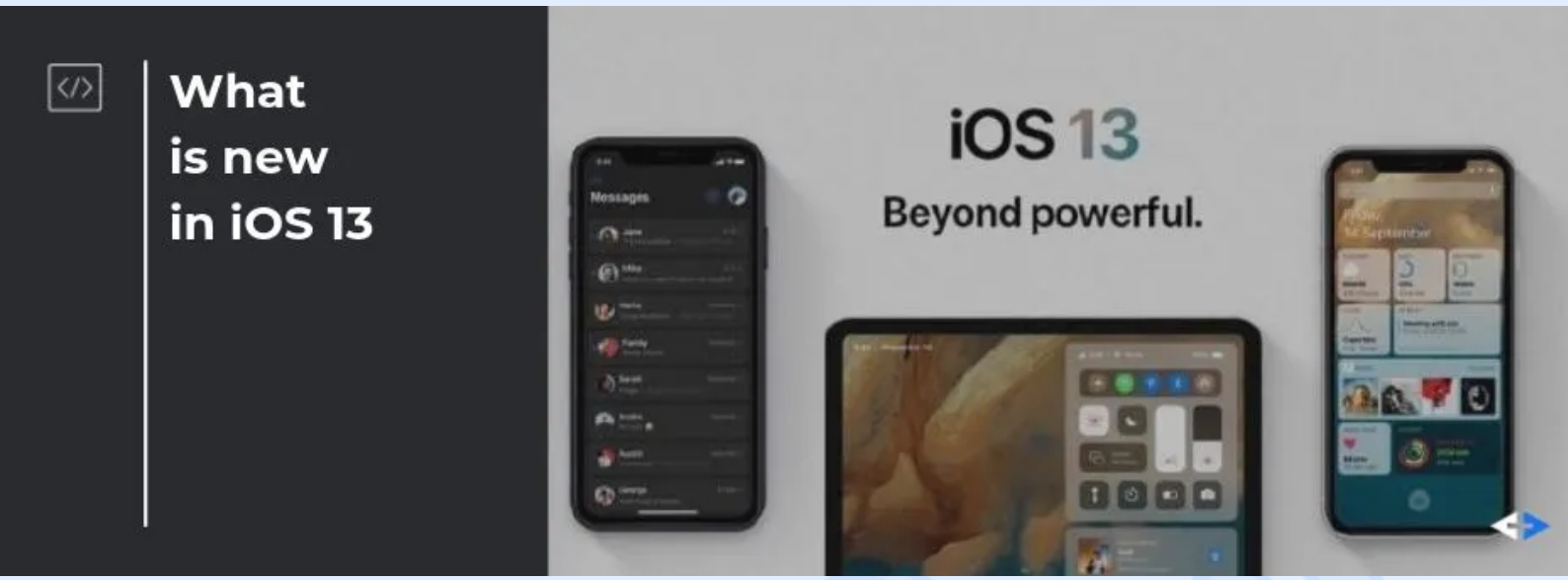


iOS is getting better and faster with every new version. At the WWDC 2019 the latest update to the iPhone operating system was announced, as well as the introduction of Dark Mode, Apple ID and renovated voice assistant Siri. For a fuller feature list, see the official Apple's website. Today, it's time to lift the veil on the technical advancements of iOS 13.

For a professional individual consultation with iOS developers, click here.

The developer's view of iOS 13

We asked our in-house experts to share the results of internal tests of the new features and the iOS 13 environment. Today our colleague, the senior iOS developer Zhangali Pernebayev, shares his perception of how the innovations in Apple's OS will affect mobile developers' approaches.

Promising SwiftUI

+44 20 7183 5820

info@magora.co.uk
sales@magora.co.uk

SwiftUI looks promising from several points:

- Incredibly fast and seems to outpace UIKit.
- Less code and easy layout. Declare the content and layout for any state of your view. SwiftUI knows when that state changes and updates your view's rendering to match.

Here is the look of code written using SwiftUI, which displays a list of landmarks with thumbnail, name, 'favourite' indicator and right arrow.

+44 20 7183 5820

info@magora.co.uk
sales@magora.co.uk

```
ist(landmarks) { landmark in
  HStack {
    Image(landmark.thumbnail)
    Text(landmark.name)
    Spacer()

    if landmark.isFavorite {
      Image(systemName: "star.fill")
        .foregroundColor(.yellow)
    }
  }
```

Let me show you how I used to display a list of landmarks using UIKit.

+44 20 7183 5820

info@magora.co.uk
sales@magora.co.uk

```
private func setupViews() {
    tableView.register(UINib(nibName: "LandMarkTableViewCell", bundle: nil), forCellReuseIdentifier:
        "cell")
}

// MARK: - Table view data source

override func numberOfSections(in tableView: UITableView) -> Int {
    // #warning Incomplete implementation, return the number of sections
    return 1
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // #warning Incomplete implementation, return the number of rows
    return landMarks.count
}
```

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath) as!
        LandMarkTableViewCell
    cell.configure(landMark: landMarks[indexPath.row])
    return cell
}

override func tableView(_ tableView: UITableView, estimatedHeightForRowAt indexPath: IndexPath) ->
    CGFloat {
    return 60
}
```

If you take into account drawing UITableViewCell, using Interface Builder implementation takes more time and code in UIKit than in SwiftUI.

+44 20 7183 5820
info@magora.co.uk
sales@magora.co.uk

```
cl class LandMarkTableViewCell: UITableViewCell {

        @IBOutlet weak var thumbnailImageView: UIImageView!
        @IBOutlet weak var nameLabel: UILabel!
        @IBOutlet weak var favoriteImageView: UIImageView!

        func configure(landMark: LandMark) {
            thumbnailImageView.image = landMark.thumbnail
            nameLabel.text = landMark.name
            favoriteImageView.isHidden = !landMark.isFavorite
        }
} }
```

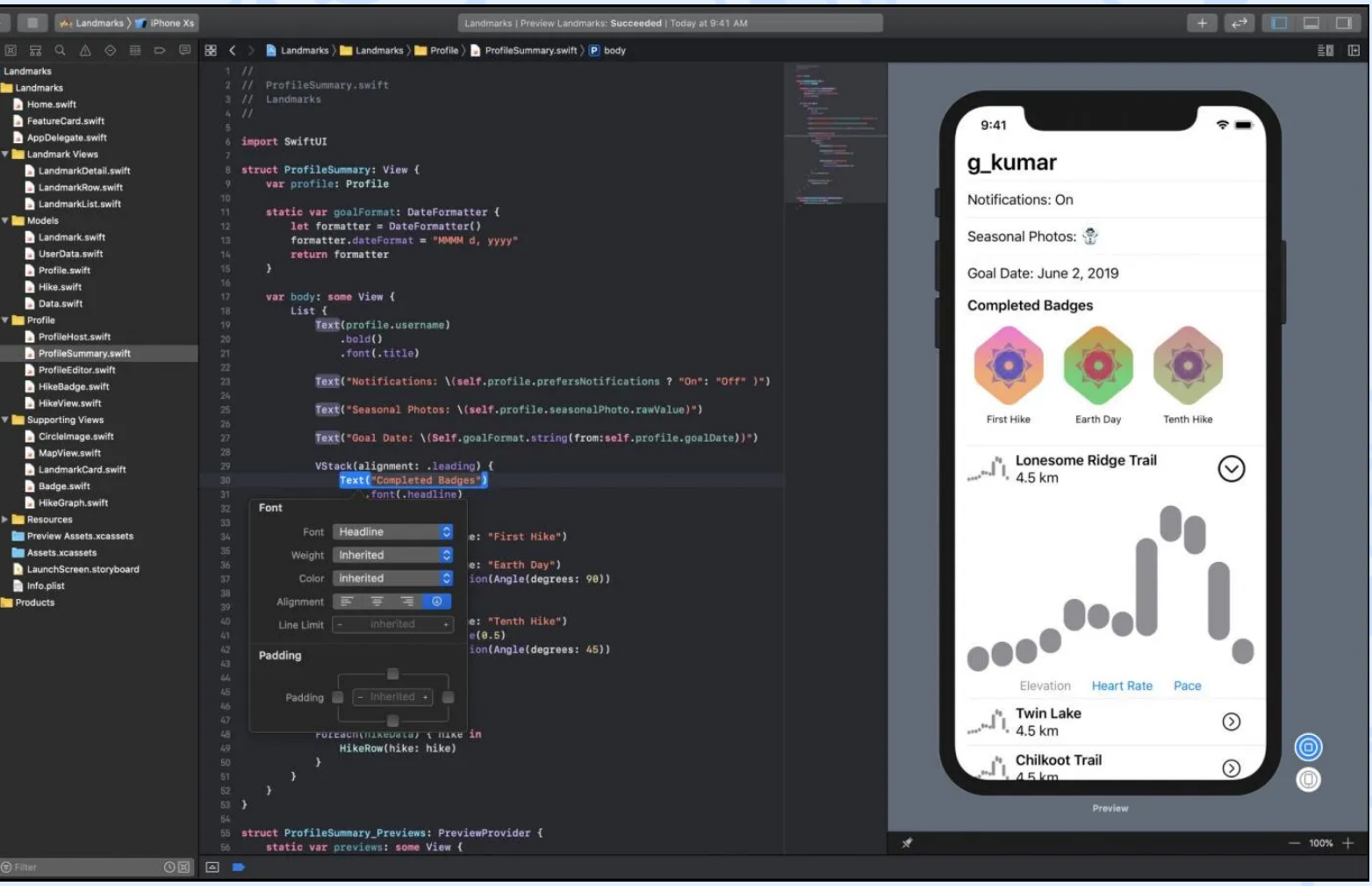
- Offers the ability to build reusable components, which you can share between apps designed for any Apple platform.
- Simplified animations. Creating smooth animations is as easy as adding a single method call. SwiftUI automatically calculates and animates transitions when needed.

+44 20 7183 5820

info@magora.co.uk
sales@magora.co.uk

```
VStack {
    Badge()
        .frame(width: 300, height: 300)
        .animation(.basic())
    Text(name)
        .font(.title)
        .animation(Animation.basic().delay(0.2))
}
```

- Live Preview lets you interact with the preview the same as if your app is running in the simulator - awesome!



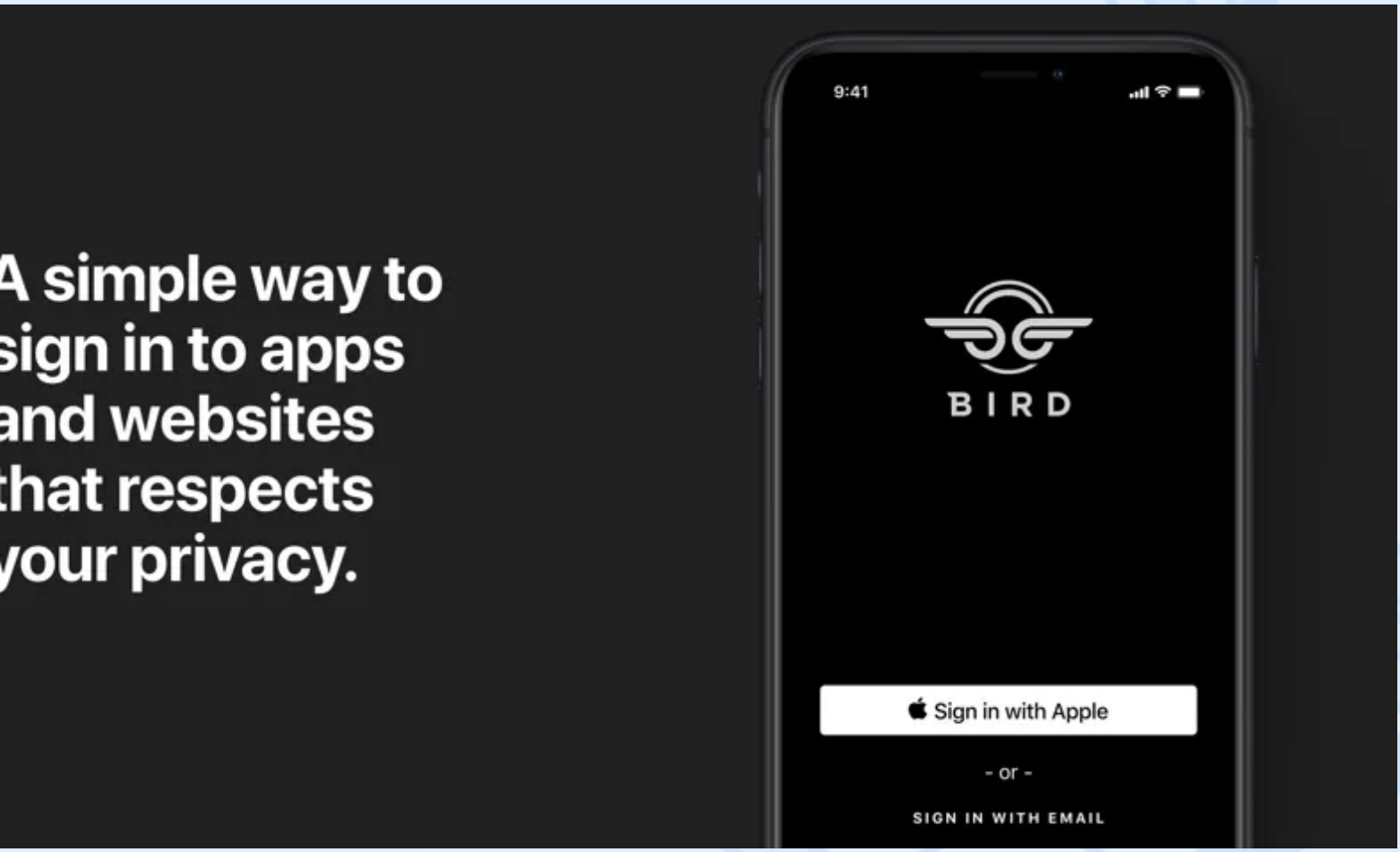
+44 20 7183 5820
info@magora.co.uk
sales@magora.co.uk

- Moreover, SwiftUI works seamlessly with the existing UI Frameworks on all Apple platforms. For instance, you can use SwiftUI views inside UIKit views and view controllers, and vice versa. I suppose SwiftUI will be a real helper for our implementation of iOS 13 into mobile app development for 2020.

Camera Capture

In iOS 13, Apple releases the opportunity to record the output from the front and back cameras into a single movie file by using AVCaptureMultiCamSession. For our development team, this is an option which simplifies native Apple application creation.

Sign In with Apple



- Sign In with Apple is a totally secure and private procedure because accounts are protected with two-factor authentication – so you can be sure that the app you develop will not provoke any security issues. Sites and apps will be able to request your email address, but you can also opt to hide it. In that case, Apple will generate a random email address that forwards messages from those services to your regular inbox.
- Apple doesn't track users' activity.

+44 20 7183 5820
info@magora.co.uk
sales@magora.co.uk

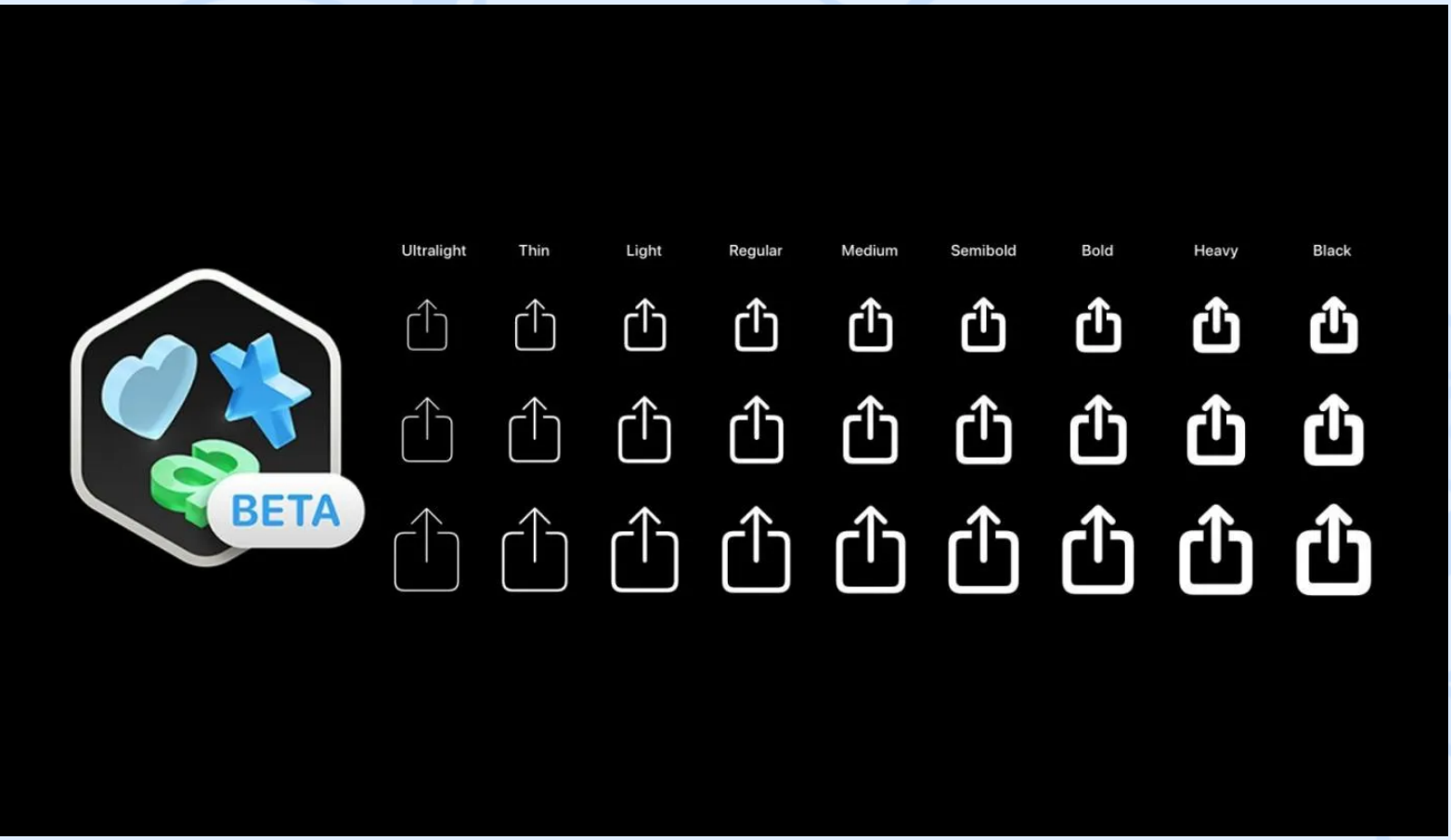
- To use it, you don’t need to integrate external libraries or SDKs such as Google and Facebook anymore, which saves the memory of your app. Moreover, you can use sign in with Apple instead of drawing your own forms for authorization.
- When designing a native or web application now, you need to remind the client of this possibility. Besides Google and Facebook, Apple Sign In can also now be used, which is one more button to place in the interface.

Here’s an illustration of adding Sign In with Apple Button:

```
let authorizationButton = ASAuthorizationAppleIDButton()
authorizationButton.addTarget(self, action:
#selector(handleLogInWithAppleIDButtonPress), for: .touchUpInside)
loginProviderStackView.addArrangedSubview(authorizationButton)
```

For detailed integration see this article.

SF Symbols



+44 20 7183 5820
info@magora.co.uk
sales@magora.co.uk

SF Symbols provides over 1 500 built-in symbols which you can use in your app without downloading and storing the same figures. These comprise a set of over 1,000 signs, each of which is a highly configurable symbol designed to integrate with Apple's system font, San Francisco.

- These can be used in iOS 13 but also in watchOS 6 and tvOS 13. Each symbol was designed with a variety of sizes and weights so you can also adjust the weight and scale of the image.

For example, here's the usage of a built-in 'gamecontroller' symbol:

```
let gamecontroller = UIImage(systemName: "gamecontroller")
```

You can also adjust the weight and scale of the image:

```
let config = UIImage.SymbolConfiguration(pointSize: 25, weight: .black, scale: .large)
let gamecontroller = UIImage(systemName: "gamecontroller", withConfiguration: config)
```

Vision

With iOS 13 Apple has now its own built-in OCR and image recognition framework – Vision. It saves time, money, memory and nerves, preventing you from being dependent on third-party frameworks like Tesseract and SwiftOCR. No more integration conflicts, additional tests and regular updates. Everything will be seamlessly clear now.

MetricKit

Personally, for me it is the best framework in iOS 13, because it helps to gather information about the problems users are currently facing and record them in your own tools.

- These metrics are in the form of histograms that record the frequency of observed values over a day. MetricKit goes beyond the metrics shown in the Metrics organiser to include average pixel

+44 20 7183 5820
info@magora.co.uk
sales@magora.co.uk

luminance, cellular network conditions and durations associated with custom OSSignpost events in your app.

Here are some specific benefits of minimising resource usage:

- Decreases app launch time and improves the user experience, while reducing the chances of the iOS watchdog timer terminating the app.
- Reduces overall memory used and the likelihood of iOS freeing your app's memory in the background, improving responsiveness when a user switches back to your app.
- Eliminates disk writes and speeds up your app's overall performance, makes it more responsive and reduces wear on users' device storage.
- Minimising hang rate and hang duration improves your users' perception of your app's performance and responsiveness.
- Reducing battery consumption and the use of power-hungry device features makes your app more reliable and helps ensure that the rest of the user's device is available when needed.

Combine

	RxSwift	Combine
Deployment Target	iOS 8.0+	iOS 13.0+
Platforms supported	iOS, macOS, tvOS, watchOS, Linux	iOS, macOS, tvOS, watchOS, UIKit for Mac [1]
Spec	Reactive Extensions (ReactiveX)	Reactive Streams (+ adjustments)
Framework Consumption	Third-party	First-party (built-in)
Maintained by	Open-Source / Community	Apple
UI Bindings	RxCocoa	SwiftUI [2]

Great news for those who use and enjoy Reactive Programming. It is the replacement for third-party asynchronous event-handling frameworks such as RxSwift and ReactiveSwift. It allows you to process values over time. Examples of these kinds of values include network responses, user interface events and other types of asynchronous data.

Overall, as you can see iOS 13 has many pleasant surprises for both users and developers. As usual, we are prepared to implement them shortly for our partners.

Check out our other materials about iOS.

+44 20 7183 5820

info@magora.co.uk
sales@magora.co.uk