Deep Dish Swift 2023

DeckUI: Coding your next presentation... in Swift?

Deep Dish Swift 2023

~~Josh Holtz~~

*The part of "Josh Holtz" will be played by Zach Brass

Deep Dish Swift 2023

fastlane

RevenueCat

Deep Dish Swift 2023

Introducing...

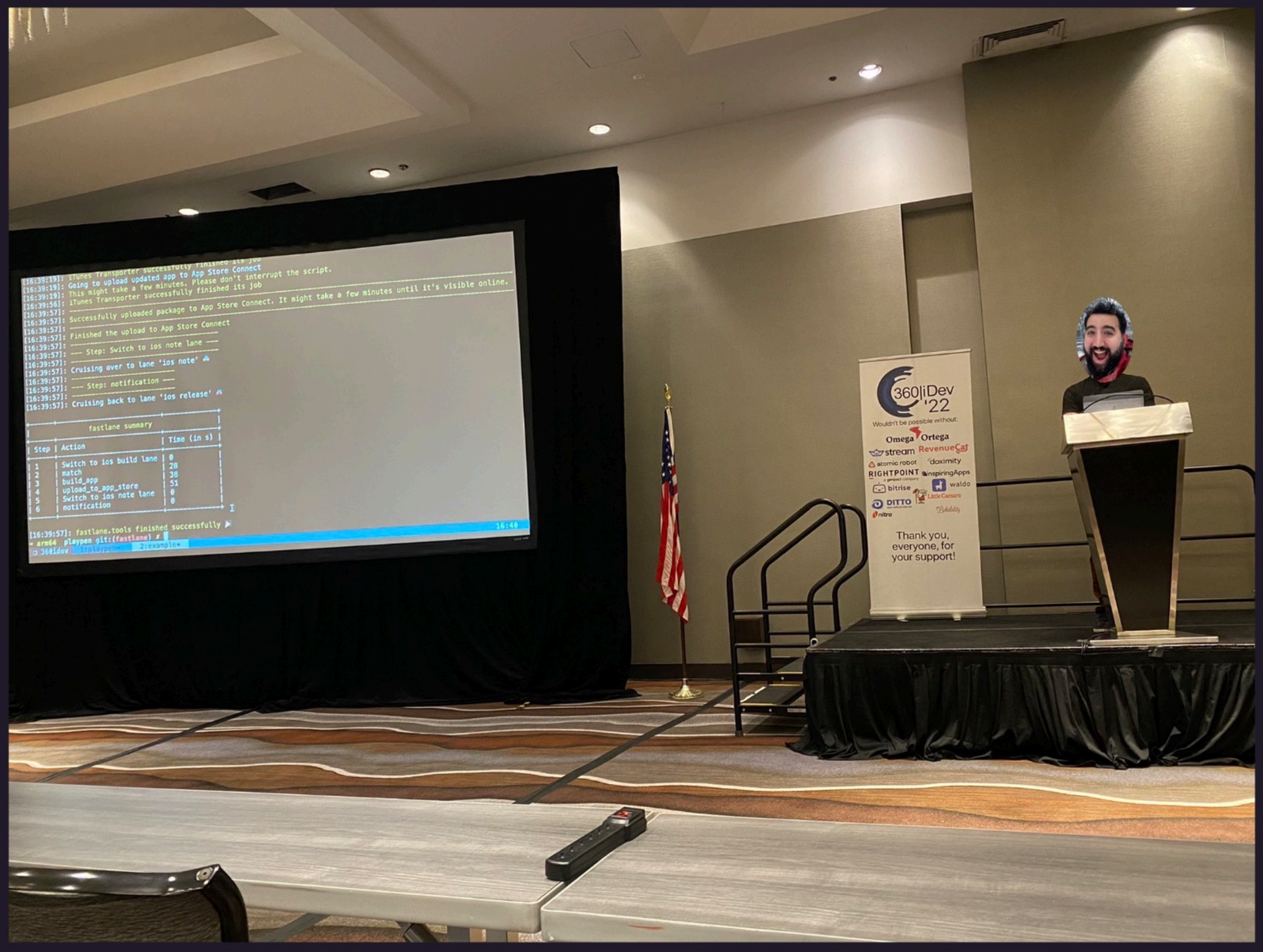

Deep Dish Swift 2023
DeckUI

What is DeckUI?

- A custom Swift DSL to make slide decks
- Distributed as a Swift Package
- Develop your slide deck in Xcode with Swift
- It's a side project that other people took too far
- It's open source!

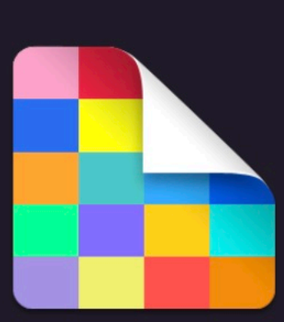
Why is DeckUI?

Deep Dish Swift 2023


[16:39:19]: iTunes Transporter successfully finished its job
[16:39:19]: Going to upload updated app to App Store Connect
[16:39:19]: This might take a few minutes. Please don't interrupt the script.
[16:39:56]: iTunes Transporter successfully finished its job
[16:39:57]: Successfully uploaded package to App Store Connect. It might take a few minutes until it's visible online.
[16:39:57]: Finished the upload to App Store Connect
[16:39:57]: --- Step: Switch to ios note lane ---
[16:39:57]: Cruising over to lane 'ios note' 🚁
[16:39:57]: --- Step: notification ---
[16:39:57]: Cruising back to lane 'ios release' 🚁
fastlane summary
Step | Action | Time (in s)
1 | Switch to ios build lane | 0
2 | match | 28
3 | build_app | 30
4 | upload_to_app_store | 51
5 | Switch to ios note lane | 0
6 | notification | 0
[16:39:57]: fastlane.tools finished successfully 🎉
360|iDev '22
Wouldn't be possible without:
Omega Ortega
stream RevenueCat
atomic robot
doximity
RIGHTPOINT
InspiringApps
bitrise
waldo
DITTO
LittleCaesars
nitro
Thank you, everyone, for your support!

Deep Dish Swift 2023

Deckset




Presentation.md
UNREGISTERED
Presentation.md
1 # Headline indicates a first level title.
2
3 ## Subheader a second level title.
4
5 ---
6
7 # Here is a new slide
Line 7, Column 22
Tab Size: 4
Markdown

Deep Dish Swift 2023

Deep Dish Swift 2023

Swift!

What's possible?

Deep Dish Swift 2023

Center alignment

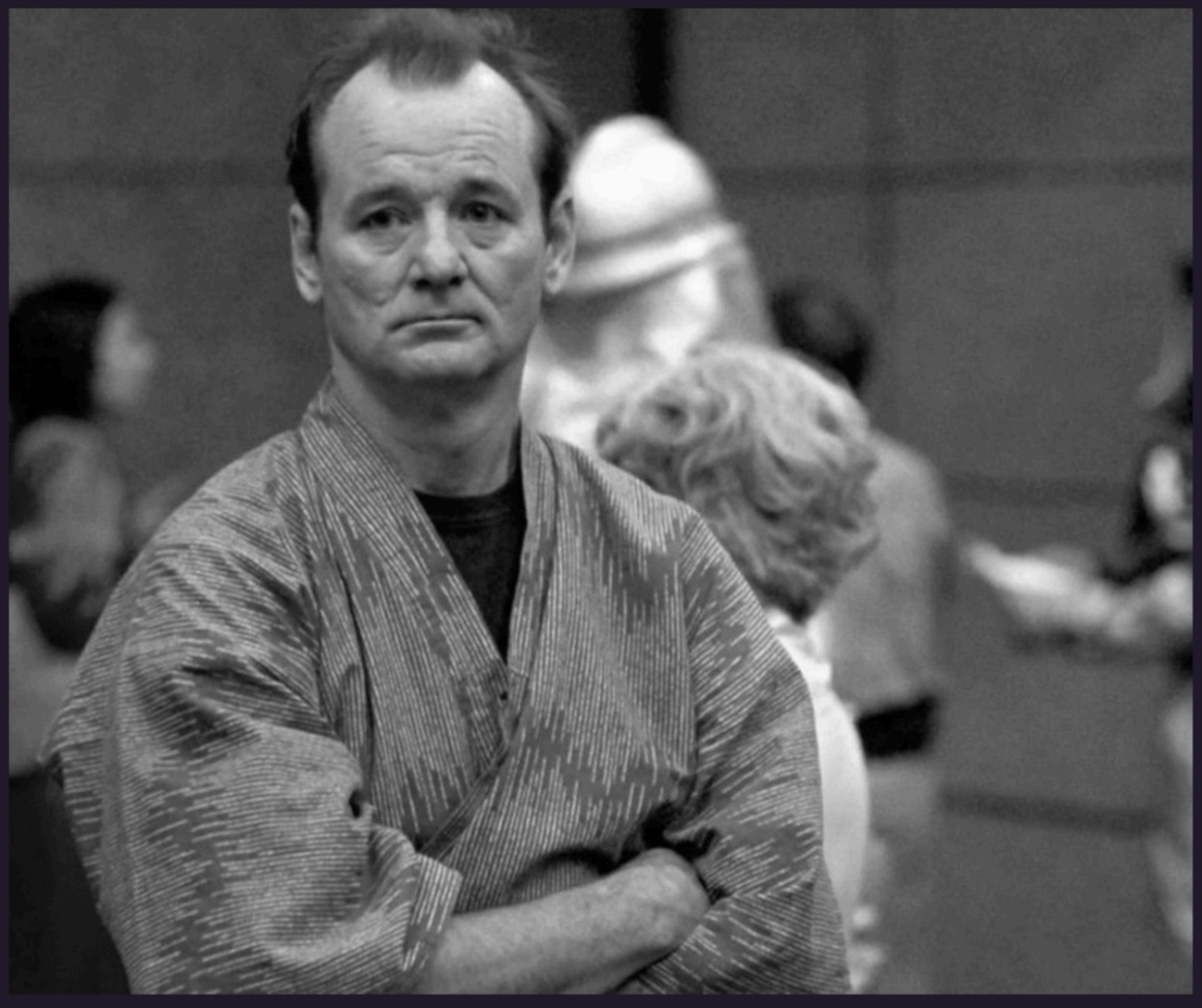
Slides can be center aligned

Top alignment

Slides also be top aligned

Deep Dish Swift 2023

Images



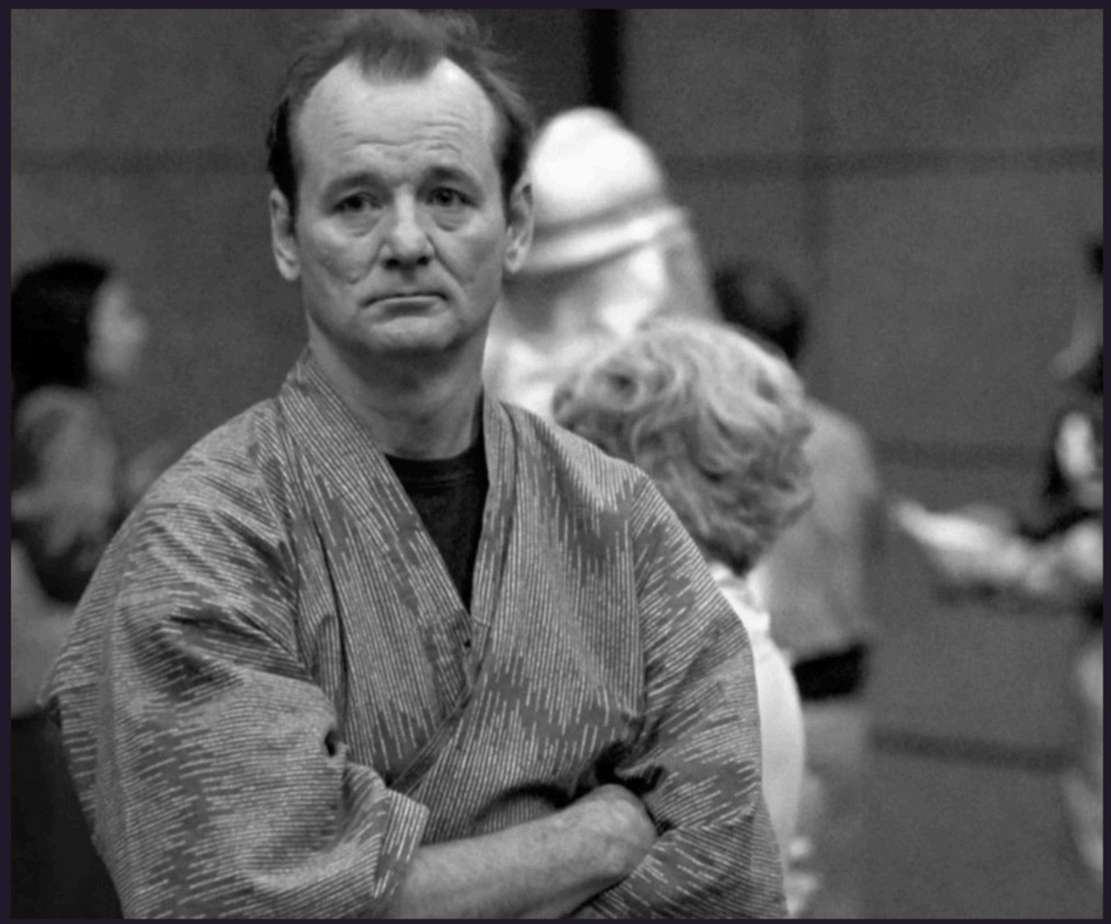
Deep Dish Swift 2023

Videos

Multiple Columns

1, 2, or more

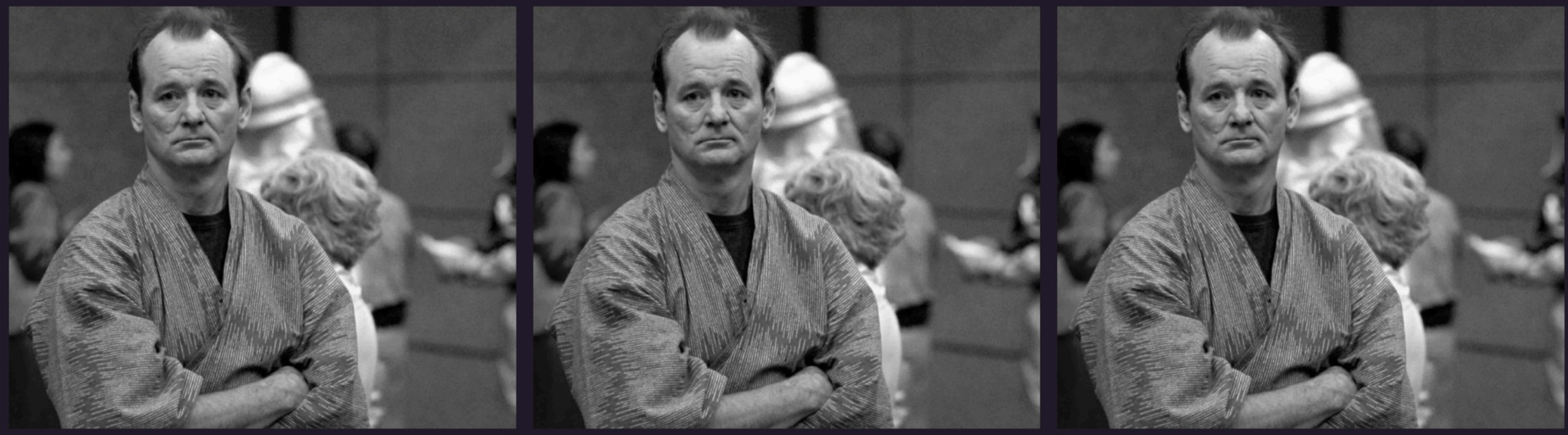
- Bill
- Murray
- Image
- 👉



Deep Dish Swift 2023

3 Bill Murraies

Don't know the plural of Murray



You can theme slides

Or entire presentations

Deep Dish Swift 2023

Code Blocks

```
struct CounterView: View {
    @State var count = 0

    var body: some View {
        Button {
            self.count += 1
        } label: {
            Text("Press me - \(self.count)")
                .font(.system(size: 60))
                .padding(.horizontal, 40)
                .padding(.vertical, 20)
                .foregroundColor(.white)
                .overlay(
                    RoundedRectangle(cornerRadius: 25)
                    .stroke(Color.white, lineWidth: 2)
                )
        }.buttonStyle(.plain)
    }
}
```

Sweet, how do I do this?

Here's a basic DeckUI Deck and how to present it

```
import SwiftUI
import DeckUI

struct ContentView: View {

    var deck: Deck {
        Deck(title: "SomeConf 2023") {
            Slide(alignment: .center) {
                Title("Welcome to DeckUI")
            }

            Slide {
                Title("Slide 2")
                Words("Some useful content")
                Media(.assetImage("murray"))
            }
        }
    }

    var body: some View {
        Presenter(deck: self.deck)
    }
}
```

- Create a `Deck` with multiple `Slides`
- Each Slide contains `ContentItems`
- Create `Presenter` and pass a Deck
- `Presenter` is a SwiftUI View to present a `Deck`

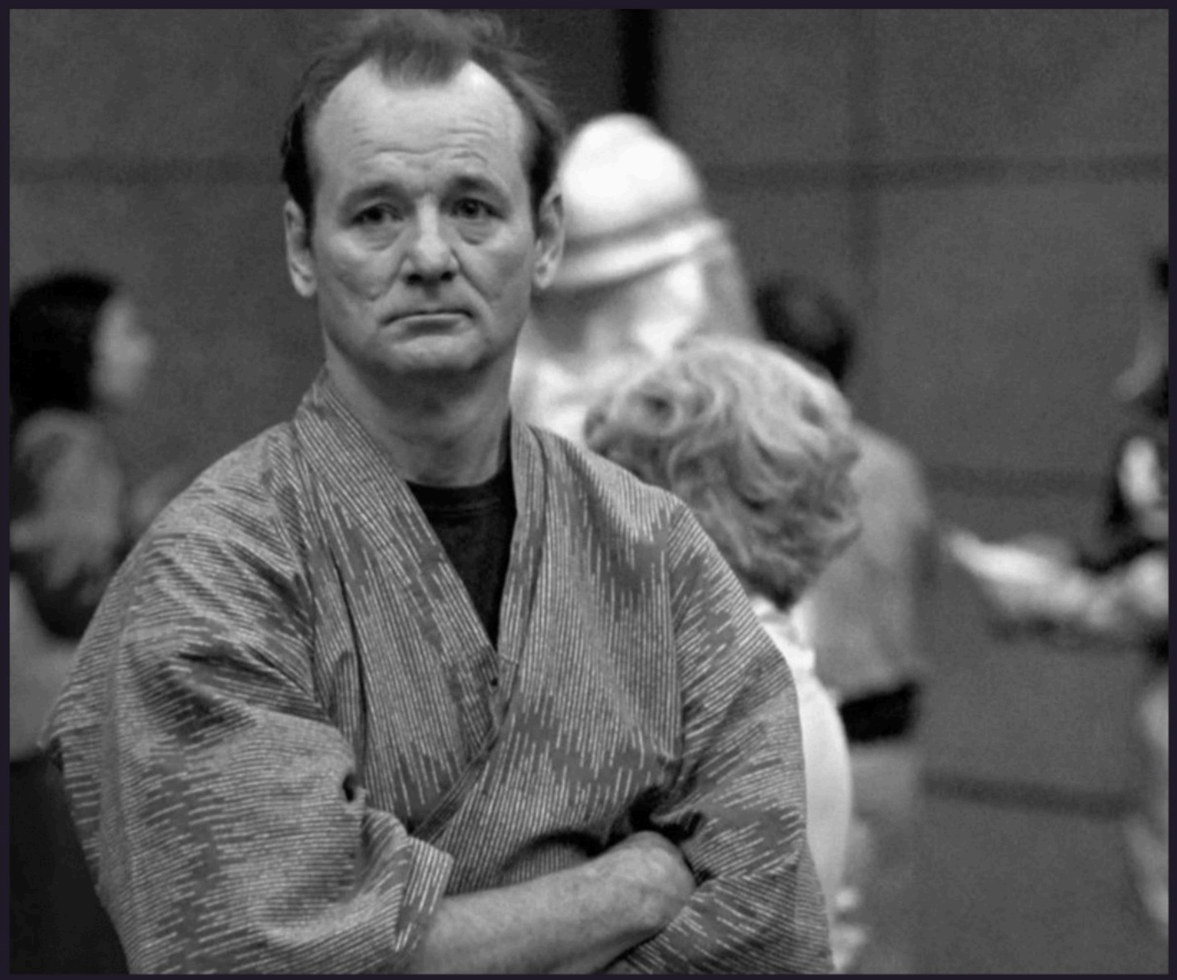
Deep Dish Swift 2023

Welcome to DeckUI

Deep Dish Swift 2023

Slide 2

Some useful content



Bullets

```
Slide {
    Bullets {
        Words("")
        Words("")
        Words("")
    }

    Bullets(style: .dash) {
        Words("")
        Words("")
        Words("")
    }
}
```

- Bullets take `Words`
- Default to circle/bullet
- Can change style

Media

```
Slide {
    Media(.assetImage(""))
    Media(.bundleImage(""))
    Media(.remoteImage(URL(string: ""))!)
    Media(.bundleVideo(""))

}
```

- Three image sources
- Video was just added

Columns

```
Slide {
    Columns {
        Column {
            Bullets {
                Words("Left")
                Words("is")
                Words("cool")
            }
        }
        Column {
            Bullets {
                Words("Right")
                Words("is")
                Words("cooler")
            }
        }
    }
}
```

- Split slide into 1 to many columns
- Each column takes ContentItems, just like a slide

Change theme

On Deck or Slide

```
extension Theme {
    public static let venonat: Theme = Theme(
        background: Color(hex: "#624a7b"),
        title: Foreground(
            color: Color(hex: "#ff5a5a"),
            font: Font.system(size: 80,
                              weight: .bold,
                              design: .default)
        ),
        subtitle: Foreground(
            color: Color(hex: "#a48bbd"),
            font: Font.system(size: 50,
                              weight: .light,
design: .default).italic()
        ),
        body: Foreground(
            color: Color(hex: "#FFFFFF"),
            font: Font.system(size: 50,
                              weight: .regular,
                              design: .default)
```

```
// Set theme on presenter
var body: some View {
    Presenter(deck: self.deck, theme: .venonat)
}

// Or on individual slide
Slide(theme: .venonat) {
    Title("Some slide")
}
```

Code

```
Slide {
    Code(.swift) {
    """
    Slide {
        Words("Your code goes here")
    }
    """
    }
}
```

- Easily drop in your code
- Up and down arrows highlight lines
- Syntax highlighting in Swift

Deep Dish Swift 2023

What's coming soon?

Presenter notes view

Deep Dish Swift 2023

iOS and iPadOS support

Deep Dish Swift 2023

Markdown ~~and~~ *attributed* **strings**

But wait... DeckUI kinda looks like SwiftUI!

Spoiler alert, not a coincidence

```
extension ContentView {
    var deck: Deck {
        Deck(title: "SomeConf 2023") {
            Slide(alignment: .center) {
                Title("Welcome to DeckUI")
            }

            Slide {
                Title("Slide 2")
                Words("Some useful content")
                Media(.assetImage("murray"))
            }
        }
    }
}
```

Deep Dish Swift 2023

Domain-Specific Language

What is a DSL?

Deep Dish Swift 2023

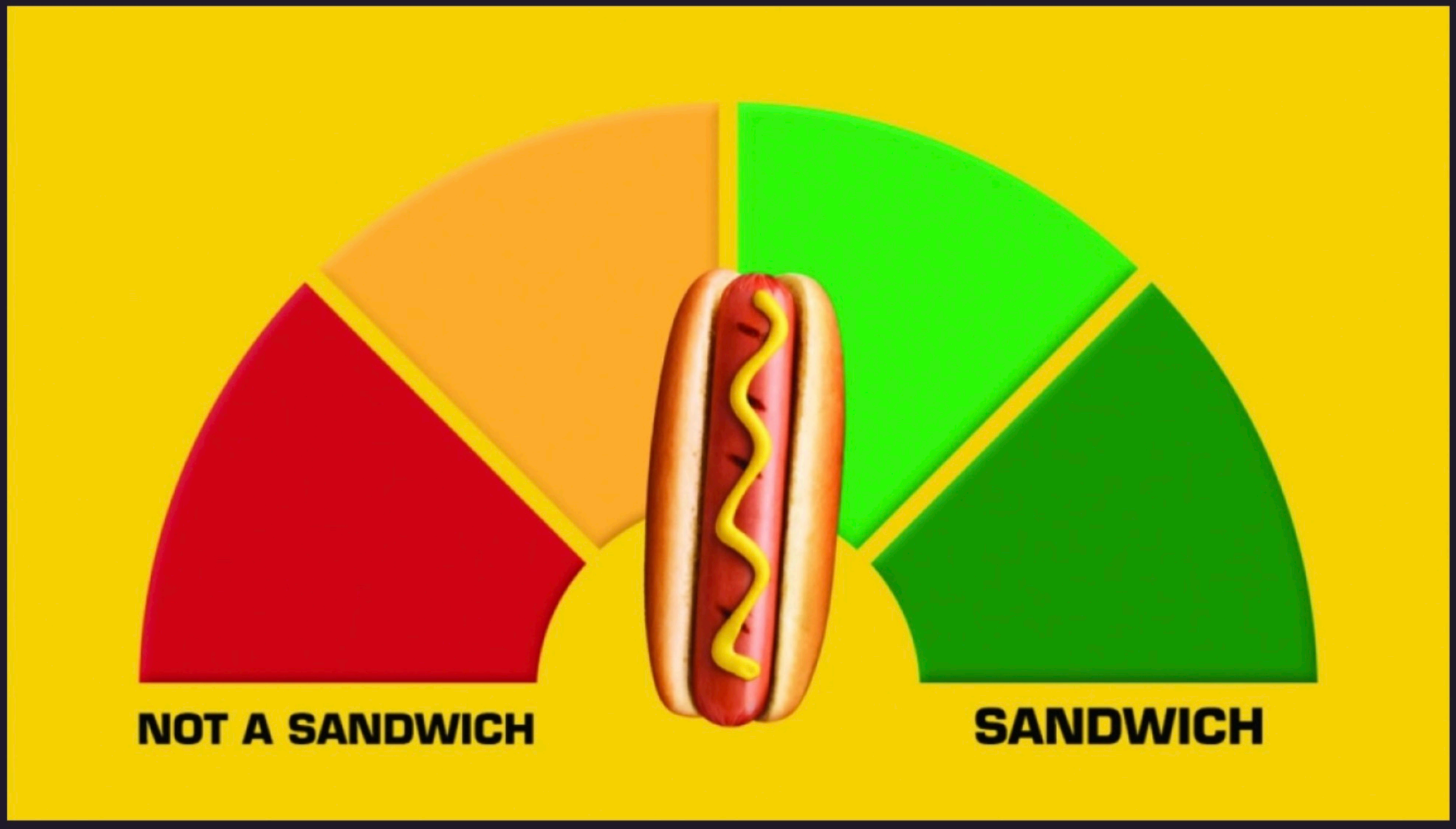
Have you used...

- SQL
- HTML
- CSS

Have you used...

- Cocoapods
- jQuery
- SwiftUI
- RegexBuilder

Deep Dish Swift 2023



How do they work in Swift?

And why does it look so much like SwiftUI?

Deep Dish Swift 2023

Result Builders

Making Swift code look funky since 2019

```
Deck(title:"Sample") {
    Slide {}
    Slide {}
    Slide {}
    Slide {}
}
```

What are these curly braces doing?

```
Deck.init(title:"Sample", slides: {
    Slide {}
    Slide {}
    Slide {}
    Slide {}
})

public struct Deck {
    public init(title: String, theme: Theme = .dark, @SlideArrayBuilder slides: @escaping () -> [Slide]) {
        ...
    }
}
```

```
Deck.init(title:"Sample", slides: {
    Slide {}
    Slide {}
    Slide {}
    Slide {}
    return ??
})

public struct Deck {
    public init(title: String, theme: Theme = .dark, @SlideArrayBuilder slides: @escaping () -> [Slide]) {
        ...
    }
}
```

```
Deck.init(title:"Sample", slides: {
    let s1 = Slide {}
    let s2 = Slide {}
    let s3 = Slide {}
    let s4 = Slide {}
    return SlideArrayBuilder.buildBlock(s1, s2, s3, s4)
})

public struct Deck {
    public init(title: String, theme: Theme = .dark, @SlideArrayBuilder slides: @escaping () -> [Slide]) {
        ...
    }
}
@resultBuilder
public enum SlideArrayBuilder {
    public static func buildBlock(_ components: Slide...) -> [Slide] {
        return [components]
    }
}
```

Deep Dish Swift 2023

HAHAHA

Deep Dish Swift 2023

NO.

```
Deck(title:"Sample") {
    Slide {}
    if (someCondition) {
        Slide {}
        Slide {}
    }
    Slide {}
}
```

```
Deck.init(title:"Sample", slides: {
    let s1 = Slide {}
    var s2: Slide?
    var s3: Slide?
    if (someCondition) {
        s2 = Slide {}
        s3 = Slide {}
    }
    let s4 = Slide {}
    return SlideArrayBuilder.buildBlock(s1, s2, s3, s4)
})

public struct Deck {
    public init(title: String, theme: Theme = .dark, @SlideArrayBuilder slides: @escaping () -> [Slide]) {
        ...
    }
}
@resultBuilder
public enum SlideArrayBuilder {
    public static func buildBlock(_ components: Slide...) -> [Slide] {
        return [components]
    }
}
```

Result builders remove some language features by default

```
Deck.init(title:"Sample", slides: {
    let s1 = Slide {}
    let s2: [Slide]
    if (someCondition) {
        let s2_1 = Slide {}
        let s2_2 = Slide {}
        let s2_block = SlideArrayBuilder.buildBlock(s2_1, s2_2)
        s2 = SlideArrayBuilder.buildOptional(s2_block)
    } else {
        s2 = SlideArrayBuilder.buildOptional(nil)
    }
    let s3 = Slide {}
    return SlideArrayBuilder.buildBlock(s1, s2, s3)
})

@resultBuilder
public enum SlideArrayBuilder {
    public static func buildBlock(_ components: Slide...) -> [Slide] {
        return [components]
    }
    public static func buildOptional(_ component: [Slide]?) -> [Slide] {
        return component ?? []
    }
}
```

```
Deck.init(title:"Sample", slides: {
    let s1 = Slide {}
    let s2: [Slide]
    if (someCondition) {
        let s2_1 = Slide {}
        let s2_2 = Slide {}
        let s2_block = SlideArrayBuilder.buildBlock(s2_1, s2_2)
        s2 = SlideArrayBuilder.buildOptional(s2_block)
    } else {
        s2 = SlideArrayBuilder.buildOptional(nil)
    }
    let s3 = Slide {}
    return SlideArrayBuilder.buildBlock(s1, s2, s3)
})

@resultBuilder
public enum SlideArrayBuilder {
    public static func buildBlock(_ components: [Slide]...) -> [Slide] {
        return components.flatMap { $0 }
    }
    public static func buildOptional(_ component: [Slide]?) -> [Slide] {
        return component ?? []
    }
}
```

```
Deck.init(title:"Sample", slides: {
    let s1 = SlideArrayBuilder.buildExpression(Slide {})
    let s2: [Slide]
    if (someCondition) {
        let s2_1 = SlideArrayBuilder.buildExpression(Slide {})
        let s2_2 = SlideArrayBuilder.buildExpression(Slide {})
        let s2_block = SlideArrayBuilder.buildBlock(s2_1, s2_2)
        s2 = SlideArrayBuilder.buildOptional(s2_block)
    } else {
        s2 = SlideArrayBuilder.buildOptional(nil)
    }
    let s3 = SlideArrayBuilder.buildExpression(Slide {})
    return SlideArrayBuilder.buildBlock(s1, s2, s3)
})

@resultBuilder
public enum SlideArrayBuilder {
    public static func buildBlock(_ components: [Slide]...) -> [Slide] {
        return components.flatMap { $0 }
    }
    public static func buildOptional(_ component: [Slide]?) -> [Slide] {
        return component ?? []
    }
    public static func buildExpression(_ expression: Slide) -> [Slide] {
        return [expression]
    }
}
```

Deep Dish Swift 2023

```
Deck(title:"Sample") {
    Slide {}
    if (someCondition) {
        Slide {}
        Slide {}
    }
    Slide {}
}

@resultBuilder
public enum SlideArrayBuilder {
    public static func buildBlock(_ components: [Slide]...) -> [Slide] {
        return components.flatMap { $0 }
    }
    public static func buildOptional(_ component: [Slide]?) -> [Slide] {
        return component ?? []
    }
    public static func buildExpression(_ expression: Slide) -> [Slide] {
        return [expression]
    }
}
```

Deep Dish Swift 2023

Method	Enables
buildArray	for..in
buildEither(first:), buildEither(second:)	if-else
buildExpression(Void)	Empty closures
buildExpression(Array)	Array variables
buildLimitedAvailability	#available

So what do I do once I make a result builder?

Whatever you want

Deep Dish Swift 2023

SwiftUI works the same way

ViewBuilder is a result builder!

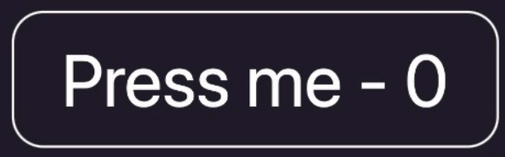
So why not just use SwiftUI?

Domain specificity

So why not just use SwiftUI? You can!

```
Slide {
    RawView {
        CounterView()
    }
}
struct CounterView: View {
    @State var count = 0

    var body: some View {
        Button {
            self.count += 1
        } label: {
            Text("Press me - \(self.count)")
                .font(.system(size: 60))
                .padding(.horizontal, 40)
                .padding(.vertical, 20)
                .foregroundColor(.white)
                .overlay(
                    RoundedRectangle(cornerRadius: 25)
                    .stroke(Color.white, lineWidth: 2)
                )
        }.buttonStyle(.plain)
    }
}
```

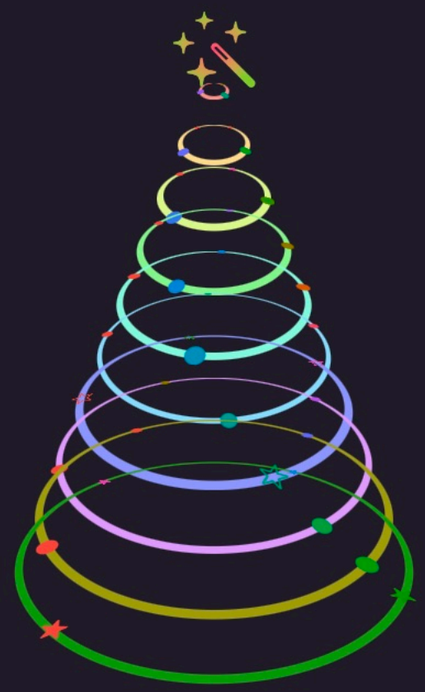


Deep Dish Swift 2023

Let's turn up the heat a bit

Deep Dish Swift 2023

Deep Dish Swift 2023



Deep Dish Swift 2023

Why you should use DeckUI

Why you should use DeckUI

- Your demos are right in the deck
- Way better than live coding
- If you like coding more than WYSIWYG editors
- You can use Git!
- Your presentation is an app!

Why you shouldn't use DeckUI

Or at least a few caveats if you want to use it

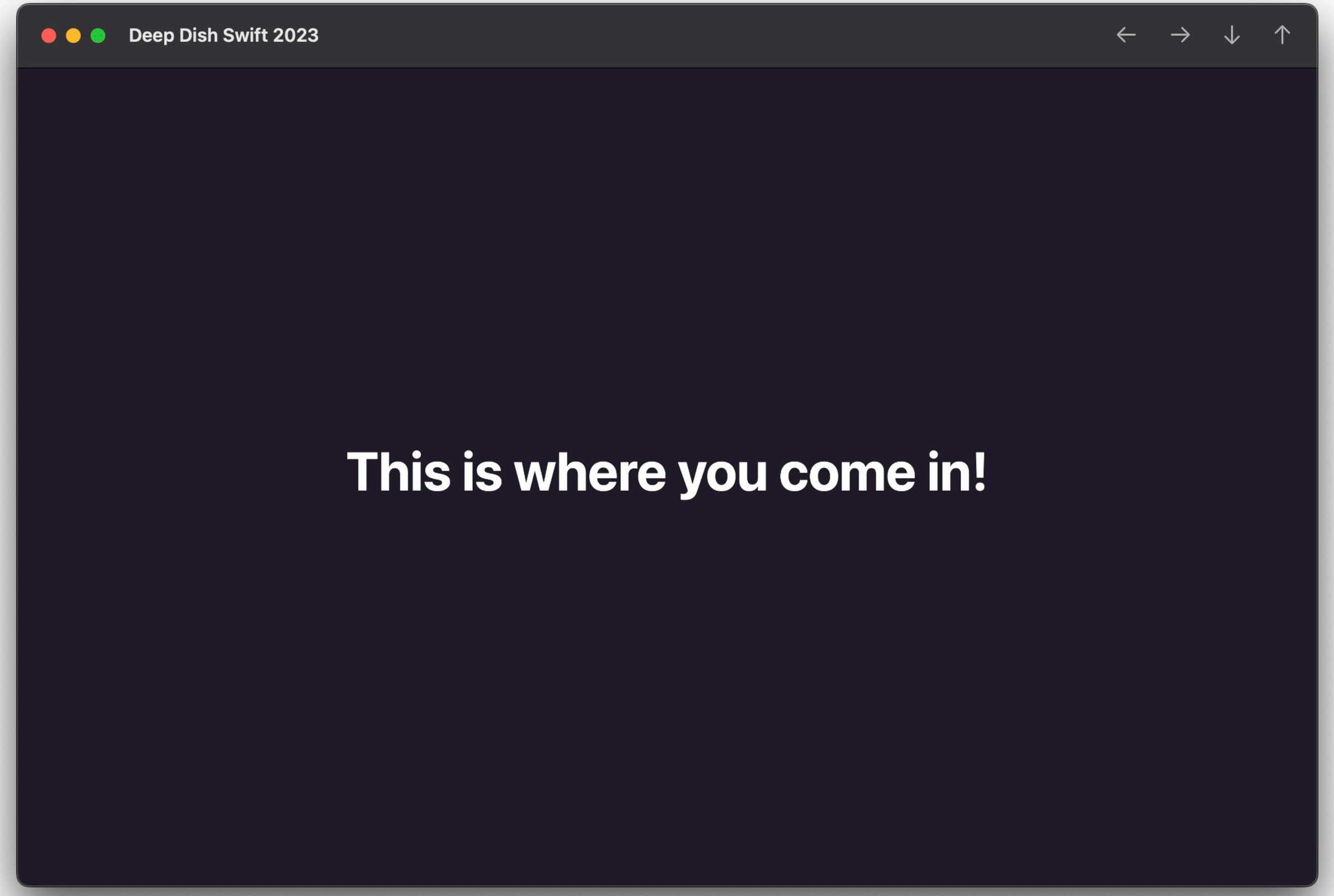
- Your presentation is an app
- Doesn't distribute very well
- Only on Apple platforms
- You'll have to drop down to SwiftUI a LOT
- Feature set is limited

Deep Dish Swift 2023

It VERY alpha stage

Deep Dish Swift 2023

#MakeBenProothiFamous


Deep Dish Swift 2023
This is where you come in!

Wishlist

- More customization of `Words`
- Nested `Bullets`
- Better slide transitions (Magic Move?)
- Animations within a slide
- DeckUI packages and plugins through Swift Package Manager
- Syntax highlighting beyond Swift
- Improvements to presenter notes view
- Generate things other than SwiftUI under the hood
- Whatever else you want to build, we want to hear it!

Thank you!

https://github.com/joshdholtz/deckui