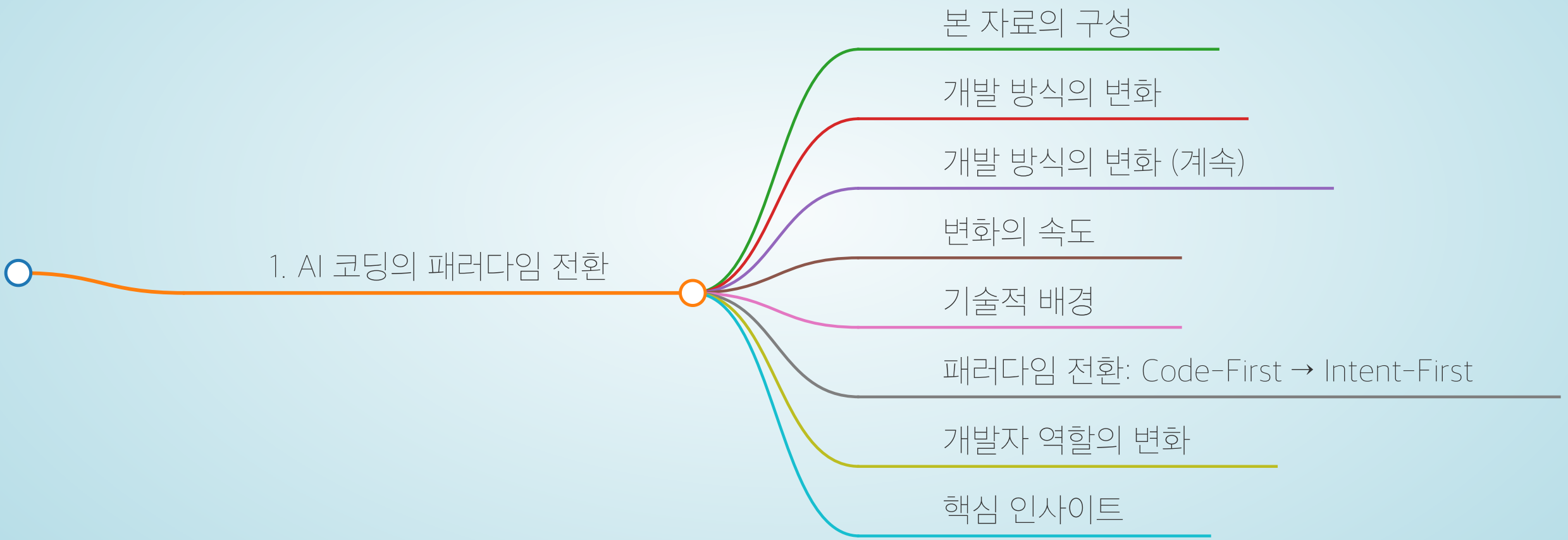
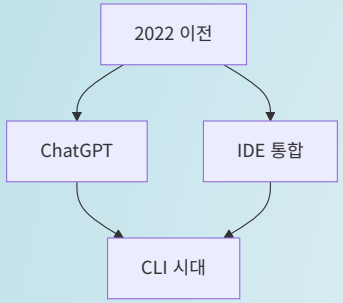
Examples for MarkdownGraph



개발 방식의 변화



2022년 이전

- 개발자가 코드를 한 줄 한 줄 직접 타이핑
- Stack Overflow와 문서 검색에 의존
- 반복적인 코딩 작업에 많은 시간 소요

2022년 ChatGPT 등장

- 자연어로 질문하고 코드 조각을 얻는 방식 시작
- 개발 보조 도구로서 LLM 활용 증가

개발 방식의 변화 (계속)

2023년 IDE 통합

- GitHub Copilot, Cursor 등 IDE 내 실시간 t 드 제식
- 검발 흐름을 끊지 않는 인라인 자동완성
- 프로젝트 컨텍스트를 일발 이해하기 시작

2024년부터 CLI 기반 코딩 본격화

- 터미널에복 실행되는 AI 도구 등장 (2024: Claude Code, 2025: Gemini CLI)
- AI가 전체 프로젝트를 이해하고 파일을 직접 수정
- 에이전트1 럼 자율적으로 멀티파일 작업 수행

변화의 속도

- **ChatGPT**: 출시 후 2개월 만에 1억 터용자 돌파
- **생산성 향상**: GitHub Copilot 도입 기업의 개발자 생S 성 55% 증가
- **시장 성장**: AI t 딩 도구 시장, 2024년 기준 연 100% 프상 성장

기술적 배경

LLM 성능 향상

- GPT-4, Claude 3.5 Sonnet 등 강력한 모델 등o
- 코드 이행 및 생성 동력 L 약적 발전

컨텍스트 윈도우 확대

- 초기 4K 체큰 → 현재 200K 체큰 이M
- 전w 코드상이스를 한 번에 이행 가동

개발 도구 통합

- IDE, CLI, 에디터와직 원활한 연동
- 파일 핑스템 직의 의근 및 S 출 가동

패러다임 전환: Code-First → Intent-First

전통적 개발 방식 (Code-First)

- C 든 t 드를 (발m 가 직접 n P
- E 법- A성 세K 사항 얻기 이요
- 반복적인 보일러플레이트 t 드 n P
- 디M 깅 사이기이 길고 반복적

AI 시대 개발 방식 (Intent-First)

- "D 엇을" 만들지 설명하A AI가 "어떻게" 구현
- 고S 준 설계원 L 즈4 스 로직에 일u
- 반복 n 업 m 동화로 생S P 향상
- 검w 원 핑질 . 리에 일u

개발자 역할의 변화

구분	전통적 개발	AI 시대 개발
주요 역할	코드 작성자	의도 설계자
필요 역량	문법 전문가	아키텍처 설계자
작업 방식	디버깅 중심	품질 검토 중심
시간 배분	반복 작업 수행	창의적 문제 해결

핵심 인사이트

> *검K 자는 : 이상 "t 드 타이핑 기게"가 아닌 **"시스템 설계자이자 품질 관리자"**로 가화하고 있색니다.*

- **반복 작업 자동화**: 반일러플A 이트, 모R생, 테스트 t 드 등
- **창의성 증대**: 비즈니스 로i 과 사용자 경항 검선에 일중
- **생산성 향상**: 같재 시간에 : 많재 가기 큰출