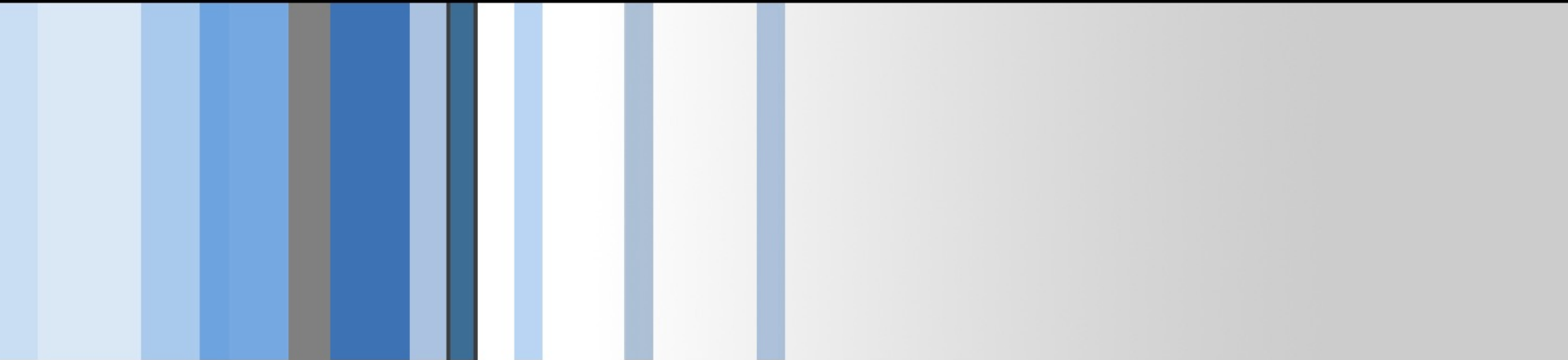
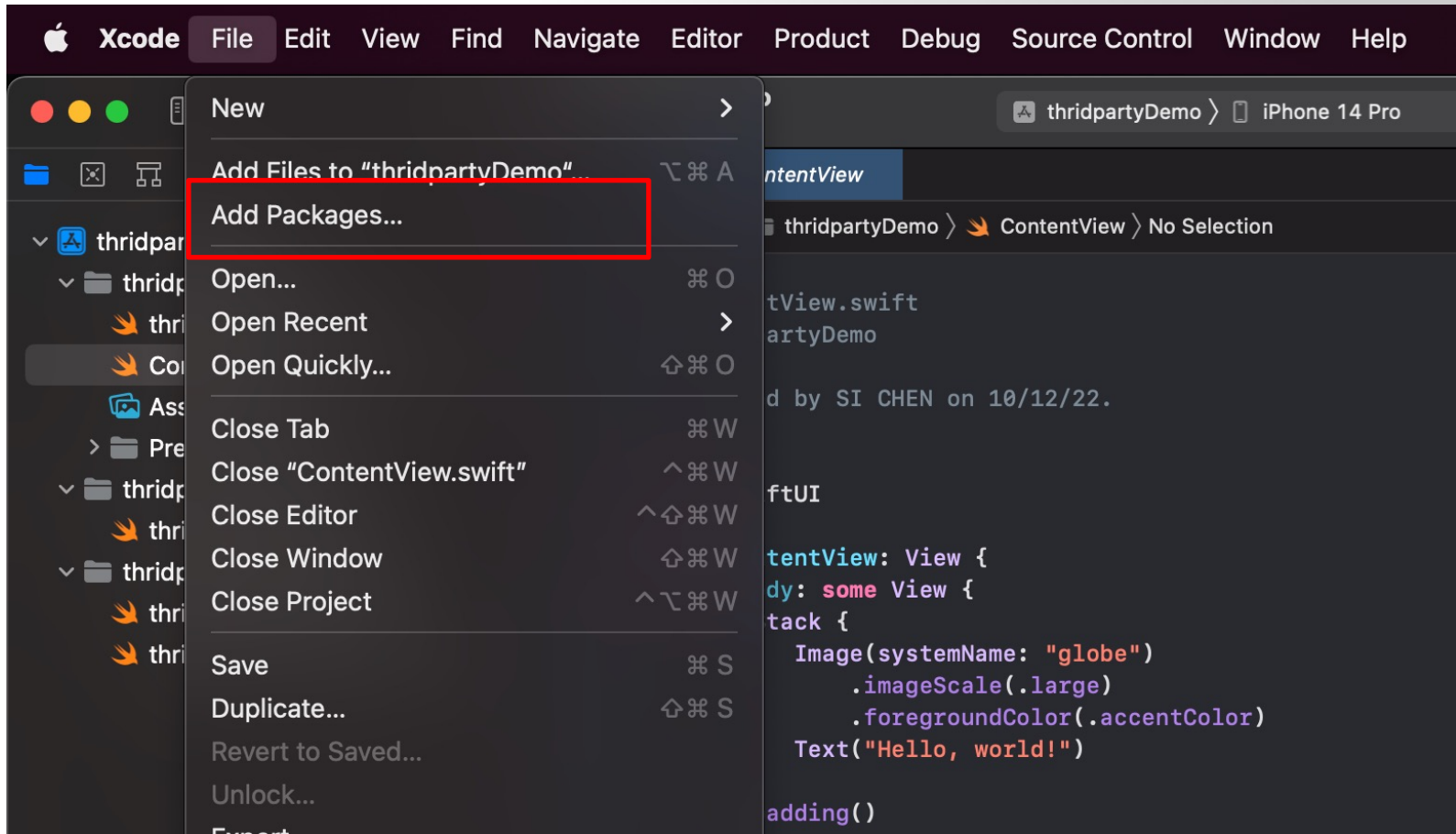


CSC 496: iOS App Development Beyond the Standard Library

Si Chen (schen@wcupa.edu)



Add Third-party packages



Add Third-party packages

Recently Used 2

Collections

Apple Swift Packages 9

Apple Swift Packages
developer.apple.com

swift-algorithms
Commonly used sequence and collection algorithms for Swift

swift-argument-parser
Straightforward, type-safe argument parsing for Swift

swift-atomics
Low-level atomic operations for Swift

swift-collections
Commonly used data structures for Swift

swift-crypto
Implements most of Apple's CryptoKit API for use across multiple platforms.

swift-nio
Supports development of asynchronous event-driven network applications that maintain high performance...

swift-numeric
Advanced mathematical types and functions for Swift

swift-system

swift-algorithms Swift ☆ 4.61k

Version 1.0.0 Authors natecook1... Release Date Sep 8, 2021 License Apache v2.0 Repository [github....](#)

Dependency Rule Up to Next Major Version 1.0.0 < 2.0.0

Add to Project thridpartyDemo

Description Release History Some metadata provided by [github.com](#)

Swift Algorithms

Swift Algorithms is an open-source package of sequence and collection algorithms, along with their related types.

Read more about the package, and the intent behind it, in the [announcement on swift.org](#).

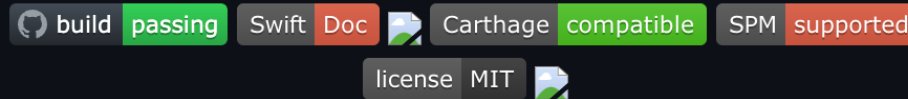
Contents

Combinations / permutations

- `combinations(ofCount:)` : Combinations of particular sizes of the elements in a collection.
- `permutations(ofCount:)` : Permutations of a particular size of

+ - ↺

Add Local... Cancel Add Package



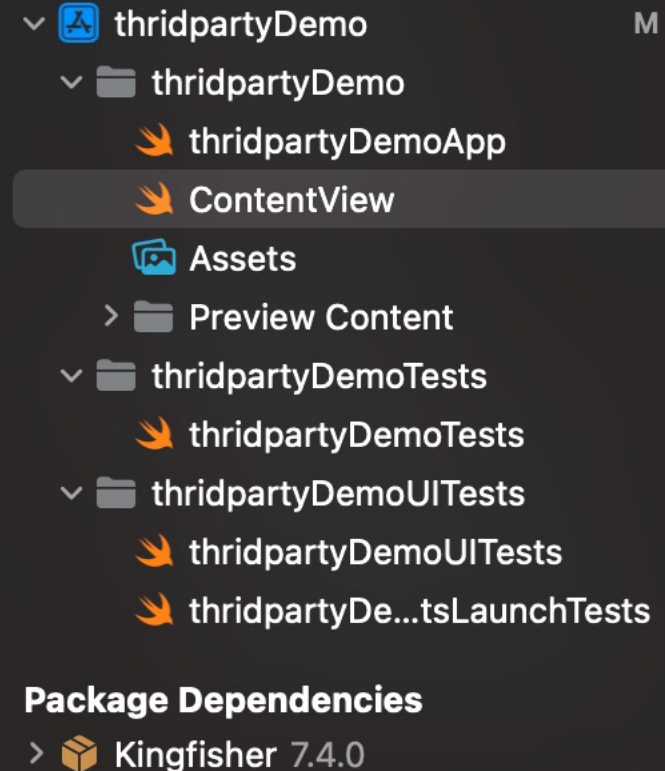
Kingfisher is a powerful, pure-Swift library for downloading and caching images from the web. It provides you a chance to use a pure-Swift way to work with remote images in your next app.

<https://github.com/onevcats/Kingfisher>

Kingfisher

Swift Package Manager

- File > Swift Packages > Add Package Dependency
- Add `https://github.com/onevc/Kingfisher.git`
- Select "Up to Next Major" with "7.0.0"



Kingfisher

```
import SwiftUI
import Kingfisher

struct ContentView: View {
    var body: some View {
        VStack {
            Image(systemName: "globe")
                .imageScale(.large)
                .foregroundColor(.accentColor)
            Text("Hello, world!")
        }
        .padding()

        KFIImage(URL(string:
            "https://www.wcupa
            .edu/_resources/includes/brand/images/headerLogo1.png"))!)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

PokemonAPI

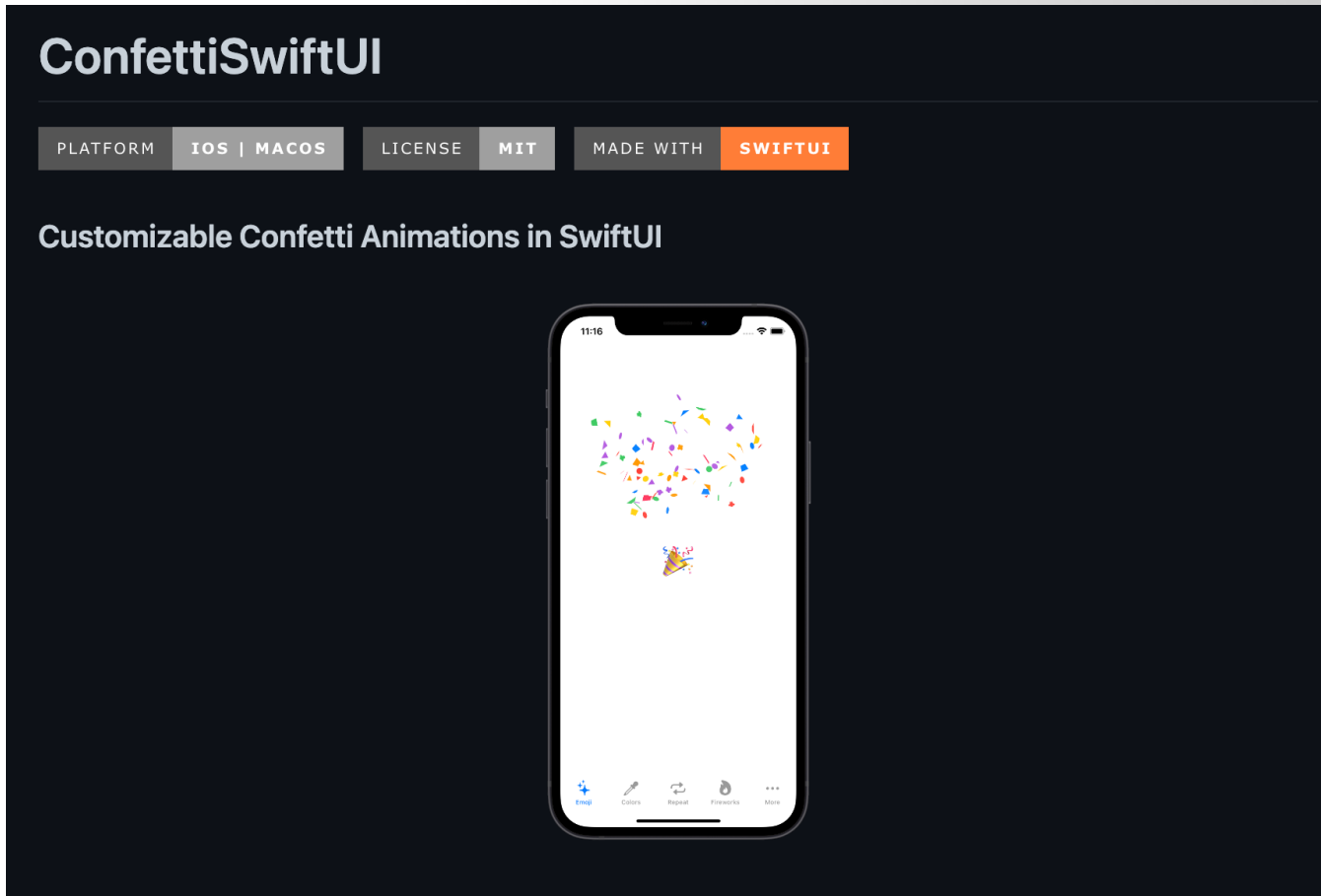
```
import PokemonAPI

struct ContentView: View {
    @State var pokemonName = ""
    @State var pokemonWeight = 0
    var body: some View {

        Text(pokemonName)
        Text(String(pokemonWeight))
        Button("Click"){
            PokemonAPI().pokemonService.fetchPokemon("150") { result in
                switch result {
                    case .success(let pokemon):
                        self.pokemonName = pokemon.name ?? "" // bulbasaur
                        self.pokemonWeight = pokemon.weight ?? 0 //
                        bulbasaur
                    case .failure(let error):
                        print(error.localizedDescription)
                }
            }
        }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

<https://github.com/kinkofer/PokemonAPI>



<https://github.com/simibac/ConfettiSwiftUI>

SUBSONIC

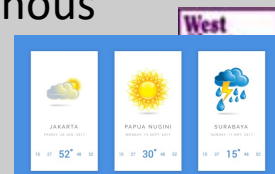
Swift 5.4 Contact @twostraws

Subsonic is a small library that makes it easier to play audio with SwiftUI, allowing you to work both imperatively ("play this sound now") and declaratively ("play this sound when some state becomes true").

Lab3: Build an app with API and third-party library (Group Project)

■ Assignment Requirements:

- 1.API Selection:** Choose an API for integration from <https://github.com/public-apis/public-apis>. Alternatively, you may opt for any other API that does not necessitate authentication.
 - 2.Data Fetching:** Develop an application capable of fetching data from the chosen API. Your application should retrieve at least one property or data field from the API.
 - 3.User Interface (UI):** Construct a user interface to display the fetched data. Implement features that allow the user to initiate new data requests. For example, if you select the Weather API, the interface should enable the user to specify a state (like PA or NY) for which to fetch data.
 - 4.Dynamic Imagery:** Incorporate at least one changeable image in the application's UI. The image should update based on the data retrieved from the API. For instance, the image could turn red if the fetched daily temperatures exceed a certain threshold.
 - 5.Library Usage:** Utilize **at least one third-party library** in the development of your application.
- ## ■ Bonus Criteria:
- Asynchronous Programming:** Earn a 2 points bonus if you implement asynchronous programming using `async/await` for data fetching and handling.



Q & A

