Package: seleniumPipes (via r-universe)

September 4, 2024

Type Package

Title R Client Implementing the W3C WebDriver Specification

Version 0.3.9

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)

Imports 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

LazyData true

Repository https://johndharrison.r-universe.dev

RemoteUrl https://github.com/johndharrison/seleniumpipes

RemoteRef HEAD

RemoteSha 53a268dc1496af84278019213f8e556df041034a

Contents

`acceptAlert` *Accept alert*

Description

`acceptAlert` accept a JavaScript alert

Usage

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

Arguments

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`...`	Additonal 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: 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.
...	Additonal 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 *Add a specific cookie.*

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 remoteDr.
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.
...	Additonal 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: 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.
...	Additonal 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)
```

---

checkResponse *Check the response from remote server*

Description

checkResponse checks the response from a remote web driver and checks against known errors. uses statusCodes in sysdata.rda see seleniumPipes:::statusCodes

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 *Close the current window.*

Description

closeWindow Close the current window.

Usage

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

Arguments

remDr	An object of class "rDriver". A remote driver object see remoteDr.
...	Additonal 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`.
`...`	Additonal 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` *Delete a given cookie.*

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 `remoteDr`.
`name`	character: The name of the cookie; may not be null or an empty string
`...`	Additonal 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`, `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` *Delete the session.*

Description

`deleteSession` Delete the session.

Usage

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

Arguments

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`...`	Additonal function arguments - Currently passes the `retry` 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: getTimeouts, newSession, setTimeouts, status

Examples

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

  # open a broswer
  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`.
`...`	Additonal 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 *Dismiss Alert*

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 `remoteDr`.
`...`	Additonal 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: 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 *Clear an elements text value.*

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 wbElement.
...	Additonal 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: elementClick, elementSendKeys

Examples

```
## Not run:
  # start a brower
  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.
...	Additonal 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 brower
  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 brower
  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 *Return the response from remote webdriver*

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", "sdofnsdofk")
errorResponse()

## End(Not run)
```

---

executeAsyncScript *Execute JavaScript asynchronously on browser.*

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 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.
...	Additonal 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 assummed 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

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.
...	Additonal 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 assummed 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 *Execute JavaScript on browser.*

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.
...	Additonal 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 assummed 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

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.
...	Additonal 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 assummed 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` *Search for an element on the page*

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 `remoteDr`.
`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.
`...`	Additonal function arguments - Currently passes the `retry` 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 wbElement.
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.
...	Additonal function arguments - Currently passes the retry 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 `remoteDr`.
`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.
`...`	Additonal function arguments - Currently passes the `retry` 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 `wbElement`.
`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.
`...`	Additonal function arguments - Currently passes the `retry` 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.
...	Additonal 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` *Make current window full-screen*

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 `remoteDr`.
`...`	Additonal 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`, `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)
```

`getActiveElement` *Get the element on the page that currently has focus.*

Description

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

Usage

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

Arguments

remDr	An object of class "rDriver". A remote driver object see remoteDr.
...	Additonal function arguments - Currently passes the retry argument.

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, 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 *Get alert text*

Description

getAlertText Get the text from a JavaScript alert.

Usage

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

Arguments

remDr	An object of class "rDriver". A remote driver object see remoteDr.
...	Additonal function arguments - Currently passes the retry 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` *Get alert text*

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 `remoteDr`.
`...`	Additonal function arguments - Currently passes the `retry` 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 *Get all current domain cookies*

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 remoteDr.
...	Additonal function arguments - Currently passes the retry 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` *Retrieve the URL of the current page.*

Description

`getCurrentUrl` Retrieve the URL of the current page.

Usage

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

Arguments

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`...`	Additonal function arguments - Currently passes the `retry` 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 `wbElement`.
`attribute`	The attribute to query as a character string.
`...`	Additonal function arguments - Currently passes the `retry` 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 `wbElement`.
`propertyName`	The property to query as a character string
`...`	Additonal function arguments - Currently passes the `retry` 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
...	Additonal 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 *Return the dimensions and coordinates of an element*

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 wbElement.
...	Additonal function arguments - Currently passes the retry 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`.
`...`	Additonal 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` *Returns the visible text for the element.*

Description

`getElementText` Returns the visible text for the element.

Usage

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

Arguments

`webElem`	An object of class "wElement". A web Element object see `wbElement`.
`...`	Additonal function arguments - Currently passes the `retry` 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 remoteDr.
name	character: The name of the cookie; may not be null or an empty string
...	Additonal function arguments - Currently passes the retry 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 *Get source of last page.*

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 remoteDr.
...	Additonal function arguments - Currently passes the retry 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)
```

getTimeouts *Get amount of time that a particular type of operation can execute in.*

Description

getTimeouts The Get Timeout command gets timeouts associated with the current session.

Usage

```
getTimeouts(remDr, type = "page load", ...)
```

Arguments

remDr	An object of class "rDriver". A remote driver object see remoteDr.
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.
...	Additonal function arguments - Currently passes the retry argument.

Value

ret2 : res$value NEED TO FILL IN specifics for each function

See Also

Other sessions functions: deleteSession, newSession, setTimeouts, status

Examples

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

  # open a broswer
  remDr %>% newSession

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

  # close Session
  remDr %>% deleteSession


## End(Not run)
```

getTitle *Get the current page title.*

Description

getTitle Get the current page title.

Usage

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

Arguments

remDr	An object of class "rDriver". A remote driver object see remoteDr.
...	Additonal function arguments - Currently passes the retry 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` *get current window handle*

Description

`getWindowHandle` Retrieve the current window handle.

Usage

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

Arguments

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`...`	Additonal function arguments - Currently passes the `retry` 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`.
`...`	Additonal 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` *Get all window handles.*

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 `remoteDr`.
`...`	Additonal function arguments - Currently passes the `retry` 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 remoteDr.
...	Additonal function arguments - Currently passes the retry 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)
```

---

getWindowPosition *Get current window position*

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

remDr	An object of class "rDriver". A remote driver object see remoteDr.
...	Additonal function arguments - Currently passes the retry 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

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`handle`	Handle of the window to query. If handle = "current" (the default) the current window is used.
`...`	Additonal function arguments - Currently passes the `retry` 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)
```

getWindowSize *getWindowSize*

Description

getWindowSize

Usage

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

Arguments

remDr	An object of class "rDriver". A remote driver object see remoteDr.
...	Additonal 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 `remoteDr`.
`handle`	Handle of the window to query. If handle = "current" (the default) the current window is used.
`...`	Additonal 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 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 remoteDr.
url	The URL to navigate to.
...	Additonal 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, 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`.
`...`	Additonal function arguments - Currently passes the `retry` argument.

Value

A logical value is returned indicating whether the elemnent 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.
...	Additonal function arguments - Currently passes the retry argument.

Value

A logical value is returned indicating whether the elemnent 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` *Maximize the current window.*

Description

`maximizeWindow` Maximize the current if not already maximized.

Usage

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

Arguments

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`...`	Additonal 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`, `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 `remoteDr`.
`handle`	Handle of the window to query. If handle = "current" (the default) the current window is used.
`...`	Additonal 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: 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 *Create a new session.*

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 remoteDr.
...	Additonal 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 sessions functions: deleteSession, getTimeouts, setTimeouts, status

Examples

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

  # open a broswer
  remDr %>% newSession

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

  # close Session
  remDr %>% deleteSession


## End(Not run)
```

---

performActions *Not currently implemented*

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 remoteDr.
...	Additonal 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 interactions functions: releasingActions

Examples

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

## End(Not run)
```

`queryDriver` *Send a query to remote Driver.*

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 `VERB`
`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` *Refresh the current page.*

Description

`refresh` Refresh the current page.

Usage

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

Arguments

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`...`	Additonal 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`, `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` *Not currently implemented*

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 `remoteDr`.
`...`	Additonal 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 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 chrome|firefox|internet explorer|iphone. 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|XP|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. n 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 newSession
...	Pass addional arguments to newSession. Currently used to pass retry

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 trys 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 trys 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)
rList <- list(noTry = 5, delay = 10)
remDr %>% findElement("id", "i am not here", retry = rList)
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 miliseconds = 5 seconds.
- `seleniumPipes_SL`: A logical value which acts as a flag indicating whether SauiceLabs 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 SauceLab tests on the package.

selKeys *Selenium key mappings*

Description

This data set contains a list of selenium key mappings. The key mappings are outlined at https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidelementidvalue. 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 *Send text to alert*

Description

`sendAlertText` Send keystrokes to JavaScript prompt() dialog

Usage

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

Arguments

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`text`	A character vector of length 1. In other words a string. The text is passed to the JavaScript alert
`...`	Additonal 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, 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 *Send text to alert*

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 remoteDr.
text	A character vector of length 1. In other words a string. The text is passed to the JavaScript alert
...	Additonal 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: 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)
```

---

setTimeouts *Configure the amount of time that a particular type of operation can execute*

Description

getTimeouts 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

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

Arguments

remDr	An object of class "rDriver". A remote driver object see remoteDr.
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.
...	Additonal 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 sessions functions: deleteSession, getTimeouts, newSession, status

Examples

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

  # open a broswer
  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 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.
...	Additonal 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, 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.
`...`	Additonal 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: 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 *Change the size of the current window.*

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 remoteDr.
width	integer The new window width.
height	integer The new window height.
...	Additonal 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, 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 remoteDr.
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.
...	Additonal 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: 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)
```

---

status *Get remote end status.*

Description

status The Status command returns information about whether a remote end is in a state in which it can create a new session. This is represented by the ready property of the response body, which has a value of false if attempting to create a session at the current time would fail.

Usage

```
status(remDr, ...)
```

Arguments

remDr	An object of class "rDriver". A remote driver object see remoteDr.
...	Additonal function arguments - Currently passes the retry argument.

Value

ret2 : res$value NEED TO FILL IN specifics for each function

See Also

Other sessions functions: deleteSession, getTimeouts, newSession, setTimeouts

Examples

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

  # open a broswer
  remDr %>% newSession

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

  # close Session
  remDr %>% deleteSession


## End(Not run)
```

---

switchToFrame *Change focus to another frame on the page.*

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 remoteDr.
Id	string|number|null|WebElement Identifier for the frame to change focus to.
...	Additonal 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, 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`.
`...`	Additonal 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` *Change focus to another window.*

Description

`switchToWindow` Change focus to another window.

Usage

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

Arguments

`remDr`	An object of class "rDriver". A remote driver object see `remoteDr`.
`name`	The handle of the window to change focus to.
`...`	Additonal 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, 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 wbElement.
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) webElem is returned to allow chaining.
...	Additonal function arguments - Currently passes the retry argument.

Value

If returnPNG is FALSE the web Element object is returned and additonal 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` *takeScreenshot*

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.
`...`	Additonal function arguments - Currently passes the `retry` argument.

Value

If `returnPNG` is FALSE the remote Driver object is returned and additonal 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