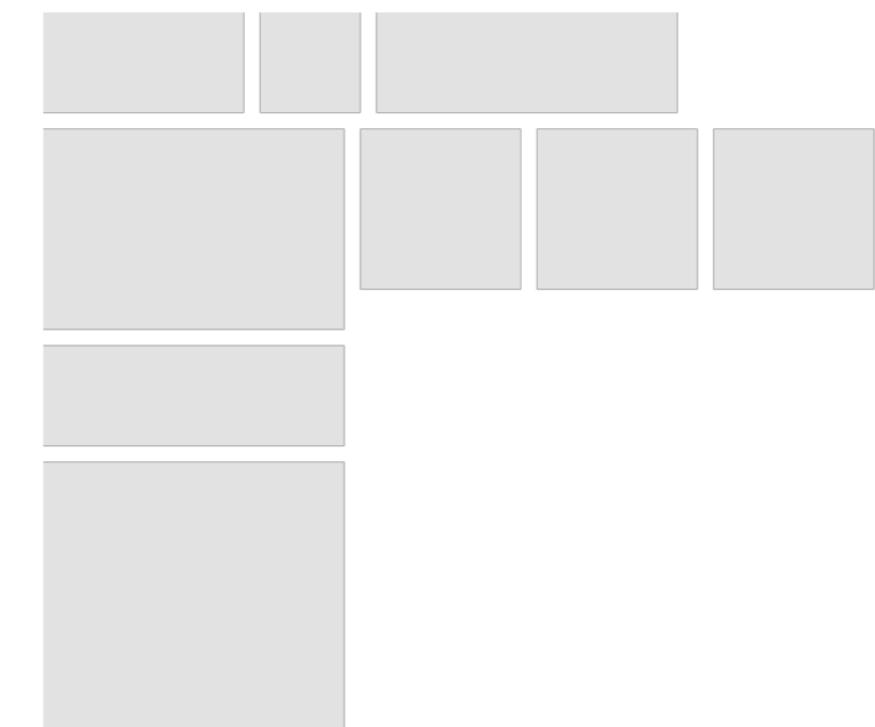
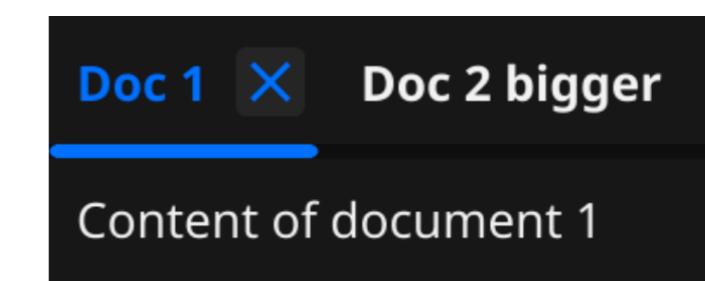
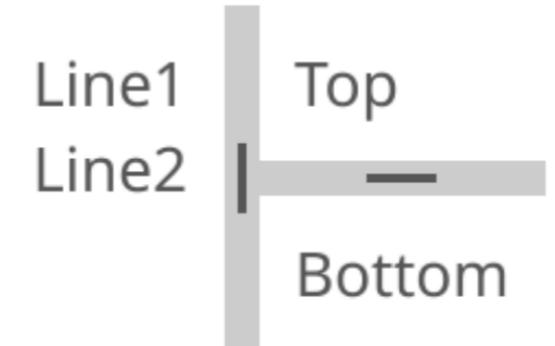
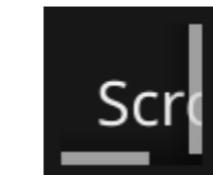
container.NewHSplit(left, right)

container.NewAppTabs(items...)

container.NewDocTabs(items...)

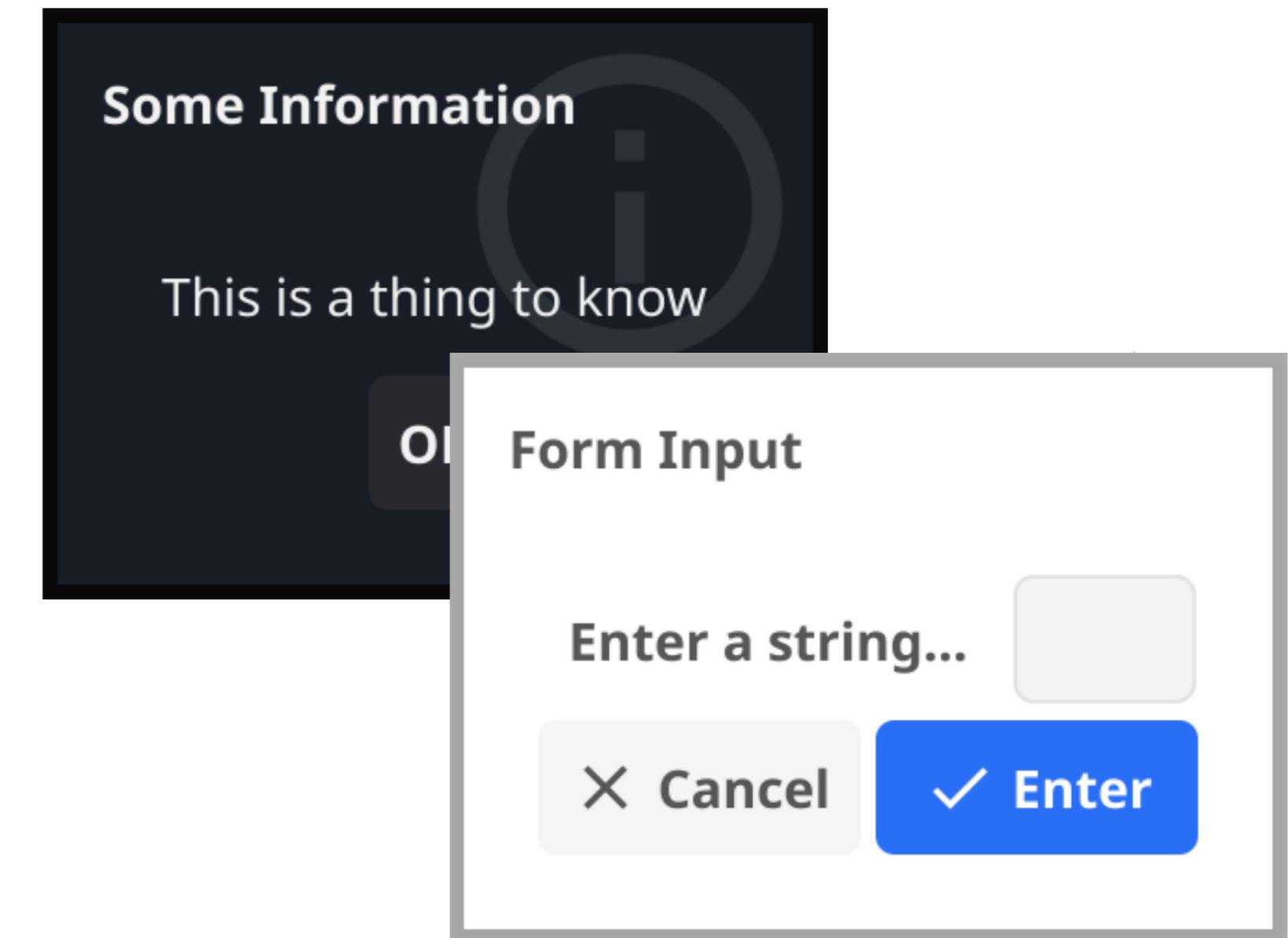
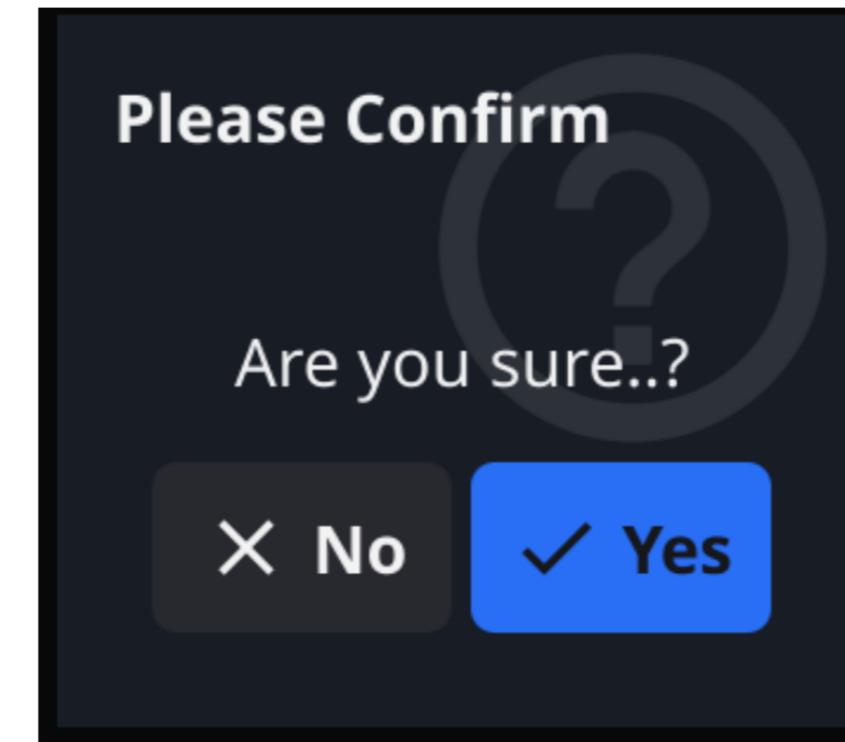
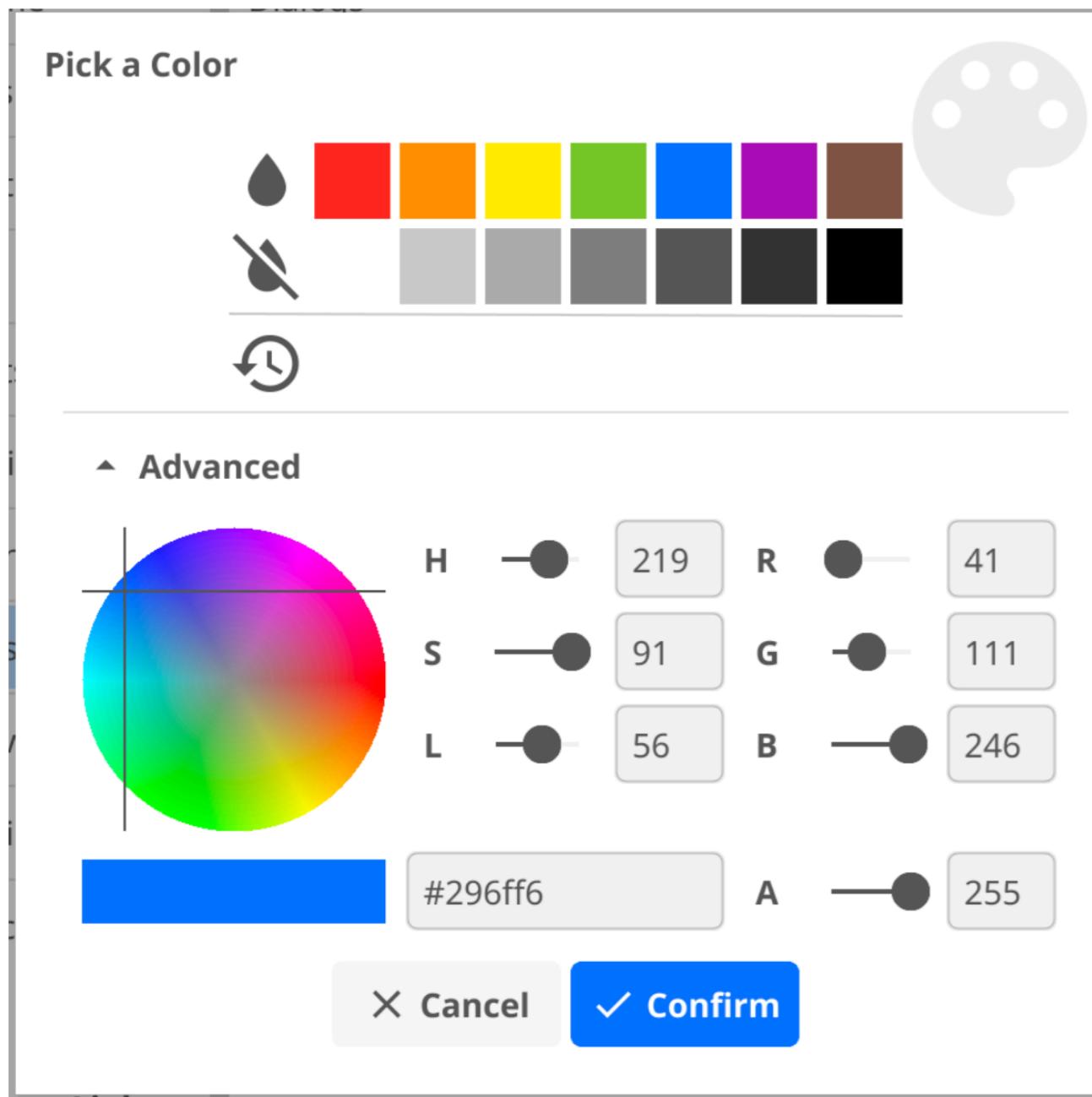
container.NewScroll(content)

container.New(layout, objects...)



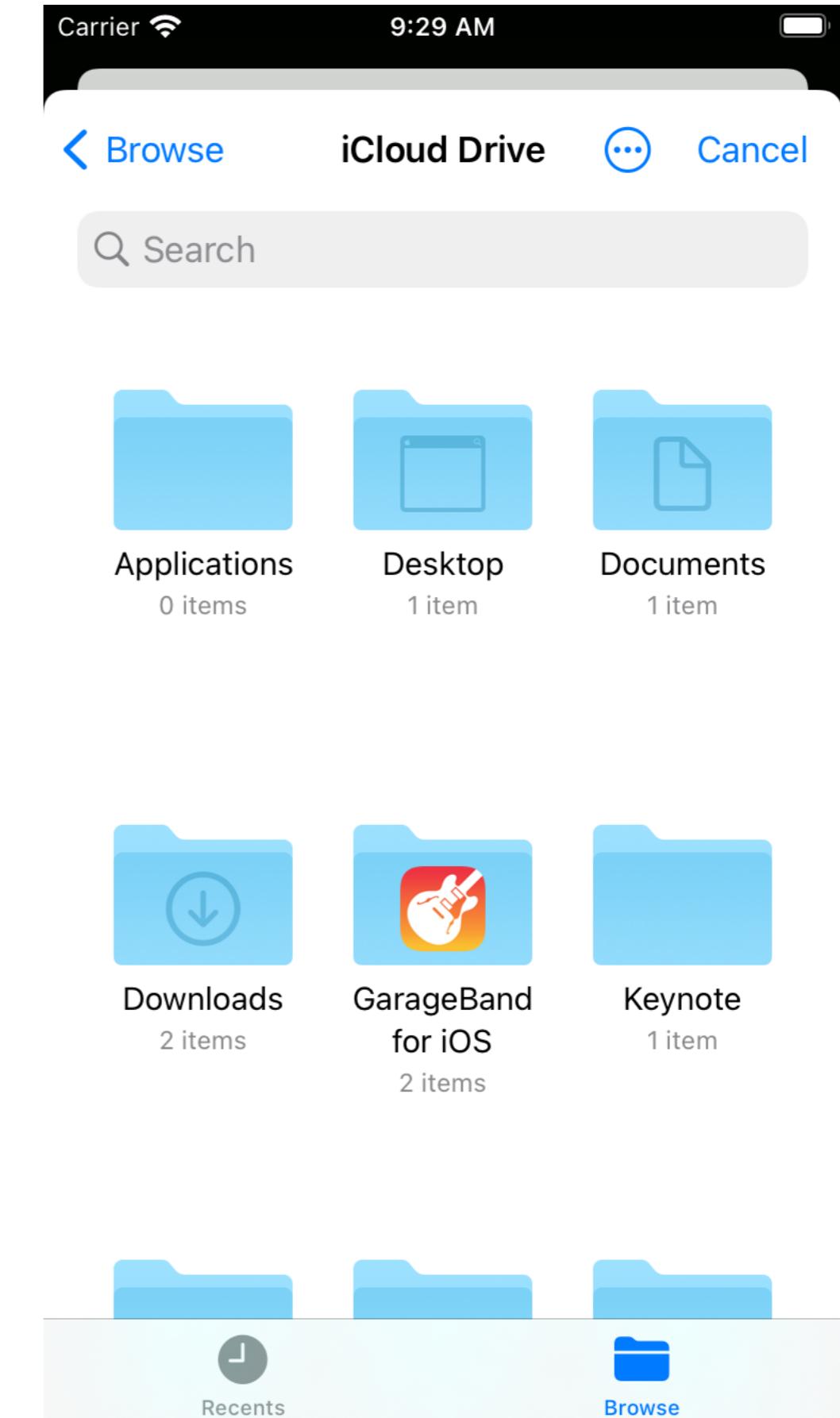
Dialogs

```
dialog.ShowConfirm("Confirmation",  
    "Are you enjoying this demo?", func(bool) {}, win)
```



File input

- dialog.ShowFileOpen
- dialog.ShowFileSave
- Use fyne.URI not path
- storage package abstraction



3rd party components too!

- Just import package and use

- Works like any widget

```
$ map := xWidget.NewMap()
```

```
$ cmdline := terminal.New()
```

<https://addons.fyne.io>



But there is more!

- Menus, menu bar
- Notifications
- System Tray
- Preferences and Documents
- Cloud storage integration



But there is EVEN MORE!

This entire presentation, and desktop, is Fyne!

But also ONE MORE THING!

Editing Fyne apps with a GUI builder!

Learn more

- Documentation: <https://docs.fyne.io>
- Videos: <https://www.youtube.com/c/fyne-io>
- Apps: <https://apps.fyne.io>
- Contribute: <https://github.com/fyne-io/fyne/>
- Sponsor! <https://fyne.io/sponsor/>
- App Builder <https://fysion.app>

Questions?

[@andydotxyz](https://twitter.com/andydotxyz) / andy@fynelabs.com