오늘 OT 의 목차

- **1. 채용시장 현황 + 2. 예비개발자 현황** : 현실 직시
- **3. 개발자 꼭 해야겠습니까?**
- **4. 채용 전략** : 마음 굳게 먹었다면 개발자 채용 변화의 역사 파악
- **5. 커리큘럼** : 그래서 ASAC 은 여러분에게 무엇을 제공할것인가
- **6. 완전한 학습을 위한 3 요소** : 여러분들이 지켜야할 최소 3 요소
- **7. 수업의 방향성** : 그래서 ASAC 은 여러분의 일주일을 이렇게 제시합니다

근거

- 쿠팡 웹 개발자 (백엔드 / 프론트엔드) 5년
- 스타트업 Technical Lead (CTO) 로 채용 및 인사 담당 1년
- 부트캠프 항해99 수강생 '기술' 멘토링 2개 기수
- 부트캠프 위코드 수료생 '기술' 멘토링 1개 기수
- 부트캠프 코드스테이츠 수료생 '취업' 멘토링
- ASAC 웹 풀스택 메인 강사 2개 기수 + 이번 기수
- 프라임닷 : 개발자 취업 컨설팅

그럼 이제 OT 시작하겠습니다

언제든지 질문이 있다면 손을 들고 질문해주세요
(직접 질문이 힘드시면 아래 Slido 링크에서 질문을 올려주세요)

#3884659

OT 구성

OT 는 다음과 같은 흐름으로 구성됩니다

우리가 참여할 개발자 채용시장이 어떤지 한번 알아봅시다

1. 개발자 채용시장 현황

현재 개발자 채용시장 실태 : 여러분들 망했어요 (진지하게)

1. 개발자 수요 증가 시기 = 과거 : **개발자 갑질 시대**
2. 개발자 공급 급증 시기
3. 개발자 수요 급감 시기 = 현재 : **채용자 갑질 시대**

여러분들과 같은 예비 개발자들이 현재 어떤 상황에 놓여있는지 한번 알아봅시다

2. 예비개발자 현황

너무 가볍게 생각하는 학생들 / 불쌍한 학생들

- 대다수의 학생들의 실력은 **"절대적"** 미달 수준
  - 개발자붐에 따라 대거 유입된 미달자들
- 일부의 학생들의 실력은 **"상대적"** 미달 수준
  - 채용 경쟁의 강도 = 채용 기준의 정도
- 부트캠프의 추락

3. 개발자 꼭 해야겠습니까?

이런 극한의 상황에서도 꼭 개발자를 하고싶나요?

- 개발자를 하고싶은 이유가 무엇인가?

개발자가 되기로 마음을 굳게 먹었다면 어떻게 채용 전략을 가져가야할까?

4. 채용 전략

매년 발전해온 채용 프로세스

1. **전공자** : 컴퓨터공학, 과학 출신 채용
2. **실무형 비전공자** : 프로젝트 혹은 인턴 경험자 선호
3. **코딩 테스트**
4. **프로젝트 코딩 과제 및 페어 코딩** (전화 인터뷰)
5. **전공자** (회귀) : 컴퓨터공학, 과학 출신 선호

개발자가 되기 위한 최소 준비사항

- **웹 기반, 네트워크 지식** : 직무 지식을 쌓기위한 근간 지식 (베이스)
- **직무에 맞는 기술에 대한 지식** : 프론트엔드 / 백엔드에 맞는 지식
- **개인/팀 프로젝트 혹은 공모전** : 실무 능력을 알아볼 수 있는 유일한 방법
- **코딩 테스트** (최근엔 사실상 필수) : 채용 비용 효율
- **인턴 경험** (선택) : 실무 검증

앞선 최소 준비사항에 맞춰 ASAC 에서 여러분들께 무엇을 제공할지

5. 커리큘럼

채용 경쟁의 강도에 따라 채용 기준의 정도가 상향평준화 되어
여러분들이 알아야할 지식의 양이 과거대비 정상적이진 않다는것 **다시 한번 유념**

- 기반, 직무 지식 : **수업 + 복습 + 질문**
  - 웹 개요 → 프론트엔드 → 백엔드
- 팀 프로젝트 : **과제**
- 인턴에 유사한 경험 : **기업협업 프로젝트**

적어도 이걸 지키지 않고서는 취업 생각하지 말것

6. 완전한 학습을 위한 3 요소

100명이 넘는 사람들을 가르치고 멘토링했던 경험을 돌이켜보면
아래 3 요소를 제대로 수행하는자들만 성공

성공하는건 어렵지 않다. 단지, 여러분들이 안할뿐

- A. 수업
- B. 복습
- C. 질문

앞선 A. 수업 + B. 복습 + C. 질문 을 어떻게 조합할것인가

7. 수업의 방향성

수업에는 2가지 타입이 존재

- Type 1 : **수업 80%** + 복습 20% (소화)
  - 한개의 주제로 3개월 (대학교 등 일반적인 커리큘럼)
- Type 2 : 수업 50% + **복습 50% (소화 + 보완)**
  - 다수의 주제로 3개월 (단기완성형 커리큘럼)

OT 상세

이제 하나씩 살펴보겠습니다.

우리가 참여할
채용시장이 어떤지 한번 알아봅시다

여러분들 망했어요 (진지하게)

1. 개발자 수요 증가 시기 = 과거 : **개발자 갑질 시대**
    - 약 2010년부터 개발자 처우가 서서히 상승하면서 개발자 수요가 증가
        - IT 기반 B2C 플랫폼 기업, 스타트업의 성장 (네이버, 카카오, 쿠팡)
        - 플랫폼 스타트업 성공 사례로 “개발자 채용 = 플랫폼 = 투자 유치” 공식
        - **개발자 임금 인플레이션**
            - 저금리 시기 활발한 투자금 → 개발자 채용 증가
            - 치열한 개발자 유치 경쟁에 따른 신입 개발자들에게까지 동반이익

2. 개발자 공급 급증 시기

- 여러 산업군에서 개발자 수요가 빨리 증가하자, 개발자 공급이 시급한 문제
  - **국비교육 및 부트캠프 양산**
    - 산업에서 개발자에 대한 수요 + 높은 초봉으로 인한 학생들의 수요

3. 개발자 수요 급감 시기 = 현재 : **채용자 갑질 시대**
    - 고금리 시기에 감소한 투자금 → 개발자 채용 감소
    - 고금리 시기에 저조한 산업계 → 산업이 예상한 개발자 수요와 현실의 괴리
        - 높게 책정됐던 개발자 예상 수요와 현재 개발자 실제 수요와의 불일치
        - 높게 책정됐던 개발자 예상 수요에 맞춘 공급 정책, 산업의 실패
            - **공급이 너무 미친듯이 쏟아져나와, 취업 경쟁 강도가 매우 높아짐**
            - **개발자 임금 인플레이션 원상복귀 중** (버블 꺼지고있는 상황)

여러분들과 같은 예비 개발자들이
현재 어떤 상황에 놓여있는지 한번 알아봅시다

너무 가볍게 생각하는 학생들

- **대다수의 학생들의 실력은 "절대적" 미달 수준**
  - 무엇이든지 '붐'에는 다수의 미달자가 존재한다.
    - 개발자가 좋다는 이유만으로 어떠한 고려도 없이 개발자를 지원한 자들
    - 개발자가 되기위한 최소한의 노력도 하지 않는 자들
      - 고연봉은 받고싶지만, 죽을만큼의 공부는 하기싫어
  - 대학교때 팀 프로젝트의 악몽을 떠올려보세요
    - 여러분이 기업이라면 그런 사람들에게 월급을 줘야한다 생각해보세요

- **제대로된 개발자 찾기 이전에, 절대적 미달자를 걸러내는게 모든 채용의 미션**
  - 개발자만큼 차별없는 직군은 없다 = 정말, 누구나 개발자를 할 수 있다.
    - 저 개발자에요 (구글)
    - 저 개발자에요 (외주업체)
    - 저 개발자에요 (프리랜서)
    - 저 개발자에요 (초등학생)
  - 연봉의 최소값과 최대값이 가장 극단적인 직군이 아닐까
  - 여기 ASAC 만 하더라도 여러분 모두 똑같은 **예비 개발자**로 수료를 마칠것

불쌍한 학생들

- **일부의 학생들의 실력은 "상대적" 미달 수준**
  - **채용 경쟁의 강도 = 채용 기준의 정도**
    - 채용 경쟁 강도가 낮은 과거엔 부족함이 있어도 채용 후 교육으로 개선
      - 2010년 정도만 하더라도 라떼에는 개발자는 각광받지 못했던 직업
    - 현재 개발자 채용 경쟁 강도가 높은 상황에서는 올라운더여야 피채용
      - 요즘 신입 개발자가 3년차 개발자보다 더 뛰어날수도 있음
      - 주변 선임 개발자들 : 요즘 취업시장이라면 우린 개발자 못했을걸

- **부트캠프의 추락**
  - 개발자 양성이라는 좋은 의도로 시작
  - 부족한 학생에게도 기회를 주기위한 공평한 교육과 수익을 위해 대거모집
    - 저급 수강생의 낮은 학습 능력 + 부트캠프의 떨어지는 강의질 및 케어
      - **저급 개발자 양산**으로 부트캠프에 대한 현업에서 부정적인 시선
      - 괜히 잘하는 학생들이 부트캠프에 한데묶여 인정을 못받는 상황

진지하게, 고연봉의 개발자가 된다는건 생각보다, 생각보다 더 힘듭니다.
겁을 주는게 아니라 심적인 안전벨트를 제가 대신 매드리는겁니다.

이렇게까지 해야하나? 라고 생각하는 그 시점을 넘어야할겁니다.

이런 극한의 상황에서도 꼭 개발자를 하고싶나요?

개발자를 하고싶은 이유가 무엇인가?

- 하지말라는게 아니라, 적어도 우리가 무엇이든 그걸 하는 이유는 명확하길
- 내적 동기가 있다면 개발자가 되는 고된 길에서 무너지지않고 버텨낼 수 있음
- 개발자는 끊임없이 공부해야하는 직군 + 끊임없는 호기심 + 협업능력
  - 어떤 이에겐 고되지만, 어떤 이에게는 즐거운 경험
    - 공부를 싫어한다 → 개발자 하지마세요
    - 복잡한걸 싫어한다 → 개발자 하지마세요
- 제가 개발자를 계속 하고있는 이유는 ... 공부를 좋아합니다

개발자가 되기로 마음을 굳게 먹었다면
우리가 어떻게 채용 전략을 가져가야할까?

매년 발전해온 채용 프로세스

1. **전공자** : 컴퓨터공학, 과학 출신 채용
    - 문제 : *컴공이라더니 왜 이렇게 못해?*
2. **실무형 비전공자** : 프로젝트 혹은 인턴 경험자 선호
    - 문제 : *찍어낸듯한 포트폴리오와 부족한 기술 면접 및 개발 실력*
3. **코딩 테스트** : 기본적인 코딩 작성 능력이 없는 사람 및 최소 노력 스크리닝
    - 문제 : *실무에서 사용하는 문법, 라이브러리, 프레임워크에 대한 검증 불가*

4. **프로젝트 코딩 과제 및 페어 코딩** (전화 인터뷰)
    - 채용 시 시간을 많이 사용해야하지만, 확실한 검증 방법
        - 채용에 진심인 회사에서 많이 활용
    - 좋은 회사인지 알아보는 방법 : 채용 프로세스에 얼마나 진심인가
5. **전공자** (회귀) : 컴퓨터공학, 과학 출신 선호
    - 결국 CS 기반이 탄탄해야 성장 기대치 및 성장 속도가 빠름
        - 비전공자도 물론 취업 전후로 계속해서 CS 보완하면 가능
        - 물론, 전공자 중에서도 물경력처럼 물전공이 존재

자 그렇다면, 개발자가 되기 위한 최소 준비사항은 무엇일까

- 지식
  - **웹 기반, 네트워크 지식** : 직무 지식을 쌓기위한 근간 지식 (베이스)
  - **직무에 맞는 기술에 대한 지식** : 프론트엔드 / 백엔드에 맞는 지식
- 실무
  - **개인/팀 프로젝트 혹은 공모전** : 실무 능력을 알아볼 수 있는 유일한 방법
  - **코딩 테스트** (최근엔 사실상 필수) : 채용비용 효율
  - **인턴 경험** (선택) : 실무 검증

수업을 어떻게 진행할것이고
여러분에게 어떤 아웃풋을 기대하는가

채용 경쟁의 강도에 따라 채용 기준의 정도가 상향평준화 되어
여러분들이 알아야할 지식의 양이 과거대비 정상적이진 않다는것 **다시 한번 유념**

- 지식
  - 기반, 직무 지식 : **수업 + 복습 + 질문**
    1. 웹 개요
    2. 프론트엔드
    3. 백엔드
- 실무
  - 팀 프로젝트 : **과제**
  - 인턴에 유사한 경험 : **기업협업 프로젝트**

수업 난이도

03 ~ 08 레벨로 구성

- 개발 지식 레벨을 00 에서 10 으로 나누어보자면
  - 00 ~ 02 : 1년차 개발자 혹은 부트캠프 수료생도 여러분께 강의 가능
    - 기초 문법을 다루지 않습니다.
    - 단, 기초 문법 강의를 과제로 드려서 따라오실수 있도록 배려합니다.
  - 09 ~ 10 : 3년차가 들어도 모르는 내용

수업 내용

진짜로, 웹 풀스택

- 웹 풀스택 강의의 "웹 풀스택"이 단지 마케팅 용어가 되지않도록
  - **양두구육하지 않도록 저는 4개월로 웹 풀스택을 진짜 가르칠겁니다.**
    - 3기, 4기 학생들의 일상이 처참해지는만큼, 개발 지식량은 급증합니다.
    - ~~4개월 뒤 아무 부트캠프 수료자 붙잡고 말싸움해도 웬만하면 이깁니다.~~

왜, 웹 풀스택?

- 가혹한 커리큘럼과 학생들이 힘들어함에도 '웹 풀스택' 강의를 유지하는 이유
  - 채용 경쟁 강도에 따라 높아진 기준엔 이것이 기본값이라 생각합니다.
  - 단기완성이라는 부트캠프의 정의와 목적에 더 부합한다고 생각합니다.

1. 웹 개요

- **웹의 등장 및 웹의 구성** : 웹 페이지, 웹 브라우저, 웹 서버, DNS, CDN
- **웹 브라우저 성능 개선 및 웹 서버 부하 완화** : SEO, Metrics, Proxy
- **웹 저장소와 웹 보안의 이해** : HTTP Cookie, Storage, Session, CORS
- **웹 페이지 생성 및 서빙 방식** : SSG, CSR, SSR 동작 방식 이해
- **Git 기초에서 심화 및 실무 활용** : Git 명령어, Staging 상태에 대한 이해
- **개발과 배포 그 사이에서의 기술과 절차** : Docker, Git-flow 와 Zone

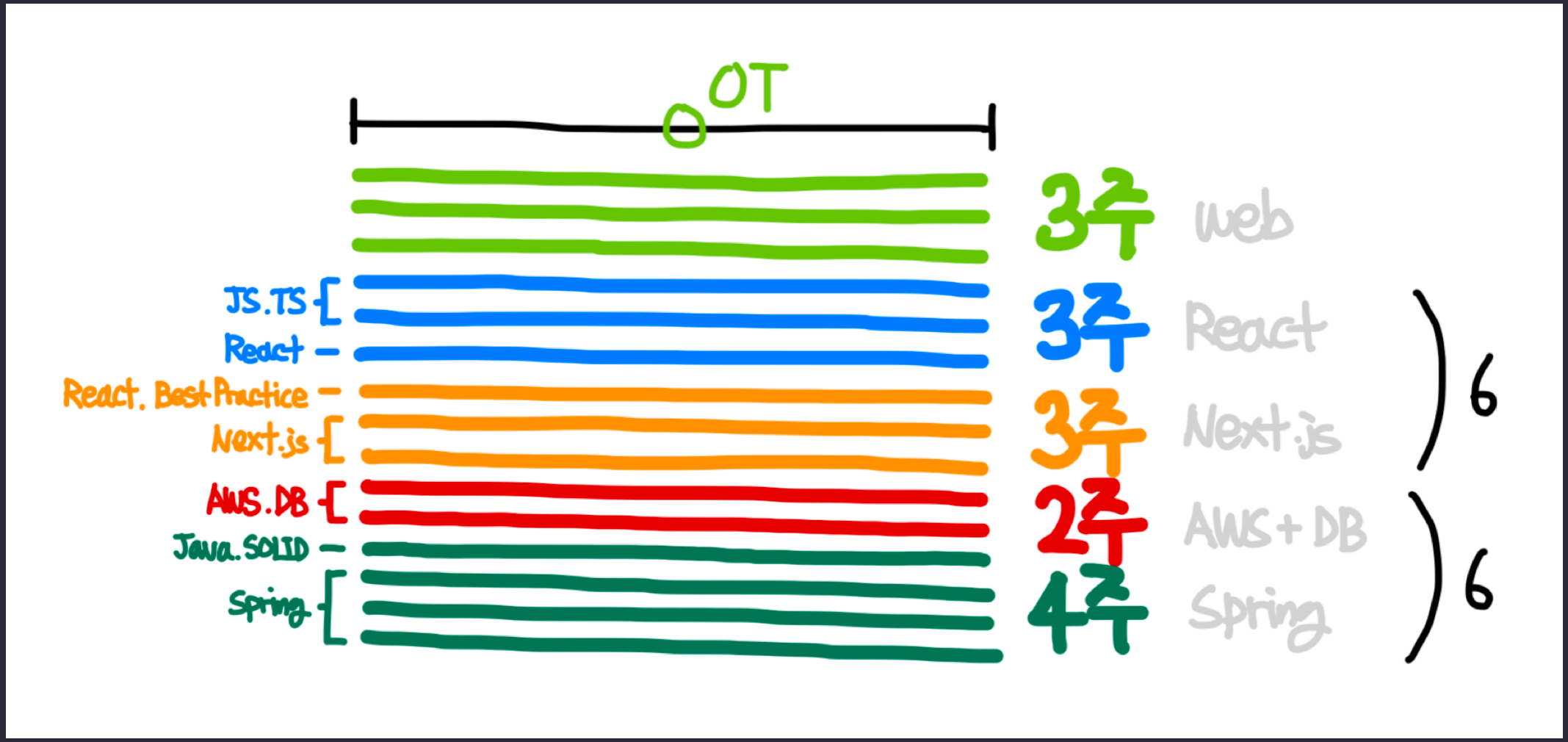
2. 프론트엔드

- **JavaScript 문법 및 엔진 동작 원리** : Lexical Scope, ES6
- **TypeScript 기초 문법 및 활용** : interface 와 type 용례
- **React 동작 원리 및 Hook 종류와 활용** : VDOM, 기본 Hook 들 살펴보기
- **React 서드파티 라이브러리와 Best Practices** : 상태관리, 비동기, 비제어
- **Next.js 원리 및 Routing, Caching 이해하기** : Prerender, Caching
- **Styled-Component, 반응형 웹 및 인터렉티브 웹 개념**

3. 백엔드

- **AWS 클라우드와 네트워크 구성** : VPC 와 Subnet 구성, AWS 서비스 종류
- **데이터베이스와 동시성 제어, 트랜잭션** : RDBMS 와 NoSQL 차이, ERD
- **Java 기본 문법 및 JVM 구성** : JVM Heap, GC, Class, Interface
- **Java 확장 문법** : Functional Interface, Stream, Lombok
- **디자인 패턴 및 객체지향 설계원칙 SOLID**
- **Spring Boot 의 등장 및 특장점** : Gradle, Spring MVC 및 3-Layered
- **Spring AOP, Data JPA, Security 구성 및 동작 원리에 따른 활용**
- **예외처리와 로깅처리** : 예외처리(Advice, Controller, Bean Validation)

5. 커리큘럼


OT
JS.TS
React
React. Best Practice
Next.js
AWS. DB
Java. SOLID
Spring
3주
web
3주
React
3주
Next.js
6
2주
AWS + DB
4주
Spring
6

안내 : 4월 15일 (월) 이전

- 이번 5기의 경우, 메인 강사인 제가 4기 수업을 진행중임에 따라
  - 타 강사가 프론트엔드(Javascript, React) 수업을 먼저 진행하시고,
  - 제가 4월 15일 (월) 부터 합류합니다.
    - 합류하자마자, 앞선 **프론트엔드 수업에 대한 Recap** 을 할겁니다
    - 그 다음 **웹 개요** 수업을 진행합니다.

안내 : 4월 15일 (월) 이후

- 제 수업은 1. 웹 개요 먼저 시작하는데
  - **모든 커리큘럼 중 가장 중요한 부분이자 시작**
  - 근간 지식 : 기초 공사가 잘되어야 그 위에 프론트엔드 / 백엔드 건설 가능
    - 모든 기술엔 이유가 있다 "왜" 를 알아야 "어떻게" 를 배울 수 있다
  - 여기서 못따라온다면 앞으로 프론트엔드 / 백엔드 이해가 고통스러울것

이것들을 제대로 학습하기위해 어떤 요소들이 필요한가

- **A. 수업** : 이론적 내용을 **수동적으로 섭취**
- **B. 복습** : 수업 내용을 **능동적으로** 내것으로 **소화** 및 깊이 **보완**
  - 기술 블로그 or 노션 정리 : 매주 수업한 내용을 복습과 함께 정리
  - 그룹 스터디 : 복습은 혼자가 아닌 둘 이상이 비용에 있어 효율적
- **C. 질문** : 시간의 제약 및 다양한 청자를 고려한 표준화 된 강의로 인한 한계
  - 강사 머릿속의 수많은 경험, 지혜를 모두 빼먹기 위해선 질문이 필수
    - 강의 내용은 제 지식의 30%
  - 질문을 통해 복습의 밀도를 높임으로써, 더 명확하게 이해할 수 있다

100명이 넘는 사람들을 가르치고 멘토링했던 경험을 돌이켜보면
아래 3 요소를 제대로 수행하는자들만 성공

성공하는건 어렵지 않다. 단지, 여러분들이 안할뿐

공부를 못하는 사람과 잘하는 사람의 차이

- 공부를 못하는 사람 : **수업**에만 의존
  - 공부 다했니? → 네, 다했어요.
- 공부를 잘하는 사람 : **복습**과 **질문**에 많은 시간을 할애
  - 공부 다했니? → 아니요, 할게 너무 많아요.

특히, 여러분들이 개발 공부를 어려워하는 이유는 모르는 용어들이 많기 때문에 수업 중 / 수업 후 제게 많이 물어봐주셔야합니다.

당신은 어떤 사람이 될것인가?

ASAC 에서 A 수업 + B 복습 + C 질문 을 어떻게 조합했는지

수업에는 2가지 타입이 존재

- Type 1 : **수업 80%** + 학습 20% (소화)
  - 대학강의식 (정상) : 시간적 여유가 많고, 긴 텀을 두고 단일과목을 학습
    - 수업에서 모든것을 다루기에, 수업만 잘들어도 B+ 학점 취득
- Type 2 : 수업 50% + **학습 50% (소화 + 보완)**
  - 단기완성형 (비정상) : 시간적 여유 없이, 짧은 시간 내 많은것을 학습
    - 수업에서 최대한 핵심적인것들을 많이 다루고,
    - 각 세부 내용은 복습과 질문으로 보완하는것이 필수

Type 2 인 ASAC 과정을 어떻게 여러분들에게 전달할지

우리의 일주일은 다음과 같이 구성됩니다

- A. 수업 50%
  - **A.1. 수업** : 월요일 오프라인
  - **A.2. 수업 + 복기 + 과제 설명** : 목요일 오프라인

- B. 복습 50% (소화 + 보완)
  - **B.1. 과제** : 목요일 발행 → 다음주 월요일 제출 및 팀 내 과제 리뷰
  - **B.2. 기술 블로그 발행 + 토의** (그룹 스터디) : 월요일에서 → 수요일까지

- C. 질문
  - **C.0. 1:1 세션** : 화, 수요일 - 수업, 과제 질문 및 수업 피드백, 고민 상담
  - **C.1. 수업 질문** : 언제나
  - **C.2. 과제 질문** : 언제나

요일별로 정리해보면 다음과 같습니다.

- 월요일 : 오프라인
  - **A.1. 수업**
  - **B.1. (전주) 과제 제출 + (팀) 팀 내 과제 리뷰**
  - **B.2. 기술 블로그 작성** 시작

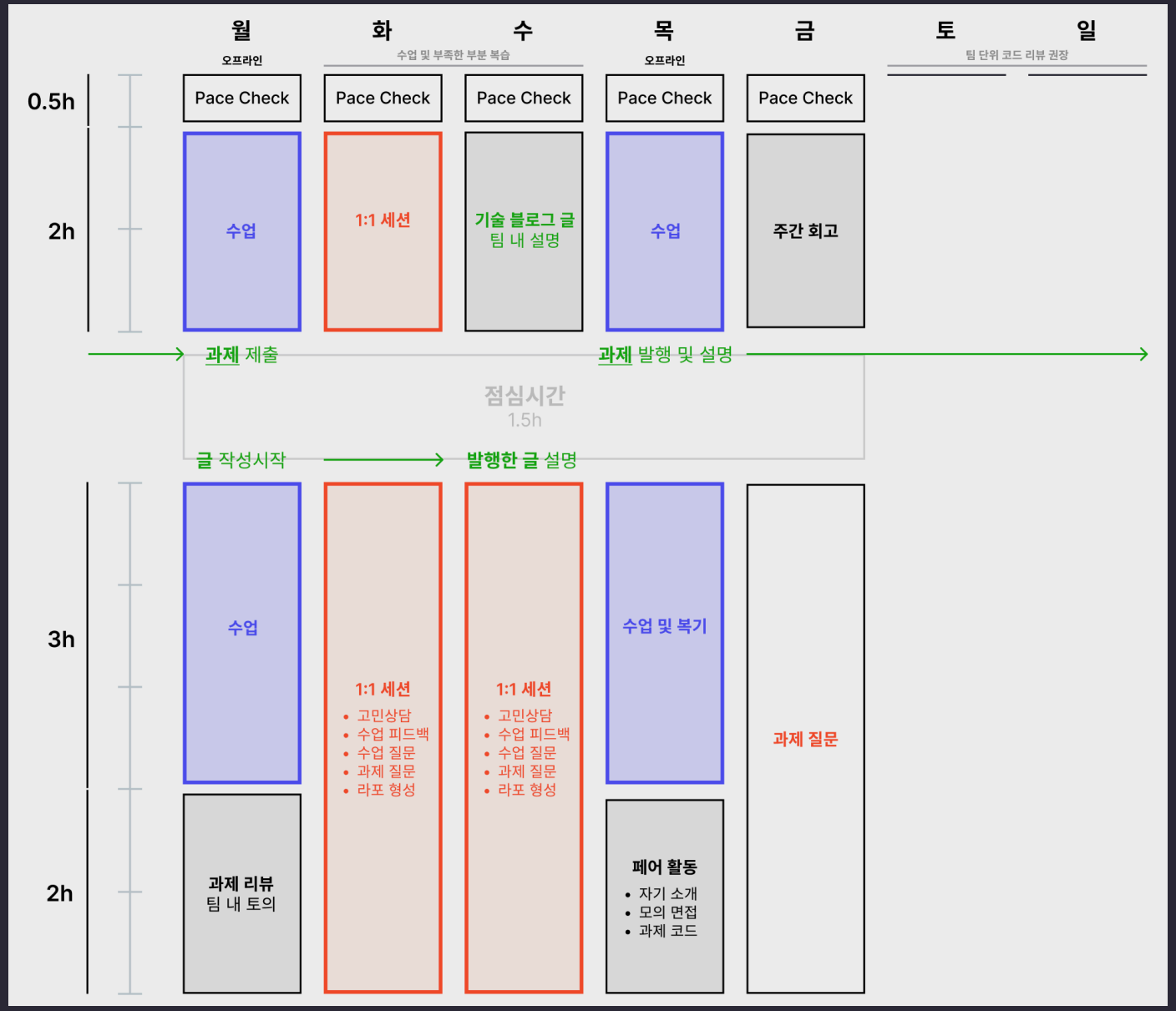
8. 일주일 구성

- 화요일
  - **C.0. 1:1 세션**
- 수요일
  - **B.2. 기술 블로그 토의** (그룹 스터디)
  - **C.0. 1:1 세션** (화요일에 이어)

- 목요일 : 오프라인
  - **A.2. 수업 + 복기 + 과제 설명**
  - **B.1. 과제 발행**
  - (팀) 페어 활동
    - 누군가에게 내 지식을 명확하게 설명 가능하도록 : 기술 면접 대비
    - 누군가에게 내 코드를 명확하게 설명 가능하도록 : 코드 설명 능력
      - 타인의 코드를 볼 줄 아는 능력을 기르도록

- 금요일
  - (팀) 주간 회고 : 자신의 현재 상태 파악 및 팀원과 함께 성장
  - **C.2. 과제 진행 및 질문**
- 토요일 + 일요일
  - 과제 진행

8. 일주일 구성


월
오프라인
화
수업 및 부족한 부분 복습
수
목
오프라인
금
토
일
팀 단위 코드 리뷰 권장
0.5h
Pace Check
Pace Check
Pace Check
Pace Check
Pace Check
2h
수업
1:1 세션
기술 블로그 글
팀 내 설명
수업
주간 회고
과제 제출
과제 발행 및 설명
점심시간
1.5h
글 작성시작
발행한 글 설명
3h
수업
1:1 세션
• 고민상담
• 수업 피드백
• 수업 질문
• 과제 질문
• 라포 형성
1:1 세션
• 고민상담
• 수업 피드백
• 수업 질문
• 과제 질문
• 라포 형성
수업 및 복기
과제 질문
2h
과제 리뷰
팀 내 토의
페어 활동
• 자기 소개
• 모의 면접
• 과제 코드

왜, 강사가 되었는지

- 유명 부트캠프들의 기술 및 취업 멘토로 활동, CTO 로 채용을 주도해본결과
  - 말로 형용할 수 없을 정도로 처참한 지원자들의 실력에 너무 충격
- 부트캠프 몸집은 불어나는데, 학생들이 그저 씹고 버려지는 껌 같다고 생각
  - 천만원 수강료가 한두푼도 아니고, 국비 아닌 사비로 내는 사람도 존재
- 과연 부트캠프, 시스템이 문제일까? 학생들이 쉽게 생각하는것이 문제일까?
  - 적어도 간절한 학생들에겐, 제대로 된 부족함 없는 교육을 하자

커리큘럼을 만들며

- 기술 면접이 어려워졌다는건 우리가 인정해야할 현실이다.
  - 필요한 지식의 수준이 높기때문에 그 기준에 맞춰 커리큘럼을 만들었다.
  - 단, 웹 풀스택인만큼 프론트엔드/백엔드 각자의 면접 기준에 맞추어서
    - 주 직무가 아닌 강의에 대해선 수준이 너무 높다고 판단할 수 있다.
- 취업용 지식이 아닌 실제로 취업 후 개발할때 필요한 모든 기반 지식을 담았다
  - 시니어 개발자들과도 대화를 할 수 있을 수준

4개월 뒤

웬만한 대학교 1년보다 더 많은것들을 배울것

- 수업을 제대로 듣고, 복습했다면 웬만한 기술면접, 개발자들과 대화 가능
- 성인이 된 후 미친듯이 공부해본 경험, 습관, 시간 활용법들은
  - 앞으로 어떤 공부를 할때도 적용할 수 있는 자산이 되어있을것

4개월 동안

내가 정말로 개발자가 되고싶은건지, 판단할 수 있는 좋은 기회가 될것

- "개발자가 될 수 있는지"가 아닌 "되고싶은건지" 임에 유의
- 아무리해도 개발자 되는길이 까마득할 수는 있지만, 절망하지마라.
  - 4개월은 개발자가 되기엔 너무 짧은 시간인것은 현실적으로 맞고
  - 정말 되고싶다면, 뜻이있는곳엔 길이 있을것이다.

저는 여러분들에게 책임있는 자세로 교육하기위해 노력할 것

어떤 피드백이든 모두에 대해 열려있고
필요하다면 어떤 정책이든, 방법론이든 수정 혹은 적용할 수 있음
여러분들이 받아갈 교육의 양과 질을 높힐수만 있다면

여러분들은 적어도 자신의 인생을 책임지는 자세로 교육받는 모습을 보여주시길