코딩은 정말 모두에게 열려 있는가?

Is coding really open to everyone?

"Anyone can learn to code." 하지만 누구나 같은 출발점에서 시작하지는 않는다.

코드 구조 + 영어 기술 언어 + 검색과 번역 + 오류 해석

Code structure + technical English + search & translation + error reading

영어는 코드 밖에만 있지 않다

English is not only inside the code

키워드 · 함수와 변수의 관습 · 오류 메시지 · 공식 문서

검색어와 커뮤니티 · GitHub와 오픈소스 협업 · AI 코딩 도구

Keywords · naming conventions · error messages · docs · search · GitHub · AI tools

두 개의 언어를 동시에 배운다

Learning two languages at once

코드를 이해한다

↓

영어를 해석한다

↓

검색 가능한 영어 문장으로 바꾼다

↓

답변을 다시 한국어 개념으로 연결한다

↓

코드에 적용한다

한 번의 학습처럼 보이지만, 여러 번의 번역이 일어난다.

개인의 부족함처럼 보이는 구조적 부담

A structural burden that looks like personal failure

에러를 늦게 해결한다 → “기술 이해가 부족하다”

질문을 짧게 못 쓴다 → “문제 정의 능력이 부족하다”

문서를 오래 읽는다 → “학습 속도가 느리다”

실제로는 — 언어 접근 조건의 차이

In reality: unequal conditions of language access

왜 '언어 식민화'인가?

Why call it linguistic colonisation?

어떤 지식이 공식적인가

누가 전문가처럼 보이는가

어떤 표현이 검색 가능한가

어떤 사고방식이 표준으로 인정되는가

누가 먼저 말하고, 누가 계속 번역해야 하는가

One language decides what counts as knowledge, expertise, and the default.

기본값 사용자

Default user

영어를 읽을 수 있다

영어로 검색할 수 있다

오류를 스스로 해석할 수 있다

최신 장비와 빠른 인터넷을 쓴다

지속적으로 자기 학습을 할 수 있다

Outside these conditions, extra work appears: translate, ask, adapt, repeat.

유지보수

Maintain

시스템이 계속 자연스럽게 작동하는 것처럼 보이도록 만드는 반복적인 조정.

용어 번역 · 오류 해석 · 검색어 수정 · 튜토리얼 비교 · 버전 확인

개념을 다시 설명하기 · 학습자의 좌절을 조정하기

Maintenance keeps the system looking natural — and stays invisible.

한국어 코딩은 영어를 삭제하는 것이 아니다.

Korean coding ≠ deleting English

한국어로 구조를 먼저 이해한다

이해한 개념과 영어 기술어를 연결한다

번역 과정 자체를 학습 내용으로 드러낸다

질문하고 설명할 수 있는 한국어 공동체를 만든다

번역이 아니라 중간층 만들기

KRThree.js: not translation, but a middle layer

```
const 장면 = new THREE.Scene()

const 카메라 = new THREE.PerspectiveCamera(...)

const 빛 = new THREE.DirectionalLight(...)
```

한국어 개념 — 영어 API — 화면에서 일어나는 변화

Explaining what the code does, in Korean.

현지화는 키워드 번역보다 넓다

Localisation is wider than keywords

키워드 · 식별자와 변수명 · 오류 메시지

문법과 어순 · 숫자와 문자 체계 · 개발 환경

문서와 예제 · 검색과 커뮤니티 · 문화적 맥락

Even with translated keywords, the rest of the stack stays in English.

오픈소스는 언어적으로도 열려 있는가?

Open source ≠ Equal access

질문은 영어로 해야 한다

기여 규칙은 영어로 작성된다

비영어 문서는 덜 발견된다

영어가 아닌 기여는 주변화될 수 있다

GitHub 2015–2025: non-English projects gain less visibility and participation.

AI는 번역 노동을 없애는가?

Does AI erase translation labour?

AI는 번역을 돕는다. 그러나 새로운 유지보수도 만든다.

번역 결과 확인 · 잘못된 코드 검증 · 영어 중심 데이터의 편향

한국어 설명과 코드의 불일치 · AI가 만든 오류의 책임

AI automates some maintenance — and adds new verification work.

유지보수를 어떻게 보이게 할 것인가?

Making maintenance visible

어떤 영어 단어에서 멈췄는가

어떤 검색어를 만들었는가

번역 전후 의미가 어떻게 달라졌는가

누가 누구의 오류를 설명했는가

어떤 설명이 다음 참가자에게 남았는가

Where did we stop? What did we search? What changed in translation?

공동체는 무엇을 생산하는가?

What does the community produce?

결과물은 3D 서재만이 아니다 —

한국어 개념어 · 주석 달린 예제 · 오류 해결 기록

검색어의 번역 · 참가자 사이의 설명 · 다음 학습자를 위한 작은 기반 시설

A small infrastructure of linguistic care, left for the next learner.

기술 지식의 입구를 다시 설계하기.

Redesigning the entrance to technical knowledge.

누가 계속 번역해야 하는가?

누구의 언어가 기본값인가?

학습이 가능하도록 누가 시스템을 유지하는가?

한국어 코딩은 영어를 거부하는 운동이 아니라, 이 질문들을 묻는 실천이다.