Instructions on setting up rules in Cursor

This guide shows you 3 ways to create standard Cursor Rules (.mdc) files in 2026 that help AI deeply understand your project, write clean code, and adhere to all programming rules.

Cursor is currently rising to become the leading code editor tool thanks to its ability to deeply understand project source code. One of its most powerful features, often overlooked by new users, is **Cursor Rules** . Setting these rules helps AI not only write code with correct syntax but also adhere to specific coding conventions, directory structures, and libraries tailored to your needs.

This article will guide you on how to master files `.mdc`to turn Cursor into a true partner, understanding your every line of code.

What are Cursor Rules and why are they important?

Before we get into the implementation steps, we need to understand the nature of Cursor Rules. Basically, these are instructions stored in a special Markdown format (`.mdc`). These files are located in the `.cursor/rules/` project directory.

When you set rules, Cursor's AI (such as Claude 4.5 Sonnet or GPT -5) will prioritize reading these files before providing answers. This effectively solves the problem of AI frequently suggesting outdated or inappropriate code snippets that don't fit your company's existing structure.

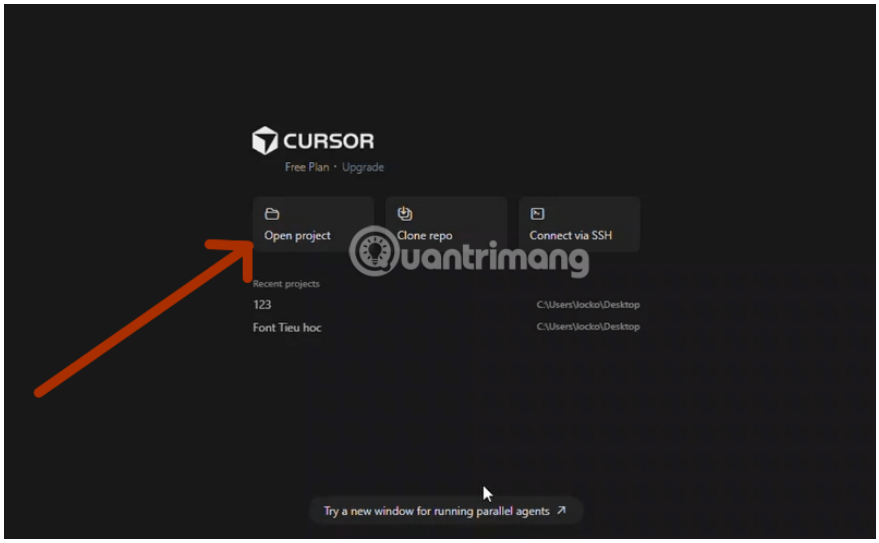
3 Ways to Create the Latest Cursor Rules in 2026

Currently, Cursor has simplified the rule creation process so that users don't need in-depth knowledge of metadata structures to create rules.

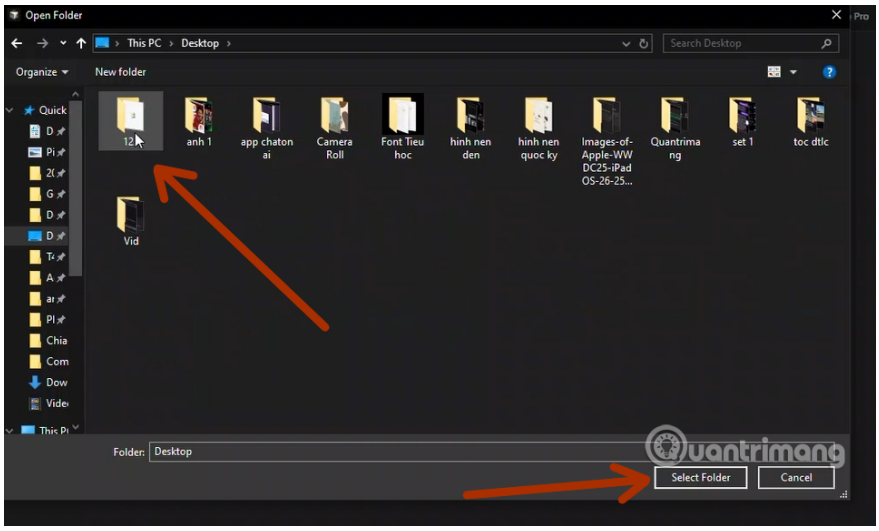
Method 1: Using Automated Chat (Recommended)

This is the fastest and smartest way because AI will automatically predict the necessary metadata based on your requirements.

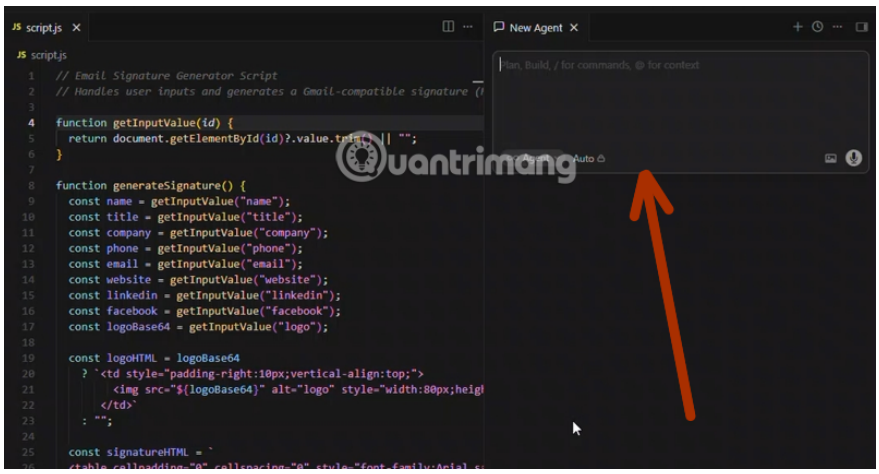
Step 1: Open the project: Launch the Cursor application and open your project folder (`File > Open Folder`).



Select the folder you want to include in Cursor.



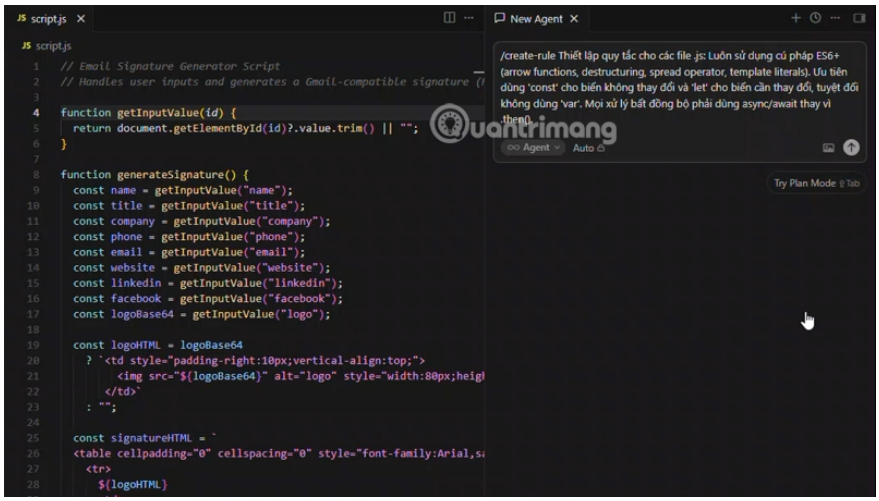
Step 2: Open the Chat window: Press the key combination `Ctrl + L`(Windows/Linux) or `Cmd + L`(Mac) to activate the AI ??Chat control panel on the right side. This control panel may also be located elsewhere in the Cursor interface.



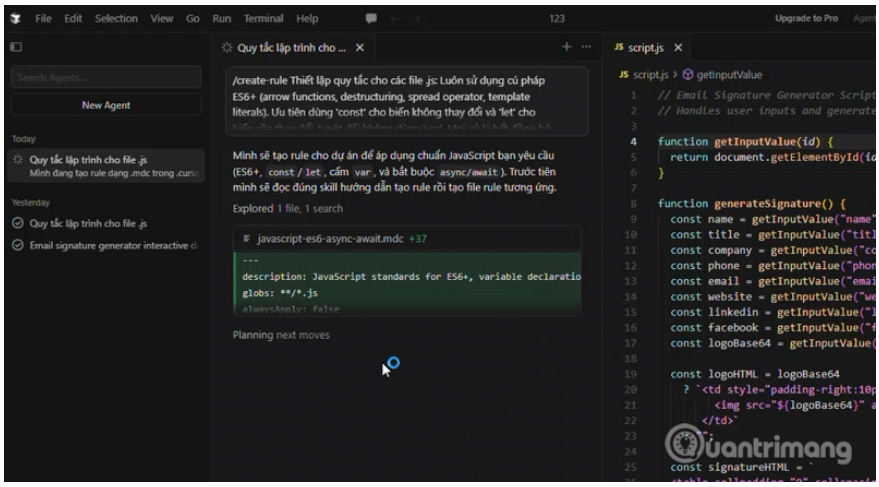
Step 3: Enter the command: Type the command using the following structure: `/create-rule [n?i dung quy t?c]`.

For example:

```
/create-rule Luôn s? d?ng Tailwind CSS cho các thành ph?n UI và ?u tiên dùng Lucide React cho icon.
```

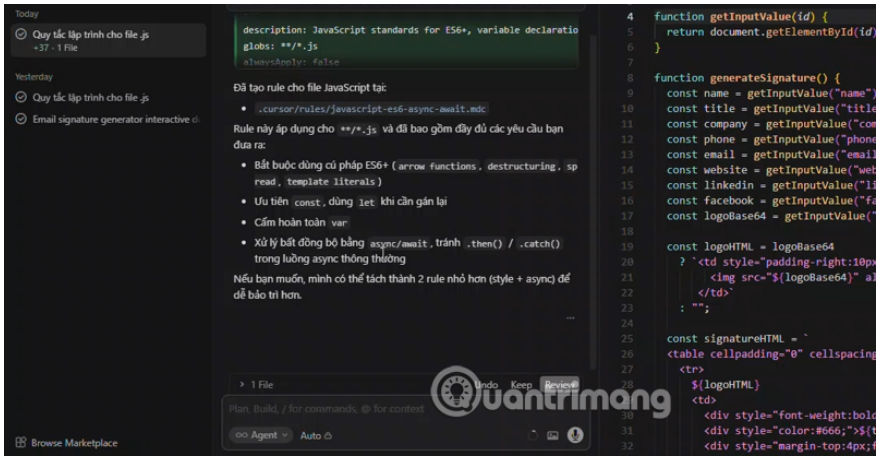


Step 4: Confirm: Press `Enter`. Cursor AI will automatically create a new file in the folder `.cursor/rules/`with the normalized filename and the Metadata Header (including rule name, description, and affected pattern files).



Step 5: Save the rule:

Click the **Review** button on the chat interface to check the content. You can request additional rules based on Cursor suggestions. The created rules will be automatically saved in the folder you opened in step 1.



Method 2: Using the Command Palette

If you prefer using keyboard shortcuts to optimize your workflow:

1. Press `Ctrl + Shift + P`(Windows/Linux) or `Cmd + Shift + P`(Mac).
2. Type the keyword **"Cursor Rules: Create New Rule"** .
3. Cursor will open an editor for a `.mdc`blank file with suggestions for metadata structure. You just need to fill in the information and save it.

Method 3: Create manually by hand

This method is for engineers who want 100% control over the rule structure:

1. Create a folder with that name `.cursor`in the root directory of your project (if it doesn't already exist).
2. Inside that folder, create another subfolder named `rules`.
3. Create a new file with the format `.mdc`. For example: `typescript-standard.mdc`.
4. Write content using this structure:

```
--- description: Quy t?
c dành cho các file TypeScript globs: **/*.ts, **/*.tsx --- # N?
i dung quy t?c t?i ?ây - S? d?ng Interface thay vì Type cho các ??nh ngh?
a Object. - Luôn export t??ng minh các hàm x? lý logic.
```

The standard structure of a professional .mdc file.

An effective Cursor Rule file is more than just a list of statements. It needs to have a structure so that the AI ?? understands when to trigger that rule.

1. **Frontmatter (Metadata) section:** Located between the two dots `---`.

    1. `description`Briefly explain the purpose of the rule.

2. `globs`: Specify the file types to which this rule applies (e.g., `src/components/*.tsx`).

2. **Content (Markdown):** Use H1, H2 tags and bullet points to structure the instructions. You should provide both "Good" and "Bad" examples so the AI ??can more accurately mimic them.

How to optimize Cursor Rules for large projects

When your project grows to hundreds of files, having too many overlapping rules can cause the AI ??to become overwhelmed. Here's a management strategy:

Classification of rules by scope

Instead of creating one huge rule file for the entire project, break it down into smaller parts:

1. **Global Rules:** Rules regarding the language (e.g., Strict TypeScript).
2. **Feature Rules:** Specific rules for each feature (e.g., payment processing rules, user authentication rules).
3. **Library Rules:** Specific guidelines for working with third-party libraries such as ShadcnUI, Redux, or Prisma.

Use the "Context Mention" feature.

In Cursor 2026, you can mention a specific rule during a chat by typing a character `@`and selecting the rule's name. This allows you to force the AI ??to adhere to a specific standard for sensitive code without it having to scan all other rules.

Common mistakes when setting up Cursor Rules

Although very powerful, if set up incorrectly, Cursor Rules can become a "double-edged sword" that slows down your progress:

1. **The rules are too vague:** Statements like "Write clean code" or "Optimize performance" are often ineffective. Be more specific with quantitative guidelines such as "Functions should not exceed 30 lines" or "Always use `useCallback`for functions passed to memoized components."
2. **Globs conflict:** When two rule files apply to the same file but provide conflicting instructions. Cursor will prioritize the file with more specific metadata, but it's best to double-check the `globs`.
3. **Forgetting to update rules:** When technology changes (e.g., upgrading from Next.js 14 to a newer version), remember to update the file `.mdc`. The AI ??won't automatically know that the old knowledge in your rules is outdated unless you fix it.