# Package 'NGLVieweR'

January 20, 2025

**Title** Interactive 3D Visualization of Molecular Structures

**Version** 1.4.0

**Maintainer** Niels van der Velden <n.s.j.vandervelden@gmail.com>

**Description** Provides an 'htmlwidgets' <https://www.htmlwidgets.org/> interface to 'NGL.js' <http://nglviewer.org/ngl/api/>. 'NGLvieweR' can be used to visualize and interact with protein databank ('PDB') and structural files in R and Shiny applications.
It includes a set of API functions to manipulate the viewer after creation in Shiny.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** htmlwidgets, magrittr, tools, shiny

**Suggests** knitr, webshot, markdown, rmarkdown, colourpicker

**VignetteBuilder** knitr

**URL** https://github.com/nvelden/NGLVieweR

**BugReports** https://github.com/nvelden/NGLVieweR/issues

**NeedsCompilation** no

**Author** Niels van der Velden [aut, cre],
Alexander Rose [cph] (NGL.js library)

**Repository** CRAN

**Date/Publication** 2024-11-22 19:10:02 UTC

## Contents

## `addRepresentation` *Add representation*

### Description

Add a representation and its parameters.

### Usage

```
addRepresentation(NGLVieweR, type, param = list())
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `type` | Type of representation. Most common options are "cartoon", "ball+stick", "line", "surface", "ribbon" and "label". For a full list of options, see the "structureRepresentation" method in the official NGL.js manual. |
| `param` | Options for the different types of representations. Most common options are `name`, `opacity`, `colorScheme`, `sele`, `colorValue` and `visibility`. For a full list of options, see the general "RepresentationParameters" method and type specific Label-, Structure- and Surface- RepresentationParameters in the official NGL.js manual. |

## Value

List of representation parameters to `NGLVieweR` `htmlwidgets` object.

## See Also

- `addSelection()`
- `NGLVieweR_example()` See example "basic".

## Examples

```
NGLVieweR("7CID") %>%
  stageParameters(backgroundColor = "black") %>%
  addRepresentation("cartoon",
  param = list(name = "cartoon", colorValue = "blue")) %>%
  addRepresentation("ball+stick",
    param = list(
      name = "ball+stick", sele = "241",
      colorScheme = "element", colorValue = "yellow"
    )
  ) %>%
  addRepresentation("label",
    param = list(
      name = "label",
      showBackground = TRUE,
      labelType = "res",
      color = "black",
      backgroundColor = "white",
      backgroundOpacity = 0.8,
      sele = ":A and 241 and .CG"
    )
  )

# Shiny context
if (interactive()) {
  library(shiny)
  ui <- fluidPage(NGLVieweROutput("structure"))
  server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        stageParameters(backgroundColor = "black") %>%
        addRepresentation("cartoon",
          param = list(name = "cartoon", colorValue = "blue")
        ) %>%
        addRepresentation("ball+stick",
          param = list(
            name = "ball+stick", sele = "241",
            colorScheme = "element"
          )
        ) %>%
        addRepresentation("label",
          param = list(
            name = "label",
            showBackground = TRUE,
            labelType = "res",
            colorValue = "black",
            backgroundColor = "white",
            backgroundOpacity = 0.8,
            sele = ":A and 241 and .CG"
          )
        )
    })
  }
  shinyApp(ui, server)
}
```

---

## addSelection *Add a selection*

---

### Description

Add a new selection to a NGLVieweR object in Shinymode.

### Usage

```
addSelection(NGLVieweR_proxy, type, param = list(), structureIndex = NULL)
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `type` | Type of representation. Most common options are "cartoon", "ball+stick", "surface", "ribbon" and "label". |
| `param` | Options for the different types of representations. Most common options are `name`, `opacity`, `colorScheme`, `sele`, `colorValue` and `visibility`. For a full list of options, see the general "RepresentationParameters" method and type specific Label-, Structure- and Surface- RepresentationParameters in the official NGL.js manual. |
| `structureIndex` | (optional) The index of the specific structure to which the selection should be added (index 0 for the first). If not specified, the selection will be applied to all loaded structures. |

### Value

API call containing `NGLVieweR id` and list of message parameters.

### See Also

- `updateRepresentation()` Update an existing NGLVieweR representation.
- `NGLVieweR_example()` See example "addSelection".

Other selections: `removeSelection()`, `updateSelection()`

## Examples

```
## Not run:
NGLVieweR_proxy("7CID") %>%
 addSelection("ball+stick", param = list(name="sel1",
                                        sele="1-20",
                                        colorValue="yellow",
                                        colorScheme="element"
                                        ))

## End(Not run)

if (interactive()) {
library(shiny)

ui <- fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      textInput("selection", "Selection", "1-20"),
      selectInput("type", "Type", c("ball+stick", "cartoon", "backbone")),
      selectInput("color", "Color", c("orange", "grey", "white")),
      actionButton("add", "Add"),
      actionButton("remove", "Remove")
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server <- function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
        param = list(name = "cartoon", colorScheme = "residueindex")
      )
  })
  observeEvent(input$add, {
    NGLVieweR_proxy("structure") %>%
      addSelection(isolate(input$type),
        param =
          list(
            name = "sel1",
            sele = isolate(input$selection),
            colorValue = isolate(input$color)
          )
      )
  })

  observeEvent(input$remove, {
    NGLVieweR_proxy("structure") %>%
      removeSelection("sel1")
  })
}
shinyApp(ui, server)
}
```

---

`addStructure` *Add structure*

---

## Description

Add a structure to the NGLVieweR object, either from a PDB entry code, a file, or directly from the R environment.

## Usage

```
addStructure(NGLVieweR, data, format = NULL)
```

## Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `data` | Structure data to be added. Can be a PDB entry code (e.g. "7CID"), a file path to a structure file, or a text representation of the structure. |
| `format` | Format of the structure file, if reading from a file. Supported formats are "pdb", "cif", etc. If the file format cannot be inferred from the file name, this parameter must be specified. |

## Details

This function allows you to add a structure to the NGLVieweR widget. You can add the structure using a PDB entry code, by specifying a local file, or by providing the structure data directly. If the format is not clear from the input, you may need to specify it using the `format` parameter.

## Value

An updated NGLVieweR object with the added structure.

## Examples

```
NGLVieweR("1CRN") %>%
  addRepresentation("cartoon", param = list(color = "blue")) %>%
  addStructure("1CRN") %>%
  addRepresentation("cartoon", param = list(color = "orange")) %>%
  setPosition(x = 20, y = 0, z = 0) %>%
  setRotation(x = 2, y = 0, z = 0, degrees = FALSE) %>%
  setScale(0.5)

# Note: The first "1CRN" structure is represented in blue, while the second
# "1CRN" structure is represented in orange. Transformations such as
# setPosition, setRotation, and setScale apply to the second
# (most recently added) structure.
```

## `NGLVieweR` *Create a NGLVieweR*

### Description

NGLVieweR can be used to visualize and interact with Protein Data Bank (PDB) and structural files in R and Shiny applications. It includes a set of API functions to manipulate the viewer after creation in Shiny.

### Usage

```
NGLVieweR(data, format = NULL, width = NULL, height = NULL, elementId = NULL)
```

### Arguments

| | |
|---|---|
| `data` | PDB file or PDB entry code |
| `format` | Input format (.mmcif, .cif, .mcif, .pdb, .ent, .pqr, .gro, .sdf, .sd, .mol2, .mmtf). Needed when no file extension is provided. |
| `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. |
| `elementId` | optional element Id |

### Details

The package is based on the NGL.js JavaScript library. To see the full set of features please read the official manual of NGL.js.

### Value

A `NGLVieweR` `htmlwidgets` object.

### See Also

- `NGLVieweR_proxy()` for handling of API calls after rendering.
- `NGLVieweR_example()` See example "API" and "basic".

### Examples

```
# Example 1: Most Basic
NGLVieweR("7CID") %>%
 addRepresentation("cartoon",
 param = list(name = "cartoon", colorScheme="residueindex"))

# Example 2: Advanced
NGLVieweR("7CID") %>%
  stageParameters(backgroundColor = "white") %>%
  setQuality("high") %>%
  setSpin(FALSE) %>%
  addRepresentation("cartoon",
    param = list(
      name = "cartoon",
      colorScheme = "residueindex"
    )
  ) %>%
  addRepresentation("ball+stick",
    param = list(
      name = "ball+stick",
      colorValue = "red",
      colorScheme = "element",
      sele = "200"
    )
  ) %>%
  addRepresentation("label",
    param = list(
      name = "label", sele = "200:A.O",
      showBackground = TRUE,
      backgroundColor = "black",
      backgroundMargin = 2,
      backgroundOpacity = 0.5,
      showBorder = TRUE,
      colorValue = "white"
    )
  ) %>%
  addRepresentation("surface",
    param = list(
      name = "surface",
      colorValue = "white",
      opacity = 0.1
    )
  ) %>%
  zoomMove("200", "200", 2000, -20)

#---------------------Using Shiny-------------------------

# App 1: Basic Example
if (interactive()) {
  library(shiny)
  ui <- fluidPage(NGLVieweROutput("structure"))
  server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        addRepresentation("cartoon",
          param = list(
            name = "cartoon",
            colorScheme = "residueindex"
          )
        ) %>%
        addRepresentation("ball+stick",
          param = list(
            name = "cartoon",
            sele = "1-20",
            colorScheme = "element"
          )
        ) %>%
        stageParameters(backgroundColor = "black") %>%
        setQuality("high") %>%
        setFocus(0) %>%
        setSpin(TRUE)
    })
  }
  shinyApp(ui, server)
}

# App 2: Example with API calls
if (interactive()) {
library(shiny)

ui <- fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      textInput("selection", "Selection", "1-20"),
      selectInput("type", "Type", c("ball+stick", "cartoon", "backbone")),
      selectInput("color", "Color", c("orange", "grey", "white")),
      actionButton("add", "Add"),
      actionButton("remove", "Remove")
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server <- function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
        param = list(name = "cartoon", colorScheme = "residueindex")
      ) %>%
      stageParameters(backgroundColor = input$backgroundColor) %>%
      setQuality("high") %>%
      setFocus(0) %>%
      setSpin(TRUE)
  })
  observeEvent(input$add, {
    NGLVieweR_proxy("structure") %>%
      addSelection(isolate(input$type),
        param =
          list(
            name = "sel1",
            sele = isolate(input$selection),
            colorValue = isolate(input$color)
          )
      )
  })

  observeEvent(input$remove, {
    NGLVieweR_proxy("structure") %>%
      removeSelection("sel1")
  })
}
shinyApp(ui, server)
}
```

---

`NGLVieweR-shiny` *Shiny bindings for NGLVieweR*

---

## Description

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

## Usage

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

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

NGLVieweR_proxy(id, session = shiny::getDefaultReactiveDomain())
```

## 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 NGLVieweR. |
| `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. |
| `id` | single-element character vector indicating the output ID of the chart to modify (if invoked from a Shiny module, the namespace will be added automatically) |
| `session` | The Shiny session object to which the map belongs; usually the default value will suffice. |

## Value

`NGLVieweR` object that can be placed in the UI.

## See Also

NGLVieweR_example()

## Examples

```
if (interactive()) {
library(shiny)

ui <- fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      textInput("selection", "Selection", "1-20"),
      selectInput("type", "Type", c("ball+stick", "cartoon", "backbone")),
      selectInput("color", "Color", c("orange", "grey", "white")),
      actionButton("add", "Add"),
      actionButton("remove", "Remove")
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server <- function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
        param = list(name = "cartoon", color = "residueindex")
      ) %>%
      stageParameters(backgroundColor = input$backgroundColor) %>%
      setQuality("high") %>%
      setFocus(0) %>%
      setSpin(TRUE)
  })
  observeEvent(input$add, {
    NGLVieweR_proxy("structure") %>%
      addSelection(isolate(input$type),
        param =
          list(
            name = "sel1",
            sele = isolate(input$selection),
            color = isolate(input$color)
          )
      )
  })

  observeEvent(input$remove, {
    NGLVieweR_proxy("structure") %>%
      removeSelection("sel1")
  })
}
shinyApp(ui, server)
}
```

## NGLVieweR_example — *Run NGLVieweR example Shiny app*

### Description

Launch an example to demonstrate how to use `NGLvieweR` in Shiny.

### Usage

```
NGLVieweR_example(example = "basic")
```

### Arguments

| | |
|---|---|
| `example` | Example type for which to see an example, possible values are: "basic", "API", "addSelection", "removeSelection", "snapshot", "updateAnimation", "updateColor", "updateFocus", "updateFullscreen", "updateRepresentation", "updateSelection", "updateStage", "updateVisibility", "updateZoomMove", "multiStructureSelection". |

### Value

Call to load Shiny example.

### Examples

```
if (interactive()) {

# Basic example
NGLVieweR_example("basic")

# Example with API calls
NGLVieweR_example("API")

# Function specific example
NGLVieweR_example("addSelection")
}
```

## removeSelection *Remove a selection*

### Description

Remove an existing NGLVieweR selection in Shinymode.

### Usage

```
removeSelection(NGLVieweR_proxy, name)
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `name` | Name of selection to be removed. |

### Value

API call containing `NGLVieweR` `id` and list of message parameters.

### See Also

- `NGLVieweR_example()` See example "removeSelection".

Other selections: `addSelection()`, `updateSelection()`

### Examples

```
## Not run:
NGLVieweR_proxy("structure") %>%
     removeSelection("sel1")

## End(Not run)

if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    titlePanel("Viewer with API inputs"),
    sidebarLayout(
      sidebarPanel(
        textInput("selection", "Selection", "1-20"),
        selectInput("type", "Type", c("ball+stick", "cartoon", "backbone")),
        selectInput("color", "Color", c("orange", "grey", "white")),
        actionButton("add", "Add"),
        actionButton("remove", "Remove")
      ),
      mainPanel(
        NGLVieweROutput("structure")
      )
    )
  )
  server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        addRepresentation("cartoon",
          param = list(name = "cartoon", colorScheme = "residueindex")
        )
    })
    observeEvent(input$add, {
      NGLVieweR_proxy("structure") %>%
        addSelection(isolate(input$type),
          param =
            list(
              name = "sel1",
              sele = isolate(input$selection),
              colorValue = isolate(input$color)
            )
        )
    })

    observeEvent(input$remove, {
      NGLVieweR_proxy("structure") %>%
        removeSelection("sel1")
    })
  }
  shinyApp(ui, server)
}
```

## `selectionParameters` *Set selection parameters*

### Description

Set selection parameters.

### Usage

```
selectionParameters(NGLVieweR, proximity = 3, level = "residue")
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `proximity` | Set distance in angstrom for atoms to return in proximity of selection. Default = `3`. |
| `level` | Set level on which atoms in proximity of selection are returned. Options are "residue" (default) or atom". |

## Value

Returns list of selection parameters to `NGLVieweR` `htmlwidgets` object.

## Examples

```
NGLVieweR("7CID") %>%
 addRepresentation("cartoon") %>%
 selectionParameters(3, "residue")

# Shiny context
if (interactive()) {
   library(shiny)
   ui <- fluidPage(NGLVieweROutput("structure"))
   server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        addRepresentation("cartoon") %>%
        selectionParameters(3, "residue")
    })
    observeEvent(input$structure_selAround, {
      NGLVieweR_proxy("structure") %>% removeSelection("selAround")
      NGLVieweR_proxy("structure") %>%
        addSelection(
          "ball+stick",
          param =
            list(
              name = "selAround",
              sele = input$structure_selAround,
              colorValue = "grey"
            )
        )
    })
  }
  shinyApp(ui, server)
}
```

---

`setFocus` *Set Focus*

---

## Description

Set Focus

## Usage

```
setFocus(NGLVieweR, focus = 0)
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `focus` | Set focus between 0 (default) to 100. |

### Value

setFocus parameter in `NGLVieweR` `htmlwidgets` object.

### See Also

`updateFocus()`

Other options: `setQuality()`, `snapShot()`, `updateFocus()`, `updateFullscreen()`

### Examples

```
NGLVieweR("7CID") %>%
  addRepresentation("cartoon",
  param=list(name="cartoon", colorValue="blue")) %>%
  setFocus(0)
```

---

## `setPosition` *Set Position*

---

### Description

Set position for the representation

### Usage

```
setPosition(NGLVieweR, x = 0, y = 0, z = 0)
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `x` | Position along the x-axis in angstroms. Default is 0. |
| `y` | Position along the y-axis in angstroms. Default is 0. |
| `z` | Position along the z-axis in angstroms. Default is 0. |

### Value

NGLVieweR object with updated setPosition parameters.

### See Also

- `setScale()`
- `zoomMove()`
- `setRotation()`

Other transformations: `setRotation()`, `setScale()`, `zoomMove()`

### Examples

```
NGLVieweR("7CID") %>%
stageParameters(backgroundColor = "white") %>%
addRepresentation("cartoon", param=list(name="cartoon", colorValue="red")) %>%
addRepresentation("ball+stick", param=list(name="ball+stick",
colorValue="yellow",
colorScheme="element",
sele="200")) %>%
setPosition(25, 0, 0)
```

---

## `setQuality` *Set Quality*

---

### Description

Set Quality

### Usage

```
setQuality(NGLVieweR, quality = "medium")
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `quality` | Set rendering quality. Can be "low", "medium" (default) or "high". |

### Value

setQuality parameter in `NGLVieweR` `htmlwidgets` object.

### See Also

Other options: `setFocus()`, `snapShot()`, `updateFocus()`, `updateFullscreen()`

### Examples

```
NGLVieweR("7CID") %>%
  addRepresentation("cartoon",
  param=list(name="cartoon", colorValue="blue")) %>%
  setQuality("medium")
```

## `setRock` *Set rock*

### Description

Set rock animation

### Usage

```
setRock(NGLVieweR, rock = TRUE)
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `rock` | If `TRUE` (default), start rocking and stop spinning. |

### Value

setRock parameter to `TRUE` or `FALSE` in `NGLVieweR` `htmlwidgets` object.

### See Also

- `setSpin()`
- `updateRock()`

Other animations: `setSpin()`, `updateRock()`, `updateSpin()`, `updateZoomMove()`

### Examples

```
NGLVieweR("7CID") %>%
 addRepresentation("cartoon", param=list(name="cartoon", colorValue="blue")) %>%
 setRock(TRUE)
```

## `setRotation` *Rotate View*

### Description

Set rotation for the representation

### Usage

```
setRotation(NGLVieweR, x = 0, y = 0, z = 0, degrees = TRUE)
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `x` | Rotation angle around the x-axis. Default is 0. |
| `y` | Rotation angle around the y-axis. Default is 0. |
| `z` | Rotation angle around the z-axis. Default is 0. |
| `degrees` | A logical value. If TRUE (default), the input angles are assumed to be in degrees and will be converted to radians. If FALSE, the angles are assumed to be in radians. |

### Value

NGLVieweR object with updated rotateView parameters.

### See Also

- `setScale()`
- `zoomMove()`
- `setPosition()`

Other transformations: `setPosition()`, `setScale()`, `zoomMove()`

### Examples

```
NGLVieweR("7CID") %>%
stageParameters(backgroundColor = "white") %>%
addRepresentation("cartoon", param=list(name="cartoon", colorValue="red")) %>%
addRepresentation("ball+stick", param=list(name="ball+stick",
colorValue="yellow",
colorScheme="element",
sele="200")) %>%
setRotation(30, 45, 60, degrees = TRUE)
```

## `setScale` *Set Scale*

### Description

Set the scale factor for the representation

### Usage

```
setScale(NGLVieweR, scale = 1)
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `scale` | A numeric value indicating the scale factor (default is 1). |

### Value

Updated NGLVieweR object with new scale parameter.

### See Also

- zoomMove()
- setRotation()
- setPosition()

Other transformations: setPosition(), setRotation(), zoomMove()

### Examples

```
NGLVieweR("7CID") %>%
addRepresentation("cartoon",
param=list(name="cartoon", colorValue="blue")) %>%
setScale(2)
```

---

## setSpin *Set Spin*

---

### Description

Set Spin animation

### Usage

```
setSpin(NGLVieweR, spin = TRUE)
```

### Arguments

| | |
|---|---|
| NGLVieweR | A NGLVieweR object. |
| spin | If TRUE (default), start spinning and stop rocking |

### Value

setSpin parameter to TRUE or FALSE in NGLVieweR htmlwidgets object.

### See Also

- setRock()
- updateSpin()

Other animations: setRock(), updateRock(), updateSpin(), updateZoomMove()

### Examples

```
NGLVieweR("7CID") %>%
 addRepresentation("cartoon", param=list(name="cartoon", colorValue="blue")) %>%
 setSpin(TRUE)
```

## setSuperpose *Set superpose*

### Description

Enable or disable superposition of multiple structures, with options to specify the reference structure and selection strings for alignment.

### Usage

```
setSuperpose(
  NGLVieweR,
  reference = 1,
  sele_reference,
  sele_target,
  superpose = TRUE
)
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `reference` | The index of the reference structure to align other structures to. Defaults to 1 (the first loaded structure). |
| `sele_reference` | Selection string for the reference structure, specifying which parts to align. Mandatory. |
| `sele_target` | Selection string for each target structure, specifying which parts to align. Mandatory. |
| `superpose` | Logical; if `TRUE` (default), enable superposition of multiple structures. Set to `FALSE` to disable. |

### Value

Sets the `superpose` list in the `NGLVieweR` `htmlwidgets` object.

### Examples

```
NGLVieweR("1GZM") %>%
  addRepresentation("cartoon", param = list(color = "blue")) %>%
  addStructure("1U19") %>%
  addRepresentation("cartoon", param = list(color = "orange")) %>%
  setSuperpose(
    reference = 1,
    sele_reference = ":A",
    sele_target = ":A",
    superpose = TRUE
  )
```

`snapShot` *Snapshot*

## Description

Make a snapshot of a NGLVieweR object in Shinymode.

## Usage

```
snapShot(NGLVieweR_proxy, fileName = "Snapshot", param = list())
```

## Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `fileName` | Optional name for Snapshot (default = "Snapshot"). |
| `param` | Of type list, can be; antialias `TRUE/FALSE`, trim `TRUE/FALSE`, transparent `TRUE/FALSE` or scale `numeric`. For a full list of options, see "makeImage" and "ImageParameters" in the official NGL.js manual. |

## Value

API call containing `NGLVieweR id` and list of message parameters.

## See Also

`NGLVieweR_example()` See example "snapshot".

Other options: `setFocus()`, `setQuality()`, `updateFocus()`, `updateFullscreen()`

## Examples

```
## Not run:
NGLVieweR_proxy("structure") %>%
snapShot("Snapshot", param = list(
                              antialias = TRUE,
                              trim = TRUE,
                              transparent = TRUE,
                              scale = 1))

## End(Not run)

if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    titlePanel("Viewer with API inputs"),
    sidebarLayout(
      sidebarPanel(
        actionButton("snapshot", "Snapshot"),
      ),
      mainPanel(
        NGLVieweROutput("structure")
      )
    )
  )
  server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        addRepresentation("cartoon",
          param = list(
            name = "cartoon",
            color = "residueindex"
          )
        )
    })
    observeEvent(input$snapshot, {
      NGLVieweR_proxy("structure") %>%
        snapShot("Snapshot",
          param = list(
            antialias = TRUE,
            trim = TRUE,
            transparent = TRUE,
            scale = 1
          )
        )
    })
  }
  shinyApp(ui, server)
}
```

---

`stageParameters` *Set stage parameters*

---

## Description

Set stage parameters.

## Usage

```
stageParameters(NGLVieweR, ...)
```

## Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `...` | Options controlling the stage. Most common options are `backgroundColor`, `rotateSpeed`, `zoomSpeed`, `hoverTimeout` and `lightIntensity`. For a full list of options, see the "stageParameters" method in the official NGL.js manual. |

### Value

Returns list of stage parameters to NGLVieweR htmlwidgets object.

### See Also

- updateStage()
- NGLVieweR_example() See example "basic".

### Examples

```
NGLVieweR("7CID") %>%
 stageParameters(backgroundColor = "white", zoomSpeed = 1) %>%
 addRepresentation("cartoon", param = list(name = "cartoon", colorScheme="residueindex"))

if (interactive()) {
  library(shiny)
  ui <- fluidPage(NGLVieweROutput("structure"))
  server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        stageParameters(backgroundColor = "white", zoomSpeed = 1) %>%
        addRepresentation("cartoon",
          param = list(name = "cartoon", colorScheme = "residueindex")
        )
    })
  }
  shinyApp(ui, server)
}
```

---

## updateColor *Update color of a selection*

---

### Description

Update color of an existing NGLVieweR selection in Shinymode.

### Usage

```
updateColor(NGLVieweR_proxy, name, color)
```

### Arguments

| | |
|---|---|
| NGLVieweR_proxy | A NGLVieweR object. |
| name | Name of selection to alter the color. |
| color | Can be a colorValue (color name or HEX code) or colorScheme (e.g. "element", "resname", "random" or "residueindex"). For a full list of options, see the "Colormaker" section in the official NGL.js manual. |

## Value

API call containing `NGLVieweR` `id` and list of message parameters.

## See Also

- `NGLVieweR_example()` See example "updateColor".

Other updates: `updateRepresentation()`, `updateStage()`, `updateVisibility()`

## Examples

```
## Not run:
NGLVieweR_proxy("structure") %>%
     updateColor("cartoon", "red")

## End(Not run)

if (interactive()) {
  library(shiny)
  library(colourpicker)

  ui <- fluidPage(
    titlePanel("Viewer with API inputs"),
    sidebarLayout(
      sidebarPanel(
        colourInput("color", "red", "red"),
        actionButton("update", "Update"),
      ),
      mainPanel(
        NGLVieweROutput("structure")
      )
    )
  )
  server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        addRepresentation("cartoon",
          param = list(name = "cartoon", color = "residueindex")
        )
    })
    observeEvent(input$update, {
      NGLVieweR_proxy("structure") %>%
        updateColor("cartoon", isolate(input$color))
    })
  }
  shinyApp(ui, server)
}
```

`updateFocus` *Update Focus*

## Description

Update the focus of an existing NGLVieweR object in Shinymode.

## Usage

```
updateFocus(NGLVieweR_proxy, focus = 0)
```

## Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `focus` | Numeric value between 0-100 (default = 0). |

## Value

API call containing `NGLVieweR id` and list of message parameters.

## See Also

- `setFocus()`
- `NGLVieweR_example()` See example "updateFocus".

Other options: `setFocus()`, `setQuality()`, `snapShot()`, `updateFullscreen()`

## Examples

```
## Not run:
NGLVieweR_proxy("structure") %>%
  updateFocus(focus = 50)

## End(Not run)

if (interactive()) {
  library(shiny)
  ui = fluidPage(
    titlePanel("Viewer with API inputs"),
    sidebarLayout(
      sidebarPanel(
        sliderInput("focus", "Focus", 0, 100, 50)
      ),
      mainPanel(
        NGLVieweROutput("structure")
      )
    )
  )
  server = function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        addRepresentation("cartoon",
        param = list(name = "cartoon", color= "red"))
    })
    observeEvent(input$focus, {
      NGLVieweR_proxy("structure") %>%
        updateFocus(input$focus)
    })
  }
shinyApp(ui, server)
}
```

---

## updateFullscreen *Fullscreen*

---

### Description

Put viewer in fullscreen. Works in Shinymode.

### Usage

```
updateFullscreen(NGLVieweR_proxy, fullscreen = TRUE)
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `fullscreen` | If `TRUE` put viewer in fullscreen. |

### Value

API call containing `NGLVieweR` `id` and list of message parameters.

### See Also

`NGLVieweR_example()` See example "updateFullscreen".

Other options: `setFocus()`, `setQuality()`, `snapShot()`, `updateFocus()`

### Examples

```
## Not run:
NGLVieweR_proxy("structure") %>% updateFullscreen()

## End(Not run)

if (interactive()) {
library(shiny)
ui <- fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      actionButton("fullscreen", "Fullscreen"),
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server = function(input, output) {
 output$structure <- renderNGLVieweR({
  NGLVieweR("7CID") %>%
    addRepresentation("cartoon",
      param = list(name = "cartoon", color = "red")
    )
})

  observeEvent(input$fullscreen, {
  NGLVieweR_proxy("structure") %>%
    updateFullscreen()
})
}
  shinyApp(ui, server)
}
```

## `updateRepresentation` *Update Representation*

### Description

Update an existing NGLVieweR representation in Shinymode.

### Usage

```
updateRepresentation(NGLVieweR_proxy, name, param = list())
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `name` | Name of representation to alter the color. |
| `param` | Options for the different types of representations. Most common options are `name`, `opacity`, `colorScheme`, `colorValue` and `visibility`. For a full list of options, see the general "RepresentationParameters" method and type specific Label-, Structure- and Surface- RepresentationParameters in the official NGL.js manual. |

## Value

API call containing `NGLVieweR` `id` and list of message parameters.

## See Also

- `addSelection()` Add a new selection to a NGLVieweR object.
- `addRepresentation()`
- `NGLVieweR_example()` See example "updateRepresentation".

Other updates: `updateColor()`, `updateStage()`, `updateVisibility()`

## Examples

```
## Not run:
NGLVieweR_proxy("structure") %>%
  updateRepresentation("cartoon",
    param = list(
      name = "cartoon",
      color = isolate(input$color),
      opacity = isolate(input$opacity)
    )
  )

## End(Not run)

if (interactive()) {
library(shiny)

ui = fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      selectInput("color", "Color", c("red", "white", "blue")),
      sliderInput("opacity", "Opacity", 0, 1, 1),
      actionButton("update", "Update"),
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server = function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
                        param = list(name = "cartoon", color="red"))
  })
observeEvent(input$update, {
  NGLVieweR_proxy("structure") %>%
    updateRepresentation("cartoon",
      param = list(
        color = isolate(input$color),
        opacity = isolate(input$opacity)
      )
    )
})
 }
shinyApp(ui, server)
}
```

## updateRock *Update Rock*

### Description

Start rock animation and stop spinning. Works on an existing NGLVieweR object in Shinymode.

### Usage

```
updateRock(NGLVieweR_proxy, rock = TRUE)
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `rock` | If `TRUE` (default), start rocking and stop spinning. |

### Value

API call containing `NGLVieweR id` and list of message parameters.

### See Also

- `setRock()`
- `NGLVieweR_example()` See example "updateAnimation".

Other animations: `setRock()`, `setSpin()`, `updateSpin()`, `updateZoomMove()`

### Examples

```
## Not run:
NGLVieweR_proxy("structure") %>% updateRock(TRUE)

## End(Not run)

if (interactive()) {
library(shiny)

ui = fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      radioButtons("animate", label = "Animation",
      choices = c("None", "Spin", "Rock"), selected = "None")
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server = function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
      param = list(name = "cartoon", color="red"))
  })

  observeEvent(input$animate,{
    if(input$animate == "Rock"){
      NGLVieweR_proxy("structure") %>%
      updateRock(TRUE)
    } else if(input$animate == "Spin") {
      NGLVieweR_proxy("structure") %>%
      updateSpin(TRUE)
    } else{
      NGLVieweR_proxy("structure") %>%
      updateRock(FALSE) %>%
      updateSpin(FALSE)
    }
  })
 }
shinyApp(ui, server)
}
```

## updateSelection *Update a selection*

### Description

Update the selected residues of an existing NGLVieweR selection in

### Usage

```
updateSelection(NGLVieweR_proxy, name = name, sele = "none")
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `name` | Name of selection. |
| `sele` | Selected atoms/residues. See the section "selection-language" in the official NGL.js manual. |

## Value

API call containing `NGLVieweR` `id` and list of message parameters.

## See Also

- `NGLVieweR_example()` See example "updateSelection".

Other selections: `addSelection()`, `removeSelection()`

## Examples

```
## Not run:
NGLVieweR_proxy("structure") %>%
  updateSelection("ball+stick", sele = "1-20")

## End(Not run)

if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    titlePanel("Viewer with API inputs"),
    sidebarLayout(
      sidebarPanel(
        textInput("selection", "Selection", "1-20"),
        actionButton("update", "Update")
      ),
      mainPanel(
        NGLVieweROutput("structure")
      )
    )
  )
  server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        addRepresentation("cartoon",
          param = list(name = "cartoon", color = "red")
        ) %>%
        addRepresentation("ball+stick",
          param = list(
            name = "ball+stick",
            colorValue = "yellow",
            colorScheme = "element",
            sele = "1-20"
          )
        )
    })
    observeEvent(input$update, {
      NGLVieweR_proxy("structure") %>%
        updateSelection("ball+stick", sele = isolate(input$selection))
    })
  }
  shinyApp(ui, server)
}
```

## updateSpin *Update Spin*

### Description

Start spin animation and stop rocking. Works on an existing NGLVieweR object in Shinymode.

### Usage

```
updateSpin(NGLVieweR_proxy, spin = TRUE)
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `spin` | If `TRUE` (default), start spinning and stop rocking. |

### Value

API call containing `NGLVieweR` `id` and list of message parameters.

### See Also

- `setSpin()`
- `NGLVieweR_example()` See example "updateAnimation".

Other animations: `setRock()`, `setSpin()`, `updateRock()`, `updateZoomMove()`

### Examples

```
## Not run:
NGLVieweR_proxy("structure") %>% updateRock(TRUE)

## End(Not run)
if (interactive()) {
library(shiny)

ui = fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      radioButtons("animate", label = "Animation",
                   choices = c("None", "Spin", "Rock"), selected = "None")
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server = function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
      param = list(name = "cartoon", color="red"))
  })

  observeEvent(input$animate,{
    if(input$animate == "Rock"){
      NGLVieweR_proxy("structure") %>%
       updateRock(TRUE)
    } else if(input$animate == "Spin") {
      NGLVieweR_proxy("structure") %>%
       updateSpin(TRUE)
    } else{
      NGLVieweR_proxy("structure") %>%
       updateRock(FALSE) %>%
       updateSpin(FALSE)
    }
  })
 }
shinyApp(ui, server)
}
```

## updateStage *Update Stage*

### Description

Update an existing NGLVieweR stage in Shinymode.

### Usage

```
updateStage(NGLVieweR_proxy, param = list())
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `param` | Of type list. Most common options are `backgroundColor`, `rotateSpeed`, `zoomSpeed`, `hoverTimeout` and `lightIntensity`. For a full list of options, see the "StageParameters" method in the official NGL.js manual. |

### Value

API call containing `NGLVieweR` `id` and list of message parameters.

## See Also

- stageParameters()
- NGLVieweR_example() See example "updateStage".

Other updates: updateColor(), updateRepresentation(), updateVisibility()

## Examples

```
## Not run:
NGLVieweR("7CID") %>%
 addRepresentation("cartoon",
                   param = list(name = "cartoon", color="red")) %>%
 stageParameters(backgroundColor = "black")

## End(Not run)

if (interactive()) {
library(shiny)

ui = fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      selectInput("background", "Background", c("black", "white", "blue")),
      actionButton("update", "Update"),
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server <- function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
        param = list(name = "cartoon", color = "red")
      ) %>%
      stageParameters(backgroundColor = "black")
  })
  observeEvent(input$update, {
    NGLVieweR_proxy("structure") %>%
      updateStage(
      param = list("backgroundColor" = isolate(input$background)))
  })
}
shinyApp(ui, server)
}
```

## updateVisibility *Update visibility*

### Description

Hide or show an existing NGLVieweR selection in Shinymode.

### Usage

```
updateVisibility(NGLVieweR_proxy, name, value = FALSE)
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `name` | Name of selection to alter the color. |
| `value` | Hide `FALSE` or show `TRUE` selection. For a full description see "setVisibility" in the official NGL.js manual. |

### Value

API call containing `NGLVieweR` `id` and list of message parameters.

### See Also

`NGLVieweR_example()` See example "updateVisibility".

Other updates: `updateColor()`, `updateRepresentation()`, `updateStage()`

### Examples

```
## Not run:
NGLVieweR_proxy("structure") %>%
 updateVisibility("cartoon", value = TRUE)

## End(Not run)

if (interactive()) {
library(shiny)

ui = fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      actionButton("show", "Show"),
      actionButton("hide", "Hide"),
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server = function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
                        param = list(name = "cartoon", color="residueindex"))
  })
  observeEvent(input$show, {
    NGLVieweR_proxy("structure") %>%
      updateVisibility("cartoon", value = TRUE)

  })
  observeEvent(input$hide, {
    NGLVieweR_proxy("structure") %>%
      updateVisibility("cartoon", value = FALSE)

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

## updateZoomMove *Update zoomMove*

### Description

Add a zoom animation on an existing NGLVieweR object.

### Usage

```
updateZoomMove(
  NGLVieweR_proxy,
  center,
  zoom,
  duration = 0,
  z_offSet = 0,
  structureIndex = NULL
)
```

### Arguments

| | |
|---|---|
| `NGLVieweR_proxy` | A NGLVieweR object. |
| `center` | Target distance of selected atoms/residues. See the section "selection-language" in the official NGL.js manual. |
| `zoom` | Target zoom of selected atoms/residues. See the section "selection-language" in the official NGL.js manual. |

| | |
|---|---|
| `duration` | Optional animation time in milliseconds (default = 0). |
| `z_offSet` | Optional zoom offset value (default = 0). |
| `structureIndex` | Optional index of the structure to target for the zoom animation. If `NULL` (default), the first structure (index 0) is targeted. |

## Value

API call containing `NGLVieweR id` and list of message parameters.

## See Also

- `zoomMove()`
- `NGLVieweR_example()` See example "updatezoomMove".

Other animations: `setRock()`, `setSpin()`, `updateRock()`, `updateSpin()`

## Examples

```
## Not run:
NGLVieweR_proxy("structure") %>% updateZoomMove(center = "200",
                                               zoom = "200",
                                               z_offSet = 80,
                                               duration = 2000)

## End(Not run)

if (interactive()) {
library(shiny)

ui = fluidPage(
  titlePanel("Viewer with API inputs"),
  sidebarLayout(
    sidebarPanel(
      textInput("center", "Center", "200"),
      textInput("zoom", "Zoom", "200"),
      numericInput("zoomOffset", "Zoom offset", 80,0,100),
      numericInput("duration", "Duration", 2000,0,2000),
      actionButton("zoom", "Zoom"),
      actionButton("reset", "Reset")
    ),
    mainPanel(
      NGLVieweROutput("structure")
    )
  )
)
server = function(input, output) {
  output$structure <- renderNGLVieweR({
    NGLVieweR("7CID") %>%
      addRepresentation("cartoon",
      param = list(name = "cartoon", color="red")) %>%
      addRepresentation("ball+stick",
      param = list(name = "ball+stick", sele="200"))
  })

observeEvent(input$zoom, {
  NGLVieweR_proxy("structure") %>%
    updateZoomMove(
      center = isolate(input$center),
      zoom = isolate(input$zoom),
      z_offSet = isolate(input$zoomOffset),
      duration = isolate(input$duration)
    )
})

observeEvent(input$reset, {
  NGLVieweR_proxy("structure") %>%
    updateZoomMove(
      center = "*",
      zoom = "*",
      z_offSet = 0,
      duration = 1000
    )
})
}
shinyApp(ui, server)
}
```

---

## `zoomMove` *Set zoomMove*

---

### Description

Add a zoom animation

### Usage

```
zoomMove(NGLVieweR, center, zoom, duration = 0, z_offSet = 0)
```

### Arguments

| | |
|---|---|
| `NGLVieweR` | A NGLVieweR object. |
| `center` | Target distance of selected atoms/residues. See the section "selection-language" in the official NGL.js manual. |
| `zoom` | Target zoom of selected atoms/residues. See the section "selection-language" in the official NGL.js manual. |
| `duration` | Optional animation time in milliseconds (default = 0). |
| `z_offSet` | Optional zoom offset value (default = 0). |

### Value

List of zoomMove parameters to `NGLVieweR` `htmlwidgets` object.

## See Also

- setScale()
- setRotation()
- setPosition()

Other transformations: setPosition(), setRotation(), setScale()

## Examples

```
NGLVieweR("7CID") %>%
stageParameters(backgroundColor = "white") %>%
  addRepresentation("cartoon", param=list(name="cartoon", colorValue="red")) %>%
  addRepresentation("ball+stick", param=list(name="ball+stick",
                                             colorValue="yellow",
                                             colorScheme="element",
                                             sele="200")) %>%
  zoomMove("200:A.C", "200:A.C", 2000, -20)
```

# Index