Package ‘upsetjs’

October 12, 2022

Type Package

Title 'HTMLWidget' Wrapper of 'UpSet.js' for Exploring Large Set Intersections

Description 'UpSet.js' is a re-implementation of 'UpSetR' to create interactive set visualizations for more than three sets. This is a 'htmlwidget' wrapper around the 'JavaScript' library 'UpSet.js'.

Version 1.11.1

Date 2022-07-13

Author Samuel Gratzl [aut, cre]

Maintainer Samuel Gratzl <sam@sgratzl.com>

URL https://github.com/upsetjs/upsetjs_r/

BugReports https://github.com/upsetjs/upsetjs_r/issues

Depends R (>= 3.2.0)

License AGPL-3 | file LICENSE

Encoding UTF-8

Imports htmlwidgets, magrittr

Suggests knitr, crosstalk, rmarkdown, formatR, tibble, testthat, styler, lintr, pkgdown

LazyData true

RoxygenNote 7.2.0

VignetteBuilder knitr

Language en-US

KeepSource true

NeedsCompilation no

Repository CRAN

Date/Publication 2022-07-13 06:00:08 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addCategoricalAttribute</td>
<td>3</td>
</tr>
<tr>
<td>addNumericAttribute</td>
<td>3</td>
</tr>
<tr>
<td>addQuery</td>
<td>4</td>
</tr>
<tr>
<td>asCombination</td>
<td>5</td>
</tr>
<tr>
<td>asSet</td>
<td>5</td>
</tr>
<tr>
<td>chartFontSizes</td>
<td>6</td>
</tr>
<tr>
<td>chartKarnaughMapLabels</td>
<td>7</td>
</tr>
<tr>
<td>chartKarnaughMapLayout</td>
<td>8</td>
</tr>
<tr>
<td>chartLabels</td>
<td>8</td>
</tr>
<tr>
<td>chartLayout</td>
<td>9</td>
</tr>
<tr>
<td>chartProps</td>
<td>11</td>
</tr>
<tr>
<td>chartStyleFlags</td>
<td>11</td>
</tr>
<tr>
<td>chartTheme</td>
<td>12</td>
</tr>
<tr>
<td>chartVennLabels</td>
<td>13</td>
</tr>
<tr>
<td>chartVennLayout</td>
<td>14</td>
</tr>
<tr>
<td>clearAttributes</td>
<td>14</td>
</tr>
<tr>
<td>clearQueries</td>
<td>15</td>
</tr>
<tr>
<td>extractSetsFromDataFrame</td>
<td>15</td>
</tr>
<tr>
<td>fromDataFrame</td>
<td>16</td>
</tr>
<tr>
<td>fromExpression</td>
<td>17</td>
</tr>
<tr>
<td>fromList</td>
<td>18</td>
</tr>
<tr>
<td>generateDistinctIntersections</td>
<td>19</td>
</tr>
<tr>
<td>generateIntersections</td>
<td>20</td>
</tr>
<tr>
<td>generateUnions</td>
<td>21</td>
</tr>
<tr>
<td>getCombinations</td>
<td>22</td>
</tr>
<tr>
<td>getElements</td>
<td>22</td>
</tr>
<tr>
<td>getSets</td>
<td>23</td>
</tr>
<tr>
<td>got</td>
<td>23</td>
</tr>
<tr>
<td>interactiveChart</td>
<td>24</td>
</tr>
<tr>
<td>queryLegend</td>
<td>24</td>
</tr>
<tr>
<td>renderUpsetjs</td>
<td>25</td>
</tr>
<tr>
<td>setAttributes</td>
<td>25</td>
</tr>
<tr>
<td>setCombinations</td>
<td>26</td>
</tr>
<tr>
<td>setElements</td>
<td>27</td>
</tr>
<tr>
<td>setQueries</td>
<td>27</td>
</tr>
<tr>
<td>setSelection</td>
<td>28</td>
</tr>
<tr>
<td>setSets</td>
<td>28</td>
</tr>
<tr>
<td>upsetjs</td>
<td>29</td>
</tr>
<tr>
<td>upsetjsDash</td>
<td>30</td>
</tr>
<tr>
<td>upsetjsEulerDiagram</td>
<td>30</td>
</tr>
<tr>
<td>upsetjsEulerDiagramProxy</td>
<td>31</td>
</tr>
<tr>
<td>upsetjsKarnaughMap</td>
<td>32</td>
</tr>
<tr>
<td>upsetjsKarnaughMapProxy</td>
<td>32</td>
</tr>
<tr>
<td>upsetjsOutput</td>
<td>33</td>
</tr>
<tr>
<td>upsetjsProxy</td>
<td>34</td>
</tr>
<tr>
<td>upsetjsSizingPolicy</td>
<td>34</td>
</tr>
</tbody>
</table>
addCategoricalAttribute

adds a new query to the plot

Description
adds a new query to the plot

Usage
addCategoricalAttribute(upsetjs, name, values, categories = NULL)

Arguments
upsetjs                  an object of class upsetjs or upsetjs_proxy
name                     name of the attribute
values                   the values as a factor
categories               optional categories otherwise the levels are used

Value
the object given as first argument

Examples
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  addCategoricalAttribute("attr", as.factor(sample(c("male", "female"), 3, replace = TRUE)))

addNumericAttribute      adds a new numeric attribute to the plot

Description
adds a new numeric attribute to the plot

Usage
addNumericAttribute(upsetjs, name, values, min_value = NULL, max_value = NULL)
addQuery

Arguments

- `upsetjs`: an object of class `upsetjs` or `upsetjs_proxy`
- `name`: name of the attribute
- `values`: the values as a numeric vector
- `min_value`: optional min domain value
- `max_value`: optional max domain value

Value

the object given as first argument

Examples

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  addNumericAttribute("attr", runif(3))
```

Description

adds a new query to the plot

Usage

```r
addQuery(upsetjs, name, color, elems = NULL, set = NULL)
```

Arguments

- `upsetjs`: an object of class `upsetjs` or `upsetjs_proxy`
- `name`: name of the query
- `color`: color of the query
- `elems`: the list of elems to highlight
- `set`: the set name, similar to the selection

Value

the object given as first argument

Examples

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  addQuery(name = "Q1", color = "red", set = "b")
```
asCombination

creates a new UpSet set combination structure

Description

creates a new UpSet set combination structure

Usage

asCombination(
  name,
  elems = c(),
  type = "intersection",
  sets = strsplit(name, "&"),
  cardinality = length(elems),
  color = NULL
)

Arguments

name name of the set combination
elems the elements of the set combination
type the set combination type (intersection, distinctIntersection, union, combination)
sets the sets this combination is part of
cardinality the cardinality of the set, default to length(elems)
color the color of the set

Value

the set object

Examples

asCombination("a", c(1, 2, 3))

asSet

creates a new UpSet set structure

Description

creates a new UpSet set structure

Usage

asSet(name, elems = c(), cardinality = length(elems), color = NULL)
Arguments

- **name**: name of the set
- **elems**: the elements of the set
- **cardinality**: the cardinality of the set, default to `length(elems)`
- **color**: the color of the set

Value

the set object

Examples

```r
asSet("a", c(1, 2, 3))
```

---

**chartFontSizes**  
*specify chart font sizes*

Description

specify chart font sizes

Usage

```r
chartFontSizes(
  upsetjs,
  font.family = NULL,
  chart.label = NULL,
  set.label = NULL,
  axis.tick = NULL,
  bar.label = NULL,
  legend = NULL,
  title = NULL,
  description = NULL,
  export.label = NULL,
  value.label = NULL
)
```

Arguments

- **upsetjs**: an object of class `upsetjs` or `upsetjs_proxy`
- **font.family**: specify the font family to render
- **chart.label**: font size of the chart label, default: 16px
- **set.label**: font size of the set label, default: 10px
- **axis.tick**: font size of the axis tick, default: 16px
- **bar.label**: font size of the bar label, default: 10px
chartKarnaughMapLabels

<table>
<thead>
<tr>
<th>argument</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>legend</td>
<td>font size of the legend label, default: 10px</td>
</tr>
<tr>
<td>title</td>
<td>font size of the chart title, default: 24px</td>
</tr>
<tr>
<td>description</td>
<td>font size of the chart description, default: 16px</td>
</tr>
<tr>
<td>export.label</td>
<td>font size of the export label, default: 10px</td>
</tr>
<tr>
<td>value.label</td>
<td>font size of the value label, (venn diagram only) default: 12px</td>
</tr>
</tbody>
</table>

**Value**

the object given as first argument

**Examples**

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartFontSizes(font.family = "serif")
```

```
chartKarnaughMapLabels

**specify chart labels**
```

**Description**

specify chart labels

**Usage**

```r
chartKarnaughMapLabels(upsetjs, title = NULL, description = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>argument</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>upsetjs</td>
<td>an object of class upsetjs_kamp or upsetjs_kmap_proxy</td>
</tr>
<tr>
<td>title</td>
<td>the chart title</td>
</tr>
<tr>
<td>description</td>
<td>the chart description</td>
</tr>
</tbody>
</table>

**Value**

the object given as first argument

**Examples**

```r
upsetjsKarnaughMap() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartKarnaughMapLabels(title = "Test")
```
chartKarnaughMapLayout

specify the chart karnaugh map layout

Description

specify the chart karnaugh map layout

Usage

chartKarnaughMapLayout(
  upsetjs,
  padding = NULL,
  bar.padding = NULL,
  numerical.scale = NULL
)

Arguments

upsetjs an object of class upsetjs_kmap or upsetjs_kmap_proxy
padding padding around the plot
bar.padding padding ratio (default 0.1) for the bar charts
numerical.scale numerical scale: linear (default) or log

Value

the object given as first argument

Examples

upsetjsKarnaughMap() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartKarnaughMapLayout(padding = 10)

chartLabels

specify chart labels

Description

specify chart labels
Usage

chartLabels(
  upsetjs,
  title = NULL,
  description = NULL,
  combination.name = NULL,
  combination.name.axis.offset = NULL,
  set.name = NULL,
  set.name.axis.offset = NULL,
  bar.label.offset = NULL
)

Arguments

  upsetjs      an object of class upsetjs or upsetjs_proxy
  title        the chart title
  description  the chart description
  combination.name
                 the label for the combination chart
  combination.name.axis.offset
                 the offset of the combination label from the axis in pixel
  set.name
                 the label for the set chart
  set.name.axis.offset
                 the offset of the set label from the axis in pixel
  bar.label.offset
                 the offset of the bar label from the bar in pixel

Value

  the object given as first argument

Examples

  upsetjs() %>%
    fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
    chartLabels(set.name = "Test")

chartLayout  specify the chart layout

Description

  specify the chart layout
Usage

chartLayout(
  upsetjs,
  height.ratios = NULL,
  width.ratios = NULL,
  padding = NULL,
  bar.padding = NULL,
  dot.padding = NULL,
  numerical.scale = NULL,
  band.scale = NULL,
  set.label.alignment = NULL,
  set.max.scale = NULL,
  combination.max.scale = NULL
)

Arguments

upsetjs an object of class upsetjs or upsetjs_proxy
height.ratios a vector of length 2 for the ratios between the combination and set plot, e.g. c(0.6, 0.4)
width.ratios a vector of length 3 for the ratios between set, label, and combination plot, e.g. c(0.3,0.2,0.5)
padding padding around the plot
bar.padding padding ratio (default 0.1) for the bar charts
dot.padding padding factor (default 0.7) for the dots
numerical.scale numerical scale: linear (default) or log
band.scale band scale: band (default)
set.label.alignment set label alignment: left, center (default), right
set.max.scale maximum value for the set scale
combination.max.scale maximum value for the combination scale

Value

the object given as first argument

Examples

upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartLayout(width.ratios = c(0.4, 0.2, 0.4))
chartProps

generic set chart props

Description

generic set chart props

Usage

chartProps(upsetjs, ...)

Arguments

upsetjs an object of class upsetjs or upsetjs_proxy
...
all upsetjs properties in R name notation

Value

the object given as first argument

Examples

upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartProps(theme = "dark")

chartStyleFlags

specify chart flags

Description

specify chart flags

Usage

chartStyleFlags(upsetjs, id = NULL, export.buttons = NULL, class.name = NULL)

Arguments

upsetjs an object of class upsetjs or upsetjs_proxy
id the optional HTML ID
export.buttons show export SVG and PNG buttons
class.name extra CSS class name to the root element
Value

the object given as first argument

Examples

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartStyleFlags(id = "test")
```

chartTheme

specify theming options

Description

specify theming options

Usage

```r
chartTheme(
  upsetjs,
  theme = NULL,
  selection.color = NULL,
  alternating.color = NULL,
  color = NULL,
  has.selection.color = NULL,
  text.color = NULL,
  hover.hint.color = NULL,
  not.member.color = NULL,
  value.text.color = NULL,
  stroke.color = NULL,
  has.selection.opacity = NULL,
  opacity = NULL,
  filled = NULL
)
```

Arguments

- **upsetjs**: an object of class `upsetjs` or `upsetjs_proxy`
- **theme**: theme to use `"dark"` or `"light"
- **selection.color**: selection color
- **alternating.color**: alternating background color
- **color**: main bar color
- **has.selection.color**: main color used when a selection is present
chartVennLabels

**text.color**  main text color
**hover.hint.color**  color of the hover hint
**not.member.color**  color of the dot if not a member
**value.text.color**  value text color (venn diagram only)
**stroke.color**  circle stroke color (venn diagram and karnaugh map only)
**has.selection.opacity**  main opacity used when a selection is present
**opacity**  main bar opacity
**filled**  enforce filled circles (venn diagram only)

**Value**

the object given as first argument

**Examples**

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartTheme(theme = "dark")
```

```r
chartVennLabels
  specify chart labels
```

**Description**

specify chart labels

**Usage**

`chartVennLabels(upsetjs, title = NULL, description = NULL)`

**Arguments**

**upsetjs**  an object of class `upsetjs_venn` or `upsetjs_venn_proxy`
**title**  the chart title
**description**  the chart description

**Value**

the object given as first argument

**Examples**

```r
upsetjsVennDiagram() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartVennLabels(title = "Test")
```
clearAttributes  

specifies the list of attributes for incremental updates

Usage

clearAttributes(upsetjs)

Arguments

upsetjs  
an object of class upsetjs or upsetjs_proxy

Value

the object given as first argument

Examples

upsetjsVennDiagram() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartVennLayout(padding = 10)

clearAttributes

specify the chart venn layout

Usage

chartVennLayout(upsetjs, padding = NULL)

Arguments

upsetjs  
an object of class upsetjs_venn or upsetjs_venn_proxy
padding  
padding around the plot

Value

the object given as first argument

Examples

upsetjsVennDiagram() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  chartVennLayout(padding = 10)
clearQueries

Examples

upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  clearAttributes()

clearQueries clears the list of queries for incremental updates

Description
clears the list of queries for incremental updates

Usage
clearQueries(upsetjs)

Arguments

upsetjs an object of class upsetjs or upsetjs_proxy

Value
the object given as first argument

Examples

upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  addQuery(name = "Q1", color = "red", set = "b") %>%
  clearQueries()

eextractSetsFromDataFrame

extract the sets from a data frame (rows = elems, columns = sets, cell = contained)

Description
extract the sets from a data frame (rows = elems, columns = sets, cell = contained)
Usage

extractSetsFromDataFrame(
  df,
  attributes = NULL,
  order.by = "cardinality",
  limit = NULL,
  colors = NULL,
  store.elems = TRUE
)

Arguments

df            the data.frame like structure
attributes    the optional column list or data frame
order.by      order intersections by cardinality or degree
limit         limit the ordered sets to the given limit
colors        the optional list with set name to color
store.elems   store the elements in the sets (default TRUE)

fromDataFrame extract the sets from a data frame (rows = elems, columns = sets, cell = contained)

Description

extract the sets from a data frame (rows = elems, columns = sets, cell = contained)

Usage

fromDataFrame(
  upsetjs,
  df,
  attributes = NULL,
  order.by = "cardinality",
  limit = NULL,
  shared = NULL,
  shared.mode = "click",
  colors = NULL,
  c_type = NULL,
  store.elems = TRUE
)
fromExpression

Arguments

upsetjs     an object of class upsetjs or upsetjs_proxy
df          the data.frame like structure
attributes  the optional column list or data frame
order.by    order intersections by cardinality or degree
limit       limit the ordered sets to the given limit
shared      a crosstalk shared data frame
shared.mode whether on 'hover' or 'click' (default) is synced
colors      the optional list with set name to color
c_type      the combination type to use
store.elems whether to store the set elements within the structures (set to false for big data frames)

Value

the object given as first argument

Examples

df <- as.data.frame(list(a = c(1, 1, 1), b = c(0, 1, 1)), row.names = c("a", "b", "c"))
upsetjs() %>% fromDataFrame(df)

df <- upsetjs()

fromExpression

generates the sets from a lists object that contained the cardinalities of both sets and combinations (&)

Description

generates the sets from a lists object that contained the cardinalities of both sets and combinations (&)

Usage

fromExpression(
  upsetjs,     
  value,       
  symbol = "&",  
  order.by = "cardinality",  
  colors = NULL,  
  type = "intersection"
)
fromList

**Arguments**

- `upsetjs` an object of class `upsetjs` or `upsetjs_proxy`
- `value` the expression list input
- `symbol` the symbol how to split list names to get the sets
- `order.by` order intersections by cardinality or name
- `colors` the optional list with set name to color
- `type` the type of intersections this data represents (intersection, union, distinctIntersection)

**Value**

the object given as first argument

**Examples**

```r
generate() %>% fromExpression(list(a = 3, b = 2, `a&b` = 2))
```

**Description**

generates the sets from a lists object

**Usage**

```r
fromList(
  upsetjs, 
  value, 
  order.by = "cardinality", 
  limit = NULL, 
  shared = NULL, 
  shared.mode = "click", 
  colors = NULL, 
  c_type = NULL
)
```

**Arguments**

- `upsetjs` an object of class `upsetjs` or `upsetjs_proxy`
- `value` the list input value
- `order.by` order intersections by cardinality or name
- `limit` limit the ordered sets to the given limit
- `shared` a crosstalk shared data frame
- `shared.mode` whether on 'hover' or 'click' (default) is synced
- `colors` the optional list with set name to color
- `c_type` the combination type to use or "none" for disabling initial generation
**generateDistinctIntersections**

**Value**

the object given as first argument

**Examples**

```r
upsetjs() %>% fromList(list(a = c(1, 2, 3), b = c(2, 3)))
```

**generateDistinctIntersections**

configure the generation of the distinct intersections

**Description**

configure the generation of the distinct intersections

**Usage**

```r
generateDistinctIntersections(
  upsetjs, 
  min = 0, 
  max = NULL, 
  empty = FALSE, 
  order.by = "cardinality", 
  limit = NULL, 
  colors = NULL
)
```

**Arguments**

- **upsetjs**: an object of class upsetjs or upsetjs_proxy
- **min**: minimum number of sets in an intersection
- **max**: maximum number of sets in an intersection
- **empty**: whether to include empty intersections or not
- **order.by**: order intersections by cardinality, degree, name or a combination of it
- **limit**: limit the number of intersections to the top N
- **colors**: the optional list with set name to color

**Value**

the object given as first argument

**Examples**

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
generateDistinctIntersections(min = 2)
```
generateIntersections configure the generation of the intersections

Description

configure the generation of the intersections

Usage

generateIntersections(
  upsetjs,
  min = 0,
  max = NULL,
  empty = FALSE,
  order.by = "cardinality",
  limit = NULL,
  colors = NULL
)

Arguments

upsetjs an object of class upsetjs or upsetjs_proxy
min minimum number of sets in an intersection
max maximum number of sets in an intersection
empty whether to include empty intersections or not
order.by order intersections by cardinality, degree, name or a combination of it
limit limit the number of intersections to the top N
colors the optional list with set name to color

Value

the object given as first argument

Examples

upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  generateIntersections(min = 2)
generateUnions

configure the generation of the unions

Description

configure the generation of the unions

Usage

generateUnions(
  upsetjs,
  min = 0,
  max = NULL,
  empty = FALSE,
  order.by = "cardinality",
  limit = NULL,
  colors = NULL
)

Arguments

upsetjs an object of class upsetjs or upsetjs_proxy
min minimum number of sets in an union
max maximum number of sets in an union
empty whether to include empty intersections or not
order.by order intersections by cardinality, degree, name or a combination of it
limit limit the number of intersections to the top N
colors the optional list with set name to color

Value

the object given as first argument

Examples

upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
generateUnions()
getCombinations  

**Description**

extract the vector of combinations

**Usage**

```r
getCombinations(upsetjs)
```

**Arguments**

- `upsetjs` 
an object of class upsetjs

**Value**

vector of sets

**Examples**

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  getCombinations()
```

ggetElements

**Description**

extract the vector of elements

**Usage**

```r
ggetElements(upsetjs)
```

**Arguments**

- `upsetjs` 
an object of class upsetjs or upsetjs_proxy

**Value**

vector of elements

**Examples**

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  getElements()
```
getSets

**Description**

extract the vector of sets

**Usage**

getsSets(upsetjs)

**Arguments**

upsetjs an object of class upsetjs

**Value**

vector of sets

**Examples**

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
getsSets()
```

---

got

Games of Thrones Character dataset for UpSet.js

**Description**

A dataset containing set information about Game of Thrones characters

**Usage**

got

**Format**

A data frame with 22 rows and 6 variables/sets:

- **Lannister** character part of the Lannister house
- **Stark** character part of the Stark house
- **female** character is female
- **male** character is male
- **royal** character is royal
- **was.killed** character was killed
interactiveChart

*make it an interactive chart*

**Description**

make it an interactive chart

**Usage**

```r
interactiveChart(upsetjs, value = TRUE, events_nonce = FALSE)
```

**Arguments**

- **upsetjs**: an object of class `upsetjs` or `upsetjs_proxy`
- **value**: whether to enable or disable or set the mode: hover, click, contextMenu
- **events_nonce**: whether to enable send a unique once (event date) for each event to prevent deduplication

**Value**

the object given as first argument

**Examples**

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  interactiveChart()
```

queryLegend

*renders a legend for the queries*

**Description**

renders a legend for the queries

**Usage**

```r
queryLegend(upsetjs, value = TRUE)
```

**Arguments**

- **upsetjs**: an object of class `upsetjs` or `upsetjs_proxy`
- **value**: whether to enable or disable
renderUpsetjs

Value

the object given as first argument

Examples

upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  addQuery(name = "Q1", color = "red", set = "b") %>%
  queryLegend(FALSE)

Description

Shiny render bindings for upsetjs

Usage

renderUpsetjs(expr, env = parent.frame(), quoted = FALSE)

Arguments

expr An expression that generates an upset
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.

Value

The output of shinyRenderWidget function

setAttributes

set the attributes

Description

set the attributes

Usage

setAttributes(upsetjs, attrs = list())
Arguments

upsetjs an object of class upsetjs or upsetjs_proxy
attrs the attributes to set

Value

the object given as first argument

Examples

upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  setAttributes(list(
    attr = runif(3),
    cat = as.factor(sample(c("male", "female"), 3, replace = TRUE))
  ))

setCombinations

describe set the vector of combinations

Description

set the vector of combinations

Usage

setCombinations(upsetjs, value)

Arguments

upsetjs an object of class upsetjs
value the vector of combinations

Value

the object given as first argument

Examples

upsetjs() %>%
  setCombinations(list(asCombination("a", c(1, 2, 3)))) %>%
  getCombinations()
**setElements**  
*set the vector of elements*

**Description**

set the vector of elements

**Usage**

```r
setElements(upsetjs, value)
```

**Arguments**

- **upsetjs**: an object of class `upsetjs`
- **value**: the vector of elements

**Value**

the object given as first argument

**Examples**

```r
upsetjs() %>%
  setElements(c(1, 2, 3, 4, 5)) %>%
  getElements()
```

**setQueries**  
*set the queries*

**Description**

set the queries

**Usage**

```r
setQueries(upsetjs, queries = list())
```

**Arguments**

- **upsetjs**: an object of class `upsetjs` or `upsetjs_proxy`
- **queries**: the queries to set

**Value**

the object given as first argument
Examples

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  setQueries(list(list(name = "Q1", color = "red", set = "b")))
```

---

**setSelection**  
*sets the selection of the chart*

**Description**

sets the selection of the chart

**Usage**

```
setSelection(upsetjs, name = NULL)
```

**Arguments**

- **upsetjs**: an object of class `upsetjs` or `upsetjs_proxy`
- **name**: the name of the set to select or a list with name and type

**Value**

the object given as first argument

**Examples**

```r
upsetjs() %>%
  fromList(list(a = c(1, 2, 3), b = c(2, 3))) %>%
  setSelection("b")
```

---

**setSets**  
*set the vector of sets*

**Description**

set the vector of sets

**Usage**

```
setSets(upsetjs, value)
```

**Arguments**

- **upsetjs**: an object of class `upsetjs`
- **value**: the vector of sets
Value

the object given as first argument

Examples

```r
upsetjs() %>%
  setCombinations(list(asSet("a", c(1, 2, 3)))) %>%
  getSets()
```

Description

upsetjs a htmlwidget wrapper around UpSet.js ([https://upset.js.org/](https://upset.js.org/))

Usage

```r
upsetjs(
  width = "100%",
  height = NULL,
  elementId = NULL,
  sizingPolicy = upsetjsSizingPolicy()
)
```

Arguments

- `width` width of the element
- `height` height of the element
- `elementId` unique element id
- `sizingPolicy` htmlwidgets sizing policy object. Defaults to `upsetjsSizingPolicy()`

Value

An object of class `upsetjs` and `htmlwidget`

Examples

```r
upsetjs() %>% fromList(list(a = c(1, 2, 3), b = c(2, 3)))
```
upsetjsDash  
create a new upsetjs dash adapter

Description
create a new upsetjs dash adapter

Usage
upsetjsDash(children = NULL, id = NULL, width = NULL, height = NULL)

Arguments

children  
dash children

id  
dash id

width  
upsetjs width

height  
upsetjs height

Value
the set object

Examples

upsetjsDash("u") %>% fromList(list(a = c(1, 2, 3), b = c(2, 3)))

upsetjsEulerDiagram  
upsetjs - factory for UpSet.js Euler Diagram HTMLWidget

Description
upsetjs - factory for UpSet.js Euler Diagram HTMLWidget

Usage
upsetjsEulerDiagram(
  width = "100%",
  height = NULL,
  elementId = NULL,
  sizingPolicy = upsetjsSizingPolicy()
)


upsetjsEulerDiagramProxy

Arguments

width      width of the element
height     height of the element
elementId  unique element id
sizingPolicy htmlwidgets sizing policy object. Defaults to upsetjsSizingPolicy()

Value

An object of class upsetjs_venn and htmlwidget

Examples

upsetjs() %>% fromList(list(a = c(1, 2, 3), b = c(2, 3)))

Description

reactive helper to update an upsetjs euler diagram in place

Usage

upsetjsEulerDiagramProxy(outputId, session)

Arguments

outputId the id of the upsetjs widget
session   current shiny session

Value

an object of class upsetjs_proxy

Examples

## Not run:
upsetjsEulerDiagramProxy("upsetjs1", session) %>% setSelection("a")
## End(Not run)
Description

upsets - factory for UpSet.js Karnaugh Map HTMLWidget

Usage

upsetjsKarnaughMap(
  width = "100%",
  height = NULL,
  elementId = NULL,
  sizingPolicy = upsetjsSizingPolicy()
)

Arguments

width width of the element
height height of the element
elementId unique element id
sizingPolicy htmlwidgets sizing policy object. Defaults to upsetjsSizingPolicy()

Value

An object of class upsetjs_venn and htmlwidget

Examples

upsetjsKarnaughMap() %>% fromList(list(a = c(1, 2, 3), b = c(2, 3)))

Description

reactive helper to update an upsetjs karnaugh map diagram in place

Usage

upsetjsKarnaughMapProxy(outputId, session)
**upsetjsOutput**  

**Arguments**  

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputId</td>
<td>the id of the upsetjs widget</td>
</tr>
<tr>
<td>session</td>
<td>current shiny session</td>
</tr>
</tbody>
</table>

**Value**  

An object of class `upsetjs_proxy`

**Examples**

```
## Not run:
upsetjsKarnaughMapProxy("upsetjs1", session) %>% setSelection("a")

## End(Not run)
```

---

**Description**

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

**Usage**

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

**Arguments**  

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputId</td>
<td>output variable to read from</td>
</tr>
<tr>
<td>width</td>
<td>Must be a valid CSS unit (like '100%', '800px', 'auto') or a number, which will be coerced to a string and have 'px' appended.</td>
</tr>
<tr>
<td>height</td>
<td>see width</td>
</tr>
</tbody>
</table>

**Value**

An output or render function that enables the use of the widget within Shiny applications.
**upsetjsProxy**  
*reactive helper to update an upsetjs inplace*

**Description**
reactive helper to update an upsetjs inplace

**Usage**

```r
upsetjsProxy(outputId, session)
```

**Arguments**
- **outputId**: the id of the upsetjs widget
- **session**: current shiny session

**Value**
an object of class `upsetjs_proxy`

**Examples**

```r
## Not run:
upsetjsProxy("upsetjs1", session) %>% setSelection("a")
## End(Not run)
```

---

**upsetjsSizingPolicy**  
*upsetjs sizing policy*

**Description**
upsetjs sizing policy

**Usage**

```r
upsetjsSizingPolicy(
  defaultWidth = "100%",
  defaultHeight = 400,
  padding = 0,
  browser.fill = TRUE,
  ...
)
```
Arguments

- `defaultWidth` defaults to "100%" of the available width
- `defaultHeight` defaults to 400px tall
- `padding` defaults to 0px
- `browser.fill` defaults to TRUE
- ... all other arguments supplied to htmlwidgets::sizingPolicy

Value

An htmlwidgets::sizingPolicy object

Examples

```r
upsetjs(sizingPolicy = upsetjsSizingPolicy(padding = 20)) %>%
fromList(list(a = c(1, 2, 3), b = c(2, 3)))
```

Description

upsetjs - factory for UpSet.js Venn Diagram HTMLWidget

Usage

```r
upsetjsVennDiagram(
    width = "100%",
    height = NULL,
    elementId = NULL,
    sizingPolicy = upsetjsSizingPolicy()
)
```

Arguments

- `width` width of the element
- `height` height of the element
- `elementId` unique element id
- `sizingPolicy` htmlwidgets sizing policy object. Defaults to upsetjsSizingPolicy()

Value

An object of class upsetjs_venn and htmlwidget

Examples

```r
upsetjs() %>% fromList(list(a = c(1, 2, 3), b = c(2, 3)))
```
upsetjsVennDiagramProxy

reactive helper to update an upsetjs venn diagram in place

Description

reactive helper to update an upsetjs venn diagram in place

Usage

upsetjsVennDiagramProxy(outputId, session)

Arguments

outputId the id of the upsetjs widget
session current shiny session

Value

an object of class upsetjs_proxy

Examples

## Not run:
upsetjsVennDiagramProxy("upsetjs1", session) %>% setSelection("a")

## End(Not run)
## Index

<table>
<thead>
<tr>
<th>* datasets</th>
<th>got, 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>addCategoricalAttribute, 3</td>
<td></td>
</tr>
<tr>
<td>addNumericAttribute, 3</td>
<td></td>
</tr>
<tr>
<td>addQuery, 4</td>
<td></td>
</tr>
<tr>
<td>asCombination, 5</td>
<td></td>
</tr>
<tr>
<td>asSet, 5</td>
<td></td>
</tr>
<tr>
<td>chartFontSizes, 6</td>
<td></td>
</tr>
<tr>
<td>chartKarnaughMapLabels, 7</td>
<td></td>
</tr>
<tr>
<td>chartKarnaughMapLayout, 8</td>
<td></td>
</tr>
<tr>
<td>chartLabels, 8</td>
<td></td>
</tr>
<tr>
<td>chartLayout, 9</td>
<td></td>
</tr>
<tr>
<td>chartProps, 11</td>
<td></td>
</tr>
<tr>
<td>chartStyleFlags, 11</td>
<td></td>
</tr>
<tr>
<td>chartTheme, 12</td>
<td></td>
</tr>
<tr>
<td>chartVennLabels, 13</td>
<td></td>
</tr>
<tr>
<td>chartVennLayout, 14</td>
<td></td>
</tr>
<tr>
<td>clearAttributes, 14</td>
<td></td>
</tr>
<tr>
<td>clearQueries, 15</td>
<td></td>
</tr>
<tr>
<td>extractSetsFromDataFrame, 15</td>
<td></td>
</tr>
<tr>
<td>fromDataFrame, 16</td>
<td></td>
</tr>
<tr>
<td>fromExpression, 17</td>
<td></td>
</tr>
<tr>
<td>fromList, 18</td>
<td></td>
</tr>
<tr>
<td>generateDistinctIntersections, 19</td>
<td></td>
</tr>
<tr>
<td>generateIntersections, 20</td>
<td></td>
</tr>
<tr>
<td>generateUnions, 21</td>
<td></td>
</tr>
<tr>
<td>getCombinations, 22</td>
<td></td>
</tr>
<tr>
<td>getElements, 22</td>
<td></td>
</tr>
<tr>
<td>getSets, 23</td>
<td></td>
</tr>
<tr>
<td>got, 23</td>
<td></td>
</tr>
<tr>
<td>interactiveChart, 24</td>
<td></td>
</tr>
<tr>
<td>queryLegend, 24</td>
<td></td>
</tr>
<tr>
<td>renderUpsetjs, 25</td>
<td></td>
</tr>
</tbody>
</table>

| setAttributes, 25           |           |
| setCombinations, 26         |           |
| setElements, 27             |           |
| setQueries, 27              |           |
| setSelection, 28            |           |
| setSets, 28                 |           |
| sizingPolicy, 35            |           |
| upsetjs, 29                 |           |
| upsetjsDash, 30              |           |
| upsetjsEulerDiagram, 30     |           |
| upsetjsEulerDiagramProxy, 31|           |
| upsetjsKarnaughMap, 32      |           |
| upsetjsKarnaughMapProxy, 32 |           |
| upsetjsOutput, 33           |           |
| upsetjsProxy, 34            |           |
| upsetjsSizingPolicy, 29, 31, 32, 34, 35 | |
| upsetjsVennDiagram, 35      |           |
| upsetjsVennDiagramProxy, 36 |           |