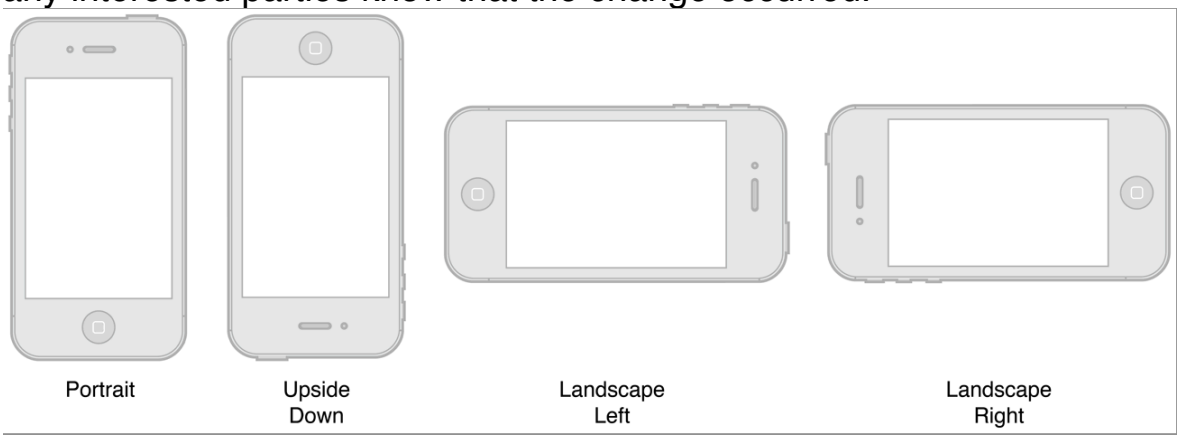
Supporting Multiple Interface Orientations

1. When the orientation of an iOS-based device changes, the system sends out a UIDeviceOrientationDidChangeNotification notification to let any interested parties know that the change occurred.



Controlling What Interface Orientations Are Supported (iOS 6)

1. When UIKit receives an orientation notification, it uses the UIApplication object and the root view controller to determine if the new orientation is allowed.

> Declaring a View Controller's Supported Interface Orientations
1. A root view controller or a view controller is presented full screen on the main window can declare what orientation it supports. (supportedInterfaceOrientations)

2. Implementing the supportedInterfaceOrientations method

```
- (NSUInteger)supportedInterfaceOrientations
{
    return UIInterfaceOrientationMaskPortrait |
UIInterfaceOrientationMaskLandscapeLeft;
}
```

> Dynamically Controlling Whether Rotation Occurs
If you want to temporarily disable automatic rotation, avoid manipulating the orientation masks to do this. Instead, override the shouldAutorotate method on the topmost view controller.

> Declaring a Preferred Presentation Orientation
1. If the view controller supports multiple orientations but appears better

in a different orientation, you can provide a prefer orientation by overriding preferredInterfaceOrientationForPresentation method.

2. Implementing the preferredInterfaceOrientationForPresentation method

```
- (UIInterfaceOrientation)preferredInterfaceOrientationForPresentation
{
    return UIInterfaceOrientationLandscapeLeft;
}
```

> Declaring the App's Supported Interface Orientations

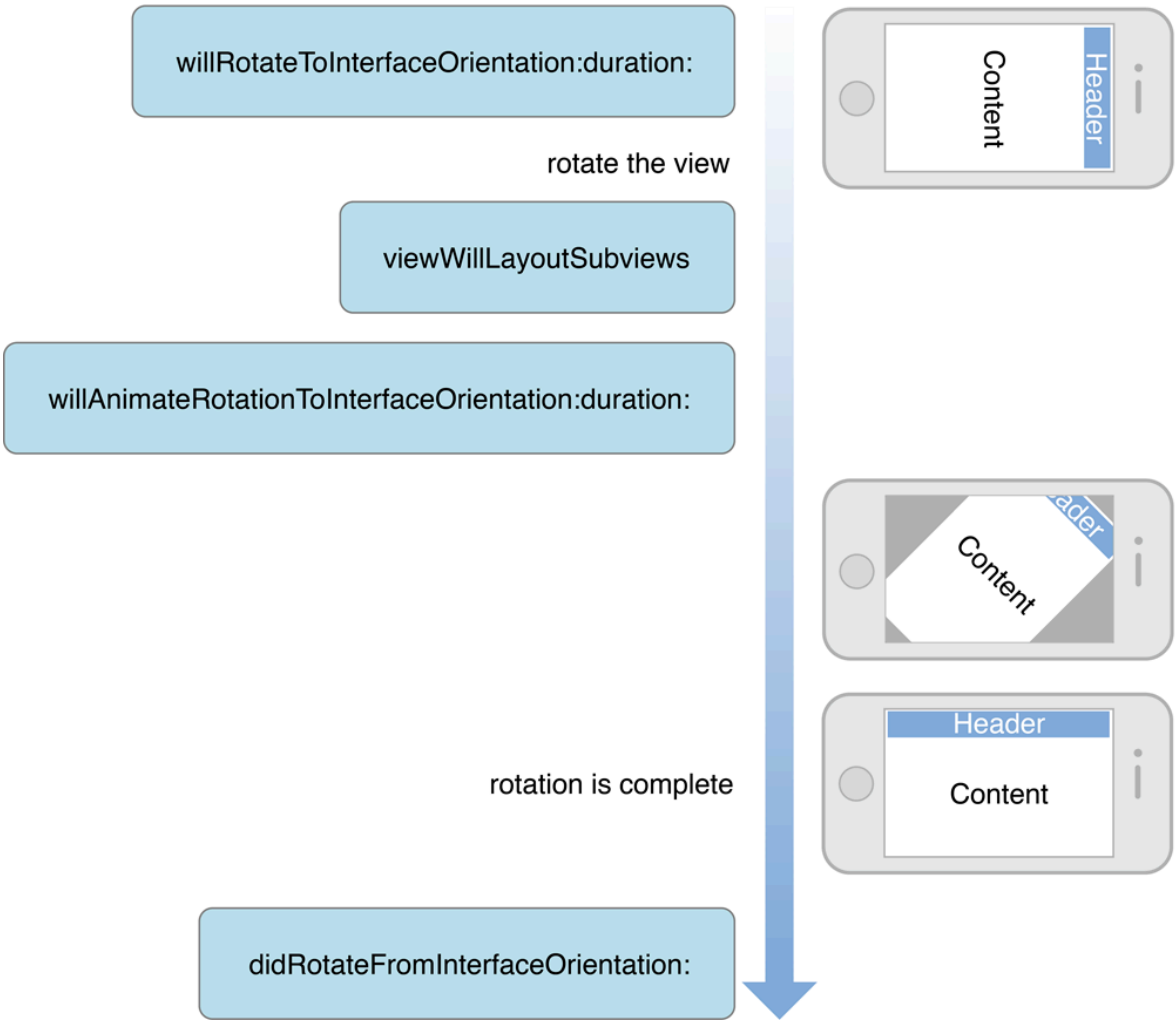
1. The easiest way to set an app's app's supported interface orientations is to edit the project's Info.plist file.

2. If you restrict your app's supported orientations, then those restrictions apply globally to all of the app's view controllers.

Responding to Orientation Changes in a Visible View Controller

1. The rotation methods are sent to the root view controller. The root view controller passes these events on as necessary to its children, and so on down the view controller hierarchy.

2. Processing an interface rotation



Rotations May Occur When Your View Controller Is Hidden
If your view controller's contents are not onscreen when a rotation occurs, then it does not see the list of rotation messages.

Creating an Alternate Landscape Interface
1. If you want to present same data differently based on whether a device is in a landscape or portrait orientation, the way to do so is using two separate view controllers.

2. To support an alternate landscape interface, you must do the following:
=> Implement two view controller objects. One to present a portrait-only interface, and the other to present a landscape-only interface.
=> Register for the UIDeviceOrientationDidChangeNotification notification. In your handler method, present or dismiss the alternate view controller based on the current device orientation.

3. resenting the landscape view controller

```
@implementation PortraitViewController
- (void)awakeFromNib
{
    isShowingLandscapeView = NO;
    [[UIDevice currentDevice]
beginGeneratingDeviceOrientationNotifications];
    [[NSNotificationCenter defaultCenter] addObserver:self
                                selector:@selector(orientationChanged:)
                                name:UIDeviceOrientationDidChangeNotification
                                object:nil];
}

- (void)orientationChanged:(NSNotification *)notification
{
    UIDeviceOrientation deviceOrientation = [UIDevice
currentDevice].orientation;
    if (UIDeviceOrientationIsLandscape(deviceOrientation) &&
        !isShowingLandscapeView)
    {
        [self performSegueWithIdentifier:@"DisplayAlternateView"
sender:self];
        isShowingLandscapeView = YES;
    }
    else if (UIDeviceOrientationIsPortrait(deviceOrientation) &&
             isShowingLandscapeView)
    {
        [self dismissViewControllerAnimated:YES completion:nil];
        isShowingLandscapeView = NO;
    }
}
```

Tips for Implementing Your Rotation Code
=> Disable event delivery temporarily during rotations.
=> Store the visible map region.
=> For complex view hierarchies, replace your views with a snapshot image.
=> Reload the contents of any visible tables after a rotation.
=> Use rotation notifications to update your app’s state information.