

[MapKit](#) / [...](#) / [MapKit annotations](#) / Annotating a Map with Custom Data

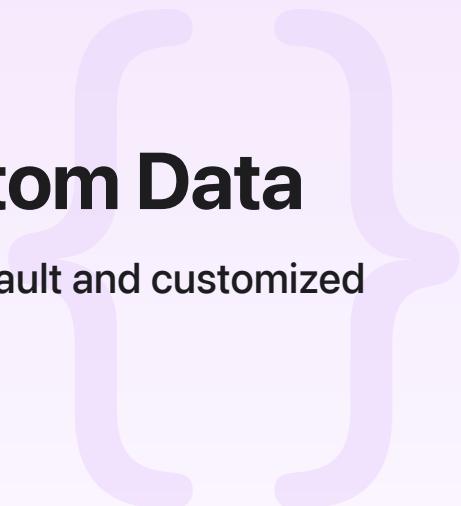
Sample Code

Annotating a Map with Custom Data

Annotate a map with location-specific data using default and customized annotation views and callouts.

[Download](#)

iOS 18.0+ | iPadOS 18.0+ | Xcode 16.1+



Overview

This sample code project demonstrates how to display a map with custom annotations, each with a customized callout provided by its [MKAnnotationView](#). In addition, the Map Callouts sample project shows how you can extend annotations with custom views, strings, and callout accessory views using [MKAnnotation](#) and [MKAnnotationView](#).

Provide an Annotation

Annotations offer a way to highlight specific coordinates on a map and provide additional information. You can use annotations to call out addresses, points of interest, and particular destinations.

To display an annotation on a map, your app must provide two distinct objects: an annotation object and an annotation view.

An annotation object conforms to the [MKAnnotation](#) protocol and manages the data for the annotation, such as the [coordinate](#), [title](#), and [subtitle](#) properties as shown in this section of the sample code.

```
class SanFranciscoAnnotation: NSObject, MKAnnotation {
```

```
// This property must be key-value observable, which the `@objc dynamic` attribu
```

```

@objc dynamic var coordinate = CLLocationCoordinate2D(latitude: 37.779_379, longitude: -122.430_687)

// Required if you set the annotation view's `canShowCallout` property to `true`
var title: String? = NSLocalizedString("SAN_FRANCISCO_TITLE", comment: "SF annotation title")

// This property defined by `MKAnnotation` is not required.
var subtitle: String? = NSLocalizedString("SAN_FRANCISCO_SUBTITLE", comment: "SF annotation subtitle")
}

```

Derived from the MKAnnotationView class, an annotation view draws the visual representation of the annotation on the map surface. Register annotation views with the MKMapView so the map view can create and efficiently reuse them. Use a default annotation view if you need to customize the content with a callout, or change the default marker. Use a custom annotation view if you want to have a completely custom view appear for the annotation.

```

private func registerMapAnnotationViews() {
    mapView.register(MKMarkerAnnotationView.self, forAnnotationViewWithReuseIdentifier: MKAnnotationViewReuseIdentifier)
    mapView.register(CustomAnnotationView.self, forAnnotationViewWithReuseIdentifier: MKAnnotationViewReuseIdentifier)
    mapView.register(MKAnnotationView.self, forAnnotationViewWithReuseIdentifier: NSUUID())
    mapView.register(MKMarkerAnnotationView.self, forAnnotationViewWithReuseIdentifier: NSUUID())
}

```

When an annotation comes into view, the map view asks the [MKMapDelegate](#) to provide the appropriate annotation view.

```

func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView {
    guard !annotation.isKind(of: MKUserLocation.self) else {
        // Make a fast exit if the annotation is the `MKUserLocation`, as it's not a bridge annotation
        return nil
    }

    var annotationView: MKAnnotationView?

    if let annotation = annotation as? BridgeAnnotation {
        annotationView = setupBridgeAnnotationView(for: annotation, on: mapView)
    } else if let annotation = annotation as? CustomAnnotation {
        annotationView = setupCustomAnnotationView(for: annotation, on: mapView)
    } else if let annotation = annotation as? SanFranciscoAnnotation {
        annotationView = setupSanFranciscoAnnotationView(for: annotation, on: mapView)
    } else if let annotation = annotation as? FerryBuildingAnnotation {
        annotationView = setupFerryBuildingAnnotationView(for: annotation, on: mapView)
    }
}

```

```
    }

    return annotationView
}
```

Before returning from the delegate call providing the annotation view, you configure the annotation view with any customizations required for the annotation. For example, use a flag icon for an annotation view representing San Francisco.

```
private func setupSanFranciscoAnnotationView(for annotation: SanFranciscoAnnotation,
    let reuseIdentifier = NSStringFromClass(SanFranciscoAnnotation.self)
    let flagAnnotationView = mapView.dequeueReusableCell(withIdentifier: reuseIdentifier)

    flagAnnotationView.canShowCallout = true

    // Provide the annotation view's image.
    let image = #imageLiteral(resourceName: "flag")
    flagAnnotationView.image = image

    // Provide the left image icon for the annotation.
    flagAnnotationView.leftCalloutAccessoryView = UIImageView(image: #imageLiteral(resourceName: "flag"))

    // Offset the flag annotation so that the flag pole rests on the map coordinate.
    let offset = CGPoint(x: image.size.width / 2, y: -(image.size.height / 2) )
    flagAnnotationView.centerOffset = offset

    return flagAnnotationView
}
```

Create Callouts

A callout is a standard or custom view that can appear with an annotation view. A standard callout displays the annotation's title, and it can display additional content such as a subtitle, images, and a control.

A callout can be customized in multiple ways. To place a disclosure button inside a callout:

```
let rightButton = UIButton(type: .detailDisclosure)
markerAnnotationView.rightCalloutAccessoryView = rightButton
```

When the disclosure button is tapped, MapKit calls `mapView(_ :annotationView:calloutAccessoryControlTapped:)` for the app to handle the tap event.

Callouts can also include images, such as the `detailCalloutAccessoryView`:

```
// Provide an image view to use as the accessory view's detail view.  
markerAnnotationView.detailCalloutAccessoryView = UIImageView(image: #imageLiteral(1
```

See Also

Location annotations

`class MKPointAnnotation`

A string-based piece of location-specific data that you apply to a specific point on a map.

`class MKMapItemAnnotation`

An annotation that represents a map item

`class MKMarkerAnnotationView`

An annotation view that displays a balloon-shaped marker at the designated location.

~~`class MKPinAnnotationView`~~

An annotation view that displays a pin image on the map.

Deprecated