

# Package: myIO (via r-universe)

June 12, 2026

**Type** Package

**Title** Interactive Data Visualizations Using 'd3.js'

**Version** 1.2.0

**Description** Create interactive 'd3.js' visualizations from R with built-in statistical transforms. Computes confidence intervals, regression fits, LOESS smoothing, moving averages, error bars, and uncertainty visualizations (quantile dot plots and fan charts) in R and renders them as composable chart layers via 'htmlwidgets'. Supports 36 chart types including boxplots, violin plots, Q-Q diagnostic plots, calendar heatmaps, survival curves, and group comparisons with pairwise significance testing. Also provides a machine-readable chart specification schema with validators so that large language model agents can author and verify charts. Works in 'RStudio', 'Shiny', and 'R Markdown'.

**License** MIT + file LICENSE

**URL** <https://mortonanalytics.github.io/myIO/>,  
<https://github.com/mortonanalytics/myIO>

**BugReports** <https://github.com/mortonanalytics/myIO/issues>

**Depends** R (>= 4.1.0)

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.2

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown, pkgdown, shiny, crosstalk (>= 1.2.0), dplyr, DT, reactable, htmltools, arrow, base64enc, cli, curl, DBI, duckdb, openssl, later, mockery, withr

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**Imports** htmlwidgets, jsonlite, stats

**Config/pak/sysreqs** cmake make libuv1-dev  
**Repository** <https://mortonanalytics.r-universe.dev>  
**Date/Publication** 2026-06-11 03:18:33 UTC  
**RemoteUrl** <https://github.com/mortonanalytics/myio>  
**RemoteRef** HEAD  
**RemoteSha** ac3d9a895664e00327bd6f53da4dd8667f078a61

## Contents

addIoLayer . . . . .	3
clear_duckdb_wasm_cache . . . . .	4
defineCategoricalAxis . . . . .	4
dragPoints . . . . .	5
duckdb_wasm_status . . . . .	6
flipAxis . . . . .	6
install_duckdb_wasm . . . . .	7
linkCharts . . . . .	8
myIO . . . . .	9
myIO-shiny . . . . .	10
myio_chart_schema . . . . .	11
myio_function_signature . . . . .	11
myIO_last_error . . . . .	12
myio_list_chart_types . . . . .	12
myio_list_functions . . . . .	13
myio_validate_call . . . . .	13
myio_validate_spec . . . . .	14
setAnnotation . . . . .	14
setAxisFormat . . . . .	15
setAxisLimits . . . . .	16
setBigData . . . . .	17
setBrush . . . . .	18
setColorScheme . . . . .	19
setExportOptions . . . . .	19
setFacet . . . . .	20
setLayerOpacity . . . . .	21
setLinked . . . . .	22
setLinkedCursor . . . . .	23
setMargin . . . . .	24
setReferenceLines . . . . .	25
setSlider . . . . .	25
setTheme . . . . .	26
setTitle . . . . .	27
setToggle . . . . .	28
setToolTipOptions . . . . .	29
setTransitionSpeed . . . . .	29
suppressAxis . . . . .	30

<i>addIoLayer</i>	3
<i>suppressLegend</i> . . . . .	31

**Index** **32**

---

<i>addIoLayer</i>	<i>Add a Layer to a myIO Chart</i>
-------------------	------------------------------------

---

**Description**

Adds individual layer to a myIO widget

**Usage**

```
addIoLayer(
  myIO,
  type,
  color = NULL,
  label,
  data = NULL,
  mapping,
  transform = "identity",
  options = list(barSize = "large", tooltipOptions = list(suppressY = FALSE))
)
```

**Arguments**

<i>myIO</i>	an htmlwidget object created by the <i>myIO()</i> function
<i>type</i>	chart type
<i>color</i>	optional CSS color string or vector for grouped layers
<i>label</i>	unique layer label
<i>data</i>	data frame backing the layer
<i>mapping</i>	named aesthetic mapping list
<i>transform</i>	transform name applied before serialization
<i>options</i>	layer options passed through to the widget config

**Value**

A modified *myIO* htmlwidget object with the new layer appended to the configuration.

**Examples**

```
myIO(data = mtcars) |>
  addIoLayer(
    type = "point", label = "points",
    mapping = list(x_var = "wt", y_var = "mpg")
  )
```

---

```
clear_duckdb_wasm_cache
```

*Remove DuckDB-WASM cache entries*

---

### Description

Remove DuckDB-WASM cache entries

### Usage

```
clear_duckdb_wasm_cache(version = NULL)
```

### Arguments

version	Character scalar naming a specific version to remove. If NULL, removes all cached versions.
---------	---------------------------------------------------------------------------------------------

### Value

Number of removed entries, invisibly.

### Examples

```
## Not run:
# Removes cached DuckDB-WASM binaries from the user cache.
clear_duckdb_wasm_cache()

## End(Not run)
```

---

```
defineCategoricalAxis Define Categorical Axis
```

---

### Description

Function to define the x variable as categorical

### Usage

```
defineCategoricalAxis(myIO, xAxis = TRUE, yAxis = FALSE)
```

### Arguments

myIO	an htmlwidget object created by the myIO() function
xAxis	a logical argument (TRUE) for defining the x axis as categorical
yAxis	a logical argument (TRUE) for defining the y axis as categorical

**Value**

A modified myIO htmlwidget object with categorical axis configured.

**Examples**

```
# Define x axis as categorical
myIO() |> defineCategoricalAxis(xAxis = TRUE)

# Define both axes as categorical
myIO() |> defineCategoricalAxis(xAxis = TRUE, yAxis = TRUE)
```

---

dragPoints	<i>Enable Draggable Points</i>
------------	--------------------------------

---

**Description**

Function to make points draggable

**Usage**

```
dragPoints(myIO, dragPoints = TRUE)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
dragPoints	a logical argument (TRUE) for creating draggable points

**Value**

A modified myIO htmlwidget object with drag interaction enabled.

**Examples**

```
# Enable draggable points
myIO() |>
  addIoLayer(
    type = "point", color = "red", label = "pts",
    data = mtcars, mapping = list(x_var = "wt", y_var = "mpg")
  ) |>
  dragPoints()
```

---

duckdb_wasm_status	<i>DuckDB-WASM cache status</i>
--------------------	---------------------------------

---

**Description**

DuckDB-WASM cache status

**Usage**

```
duckdb_wasm_status()
```

**Value**

A list with class 'myIO\_duckdb\_wasm\_status' and fields 'installed' (logical), 'version' (chr or NA), 'cache\_dir' (chr), 'size\_bytes' (numeric).

**Examples**

```
duckdb_wasm_status()
```

---

flipAxis	<i>Flip Chart Axes</i>
----------	------------------------

---

**Description**

Function to flip the x and y axes

**Usage**

```
flipAxis(myIO, flipAxis = TRUE)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
flipAxis	a logical argument (TRUE) for flipping the x and y axes

**Value**

A modified myIO htmlwidget object with axes flipped.

**Examples**

```
# Flip the axes for a horizontal bar chart
myIO() |>
  addIoLayer(
    type = "bar", color = "steelblue", label = "bars",
    data = mtcars, mapping = list(x_var = "wt", y_var = "mpg")
  ) |>
  flipAxis()
```

---

install\_duckdb\_wasm    *Install the DuckDB-WASM binary for large-dataset virtualization*

---

**Description**

Downloads and verifies the DuckDB-WASM runtime into a user-local cache so the in-browser big-data engine is available for subsequent widgets. This follows the keras3 / torch / reticulate pattern: the binary is NOT bundled with the R package and is not required for small-data use.

**Usage**

```
install_duckdb_wasm(
  version = NULL,
  from = NULL,
  force = FALSE,
  quiet = !interactive()
)
```

**Arguments**

version	Character scalar naming the version to install. Defaults to the latest row in ‘inst/duckdb-wasm-manifest.csv’.
from	Optional local directory containing pre-downloaded binaries (‘duckdb-mvp.wasm’ and ‘duckdb-browser-mvp.worker.js’). Airgap path.
force	Logical. If TRUE, overwrite existing cached binaries.
quiet	Logical. Suppress progress output. Defaults to ‘!interactive()’.

**Value**

Invisibly returns the cache path where the binary was installed.

**Examples**

```
## Not run:
# Downloads ~22 MB from the upstream mirror into the user cache.
install_duckdb_wasm()

## End(Not run)
```

---

`linkCharts`*Link Charts for Cross-Selection*

---

### Description

A convenience wrapper that sets matching link config on multiple widgets so that brush selections propagate between them. Unlike `setLinked`, this does not require Crosstalk — it uses a shared group identifier and key column to coordinate selections across charts rendered in the same page.

### Usage

```
linkCharts(..., on, group = NULL, cursor = FALSE, cursorAxis = "x")
```

### Arguments

<code>...</code>	myIO widget objects to link.
<code>on</code>	Character. Column name to match rows across charts.
<code>group</code>	Character. Group identifier. Default auto-generated.
<code>cursor</code>	Logical. When TRUE, a hover in any linked chart draws a synchronized crosshair (and optional tooltip) on every sibling chart in the same group. Default FALSE.
<code>cursorAxis</code>	Character. Which axis to sync: "x" (default), "y", or "xy". Only "x" is active in v1.2; other values are accepted but not yet rendered.

### Value

A list of modified myIO widgets with matching link config.

### Examples

```
w1 <- myIO() |>
  addIoLayer(type = "point", label = "scatter",
    data = mtcars, mapping = list(x_var = "wt", y_var = "mpg"))
w2 <- myIO() |>
  addIoLayer(type = "bar", label = "bars",
    data = mtcars, mapping = list(x_var = "cyl", y_var = "mpg"))
linked <- linkCharts(w1, w2, on = "cyl")
```

## Description

Create an interactive D3.js chart widget

## Usage

```
myIO(  
  data = NULL,  
  width = "100%",  
  height = "400px",  
  elementId = NULL,  
  title = NULL,  
  sparkline = FALSE,  
  engine = "auto",  
  webgl_threshold = 50000L,  
  unify_data_path = FALSE  
)
```

## Arguments

data	an optional point of entry for the data frame or vector
width	a string of either pixel width or a percentage width
height	a string of pixel height
elementId	a unique id for the htmlwidget object
title	Optional chart title rendered inside the widget SVG.
sparkline	Logical. If TRUE, renders a compact sparkline suitable for embedding in table cells. Strips axes, legend, and interactions. Only "line", "bar", and "area" layer types are supported. Default FALSE.
engine	one of "auto" (default), "server", "wasm", or "svg". Only consulted when big-data features are attached via <code>setBigData()</code> . With no big-data attachment, charts render identically to the small-data SVG path regardless of this argument. See vignette("large-data-linking").
webgl_threshold	Positive integer row-count threshold for the big-data WebGL render path. Use <code>Inf</code> to disable WebGL. With <code>unify_data_path = TRUE</code> , disabled WebGL uses the SVG coordinator path; otherwise the existing inline SVG path is preserved. Default 50000.
unify_data_path	Logical. If TRUE, coordinator results also replace SVG layer data below <code>webgl_threshold</code> . Default FALSE preserves the inline small-data SVG render path.

**Value**

An htmlwidget object of class myIO.

**Examples**

```
myIO(data = mtcars) |>
  setMargin(top = 40, bottom = 80, left = 60, right = 10)
```

---

 myIO-shiny

*Shiny Bindings for myIO*


---

**Description**

Shiny Bindings for myIO

**Usage**

```
myIOOutput(outputId, width = "100%", height = "400px")
renderMyIO(expr, env = parent.frame(), quoted = FALSE)
```

**Arguments**

outputId	output variable to read from
width, height	Must be a valid CSS unit or a number.
expr	An expression that generates a myIO
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression?

**Value**

myIOOutput returns a Shiny UI element for placement in a UI definition. renderMyIO returns a Shiny render function for use in a server definition.

**Examples**

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(myIOOutput("chart"))
  server <- function(input, output) {
    output$chart <- renderMyIO({
      myIO(data = mtcars) |>
        addIoLayer(type = "point", label = "scatter",
          mapping = list(x_var = "wt", y_var = "mpg"))
    })
  }
}
```

```
  shinyApp(ui, server)
}
```

---

myio\_chart\_schema      *Get myIO Chart Schema for LLM Tool Calling*

---

**Description**

Get myIO Chart Schema for LLM Tool Calling

**Usage**

```
myio_chart_schema(type = NULL)
```

**Arguments**

type                      Optional chart type. When NULL, returns every type schema.

**Value**

A list containing one chart schema or all chart schemas.

**Examples**

```
myio_chart_schema("boxplot")
```

---

myio\_function\_signature      *Get a myIO Function Signature for LLM Tool Calling*

---

**Description**

Get a myIO Function Signature for LLM Tool Calling

**Usage**

```
myio_function_signature(fn = NULL)
```

**Arguments**

fn                         Optional exported function name. When NULL, returns all signatures.

**Value**

A character vector of argument names or a named list of signatures.

**Examples**

```
myio_function_signature("setAxisFormat")
```

---

```
myIO_last_error      Diagnose myIO Rendering Errors
```

---

**Description**

Prints guidance on how to find the most recent JavaScript error from a myIO widget. In Shiny, errors are available as reactive inputs. Outside Shiny, errors appear in the browser's developer console.

**Usage**

```
myIO_last_error(outputId = NULL)
```

**Arguments**

`outputId` optional Shiny output ID (character string). If provided, prints the exact Shiny input key to read.

**Value**

Invisibly returns NULL. Called for its side effect (printing diagnostic guidance).

**Examples**

```
myIO_last_error()
myIO_last_error("chart1")
```

---

```
myio_list_chart_types List myIO Chart Types for LLM Tool Calling
```

---

**Description**

Returns chart type names from the generated myIO schema.

**Usage**

```
myio_list_chart_types()
```

**Value**

A character vector of chart type names.

**Examples**

```
head(myio_list_chart_types())
```

---

myio\_list\_functions     *List myIO Functions for LLM Tool Calling*

---

**Description**

List myIO Functions for LLM Tool Calling

**Usage**

```
myio_list_functions()
```

**Value**

A character vector of exported myIO function names.

**Examples**

```
head(myio_list_functions())
```

---

myio\_validate\_call     *Validate a myIO Function Call*

---

**Description**

Validate a myIO Function Call

**Usage**

```
myio_validate_call(fn, args = list())
```

**Arguments**

fn	Exported myIO function name.
args	Named list of proposed arguments.

**Value**

A list with valid and errors. Errors use stable machine-readable code values.

**Examples**

```
myio_validate_call("setAxisFormat", list(axis_x = ".0f"))
```

---

myio_validate_spec	<i>Validate a myIO Chart Specification</i>
--------------------	--------------------------------------------

---

**Description**

Validates a proposed chart specification against the generated myIO schema.

**Usage**

```
myio_validate_spec(spec, columns = NULL)
```

**Arguments**

spec	A list with type, mapping, optional transform, and optional columns.
columns	Optional named column type map. When supplied, overrides spec\$columns and enables missing-column and numeric-column checks.

**Value**

A list with valid and errors. Errors use stable machine-readable code values.

**Examples**

```
myio_validate_spec(list(
  type = "boxplot",
  mapping = list(column_var = "Species", value_var = "Sepal.Width")
))
```

---

setAnnotation	<i>Enable Click-to-Annotate</i>
---------------	---------------------------------

---

**Description**

Enables annotation mode where clicking a data point opens a label input. Annotations are stored as data and can be exported or accessed as a Shiny reactive input.

**Usage**

```
setAnnotation(myIO, labels = NULL, colors = NULL, mode = "click")
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
labels	character vector of preset label options (optional). If provided, shows a drop-down instead of free-text input.
colors	named character vector of category colors (optional). Names are category labels, values are CSS colors.
mode	"click" (default) to annotate individual points.

**Value**

A modified myIO htmlwidget object with annotation enabled.

**Examples**

```
myIO(data = mtcars) |>
  addIoLayer(
    type = "point", label = "pts",
    mapping = list(x_var = "wt", y_var = "mpg")
  ) |>
  setAnnotation(labels = c("outlier", "normal"))
```

---

 setAxisFormat

*Set Axis Format*


---

**Description**

Sets axis for x axis, y axis, and/or tool tip

**Usage**

```
setAxisFormat(
  myIO,
  xAxis = NULL,
  yAxis = NULL,
  toolTip = NULL,
  xLabel = NULL,
  yLabel = NULL
)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
xAxis	Optional string indicating a d3.js format for the x axis. When NULL, leaves the existing setting unchanged.
yAxis	Optional string indicating a d3.js format for the y axis. When NULL, leaves the existing setting unchanged.
toolTip	Optional string indicating a d3.js format for tooltips. When NULL, leaves the existing setting unchanged.
xLabel	Optional string label for the x axis. When NULL, leaves the existing setting unchanged.
yLabel	Optional string label for the y axis. When NULL, leaves the existing setting unchanged.

**Value**

A modified myIO htmlwidget object with updated axis format configuration. with options set for the tooltip formats

**Examples**

```
# Set axis formats using d3.js format strings
myIO() |> setAxisFormat(xAxis = ".0f", yAxis = ".1f")

# Set axis labels
myIO() |> setAxisFormat(xLabel = "Weight (lbs)", yLabel = "MPG")

# Label-only calls preserve previously configured formats
myIO() |> setAxisFormat(yAxis = ".2f") |> setAxisFormat(yLabel = "Rate")
```

---

setAxisLimits

*Set Axis Limits*

---

**Description**

Sets the minimum and maximum values for chart axes

**Usage**

```
setAxisLimits(
  myIO,
  xlim = list(min = NULL, max = NULL),
  ylim = list(min = NULL, max = NULL)
)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
xlim	a list of min and max values
ylim	a list of min and max values

**Value**

A modified myIO htmlwidget object with updated axis limits.

**Examples**

```
# Set x axis limits
myIO() |> setAxisLimits(xlim = list(min = 0, max = 100))

# Set both axis limits
myIO() |> setAxisLimits(
  xlim = list(min = 0, max = 50),
  ylim = list(min = -10, max = 200)
)
```

---

setBigData

*Attach a big-data source to a myIO widget*


---

**Description**

'setBigData()' declares the data source that a myIO widget should use for large-dataset rendering and linked-selection coordination.

**Usage**

```
setBigData(widget, source, rowkey_col = NULL, ...)
```

**Arguments**

widget	A myIO htmlwidget object.
source	A supported big-data source: a 'data.frame', an Arrow table or record-batch reader, a single file path or URL ending in '.parquet', '.arrow', '.feather', or '.csv', or a 'DBIConnection'.
rowkey_col	Optional name of the column that uniquely identifies rows for Crosstalk-compatible linked selections.
...	Additional source-specific options. For DBI sources, pass 'table = "name"' so myIO can query the table schema. For file path or URL sources, pass 'schema = c("col1", "col2", ...)' or a schema field list.

**Details**

This function writes the 'x.bigdata.\*' payload fields consumed by the widget's large-dataset virtualization path and follows its row-key contract. DBI sources are stored as an internal session-scoped marker in 'x.bigdata\$dbi\_handle\_internal'; render-time source registration is handled by the source registry phase.

**Value**

A modified myIO htmlwidget object.

**Examples**

```

myIO(data = mtcars) |>
  setBigData(mtcars)

myIO() |>
  setBigData("data/large.parquet", schema = c("id", "x", "y"), rowkey_col = "id")

if (requireNamespace("duckdb", quietly = TRUE)) {
  con <- DBI::dbConnect(duckdb::duckdb())
  obs <- data.frame(id = seq_len(nrow(mtcars)), mpg = mtcars$mpg)
  DBI::dbWriteTable(con, "observations", obs)
  myIO() |>
    setBigData(con, table = "observations", rowkey_col = "id")
  DBI::dbDisconnect(con, shutdown = TRUE)
}

```

---

setBrush

*Enable Brush Selection*


---

**Description**

Enables rectangle brush selection on the chart. When a user drags to select a region, the selected data points are available as a reactive input in Shiny or exportable in static HTML.

**Usage**

```
setBrush(myIO, direction = "xy", on_select = "highlight")
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
direction	brush direction: "xy" (default), "x", or "y"
on_select	behavior in static mode: "highlight" (default) or "export" (scopes CSV download to selected points)

**Value**

A modified myIO htmlwidget object with brush interaction enabled.

**Examples**

```

myIO(data = mtcars) |>
  addIoLayer(
    type = "point", label = "pts",
    mapping = list(x_var = "wt", y_var = "mpg")
  ) |>
  setBrush()

```

---

setColorScheme	<i>Set Color Scheme</i>
----------------	-------------------------

---

**Description**

Sets color scheme for a chart and the category names (optional)

**Usage**

```
setColorScheme(myIO, colorScheme = NULL, setCategories = NULL)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
colorScheme	a vector of colors in the order you want them used
setCategories	an optional vector of names that will be mapped to the corresponding color in the colorScheme

**Value**

A modified myIO htmlwidget object with updated color scheme.

**Examples**

```
# Set a custom color scheme
myIO() |> setColorScheme(colorScheme = list("red", "blue", "green"))

# Set colors with category labels
myIO() |> setColorScheme(
  colorScheme = list("steelblue", "orange"),
  setCategories = c("Group A", "Group B")
)
```

---

setExportOptions	<i>Configure Export Options</i>
------------------	---------------------------------

---

**Description**

Controls which export buttons appear in the chart toolbar.

**Usage**

```

setExportOptions(
  myIO,
  png = TRUE,
  svg = TRUE,
  pdf = TRUE,
  clipboard = TRUE,
  csv = TRUE,
  title = NULL
)

```

**Arguments**

myIO	A myIO widget object.
png	Logical. Show PNG download button. Default TRUE.
svg	Logical. Show SVG download button. Default TRUE.
pdf	Logical. Show PDF download button. Default TRUE.
clipboard	Logical. Show clipboard copy button. Default TRUE.
csv	Logical. Show CSV export button. Default TRUE.
title	Character or NULL. Chart title for PDF metadata.

**Value**

Modified myIO widget.

**Examples**

```

myIO(iris) |>
  addIoLayer("point", label = "pts",
            mapping = list(x_var = "Sepal.Length", y_var = "Sepal.Width")) |>
  setExportOptions(svg = TRUE, clipboard = TRUE, pdf = FALSE)

```

---

 setFacet

*Set Faceting (Small Multiples)*


---

**Description**

Splits the chart into a grid of panels, one per unique value of the faceting variable. Each panel shows the same layers filtered to that subset of the data.

**Usage**

```
setFacet(
  myIO,
  var,
  ncol = NULL,
  min_width = 200,
  scales = "fixed",
  label_position = "top"
)
```

**Arguments**

myIO	A myIO widget object.
var	Character. Column name to facet by. Must exist in at least one layer's data.
ncol	Integer or NULL. Number of columns in the grid. If NULL, auto-computes from min_width and container width.
min_width	Numeric. Minimum panel width in pixels when ncol is NULL. Default 200.
scales	Character. Scale sharing mode: <ul style="list-style-type: none"> <li>• "fixed" – all panels share x and y scales (default)</li> <li>• "free_x" – independent x scales per panel</li> <li>• "free_y" – independent y scales per panel</li> <li>• "free" – independent x and y scales per panel</li> </ul>
label_position	Character. Where to show panel labels: "top" (default) or "bottom".

**Value**

Modified myIO widget.

**Examples**

```
myIO(iris) |>
  addIoLayer("point", label = "pts",
            mapping = list(x_var = "Sepal.Length", y_var = "Sepal.Width")) |>
  setFacet("Species", ncol = 3)
```

---

setLayerOpacity	<i>Set Layer Opacity</i>
-----------------	--------------------------

---

**Description**

Set Layer Opacity

**Usage**

```
setLayerOpacity(myIO, label, opacity)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
label	character. The layer label to target.
opacity	numeric. Opacity value between 0 (transparent) and 1 (opaque).

**Value**

A modified myIO htmlwidget object.

**Examples**

```
myIO(iris) |>
  addIoLayer("point", label = "pts",
            mapping = list(x_var = "Sepal.Length", y_var = "Sepal.Width")) |>
  setLayerOpacity("pts", 0.5)
```

---

 setLinked
 

---



---

*Enable Linked Brushing via Crosstalk*


---

**Description**

Connects the chart to a Crosstalk SharedData object so that brush selections propagate to other linked widgets.

**Usage**

```
setLinked(
  myIO,
  shared_data,
  mode = "both",
  filter = FALSE,
  key = NULL,
  group = NULL,
  cursor = FALSE,
  cursorAxis = "x"
)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
shared_data	a crosstalk::SharedData object
mode	"source", "target", or "both" (default)
filter	if TRUE, Crosstalk filter operations hide non-matching points. Default FALSE (dim only).

key	Optional character vector of row keys. When supplied, overrides the keys extracted from shared_data. Useful when the SharedData keys do not match the column used for cross-chart matching.
group	Optional character string. When supplied, overrides the Crosstalk group name from shared_data, allowing manual control over which widgets share selections.
cursor	Logical. When TRUE, a hover in any linked chart draws a synchronized crosshair on every sibling chart in the same group. Default FALSE.
cursorAxis	Character. Which axis to sync: "x" (default), "y", or "xy". Only "x" is active in v1.2.

### Value

A modified myIO htmlwidget with Crosstalk linking.

### Examples

```
if (interactive() && requireNamespace("crosstalk", quietly = TRUE)) {
  shared <- crosstalk::SharedData$new(mtcars, key = ~rownames(mtcars))
  myIO() |>
    addIoLayer(
      type = "point", label = "scatter",
      data = shared$data(), mapping = list(x_var = "wt", y_var = "mpg")
    ) |>
    setLinked(shared)
}
```

---

setLinkedCursor      *Toggle Linked Cursor Sync on a myIO Widget*

---

### Description

Enables or disables synchronized hover crosshair across linked charts. Composable with [linkCharts](#) and [setLinked](#) via the pipe: call after linking to opt into cursor sync without re-linking. Preserves any existing interactions\$link configuration.

### Usage

```
setLinkedCursor(myIO, enabled = TRUE, axis = "x")
```

### Arguments

myIO	A myIO htmlwidget.
enabled	Logical. TRUE to turn cursor sync on, FALSE to turn it off. Default TRUE.
axis	Character. Which axis to sync: "x" (default), "y", or "xy". Only "x" is active in v1.2; other values are accepted and persisted but not yet rendered.

**Value**

A modified myIO htmlwidget.

**Examples**

```
w1 <- myIO() |>
  addIoLayer(type = "point", label = "a",
    data = mtcars, mapping = list(x_var = "wt", y_var = "mpg"))
w2 <- myIO() |>
  addIoLayer(type = "point", label = "b",
    data = mtcars, mapping = list(x_var = "hp", y_var = "mpg"))
linked <- linkCharts(w1, w2, on = "cyl")
linked[[1]] <- setLinkedCursor(linked[[1]])
linked[[2]] <- setLinkedCursor(linked[[2]])
```

---

 setMargin

*Set Chart Margins*


---

**Description**

Sets margins for the top, bottom, left, and right sides of the chart

**Usage**

```
setMargin(myIO, top = 20, bottom = 40, left = 50, right = 50)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
top	a numeric value representing in pixels the top margin
bottom	a numeric value representing in pixels the bottom margin
left	a numeric value representing in pixels the left margin
right	a numeric value representing in pixels the right margin

**Value**

A modified myIO htmlwidget object with updated margin configuration.

**Examples**

```
# Set custom margins
myIO() |> setMargin(top = 50, bottom = 80, left = 60, right = 20)
```

---

setReferenceLines	<i>Set Reference Lines</i>
-------------------	----------------------------

---

**Description**

Sets x and y reference lines

**Usage**

```
setReferenceLines(myIO, xRef = 0, yRef = 0)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
xRef	a list of the reference line value of x
yRef	a list of the reference line value of y

**Value**

A modified myIO htmlwidget object with reference lines added.

**Examples**

```
# Add reference lines at x=5 and y=20  
myIO() |> setReferenceLines(xRef = 5, yRef = 20)
```

---

setSlider	<i>Add a Parameter Slider (Shiny Only)</i>
-----------	--------------------------------------------

---

**Description**

Adds a slider control below the chart that adjusts a transform option and triggers reactive re-rendering in Shiny.

**Usage**

```
setSlider(myIO, param, label, min, max, value, step = NULL, debounce = 200)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
param	transform option name (e.g., "ci_level", "degree")
label	display label for the slider
min	minimum value
max	maximum value
value	default value
step	step size (default: NULL for auto)
debounce	debounce delay in milliseconds (default: 200)

**Value**

A modified myIO htmlwidget with slider config attached.

**Examples**

```
## Not run:
# In a Shiny server function:
output$chart <- renderMyIO({
  myIO(data = mtcars) |>
    addIoLayer(
      type = "regression", label = "fit",
      mapping = list(x_var = "wt", y_var = "mpg"),
      options = list(ci_level = 0.95)
    ) |>
    setSlider("ci_level", "Confidence level", 0.80, 0.99, 0.95, 0.01)
})

## End(Not run)
```

---

 setTheme

*Set Chart Theme*


---

**Description**

Sets chart theme tokens using CSS custom properties

**Usage**

```
setTheme(
  myIO,
  text_color = NULL,
  grid_color = NULL,
  bg = NULL,
  font = NULL,
```

```

    mode = NULL,
    preset = NULL,
    overrides = list(),
    ...
)

```

### Arguments

myIO	an htmlwidget object created by the myIO() function
text_color	text and label color
grid_color	grid line color
bg	background color
font	font family
mode	Character or NULL. Theme mode: "light", "dark", or "auto". Default NULL (no mode, manual CSS vars only).
preset	Character or NULL. Named preset (reserved for future use). Default NULL.
overrides	Named list of CSS custom property overrides (e.g., list("--chart-tooltip-bg" = "#222")).
...	additional CSS custom property overrides with '-' prefix

### Value

A modified myIO htmlwidget object with updated theme configuration.

### Examples

```

myIO() |>
  setTheme(text_color = "#222222", grid_color = "#d9d9d9")

myIO() |>
  setTheme(mode = "dark", bg = "#1a1a2e")

```

---

setTitle

*Set Chart Title*

---

### Description

Sets the title rendered inside the myIO widget SVG.

### Usage

```
setTitle(myIO, title = NULL)
```

**Arguments**

myIO	A myIO widget object.
title	Character title or NULL to remove the title.

**Value**

A modified myIO htmlwidget object.

**Examples**

```
myIO() |>
  setTitle("Miles per gallon") |>
  addIoLayer("point", label = "cars",
            data = mtcars,
            mapping = list(x_var = "wt", y_var = "mpg"))
```

---

 setToggle

*Set Toggle Interaction*


---

**Description**

Sets toggle options for y\_var and adds a toggle button for chart

**Usage**

```
setToggle(myIO, variable, format = NULL)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
variable	a string indicating the variable name in data for toggle
format	a string indicating the format for the toggled variable

**Value**

A modified myIO htmlwidget object with toggle interaction configured.

**Examples**

```
# Add a toggle button to switch y variable
myIO() |> setToggle(variable = "Percent", format = ".0%")
```

---

setToolTipOptions      *Set Tooltip Options*

---

**Description**

Generic function for setting tool tip options for a chart

**Usage**

```
setToolTipOptions(myIO, suppressY = NULL)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
suppressY	a boolean

**Value**

A modified myIO htmlwidget object with updated tooltip options.

**Examples**

```
# Suppress the y value in tooltips  
myIO() |> setToolTipOptions(suppressY = TRUE)
```

---

setTransitionSpeed      *Set Transition Speed*

---

**Description**

Sets transition speeds across the chart (set to 0 to suppress)

**Usage**

```
setTransitionSpeed(myIO, speed)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
speed	a number indicating the speed of transition in milliseconds

**Value**

A modified myIO htmlwidget object with updated transition speed.

**Examples**

```
# Set transition speed to 500ms
myIO() |> setTransitionSpeed(speed = 500)

# Disable transitions
myIO() |> setTransitionSpeed(speed = 0)
```

---

suppressAxis	<i>Suppress Axis Display</i>
--------------	------------------------------

---

**Description**

Suppresses axes from printing

**Usage**

```
suppressAxis(myIO, xAxis = NULL, yAxis = NULL)
```

**Arguments**

myIO	an htmlwidget object created by the myIO() function
xAxis	a logical operator defining whether the x axis should be printed or not
yAxis	a logical operator defining whether the y axis should be printed or not

**Value**

A modified myIO htmlwidget object with axis display suppressed.

**Examples**

```
# Suppress both axes
myIO() |> suppressAxis(xAxis = TRUE, yAxis = TRUE)

# Suppress only the x axis
myIO() |> suppressAxis(xAxis = TRUE)
```

---

suppressLegend	<i>Suppress Legend Display</i>
----------------	--------------------------------

---

**Description**

Suppresses legend from printing

**Usage**

```
suppressLegend(myIO, suppressLegend = TRUE)
```

**Arguments**

myIO                    an htmlwidget object created by the myIO() function

suppressLegend    a logical operator defining whether the legend should be printed or not

**Value**

A modified myIO htmlwidget object with legend display suppressed.

**Examples**

```
# Hide the legend  
myIO() |> suppressLegend()
```

# Index

addIoLayer, 3

clear\_duckdb\_wasm\_cache, 4

defineCategoricalAxis, 4  
dragPoints, 5  
duckdb\_wasm\_status, 6

flipAxis, 6

install\_duckdb\_wasm, 7

linkCharts, 8, 23

myIO, 9  
myIO-shiny, 10  
myio\_chart\_schema, 11  
myio\_function\_signature, 11  
myIO\_last\_error, 12  
myio\_list\_chart\_types, 12  
myio\_list\_functions, 13  
myio\_validate\_call, 13  
myio\_validate\_spec, 14  
myIOOutput (myIO-shiny), 10

renderMyIO (myIO-shiny), 10

setAnnotation, 14  
setAxisFormat, 15  
setAxisLimits, 16  
setBigData, 9, 17  
setBrush, 18  
setColorScheme, 19  
setExportOptions, 19  
setFacet, 20  
setLayerOpacity, 21  
setLinked, 8, 22, 23  
setLinkedCursor, 23  
setMargin, 24  
setReferenceLines, 25  
setSlider, 25  
setTheme, 26  
setTitle, 27  
setToggle, 28  
setToolTipOptions, 29  
setTransitionSpeed, 29  
suppressAxis, 30  
suppressLegend, 31