QR Coder static library

QRCoder is a simple code reader (initially only QR Code) for iOS in Swift. It is based on the `AVFoundation` framework from Apple.

It provides a default view controller to display the camera view with the scan area overlay and it also provides methods to decode a provided image to string.

It also provides methods to create QR codes image for a provided JSON, string or data.

Requirements

iOS 13.0+

Xcode 11.0+

Swift 5.0+

Usage

In iOS13+, you will need first to reasoning about the camera use. For that you'll need to add the **Privacy - Camera Usage Description** *(NSCameraUsageDescription)* field in your Info.plist

REVISION HISTORY

DATE	VERSION	AUTHOR	UPDATE DESCRIPTION
Aug 8, 2020	1.0	Gautham Velappan	Initial Draft
Aug 17, 2020	1.1	Gautham Velappan	Scanner View updates and bug fixes
Aug 25, 2020	1.2	Gautham Velappan	Compression feature for input data

QRCodeDecoder

public class QRCodeDecoder
A collection of helper functions for extracting details from a QR code image.

- decode(image:detectorAccuracy:completion:)
  Generates decoded messages for a given image(QR Code) along with additional optional properties.

Declaration
SWIFT
```
public func decode(image: UIImage,
                   detectorAccuracy: DetectorAccuracy = .high,
                   completion: ((_ message:[String]?, _ details:
[String: Any?]]?, _error: Error?) -> Void))
```

Parameters

image	UIImage for which decoded message is generated
detectorAccuracy	The detection accuracy for decoding. The default value is .high.
completion	A closure of type ([String], [[String: Any?]]?, Error) to be executed once the request has finished. This will provide the decoded message for the given image

discussion

- **mesage**: Returns the receiver's QR Code decoded into a human-readable string.
- **isBase64Encoded**: Indicates if the data is base64 encoded
- **rawMessage**: The raw string that comprise the QR code symbol.
- **errorCorrection**: The error correction level of the QR code.
- **errorCorrectedPayload**: The error-corrected codewords that comprise the QR code symbol.
- **symbolVersion**: The version property corresponds to the size of the QR Code.

DetectorAccuracy

public enum DetectorAccuracy
The key in the options dictionary used to specify an accuracy / performance tradeoff to be used.

There is a performance / accuracy tradeoff to be made. The default value will work well for most situations but using these the detector will favor performance over accuracy or accuracy over performance.

case low
Lower accuracy, higher performance

case high
Lower performance, higher accuracy

QRCodeEncoder

public class QRCodeEncoder
A collection of helper functions for generating a QR code images.

- encode(for:size:color:backgroundColor:errorCorrection:completion:)
  Generates a QR code image for a given dictionary along with additional optional properties.

Declaration
SWIFT
```
public func encode(for dictionary: [String: Any],
                   size: CGSize? = nil,
                   color: UIColor = .black,
                   backgroundColor: UIColor = .white,
                   errorCorrection: ErrorCorrection = .low,
                   compressData: Bool? = nil,
                   completion: ((UIImage?, Error?) -> Void))
```

Parameters

dictionary	Dictionary for which QRCode is generated
size	Size of the output image. Defaults to `nil`, implies no scaling
color	Foreground color of the output. Defaults to `.black`
backgroundColor	Background color of the output. Defaults to `.white`
errorCorrection	The error correction. The default value is `.low`.
compressData	The compress flag which tells if the input data should be compressed. The default value is `nil`.
completion	A closure of type (UIImage, Error) to be executed once the request has finished. This will provide the QR code for the dictionary

- encode(for:size:color:backgroundColor:errorCorrection:completion:)
  Generates a QR code image for a given string along with additional optional properties.

Declaration

SWIFT

```
public func encode(for string: String,
                   size: CGSize? = nil,
                   color: UIColor = .black,
                   backgroundColor: UIColor = .white,
                   errorCorrection: ErrorCorrection = .low,
                   compressData: Bool? = nil,
                   completion: ((UIImage?, Error?) -> Void))
```

Parameters

string	String for which QR Code is generated
size	Size of the output image. Defaults to `nil`, implies no scaling
color	Foreground color of the output. Defaults to `.black`
backgroundColor	Background color of the output. Defaults to `.white`
errorCorrection	The error correction. The default value is `.low`.
compressData	The compress flag which tells if the input data should be compressed. The default value is `nil`.
completion	A closure of type (UIImage, Error) to be executed once the request has finished. This will provide the QR code for the string

- encode(for:size:color:backgroundColor:errorCorrection:completion:)
  Generates a QR code image for a given data along with additional optional properties.

Declaration

SWIFT

```
public func encode(for data: Data,
                   size: CGSize? = nil,
                   color: UIColor = .black,
                   backgroundColor: UIColor = .white,
                   errorCorrection: ErrorCorrection = .low,
                   compressData: Bool? = nil,
                   completion: ((UIImage?, Error?) -> Void))
```

Parameters

data	Data for which QR Code is generated
size	Size of the output image. Defaults to `nil`, implies no scaling
color	Foreground color of the output. Defaults to `.black`
backgroundColor	Background color of the output. Defaults to `.white`
errorCorrection	The error correction. The default value is `.low`.
compressData	The compress flag which tells if the input data should be compressed. The default value is `nil`.
completion	A closure of type (UIImage, Error) to be executed once the request has finished. This will provide the QR code for the data

ErrorCorrection

public enum ErrorCorrection : String
The level of error correction. This controls the amount of additional data encoded in the output image to provide error correction. Higher levels result in larger output images but allow larger areas of the code to be damaged.

case low
7%

case medium
15%

case quartile
25%

case high
30%

QRCodeScannerView

```
public class QRCodeScannerView : UIView,
AVCaptureMetadataOutputObjectsDelegate
```

A custom UIView which can bring a device camera and start scanning for QR code images. The scanned QR code image also provides details about it.

- `delegate`
  Defines an interface for delegates of QRCodeScannerViewDelegate to receive emitted objects.

  Declaration
  SWIFT
  ```
  public weak var delegate: QRCodeScannerViewDelegate?
  ```

- `stopScannerAfterDecode`
  Indicates whether the scanner should stop after scanning a QR code.

  Declaration
  SWIFT
  ```
  public var stopScannerAfterDecode: Bool
  ```

- startScanner()
  Starts the Scanner instance running.

  Declaration
  SWIFT
  ```
  public func startScanner()
  ```

- stopScanner()
  Tells the Scanner to stop running.

  Declaration
  SWIFT
  ```
  public func stopScanner()
  ```

- toggleFlash(enable:)
  Toggles the illumination of the torch mode.

  Declaration
  SWIFT
  ```
  public func toggleFlash(enable: Bool? = nil)
  ```

  Parameters

enable	Boolean which indicates if the flashlight to be toggled on/off

QRCodeScannerViewDelegate

```
@objc
public protocol QRCodeScannerViewDelegate : AnyObject
```

- scannerView(_ scannerView: QRCoderScannerView, didScanQRCodeMessages messages:)
  Called whenever QR code was scanned and was successfully be able to be decoded

Declaration

SWIFT

```
func scannerView(_ scannerView: QRCodeScannerView, didScanQRCode
messages: [String])
```

- scannerView(_ scannerView: QRCoderScannerView, didScanQRCodeDetails details:)
  Called whenever QR codes were scanned successfully at frame

Parameters

scannerView	Current QR Coder Scanner view
details	Returns the receiver's QR Code decided into a detailed object.

Discussion

- **mesage**: Returns the receiver's QR Code decoded into a human-readable string.
- **isBase64Encoded**: Indicates if the data is base64 encoded
- **rawMessage**: The raw string that comprise the QR code symbol.
- **errorCorrection**: The error correction level of the QR code.
- **errorCorrectedPayload**: The error-corrected codewords that comprise the QR code symbol.
- **symbolVersion**: The version property corresponds to the size of the QR Code.
- **frame**: The bounding rectangle of the QR code

Declaration
SWIFT
```
@objc optional func scannerView(_ scannerView: QRCodeScannerView,
didScanQRCodeDetails details: [[String: Any]])
```

- scannerView(_ scannerView: QRCoderScannerView, didReceiveError error:)
Called when a device setup or a decode fails

Declaration
SWIFT
```
@objc
optional func scannerView(_ scannerView: QRCodeScannerView,
didReceiveError error: Error)
```