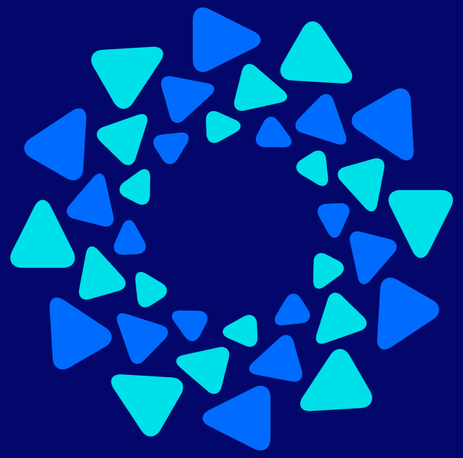


fyne

How Go Taught Me to Love Building Apps Again

Andrew Williams - Conf42

25 April 2024

About Me

- Software Engineer, Author, Tech Leader and Open Source advocate
- Core developer on Maven, Enlightenment, EFL
- Founder of the Fyne project and Go developer since 2018
- CEO Fyne Labs

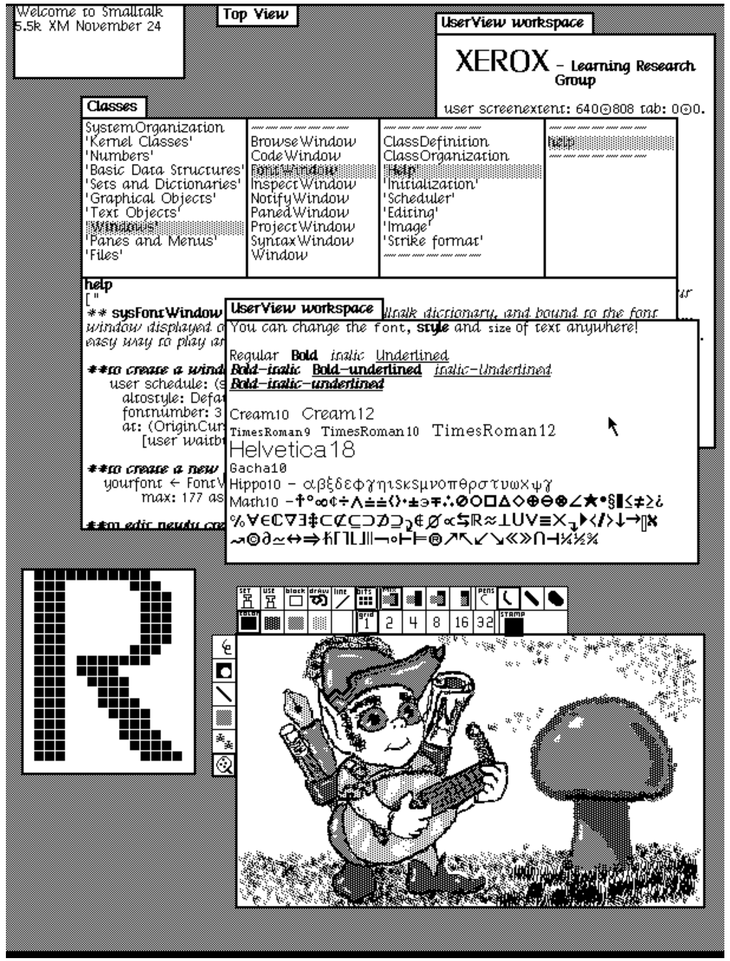
About Me

fyne labs


THE MAGIC KINGDOM
WALT DISNEY WORLD
WALT DISNEY WORLD RAILROAD

Background: Frustrations of GUI development

- Mostly C language heritage
  - Memory management
  - Thread handling
  - Lacking web services (or even strings / unicode!)
- Modern alternatives often bindings over old
- Complex to learn
- Toolkits either massive or complex installs
- Modern alternatives only for some platforms



Go: A delightful alternative

fyne labs

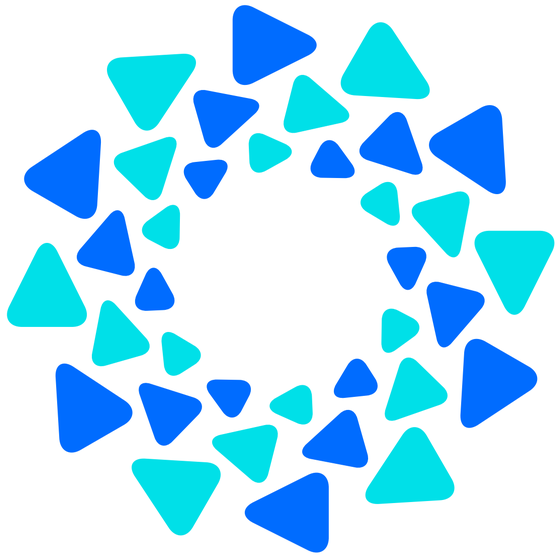
- Write once, run anywhere
- Apps that just work, no libraries or setup
- Native performance, on all platforms
- Modern language standards and techniques
- Lower barrier of entry to building GUI apps

Fyne: A new hope for app developers

An open source native app toolkit?

fyne labs

- A fresh start 😀
- Easy to understand, simple to contribute
- Platform independent native apps
- Light-weight with great performance
- Community driven

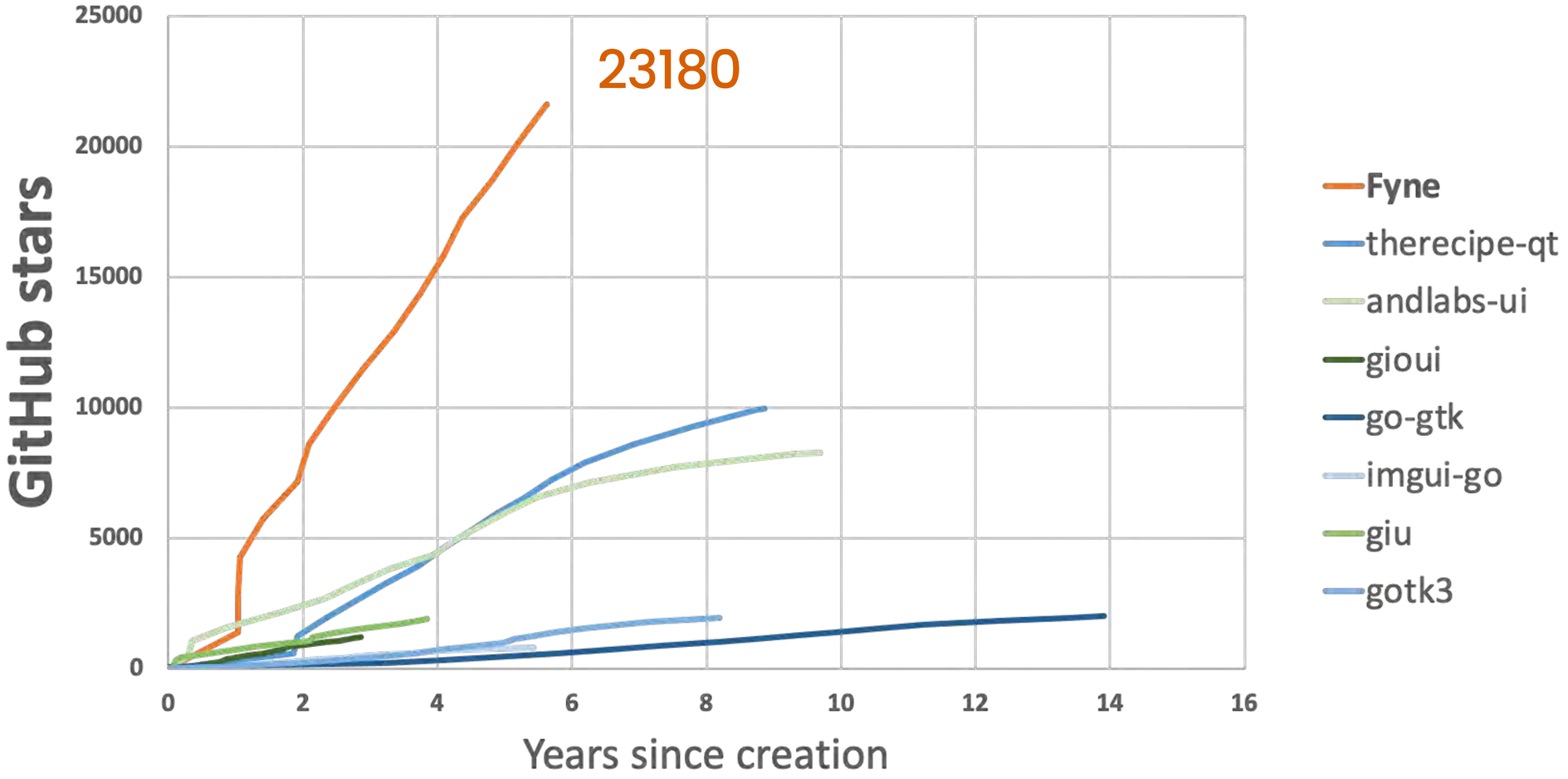


Fyne Project

fyne labs

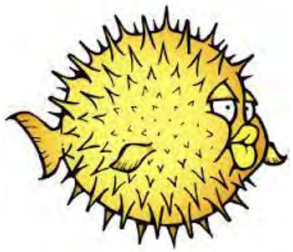
Fyne aims to be the simplest toolkit for developing beautiful and user-friendly native graphical applications for desktop, mobile and beyond.

Fyne Toolkit Stargazers



Fyne Runs on...

- Desktop & Laptop computers:
  Windows, macOS, Linux, BSDs
- Mobile Devices
  iOS, Android
- Web browsers
  WASM / Javascript
- Embedded devices



Your First App in Just Minutes!

Building our first - Prerequisites

fyne labs

- Working Go compiler (>= 1.17)
- Installed C compiler (gcc, clang etc)
  - Linux / BSD install gcc package
  - macOS (& iOS) via Xcode
  - Windows through MSYS2 & mingw
  - Android requires SDK & NDK

Building our first app - Setup


```
$ mkdir project; cd project
$ go mod init project
$ go get fyne.io/fyne/v2@latest
$ vim ui.go
```

Building our first app - Code

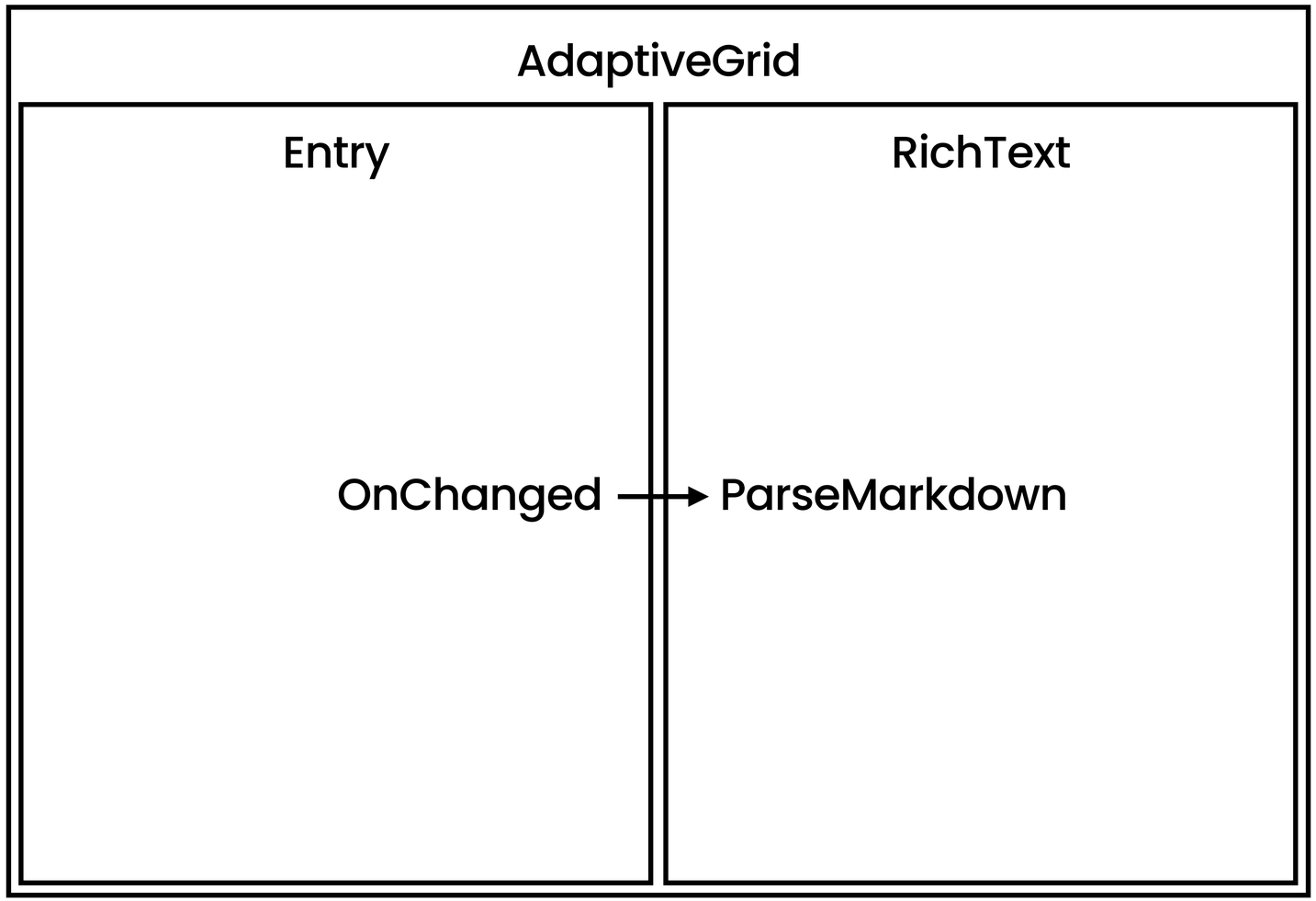
fyne labs

```go
package main

import (
	"fyne.io/fyne/v2/app"
	"fyne.io/fyne/v2/widget"
)

func main() {
    a := app.New()
    w := a.NewWindow("Hello")

    w.SetContent(widget.NewLabel("Hello Fyne!"))

    w.ShowAndRun()
}
```

Building our first app - Run

```
$ mkdir project; cd project
$ go mod init project
$ go get fyne.io/fyne/v2@latest
$ vim ui.go
$ go mod tidy
$ go run .
```

Markdown Editor - Layout

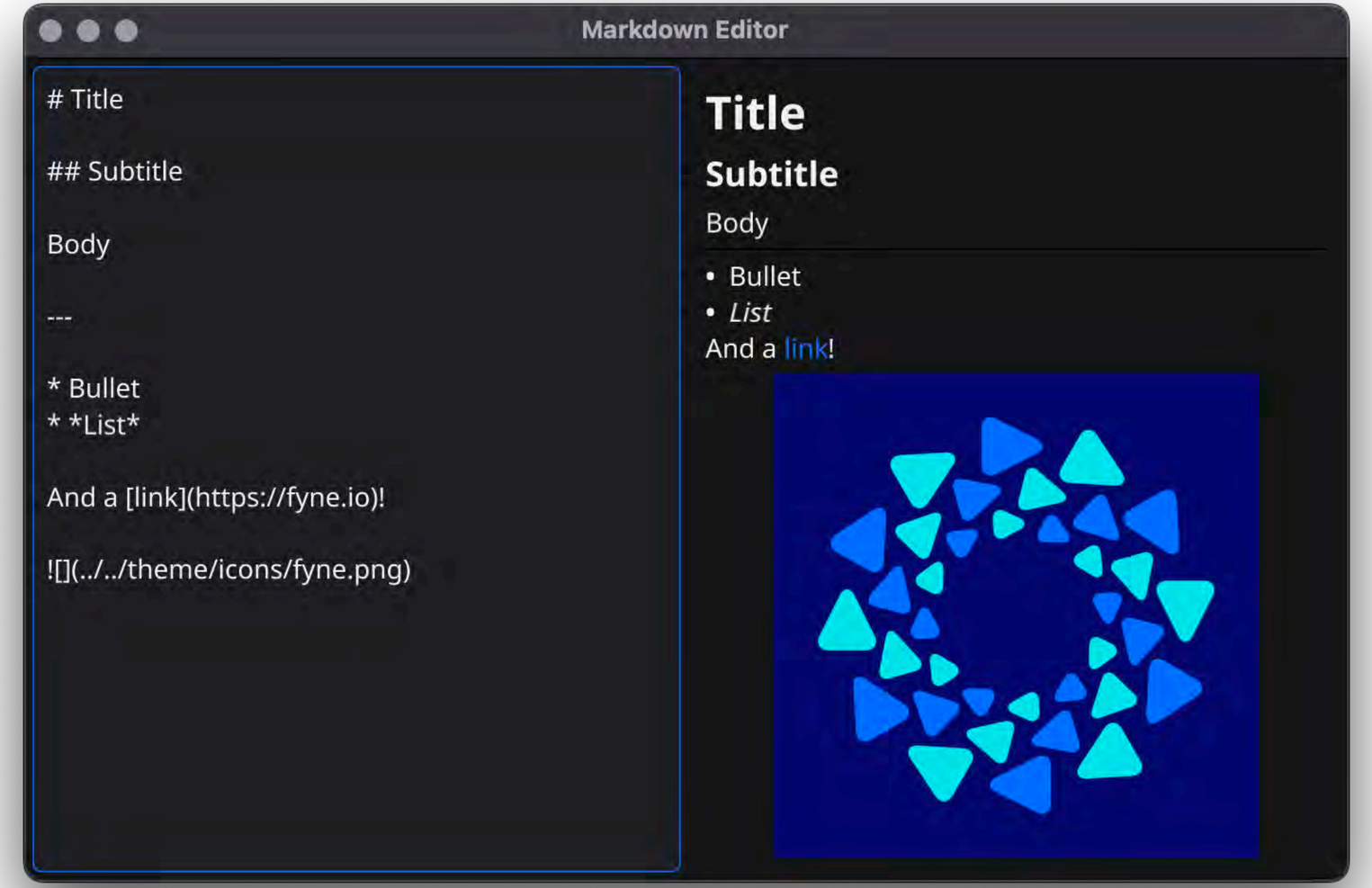


Markdown Editor - Code

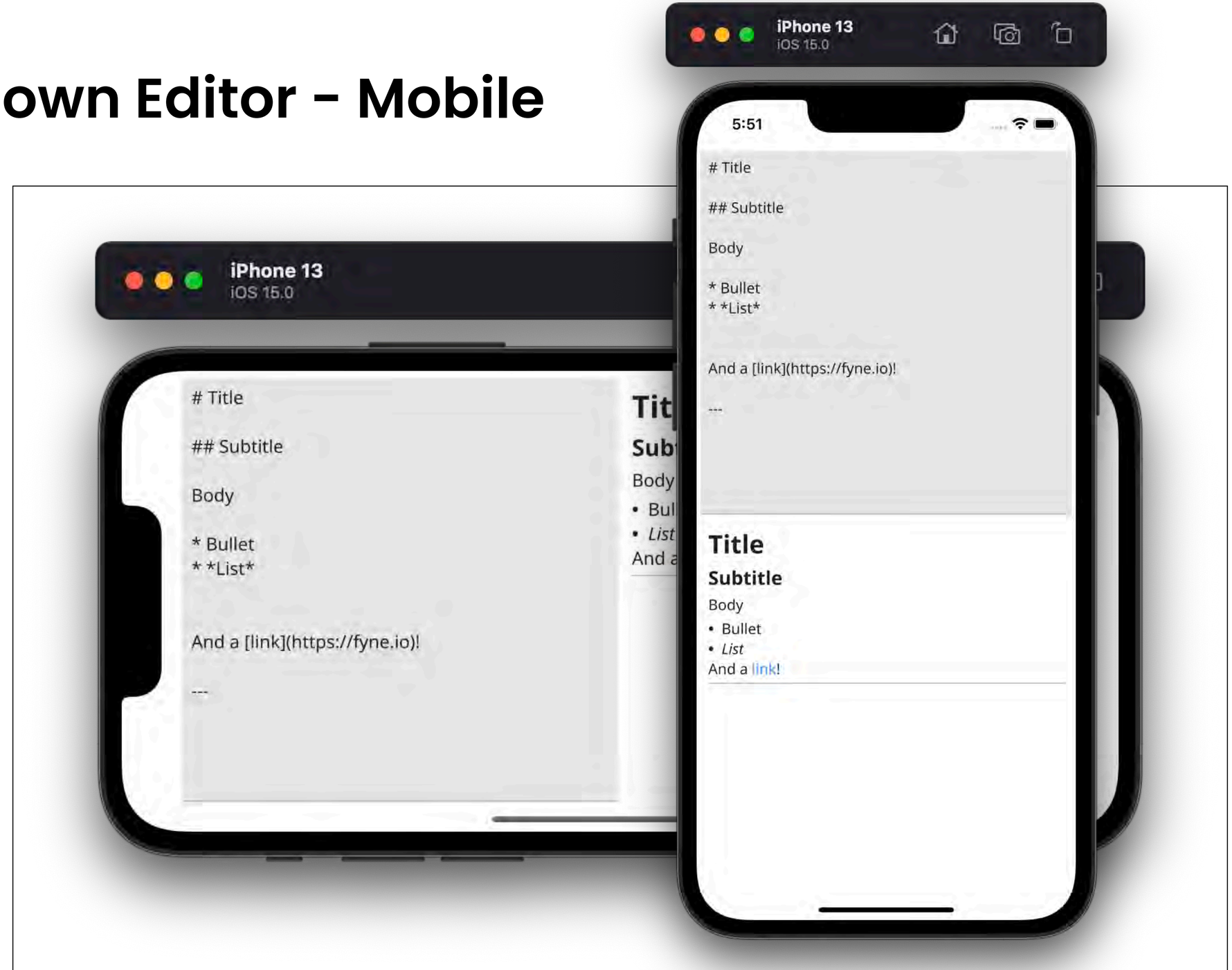
fyne labs

```go
package main

import (
    "fyne.io/fyne/v2/app"
    "fyne.io/fyne/v2/container"
    "fyne.io/fyne/v2/widget"
)

func main() {
    a := app.New()
    w := a.NewWindow("Markdown Editor")

    edit := widget.NewMultiLineEntry()
    preview := widget.NewRichTextFromMarkdown("")
    edit.OnChanged = preview.ParseMarkdown

    w.SetContent(
        container.NewAdaptiveGrid(2, edit, preview))
    w.ShowAndRun()
}
```

Markdown Editor - Complete :)



Markdown Editor - Mobile



Testing and distribution


```
$ mkdir project; cd project
$ go mod init project
$ go get fyne.io/fyne/v2

$ vim ui.go
$ go run .

$ vim ui_test.go
$ go test .
```

Testing and distribution - Code

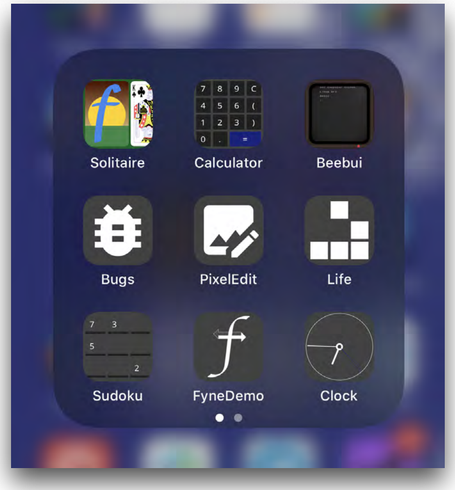
fyne labs

```go
package test

import (
    "testing"

    "github.com/stretchr/testify/assert"
    "fyne.io/fyne/v2/test"
    "fyne.io/fyne/v2/widget"
)

func TestText_Selected(t *testing.T) {
    e := widget.NewEntry()
    test.Type(e, "Hello")
    assert.Equal(t, "Hello", e.Text)

    test.DoubleTap(e)
    assert.Equal(t, "Hello", e.SelectedText())
    assert.Equal(t, 5, e.CursorColumn)
}
```

Package

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

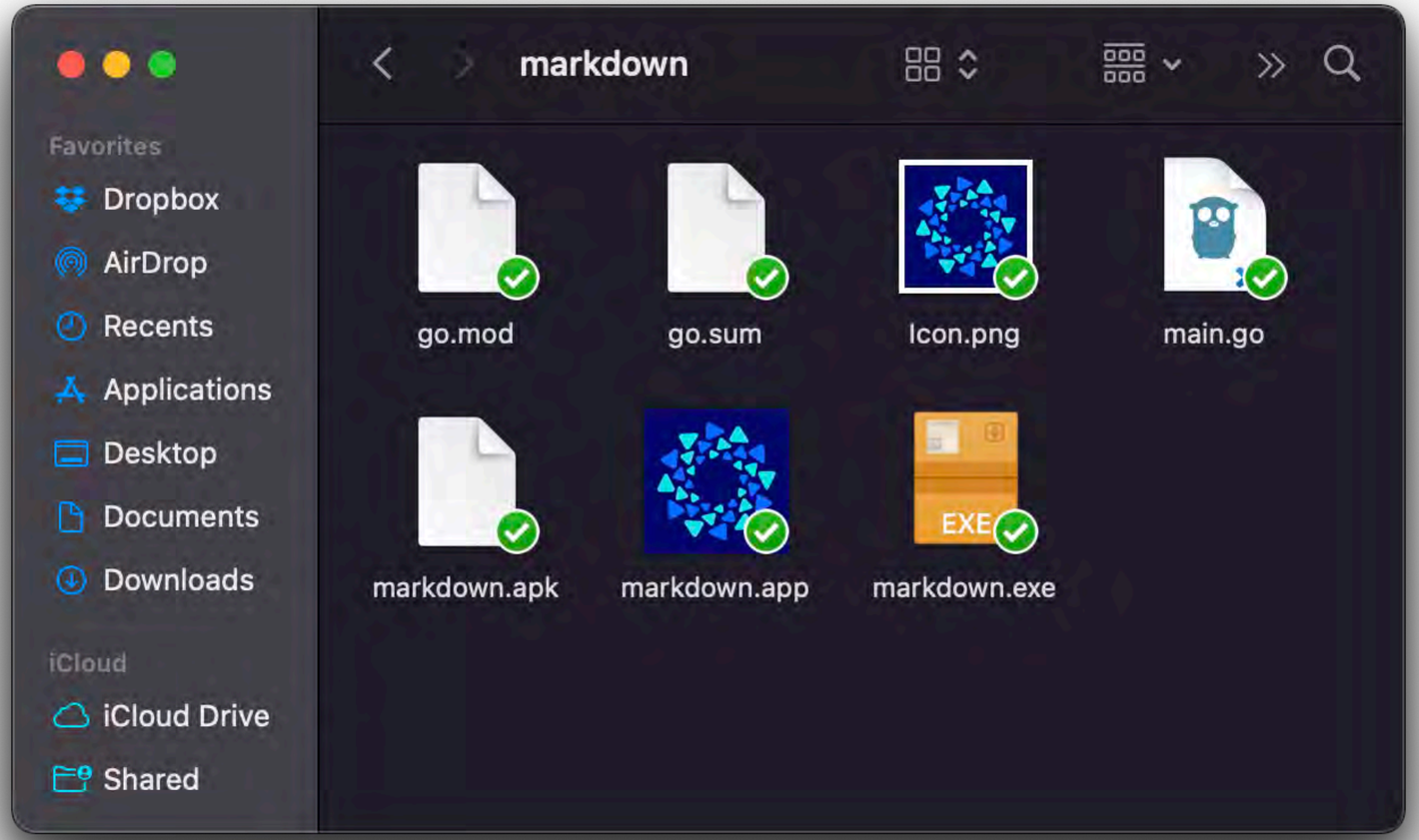

```
$ fyne package -os windows
$ fyne install -os android -appID com.mydomain.myProject
```



(or use fyne-cross instead of installing platform tools)

Package

fyne labs



Distribution

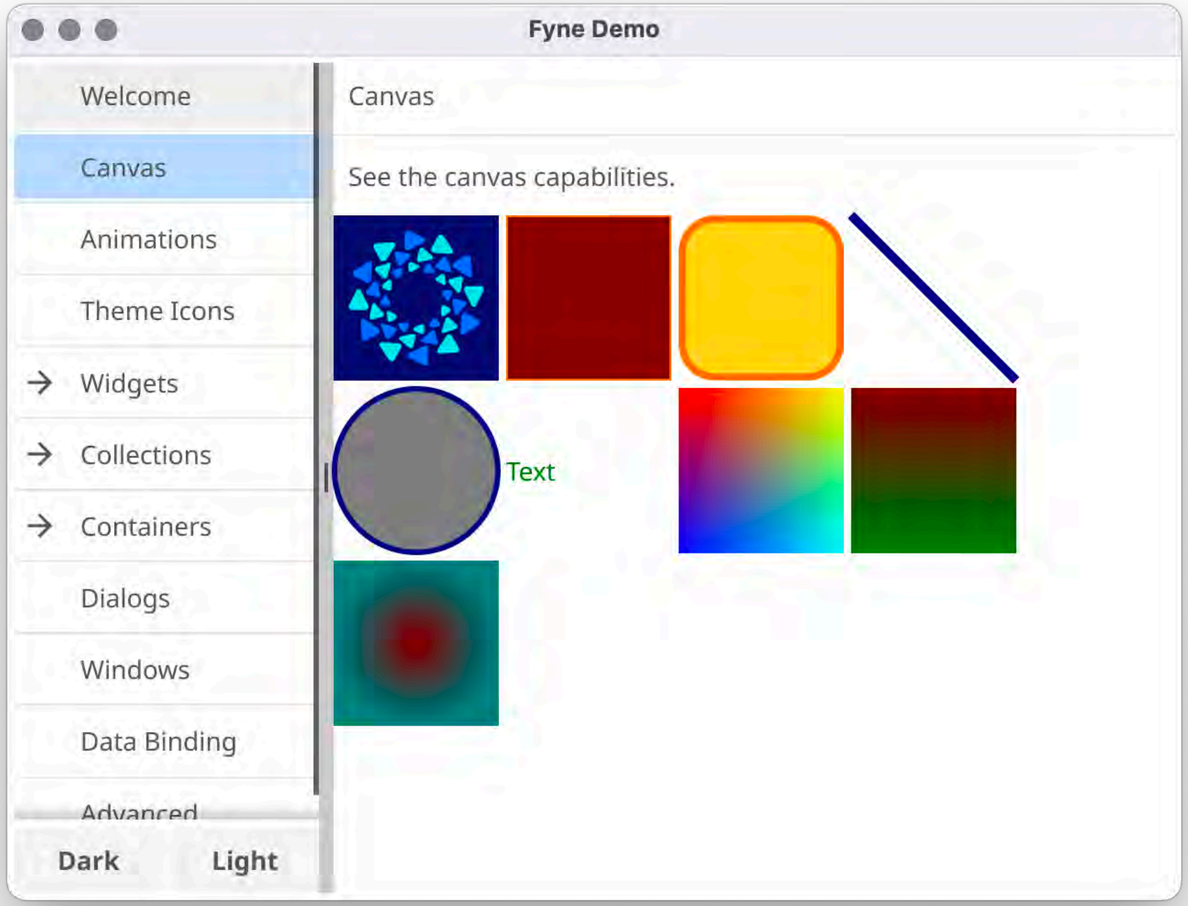
```
$ fyne release -appID com.mydomain.appname
$ fyne release -os ios -profile Xxx -certif
$ fyne release -os android -keystore ~/Zzz.
```

(many options can be omitted using FyneApp.toml)

Exploring Further...

Graphical capabilities

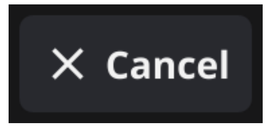
- Line
- (Rounded)Rectangle
- Circle
- Text
- (Linear/Radial)Gradient
- Image (SVG, bitmap), Raster
- Icon and themes bundled

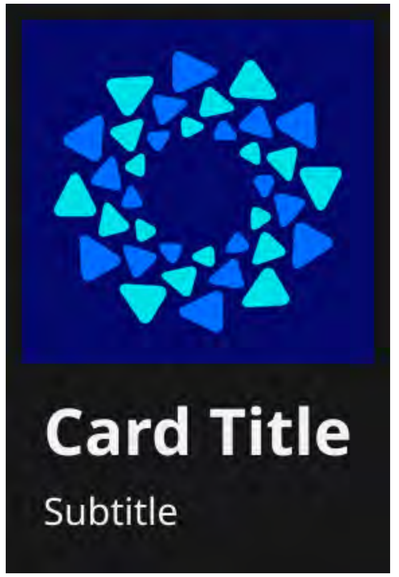


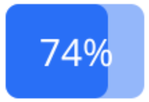
Widgets - basic

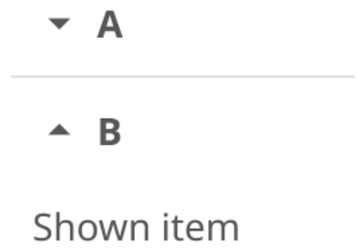
Text label

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



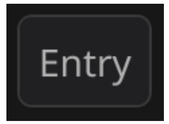


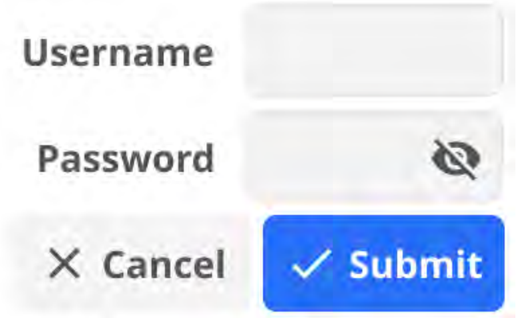


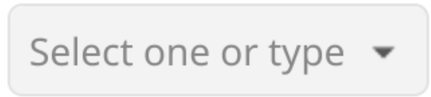


Widgets – input

- `widget.NewEntry()`
- `widget.NewPasswordEntry()`
- `widget.NewSlider(0, 100)`
- `widget.NewCheck("Check", func(bool) {})`
- `widget.NewRadioGroup([]string{ "Option 1", "Option 2"}, func(string) {})`
- `widget.NewSelect([]string{"Option A", "Option B"}, func(string) {})`

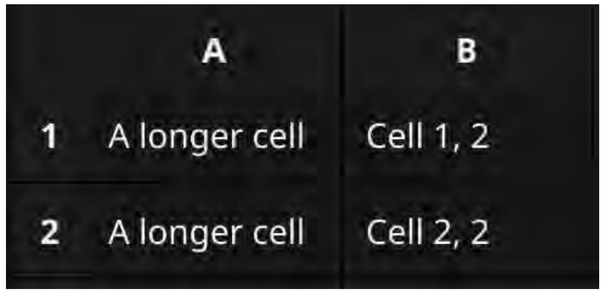


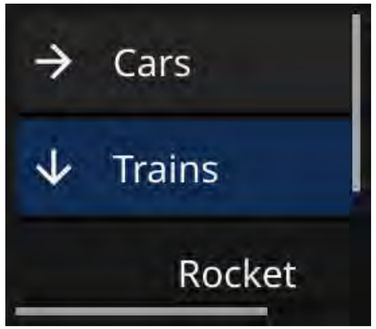


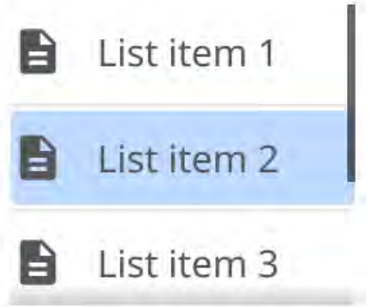


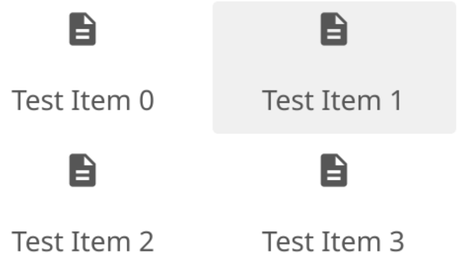
Widgets - collection

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

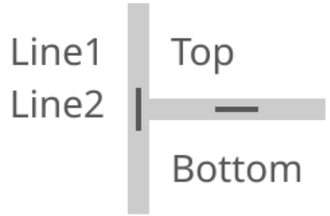




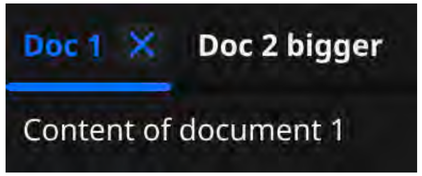


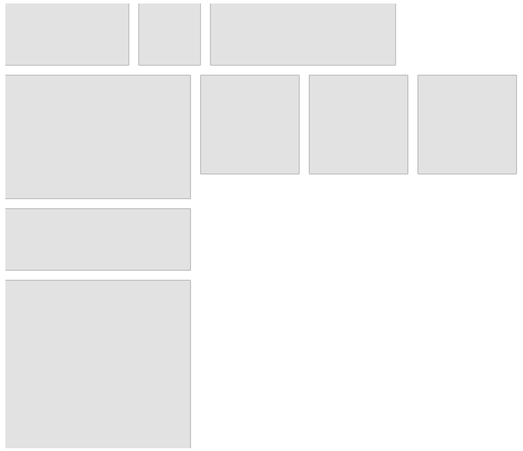


Containers



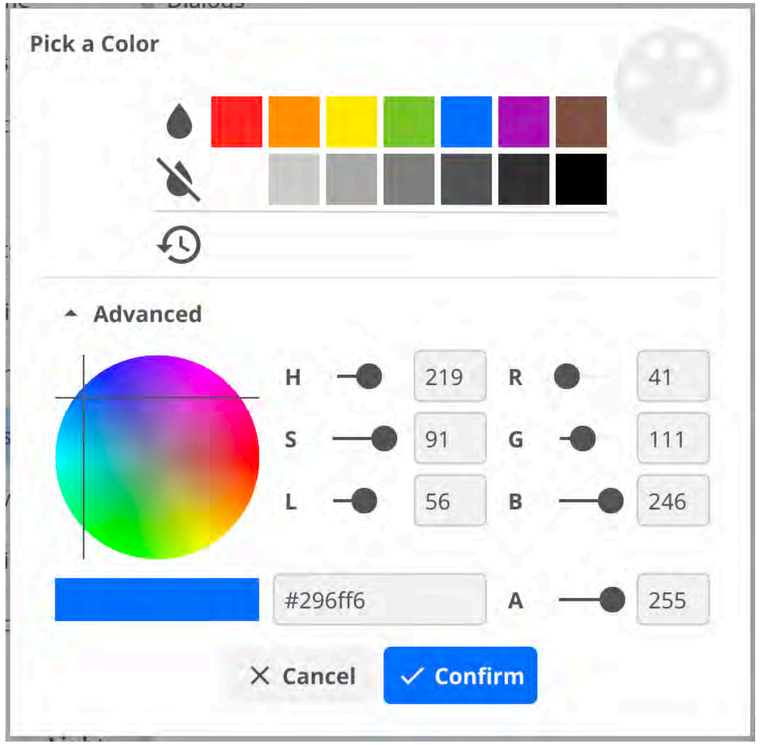
- `container.NewHSplit(left, right)`
- `container.NewAppTabs(items…)`
- `container.NewDocTabs(items…)`
- `container.NewScroll(content)`
- `container.New(layout, objects…)`

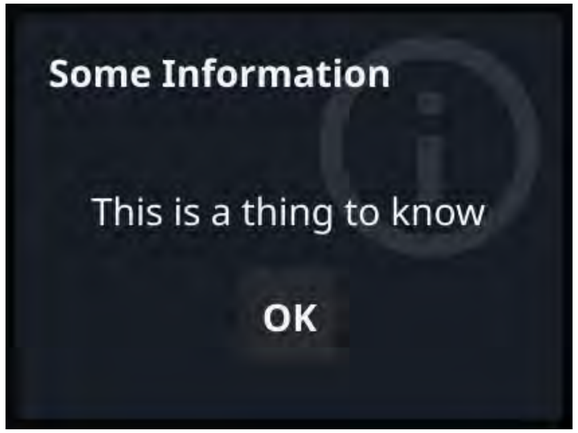


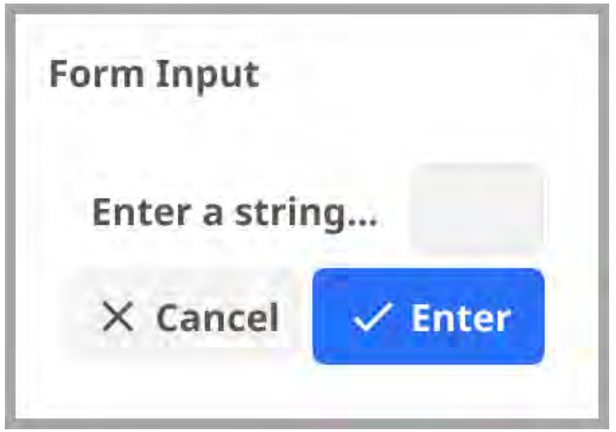


Dialogs

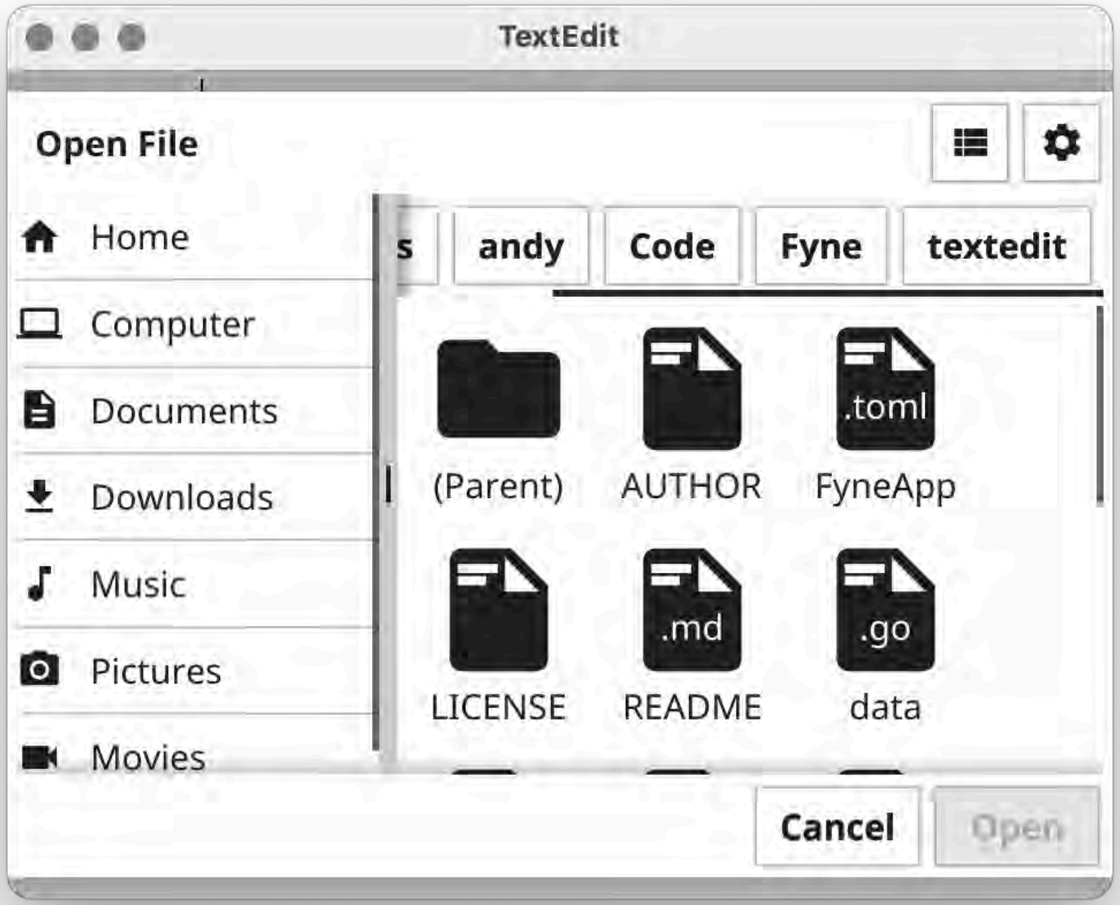
- `dialog.ShowConfirm("Confirmation", "Are you enjoying this demo?", func(bool) {}, win)`







File and I/O

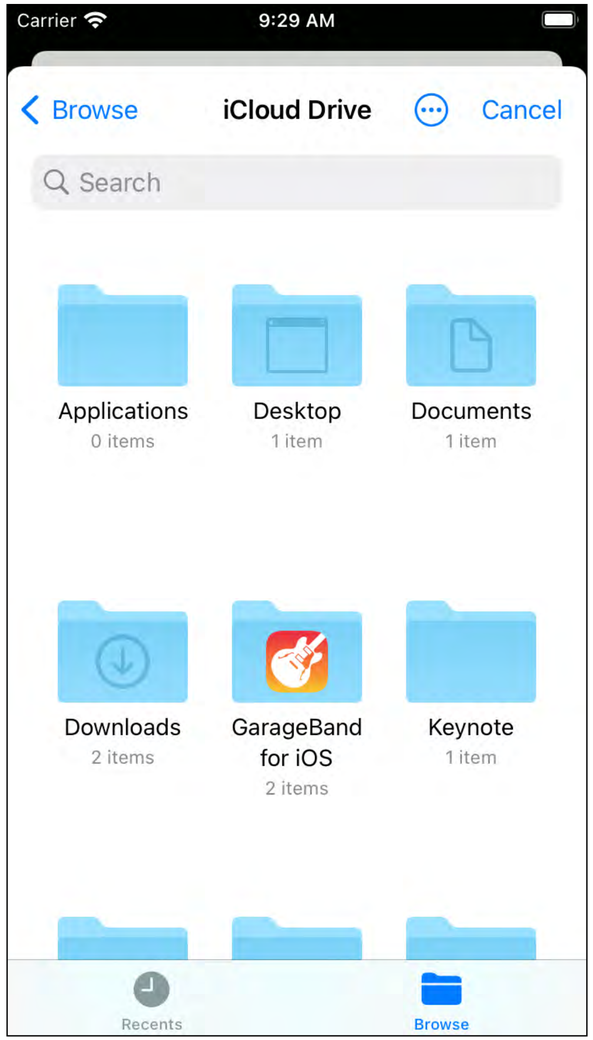

TextEdit
Open File
Home
Computer
Documents
Downloads
Music
Pictures
Movies
s
andy
Code
Fyne
textedit
(Parent)
AUTHOR
.toml
FyneApp
LICENSE
.md
README
.go
data
Cancel
Open

File and I/O - Code

fyne labs

```
func open() {
    dialog.ShowFileOpen(func(r fyne.URIReadCloser, err error) {
        if err != nil {
            dialog.ShowError(err, window)
            return
        }
        if r == nil {
            return
        }

        data, err := ioutil.ReadAll(r)
        if err == nil {
            _ = r.Close()
            e.entry.SetText(string(data))
            savedURI := r.URI()
        } else {
            dialog.ShowError(err, window)
        }
    }, window)
}
```

File and I/O - Data from anywhere

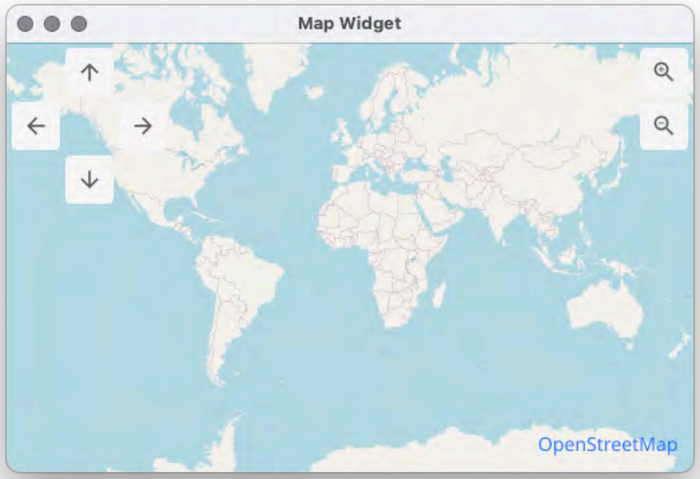


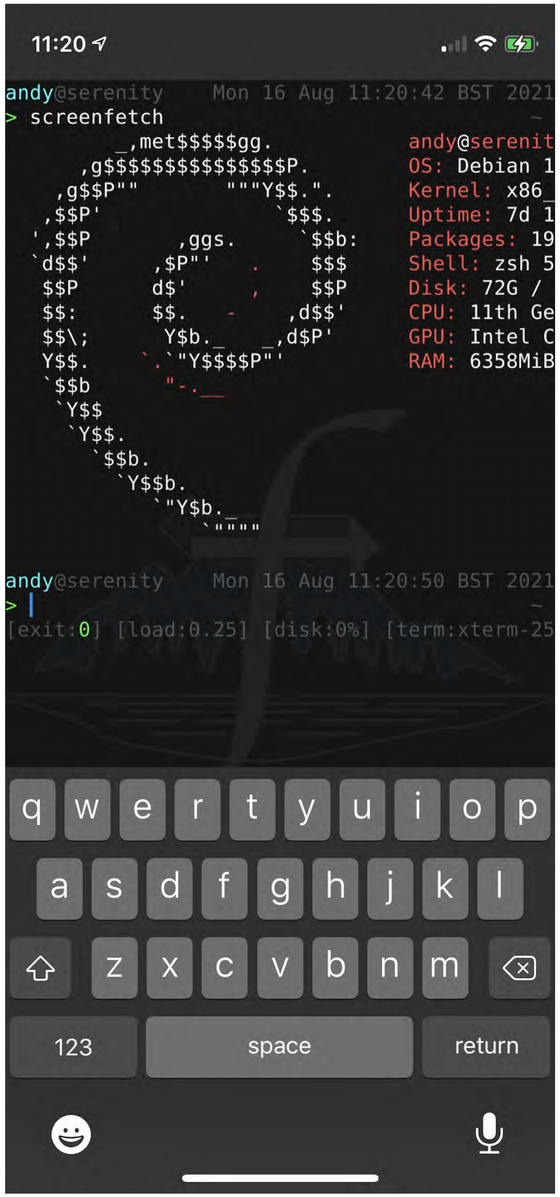
Additional possibilities…

- Window.SetMainMenu
- (desktop.App).SetSystemTrayMenu()
- Use data binding to avoid some manual code
- So much more, see fyne_demo:
  `go install fyne.io/fyne/v2/cmd/fyne_demo@latest`

3rd party components too!

- Just import package and call constructor
- Works like any widget
  - `map := xWidget.NewMap()`
  - `cmdline := terminal.New()`



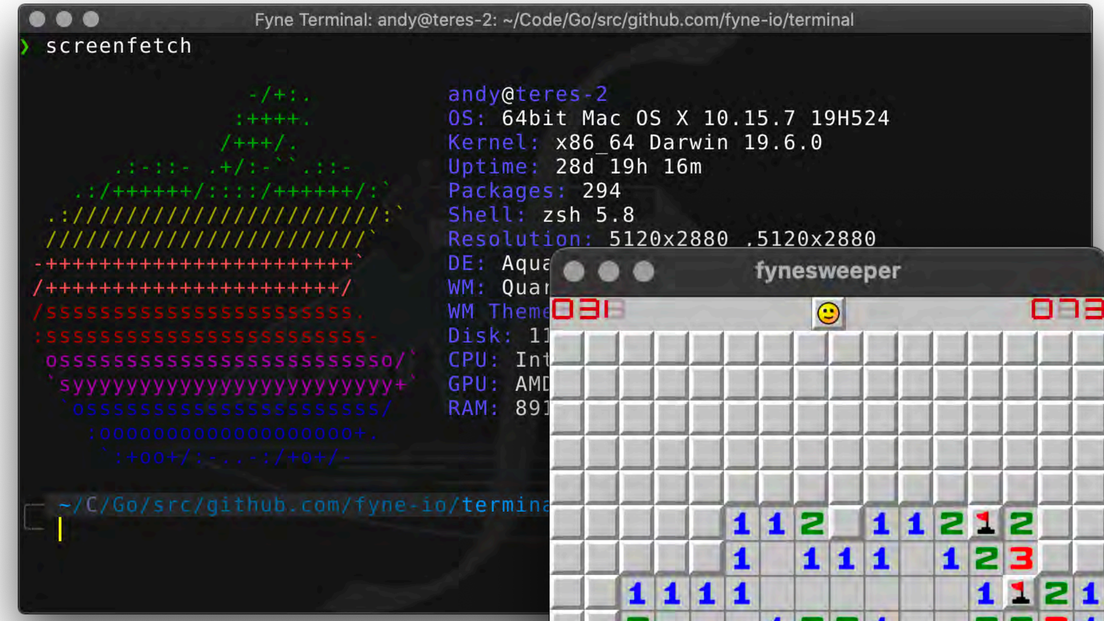


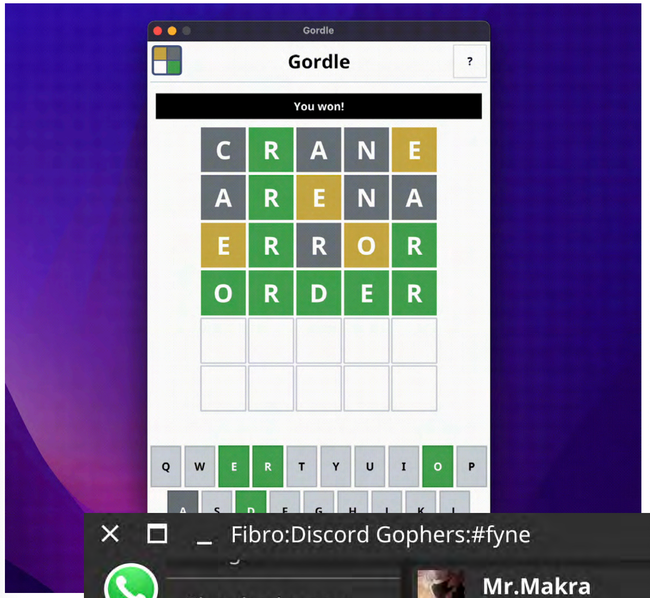
https://github.com/fyne-io/fyne-x

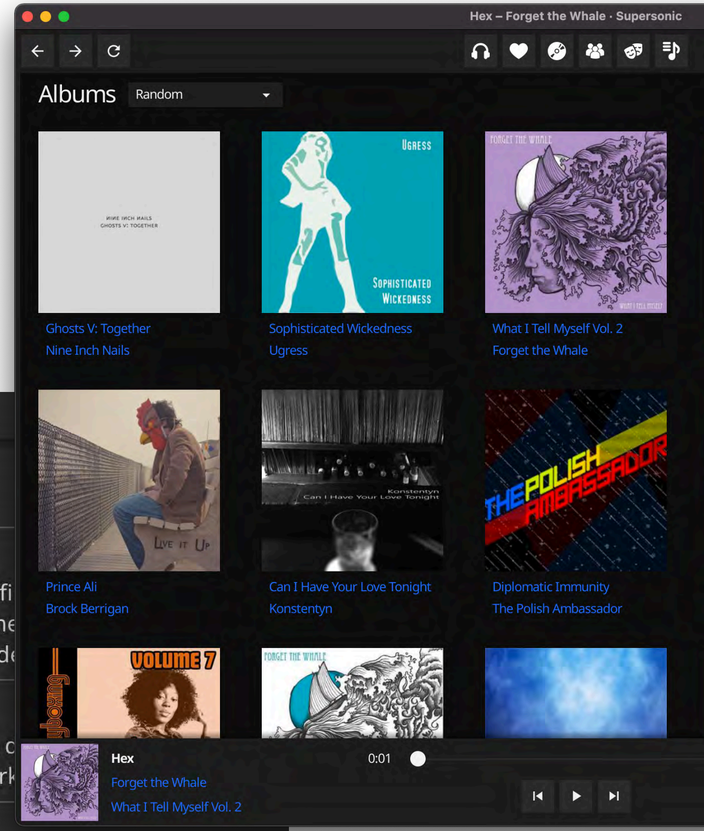
https://addons.fyne.io

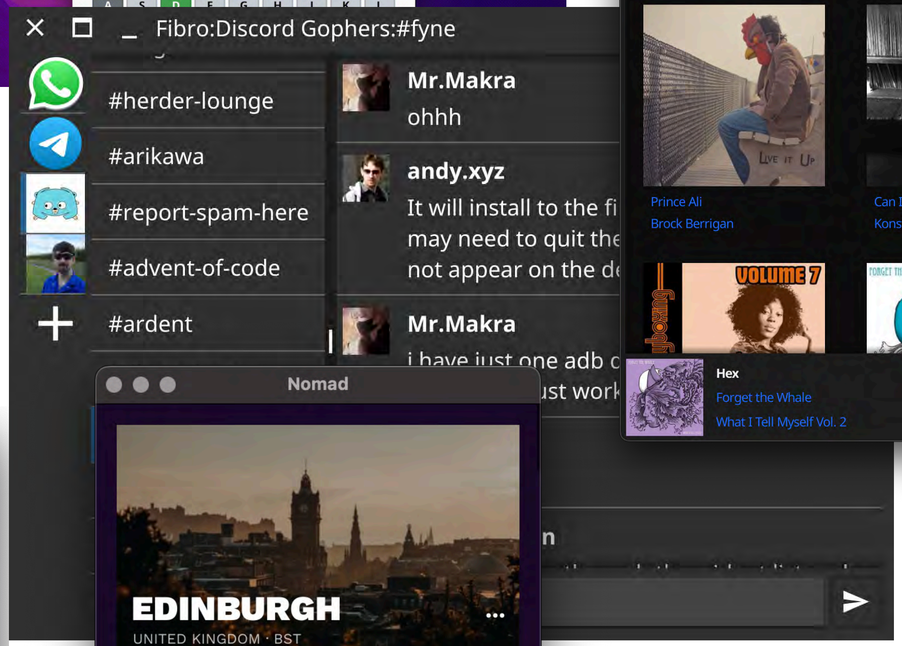
So many possibilities!

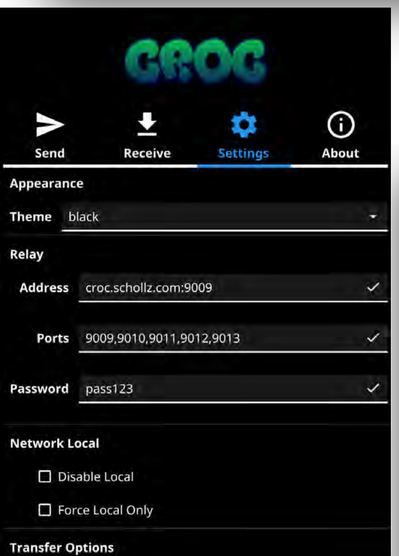
Other Fyne apps

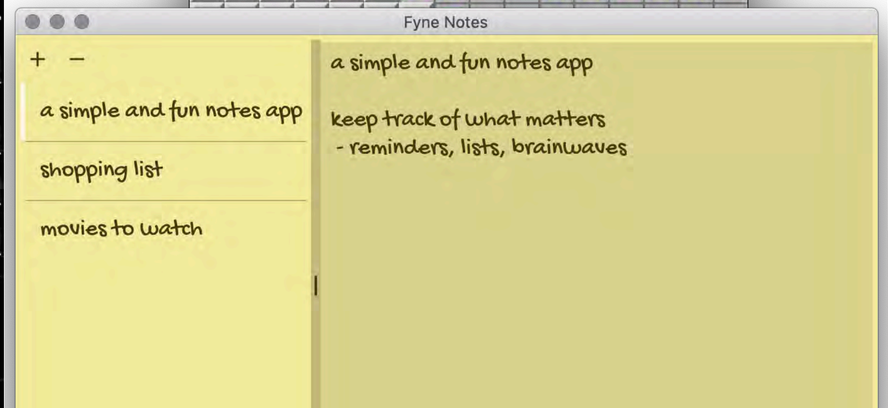






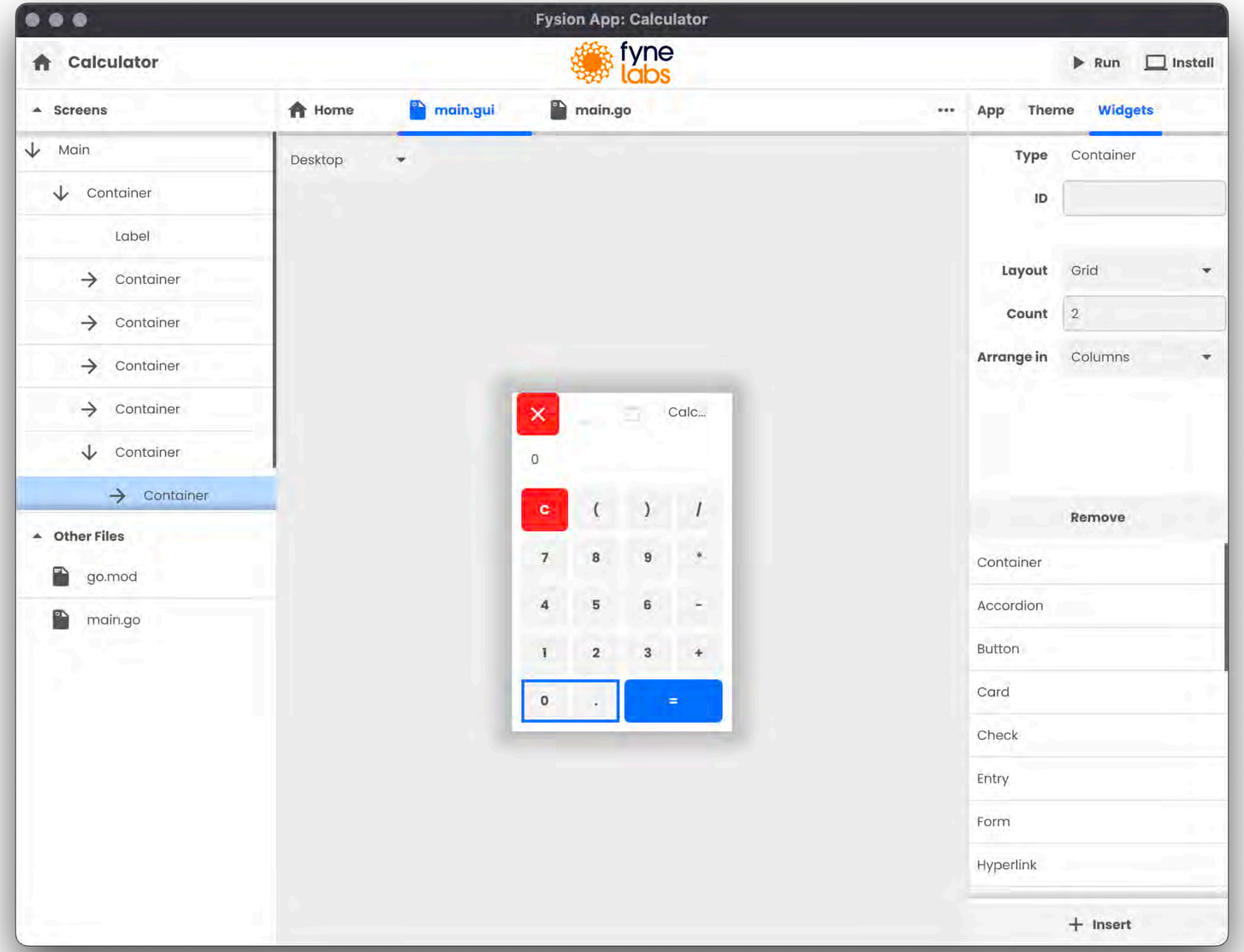






https://apps.fyne.io

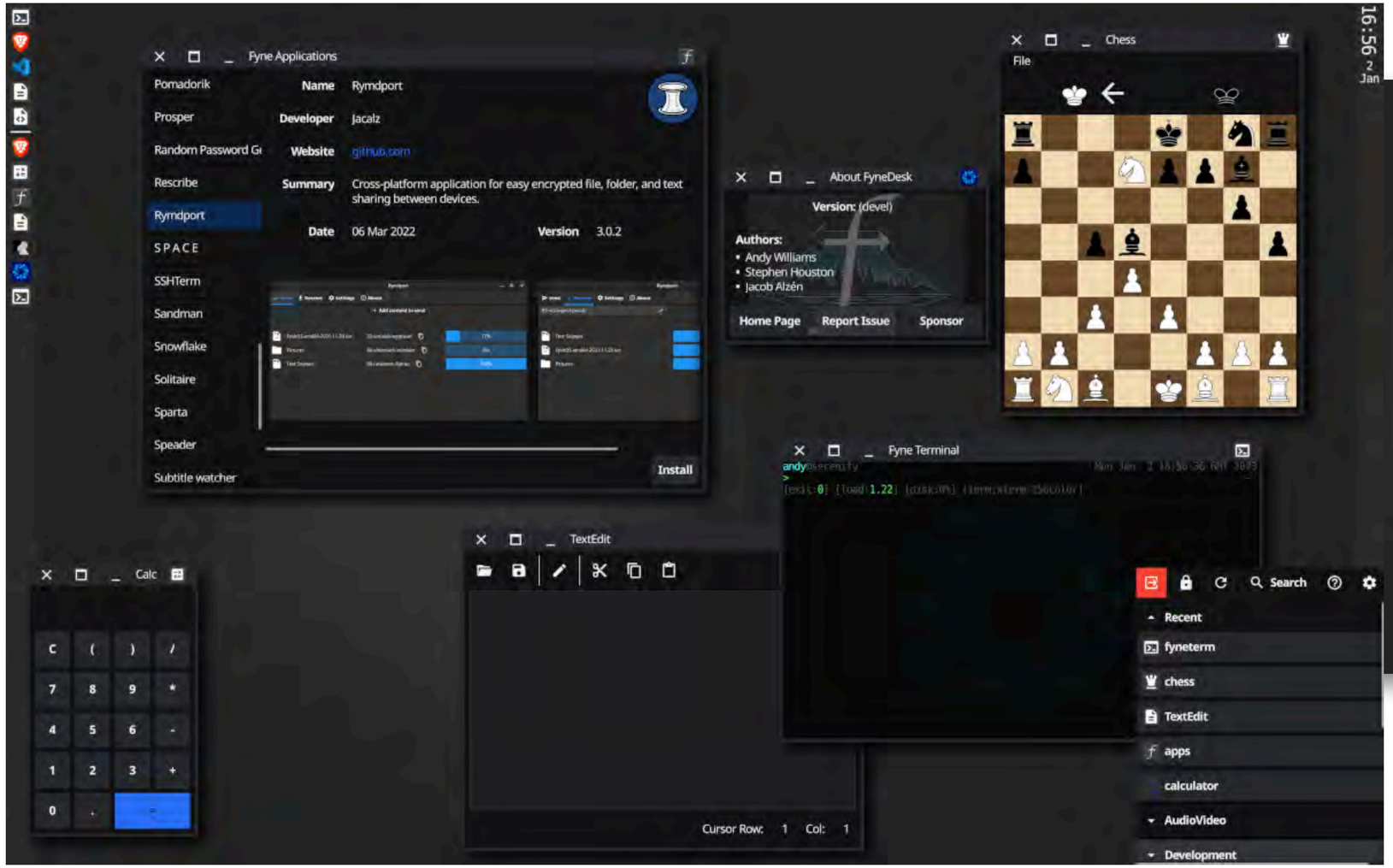
And Fysion!

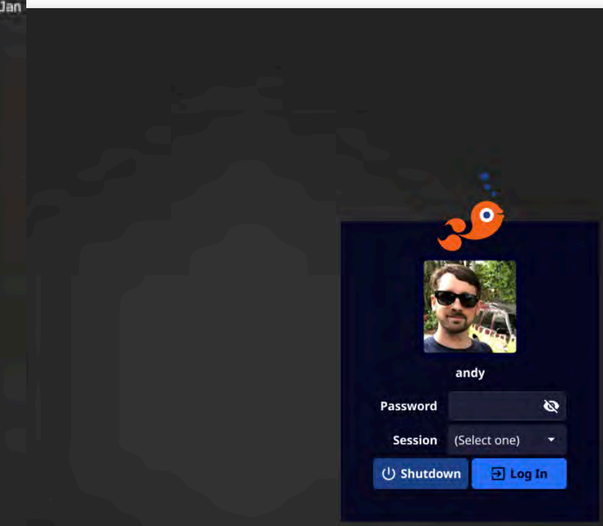


https://fysion.app

And FyshOS...

fyne labs





https://fyshos.com

More about Fyne

- Learn more about using Fyne

  https://docs.fyne.io - https://www.youtube.com/@fyneio

- Read: Building Cross-Platform GUI Applications with Fyne

  https://packtpub.com/ - https://amazon.com/

- Contribute to the project - Code, Test, Document, Design

  https://github.com/fyne-io/fyne/

- Sponsor us!

  https://fyne.io/sponsor/

fyne labs

Thank you

Andrew Williams

@andydotxyz

andy@fynelabs.com