MY

# NAVSTACK

BRINGS ALL THE

BOYS TO THE YARD


CRAIG FEDERIGHI

# 🙋🏾 About Me

**Lead Dev @ Bally's Interactive**

Currently building amazing experiences within the Sports Product Studio

**I'm a Content Creator**

I currently teach iOS development mainly Swift & SwiftUI on my **YouTube** channel **tundsdev**

My NavStack Brings All The Boys To The Yard

# 🙋🏾 About Me

My NavStack Brings All The Boys To The Yard

# 🙋🏾 About Me

## Lead Dev @ Bally's Interactive

Currently building amazing experiences within the Sports Product Studio

## I'm a Content Creator

I currently teach iOS development mainly Swift & SwiftUI on my **YouTube** channel **tundsdev**

## I love Anime

When I'm not coding, to relax I love watching Anime & hope to visit Japan next year

## I live for random

I love travelling or exploring new things, just like you can see me landing in this ball pit like a potato

**My NavStack Brings All The Boys To The Yard**

# 🙋🏾‍♂️ About Me


My NavStack Brings All The Boys To The Yard

# ⏪ A Lil Rewind

My NavStack Brings All The Boys To The Yard

# ⏪ A Lil Rewind


It is production ready

My NavStack Brings All The Boys To The Yard

# ⏪ A Lil Rewind


My NavStack Brings All The Boys To The Yard

# 😜 The Good Ol Days?


My NavStack Brings All The Boys To The Yard

Structure

# NavigationView Deprecated

A view for presenting a stack of views that represents a visible path in a navigation hierarchy.

iOS 13.0–16.4 Deprecated | iPadOS 13.0–16.4 Deprecated | macOS 10.15–13.3 Deprecated | Mac Catalyst 13.0–16.4 Deprecated | tvOS 13.0–16.4 Deprecated | watchOS 7.0–9.4 Deprecated

**Deprecated**

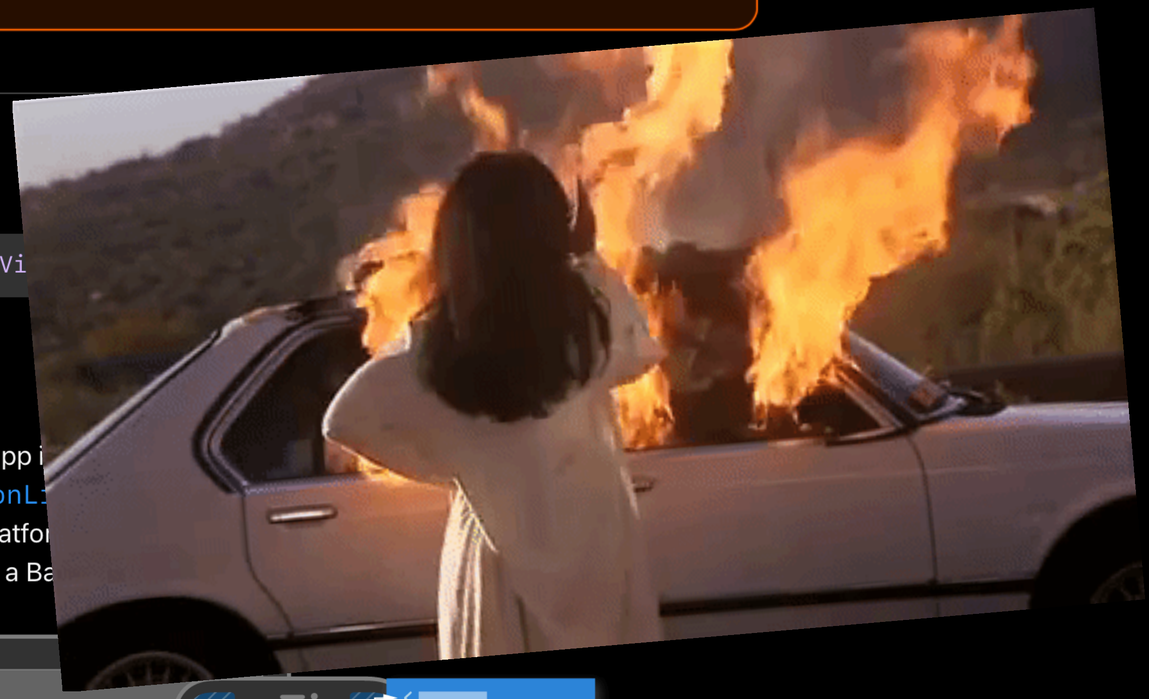
Use `NavigationStack` and `NavigationSplitView` instead. For more information, see Migrating to new navigation types.

## Declaration

```
struct NavigationView<Content> where Content : Vi
```

## Overview

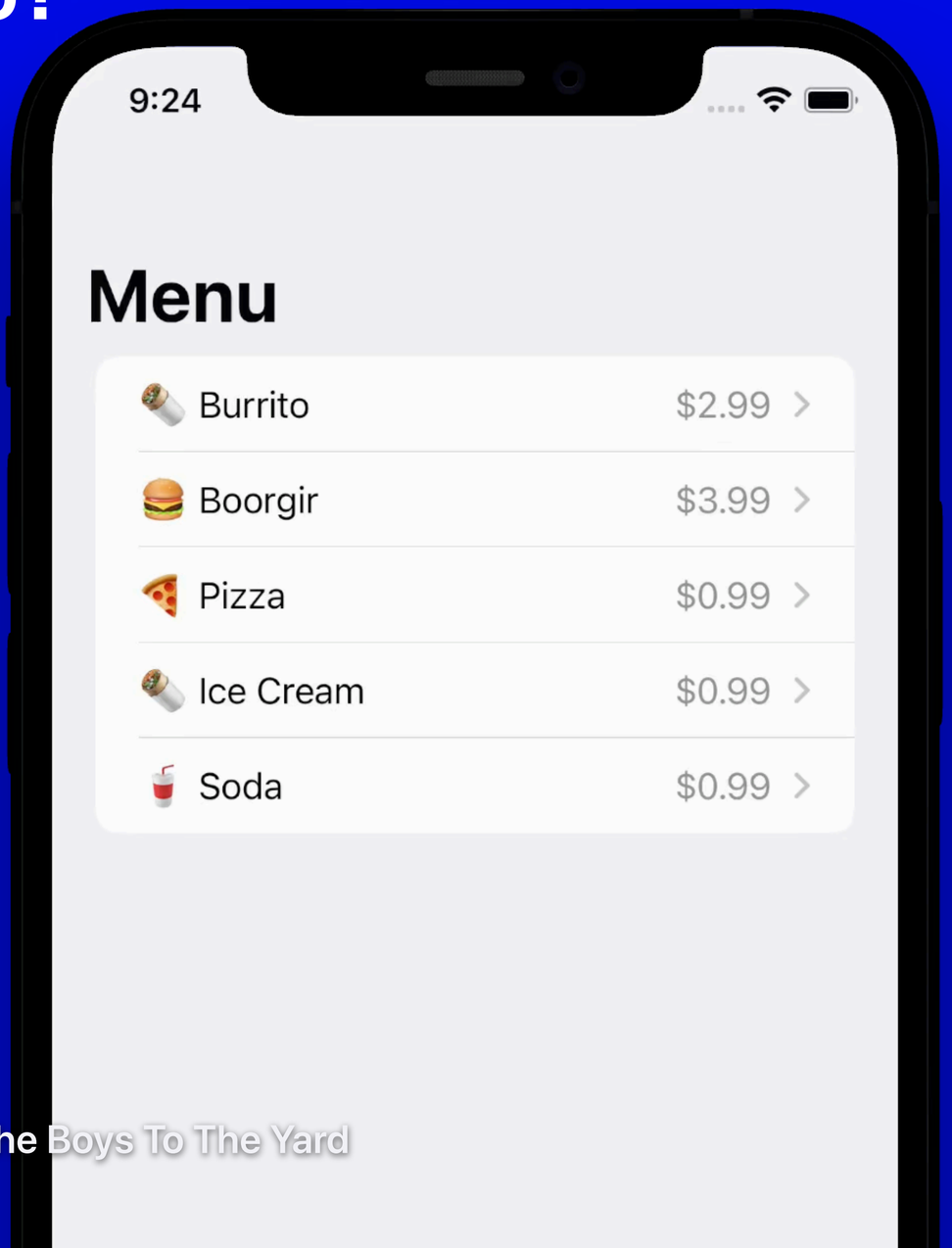
Use a `NavigationView` to create a navigation-based app i
navigate to a destination view by selecting a `NavigationLi`
destination content appears in the next column. Other platfo
items from the stack with platform-specific controls, like a Ba



# 🫂 Don’t Worry I Got You


My NavStack Brings All The Boys To The Yard

# 📣 NavigationWho?



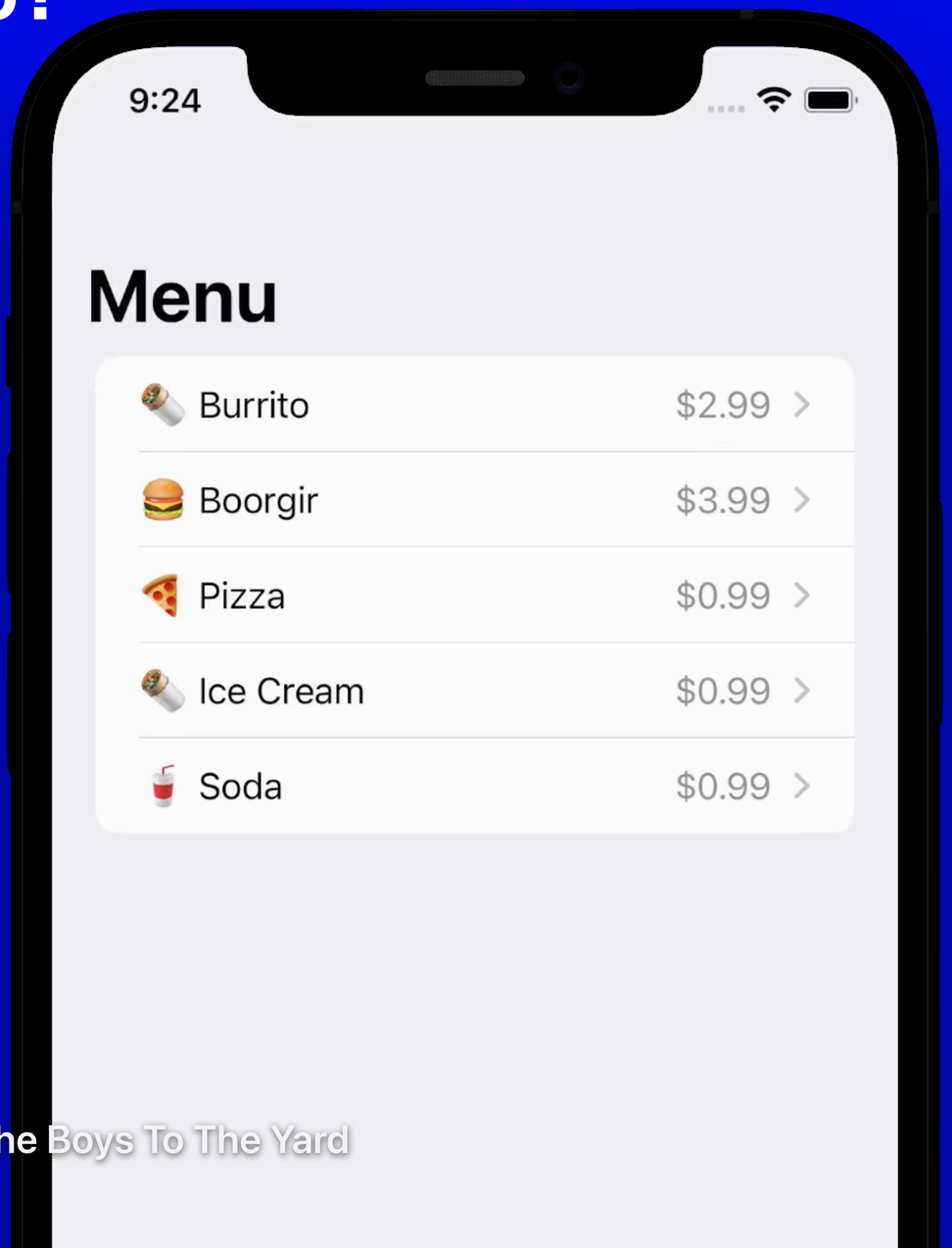

My NavStack Brings All The Boys To The Yard

# 📣 NavigationWho?

```
struct ContentView: View {
  var body: some View {
    NavigationView {
      List {
         ForEach(Item.data, id: \.name) { item in
          NavigationLink(destination: ItemDetailView(item: item)) {
              content(item)
          }
        }
      }
      .navigationTitle("Menu")
    }
  }
}
```

My NavStack Brings All The Boys To The Yard

# 📣 NavigationWho?




My NavStack Brings All The Boys To The Yard

# 📣 NavigationWho?


My NavStack Brings All The Boys To The Yard

# 📣 NavigationWho?

```
struct ContentView: View {
  var body: some View {
    NavigationView {
      List {
         ForEach(Item.data, id: \.name) { item in
          NavigationLink(destination: ItemDetailView(item: item)) {
              content(item)
          }
        }
      }
      .navigationTitle("Menu")
    }
  }
}
```

My NavStack Brings All The Boys To The Yard

# 🙃 Le Problem

```
ForEach(Item.data, id: \.name) { item in
 NavigationLink(destination: ItemDetailView(item: item)) {
     content(item)
 }

 NavigationLink(destination: PromoView(item: ???)) {
     content(item)
 }

 NavigationLink(destination: StatsView()) {
     content(item)
 }
}
```

My NavStack Brings All The Boys To The Yard

# 🙃 Le Problem

```
ForEach(Item.data, id: \.name) { item in
 NavigationLink(destination: ItemDetailView(item: item)) {
     content(item)
 }

 NavigationLink(destination: PromoView(item: ???)) {
     content(item)
 }

 NavigationLink(destination: StatsView()) {
     content(item)
 }
}
```

My NavStack Brings All The Boys To The Yard

# 🙃 Le Problem

## 💏 Boo'd up

Views become tightly coupled with one another

## 🤬 Bruh

Passing data can be difficult

## 🙅🏾‍♂️ No flex zone

The stack for the navigation can't be manipulated

## 🫠 Weirdo

You can sometimes get weird behaviour, due to side effects

My NavStack Brings All The Boys To The Yard

# 🤝 Bridging With UIKit


My NavStack Brings All The Boys To The Yard

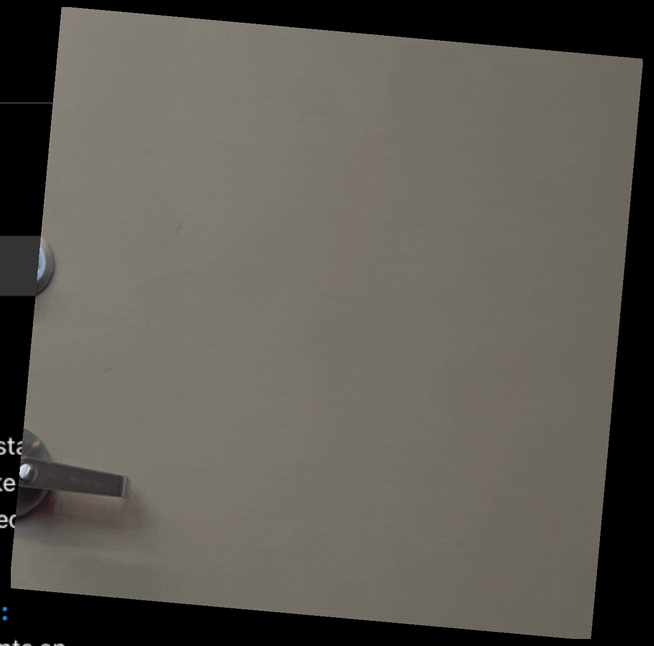
# 🤝 Bridging With UIKit

My NavStack Brings All The Boys To The Yard

# NavigationStack

A view that displays a root view and enables you to present additional views over the root view.

iOS 16.0+ | iPadOS 16.0+ | macOS 13.0+ | Mac Catalyst 16.0+ | tvOS 16.0+ | watchOS 9.0+



## Declaration

```
@MainActor struct NavigationStack<Data, Root> where Root : View
```

## Overview

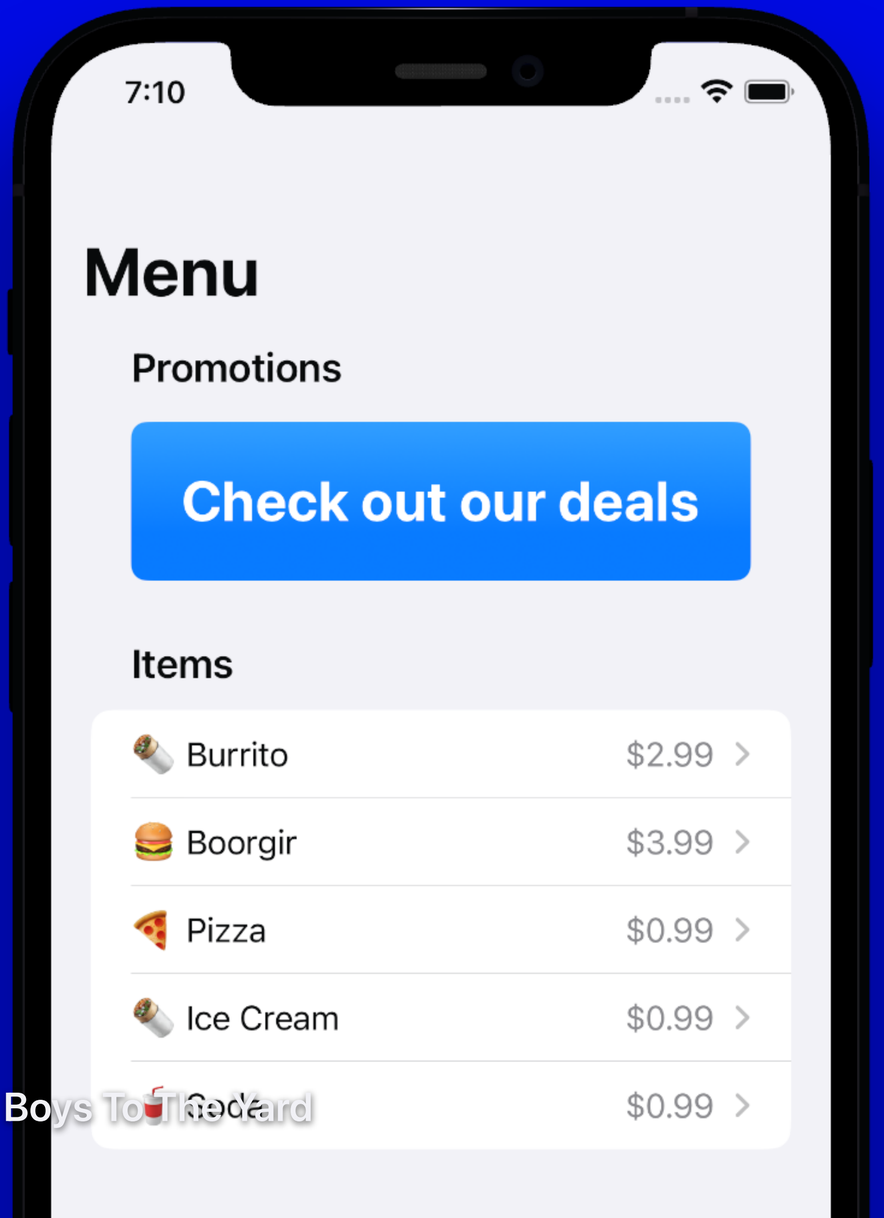
Use a navigation stack to present a stack of views over a root view. People can add views to the top of the sta
clicking or tapping a `NavigationLink`, and remove views using built-in, platform-appropriate controls, like
button or a swipe gesture. The stack always displays the most recently added view that hasn't been removed
doesn't allow the root view to be removed.

To create navigation links, associate a view with a data type by adding a `navigationDestination(for:destination:)` modifier inside the stack's view hierarchy. Then initialize a `NavigationLink` that presents an instance of the same kind of data. The following stack displays a `ParkDetails` view for navigation links that present data of type `Park`:

```
NavigationStack {
    List(parks) { park in
        NavigationLink(park.name, value: park)
    }
    .navigationDestination(for: Park.self) { park in
        ParkDetails(park: park)
    }
}
```

In this example, the `List` acts as the root view and is always present. Selecting a navigation link from the list adds a `ParkDetails` view to the stack, so that it covers the list. Navigating back removes the detail view and reveals the list

# 👋🏾 NavigationStack




My NavStack Brings All The Boys To The Yard

# 👋🏾 NavigationStack

```
struct ContentView: View {
  var body: some View {
    NavigationView {
      List {
         ForEach(Item.data, id: \.name) { item in
          NavigationLink(destination: ItemDetailView(item: item)) {
              content(item)
          }
        }
      }
      .navigationTitle("Menu")
    }
  }
}
```

My NavStack Brings All The Boys To The Yard

# Topics


## Presenting a destination view

`init(LocalizedStringKey, destination: () -> Destination)`

Creates a navigation link that presents a destination view, with a text label that th
string key.

Available when `Label` is `Text` and `Destination` conforms to `View`.

`init<S>(S, destination: () -> Destination)`

Creates a navigation link that presents a destination view, with a text label that t

Available when `Label` is `Text` and `Destination` conforms to `View`.

`init(destination: () -> Destination, label: () -> Label)`

Creates a navigation link that presents the destination view.

## Presenting a data value

`init<P>(LocalizedStringKey, value: P?)`

Creates a navigation link that presents the view corresponding to a value, with a text label that the link generates from a localized string key.

Available when `Label` conforms to `View` and `Destination` is `Never`.

`init<S, P>(S, value: P?)`

Creates a navigation link that presents the view corresponding to a value, with a text label that the link generates from a title string.

Available when `Label` conforms to `View` and `Destination` is `Never`.

`init<P>(value: P?, label: () -> Label)`

Creates a navigation link that presents the view corresponding to a value.

Available when `Label` conforms to `View` and `Destination` is `Never`.

## Presenting a codable value

# 👋🏾 NavigationStack

```
struct ContentView: View {
  var body: some View {
    NavigationView {
      List {
         ForEach(Item.data, id: \.name) { item in
          NavigationLink(destination: ItemDetailView(item: item)) {
              content(item)
          }
        }
      }
      .navigationTitle("Menu")
    }
  }
}
```

My NavStack Brings All The Boys To The Yard

# 👋🏾 NavigationStack

```
struct ContentView: View {
  var body: some View {
    NavigationView {
      List {
         ForEach(Item.data, id: \.name) { item in
             content(item)
       }
      }
      .navigationTitle("Menu")
    }
  }
}
```

My NavStack Brings All The Boys To The Yard

# 👋🏾 NavigationStack

```
struct ContentView: View {
  var body: some View {
    NavigationView {
      List {
         ForEach(Item.data, id: \.name) { item in
             content(item)
       }
      }
      .navigationTitle("Menu")
    }
  }
}
```

My NavStack Brings All The Boys To The Yard

# 👋🏾 NavigationStack

```
struct ContentView: View {
  var body: some View {
    NavigationStack {
      List {
         ForEach(Item.data, id: \.name) { item in

            content(item)

      }
      }
      .navigationTitle("Menu")
    }
  }
}
```

My NavStack Brings All The Boys To The Yard

# 👋🏾 NavigationStack

```
struct ContentView: View {
  var body: some View {
    NavigationStack {
      List {
          ForEach(Item.data, id: \.name) { item in
           NavigationLink(value: item) {
               content(item)
           }
        }
      }
      .navigationTitle("Menu")
    }
  }
}
```

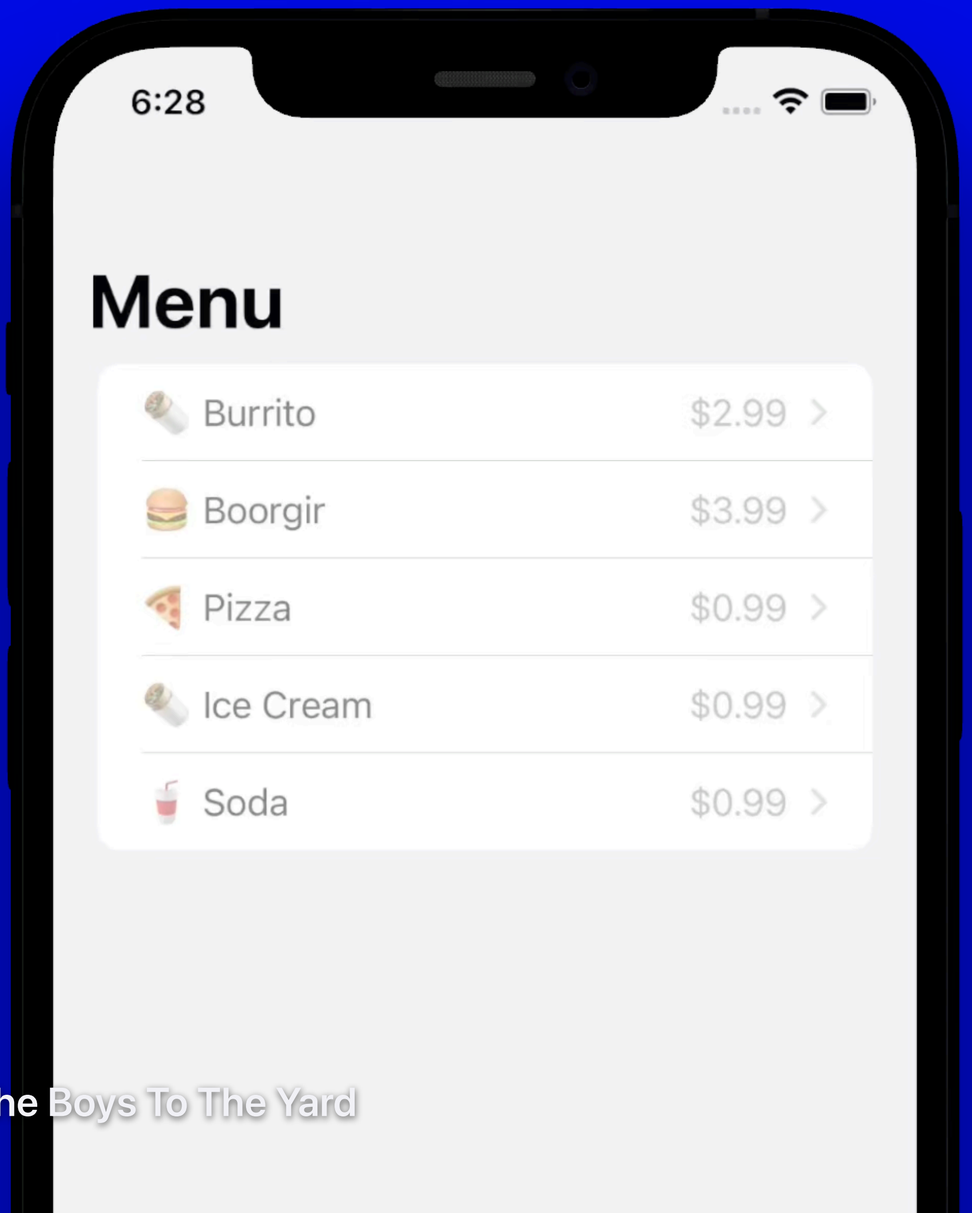
My NavStack Brings All The Boys To The Yard

# 🥔 Don’t Forget The Hash

```
struct Item: Hashable {
    let name: String
    let emoji: String
    let price: Decimal
}
```

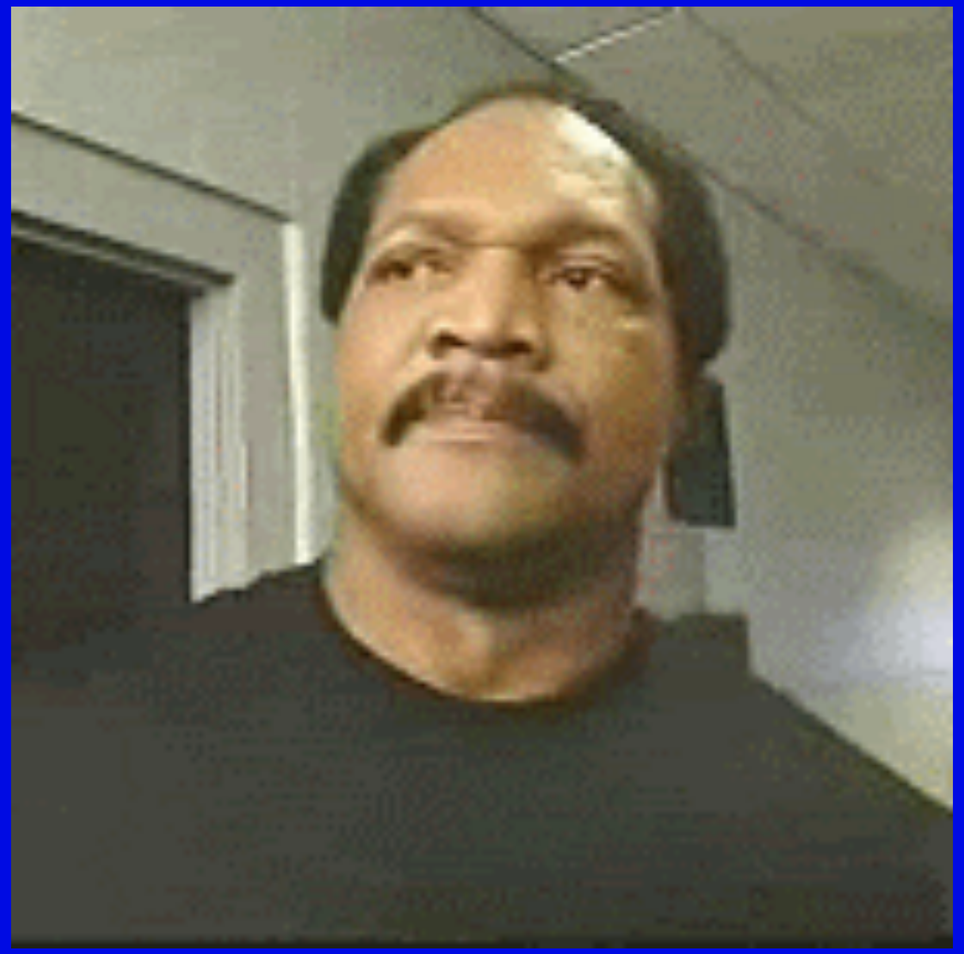

My NavStack Brings All The Boys To The Yard

# 👋 NavigationStack




My NavStack Brings All The Boys To The Yard

# 👋🏾 NavigationStack



My NavStack Brings All The Boys To The Yard

Instance Method

# navigationDestination(for:destination:)

Associates a destination view with a presented data type for use within a navigation stack.

iOS 16.0+ | iPadOS 16.0+ | macOS 13.0+ | Mac Catalyst 16.0+ | tvOS 16.0+ | watchOS 9.0+

---

## Declaration

```
func navigationDestination<D, C>(for data: D.Type, @ViewBuilder destination: @escaping (D) -> C) -> some View where D : Hashable, C : View
```

## Parameters

`data`
The type of data that this destination matches.

`destination`
A view builder that defines a view to display when the stack's navigation state contains a value of type `data`. The closure takes one argument, which is the value of the data to present.

## Discussion

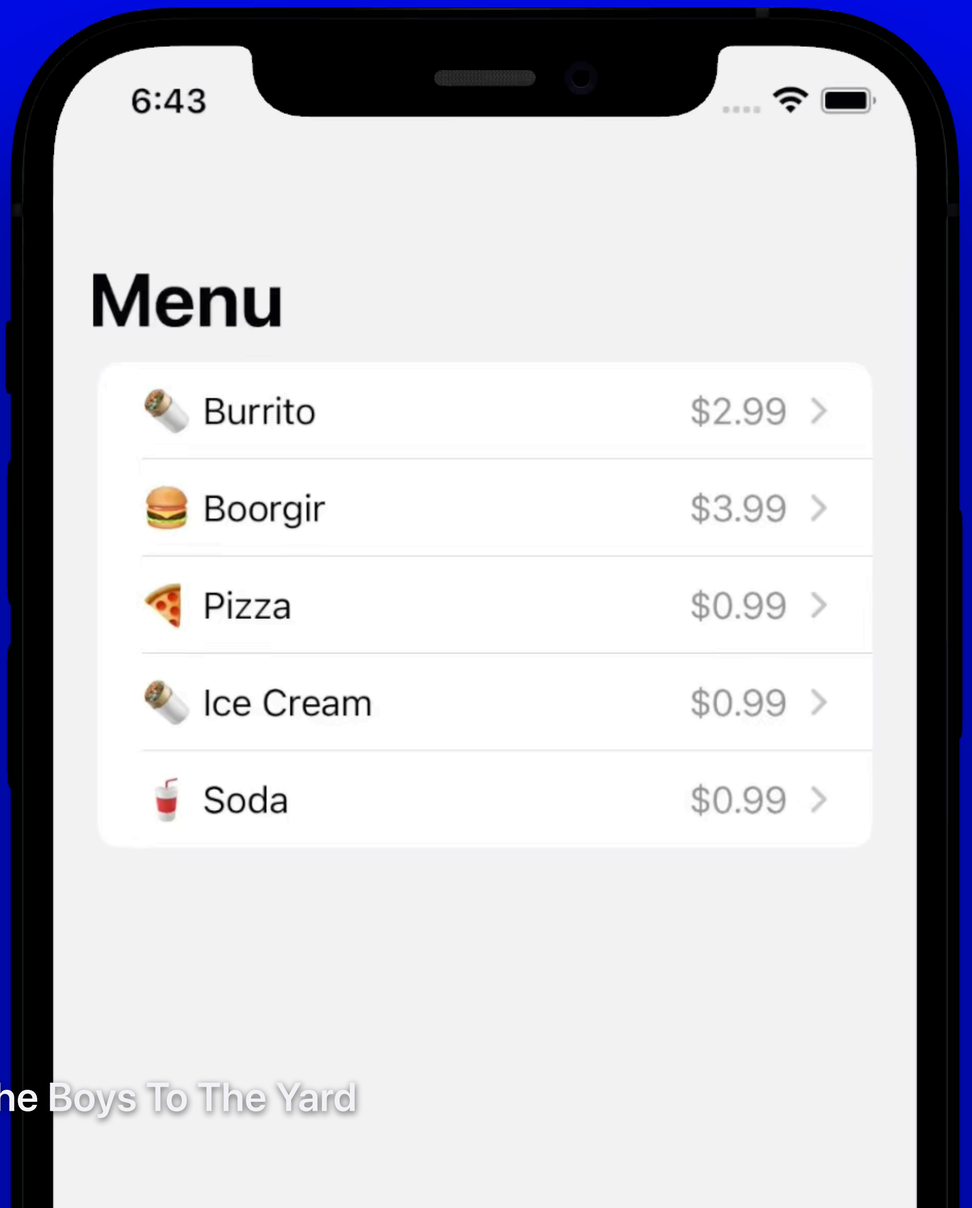
Add this view modifer to a view inside a `NavigationStack` to describe the view that the stack displays when presenting a particular kind of data. Use a `NavigationLink` to present the data. For example, you can present a `ColorDetail` view for each presentation of a `Color` instance:

# 👋🏾 NavigationStack

```
struct ContentView: View {
  var body: some View {
    NavigationStack {
      List {
          ForEach(Item.data, id: \.name) { item in
           NavigationLink(value: item) {
               content(item)
           }
        }
      }
      .navigationTitle(“Menu")
      .navigationDestination(for: Item.self) { item in
        ItemDetailView(item: item)
      }
    }
  }
}
```

My NavStack Brings All The Boys To The Yard

# 👋 NavigationStack




My NavStack Brings All The Boys To The Yard

# 👀 Wut’s goin on ere

**NavigationStack**

🙅🏾 No items on the navigation stack

My NavStack Brings All The Boys To The Yard

# 👀 Wut’s goin on ere

NavigationStack

Boorgir

1️⃣ Item on the navigation stack

My NavStack Brings All The Boys To The Yard

# 👀 Wut’s goin on ere

NavigationStack

💨 Popping an item on the stack

My NavStack Brings All The Boys To The Yard

# 👀 Wut’s goin on ere

NavigationStack

Promos

## 🛍️ Multiple items on the stack

My NavStack Brings All The Boys To The Yard

# 👀 Wut’s goin on ere


## 🛍️ Multiple items on the stack

My NavStack Brings All The Boys To The Yard

# 🌈 Handling Different Routes

My NavStack Brings All The Boys To The Yard

```
List {
    Section("Promotions") {
       PromoView()
            .background(NavigationLink("", value: Promo.data))
    }
    .listRowBackground(Color.clear)
    .headerProminence(.increased)

    Section("Items") {
        ForEach(Item.data, id: \.name) { item in
            NavigationLink(value: item) {
                content(item)
            }
        }
    }
    .headerProminence(.increased)
}
.navigationTitle("Menu")
.navigationDestination(for: Item.self) { item in
    ItemDetailView(item: item)
}
```

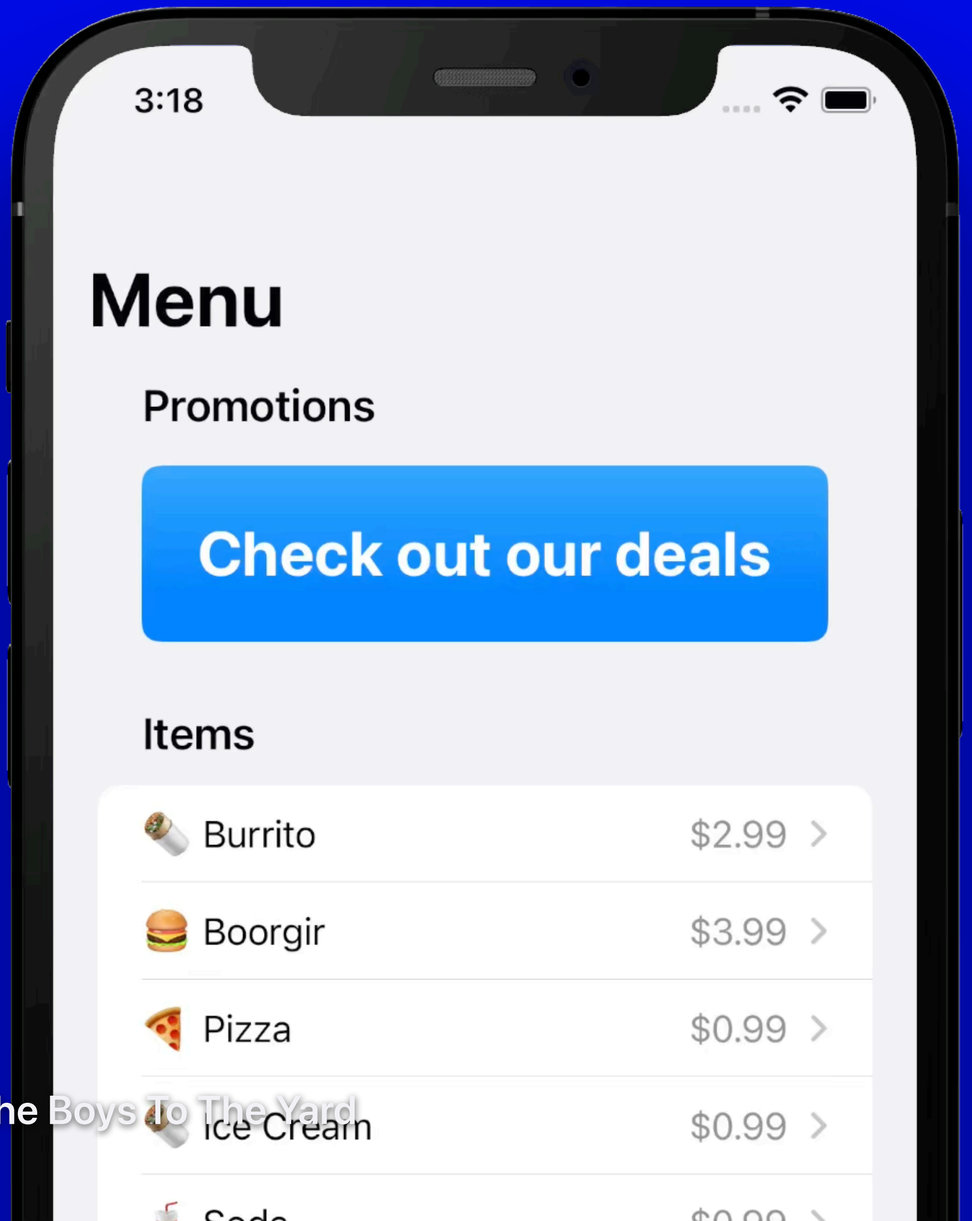
```
List {
    Section("Promotions") {
       PromoView()
            .background NavigationLink("", value: Promo.data )
    }
    .listRowBackground(Color.clear)
    .headerProminence(.increased)

    Section("Items") {
        ForEach(Item.data, id: \.name) { item in
            NavigationLink(value: item) {
                content(item)
            }
        }
    }
    .headerProminence(.increased)
}
.navigationTitle("Menu")
.navigationDestination(for: Item.self) { item in
    ItemDetailView(item: item)
}
```

# 🌈 Handling Different Routes

```
PromoView()
  .background(NavigationLink(“”, value: Promo.data))
```

My NavStack Brings All The Boys To The Yard

# 🌈 Handling Different Routes



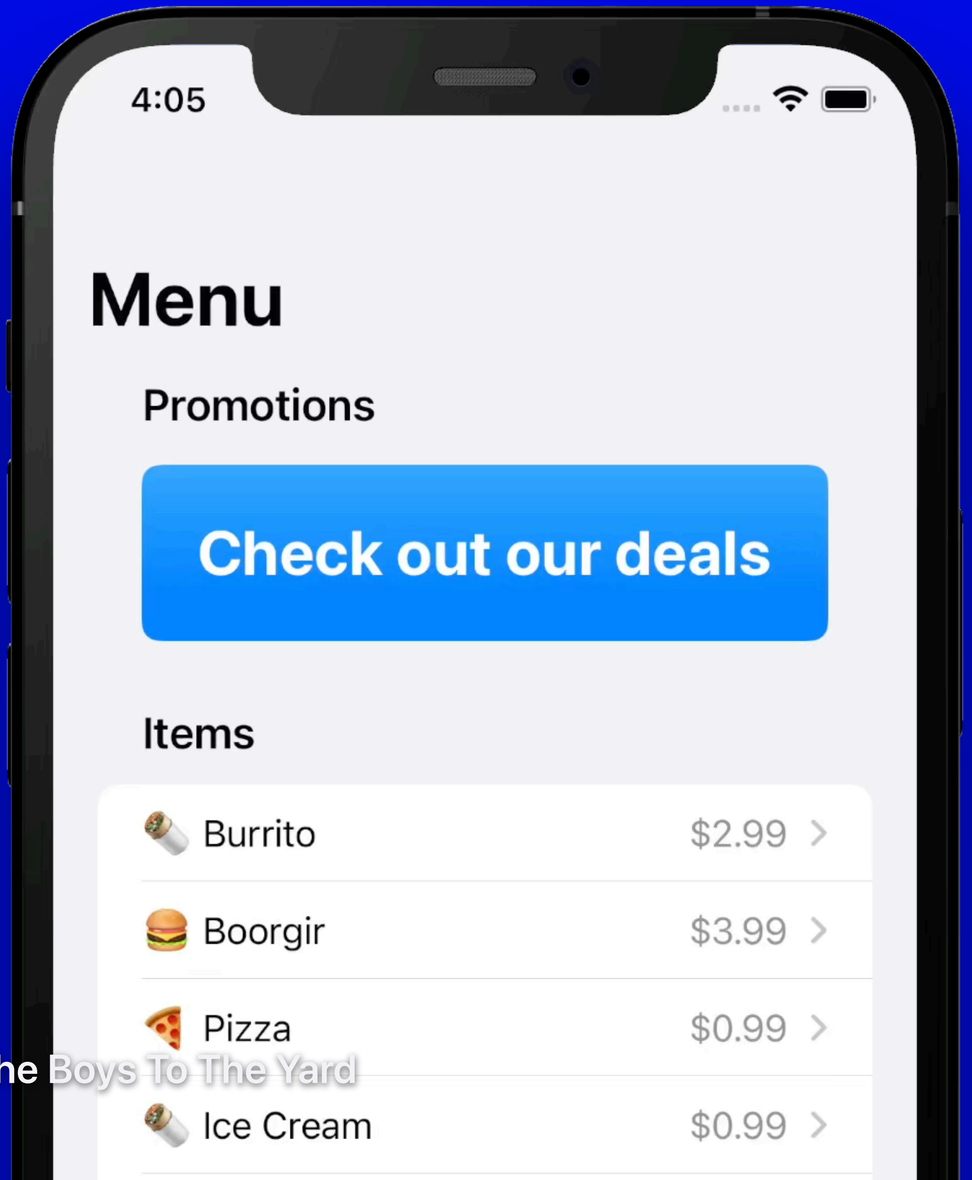

My NavStack Brings All The Boys To The Yard

# 🌈 Handling Different Routes

```
List {

  /*
    Section code here..
   */

}
.navigationDestination(for: Item.self) { item in
    ItemDetailView(item: item)
}
.navigationDestination(for: Promo.self) { data in
    PromoListView(promos: data)
}
```

My NavStack Brings All The Boys To The Yard

# 🌈 Handling Different Routes




My NavStack Brings All The Boys To The Yard

# 🤝 Don’t You Just Love Enums

```
List {

   Section("Promotions") {
     PromoView()
       .background(NavigationLink(“", value: Promo.data[0]))
   }

  /*
    Section code here..
   */

}
.navigationDestination(for: Item.self) { item in
    ItemDetailView(item: item)
}
.navigationDestination(for: Promo.self) { data in
    PromoListView(promos: data)
}
```

# 🤝 Don’t You Just Love Enums

```
List {

   Section("Promotions") {
     PromoView()
       .background(NavigationLink(“", value: Promo.data[0]))
   }

  /*
    Section code here..
   */

}
.navigationDestination(for: Item.self) { item in
     ItemDetailView(item: item)
}
.navigationDestination(for: Promo.self) { data in
     PromoListView(promos: data)
}
```

# 🤝 Don’t You Just Love Enums

```
enum Route: Hashable {
    case item(Item)
    case promos([Promo])
}
```

My NavStack Brings All The Boys To The Yard

# 🤝 Don’t You Just Love Enums

```
Section("Promotions") {
        PromoView()
        .background(NavigationLink("", value: Route.promos(Promo.data)))
}

Section("Items") {
    ForEach(Item.data, id: \.name) { item in
        NavigationLink(value: Route.item(item)) {
            content(item)
        }
    }
}
```

My NavStack Brings All The Boys To The Yard

# 🤝 Don’t You Just Love Enums

```
List {
    // Section code here…
 }
 .navigationTitle("Menu")
  .navigationDestination(for: Route.self) { route in
      switch route {
      case .item(let data):
          ItemDetailView(item: data)
      case .promos(let data):
          PromoListView(promos: data)
      }
  }
```

My NavStack Brings All The Boys To The Yard

# 🤝 Don’t You Just Love Enums

```
List {
    // Section code here…
 }
 .navigationTitle("Menu")
  .navigationDestination(for: Rout
      switch route {
      case .item(let data):
          ItemDetailView(item: dat
      case .promos(let data):
          PromoListView(promos: da
      }
  }
```

My NavStack Brings All The Boys To The Yard

# 🤝 Don’t You Just Love Enums

```
enum Route: Hashable {
    case item(Item)
    case promos([Promo])
}
```

My NavStack Brings All The Boys To The Yard

# 🤝 Don’t You Just Love Enums

```
extension Route: View {

    var body: some View {
        switch self {
        case .item(let data):
            ItemDetailView(item: data)
        case .promos(let data):
            PromoListView(promos: data)
        }
    }
}
```

My NavStack Brings All The Boys To The Yard

# 🤝 Don’t You Just Love Enums

```
List {
    // Section code here…
 }
 .navigationTitle("Menu")
  .navigationDestination(for: Route.self) { route in
      switch route {
      case .item(let data):
          ItemDetailView(item: data)
      case .promos(let data):
          PromoListView(promos: data)
      }
  }
```

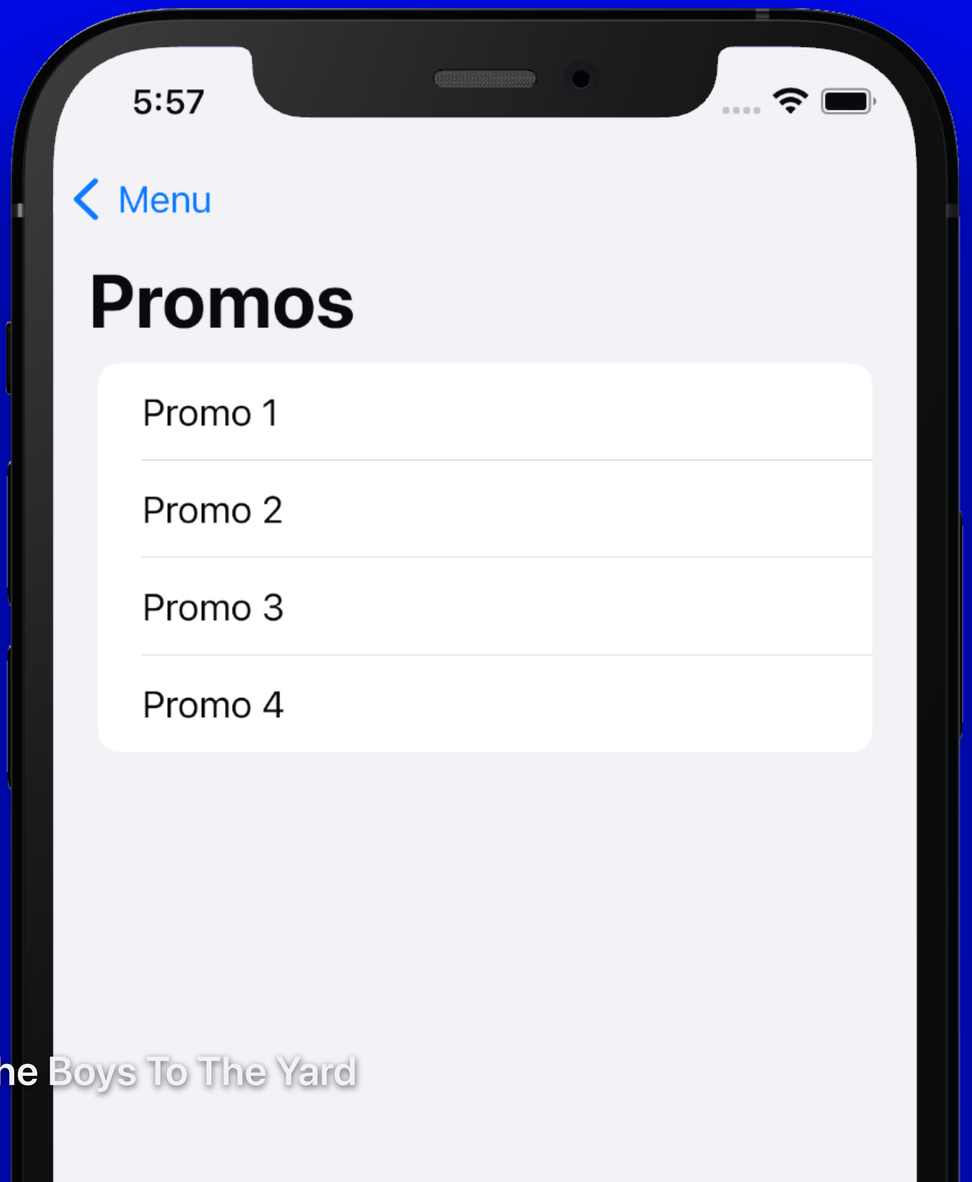
My NavStack Brings All The Boys To The Yard

# 🤝 Don’t You Just Love Enums

```
List {
    // Section code here…
 }
 .navigationTitle("Menu")
 .navigationDestination(for: Route.self) { $0 }
```

My NavStack Brings All The Boys To The Yard

# 👱 Think About The Kids MAN




My NavStack Brings All The Boys To The Yard

# 👦 Think About The Kids MAN


My NavStack Brings All The Boys To The Yard

# 👦 Think About The Kids MAN

```
enum Route: Hashable {
    case item(Item)
    case promos([Promo])
    case promo(Promo)
}
```

My NavStack Brings All The Boys To The Yard

# 👦 Think About The Kids MAN

```
enum Route: Hashable {
    case item(Item)
    case promos([Promo])
    case promo(Promo)
}
```

My NavStack Brings All The Boys To The Yard

# 👦 Think About The Kids MAN

```swift
extension Route: View {

    var body: some View {
        switch self {
        case .item(let data):
            ItemDetailView(item: data)
        case .promos(let data):
            PromoListView(promos: data)
        case .promo(let data):
            PromoDetailView(promo: data)
        }
    }
}
```

My NavStack Brings All The Boys To The Yard

# 👱 Think About The Kids MAN

```
extension Route: View {

    var body: some View {
        switch self {
        case .item(let data):
            ItemDetailView(item: data)
        case .promos(let data):
            PromoListView(promos: data)
        case .promo(let data):
            PromoDetailView(promo: data)
        }
    }
}
```

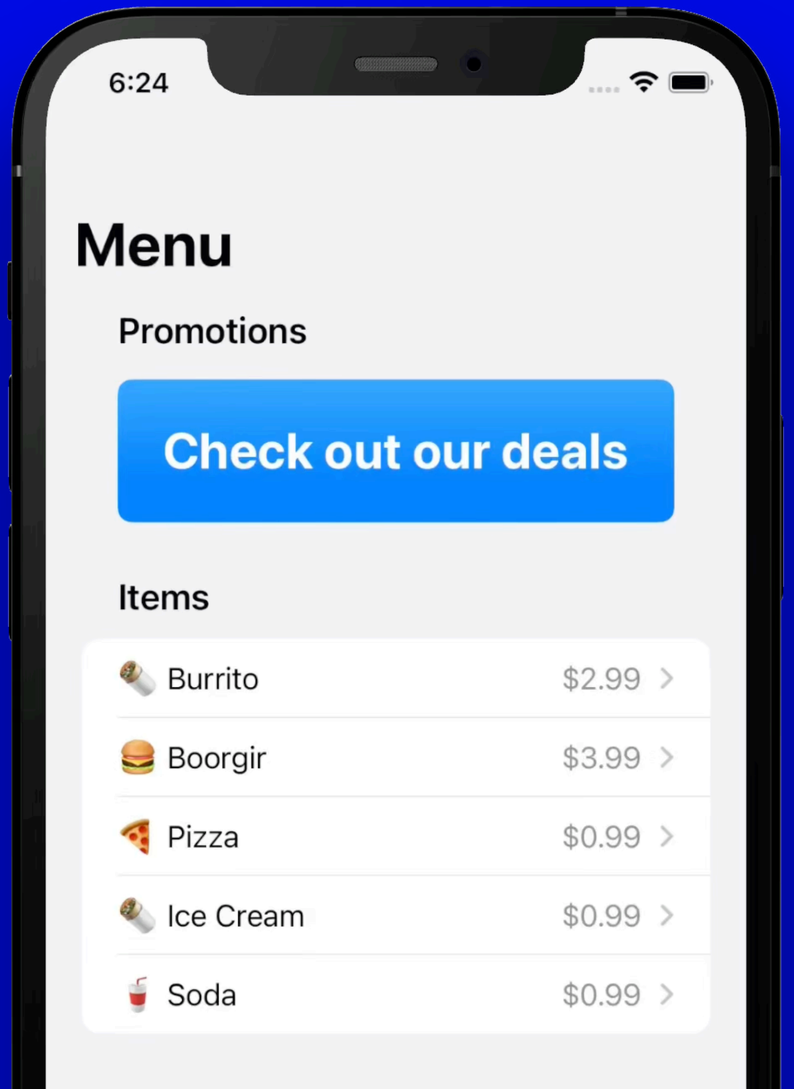
My NavStack Brings All The Boys To The Yard

# 👦 Think About The Kids MAN

```
List {
    ForEach(promos, id: \.name) { promo in
        NavigationLink(promo.name, value: Route.promo(promo))
    }
}
.navigationTitle("Promos")
```

My NavStack Brings All The Boys To The Yard

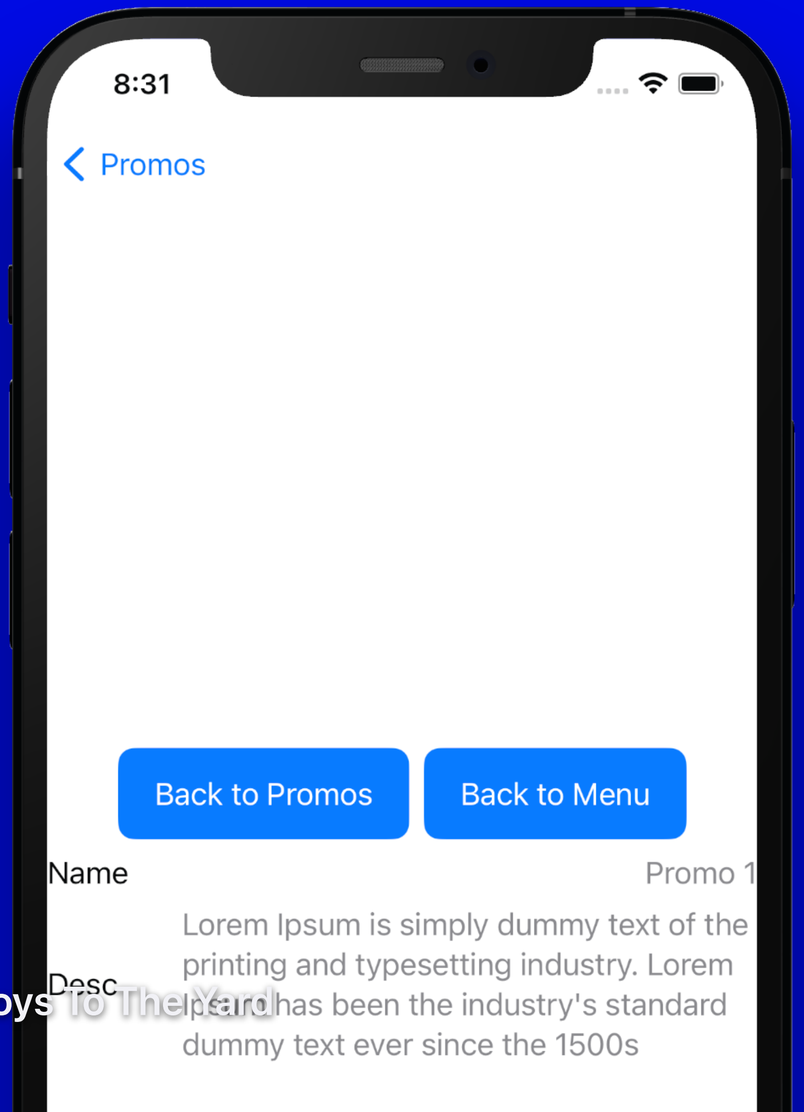
# 👦 Think About The Kids MAN



# 🤖 Errr But What About Programmatic

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic




My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic


My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

```
class RouteManager: ObservableObject {


}
```

My NavStack Brings All The Boys To The Yard

# Topics

## Creating a navigation stack

`init(root: () -> Root)`

Creates a navigation stack that manages its own navigation state.

## Creating a navigation stack with a path

`init(path: Binding<Data>, root: () -> Root)`

Creates a navigation stack with homogeneous navigation state that you can control.

`init(path: Binding<NavigationPath>, root: () -> Root)`

Creates a navigation stack with heterogeneous navigation state that you can control.

## Setting a destination for a value

`func navigationDestination<D, C>(for: D.Type, destination: (D) -> C) -> some View`

Associates a destination view with a presented data type for use within a navigation stack.

## Supporting types

`var body: some View`

The content and behavior of the view.

`typealias Body`

The type of view representing the body of this view.

## Default Implementations

View Implementations

# Topics

## Creating a navigation stack

```
init(root: () -> Root)
```

Creates a navigation stack that manages its own navigation state.

## Creating a navigation stack with a path

```
init(path: Binding<Data>, root: () -> Root)
```

Creates a navigation stack with homogeneous navigation state that you can control.

```
init(path: Binding<NavigationPath>, root: () -> Root)
```

Creates a navigation stack with heterogeneous navigation state that you can control.

## Setting a destination for a value

```
func navigationDestination<D, C>(for: D.Type, destination: (D) -> C) -> some View
```

Associates a destination view with a presented data type for use within a navigation stack.

# 🤖 Errr But What About Programmatic

```
class RouteManager: ObservableObject {

    @Published var routes: [Route] = []

}
```

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

```
class RouteManager: ObservableObject {

    @Published var routes: [Route] = []

    func pop() {
        _ = routes.popLast()
    }

    func reset() {
        routes = []
    }

    func push(_ route: Route) {
        routes.append(route)
    }
}
```

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

```
struct ContentView: View {
    @StateObject private var routeManager = RouteManager()
    var body: some View {
        NavigationStack {
            /****/
```

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

```
struct ContentView: View {
    @StateObject private var routeManager = RouteManager()
    var body: some View {
        NavigationStack(path: $routeManager.routes) {
            /****/
```

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

```
struct ContentView: View {

    @StateObject private var routeManager = RouteManager()

    var body: some View {

        NavigationStack(path: $routeManager.routes) {

            /****/
```

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

```
NavigationStack(path: $routeManager.routes) { /***/ }
 .environmentObject(routeManager)
```

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

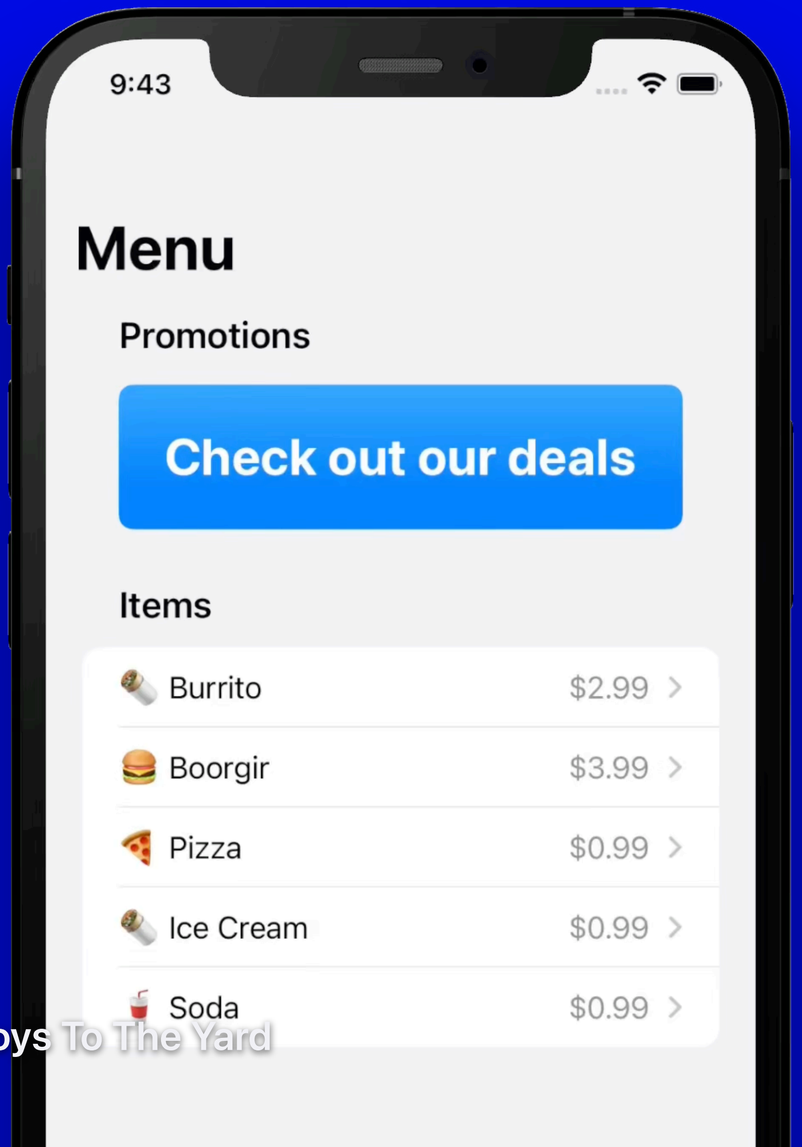
```
struct PromoDetailView: View {

    @EnvironmentObject var routeManager: RouteManager

    let promo: Promo

    /***/
```

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

```
struct PromoDetailView: View {

    @EnvironmentObject var routeManager: RouteManager

    let promo: Promo

    /***/

    Button("Back to Promos") {
      routeManager.pop()
    }

    Button("Back to Menu") {
      routeManager.reset()
    }
```

# 🤖 Errr But What About Programmatic



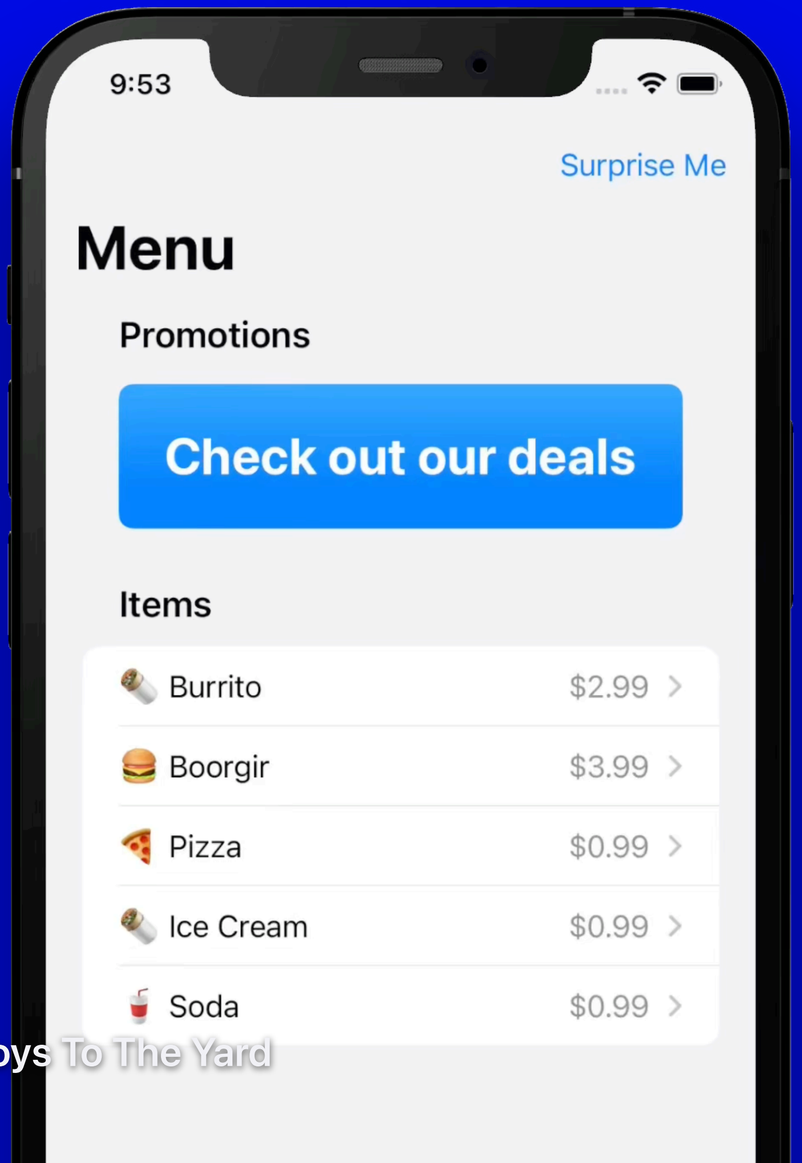

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic

```
.toolbar {
    Button("Surprise Me") {
        if let randomItem = Item.data.randomElement() {
            routeManager.push(Route.item(randomItem))
        }
    }
}
```

My NavStack Brings All The Boys To The Yard

# 🤖 Errr But What About Programmatic




My NavStack Brings All The Boys To The Yard

# 🤷🏾 Why Not NavigationPath?

My NavStack Brings All The Boys To The Yard

# Topics

## Creating a navigation stack

```
init(root: () -> Root)
```

Creates a navigation stack that manages its own navigation state.

## Creating a navigation stack with a path

```
init(path: Binding<Data>, root: () -> Root)
```

Creates a navigation stack with homogeneous navigation state that you can control.

```
init(path: Binding<NavigationPath>, root: () -> Root)
```

Creates a navigation stack with heterogeneous navigation state that you can control.

## Setting a destination for a value

```
func navigationDestination<D, C>(for: D.Type, destination: (D) -> C) -> some View
```

Associates a destination view with a presented data type for use within a navigation stack.

## Managing path contents

`var isEmpty: Bool`

A Boolean that indicates whether this path is empty.

`var count: Int`

The number of elements in this path.

`func append<V>(V)`

Appends a new codable value to the end of this path.

`func append<V>(V)`

Appends a new value to the end of this path.

`func removeLast(Int)`

Removes values from the end of this path.

## Encoding a path

`var codable: NavigationPath.CodableRepresentation?`

A value that describes the contents of this path in a serializable format.

`struct CodableRepresentation`

A serializable representation of a navigation path.

## Default Implementations

Equatable Implementations

---

# Relationships

## Managing path contents

```
var isEmpty: Bool
```

A Boolean that indicates whether this path is empty.

```
var count: Int
```

The number of elements in this path.

```
func append<V>(V)
```

Appends a new codable value to the end of this path.

```
func append<V>(V)
```

Appends a new value to the end of this path.

```
func removeLast(Int)
```

Removes values from the end of this path.

# 🤷🏾 Why Not NavigationPath?

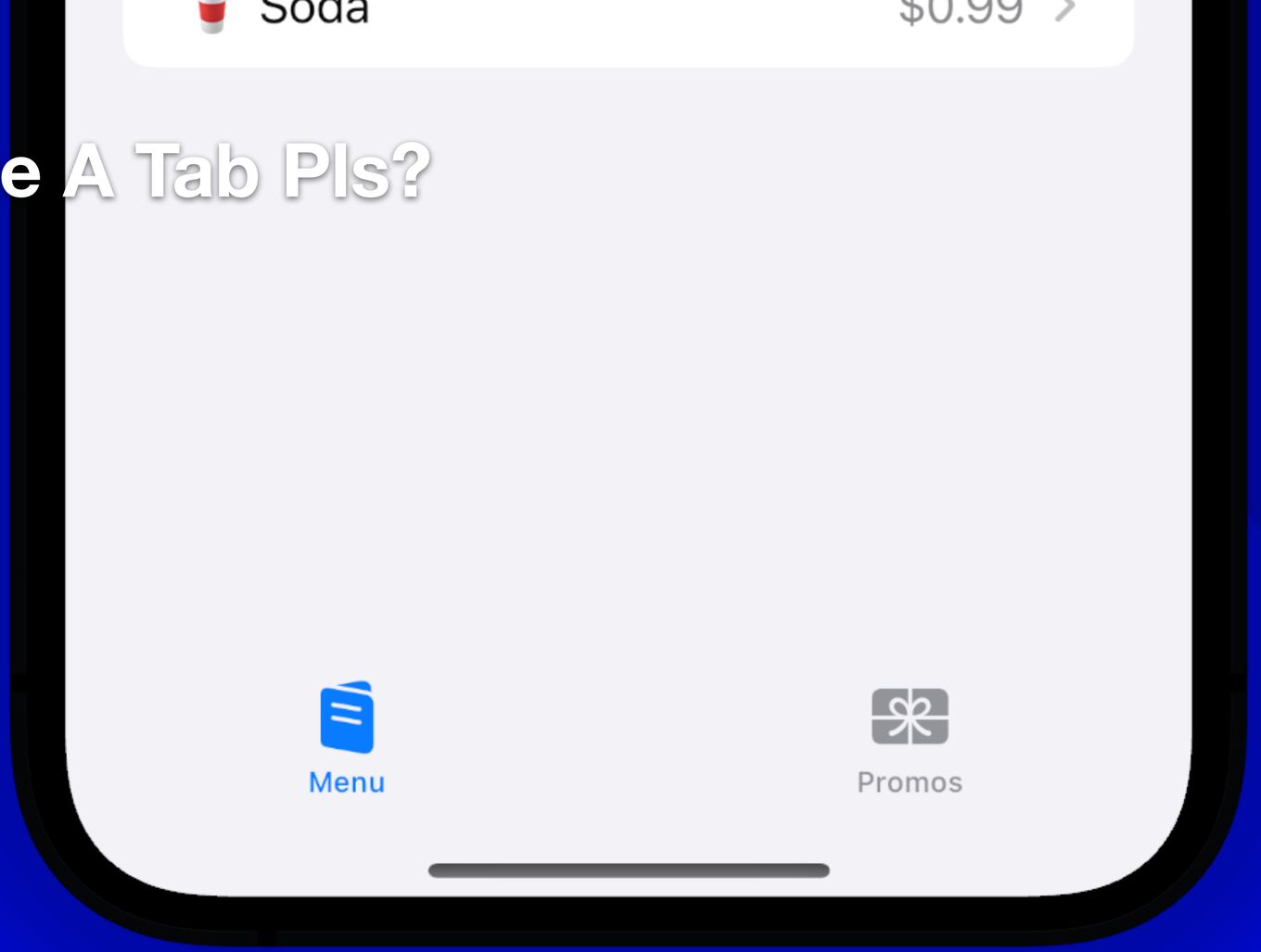
# 💎 ENUMS ARE A HIDDEN GEM 💎

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?




My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

```
TabView {
    MenuScreen()
        .tabItem {
            Label("Menu", systemImage: "menucard")
        }
    PromoScreen()
        .tabItem {
            Label("Promos", systemImage: "giftcard")
        }
}
```

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

📱

## Screens Represent The Parent

A parent holds a bunch components, that you'll see on the screen.

📦

## Views Represent The Components

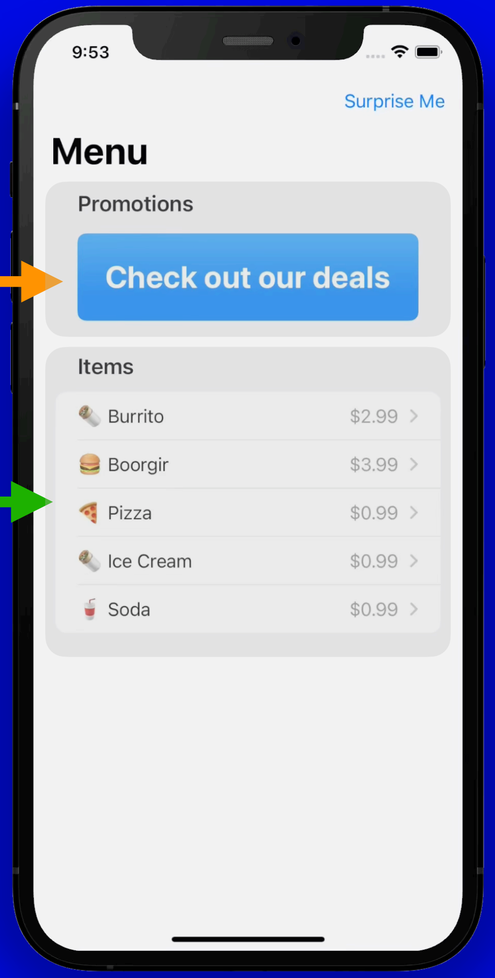
A view is a component that has a single purpose on the screen and can be reused.

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

PromoView

MenuView




MenuScreen

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

📱

## Screens Represent The Parent

A parent holds a bunch components, that you'll see on the screen.

📦

## Views Represent The Components

A view is a component that has a single purpose on the screen and can be reused.

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

```
TabView {
    MenuScreen()
        .tabItem {
            Label("Menu", systemImage: "menucard")
        }
    PromoScreen()
        .tabItem {
            Label("Promos", systemImage: "giftcard")
        }
}
```

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

```
@StateObject private var manager = RouteManager()

TabView {
    MenuScreen()
        .tabItem {
            Label("Menu", systemImage: "menucard")
        }
    PromoScreen()
        .tabItem {
            Label("Promos", systemImage: "giftcard")
        }
}
```

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

```
@StateObject private var manager = RouteManager()

TabView {

    MenuScreen()
        .tabItem {
            Label("Menu", systemImage: "menucard")
        }

    PromoScreen()
        .tabItem {
            Label("Promos", systemImage: "giftcard")
        }
```

My NavStack Brings All The Boys To The Yard

```
Group {

    NavigationStack(path: $manager.routes) {
        MenuScreen()
            .navigationDestination(for: Route.self) { $0 }
    }
    .tabItem {
        Label("Menu", systemImage: "menucard")
    }

    NavigationStack(path: $manager.routes) {
        PromoScreen()
            .navigationDestination(for: Route.self) { $0 }
    }
    .tabItem {
        Label("Promos", systemImage: "giftcard")
    }
}
.environmentObject(manager)
```

```
Group {

    NavigationStack(path: $manager.routes) {
        MenuScreen()
            .navigationDestination(for: Route.self) { $0 }
    }
    .tabItem { /***/ }

    NavigationStack(path: $manager.routes) {
       PromoScreen()
            .navigationDestination(for: Route.self) { $0 }
    }
    .tabItem { /***/ }

}
.environmentObject(manager)
```

```swift
enum Tab {
    case menu
    case promo
}

@Published var menuRoutes: [Route] = []
@Published var promoRoutes: [Route] = []

func pop(_ tab: Tab) {
    switch tab {
    case .menu:
        _ = menuRoutes.popLast()
    case .promo:
        _ = promoRoutes.popLast()
    }
}

func reset(_ tab: Tab) {
    switch tab {
    case .menu:
        menuRoutes = []
    case .promo:
        promoRoutes = []
    }
}
```

```swift
enum Tab {
    case menu
    case promo
}

@Published var menuRoutes: [Route] = []
@Published var promoRoutes: [Route] = []

func pop(_ tab: Tab) {
    switch tab {
    case .menu:
        _ = menuRoutes.popLast()
    case .promo:
        _ = promoRoutes.popLast()
    }
}

func reset(_ tab: Tab) {
    switch tab {
    case .menu:
        menuRoutes = []
    case .promo:
        promoRoutes = []
    }
}
```

```
func pop(_ tab: Tab) {
    switch tab {
    case .menu:
        _ = menuRoutes.popLast()
    case .promo:
        _ = promoRoutes.popLast()
    }
}

func reset(_ tab: Tab) {
    switch tab {
    case .menu:
        menuRoutes = []
    case .promo:
        promoRoutes = []
    }
}

func push(_ route: Route, on tab: Tab) {
    switch tab {
    case .menu:
        menuRoutes.append(route)
    case .promo:
        promoRoutes.append(route)
    }
}
```

```
func pop(_ tab: Tab) {
    switch tab {
    case .menu:
        _ = menuRoutes.popLast()
    case .promo:
        _ = promoRoutes.popLast()
    }
}

func reset(_ tab: Tab) {
    switch tab {
    case .menu:
        menuRoutes = []
    case .promo:
        promoRoutes = []
    }
}

func push(_ route: Route, on tab: Tab) {
    switch tab {
    case .menu:
        menuRoutes.append(route)
    case .promo:
        promoRoutes.append(route)
    }
}
```

# 😅 Can I Have A Tab Pls?

```
NavigationStack(path: $manager.menuRoutes) {
    MenuScreen()
        .navigationDestination(for: Route.self) { $0 }
}
.tabItem { /***/ }

NavigationStack(path: $manager.promoRoutes) {
    PromoScreen()
        .navigationDestination(for: Route.self) { $0 }
}
.tabItem { /***/ }
```

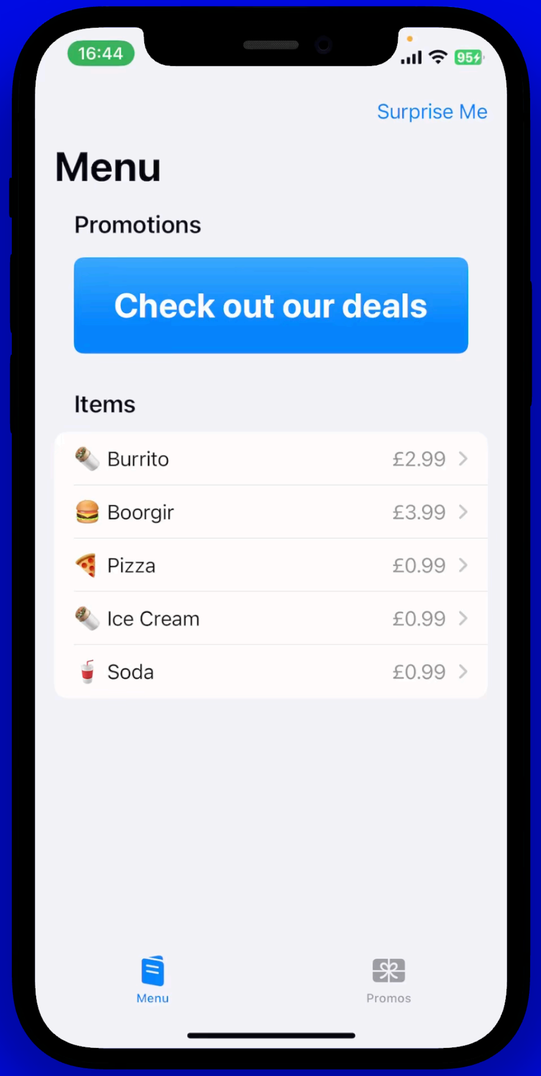
My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?

```
.toolbar {
    Button("Surprise Me") {
        if let randomItem = Item.data.randomElement() {
            routeManager.push(Route.item(randomItem),
                              on: .menu)
        }
    }
}
```

My NavStack Brings All The Boys To The Yard

# 😅 Can I Have A Tab Pls?




My NavStack Brings All The Boys To The Yard

# 🔗 Deep Links

My NavStack Brings All The Boys To The Yard

# 🔗 Deep Links

💭

## What is a deep link?

A link that opens up a specific screen with your app, to display content.

👀

## URL Format

It follows the same format of a URL where we have paths & query items.

🔗

## Custom URL Scheme

A link that is registered within your project info.plist

🍎

## Universal Links

An ***Apple Associate*** file is hosted on your website, and you're app handles website routes to display content in your app

**My NavStack Brings All The Boys To The Yard**

# 🔗 Some Possible Deep Links Use Cases

🌠

## Promoting A Product

Deep links into promotions or specific products within your app you want to push

🛍️

## Access To Content

Links to specific screens, such as new items being added that you want users to see

🤝

## Sharing In App Content

A link that you generate that when shared with other users, it will open up your app and access that content

My NavStack Brings All The Boys To The Yard

# RocketSim

## Enhancing the iOS Simulator

Compare designs, show rulers, add a grid, quick actions for recent builds. Create recordings with touches & audio, trim and export them into MP4 or GIF and share them anywhere using drag & drop. Add bezels to screenshots and videos.

And that's not all. Organise and launch Universal Links (deeplinks) with ease.

Download now

Team Licenses >

4.7 Rating

App Store Reviews

Essential Apps for Developers

App Store Featured

## Trusted by Teams Around the World

Google RevenueCat we CHECK24

Apple DATADOG BBC sky NEWS adidas

MenuScreen | NavigationViewExample | NavigationViewExampleApp | NavRouter | PromoListView | PromoDetailView | PromoView

NavigationViewExample

General | Signing & Capabilities | Resource Tags | Info | Build Settings | Build Phases | Build Rules

PROJECT

NavigationViewExa...

TARGETS

NavigationViewExa...

## Custom iOS Target Properties

| Key | Type | Value |
|---|---|---|
| Bundle name | String | $(PRODUCT_NAME) |
| Bundle identifier | String | $(PRODUCT_BUNDLE_IDENTIFIER) |
| InfoDictionary version | String | 6.0 |
| Supported interface orientations (iPhone) | Array | (3 items) |
| Bundle version | String | $(CURRENT_PROJECT_VERSION) |
| Application supports indirect input events | Boolean | YES |
| Application Scene Manifest | Dictionary | (2 items) |
| Application requires iPhone environment | Boolean | YES |
| Executable file | String | $(EXECUTABLE_NAME) |
| Bundle OS Type code | String | $(PRODUCT_BUNDLE_PACKAGE_TYPE) |
| Launch Screen | Dictionary | (1 item) |
| Default localization | String | $(DEVELOPMENT_LANGUAGE) |
| Supported interface orientations (iPad) | Array | (4 items) |
| Bundle version string (short) | String | $(MARKETING_VERSION) |

## Document Types (0)

## Exported Type Identifiers (0)

## Imported Type Identifiers (0)

## URL Types (1)

com.tundsdev.NavigationViewExample

No image specified

Identifier: com.tundsdev.NavigationViewExample

Icon: None

URL Schemes: mynavapp

Role: Editor

Additional url type properties (0)

| Key | Type | Value |
|---|---|---|

Click here to add additional url type properties

Bundle version string (short) String $(MARKETING_VERSION)

rs (0)

rs (0)

com.tundsdev.NavigationViewExample

No image specified

Identifier com.tundsdev.NavigationViewExample

Icon None

URL Schemes mynavapp

Role Editor

Additional url type properties (0)

| Key | Type | Value |
| --- | --- | --- |

Click here to add additional url type properties

+

Instance Method

# onOpenURL(perform:)

Registers a handler to invoke when the view receives a url for the scene or window the view is in.

iOS 14.0+ | iPadOS 14.0+ | macOS 11.0+ | Mac Catalyst 14.0+ | tvOS 14.0+ | watchOS 7.0+

## Declaration

```
func onOpenURL(perform action: @escaping (URL) -> ()) -> some View
```

## Parameters

`action`
A function that takes a URL object as its parameter when delivering the URL to the scene or window the view is in.

## Discussion

> **Note**
> This method handles the reception of Universal Links, rather than a `NSUserActivity`.

## See Also

### Handling URLs

# 🔗 Deep Links

mynavapp://product?id=burrito

My NavStack Brings All The Boys To The Yard

```swift
class RouteManager: ObservableObject {

    enum Tab {
        case menu
        case promo
    }

    @Published var menuRoutes: [Route] = []
    @Published var promoRoutes: [Route] = []
    @Published var tab: Tab = .menu

    /**
     Action Code Here
    */

    func change(to tab: Tab) {
        self.tab = tab
    }
}
```

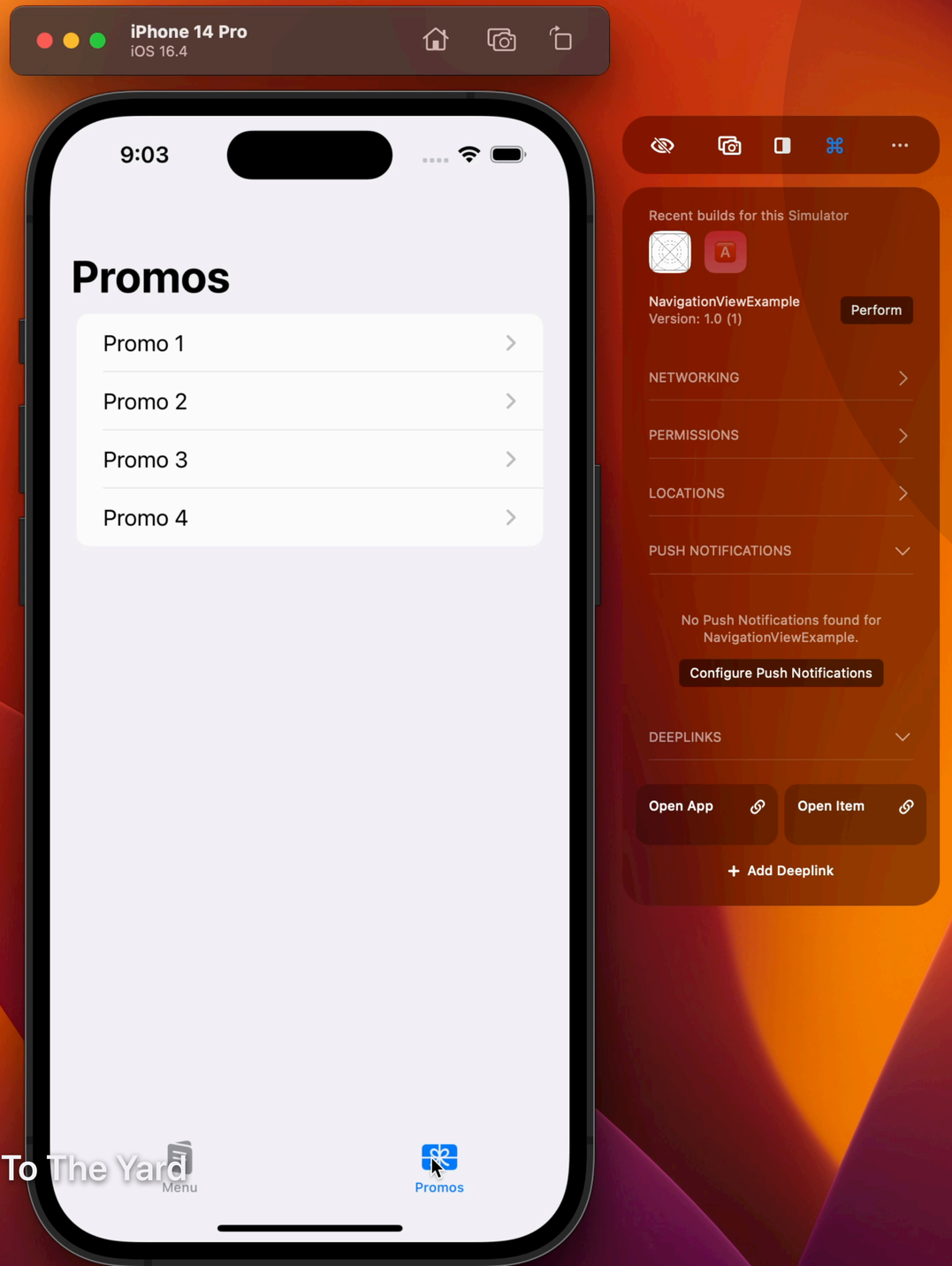
```
TabView(selection: $manager.tab) {

    Group {

        NavigationStack(path: $manager.menuRoutes) {
            MenuScreen()
                .navigationDestination(for: Route.self) { $0 }
        }
        .tag(RouteManager.Tab.menu)
        .tabItem { /***/ }

        NavigationStack(path: $manager.promoRoutes) {
            PromoScreen()
                .navigationDestination(for: Route.self) { $0 }
       }
        .tag(RouteManager.Tab.promo)
        .tabItem { /***/ }
    }
    .environmentObject(manager)
}
```

# 🔗 Deep Links

```
TabView(selection: $manager.tab) {•••}
.onOpenURL { url in

    // 1. Get the URL and build URLComponents
    guard let urlComponents = URLComponents(url: url, resolvingAgainstBaseURL: true),
          // 2. Check to see if the url matches product
          urlComponents.host == "product",
          // 3. Get the url's query items, get the key "id" and it's value lowercased
          let itemName = urlComponents.queryItems?.first(where: { $0.name == "id" })?.value?.lowercased(),
          // 4. Find the product based on the value associated to the key "id" lowercased
          let item = Item.data.first(where: { $0.name.lowercased() == itemName }) else { return }

    // 5. Switch to the tab
    manager.change(to: .menu)
    // 6. Push to the item on the menu screen using the active tab we've switched too
    manager.push(.item(item), on: manager.tab)
}
```

My NavStack Brings All The Boys To The Yard

# 🔗 Deep Links




My NavStack Brings All The Boys To The Yard

# 👈 Summary

NavigationView Limitations

New iOS16+ APIs

Organising Navigation Routes

Programmatic Navigation

NavigationPath

Handling Tab Bars

Deep Links

Architecture

Child Navigation

My NavStack Brings All The Boys To The Yard

# 📲 Check Out My Navigation Course


My NavStack Brings All The Boys To The Yard

SCAN ME

# 💅🏾 Treat Yourself


My NavStack Brings All The Boys To The Yard

# 👀 Check Out My Courses

# tunds.dev/#courses

## 👨🏾‍🏫 iOS Courses

I also offer free iOS development courses on my YouTube Channel in the form of videos and playlists, as well as courses on both Udemy & Teachable for more intermediate and advanced topics.

Check out my courses down below and see which one catches your eye 👀


**Become A SwiftUI Navigation Pro (All Levels | SwiftUI Navigation Stack | iOS Deep Links | iOS Push Notifications | SwiftUI Multiplatform)**

In this YouTube series, we'll look at everything you need to know when working with the new

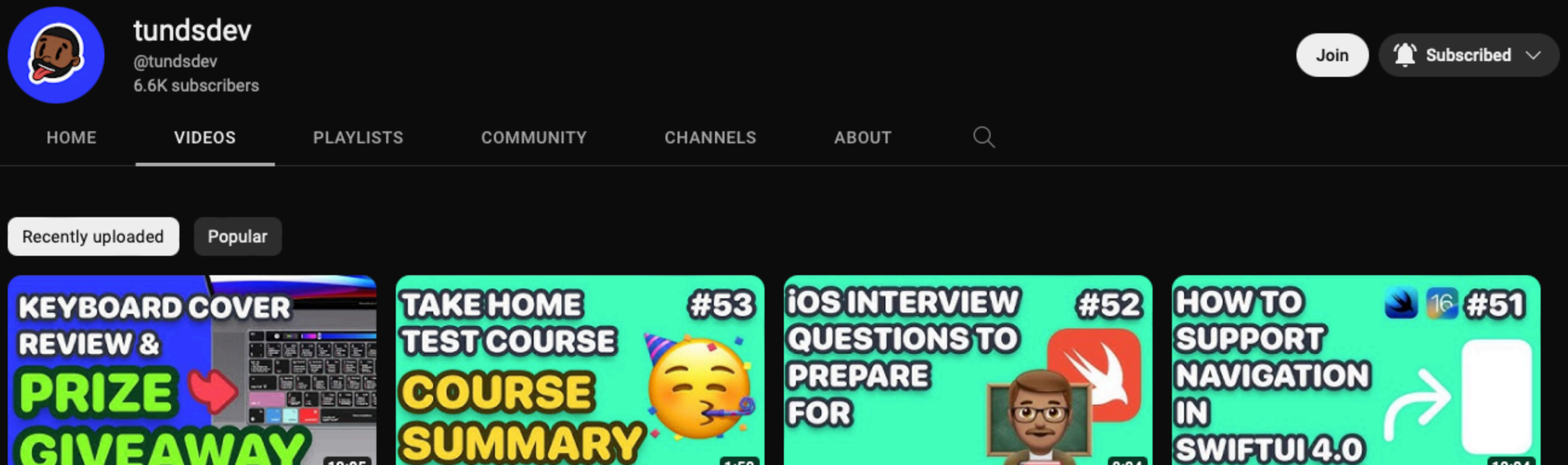

**Master Core Data With SwiftUI 👀 Build A Contacts App In SwiftUI**

In this YouTube series, we'll look at everything you need to know when getting started with


**SwiftUI Take Home Test Course**

In this YouTube series, we'll look at everything you need to know to ace your next iOS interview and take-home test. We'll look at how to build an iOS app that interacts with an API to fetch JSON data and display it on the screen.

# 📽️ Check Out My Channel

# Which Is The Lie? 🤥

A. I used to play for a Football academy (Not the egg game)

B. I enjoy eating dry cereal

C. I was once captured on TV in the background eating a McDonald's Apple

# Which Is The Lie? 🤥

A. I used to play for a Football academy (Not the egg game)

B. I enjoy eating dry cereal

C. I was once captured on TV in the background eating a McDonald's Apple


ritain's Favourite Supermarket Foods

2 Share

About this episode: Britain's Favourite Supermarket Foods

Clips and more...

# THANK YOU FOR LISTENING

# DEUCES ✌🏾

@tundsdev

@tundsdev

tunds.dev