Introduction
What is Angular?
Installation
Essentials
Start coding!
In-depth Guides
Signals
Components
Templates
Directives
Dependency Injection Updated
Routing Updated
Forms
HTTP Client
Server-side & hybrid-rendering
Testing
Internationalization
Animations Updated
Drag and drop
Build with AI New
Get Started
LLM prompts and AI IDE setup
Design Patterns
Angular CLI MCP Server setup
Developer Tools
Angular CLI
Libraries
DevTools
Language Service
Best Practices
Style Guide Updated
Security
Accessibility
Unhandled errors in Angular
Performance
Keeping up-to-date
Developer Events
Angular v21 Release New
Extended Ecosystem
NgModules
Legacy Animations
Using RxJS with Angular
Service Workers & PWAs
Web workers
Custom build pipeline
Tailwind New
Angular Fire
Google Maps
Google Pay
YouTube player
Angular CDK
Angular Material

LLM prompts and AI IDE setup

Generating code with large language models (LLMs) is a rapidly growing area of interest for developers. While LLMs are often capable of generating working code it can be a challenge to generate code for consistently evolving frameworks like Angular.

Advanced instructions and prompting are an emerging standard for supporting modern code generation with domain specific details. This section contains curated content and resources to support more accurate code generation for Angular and LLMs.

On this page

Custom Prompts and System Instructions
Rules Files
Angular CLI MCP Server setup
Providing Context with llms.txt
Web Codegen Scorer

Custom Prompts and System Instructions

Improve your experience generating code with LLMs by using one of the following custom, domain specific files.

> **NOTE:** These files will be updated on a regular basis staying up to date with Angular's conventions.

Here is a set of instructions to help LLMs generate correct code that follows Angular best practices. This file can be included as system instructions to your AI tooling or included along with your prompt as context.

```
- Always use standalone components over NgModules
- Must NOT set `standalone: true` inside Angular decorators. It's the default in Angular v20+.
- Use signals for state management
- Implement lazy loading for feature routes
- Do NOT use the `@HostBinding` and `@HostListener` decorators. Put host bindings inside the `host` object
- Use `NgOptimizedImage` for all static images.
  - `NgOptimizedImage` does not work for inline base64 images.

## Accessibility Requirements

- It MUST pass all AXE checks.
- It MUST follow all WCAG AA minimums, including focus management, color contrast, and ARIA attributes.
```

Click here to download the best-practices.md file.

Rules Files

Several editors, such as Firebase Studio have rules files useful for providing critical context to LLMs.

Environment/IDE	Rules File	Installation Instructions
Firebase Studio	airules.md	Configure `airules.md`
Copilot powered IDEs	copilot-instructions.md	Configure `.github/copilot-instructions.md`
Cursor	cursor.md	Configure `cursorrules.md`
JetBrains IDEs	guidelines.md	Configure `guidelines.md`
VS Code	.instructions.md	Configure `.instructions.md`
Windsurf	guidelines.md	Configure `guidelines.md`

Angular CLI MCP Server setup

The Angular CLI includes an experimental Model Context Protocol (MCP) server that allows AI assistants in your development environment to interact with the Angular CLI.

Learn how to set up the Angular CLI MCP Server

Providing Context with `llms.txt`

`llms.txt` is a proposed standard for websites designed to help LLMs better understand and process their content. The Angular team has developed two versions of this file to help LLMs and tools that use LLMs for code generation to create better modern Angular code.

- llms.txt - an index file providing links to key files and resources.
- llms-full.txt - a more robust compiled set of resources describing how Angular works and how to

Be sure to check out the overview page for more information on how to integrate AI into your Angular a

Web Codegen Scorer

The Angular team developed and open-sourced the Web Codegen Scorer, a tool to evaluate and score the quality of AI generated web code. You can use this tool to make evidence-based decisions relating to AI-generated code, such as fine-tuning prompts to improve the accuracy of LLM-generated code for Angular. These prompts can be included as system instructions for your AI tooling or as context with your prompt. You can also use this tool to compare the quality of code produced by different models and monitor quality over time as models and agents evolve.