

fyne



How Go Taught Me to Love Building Apps Again

Andrew Williams – GoLab

20th November 2023

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

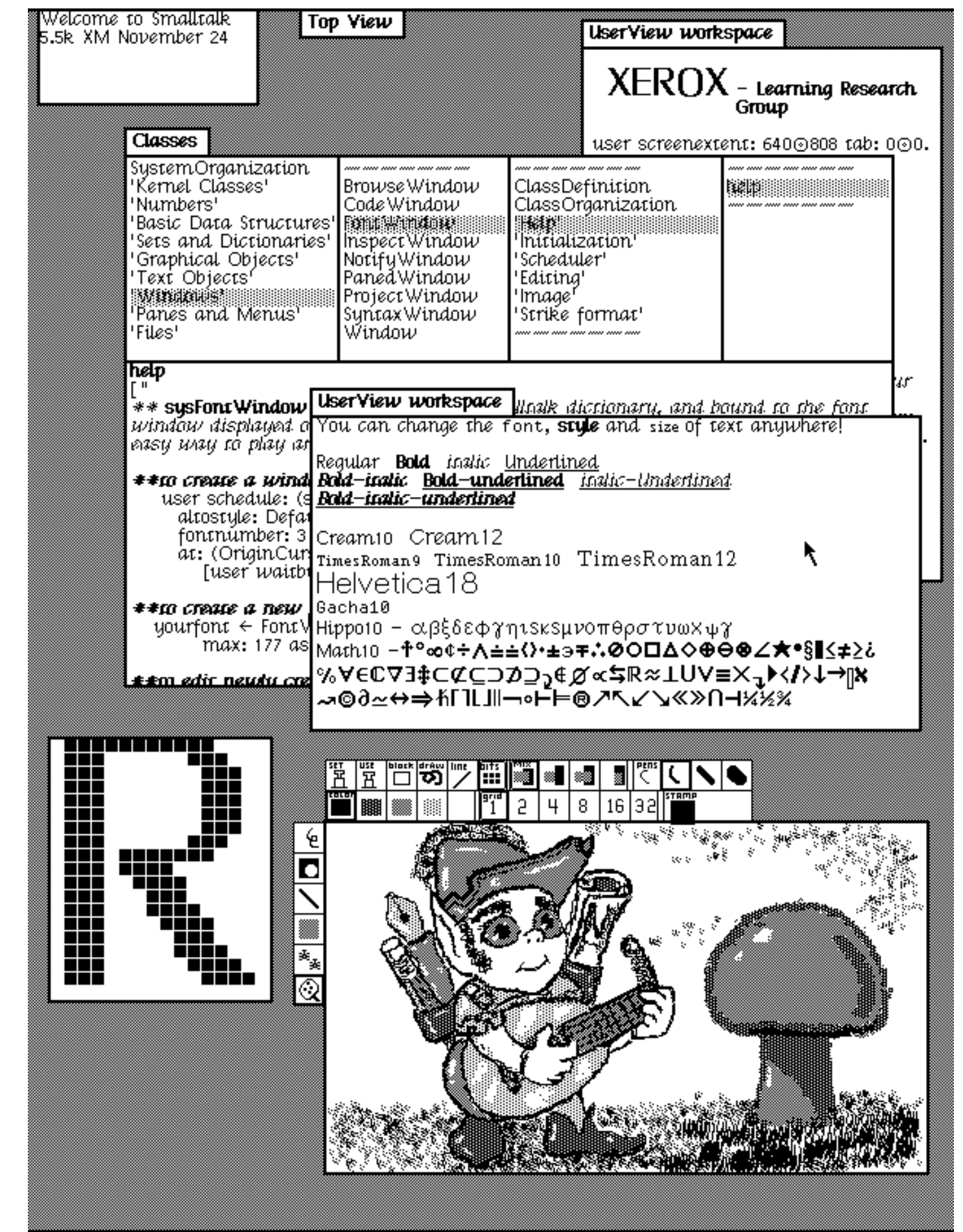
About Me



Background: Frustrations of GUI development



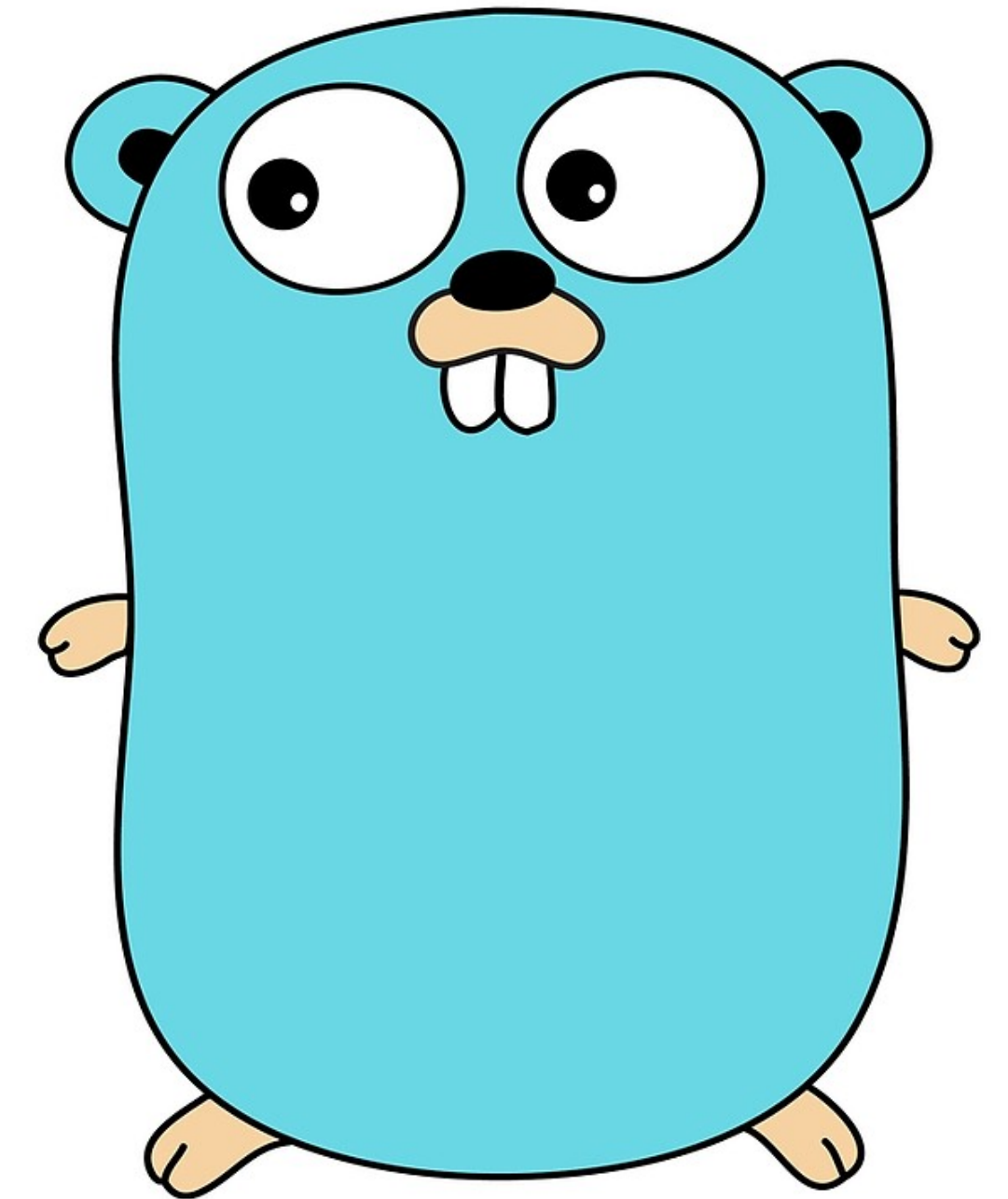
- Mostly C language heritage
 - Memory management
 - Thread handling
 - Lacking web services (or even strings / unicode!)
- Modern alternatives often bindings over old
- Complex to learn
- Toolkits either massive or complex installs
- Modern alternatives only for some platforms



Go: A delightful alternative



- 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



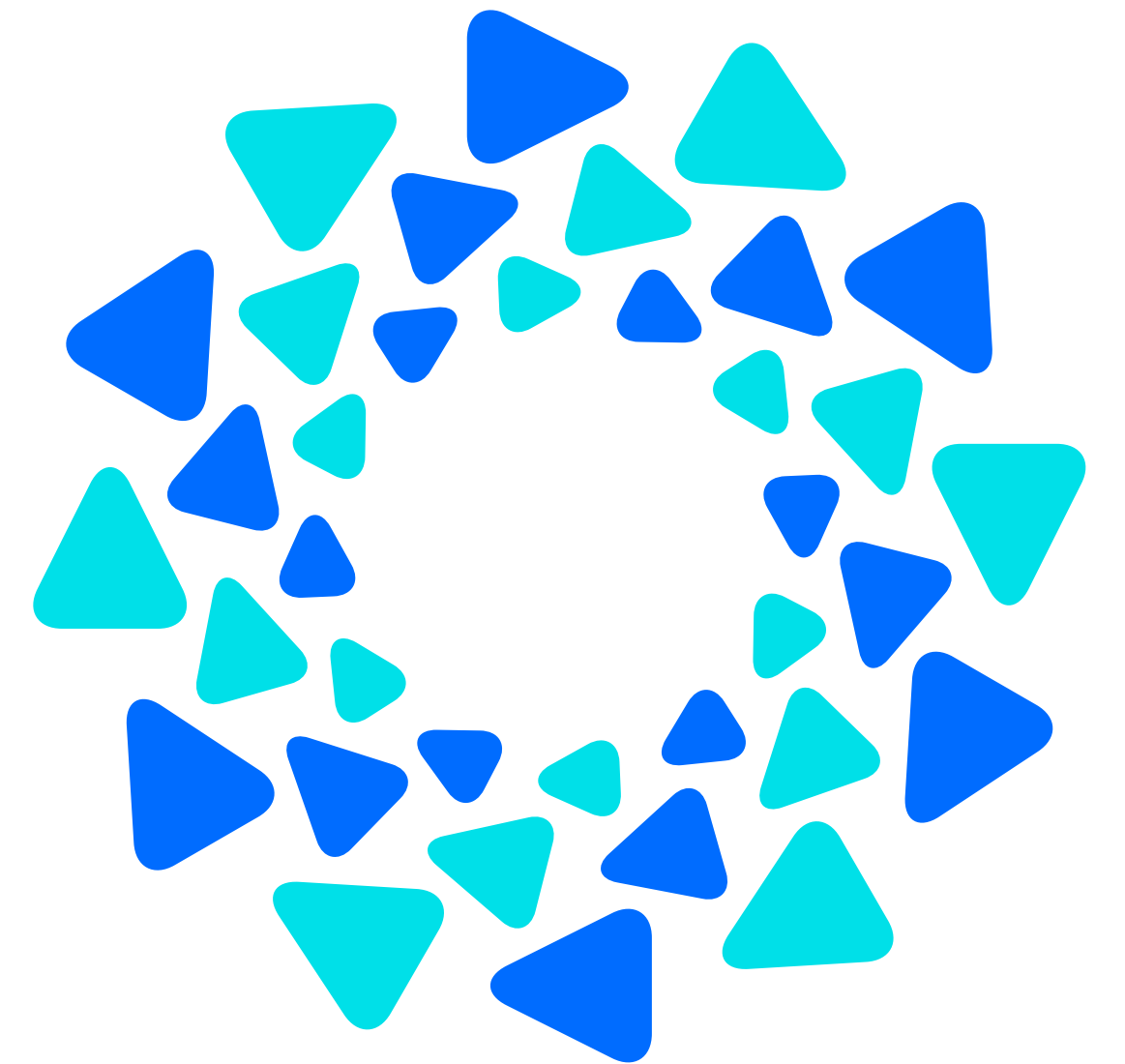
Fyne: A new hope for app developers



An open source native app toolkit?



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

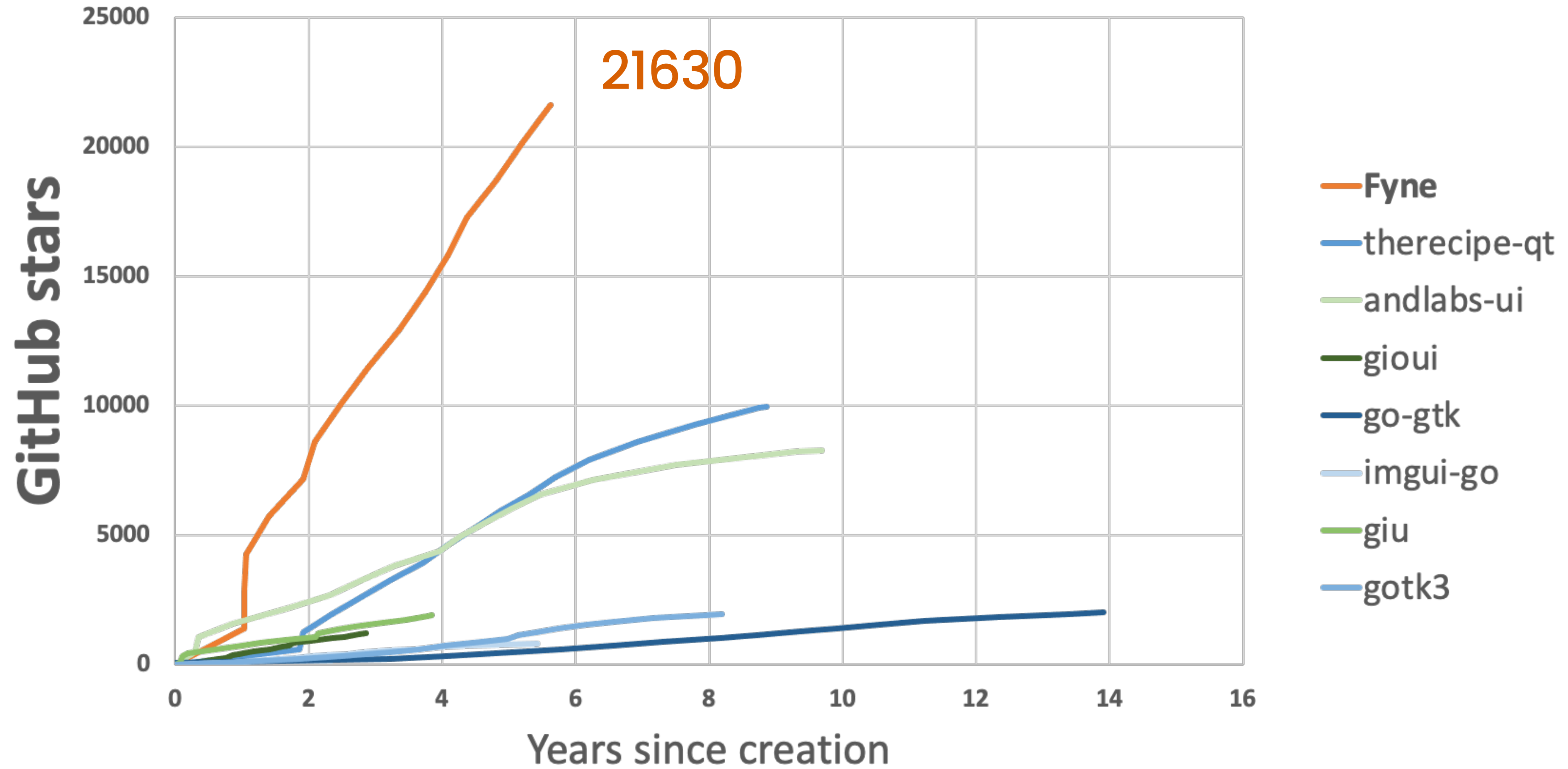


Fyne Project



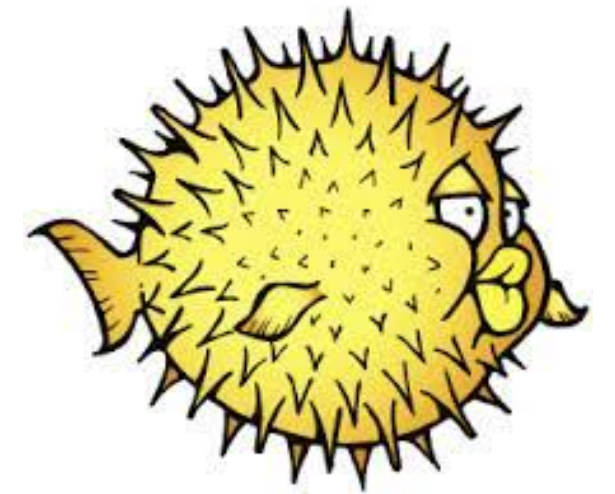
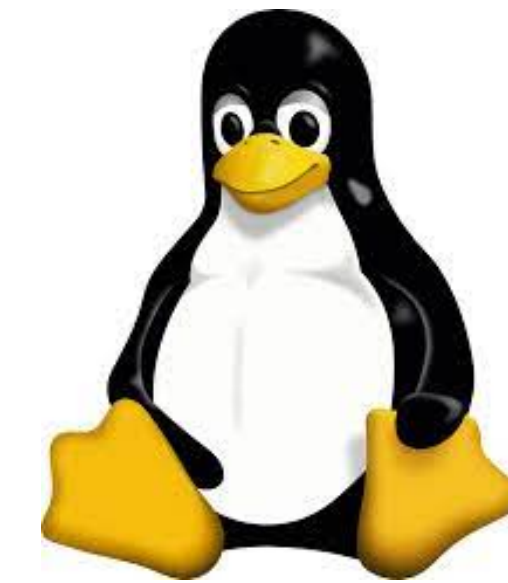
Fyne aims to be the simplest toolkit for developing beautiful and user-friendly native graphical applications for desktop, mobile and beyond.

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



- 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



```
package main

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

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

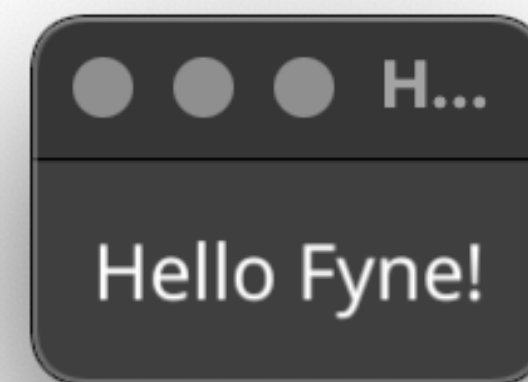
    w.SetContent(widget.NewLabel("Hello Fyne!"))

    w.ShowAndRun()
}
```

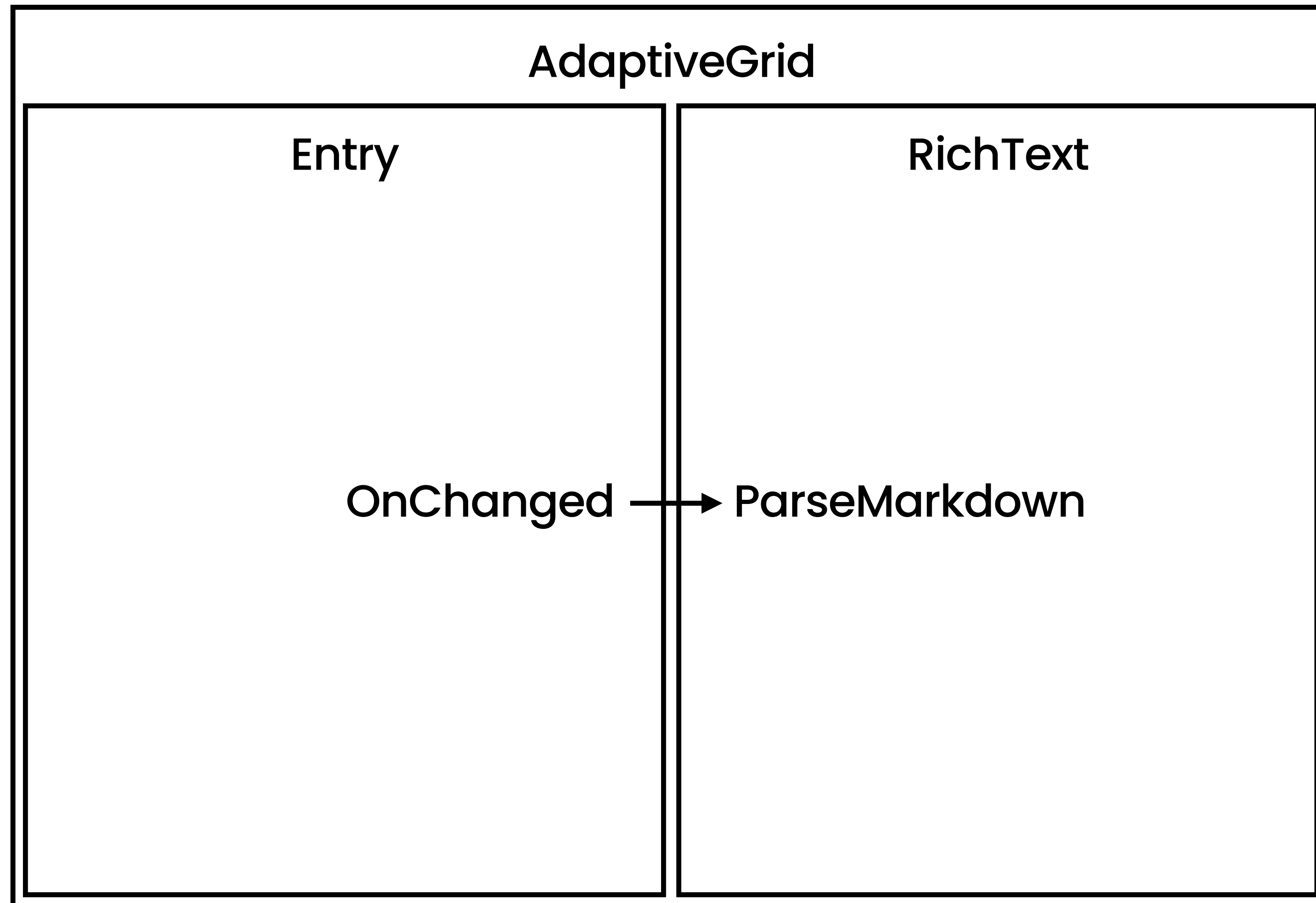
Building our first app - Run



```
$ 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



```
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")

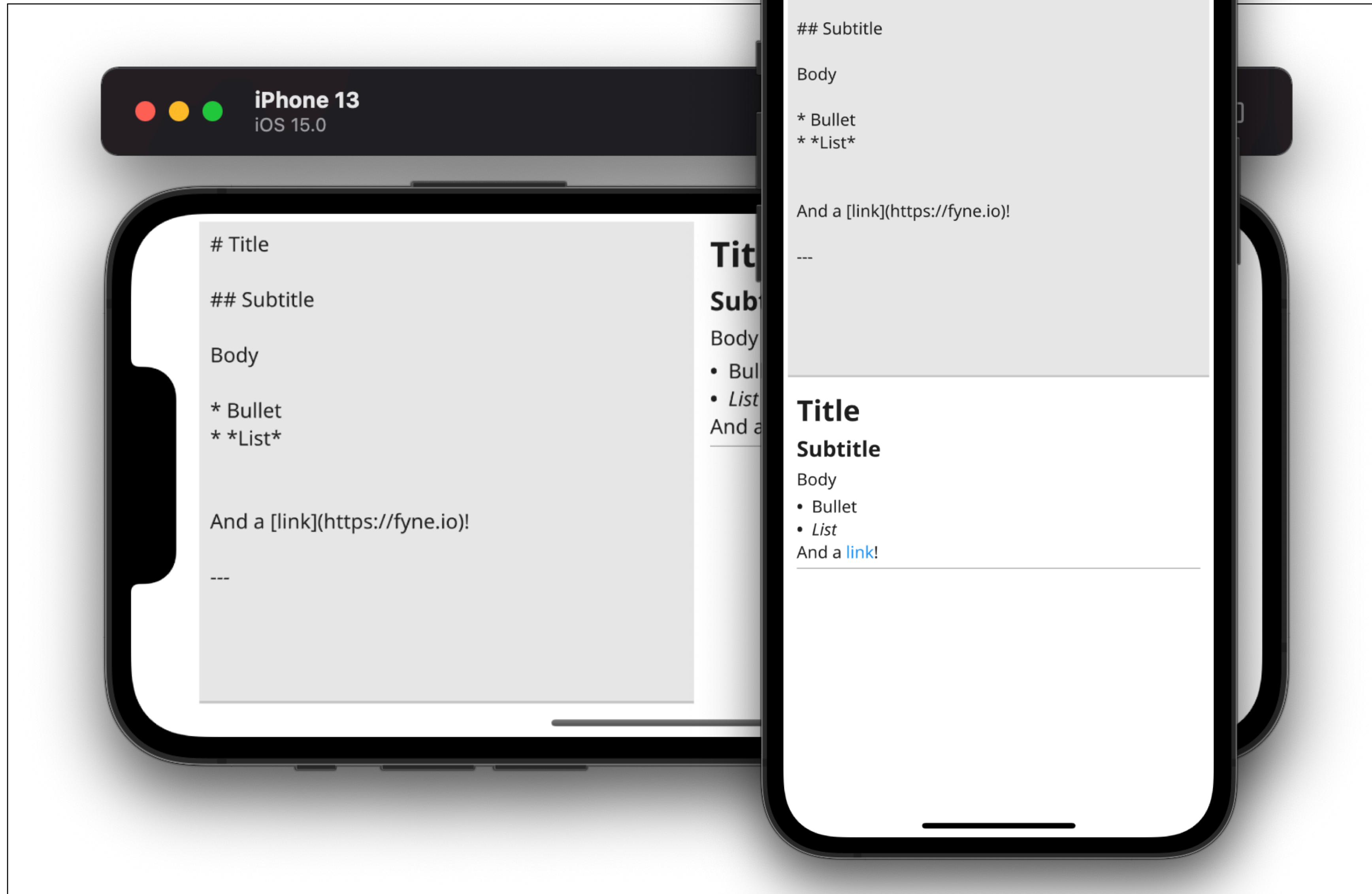
    edit := widget.NewMultiLineEntry()
    preview := widget.NewRichTextFromMarkdown("")
    edit.OnChanged = preview.ParseMarkdown

    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



```
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
```

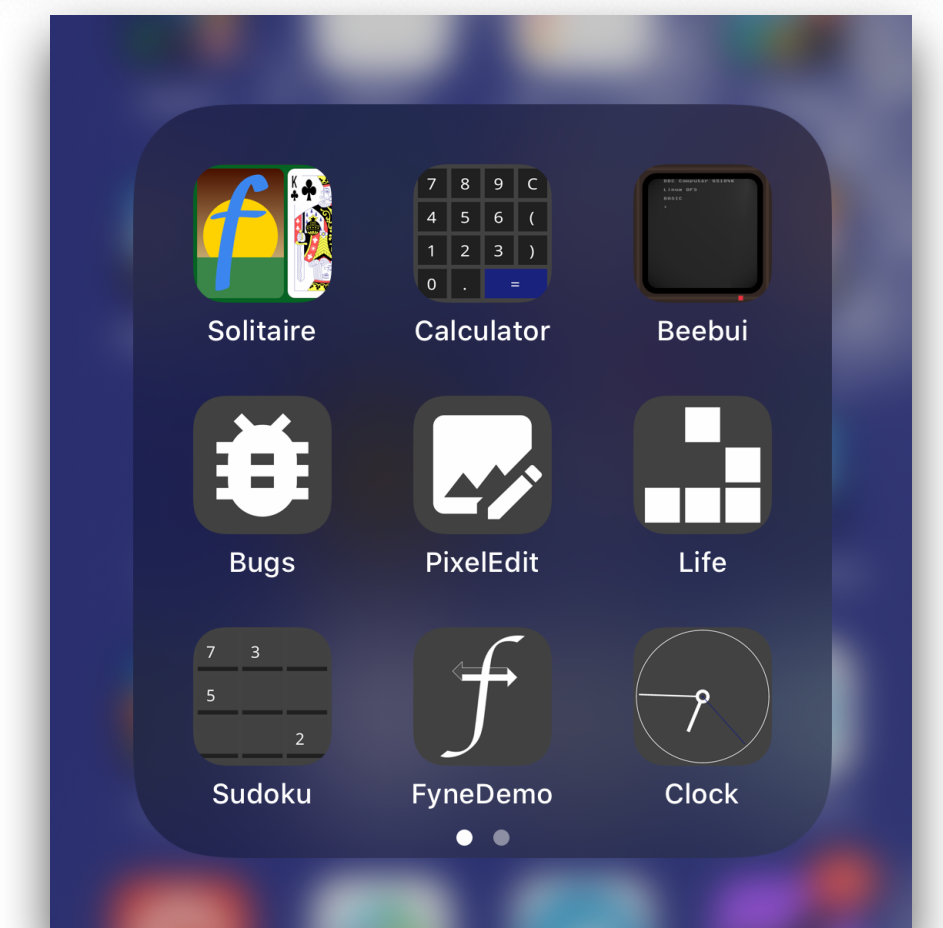
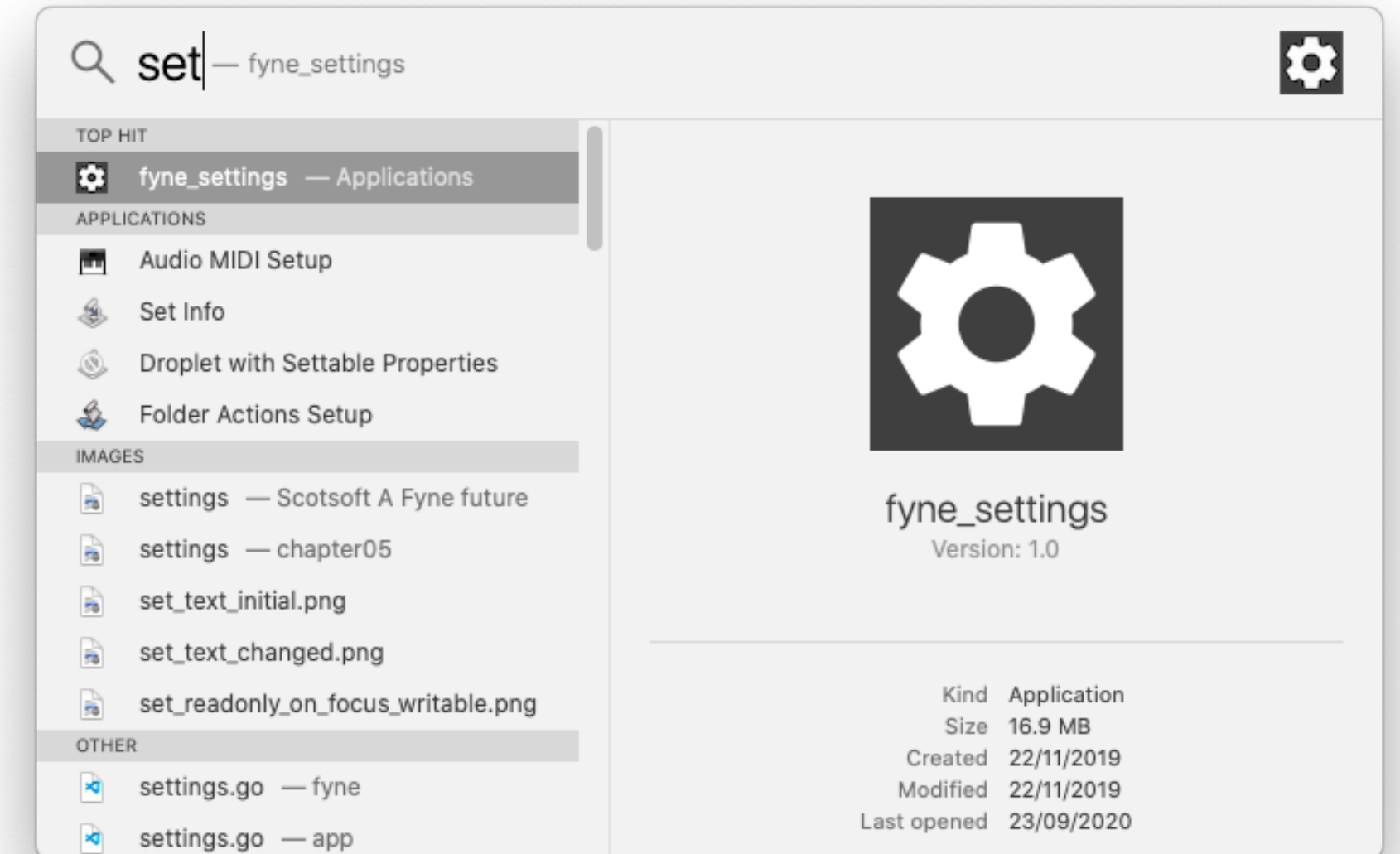
```
$ fyne package
```

```
$ fyne install
```

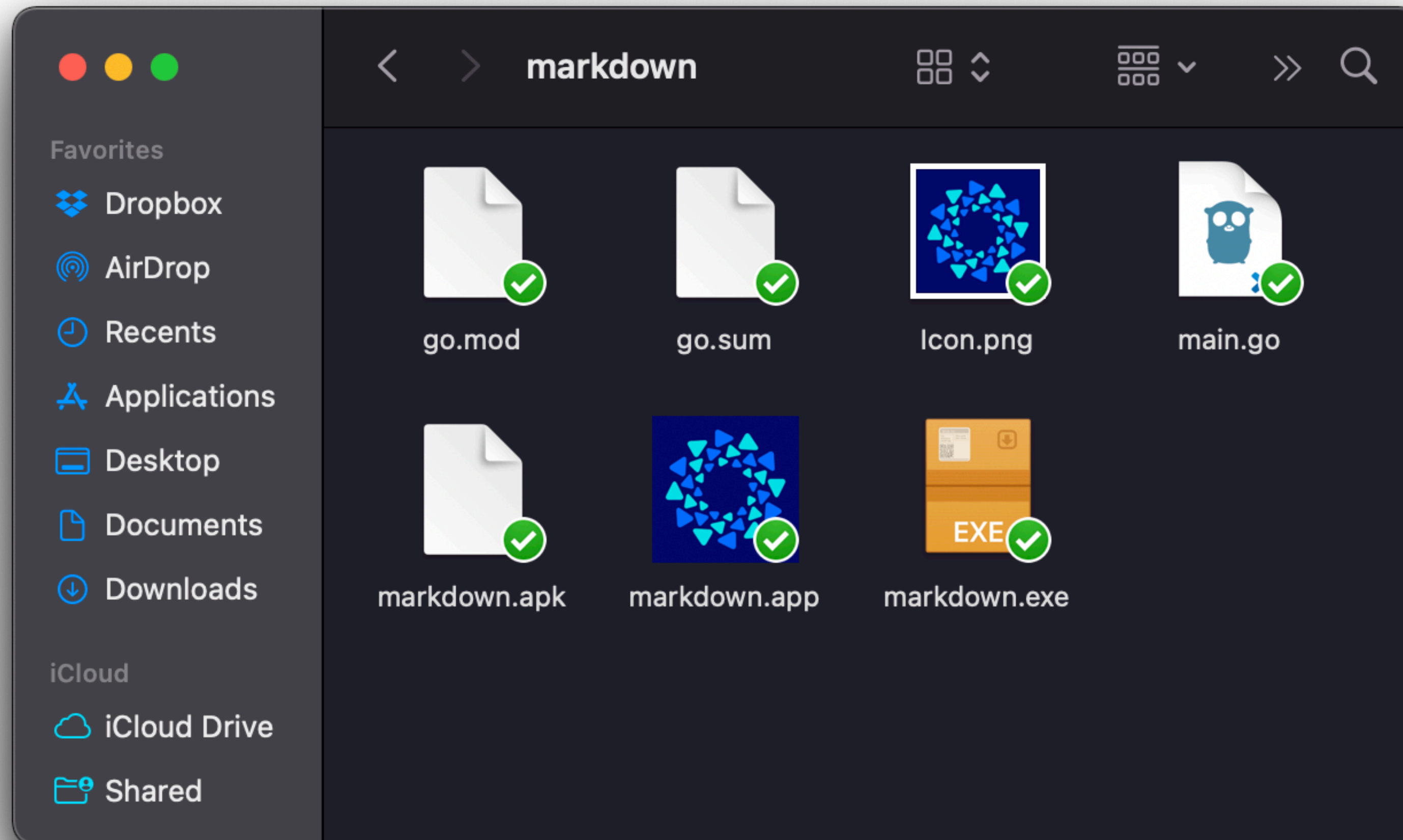
```
$ fyne package -os windows
```

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

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



Package



Distribution



```
$ fyne release -appID com.mydomain.appname
```

```
$ fyne release -os ios -profile Xxx -certif
```

```
$ fyne release -os android -keystore ~/Zzz.
```



```
Website = "https://fyne.io"
```

```
[Details]
```

```
Icon = "img/fyne.png"
```

```
Name = "Fyne Demo"
```

```
ID = "io.fyne.demo"
```

```
Version = "1.0.0"
```

```
Build = 5
```

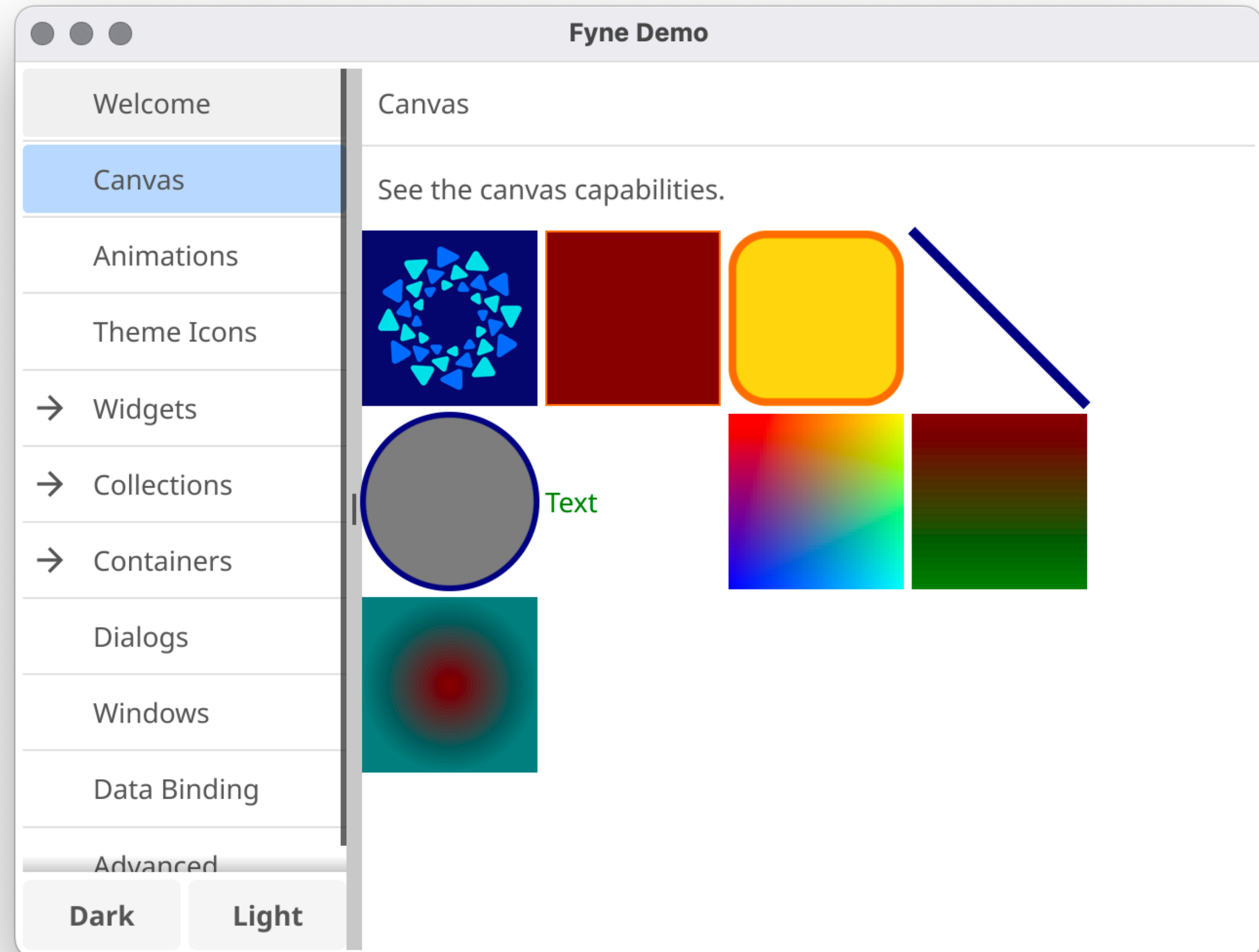
(many options can be omitted using `FyneApp.toml`)

Exploring Further...



Graphical capabilities

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

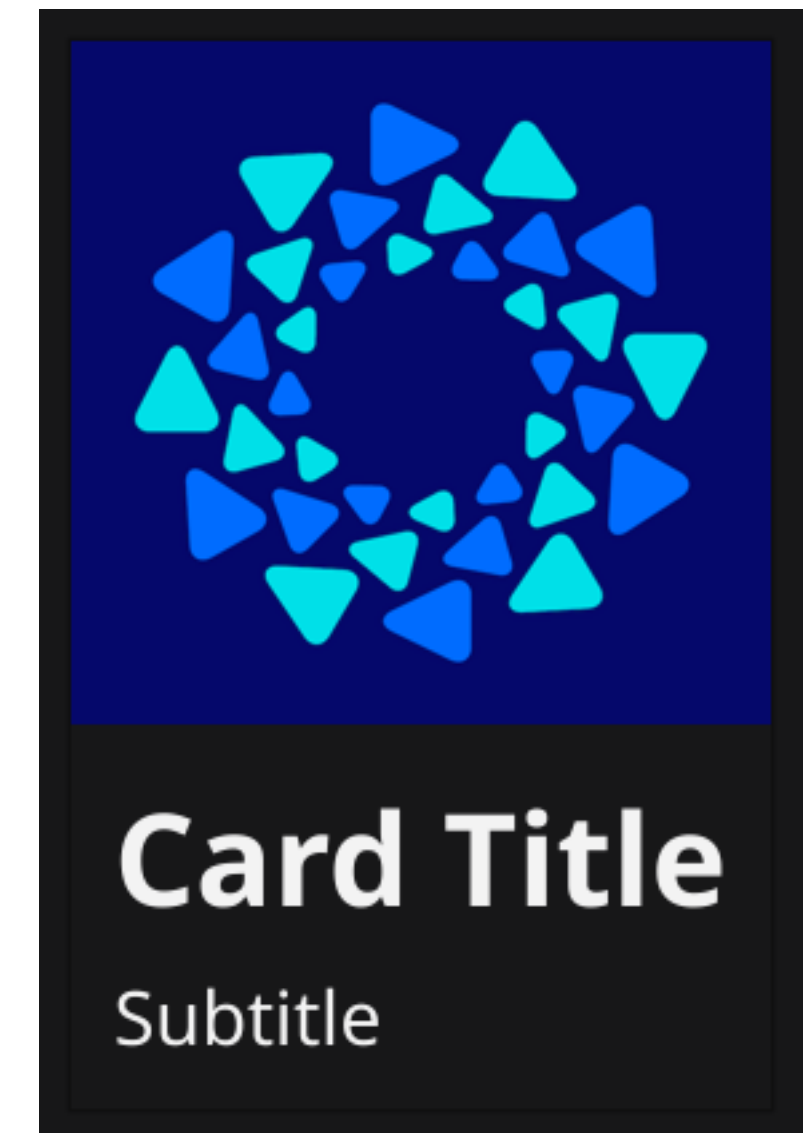
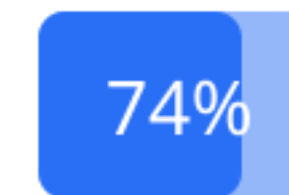


Widgets - basic



Text label

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



RichText Heading
A Sub Heading



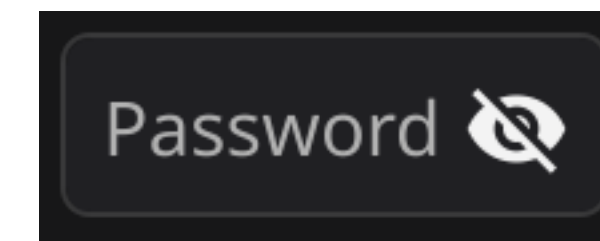
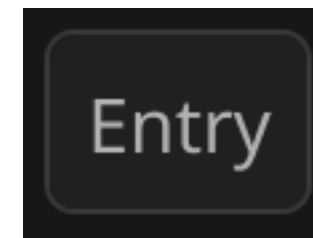
Shown item

```
1 | TextGrid  
2 | ··Content
```

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) {})`



Username

Password

(Select one) ▼

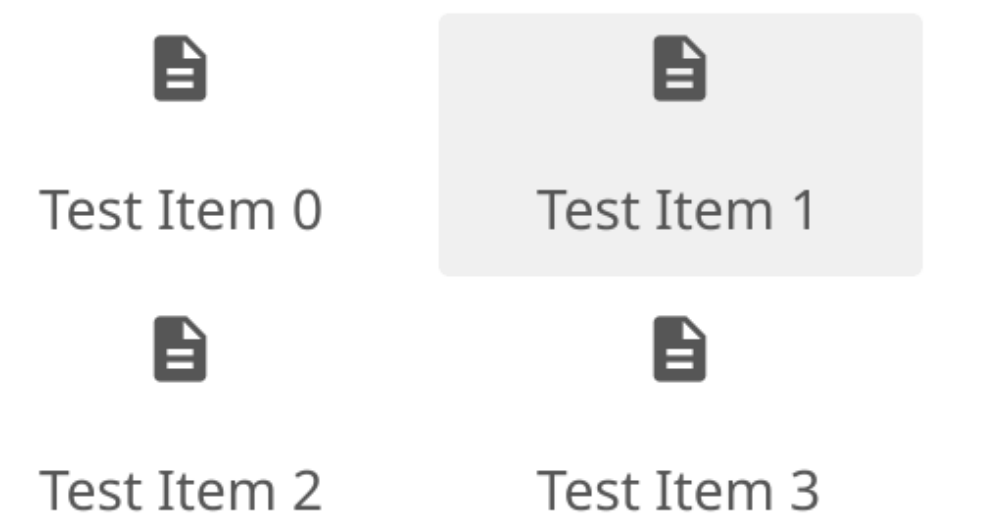
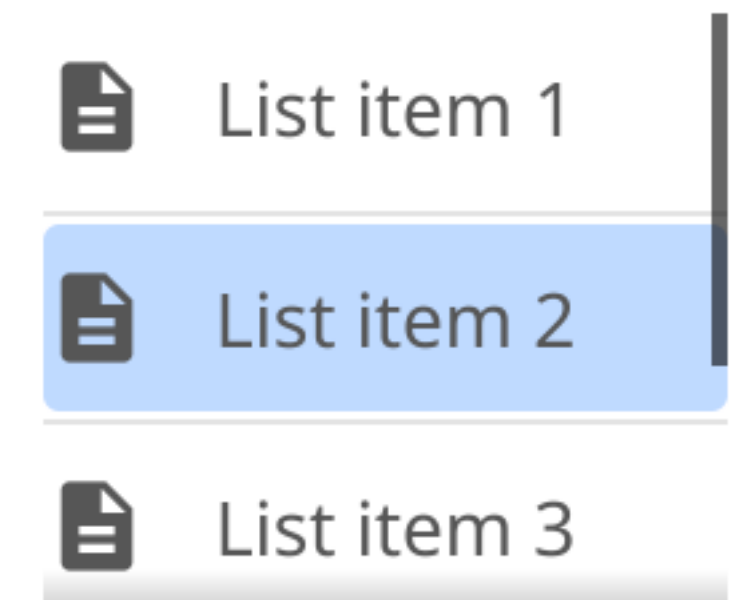
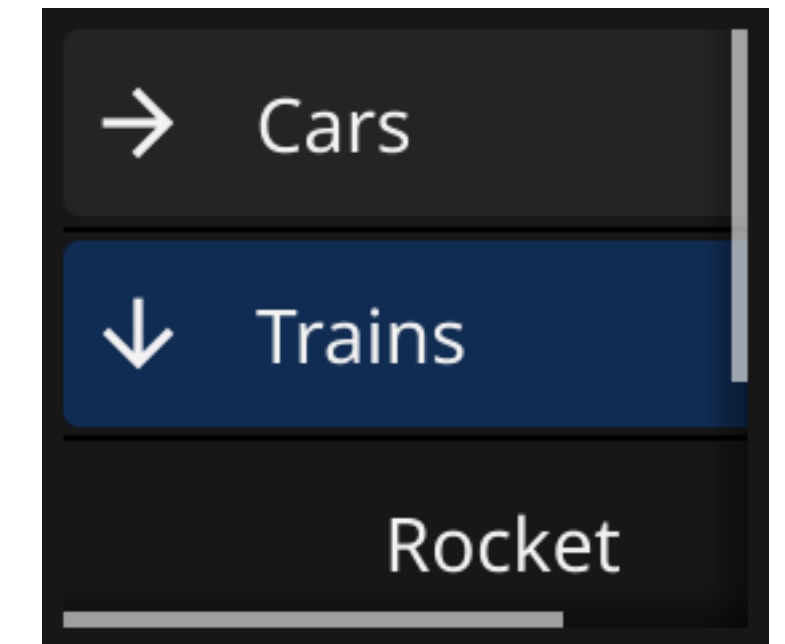
Select one or type ▼

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])  
    },  
)
```

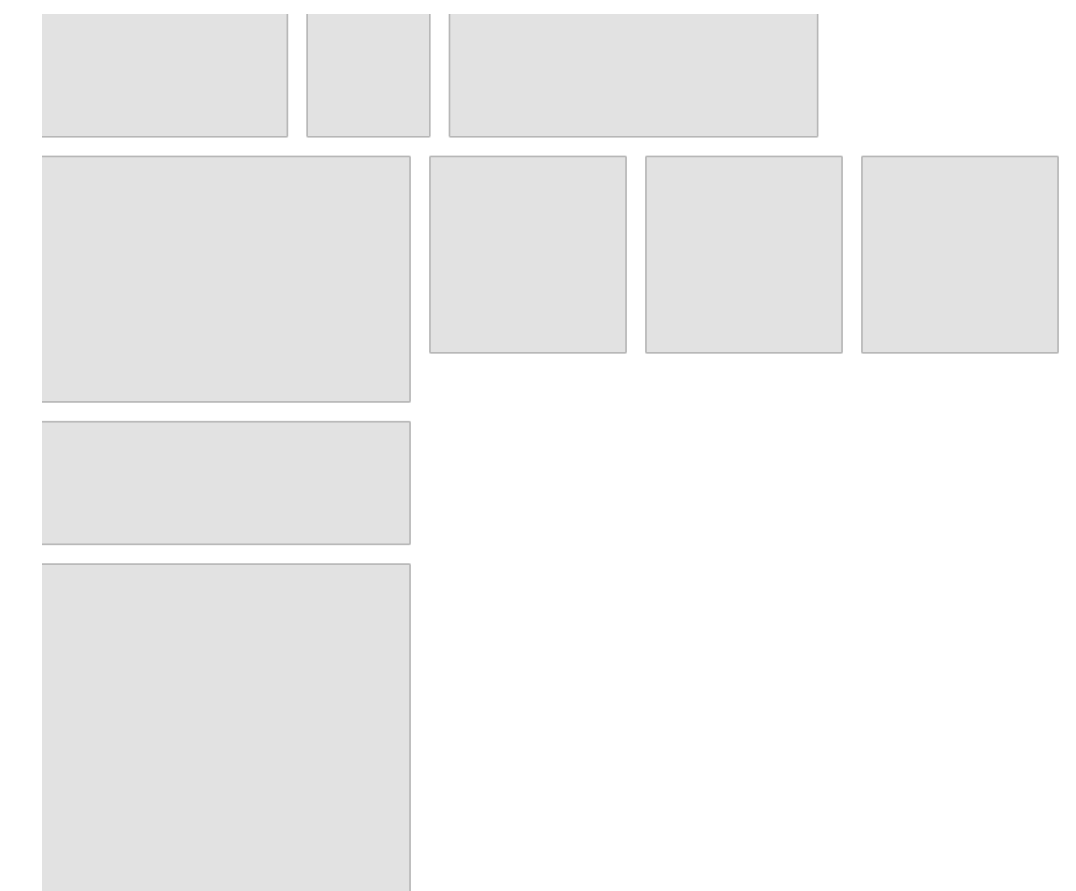
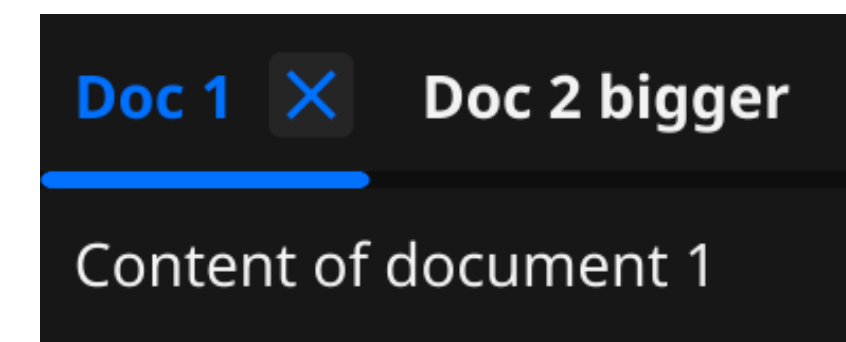
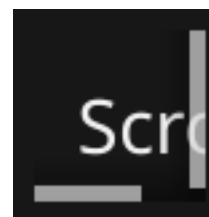
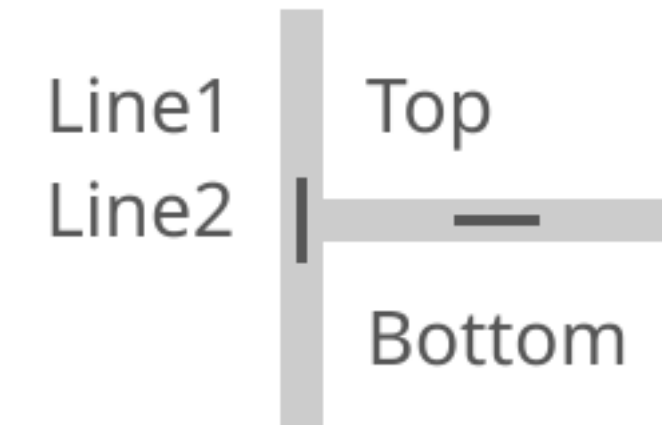
	A	B
1	A longer cell	Cell 1, 2
2	A longer cell	Cell 2, 2



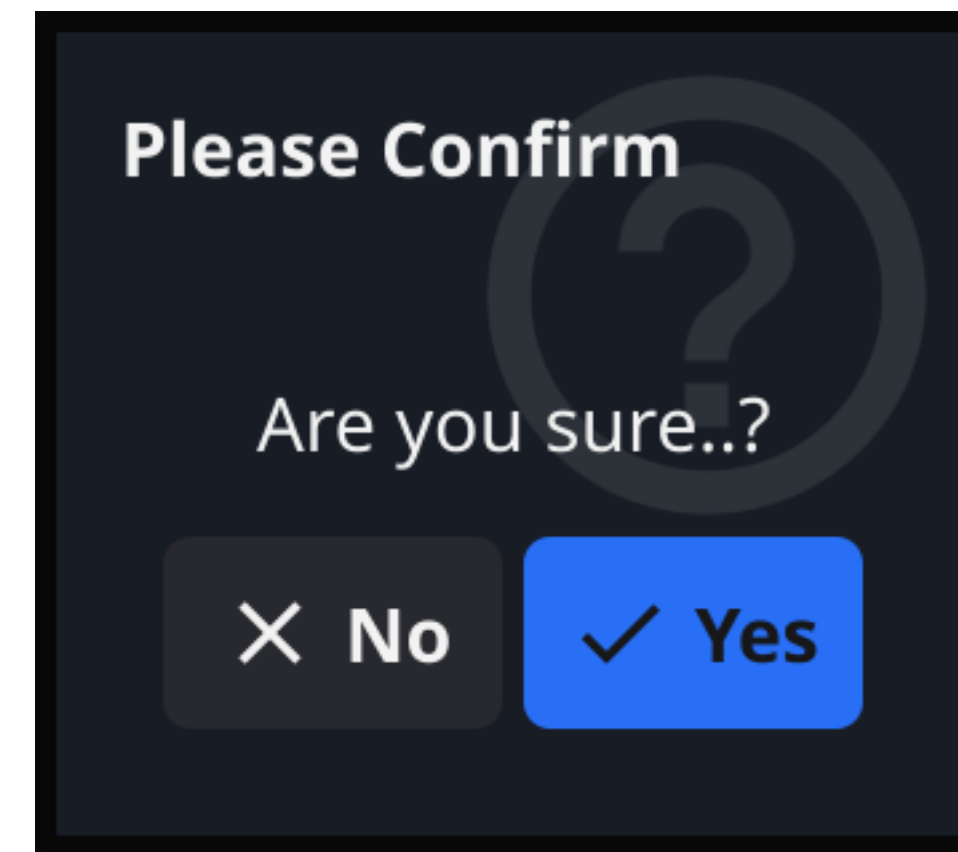
Containers



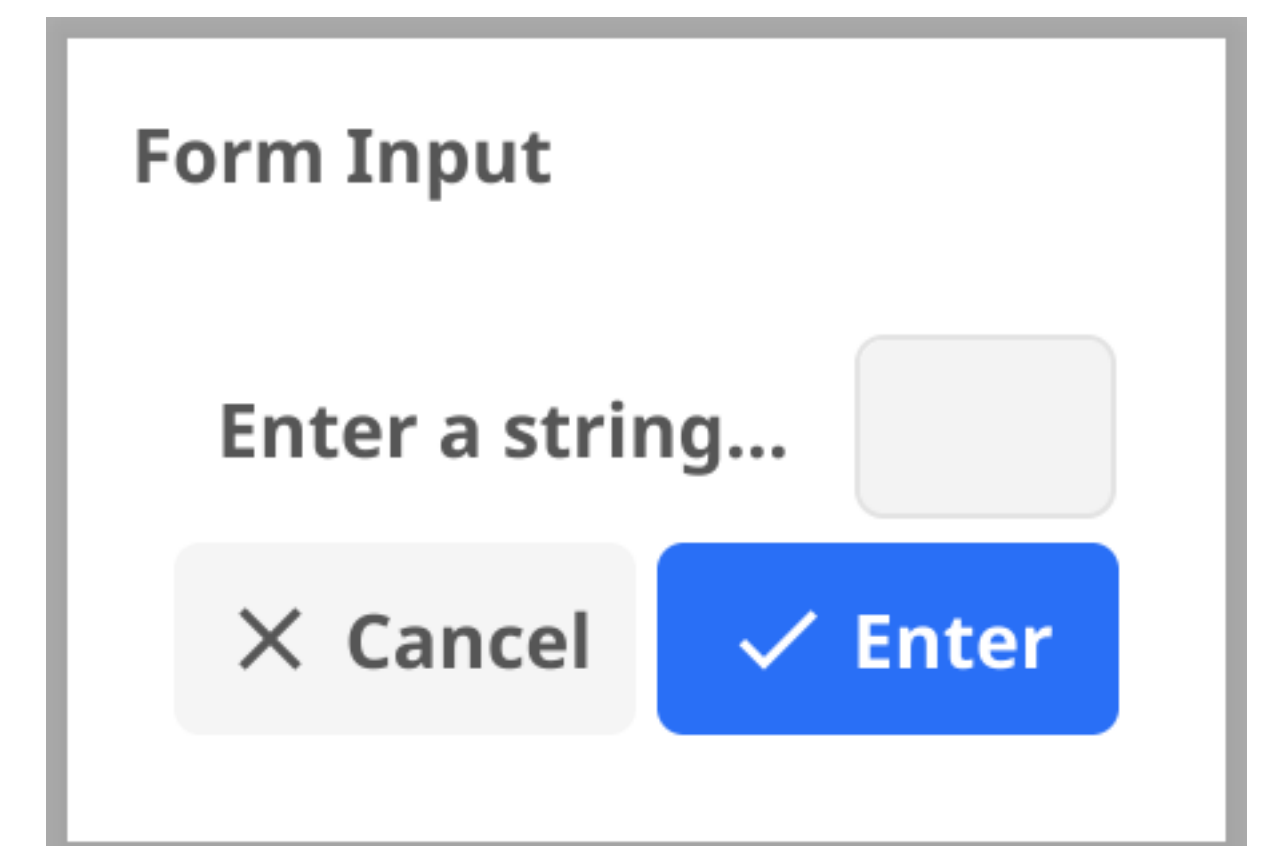
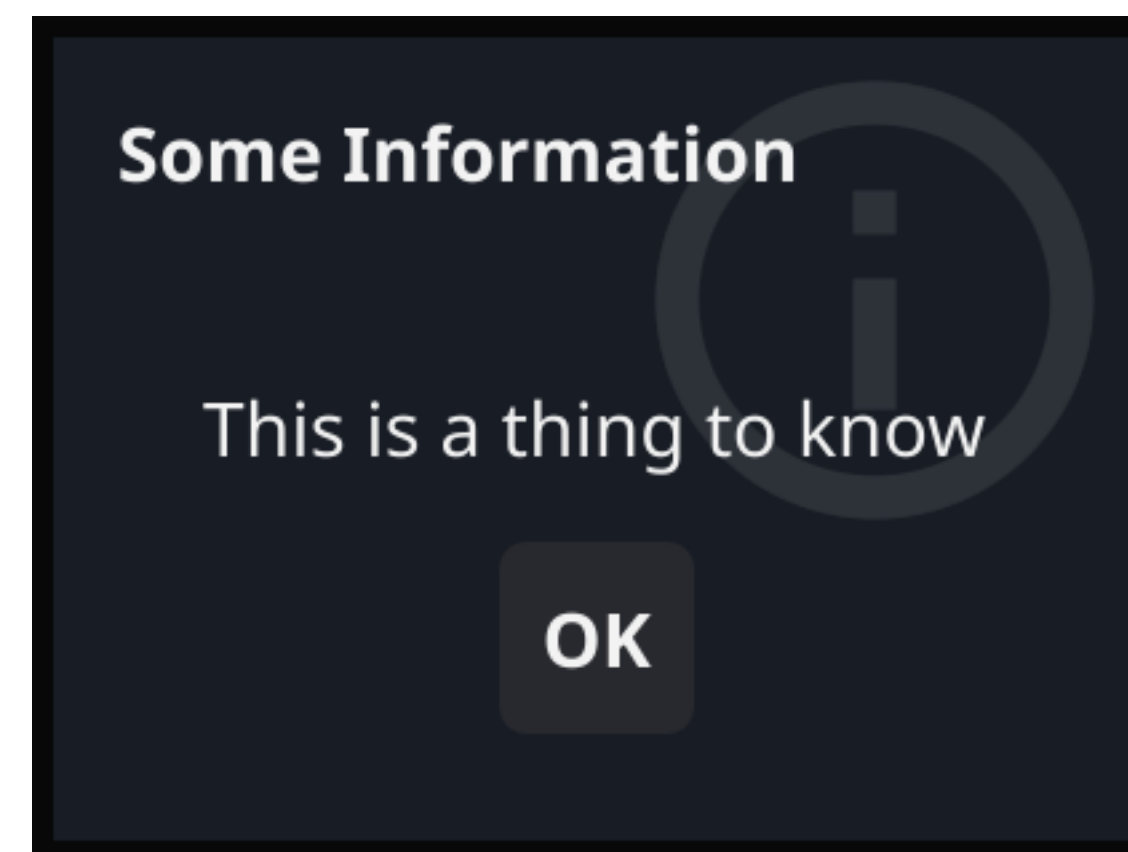
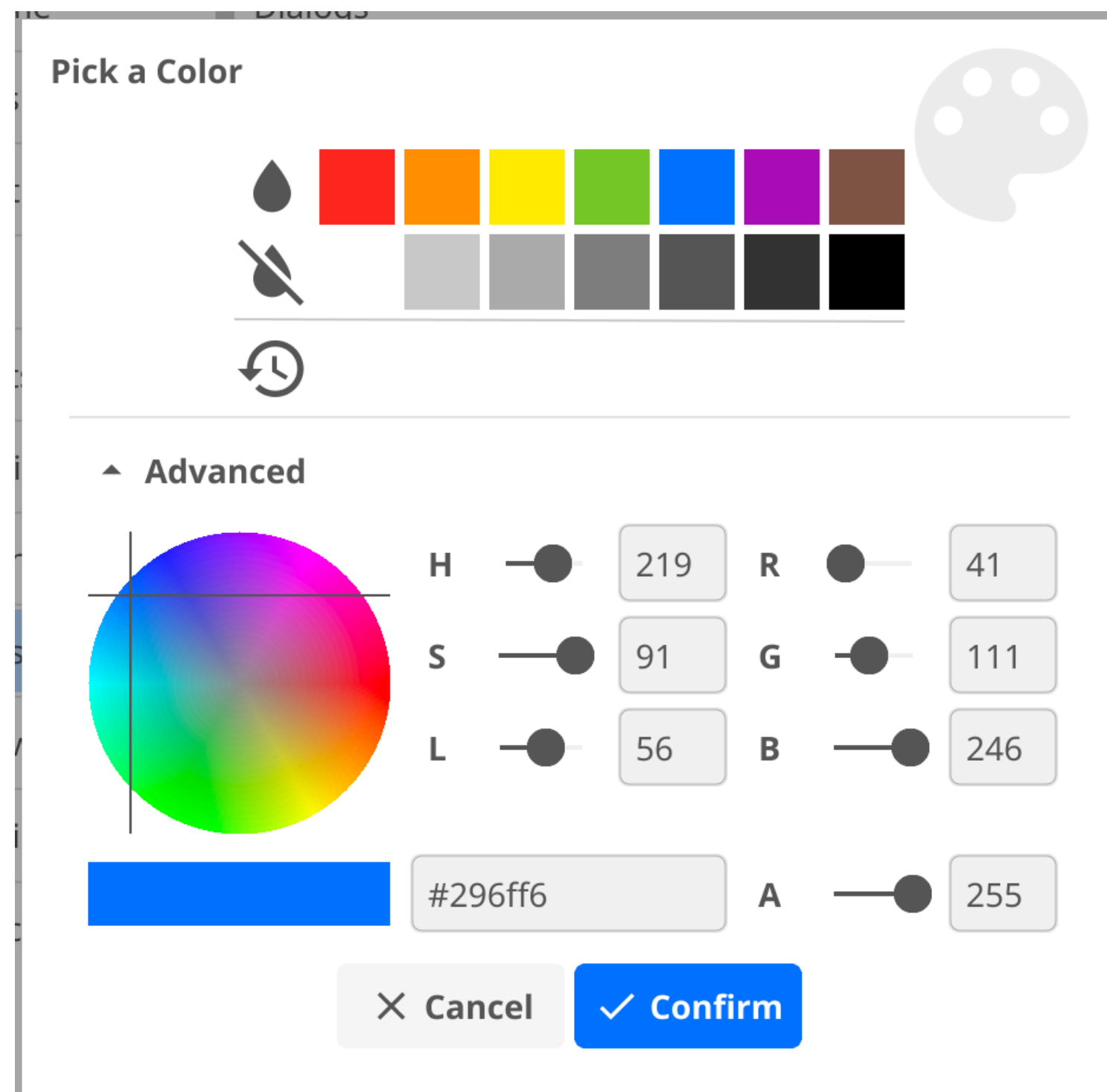
- `container.NewHSplit(left, right)`
- `container.NewAppTabs(items...)`
- `container.NewDocTabs(items...)`
- `container.NewScroll(content)`
- `container.New(layout, objects...)`



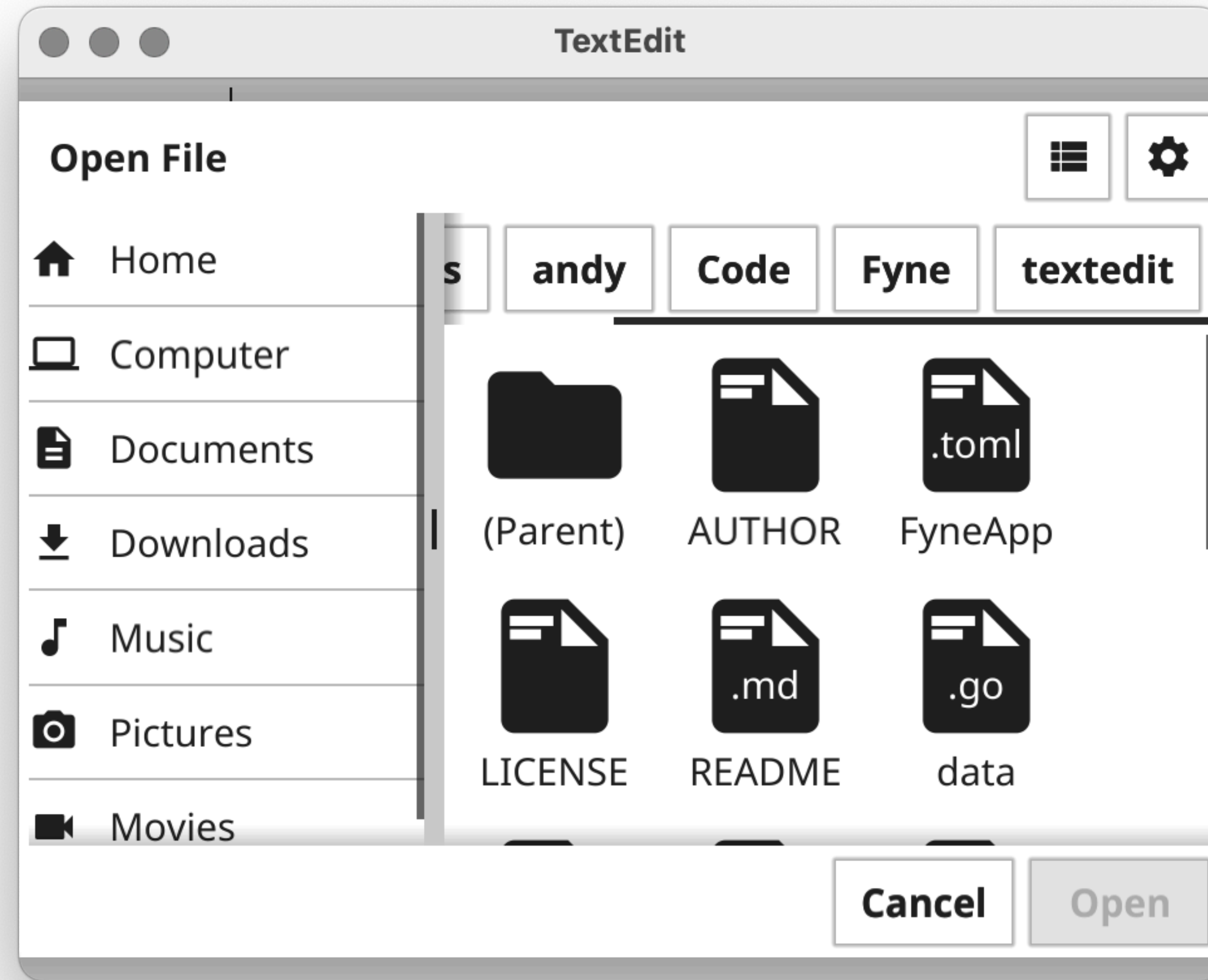
Dialogs



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



File and I/O



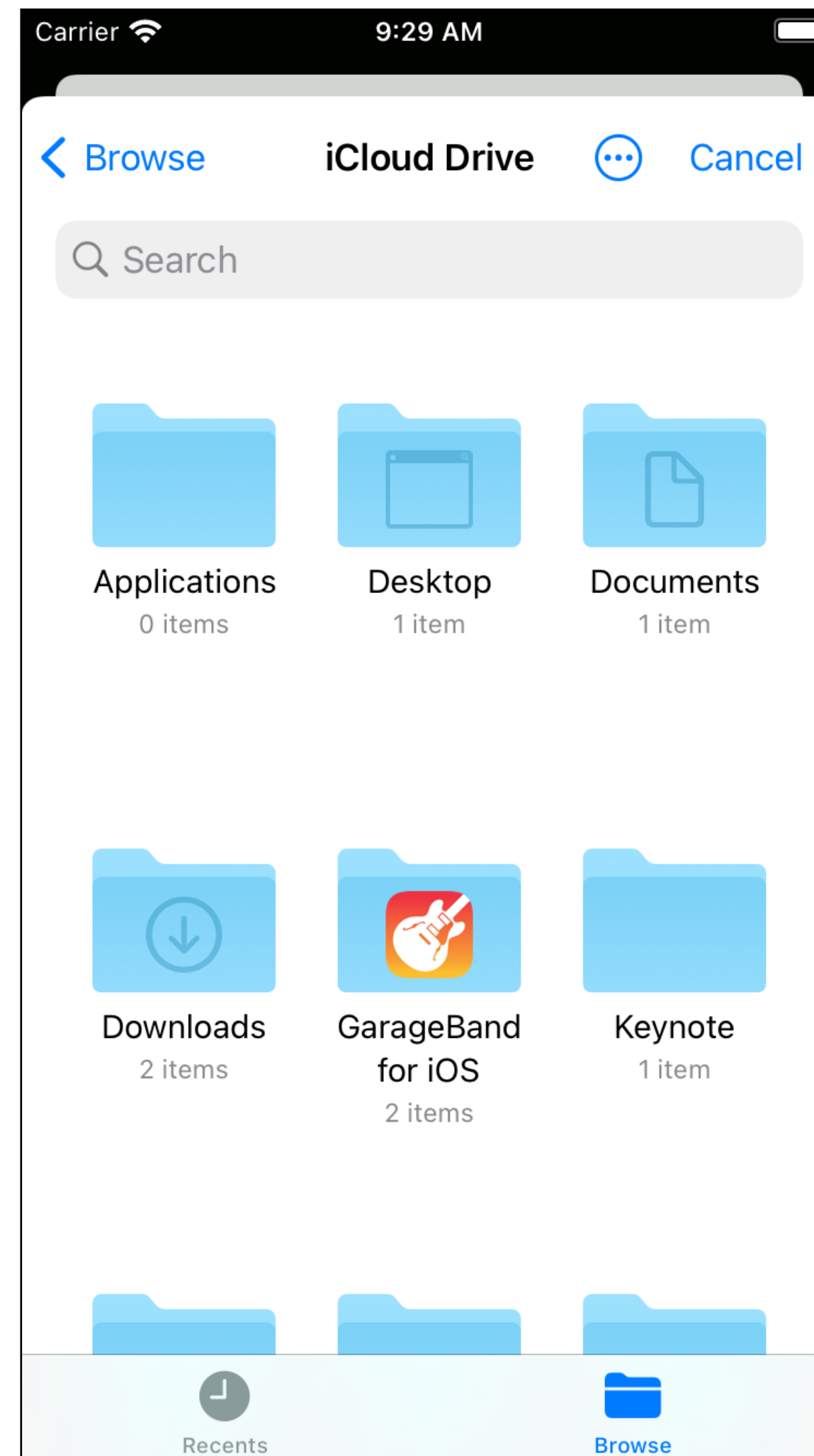
File and I/O - Code



```
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



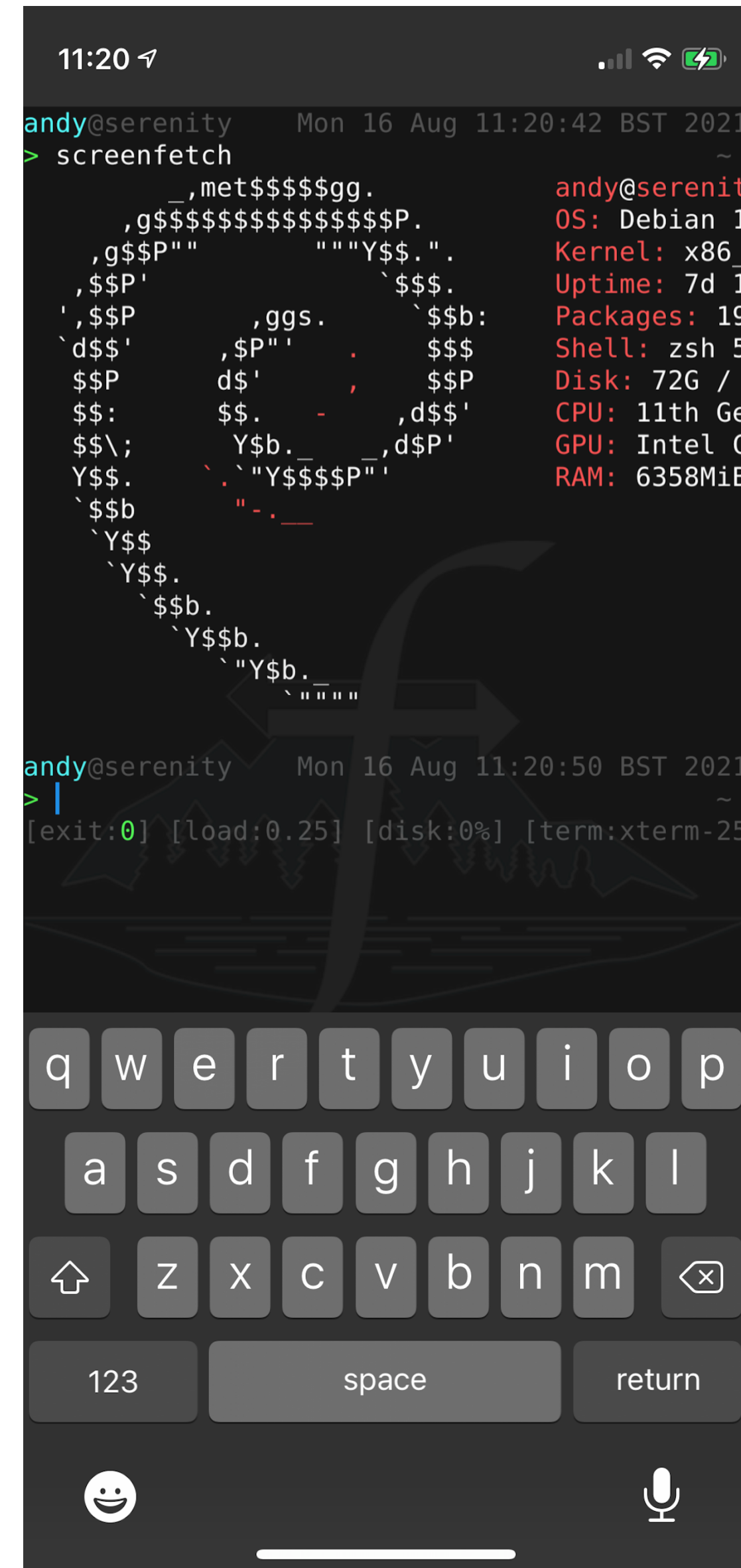
Additional possibilities...



- `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!

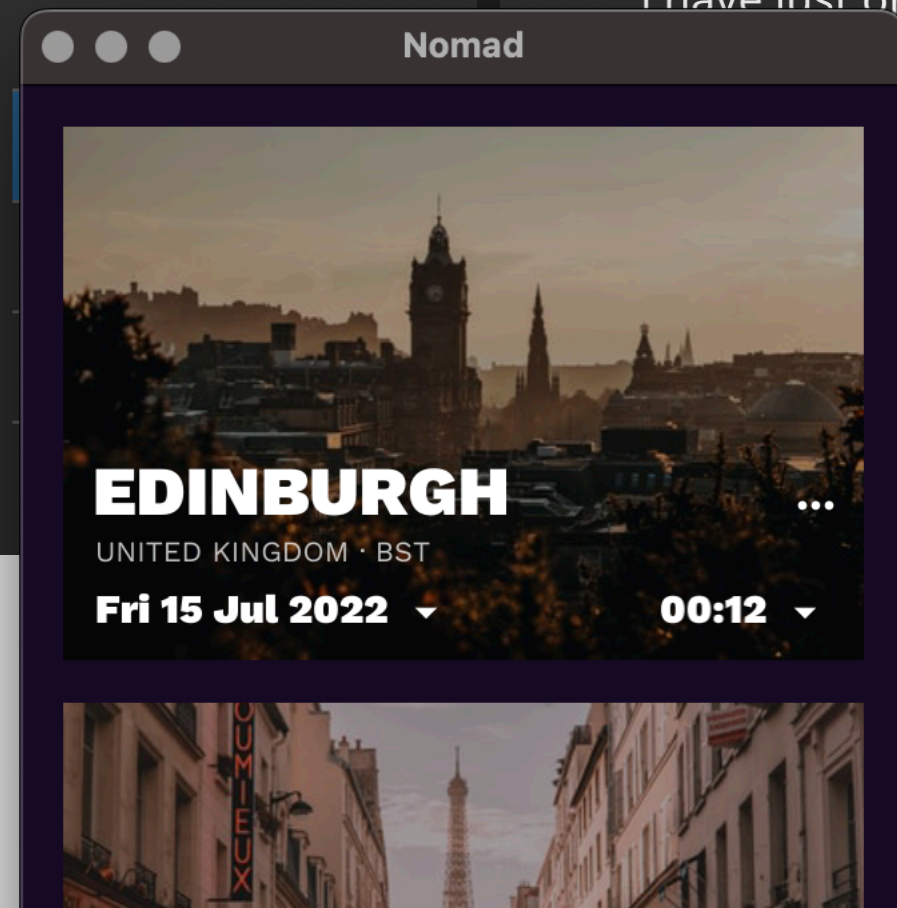
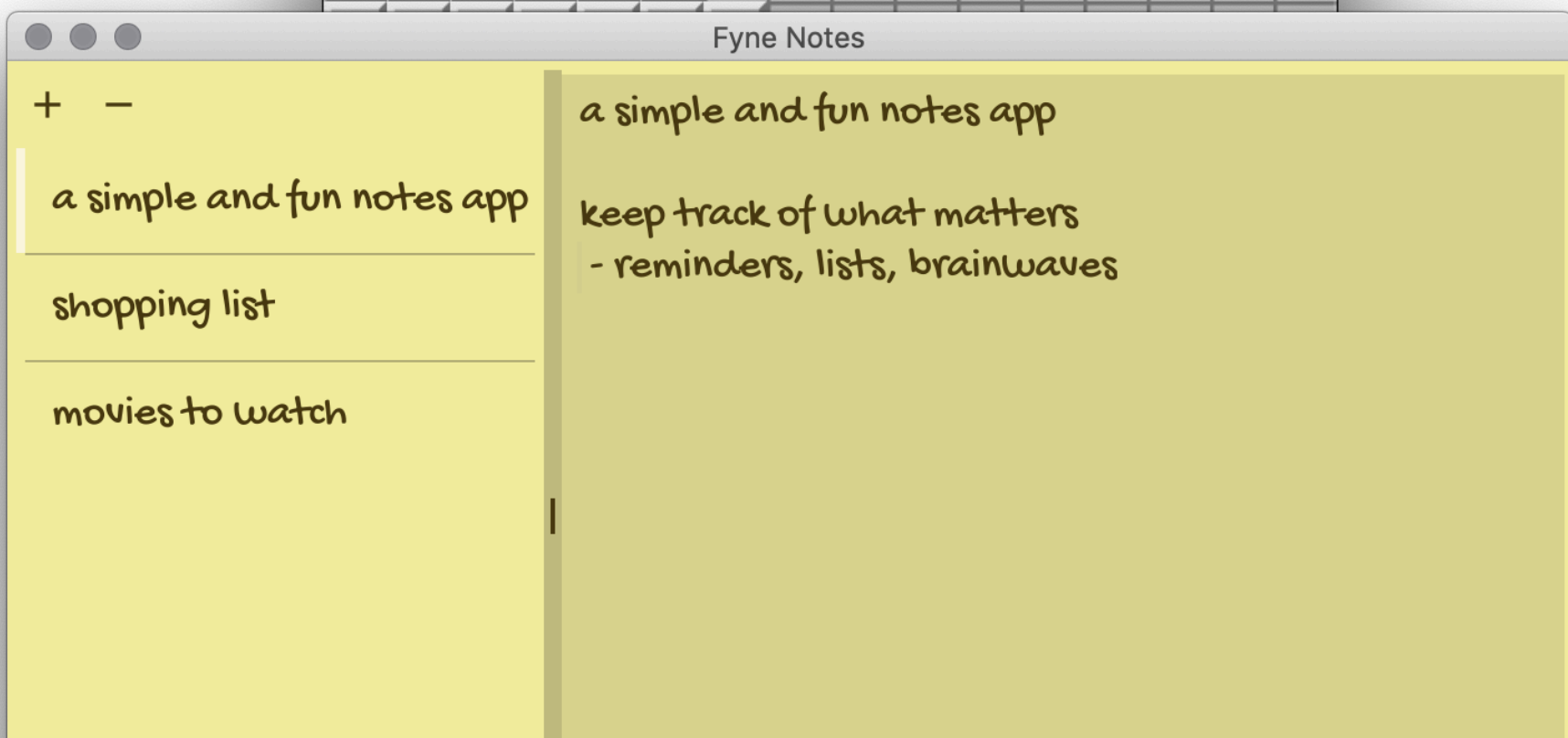
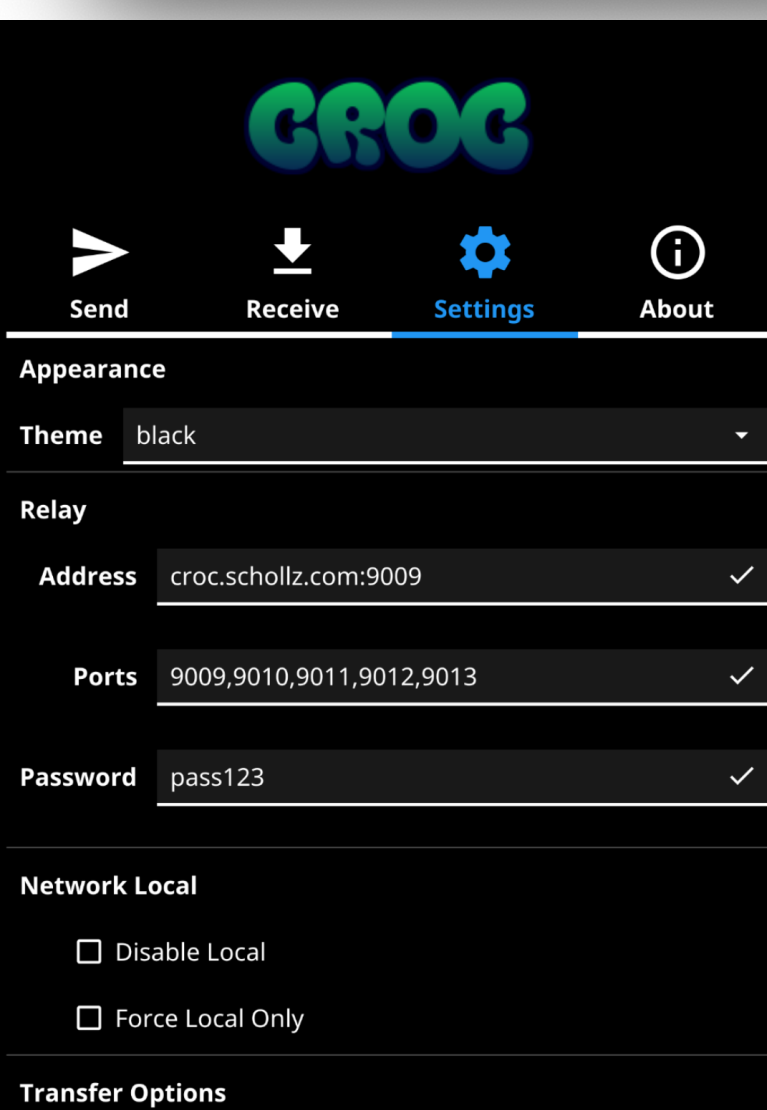
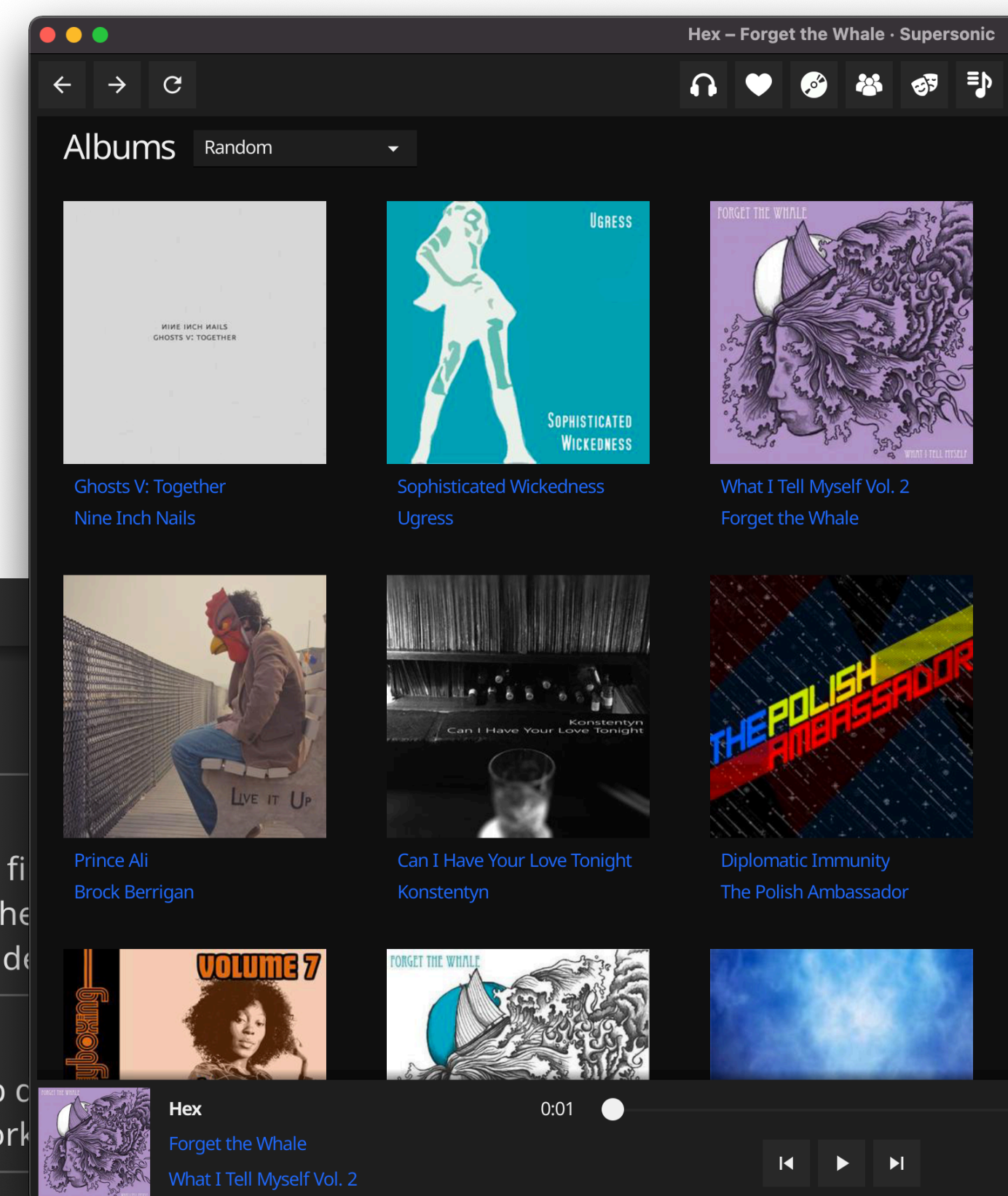
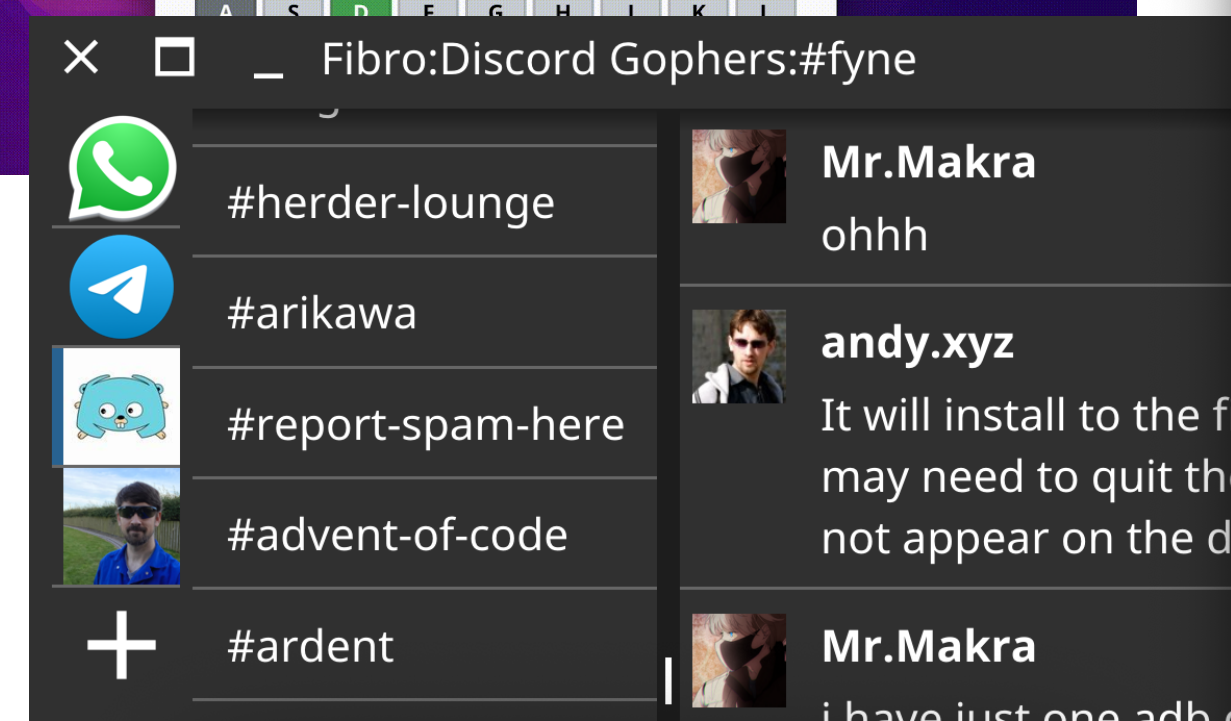
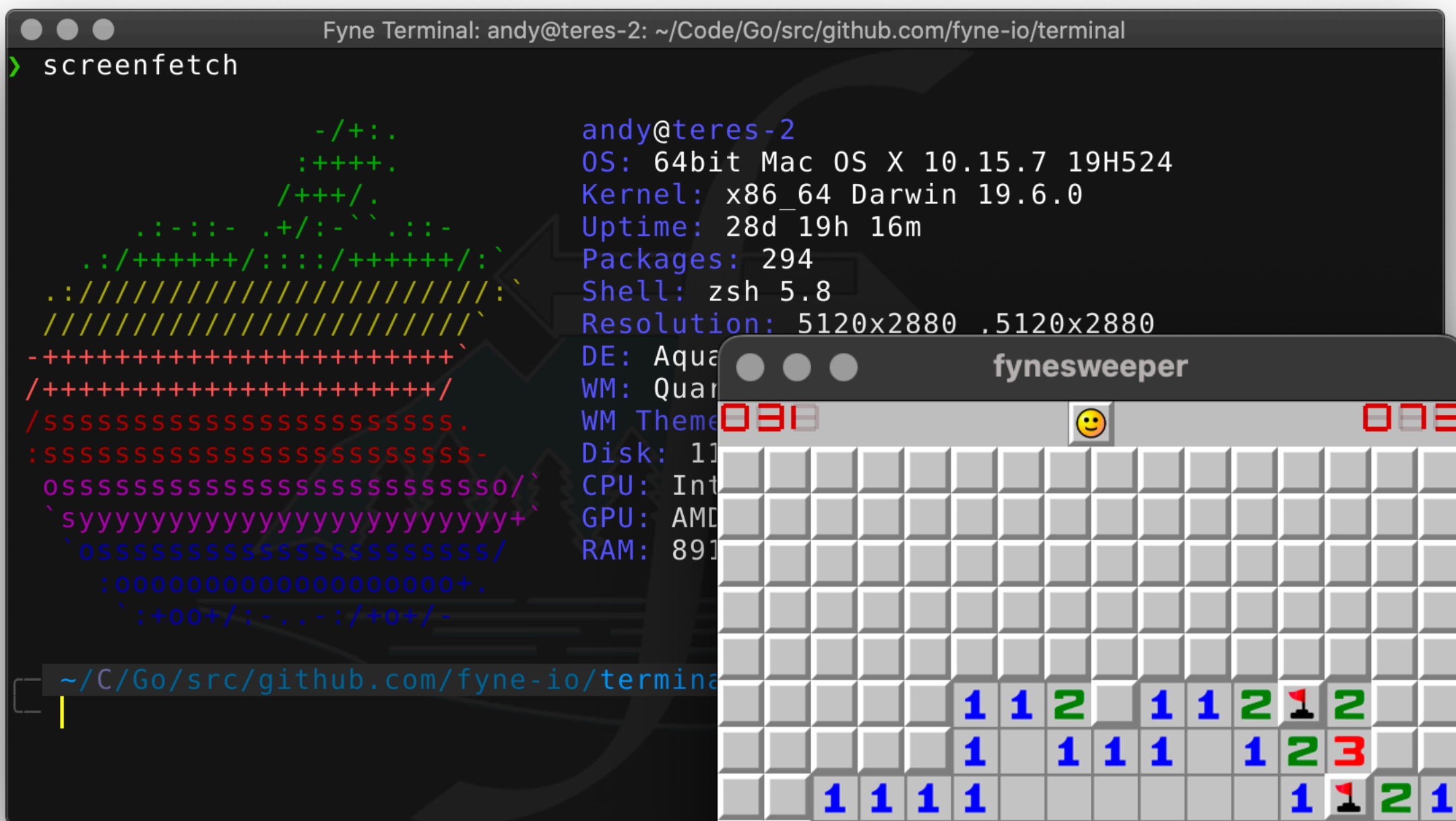
- Just import package and call constructor
- Works like any widget
 - `map := xWidget.NewMap()`
 - `cmdline := terminal.New()`



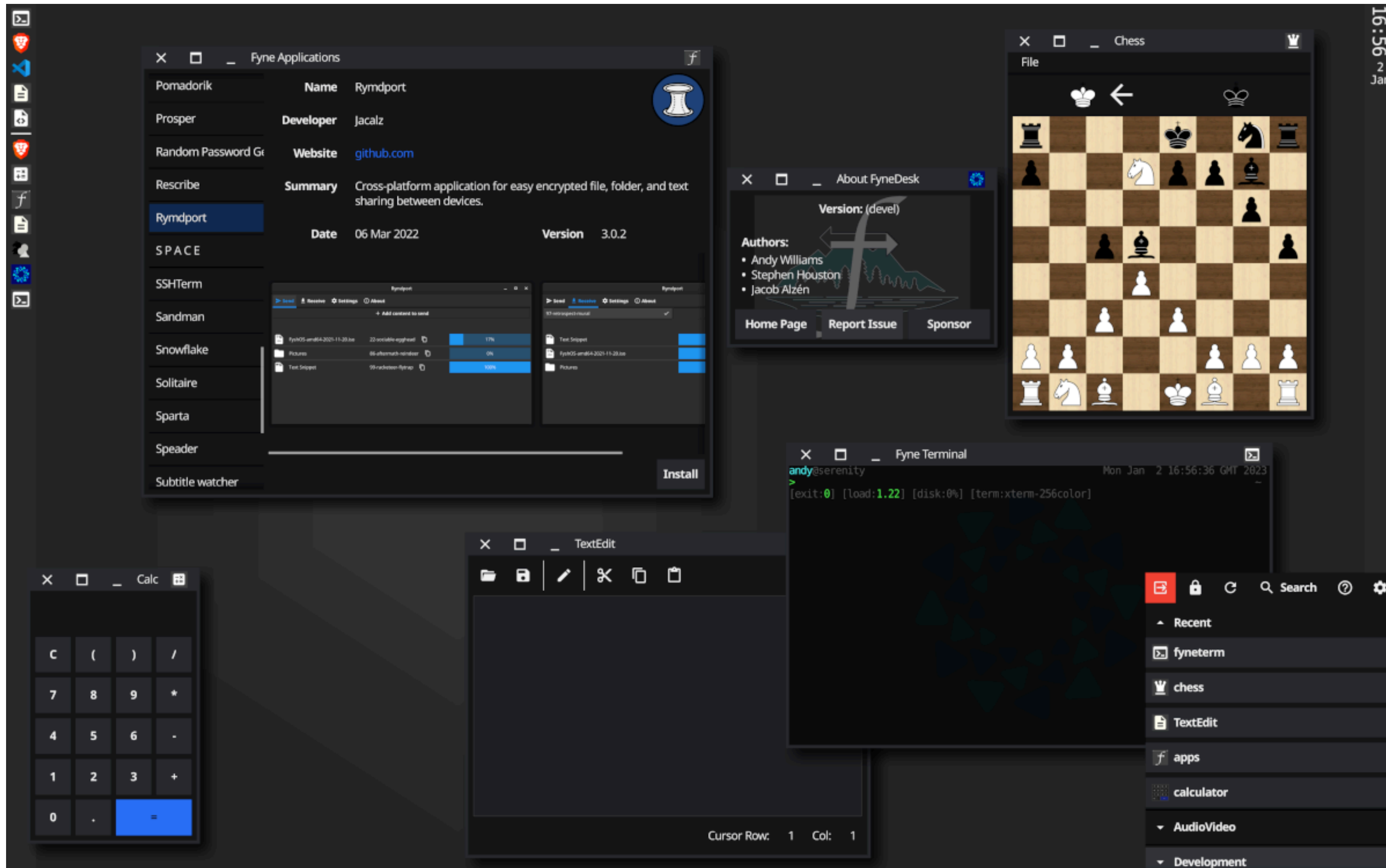
So many possibilities!



Other Fyne apps



And FyshOS...



And Fysion!



A screenshot of the Fysion IDE interface for building a calculator application. The window title is "Fysion App: Calculator". The top bar includes a home icon, the text "Calculator", the fyne labs logo, and "Run" and "Install" buttons. Below the top bar, there are tabs for "Home", "main.gui", and "main.go". The main workspace is titled "Desktop" and contains a preview of a calculator application window. The left sidebar shows a tree view of the application structure: "Screens" (Main) and "Other Files" (go.mod, main.go). The right sidebar is a "Widgets" panel with settings for "Type" (Container), "ID", "Layout" (Grid), "Count" (2), and "Arrange in" (Columns). A list of widget types is shown below, including Container, Accordion, Button, Card, Check, Entry, Form, and Hyperlink, with an "Insert" button at the bottom.

Hot off the GoLab press...



Build Results


GoLab 2023

Build Status


 Success 19 Nov 2023 23:19:16 UTC

 Publish


Android

 Success



 Universal


macOS


 Success

 Intel processor

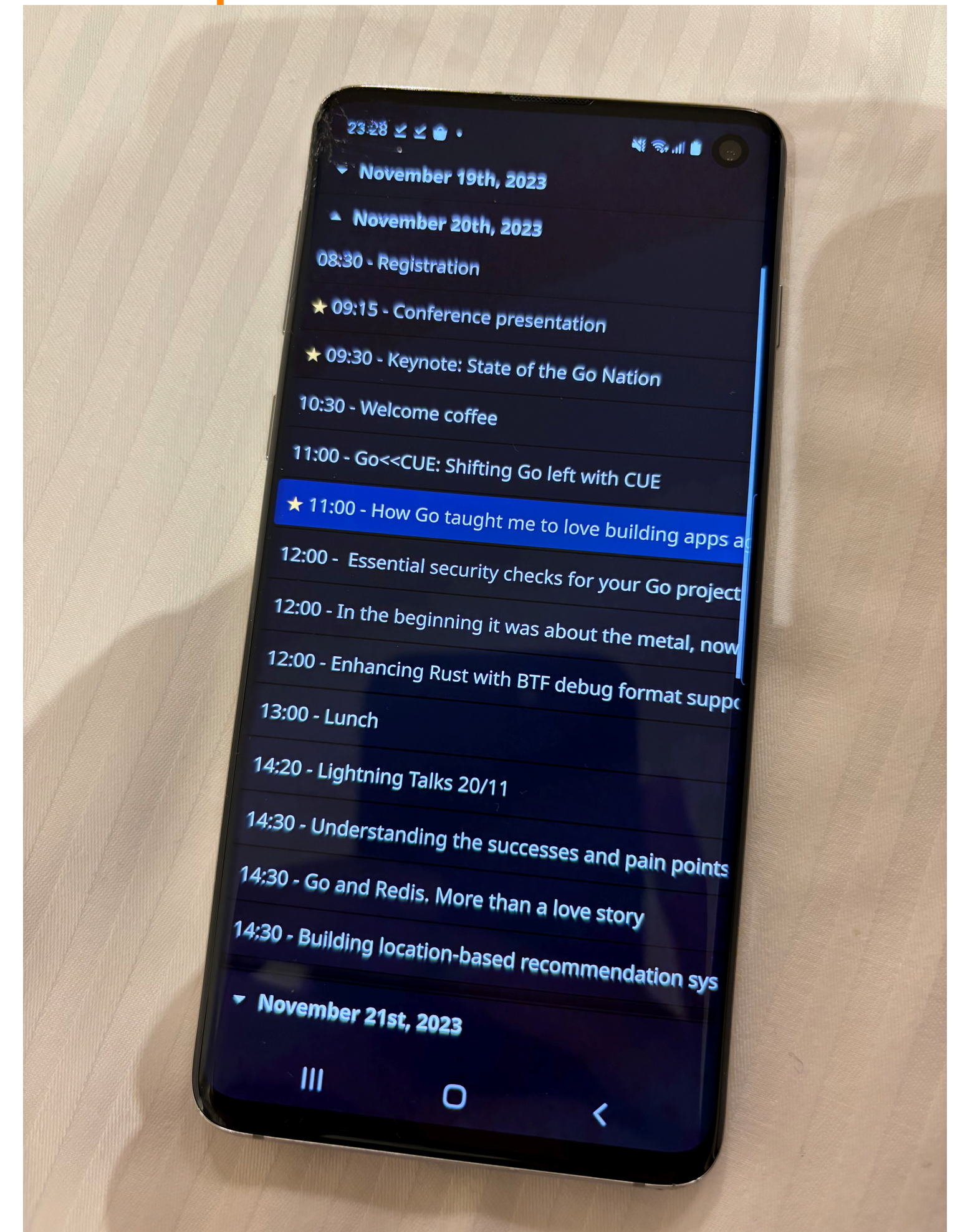
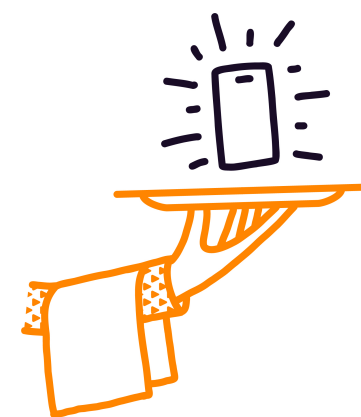
 Apple M1/2

Linux

 Success

 Amd64

 Arm64



More about Fyne



- Learn more about using Fyne

<https://developer.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