

Package ‘shinipsum’

October 14, 2022

Title Lorem-Ipsum Helper Function for 'shiny' Prototyping

Version 0.1.0

Description Prototype your 'shiny' apps quickly with these Lorem-Ipsum helper functions. Generate random elements for 'shiny' outputs that can be used as placeholder in your application.

License MIT + file LICENSE

URL <https://github.com/Thinkr-open/shinipsum>

BugReports <https://github.com/Thinkr-open/shinipsum/issues>

Depends R (>= 2.10)

Imports attempt, datasets, DT, dygraphs, ggplot2 (>= 3.0.0), magrittr, plotly, stats, utils

Suggests covr, pkgdown, testthat

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Colin Fay [cre, aut, cph] (<<https://orcid.org/0000-0001-7343-1846>>),
Sebastien Rochette [aut] (<<https://orcid.org/0000-0002-1565-9313>>),
ThinkR [cph, fnd]

Maintainer Colin Fay <contact@colinfay.me>

Repository CRAN

Date/Publication 2020-04-30 18:40:03 UTC

R topics documented:

ipsum_examples	2
lorem	2
lorem_words	3
random_DT	3
random_dygraph	4

random_ggplot	4
random_ggplotly	5
random_image	5
random_print	6
random_table	6
random_text	7

Index	8
--------------	----------

ipsum_examples	<i>Get a shinipsum example</i>
----------------	--------------------------------

Description

Get a shinipsum example

Usage

```
ipsum_examples(which = NULL)
```

Arguments

`which` The example to run. If empty, all the available examples are listed.

Value

A path to the example.

Examples

```
ipsum_examples()
```

lorem	<i>Lorem text</i>
-------	-------------------

Description

A long lorem ipsum text

Usage

```
lorem
```

Format

An object of class character of length 1.

lorem_words	<i>Lorem test as vector</i>
-------------	-----------------------------

Description

A long vector of words

Usage

```
lorem_words
```

Format

An object of class character of length 13657.

random_DT	<i>A Random DT</i>
-----------	--------------------

Description

This function creates a random DT::datatable, and can be passed into renderDT & DTOutput.

Usage

```
random_DT(
  nrow,
  ncol,
  type = c("random", "numeric", "character", "numchar"),
  ...
)
```

Arguments

nrow	number of row of the output
ncol	number of cols of the output
type	type of the columns, can be "random", "numeric", "character", "numchar". Default is random.
...	arguments to be passed to DT::datatable

Value

a DT

Examples

```
if(interactive()){
  random_DT(10, 10)
  random_DT(10, 10, "numeric")
}
```

random_dygraph	<i>A Random Dygraph</i>
----------------	-------------------------

Description

This function returns a dygraph object, which can be passed to renderDygraph and dygraphOutput

Usage

```
random_dygraph(...)
```

Arguments

... args passed to dygraph

Value

a dygraph

Examples

```
if(interactive()){
  random_dygraph()
}
```

random_ggplot	<i>A Random ggplot</i>
---------------	------------------------

Description

This function returns a ggplot object, which can be passed to renderPlot and plotOutput

Usage

```
random_ggplot(
  type = c("random", "point", "bar", "boxplot", "col", "tile", "line", "bin2d",
    "contour", "density", "density_2d", "dotplot", "hex", "freqpoly", "histogram",
    "ribbon", "raster", "tile", "violin")
)
```

Arguments

`type` type of the geom. Can be any of "random", "point", "bar", "boxplot", "col", "tile", "line", "bin2d", "contour", "density", "density_2d", "dotplot", "hex", "freqpoly", "histogram", "ribbon", "raster", "tile", "violin" and defines the geom of the ggplot. Default is "random", and chooses a random geom for you.

Value

a ggplot

Examples

```
random_ggplot("point")
random_ggplot("histogram")
```

random_ggplotly	<i>A Random ggplotly</i>
-----------------	--------------------------

Description

This function returns a ggplotly object, which can be passed to `renderPlotly` and `plotlyOutput`

Usage

```
random_ggplotly(...)
```

Arguments

`...` arg to pass to `random_ggplot`.

Value

a ggplotly

random_image	<i>A Random Image</i>
--------------	-----------------------

Description

This function returns a random image that can be passed into `renderImage` and `plotOutput`.

Usage

```
random_image()
```

Value

an image

Examples

```
random_image()
```

random_print	<i>A Random print output</i>
--------------	------------------------------

Description

This function returns a random print output that can be passed to `renderPrint` and `verbatimTextOutput`.

Usage

```
random_print(type = c("character", "numeric", "integer", "model", "table"))
```

Arguments

`type` type of the output ("character", "numeric", "model", "table")

Value

a random print

Examples

```
random_print("character")
random_print("numeric")
```

random_table	<i>A Random Table</i>
--------------	-----------------------

Description

This function returns a table that can be passed to `renderTable` and `tableOutput`.

Usage

```
random_table(nrow, ncol, type = c("random", "numeric", "character", "numchar"))
```

Arguments

nrow	number of row of the output
ncol	number of cols of the output
type	type of the columns, can be "random", "numeric", "character", "numchar". Default is random.

Value

a table

Examples

```
random_table(10, 10)
random_table(10, 10, "numeric")
```

random_text	<i>A Random Lorem Ipsum</i>
-------------	-----------------------------

Description

A Random Lorem Ipsum

Usage

```
random_text(nchars = NULL, nwords = NULL)
```

Arguments

nchars	number of characters. One of the two params should be left NULL.
nwords	number of words to return. One of the two params should be left NULL.

Value

a text

Examples

```
random_text(nchars = 100)
random_text(nwords = 100)
```

Index

* datasets

lorem, [2](#)

lorem_words, [3](#)

ipsum_examples, [2](#)

lorem, [2](#)

lorem_words, [3](#)

random_DT, [3](#)

random_dygraph, [4](#)

random_ggplot, [4](#)

random_ggplotly, [5](#)

random_image, [5](#)

random_print, [6](#)

random_table, [6](#)

random_text, [7](#)