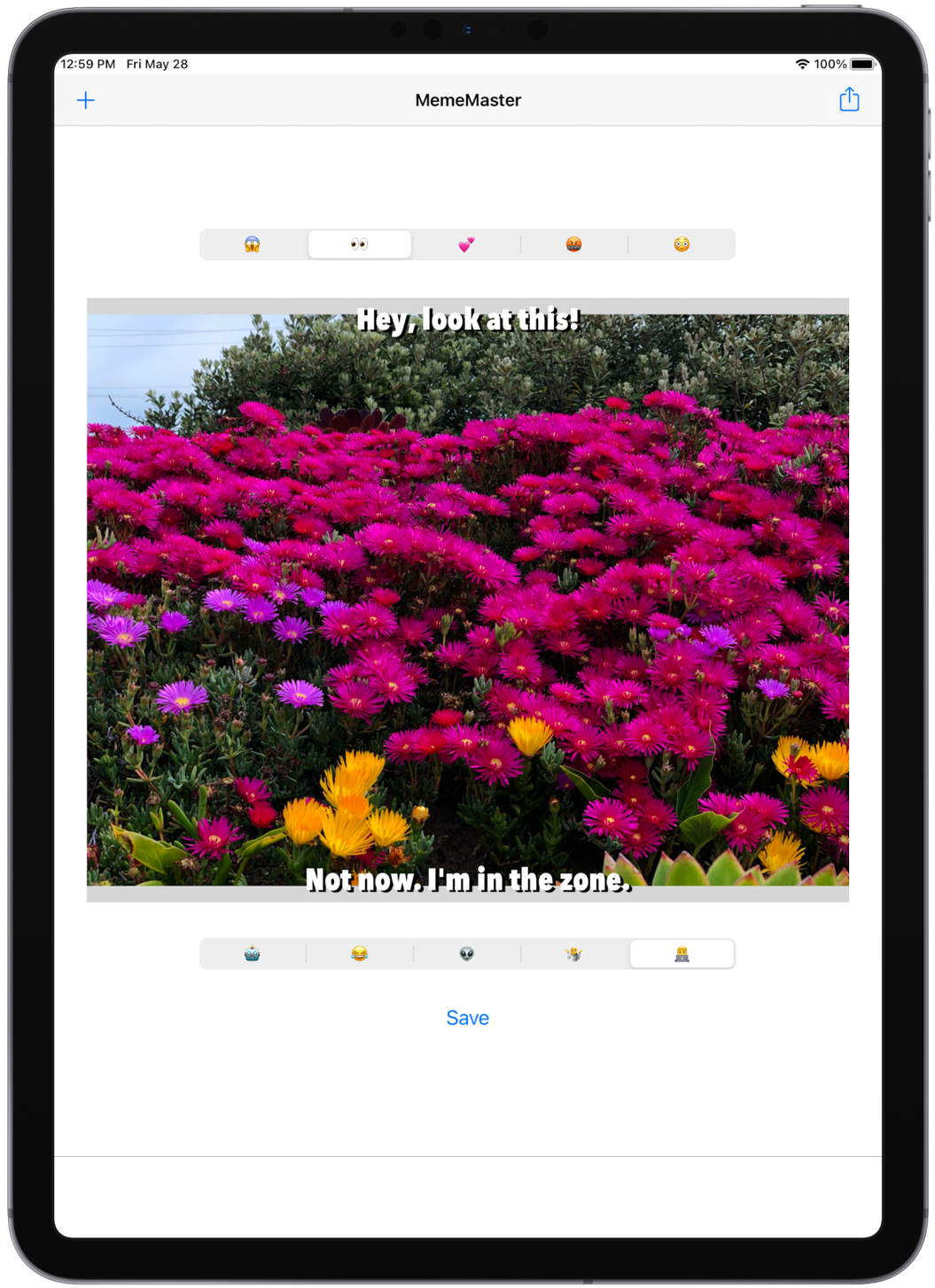
Brian Foutty aka The Swift Teacher

MemeMaster

A Little Meme Fun

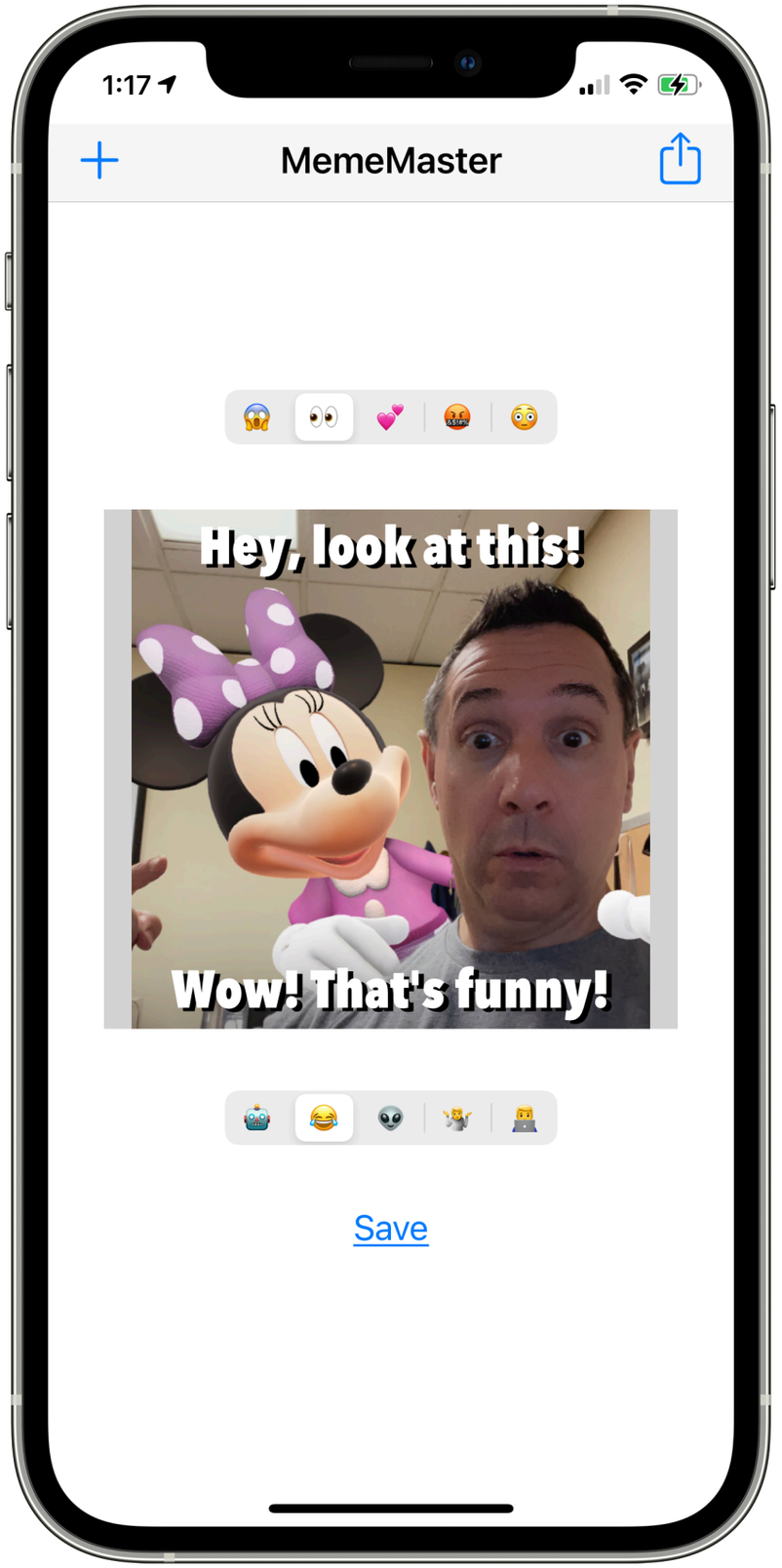


Lorem

MemeMaster app project

Activate, Explore, Apply

A single-view app where the user can choose a photo from their Photo Library and apply one of five phrases for the top and bottom of the photo. This app is an extension of the MemeMaker app created as part of Apple's **Develop in Swift Explorations** Teacher and Student course. The project can be found in Unit 4 starting on page 384 in the Student book and starting on page 589 in the Teacher Guide.



The general process and flow of this project is in the format of an Apple Teacher Portfolio lesson. There are three parts/phases to the lesson:

- Activate
- Explore
- Apply.

Activate

We want our students to activate any prior knowledge on the topic. Since the students have previously made the MemeMaker app, they will all have the starting point of the project. However, the goal of the project is to greatly improve the app and enable the user to use their own photos create a meme. Here is the activity I use with my students:

1. Take a selfie.
2. Open Keynote and pick a basic white or basic black project.
3. Insert your selfie into the slide, adjust the size of the picture so that it covers the entire slide.
4. You can adjust the transparency level of the picture to mimic a filter feature.
5. Add a text box to the top and bottom of your slide.
6. Add phrases of your choosing to your textbooks.
7. Think, pair, share with a partner about how easy it was to make your meme and how we could make the process of making the meme easier.

Explore

Students will now create a prototype of a meme making app.

1. Open Keynote.
2. We are going to prototype a meme making app.
3. Set the slide background color to be the same color you would have for a meme making app that you would build.
4. Search shapes for phone. Drag the iPhone onto the screen and resize it so that it is the full height of the slide.
5. Use elements in the Shape section and text boxes to create your meme app prototype. Be sure to include your meme from the Activate lesson as the example meme for your prototype.

Apply

In this part of the project we build the app.

MemeMaster GitHub Repo

1. Open existing MemeMaker app
2. Click Source **Control > New Git Repositories** to place it under version control

Steps 2 - 17 screencast

3. Create new branch called “**add-photo**”
4. Select segmented controllers and image view and then click **Editor > Resolve Auto Layout Issues > Add Missing Constraints**
5. Remove the image from the drop down menu in the **Attributes Inspector** for the **imageView**
6. Select **imageView** and add a **Background Color** in the **Attributes Inspector**
7. Open the **Object Library**. Add a **Navigation Bar** to the view.
8. Open the **Object Library**. Add a **Bar Button Item** to the top left (or top right if you want to have two side-by-side). In the **Attribute Inspector** set the **System Item** to be **Add**.
9. Open the **Assistant Editor**.
10. Create an outlet from the **imageView** and name it `imageView`.
11. Create an action from the **Add** button and name it `importPicture()`.

```
// #11
@IBAction func importPicture(_ sender: Any) {

}
```

12. Add this code to `importPicture()` - the code will generate an error. We will fix it in the next step.

1. Add this code to `importPicture()` - the code will generate an error. We will fix it in the next step.

```
let picker = UIImagePickerController()
picker.allowsEditing = true
picker.delegate = self
present(picker, animated: true)
```

13. Add the following to the ViewController class declarations: `UIImagePickerControllerDelegate`, `UINavigationControllerDelegate`.

```
class ViewController: UIViewController, UIImagePickerControllerDelegate,
    UINavigationControllerDelegate {
```

14. Next we need to create a method to that will take the photo we select from our photo library and add it to the imageView. It is a built in Apple method (API) that we use to to pick photos or videos from the Photo Library. Start typing `didFinishPickingMediaWithInfo` and use autocomplete to get the entire method. The entire method looks like this:

```
// method to add selected photo from photo library to imageView
// #14
func imagePickerController(_ picker: UIImagePickerController,
    didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {

}
```

15. Then add this code to your imagePickerController method:

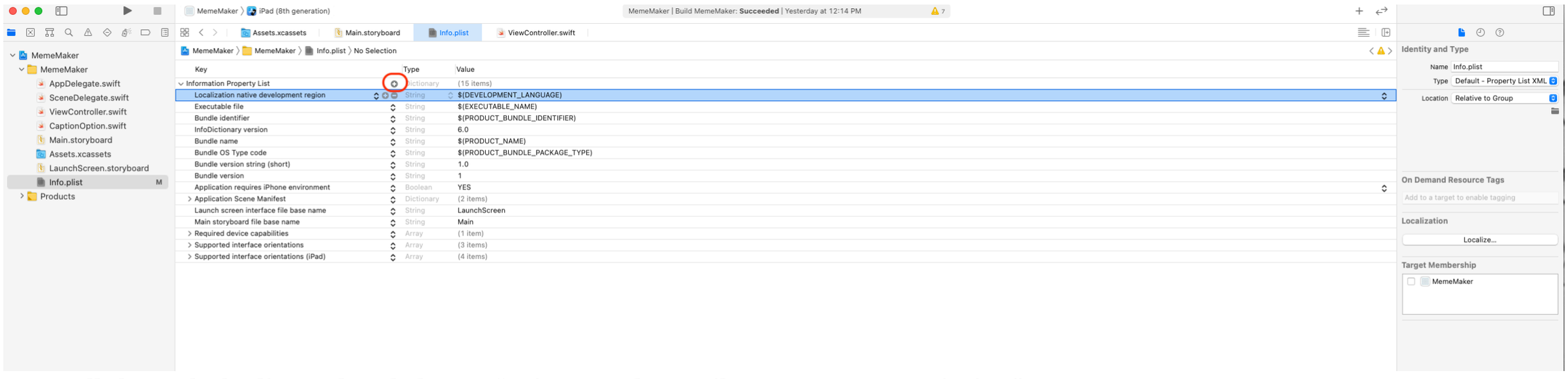
```
// #15
guard let image = info[.editedImage] as? UIImage else { return }

imageView.image = image
dismiss(animated: true)
```

16. Build and run. Tap the “+” button in the Navigation Bar and choose a photo and add it. Change your meme phrases.

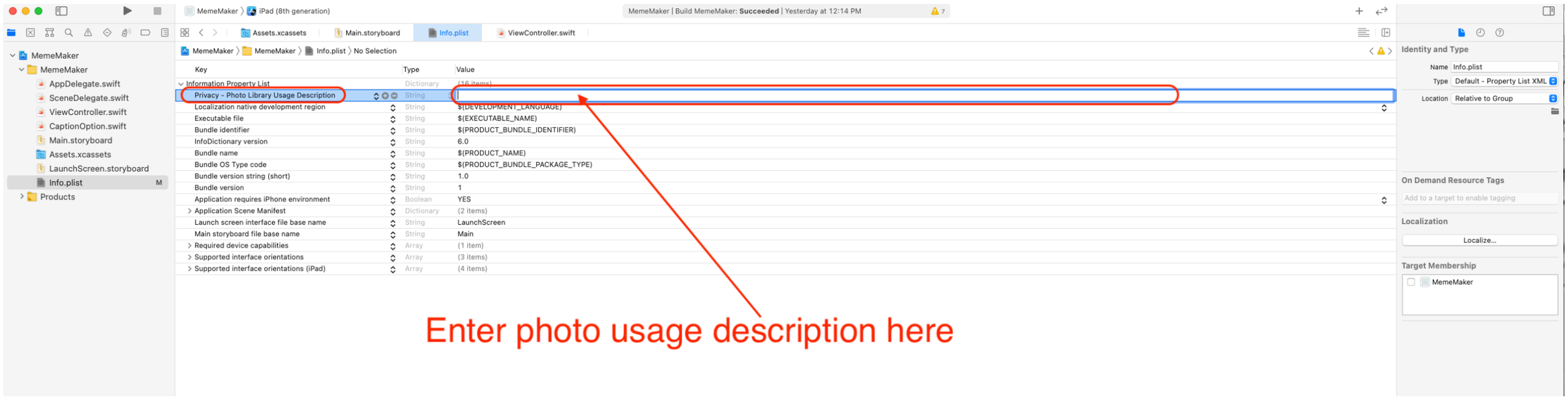
17. If everything works as expected:
    1. Commit the changes and create the remote branch in the process
    2. Right-click the "main" branch and click the "**Checkout**" button.
    3. Right-click on the "**add-photo**" branch and choose "**Merge** "add-photo" into "main". Then click the "**Merge**" button.
    4. In the Menu bar, click **Source Control > Push**. This will send the changes just merged into the main branch to the main branch on GitHub or your chosen repository hosting service.

Steps 18 - 36 screencast

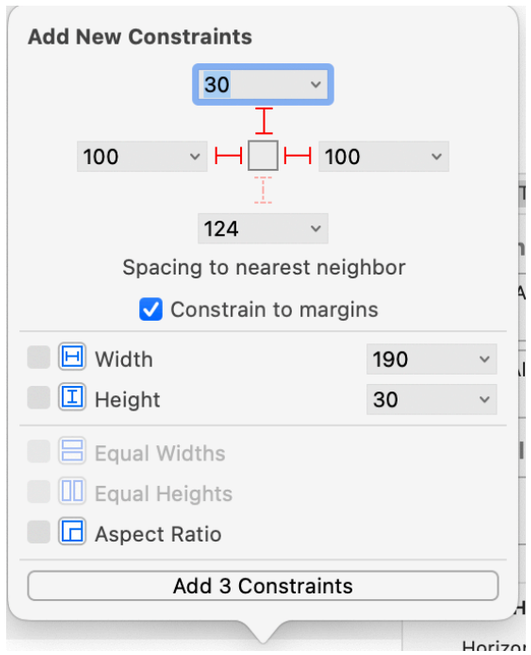
18. Next we need to add the functionality to save the meme to the user's Photo Library. This is actually a somewhat involved process as we have to take the image and the text and combine them into a new image.
19. Go to **Source Control** and create new branch called "save-meme"
20. We need to ask for access to photos to save the meme to the photo library. This is done in the info.plist file. At the top of the list (on the same line that says Information Property List) click the "+" button.



21. Scroll through the list and and choose "Privacy - Photo Library Usage Description".
22. Click in the "**Value**" column and enter something like this: Please allow access to your photos so that you can save your meme or Access to your photos is needed to save your meme. Either phrase will work (see screenshot on next page for 21 & 21).



23. Open **Main.storyboard** and and a button to the bottom center under the bottom captions segmented controller.
24. Click on the button. Name the button "**Save**" and set the font to size **20**.
25. Click on the **Add New Constraints** button in the bottom right and pin the location to set the constraints as shown.



26. Let's give ourselves more meme options. Add two emoji and phrases to topChoices and bottomChoices in ViewController.swift. Those choices can be the same as what you made in your app prototype.

```
// #26a
let topChoices = [
    CaptionOption(emoji: "😱", caption: "OMG!!"),
    CaptionOption(emoji: "👀", caption: "Hey, look at this!"),
    CaptionOption(emoji: "💕", caption: "You know what I love…"),
    CaptionOption(emoji: "😡", caption: "You know what makes me mad?"),
    CaptionOption(emoji: "😳", caption: "Holy crap!!💩")
]
// #26b
let bottomChoices = [
    CaptionOption(emoji: "🤖", caption: "You are heartless."),
    CaptionOption(emoji: "😂", caption: "Wow! That's funny!"),
    CaptionOption(emoji: "👽", caption: "Dumb humans!"),
    CaptionOption(emoji: "🤷‍♂️", caption: "Beats me!?!?"),
    CaptionOption(emoji: "👨‍💻", caption: "Not now. I'm in the zone.")
]
```

27. At the very top of ViewController.swift file above the import UIKit statement, add this statement: `import CoreImage`.

```
8  import UIKit
9  import CoreImage // #27
```

28. Create a `currentImage` property towards the top of the class right under the `bottomChoices` array. Declare it to be of type `UIImage!`.

```
// #28
var currentImage: UIImage!
```

29. Create properties for the size and type of font to be rendered. We will use a font size of **70** and font type **Avenir Next Heavy**:

```
// #29
var fontSize: Double = 70.0
var selectedFont = "Avenir Next Heavy"
```

30. We are going to write a lengthy function that will do four things:
    1. Send the user an error message if they did not pick an image to add the text to create a meme.
    2. Get the size of the chosen image.
    3. Add the text to the chosen image.
    4. Call the Objective-C function that will save the meme to the Photo Library.

31. Write the `saveImageAndText()` method:

```
// method to get size of photo, add the text to the photo, and then call the function to
    save the image to Photo Library
// #31
func saveImageAndText() {
    if imageView.image == nil {
        let alert = UIAlertController(title: "You did not pick an image", message: "Please
            pick an image and try again", preferredStyle: .alert)
        alert.addAction(UIAlertAction(title: "OK", style: .cancel, handler: nil))
        present(alert, animated: true)
    }

    // get size of image from chosen image
    guard let width = imageView.image?.size.width else { return }
    guard let height = imageView.image?.size.height else { return }

    let renderer = UIGraphicsImageRenderer(size: CGSize(width: width, height: height))
    let pic = renderer.image { context in

        guard let image = imageView.image else { return }
        image.draw(at: CGPoint(x: 0, y: 0))

        let paragraphStyle = NSMutableParagraphStyle()
        paragraphStyle.alignment = .center

        let strokeAttributes: [NSAttributedString.Key : Any] = [
            .strokeColor : UIColor.black,
            .foregroundColor : UIColor.white,
            .strokeWidth : -2.0,
            .paragraphStyle : paragraphStyle,
            .font : UIFont(name: selectedFont, size: CGFloat(fontSize)) ??
                UIFont.systemFont(ofSize: 50)
        ]
```

```
        let topAttributedString = NSAttributedString(string: topCaptionLabel.text!,
            attributes: strokeAttributes)
        // start with (with: CGRect(x: 0, y: 0,...
        topAttributedString.draw(with: CGRect(x: 0, y: 5, width: width, height: height /
            4), options: .usesLineFragmentOrigin, context: nil)

        let bottomAttributedString = NSAttributedString(string: bottomCaptionLabel.text!,
            attributes: strokeAttributes)

        // use 100 for single line and 120 for 2 lines
        // start with (with: CGRect(x: 0, y: 0,...
        bottomAttributedString.draw(with: CGRect(x: 0, y: height - CGFloat(fontSize +
            120), width: width, height: height /  4), options: .usesLineFragmentOrigin,
            context: nil)

    }

    UIImageWriteToSavedPhotosAlbum(pic, self,
        #selector(imageSave(_:didFinishSavingWithError:contextInfo:)), nil)

    currentImage = pic
}
```

32. Now we are going to write the Objective-C function called `imageSave` that will save the image to the Photo Library by bridging to Objective-C.

```
// method to save image to Photo Library
// #32
@objc func imageSave(_ image: UIImage, didFinishSavingWithError error: Error?,
    contextInfo: UnsafeRawPointer) {
    if let error = error {
        // if we get back an error
        let alert = UIAlertController(title: "Save error", message:
            error.localizedDescription, preferredStyle: .alert)
        alert.addAction(UIAlertAction(title: "OK", style: .default))
        present(alert, animated: true)
    } else {
        let alert = UIAlertController(title: "Saved!", message: "Your meme has been saved
            to your photos.", preferredStyle: .alert)
        alert.addAction(UIAlertAction(title: "OK", style: .default))
        present(alert, animated: true)
    }
}
```

33.Call `imageSave` inside of `saveImageAndText()`.

```
UIImageWriteToSavedPhotosAlbum(pic, self,
    #selector(imageSave(_:didFinishSavingWithError:contextInfo:)), nil) // #33
```

34. Create an action from from the **Save** button. Name it `saveMeme`.

```
@IBAction func saveMeme(_ sender: Any) {
    saveImageAndText()
}
```

35. Call `saveImageAndText` inside of `saveMeme`.

36. Build and Run. Choose an image. Create a meme and save it the Photo Library.

Steps 37 - 45 screencast

37. Open the **Object Library**. Add a **Bar Button Item** to the top right. In the **Attributes Inspector** set the **System Item** to be **Action**.
38. Create an outlet from the **Action** (+) button in the Navigation Bar and name it `shareButton`.

```
23
    @IBOutlet weak var shareButton: UIBarButtonItem!
```

39. Create an action from the **Action** (+) button in the Navigation Bar and name it `shareMeme`.

```
@IBAction func shareMeme(_ sender: Any) {

}
```

40. We are going to write a function that will give us access to the iOS system share sheet. Create a function called `shareTapped` that takes no parameters.

```
func shareTapped() {

}
```

41. Add this code to `shareTapped()`:

```
    saveImageAndText() // leave this out of initial share and use
        imageView.image? instead of currentImage
    guard let image = currentImage.jpegData(compressionQuality: 0.8) else {
        return }
    let view = UIActivityViewController(activityItems: [image],
        applicationActivities: [])
    view.popoverPresentationController?.barButtonItem = shareButton
    present(view, animated: true)
}
```

42. Call `shareTapped` inside of `shareMeme`.

```
@IBAction func shareMeme(_ sender: Any) {

    shareTapped()
}
```

43. Click on the **Navigation Bar** and change the **Title** to **MemeMaster**.

44. Build and run. Create a meme. Tap the **share** button to determine the share button functions as it should.

45. If you are happy with the updated version of the Skills screen:
    1. Commit the changes to the "**save-meme**" branch.
    2. Right-click the "**main**" branch and choose "**Checkout**". Click the "**Checkout**" button.
    3. Right-click on the "**save-meme**" branch and choose "**Merge** "save-meme" into "main"". Then click the "**Merge**" button.
    4. In the Menu bar, click **Source Control > Push**. This will send the changes just merged into the main branch to the main branch on GitHub or your chosen repository hosting service.

46. This updated version of the app is now complete. Congratulations!!

MemeMaster-Tutorial GitHub Repo