
N-TIME
NAVER CLOUD CAMP SEMINAR
SW 엔지니어가 맞닥뜨리는 도전들
: 훈련기관 밖의 코드와 고객사 만나기

소 개

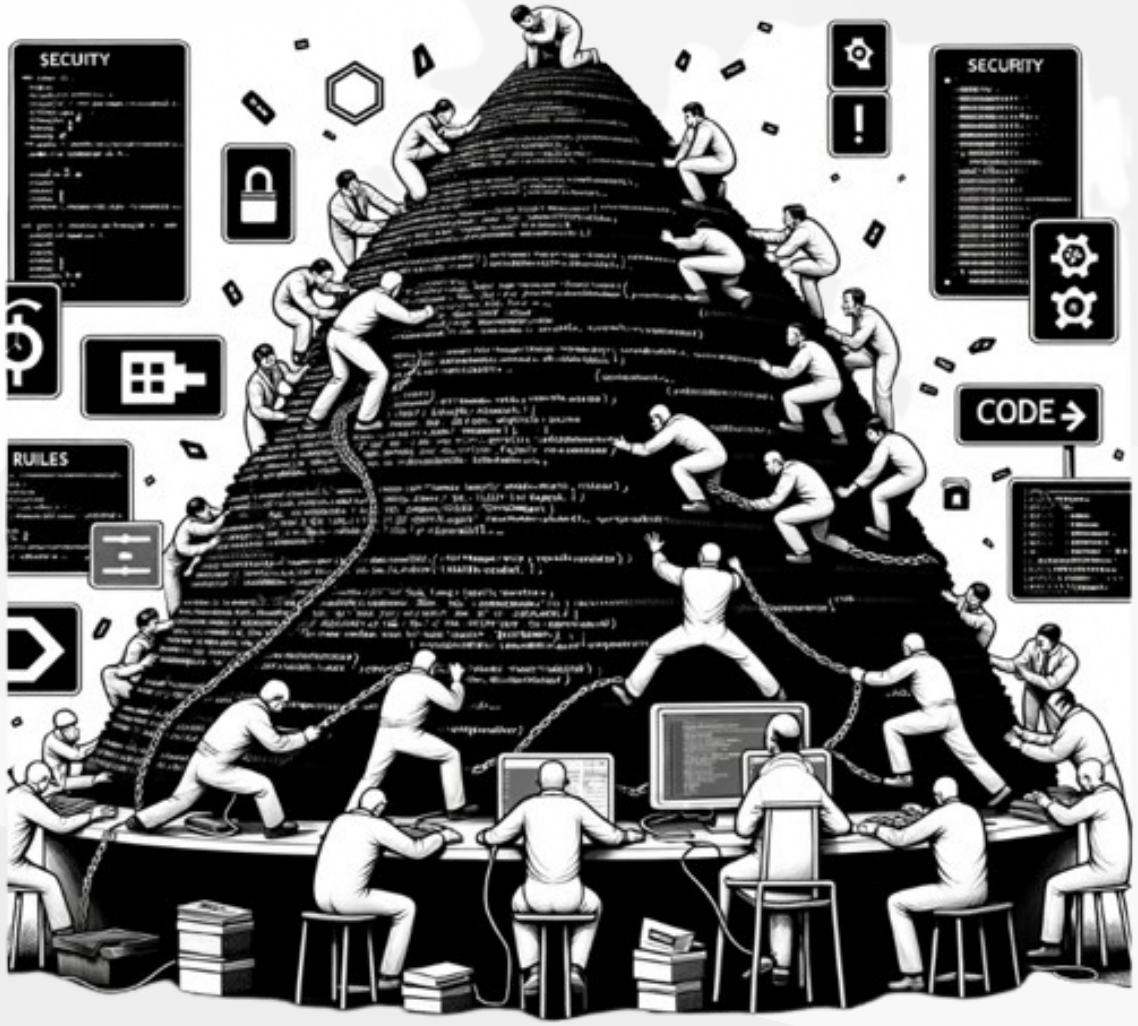
- 김한웅 / 오픈소스 개발자
- 오픈소스를 활용한 기업 컨설팅
- 오픈소스를 활용한 B2B SaaS 개발

첫 출근, 이어지는 도전들

- 긴장감, 잘하려고 하는 마음 바라보기
- 기다림, 공부할 용어와 체계들
- 사실 맨 처음에 할 수 있는 것은 없다는 것을 인정하기
- 이 과정은 자연스럽고 최소 2~5년간 지속될 것
    - 직무 또는 스킬이 아닌 회사에 대한 산업에 대한 이해 과정의 시간이 필요

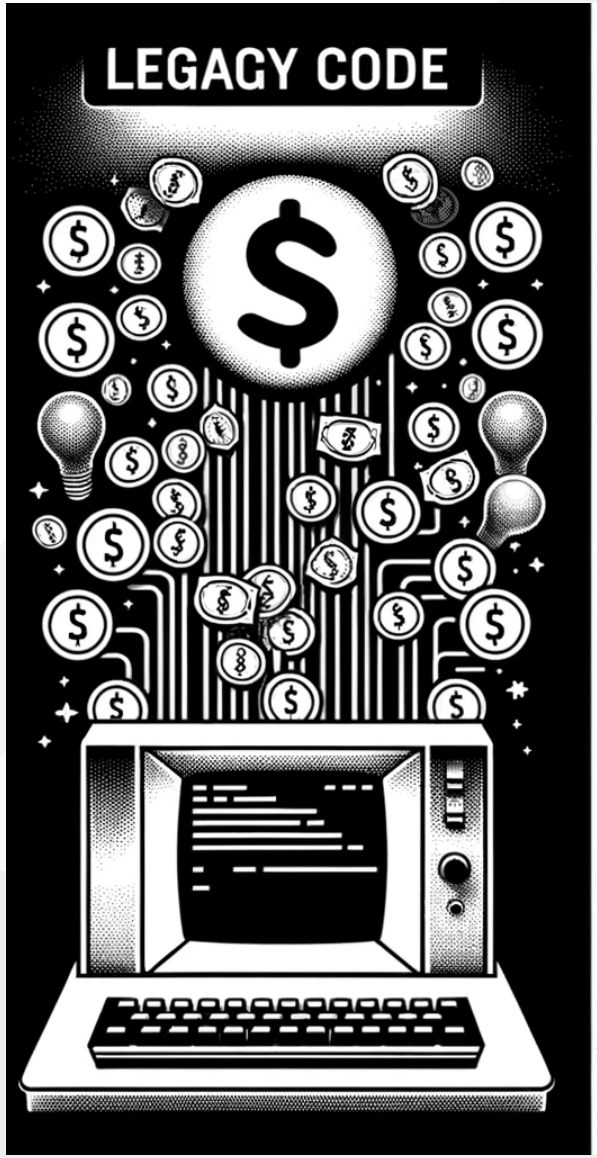
기술 제약에 익숙해지기

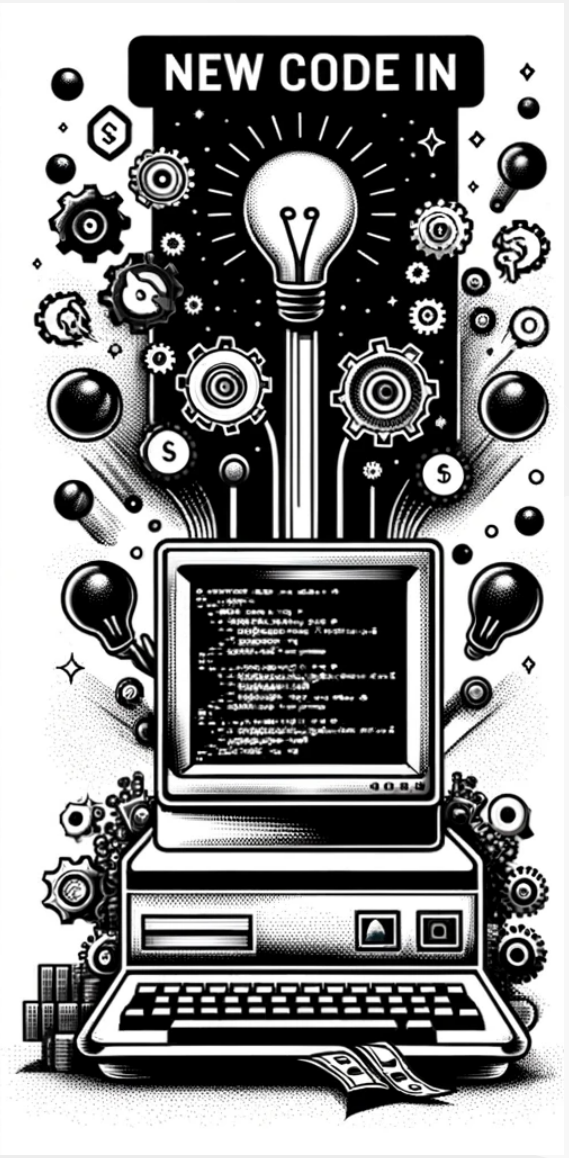
- 여러 팀과 일하게 된다는 것!
  == 많은 검토 지점이 있는 것!
- 테스트 환경
- 팀 코딩 규칙
- 회사 보안팀 규칙에 위배되는가?
- 레거시 코드와 라이브러리



내가 배운 최신 기술, 회사의 오래된 코드

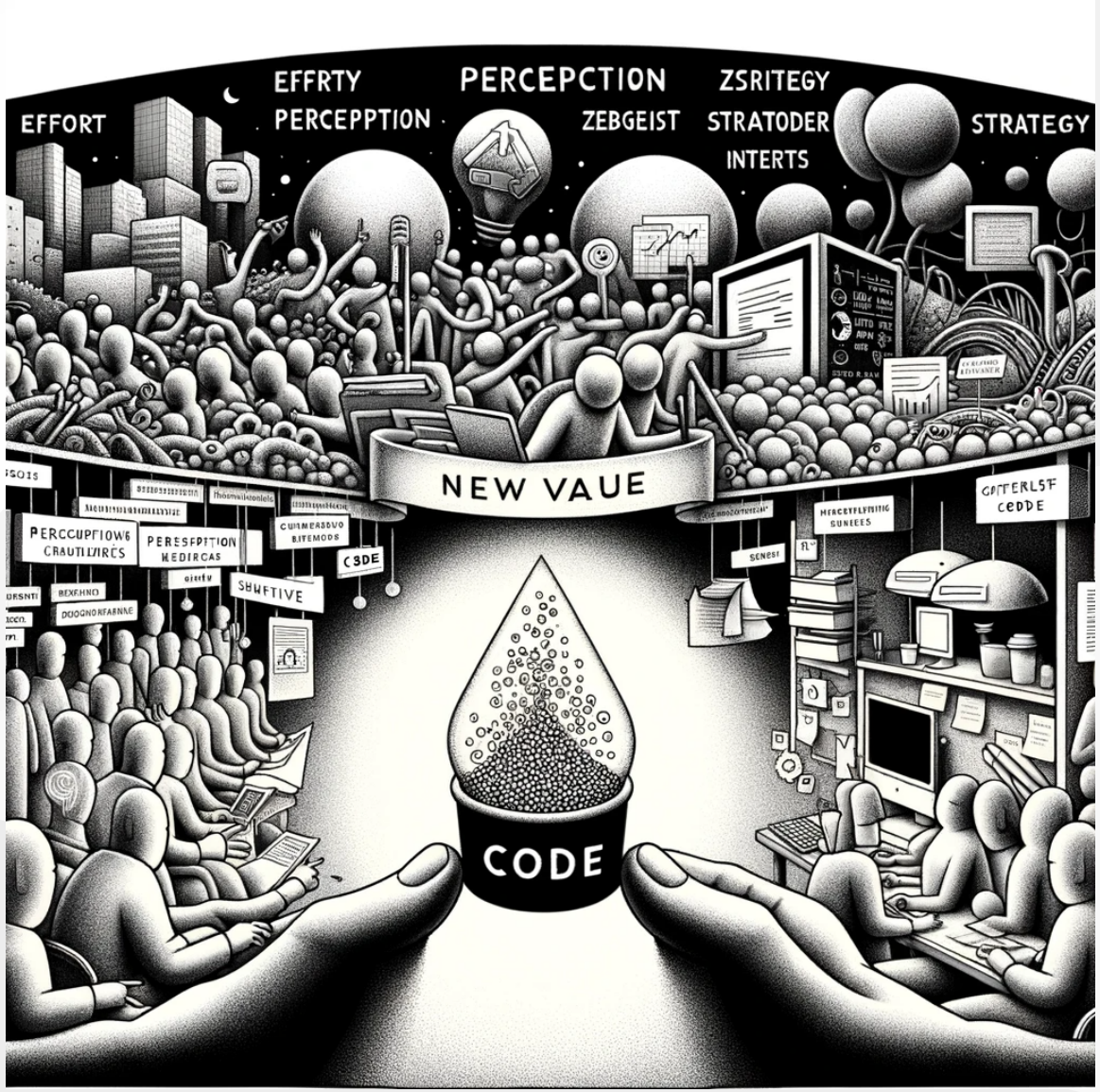
- 기술로 매출을 만든 다는것의 의미는
- 과거에 작성된 코드와 라이브러리로 수익을 만들고 운용한다는 것
- 코드에는 사연이 있다.
- 기능을 추가하는 일도 사연이 있다.
- 항상 레거시 코드를 신경을 쓰고 예측할 수 있는 습관을 지닐 것.
- 절대 바로 코드를 Merge 하지 말 것!





공부했던 기술의 반대편

- 공부할 때 무언가를 처음부터 끝까지 만들어 보는 경험은 중요
- 여러 사람과 같이 일을 할 때 배우는 것
- 여러 사람의 노력, 인식, 문화, 시대상, 전략, 이해관계가 어우러져서 기업의 생존 자본이 만들어지는 것을 알게 될 때까지
- 기존 레거시 코드의 이해, 전반적인 사업 영역의 이해가 더 중요하다는 것을 깨닫는 데 오랜 시간이 걸림



준비 과정: 보안에 따른 제약

- CASE: 콘텐츠 보안 때문에 물리적으로 인터넷이 되지 않는 상황
- 라이브러리, 라이브러리 의존성
- 하나의 기능을 사용하기 위해 많은 팀과 대화해야 함
- 왜 해당 기술이 필요한지 리포트 작성, 보안 결점 정리 및 정보 발송
- 때때로 IT 담당자와 방화벽 정책 논의

리서치: 요구사항 및 이슈 수집

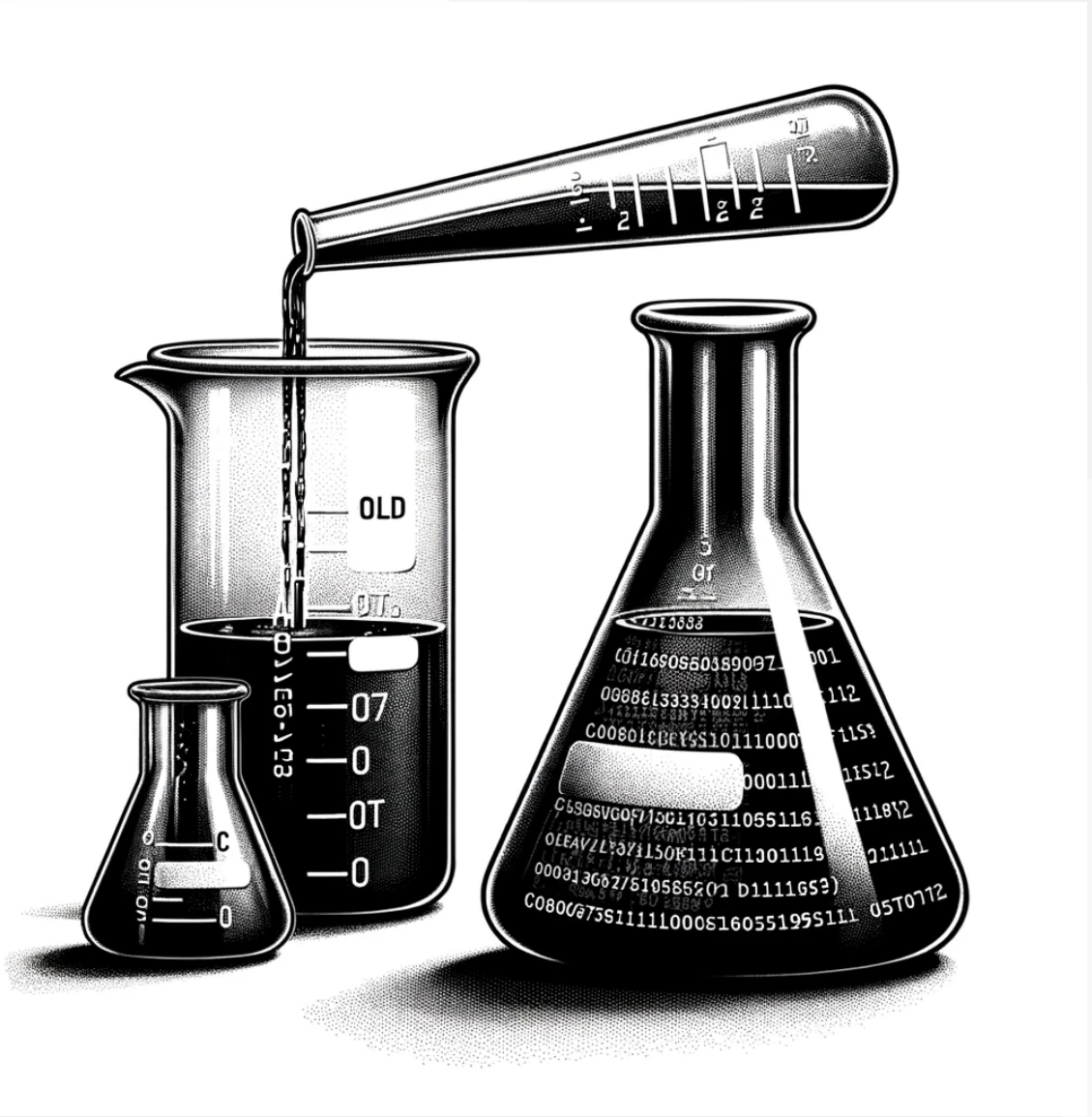
- 요구사항을 잘 정리하기 위해 실제로 알아야 하는 배경지식
- 과거 역사 이해하기
- 이해와 공유를 위한 로그 내용 기록
- 이해와 공유를 위한 스크린샷 남기기
- 실행 환경: OS, 브라우저, 사용 Version
- 바로 해결하기보다는 이슈 묵히기! (성급한 판단)

리서치: 정보(생각)를 머금기

- 어떠한 한 조직에 편향되도록 작성하지 않기
- 많은 팀과 대화하기
- 기획이나 회사 전략상 영향력 체크
- 문제점이나 해결점이 정리되기까지는 조직 내 또는 개인에게 머금는 시간이 필요함
- 중요 점: 빨리 처리하면 인정받을 거야! == 인정욕구에 대한 성급한 액션
- 넓은 시야 필요, 많은 입장에 대한 경청

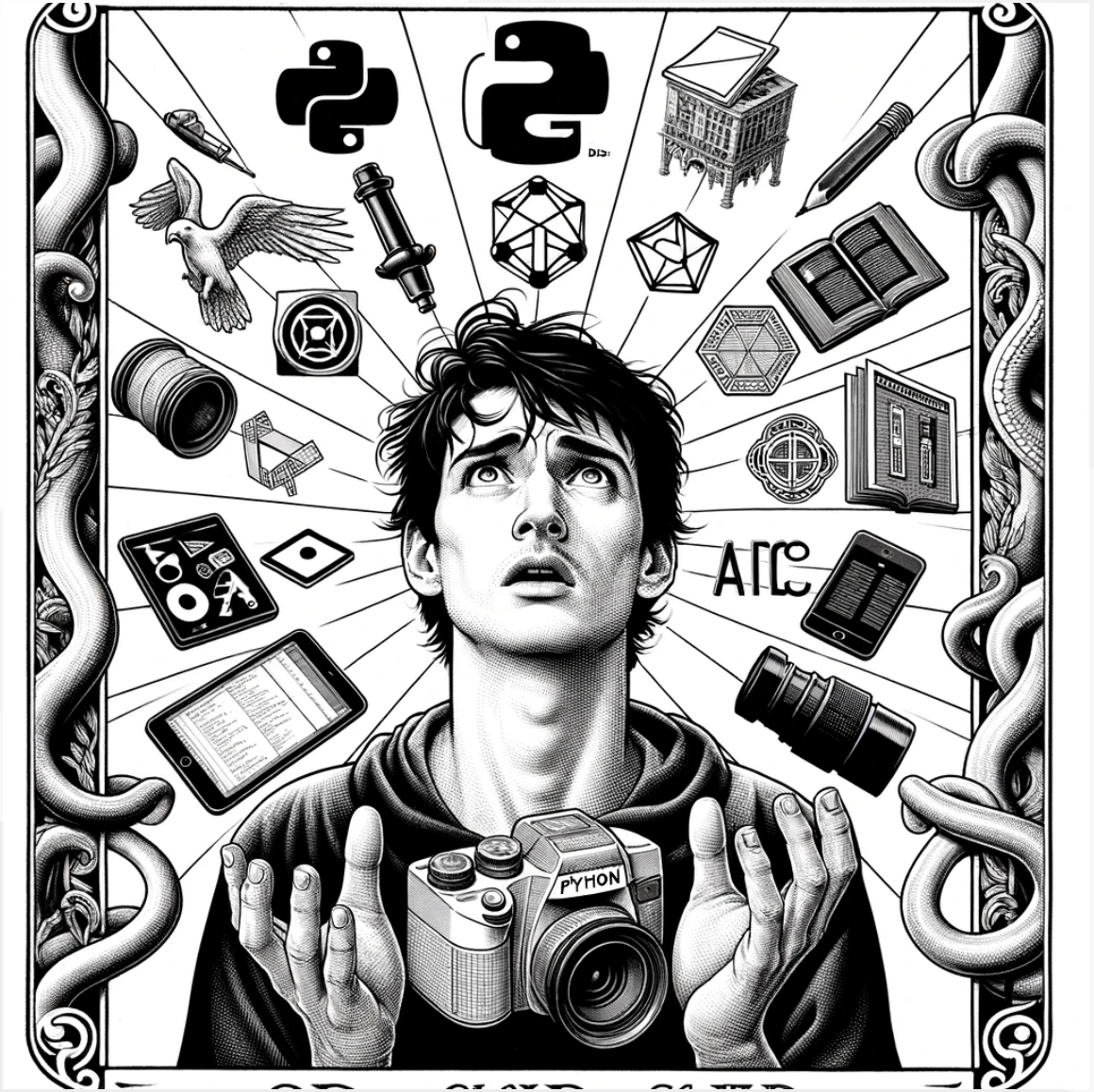
시장구조: 자료구조

- 초기 업무는 버그 및 기능 개선일 것
- 이후, 새로운 기능이라면 자료구조 논의가 필요함.
- 새로운 기능에 필요한 자료구조 작성
- 앞으로 어떻게 될 것인가?
- 이 단계에서는 코드로 이야기하기보다는 이런 것들이 추가된다면 과연 우리의 SW에 무슨 일들이 일어날까? 라는 지점을 대화해 보는 것이 중요하다.



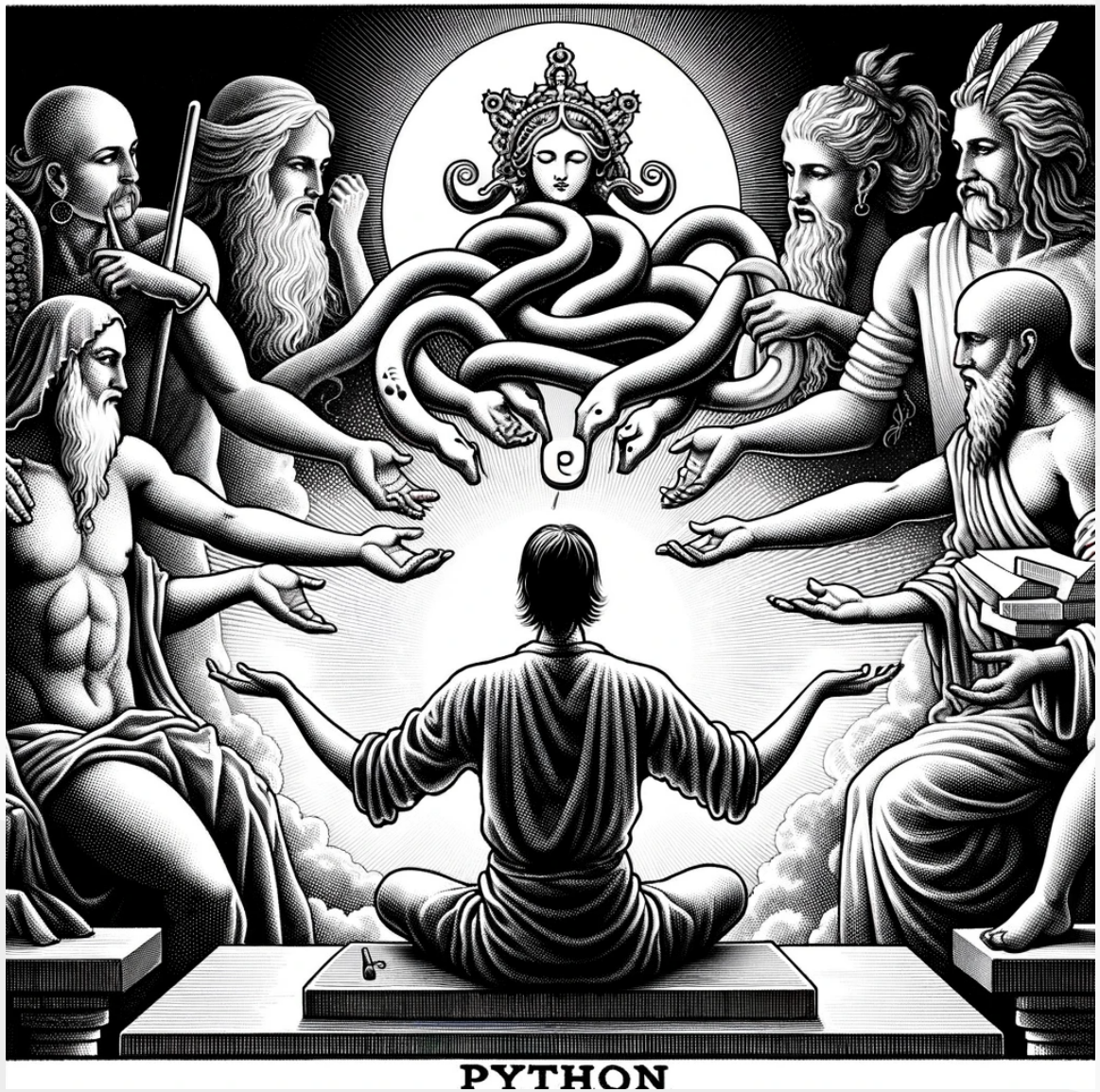
사례: 연결된 정보들(이미지, 영상, Color)

- 코드를 짜며 추가로 다른 지식이 필요했다.
- 이미지, 영상, 컬러에 대한 시행착오
- 많은 방법, 지식, 시장 상황, 생태계 경험이 있어야 잘못된 설계, 토론, 이슈 작성을 막을 수 있다.
- 여러 오픈소스의 이슈를 살펴보는 것도 도움이 되었다.
- Python > DCC > Camera > API > OpenColorIO > OpenImageIO



사례: DCC Tools 연결

- DCC Tools: Digital Contents Creation Tools
- 개발 기능이 DCC 툴과 연동 필요
- 많은 DCC 툴은 Python 또는 내장 언어가 존재
- API 또는 Python 코드를 추가 작성
- DCC 툴의 별도 공부, 이해
- 다른 분야에서도 비슷한 현상 존재
- 하나의 기능이 탄생할 땐 많은 것이 연결되어 있다.

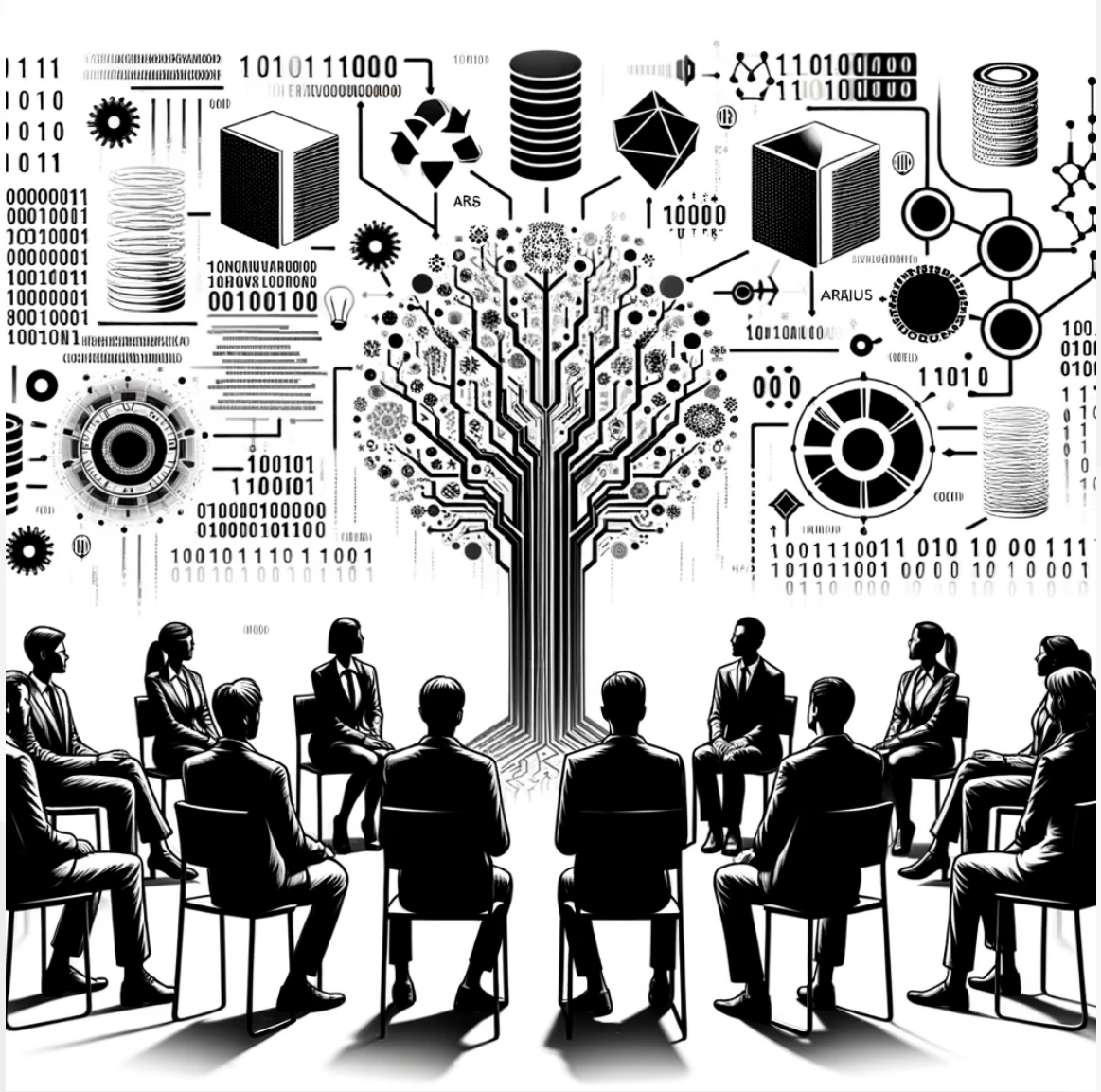


Git로 대화하는 법 배우기

- Pull Request를 위한 글 작성. 글을 작성할 때는 이슈가 발생한 근거나 문제 발생 지점에 대한 리서치 내용의 링크나 설명이 필요해요.
- 리서치 문서를 보관하면 개발자, 구성원끼리 학습 가능
- 새로운 입사자 교육자료 생성
- 문서를 작성하는 과정에서 놓치거나 또는 사고의 과정을 단단하게 만들 수 있어요.
- Git을 꼭 배우세요. Pro Git 추천

자료구조로 대화하기

- SW 개발을 이야기할 때 기능(Function), UX로 이야기하면 너무 대화가 광범위해져요.
- 최대한 자료구조로 이야기하는 것이 심플합니다.
- 자료구조로 대화하면 알아서 함수와 UX가 만들어져요.



Command-Line 또는 API 제작

- UX를 만들기 전에 간단하게 Cmd를 만들고 테스트
- 이 방식은 Linux, Unix에서 오랫동안 사용되는 방식
- 많은 데이터를 넣어보기 위해 편리하고, 여러 개발자를 통해 테스트할 수 있는 환경구축
- 많은 데이터가 들어가면 예상치 못한 문제점이 보여요.
- UX 제작 전 Backend에서 많은 것을 테스트

UI / UX 라이브러리 검토

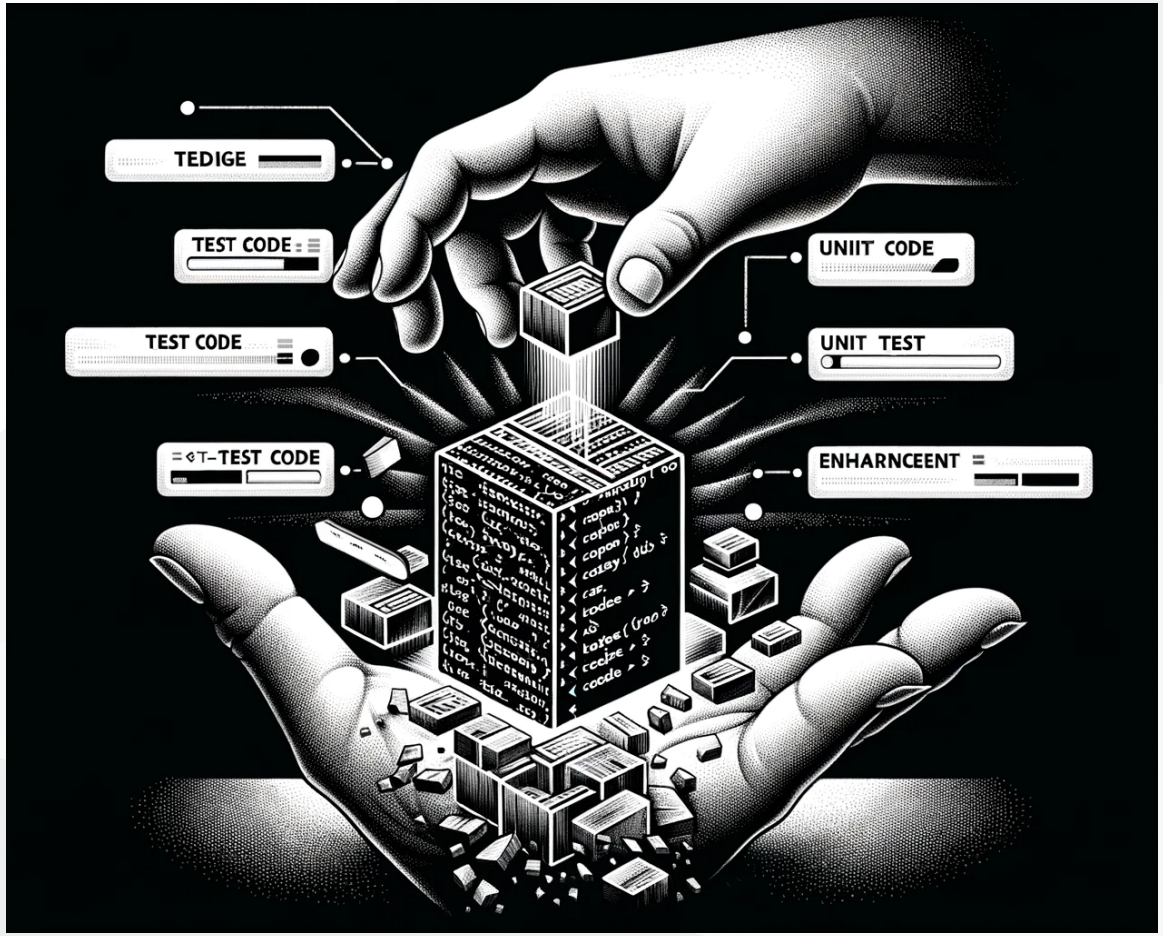
- 수많은 UI/UX 라이브러리
- TEST: 간단한 버튼
- B2B: 부트스트랩으로 빠르게 UX 제작
- B2C: Figma, React, Flutter를 활용
- 아이디어로 필요한 버튼을 생성하고 나중에 UX 디자이너가 붙는 것이 더 효율적일 때도 있다.

피드백이 몰리는 타이밍

- UX와 작동되는 버튼이 보일 때부터 실제 피드백이 들어옵니다.
- 개발보다는 이슈에 담겨있는 의미를 중요하게 바라보세요.
- 들어오는 이슈들에 쫓겨가며 성급하게 개발을 처리하지 마세요.

테스트코드 & 유닛테스트

- 코드를 작성했다면 꼭 테스트 코드를 작성하는 습관을 들이세요.
- 테스트 코드를 작성하는 프로세스를 익혀두면 좋습니다.
- 여러분의 개발 언어로 테스트 코드 작성, 유닛 테스트하는 방법을 익혀두면 단단한 코드와 함수 작성 가능

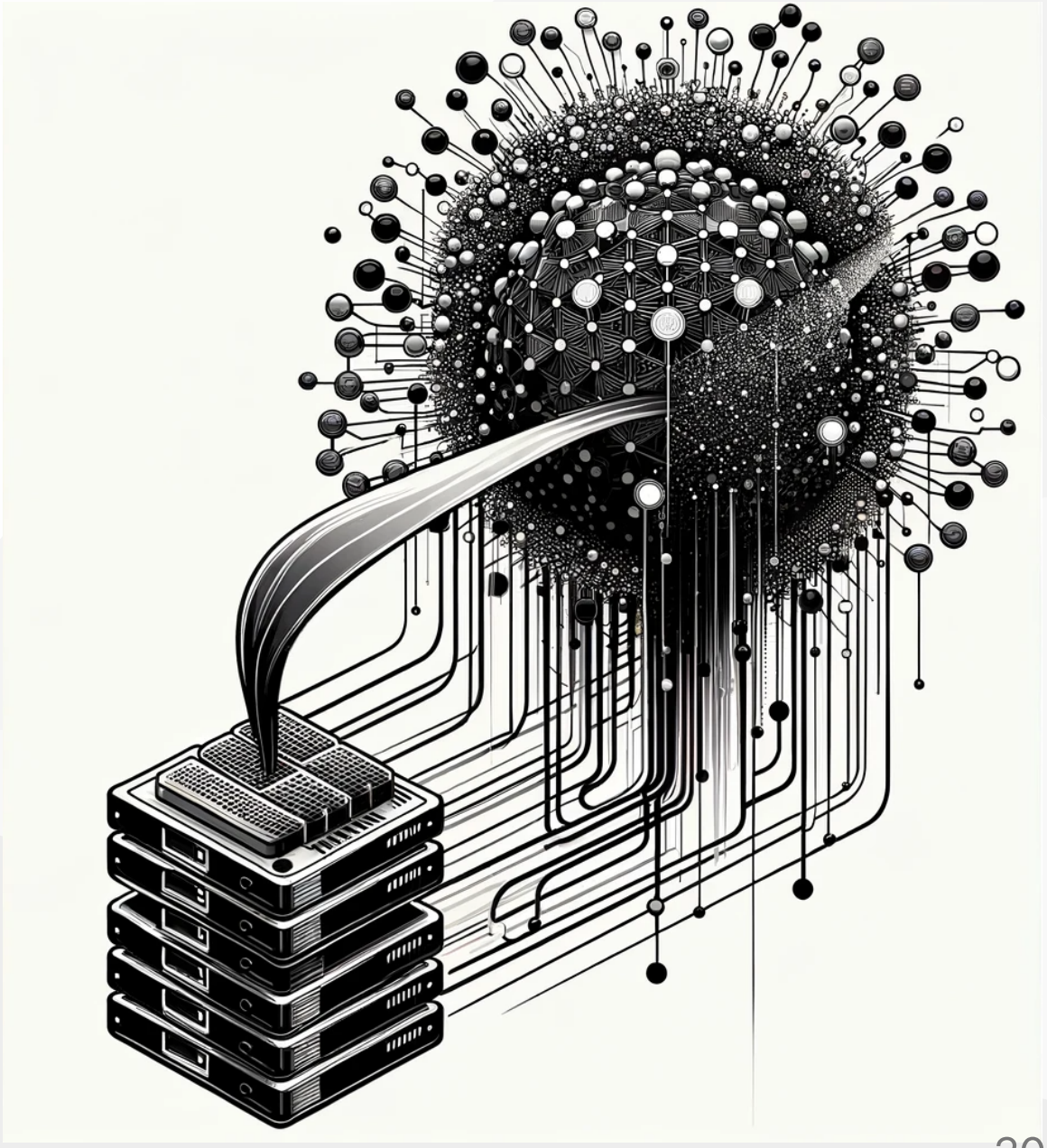


테스트 서버 배포

- 예측하지 못한 문제점을 알기 위해서 본 서버 이전에 개발 서버에 배포하고 모니터링합니다.
- 이때 나오는 개발 이슈들을 정리해서 보완합니다.

본 서버 배포

- 실 서버에 배포하고 모니터링합니다.
- 감시할 수 있는 수단을 만들거나 대비하세요.
- 무중단 서비스를 위해 클라우드 또는 가상화, 쿠버네티스, 클러스터 개념을 배우세요.
- 안정적으로 서비스를 운용할 수 있는 기초 지식을 얻을 수 있습니다.
- 클라우드 사용 전 개념을 공부하기에 좋습니다.



소프트웨어 판매하기

- 매뉴얼, 튜토리얼, 마케팅에도 관심을 가지세요.
- 매출 전략상 개발팀과 긴밀한 협업이 필요합니다.
- 매출이 일어나기 전까지 만들어야 하는 문서들이 매우 많습니다.

균형잡기

- SW를 사용하는 모든 고객사의 요구사항을 들어주면 툴이 망가집니다.
- 요구사항을 들어주지 않으면 계약이 되지 않습니다.
    - 기업 생존과 직결
- 두 지점에서 균형을 잡으면서 SW 로드맵을 세우고 발전시킬 필요가 있음
- 회사의 모든 역사를 아는 노련한 사람의 시야를 배우세요. 그런 사람들은 때론 감각으로 선택합니다.

Q&A

khw7096 @ gmail.com