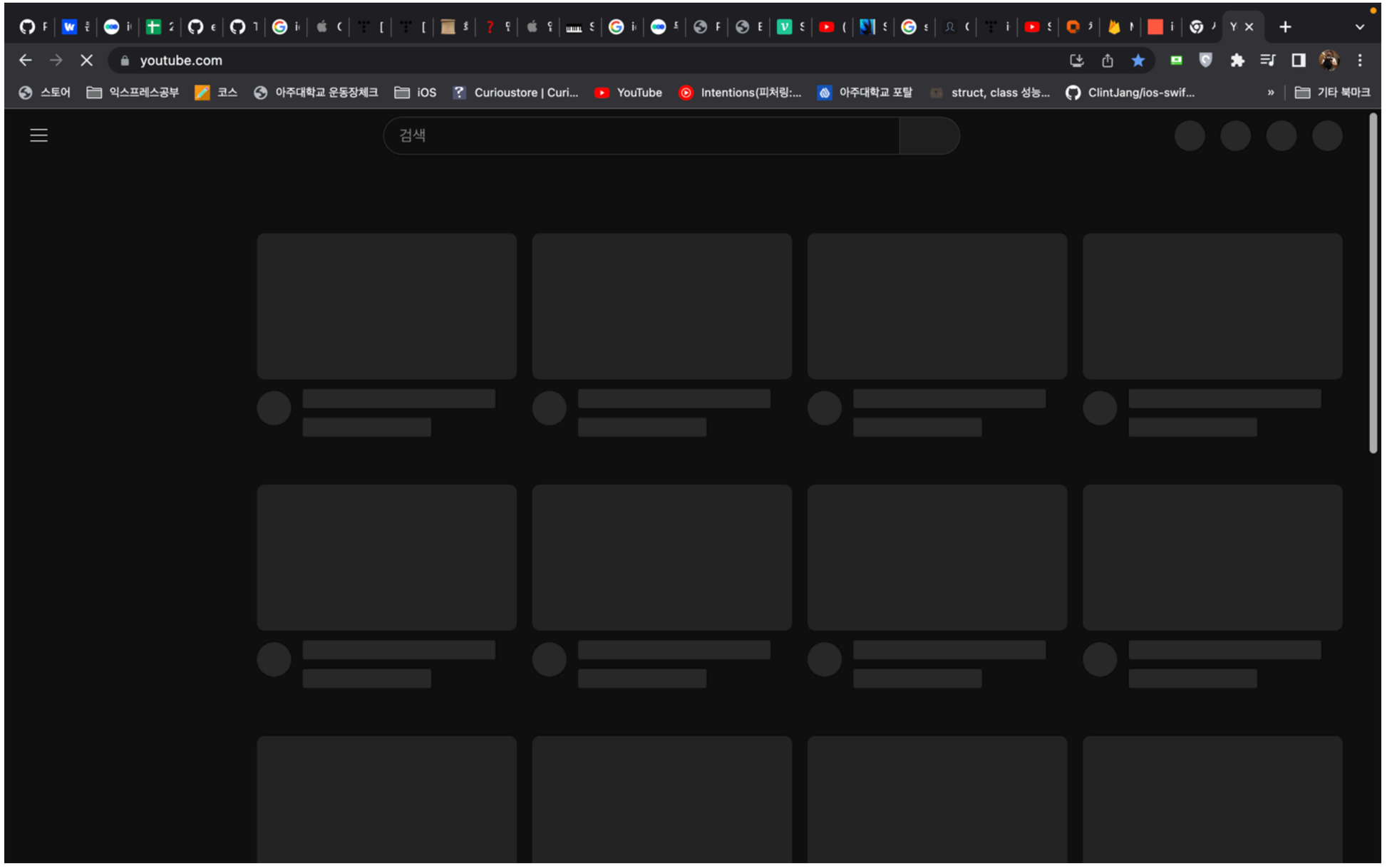
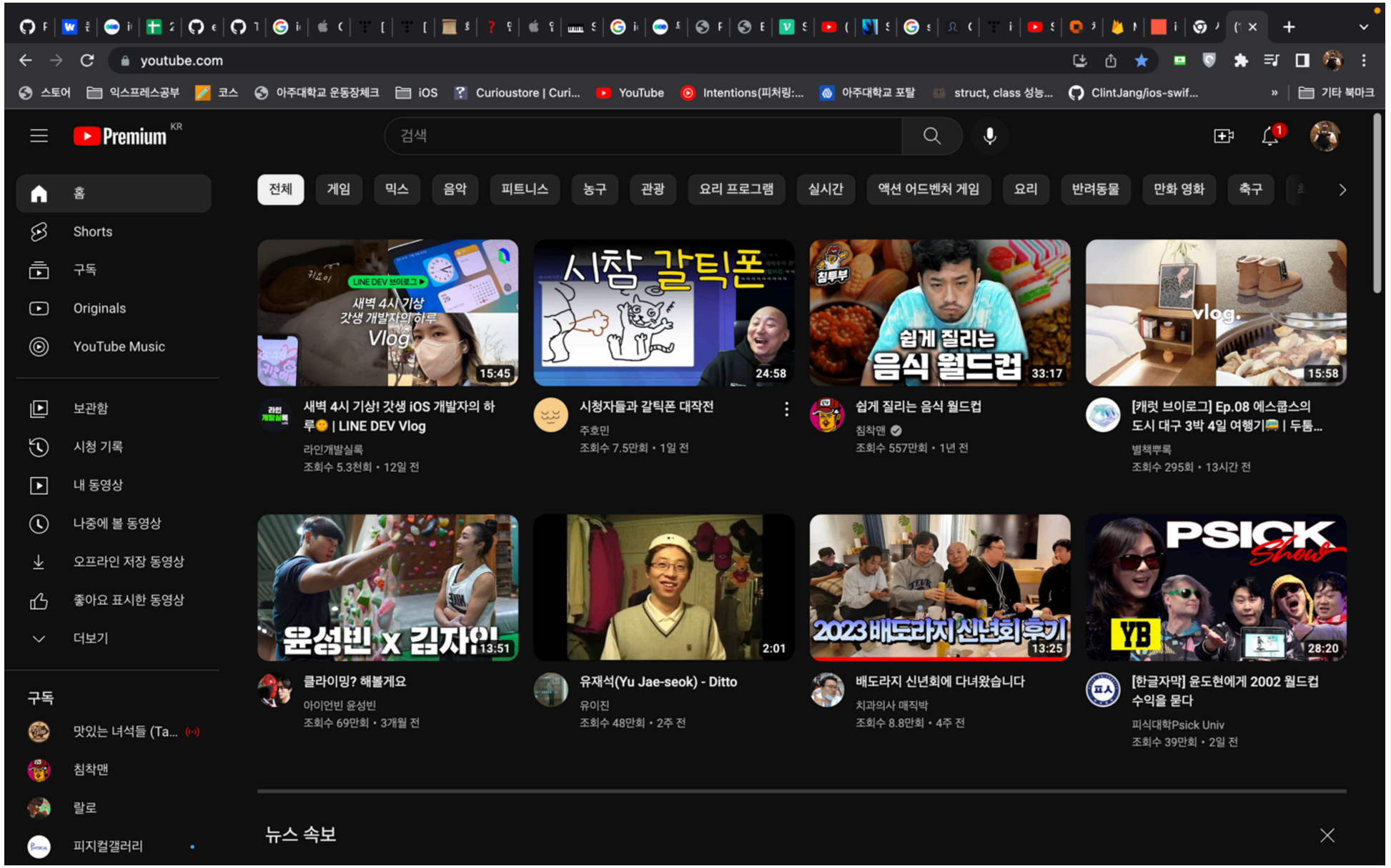
비동기 Swift 코드

Closure vs Combine vs Async/Await

허두영 23/02/08

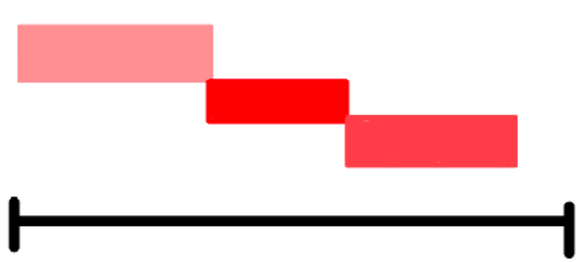

youtube.com
스토어
익스프레스공부
코스
아주대학교 운동장체크
iOS
Curioustore | Curi...
YouTube
Intentions(피처링:...
아주대학교 포탈
struct, class 성능...
ClintJang/ios-swif...
기타 북마크
검색


youtube.com
Premium KR
검색
홈
Shorts
구독
Originals
YouTube Music
보관함
시청 기록
내 동영상
나중에 볼 동영상
오프라인 저장 동영상
좋아요 표시한 동영상
더보기
구독
맛있는 녀석들 (Ta...
침착맨
랄로
피지컬갤러리
전체
게임
믹스
음악
피트니스
농구
관광
요리 프로그램
실시간
액션 어드벤처 게임
요리
반려동물
만화 영화
축구
LINE DEV 브이로그
새벽 4시 기상
갓생 개발자의 하루
Vlog
15:45
새벽 4시 기상! 갓생 iOS 개발자의 하루 | LINE DEV Vlog
라인개발실록
조회수 5.3천회 • 12일 전
시참 갈틱폰
24:58
시청자들과 갈틱폰 대작전
주호민
조회수 7.5만회 • 1일 전
쉽게 질리는
음식 월드컵
33:17
쉽게 질리는 음식 월드컵
침착맨
조회수 557만회 • 1년 전
vlog.
15:58
[캐럿 브이로그] Ep.08 에스쿱스의 도시 대구 3박 4일 여행기 | 두툼...
별책뿌록
조회수 295회 • 13시간 전
윤성빈 x 김자인
13:51
클라이밍? 해볼게요
아이언빈 윤성빈
조회수 69만회 • 3개월 전
2:01
유재석(Yu Jae-seok) - Ditto
유이진
조회수 48만회 • 2주 전
2023 배도라지 신년회 후기
13:25
배도라지 신년회에 다녀왔습니다
치과의사 매직박
조회수 8.8만회 • 4주 전
PSICK Show
YB
28:20
[한글자막] 윤도현에게 2002 월드컵 수익을 묻다
피식대학Psick Univ
조회수 39만회 • 2일 전
뉴스 속보

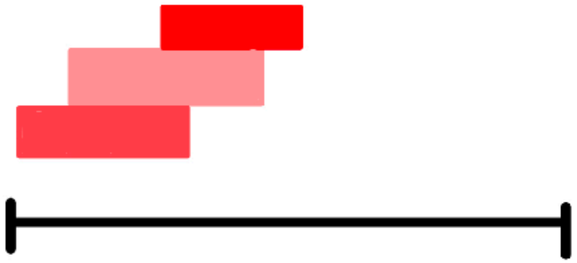
비동기 프로그래밍

- 프로그램의 주요 실행 흐름에서 특정 작업에 시간이 많이 소요된다면, 그 작업을 수행하면서 아래 프로그램 코드를 계속 수행하는 방식입니다.

- 구현방법 in iOS
  - 후행 클로저 (콜백함수)
  - Combine or RxSwift
  - Async/Await



Synchronous
동기



Asynchronous
비동기

클로저 방식

- Completion 같은 클로저를 파라미터로 받아 함수가 끝나면 클로저를 실행시킵니다 (콜백 함수)
- @escaping 클로저 : 클로저가 함수의 파라미터로 전달됐을 때, 함수의 실행이 종료된 후 실행되는 클로저
- Result type : 성공과 실패를 나타내는 값 타입 (에러 처리를 위해)

```
func perform(_ request: URLRequest, completion: @escaping (Result<Data, Error>) -> Void) {
    let task = URLSession.shared.dataTask(for: request) { data, response, error in
        if let error = error {
            completion(.failure(error))
            return
        }
        guard let data = data,
              let httpResponse = response as? HTTPURLResponse,
              (200..<400).contains(httpResponse.statusCode) else {
            completion(.failure(URLError(.badServerResponse)))
            return
        }
        completion(.success(data))
    }
    task.resume()
}
```

```swift
func perform(_ request: URLRequest, completion: @escaping (Result<Data, Error>) -> Void) {
    let task = URLSession.shared.dataTask(for: request) { data, response, error in
        if let error = error {
            completion(.failure(error))
            return
        }
        guard let data = data,
              let httpResponse = response as? HTTPURLResponse,
              (200..<400).contains(httpResponse.statusCode) else {
            completion(.failure(URLError(.badServerResponse)))
            return
        }
        completion(.success(data))
    }
    task.resume()
}
```

```swift
@frozen public enum Result<Success, Failure> where Failure : Error {

    /// A success, storing a `Success` value.
    case success(Success)

    /// A failure, storing a `Failure` value.
    case failure(Failure)
```

```
func login(username: String, password : String, completion: @escaping
    (Result<String,AuthenticationError>) -> Void) {
```

```
    LoginManager().login(username: username, password: password) { result in
        switch result {
        case .success(let token):
            print(token)
            defaults.setValue(token, forKey: "jsonwebtoken")
            DispatchQueue.main.async {
                self.isAuthenticated = true
            }
            LoginManager().getAllAccounts(token) { result in
                switch result {
                case .success(let html) :
                    DispatchQueue.main.async {
                        print("성공 \(html)")
                        if html == "User Content." {
                            print("로그인 성공")
                            self.isAuthenticated = true
                        }else{
                            print("등록되지 않은 계정입니다")
                        }
                    }
                case .failure(let error):
                    print(error.localizedDescription)
                }
            }
        case .failure(let error):
            print("실패, \(error.localizedDescription)")
        }
```

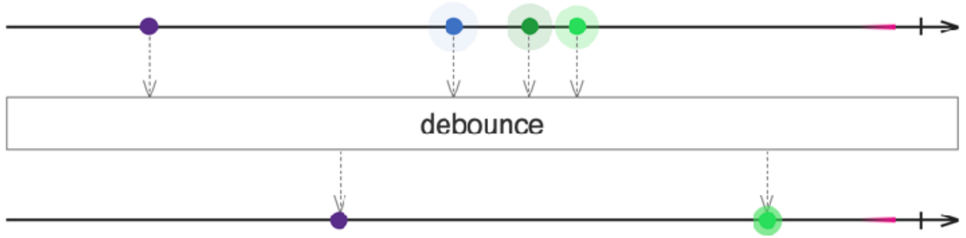
```
func processImageData2a(completionBlock: (_ result: Image?, _ error: Error?) -> Void) {
    loadWebResource("dataprofile.txt") { dataResource, error in
        guard let dataResource = dataResource else {
            completionBlock(nil, error)
            return
        }
        loadWebResource("imagedata.dat") { imageResource, error in
            guard let imageResource = imageResource else {
                completionBlock(nil, error)
                return
            }
            decodeImage(dataResource, imageResource) { imageTmp, error in
                guard let imageTmp = imageTmp else {
                    completionBlock(nil, error)
                    return
                }
                dewarpAndCleanupImage(imageTmp) { imageResult, error in
                    guard let imageResult = imageResult else {
                        completionBlock(nil, error)
                        return
                    }
                    completionBlock(imageResult)
```

클로저의 클로저의 클로저의 …

콜백 함수의 연속
-> 가독성이 떨어지고, 에러핸들링이 복잡해진다.

Combine

- iOS 13 버전 부터 나온 애플의 퍼스트파티 비동기 이벤트 처리 프레임워크?!
- 함수형 프로그래밍 방식으로 복잡한 비동기 흐름을 다룰 수 있게 됨
- 러닝커브가 높음 (Rx는 더 높음)




```
func perform(_ request: URLRequest) -> AnyPublisher<Data, Error> {
    URLSession.shared.dataTaskPublisher(for: request)
        .tryMap { data, response in
            guard let httpResponse = response as? HTTPURLResponse,
                  (200..<400).contains(httpResponse.statusCode) else {
                throw URLError(.badServerResponse)
            }
            return data
        }
        .eraseToAnyPublisher()
}
```

```
private extension SearchViewModel {
    func configureDataPipeline() {
        $query
            .dropFirst()
            .debounce(for: 0.5, scheduler: DispatchQueue.main)
            .removeDuplicates()
            .combineLatest($filter)
            .map { [loader] query, filter in
                loader.loadResults(
                    forQuery: query,
                    filter: filter
                )
                .asResult()
            }
            .switchToLatest()
            .receive(on: DispatchQueue.main)
            .assign(to: &$output)
    }
}
```

Async - Await

- 2021년 Swift 5.5 버전에서 등장함 (Task 구조체와 함께 iOS 13 부터 사용 가능)
- 이미 다른 언어들에서 흔히 사용되는 방식
- Combine에 비해 난이도가 낮고, 가독성도 좋음

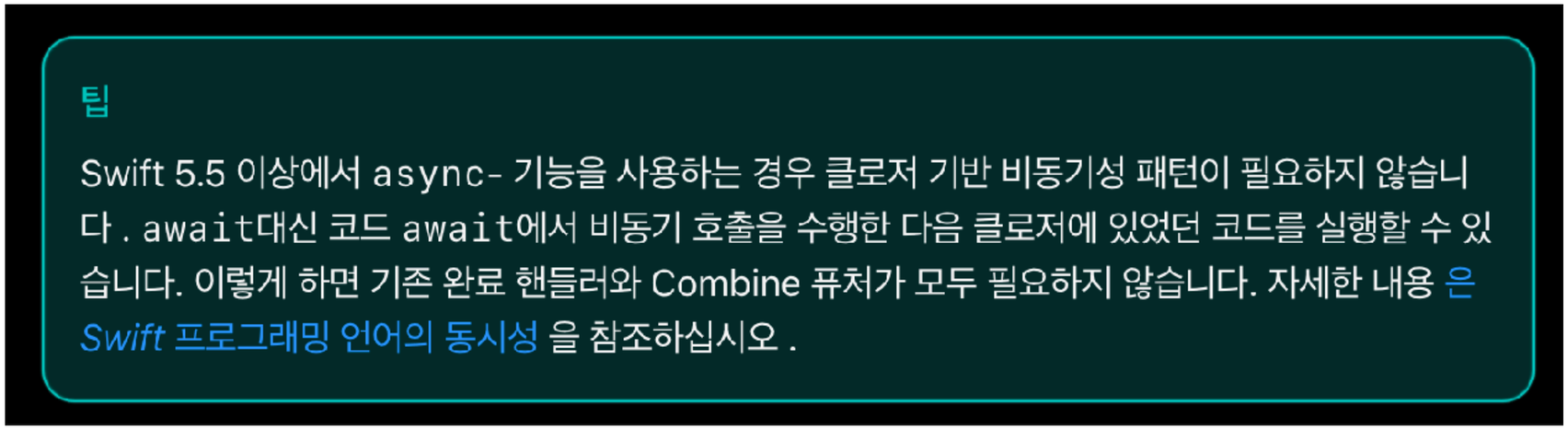
```
func perform(_ request: URLRequest) async throws -> Data {
    let (data, response) = try await URLSession.shared.data(for: request)
    guard let httpResponse = response as? HTTPURLResponse,
          (200..<400).contains(httpResponse.statusCode) else {
        throw URLError(.badServerResponse)
    }
    return data
}
```

```
func processImageData1(completionBlock: (_ result: Image) -> Void) {
    loadWebResource("dataprofile.txt") { dataResource in
        loadWebResource("imagedata.dat") { imageResource in
            decodeImage(dataResource, imageResource) { imageTmp in
                dewarpAndCleanupImage(imageTmp) { imageResult in
                    completionBlock(imageResult)
                }
            }
        }
    }
}

processImageData1 { image in
    display(image)
}
```

```
func loadWebResource(_ path: String) async throws -> Resource
func decodeImage(_ r1: Resource, _ r2: Resource) async throws -> Image
func dewarpAndCleanupImage(_ i : Image) async throws -> Image

func processImageData() async throws -> Image {
  let dataResource  = try await loadWebResource("dataprofile.txt")
  let imageResource = try await loadWebResource("imagedata.dat")
  let imageTmp      = try await decodeImage(dataResource, imageResource)
  let imageResult   = try await dewarpAndCleanupImage(imageTmp)
  return imageResult
}
```

컴바인 진짜 공부해야 할까?

- 컴바인은 Rx와 Completion Handler에 대한 방안으로 애플이 직접 만들어 나오게 됨
- 하지만 async - await 의 등장으로 애플 공식 문서에서도 Combine이 굳이 필요 없다고 설명하고 있습니다.
- Swift 공식 문서에 Concurrency 챕터에도 Combine 키워드는 하나도 검색되지 않고, 동시성 프로그래밍에서 async - await 을 강력하게 밀고 있습니다.
- 컴바인의 미래는 불확실하다고 생각합니다.



그래도 Combine 왜 공부하나?

우대사항

- 의존성 관리 툴 경험
- 배포 자동화/테스트 자동화 관련 경험
- Rx 경험
- MVVM 디자인 패턴에 대한 높은 이해도
- Code-based UI 경험

우대사항

- Swift UI 학습 및 경험 보유
- 컴퓨터 관련 전공자 또는 프로그래밍에 대한 지식 보유
- MVC, MVVM 등 디자인 패턴에 대한 높은 이해도
- RxSwift 사용경험이 있으신 분
- 배포 자동화 및 테스트 자동화 관련 경험
- Git을 통한 협업 및 코드 관리가 능숙하신 분
- 본인이 직접 iOS 앱을 개발하여꾸준히 운영한 경험이 있는 분
- 새로운 기술에 대한 호기심과 즐겁게 일하실 분

우대사항

- 서비스 아키텍쳐에 대해 관심 많으신 분 (RIBs, VIPER 등등)
- Instruments를 활용하여 성능 분석 및 디버깅 경험이 있으신 분
- Snapkit, FlexLayout 등 코드 기반 UI 구현에 불편함이 없으신 분
- 모듈화, 디자인 시스템 구축 경험이 있으신 분
- Combine, async/await 등 Rx 이외의 비동기 처리 방식에 관심 많으신 분
- Fastlane, Github Action 등 CI/CD에 관심 많으신 분
- 좋은 코드, 좋은 설계가 무엇인가 고민해보신 분

이런 분을 찾습니다!

SwiftUI(or UIKit), Combine(or RxSwift) 등 주요 iOS Framework에 대한 이해와 개발 경험이 있으신 분

Combine(or RxSwift)을 활용한 Reactive Application을 개발하는 분

자격요건

- 3년 이상 Native 개발 경험이 있으신 분
- SwiftUI, Combine(또는 RxSwift)에 대한 기본적인 이해가 있고 사용해 보신 분
- Restful API 연동 경험
- 협업에 익숙하고 커뮤니케이션이 원활한 분

우대사항

- SwiftUI 개발 경험
- Combine 및 RxSwift 등 비동기 프로그래밍 경험
- MVVM 아키텍처 이해 및 개발 경험
- 다양한 iOS 오픈소스 활용 경험
- iOS 앱 배포 및 상용화 경험
- 최신 개발 트렌드에 대한 관심이 많으신 분

※ 면접 전 과제 전형이 있을 수 있으며 진행 시 안내드립니다.

참고

- https://quickbirdstudios.com/blog/async-await-combine-closures/
- https://github.com/apple/swift-evolution/blob/main/proposals/0296-async-await.md#motivation-completion-handlers-are-suboptimal
- https://www.andyibanez.com/posts/modern-concurrency-in-swift-introduction/
- https://zeddios.tistory.com/1230