## Error Handling Intro

Now let's discuss how to handle error conditions in our application. In Swift 1.2 there's really no formal error handling - it's very primitive. What we're going to do is basically wrap potential failure points in conditionals.

## Handling Errors Conditions

The first thing to do is to identify some areas that might fail. Our application is very simple, and we really restrict the user input. One way to handle errors is to try and limit the areas that might fail.

For example, you can limit the user input. User can add something only on the "Add Activity" tab. We can make sure they've actually typed something in. So let's add a conditional:

```
if(activityLabel.text == ""){
  var alert = UIAlertView(title: "Empty Activity", message: "Please type in a valid activity", del
  alert.show()

} else {
  ActivityManager.activities.append(Activity(activityName: activityLabel.text, totalTime: 0))
  activityLabel.text = ""
  activityLabel.resignFirstResponder()
}
```

## Displaying an Alert Message

In this conditional we are displaying a message by using `UIAlertView` which pops up the message and. The user clicks "OK" and the message goes away. However they can't continue until they fix this issue with the empty string.

Now let's look at performance view controller. This is really always going to work right with this array. However what if the index was greater than the number of items inside of the array? That would throw a runtime error. That's not really going to happen here, but let's discuss how you might handle this scenario if you are doing something more complex.

Suppose you are passing in your own index, which could be a failure point. So you want to check, whether the index is within the range of the array. To do that, I'm gonna add another conditional:

```
if(indexPath.item <= ActivityManager.activities.count) {
    var acitivity = ActivityManager.activities[indexPath.item]
    cell.textLabel!.text = "\(activity.activityName) \(activity.totalTime)"
}
```

So to summarize on how you might do some error handling in Swift 1.2, it’s basically about wrapping potential failure points inside of a conditional. You identify areas in your app that can fail and then put the conditional around them.