

# Package ‘seleniumPipes’

October 14, 2022

**Type** Package

**Title** R Client Implementing the W3C WebDriver Specification

**Version** 0.3.7

**Description** The W3C WebDriver specification defines a way for out-of-process programs to remotely instruct the behaviour of web browsers. It is detailed at <https://w3c.github.io/webdriver/webdriver-spec.html>. This package provides an R client implementing the W3C specification.

**Depends** R (>= 3.0.0),httr,jsonlite,xml2,magrittr,whisker

**Encoding** UTF-8

**License** GPL-3

**Suggests** knitr, rmarkdown, data.table, testthat, covr, RSauceLabs

**RoxygenNote** 5.0.1

**URL** <https://github.com/johndharrison/seleniumPipes>

**URLNote** <https://github.com/johndharrison/seleniumPipes>

**BugReports** <https://github.com/johndharrison/seleniumPipes/issues>

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** John Harrison [aut, cre]

**Maintainer** John Harrison <johndharrison@gmail.com>

**Repository** CRAN

**Date/Publication** 2016-10-01 00:36:25

## R topics documented:

acceptAlert	3
acceptAlertOld	4
addCookie	5
back	6
checkResponse	7

closeWindow . . . . .	8
deleteAllCookies . . . . .	10
deleteCookie . . . . .	11
deleteSession . . . . .	12
dismissAlert . . . . .	13
dismissAlertOld . . . . .	14
elementClear . . . . .	15
elementClick . . . . .	16
elementSendKeys . . . . .	17
errorContent . . . . .	18
errorResponse . . . . .	19
executeAsyncScript . . . . .	19
executeAsyncScriptOld . . . . .	21
executeScript . . . . .	22
executeScriptOld . . . . .	24
findElement . . . . .	25
findElementFromElement . . . . .	27
findElements . . . . .	28
findElementsFromElement . . . . .	30
forward . . . . .	32
fullscreenWindow . . . . .	33
getActiveElement . . . . .	34
getAlertText . . . . .	36
getAlertTextOld . . . . .	37
getAllCookies . . . . .	38
getCurrentUrl . . . . .	39
getElementAttribute . . . . .	40
getElementCssValue . . . . .	41
getElementProperty . . . . .	42
getElementRect . . . . .	44
getElementTagName . . . . .	45
getElementText . . . . .	46
getNamedCookie . . . . .	47
getPageSource . . . . .	49
getTitle . . . . .	50
getWindowHandle . . . . .	52
getWindowHandleOld . . . . .	53
getWindowHandles . . . . .	54
getWindowHandlesOld . . . . .	55
getWindowPosition . . . . .	56
getWindowPositionOld . . . . .	57
getWindowSize . . . . .	58
getWindowSizeOld . . . . .	60
go . . . . .	61
isElementEnabled . . . . .	62
isElementSelected . . . . .	63
maximizeWindow . . . . .	64
maximizeWindowOld . . . . .	66

newSession . . . . .	67
performActions . . . . .	68
queryDriver . . . . .	68
refresh . . . . .	69
releasingActions . . . . .	70
remoteDr . . . . .	71
retry . . . . .	72
seleniumPipes . . . . .	73
sendKeys . . . . .	74
sendAlertText . . . . .	74
sendAlertTextOld . . . . .	76
setTimeout . . . . .	77
setWindowPosition . . . . .	78
setWindowPositionOld . . . . .	79
setWindowSize . . . . .	80
setWindowSizeOld . . . . .	82
switchToFrame . . . . .	83
switchToParentFrame . . . . .	84
switchToWindow . . . . .	86
takeElementScreenshot . . . . .	87
takeScreenshot . . . . .	88
wbElement . . . . .	89

## **Index** **91**

---

acceptAlert	<i>Accept alert</i>
-------------	---------------------

---

### **Description**

acceptAlert accept a JavaScript alert

### **Usage**

```
acceptAlert(remDr, ...)
```

### **Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### **Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other userPrompts functions: [dismissAlert](#), [getAlertText](#), [sendAlertText](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
sScript <- "setTimeout(function(){alert('HELLO')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% dismissAlert()

sScript <- "setTimeout(function(){confirm('Press a button')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% acceptAlert()

sScript <- "setTimeout(function(){confirm('Press a button')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% getAlertText()
remDr %>% dismissAlert()

sScript <- "setTimeout(function(){prompt('Please enter your name', '')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% getAlertText()
remDr %>% sendAlertText("Buck Rogers?")

remDr %>% deleteSession()

## End(Not run)
```

---

acceptAlertOld

*Accept alert*

---

**Description**

acceptAlertOld accept a JavaScript alert This uses the old JSONwireprotocol endpoints.

**Usage**

```
acceptAlertOld(remDr, ...)
```

**Arguments**

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).

...              Additional function arguments - Currently passes the [retry](#) argument.

**Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other oldMethods functions: [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

**Examples**

```
## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)
```

---

addCookie	<i>Add a specific cookie.</i>
-----------	-------------------------------

---

**Description**

addCookie Add a specific cookie.

**Usage**

```
addCookie(remDr, name, value, path = NULL, domain = NULL, secure = FALSE,
  httpOnly = NULL, expiry = NULL, ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
name	character: The name of the cookie; may not be null or an empty string
value	character: The cookie value; may not be null.
path	character: The path the cookie is visible to. If left blank or set to null, will be set to "/".
domain	character: The domain the cookie is visible to. It should be null or the same as the domain of the current URL.
secure	logical: Whether this cookie requires a secure connection(https?). It should be null or equal to the security of the current URL.
httpOnly	logical: Whether the cookie is an httpOnly cookie.
expiry	The cookie's expiration date; may be null.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other cookies functions: [deleteAllCookies](#), [deleteCookie](#), [getAllCookies](#), [getNamedCookie](#)

**Examples**

```
## Not run:
# assume a server is running at default location
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
# get the cookies
remDr %>% getCookie()
# get a named cookie
remDr %>% getCookie("NID")
# add our own cookie
remDr %>% addCookie(name = "myCookie", value = "12")
# check its value
remDr %>% getCookie("myCookie")
# delete our cookie
remDr %>% deleteCookie("myCookie")
# check its deleted
remDr %>% getCookie("myCookie")

# delete all cookies
remDr %>% getCookie()
remDr %>% deleteAllCookies() %>%
  getCookie()

remDr %>% deleteSession()

## End(Not run)
```

---

back

*Navigate backwards*

---

**Description**

back Navigate backwards in the browser history, if possible.

**Usage**

```
back(remDr, ...)
```

## Arguments

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).  
...              Additional function arguments - Currently passes the [retry](#) argument.

## Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

## See Also

Other navigation functions: [forward](#), [getCurrentUrl](#), [getTitle](#), [go](#), [refresh](#)

## Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# get the title
remDr %>% getTitle

# get the current page url
remDr %>% getCurrentUrl

# navigate
remDr %>% go("http://www.bbc.co.uk")

# go back
remDr %>% (seleniumPipes::back)

# go forward
remDr %>% forward

# refresh page
remDr %>% refresh

# close browser
remDr %>% deleteSession

## End(Not run)
```

**Description**

checkResponse checks the response from a remote web driver and checks against known errors.

**Usage**

```
checkResponse(response)
```

**Arguments**

response            The value returned by a http method from httr see [VERB](#)

**Value**

Stops with appropriate error if any found. On error [errorResponse](#) and [errorContent](#) may provide additional detail.

**Examples**

```
## Not run:
# internal method

## End(Not run)
```

---

closeWindow	<i>Close the current window.</i>
-------------	----------------------------------

---

**Description**

closeWindow Close the current window.

**Usage**

```
closeWindow(remDr, ...)
```

**Arguments**

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).  
 ...            Additional function arguments - Currently passes the [retry](#) argument.

**Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.



## See Also

Other commandContexts functions: [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)

## Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)
```

---

deleteAllCookies      *Delete all the cookies.*

---

### Description

deleteAllCookies Delete all the cookies that are currently visible.

### Usage

```
deleteAllCookies(remDr, ...)
```

### Arguments

remDr                  An object of class "rDriver". A remote driver object see [remoteDr](#).  
 ...                    Additional function arguments - Currently passes the [retry](#) argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other cookies functions: [addCookie](#), [deleteCookie](#), [getAllCookies](#), [getNamedCookie](#)

### Examples

```
## Not run:
# assume a server is running at default location
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
# get the cookies
remDr %>% getCookie()
# get a named cookie
remDr %>% getCookie("NID")
# add our own cookie
remDr %>% addCookie(name = "myCookie", value = "12")
# check its value
remDr %>% getCookie("myCookie")
# delete our cookie
remDr %>% deleteCookie("myCookie")
# check its deleted
remDr %>% getCookie("myCookie")

# delete all cookies
remDr %>% getCookie()
remDr %>% deleteAllCookies() %>%
```

```

    getCookie()

remDr %>% deleteSession()

## End(Not run)

```

---

deleteCookie	<i>Delete a given cookie.</i>
--------------	-------------------------------

---

### Description

deleteCookie Delete the cookie with the give name.

### Usage

```
deleteCookie(remDr, name = NULL, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
name	character: The name of the cookie; may not be null or an empty string
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other cookies functions: [addCookie](#), [deleteAllCookies](#), [getAllCookies](#), [getNamedCookie](#)

### Examples

```

## Not run:
# assume a server is running at default location
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
# get the cookies
remDr %>% getCookie()
# get a named cookie
remDr %>% getCookie("NID")
# add our own cookie
remDr %>% addCookie(name = "myCookie", value = "12")
# check its value
remDr %>% getCookie("myCookie")
# delete our cookie

```

```

remDr %>% deleteCookie("myCookie")
# check its deleted
remDr %>% getCookie("myCookie")

# delete all cookies
remDr %>% getCookie()
remDr %>% deleteAllCookies() %>%
  getCookie()

remDr %>% deleteSession()

## End(Not run)

```

---

deleteSession	<i>Delete the session.</i>
---------------	----------------------------

---

### Description

deleteSession Delete the session.

### Usage

```
deleteSession(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

Returns an object of class "rDriver". The sessionId associated with this driver has been removed however and the associated browser should be closed by the server.

### See Also

Other sessions functions: [newSession](#), [setTimeout](#)

### Examples

```

## Not run:
# start a driver without opening a browser
remDr <- remoteDr(newSession = FALSE)

# open a browser
remDr %>% newSession

# set timeout on waiting for elements
remDr %>% setTimeout(type = "implicit", 5000)

```

```
# close Session
remDr %>% deleteSession

## End(Not run)
```

---

dismissAlert

*Dismiss Alert*


---

## Description

dismissAlert dismiss a JavaScript alert

## Usage

```
dismissAlert(remDr, ...)
```

## Arguments

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).  
 ...              Additional function arguments - Currently passes the [retry](#) argument.

## Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

## See Also

Other userPrompts functions: [acceptAlert](#), [getAlertText](#), [sendAlertText](#)

## Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
sScript <- "setTimeout(function(){alert('HELLO')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% dismissAlert()

sScript <- "setTimeout(function(){confirm('Press a button')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% acceptAlert()

sScript <- "setTimeout(function(){confirm('Press a button')},1000); return 'DONE';"
```

```

remDr %>% executeScript(sScript, args = list())
remDr %>% getAlertText()
remDr %>% dismissAlert()

sScript <- "setTimeout(function(){prompt('Please enter your name', '')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% getAlertText()
remDr %>% sendAlertText("Buck Rogers?")

remDr %>% deleteSession()

## End(Not run)

```

---

dismissAlertOld	<i>Dismiss Alert</i>
-----------------	----------------------

---

## Description

dismissAlertOld dismiss a JavaScript alert. This uses the old JSONwireprotocol endpoints.

## Usage

```
dismissAlertOld(remDr, ...)
```

## Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

## Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

## See Also

Other oldMethods functions: [acceptAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

## Examples

```

## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)

```

---

elementClear	<i>Clear an elements text value.</i>
--------------	--------------------------------------

---

### Description

elementClear Clear a TEXTAREA or text INPUT element's value.

### Usage

```
elementClear(webElem, ...)
```

### Arguments

webElem	An object of class "wElement". A web Element object see <a href="#">wbElement</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(webElem): An object of class "wElement" is invisibly returned. A webElement object see [wbElement](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other elementInteraction functions: [elementClick](#), [elementSendKeys](#)

### Examples

```
## Not run:
# start a browser
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

queryBox <- remDr %>% findElement("name", "q")

# send text to the query box
queryBox %>% elementSendKeys("Some ", "text")

# clear the query box
queryBox %>% elementClear

# get the search button
searchBtn <- remDr %>% findElement("name", "btnG")
# send text to query box
queryBox %>% elementSendKeys("R project")

# click the search button
searchBtn %>% elementClick
```

```
# close browser
remDr %>% deleteSession

## End(Not run)
```

---

elementClick                      *Click on an element.*

---

### Description

elementClick The elementClick function scrolls into view the element and clicks the in-view centre point. If the element is not pointer-interactable, an element not interactable error is returned.

### Usage

```
elementClick(webElem, ...)
```

### Arguments

webElem                      An object of class "wElement". A web Element object see [wbElement](#).  
 ...                            Additional function arguments - Currently passes the [retry](#) argument.

### Value

invisible(webElem): An object of class "wElement" is invisibly returned. A webElement object see [wbElement](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other elementInteraction functions: [elementClear](#), [elementSendKeys](#)

### Examples

```
## Not run:
# start a browser
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

queryBox <- remDr %>% findElement("name", "q")

# send text to the query box
queryBox %>% elementSendKeys("Some ", "text")

# clear the query box
queryBox %>% elementClear

# get the search button
searchBtn <- remDr %>% findElement("name", "btnG")
```



```

# send text to query box
queryBox %>% elementSendKeys("R project")

# click the search button
searchBtn %>% elementClick

# close browser
remDr %>% deleteSession

## End(Not run)

```

---

elementSendKeys      *Send a sequence of key strokes to an element.*

---

### Description

elementSendKeys The elementSendKeys function scrolls into view the form control element and then sends the provided keys to the element. In case the element is not keyboard interactable, an element not interactable error is returned.

### Usage

```
elementSendKeys(webElem, ...)
```

### Arguments

webElem      An object of class "wElement". A web Element object see [wbElement](#).

...      keys to send the element. seleniumPipes includes mappings to unicode keys see [selKeys](#). To use one of this name the string using key. See examples.

### Value

invisible(webElem): An object of class "wElement" is invisibly returned. A webElement object see [wbElement](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other elementInteraction functions: [elementClear](#), [elementClick](#)

### Examples

```

## Not run:
# start a browser
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

queryBox <- remDr %>% findElement("name", "q")

```

```
# send text to the query box
queryBox %>% elementSendKeys("Some ", "text")

# clear the query box
queryBox %>% elementClear

# get the search button
searchBtn <- remDr %>% findElement("name", "btnG")
# send text to query box
queryBox %>% elementSendKeys("R project")

# click the search button
searchBtn %>% elementClick

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

errorContent

*Returns the content from remote webdriver*

---

## Description

errorContent returns the content from the remote webdriver on an error.

## Usage

```
errorContent()
```

## Value

returns content see [content](#)

## Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% findElement("name", "sdofnsdofk")
errorContent()

## End(Not run)
```

---

errorResponse	<i>Return the response from remote webdriver</i>
---------------	--

---

### Description

errorResponse returns the response from the remote webdriver on an error.

### Usage

```
errorResponse()
```

### Value

returns response see [VERB](#). Headers, request etc. can be examined from this object.

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% findElement("name", "sdfnsdofk")
errorResponse()

## End(Not run)
```

---

executeAsyncScript	<i>Execute JavaScript asynchronously on browser.</i>
--------------------	--

---

### Description

executeAsyncScript Inject a snippet of JavaScript into the page for asynchronous execution in the context of the currently selected frame.

### Usage

```
executeAsyncScript(remDr, script, args = list(), replace = TRUE, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
script	character: The script to inject.
args	The arguments of the script as a list.
replace	logical: If TRUE any elements identify as web Elements are converted to such.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Details**

The driver will pass a callback as the last argument to the snippet, and block until the callback is invoked.

**Value**

If `replace` is `TRUE` parses the response from the server for web Elements and converts as such. Otherwise the content returned is assumed to be a simple list.

**See Also**

Other documentHandling functions: [executeScript](#), [getPageSource](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
# Get the page source
remDr %>% go("https://www.google.com/ncr") %>%
  getPageSource

remDr %>% getTitle()
webElem <- remDr %>% findElement("css", "img#hplogo")
# check if the logo is hidden
remDr %>% executeScript("return document.getElementById('hplogo').hidden;", args = list())
# [1] FALSE
# hide the logo
remDr %>% executeScript("document.getElementById('hplogo').hidden = true;", args = list())
# Make the logo visible this time passing a web Element as an argument
remDr %>% executeScript(script = "return arguments[0].hidden = false;", args = list(webElem))

# Pass arguments
remDr %>% executeScript(script = "return argument[1] + argument[2];"
  , args = list(1, 2))

# Return a web Element
remDr %>% executeScript(script = "return document.getElementById('hplogo');", args = list())
# ElementId: 0
# Remote Driver:
# Remote Ip Address: http://localhost:4444/wd/hub
# Remote sessionId: 9a83672a-d72b-4873-aa7d-96f7f1f80fa0

# Return a web Element in a more complex object
script <- "var test = {num:1, str:'a', el:document.getElementById('hplogo')};return test;"
remDr %>% executeScript(script = script
  , args = list())

# $str
# [1] "a"
#
# $num
# [1] 1
#
```

```

# $el
# ElementId: 0
# Remote Driver:
#   Remote Ip Address: http://localhost:4444/wd/hub
#   Remote sessionId: 9a83672a-d72b-4873-aa7d-96f7f1f80fa0

# Run with replace = FALSE
remDr %>% executeScript(script = script
                        , args = list(), replace = FALSE)

# $str
# [1] "a"
#
# $num
# [1] 1
#
# $el
# $el$ELEMENT
# [1] "0"

remDr %>% setTimeout("script")

asScript <- "cb = arguments[0];setTimeout(function(){cb('DONE');},5000); "
system.time(test1 <- remDr %>% executeAsyncScript(asScript, args = list()))
sScript <- "setTimeout(function(){},5000); return 'DONE';"
system.time(test2 <- remDr %>% executeScript(sScript, args = list()))

remDr %>% deleteSession()

## End(Not run)

```

---

executeAsyncScriptOld *executeAsyncScriptOld*

---

## Description

`executeAsyncScriptOld` This function uses the old JSONwireprotocol end points. Inject a snippet of JavaScript into the page for asynchronous execution in the context of the currently selected frame.

## Usage

```
executeAsyncScriptOld(remDr, script, args = list(), replace = TRUE, ...)
```

## Arguments

<code>remDr</code>	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
<code>script</code>	character: The script to inject.
<code>args</code>	The arguments of the script as a list.

replace            logical: If TRUE any elements identify as web Elements are converted to such.  
 ...                Additional function arguments - Currently passes the [retry](#) argument.

### Details

The driver will pass a callback as the last argument to the snippet, and block until the callback is invoked.

### Value

If replace is TRUE parses the response from the server for web Elements and converts as such. Otherwise the content returned is assumed to be a simple list.

### See Also

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

### Examples

```
## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)
```

---

executeScript	<i>Execute JavaScript on browser.</i>
---------------	---------------------------------------

---

### Description

executeScript Inject a snippet of JavaScript into the page for execution in the context of the currently selected frame. The executed script is assumed to be synchronous and the result of evaluating the script will be returned.

### Usage

```
executeScript(remDr, script, args = list(), replace = TRUE, ...)
```

### Arguments

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).  
 script           character: The script to inject.  
 args             The arguments of the script as a list.  
 replace         logical: If TRUE any elements identify as web Elements are converted to such.  
 ...              Additional function arguments - Currently passes the [retry](#) argument.

**Value**

If `replace` is `TRUE` parses the response from the server for web Elements and converts as such. Otherwise the content returned is assumed to be a simple list.

**See Also**

Other documentHandling functions: [executeAsyncScript](#), [getPageSource](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
# Get the page source
remDr %>% go("https://www.google.com/ncr") %>%
  getPageSource

remDr %>% getTitle()
webElem <- remDr %>% findElement("css", "img#hplogo")
# check if the logo is hidden
remDr %>% executeScript("return document.getElementById('hplogo').hidden;", args = list())
# [1] FALSE
# hide the logo
remDr %>% executeScript("document.getElementById('hplogo').hidden = true;", args = list())
# Make the logo visible this time passing a web Element as an argument
remDr %>% executeScript(script = "return arguments[0].hidden = false;", args = list(webElem))

# Pass arguments
remDr %>% executeScript(script = "return argument[1] + argument[2];"
  , args = list(1, 2))

# Return a web Element
remDr %>% executeScript(script = "return document.getElementById('hplogo');", args = list())
# ElementId: 0
# Remote Driver:
# Remote Ip Address: http://localhost:4444/wd/hub
# Remote sessionId: 9a83672a-d72b-4873-aa7d-96f7f1f80fa0

# Return a web Element in a more complex object
script <- "var test = {num:1, str:'a', el:document.getElementById('hplogo')};return test;"
remDr %>% executeScript(script = script
  , args = list())

# $str
# [1] "a"
#
# $num
# [1] 1
#
# $el
# ElementId: 0
# Remote Driver:
# Remote Ip Address: http://localhost:4444/wd/hub
# Remote sessionId: 9a83672a-d72b-4873-aa7d-96f7f1f80fa0
```

```

# Run with replace = FALSE
remDr %>% executeScript(script = script
                        , args = list(), replace = FALSE)

# $str
# [1] "a"
#
# $num
# [1] 1
#
# $el
# $el$ELEMENT
# [1] "0"

remDr %>% setTimeout("script")

asScript <- "cb = arguments[0];setTimeout(function(){cb('DONE');},5000); "
system.time(test1 <- remDr %>% executeAsyncScript(asScript, args = list()))
sScript <- "setTimeout(function(){},5000); return 'DONE';"
system.time(test2 <- remDr %>% executeScript(sScript, args = list()))

remDr %>% deleteSession()

## End(Not run)

```

---

executeScriptOld

*Execute JavaScript asynchronously on browser.*

---

## Description

`executeScriptOld` This function uses the old JSONwireprotocol end points. Inject a snippet of JavaScript into the page for execution in the context of the currently selected frame. The executed script is assumed to be synchronous and the result of evaluating the script will be returned.

## Usage

```
executeScriptOld(remDr, script, args = list(), replace = TRUE, ...)
```

## Arguments

<code>remDr</code>	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
<code>script</code>	character: The script to inject.
<code>args</code>	The arguments of the script as a list.
<code>replace</code>	logical: If TRUE any elements identify as web Elements are converted to such.
<code>...</code>	Additional function arguments - Currently passes the <a href="#">retry</a> argument.



**Value**

If replace is TRUE parses the response from the server for web Elements and converts as such. Otherwise the content returned is assumed to be a simple list.

**See Also**

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

**Examples**

```
## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)
```

---

findElement	<i>Search for an element on the page</i>
-------------	--

---

**Description**

findElement Search for an element on the page, starting from the document root. The located element will be returned as an object of "wElement" class

**Usage**

```
findElement(remDr, using = c("xpath", "css selector", "id", "name",
  "tag name", "class name", "link text", "partial link text"), value, ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
using	Locator scheme to use to search the element, available schemes: "class name", "css selector", "id", "name", "link text", "partial link text", "tag name", "xpath". Defaults to 'xpath'. Partial string matching is accepted.
value	The search target. See examples.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Details**

Details of possible locator schemes

**"class name"** : Returns an element whose class name contains the search value; compound class names are not permitted.

**"css selector"** : Returns an element matching a CSS selector.

- "id"** : Returns an element whose ID attribute matches the search value.
- "name"** : Returns an element whose NAME attribute matches the search value.
- "link text"** : Returns an anchor element whose visible text matches the search value.
- "partial link text"** : Returns an anchor element whose visible text partially matches the search value.
- "tag name"** : Returns an element whose tag name matches the search value.
- "xpath"** : Returns an element matching an XPath expression.

### Value

`invisible(wbElement(res$value, remDr))`: An object of class "wElement" is invisibly returned. A `webElement` object see [wbElement](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other elementRetrieval functions: [findElementFromElement](#), [findElementsFromElement](#), [findElements](#), [getActiveElement](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# find the search form query box and search for "R project"
webElem <- remDr %>% findElement("name", "q") %>%
  elementSendKeys("R project", key = "enter")
# click the first link hopefully should be www.r-project.org
remDr %>% findElement("css", "h3.r a") %>% elementClick

# get the navigation div
navElem <- remDr %>% findElement("css", "div[role='navigation']")

# find all the links in this div
navLinks <- navElem %>% findElementsFromElement("css", "a")

# check the links
nLinks <- sapply(navLinks, function(x) x %>% getElementText)

# compare with all links
allLinks <- remDr %>% findElements("css", "a")
aLinks <- sapply(allLinks, function(x) x %>% getElementText)

# show the effect of searching for elements from element
aLinks %in% nLinks

remDr %>% deleteSession

## End(Not run)
```

---

 findElementFromElement

*Search for an element on the page, starting from another element*


---

### Description

findElementFromElement Search for an element on the page, starting from the node defined by the parent webElement. The located element will be returned as an object of wElement class.

### Usage

```
findElementFromElement(webElem, using = c("xpath", "css selector", "id",
    "name", "tag name", "class name", "link text", "partial link text"), value,
    ...)
```

### Arguments

webElem	An object of class "wElement". A web Element object see <a href="#">wbElement</a> .
using	Locator scheme to use to search the element, available schemes: "class name", "css selector", "id", "name", "link text", "partial link text", "tag name", "xpath". Defaults to 'xpath'. Partial string matching is accepted.
value	The search target. See examples.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Details

Details of possible locator schemes

**"class name"** : Returns an element whose class name contains the search value; compound class names are not permitted.

**"css selector"** : Returns an element matching a CSS selector.

**"id"** : Returns an element whose ID attribute matches the search value.

**"name"** : Returns an element whose NAME attribute matches the search value.

**"link text"** : Returns an anchor element whose visible text matches the search value.

**"partial link text"** : Returns an anchor element whose visible text partially matches the search value.

**"tag name"** : Returns an element whose tag name matches the search value.

**"xpath"** : Returns an element matching an XPath expression.

### Value

invisible(wbElement(res\$value, webElem\$remDr)): An object of class "wElement" is invisibly returned. A webElement object see [wbElement](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other elementRetrieval functions: [findElementsFromElement](#), [findElements](#), [findElement](#), [getActiveElement](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# find the search form query box and search for "R project"
webElem <- remDr %>% findElement("name", "q") %>%
  elementSendKeys("R project", key = "enter")
# click the first link hopefully should be www.r-project.org
remDr %>% findElement("css", "h3.r a") %>% elementClick

# get the navigation div
navElem <- remDr %>% findElement("css", "div[role='navigation']")

# find all the links in this div
navLinks <- navElem %>% findElementsFromElement("css", "a")

# check the links
nLinks <- sapply(navLinks, function(x) x %>% getElementText)

# compare with all links
allLinks <- remDr %>% findElements("css", "a")
aLinks <- sapply(allLinks, function(x) x %>% getElementText)

# show the effect of searching for elements from element
aLinks %in% nLinks

remDr %>% deleteSession

## End(Not run)
```

---

findElements

*Search for multiple elements on the page*

---

**Description**

findElements Search for multiple elements on the page, starting from the document root. The located elements will be returned as a list of objects of class wElement.

**Usage**

```
findElements(remDr, using = c("xpath", "css selector", "id", "name",
  "tag name", "class name", "link text", "partial link text"), value, ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
using	Locator scheme to use to search the element, available schemes: "class name", "css selector", "id", "name", "link text", "partial link text", "tag name", "xpath". Defaults to 'xpath'. Partial string matching is accepted.
value	The search target. See examples.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Details**

Details of possible locator schemes

**"class name"** : Returns an element whose class name contains the search value; compound class names are not permitted.

**"css selector"** : Returns an element matching a CSS selector.

**"id"** : Returns an element whose ID attribute matches the search value.

**"name"** : Returns an element whose NAME attribute matches the search value.

**"link text"** : Returns an anchor element whose visible text matches the search value.

**"partial link text"** : Returns an anchor element whose visible text partially matches the search value.

**"tag name"** : Returns an element whose tag name matches the search value.

**"xpath"** : Returns an element matching an XPath expression.

**Value**

invisible(lapply(res\$value, wbElement, remDr = remDr)): A list of objects of class "wElement" is invisibly returned. A webElement object see [wbElement](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other elementRetrieval functions: [findElementFromElement](#), [findElementsFromElement](#), [findElement](#), [getActiveElement](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# find the search form query box and search for "R project"
webElem <- remDr %>% findElement("name", "q") %>%
  elementSendKeys("R project", key = "enter")
# click the first link hopefully should be www.r-project.org
remDr %>% findElement("css", "h3.r a") %>% elementClick

# get the navigation div
```

```

navElem <- remDr %>% findElement("css", "div[role='navigation']")

# find all the links in this div
navLinks <- navElem %>% findElementsFromElement("css", "a")

# check the links
nLinks <- sapply(navLinks, function(x) x %>% getElementText)

# compare with all links
allLinks <- remDr %>% findElements("css", "a")
aLinks <- sapply(allLinks, function(x) x %>% getElementText)

# show the effect of searching for elements from element
aLinks %in% nLinks

remDr %>% deleteSession

## End(Not run)

```

---

findElementsFromElement

*Search for multiple elements on the page, starting from another element.*

---

## Description

findElementsFromElement Search for multiple elements on the page, starting from the node defined by the parent webElement. The located elements will be returned as an list of objects of class wElement.

## Usage

```

findElementsFromElement(webElem, using = c("xpath", "css selector", "id",
  "name", "tag name", "class name", "link text", "partial link text"), value,
  ...)

```

## Arguments

webElem	An object of class "wElement". A web Element object see <a href="#">wbElement</a> .
using	Locator scheme to use to search the element, available schemes: "class name", "css selector", "id", "name", "link text", "partial link text", "tag name", "xpath". Defaults to 'xpath'. Partial string matching is accepted.
value	The search target. See examples.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

## Details

Details of possible locator schemes

**"class name"** : Returns an element whose class name contains the search value; compound class names are not permitted.

**"css selector"** : Returns an element matching a CSS selector.

**"id"** : Returns an element whose ID attribute matches the search value.

**"name"** : Returns an element whose NAME attribute matches the search value.

**"link text"** : Returns an anchor element whose visible text matches the search value.

**"partial link text"** : Returns an anchor element whose visible text partially matches the search value.

**"tag name"** : Returns an element whose tag name matches the search value.

**"xpath"** : Returns an element matching an XPath expression.

## Value

`invisible(lapply(res$value, wbElement, remDr = webElem$remDr))`: A list of objects of class "wElement" is invisibly returned. A webElement object see [wbElement](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

## See Also

Other elementRetrieval functions: [findElementFromElement](#), [findElements](#), [findElement](#), [getActiveElement](#)

## Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# find the search form query box and search for "R project"
webElem <- remDr %>% findElement("name", "q") %>%
  elementSendKeys("R project", key = "enter")
# click the first link hopefully should be www.r-project.org
remDr %>% findElement("css", "h3.r a") %>% elementClick

# get the navigation div
navElem <- remDr %>% findElement("css", "div[role='navigation']")

# find all the links in this div
navLinks <- navElem %>% findElementsFromElement("css", "a")

# check the links
nLinks <- sapply(navLinks, function(x) x %>% getElementText)

# compare with all links
allLinks <- remDr %>% findElements("css", "a")
aLinks <- sapply(allLinks, function(x) x %>% getElementText)
```

```

# show the effect of searching for elements from element
aLinks %in% nLinks

remDr %>% deleteSession

## End(Not run)

```

---

forward

*Navigate forwards*


---

### Description

forward Navigate forwards in the browser history, if possible.

### Usage

```
forward(remDr, ...)
```

### Arguments

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).  
...                Additional function arguments - Currently passes the [retry](#) argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other navigation functions: [back](#), [getCurrentUrl](#), [getTitle](#), [go](#), [refresh](#)

### Examples

```

## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# get the title
remDr %>% getTitle

# get the current page url
remDr %>% getCurrentUrl

# navigate
remDr %>% go("http://www.bbc.co.uk")

```



```
# go back
remDr %>% (seleniumPipes::back)

# go forward
remDr %>% forward

# refresh page
remDr %>% refresh

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

fullscreenWindow	<i>Make current window full-screen</i>
------------------	--

---

## Description

fullscreenWindow The Fullscreen Window command invokes the window manager-specific “full screen” operation, if any, on the window containing the current top-level browsing context. This typically increases the window to the size of the physical display and can hide browser UI elements such as toolbars.

## Usage

```
fullscreenWindow(remDr, ...)
```

## Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

## Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

## See Also

Other commandContexts functions: [closeWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowold()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)
```

**Description**

getActiveElement Get the element on the page that currently has focus. The located element will be returned as a WebElement object.

**Usage**

```
getActiveElement(remDr, ...)
```

**Arguments**

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).  
 ...             Additional function arguments - Currently passes the [retry](#) argument.

**Value**

invisible(webElement(res\$value, remDr)): An object of class "wElement" is invisibly returned. A webElement object see [webElement](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other elementRetrieval functions: [findElementFromElement](#), [findElementsFromElement](#), [findElements](#), [findElement](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# find the search form query box and search for "R project"
webElem <- remDr %>% findElement("name", "q") %>%
  elementSendKeys("R project", key = "enter")
# click the first link hopefully should be www.r-project.org
remDr %>% findElement("css", "h3.r a") %>% elementClick

# get the navigation div
navElem <- remDr %>% findElement("css", "div[role='navigation']")

# find all the links in this div
navLinks <- navElem %>% findElementsFromElement("css", "a")

# check the links
nLinks <- sapply(navLinks, function(x) x %>% getElementText)

# compare with all links
allLinks <- remDr %>% findElements("css", "a")
aLinks <- sapply(allLinks, function(x) x %>% getElementText)

# show the effect of searching for elements from element
aLinks %in% nLinks
```

```
remDr %>% deleteSession
## End(Not run)
```

---

getAlertText	<i>Get alert text</i>
--------------	-----------------------

---

### Description

getAlertText Get the text from a JavaScript alert.

### Usage

```
getAlertText(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

The text from the alert is returned.

### See Also

Other userPrompts functions: [acceptAlert](#), [dismissAlert](#), [sendAlertText](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
sScript <- "setTimeout(function(){alert('HELLO')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% dismissAlert()

sScript <- "setTimeout(function(){confirm('Press a button')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% acceptAlert()

sScript <- "setTimeout(function(){confirm('Press a button')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% getAlertText()
remDr %>% dismissAlert()
```

```
sScript <- "setTimeout(function(){prompt('Please enter your name', '')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% getAlertText()
remDr %>% sendAlertText("Buck Rogers?")

remDr %>% deleteSession()

## End(Not run)
```

---

getAlertTextOld	<i>Get alert text</i>
-----------------	-----------------------

---

### Description

getAlertTextOld Get the text from a JavaScript alert. This uses the old JSONwireprotocol endpoints.

### Usage

```
getAlertTextOld(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

The alert text is returned as a string.

### See Also

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

### Examples

```
## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)
```

---

getAllCookies	<i>Get all current domain cookies</i>
---------------	---------------------------------------

---

### Description

getAllCookies Get all the cookies for the current domain.

### Usage

```
getAllCookies(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

A list of all the cookies on the current domain are returned. These cookies have values as stipulated by the arguments given in [addCookie](#).

### See Also

Other cookies functions: [addCookie](#), [deleteAllCookies](#), [deleteCookie](#), [getNamedCookie](#)

### Examples

```
## Not run:
# assume a server is running at default location
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
# get the cookies
remDr %>% getCookie()
# get a named cookie
remDr %>% getCookie("NID")
# add our own cookie
remDr %>% addCookie(name = "myCookie", value = "12")
# check its value
remDr %>% getCookie("myCookie")
# delete our cookie
remDr %>% deleteCookie("myCookie")
# check its deleted
remDr %>% getCookie("myCookie")

# delete all cookies
remDr %>% getCookie()
remDr %>% deleteAllCookies() %>%
  getCookie()
```

```
remDr %>% deleteSession()

## End(Not run)
```

---

getCurrentUrl	<i>Retrieve the URL of the current page.</i>
---------------	--

---

### Description

getCurrentUrl Retrieve the URL of the current page.

### Usage

```
getCurrentUrl(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

A character string is returned giving the current page URL.

### See Also

Other navigation functions: [back](#), [forward](#), [getTitle](#), [go](#), [refresh](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# get the title
remDr %>% getTitle

# get the current page url
remDr %>% getCurrentUrl

# navigate
remDr %>% go("http://www.bbc.co.uk")

# go back
remDr %>% (seleniumPipes::back)

# go forward
remDr %>% forward
```

```
# refresh page
remDr %>% refresh

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

getElementAttribute *Get the value of an element's attribute.*

---

### Description

getElementAttribute Get the value of an element's attribute.

### Usage

```
getElementAttribute(webElem, attribute, ...)
```

### Arguments

webElem	An object of class "wElement". A web Element object see <a href="#">wbElement</a> .
attribute	The attribute to query as a character string.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

The value of the attribute, or null if it is not set on the element.

### See Also

Other elementState functions: [getElementCssValue](#), [getElementProperty](#), [getElementRect](#), [getElementTagName](#), [getElementText](#), [isElementEnabled](#), [isElementSelected](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# select the search box
searchElem <- remDr %>% findElement("name", "q")

# get the name of our element
searchElem %>% getElementAttribute("name")

# get the css background-color
```



```

searchElem %>% getElementCssValue("background-color")
blueScript <- "arguments[0].style.backgroundColor = 'blue';"
remDr %>% executeScript(blueScript, list(searchElem))
searchElem %>% getElementCssValue("background-color")

# get the javascript property
# searchElem %>% getElementProperty("backgroundColor")

# get dimensions
searchElem %>% getElementRect

searchElem %>% getElementTagName

# get some text and return it
remDr %>% go("http://r-project.org")
remDr %>% findElement("css", "h1") %>% getElementText

# close browser
remDr %>% deleteSession

## End(Not run)

```

---

getElementCssValue      *Query the value of an element's computed CSS property.*

---

## Description

getElementCssValue Query the value of an element's computed CSS property. The CSS property to query should be specified using the CSS property name, not the JavaScript property name (e.g. background-color instead of backgroundColor).

## Usage

```
getElementCssValue(webElem, propertyName, ...)
```

## Arguments

webElem	An object of class "wElement". A web Element object see <a href="#">wbElement</a> .
propertyName	The property to query as a character string
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

## Value

The value of the specified CSS property.

## See Also

Other elementState functions: [getElementAttribute](#), [getElementProperty](#), [getElementRect](#), [getElementTagName](#), [getElementText](#), [isElementEnabled](#), [isElementSelected](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# select the search box
searchElem <- remDr %>% findElement("name", "q")

# get the name of our element
searchElem %>% getElementAttribute("name")

# get the css background-color
searchElem %>% getElementCssValue("background-color")
blueScript <- "arguments[0].style.backgroundColor = 'blue';"
remDr %>% executeScript(blueScript, list(searchElem))
searchElem %>% getElementCssValue("background-color")

# get the javascript property
# searchElem %>% getElementProperty("backgroundColor")

# get dimensions
searchElem %>% getElementRect

searchElem %>% getElementTagName

# get some text and return it
remDr %>% go("http://r-project.org")
remDr %>% findElement("css", "h1") %>% getElementText

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

getElementProperty      *Query the value of an elements property.*

---

**Description**

getElementProperty Query the value of an elements property.

**Usage**

```
getElementProperty(webElem, property, ...)
```

**Arguments**

webElem                  An object of class "wElement". A web Element object see [wbElement](#).

property            The property to query as a character string  
...                 Additional function arguments - Currently passes the [retry](#) argument.

### Value

The value of the elements specified property.

### See Also

Other elementState functions: [getElementAttribute](#), [getElementCssValue](#), [getElementRect](#), [getElementTagName](#), [getElementText](#), [isElementEnabled](#), [isElementSelected](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# select the search box
searchElem <- remDr %>% findElement("name", "q")

# get the name of our element
searchElem %>% getElementAttribute("name")

# get the css background-color
searchElem %>% getElementCssValue("background-color")
blueScript <- "arguments[0].style.backgroundColor = 'blue';"
remDr %>% executeScript(blueScript, list(searchElem))
searchElem %>% getElementCssValue("background-color")

# get the javascript property
# searchElem %>% getElementProperty("backgroundColor")

# get dimensions
searchElem %>% getElementRect

searchElem %>% getElementTagName

# get some text and return it
remDr %>% go("http://r-project.org")
remDr %>% findElement("css", "h1") %>% getElementText

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

getElementRect	<i>Return the dimensions and coordinates of an element</i>
----------------	--

---

### Description

getElementRect The getElementRect fuinction returns the dimensions and coordinates of the given web element.

### Usage

```
getElementRect(webElem, ...)
```

### Arguments

webElem	An object of class "wElement". A web Element object see <a href="#">wbElement</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

The returned value is a list including the following members:

**x** X axis position of the top-left corner of the web element relative to the current browsing context's document element in CSS reference pixels.

**y** Y axis position of the top-left corner of the web element relative to the current browsing context's document element in CSS reference pixels.

**height** Height of the web element's bounding rectangle in CSS reference pixels.

**width** Width of the web element's bounding rectangle in CSS reference pixels.

### See Also

Other elementState functions: [getElementAttribute](#), [getElementCssValue](#), [getElementProperty](#), [getElementTagName](#), [getElementText](#), [isElementEnabled](#), [isElementSelected](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# select the search box
searchElem <- remDr %>% findElement("name", "q")

# get the name of our element
searchElem %>% getElementAttribute("name")

# get the css background-color
searchElem %>% getElementCssValue("background-color")
blueScript <- "arguments[0].style.backgroundColor = 'blue';"
```

```
remDr %>% executeScript(blueScript, list(searchElem))
searchElem %>% getElementCssValue("background-color")

# get the javascript property
# searchElem %>% getElementProperty("backgroundColor")

# get dimensions
searchElem %>% getElementRect

searchElem %>% getElementTagName

# get some text and return it
remDr %>% go("http://r-project.org")
remDr %>% findElement("css", "h1") %>% getElementText

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

getElementTagName      *Query for an element's tag name.*

---

## Description

getElementTagName Query for an element's tag name.

## Usage

```
getElementTagName(webElem, ...)
```

## Arguments

webElem      An object of class "wElement". A web Element object see [wbElement](#).  
...      Additional function arguments - Currently passes the [retry](#) argument.

## Value

The element's tag name, as a lowercase character string.

## See Also

Other elementState functions: [getElementAttribute](#), [getElementCssValue](#), [getElementProperty](#), [getElementRect](#), [getElementText](#), [isElementEnabled](#), [isElementSelected](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# select the search box
searchElem <- remDr %>% findElement("name", "q")

# get the name of our element
searchElem %>% getElementAttribute("name")

# get the css background-color
searchElem %>% getElementCssValue("background-color")
blueScript <- "arguments[0].style.backgroundColor = 'blue';"
remDr %>% executeScript(blueScript, list(searchElem))
searchElem %>% getElementCssValue("background-color")

# get the javascript property
# searchElem %>% getElementProperty("backgroundColor")

# get dimensions
searchElem %>% getElementRect

searchElem %>% getElementTagName

# get some text and return it
remDr %>% go("http://r-project.org")
remDr %>% findElement("css", "h1") %>% getElementText

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

getElementText	<i>Returns the visible text for the element.</i>
----------------	--

---

**Description**

getElementText Returns the visible text for the element.

**Usage**

```
getElementText(webElem, ...)
```

**Arguments**

webElem	An object of class "wElement". A web Element object see <a href="#">wbElement</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

The visible text for an element is returned as a character string.

**See Also**

Other elementState functions: [getElementAttribute](#), [getElementCssValue](#), [getElementProperty](#), [getElementRect](#), [getElementTagName](#), [isElementEnabled](#), [isElementSelected](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# select the search box
searchElem <- remDr %>% findElement("name", "q")

# get the name of our element
searchElem %>% getElementAttribute("name")

# get the css background-color
searchElem %>% getElementCssValue("background-color")
blueScript <- "arguments[0].style.backgroundColor = 'blue';"
remDr %>% executeScript(blueScript, list(searchElem))
searchElem %>% getElementCssValue("background-color")

# get the javascript property
# searchElem %>% getElementProperty("backgroundColor")

# get dimensions
searchElem %>% getElementRect

searchElem %>% getElementTagName

# get some text and return it
remDr %>% go("http://r-project.org")
remDr %>% findElement("css", "h1") %>% getElementText

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

getNamedCookie

*Get a named cookie*


---

**Description**

getNamedCookie Get the cookie with a given name.

**Usage**

```
getNamedCookie(remDr, name = NULL, ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
name	character: The name of the cookie; may not be null or an empty string
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

A named cookies from the current domain is returned if it exists. These cookies have values as stipulated by the arguments given in [addCookie](#).

**See Also**

Other cookies functions: [addCookie](#), [deleteAllCookies](#), [deleteCookie](#), [getAllCookies](#)

**Examples**

```
## Not run:
# assume a server is running at default location
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
# get the cookies
remDr %>% getCookie()
# get a named cookie
remDr %>% getCookie("NID")
# add our own cookie
remDr %>% addCookie(name = "myCookie", value = "12")
# check its value
remDr %>% getCookie("myCookie")
# delete our cookie
remDr %>% deleteCookie("myCookie")
# check its deleted
remDr %>% getCookie("myCookie")

# delete all cookies
remDr %>% getCookie()
remDr %>% deleteAllCookies() %>%
  getCookie()

remDr %>% deleteSession()

## End(Not run)
```



---

getPageSource	<i>Get source of last page.</i>
---------------	---------------------------------

---

### Description

getPageSource Get the source of the last loaded page.

### Usage

```
getPageSource(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

read\_html(res\$value): An xml document is returned. The content from the remote webdriver is parsed (see [read\\_html](#)) and returned as xml.

### See Also

Other documentHandling functions: [executeAsyncScript](#), [executeScript](#)

### Examples

```
## Not run:
remDr <- remoteDr()
# Get the page source
remDr %>% go("https://www.google.com/ncr") %>%
  getPageSource

remDr %>% getTitle()
webElem <- remDr %>% findElement("css", "img#hplogo")
# check if the logo is hidden
remDr %>% executeScript("return document.getElementById('hplogo').hidden;", args = list())
# [1] FALSE
# hide the logo
remDr %>% executeScript("document.getElementById('hplogo').hidden = true;", args = list())
# Make the logo visible this time passing a web Element as an argument
remDr %>% executeScript(script = "return arguments[0].hidden = false;", args = list(webElem))

# Pass arguments
remDr %>% executeScript(script = "return argument[1] + argument[2];",
  , args = list(1, 2))

# Return a web Element
remDr %>% executeScript(script = "return document.getElementById('hplogo');", args = list())
```

```

# ElementId: 0
# Remote Driver:
# Remote Ip Address: http://localhost:4444/wd/hub
# Remote sessionId: 9a83672a-d72b-4873-aa7d-96f7f1f80fa0

# Return a web Element in a more complex object
script <- "var test ={num:1, str:'a', el:document.getElementById('hplogo')};return test;"
remDr %>% executeScript(script = script
                        , args = list())

# $str
# [1] "a"
#
# $num
# [1] 1
#
# $el
# ElementId: 0
# Remote Driver:
# Remote Ip Address: http://localhost:4444/wd/hub
# Remote sessionId: 9a83672a-d72b-4873-aa7d-96f7f1f80fa0

# Run with replace = FALSE
remDr %>% executeScript(script = script
                        , args = list(), replace = FALSE)

# $str
# [1] "a"
#
# $num
# [1] 1
#
# $el
# $el$ELEMENT
# [1] "0"

remDr %>% setTimeout("script")

asScript <- "cb = arguments[0];setTimeout(function(){cb('DONE');},5000); "
system.time(test1 <- remDr %>% executeAsyncScript(asScript, args = list()))
sScript <- "setTimeout(function(){},5000); return 'DONE';"
system.time(test2 <- remDr %>% executeScript(sScript, args = list()))

remDr %>% deleteSession()

## End(Not run)

```

**Description**

`getTitle` Get the current page title.

**Usage**

```
getTitle(remDr, ...)
```

**Arguments**

<code>remDr</code>	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
<code>...</code>	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

The title of the current page is returned as a character string.

**See Also**

Other navigation functions: [back](#), [forward](#), [getCurrentUrl](#), [go](#), [refresh](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# get the title
remDr %>% getTitle

# get the current page url
remDr %>% getCurrentUrl

# navigate
remDr %>% go("http://www.bbc.co.uk")

# go back
remDr %>% (seleniumPipes::back)

# go forward
remDr %>% forward

# refresh page
remDr %>% refresh

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

getWindowHandle	<i>get current window handle</i>
-----------------	----------------------------------

---

### Description

getWindowHandle Retrieve the current window handle.

### Usage

```
getWindowHandle(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

Returns a string which is the "handle" for the current window.

### See Also

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
```

```

remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

getWindowHandleOld      *Retrieve the current window handle.*

---

### Description

getWindowHandleOld Retrieve the current window handle. Uses the old JSONwireprotocol end points

### Usage

```
getWindowHandleOld(remDr, ...)
```

### Arguments

remDr                    An object of class "rDriver". A remote driver object see [remoteDr](#).  
...                        Additional function arguments - Currently passes the [retry](#) argument.

### Value

Returns a string which is the "handle" for the current window.

### See Also

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

**Examples**

```
## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)
```

---

getWindowHandles	<i>Get all window handles.</i>
------------------	--------------------------------

---

**Description**

getWindowHandles Retrieve the list of all window handles available to the session.

**Usage**

```
getWindowHandles(remDr, ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

Returns a list of windows handles. Each element of the list is a string. The order window handles are returned is arbitrary.

**See Also**

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
```

```

remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

getWindowHandlesOld *Get all window handles.*

---

### Description

getWindowHandlesOld Uses the old JSONwireprotocol end points. Retrieve the list of all window handles available to the session.

### Usage

```
getWindowHandlesOld(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

Returns a list of windows handles. Each element of the list is a string. The order window handles are returned is arbitrary.

**See Also**

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

**Examples**

```
## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)
```

---

<code>getWindowPosition</code>	<i>Get current window position</i>
--------------------------------	------------------------------------

---

**Description**

`getWindowPosition` The Get Window Position command returns the position on the screen of the operating system window corresponding to the current top-level browsing context.

**Usage**

```
getWindowPosition(remDr, ...)
```

**Arguments**

<code>remDr</code>	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
<code>...</code>	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

Returns a list which contains the x coordinate to position the window at, relative to the upper left corner of the screen and the Y coordinate to position the window at, relative to the upper left corner of the screen

**See Also**

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)



**Examples**

```

## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowold()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

getWindowPositionOld *Get window position*

---

**Description**

`getWindowPositionOld` Get the position of the specified window. Uses the old JSONwireprotocol end points.

**Usage**

```
getWindowPositionOld(remDr, handle = "current", ...)
```

**Arguments**

<code>remDr</code>	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
<code>handle</code>	Handle of the window to query. If <code>handle = "current"</code> (the default) the current window is used.
<code>...</code>	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

Returns a list which contains the x coordinate to position the window at, relative to the upper left corner of the screen and the Y coordinate to position the window at, relative to the upper left corner of the screen

**See Also**

Other `oldMethods` functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

**Examples**

```
## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)
```

---

<code>getWindowSize</code>	<i>getWindowSize</i>
----------------------------	----------------------

---

**Description**

`getWindowSize`

**Usage**

```
getWindowSize(remDr, ...)
```

**Arguments**

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).  
 ...              Additional function arguments - Currently passes the [retry](#) argument.

**Value**

The width and height of the window are returned as elements in a list.

**See Also**

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
```

```

# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

getWindowSizeOld      *Get window size*

---

### Description

getWindowSizeOld Get the size of the specified window. Uses the old JSONwireprotocol end points.

### Usage

```
getWindowSizeOld(remDr, handle = "current", ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
handle	Handle of the window to query. If handle = "current" (the default) the current window is used.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

The width and height of the window are returned as elements in a list.

### See Also

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

### Examples

```

## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)

```

---

go *Navigate to a new URL.*

---

### Description

go Navigate to a new URL.

### Usage

```
go(remDr, url, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
url	The URL to navigate to.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other navigation functions: [back](#), [forward](#), [getCurrentUrl](#), [getTitle](#), [refresh](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# get the title
remDr %>% getTitle

# get the current page url
remDr %>% getCurrentUrl

# navigate
remDr %>% go("http://www.bbc.co.uk")

# go back
remDr %>% (seleniumPipes::back)

# go forward
remDr %>% forward

# refresh page
```

```

remDr %>% refresh

# close browser
remDr %>% deleteSession

## End(Not run)

```

---

isElementEnabled      *Determine if an element is currently enabled.*

---

### Description

isElementEnabled Determine if an element is currently enabled.

### Usage

```
isElementEnabled(webElem, ...)
```

### Arguments

webElem            An object of class "wElement". A web Element object see [wbElement](#).  
...                Additional function arguments - Currently passes the [retry](#) argument.

### Value

A logical value is returned indicating whether the element is enabled.

### See Also

Other elementState functions: [getElementAttribute](#), [getElementCssValue](#), [getElementProperty](#), [getElementRect](#), [getElementTagName](#), [getElementText](#), [isElementSelected](#)

### Examples

```

## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# select the search box
searchElem <- remDr %>% findElement("name", "q")

# get the name of our element
searchElem %>% getElementAttribute("name")

# get the css background-color
searchElem %>% getElementCssValue("background-color")
blueScript <- "arguments[0].style.backgroundColor = 'blue';"
remDr %>% executeScript(blueScript, list(searchElem))

```

```
searchElem %>% getElementCssValue("background-color")

# get the javascript property
# searchElem %>% getElementProperty("backgroundColor")

# get dimensions
searchElem %>% getElementRect

searchElem %>% getElementTagName

# get some text and return it
remDr %>% go("http://r-project.org")
remDr %>% findElement("css", "h1") %>% getElementText

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

isElementSelected      *Determine if an element is currently selected.*

---

## Description

isElementSelected Determines if an OPTION element, or an INPUT element of type checkbox or radiobutton is currently selected.

## Usage

```
isElementSelected(webElem, ...)
```

## Arguments

webElem      An object of class "wElement". A web Element object see [wbElement](#).  
...      Additional function arguments - Currently passes the [retry](#) argument.

## Value

A logical value is returned indicating whether the element is selected.

## See Also

Other elementState functions: [getElementAttribute](#), [getElementCssValue](#), [getElementProperty](#), [getElementRect](#), [getElementTagName](#), [getElementText](#), [isElementEnabled](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# select the search box
searchElem <- remDr %>% findElement("name", "q")

# get the name of our element
searchElem %>% getElementAttribute("name")

# get the css background-color
searchElem %>% getElementCssValue("background-color")
blueScript <- "arguments[0].style.backgroundColor = 'blue';"
remDr %>% executeScript(blueScript, list(searchElem))
searchElem %>% getElementCssValue("background-color")

# get the javascript property
# searchElem %>% getElementProperty("backgroundColor")

# get dimensions
searchElem %>% getElementRect

searchElem %>% getElementTagName

# get some text and return it
remDr %>% go("http://r-project.org")
remDr %>% findElement("css", "h1") %>% getElementText

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

maximizeWindow	<i>Maximize the current window.</i>
----------------	-------------------------------------

---

**Description**

maximizeWindow Maximize the current if not already maximized.

**Usage**

```
maximizeWindow(remDr, ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.



**Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")
```

```

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

maximizeWindowOld      *Maximize the current window.*

---

### Description

maximizeWindowOld Maximize the specified window if not already maximized.

### Usage

```
maximizeWindowOld(remDr, handle = "current", ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
handle	Handle of the window to query. If handle = "current" (the default) the current window is used.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

### Examples

```

## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)

```

---

newSession	<i>Create a new session.</i>
------------	------------------------------

---

### Description

newSession The server should attempt to create a session that most closely matches the desired and required capabilities. Required capabilities have higher priority than desired capabilities and must be set for the session to be created.

### Usage

```
newSession(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other sessions functions: [deleteSession](#), [setTimeout](#)

### Examples

```
## Not run:
# start a driver without opening a browser
remDr <- remoteDr(newSession = FALSE)

# open a browser
remDr %>% newSession

# set timeout on waiting for elements
remDr %>% setTimeout(type = "implicit", 5000)

# close Session
remDr %>% deleteSession

## End(Not run)
```

---

performActions	<i>Not currently implemented</i>
----------------	----------------------------------

---

### Description

performActions The Perform Actions command allows you to create sequential interactions that can be sent over from the local end to the remote end. This type of interactions allow emulations like drag and drop.

### Usage

```
performActions(remDr, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other interactions functions: [releasingActions](#)

### Examples

```
## Not run:
# functions not currently implemented

## End(Not run)
```

---

queryDriver	<i>Send a query to remote Driver.</i>
-------------	---------------------------------------

---

### Description

queryDriver A function to send a query to a remote driver. Intended for seleniumPipes internal use mainly.

### Usage

```
queryDriver(verb = GET, url, source, drvID, ...)
```

**Arguments**

verb	The http method to use. See <a href="#">VERB</a>
url	The url of the remote server endpoint.
source	The name of the seleniumPipes function that called queryDriver.
drvID	The driver id of the session as given by an object of class "remoteDr"
...	additonal arguments

**Value**

The contents of the response from the remote server. See [content](#) for details.

**Examples**

```
## Not run:
# internal method

## End(Not run)
```

---

refresh	<i>Refresh the current page.</i>
---------	----------------------------------

---

**Description**

refresh Refresh the current page.

**Usage**

```
refresh(remDr, ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additonal function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other navigation functions: [back](#), [forward](#), [getCurrentUrl](#), [getTitle](#), [go](#)

## Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")

# get the title
remDr %>% getTitle

# get the current page url
remDr %>% getCurrentUrl

# navigate
remDr %>% go("http://www.bbc.co.uk")

# go back
remDr %>% (seleniumPipes::back)

# go forward
remDr %>% forward

# refresh page
remDr %>% refresh

# close browser
remDr %>% deleteSession

## End(Not run)
```

---

releasingActions	<i>Not currently implemented</i>
------------------	----------------------------------

---

## Description

releasingActions The Release Actions command is used to cancel all current action chains. This is the equivalent of releasing all modifiers from input sources.

## Usage

```
releasingActions(remDr, ...)
```

## Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other interactions functions: [performActions](#)

**Examples**

```
## Not run:
# functions not currently implemented

## End(Not run)
```

---

remoteDr

*Create a remote driver*


---

**Description**

remoteDr: Create a remote Driver object

**Usage**

```
remoteDr(remoteServerAddr = "http://localhost", port = 4444L,
  browserName = "firefox", version = "", platform = "ANY",
  javascript = TRUE, nativeEvents = TRUE, extraCapabilities = list(),
  path = "wd/hub", newSession = TRUE)
```

**Arguments**

remoteServerAddr	Object of class "character", giving the ip of the remote server. Defaults to localhost
port	Object of class "integer", the port of the remote server on which to connect
browserName	Object of class "character". The name of the browser being used; choices include chromelfirefoxinternet explorerliphonelhtmlunit. Defaults to firefox.
version	Object of class "character". The browser version, or the empty string if unknown.
platform	Object of class "character". A key specifying which platform the browser is running on. This value should be one of WINDOWS XPI VISTA MAC LINUX UNIX. When requesting a new session, the client may specify "ANY" to indicate any available platform may be used.

javascript	Object of class "logical". Whether the session supports executing user supplied JavaScript in the context of the current page.
nativeEvents	Object of class "logical". Whether the session supports native events. In WebDriver advanced user interactions are provided by either simulating the Javascript events directly (i.e. synthetic events) or by letting the browser generate the Javascript events (i.e. native events). Native events simulate the user interactions better.
extraCapabilities	A list containing any os/platform/driver specific arguments.
path	Path on the server side to issue webdriver calls to. Normally use the default value.
newSession	Logical value whether to start an instance of the browser. If TRUE a browser will be opened using <code>newSession</code>

### Value

An object of class "rDriver" is returned. This is a remote Driver object that is used in many of the remote driver specific functions. Many functions that take a remote driver object as input also return the remote driver object. This allows chaining of commands. See the examples for chaining in action.

### Examples

```
## Not run:
# assume a server is available at the default location.
remDr <- remoteDr()
remDr %>% go("http://www.google.com") %>%
  findElement("name", "q") %>%
  elementSendKeys("R project", key = "enter")
# close our browser
remDr %>% deleteSession

## End(Not run)
```

---

retry

*Documetation of retry argument*

---

### Description

The ability to retry function code is enabled by default. `retry` can be a logical value. if it is TRUE then `noTry = getOption("seleniumPipes_no_try")` and `delay = getOption("seleniumPipes_no_try_delay")`. If it is FALSE the facility to have multiple tries of the function call is removed. `retry` can also be a list with the following named arguments that will override the values in options

**"noTry"** Integer indicating how many times to try the function call

**"delay"** Integer indicating delay between tries of the function call



**Examples**

```
## Not run:
remDr <- remoteDr()
webElem <- remDr %>% go("http://www.google.com/ncr") %>%
  findElement("name", "q")
# change the name of q with an 8 second delay
myscript <- "var myElem = arguments[0];
window.setTimeout(function(){
  myElem.setAttribute('name','funkyname');
}, 8000);"
remDr %>% executeScript(myscript, args = list(webElem))

newWebElem <- remDr %>% findElement("name", "funkyname")

# > newWebElem <- remDr %>% findElement("name", "funkyname")
#
# Calling findElement - Try no: 1 of 3
#
# Calling findElement - Try no: 2 of 3

newWebElem %>% getElementAttribute("name")

# compare with a function that will fail (no element present)
remDr %>% findElement("id", "i am not here", retry = list(noTry = 5, delay = 10))
remDr %>% findElement("id", "i am not here", retry = FALSE)

## End(Not run)
```

---

seleniumPipes

*Implements the W3C webdriver specification.*


---

**Description**

Implements the W3C webdriver specification.

**Package options**

seleniumPipes uses the following [options](#) to configure behaviour:

- `seleniumPipes_display_screenshot`: Logical value indicating whether to display PNG returned by `takeScreenshot` and `takeElementScreenshot`. Defaults to TRUE
- `seleniumPipes_no_try`: An integer giving the number of time to try calling an endpoint on the Selenium Server. Defaults to 3 attempts
- `seleniumPipes_no_try_delay`: An integer detailing the delay between attempts to call a failing endpoint on the Selenium Server. Defaults to 5000 milliseconds = 5 seconds.
- `seleniumPipes_SL`: A logical value which acts as a flag indicating whether SauceLabs is being used for package testing.

- seleniumPipes\_selOptions: A list used to store options to pass to [remoteDr](#) when running tests.
- seleniumPipes\_sauceID: A character used to store remote session ids when running Sauce-Lab tests on the package.

---

selKeys	<i>Selenium key mappings</i>
---------	------------------------------

---

### Description

This data set contains a list of selenium key mappings. The key mappings are outlined at <https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#session/sessionid/element/id/value>. selKeys is used when a sendKeys variable is needed. sendKeys is defined as a list. If an entry is needed from selKeys it is denoted by key.

### Usage

```
selKeys
```

### Format

A named list. The names are the descriptions of the keys. The values are the "UTF-8" character representations.

### Author(s)

John Harrison, 2012-10-05

### Source

<http://code.google.com/p/selenium/wiki/JsonWireProtocol#/session/:sessionId/element/:id/value>

---

sendAlertText	<i>Send text to alert</i>
---------------	---------------------------

---

### Description

sendAlertText Send keystrokes to JavaScript prompt() dialog

### Usage

```
sendAlertText(remDr, text = "", ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
text	A character vector of length 1. In other words a string. The text is passed to the JavaScript alert
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other userPrompts functions: [acceptAlert](#), [dismissAlert](#), [getAlertText](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% go("https://www.google.com/ncr") %>%
  getTitle()
sScript <- "setTimeout(function(){alert('HELLO')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% dismissAlert()

sScript <- "setTimeout(function(){confirm('Press a button')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% acceptAlert()

sScript <- "setTimeout(function(){confirm('Press a button')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% getAlertText()
remDr %>% dismissAlert()

sScript <- "setTimeout(function(){prompt('Please enter your name', '')},1000); return 'DONE';"
remDr %>% executeScript(sScript, args = list())
remDr %>% getAlertText()
remDr %>% sendAlertText("Buck Rogers?")

remDr %>% deleteSession()

## End(Not run)
```

---

sendAlertTextOld	<i>Send text to alert</i>
------------------	---------------------------

---

### Description

sendAlertTextOld Send keystrokes to JavaScript prompt() dialog. This uses the old JSONwire-protocol endpoints.

### Usage

```
sendAlertTextOld(remDr, text = "", ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
text	A character vector of length 1. In other words a string. The text is passed to the JavaScript alert
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [setWindowPositionOld](#), [setWindowSizeOld](#)

### Examples

```
## Not run:  
# functions in this group are using the old JSONwireprotocol end points  
  
## End(Not run)
```

---

setTimeout	<i>Configure the amount of time that a particular type of operation can execute</i>
------------	---

---

### Description

setTimeout Configure the amount of time that a particular type of operation can execute for before they are aborted and a `!Timeout!` error is returned to the client.

### Usage

```
setTimeout(remDr, type = "page load", milliseconds = 10000, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
type	The type of operation to set the timeout for. Valid values are: "script" for script timeouts, "implicit" for modifying the implicit wait timeout and "page load" for setting a page load timeout.
milliseconds	The amount of time, in milliseconds, that time-limited commands are permitted to run.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

`invisible(remDr)`: An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other sessions functions: [deleteSession](#), [newSession](#)

### Examples

```
## Not run:
# start a driver without opening a browser
remDr <- remoteDr(newSession = FALSE)

# open a browser
remDr %>% newSession

# set timeout on waiting for elements
remDr %>% setTimeout(type = "implicit", 5000)

# close Session
remDr %>% deleteSession
```

```
## End(Not run)
```

---

```
setWindowPosition      Change the position of the current window.
```

---

### Description

setWindowPosition Change the position of the current window.

### Usage

```
setWindowPosition(remDr, x, y, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
x	integer The X coordinate to position the window at, relative to the upper left corner of the screen.
y	integer The Y coordinate to position the window at, relative to the upper left corner of the screen.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld
```

```

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

setWindowPositionOld *Change the position of the specified window.*

---

### Description

setWindowSize Change the position of the specified window.

### Usage

```
setWindowPositionOld(remDr, x, y, handle = "current", ...)
```

### Arguments

remDr An object of class "rDriver". A remote driver object see [remoteDr](#).

x	integer The X coordinate to position the window at, relative to the upper left corner of the screen.
y	integer The Y coordinate to position the window at, relative to the upper left corner of the screen.
handle	Handle of the window to query. If handle = "current" (the default) the current window is used.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

**Value**

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowSizeOld](#)

**Examples**

```
## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)
```

---

setWindowSize	<i>Change the size of the current window.</i>
---------------	---

---

**Description**

setWindowSize Change the size of the current window.

**Usage**

```
setWindowSize(remDr, width, height, ...)
```

**Arguments**

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
width	integer The new window width.
height	integer The new window height.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.



## Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

## See Also

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [switchToFrame](#), [switchToParentFrame](#), [switchToWindow](#)

## Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")
```

```

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

setWindowSizeOld      *Change the size of the specified window.*

---

### Description

setWindowSize Change the size of the specified window.

### Usage

```
setWindowSizeOld(remDr, width, height, handle = "current", ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
width	integer The new window width.
height	integer The new window height.
handle	Handle of the window to query. If handle = "current" (the default) the current window is used.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other oldMethods functions: [acceptAlertOld](#), [dismissAlertOld](#), [executeAsyncScriptOld](#), [executeScriptOld](#), [getAlertTextOld](#), [getWindowHandleOld](#), [getWindowHandlesOld](#), [getWindowPositionOld](#), [getWindowSizeOld](#), [maximizeWindowOld](#), [sendAlertTextOld](#), [setWindowPositionOld](#)

### Examples

```

## Not run:
# functions in this group are using the old JSONwireprotocol end points

## End(Not run)

```

---

switchToFrame	<i>Change focus to another frame on the page.</i>
---------------	---

---

### Description

switchToFrame Change focus to another frame on the page. If the frame id is null, the server should switch to the page's default content.

### Usage

```
switchToFrame(remDr, Id = NULL, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
Id	string number null WebElement Identifier for the frame to change focus to.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToParentFrame](#), [switchToWindow](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld
```

```

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

switchToParentFrame *Change focus to the parent context.*

---

## Description

switchToParentFrame Change focus to the parent context. If the current context is the top level browsing context, the context remains unchanged.

## Usage

```
switchToParentFrame(remDr, ...)
```

## Arguments

remDr An object of class "rDriver". A remote driver object see [remoteDr](#).  
 ... Additional function arguments - Currently passes the [retry](#) argument.

## Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

**See Also**

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToWindow](#)

**Examples**

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)
```

---

switchToWindow	<i>Change focus to another window.</i>
----------------	--

---

### Description

switchToWindow Change focus to another window.

### Usage

```
switchToWindow(remDr, name, ...)
```

### Arguments

remDr	An object of class "rDriver". A remote driver object see <a href="#">remoteDr</a> .
name	The handle of the window to change focus to.
...	Additional function arguments - Currently passes the <a href="#">retry</a> argument.

### Value

invisible(remDr): An object of class "rDriver" is invisibly returned. A remote driver object see [remoteDr](#). This allows for chaining from this function to other functions that take such an object as an argument. See examples for further details.

### See Also

Other commandContexts functions: [closeWindow](#), [fullscreenWindow](#), [getWindowHandles](#), [getWindowHandle](#), [getWindowPosition](#), [getWindowSize](#), [maximizeWindow](#), [setWindowPosition](#), [setWindowSize](#), [switchToFrame](#), [switchToParentFrame](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% getWindowHandle() # The current window handle
remDr %>% getWindowHandles() # All windows in the session

# Get the window position
remDr %>% getWindowPosition

# Some browsers are still using the old JSON wire end points
remDr %>% getWindowPositionOld

# Get the size of the window
remDr %>% getWindowSize

# Some browsers are still using the old JSON wire end points
# remDr %>% getWindowSizeOld
```

```

# Set the window size
remDr %>% setWindowSize(500, 500)

# Some browsers are still using the old JSON wire end points
remDr %>% setWindowSizeOld(500, 500)

# Set the position of the window
remDr %>% setWindowPositionOld(400, 100)

# Some browsers are still using the old JSON wire end points
# remDr %>% setWindowPositionOld(400, 100)

# Maximise the window
remDr %>% maximizeWindow
# Some browsers are still using the old JSON wire end points
# remDr %>% maximizeWindowOld()

remDr %>% go("http://www.google.com/ncr")
# search for the "R project"

remDr %>% findElement("name", "q") %>% elementSendKeys("R project", key = "enter")

webElem <- remDr %>% findElement("css", "h3.r a")

remDr %>% deleteSession

## End(Not run)

```

---

takeElementScreenshot *takeElementScreenshot*

---

## Description

takeElementScreenshot

## Usage

```

takeElementScreenshot(webElem, file = NULL,
  display = getOption("seleniumPipes_display_screenshot"),
  useViewer = !is.null(getOption("viewer")), returnPNG = FALSE, ...)

```

## Arguments

webElem	An object of class "wElement". A web Element object see <a href="#">wbElement</a> .
file	If not null the decoded PNG is written to file using the string provided here. Defaults to NULL.
display	logical Display the PNG or not (default is set in <code>getOption("seleniumPipes_display_screenshot")</code> ).
useViewer	A viewer to view the PNG. Looks for the RStudio viewer by default.

returnPNG      logical return the decoded PNG. If false (default) webElem is returned to allow chaining.

...              Additional function arguments - Currently passes the [retry](#) argument.

### Value

If returnPNG is FALSE the web Element object is returned and additional chaining is possible. If TRUE then the decoded base64 image is returned see [base64\\_dec](#)

### See Also

Other screenCapture functions: [takeScreenshot](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")
remDr %>% takeScreenshot

# select the search box
searchElem <- remDr %>% findElement("name", "q")
#searchElem %>% takeElementScreenshot()

## End(Not run)
```

---

takeScreenshot	<i>takeScreenshot</i>
----------------	-----------------------

---

### Description

takeScreenshot

### Usage

```
takeScreenshot(remDr, file = NULL,
  display = getOption("seleniumPipes_display_screenshot"),
  useViewer = !is.null(getOption("viewer")), returnPNG = FALSE, ...)
```

### Arguments

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).

file             If not null the decoded PNG is written to file using the string provided here. Defaults to NULL.

display         logical Display the PNG or not (default is set in `getOption("seleniumPipes_display_screenshot")`).

useViewer       A viewer to view the PNG. Looks for the RSudio viewer by default.



returnPNG      logical return the decoded PNG. If false (default) remDr is returned to allow chaining.

...              Additional function arguments - Currently passes the [retry](#) argument.

### Value

If returnPNG is FALSE the remote Driver object is returned and additional chaining is possible. If TRUE then the decoded base64 image is returned see [base64\\_dec](#)

### See Also

Other screenCapture functions: [takeElementScreenshot](#)

### Examples

```
## Not run:
remDr <- remoteDr()
remDr %>% go("http://www.google.com/ncr")
remDr %>% takeScreenshot

# select the search box
searchElem <- remDr %>% findElement("name", "q")
#searchElem %>% takeElementScreenshot()

## End(Not run)
```

---

wbElement

*Create a Web Element*


---

### Description

wbElement Create a Web Element object of class "wElement"

### Usage

```
wbElement(elementId, remDr)
```

### Arguments

elementId      This is a string returned by the web driver that identifies the web element.

remDr            An object of class "rDriver". A remote driver object see [remoteDr](#).

### Value

An object of class "wElement" is returned. This is a web element object that is used in many of the web Element specific functions. Many functions that take a web Element object as input also return the web Element object. This allows chaining of commands. See the examples for chaining in action.

**Examples**

```
## Not run:
remDr <- remoteDr()
webElem <- remDR %>% go("http://www.google.com") %>%
  findElement("name", "q")
# print the webElement
webElem

# send keys to the web Element
webElem %>% elementSendKeys("R project", key = "enter")

# close browser
remDr %>% deleteSession()

## End(Not run)
```

# Index

- \* **datasets**
  - selKeys, 74
- acceptAlert, 3, 13, 36, 75
- acceptAlertOld, 4, 14, 22, 25, 37, 53, 56, 58, 60, 66, 76, 80, 82
- addCookie, 5, 10, 11, 38, 48
- back, 6, 32, 39, 51, 61, 69
- base64\_dec, 88, 89
- checkResponse, 7
- closeWindow, 8, 33, 52, 54, 56, 59, 65, 78, 81, 83, 85, 86
- content, 18, 69
- deleteAllCookies, 6, 10, 11, 38, 48
- deleteCookie, 6, 10, 11, 38, 48
- deleteSession, 12, 67, 77
- dismissAlert, 4, 13, 36, 75
- dismissAlertOld, 5, 14, 22, 25, 37, 53, 56, 58, 60, 66, 76, 80, 82
- elementClear, 15, 16, 17
- elementClick, 15, 16, 17
- elementSendKeys, 15, 16, 17
- errorContent, 8, 18
- errorResponse, 8, 19
- executeAsyncScript, 19, 23, 49
- executeAsyncScriptOld, 5, 14, 21, 25, 37, 53, 56, 58, 60, 66, 76, 80, 82
- executeScript, 20, 22, 49
- executeScriptOld, 5, 14, 22, 24, 37, 53, 56, 58, 60, 66, 76, 80, 82
- findElement, 25, 28, 29, 31, 35
- findElementFromElement, 26, 27, 29, 31, 35
- findElements, 26, 28, 28, 31, 35
- findElementsFromElement, 26, 28, 29, 30, 35
- forward, 7, 32, 39, 51, 61, 69
- fullscreenWindow, 9, 33, 52, 54, 56, 59, 65, 78, 81, 83, 85, 86
- getActiveElement, 26, 28, 29, 31, 34
- getAlertText, 4, 13, 36, 75
- getAlertTextOld, 5, 14, 22, 25, 37, 53, 56, 58, 60, 66, 76, 80, 82
- getAllCookies, 6, 10, 11, 38, 48
- getCurrentUrl, 7, 32, 39, 51, 61, 69
- getElementAttribute, 40, 41, 43–45, 47, 62, 63
- getElementCssValue, 40, 41, 43–45, 47, 62, 63
- getElementProperty, 40, 41, 42, 44, 45, 47, 62, 63
- getElementRect, 40, 41, 43, 44, 45, 47, 62, 63
- getElementTagName, 40, 41, 43, 44, 45, 47, 62, 63
- getElementText, 40, 41, 43–45, 46, 62, 63
- getNamedCookie, 6, 10, 11, 38, 47
- getPageSource, 20, 23, 49
- getTitle, 7, 32, 39, 50, 61, 69
- getWindowHandle, 9, 33, 52, 54, 56, 59, 65, 78, 81, 83, 85, 86
- getWindowHandleOld, 5, 14, 22, 25, 37, 53, 56, 58, 60, 66, 76, 80, 82
- getWindowHandles, 9, 33, 52, 54, 56, 59, 65, 78, 81, 83, 85, 86
- getWindowHandlesOld, 5, 14, 22, 25, 37, 53, 55, 58, 60, 66, 76, 80, 82
- getWindowPosition, 9, 33, 52, 54, 56, 59, 65, 78, 81, 83, 85, 86
- getWindowPositionOld, 5, 14, 22, 25, 37, 53, 56, 57, 60, 66, 76, 80, 82
- getWindowSize, 9, 33, 52, 54, 56, 58, 65, 78, 81, 83, 85, 86
- getWindowSizeOld, 5, 14, 22, 25, 37, 53, 56, 58, 60, 66, 76, 80, 82
- go, 7, 32, 39, 51, 61, 69

- isElementEnabled, [40](#), [41](#), [43–45](#), [47](#), [62](#), [63](#)
- isElementSelected, [40](#), [41](#), [43–45](#), [47](#), [62](#), [63](#)
  
- maximizeWindow, [9](#), [33](#), [52](#), [54](#), [56](#), [59](#), [64](#), [78](#),  
[81](#), [83](#), [85](#), [86](#)
- maximizeWindowOld, [5](#), [14](#), [22](#), [25](#), [37](#), [53](#), [56](#),  
[58](#), [60](#), [66](#), [76](#), [80](#), [82](#)
  
- newSession, [12](#), [67](#), [72](#), [77](#)
  
- options, [73](#)
  
- performActions, [68](#), [71](#)
  
- queryDriver, [68](#)
  
- read\_html, [49](#)
- refresh, [7](#), [32](#), [39](#), [51](#), [61](#), [69](#)
- releasingActions, [68](#), [70](#)
- remoteDr, [3–8](#), [10–14](#), [19](#), [21](#), [22](#), [24](#), [25](#), [29](#),  
[32](#), [33](#), [35–39](#), [48](#), [49](#), [51–56](#), [58–61](#),  
[64–71](#), [71](#), [74–84](#), [86](#), [88](#), [89](#)
- retry, [3–5](#), [7](#), [8](#), [10–16](#), [19](#), [22](#), [24](#), [25](#), [27](#), [29](#),  
[30](#), [32](#), [33](#), [35–41](#), [43–46](#), [48](#), [49](#),  
[51–56](#), [58–64](#), [66–70](#), [72](#), [75–78](#), [80](#),  
[82–84](#), [86](#), [88](#), [89](#)
  
- seleniumPipes, [73](#)
- seleniumPipes-package (seleniumPipes),  
[73](#)
- selKeys, [17](#), [74](#)
- sendAlertText, [4](#), [13](#), [36](#), [74](#)
- sendAlertTextOld, [5](#), [14](#), [22](#), [25](#), [37](#), [53](#), [56](#),  
[58](#), [60](#), [66](#), [76](#), [80](#), [82](#)
- setTimeout, [12](#), [67](#), [77](#)
- setWindowPosition, [9](#), [33](#), [52](#), [54](#), [56](#), [59](#), [65](#),  
[78](#), [81](#), [83](#), [85](#), [86](#)
- setWindowPositionOld, [5](#), [14](#), [22](#), [25](#), [37](#), [53](#),  
[56](#), [58](#), [60](#), [66](#), [76](#), [79](#), [82](#)
- setWindowSize, [9](#), [33](#), [52](#), [54](#), [56](#), [59](#), [65](#), [78](#),  
[80](#), [83](#), [85](#), [86](#)
- setWindowSizeOld, [5](#), [14](#), [22](#), [25](#), [37](#), [53](#), [56](#),  
[58](#), [60](#), [66](#), [76](#), [80](#), [82](#)
- switchToFrame, [9](#), [33](#), [52](#), [54](#), [56](#), [59](#), [65](#), [78](#),  
[81](#), [83](#), [85](#), [86](#)
- switchToParentFrame, [9](#), [33](#), [52](#), [54](#), [56](#), [59](#),  
[65](#), [78](#), [81](#), [83](#), [84](#), [86](#)
- switchToWindow, [9](#), [33](#), [52](#), [54](#), [56](#), [59](#), [65](#), [78](#),  
[81](#), [83](#), [85](#), [86](#)
  
- takeElementScreenshot, [73](#), [87](#), [89](#)
- takeScreenshot, [73](#), [88](#), [88](#)
  
- VERB, [8](#), [19](#), [69](#)
  
- wbElement, [15–17](#), [26](#), [27](#), [29–31](#), [35](#), [40–42](#),  
[44–46](#), [62](#), [63](#), [87](#), [89](#)