### 내용 요약

part 1(byukim): **C언어에 대한 생각**, 함수형 언어와 절차적 언어, **좋은 코드는 읽기 쉬운 코드**, 천재 개발자들, **책의 중요성(용어를 정확히 아는 것),** **동료평가(감정적으로 받아들이지 말 것)**

part 2(jwon): **학장을 맡게 된 계기**, 나를 팔아야한다. 성장하기 위한 방법(박람회 참여), 42의 불친절함에 대한 질문, {**아우터서클의 필요성, 학생들이 회사를 모른다(현실적인 프로젝**트)}, **동료학습이 중요하다(**혼자 할 거면 여기 왜 왔냐)

part 3(yechoi): **소프트웨어 시장에서의 이직, 시간 관리(하는 방법, 하고 났을 때의 이점)**, **커뮤니티의 중요성(새로운 것을 배우는 데 있어서)**, **창업지원 계획 및 창업 유망 분야**, 나중 계획/취미

### 내용 구성

**42 - ( 6월 25일 금요일 저녁 9시 )**

학장을 맡게 된 계기 - jwon

동료평가(part 1: 감정적으로 받아들이지 말 것/ 효율적인 이유, part 2: 숨어서 하지 말고 끌어내서 해야한다) - byukim, jwon

아우터서클의 필요성, 현실적인 프로젝트 블라블라 - jwon

커뮤니티에 관한 생각

창업지원 계획 및 창업 유망 분야

**일반적 - (6월 28일 월요일 저녁 21시)**

C언어에 대한 생각 - byukim

책의 중요성 - byukim

좋은 코드는 읽기 쉬운 코드(New) - byukim

내가 할 수 있는걸 써먹을 수 있는 방법을 찾아야한다. 자신을 팔아라(박람회) - jwon

소프트웨어 시장에서의 이직

시간관리

Part1(피드백 전 초안)

“새로운 교육 시스템을 만든다는 사실은 매우 중요하고 의미 있는 일이잖아요. 개발자들은 모두 같은 고민을 해요.’ 소프트웨어를 체계적으로 배웠으면 더 빨리 성장했을 텐데’라고 생각하죠. 배움에 대해 아쉬움이 있는 거예요. 만렙 끝판왕인 멘토님들과 함께 시스템을 설계하면서 그 고민을 해결할 기회가 생기는 거예요. 훌륭한 시스템을 만들어서 지도자의 희생 없이 큰 규모의 교육을 가능케 하는 것도 중요하다고 생각하고 있어요. 그 시스템화를 이노베이션 아카데미에서 하는 거죠.”

“본과정에 대한 필요성을 잘 느끼지 못하는 경우가 있다면, 지금 배우는 게 무엇을 목표로 하는지 안 가르쳐준 탓이라고 생각해요. C언어를 배운다고 네카라쿠배에서 원하는 표면적인 역량이 키워지지 않잖아요. 그런데 막상 면접을 가보면 내가 무엇을 놓치고 있었는지 알게 돼요. 언어 이면의 것들이죠. 그런 것들을 배우라고 본과정이 만들어져있어요.”

“혼자 프로젝트하고 온라인 강의나 잘 정리된 자료를 보면서 공부해도 시간을 들이면 할 수 있어요. 근데 그건 빨리 가는 방법이 전혀 아니죠. 그리고 동료 학습을 해서 얻을 수 있는 것들에 비하면 많은 것을 놓칠 가능성이 높아요. 옛날에 제가 배울 때는 혼자 학습하는 게 비교적 용서가 됐어요. 책에 있는 거 진도 나가면 되는 거고 언어도 많지 않았고요. 하지만 지금은 전혀 그렇지 않거든요. 배워야 할 게 정말 많아요. 그래서 같이 배우는 게 훨씬 효율적인 거죠.”

“개발 경험이 충분히 쌓이고 나면, 결국 문제에 대한 해결방법이 어느 정도 정해져 있다는 걸 알게 돼요. 그런데 그 이전까지는 정해진 답에 얽매이지 않고 다양한 관점으로 바라보는 게 중요해요. 해결법을 가르쳐주지 않고 과제를 하라고 하면 서로 전혀 다른 방법으로 문제를 해결해요. 그걸 서로 이야기를 하면 다른 사람의 관점으로 배워볼 수 있죠. 이걸 손쉽게 할 수 있는 것이 동료학습의 큰 장점이에요.”

“카뎃들의 동료평가를 보면서 한 가지 아쉬운 점이 있어요. 우리는 평가가 감정적인 행위가 아니라는 믿음을 가져야 해요. 그래서 치팅이 의심될 때도 엄격하게 평가하고, 그 과정에서 감정을 안 쌓는 것이 중요해요. 협업할 때 꼭 필요하거든요. 회사에서 코드 리뷰를 할 때도 내 코드가 난도질을 당할 수도 있지만, UFC 경기가 끝나면 다시 화해하는 것처럼 쿨하게 받아들여야 하는데 아직은 그런 훈련이 좀 덜 돼 있는 것 같아요. 실패를 감정적으로 받아들이면 실력으로 쌓이지 않아요. 그러니까 평가를 감정적으로 받아들이지 않는 훈련이 필요한 거죠.”

“많은 학생이 동아리에 대해 완전히 다른 방식으로 생각하고 있다는 걸 많이 느껴요. 동아리에 목적이 있어야 한다고 생각하죠. 그런데 커뮤니티라는 건 여러 사람의 좋은 면을 보고 배우는 곳이에요. 20명 규모의 동아리가 있다면 미션을 열심히 수행하는 사람은 네 명만 있으면 돼요. 나머지 16명은 놀아야 그 동아리가 유지가 돼요. 우리 운영팀장님은 별로 안 좋아할 얘기겠지만(웃음). 그러니까 커뮤니티는 사람이 모이는 게 먼저고 그다음이 ‘미션(mission)’인 거죠.”

“그래야 호기심이 숙제가 아닌 재미가 돼요. 내가 관심 있는 대상 그 주변부를 팔 때 재미가 생겨요. 관심 있는 대상을 ‘어떻게 달성할지, 어떻게 공부할지' 에 대한 일정이 꽉 짜여 있으면 피곤해서 인생을 살 수가 없어요. 완전히 새로운 걸 공부할 땐 여유를 가져야 해요. 조급하게 생각할 필요도 없어요. 지금 있는 시간을 많이 써서 공부하는 게 진짜 좋은 거예요.”

<사진>

“대학에서 창업지원단장을 했을 때의 경험으로 미뤄보면, 창업 잘하는 사람은 ‘실행력'이 정말 좋아요. 사람들이 아파하는 영역이라면 고민하지 않고 해결해주겠다고 주장하죠. 아직 해결 방법이 다 완성되지 않은 상태에서요. 그렇게 ‘해결해주겠다'고 소문내고 다니다가 누군가가 정말 해결해달라고 나타나면, 그 사람과 그 문제를 이야기하면서 답을 찾아요. 이런 게 창업하는 사람들의 기본적인 에티튜드죠.”

“창업하는 사람이 배워야 할 것은 ‘남이 돈을 지불하는 메커니즘'입니다. 창업 생각이 있는 학생들에게 ‘비 오는 날에 우산을 팔아봐라’라고 얘기해요. 알리 익스프레스에서 2불짜리 우산을 한 박스 사서 지하철역 앞에서 팔아요. 그럼 어떤 사람은 사고, 어떤 사람은 사지 않는다고요. 이런 고객의 패턴을 이해하지 못하면 장사라는 건 할 수 없는 거예요. 이런 감각을 몸으로 배워야 해요.”

“미국에서 괜찮은 창업 케이스가 많은 이유는 개발자들이 창업을 많이 하기 때문이에요. 개발자들이 소프트웨어 분야 회사가 필요한 솔루션을 만드는 경우가 많아요. 소프트웨어 회사들이 쓸 소프트웨어를 만드는 거죠. 회사 입장에선 이런 서비스에 당연히 비용을 지불합니다. 회사에서 필요한 서비스를 내부 인력으로 만든다고 했을 때 인건비가 1억이라고 합시다. 그런데 어떤 회사가 이런 솔루션을 2000만 원으로 제공하고 있어요. 그러면 개발자 안 뽑고 사서 쓰는 거죠. 기업이 슬랙 사용료로 1인당 7~8불씩 내잖아요. 2,000명이 사용한다고 하면 우리나라 돈으로 한 달 150만 원이 슬랙 비용으로 나가는 거예요. 그런데도 그 비용이 안 아까워요. 왜냐면 직접 만들었으면 비용이 훨씬 많이 들었을 테니까.”

“우리나라도 점점 이렇게 사람의 가치를 돈으로 환산하는 게 가능해지고 있어요. 많은 IT 기업들이 그런 유료 솔루션을 구매합니다. 소프트웨어 시장이 워낙 크기 때문에 이렇게 개발자들을 위한 서비스를 만들어서 파는 창업이 유망해요. 이런 창업은 대단히 창의적인 아이디어가 필요하지도 않아요. 회사에서 필요한 서비스가 있다고 하면 만들면 되는 거죠. 창업은 멀고 새로운 일이 아니라 ‘우리 주변에 있다’는 게 중요한 것 같아요.”

“아직 창업에 대한 현실적인 지원 계획은 없지만, 네트워크를 활용해 적극적으로 도와드릴 겁니다. 곧 있으면 자신의 프로젝트를 의미 있게 하는 카뎃들이 늘어날 거예요. 여러 기회를 경험하며 창업 아이템을 다룬다면, VC와 연결하고 데모데이도 진행하는 등 ‘진짜' 창업 아이템이 될 수 있도록 멘토님들과 함께 도와드릴 겁니다. 혹시 아이디어가 없으면 저한테 찾아오세요. 아이디어 드릴게요.”

Part 2

“가장 좋아하는 프로그래밍 언어는 C언어에요. C를 접한 이후로는 거의 완벽한 언어라고 생각하고 있어요. low-level에서 원하는 것을 다 할 수 있어서 좋아요. 우리나라는 C보다는 자바, 자바스크립트, Python 등이 대세인데, 요즘에는 언어와 라이브러리의 경계가 희미해지면서 점점 지저분해진다고 느껴져요. 반대로 C는 프로그래밍 언어가 필요한 기능에다가 실제 하드웨어를 고려한 포인터 같은 기능들만 추가해서 상대적으로 깨끗해요.”

“다른 언어들이 C에 비해 쉽다는 의견에는 동의를 못 하겠어요. 예로 들어보면 Python이 배우기 쉬운 이유는 C의 진입장벽인 포인터가 없고, 문법이 간단하다는 것 외에는 별거 없어요. 물론 파이썬은 자동으로 자료형을 정해줘서 편리하긴 해요. 근데 자료형이 엉키면 나중에 사고를 치게 되어있어요. 그걸 겪고 나면 자료형이 잘 정의된 C 같은 언어가 낫다고 생각하게 돼요. 자료형이 명시적으로 나타나니 읽기도 쉽고요. 그리고 진짜 문제는 로직에서 나오는 에러거든요. 이건 언어가 다르다고 해서 피할 수 있는 것도 아니고요.

“코드는 읽기 쉬운 코드가 최고예요. 코드는 나만을 위한 것이 아니라 모두를 위한 것이거든요. 오픈 소스든 회사의 프로그램이든 다른 사람이 나의 코드를 보고 리뷰를 하기 때문이죠. 요즘 언어들이 간단하지만, 추상적인 문법으로 생산성을 높이려고 하는데, 제 생각엔 아름답지 않은 모습 같아요. 코드는 읽는 사람이 고민하게 만들면 안 되거든요. 그래서 쓰기 쉬운 코드가 아니라 읽기 쉬운 코드를 지향해야 합니다. C언어가 좋은 이유도 그 이유고요.”

“좋은 개발자가 되려면 해결 방법을 찾고 설명하는 역량이 필수적이에요. 이때 설명할 때 쓰는 용어가 매우 중요해요. 소프트웨어를 배울 때 책으로 공부하면서 접했을 용어들이 있거든요? 그 용어들을 써서 다른 개발자들에게 설명해야 해요. 제가 42학생들한테 ‘문제를 풀 때 주어진 책을 공부해야 한다’고 많이 얘기해요. 책으로 공부하지 않으면 나중에 질문할 때 용어를 써서 설명할 수가 없어요.”

“예시를 몇 개 들어보면, 42에 웹서버를 만드는 과제가 있잖아요? 웹서버를 만들면 네트워크 프로토콜, 패킷 등을 알아야 하고 그런 용어들은 책에 다 쓰여 있어요. Shell을 만들려면 ‘프로세스가 무엇인지’, ‘어떻게 프로세스 통신이 일어나는지’, 그리고 ‘시스템 콜이 무엇인지’ 알아야 하는데, 책을 안 보면 모를 수밖에 없는 거죠. 프로그램에 대한 질문을 던졌을 때 단어조차 처음 들어봤다는 반응을 하면 천재가 아닌 이상 코드를 가져다 썼다는 인상을 줄 수밖에 없게 되죠. 그래서 선수들이 쓰는 용어를 배우기 위해 책을 공부해야 합니다.”

“아마추어와 프로의 차이는 ‘자신을 평가할 줄 아는 능력'이 있다는 거예요. 시간을 어떻게 쓰는지 평가하는 것도 그 일부죠. 학창 시절 방학이면 동그란 계획표를 만들어 하루의 일과를 채워 넣잖아요. 지금도 이런 걸 해야 한다고 봐요. 24시간을 어떻게 썼는지 매일 저녁 적어보세요. 당연히 처음에는 24시간이랑 안 맞아요. 시간을 낭비한 듯한 느낌이 들죠. 하지만 이 작업을 계속하다 보면 어느새 하루를 어떻게 썼는지 딱 맞출 수 있어요. 허투루 쓴 시간이 없어지는 거예요.”

“해야 할 일에 대한 소요 시간도 예측해보세요. ‘어떤 함수를 짤 때 몇 시간이 걸릴 것이다’, ‘어떤 기능을 구현하는 데 며칠 걸릴 것이다' 이런 식으로요. 일을 마친 뒤에는 실제 얼마나 걸렸는지 적어뒀던 예측 시간과 비교해보세요. 이 두 가지 모두 5분도 안 걸리는 간단한 일이에요. 그렇지만 반복하다 보면 나의 예상과 실제 결과가 그럭저럭 비슷해지고 나를 평가할 줄 아는 능력이 생겨요.”

“그 ‘끝판왕'은 내가 택해왔던 공부 방법을 바꿀 줄 알게 되는 겁니다. 운동선수들이 슬럼프에 빠졌다가 극복하고 난 뒤 하는 얘기가 있어요. ‘처음부터 다시 배웠다'고 하죠. 소프트웨어 분야도 마찬가지예요. 스스로 평가해서 예전에 해왔던 방식을 버릴 줄 아는 게 능력이에요.”

“뭔가를 진정성 있게 한다는 것은 그냥 시간을 많이 쓰는 걸 의미해요. 어른으로서의 부탁인데 젊은 시기에 공부에 시간을 정말 많이 썼으면 좋겠어요. 일주일에 적어도 50~60시간 정도는 코딩하는 데 써야 실력이 늘어요. 이 얘기를 굵은 폰트로 써주세요. 이 말은 굵은 폰트로 써달라고 얘기한 것까지 써주세요. 그 정도는 해야지 나중에 봤을 때 ‘어 코딩 좀 하네’ 그런 느낌이 나요. 어설프게 해서 이룰 수 있는 건 생각보다 별로 없어요.”

<사진>

“취업을 할 때는 내가 들어가고 싶은 회사에 대해 잘 알아야 해요. 예를 들어 이런 것들이요. '업의 본질이 무엇인지', '매출이 얼마인지', '소프트웨어를 얼마나 중요시하는 곳인지', '개발자가 성장을 할 수 있는 곳인지' 같은 것들이요. 하지만 이것들은 DART 공시에도 안 나와 있고 재무제표에도 공개되지 않는 내용이죠.”

“최근 들어 다시 활발히 열리고 있는 박람회에 가는 게 한 방법이에요. 가서 '이런 거 만들 때 소프트웨어 필요하지 않아요?', '뭐로 개발하세요?' 등 미친 척하고 질문해보는 거죠. 그곳에 있는 사람들은 온종일 질문을 고대하고 있기 때문에 영업비밀까지 이야기해 줄 수도 있단 말이에요. 그리고 비슷한 업종의 다른 회사 부스에 가서 방금 들은 이야기를 이용해 다시 질문해보는 거예요. '다른 곳에서는 이렇게 개발하는데 여기는 어떻게 하세요?'라고 이야기하면 엄청 있어 보이잖아요. 그리곤 '소프트웨어를 배웠고 인턴을 가고 싶은데 필요하면 연락해주세요' 하고 명함을 드리는 거죠. 이렇게 시장에서 내 능력을 사용할 기회를 찾는 건 굉장히 중요한 일입니다. 내 관심사를 계속 이야기하고 그런 영역에 있는 사람들을 만나는 노력을 해야 하는 거죠.”

“소프트웨어 시장은 분야 전체가 한 회사 같아요. 한 회사에서 2~3년 있으면서 배움에 한계가 느껴지면 다른 회사로 옮겨가면 돼요. 비록 회사에 잘 이끌어 주는 사람이 있고 후배들이 똘똘하다고 하더라도, 갈증이 생기면 이직해야 해요. 월급이 깎인다고 하더라도요. 회사를 옮기는 것에 대해 두려워하지 않는 게 중요해요. 프레임워크, 클라우드 등 같은 스택을 사용하더라도 다른 도메인의 데이터를 다루면 엄청난 챌린지로 느껴질 거예요. 도메인이 다르면 데이터가 만들어지고 처리되는 과정이 다 다르거든요. 이런 디테일을 경험하면 실력이 굉장히 높아질 겁니다.”