

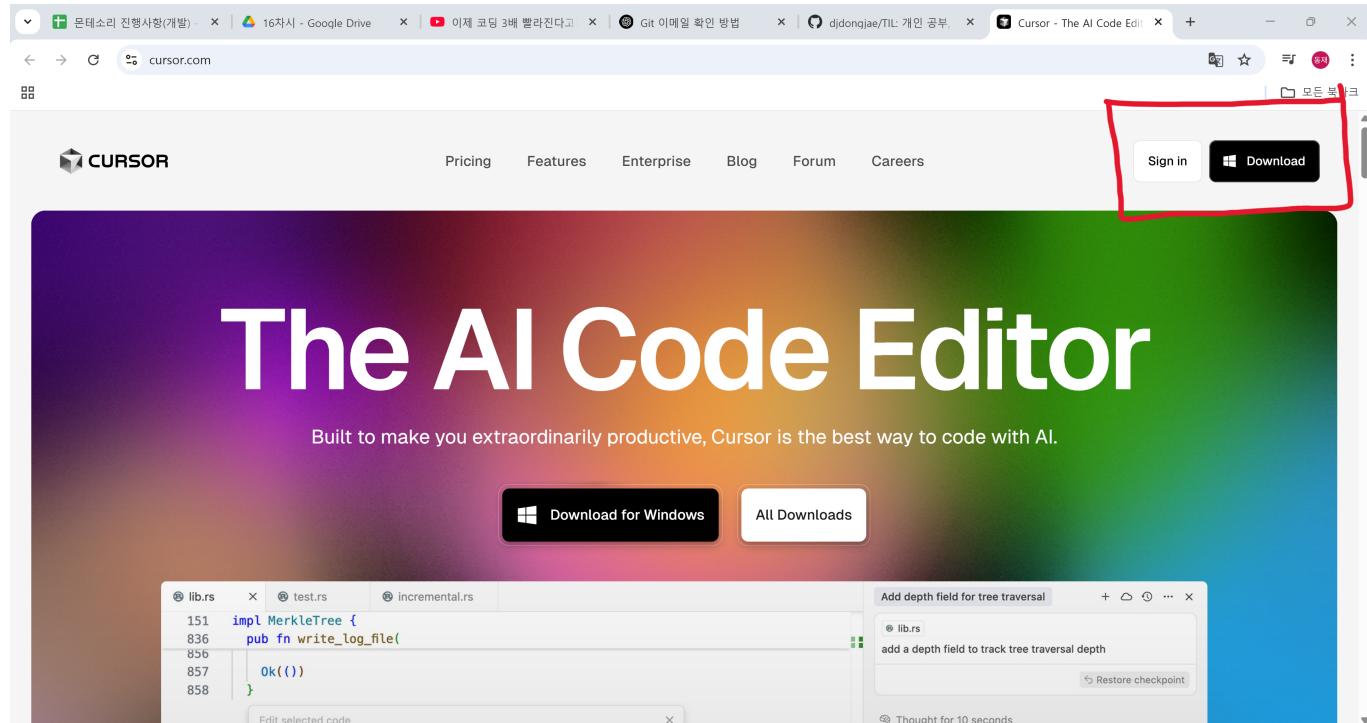
Cursor AI

1. 개요

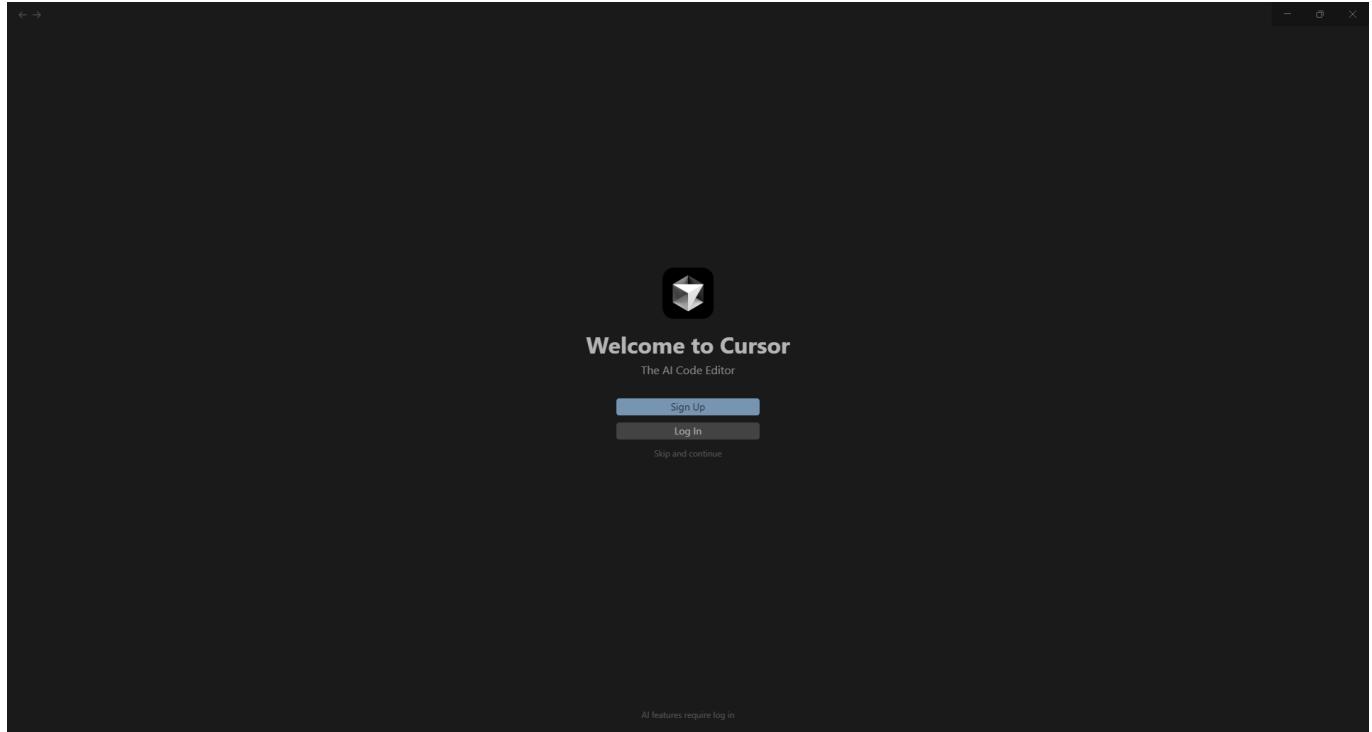
ChatGPT나 Claude와 같은 단순 대화형 AI의 경우 기존 코드의 흐름을 모른 상태에서 생성해 주는 코드를 일일히 복사하여 ide에 붙여넣는 번거로움이 존재했다. Cursor AI의 경우 코드의 맥락을 파악할 수 있는 **AI 기능이 완전히 통합된 코드 편집기**이다. 즉, 코딩을 하는 실시간으로 AI와 대화를 하며 코드 수정을 제안받고 적용할 수 있는 코드 편집기라는 뜻이다. 또한 Cursor AI는 VS Code 기반으로 제작되었기 때문에 기존에 이를 사용하던 사용자라면 쉽게 적응할 수 있는 인터페이스로 구성되어 있다.

2. Cursor AI 설치 및 세팅

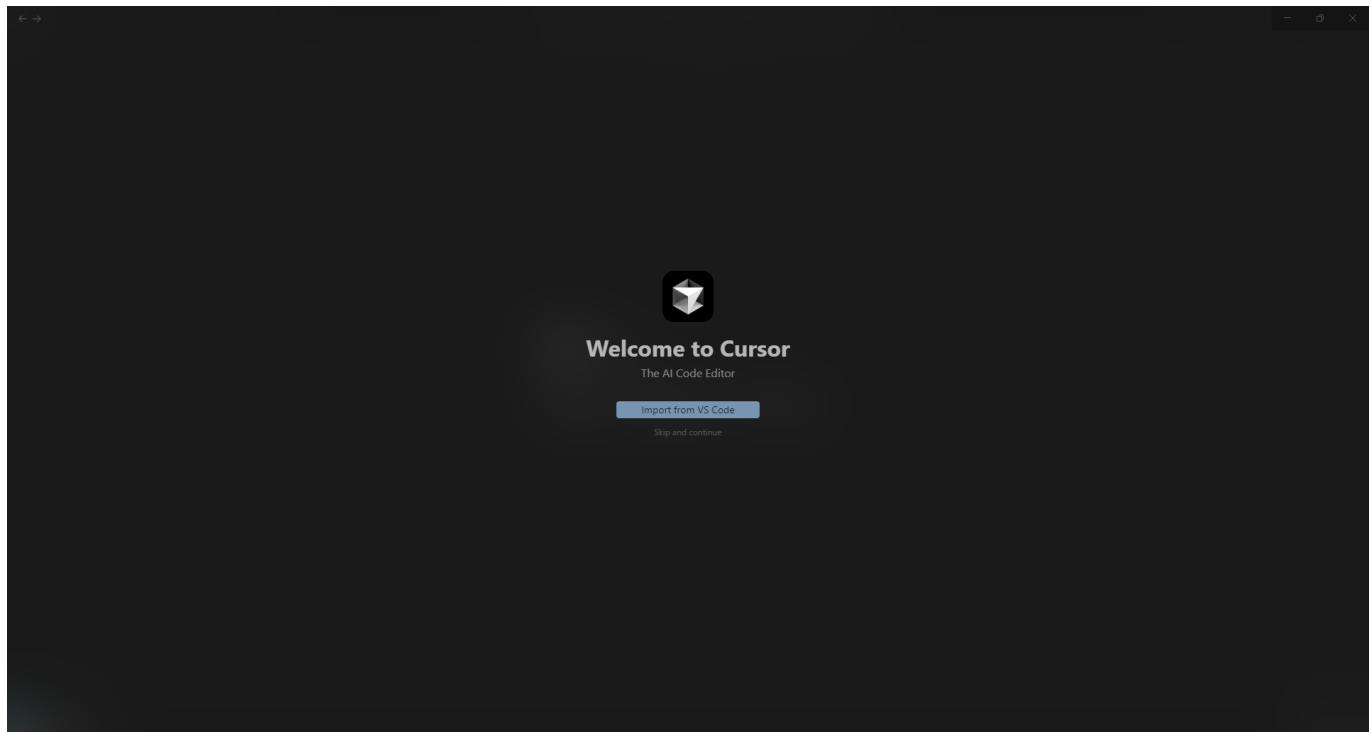
우선 [Cursor 공식 홈페이지](https://cursor.com)로 접속하여 우측 상단에 본인 OS에 맞는 Cursor AI를 다운로드 받는다.



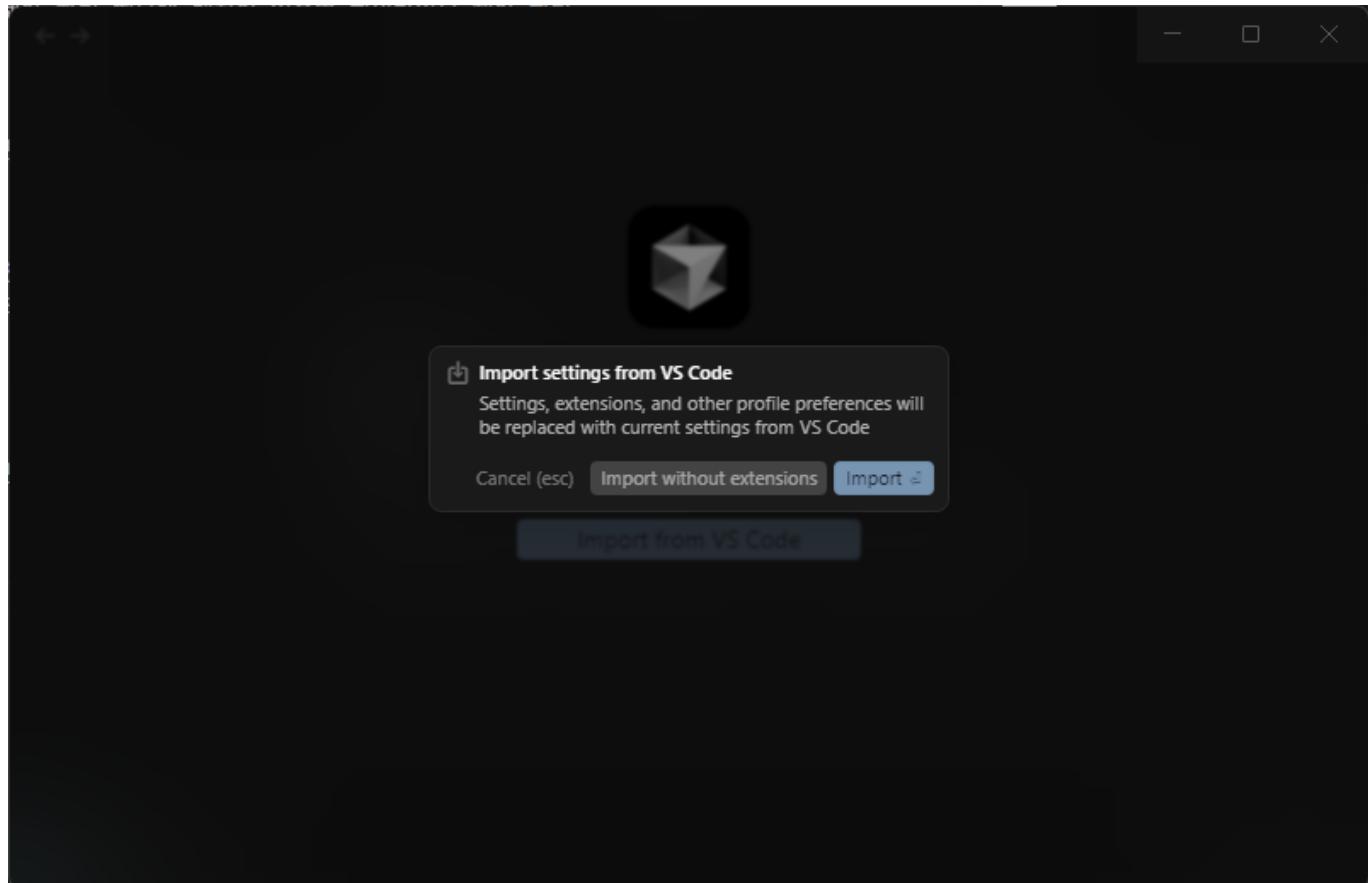
다운로드 받고 실행을 하면 다음과 같은 초기 화면이 등장한다. AI 기능을 사용하기 위해서는 로그인을 해야 하기 때문에 계정이 없으면 회원가입을 해야 한다.



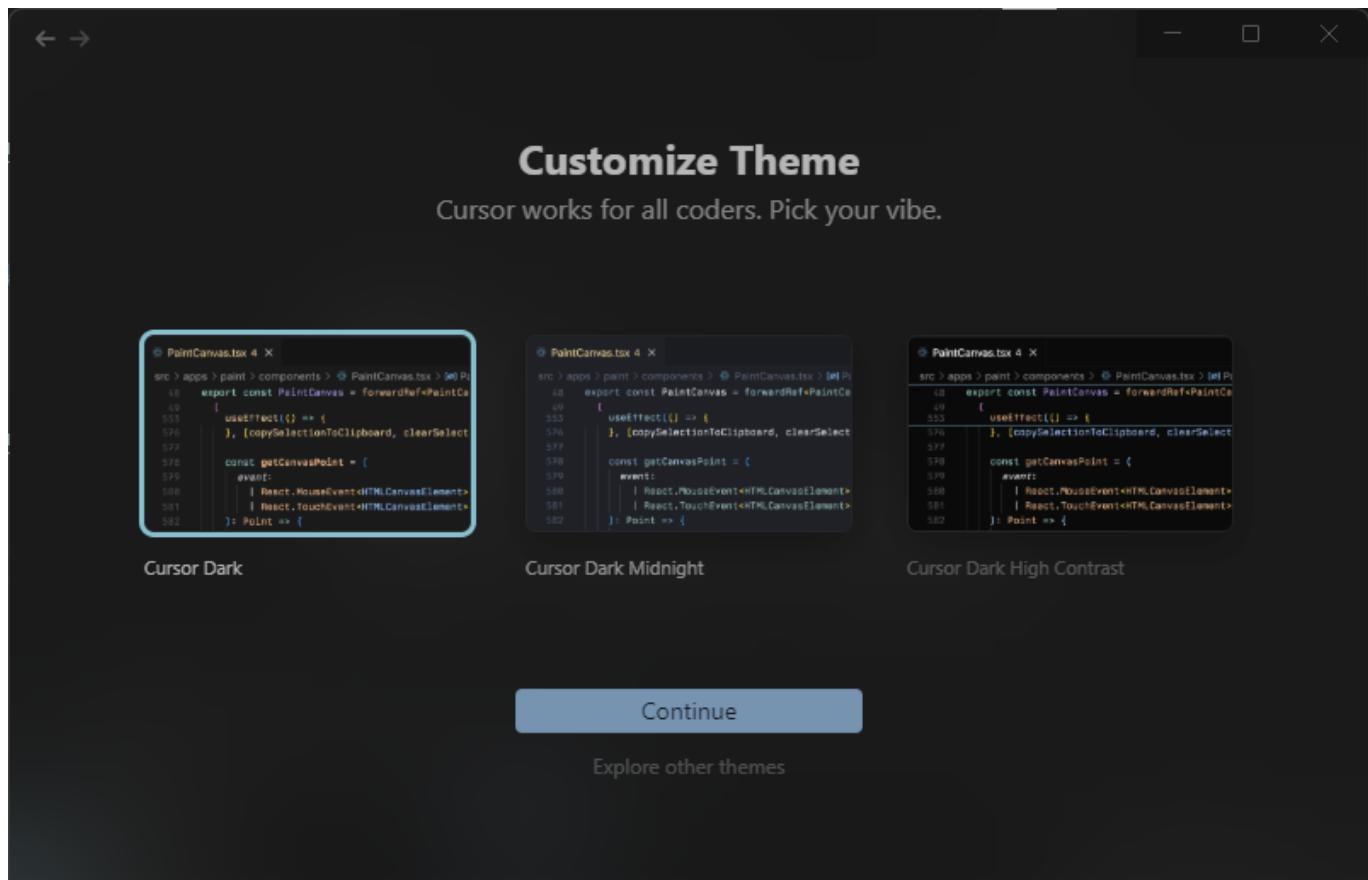
회원가입 또는 로그인을 마치고 나면 기존의 VS Code의 설정을 적용할 수 있는 버튼이 등장한다. 기존에 VS Code를 사용했던 사용자라면 해당 버튼을 클릭하고 아니라면 Skip 버튼을 클릭한다.



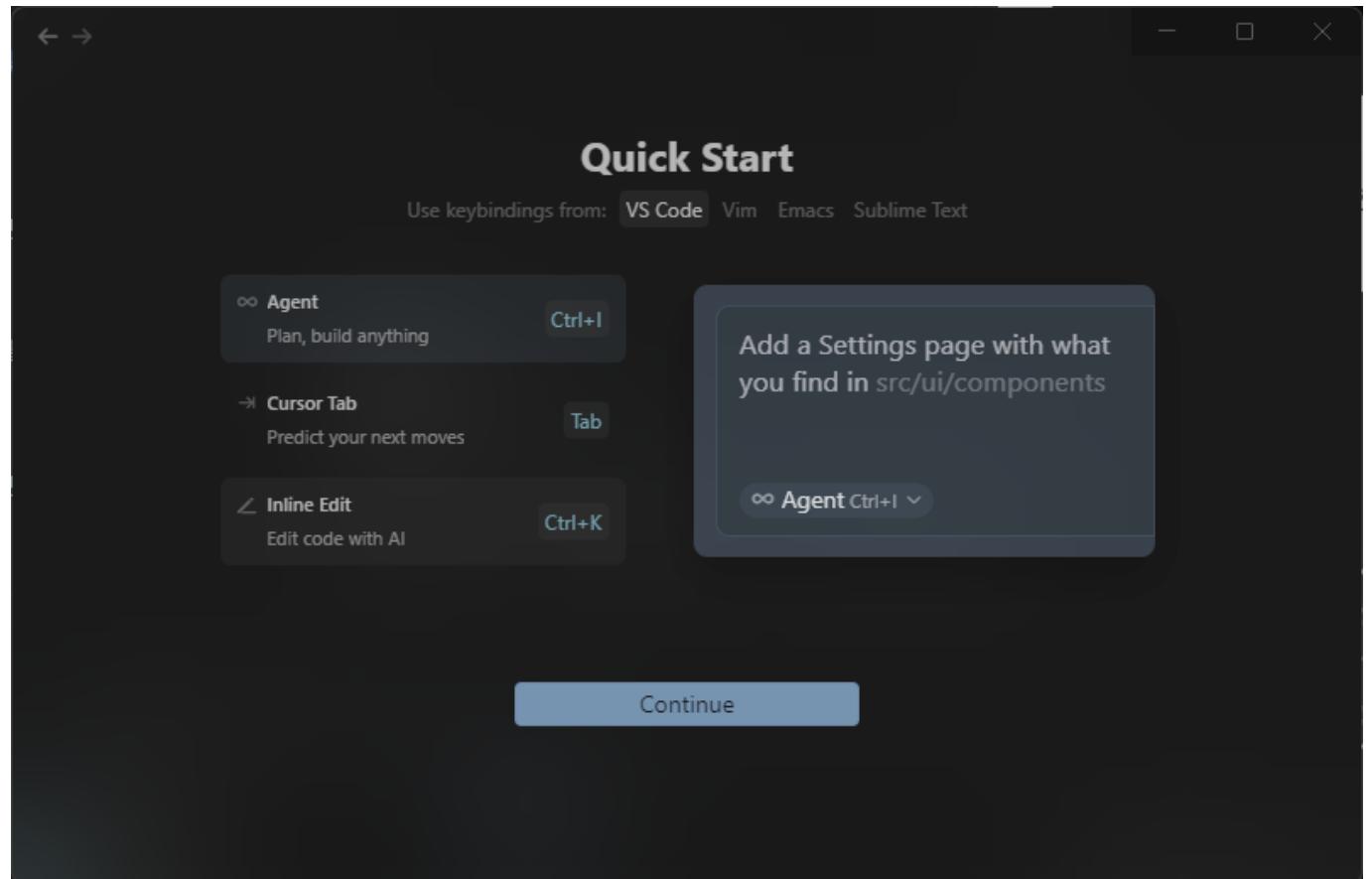
VS Code의 설정, 확장 프로그램, 프로필 등을 모두 적용하고 싶다면 import를 누르고 아니면 왼쪽 버튼을 클릭한다.



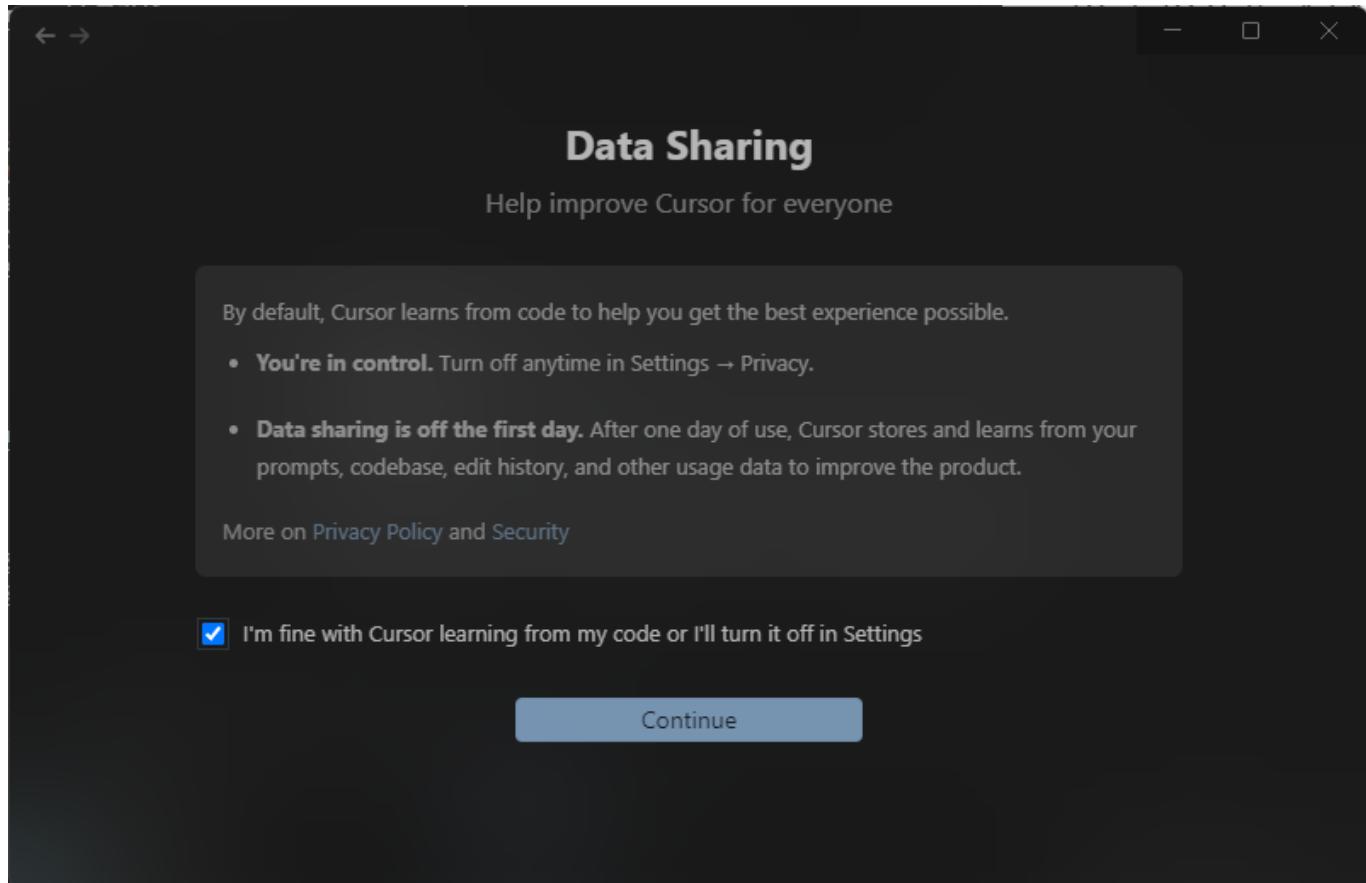
그리고 원하는 테마를 선택한다.



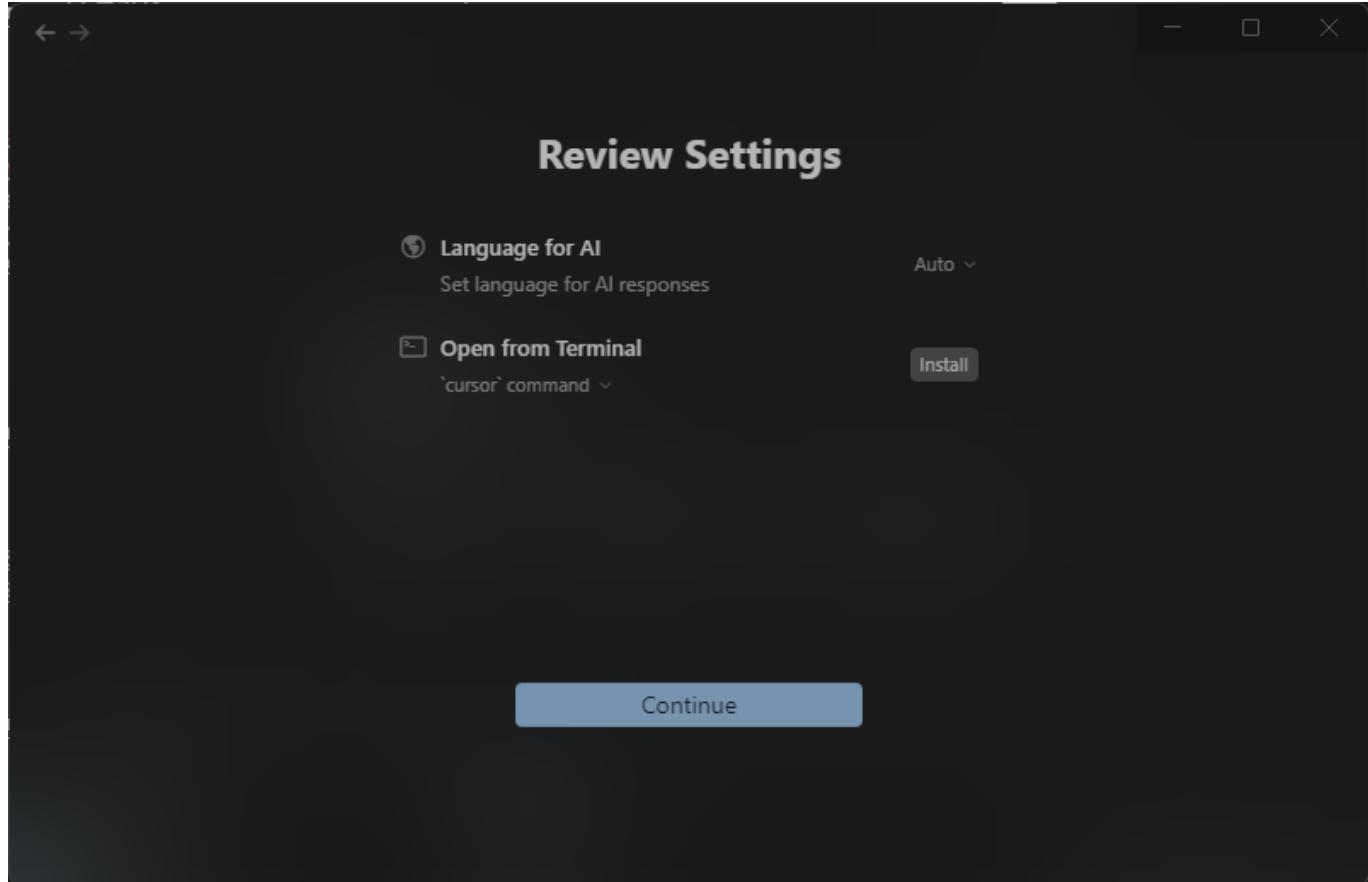
다음으로 '어떤 코드 편집기의 단축키를 사용할 것인가?'에 대한 선택을 할 수 있는 창이 등장한다. 원하는 코드 편집기의 단축키를 선택한다. 간단하게 기능을 설명하자면 **Agent** 기능의 경우 복잡한 코딩 작업을 최소한의 지침으로 스스로 탐색하고 계획해서 완료하는 AI 비서라고 볼 수 있다. **Cursor Tab**의 경우 코드 작성 시 다음 동작을 예측해 주는 기능이라고 보면 된다. **Ctrl+K**의 경우 원하는 영역을 지정하여 AI 기능을 활용해서 편집할 수 있는 기능이다.



다음으로 데이터 수집 및 공유에 대한 개인정보 동의에 관한 문서가 등장한다. 해당 설정은 추후에도 수정할 수 있으니 체크하고 넘어간다.

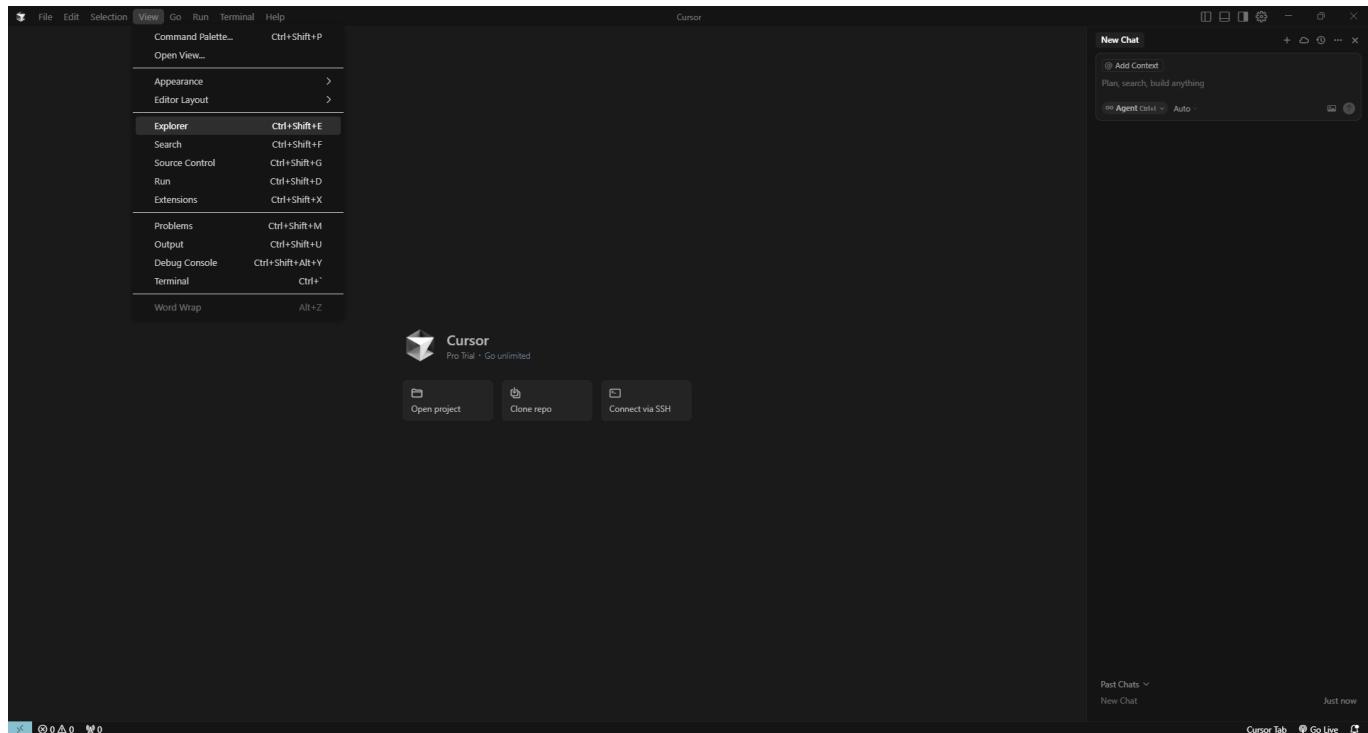


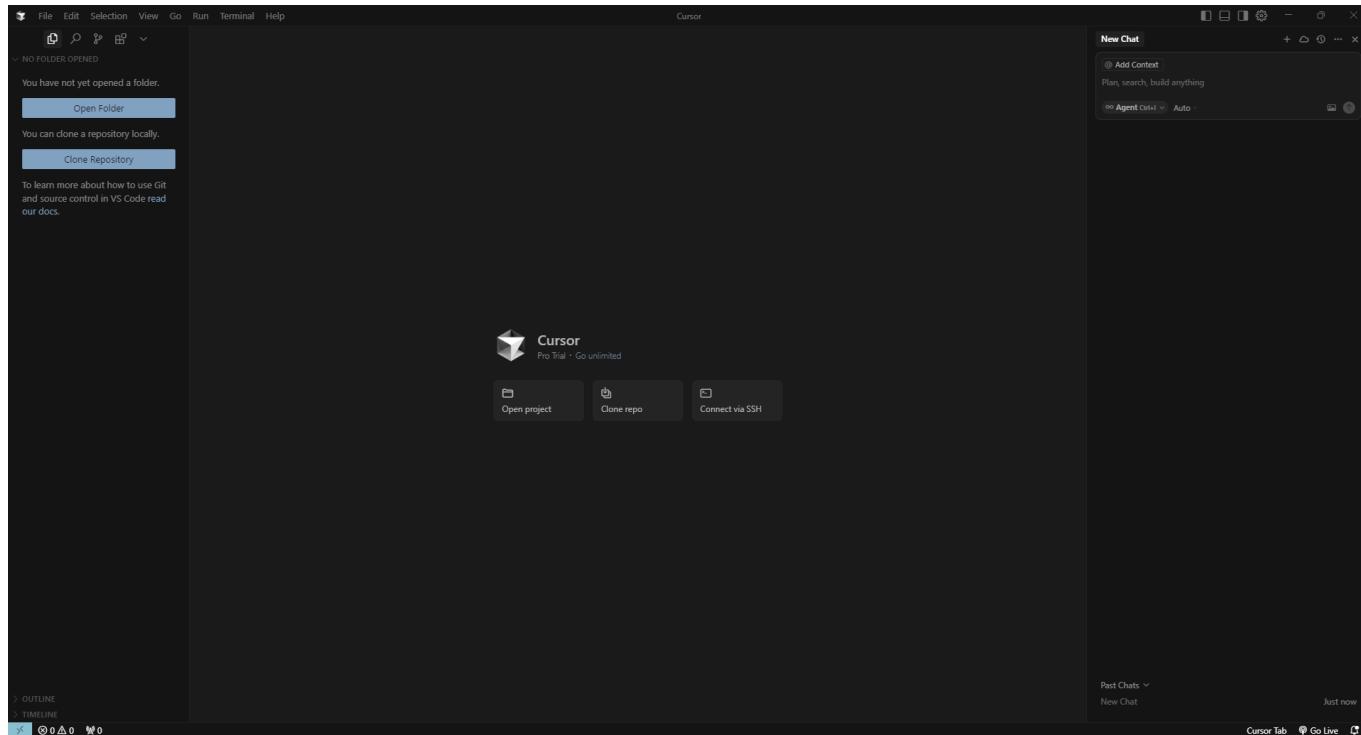
Cursor를 활용할 때 사용할 언어와 터미널에서 Cursor를 사용할 수 있는 명령어를 설치할 것인지 여부를 결정하는 창이 등장한다. 언어는 한국어를 선택하고 터미널 명령어 또한 설치를 진행한다.



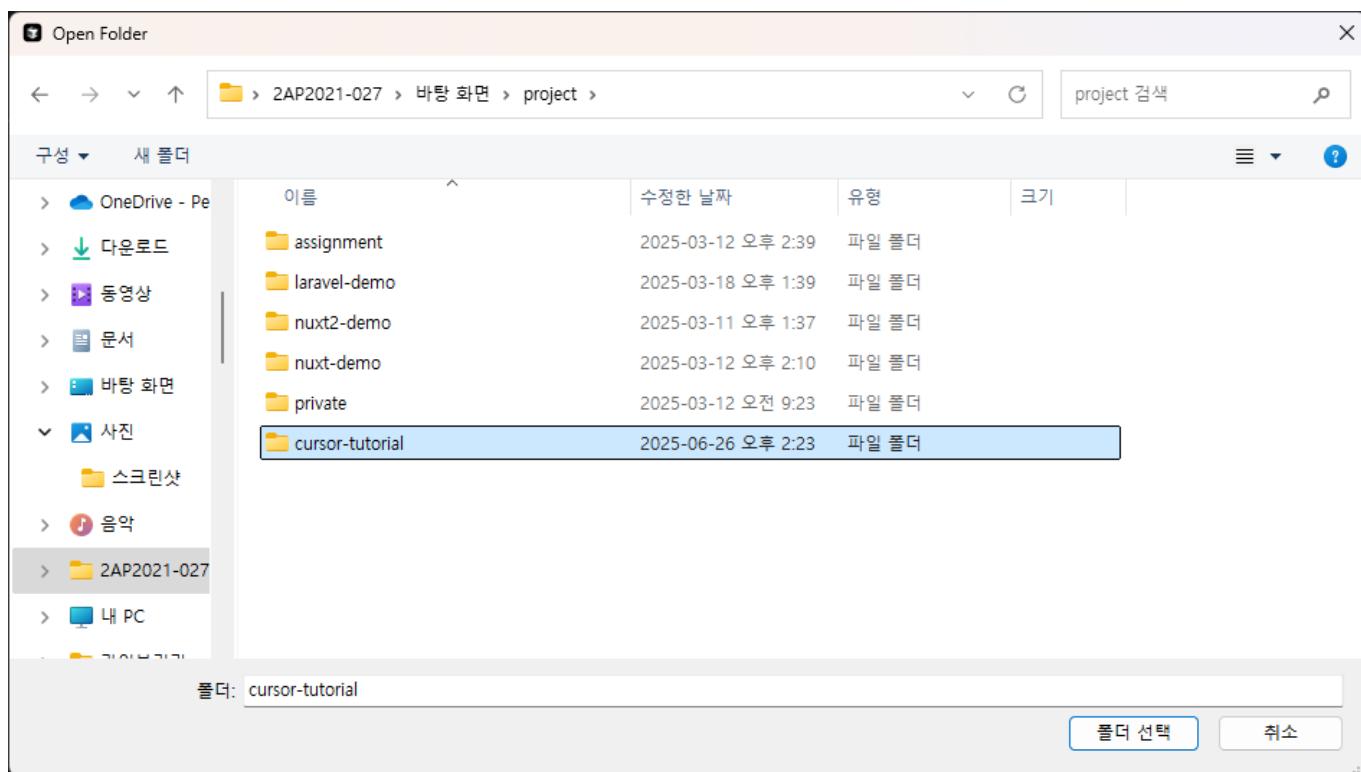
3. Cursor AI 툴 소개

모든 초기 설정을 마쳤다면 Cursor AI 화면이 등장하게 된다. 좌측 사이드 바를 열기 위해서는 다음과 같이 [View -> Explorer](#)를 클릭한다.

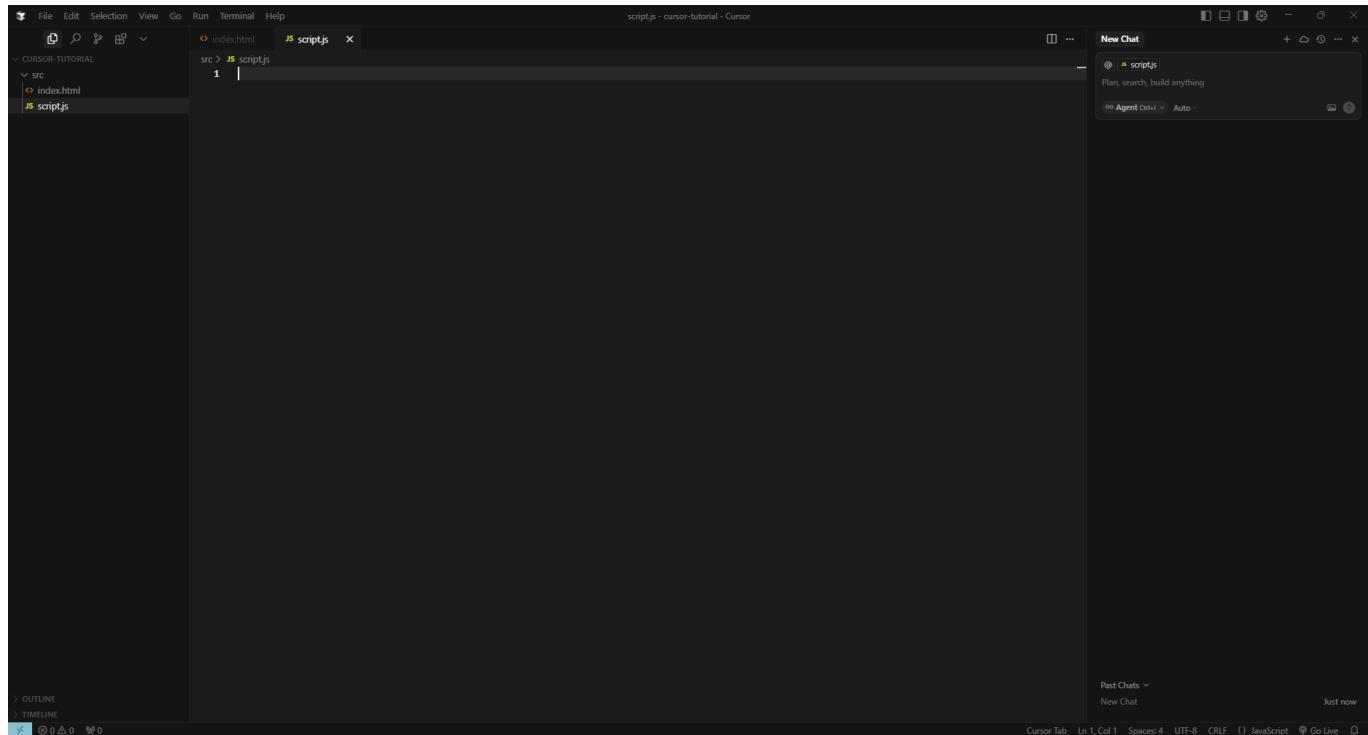




적당한 위치에 빈 프로젝트 폴더를 생성하고 **Open Folder** 버튼을 클릭하여 해당 폴더를 열어준다.

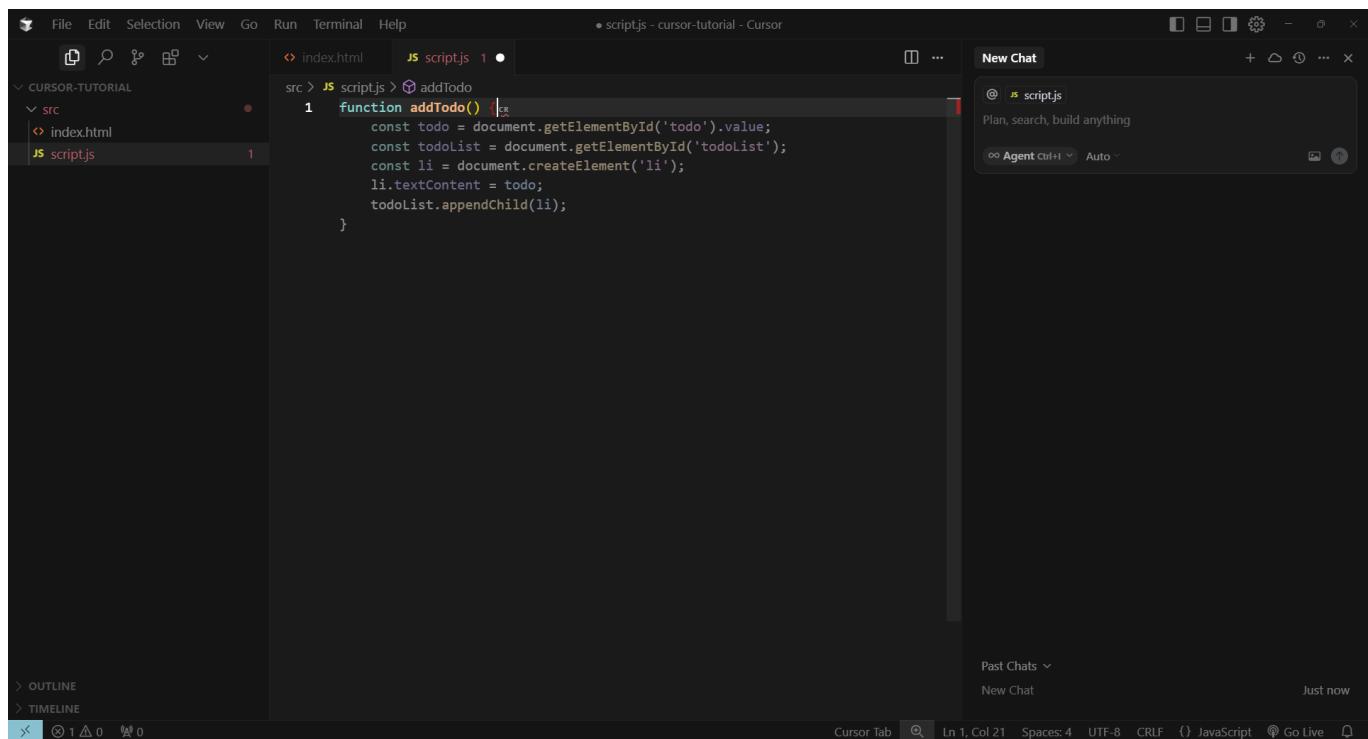


그리고 다음과 같이 src 폴더, 간단한 스크립트와 html 파일을 생성한다.



3.1 Cursor Tab 기능 소개

`function addTodo() {` 를 입력하고 tab키를 누르면 다음과 같이 Cursor가 제안한 코드를 완성하여 입력하는 장면을 볼 수 있다.



해당 기능을 잠시 Off하고 싶다면 하단의 Cursor Tab 버튼을 통해 설정을 변경할 수 있다.

A screenshot of a code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The current file is 'script.js' under the 'scriptjs' folder. The code in the editor is:

```

src > JS scriptjs > addTodo
1 function addTodo() {
2     const todo = document.getElementById("todo");
3     const todoText = todo.value;
4     if (todoText) {
5         const li = document.createElement("li");
6         li.textContent = todoText;
7         document.getElementById("todo-list").appendChild(li);
8     }
9 }

```

The bottom status bar shows 'Cursor Tab' and 'Ln 9, Col 2'. To the right of the editor is a 'New Chat' sidebar with a message input field and a 'Plan, search, build anything' placeholder.

Cursor 탭 기능을 잘 활용하기 위해서는 다음과 같이 주석으로 지시사항을 명확하게 작성하고 코드를 입력해야 한다. 즉 Cursor는 명확한 주석으로 맥락을 파악한다.

A screenshot of a code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The current file is 'script.js' under the 'src' folder. The code in the editor is:

```

src > JS scriptjs > filterCompleted
1 function addTodo() {
2     const todo = document.getElementById("todo");
3     const todoText = todo.value;
4     if (todoText) {
5         const li = document.createElement("li");
6         li.textContent = todoText;
7         document.getElementById("todo-list").appendChild(li);
8     }
9 }
10 // 완료된 할 일들만 필터링 하는 함수
11 function filterCompleted() {
12     const todoList = document.getElementById("todo-list");
13     const todos = todoList.getElementsByTagName("li");
14     for (let i = 0; i < todos.length; i++) {
15         if (todos[i].classList.contains("completed")) {
16             todos[i].style.display = "none";
17         }
18     }
}

```

The bottom status bar shows 'Cursor Tab' and 'Ln 12, Col 29'. To the right of the editor is a 'New Chat' sidebar with a message input field and a 'Plan, search, build anything' placeholder.

3.2 Cursor 인라인 편집 기능

인라인 편집 기능이란 **Ctrl + K** 단축키를 통해 원하는 코드 영역을 AI를 활용해 수정할 수 있도록 도와주는 기능이다. 먼저 다음과 같이 수정할 부분을 드래그를 통해 지정하고 **Ctrl + K**를 통해 요구사항을 입력한다.

```

1 function addTodo() {
2     const todo = document.getElementById("todo");
3     const todoText = todo.value;
4     if (todoText) {
5         const li = document.createElement("li");
6         li.textContent = todoText;
7         document.getElementById("todo-list").appendChild(li);
8     }
9 }
10
11 // 완료된 할 일들만 필터링 하는 함수
12 function filterCompleted() {
13     const todoList = document.getElementById("todo-list");
14     const todos = todoList.querySelectorAll("li");
15     todos.forEach(todo => {
16         if (todo.classList.contains("completed")) {
17             todo.style.display = "none";
18         } else {
19             todo.style.display = "block";
20         }
21     });
22 }

```

Past Chats < New Chat Just now

Cursor Tab Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript ⌂ Go Live ⌂

요구사항을 입력하고 엔터를 누르면 다음과 같이 코드의 수정 제안을 미리보기 형식을 통해 제공해주는 화면을 확인할 수 있다. 이때 **Ctrl + Shift + Y**를 누르면 제안 반영, **Ctrl + N**을 누르면 제안을 거부한다.

```

1 function addTodo() {
2     const todo = document.getElementById("todo");
3     const todoText = todo.value;
4     if (todoText) {
5         const li = document.createElement("li");
6         li.textContent = todoText;
7         document.getElementById("todo-list").appendChild(li);
8     }
9 }
10
11 // 유효성 검증
12 if (!todoText) {
13     alert("할 일을 입력해주세요!");
14     todo.focus();
15     return;
16 }
17
18 // 중복 검사
19 const existingTodos = document.querySelectorAll("#todo-list li");
20 for (let existingTodo of existingTodos) {
21     if (existingTodo.textContent === todoText) {
22         alert("이미 존재하는 할 일입니다!");
23         todo.focus();
24         return;
25     }
26 }
27
28 // 할 일 추가
29 const li = document.createElement("li");
30 li.textContent = todoText;
31 document.getElementById("todo-list").appendChild(li);
32
33 // 투명 필드 초기화
34 todo.value = "";
35 todo.focus();
36 }

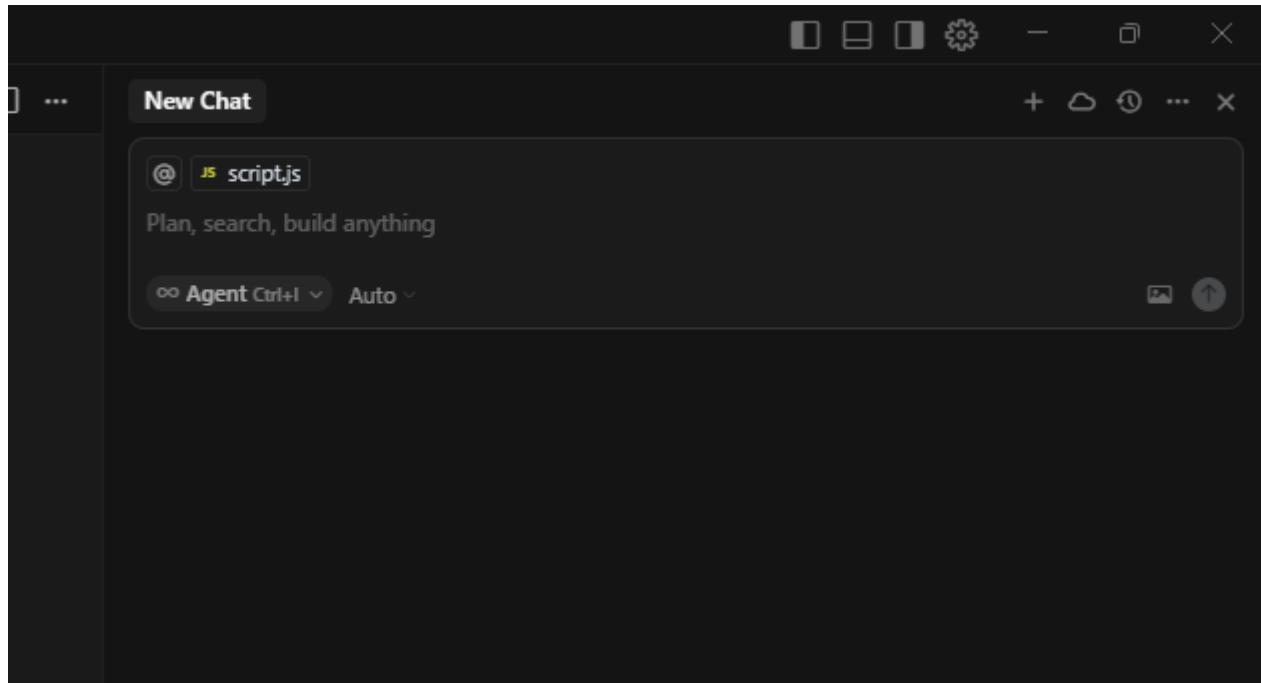
```

Past Chats < New Chat Just now

Cursor Tab Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript ⌂ Go Live ⌂

3.3 Cursor Chat 기능

말 그대로 자연어를 통해 Cursor AI와 직접 대화할 수 있는 기능이다. 채팅 기능을 제대로 활용하기 위해서는 Cursor에서 **Context**가 가지는 의미가 무엇인지 정확하게 알 필요가 있다.



Ctrl + i를 통해 채팅을 열 수 있다. **Context**를 추가하기 위해서는 **@**를 활용하여 필요한 파일, 문서 등을 추가 한다.

