Mini Seminar

그 때의 내가 알았더라면 좋았을 이야기

김태현, SOPT 30기 iOS 파트장

CONTENTS

들어가며

어떤 이야기를 하면 좋을까요?

0. 주제 선정

Dependency Injection

SwiftUI

Architecture

Tuist

GCD

ARC

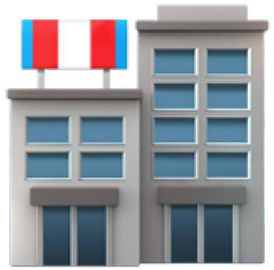
기술

취업 준비에 대한 이야기

현업에 대한 이야기

신입의 자세

...



취업

학습방법, 이야기

0. 주제 선정

내가 다시 돌아가면 이것부터 할거야

기록하는 삶은 무너지지 않는다

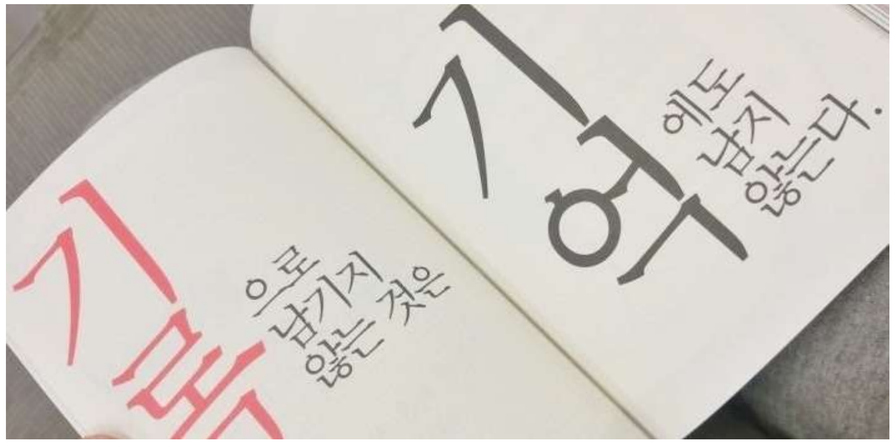
기록의 중요성

1. 기록이 왜 중요할까요?

노션 깃허브 메모장 블로그 노트

여러분은 기록을 잘하고 계신가요?

1. 기록이 왜 중요할까요?



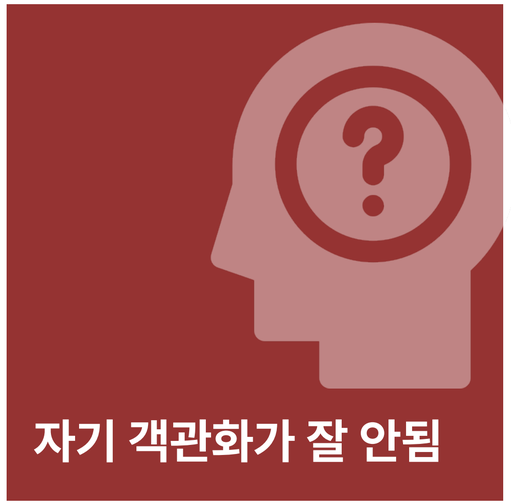
‘기록으로 남기지 않는 것은 기억에도 남지 않는다.’

사람의 기억력은 생각보다 좋지 않습니다.
1달만 지나더라도 우리가 이전에 학습한 내용은 머릿속에서 지워지게 되죠.
조금 더 과장해서 우리는 1주만 지나더라도 대부분의 내용을 기억하지 못합니다.

1. 기록이 왜 중요할까요?

자소서 쓸 때
쓸 내용이 기억 안남

시간을 비효율적으로
사용하게 됨



자기 객관화가 잘 안됨

나만의 것(ex. 실력)을
만들기 어려움

학습 단계에서의 기록

기록하는 삶은 무너지지 않는다

1. 학습 단계에서의 기록

1. 학습할 때, 어떤 형태로든 **기록**하는 습관을 들이자
    a. 리스트를 한눈에 살펴볼 수 있는 수단 : 노션, 깃허브, 메모장, 블로그 등
    b. 정리하는 것도 중요하지만 찾아보기 쉬워야 한다.

1. 학습 단계에서의 기록

- 개념 정리
- 키워드 정리
  - 키워드에 따른 레퍼런스 url 마킹
- TIL(Today I Learned) 정리
- 블로그에 깊이 파고 든 내용 정리
- 하나의 주제를 몇 챕터에 걸쳐서 공부하고 정리
- 스터디나 자체 기술 컨퍼런스를 열어서 발표자료의 형태로 정리
- ...

1. 학습 단계에서의 기록

- 키워드, 주제를 많이 아카이빙 하는 것부터 시작해보자
- 몰랐던 부분은 반드시 메모해놓고, 나 또는 다른 사람에게 설명한다고 생각하고 정리해보자
- 개발하면서 순간 순간 떠올랐던 생각, 아이디어를 정리해보자

회고/정리 단계에서의 기록

기록하는 삶은 무너지지 않는다

32기 Mini Seminar

1. 회고/정리 단계에서의 기록

✨ 가치 있는 기록

안 쓰는 것 < 단순 카피, 스크랩, 북마크 < **나의 생각과 고민/문제 정리**

1. 회고/정리 단계에서의 기록

- 생각 정리란 고민해봤던 내용
- 회고(- 개발자라면 프로젝트 자체에 대한 회고 뿐만 아니라 기술적인 회고도 중요함)
- **문제 정리(- 꼭 어려운 문제가 아니더라도 간단하게 기록해놓기)**
  - 문제 정의
  - 가설, 가정
  - 아이디어 도출
  - 해결방안 적용
  - 결과(어떻게 해결할 수 있었는지 / 해결을 못했다면 어디까지 시도해봤는지)
  - 개선점 / 회고

1. 회고/정리 단계에서의 기록

- 생각 정리란 고민해봤던 내용
- 회고(- 개발자라면 프로젝트 자체에 대한 회고 뿐만 아니라 기술적인 회고도 중요함)
- **문제 정리(- 꼭 어려운 문제가 아니더라도 간단하게 기록해놓기)**
  - 문제 정의 / ex. 로그인이 실패한다.
  - 가설, 가정 / ex. 네트워크 에러, 토큰 이슈, 서버 에러, 코드 누락
  - 아이디어 도출 / ex. 가장 유력한 것부터 해결을 시도한다.
  - 해결방안 적용 / ex. Postman - Data model 형식이 다름, Api 수정 요청 ...
  - 결과(어떻게 해결할 수 있었는지 / 해결을 못했다면 어디까지 시도해봤는지)
  - 개선점 / 회고

1. 회고/정리 단계에서의 기록

- 생각 정리란 고
- 회고(- 개발자
- **문제 정리(- 꼭**
  - 문제 정의 /
  - 가설, 가정 /
  - 아이디어 도
  - 해결방안 적
  - 결과(어떻게
  - 개선점 / 회

회고도 중요함)

요청 ...

해봤는지)

문제 상황

로그아웃 이후에 재로그인시 데이터를 로드해오지 못하는 문제 발생

가정

1. 토큰 문제일 것이다.
   - [x] 토큰이 갱신되지 않았을 것이다.
   - [ ] 회원탈퇴가 정상적으로 이뤄지지 않아서 토큰이 꼬였을 것이다.
2. 네트워크 문제일 것이다.
3. 서버 문제일 것이다.

가정 확인

토큰 문제

우선 토큰 문제의 가능성이 있어보여 프로젝트를 켜고 디버깅을 해보았다. 로그를 자세히 살펴보니, 로그인 이후에 accesstoken은 잘 넘어오는데 다른 api 요청 시에 header에 담기지 않는 것이었다. 그렇다면 여기서 token을 정상적으로 저장하지 못한다는 문제를 확인할 수 있다.

문제 재정의

토큰을 정상적으로 저장하지 못하는 문제

가정

1. 가장 최근에 Reponse data에서 accesstoken 변수명을 accessToken으로 바꾼 것이 문제이다.

설계, 코드 작성 단계에서의 기록

기록하는 삶은 무너지지 않는다

1. 설계, 코드 작성 단계에서의 기록

생각보다 많은 사람들이
설계하는 작업을 하지 않습니다

1. 설계, 코드 작성 단계에서의 기록

1. 설계 단계에서도 기록, 메모하는 습관을 들이면 좋습니다.
    a. UI 설계 -> 데이터 설계
    b. Layer, Architecture, Module, Components에 대한 설계 / 다이어그램

1. 설계, 코드 작성 단계에서의 기록 - UI

1. 설계, 코드 작성 단계에서의 기록 - UI

1. 설계, 코드 작성 단계에서의 기록 - UI

이미지 + 라벨이 반복되는 형태네?
또는 그냥 전체를 하나의 이미지로 봐도 되겠다.

어 그리고 나머지 친구들도 UI 구성이 동일하니까
반복해서 사용해도 되겠다!
(뒤에서 배울 스택뷰 또는 컬렉션뷰 이용 가능)

클릭했을 때 주소 설정하는 창이 나와야 하니까
라벨이 아니라 버튼을 써야겠네?

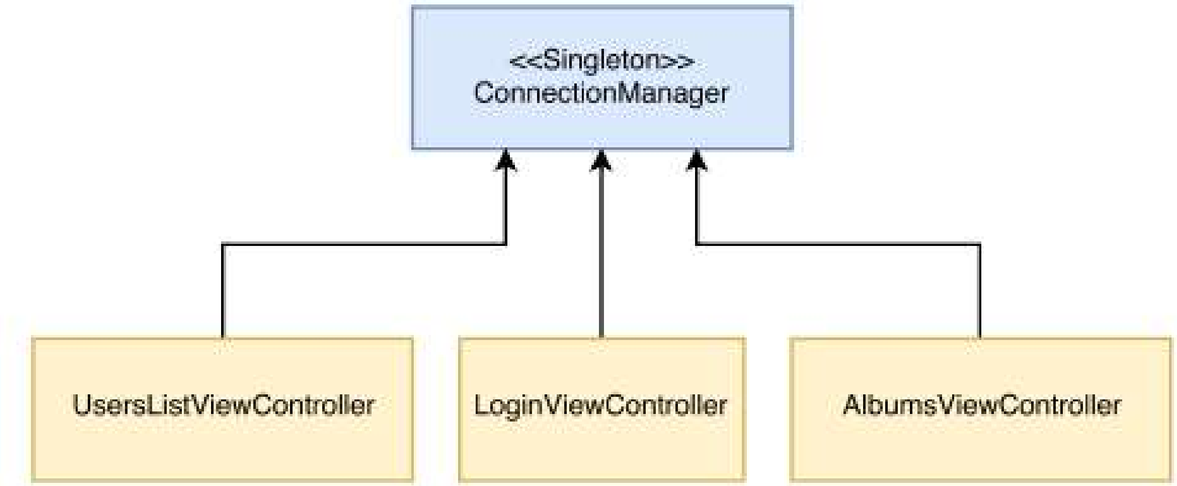
터치 영역은 화살표 버튼 영역만이 아니라
텍스트 있는 영역까지로 잡아야 겠다!

종이나 패드를 이용해서 스케치하는 작업을 해보는거예요.
저는 패드를 이용해서 스케치를 많이 해보는 편입니다.

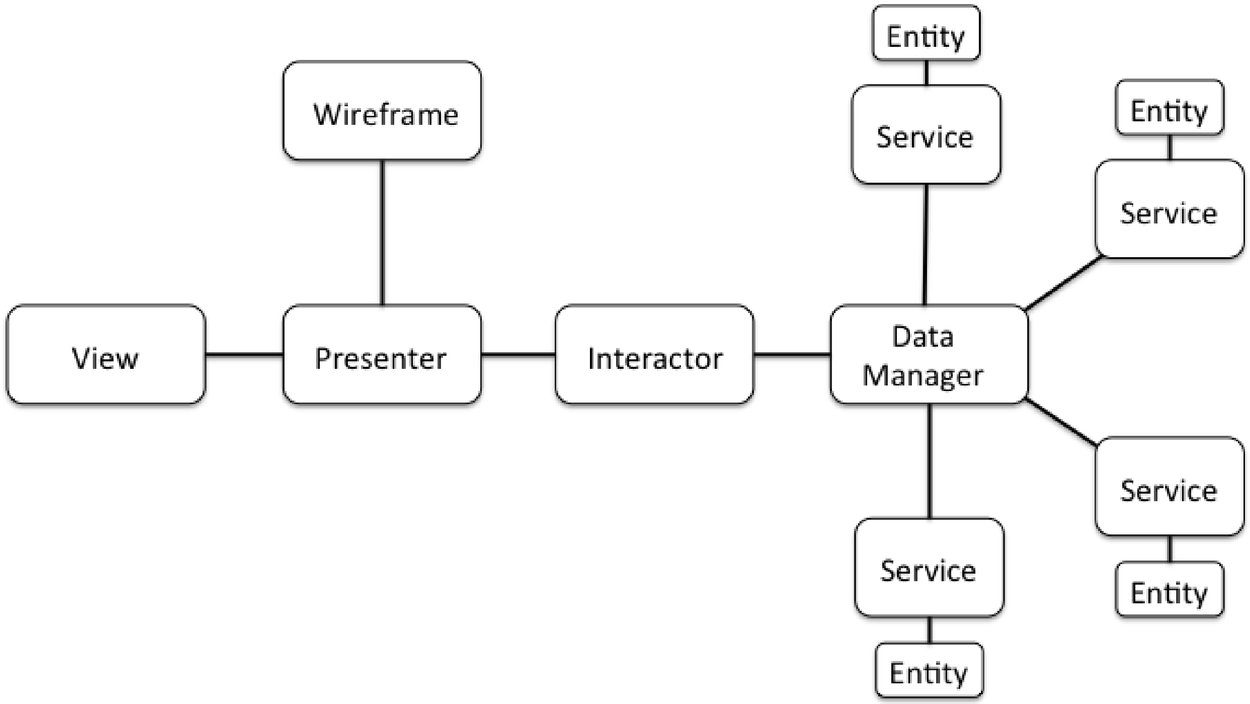
1. 설계, 코드 작성 단계에서의 기록 - UI

거창하게 할 필요 없습니다.
간단하게 몇 글자만 적어놔도 됩니다

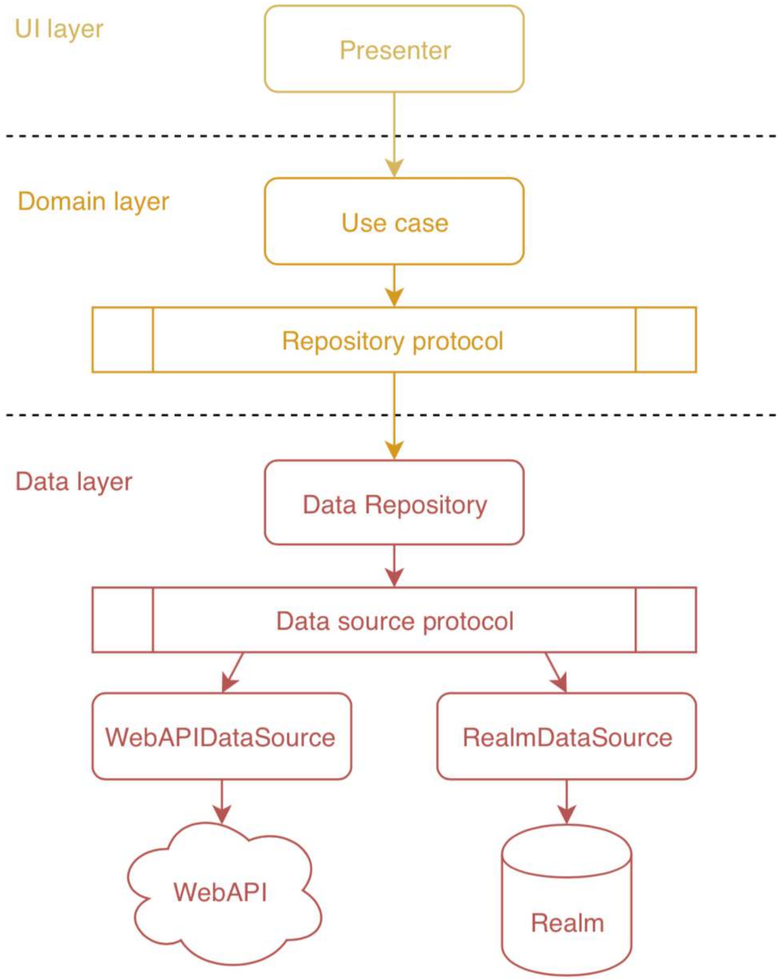
1. 설계, 코드 작성 단계에서의 기록 - 다이어그램



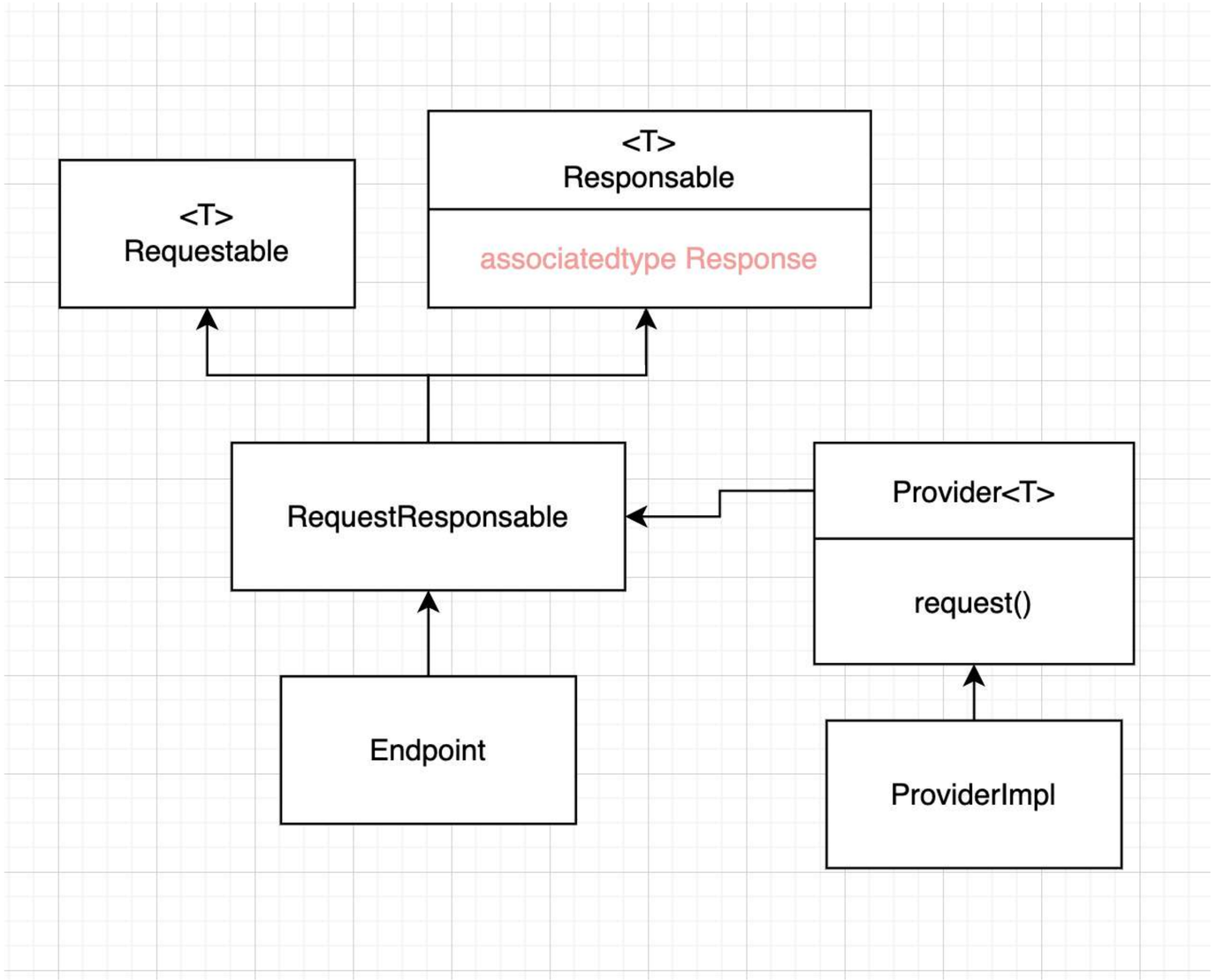
1. 설계, 코드 작성 단계에서의 기록 - 다이어그램



1. 설계, 코드 작성 단계에서의 기록 - 다이어그램



1. 설계, 코드 작성 단계에서의 기록 - 다이어그램



다이어그램 그리기가 어려울 수 있어요

1. 설계, 코드 작성 단계에서의 기록

- 객체 지향 프로그래밍에 대한 학습을 진행하고, 최대한 이해해보도록 한다.
- UML, 클래스 다이어그램 요소에 대해서 간단하게 살펴본다.
- 작은 기능 단위부터 연습해본다. (로그인, 회원가입, ..., 또는 계산기, 타이머 등등)

기초를 탄탄히 하자

그래야 확장과 응용이 가능하다

32기 Mini Seminar

기초라고 한다면 언어에 대한 이해

2. 기초를 탄탄히 하자

- 객체지향프로그래밍에 대한 특징을 정확히 알고 계신가요?
- 객체지향프로그래밍의 SOLID 원칙을 이해하고 적용할 수 있으신가요?
- 객체간의 관계를 그림으로 표현할 수 있나요?

- 클로저를 정확히 이해하고 계신가요? 적재적소에 활용할 수 있나요?
- 인스턴스 프로퍼티, 타입 프로퍼티의 차이를 아시나요? 언제 사용하는 것이 적절한가요?
- 객체간의 관계를 그림으로 표현할 수 있나요?
- swift.org, developer.apple.com에서 학습한 내용을 찾아서 복습했나요?

세미나를 듣고 난 뒤

2. 기초를 탄탄히 하자

1. 키워드를 몇 개 뽑아본다.
2. 누군가에게 설명할 수 있을 정도로 확실히 이해하고 있나 확인해본다.
3. 공식문서를 찾아서 읽어본다. (정리까지 할 수 있으면 좋다.)
4. 공식문서를 읽다가 모르는 키워드가 또 나올 것이다.
5. 그럼 또 키워드를 몇 개 정리한다.
6. 위의 과정을 반복한다.

이렇게만 해도 저는 실력이 늘 수 있다고 생각합니다

그런데 나는 왜?

2. 기초를 탄탄히 하자

- 근데 왜 안할까?
  - 방법을 몰라서?
  - 어려워서?
  - 시간이 없어서?

- 근데 왜 안할까?
  - 하기 귀찮아서
  - 귀찮아서
  - 다음에 할래..ㅎ

→ **이것이 아닐까 싶어요...**

생각하면서 코드를 작성하자

생각해라 휴먼. 그래야 사고력이 확장된다

32기 Mini Seminar

3. 생각하면서 코드를 작성하자

1. API Design Guides
    a. 네이밍, 코드 스타일, 컨벤션 작성할때에도 생각하자
    b. 가독성을 높이도록 하자
2. 각 객체(ViewController, Data, Class ...)의 책임과 역할을 생각하자.
    a. 메소드를 작은 단위로 쪼개보자.
    b. 클래스를 최소의 기능, 역할 단위로 쪼개보자.
3. 기타
    a. 프로젝트 또는 뷰에서 재사용될 부분(UI View, Class ..)이 있는지 찾아보자.
        i. 재사용 함으로써 더 복잡해지는 것은 아닌지 고민해보자.

3. 생각하면서 코드를 작성하자

1. API Design Guides
    a. 네이밍, 코드 스
    b. 가독성을 높이
2. 각 객체(ViewCon
    a. 메소드를 작은
    b. 클래스를 최소
3. 기타
    a. 프로젝트 또는
        i. 재사용 함

자.

는지 찾아보자.

Naming

Promote Clear Usage

- **Include all the words needed to avoid ambiguity** for a person reading code where the name is used.

  ▼ LESS DETAIL

  For example, consider a method that removes the element at a given position within a collection.

```
extension List {
  public mutating func remove(at position: Index) -> Element
}
employees.remove(at: x)
```

  If we were to omit the word `at` from the method signature, it could imply to the reader that the method searches for and removes an element equal to `x`, rather than using `x` to indicate the position of the element to remove.

```
employees.remove(x) // unclear: are we removing x?
```

- **Omit needless words.** Every word in a name should convey salient information at the use site.

  ▶ MORE DETAIL

내가 알고 있는 것에만 갇혀있지 말자

4. 내가 알고 있는 것에만 갇혀있지 말자

왜 나는 실력이 늘지를 않을까?
왜 나는 지금 정체되어 있을까?

4. 내가 알고 있는 것에만 갇혀있지 말자

친구 코드 블로그 세미나 책

처음 학습했던 내용에 갇혀있어서
그럴 수도 있습니다.

4. 내가 알고 있는 것에만 갇혀있지 말자

앞에서 이야기했던 것처럼
(기초 + 생각)을 다시 해 볼 필요가 있어요.

4. 내가 알고 있는 것에만 갇혀있지 말자

1. 내가 이 코드를 생각하고 짰을까?
2. 지금 코드가 최선이라고 보는가?
3. 객체 간의 역할을 분리해보려고 노력했는가?
4. 객체 간의 역할과 책임이 명확한가?
5. UI를 쪼개보려고 노력했는가?
6. API 가이드라인을 지키려고 했는가?
7. 공식문서도 꼼꼼히 다 읽어보고 학습했는가?
8. 정말 알아야 하는 내용을 알고 있는가?
    a. GCD(Thread), ARC, Networking

만약 다 했는데도 그렇다면

→ **디자인 패턴 이론 학습**

→ **오픈 소스 뜯어보기**

Alamofire, Moya, Kingfisher ...

맺음말

5. 맺음말

완벽주의의 가면을 쓴 게으름을 경계하자

5. 맺음말

어떤 개발자가 되고 싶은지
계속 생각해보세요

기초가 탄탄한 개발자　　재사용, 모듈화를 중요시하는 개발자　　근거 있는 코드를 쓰는 개발자

사용자 경험을 중요시하는 개발자　　성능 개선을 좋아하는 개발자

5. 맺음말

그리고 그 모습을 위해
역으로 근거를 계속 쌓아가보세요

5. 맺음말

여러분의 길을 응원하겠습니다.
다음에 좋은 기회로 다시 만났으면 좋겠습니다 !