D2

DECLARATIVE DIAGRAMMING (D2) CHEAT SHEET

Getting started

introduction

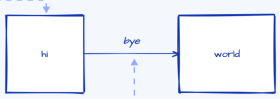
```
hello -> world
```

labels on nodes

```
hello: hi
```

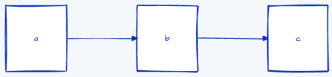
labels on connections

```
hello -> world: bye
```



chained connections

```
a -> b -> c
```



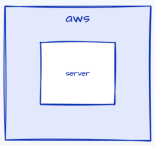
containers. there 2 ways of making containers

1st: add “.” anywhere
(quick and easy method)

```
aws.server
```

2nd: create a map
(more structured method)

```
aws: {
   server
}
```

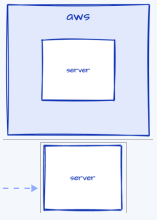


pro tip: make sure you reference the container to target a child

```
aws: {
   server
}
aws.server
server
```

points to this

This will create a new shape labeled “server” outside of “aws”

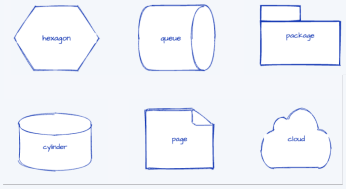


reserved keywords list

```
myNode: {shape: circle}
```

pro tip: some keywords are reserved, for example “label” and “shape.” You can choose from any of these shapes:

```
hexagon: {shape: hexagon}
queue: {shape: queue}
package: {shape: package}
cylinder: {shape: cylinder}
page: {shape: page}
cloud: {shape: cloud}
```



etc. choose any shape in the shapes panel

comments

```
# to do
```

short one line comments

Advanced

special shapes

classes

```
D2 AST Parser: {
  shape: class

  +prevRune: rune
  prevColumn: int

  +eatSpace(eatNewlines bool): (rune, error)
  unreadRune()

  \#scanKey(r rune): (k Key, _ error)
}
```

Classes have special syntax

- `+` for public visibility
- `-` for private
- `#` for protected

tables

```
board: {
    shape: sql_table

    id: int {constraint: primary_key}
    frame: int {constraint: foreign_key}
    diagram: int {constraint: foreign_key}
    board_objects: jsonb
    last_updated: timestamp with time zone
    last_thumbgen: timestamp with time zone
    dsl: text
  }
```

pro tip: You can define relationships between tables by:

```
diagram: {
  shape: sql_table
  id: int {constraint: primary_key}
}

board.diagram -> diagrams.id
```

markdown and code

- Usually, when you give shapes a value, they’re treated as labels, e.g. `a: b`
- However, you can also give it a Markdown-style block syntax, or define a language tag

markdown

```
a: |`
   # Header
   - This text
   - Will be Markdown
`|
```

D2 renders this as Markdown text with no border.

code

```
a: |`go
   func() {
      fmt.Printf("hello")
   }()
`|
```

D2 renders this as a code block

full language specifications

https://d2-lang.com

Updated for D2 v0.12+