

# Package ‘nomnoml’

July 22, 2025

**Type** Package

**Title** Sassy 'UML' Diagrams

**Version** 0.3.0

**Description** A tool for drawing sassy 'UML' (Unified Modeling Language) diagrams based on a simple syntax, see <<https://www.nomnoml.com>>. Supports styling, R Markdown and exporting diagrams in the PNG format. Note: you need a chromium based browser installed on your system.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 3.1.2)

**Imports** htmlwidgets, png, webshot2, lifecycle, rlang

**RoxygenNote** 7.2.3

**Suggests** V8, knitr, rmarkdown, testthat, shinytest, covr, spelling

**SystemRequirements** A chromium based browser, required by ``webshot2" <<https://rstudio.github.io/webshot2/>>

**URL** <https://rstudio.github.io/nomnoml/>

**BugReports** <https://github.com/rstudio/nomnoml/issues>

**RdMacros** lifecycle

**Language** en-US

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Andrie de Vries [aut, cre],  
Javier Luraschi [aut],  
Daniel Kallin [cph] (nomnoml.js library, <<https://nomnoml.com>>),  
RStudio [cph, fnd]

**Maintainer** Andrie de Vries <apdevries@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-11-20 10:10:02 UTC

## Contents

nomnoml	2
nomnoml-shiny	3
nomnoml_syntax	4
nomnoml_validate	6

<b>Index</b>	<b>8</b>
--------------	----------

---

nomnoml	<i>Create and render a nomnoml diagram.</i>
---------	---

---

### Description

#### [Experimental]

Renders a 'nomnoml' diagram as an 'htmlwidget' or saves it as a '.png' or '.svg' image.

### Usage

```
nomnoml(
  code = "[Hello]-[World!]",
  png = NULL,
  width = NULL,
  height = NULL,
  svg = FALSE,
  ...
)
```

### Arguments

code	The nomnoml diagram code.
png	Optional file name to export diagram as 'png'.
width	Optional width in pixels for the exported 'png'.
height	Optional height in pixels for the exported 'png'.
svg	Use 'svg' output instead of 'png'? Notice that rendering in 'svg' is not at a par with 'png' and renders incorrectly at times.
...	Additional parameters.

### Details

The 'nomnoml' syntax is simple and intuitive, a "Hello World" example can be rendered as an 'htmlwidget' as follows:

```
nomnoml::nomnoml("[Hello]-[World!]")
```

You can also render as a 'png' file with specific dimensions:

```
nomnoml::nomnoml("[Hello]-[World!]", png = "hello.png", 600, 100)
```

Still, complex diagrams can be defined by combining multiple association types, classifier types, directives and custom classifier styles.

You can also use of the nomnoml 'knitr' chunk to render inline diagrams in R Markdown documents.

## Syntax

For a summary of available nomnoml syntax, including association types, directives and customer classifier styles, see [nomnoml\\_syntax](#)

## See Also

[nomnomlOutput\(\)](#), [renderNomnoml\(\)](#), [nomnoml\\_validate\(\)](#), [nomnoml\\_syntax\(\)](#)

## Examples

```
# Render simple diagram:
nomnoml::nomnoml("[Hello]-[World!]")

# Render complex diagram:
nomnoml::nomnoml(
  #stroke: #a86128
  [<frame>Decorator pattern|
  [<abstract>Component| |+ operation()]
  [Client] depends --> [Component]
  [Decorator|- next: Component]
  [Decorator] decorates -- [ConcreteComponent]
  [Component] <:- [Decorator]
  [Component] <:- [ConcreteComponent]
  ]")
```

## Description

### [Experimental]

Output and render functions for using nomnoml within Shiny applications and interactive Rmd documents.

## Usage

```
nomnomlOutput(outputId, width = "100%", height = "400px")
```

```
renderNomnoml(expr, env = parent.frame(), quoted = FALSE)
```

**Arguments**

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a nomnoml
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

**See Also**

[nomnoml\(\)](#)

---

nomnoml_syntax	<i>Summary of nomnoml syntax.</i>
----------------	-----------------------------------

---

**Description**

Summary of nomnoml syntax.

**Association Types**

```

association -
association ->
association <->
dependency -->
dependency <-->
generalization -:>
generalization <:-
implementation --:>
implementation <:--
composition +-
composition +->
aggregation o-
aggregation o->
note --
hidden -/-
weightless edge _>
weightless dashed __

```

**Classifier Types**

```

[name]
[<abstract> name]
[<instance> name]
[<note> name]

```

```

[<reference> name]
[<package> name]
[<frame> name]
[<database> name]
[<start> name]
[<end> name]
[<state> name]
[<choice> name]
[<input> name]
[<sender> name]
[<receiver> name]
[<transceiver> name]
[<actor> name]
[<usecase> name]
[<label> name]
[<hidden> name]
[<table> table | a | 5 | b | 7]

```

### Directives

```

#arrowSize: 1
#bendSize: 0.3
#direction: down | right
#gutter: 5
#edgeMargin: 0
#edges: hard | rounded
#background: transparent
#fill: #eee8d5; #fdf6e3
#fillArrows: false
#font: Calibri
#fontSize: 12
#leading: 1.25
#lineWidth: 3
#padding: 8
#spacing: 40
#stroke: #33322E
#title: filename
#zoom: 1
#acyclicer: greedy
#ranker: network-simplex | tight-tree | longest-path

```

### Custom Classifier Styles

A directive that starts with `.` define a classifier style. The style is written as a space separated list of modifiers and key/value pairs.

```

#.box: fill=#8f8 dashed
#.blob: visual=ellipse
[<box> GreenBox]
[<blob> HideousBlob]

```

**Available key/value pairs are::**

fill=(any css color)  
stroke=(any css color)  
align=center  
align=left  
direction=right  
direction=down  
visual=actor  
visual=class  
visual=database  
visual=ellipse  
visual=end  
visual=frame  
visual=hidden  
visual=input  
visual=none  
visual=note  
visual=package  
visual=receiver  
visual=rhomb  
visual=roundrect  
visual=sender  
visual=start  
visual=transceiver

**Style title and text body:**

title=left,italic,bold  
body=center,italic,bold

**Text modifiers::**

bold  
underline  
italic  
dashed  
empty

**Description****[Experimental]**

Although the nomnoml widgets render very quickly in the IDE, it can take a few seconds to grab a static screenshot and create a png file. In these situations it can be helpful to validate if the nomnoml JS library can parse a diagram.

This function returns TRUE if a diagram can be parsed, and FALSE otherwise. If FALSE the function also throws a warning.

**Usage**

```
nomnoml_validate(diagram = "[test]")
```

**Arguments**

diagram            A nomnoml diagram to validate

**Value**

Either TRUE or FALSE

**See Also**

[nomnoml\(\)](#)

**Examples**

```
## Not run:  
if (requireNamespace("V8", quietly = TRUE)) nomnoml_validate("[hello] -> [world]")  
  
## End(Not run)
```

# Index

`'nomnoml-syntax'` (`nomnoml_syntax`), 4

`nomnoml`, 2

`nomnoml()`, 4, 7

`nomnoml-shiny`, 3

`nomnoml_syntax`, 3, 4

`nomnoml_syntax()`, 3

`nomnoml_validate`, 6

`nomnoml_validate()`, 3

`nomnomlOutput` (`nomnoml-shiny`), 3

`nomnomlOutput()`, 3

`renderNomnoml` (`nomnoml-shiny`), 3

`renderNomnoml()`, 3