

# Vistas, estructuras y propiedades

## Personalizar las vistas con propiedades



Crea una aplicación SwiftUI creando vistas personalizadas para hacer un pronóstico del tiempo de varios días. En tu vista personalizada, usarás propiedades para personalizar la pantalla para cada día.

Diseñas vistas personalizadas utilizando estructuras. Las estructuras son una forma de organizar su código para que las piezas relacionadas se empaqueten juntas. Cada estructura tiene un nombre que te permite reutilizarla como una plantilla en cualquier lugar de tu aplicación. Con las estructuras, puedes escribir código que sea más eficiente y tenga menos errores.

### Prototipo de una vista de pronóstico

Crea una interfaz sencilla para mostrar dos pronósticos meteorológicos.

Fíjate en el patrón repetido en la interfaz.

Mon	Tue
	
High: 70	High: 60
Low: 50	Low: 40

### Paso 1

En ContentView, elimine el código dentro del VStack y reemplázelo con un pronóstico del tiempo simple.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {

            Text("Mon")
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}

#Preview {
    ContentView()
}
```

Mon  
High: 70  
Low: 50

## Paso 2

Añade una vista de imagen a tu pronóstico y utilízala para mostrar un símbolo de SF de un sol.

Este tipo de imagen requiere un argumento, cuya etiqueta de argumento es `systemName`.

### Nota

Descarga la aplicación [SF Symbols](#) para navegar por miles de imágenes que puedes usar en toda una aplicación.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}

#Preview {
    ContentView()
}
```

Mon  
☀️  
High: 70  
Low: 50

## Paso 3

Usa el modificador `.foregroundColor` para hacer que el icono del sol sea amarillo.

---

Utilizas el modificador `.foregroundColor` para aplicar un color a los elementos de primer plano de cualquier vista. Por ejemplo, establecer un color como el estilo de primer plano de una vista de texto cambia el color del texto.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}

#Preview {
    ContentView()
}
```



Mon  
High: 70  
Low: 50

## Paso 4

Prepárate para añadir un segundo pronóstico haciendo clic con el botón derecho en VStack y eligiendo Insertar en HStack.

Un HStack es una vista de contenedor que organiza las vistas horizontalmente. Un VStack organiza las vistas que contiene verticalmente.

### Nota

No hay ningún cambio en la interfaz, porque el HStack solo contiene un elemento.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {
            VStack {

                Text("Mon")
                Image(systemName: "sun.max.fill")
                    .foregroundColor(Color.yellow)
                Text("High: 70")
                Text("Low: 50")

            }
            .padding()
        }
    }
}

#Preview {
    ContentView()
}
```

Mon  
☀️  
High: 70  
Low: 50



# Paso 5

Copie todo el VStack, incluido el modificador .padding, y luego pegue el código debajo del primer pronóstico para crear una segunda vista de pronóstico. Actualice los datos para mostrar el pronóstico del día siguiente.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {
            VStack {
                Text("Mon")
                Image(systemName: "sun.max.fill")
                    .foregroundColor(Color.yellow)
                Text("High: 70")
                Text("Low: 50")
            }
            .padding()
            VStack {
                Text("Tue")
                Image(systemName: "cloud.rain.fill")
                    .foregroundColor(Color.blue)
                Text("High: 60")
                Text("Low: 40")
            }
            .padding()
        }
    }
}

#Preview {
    ContentView()
}
```

Mon	Tue
	
High: 70	High: 60
Low: 50	Low: 40

# Crea una subvista personalizada

En lugar de seguir copiando y pegando su código, aprenda a encapsular el pronóstico de un día en su propia estructura para hacer una tarjeta de pronóstico personalizable.

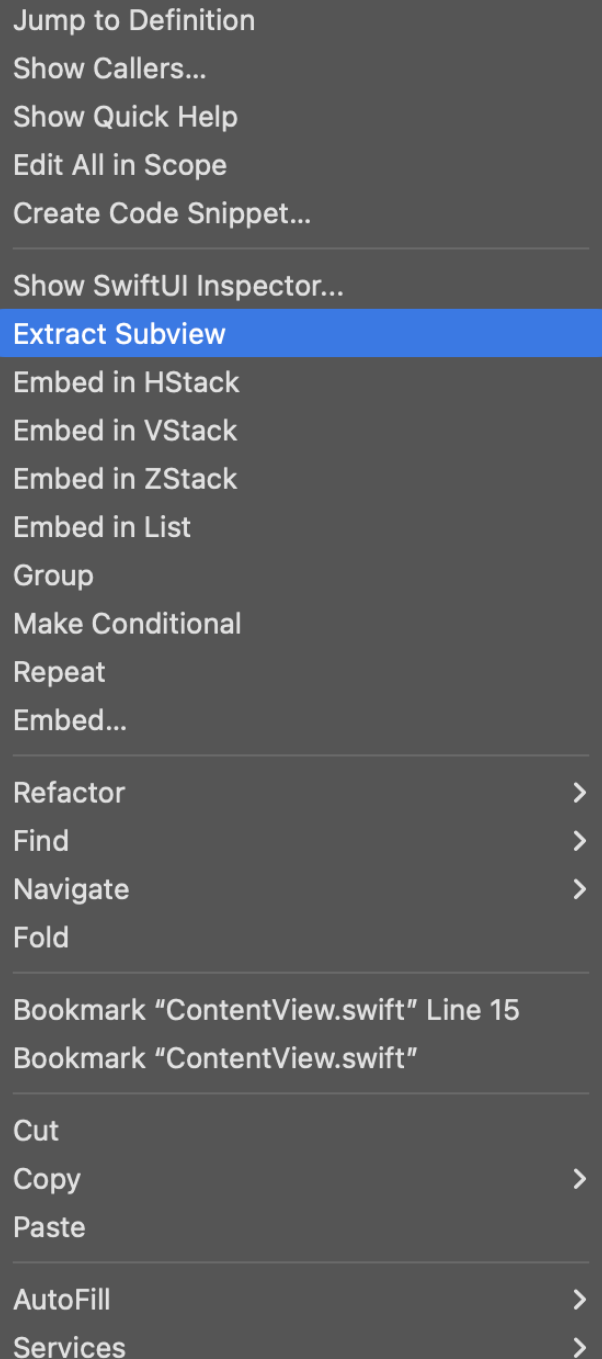
## Paso 1

Haga clic con el botón derecho en el VStack para ver el pronóstico del primer día. Xcode presenta un menú de acciones comunes. Elija Extraer Subvista.

Xcode reemplaza el VStack con ExtractedView(). Mire más abajo en su código para encontrar su nueva vista personalizada, struct ExtractedView.

### Nota

Una subvista es una vista que se utiliza dentro de otra vista. Por ejemplo, en el código predeterminado de un nuevo proyecto, el texto y la imagen son subvistas de ContentView.



## Paso 2

En ambas ubicaciones, cambio el nombre de `ExtractedView` a `DayForecast`. Ahora, en cualquier lugar que escribas `DayForecast()`, estás usando todo el código que extrajiste en la nueva vista.

---

Ahora que has creado una vista de `DayForecast`, puedes usarla para el pronóstico de cada día.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast()



            VStack {
                Text("Tue")
                Image(systemName: "cloud.rain.fill")
                    .foregroundColor(Color.blue)
                Text("High: 60")
                Text("Low: 40")
            }
                .padding()
        }
    }
}

#Preview {
    ContentView()
}

struct DayForecast: View {
    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
            .padding()
    }
}
```

Mon	Tue
	
High: 70	High: 60
Low: 50	Low: 40

# Paso 3

Reemplace el segundo pronóstico por otra vista de DayForecast.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast()

            DayForecast()

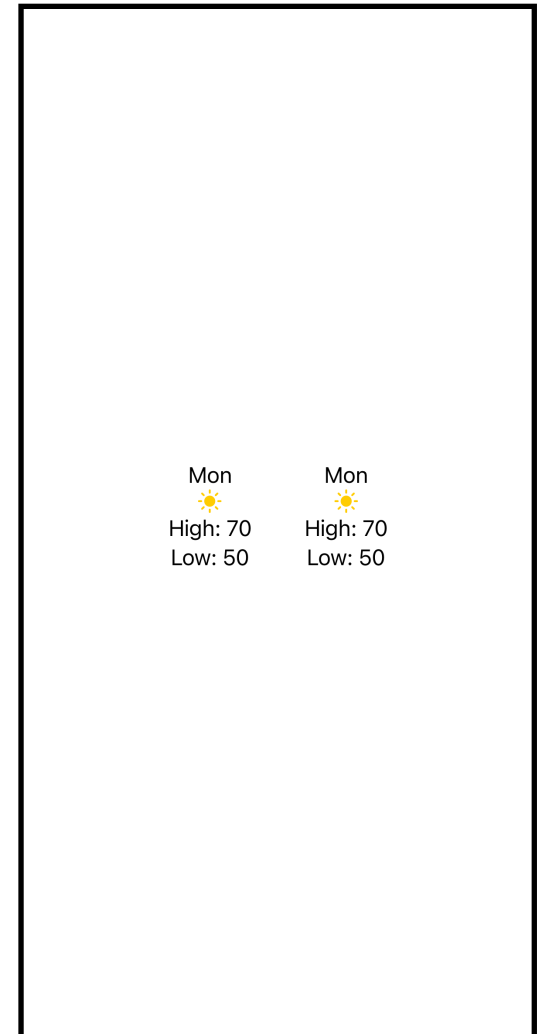
        }
    }
}

#Preview {
    ContentView()
}

struct DayForecast: View {
    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}
```





## Paso 4

Examina tu código. Hay tres lugares en los que hace referencia a DayForecast. Dentro de HStack, estás creando dos instancias de DayForecast para mostrarlas en la pantalla. En la parte inferior, estás definiendo la vista y su contenido.

---

### Nota

Usted define o declara - una estructura usando la palabra clave struct.

Utilizas o creas una instancia de la estructura escribiendo su nombre seguido de paréntesis.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast()

            DayForecast()

        }
    }
}



#Preview {
    ContentView()
}

struct DayForecast: View {

    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}
```

Mon	Mon
	
High: 70	High: 70
Low: 50	Low: 50

# Generalizar el día con una propiedad

Puede usar propiedades para mostrar cualquier dato de pronóstico utilizando su vista personalizada. Empieza por hacer que el día de la semana sea personalizable.

## Paso 1

Añade una nueva propiedad a la vista DayForecast para almacenar datos sobre el día de la semana. Añade una línea en blanco entre la propiedad y la propiedad del cuerpo debajo de ella.

---

Puedes usar líneas en blanco en Swift para hacer que tu código sea más legible; no tienen ningún impacto en la forma en que se ejecuta tu código.

### Importante

Xcode muestra un error, que corregirás en un momento.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast()

            DayForecast()

        }
    }
}

#Preview {
    ContentView()
}

struct DayForecast: View {
    let day: String

    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}
```

## Paso 2

Antes de corregir el error, eche un vistazo a la propiedad que acaba de crear.

---

Para declarar una propiedad, utilice la palabra clave `let`. Una propiedad tiene un nombre, seguido de dos puntos y luego el tipo de datos que contiene. El nombre de esta propiedad es `day`, y tiene una cadena.

### Nota

En Swift, los tipos de datos se llaman tipos. Puedes leer el código que agregaste a la estructura de `DayForecast` de la siguiente manera: "day es una propiedad cuyo tipo es `String`". O podrías decir: "el día es una propiedad de tipo `String`".

Nombre

Tipo

```
let day: String
```

## Paso 3

Mira el error en el que usas DayForecast por primera vez: "Argumento que falta para el parámetro 'día' en la llamada". Haga clic en el icono junto al mensaje de error en Xcode para expandirlo.

---

Cuando agregas una propiedad a una estructura, debes darle un valor cada vez que creas una instancia de la estructura. Este error significa que no le has dado un valor a la propiedad del día.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast()

            DayForecast()

        }
    }
}

#Preview {
    ContentView()
}

struct DayForecast: View {
    let day: String

    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}
```

## Paso 4

Haga clic en el botón Fix en el banner de error para agregar la etiqueta del argumento, día:.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast(day: String)

            DayForecast()

        }
    }
}

#Preview {
    ContentView()
}

struct DayForecast: View {
    let day: String

    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}
```

# Paso 5

Reemplace el marcador de posición con un día de la semana entre comillas.

---

Estás llamando al inicializador de `DayForecast` y pasando un argumento `String` para el parámetro `day`.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast(day: "Mon")

            DayForecast()

        }
    }
}

#Preview {
    ContentView()
}

struct DayForecast: View {
    let day: String

    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}
```

# Paso 6

Corrige el segundo error de la misma manera, usando un día diferente para el argumento.

La vista previa no cambia, porque todavía estás pasando una cadena literal en la primera vista de texto en la estructura DayForecast.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast(day: "Mon")

            DayForecast(day: "Tue")

        }
    }
}



#Preview {
    ContentView()
}

struct DayForecast: View {
    let day: String

    var body: some View {
        VStack {

            Text("Mon")
            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}
```

Mon	Mon
	
High: 70	High: 70
Low: 50	Low: 50

# Paso 7

Utilice la propiedad `day` en la vista `Texto` de su estructura `DayForecast` para que su vista previa refleje los datos que agregó anteriormente.

---

La propiedad `day` tiene un valor de cadena. Al usar la propiedad como argumento, estás pasando su valor a la vista `Texto`.

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack {

            DayForecast(day: "Mon")

            DayForecast(day: "Tue")

        }
    }
}

#Preview {
    ContentView()
}



struct DayForecast: View {
    let day: String

    var body: some View {
        VStack {

            Text(day)

            Image(systemName: "sun.max.fill")
                .foregroundColor(Color.yellow)
            Text("High: 70")
            Text("Low: 50")

        }
        .padding()
    }
}
```

Mon	Tue
	
High: 70	High: 70
Low: 50	Low: 50



## Paso 8

Echa un vistazo a cómo el valor de la cadena viaja desde tu código ContentView a una instancia de DayForecast y desde allí hasta tu interfaz.

---

Has generalizado DayForecast para mostrar el nombre de cualquier día.

