# Found
# Lesson 5

## Description

Add a button that places a custom `MKAnnotation` on the map.


## Learning Outcomes

- Practice adding a Toolbar and customizing a Bar Button Item with Interface Builder.
- Practice establishing connections from interface elements to controller actions.
- Implement a class that adopts a protocol.
- Describe the purpose and use of Swift protocols.
- Discover how to use Map Kit and Core Location to add annotations to a map.

## Vocabulary

| Toolbar | Bar Button Item | controller action |
|---|---|---|
| `MKAnnotation` | protocol | protocol adoption |
| `CLLocationCoordinate2D` | property | initializer |

## Materials

- **Found Lesson 5** Xcode project
- **Protocols** presentation

# Opening

How can we add a pin to the map?

# Agenda

- Using Interface Builder and the Object Library (⌥⌘L), add a Toolbar to the bottom of the view.
- Add layout constraints for the toolbar leading, trailing, and bottom space by Control-dragging from the canvas to the main View in the Document Outline ().
- Adjust the bottom edge of the map view to match the top edge of the toolbar, and update the map view constraints with the menu item *Editor* > *Resolve Auto Layout Issues* > *Update Constraints* (⇧⌘=).
- Click on the default Item button and use the Attributes Inspector (⌥⌘4) to change the Identifier attribute to **Add**. Observe how the button appearance changes.
- Run the app (⌘R), and observe that the toolbar appears at the bottom of the screen.
- Using Interface Builder and the Assistant Editor (⌥⌘↩), Control-drag from the Add button in the Document Outline () to the `ViewController` class definition to create an action called `dropPin:`.

```
@IBAction func dropPin(sender: UIBarButtonItem) {
   // drop pin at current location
}
```

- Explain how Map Kit uses the concept of an `MKAnnotation` to represent markers on a map.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `MKAnnotation` protocol reference.
- Discuss how there is no `MKAnnotation` class, only a protocol, and how one will need to create a class for an annotation, adopting the `MKAnnotation` protocol.
- Present the concept of Swift protocols.
- Add a new `Pin` class (⌘N) to the project.

```
class Pin {

}
```

- Add an import statement for Map Kit above the `Pin` class definition, and update the class definition to extend `NSObject` and adopt the `MKAnnotation` protocol.

```
import MapKit

class Pin: NSObject, MKAnnotation {
```

- Using the Xcode Documentation and API Reference (⇧⌘0), review the requirements of the `MKAnnotation` protocol, drawing attention to the requirement of having a `CLLocationCoordinate2D` property.
- Add a `CLLocationCoordinate2D` property and an initializer to the `Pin` class.

```
class Pin: NSObject, MKAnnotation {

   let coordinate: CLLocationCoordinate2D

   init(coordinate: CLLocationCoordinate2D) {
      self.coordinate = coordinate
   }

}
```

- Explain how `CLLocationCoordinate2D` is a struct, and not an Swift object type.
- Update the implementation of the `dropPin:` method in the controller class.

```
@IBAction func dropPin(sender: UIBarButtonItem) {
   let pin = Pin(coordinate: mapView.userLocation.coordinate)
   mapView.addAnnotation(pin)
}
```

- Run the app (⌘R), observe the current location beacon, tap the Add button, and observe the annotation appear.

# Closing

How would you enable the user to add a label to the pin? How would you handle cases when the user location is not available?

# Modifications and Extensions

- Investigate the additional `MKAnnotation` protocol properties, and display an alert view with text field to enable the user to add a name to the pin when tapping the Add button.
- Store each `MKAnnotation` in an array, and add a new toolbar button that enables the user to clear all of the existing map annotations.
- Enable the user to tap on a pin, and share that location with other people.

# Resources

UIKit User Interface Catalog: Toolbar https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/UIToolbar.html

Interface Builder Help: Adding an Object to Your Interface https://developer.apple.com/library/ios/recipes/xcode_help-IB_objects_media/Chapters/AddingObject.html

Interface Builder Help: Creating an Action Connection https://developer.apple.com/library/ios/recipes/xcode_help-IB_connections/chapters/CreatingAction.html

Interface Builder Help: Configuring Object Attributes https://developer.apple.com/library/ios/recipes/xcode_help-IB_objects_media/Chapters/ObjectAttributes.html

MKAnnotation Protocol Reference https://developer.apple.com/library/ios/documentation/MapKit/Reference/MKAnnotation_Protocol/index.html

Cocoa Core Competencies: Protocol http://developer.apple.com/library/ios/documentation/general/conceptual/DevPedia-CocoaCore/Protocol.html

The Swift programming Language: Protocols https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Protocols.html

The Swift Programming Language: Classes and Structures https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/ClassesAndStructures.html

Core Location Data Types Reference https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocationDataTypesRef/index.html