

한 걸음 앞선 개발자가 지금 꼭 알아야 할 클로드  
코드

한 걸음 앞선 개발자가

지금 꼭 알아야 할

MASTERING  
CLAUDE  
CODE

조훈, 정찬훈 지음

클로드

코드

CLAUDE  
CODE

실무에서 검증된 개발 방식 그대로,

매일 1시간 4주

Claude Code 에이전트 실전 훈련!

```
# 설치(macOS, Windows)
# CLAUDE.md 설정
# MCP 연동/다양한 활용 전략
# 클로드 워크플로 전략
# 설계 → 부트스트래핑 →
  테스트 → 개선 → 명세
# 클로드 코드의 효율을 극대화
  하는 다양한 방법
```

갈벗

깃허브: [https://github.com/sysnet4admin/\\_Book\\_Claude-Code](https://github.com/sysnet4admin/_Book_Claude-Code)

## AI 기반 코딩에서 라이브러리 버전 관리의 어려움

요즘 개발 현장에서 AI를 쓰지 않는 팀을 찾기 힘들 정도입니다. ChatGPT, Claude, Copilot 등 다양한 LLM 기반 도구들이 이미 코드 생성의 보조 엔진으로 자리를 잡았죠. 하지만 실제로 사용해 보면, 늘 아쉬운 지점이 따라옵니다. 바로 LLM이 최신 기술 환경을 잘 따라잡지 못한다는 점입니다.

개발에 쓰이는 라이브러리와 패키지는 하루가 다르게 버전이 올라가고, API는 예고 없이 바뀌기도 합니다. 예를 들어, 어제까지만 해도 잘 돌아가던 코드가 오늘은 **deprecated** 경고를 뿜어내거나, 아예 실행조차 되지 않는 상황이 벌어지곤 합니다. 하지만 우리가 활용하는 LLM들은 훈련 시점의 지식에 머물러 있습니다. 모델이 2024년 기준으로 학습되었다면, 그 이후에 발표된 프레임워크의 변화나 내부 API 업데이트는 반영하지 못하게 됩니다.

이 문제를 해결하려고 등장한 방식으로 RAG(Retrieval Augmented Generation)를 꼽을 수 있습니다. RAG는 간단히 말해, AI가 답변을 하기 전에 최신 데이터를 검색해서 참고하도록 만드는 구조입니다. 문서 검색과 생성형 모델을 조합해, 최신성을 보강하려는 시도입니다. ChatGPT 등 우리가 사용하는 LLM들은 사용자의 프롬프트를 분석하여 검색이 필요한 경우 이를 직접 검색해서 결과물을 보강하기도 합니다.

하지만 RAG에도 분명 한계가 있습니다.

- 검색 품질에 따라 결과의 정확도가 들쭉날쭉합니다.
- 공개 문서에 없는 내부 전용 API, 사내 설정 값 같은 건 여전히 반영하기 어렵습니다.
- 코드 생성 과정에 자연스럽게 녹아들지 않고, 검색 결과를 붙여주는 보조 수단 수준에 머무르는 경우가 많습니다.

최신 라이브러리만이 문제가 아닙니다. 우리가 개발하고 있는 프로젝트는 여러 라이브러리가 사용되고 있고 서로간의 호환성을 잘유지해야합니다. 무작정 최신 버전을 사용해서는 안되고 현재 사용중인 라이브러리와 버전을 잘유지하고 해당 버전에 맞는 API와 코드 사용이 필요합니다. 하지만 LLM기반의 코드 생성은 이러한 정보들을 제대로 반영하지 못하고 있습니다.

이렇게 실제 개발 환경에서 필요로 하는 내용을 충분히 메워주지는 못하고 있습니다. 이런 문제를 보다 근본적으로 해결하기 위해 등장한 것이 **Context7**입니다.

# Context7?

Context7은 Upstash에서 관리하는 MCP 서비스로 LLM에게 다양한 버전의 라이브러리에 대한 정보를 정확하게 전달해주기 위해 만들어졌습니다. Context7은 단순히 데이터를 검색해서 던져주는 것이 아니라, AI가 코드 생성 시점에 실제 개발 컨텍스트를 직접 주입받을 수 있는 환경을 제공합니다. 다르게 풀자면 AI가 우리의 개발 환경을 좀 더 잘 이해할 수 있도록 좀 더 직접적인 컨텍스트를 제공해주는 역할을 합니다.

아래는 공식 페이지의 소개 내용입니다.

## What is Context7?

LLMs rely on outdated or generic information about the libraries you use.

Context7 pulls up-to-date, version-specific documentation and code examples directly from the source. Paste accurate, relevant documentation directly into tools like Cursor, Claude, or any LLM. **Get better answers, no hallucinations and an AI that actually understands your stack.**

### Without Context7

- ❌ Outdated documentation from training data
- ❌ Hallucinated code examples that don't work
- ❌ Generic answers not specific to your version
- ❌ Time wasted verifying AI responses
- ❌ Frustrating back-and-forth with LLMs

### With Context7

- ✅ Up-to-date, version-specific documentation
- ✅ Real, working code examples from the source
- ✅ Concise, relevant information with no filler
- ✅ Free for personal use
- ✅ Simple integration with your coding tools

Context7 is built & maintained by the Upstash team. Reach out to us at [context7@upstash.com](mailto:context7@upstash.com)

<https://context7.com/about>

요약해서 표로 나타내면 아래와 같게 비교해볼 수 있습니다.

without Context7	with Context7
- LLM이 오래된 라이브러리 정보를 활용할 수 있음 - 내가 사용하는 버전의 라이브러리에서	- 최신 라이브러리 정보를 가져올 수 있음 - 실제 내가 사용하고있는 라이브러리의 명세를 활용하게 함

존재하지 않은 함수를 사용할 수 있음 - 복수의 라이브러리를 사용하고 있을때, 서로 호환이 잘 이뤄지지 않을 수 있음	- LLM이 능동적으로 내 컨텍스트를 파악하여 코드를 생성하게끔 유도할 수 있음
---	---

이렇게 **Context7**을 통해 다음과 같은 효과를 기대할 수 있습니다.

1. 최신성 확보: 에이전트가 과거에 학습한 데이터를 넘어 현재 개발 중인 프로젝트에서 실제로 사용 중인 라이브러리 버전, **API** 스펙, 개발 문서 등 최신화된 정보를 그대로 참고할 수 있습니다.
2. 정합성 개선: 모델이 생성한 코드가 실행 환경과 어긋나는 일을 크게 줄일 수 있습니다. 좀 더 에이전트가 생성한 코드가 즉시 사용가능한 형태에 가까워질 수 있습니다.
3. 도메인 특화: 공개되어 있지 않은 내부 코드베이스도 에이전트에서 컨텍스트로 활용할 수 있게 됩니다.

결과적으로 **AI**의 산출물을 우리의 환경에 좀 더 강하게 밀착시켜주는 역할을 합니다.

## Context7 시작하기

좀 더 알아보기전에 **Context7**를 사용하는 방법들을 살펴봅니다. 현재까지 여러가지 방법으로 사용가능하나 일단 공식적인 가이드는 아래 **Context7**에 가입, **API Key**를 확보해야합니다.

<https://context7.com/>



Context7

Sign in

MCP Server

+ Add

## Up-to-date Docs for LLMs and AI code editors





















Copy latest docs & code — paste into Cursor, Claude, or other LLMs

Search a library (e.g. Next, React)

☆ Popular

↗ Trending

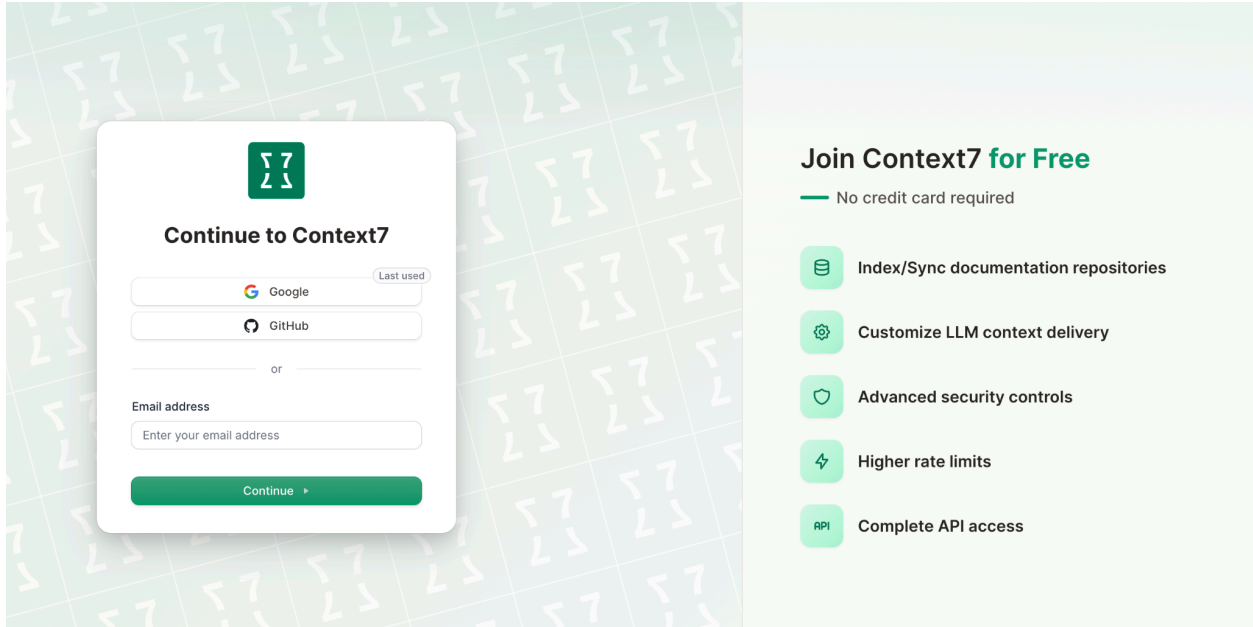
🕒 Recent

	SOURCE	TOKENS	SNIPPETS	UPDATE
Next.js	 /vercel/next.js	541K	3.2K	19 hours 
Shadcn UI	 /shadcn-ui/ui	201K	1.2K	1 day 
Vercel AI SDK	 /vercel/ai	515K	2.2K	1 week 
Better Auth	 /better-auth/better-auth	226K	1.3K	1 day 
MongoDB	 /mongodb/docs	15.1M	148K	1 week 
Supabase	 /supabase/supabase	1.2M	4.5K	1 week 
React	 /reactjs/react.dev	618K	2.4K	1 week 
Tailwind CSS	 tailwindcss.com/docs	588K	1.8K	2 days 
OpenAI API Docu...	 platform.openai.com/docs	6.4M	27K	3 days 
Tailwind CSS	 /tailwindlabs/tailwindcss.com	559K	1.7K	13 hours 

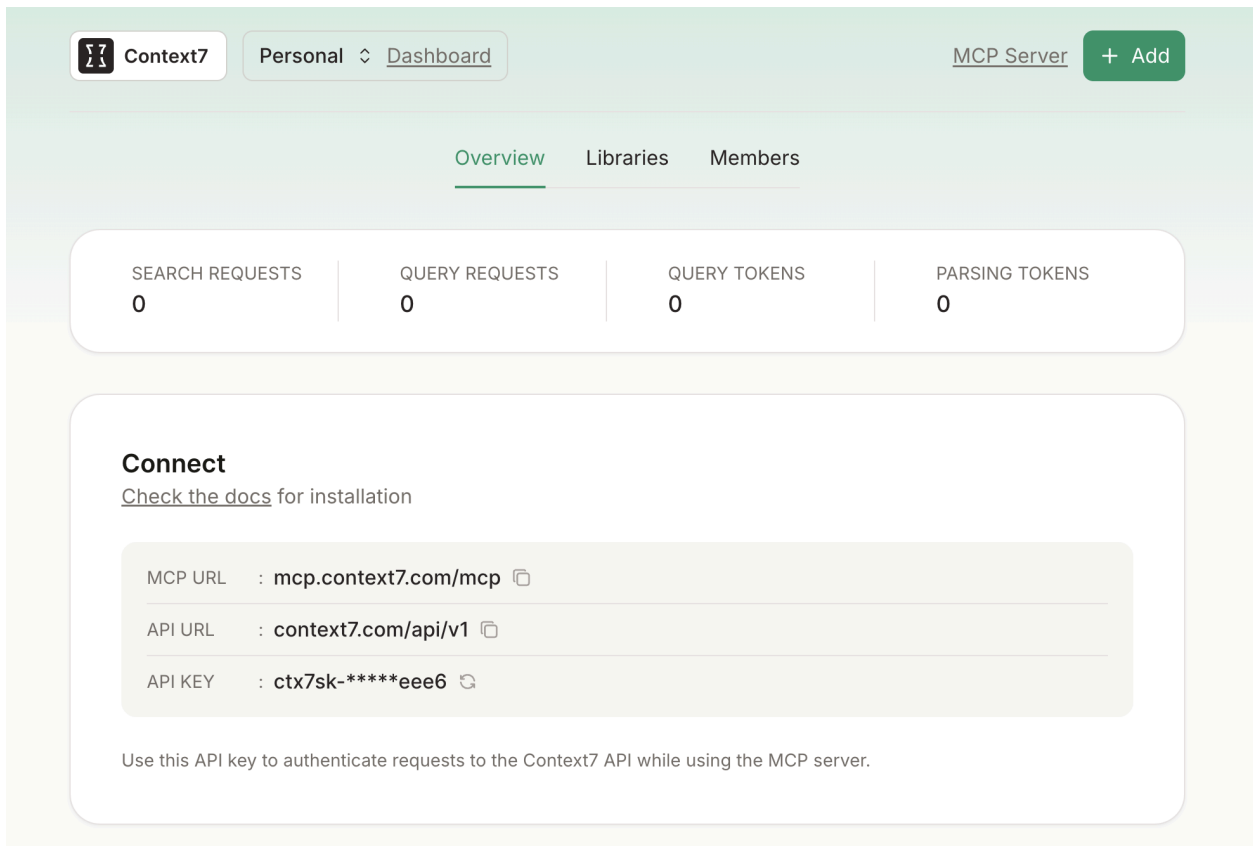
44,813 LIBRARIES

[SEE TASKS IN PROGRESS](#) →

Sign in 을 선택합니다.



Google이나 Github 계정으로 시작할 수 있습니다. 다행히 번거로운 결제정보 입력이 필요하지 않습니다.



바로 사용 가능한 **API Key**를 받을 수 있습니다. 이제 이걸 토대로 아래와 같이 클로드 코드에 **MCP**를 등록합니다.

<Terminal박스>

```
$ claude mcp add --transport http context7 https://mcp.context7.com/mcp --header  
"CONTEXT7_API_KEY: YOUR_API_KEY"
```

</Terminal박스>

위는 원격지의 **MCP** 서버를 **Streamable HTTP**를 통해 사용하는 방식입니다. 아래와 같이 로컬 **MCP**서버를 쓸 수도 있습니다. 조금 더 보안적으로 안전하다고 볼 수도 있습니다. 혹은 **MCP**가 구체적으로 뭘하고 있는지 보고자하는 요건이 있는 경우 시도해볼 만합니다. 하지만 따로 **debug** 모드 같은걸 제공하고 있지는 않아서 별도의 디버깅 도구나 리버스 엔지니어링을 활용해야 합니다.

<Terminal박스>

```
$ claude mcp add context7 -- npx -y @upstash/context7-mcp --api-key YOUR_API_KEY
```

</Terminal박스>

따로 가입을 원하지 않으면, 다음과 같이 **SSE** 엔드포인트를 활용해서 **MCP**를 등록할 수도 있습니다. 이 **SSE** 엔드포인트는 과거 사용되던것으로 현재 공식적으로 지원되는 내용은 아니지만 아직까지 유효합니다. **MCP** 프로토콜이 이제 **stdio**와 **streamable HTTP**만을 지원하며, **SSE**는 **legacy**로 남게되었는데 점차 지원이 종료될것으로 예상됩니다.

<Terminal박스>

```
$ claude mcp add --transport sse context7 https://mcp.context7.com/sse
```

</Terminal박스>

Context7은 아래 2가지 기능만을 제공합니다.

- **resolve-library-id**
- **get-library-docs**

Context7은 내부적으로 라이브러리마다 고유한 **ID**를 부여해서 관리합니다.

**resolve-library-id**는 사용자의 프롬프트 및 환경에서 적절한 라이브러리 **ID**를 찾는데 활용됩니다. **get-library-docs**는 주어진 Context7 내부에 저장된 해당 라이브러리와 관련된 문서들을 가져오는데 사용됩니다.

## Context7 효과 확인하기

간단한 예제로 Context7 사용 전후를 비교해보겠습니다. 다른 MCP 동작과 섞이면 Context7의 효과를 살펴보기가 어려운 부분이 있습니다. 다른 MCP를 안쓰도록 하기 위해 아래와 같이 `claude`를 실행합니다. `--strict-mcp-config`는 명시적으로 주어진 MCP만을 허용해주도록 하는 플래그입니다. 또한 편의상 `--dangerously-skip-permissions`으로 따로 권한요청을 하지 않도록 하겠습니다.

<Terminal박스>

```
$ claude --strict-mcp-config --dangerously-skip-permissions
```

</Terminal박스>

이제 예제 작성을 주문합니다. 최근에 Anthropic은 Claude Agent SDK를 발표한바 있습니다. 간단한 Claude Agent SDK 예제 작성을 주문해보겠습니다.

<Claude Code박스>

> 간단한 **claude agent sdk** 예제를 작성해줘

● 간단한 Claude Agent SDK 예제를 작성해드리겠습니다.

● Write(claude\_agent\_example.py)

└ Wrote 143 lines to claude\_agent\_example.py

```
#!/usr/bin/env python3
```

```
"""
```

Simple Claude Agent SDK Example

This example demonstrates basic usage of the Claude API to create a simple agent

```
"""
```

```
import anthropic
```

```
import os
```

```
... +133 lines (ctrl+o to expand)
```

● Write(README.md)

└ Wrote 36 lines to README.md

```
# Claude Agent SDK Example
```



간단한 Claude API 사용 예제 모음입니다.

## 설치

```
```bash
```

```
pip install anthropic
```

```
```
```

... +26 lines (ctrl+o to expand)

🟡 Claude Agent SDK 예제를 작성했습니다:

claude\_agent\_example.py - 4가지 예제 포함:

1. 기본 채팅 에이전트
2. 시스템 프롬프트를 사용한 역할 정의
3. 다중 턴 대화 (대화 기록 유지)
4. 스트리밍 응답

실행 방법:

```
pip install anthropic
```

```
export ANTHROPIC_API_KEY='your-api-key'
```

```
python claude_agent_example.py
```

</Claude Code박스>

작성된 예제 전문은 다음과 같습니다.

<Terminal박스>

```
#!/usr/bin/env python3
```

```
"""
```

```
Simple Claude Agent SDK Example
```

```
This example demonstrates basic usage of the Claude API to create a  
simple agent
```

```
"""

import anthropic

import os


def simple_chat_agent():

    """Simple chat agent that responds to user messages"""

    # Initialize the Anthropic client

    client = anthropic.Anthropic(

        api_key=os.environ.get("ANTHROPIC_API_KEY")

    )

    # Create a simple conversation

    message = client.messages.create(

        model="claude-sonnet-4-5-20250929",

        max_tokens=1024,

        messages=[

            {"role": "user", "content": "Hello! What can you help me

with?"}

        ]

    )
```

```
print("Agent Response:")  
  
print(message.content[0].text)
```

[중략]

```
if __name__ == "__main__":  
  
    print("=== Claude Agent SDK Examples ===\n")  
  
    # Check if API key is set  
  
    if not os.environ.get("ANTHROPIC_API_KEY"):  
        print("Error: ANTHROPIC_API_KEY environment variable not  
set")  
  
        print("Set it with: export ANTHROPIC_API_KEY='your-api-key'")  
  
        exit(1)  
  
    # Run examples  
  
    try:  
  
        simple_chat_agent()  
  
        agent_with_system_prompt()  
  
        multi_turn_conversation()  
  
        agent_with_streaming()  
  
  
        print("\n=== All examples completed successfully! ===")
```

```
except Exception as e:

    print(f"\nError: {e}")
```

</Terminal박스>

전문을 다 살펴볼 필요없이 맨윗줄에 어떤 라이브러리 쓰고있는지 확인해봅니다.

<Terminal박스>

```
import anthropic
```

</Terminal박스>

이게 맞을까요? 이제 Anthropic에서 관련 안내를 살짝 살펴봅니다.

<https://docs.claude.com/ko/api/agent-sdk/overview>

## AGENT SDK

# Agent SDK 개요

📄 페이지 복사



Claude Agent SDK로 맞춤형 AI 에이전트 구축하기

❗ Claude Code SDK가 **Claude Agent SDK**로 이름이 변경되었습니다. 기존 SDK에서 마이그레이션하는 경우 [마이그레이션 가이드](#)를 참조하세요.

## 설치

TypeScript Python



```
pip install claude-agent-sdk
```

**claude-agent-sdk**가 쓰여야합니다! 아무래도 라이브러리 선택부터 잘못된것 같습니다!

이번엔 Context7과 함께해봅니다. 앞서 설정 과정을 봤지만 다시 정확히 Context7만 사용하기 위해 현재의 프로젝트 스코프로 Context7을 설정하기 실행하도록 합니다.

<Terminal박스>

```
$ claude mcp add -s project --transport http context7
https://mcp.context7.com/mcp --header "CONTEXT7_API_KEY:
YOUR_API_KEY"

$ claude --strict-mcp-config --mcp-config ./mcp.json
--dangerously-skip-permissions
```

</Terminal박스>

Context7이 사용되는걸 확인해봅니다.

/mcp로 Context7 설정이 문제 없는지 확인하고

```
> /mcp

Manage MCP servers

> 1. context7      ✓ connected · Enter to view details

MCP Config locations (by scope):
  • User config (available in all your projects):
    • /Users/gavin.jeong/.claude.json
  • Project config (shared via .mcp.json):
    • /Users/gavin.jeong/testtest1/.mcp.json
  • Local config (private to you in this project):
    • /Users/gavin.jeong/.claude.json [project: /Users/gavin.jeong/testtest1]

For help configuring MCP servers, see: https://docs.claude.com/en/docs/claude-code/mcp

Esc to exit
```

제공하는 `Tools`도 확인합니다.

```
> /mcp

Tools for context7 (2 tools)

> 1. resolve-library-id
   2. get-library-docs

Esc to go back
```

이제 똑같은 프롬프트를 입력해봅니다.

<Claude Code박스>

> 간단한 **claude agent sdk** 예제를 작성해줘

● Claude Agent SDK의 최신 문서를 확인하여 예제를 작성하겠습니다.

● context7 - resolve-library-id (MCP)(libraryName: "anthropic claude sdk")

└ Available Libraries (top matches):

Each result includes:

... +219 lines (ctrl+o to expand)

● context7 - get-library-docs (MCP)(context7CompatibleLibraryID: "/anthropics/claude-agent-sdk-python", topic: "getting started basic example", tokens: 3000)

└ =====

CODE SNIPPETS

=====

... +459 lines (ctrl+o to expand)

● 간단한 Claude Agent SDK 예제를 작성하겠습니다.

● Write(claude\_agent\_examples.py)

└ Wrote 168 lines to claude\_agent\_examples.py

"""

Claude Agent SDK 예제 모음

Requirements: pip install claude-agent-sdk

"""

import anyio

from claude\_agent\_sdk import (

query,

ClaudeAgentOptions,

ClaudeSDKClient,

... +158 lines (ctrl+o to expand)

● Write(README.md)

└ Wrote 55 lines to README.md

# Claude Agent SDK 예제

Claude Agent SDK를 사용한 간단한 파이썬 예제 모음입니다.

## 설치

```
```bash
pip install claude-agent-sdk
...

... +45 lines (ctrl+o to expand)
```

● Claude Agent SDK 예제를 작성했습니다:

claude\_agent\_examples.py - 5가지 예제 포함:

1. 기본 쿼리
2. 옵션을 사용한 쿼리
3. 파일 작업 도구 사용
4. 커스텀 도구 정의 및 사용
5. 대화형 클라이언트

실행 방법:

```
pip install claude-agent-sdk
python claude_agent_examples.py
```

각 예제는 주석으로 설명되어 있으며, 기본 사용법부터 고급 기능까지 단계별로 보여줍니다.

</Claude Code박스>

세션에서 **Context7**을 사용하여 라이브러리를 확인하고 관련 문서를 가져오는 과정이 포함된걸 알 수 있습니다. 이제 산출물을 확인해봅시다.

<Terminal박스>

```
"""

Claude Agent SDK 예제 모음

Requirements: pip install claude-agent-sdk

"""

import anyio

from claude_agent_sdk import (
```

```
    query,  
    ClaudeAgentOptions,  
    ClaudeSDKClient,  
    AssistantMessage,  
    TextBlock,  
    tool,  
    create_sdk_mcp_server,  
)
```

# 예제 1: 가장 기본적인 쿼리

```
async def basic_query():  
    """가장 간단한 Claude 쿼리 예제"""  
    print("=== 예제 1: 기본 쿼리 ===")  
    async for message in query(prompt="2 + 2는 무엇인가요?"):  
        if isinstance(message, AssistantMessage):  
            for block in message.content:  
                if isinstance(block, TextBlock):  
                    print(block.text)  
    print()
```

[중략]

# 메인 함수 - 모든 예제 실행

```
async def main():
```



```

"""모든 예제를 순차적으로 실행"""

await basic_query()

await query_with_options()

# await query_with_tools() # 실제 파일을 생성하므로 주석 처리

await custom_tool_example()

await interactive_client()


if __name__ == "__main__":

    # anyio를 사용하여 비동기 함수 실행

    anyio.run(main)

```

</Terminal박스>

다시 라이브러리 임포트 부분만 살펴봅니다.

<Terminal박스>

```

import anyio

from claude_agent_sdk import (

    query,

    ClaudeAgentOptions,

    ClaudeSDKClient,

    AssistantMessage,

    TextBlock,

    tool,

    create_sdk_mcp_server,

)

```

</Terminal박스>

`claude_agent_sdk`가 제대로 활용되고 있는걸 알 수 있습니다!

## Context7을 좀 더 이해해보기

Context7은 이렇게 라이브러리 정보를 제공해주는 MCP입니다. 이 정보들은 Context7에서 주기적으로 수집하여 관리가 되고 있습니다. 자연스럽게 Context7에서 관리되고 있지 않은 라이브러리는 인식할 수 없습니다. 웹페이지에서 어떤 라이브러리들을 현재 관리되고 있는지 볼 수 있습니다.

Context7





















MCP ServerSign In+ Add

Up-to-date documentation for LLMs and AI code editors

Copy the latest docs and code for any library — paste into Cursor, Claude, or other LLMs.

Search a library (e.g. Next, React)

POPULARTRENDINGRECENT

NAME	SOURCE	TOKENS	SNIPPETS	UPDATE	
Next.js	 /vercel/next.js	537K	3.2K	5 days	
Shadcn UI	 /shadcn-ui/ui	194K	1.1K	5 days	
Vercel AI SDK	 /vercel/ai	515K	2.2K	3 hours	
Better Auth	 /better-auth/better-auth	222K	1.3K	5 days	
Supabase	 /supabase/supabase	1.2M	4.5K	2 days	
Tailwind CSS	 /tailwindlabs/tailwindcss.com	559K	1.7K	5 days	
MongoDB	 /mongodb/docs	15.1M	148K	21 hours	
FastAPI	 /tiangolo/fastapi	133K	642	5 days	
LangGraph	 /langchain-ai/langgraph	620K	2K	5 days	
React	 /reactjs/react.dev	613K	2.3K	5 days	

43,071 LibrariesSee Tasks in Progress >

주요 오픈소스 라이브러리들은 관리대상으로 들어가있지만 혹시 없으면 따로 Add 버튼을 눌러 추가해줘야 합니다.


Context7


Personal ⌵ [Dashboard](#)


MCP Server + Add

### Add Libraries

Select and add your documentation source

 GitHub

 LLMs.txt

 Website

Use public/open-source GitHub repos with URLs like: <https://github.com/username/repo>

☐ Private Repo

Submit

퍼블릭 리포는 별도 플랜없이 등록가능하지만, 프라이빗 리포에 경우 유료 플랜 사용이 요구됩니다.

## Choose your plan

Everything is simple, choose what feels right for you

### Free

\$0

FEATURES

**Public Repos**  
**Access Control**

Choose plan

### Pro

**\$7 per seat / month**

FREE FEATURES

**Private Repos**  
**Team Collaboration**

Choose plan

### Enterprise

-

PRO FEATURES

**SOC-2**  
**SSO**

Contact us

Context7이 어떻게 라이브러리의 정보를 수집하고 어떤 정보를 제공하는지 조금 더 살펴보겠습니다. 웹페이지에서 각 라이브러리를 선택하면 수집, 정리된 정보들을 좀 더 자세히 볼 수 있습니다.

React

<https://github.com/reactjs/react.dev>

React.dev is the official documentation website for React, a ...

Latest ▾

✔ Completed

Tokens: 612,625

Snippets: 2,321

Update: 5 days ago

Benchmark: 74.2

SEARCH BY TOPIC

Show docs for...

Show Results

RESULT

Tokens:

Raw

Copy

Link

```
=====
CODE SNIPPETS
=====
TITLE: React Application Setup and Styling for Strict Mode Example
DESCRIPTION: This snippet provides the foundational setup for a React application,
including the main entry point, a root App component managing initial data with
`useState`, and basic CSS styling. This boilerplate demonstrates how a typical React
application is structured to run the `StoryTray` component examples.

SOURCE:
https://github.com/reactjs/react.dev/blob/main/src/content/reference/react/StrictMode.md#_
snippet_3

LANGUAGE: javascript
CODE:
```
import { createRoot } from 'react-dom/client';
import './styles.css';

import App from './App';
```

언뜻봐도 우측 스크롤바 크기가 꽤 작고 많은 정보를 제공하는걸 볼 수 있습니다. 달리 생각하면 **Context7**을 쓰면 에이전트가 생산하는 코드의 정확도는 올라가지만 그만큼 많은 토큰이 소모될 수 있다는걸 가늠할 수 있습니다. 사용되는 패턴을 생각해보면 토큰을 줄이기 위해선 **resolve-library-id** 과정을 생략시키는 방법도 생각해볼 수 있습니다. 미리 **Context7** 웹페이지에서 정확한 라이브러리 ID를 확인해서 프롬프트에 활용하면 도움이 될것입니다.

이 정보는 어떤 기준으로 수집될까요? 웹페이지에서 아래와 같은 정보를 확인할 수 있습니다.

#### Guidelines

- Context7 extracts code only from docs with extensions: `.md`, `.mdx`, `.html`, `.rst`, `.ipynb`
- The system processes code snippets in docs and ignores pages without code.

기본적으로 문서포맷의 파일에 명시된 코드만을 가져간다고 명시되어있습니다. 라이브러리를 작성할때 문서에 관련된 스니펫이나 예제가 잘작성되어있지 않으면 기대만큼 동작이 잘안될 수도 있겠습니다.

## 마무리

**Context7**이 어떻게 라이브러리의 정보를 수집하는지 보면, 언뜻 다양한 생각이 들게 하는 부분이 있습니다. 앞으로는 **AI**에서 잘사용될 수 있도록 라이브러리 리포를 관리하고 챙기는게 필요해지겠다라는 생각도들고 좀 더 확장해보면 언어던 플랫폼이던 **AI** 입장에서 생산성이 담보되는게 굉장히 중요한 포인트가 되겠다는걸 예상해볼 수 있습니다.

**AI**가 만들어내는 코드가 최신성을 보장하지 못하면, 그저 데모 수준에서 만족해야 합니다. 반대로, **Context7**처럼 실제 컨텍스트를 반영할 수 있다면 **AI**는 단순한 도움말 도구가 아니라 실질적인 생산성 엔진으로 도약할 수 있습니다.

**AI**와 함께하는 개발의 다음 단계는 단순히 더 똑똑한 모델을 기다리는 게 아닙니다. 우리의 맥락을 **AI**에 어떻게 효과적으로 주입할 것인가, 이게 엔지니어가 집중해야할 부분입니다. 달리 용어로 컨텍스트 엔지니어링이라고도 표현합니다. **Context7**은 그 해법 중 한면을 제시합니다. **AI**가 현실의 개발 환경을 온전히 이해하고, 최신 정보와 내부 컨텍스트를 기반으로 코드를 생성, 바로 실무에 투입 가능한 결과물로 바꿔주는 전환점이 되고 있습니다.