/sp-using-superpowers

Source: `~/.claude/skills/sp-using-superpowers/SKILL.md`

---

name: using-superpowers description: Use when starting any conversation - establishes how to find and use skills, requiring Skill tool invocation before ANY response including clarifying questions

If you think there is even a 1% chance a skill might apply to what you are doing, you ABSOLUTELY MUST invoke the skill.
IF A SKILL APPLIES TO YOUR TASK, YOU DO NOT HAVE A CHOICE. YOU MUST USE IT.

This is not negotiable. This is not optional. You cannot rationalize your way out of this.
</EXTREMELY-IMPORTANT>

How to Access Skills

In Claude Code: Use the `Skill` tool. When you invoke a skill, its content is loaded and presented to you—follow it directly. Never use the Read tool on skill files.

In other environments: Check your platform's documentation for how skills are loaded.

Using Skills

The Rule

Invoke relevant or requested skills BEFORE any response or action. Even a 1% chance a skill might apply means that you should invoke the skill to check. If an invoked skill turns out to be wrong for the situation, you don't need to use it.

```
digraph skill_flow {
    "User message received" [shape=doublecircle];
    "Might any skill apply?" [shape=diamond];
    "Invoke Skill tool" [shape=box];
    "Announce: 'Using [skill] to [purpose]'" [shape=box];
    "Has checklist?" [shape=diamond];
    "Create TodoWrite todo per item" [shape=box];
    "Follow skill exactly" [shape=box];
    "Respond (including clarifications)" [shape=doublecircle];

    "User message received" -> "Might any skill apply?";
    "Might any skill apply?" -> "Invoke Skill tool" [label="yes, even 1%"];
    "Might any skill apply?" -> "Respond (including clarifications)" [label="definitely not"];
    "Invoke Skill tool" -> "Announce: 'Using [skill] to [purpose]'";
    "Announce: 'Using [skill] to [purpose]'" -> "Has checklist?";
    "Has checklist?" -> "Create TodoWrite todo per item" [label="yes"];
    "Has checklist?" -> "Follow skill exactly" [label="no"];
    "Create TodoWrite todo per item" -> "Follow skill exactly";
}
```

Red Flags

These thoughts mean STOP—you're rationalizing:

Thought	Reality
"This is just a simple question"	Questions are tasks. Check for skills.
"I need more context first"	Skill check comes BEFORE clarifying questions.

Thought	Reality
"Let me explore the codebase first"	Skills tell you HOW to explore. Check first.
"I can check git/files quickly"	Files lack conversation context. Check for skills.
"Let me gather information first"	Skills tell you HOW to gather information.
"This doesn't need a formal skill"	If a skill exists, use it.
"I remember this skill"	Skills evolve. Read current version.
"This doesn't count as a task"	Action = task. Check for skills.
"The skill is overkill"	Simple things become complex. Use it.
"I'll just do this one thing first"	Check BEFORE doing anything.
"This feels productive"	Undisciplined action wastes time. Skills prevent this.
"I know what that means"	Knowing the concept ≠ using the skill. Invoke it.

Skill Priority

When multiple skills could apply, use this order:

1. **Process skills first** (brainstorming, debugging) - these determine HOW to approach the task
2. **Implementation skills second** (frontend-design, mcp-builder) - these guide execution

"Let's build X" → brainstorming first, then implementation skills. "Fix this bug" → debugging first, then domain-specific skills.

Skill Types

Rigid (TDD, debugging): Follow exactly. Don't adapt away discipline.

Flexible (patterns): Adapt principles to context.

The skill itself tells you which.

User Instructions

Instructions say WHAT, not HOW. "Add X" or "Fix Y" doesn't mean skip workflows.

Revision #5
Created 2026-02-18 08:39:56 UTC by John
Updated 2026-07-05 20:01:24 UTC by John