Fyne apps from Start to Store

Andrew Williams - GoLab

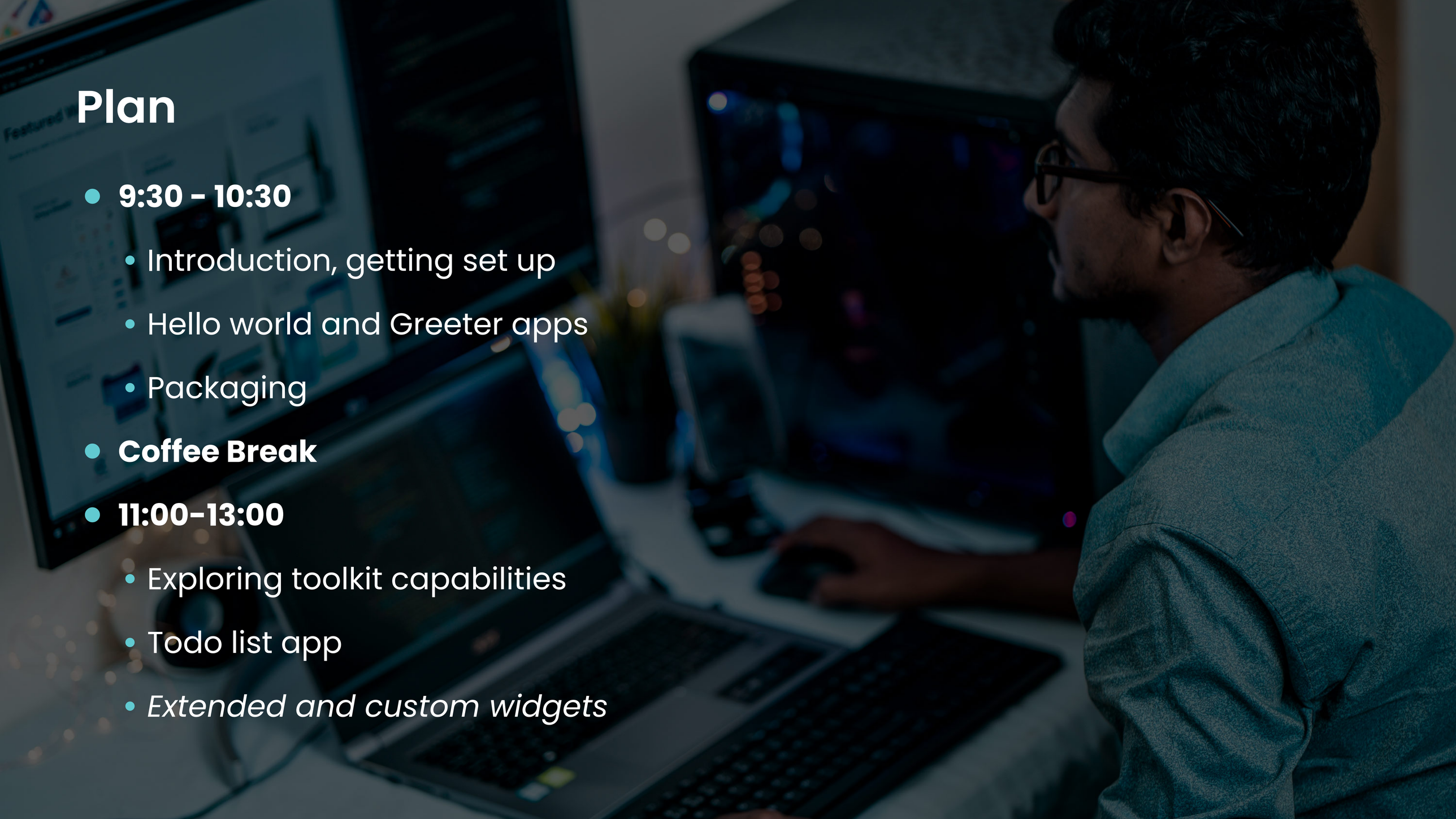
11th November 2023

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

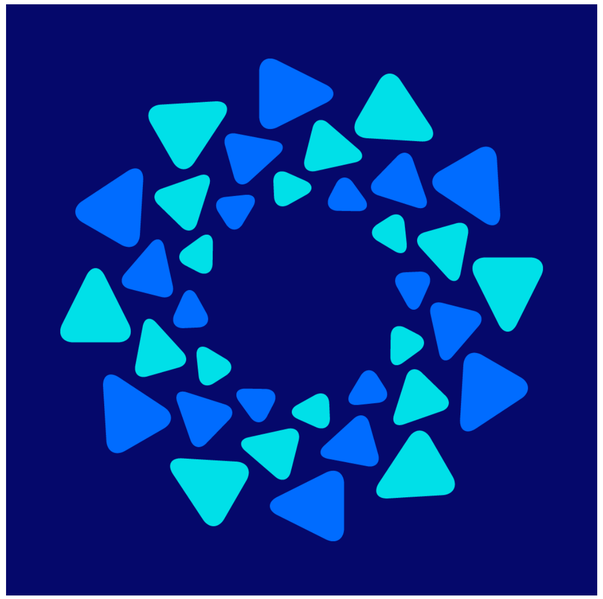
Plan

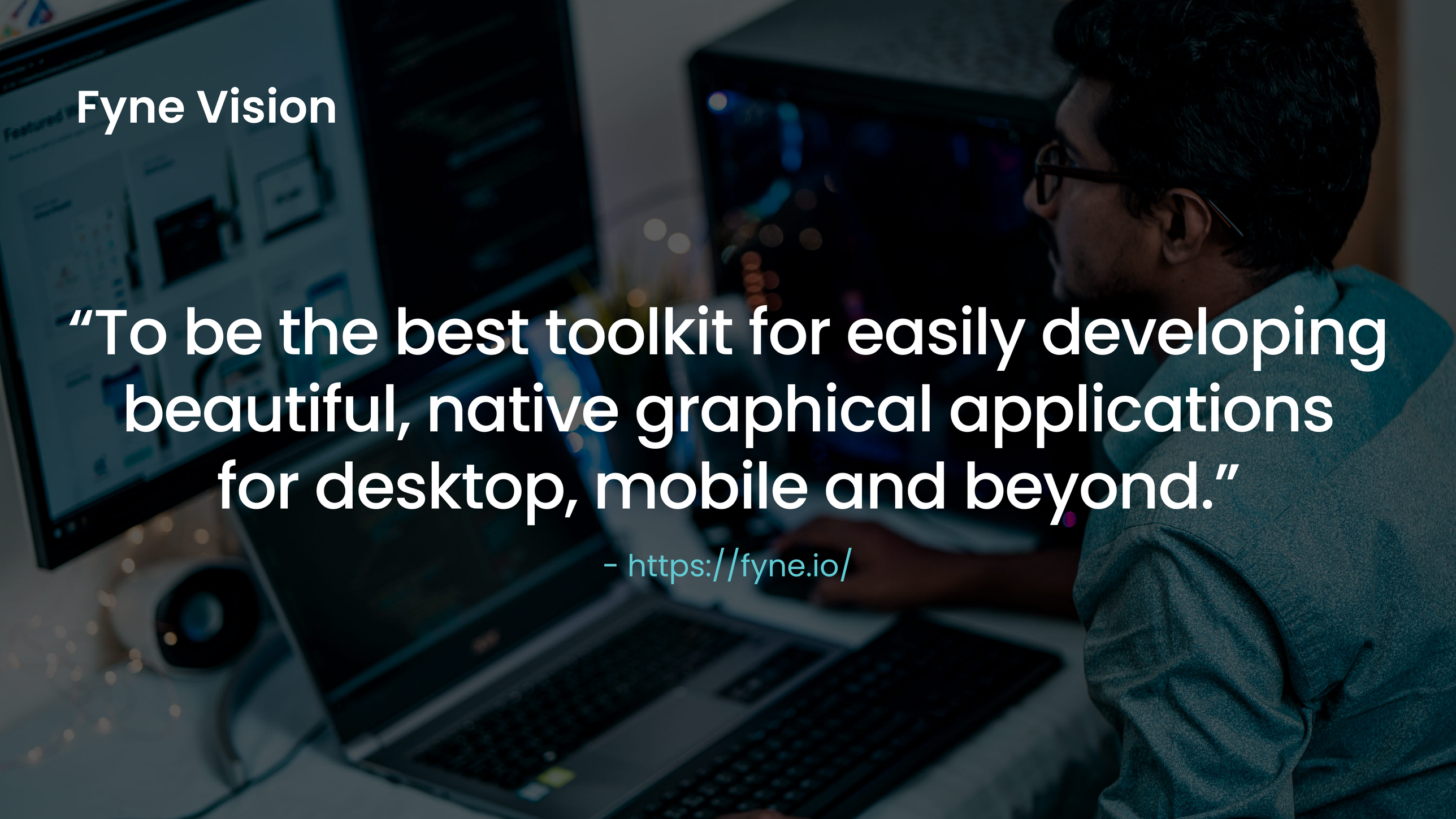
- **9:30 – 10:30**
  - Introduction, getting set up
  - Hello world and Greeter apps
  - Packaging
- **Coffee Break**
- **11:00–13:00**
  - Exploring toolkit capabilities
  - Todo list app
  - *Extended and custom widgets*



Fyne Aims

- Write once, run anywhere
- Apps that just work, do not require libraries or setup
- Leverage modern languages and techniques
- Great looking apps as standard
- Lower barrier of entry to building GUI apps
- Promote good engineering principles too





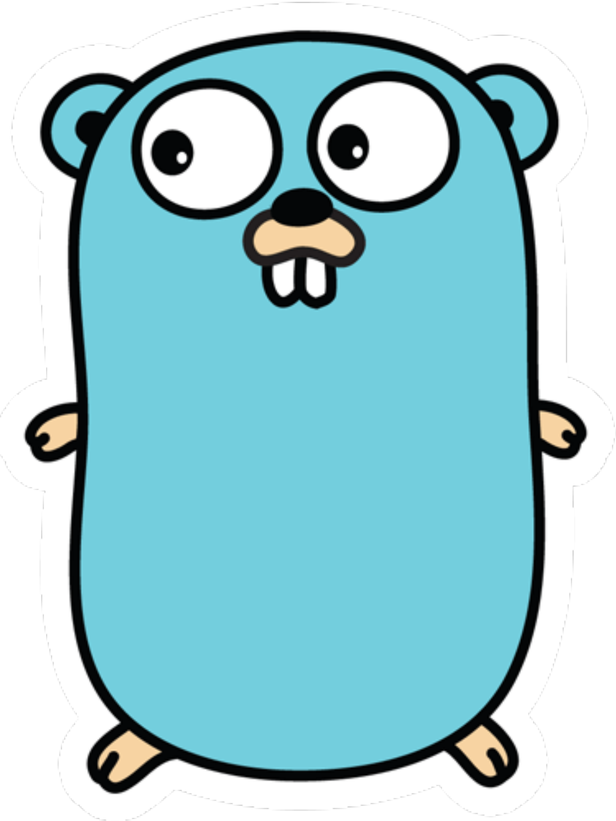
Fyne Vision

"To be the best toolkit for easily developing beautiful, native graphical applications for desktop, mobile and beyond."

- https://fyne.io/

Introduction

- Go
- (And C)
- Fyne
- (And OpenGL)
- Behaviour vs Control
- Consistency vs Customisation



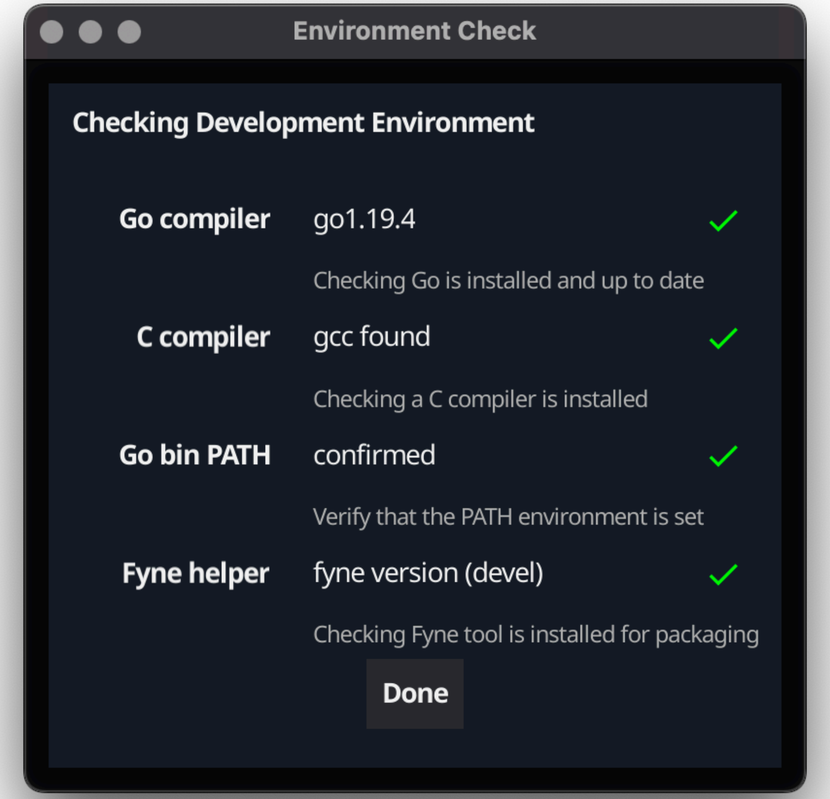
Install Developer Tools

- Install Go and Git
- Install Fyne helper tool (`go install` fyne.io/fyne/v2/cmd/fyne@latest)
- Set up C / graphics compile tools:
- *Windows*:
  - MSYS2 & Mingw64
- *macOS*:
  - Xcode
- *Linux / BSD*:
  - Gcc, system library headers (see docs)

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

Setup test

- Run in your terminal:
  - `fyne --version`
- Or use the setup tool from:
  - https://docs.fyne.io/started
- More hints and tips at
  - https://docs.fyne.io/faq/troubleshoot



Time to code!

```
mkdir myproject; cd myproject

go mod init myproject

go get fyne.io/fyne/v2@latest

code main.go
```

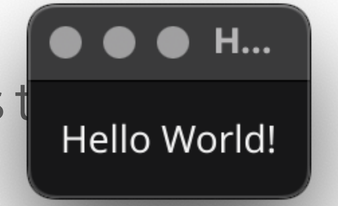
“hello” from https://github.com/fyne-io/workshops/GoLab2024

Hello World!

```go
package main

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

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

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

    w.ShowAndRun()
}
```

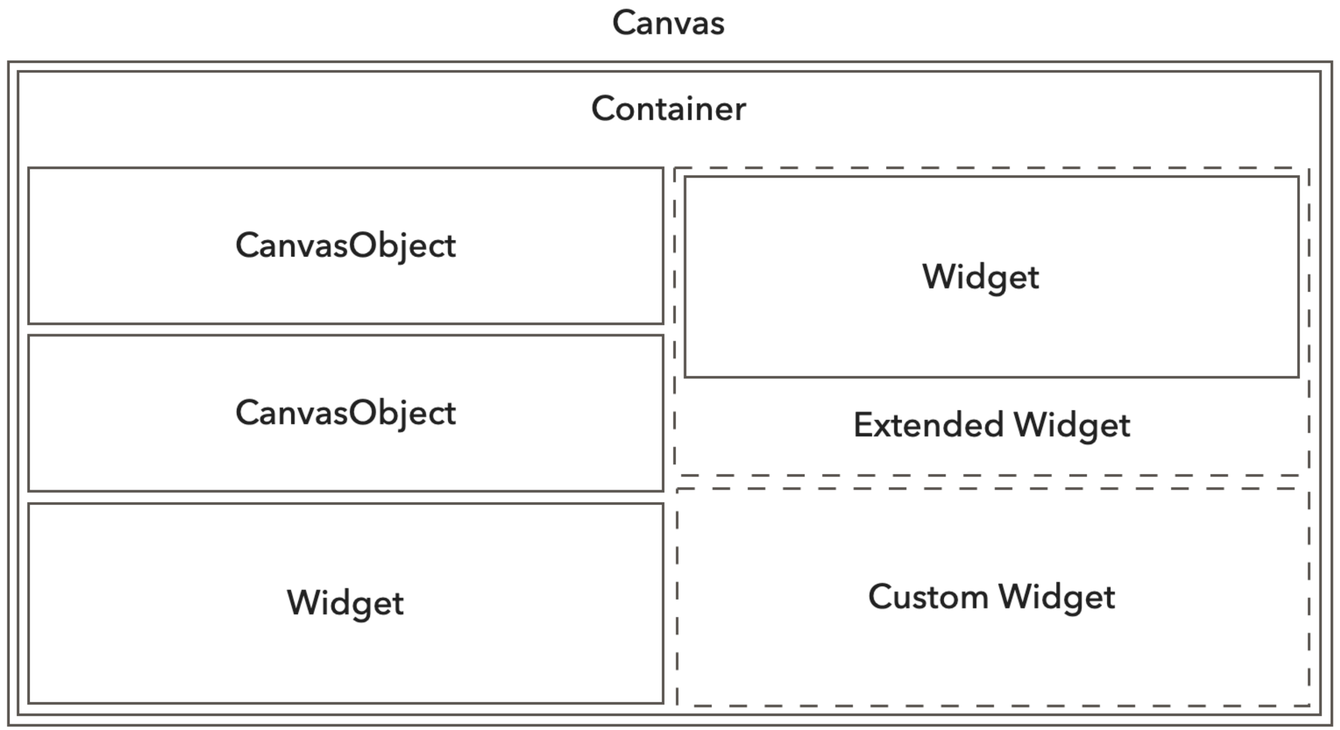
Run the App

```
mkdir myproject; cd myproject
go mod init myproject
go get fyne.io/fyne/v2@latest
code main.go
go mod tidy
go run .
```



Terminology

- Window
- Canvas
- CanvasObject
- Widget
- Container



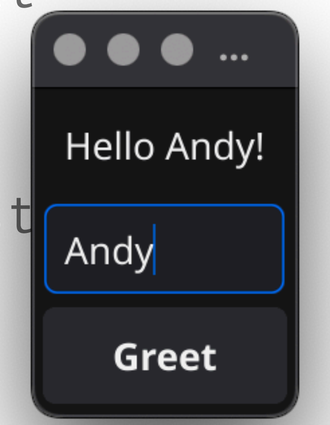
Container and Update

```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("Hello Person")

    out := widget.NewLabel("Hello World!")
    in := widget.NewEntry()
    in.SetPlaceHolder("World")
    w.SetContent(container.NewVBox(
        out, in,
        widget.NewButton("Greet", func() {
            out.SetText("Hello " + in.Text + "!")
        }),
    ))

    w.ShowAndRun()
}
```

“greeter” from https://github.com/fyne-io/workshops/GoLab2024

Run the App

```
mkdir myproject; cd myproject
go mod init myproject
go get fyne.io/fyne/v2@latest
code main.go
go mod tidy
go run .
```



Run the App

```
mkdir myproject; cd myproject
go mod init myproject
go get fyne.io/fyne/v2@latest
code main.go
go mod tidy
go run -tags debug .
```

Package & Install

- Remember Fyne helper tool (`go install fyne.io/fyne/v2/cmd/fyne@latest`)
- Add `Icon.png` and optional `FyneApp.toml`
- Package for sharing

```
fyne package
```

- Install on current computer

```
fyne install
```

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

[Details]
Icon = "img/fyne.png"
Name = "Fyne Demo"
ID = "io.fyne.demo"
Version = "1.0.0"
Build = 5
```

Building for Mobile/Other Platforms

- If you have the tools installed:

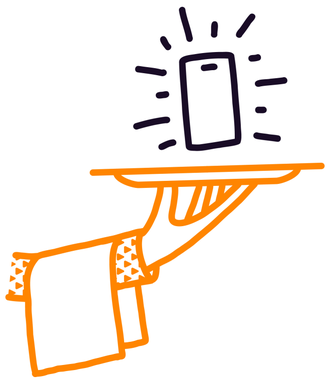
```
fyne package -os windows

fyne package -os android
```

- Otherwise...

https://docs.fyne.io/started/cross-compiling

Fyne-Cross and Other tools

- Avoid the need to manage developer tools for each platform
- Just install Docker or Podman
- Install Fyne Cross:
  - `go install github.com/fyne-io/fyne-cross@latest`
- Run the package using fyne-cross:
  - `fyne-cross windows`
- Also possible to use other tools like GitHub Actions or FyneLabs Geoffrey



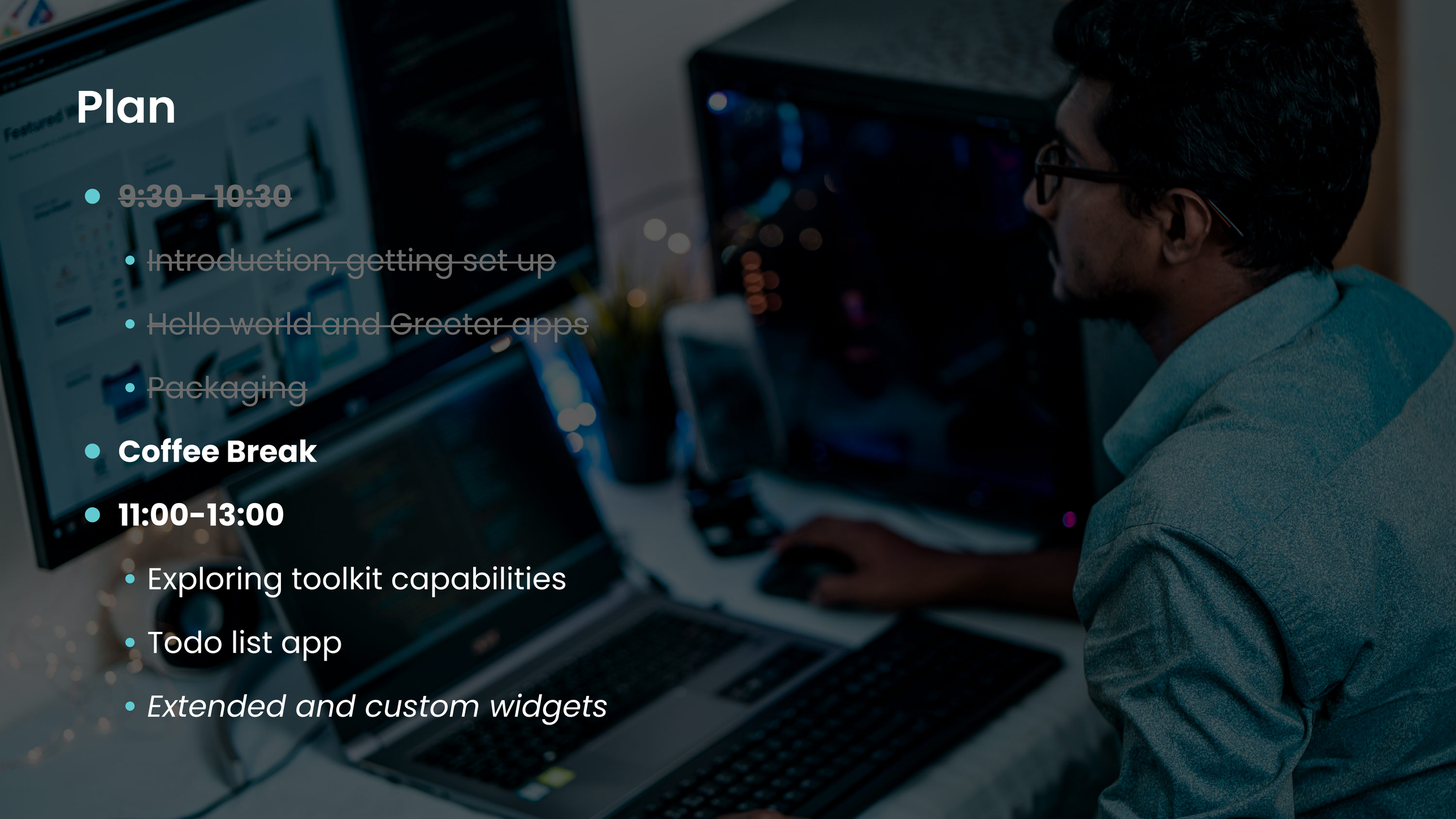
https://fynelabs.com/geoffrey

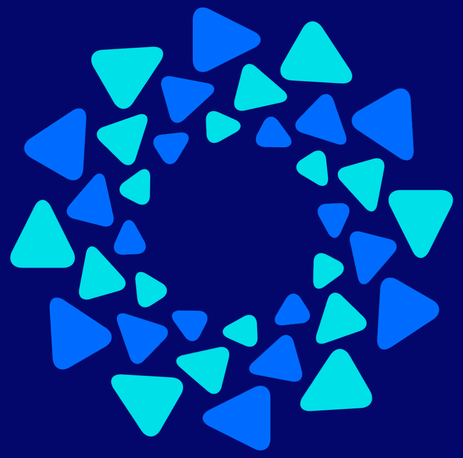
Congratulations!

- Up and running with Fyne
- Built your first app
- For many different computers!

Plan

- ~~**9:30 - 10:30**~~
  - ~~Introduction, getting set up~~
  - ~~Hello world and Greeter apps~~
  - ~~Packaging~~
- **Coffee Break**
- **11:00-13:00**
  - Exploring toolkit capabilities
  - Todo list app
  - *Extended and custom widgets*





fyne

Exploring Toolkit Capabilities

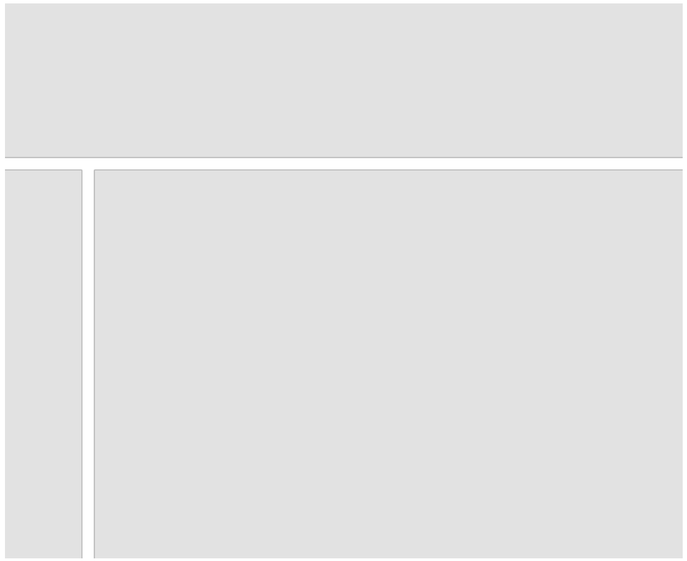
Collections, Preferences, Data Binding and more!

Introduction

- Border Container
- List & Collection Widgets
- Preference storage

Building a TODO list

- Border Container (top, bottom, left, right, middle)
- Text input and “Add” button
- List of items
- Checkbox to mark done



https://developer.fyne.io/explore/layouts

List Widget

- Part of the “collections” widgets (List, Tree, Table and GridWrap)
- Optimised for large data
- Re-use of graphical objects for performance

Time to code!

```
mkdir mytodo; cd mytodo

go mod init mytodo

go get fyne.io/fyne/v2@latest

code main.go
```

“hello” from https://github.com/fyne-io/workshops/GoLab2024

And main Function

```go
package main

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

func main() {
    a := app.NewWithID("io.fyne.workshop.todo")
    w := a.NewWindow("TODO")

    w.SetContent(makeUI())

    w.Resize(fyne.NewSize(180, 240))
    w.ShowAndRun()
}
```

“todo” from https://github.com/fyne-io/workshops/GoLab2024

Time to code!

```
mkdir mytodo; cd mytodo

go mod init mytodo

go get fyne.io/fyne/v2@latest

code main.go

code ui.go
```

“hello” from https://github.com/fyne-io/workshops/GoLab2024

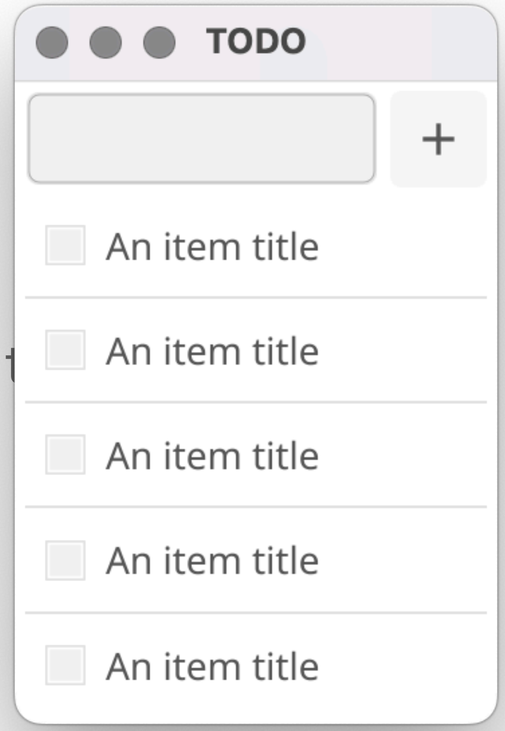
Coding the layout

```go
package main

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

func makeUI() fyne.CanvasObject {
    list := widget.NewList(
        func() int {
            return 5
        },
        func() fyne.CanvasObject {
            return widget.NewCheck("An item title", nil)
        },
        func(i widget.ListItemID, co fyne.CanvasObject) {
        })

    input := widget.NewEntry()
    add := widget.NewButtonWithIcon("",
        theme.ContentAddIcon(), func() {})

    top := container.NewBorder(nil, nil, nil, add, input)
    return container.NewBorder(top, nil, nil, nil, list)
}
```

“todo” from https://github.com/fyne-io/workshops/GoLab2024

Time to code!

```
mkdir mytodo; cd mytodo
go mod init mytodo
go get fyne.io/fyne/v2@lat
code main.go
code ui.go
go run .
```



Preferences & Storage

- Storage is complex in platform-agnostic
- Preferences API for simple data
- storage package for more complex needs
- Avoid direct file access or io/fs to remain portable

Updating TODO Data

- Use preferences to store our user data
- Add new when we tap the add button
- Tapping the checkbox removes TODO
- Update user data on change

Completing the App

```
func makeUI(data []string) fyne.CanvasObject {
    var list *widget.List
    list = widget.NewList(
        func() int {
            return len(data)
        },
        func() fyne.CanvasObject {
            return widget.NewCheck("An item title", nil)
        },
        func(i widget.ListItemID, co fyne.CanvasObject) {
            ch := co.(*widget.Check)
            ch.SetText(data[i])

            ch.OnChanged = func(done bool) {
                if !done {
                    return
                }
                ch.SetChecked(false)

                data = slices.Delete(data, i, i+1)
                list.Refresh()
                fyne.CurrentApp().Preferences().SetStringList("todos", data)
            }
        })

...
```

“todo” from https://github.com/fyne-io/workshops/GoLab2024

Completing the App

```
...

    input := widget.NewEntry()
    add := widget.NewButtonWithIcon("",
        theme.ContentAddIcon(), func() {
            if input.Text == "" {
                return
            }

            data = append(data, input.Text)
            list.Refresh()
            input.SetText("")
            fyne.CurrentApp().Preferences().SetStringList("todos", data)
        })

    top := container.NewBorder(nil, nil, nil, add, input)
    return container.NewBorder(top, nil, nil, nil, list)
}
```

“todo” from https://github.com/fyne-io/workshops/GoLab2024

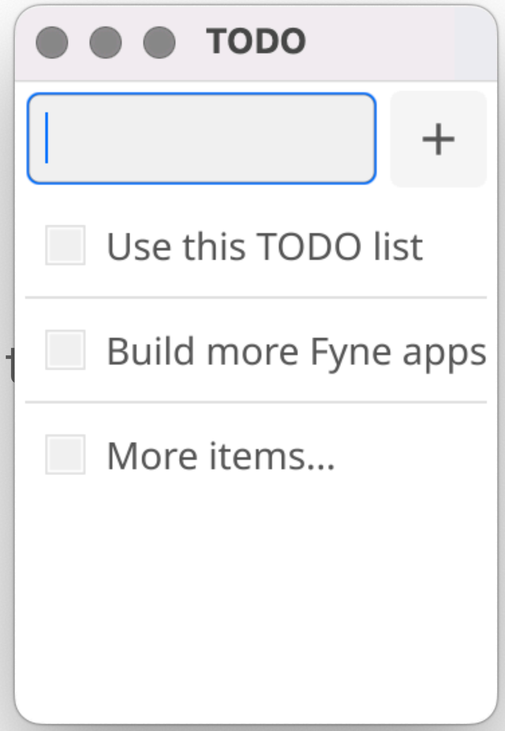
Completing the App

```
    data := a.Preferences().StringListWithFallback("todos",
        []string{"Use this TODO list", "Build more Fyne apps"})
    w.SetContent(makeUI(data))
```

"todo" from https://github.com/fyne-io/workshops/GoLab2024

Completing the App

```
mkdir mytodo; cd mytodo
go mod init mytodo
go get fyne.io/fyne/v2@lat
code main.go
code ui.go
go run .
```



Release to Stores!

- Prepare certification!
  - Android: Create an upload key in a Java keystroke (Android Studio)

    ```
    fyne release --os android --keyStore --keyStorePass --keyName --keyPass
    ```

  - iOS: Request a certificate and provisioning profile from Apple (Online)

    ```
    fyne release --os ios --profile --certificate
    ```

  - macOS: As iOS plus a packaging certificate

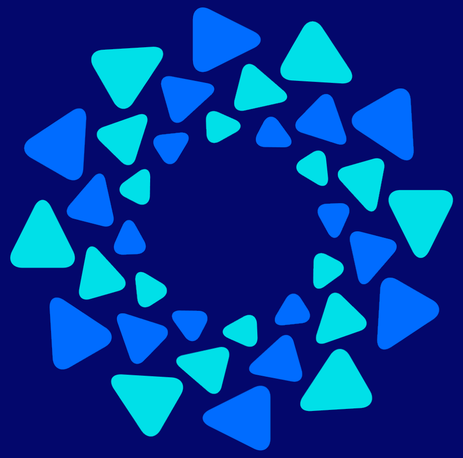
    ```
    fyne release --os ios --profile --certificate —category
    ```

  - Windows: Request a certificate from Microsoft

    ```
    fyne release --os ios --profile --certificate --category
    ```

Congratulations!

- Created a complete app
- Saved state between launches
- Ready to build anything - for all devices!
- AND release it to the stores :)



fyne

Extending Fyne Functionality

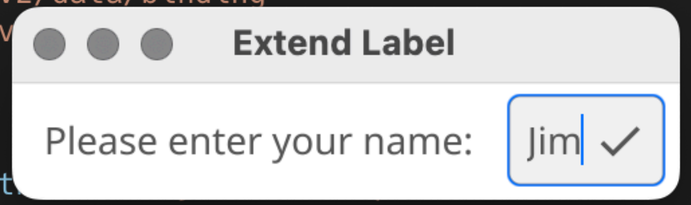
Adding new widgets and enhancing existing ones

Introduction

- Data Binding
- Extending widgets
- Custom widgets

Hello World Data Binding

```go
package main

import (
    "fyne.io/fyne/v2/app"
    "fyne.io/fyne/v2/container"
    "fyne.io/fyne/v2/data/binding"
    "fyne.io/fyne/v
)

func main() {
    a := app.NewWit
    w := a.NewWindow("Bind Name")

    name := binding.BindPreferenceString("user.name", a.Preferences())
    w.SetContent(container.NewHBox(
        widget.NewLabel("Please enter your name:"),
        widget.NewEntryWithData(name),
    ))

    w.ShowAndRun()
}
```



"more" from https://github.com/fyne-io/workshops/GoLab2024

Data Binding

- Two-way communication of data
- Can be used to update any object
- Many widgets provide data binding helpers
- Increase decoupling of data and GUI

Binding our Data - Data Source

```
@@ -3,16 +3,24 @@ package main
 import (
     "fyne.io/fyne/v2"
     "fyne.io/fyne/v2/app"
+    "fyne.io/fyne/v2/data/binding"
 )

 func main() {
     a := app.NewWithID("io.fyne.workshop.todo")
     w := a.NewWindow("TODO")

-    data := a.Preferences().StringListWithFallback("todos",
-        []string{"Use this TODO list", "Build more Fyne apps"})
+    data := binding.NewStringList()
+    data.Set(a.Preferences().StringListWithFallback("todos",
+        []string{"Use this TODO list", "Build more Fyne apps"}))
     w.SetContent(makeUI(data))

+    // work around missing BindPreferenceString
+    data.AddListener(binding.NewDataListener(func() {
+        vals, _ := data.Get()
+        a.Preferences().SetStringList("todos", vals)
+    }))
+
     w.Resize(fyne.NewSize(180, 240))
     w.ShowAndRun()
 }
```

Binding our Data - Bind List Widget

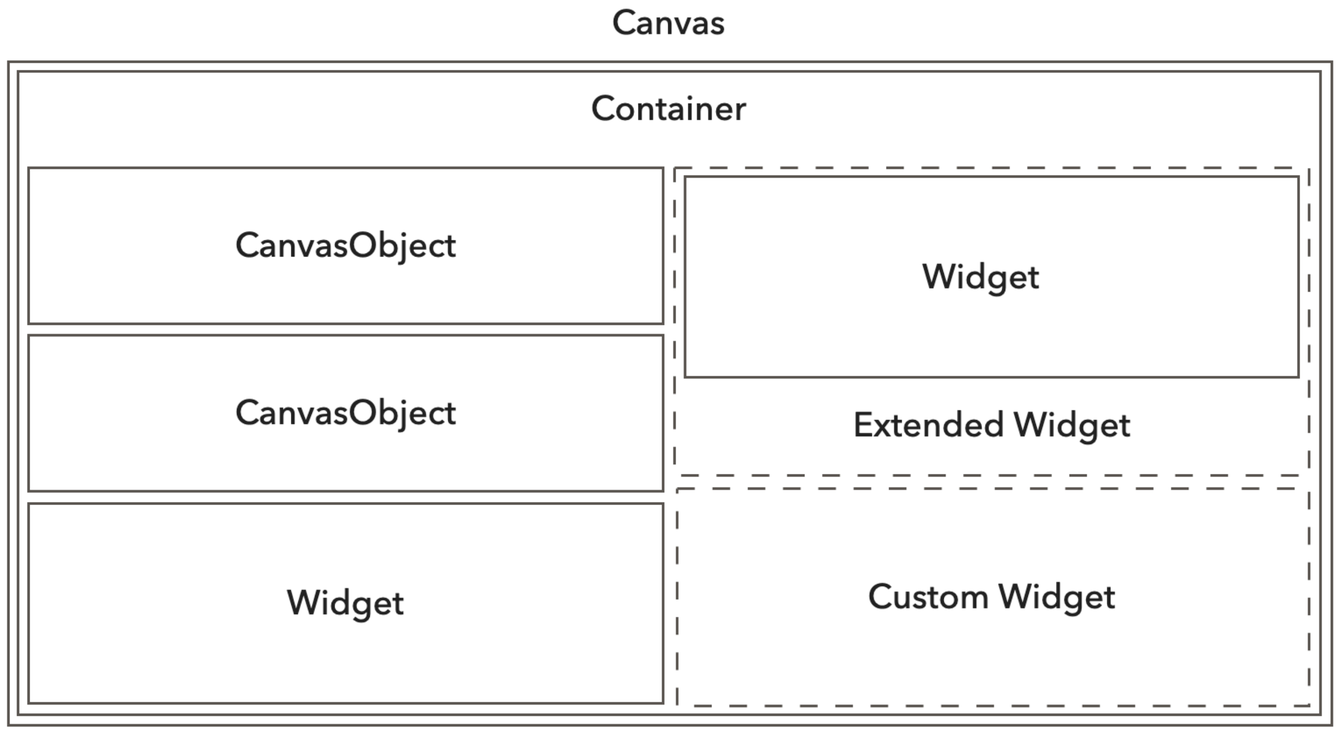
```
@@ -1,26 +1,22 @@
 package main

 import (
-    "slices"
-
     "fyne.io/fyne/v2"
     "fyne.io/fyne/v2/container"
+    "fyne.io/fyne/v2/data/binding"
     "fyne.io/fyne/v2/theme"
     "fyne.io/fyne/v2/widget"
 )

-func makeUI(data []string) fyne.CanvasObject {
-    var list *widget.List
-    list = widget.NewList(
-        func() int {
-            return len(data)
-        },
+func makeUI(data binding.StringList) fyne.CanvasObject {
+    list := widget.NewListWithData(data,
         func() fyne.CanvasObject {
             return widget.NewCheck("An item title", nil)
         },
-        func(i widget.ListItemID, co fyne.CanvasObject) {
+        func(di binding.DataItem, co fyne.CanvasObject) {
+            text, _ := di.(binding.String).Get()
             ch := co.(*widget.Check)
-            ch.SetText(data[i])
+            ch.SetText(text)

...
```

Binding our Data - Updating on Change

```
...

@@ -28,9 +24,7 @@ func makeUI(data []string) fyne.CanvasObject {
                }
                ch.SetChecked(false)

-               data = slices.Delete(data, i, i+1)
-               list.Refresh()
-               fyne.CurrentApp().Preferences().SetStringList("todos", data)
+               data.Remove(text)
            }
        })

@@ -41,10 +35,8 @@ func makeUI(data []string) fyne.CanvasObject {
                return
            }

-           data = append(data, input.Text)
-           list.Refresh()
+           data.Append(input.Text)
            input.SetText("")
-           fyne.CurrentApp().Preferences().SetStringList("todos", data)
        })
```

More Terminology

- Window
- Canvas
- CanvasObject
- Widget
- Container
- Extended Widget
- Custom Widget



Extending a Widget

- Create a new struct, embed the old type
- Add a constructor function
  - Call ExtendBaseWidget
- Implement interfaces to get events
  - See https://github.com/fyne-io/fyne/blob/master/canvasobject.go

Extend a Widget - Example

```go
type rightClickLabel struct {
    widget.Label

    canvas fyne.Canvas
}

func newRightClickLabel(t                    vas) fyne.Widget {
    r := &rightClickLabel
    r.Text = text
    r.ExtendBaseWidget(r)

    return r
}

func (r *rightClickLabel) TappedSecondary(e *fyne.PointEvent) {
    m := fyne.NewMenu("",
        fyne.NewMenuItem("Item 1", func() {}),
        fyne.NewMenuItem("Item 2", func() {}))
    widget.ShowPopUpMenuAtPosition(m, r.canvas, e.AbsolutePosition)
}
```

"more" from https://github.com/fyne-io/workshops/GoLab2024

Custom Widget - Architecture

- Widget
  - Defines the state or behaviour at API level
  - Expose API that is clear, document everything!
- WidgetRenderer
  - Handles how things will appear on screen
  - Synchronise widget state with output objects
  - Remember to apply theme to your objects

Custom Widget - Creating

- Implement the Widget interface
  - Can be done by extending widget.BaseWidget
  - Remember to ExtendBaseWidget if using the helper!
- Create a renderer and return from CreateRenderer
  - Often just returning widget.NewSimpleRenderer
- Add refresh code to sync data and apply current theme

Custom Widget - Example Widget

```go
type imageSwapper struct {
    widget.BaseWidget

    Reversed bool

    img1, img2 fyne.Resource
}

func newImageSwapper(img1, img2 fyne.Resource) fyne.Widget {
    i := &imageSwapper{img1: img1, img2: img2}
    i.ExtendBaseWidget(i)

    return i
}

func (i *imageSwapper) CreateRenderer() fyne.WidgetRenderer {
    return &imageSwapRenderer{img: i}
}

func (i *imageSwapper) Swap() {
    i.Reversed = !i.Reversed
    i.Refresh()
}

func (i *imageSwapper) Tapped(_ *fyne.PointEvent) {
    i.Swap()
}
```

"more" from https://github.com/fyne-io/workshops/GoLab2024

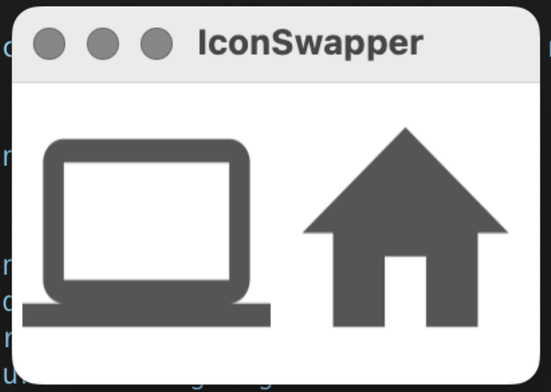
Custom Widget - Example WidgetRenderer

```go
type imageSwapRenderer struct {
    img *imageSwapper

    left, right *canvas.Image
    content     *fyne.Container
}

func (r *imageSwapRenderer) Destroy() {}

func (r *imageSwapRenderer) Layout(s fyne.Size) {
    if r.content == nil {
        return
    }

    r.content.Resize(s)
}

func (r *imageSwapRenderer) MinSize() fyne.Size {
    iconMin := theme.SizeForWidget(theme.SizeNameInlineIcon, r.img)
    pad := theme.SizeForWidget(theme.SizeNamePadding, r.img)

    return fyne.NewSize(iconMin*2+pad, iconMin)
}

...
```

“more” from https://github.com/fyne-io/workshops/GoLab2024

Custom Widget - Example WidgetRenderer (continued)

```
...

func (r *imageSwapRenderer) Objects() []fyne.CanvasObject {
    if r.content == nil {
        r.left = canvas.NewImageFromResource(r.img.img1)
        r.right = canvas.NewImageFromResource(r.img.img2)

        r.content = c                                    r.left, r.right)
    }

    return []fyne.Can
}

func (r *imageSwapRen
    if r.img.Reversed
        r.left.Resour
        r.right.Resou
    } else {
        r.left.Resource = r.img.img1
        r.right.Resource = r.img.img2
    }

    r.left.Refresh()
    r.right.Refresh()
}
```



"more" from https://github.com/fyne-io/workshops/GoLab2024

Congratulations!

- Wow, much stretch goal!
- Binding data
- Extending widgets
- Creating new a widget

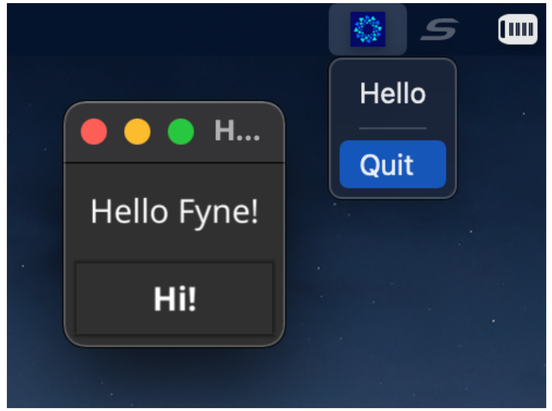
And so much more...

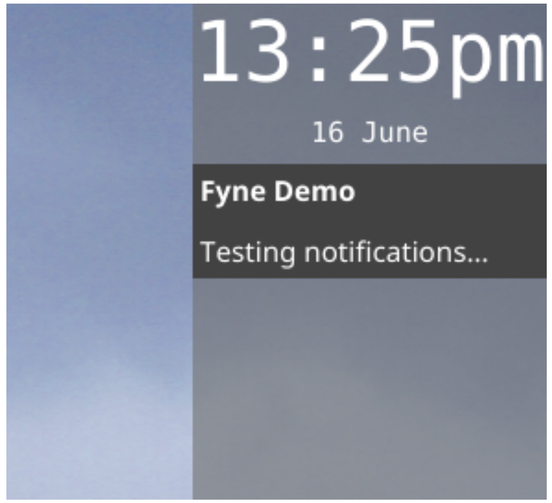
Features we missed

- Dialogs
- File/data handling
- Animations
- Shortcuts
- Multiple windows
- Custom themes

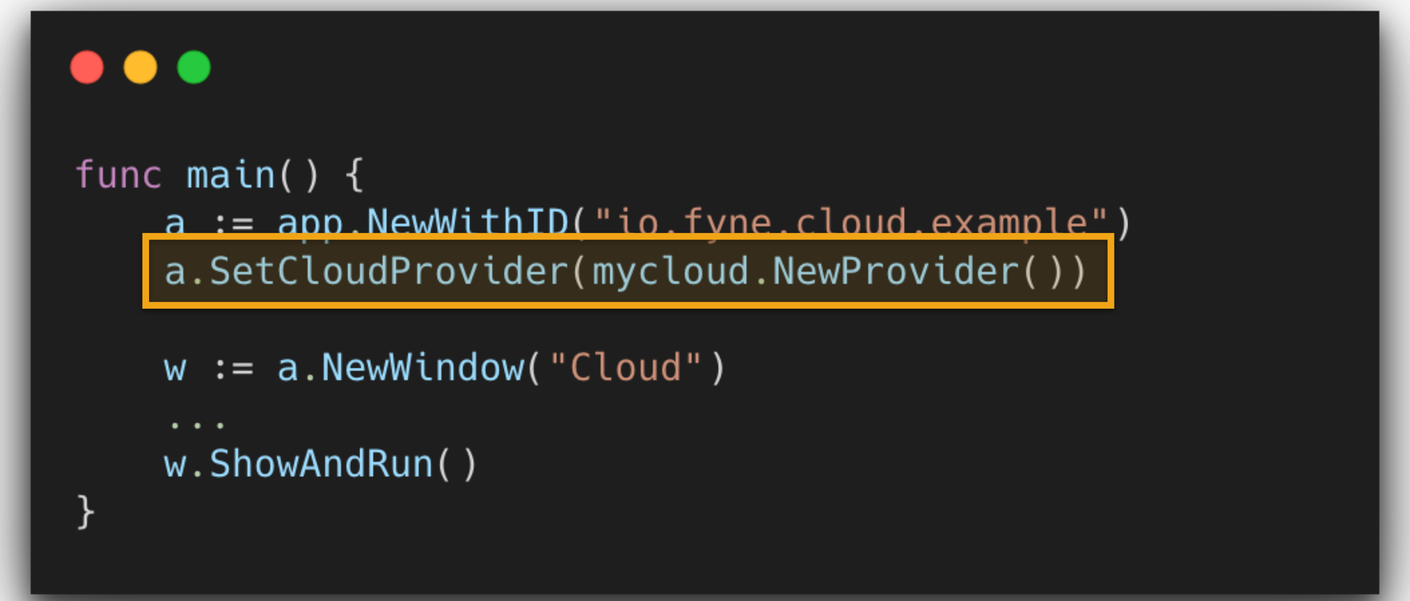
System integration

- System Tray
- Notifications
- Open browser windows
- Keyboard/Mouse/Touch events
- Control virtual keyboards



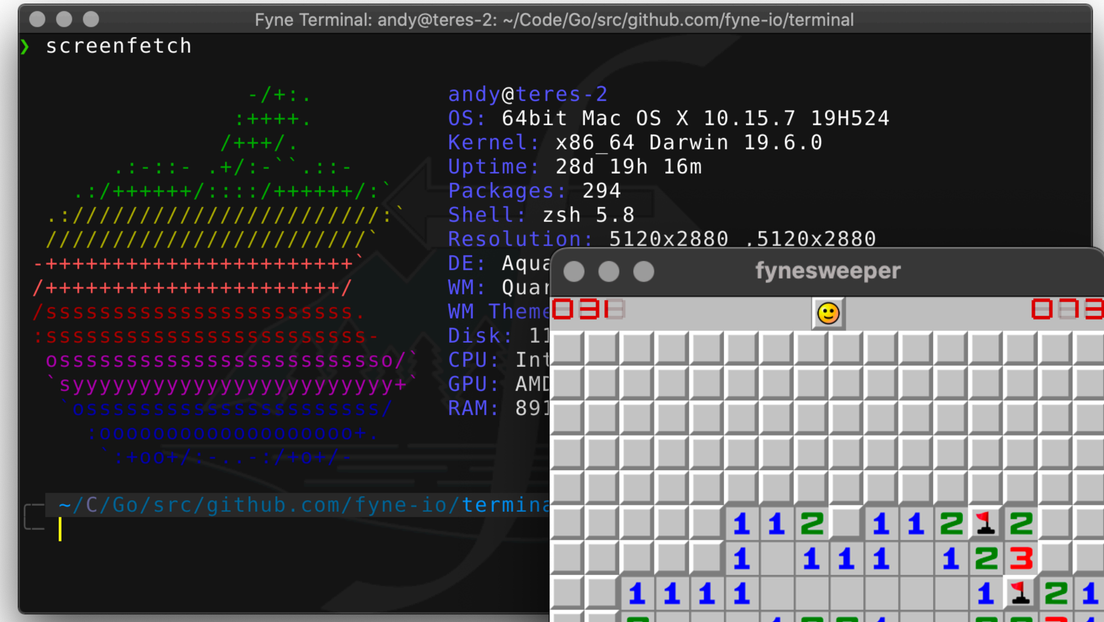


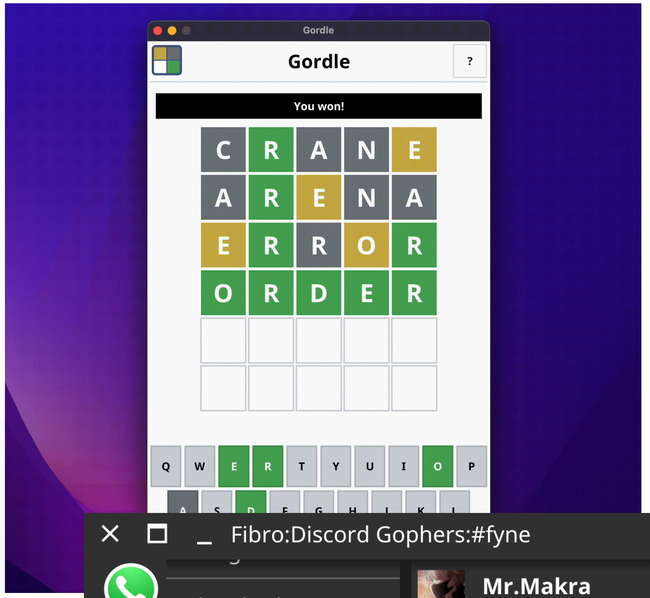
Integrate a Cloud Provider

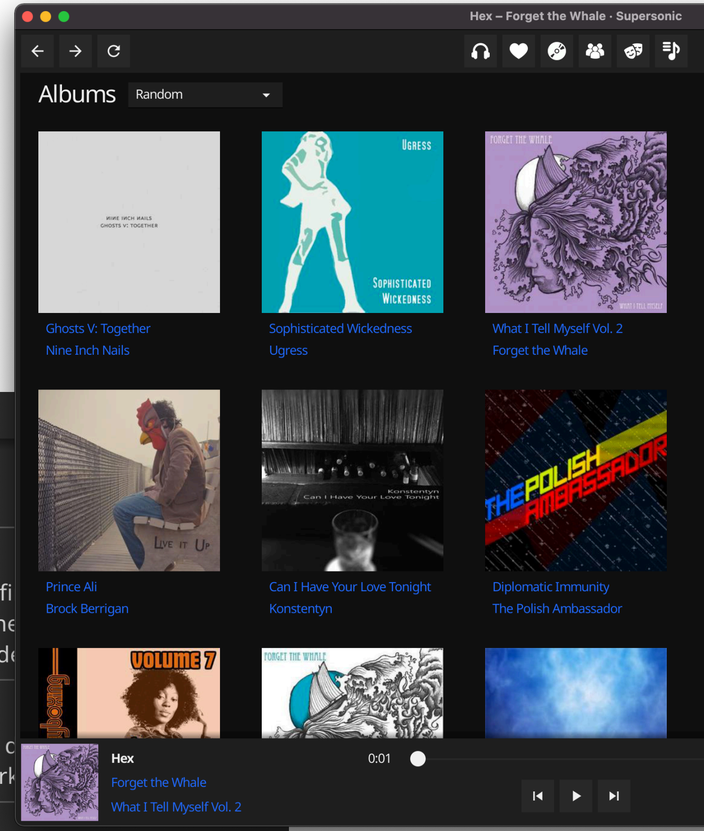


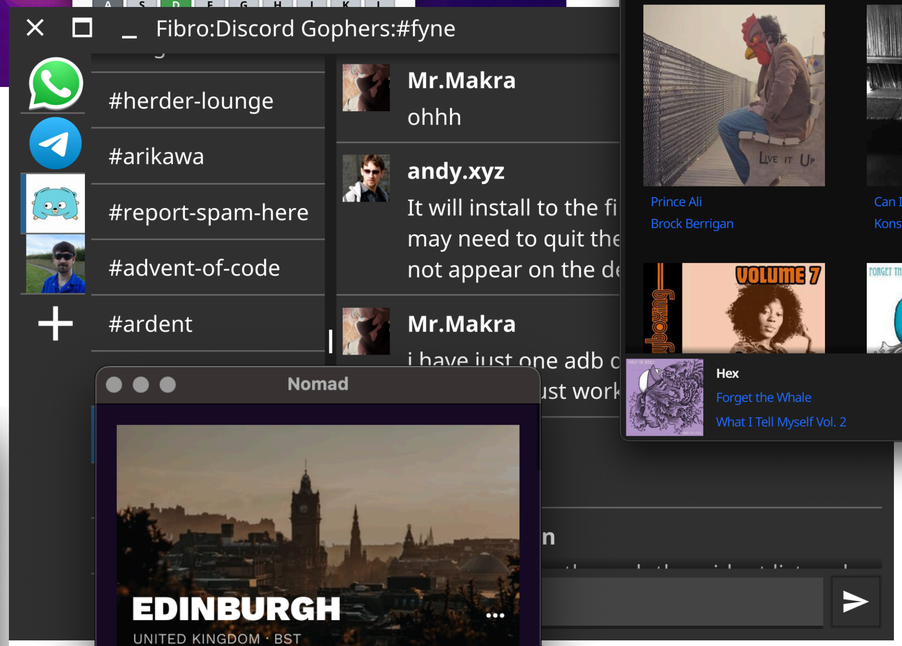
```
func main() {
    a := app.NewWithID("io.fyne.cloud.example")
    a.SetCloudProvider(mycloud.NewProvider())

    w := a.NewWindow("Cloud")
    ...
    w.ShowAndRun()
}
```

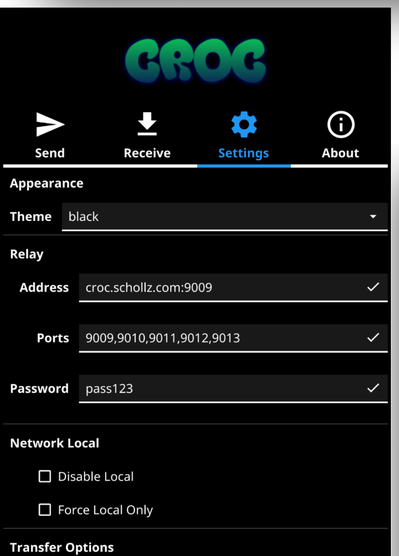
Other Fyne apps

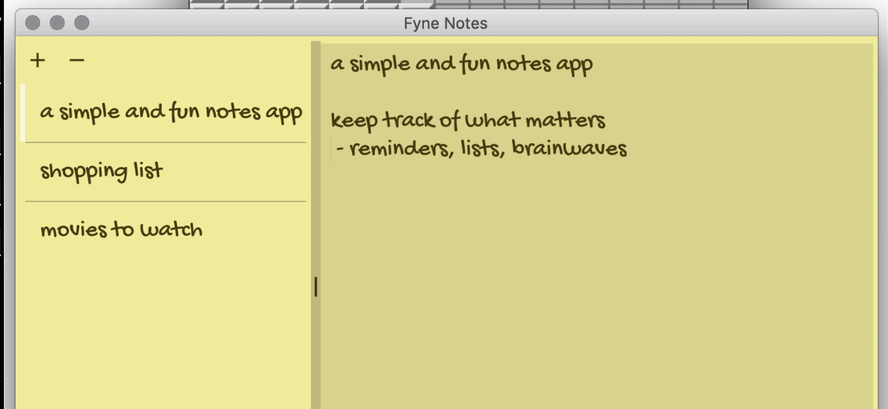






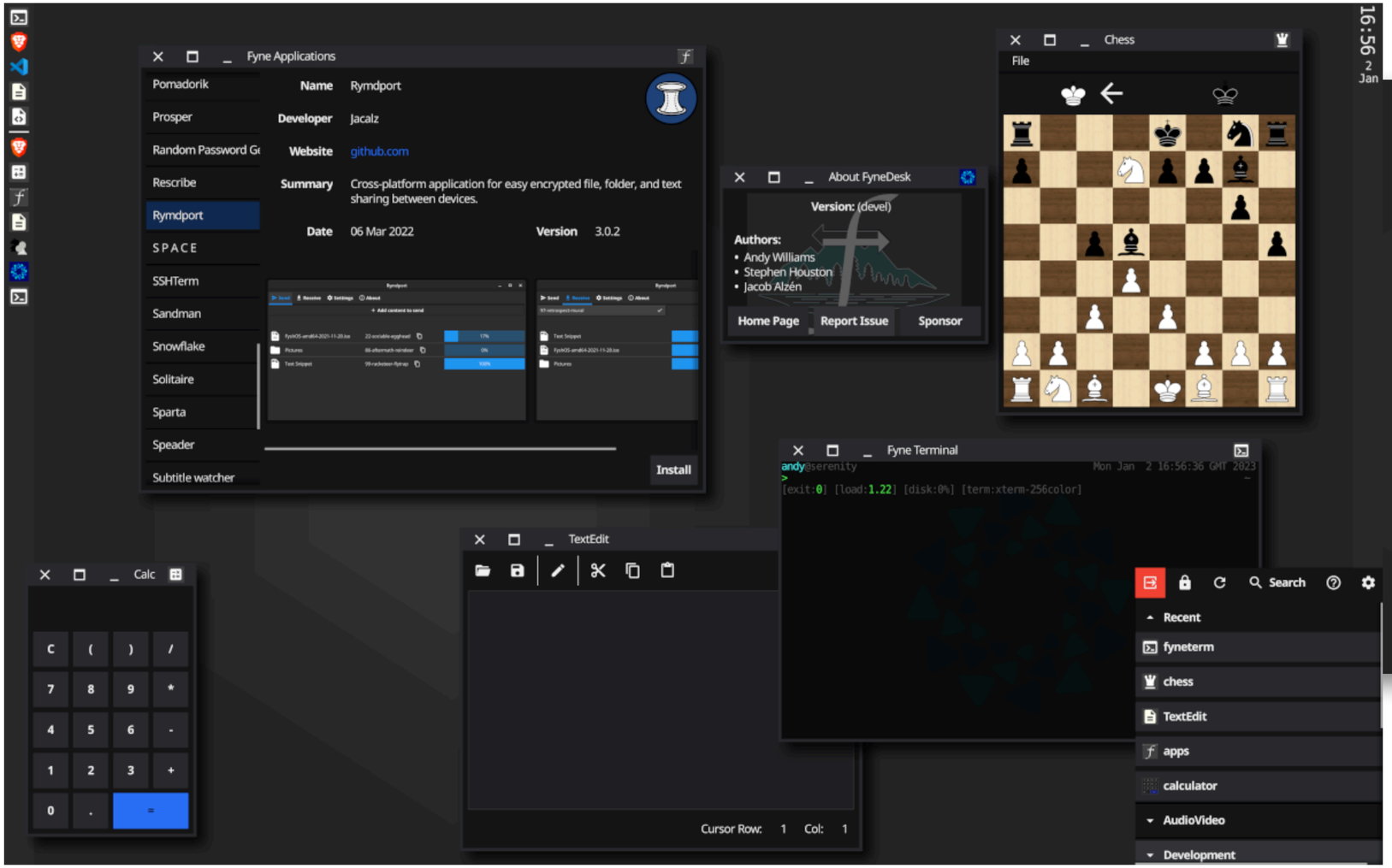


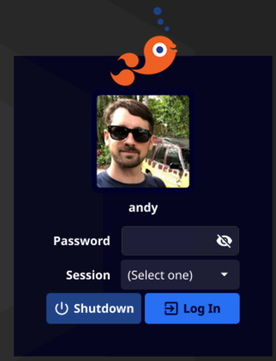




https://apps.fyne.io

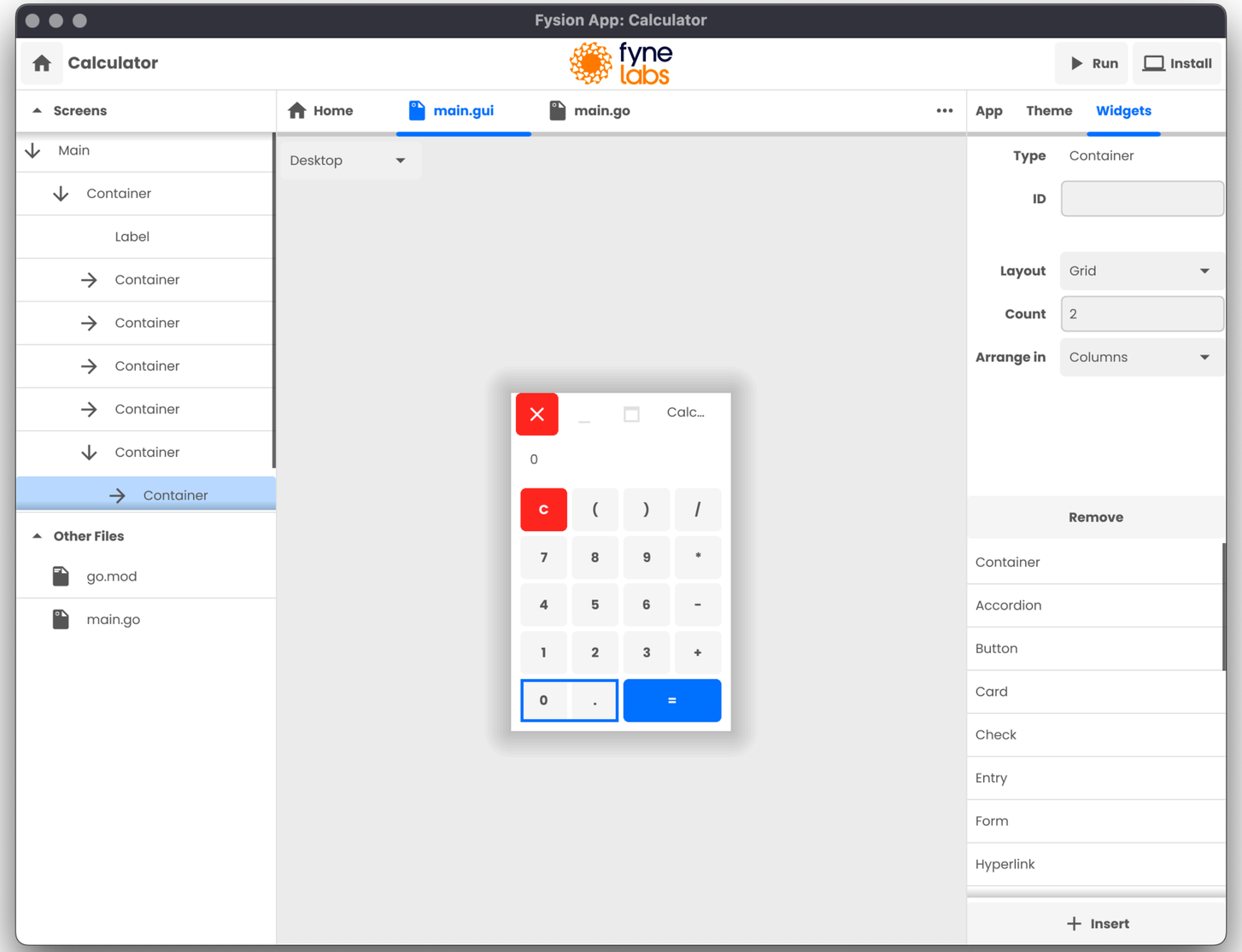
And FyshOS...





https://fyshos.com

And Fysion!



https://fysion.app

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