# Cursor IDE

## Teaching an AI to Teach Itself

A deep dive into Cursor's self-learning capabilities through `.cursor/rules`

## What We'll Cover

1. What is Cursor?
2. Understanding Cursor Rules
3. The Power of Self-Learning
4. Live Demo
5. Best Practices & Tips

## What is Cursor?

An AI-first IDE that revolutionizes how we write code

- 💡 Intelligent Code Completion
- 🔍 Context-Aware Understanding
- 🤝 Natural Language Interaction
- 📚 Documentation Generation
- 🧠 Self-Learning Capabilities

## Why Cursor?

- **Faster Development**: AI assistance speeds up coding
- **Smarter Suggestions**: Understands project context
- **Learning System**: Improves over time
- **Customizable**: Adapts to your codebase
- **Open Source**: Community-driven improvements

## Understanding Cursor Rules

The `.cursor/rules` system: Your AI's knowledge base

# What are Cursor Rules?

- Project-specific guidelines
- AI behavior modifiers
- Codebase documentation
- Learning framework

## Rule Types

1. **Auto-attached**
   - Triggered by file patterns
2. **Agent-requested**
   - Used when relevant
3. **Always**
   - Applied to every interaction
4. **Manual**
   - Explicitly referenced

## Rule Structure

```
---
description: Rule purpose
globs: ["*.ts", "*.tsx"]
type: auto_attached
---

# Rule Title

## Context
What the rule addresses

## Guidelines
Specific instructions

## Examples
Code samples
```

# The Power of Self-Learning

How Cursor evolves with your project

1. 🔍 Identify patterns & errors
2. 📝 Document solutions
3. 🏗️ Create new rules
4. 🔄 Apply & refine

## Error Analysis Framework

```
1. Identify root cause
2. Document incorrect approach
3. Record correct solution
4. Create/update rules
```

## Rule Categories

- Type Errors
- Integration Issues
- Performance Patterns
- Dependency Management
- Project Conventions

## Benefits of Self-Learning

- 📈 Continuous improvement
- 🎯 More accurate suggestions
- 🚀 Faster development
- 🛠️ Reduced errors
- 📚 Growing knowledge base

## Live Demo

Watch Cursor learn and create new rules

1. Encounter a common error
2. Analyze the problem
3. Create a new rule
4. Test the improvement

## Example Rule Creation

```
---
description: Handle TypeScript enum type safety
globs: ["*.ts", "*.tsx"]
type: auto_attached
---

# TypeScript Enum Best Practices

## Problem
Inconsistent enum usage leading to type errors

## Solution
Standardized enum pattern with type safety

## Examples
Before/After code samples
```

## Getting Started

1. Install Cursor IDE
2. Create `.cursor/rules` directory
3. Add your first rule
4. Watch it learn and grow

## Resources

- Cursor IDE Website
- Documentation
- GitHub Repository
- Community Discord

## Thank You!

Start small with rules, and let them grow naturally as you encounter new patterns and challenges

cursor.sh