

Package ‘chromote’

April 19, 2022

Title Headless Chrome Web Browser Interface

Version 0.1.0

Description An implementation of the 'Chrome DevTools Protocol', for controlling a headless Chrome web browser.

License GPL-2

Encoding UTF-8

SystemRequirements Google Chrome or other Chromium-based browser.
chromium: chromium (rpm) or chromium-browser (deb)

Imports curl, jsonlite, websocket (>= 1.2.0), processx, R6, later (>= 1.1.0), promises (>= 1.1.1), magrittr, rlang, fastmap

Suggests testthat (>= 3.0.0), showimage

RoxygenNote 7.1.2

URL <https://github.com/rstudio/chromote>

BugReports <https://github.com/rstudio/chromote/issues>

Config/testthat/edition 3

Config/Needs/website tidyverse/tidytemplate

Language en-US

NeedsCompilation no

Author Winston Chang [aut, cre],
Barret Schloerke [aut],
RStudio [cph, fnd]

Maintainer Winston Chang <winston@rstudio.com>

Repository CRAN

Date/Publication 2022-04-19 09:02:29 UTC

R topics documented:

Browser	2
Chrome	3

ChromeRemote	4
Chromote	5
ChromoteSession	9
default_chrome_args	20
default_chromote_object	22
find_chrome	22

Index	23
--------------	-----------

Browser	<i>Browser base class</i>
---------	---------------------------

Description

Browser base class

Browser base class

Details

Base class for browsers like Chrome, Chromium, etc. Defines the interface used by various browser implementations. It can represent a local browser process or one running remotely.

The `initialize()` method of an implementation should set `private$host` and `private$port`. If the process is local, the `initialize()` method should also set `private$process`.

Methods

Public methods:

- `Browser$is_local()`
- `Browser$get_process()`
- `Browser$get_host()`
- `Browser$get_port()`
- `Browser$close()`
- `Browser$clone()`

Method `is_local()`: Is local browser? Returns TRUE if the browser is running locally, FALSE if it's remote.

Usage:

`Browser$is_local()`

Method `get_process()`: Browser process

Usage:

`Browser$get_process()`

Method `get_host()`: Browser Host

Usage:

`Browser$get_host()`

Method `get_port()`: Browser port

Usage:

`Browser$get_port()`

Method `close()`: Close the browser

Usage:

`Browser$close()`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`Browser$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Chrome

Local Chrome process

Description

Local Chrome process

Local Chrome process

Details

This is a subclass of [Browser](#) that represents a local browser. It extends the [Browser](#) class with a [processx::process](#) object, which represents the browser's system process.

Super class

[chromote::Browser](#) -> Chrome

Methods

Public methods:

- [Chrome\\$new\(\)](#)
- [Chrome\\$get_path\(\)](#)
- [Chrome\\$clone\(\)](#)

Method `new()`: Create a new Chrome object.

Usage:

`Chrome$new(path = find_chrome(), args = get_chrome_args())`

Arguments:

`path` Location of chrome installation

args A character vector of command-line arguments passed when initializing Chrome. Single on-off arguments are passed as single values (e.g. "--disable-gpu"), arguments with a value are given with a nested character vector (e.g. c("--force-color-profile", "srgb")). See [here](#) for a list of possible arguments. Defaults to `get_chrome_args()`.

Returns: A new Chrome object.

Method `get_path()`: Browser application path

Usage:

`Chrome$get_path()`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`Chrome$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

[get_chrome_args\(\)](#)

ChromeRemote

Remote Chrome process

Description

Remote Chrome process

Remote Chrome process

Super class

[chromote::Browser](#) -> ChromeRemote

Methods

Public methods:

- [ChromeRemote\\$new\(\)](#)
- [ChromeRemote\\$clone\(\)](#)

Method `new()`: Create a new ChromeRemote object.

Usage:

`ChromeRemote$new(host, port)`

Arguments:

`host` A string that is a valid IPv4 or IPv6 address. "0.0.0.0" represents all IPv4 addresses and "::/0" represents all IPv6 addresses.

`port` A number or integer that indicates the server port.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ChromeRemote$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Chromote

Chromote class

Description

Chromote class

Chromote class

Details

This class represents the browser as a whole.

A Chromote object represents the browser as a whole, and it can have multiple *targets*, which each represent a browser tab. In the Chrome DevTools Protocol, each target can have one or more debugging *sessions* to control it. A ChromoteSession object represents a single *session*.

A Chromote object can have any number of ChromoteSession objects as children. It is not necessary to create a Chromote object manually. You can simply call:

```
b <- ChromoteSession$new()
```

and it will automatically create a Chromote object if one has not already been created. The **chromote** package will then designate that Chromote object as the *default* Chromote object for the package, so that any future calls to `ChromoteSession$new()` will automatically use the same Chromote. This is so that it doesn't start a new browser for every ChromoteSession object that is created.

Public fields

`default_timeout` Default timeout in seconds for **chromote** to wait for a Chrome DevTools Protocol response.

`protocol` Dynamic protocol implementation. For expert use only!

Methods

Public methods:

- [Chromote\\$new\(\)](#)
- [Chromote\\$view\(\)](#)
- [Chromote\\$get_auto_events\(\)](#)
- [Chromote\\$get_child_loop\(\)](#)

- `Chromote$wait_for()`
- `Chromote$new_session()`
- `Chromote$get_sessions()`
- `Chromote$register_session()`
- `Chromote$send_command()`
- `Chromote$invoke_event_callbacks()`
- `Chromote$debug_messages()`
- `Chromote$debug_log()`
- `Chromote$url()`
- `Chromote$is_active()`
- `Chromote$get_browser()`
- `Chromote$close()`

Method `new()`:

Usage:

```
Chromote$new(browser = Chrome$new(), multi_session = TRUE, auto_events = TRUE)
```

Arguments:

`browser` A [Browser](#) object

`multi_session` Should multiple sessions be allowed?

`auto_events` If TRUE, enable automatic event enabling/disabling; if FALSE, disable automatic event enabling/disabling.

Method `view()`: Display the current session in the browser

If a [Chrome](#) browser is being used, this method will open a new tab using your [Chrome](#) browser. When not using a [Chrome](#) browser, set `options(browser=)` to change the default behavior of `browseURL()`.

Usage:

```
Chromote$view()
```

Method `get_auto_events()`: `auto_events` value.

For internal use only.

Usage:

```
Chromote$get_auto_events()
```

Method `get_child_loop()`: Local **later** loop.

For expert async usage only.

Usage:

```
Chromote$get_child_loop()
```

Method `wait_for()`: Wait until the promise resolves

Blocks the R session until the promise (`p`) is resolved. The loop from `$get_child_loop()` will only advance just far enough for the promise to resolve.

Usage:

```
Chromote$wait_for(p)
```

Arguments:

p A promise to resolve.

Method `new_session()`: Create a new tab / window

Usage:

```
Chromote$new_session(width = 992, height = 1323, targetId = NULL, wait_ = TRUE)
```

Arguments:

width, height Width and height of the new window.

targetId **Target** ID of an existing target to attach to. When a targetId is provided, the width and height arguments are ignored. If NULL (the default) a new target is created and attached to, and the width and height arguments determine its viewport size.

wait_ If FALSE, return a `promises::promise()` of a new `ChromoteSession` object. Otherwise, block during initialization, and return a `ChromoteSession` object directly.

Method `get_sessions()`: Retrieve all `ChromoteSession` objects

Usage:

```
Chromote$get_sessions()
```

Returns: A list of `ChromoteSession` objects

Method `register_session()`: Register `ChromoteSession` object

Usage:

```
Chromote$register_session(session)
```

Arguments:

session A `ChromoteSession` object

For internal use only.

Method `send_command()`: Send command through Chrome DevTools Protocol.

For expert use only.

Usage:

```
Chromote$send_command(
  msg,
  callback = NULL,
  error = NULL,
  timeout = NULL,
  sessionId = NULL
)
```

Arguments:

msg A JSON-serializable list containing method, and params.

callback Method to run when the command finishes successfully.

error Method to run if an error occurs.

timeout Number of milliseconds for Chrome DevTools Protocol execute a method.

sessionId Determines which `ChromoteSession` with the corresponding to send the command to.

Method `invoke_event_callbacks()`: Immediately call all event callback methods.
For internal use only.

Usage:

```
Chromote$invoke_event_callbacks(event, params)
```

Arguments:

event A single event string

params A list of parameters to pass to the event callback methods.

Method `debug_messages()`: Enable or disable message debugging
If enabled, R will print out the

Usage:

```
Chromote$debug_messages(value = NULL)
```

Arguments:

value If TRUE, enable debugging. If FALSE, disable debugging.

Method `debug_log()`: Submit debug log message

Usage:

```
Chromote$debug_log(...)
```

Arguments:

... Arguments pasted together with `paste0(..., collapse = "")`.

Examples:

```
\dontrun{b <- ChromoteSession$new()
b$parent$debug_messages(TRUE)
b$Page$navigate("https://www.r-project.org/")
#> SEND {"method": "Page.navigate", "params": {"url": "https://www.r-project.org/"} | __truncated__}
# Turn off debug messages
b$parent$debug_messages(FALSE)}
```

Method `url()`: Create url for a given path

Usage:

```
Chromote$url(path = NULL)
```

Arguments:

path A path string to append to the host and port

Method `is_active()`: Retrieve active status Once initialized, the value returned is TRUE. If `$close()` has been called, this value will be FALSE.

Usage:

```
Chromote$is_active()
```

Method `get_browser()`: Retrieve `Browser` object

Usage:

```
Chromote$get_browser()
```

Method `close()`: Close the `Browser` object

Usage:

```
Chromote$close()
```


Examples

```
## -----
## Method `Chromote$debug_log`
## -----

## Not run: b <- ChromoteSession$new()
b$parent$debug_messages(TRUE)
b$page$navigate("https://www.r-project.org/")
#> SEND {"method":"Page.navigate","params":{"url":"https://www.r-project.org/"}| __truncated__}
# Turn off debug messages
b$parent$debug_messages(FALSE)
## End(Not run)
```

ChromoteSession	<i>ChromoteSession class</i>
-----------------	------------------------------

Description

ChromoteSession class

ChromoteSession class

Public fields

parent [Chromote](#) object

default_timeout Default timeout in seconds for **chromote** to wait for a Chrome DevTools Protocol response.

protocol Dynamic protocol implementation. For expert use only!

Methods

Public methods:

- [ChromoteSession\\$new\(\)](#)
- [ChromoteSession\\$view\(\)](#)
- [ChromoteSession\\$close\(\)](#)
- [ChromoteSession\\$screenshot\(\)](#)
- [ChromoteSession\\$screenshot_pdf\(\)](#)
- [ChromoteSession\\$new_session\(\)](#)
- [ChromoteSession\\$get_session_id\(\)](#)
- [ChromoteSession\\$wait_for\(\)](#)
- [ChromoteSession\\$debug_log\(\)](#)
- [ChromoteSession\\$get_child_loop\(\)](#)
- [ChromoteSession\\$send_command\(\)](#)
- [ChromoteSession\\$get_auto_events\(\)](#)
- [ChromoteSession\\$invoke_event_callbacks\(\)](#)

- `ChromoteSession$mark_closed()`
- `ChromoteSession$is_active()`
- `ChromoteSession$init_promise()`

Method `new()`: Create a new `ChromoteSession` object.

Usage:

```
ChromoteSession$new(
  parent = default_chromote_object(),
  width = 992,
  height = 1323,
  targetId = NULL,
  wait_ = TRUE,
  auto_events = NULL
)
```

Arguments:

`parent` `Chromote` object to use; defaults to `default_chromote_object()`

`width` Width, in pixels, of the Target to create if `targetId` is NULL

`height` Height, in pixels, of the Target to create if `targetId` is NULL

`targetId` **Target** ID of an existing target to attach to. When a `targetId` is provided, the width and height arguments are ignored. If NULL (the default) a new target is created and attached to, and the width and height arguments determine its viewport size.

`wait_` If FALSE, return a `promises::promise()` of a new `ChromoteSession` object. Otherwise, block during initialization, and return a `ChromoteSession` object directly.

`auto_events` If NULL (the default), use the `auto_events` setting from the parent `Chromote` object. If TRUE, enable automatic event enabling/disabling; if FALSE, disable automatic event enabling/disabling.

Returns: A new `ChromoteSession` object.

Examples:

```
\dontrun{# Create a new `ChromoteSession` object.
b <- ChromoteSession$new()

# Create a ChromoteSession with a specific height,width
b <- ChromoteSession$new(height = 1080, width = 1920)

# Navigate to page
b$Page$navigate("http://www.r-project.org/")

# View current chromote session
if (interactive()) b$view()}
```

Method `view()`: Display the current session in the `Chromote` browser.

If a `Chrome` browser is being used, this method will open a new tab using your `Chrome` browser. When not using a `Chrome` browser, set `options(browser=)` to change the default behavior of `browseURL()`.

Usage:

```
ChromoteSession$view()
```

Examples:

```
\dontrun{# Create a new `ChromoteSession` object.
b <- ChromoteSession$new()
```

```
# Navigate to page
b$page$navigate("http://www.r-project.org/")
```

```
# View current chromote session
if (interactive()) b$view()}
```

Method `close()`: Close the Chromote session.

Usage:

```
ChromoteSession$close(wait_ = TRUE)
```

Arguments:

`wait_` If FALSE, return a `promises::promise()` that will resolve when the `ChromoteSession` is closed. Otherwise, block until the `ChromoteSession` has closed.

Examples:

```
\dontrun{# Create a new `ChromoteSession` object.
b <- ChromoteSession$new()
```

```
# Navigate to page
b$page$navigate("http://www.r-project.org/")
```

```
# Close current chromote session
b$close()}
```

Method `screenshot()`: Take a PNG screenshot

Usage:

```
ChromoteSession$screenshot(
  filename = "screenshot.png",
  selector = "html",
  cliprect = NULL,
  region = c("content", "padding", "border", "margin"),
  expand = NULL,
  scale = 1,
  show = FALSE,
  delay = 0.5,
  wait_ = TRUE
)
```

Arguments:

`filename` File path of where to save the screenshot.

`selector` CSS selector to use for the screenshot.

`cliprect` A list containing x, y, width, and height. See [Page.Viewport](#) for more information. If provided, `selector` and `expand` will be ignored. To provide a scale, use the `scale` parameter.

region CSS region to use for the screenshot.

expand Extra pixels to expand the screenshot. May be a single value or a numeric vector of top, right, bottom, left values.

scale Page scale factor

show If TRUE, the screenshot will be displayed in the viewer.

delay The number of seconds to wait before taking the screenshot after resizing the page. For complicated pages, this may need to be increased.

wait_ If FALSE, return a `promises::promise()` that will resolve when the `ChromoteSession` has saved the screenshot. Otherwise, block until the `ChromoteSession` has saved the screenshot.

Examples:

```
\dontrun{# Create a new `ChromoteSession` object.
b <- ChromoteSession$new()

# Navigate to page
b$page$navigate("http://www.r-project.org/")

# Take screenshot
tmppngfile <- tempfile(fileext = ".png")
is_interactive <- interactive() # Display screenshot if interactive
b$screenshot(tmppngfile, show = is_interactive)

# Show screenshot file info
unlist(file.info(tmppngfile))

# Take screenshot using a selector
sidebar_file <- tempfile(fileext = ".png")
b$screenshot(sidebar_file, selector = ".sidebar", show = is_interactive)

# -----
# Take screenshots in parallel

urls <- c(
  "https://www.r-project.org/",
  "https://github.com/",
  "https://news.ycombinator.com/"
)
# Helper method that:
# 1. Navigates to the given URL
# 2. Waits for the page loaded event to fire
# 3. Takes a screenshot
# 4. Prints a message
# 5. Close the ChromoteSession
screenshot_p <- function(url, filename = NULL) {
  if (is.null(filename)) {
    filename <- gsub("^.*://", "", url)
  }
}
```

```

    filename <- gsub("/", "_", filename)
    filename <- gsub("\\\\.", "_", filename)
    filename <- sub("_$", "", filename)
    filename <- paste0(filename, ".png")
  }

  b2 <- b$new_session()
  b2$Page$navigate(url, wait_ = FALSE)
  b2$Page$loadEventFired(wait_ = FALSE)$
    then(function(value) {
      b2$screenshot(filename, wait_ = FALSE)
    })$
    then(function(value) {
      message(filename)
    })$
    finally(function() {
      b2$close()
    })
  }

  # Take multiple screenshots simultaneously
  ps <- lapply(urls, screenshot_p)
  pa <- promises::promise_all(.list = ps)$then(function(value) {
    message("Done!")
  })

  # Block the console until the screenshots finish (optional)
  b$wait_for(pa)
  #> www_r-project_org.png
  #> github_com.png
  #> news_ycombinator_com.png
  #> Done!}

```

Method `screenshot_pdf()`: Take a PDF screenshot

Usage:

```

ChromoteSession$screenshot_pdf(
  filename = "screenshot.pdf",
  pagesize = "letter",
  margins = 0.5,
  units = c("in", "cm"),
  landscape = FALSE,
  display_header_footer = FALSE,
  print_background = FALSE,
  scale = 1,
  wait_ = TRUE
)

```

Arguments:

`filename` File path of where to save the screenshot.

pagesize A single character value in the set "letter", "legal", "tabloid", "ledger" and "a0" through "a1". Or a numeric vector `c(width,height)` specifying the page size.
margins A numeric vector `c(top,right,bottom,left)` specifying the page margins.
units Page and margin size units. Either "in" or "cm" for inches and centimeters respectively.
landscape Paper orientation.
display_header_footer Display header and footer.
print_background Print background graphics.
scale Page scale factor.
wait_ If FALSE, return a `promises::promise()` that will resolve when the `ChromoteSession` has saved the screenshot. Otherwise, block until the `ChromoteSession` has saved the screenshot.

Examples:

```

\dontrun{# Create a new `ChromoteSession` object.
b <- ChromoteSession$new()

# Navigate to page
b$page$navigate("http://www.r-project.org/")

# Take screenshot
tmpdffile <- tempfile(fileext = ".pdf")
b$screenshot_pdf(tmpdffile)

# Show PDF file info
unlist(file.info(tmpdffile))}
  
```

Method `new_session()`: Create a new tab / window

Usage:

```

ChromoteSession$new_session(
  width = 992,
  height = 1323,
  targetId = NULL,
  wait_ = TRUE
)
  
```

Arguments:

width, height Width and height of the new window.
targetId **Target** ID of an existing target to attach to. When a `targetId` is provided, the width and height arguments are ignored. If `NULL` (the default) a new target is created and attached to, and the width and height arguments determine its viewport size.
wait_ If FALSE, return a `promises::promise()` that will resolve when the `ChromoteSession` has created a new session. Otherwise, block until the `ChromoteSession` has created a new session.

Examples:

```

\dontrun{b1 <- ChromoteSession$new()
b1$page$navigate("http://www.google.com")
b2 <- b1$new_session()}
  
```

```

b2$page$navigate("http://www.r-project.org/")
b1$runtime$evaluate("window.location", returnByValue = TRUE)$result$value$href
#> [1] "https://www.google.com/"
b2$runtime$evaluate("window.location", returnByValue = TRUE)$result$value$href
#> [1] "https://www.r-project.org/"

```

Method `get_session_id()`: Retrieve the session id

Usage:

```
ChromoteSession$get_session_id()
```

Examples:

```

\dontrun{b <- ChromoteSession$new()
b$get_session_id()
#> [1] "05764F1D439F4292497A21C6526575DA"}

```

Method `wait_for()`: Wait for a Chromote Session to finish. This method will block the R session until the provided promise resolves. The loop from `$get_child_loop()` will only advance just far enough for the promise to resolve.

Usage:

```
ChromoteSession$wait_for(p)
```

Arguments:

`p` A promise to resolve.

Examples:

```

\dontrun{b <- ChromoteSession$new()

# Async with promise
p <- b$browser$getVersion(wait_ = FALSE)
p$then(str)

# Async with callback
b$browser$getVersion(wait_ = FALSE, callback_ = str)}

```

Method `debug_log()`: Send a debug log message to the parent [Chromote](#) object

Usage:

```
ChromoteSession$debug_log(...)
```

Arguments:

`...` Arguments pasted together with `paste0(..., collapse = "")`.

Examples:

```

\dontrun{b <- ChromoteSession$new()
b$parent$debug_messages(TRUE)
b$page$navigate("https://www.r-project.org/")
#> SEND {"method": "Page.navigate", "params": {"url": "https://www.r-project.org/"} | __truncated__}
# Turn off debug messages
b$parent$debug_messages(FALSE)}

```

Method `get_child_loop()`: **later** loop.

For expert async usage only.

Usage:

```
ChromoteSession$get_child_loop()
```

Method `send_command()`: Send command through Chrome DevTools Protocol.

For expert use only.

Usage:

```
ChromoteSession$send_command(
  msg,
  callback = NULL,
  error = NULL,
  timeout = NULL
)
```

Arguments:

`msg` A JSON-serializable list containing method, and params.

`callback` Method to run when the command finishes successfully.

`error` Method to run if an error occurs.

`timeout` Number of milliseconds for Chrome DevTools Protocol execute a method.

Method `get_auto_events()`: Resolved `auto_events` value.

For internal use only.

Usage:

```
ChromoteSession$get_auto_events()
```

Method `invoke_event_callbacks()`: Immediately call all event callback methods.

For internal use only.

Usage:

```
ChromoteSession$invoke_event_callbacks(event, params)
```

Arguments:

`event` A single event string

`params` A list of parameters to pass to the event callback methods.

Method `mark_closed()`: Disable callbacks for a given session.

For internal use only.

Usage:

```
ChromoteSession$mark_closed()
```

Method `is_active()`: Retrieve active status Once initialized, the value returned is TRUE. If `$close()` has been called, this value will be FALSE.

Usage:

```
ChromoteSession$is_active()
```

Method `init_promise()`: Initial promise

For internal use only.

Usage:

```
ChromoteSession$init_promise()
```


Examples

```

## -----
## Method `ChromoteSession$new`
## -----

## Not run: # Create a new `ChromoteSession` object.
b <- ChromoteSession$new()

# Create a ChromoteSession with a specific height,width
b <- ChromoteSession$new(height = 1080, width = 1920)

# Navigate to page
b$page$navigate("http://www.r-project.org/")

# View current chromote session
if (interactive()) b$view()
## End(Not run)

## -----
## Method `ChromoteSession$view`
## -----

## Not run: # Create a new `ChromoteSession` object.
b <- ChromoteSession$new()

# Navigate to page
b$page$navigate("http://www.r-project.org/")

# View current chromote session
if (interactive()) b$view()
## End(Not run)

## -----
## Method `ChromoteSession$close`
## -----

## Not run: # Create a new `ChromoteSession` object.
b <- ChromoteSession$new()

# Navigate to page
b$page$navigate("http://www.r-project.org/")

# Close current chromote session
b$close()
## End(Not run)

## -----
## Method `ChromoteSession$screenshot`
## -----

## Not run: # Create a new `ChromoteSession` object.

```

```

b <- ChromoteSession$new()

# Navigate to page
b$page$navigate("http://www.r-project.org/")

# Take screenshot
tmppngfile <- tempfile(fileext = ".png")
is_interactive <- interactive() # Display screenshot if interactive
b$screenshot(tmppngfile, show = is_interactive)

# Show screenshot file info
unlist(file.info(tmppngfile))

# Take screenshot using a selector
sidebar_file <- tempfile(fileext = ".png")
b$screenshot(sidebar_file, selector = ".sidebar", show = is_interactive)

# -----
# Take screenshots in parallel

urls <- c(
  "https://www.r-project.org/",
  "https://github.com/",
  "https://news.ycombinator.com/"
)
# Helper method that:
# 1. Navigates to the given URL
# 2. Waits for the page loaded event to fire
# 3. Takes a screenshot
# 4. Prints a message
# 5. Close the ChromoteSession
screenshot_p <- function(url, filename = NULL) {
  if (is.null(filename)) {
    filename <- gsub("^.*://", "", url)
    filename <- gsub("/", "_", filename)
    filename <- gsub("\\.", "_", filename)
    filename <- sub("_$", "", filename)
    filename <- paste0(filename, ".png")
  }

  b2 <- b$new_session()
  b2$page$navigate(url, wait_ = FALSE)
  b2$page$loadEventFired(wait_ = FALSE)$
    then(function(value) {
      b2$screenshot(filename, wait_ = FALSE)
    })$
    then(function(value) {
      message(filename)
    })$
    finally(function() {
      b2$close()
    })
}

```

```

}

# Take multiple screenshots simultaneously
ps <- lapply(urls, screenshot_p)
pa <- promises::promise_all(.list = ps)$then(function(value) {
  message("Done!")
})

# Block the console until the screenshots finish (optional)
b$wait_for(pa)
#> www_r-project_org.png
#> github_com.png
#> news_ycombinator_com.png
#> Done!
## End(Not run)

## -----
## Method `ChromoteSession$screenshot_pdf`
## -----

## Not run: # Create a new `ChromoteSession` object.
b <- ChromoteSession$new()

# Navigate to page
b$Page$navigate("http://www.r-project.org/")

# Take screenshot
tmppdffile <- tempfile(fileext = ".pdf")
b$screenshot_pdf(tmppdffile)

# Show PDF file info
unlist(file.info(tmppdffile))
## End(Not run)

## -----
## Method `ChromoteSession$new_session`
## -----

## Not run: b1 <- ChromoteSession$new()
b1$Page$navigate("http://www.google.com")
b2 <- b1$new_session()
b2$Page$navigate("http://www.r-project.org/")
b1$Runtime$evaluate("window.location", returnByValue = TRUE)$result$value$href
#> [1] "https://www.google.com/"
b2$Runtime$evaluate("window.location", returnByValue = TRUE)$result$value$href
#> [1] "https://www.r-project.org/"
## End(Not run)

## -----
## Method `ChromoteSession$get_session_id`
## -----

## Not run: b <- ChromoteSession$new()

```

```

b$get_session_id()
#> [1] "05764F1D439F4292497A21C6526575DA"
## End(Not run)

## -----
## Method `ChromoteSession$wait_for`
## -----

## Not run: b <- ChromoteSession$new()

# Async with promise
p <- b$Browser$getVersion(wait_ = FALSE)
p$then(str)

# Async with callback
b$Browser$getVersion(wait_ = FALSE, callback_ = str)
## End(Not run)

## -----
## Method `ChromoteSession$debug_log`
## -----

## Not run: b <- ChromoteSession$new()
b$parent$debug_messages(TRUE)
b$Page$navigate("https://www.r-project.org/")
#> SEND {"method": "Page.navigate", "params": {"url": "https://www.r-project.org/"} | __truncated__}
# Turn off debug messages
b$parent$debug_messages(FALSE)
## End(Not run)

```

default_chrome_args *Default Chrome arguments*

Description

A character vector of command-line arguments passed when initializing any new instance of [Chrome](#). Single on-off arguments are passed as single values (e.g. "`--disable-gpu`"), arguments with a value are given with a nested character vector (e.g. `c("--force-color-profile", "srgb")`). See [here](#) for a list of possible arguments.

Usage

```

default_chrome_args()

get_chrome_args()

set_chrome_args(args)

```

Arguments

`args` A character vector of command-line arguments (or NULL) to be used with every new [ChromoteSession](#).

Details

Default chromote arguments are composed of the following values (when appropriate):

- `--disable-gpu`
 - Disables GPU hardware acceleration. If software renderer is not in place, then the GPU process won't launch.
- `--no-sandbox`
 - Only added when CI system environment variable is set, when the user on a Linux system is not set, or when executing inside a Docker container.
 - Disables the sandbox for all process types that are normally sandboxed. Meant to be used as a browser-level switch for testing purposes only
- `--disable-dev-shm-usage`
 - Only added when CI system environment variable is set or when inside a docker instance.
 - The `/dev/shm` partition is too small in certain VM environments, causing Chrome to fail or crash
- `--force-color-profile=srgb`
 - This means that screenshots taken on a laptop plugged into an external monitor will often have subtly different colors than one taken when the laptop is using its built-in monitor. This problem will be even more likely across machines.
 - Force all monitors to be treated as though they have the specified color profile.
- `--disable-extensions`
 - Disable extensions.
- `--mute-audio`
 - Mutes audio sent to the audio device so it is not audible during automated testing

Value

A character vector of default command-line arguments to be used with every new [ChromoteSession](#)

Functions

- `default_chrome_args`: Returns a character vector of command-line arguments passed when initializing Chrome. See Details for more information.
- `get_chrome_args`: Retrieves the default command-line arguments passed to [Chrome](#) during initialization. Returns either NULL or a character vector.
- `set_chrome_args`: Sets the default command-line arguments passed when initializing. Returns the updated defaults.

Examples

```
old_chrome_args <- get_chrome_args()

# Only disable the gpu and using `/dev/shm`
set_chrome_args(c("--disable-gpu", "--disable-dev-shm-usage"))

#... Make new `Chrome` or `ChromoteSession` instance

# Restore old defaults
set_chrome_args(old_chrome_args)
```

default_chromote_object

Default Chromote object

Description

Returns the Chromote package's default [Chromote](#) object. If there is not currently a default Chromote object that is active, then one will be created and set as the default.

Usage

```
default_chromote_object()

has_default_chromote_object()

set_default_chromote_object(x)
```

Arguments

x A [Chromote](#) object.

Details

[ChromoteSession\\$new\(\)](#) calls this function by default, if the parent is not specified. That means that when [ChromoteSession\\$new\(\)](#) is called and there is not currently an active default Chromote object, then a new Chromote object will be created and set as the default.

find_chrome

Find path to Chrome or Chromium browser

Description

Find path to Chrome or Chromium browser

Usage

```
find_chrome()
```

Index

Browser, [2](#), [3](#), [6](#), [8](#)
browseURL(), [6](#), [10](#)

Chrome, [3](#), [6](#), [10](#), [20](#), [21](#)
ChromeRemote, [4](#)
Chromote, [5](#), [9](#), [10](#), [15](#), [22](#)
chromote::Browser, [3](#), [4](#)
ChromoteSession, [7](#), [9](#), [21](#), [22](#)

default_chrome_args, [20](#)
default_chromote_object, [22](#)
default_chromote_object(), [10](#)

find_chrome, [22](#)

get_chrome_args (default_chrome_args),
[20](#)
get_chrome_args(), [4](#)

has_default_chromote_object
(default_chromote_object), [22](#)

processx::process, [3](#)
promises::promise(), [7](#), [10–12](#), [14](#)

set_chrome_args (default_chrome_args),
[20](#)
set_default_chromote_object
(default_chromote_object), [22](#)