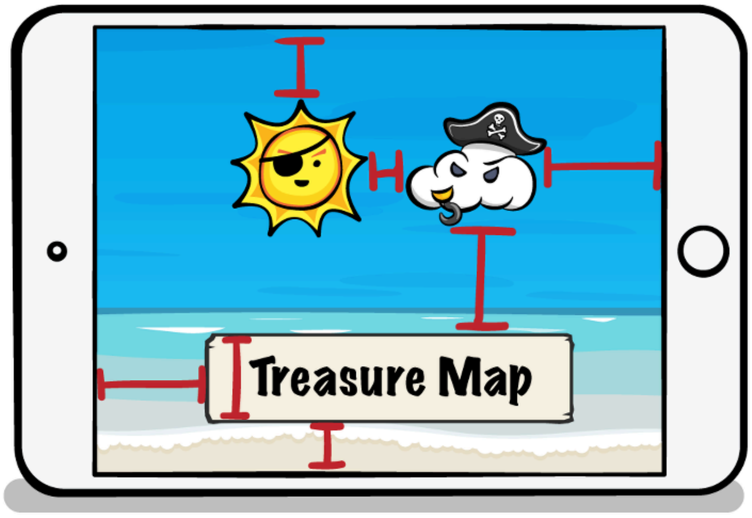
MASTERING AUTO LAYOUT



HANDS-ON CHALLENGES

Mastering Auto Layout

Jerry Beers

Challenge #19: Adaptive Presentation

By Jerry Beers

When you present a view in a popover, like we do for the dictionary view in the demo, the system does the work of deciding how that view should actually be presented. In a horizontally compact environment, it will present the view as a full screen modal. In a horizontally normal, but vertically compact environment (like the iPhone 7 Plus in landscape) it will display as `formSheet`, like a modal, but with translucent space on the sides. For this challenge, you're going to override that default behavior and make the popover display as a popover in all environments. Then you'll see how to limit it to only certain environments.

Prepare MenuViewController

First, there's some code you need to remove from `MenuViewController`. The `prepare(for:sender:)` method sets the delegate for the popover controller. You're going to set that a different way, so you can remove that whole method. The private extension of `String` at the top was only used by this method, so you can remove it too.

Controlling the Adaptive Style

In `awakeFromNib` in the `DictionaryViewController`, you're going to add the code to assign the delegate. Add these lines right below the call to `super`:

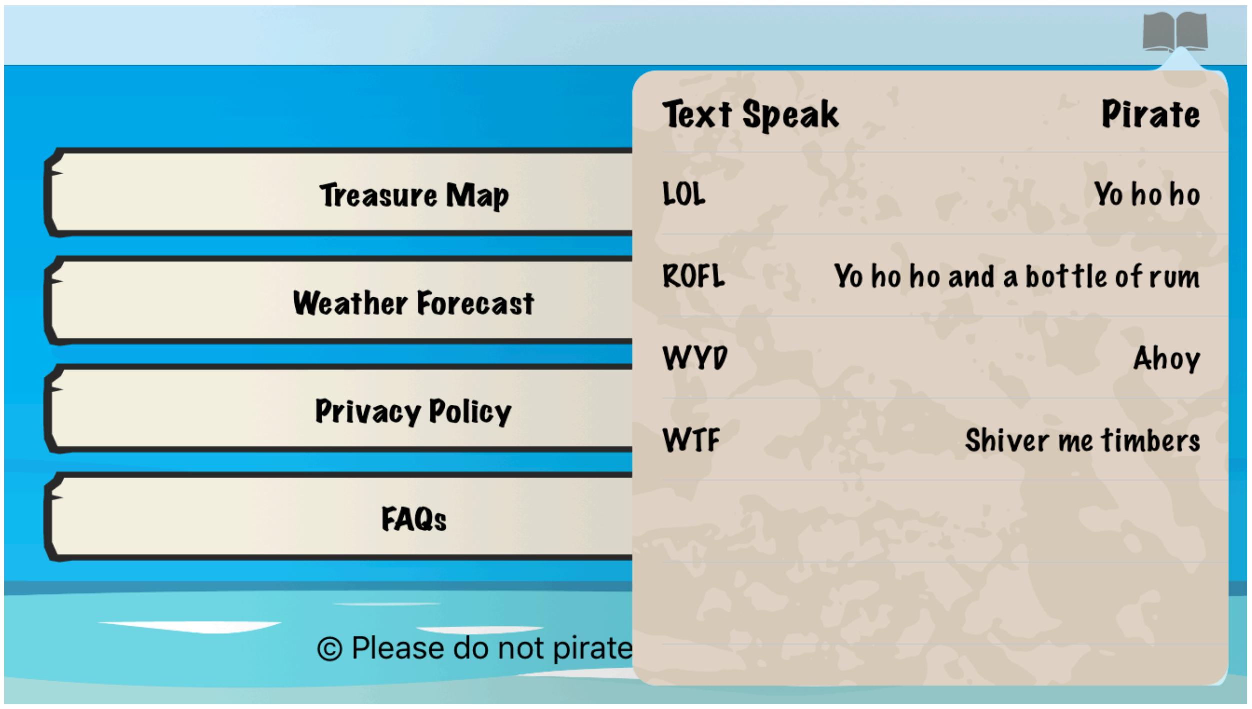
```
modalPresentationStyle = .popover
popoverPresentationController?.delegate = self
```

The order of these is important. As you can see the `popoverPresentationController` property is optional. If you don't set the `modalPresentationStyle` to `.popover`, it will be nil and you won't be able to assign the delegate. Now with that out of the way, let's control the adaptive style. Add this method to the extension, at the bottom:

```
func adaptivePresentationStyle(for controller:
    UIPresentationController, traitCollection:
    UITraitCollection) -> UIModalPresentationStyle {
  return .none
}
```

This code is simple, but what it does is override the default behavior. It basically says that there should be *no* adaptive style for this view controller. So in every case, it will be presented as a popover and not adapt to the environment.

If you build and run, no matter the device or orientation, the dictionary should now present as a popover.



Text Speak	Pirate
LOL	Yo ho ho
ROFL	Yo ho ho and a bottle of rum
WYD	Ahoy
WTF	Shiver me timbers

Limiting the Environment

Now that you have control over that presentation, let's add our own limits to the popover presentation. If we only want it to present as a popover when the vertical size class is regular, but go back to using the modal when the vertical size class is compact, we'd add logic to the `adaptivePresentationStyle` method to accomplish this. Replace the code in that method with the following:

```
if traitCollection.verticalSizeClass == .regular {
  return .none
} else {
  return .fullScreen
}
```

Remember, in this case `.none` means, "don't adapt the presentation, just use popover". It's important to note that although the `UIModalPresentationStyle` has several choices, the only ones that are valid to return from this method are fullScreen, overFullScreen, formSheet, or none.