프로그래머의 배움

안희종 @ FRONT-ENDGAME

2019-06-22

반갑습니다

- 주로 웹 제품을 만듭니다
  - 스포카 (2017-01 - 2017-09)
  - 하이퍼커넥트 (2017-09 - 2018-04)
  - 비바리퍼블리카 (2018-04 -)
- https://ahnheejong.name
- twitter, GitHub @heejongahn

목차

- 무엇을 배울까: 변하는 세상 속에서
- 어떻게 배울까: 이론과 실습 사이의 핑퐁, 사람들은 틀린 말을 한다
- 왜 배울까: 나는 작지만 내가 할 수 있는 일이 있다

무엇을 배울까:
변하는 세상 속에서

코드는 빚이다

- 코드는 적게 짤수록 좋고, 코드를 짜지 않고도 할 수 있으면 더욱 좋다.
- 왜냐고? 소프트웨어는 가만 냅두면 녹이 슨다.
- 왜냐고? **소프트웨어는 세상에 대한 특정한 가정의 집합 위에 세워지기 때문**이다.

소프트웨어는 세상에 대한 가정 위에 세워진다

- 프로그래머가 인식하든 인식하지 않든 프로그램을 지탱하는 가정들이 존재한다
- 모든 가정은 나름의 방식으로 당시 세상의 모습을 반영한다
  - 컴퓨터에서 사용할 인코딩에는 알파벳을 포함해 128개의 문자만 들어가면 된다
  - 프로그램의 메모리 공간을 표현하기 위해서는 32비트면 충분하다
  - …
  - 자바스크립트는 브라우저에서 간단한 작업을 하는 데나 필요한 언어다.
    따라서 뭔가 잘못되더라도 에러를 뱉기보단 적당히 그럴듯한 결과를 내는게 낫다

세상은 끊임없이 변한다

- 세상이 변하면 가정은 깨지고, 그 가정에 기반해 세워진 프로그램 역시 영향을 받는다
  - 진공 속에 존재해 “한 번 짜면 그 이후로는 알아서 잘 돌아가는” 프로그램은 거의 없다
- **가정의 수명이 어떨지 판단하고, 그 정보를 코드에 반영하는 통찰이 필요하다**
  - 지금 들어온 요구사항이 한 달 후에도 바뀌지 않을 가능성은 얼마나 될까?
  - 만약 바뀐다면 어떤 부분이 바뀌고, 얼마나 바뀔까?
  - 나중에 변경사항이 들이닥칠 때 적은 노력으로도 대응하려면 코드를 어떻게 짜야 할까?
- + 판단이 얼마나 정확했는지, 다음엔 어떻게 더 잘 할지 꾸준히 회고해야 한다

세상은 끊임없이 변하지만...

- 나를 둘러싼 세상이 변할 때 **여전히 가치 있게 남을 지식**은 무엇일까?
- 오랜 기간 다양한 사용자 요구, 시대적 환경에 발맞춰 여전히 쓰이는 도구들
  - 변하는 세상 속에서 여전히 가치 있음을 증명한 제품(들)이 힌트를 갖고 있다
  - 실용적인 용도로 언어나 라이브러리 문서를 볼 때는 보통 '어떻게'를 생각한다
  - 학습을 위해 소스 코드나 API를 볼 때는 '왜'를 생각하자

왜 - 무엇이 - 살아남았을까?

- 왜 표준 라이브러리를 이렇게 디자인 했을까?
- 왜 이런 기능을 표준 라이브러리에 포함시켰을까? / 포함시키지 않았을까?
- 왜 이 API는 인자를 이런 순서로 받을까?
- 왜 이 정보는 인터페이스로 노출하고, 이 정보는 감췄을까?

Last but not least: 겸허함

- “프로그래머는, 그리고 오직 프로그래머만이 특별하다!”
  - **사실이 아닙니다**
- 혼자서, 프로그래밍 만으로 만들 수 있는 임팩트의 크기는 제한적이다
- 다른 직군에게 필요한 자질 대부분이 프로그래머에게도 똑같이 요구된다
  - 글과 말을 이용해 생각을 정돈, 표현, 설득하는 능력
  - 팀워크, 친절함, 배려: Don’t be an asshole
- 함께 일하고 싶은 동료가 되세요: 「Being a Multiplier」

요약: 변하는 세상 속에서

- 올바른 가정을 세우고, 어떤 가정이 오래 살아남을지 판단하는 능력을 기르자
- 세상이 변해도 가치가 사라지지 않을 지혜를 배우자
  - ‘어떻게’보다 ‘왜’를 고민하는 태도
- 무엇보다 **좋은 동료가 되자**

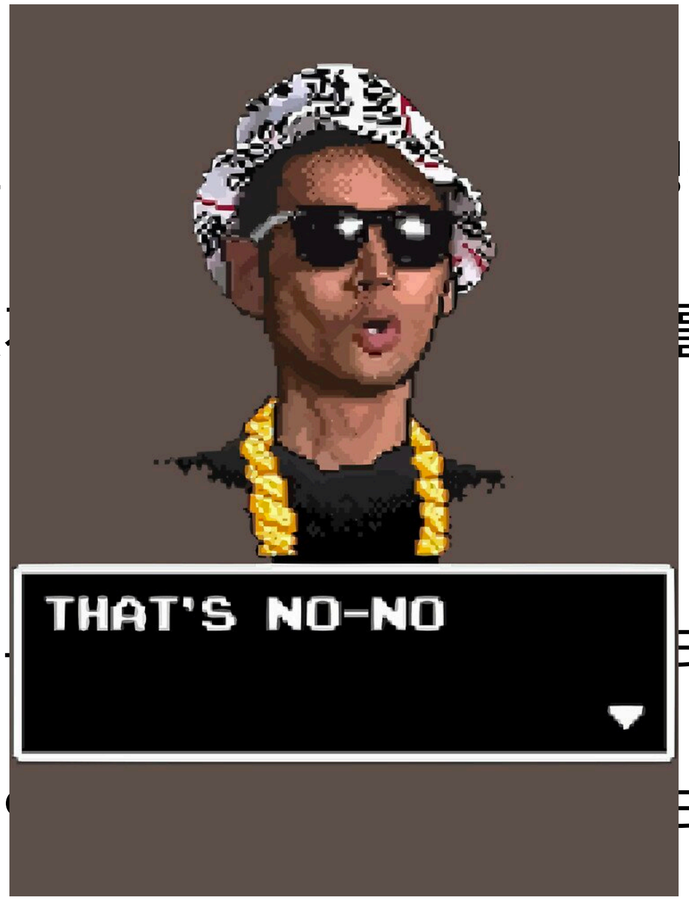
어떻게 배울까:
이론과 실습 사이의 핑퐁

새로운 도구를 배우는 두 태도

- 프로그래머 A:
  - 코드는 한 줄도 짜지 않고 처음부터 끝까지 스펙을 정독
  - 한참 후 스펙 읽기를 마쳤지만 간단한 프로그램 하나를 짜는데도 고생
- 프로그래머 B:
  - 기본 문법만 배운 후 문서는 쳐다보지도 않고 프로그래밍에 돌입
  - 문서를 한 번만 읽었어도 안 했을 고생을 헤쳐나가느라 엄청난 시간을 소요

새로운 도구를 배우는 두 태도

- 프로그래머 A:
  - 코드는 한 줄도 짜지 않고 독
  - 한참 후 스펙 읽기를 마쳤 를 짜는데도 고생
- 프로그래머 B:
  - 기본 문법만 배운 후 문서 래밍에 돌입
  - 문서를 한 번만 읽었어도 라 엄청난 시간을 소요

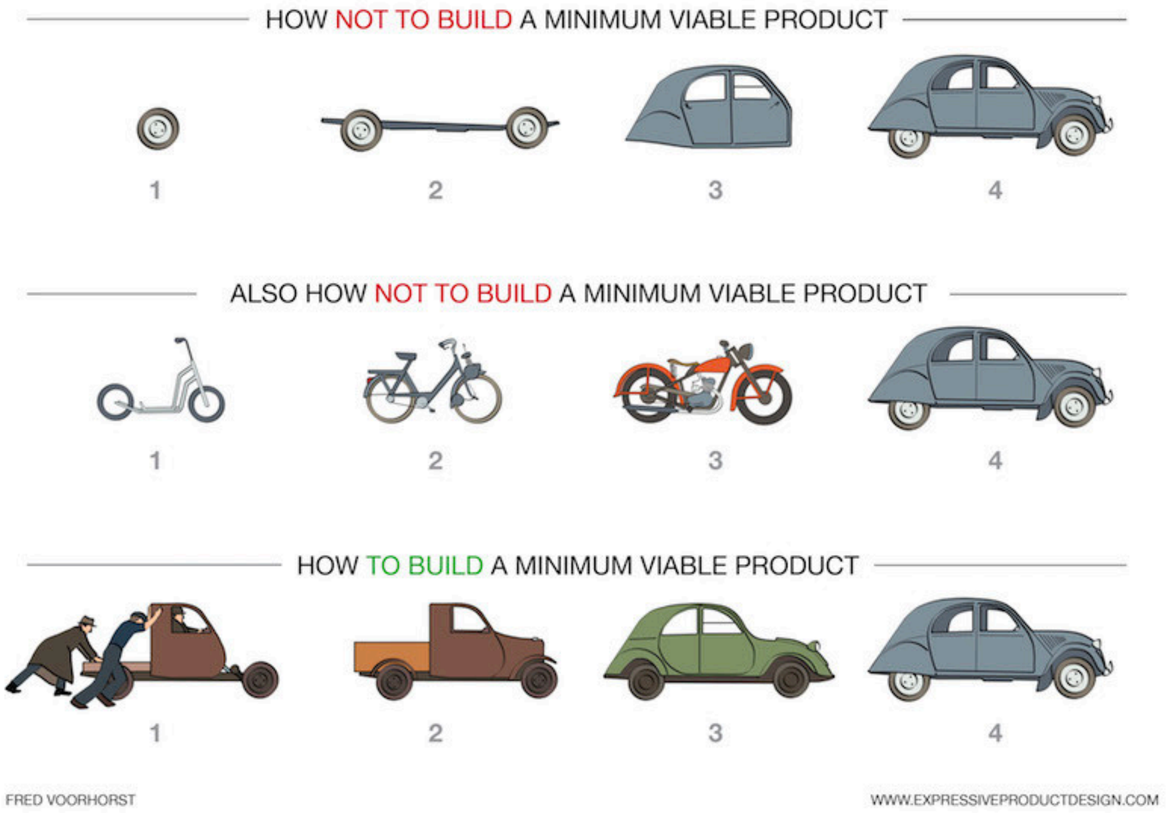


균형을 찾아서

- 이론만 파는 것도, 이론을 아예 보지 않는 것도 최선은 아니다
- 이론적인 배경 공부와 그렇게 쌓은 지식을 활용하는 실습, **두 상태를 빠르게 오가기**
- 주의: 배운 이론을 이용해 무언가를 구현했을 때, 결과물이 '**동작하는**' 상태여야 한다
  - 짧은 주기의 반복을 통한 빠른 피드백을 얻기 위해
  - 돌아올 수 있는 체크포인트를 만들기 위해
- 그런데 '동작한다'는 게 정확히 무슨 뜻인가요?
  - MVP (Minimum Viable Product)

MVP 잘 설정하기

- MVP(Minimum Viable Product): **가치를 전달할 수 있는** 가장 단순한 형태의 제품



https://www.linkedin.com/pulse/mvp-bike-car-fred-voorhorst/

MVP 잘 설정하기

- 예시: 전통적인 MVC 프레임워크를 사용해 어플리케이션을 작성할 때

Don’t:

1. 모델을 짜는 단계
2. 컨트롤러를 짜는 단계
3. 뷰를 짜는 단계

Do:

1. 하드코딩 된 데이터를 읽기만 가능한 단계
2. 데이터의 추가, 삭제, 수정을 구현하는 단계
3. 인증 로직을 추가하는 단계

만들면서 배워요

- 어떤 프로그램을 짤지 생각하면서 문서를 읽고,
  그 프로그램을 작성할 수 있겠다는 생각이 들 때 바로 코드를 짜기 시작하라
  - 좋은 타겟: 단어 세기 프로그램, TodoApp, 텍스트 기반 2048 게임 등
- 어떤 프로그램을 짤지 전혀 떠오르지 않는다면?
  - 그 기술을 정말 배워야 하는 것이 맞는지 다시 생각해보자

요약: 이론과 실습 사이의 핑퐁

- 기억해야 할 두 질문
  - 이 이론(언어, 라이브러리, 패턴)을 사용해 만들 수 있는
    가장 단순하되 실질적인 쓸모가 있는 기능이 뭘까?
  - 그 기능을 만들기 위해 알아야 하는
    최소한의 지식(언어 문법, 라이브러리 API, 패턴의 구성 요소)은 어느 정도일까?

어떻게 배울까:
사람들은 틀린 말을 한다

불편한 진실

- 사람들은 틀린 말을 (많이) 한다
  - 때로는 의도적으로, 때로는 의도치 않게
- 사람들이 틀린 말을 한다는 사실에 비하면 의도는 사실 별로 중요하지 않다
- 배움의 과정에서 사람들의 틀린 말에 덜 휘둘리기 위해선 어떻게 해야 할까?
  - 재해석의 단계 줄이기
  - 정말 사람을 믿어야 할 상황인지 묻기

재해석의 단계 줄이기

- ‘쉽게’ ‘풀어서’ 설명해주는 블로그 글, 유튜브 영상…
  - 저자의 재해석과 무엇을 말하고 무엇을 말하지 않을지에 대한 결정이 들어간다
  - 재해석의 단계가 깊어질수록 가장 사실에 가까운 정보로부터 멀어진다
- 무언가 배울 때 가장 근원이 되는 정보를 직접 찾아보는 습관을 들이자
  - 소스 코드, 언어·라이브러리 공식 문서, 어떤 개념을 처음으로 주창한 이의 논문, 글

정말 사람을 믿어야 할 상황인지 묻기

- 자동화할 수 있는 검증의 책무는 사람이 아닌 기계에게 맡긴다
- 사람의 ‘이 풀 리퀘스트는 기존의 기능을 깨먹지 않았습니다’라는 말을 믿는 대신:
  - 자동화된 회귀 테스트가 검증하게 하기
- 문서에 적힌 ‘이 함수를 이런 파라미터를 받습니다’라는 정보를 믿는 대신:
  - 정말 그런지, 잘못 사용 중인 곳은 없는지 타입 체커가 자동으로 검사하게 만들기

요약: 사람들은 틀린 말을 한다

- 더 견고한 프로그램을 위해서:
  - 스스로 검증하지 않은 정보를 함부로 믿지 않는다
  - 다른 프로그래머와 사용자를 올바르게 의심한다
- 프로그래머가 부주의하면 회사와 사용자가 비용을 치른다
- **최선을 바라되, 최악을 대비하기**

왜 배울까:
나는 작지만 내가 할 수 있는 일이 있다

FOMO(fear of missing out) 부수기

- 세상은 넓다: 지금도 수많은 블로그 포스팅, 라이브러리, 유튜브 영상이 쏟아지는 중
- “새로 나오는 정보를 절대 놓치지 않겠어!”
  - 불가능할 뿐 아니라 비효율적인 전략
- 수천, 수만 명이 있는 페이스북 그룹에 모조리 가입
  - 신호대잡음비만 낮아지고, 어차피 구성원 대부분과는 소통할 수 없다
- 유명하다는 뉴스레터 100개를 구독
  - 결국 아무 뉴스레터도 읽지 않고 메일함에 쌓이게 내버려 두는 지름길

FOMO(fear of missing out) 부수기

- **학습에 있어 병목은 외부 정보량이 아닌 나의 시간과 에너지**
- 정보가 얼마나 들어오느냐가 아니라 어떤 정보가 들어오느냐가 중요하다
- 관심사를 공유하는 열정적이고 훌륭한 동료 한 명 > 수천, 수만명짜리 FB 그룹
- “관심사를 공유하는 열정적이고 훌륭한 동료”
  - 당연히 좋겠지. 근데 어디서 찾나요?
  - 스스로 먼저 그런 좋은 동료가 되려 노력하는 것이 도움이 된다

동료가 당신에게 무엇을 줄 수 있는지 묻지 말고...

- 내가 원하는 동료를 생각해보자
- 남들에게 바로 그런 동료가 되기 위해서 내가 할 수 있는 일을 생각해보자
  - 어떤 기술을 공부하는데 문서가 부실하고 관련 자료가 없어 고생했다면,
    공부한 내용과 경험을 블로깅
  - 어떤 버그 때문에 고생했다면,
    다음 사람을 위해 그 버그를 고치는 PR을 작성
  - 너무 좋은 아티클을 언어의 장벽 탓에 못 읽을 이들을 위해 저자의 허락을 얻어 번역

나는 작지만 내가 할 수 있는 일이 있다

- 우리 모두 빚을 지고 있다
  - 귀중한 경험과 지식을 나누고, 남의 고생을 덜어주려 노력하는 수많은 프로그래머들
- **받기만 하는 입장에서 주기도 하는 입장으로 가고자 넘어야 할 문턱은 생각보다 높지 않다**
- 도움을 받은 생태계에 돌려주는 과정에 따라올 것들:
  - 좋은 동료, 기술자로서의 성장
  - 다양한 기회들 (훌륭한 직장, 출판, 강연…)
  - 인정과 보람: making the world a better place™

요약

- 무엇을 배울까: 변하는 세상 속에서
  - 가정의 수명 파악하기, 세상이 변해도 가치있을 지식, 좋은 동료가 되기
- 어떻게 배울까: 이론과 실습 사이의 핑퐁, 사람들은 틀린 말을 한다
  - 빠른 피드백과 잦은 체크포인트
  - 최선을 바라되, 최악을 대비하기
- 왜 배울까: 나는 작지만 내가 할 수 있는 일이 있다

Thanks To

- 서상현 님의 2015년 3월 23일 페이스북 포스팅
- 소프트웨어는 녹이 슨다
- 프로그래밍 언어 배우기의 달인 을 비롯한 애자일 이야기 블로그, 함께 자라기
- 실용주의 프로그래머
- 프로그래밍 심리학, 테크니컬 리더