

Package: NGLVieweR (via r-universe)

October 19, 2026

Title Interactive 3D Visualization of Molecular Structures

Version 1.5.0

Copyright See the file inst/COPYRIGHTS for the copyright holders and licenses of the bundled NGL.js library and the code it includes.

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

Description Provides an 'htmlwidgets' <<https://www.htmlwidgets.org/>> interface to 'NGL.js' <<https://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

Roxygen list(markdown = TRUE)

Imports grDevices, htmlwidgets, magrittr, tools, shiny

Suggests knitr, webshot, markdown, rmarkdown, colourpicker, testthat (>= 3.0.0), shinytest2, chromote, withr, pkgload, htmltools, callr

VignetteBuilder knitr

URL <https://nvelden.github.io/NGLVieweR/>,
<https://github.com/nvelden/NGLVieweR>

BugReports <https://github.com/nvelden/NGLVieweR/issues>

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository <https://nvelden.r-universe.dev>

Date/Publication 2026-10-19 17:56:32 UTC

RemoteUrl <https://github.com/nvelden/NGLVieweR>

RemoteRef HEAD

RemoteSha d0a70add6dfab6c69212aaf5e5931b3be11892f7

Contents

addRepresentation	2
addSelection	5
addStructure	7
colorByValue	8
NGLVieweR	9
NGLVieweR-shiny	13
NGLVieweR_example	15
removeSelection	15
selectionParameters	17
setFocus	18
setPosition	19
setQuality	20
setRock	21
setRotation	22
setScale	23
setSpin	23
setSuperpose	24
snapshot	25
stageParameters	27
updateColor	28
updateFocus	30
updateFullscreen	31
updateRepresentation	33
updateRock	34
updateSelection	36
updateSpin	37
updateStage	39
updateVisibility	40
updateZoomMove	41
zoomMove	44
Index	46

addRepresentation	<i>Add representation</i>
-------------------	---------------------------

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.

Details

type and param are passed to NGL.js as they are, so every NGL.js representation and parameter works, including ones not listed above. Some useful ones:

- **Other styles:** "spacefill", "licorice", "hyperball", "backbone", "trace", "tube", "rope", "rocket", "point", "axes", "helixorient", "unitcell" and "contact".
- **Measurements:** "distance" (atomPair), "angle" (atomTriple) and "dihedral" (atomQuad). Give each as a list of selections, e.g. atomPair = list(list("1.CA", "20.CA")). These are only drawn; the measured values are not returned to R or Shiny.
- **Biological assemblies and crystal packing:** the assembly parameter shows a biological unit such as "BU1", the unit cell ("UNITCELL") or a 3x3x3 block of unit cells ("SUPERCELL"). The default is the asymmetric unit ("AU"). Only assemblies defined in the file can be shown.

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"
    )
)

# Biological assembly
NGLViewer("1RUZ") %>%
  addRepresentation("cartoon", param = list(assembly = "BU1"))

# Distance measurement
NGLViewer("1CRN") %>%
  addRepresentation("cartoon") %>%
  addRepresentation("distance",
    param = list(atomPair = list(list("1.CA", "20.CA"))))
)

# 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	<i>Add a selection</i>
--------------	------------------------

Description

Add a new selection to a NGLVieweR object in Shiny mode.

Usage

```
addSelection(
  NGLVieweR_proxy,
  type,
  param = list(),
  structureIndex = NULL,
  structure = 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	Deprecated; use structure. The same, but counting from 0.
structure	(optional) Number of the structure to add the selection to, counting from 1 in the order the structures were added (see addStructure()). In Shiny, input\$<outputId>_selectionStructure gives the number of the last clicked structure. If not specified, the selection is added 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, in the same forms as in NGLViewer() : a PDB entry code (e.g. "7CID"), a file path (also compressed with gzip, bzip2 or xz), or the file contents as a single string or a vector of lines.
format	Format of the structure, as in NGLViewer() (e.g. "pdb", "cif"). Needed for file contents and when the format cannot be inferred from the file name. For a PDB entry code it picks the file downloaded from RCSB, as in NGLViewer() .

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.
```

colorByValue

Color residues by value

Description

Color residues by a numeric value computed in R, such as a conservation score, a mutation effect or a p-value. The values are mapped onto a color gradient, and the result can be used as color wherever a representation takes one: [addRepresentation](#), [addSelection](#), [updateColor](#) and [updateRepresentation](#).

Usage

```
colorByValue(
  values,
  palette = c("blue", "white", "red"),
  domain = NULL,
  na_color = "grey"
)
```

Arguments

values	Either a data frame with columns <code>resno</code> and <code>value</code> , and optionally <code>chain</code> , <code>inscode</code> (insertion code) and <code>structure</code> (the structure number, counting from 1); or a named numeric vector with names <code>"chain:resno"</code> , such as <code>"A:12"</code> , <code>"A:12B"</code> (with insertion code B) or <code>"12"</code> (any chain). Without <code>chain</code> a row applies to that residue number in every chain, and without <code>structure</code> to every structure. The data frame of the Shiny input <code>input\$<outputId>_residues</code> with an added <code>value</code> column can be used as is.
palette	Two or more colors (names or HEX codes) for the gradient, from the lowest to the highest value.
domain	The values at the two ends of the palette, <code>c(min, max)</code> . Defaults to the range of values. Values outside it get the color of the nearest end.
na_color	Color for residues with a missing value (NA) and for residues not listed in <code>values</code> .

Details

The colors are computed in R, so the viewer shows exactly the colors that `grDevices::colorRamp(palette)` gives. A row for a specific chain or structure takes precedence over a row for any chain or structure. NGL.js cannot show a color legend; the returned object contains `palette` and `domain` (as HEX colors and numbers) so a legend can be drawn next to the viewer, e.g. with a plot.

Value

A `NGLViewerR_colorByValue` object to use as color.

See Also

[addRepresentation\(\)](#), [updateColor\(\)](#), [NGLVieweR-shiny](#)

Examples

```
scores <- data.frame(chain = "A", resno = 1:46, value = sin(1:46 / 5))
NGLVieweR("1CRN") %>%
  addRepresentation("cartoon",
    param = list(color = colorByValue(scores,
                                         palette = c("blue", "white", "red"),
                                         domain = c(-1, 1))))

# A named vector, for a few residues; all others are grey
NGLVieweR("1CRN") %>%
  addRepresentation("cartoon",
    param = list(color = colorByValue(c("A:10" = 1, "A:20" = 2, "A:30" = 3))))

if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    sliderInput("k", "Score = sin(resno / k)", 1, 20, 5),
    NGLVieweROutput("structure")
  )
  server <- function(input, output) {
    output$structure <- renderNGLVieweR({
      NGLVieweR("1CRN") %>%
        addRepresentation("cartoon", param = list(name = "cartoon"))
    })
    observeEvent(input$k, {
      residues <- input$structure_residues
      req(residues)
      residues$value <- sin(residues$resno / input$k)
      NGLVieweR_proxy("structure") %>%
        updateColor("cartoon", colorByValue(residues, domain = c(-1, 1)))
    })
  }
  shinyApp(ui, server)
}
```

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

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

Arguments

data	A PDB entry code (e.g. "7CID"), the path to a structure file, or the file contents as a single string or a vector of lines (e.g. from <code>readLines()</code>). Files compressed with gzip, bzip2 or xz (.gz, .bz2, .xz) are decompressed.
format	Input format (cif, mmCIF, mcif, pdb, ent, pqr, gro, sdf, sd, mol2), with or without a leading dot. Needed for file contents and for files without a recognisable extension. Binary formats such as MMTF are not supported. For a PDB entry code, "pdb" or "cif" picks the file downloaded from RCSB; without format the PDB-format file is used, and the mmCIF file if there is none (large entries such as ribosomes only exist as mmCIF). For such entries, format = "cif" saves the failed request for the PDB-format file.
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 NGLViewerR 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
NGLViewerR("7CID") %>%
  addRepresentation("cartoon",
    param = list(name = "cartoon", colorScheme="residueindex"))

# Example 2: Advanced
NGLViewerR("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 = "ball+stick",
            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$structure <- renderNGLVieweR({
      NGLVieweR("7CID") %>%
        addRepresentation("cartoon",
          param = list(name = "cartoon", colorScheme = "residueindex")
        ) %>%
        stageParameters(background-color = "black") %>%
        setQuality("high") %>%
        setFocus(0) %>%
        setSpin(TRUE)
    })
    observeEvent(input$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(
        NGLViewROutput("structure")
      )
    )
  )
  server <- function(input, output) {
    output$structure <- renderNGLViewR({
      NGLViewR("7CID") %>%
        addRepresentation("cartoon",
          param = list(name = "cartoon", color = "residueindex")
        ) %>%
        stageParameters(backgroundcolor = "black") %>%
        setQuality("high") %>%
        setFocus(0) %>%
        setSpin(TRUE)
    })
    observeEvent(input$add, {
      NGLViewR_proxy("structure") %>%
        addSelection(isolate(input$type),
          param =
            list(
              name = "sel1",
              sele = isolate(input$selection),
              color = isolate(input$color)
            )
        )
    })
    observeEvent(input$remove, {
      NGLViewR_proxy("structure") %>%
        removeSelection("sel1")
    })
  }
  shinyApp(ui, server)
}

```

NGLVieweR_example	<i>Run NGLVieweR example Shiny app</i>
-------------------	--

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	<i>Remove a selection</i>
-----------------	---------------------------

Description

Remove an existing NGLVieweR selection in Shiny mode.

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	<i>Set selection parameters</i>
---------------------	---------------------------------

Description

Set selection parameters.

Usage

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

Arguments

NGLViewerR	A NGLViewerR 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 NGLViewerR htmlwidgets object.

Examples

```

NGLViewerR("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.

Details

The focus sets the clipping planes and the fog (the NGL.js stage parameters clipNear, clipFar, fogNear and fogFar). Without setFocus() the viewer uses the NGL.js defaults (with the default clipScale = "relative" these equal focus 0). Values of these four parameters given in [stageParameters\(\)](#) win over setFocus().

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	<i>Set Position</i>
-------------	---------------------

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".

Details

Without setQuality() the viewer uses the NGL.js default ("medium"). A quality given in [stageParameters\(\)](#) wins over setQuality().

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	<i>Set rock</i>
---------	-----------------

Description

Set rock animation

Usage

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

Arguments

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

Details

Only one animation can run at a time. If both `setRock(TRUE)` and `setSpin(TRUE)` are used, the structure spins.

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	<i>Rotate View</i>
-------------	--------------------

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	<i>Set Scale</i>
----------	------------------

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	<i>Set Spin</i>
---------	-----------------

Description

Set Spin animation

Usage

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

Arguments

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

Details

Only one animation can run at a time. If both setSpin(TRUE) and setRock(TRUE) are used, the structure spins.

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). The reference keeps its coordinates; every other structure is moved onto it.
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 Shiny mode.

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	<i>Set stage parameters</i>
-----------------	-----------------------------

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.

Details

Options that are not given use the NGL.js defaults. In Shiny, this also holds when `renderNGLVieweR()` re-renders: an option set in an earlier render but not in the current one returns to its default.

`setFocus()` and `setQuality()` are shortcuts that set clipping, fog and quality. Options named here win over them: for example `setFocus(50) %>% stageParameters(fogFar = 60)` keeps the clipping of focus 50 with fogFar 60.

All NGL.js stage parameters can be set here or changed later with `updateStage()`. For example:

- **Clipping:** clipNear, clipFar, clipDist, clipMode ("scene" or "camera") and clipScale ("relative" or "absolute").
- **Fog:** fogNear and fogFar.
- **Lighting:** lightColor, lightIntensity, ambientColor and ambientIntensity.
- **Camera:** cameraType ("perspective", "orthographic" or "stereo"), cameraFov and cameraEyeSep.
- **Mouse controls:** mousePreset ("default", "pymol", "coot" or "astexviewer").

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"))

# Orthographic camera, PyMOL mouse controls and fog
NGLViewer("7CID") %>%
  stageParameters(
    cameraType = "orthographic",
    mousePreset = "pymol",
    fogNear = 40,
    fogFar = 80
  ) %>%
  addRepresentation("cartoon")

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 Shiny mode.

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 Shiny mode.

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	<i>Fullscreen</i>
------------------	-------------------

Description

Enter or leave fullscreen. Works in Shiny mode.

Usage

```

updateFullscreen(
  NGLVieweR_proxy,
  fullscreen = TRUE,
  element = c("page", "viewer")
)

```

Arguments

NGLVieweR_proxy	A NGLVieweR object.
fullscreen	If TRUE (default) enter fullscreen, if FALSE leave it. Nothing happens if the page is already in the requested state.
element	What to show fullscreen: "page" (default) the whole page, so the app's own controls stay visible, or "viewer" only the viewer. Only used when entering fullscreen; users can always leave with Esc.

Details

Browsers only allow entering fullscreen in response to a user action, such as clicking a button that triggers `updateFullscreen()`.

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("enter", "Enter fullscreen"),
        actionButton("exit", "Exit fullscreen")
      ),
      mainPanel(
        NGLViewerOutput("structure")
      )
    )
  )
  server = function(input, output) {
    output$structure <- renderNGLViewer({
      NGLViewer("7CID") %>%
        addRepresentation("cartoon",
          param = list(name = "cartoon", color = "red")
        )
    })

    observeEvent(input$enter, {
      NGLViewer_proxy("structure") %>%
        updateFullscreen(TRUE)
    })

    observeEvent(input$exit, {
      NGLViewer_proxy("structure") %>%
        updateFullscreen(FALSE)
    })
  }
  shinyApp(ui, server)
}
```

updateRepresentation	<i>Update Representation</i>
----------------------	------------------------------

Description

Update an existing NGLVieweR representation in Shiny mode.

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 Shiny mode.

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	<i>Update a selection</i>
-----------------	---------------------------

Description

Update the selected residues of an existing NGLVieweR selection in Shiny mode.

Usage

```
updateSelection(NGLVieweR_proxy, 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 Shiny mode.

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	<i>Update Stage</i>
-------------	---------------------

Description

Update an existing NGLVieweR stage in Shiny mode.

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. Clipping, fog, lighting, camera type and mouse preset can be changed here too; see stageParameters() for those options. 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 Shiny mode.

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)
}
```



```

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).

Details

Applies to the last structure added. The camera moves after all structures are loaded and superposed (see [setSuperpose\(\)](#)). If [zoomMove\(\)](#) is used for several structures, the last one is used (with a warning). The center is that of the atom coordinates, so it does not follow [setPosition\(\)](#), [setRotation\(\)](#) or [setScale\(\)](#).

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

- * **animations**
 - setRock, 21
 - setSpin, 23
 - updateRock, 34
 - updateSpin, 37
 - updateZoomMove, 41
- * **options**
 - setFocus, 18
 - setQuality, 20
 - snapShot, 25
 - updateFocus, 30
 - updateFullscreen, 31
- * **selections**
 - addSelection, 5
 - removeSelection, 15
 - updateSelection, 36
- * **transformations**
 - setPosition, 19
 - setRotation, 22
 - setScale, 23
 - zoomMove, 44
- * **updates**
 - updateColor, 28
 - updateRepresentation, 33
 - updateStage, 39
 - updateVisibility, 40
- addRepresentation, 2, 8
- addRepresentation(), 9, 33
- addSelection, 5, 8
- addSelection(), 3, 16, 33, 36
- addStructure, 7
- addStructure(), 5, 42
- colorByValue, 8
- NGLViewer, 9
- NGLViewer(), 7
- NGLViewer-shiny, 9, 13
- NGLViewer_example, 15
- NGLViewer_example(), 3, 5, 10, 13, 16, 26, 28–30, 32, 33, 35, 36, 38, 39, 41, 42
- NGLViewer_proxy (NGLViewer-shiny), 13
- NGLViewer_proxy(), 10
- NGLViewerOutput (NGLViewer-shiny), 13
- removeSelection, 15
- removeSelection(), 5, 36
- renderNGLViewer (NGLViewer-shiny), 13
- selectionParameters, 17
- setFocus, 18
- setFocus(), 20, 26, 27, 30, 32
- setPosition, 19
- setPosition(), 22, 23, 44
- setQuality, 20
- setQuality(), 19, 26, 27, 30, 32
- setRock, 21
- setRock(), 24, 35, 38, 42
- setRotation, 22
- setRotation(), 19, 23, 44
- setScale, 23
- setScale(), 19, 22, 44
- setSpin, 23
- setSpin(), 21, 35, 38, 42
- setSuperpose, 24
- setSuperpose(), 44
- snapShot, 25
- snapShot(), 19, 20, 30, 32
- stageParameters, 27
- stageParameters(), 18, 20, 39
- updateColor, 8, 28
- updateColor(), 9, 33, 39, 41
- updateFocus, 30
- updateFocus(), 19, 20, 26, 32
- updateFullscreen, 31
- updateFullscreen(), 19, 20, 26, 30
- updateRepresentation, 8, 33
- updateRepresentation(), 5, 29, 39, 41

updateRock, [34](#)
updateRock(), [21](#), [24](#), [38](#), [42](#)
updateSelection, [36](#)
updateSelection(), [5](#), [16](#)
updateSpin, [37](#)
updateSpin(), [21](#), [24](#), [35](#), [42](#)
updateStage, [39](#)
updateStage(), [27–29](#), [33](#), [41](#)
updateVisibility, [40](#)
updateVisibility(), [29](#), [33](#), [39](#)
updateZoomMove, [41](#)
updateZoomMove(), [21](#), [24](#), [35](#), [38](#)

zoomMove, [44](#)
zoomMove(), [19](#), [22](#), [23](#), [42](#)