ADC
Applaudo
Developers
Conference
2020

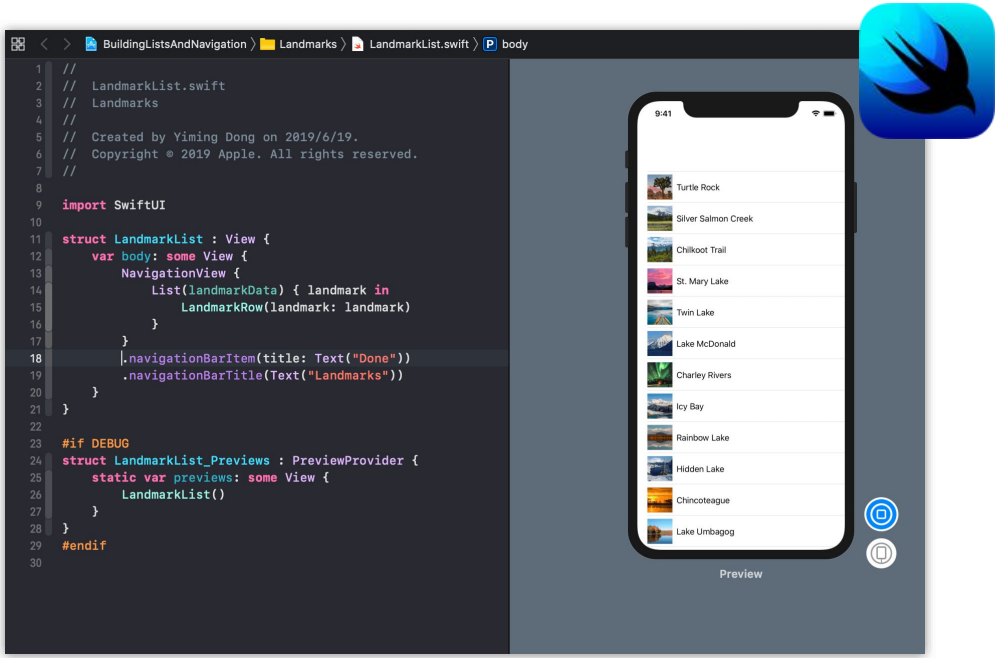
Powered by ApplaudoStudios

Getting Started with Combine and UIKit.

Introducción a Combine y su potencial uso junto a UIKit.

Potenciando el desarrollo iOS

Apple tiene una mejora continua en todas las herramientas para desarrolladores, con el propósito de que estas herramientas potencie y facilite el desarrollo en sus plataforma nativa. Entre estas mejoras las más destacadas son Combine y SwiftUI.



Emilio Vásquez

iOS Developer

@RagdeSey

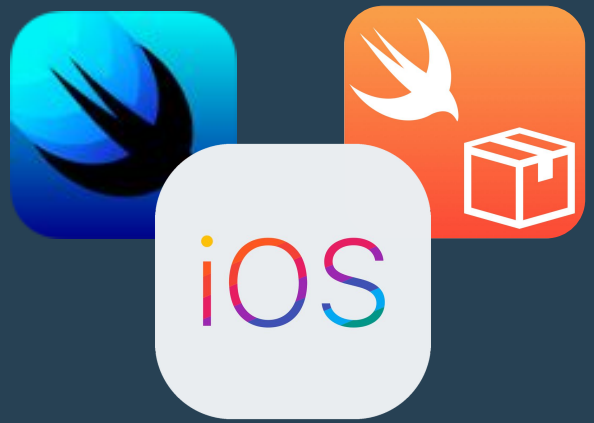
@EmilioVasquez

Agenda

1. What is new in iOS?
2. Asynchronous programming
3. Reactive Programming
4. Combine Basis
5. Combify your UIKit Apps
6. Advantages and Disadvantages
7. Where to learn?
7. Ask and Questions

What is new on iOS

Features, updates y herramientas a partir de iOS 13+

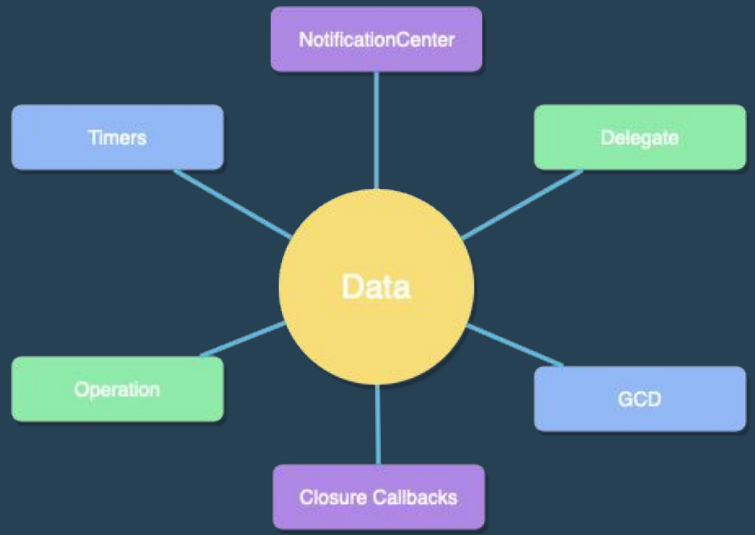


iOS 13+

- Combine Framework
- SwiftUI
- New and Fresh Diffable Datasources
- Compositional Layout
- DI en storyboards
- Dark mode
- Multi-Window Support (AppDelegate y SceneDelegate)
- SPM support en iOS targets
- Apple Sign in

Asynchronous programming

Foundation, UIKIt and Asynchronous programming

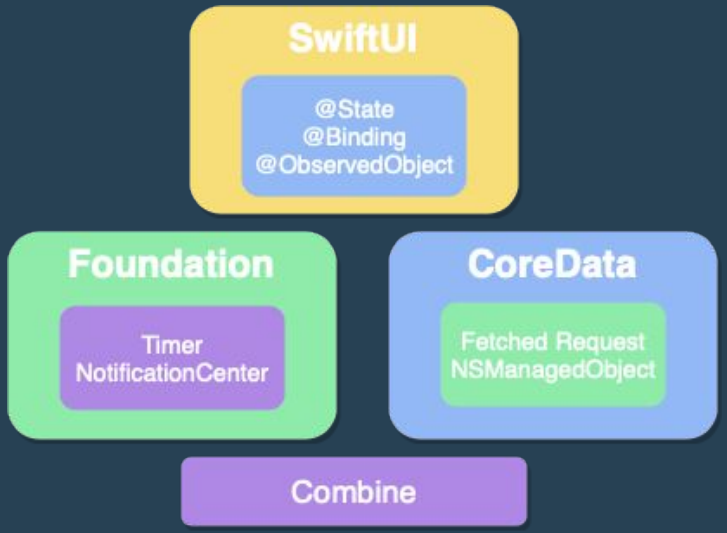


Asynchronous programming before iOS 13

- Notification Center
- Delegate Pattern
- GDC (Grand Central Dispatch)
- Closures

Framework útiles para facilitar la programación asíncrona.

- PromiseKit



Asynchronous programming after iOS 13

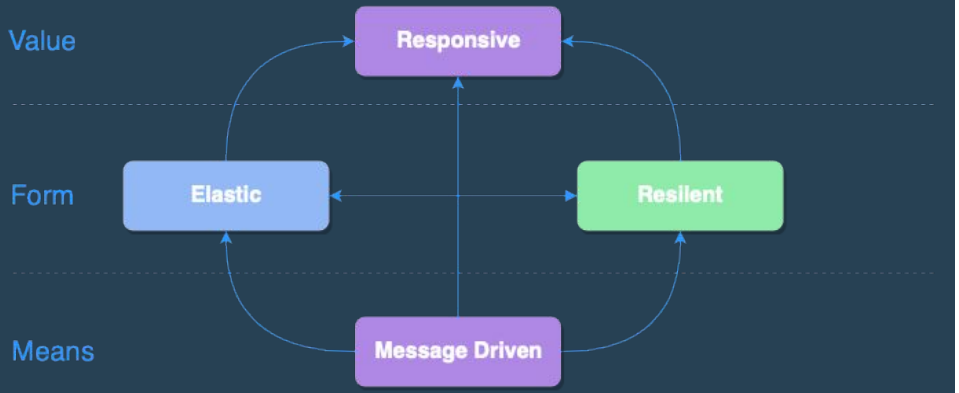
- Notification Center
- Delegate Pattern
- GDC (Grand Central Dispatch)
- Closures

Y.....

- Combine

Reactive Programming

The beginning of Combine



Conceptos

La programación reactiva es un paradigma de programación orientado a los flujos de datos y la propagación del cambio. Esto significa que debería ser posible expresar los flujos de datos estáticos o dinámicos con facilidad en los lenguajes de programación utilizados, y que el modelo de ejecución correspondiente propaga automáticamente los cambios a través del flujo de datos.

La principal ventaja es que los programadores pueden centrarse más en el business logic en lugar de el estado de la data y sus efectos secundarios (*side effects*).

Reactive Frameworks

Las primera solución reactive vino en gran medida en 2009 cuando Microsoft lanzó la libreria llamada *Reactive Extension* para .Net (Rx.Net), Microsoft hizo Rx.Net open source en 2012 y desde ese momento, muchos lenguajes han comenzado a utilizar sus conceptos.

Actualmente existen varias implementaciones del estándar Rx como:

- RxJS
- RxKotlin
- RxScala
- RxPHP
- RxJava
- RxGo

Third-party reactive frameworks for iOS

Combine Basis

Fundamentos basicos de Combine

Publishers

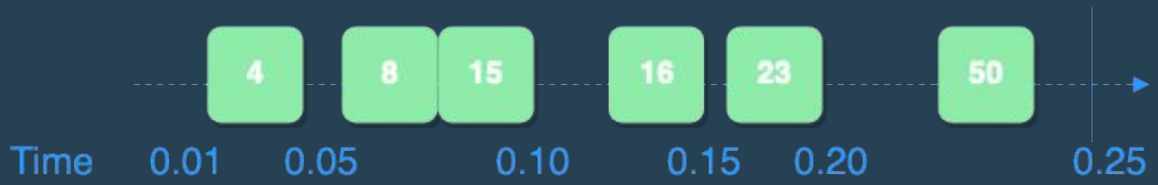
Definición

Son tipos que pueden emitir valores a través del tiempo a uno o más partes interesadas, conocidas como **subscribers**. Los publisher pueden emitir múltiples eventos de estos tres tipos:

- Un valor generico resultate conocido como **Output**
- Un evento de terminacion exitosa (**successful completion**)
- Un evento de fallo con un error de tipo **Failure**

Importante

Un publisher puede emitir cero o más valores, y si el flujo es completado debido a un evento de éxito o error, una vez emitido uno de estos eventos el publisher no emitirá otro evento.



Publisher

Aquí se muestra un publisher que emite valores enteros los cuales puede visualizarse en una línea de tiempo, y los números representan los valores emitidos. La línea vertical a la derecha representa la terminación exitosa del flujo.

Subjects

Definición

Son un tipo especial de Publishers, estos puede ser utilizados para “inyectar” valores a un flujo de data, llamando el método **send(_:)** , este es útil para integrar código imperativo existente con Combine. Hay dos tipos de subjects:

- CurrentValueSubject
- PassthroughSubject

Diferencia

Los dos Subjects mencionados tienen un funcionamiento similar sin embargo con una diferencia sustancial y es que CurrentValueSubject requiere un valor inicial, mientras que PassthroughSubject no.

Operadores

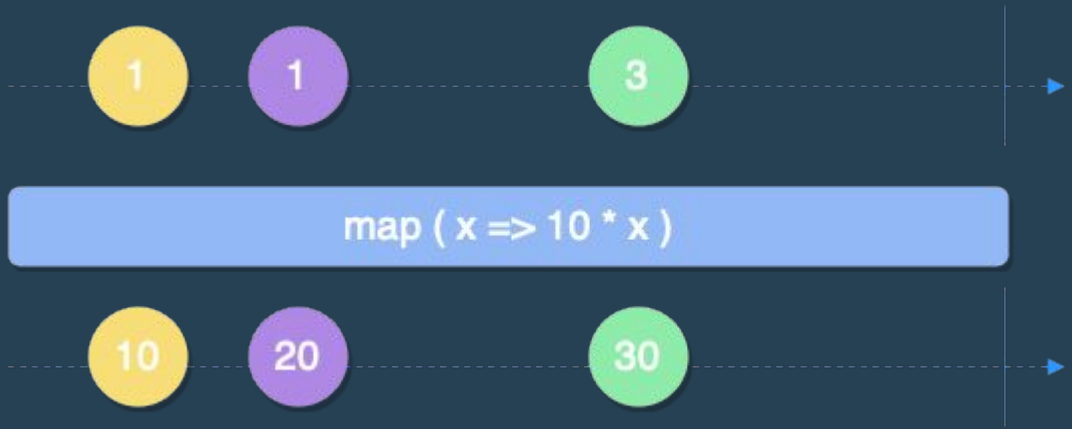
Definición

Son métodos declarados en el protocolo de Publisher que retornan el mismo publisher o un nuevo, son útiles porque es posible llamar una cantidad de operadores uno tras otro, creando una cadena entre ellos. Con estos es posible implementar lógica bastante compleja en una sola Subscription.

Naming

Algunos operadores pueden hacerse familiar de los métodos de transformación la Collection en la librería estándar de Swift, la mayoría tiene su mismo funcionamiento pero otros no como **flatmap**.

- Transforming Operators: **map, collect, flatMap, tryMap, scan.**
- Filtering Operator como **compactMap, filter, drop, last, first.**
- Combining Operator como **prepend, append, merge, zip, combineLatest**
- Time Manipulation Operators such as debounce, timeout, throttle
- Sequence Operators such as min, max, first, last, output, count, contains



Operador map

Aquí se muestra como el operador map modifica el data en el tiempo.

Subscribers

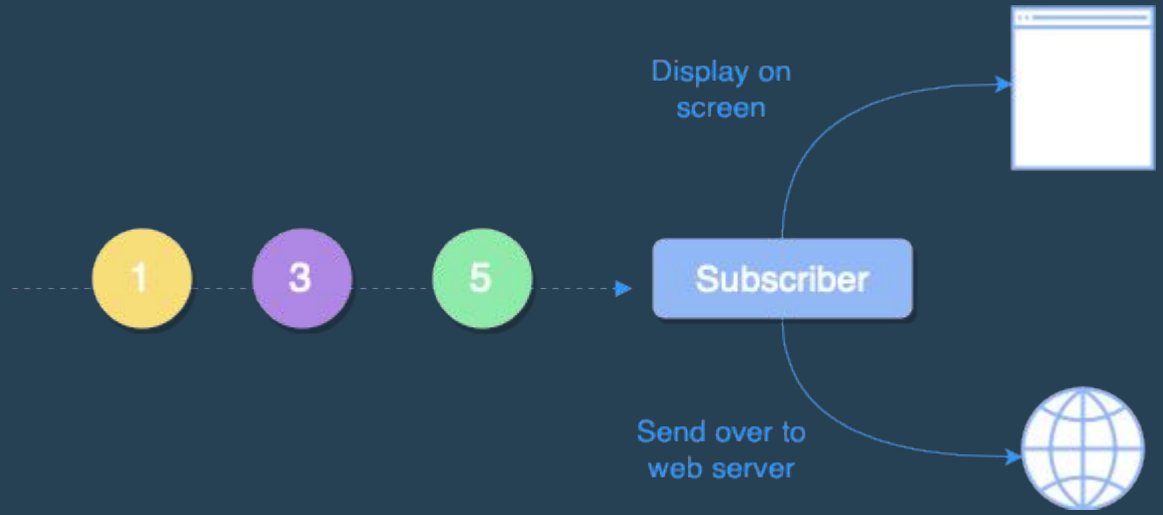
Definición

Usualmente por fácil comprensión se refiere como al final del flujo, cada Subscription termina con un Subscriber, en general es el punto donde podemos hacer una acción con el valor emitido o los completion de los eventos como success y error.

Tipos de subscribers

Combine tiene dos tipos de Subscribers, que facilita el trabajo final con la data:

- **sink:** provee closures que reciben los valores (output) y los completion (error, y success).
- **assign:** este permite sin ningun codigo personalizado, enlazar el resultado de la suscripción directamente a una propiedad de tu modelo o UI vía keypaths.



Subscriber

Aquí se muestra como gráficamente se vería un Subscriber en un Subscription.

Subscription

Definición

Subscription es el concepto del flujo completo iniciando desde un Publisher, pasando por los operadores y finalizando en un Subscriber, esto permite declarar una cadena de eventos asíncronos con su propio código personalizado y manejo de error una sola vez, y nunca volver ha realizar un cambio a menos que lo requiera.

Importante

Cuando se define un Subscriber al final de la Subscription, el Publisher empieza a emitir valores, sin embargo sin no hay ningún Subscriber definido el Publisher no emite ningún valor por lo tanto la Subscription no es ejecutada.

Combify your UIKit apps

Implementa Combine junto a UIKit

Closures

Una simple call de URLSession con closures

```
let session = URLSession.shared

let task = session.dataTask(with: urlRequest) { (data, response, error)  in
    if let data = response.data, !data.isEmpty {
        do {
            if let decoded = try JSONDecoder().decode(Foo.self, from: data){
                // Do something with the data decoded
            } else {
                // Return an error for nil decoded data
            }
        } catch {
            // Return an error for bad decoding
        }
    } else if let error = error {
        //return the error
    } else {
        // return the error for empty data
    }
}

task.resume()
```

Promises

Una simple call de URLSession con promises

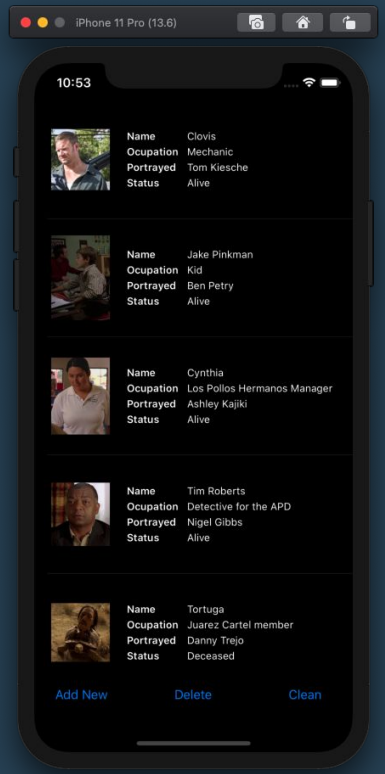
```
firstly {
   URLSession.shared.dataTask(.promise, with: urlRequest).validate()
}.map {
   try JSONDecoder().decode(Foo.self, from: $0.data)
}.done { foo in
 // Do something with the decoded data
}.catch { error in
   // Do something with the error
}
```

Combine

Una simple call de URLSession con Combine.

```
URLSession.shared.dataTaskPublisher(for: urlRequest)
            .map(\.data)
            .decode(type: [Character].self, decoder: JSONDecoder())
            .eraseToAnyPublisher()
```

Ejemplo práctico



Ventajas y desventajas

Ventajas

Combine está integrado a nivel del sistema.

Abstrae todos las operaciones asíncronas en operadores.

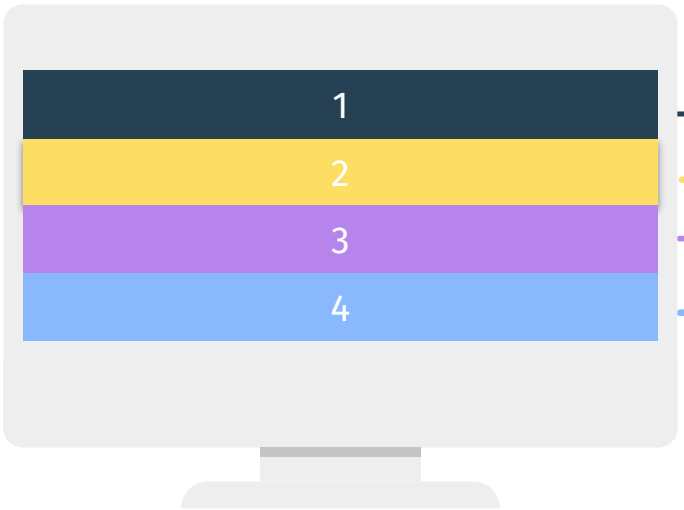
Los operadores de Combine son compuestos.

Combine no altera la arquitectura de una aplicación

Existen librerías de la comunidad de desarrolladores que pueden extender su funcionalidad.

El código será más legible y mantenible, eliminando estrategias como closures encadenados

Desventajas



La curva de aprendizaje puede ser alta.

Combine no es open-source

El debugging puede llegar a ser difícil pero no imposible.

Combine esta disponible solamente en iOS 13 en adelante.

Where to go from here?

Lecturas Recomendadas

- Combine: Asynchronous Programming with Swift by RayWenderlich
- Using Combine by Joe Heck
- Apple Combine Documentation by Apple
- Introducing Combine WWDC19 by Apple.
- Combine in Practice WWDC19 by Apple
- Advances in UI Data Sources WWDC19 by Apple
- Combine framework in Swift by Antoine Van der Lee
- Combine by John Sundell
- RxMarbles by Staltz

Preguntas y Respuestas

Applaudo
Developers
Conference
2020

Powered by ApplaudoStudios