Mermaid Diagrams from text

CONCEPTS

A diagram is plain text: the first line names the type (flowchart, sequenceDiagram, erDiagram), the rest is its own syntax.

Text .mmd or ```mermaid
↓ parse
Mermaid JS library
↓ render
SVG page, PNG, PDF

FLOWCHART

`flowchart TD / LR / BT / RL`
Set direction

`A[Text]`
Rectangle node

`A{Text}`
Decision diamond

`A ⟶ B`
Arrow

`A ⟶|text| B / A -- text ⟶ B`
Labelled link

`A ⟶ B & C`
One to many

`subgraph` *<id>* **`[`***<title>***`]`**
Group nodes, close with end

`classDef` *<c>* **`fill:#f96,stroke:#333`**
Define a style class

`class A,B` *<c>* **`/ A :::`** *<c>*
Apply a class

`style A fill:#f9f`
Style one node

`A["Text (with) symbols"]`
Quote text with special characters

`%%` *<comment>*
Comment line

SEQUENCE DIAGRAM

`participant A as` *<alias>*
Declare, order, rename

`A⟶»B: text`
Solid arrow message

`A-->>B: text`
Dotted arrow (reply)

SEQUENCE DIAGRAM (cont.)

`A⟶»+B: text`
Activate B on arrival (- to end)

`Note right of A: text`
Note beside a participant

`loop` *<label>* **`... end`**
Repeat

`alt` *<c>* **`... else ... end`**
Alternatives

`autonumber`
Number the messages

CLASS DIAGRAM

`classDiagram`
Start a class diagram

`class A { +type field / +method() }`
Class with members

`A ◁-- B`
Inheritance

`A ⟶ B / A -- B`
Association / link

STATE, ER, GANTT, PIE

`stateDiagram-v2`
Start a state diagram

`[*] ⟶ A : label`
Start state and transition

`A ⟶ [*]`
End state

`erDiagram`
Start an ER diagram

`A ||⊢ -o{ B : has`
One to zero-or-more

`A { string name PK }`
Entity attributes with keys

<task> **`:done, t1, 2026-01-01, 3d`**
Task: tags, id, start, length

CLI AND EMBEDDING

`npm i -g @mermaid-js/mermaid-cli`
Install mmdc

`mmdc -i` *<in.mmd>* **`-o`** *<out.svg>*
Render, format by extension