Mermaid Diagrams from text

START HERE

`flowchart TD`
Top-down flowchart

`mmdc -i in.mmd -o out.svg`
Render to file

`A[Text] ⟶ B(Text)`
Two nodes, one arrow

```` ```mermaid ````
Fence in Markdown

`A ⟶|yes| B`
Label a link

`sequenceDiagram`
Message sequence

`subgraph <name> ... end`
Group nodes

`%% comment`
Ignored line

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

CONCEPTS

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

FLOWCHART

`flowchart TD / LR / BT / RL`
Set direction
`graph` is an accepted alias; `TB` equals `TD`

`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

FLOWCHART (cont.)

`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

CLASS DIAGRAM (cont.)

`A ..> B`
Dependency

`A "1" ⟶ "*" B : has`
Cardinality and label

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

`section <name>`
Group tasks

STATE, ER, GANTT, PIE (cont.)

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

`<task> :after t1, 5d`
Start after another task

`pie showData`
Pie chart with values

`"Label" : 42`
Pie slice

CLI AND EMBEDDING

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

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

`mmdc -t dark -b transparent`
Theme and background

CLI AND EMBEDDING (cont.)

`mmdc -i in.md -o out.md`
Render fences in a Markdown file

`<pre class="mermaid">`
Embed in an HTML page

GitHub, GitLab and most Markdown tools render a ```mermaid fence without any install.

RECIPES

Request flow

```
sequenceDiagram
  participant U as User
  participant S as Server
  U⟶»+S: GET /items
  S--»⟶U: 200 JSON
```

Pipeline with a branch

```
flowchart LR
  A[Commit] ⟶ B{Tests pass?}
  B ⟶|yes| C[Deploy]
  B ⟶|no| D[Fix]
  D ⟶ A
```