

# Package ‘chromote’

May 8, 2026

**Title** Headless Chrome Web Browser Interface

**Version** 0.5.1

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

**License** MIT + file LICENSE

**URL** <https://rstudio.github.io/chromote/>,  
<https://github.com/rstudio/chromote>

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

**Imports** cli, curl, fastmap, jsonlite, later (>= 1.1.0), magrittr, processx, promises (>= 1.1.1), R6, rlang (>= 1.1.0), utils, websocket (>= 1.2.0), withr, zip

**Suggests** knitr, rmarkdown, showimage, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/Needs/website** r-lib/pkgdown, rstudio/bslib

**Config/testthat/edition** 3

**Config/testthat/parallel** FALSE

**Config/testthat/start-first** chromote\_session

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.2

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

**NeedsCompilation** no

**Author** Garrick Aden-Buie [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-7111-0077>>),  
Winston Chang [aut],  
Barret Schloerke [aut] (ORCID: <<https://orcid.org/0000-0001-9986-114X>>),  
Posit Software, PBC [cph, fnd] (ROR: <<https://ror.org/03wc8by49>>)

**Maintainer** Garrick Aden-Buie <[garrick@posit.co](mailto:garrick@posit.co)>

**Repository** CRAN

**Date/Publication** 2025-04-24 03:40:02 UTC

## Contents

Browser . . . . .	2
Chrome . . . . .	3
ChromeRemote . . . . .	4
chrome_versions . . . . .	5
chrome_versions_list . . . . .	7
Chromote . . . . .	8
chromote-options . . . . .	13
ChromoteSession . . . . .	13
chromote_info . . . . .	25
default_chrome_args . . . . .	26
default_chromote_object . . . . .	27
find_chrome . . . . .	28
with_chrome_version . . . . .	29
<b>Index</b>	<b>32</b>

---

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

---

### Description

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.

### Details

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$is_alive()`
- `Browser$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** `is_alive()`: Is the process alive?

*Usage:*

```
Browser$is_alive()
```

**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(wait = FALSE)
```

*Arguments:*

`wait` If an integer, waits a number of seconds for the process to exit, killing the process if it takes longer than `wait` seconds to close. Use `wait = TRUE` to wait for 10 seconds.

**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

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

## Super class

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

## Methods

### Public methods:

- [ChromeRemote\\$new\(\)](#)
- [ChromeRemote\\$is\\_alive\(\)](#)
- [ChromeRemote\\$close\(\)](#)
- [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** `is_alive()`: Is the remote service alive?

*Usage:*

```
ChromeRemote$is_alive()
```

**Method** `close()`: chromote does not manage remote processes, so closing a remote Chrome browser does nothing. You can send a `Browser$close()` command if this is really something you want to do.

*Usage:*

```
ChromeRemote$close()
```

**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.

---

chrome\_versions

*Chrome versions cache helpers*

---

## Description

### [E]

These functions help interact with the cache used by **chromote**'s for storing versioned Chrome for Testing binaries:

- `chrome_versions_path()`: Returns a path or paths to specific Chrome binaries in the cache.
- `chrome_versions_add()`: Add a specific version to the Chrome versions cache.

- `chrome_versions_remove()`: Remove specific versions and binaries from the Chrome cache. The `version`, `binary` and `platform` arguments can each take "all" to remove all installed copies of that version, binary or platform.
- `chrome_versions_path_cache()`: Returns the path to the cache directory used for Chrome binaries.

Managed Chrome installations is an experimental feature introduced in chromote v0.5.0 and was inspired by similar features in [playwright](#).

### Usage

```
chrome_versions_path_cache(...)
```

```
chrome_versions_path(version = "latest", binary = "chrome", platform = NULL)
```

```
chrome_versions_add(version, binary, platform = NULL)
```

```
chrome_versions_remove(version, binary, platform = NULL, ask = TRUE)
```

### Arguments

...	Additional path parts.
version	A character string specifying the version to list, add or remove.
binary	A character string specifying which binary to list. Defaults to "all" to show all binaries, or can be one or more of "chrome", "chrome-headless-shell", or "chromedriver".
platform	A character string specifying the platform(s) to list. If NULL (default), the platform will be automatically detected, or if "all", then binaries for all platforms will be listed.
ask	Whether to ask before removing files.

### Value

A character vector of Chrome binary paths.

### See Also

[chrome\\_versions\\_list\(\)](#)

---

chrome\_versions\_list *List installed or available Chrome binary versions*

---

## Description

### [E]

By default lists the installed Chrome versions in the `chrome_versions_path_cache()`, or list all Chrome versions available via Google's [Chrome for Testing](#) service.

Managed Chrome installations is an experimental feature introduced in chromote v0.5.0 and was inspired by similar features in [playwright](#).

## Usage

```
chrome_versions_list(  
  which = c("installed", "all"),  
  binary = c("all", "chrome", "chrome-headless-shell", "chromedriver"),  
  platform = NULL  
)
```

## Arguments

which	Whether to list "installed" local binaries or to list "all" chrome versions available from online sources.
binary	A character string specifying which binary to list. Defaults to "all" to show all binaries, or can be one or more of "chrome", "chrome-headless-shell", or "chromedriver".
platform	A character string specifying the platform(s) to list. If NULL (default), the platform will be automatically detected, or if "all", then binaries for all platforms will be listed.

## Value

Returns a `data.frame()` of Chrome for Testing versions with columns: version, revision, binary, platform, url (where the binary can be downloaded), and—if which = "installed"—the local path to the binary in the `chrome_versions_path_cache()`.

## Examples

```
chrome_versions_list()
```

---

Chromote

*Chromote class*

---

## Description

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\\$connect\(\)](#)
- [Chromote\\$view\(\)](#)
- [Chromote\\$get\\_auto\\_events\(\)](#)
- [Chromote\\$auto\\_events\\_enable\\_args\(\)](#)
- [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$is_alive()`
- `Chromote$check_active()`
- `Chromote$get_browser()`
- `Chromote$close()`
- `Chromote$print()`

**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** `connect()`: Re-connect the websocket to the browser. The Chrome browser automatically closes websockets when your computer goes to sleep; you can use this to bring it back to life with a new connection.

*Usage:*

```
Chromote$connect(multi_session = TRUE, wait_ = TRUE)
```

*Arguments:*

`multi_session` Should multiple sessions be allowed?

`wait_` If FALSE, return a promise; if TRUE wait until connection is complete.

**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** `auto_events_enable_args()`: Set or retrieve the enable command arguments for a domain. These arguments are used for the enable command that is called for a domain, e.g. `Fetch$enable()`, when accessing an event method.

*Usage:*

```
Chromote$auto_events_enable_args(domain, ...)
```

*Arguments:*

`domain` A command domain, e.g. "Fetch".

... Arguments to use for auto-events for the domain. If not provided, returns the argument values currently in place for the domain. Use NULL to clear the enable arguments for a domain.

**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

*Examples:*

```
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)
```

*Usage:*

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

*Arguments:*

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

**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()`: Is there an active websocket connection to the browser process?

*Usage:*

```
Chromote$is_active()
```

**Method** `is_alive()`: Is the underlying browser process running?

*Usage:*

```
Chromote$is_alive()
```

**Method** `check_active()`: Check that a chromote instance is active and alive. Will automatically reconnect if browser process is alive, but there's no active web socket connection.

*Usage:*

```
Chromote$check_active()
```

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

*Usage:*

```
Chromote$get_browser()
```

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

*Usage:*

```
Chromote$close(wait = TRUE)
```

*Arguments:*

wait If an integer, waits a number of seconds for the process to exit, killing the process if it takes longer than wait seconds to close. Use `wait = TRUE` to wait for 10 seconds, or `wait = FALSE` to close the connection without waiting for the process to exit. Only applies when Chromote is connected to a local process.

**Method** `print()`: Summarise the current state of the object.

*Usage:*

```
Chromote$print(..., verbose = FALSE)
```

*Arguments:*

... Passed on to `format()` when `verbose = TRUE`

verbose The print method defaults to a brief summary of the most important debugging info; use `verbose = TRUE` to see the complex R6 object.

---

chromote-options      *chromote Options*

---

### Description

These options and environment variables that are used by chromote. Options are lowercase and can be set with `options()`. Environment variables are uppercase and can be set in an `.Renvi` file, with `Sys.setenv()`, or in the shell or process running R. If both an option or environment variable are supported, chromote will use the option first.

- `CHROMOTE_CHROME`  
Path to the Chrome executable. If not set, chromote will attempt to find and use the system installation of Chrome.
- `chromote.headless`, `CHROMOTE_HEADLESS`  
Headless mode for Chrome. Can be "old" or "new". See [Chrome Headless mode](#) for more details.
- `chromote.timeout`  
Timeout (in seconds) for Chrome to launch or connect. Default is 10.
- `chromote.launch.echo_cmd`  
Echo the command used to launch Chrome to the console for debugging. Default is FALSE.

---

ChromoteSession      *ChromoteSession class*

---

### Description

This represents one *session* in a Chromote object. Note that in the Chrome DevTools Protocol a session is a debugging session connected to a *target*, which is a browser window/tab or an iframe.

A single target can potentially have more than one session connected to it, but this is not currently supported by chromote.

### 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$close()`
- `ChromoteSession$get_viewport_size()`
- `ChromoteSession$set_viewport_size()`
- `ChromoteSession$go_to()`
- `ChromoteSession$screenshot()`
- `ChromoteSession$screenshot_pdf()`
- `ChromoteSession$new_session()`
- `ChromoteSession$get_session_id()`
- `ChromoteSession$respawn()`
- `ChromoteSession$get_target_id()`
- `ChromoteSession$wait_for()`
- `ChromoteSession$debug_log()`
- `ChromoteSession$get_child_loop()`
- `ChromoteSession$send_command()`
- `ChromoteSession$get_auto_events()`
- `ChromoteSession$auto_events_enable_args()`
- `ChromoteSession$invoke_event_callbacks()`
- `ChromoteSession$mark_closed()`
- `ChromoteSession$is_active()`
- `ChromoteSession$check_active()`
- `ChromoteSession$get_init_promise()`
- `ChromoteSession$print()`

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

*Examples:*

```
# 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$go_to("http://www.r-project.org/")

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

*Usage:*

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

*Arguments:*

parent [Chromote](#) object to use; defaults to `default_chromote_object()`

width, height Width and height of the new window in integer pixel values.

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.

mobile Whether to emulate mobile device. When TRUE, Chrome updates settings to emulate browsing on a mobile phone; this includes viewport meta tag, overlay scrollbars, text auto-sizing and more. The default is FALSE.

*Returns:* A new ChromoteSession object.

**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()`.

*Examples:*

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

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

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

*Usage:*

```
ChromoteSession$view()
```

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

*Examples:*

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

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

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

*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.

**Method** `get_viewport_size()`: Get the viewport size

*Usage:*

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

*Arguments:*

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

*Returns:* Returns a list with values `width`, `height`, `zoom` and `mobile`. See `$set_viewport_size()` for more details.

**Method** `set_viewport_size()`: Set the viewport size

Each `ChromoteSession` is associated with a page that may be one page open in a browser window among many. Each page can have its own viewport size, that can be thought of like the window size for that page.

This function uses the `Emulation.setDeviceMetricsOverride` command to set the viewport size. If you need more granular control or access to additional settings, use `$Emulation$setDeviceMetricsOverride()`.

*Usage:*

```
ChromoteSession$set_viewport_size(
  width,
  height,
  zoom = NULL,
  mobile = NULL,
  wait_ = TRUE
)
```

*Arguments:*

`width`, `height` Width and height of the new window in integer pixel values.

`zoom` The zoom level of displayed content on a device, where a value of 1 indicates normal size, greater than 1 indicates zoomed in, and less than 1 indicates zoomed out.

`mobile` Whether to emulate mobile device. When TRUE, Chrome updates settings to emulate browsing on a mobile phone; this includes viewport meta tag, overlay scrollbars, text auto-sizing and more. The default is FALSE.

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

*Returns:* Invisibly returns the previous viewport dimensions so that you can restore the viewport size, if desired.

**Method** `go_to()`: Navigate to a URL and wait for the page to load

This method navigates to a specified URL and waits for the page load event to complete. This is a more reliable alternative to directly calling `Page$navigate()`, which can return before the page is actually loaded. This method also allows for an optional delay after the load event has fired, in case the page needs to load additional assets after that event.

*Usage:*

```
ChromoteSession$go_to(
  url,
  ...,
  delay = 0,
  callback_ = NULL,
  error_ = NULL,
  timeout_ = self$default_timeout,
  wait_ = TRUE
)
```

*Arguments:*

`url` The URL to navigate to.

`...` Additional parameters passed to `Page$navigate()`.

`delay` Number of seconds to wait after the page load event fires.

`callback_` Function to call when the page load event fires.

`error_` Function to call if an error occurs during navigation.

`timeout_` Maximum time in seconds to wait for the page load event (defaults to session's `'default_timeout'`).

`wait_` If `FALSE`, returns a promise that resolves when navigation is complete. If `TRUE` (default), blocks until navigation is complete.

*Returns:* If `wait_` is `TRUE`, returns `invisible(NULL)`. If `wait_` is `FALSE`, returns a promise that resolves when navigation is complete. The promise resolves with the value from the navigate command.

*Examples:*

```
\dontrun{
# Basic navigation
b$go_to("https://www.r-project.org")

# Navigation with delay
b$go_to("https://www.r-project.org", delay = 2)

# Asynchronous navigation
p <- b$go_to("https://www.r-project.org", wait_ = FALSE)
p$then(function(value) print("Navigation complete!"))
}
```

**Method** `screenshot()`: Take a PNG screenshot

*Examples:*

```

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

# Navigate to page
b$go_to("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$new_session()
  b2$go_to(url, 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!

```

*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,
  options = list(),
  wait_ = TRUE
)

```

*Arguments:*

**filename** File path of where to save the screenshot. The format of the screenshot is inferred from the file extension; use `options = list(format = "jpeg")` to manually choose the format. See [Page.captureScreenshot](#) for supported formats; at the time of this release the format options were "png" (default), "jpeg", or "webp".

**selector** CSS selector to use for the screenshot.

**cliprect** An unnamed vector or list containing values for top, left, width, and height, in that order. 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.

`options` Additional options passed to `Page.captureScreenshot`.

`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.

**Method** `screenshot_pdf()`: Take a PDF screenshot

*Examples:*

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

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

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

```
# Show PDF file info
unlist(file.info(tmppdf file))
```

*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.

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

*Examples:*

```
b1 <- ChromoteSession$new()
b1$go_to("http://www.google.com")
b2 <- b1$new_session()
b2$go_to("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/"
```

*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.

**Method** `get_session_id()`: Retrieve the session id

*Usage:*

```
ChromoteSession$get_session_id()
```

**Method** `respawn()`: Create a new session that connects to the same target (i.e. page) as this session. This is useful if the session has been closed but the target still exists.

*Usage:*

```
ChromoteSession$respawn()
```

**Method** `get_target_id()`: Retrieve the target id

*Usage:*

```
ChromoteSession$get_target_id()
```

**Method** `wait_for()`: Wait for a `ChromoteSession` 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.

*Examples:*

```

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)

```

*Usage:*

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

*Arguments:*

p A promise to resolve.

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

*Examples:*

```

b <- ChromoteSession$new()
b$parent$debug_messages(TRUE)
b$go_to("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)

```

*Usage:*

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

*Arguments:*

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

**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** `auto_events_enable_args()`: Set or retrieve the enable command arguments for a domain. These arguments are used for the enable command that is called for a domain, e.g. `Fetch$enable()`, when accessing an event method.

*Usage:*

```
ChromoteSession$auto_events_enable_args(domain, ...)
```

*Arguments:*

domain A command domain, e.g. "Fetch".

... Arguments to use for auto-events for the domain. If not provided, returns the argument values currently in place for the domain. Use NULL to clear the enable arguments for a domain.

*Examples:*

```
if (interactive()) {
  b <- ChromoteSession$new(
    auto_events_enable_args = list(
      Fetch = list(handleAuthRequests = TRUE)
    )
  )

  # Get current `Fetch.enable` args
  b$auto_events_enable_args("Fetch")

  # Update the `Fetch.enable` args
  b$auto_events_enable_args("Fetch", handleAuthRequests = FALSE)

  # Reset `Fetch.enable` args
  b$auto_events_enable_args("Fetch", NULL)
}
```

**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()`: Mark a session, and optionally, the underlying target, as closed. For internal use only.

*Usage:*

```
ChromoteSession$mark_closed(target_closed)
```

*Arguments:*

`target_closed` Has the underlying target been closed as well as the active debugging session?

**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** `check_active()`: Check that a session is active, erroring if not.

*Usage:*

```
ChromoteSession$check_active()
```

**Method** `get_init_promise()`: Initial promise

For internal use only.

*Usage:*

```
ChromoteSession$get_init_promise()
```

**Method** `print()`: Summarise the current state of the object.

*Usage:*

```
ChromoteSession$print(..., verbose = FALSE)
```

*Arguments:*

`...` Passed on to `format()` when `verbose = TRUE`

`verbose` The print method defaults to a brief summary of the most important debugging info; use `verbose = TRUE` to see the complex R6 object.

**Examples**

```
## -----
## Method `ChromoteSession$go_to`
## -----

## Not run:
# Basic navigation
b$go_to("https://www.r-project.org")

# Navigation with delay
b$go_to("https://www.r-project.org", delay = 2)

# Asynchronous navigation
p <- b$go_to("https://www.r-project.org", wait_ = FALSE)
p$then(function(value) print("Navigation complete!"))

## End(Not run)

## -----
## Method `ChromoteSession$auto_events_enable_args`
```

```
## -----
if (interactive()) {
  b <- ChromoteSession$new(
    auto_events_enable_args = list(
      Fetch = list(handleAuthRequests = TRUE)
    )
  )

  # Get current `Fetch.enable` args
  b$auto_events_enable_args("Fetch")

  # Update the `Fetch.enable` args
  b$auto_events_enable_args("Fetch", handleAuthRequests = FALSE)

  # Reset `Fetch.enable` args
  b$auto_events_enable_args("Fetch", NULL)
}
```

---

chromote\_info

*Show information about the chromote package and Chrome browser*


---

### Description

This function gathers information about the operating system, R version, chromote package version, environment variables, Chrome path, and Chrome arguments. It also verifies the Chrome installation and retrieves its version.

### Usage

```
chromote_info()
```

### Value

A list containing the following elements:

- os** The operating system platform.
- version\_r** The version of R.
- version\_chromote** The version of the chromote package.
- envvar** The value of the CHROMOTE\_CHROME environment variable.
- path** The path to the Chrome browser.
- args** A vector of Chrome arguments.
- version** The version of Chrome (if verification is successful).
- error** The error message (if verification fails).
- .check** A list with the status and output of the Chrome verification.

## Examples

```
chromote_info()
```

---

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"`
  - Only added on Windows, as empirically it appears to be needed (if not, check runs on GHA never terminate).
  - 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()

# Disable the gpu and use of `/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**

**chromote** requires a Chrome- or Chromium-based browser with support for the Chrome DevTools Protocol. There are many such browser variants, including [Google Chrome](#), [Chromium](#), [Microsoft Edge](#) and others.

If you want **chromote** to use a specific browser, set the CHROMOTE\_CHROME environment variable to the full path to the browser's executable. Note that when CHROMOTE\_CHROME is set, **chromote** will use the value without any additional checks. On Mac, for example, one could use Microsoft Edge by setting CHROMOTE\_CHROME with the following:

```

Sys.setenv(
  CHROMOTE_CHROME = "/Applications/Microsoft Edge.app/Contents/MacOS/Microsoft Edge"
)

```

When CHROMOTE\_CHROME is not set, find\_chrome() will perform a limited search to find a reasonable executable. On Windows, find\_chrome() consults the registry to find chrome.exe. On Mac, it looks for Google Chrome in the /Applications folder (or tries the same checks as on Linux). On Linux, it searches for several common executable names.

**Usage**

```
find_chrome()
```

**Value**

A character vector with the value of CHROMOTE\_CHROME, or a path to the discovered Chrome executable. If no path to is found, find\_chrome() returns NULL.

**Examples**

```
find_chrome()
```

---

with\_chrome\_version    *Use a specific version of Chrome or related binaries*

---

**Description****[E]**

This function downloads and sets up a specific version of Chrome, using the [Google Chrome for Testing builds](#) for chrome, chrome-headless-shell or chromedriver for use with chromote.

Managed Chrome installations is an experimental feature introduced in chromote v0.5.0 and was inspired by similar features in [playwright](#).

**Usage**

```
with_chrome_version(
  version = "latest-stable",
  code,
  ...,
  binary = c("chrome", "chrome-headless-shell", "chromedriver"),
  platform = NULL,
  quiet = TRUE
)
```

```
local_chrome_version(
  version = "latest-stable",
  binary = c("chrome", "chrome-headless-shell", "chromedriver"),
  platform = NULL,
  ...,
  quiet = FALSE,
  .local_envir = parent.frame()
)
```

```
local_chromote_chrome(path, ..., .local_envir = parent.frame())
```

```
with_chromote_chrome(path, code, ...)
```

**Arguments**

**version**            A character string specifying the version to use. The default value is "latest-stable" to follow the latest stable release of Chrome. For robust results, and to avoid frequently downloading new versions of Chrome, use a fully qualified version number, e.g. "133.0.6943.141".

If you specify a partial version, e.g. "133", chromote will find the most recent release matching that version, preferring to use the latest *installed* release that matches the partially-specified version. chromote also supports a few special version names:

- "latest-installed": The latest version currently installed locally in chromote's cache. If you don't have any installed versions of the binary, chromote uses "latest".
- "latest": The most recent Chrome for Testing release, which may be a beta or canary release.
- "latest-stable", "latest-beta", "latest-extended", "latest-canary" or "latest-dev": Installs the latest release from one of Chrome's version channels, queried from the [VersionHistory API](#). "latest-stable" is the default value of `with_chrome_version()` and `local_chrome_version()`.
- "system": Use the system-wide installation of Chrome.

Chromote also supports

code	[any] Code to execute in the temporary environment
...	Ignored, used to require named arguments and for future feature expansion.
binary	A character string specifying which binary to use. Must be one of "chrome", "chrome-headless-shell", or "chromedriver". Default is "chrome".
platform	A character string specifying the platform. If NULL (default), the platform will be automatically detected.
quiet	Whether to print a message indicating which version and binary of Chrome is being used. By default, this message is suppressed for <code>with_chrome_version()</code> and enabled for <code>local_chrome_version()</code> .
.local_envir	[environment] The environment to use for scoping.
path	A direct path to the Chrome (or Chrome-based) binary. See <code>find_chrome()</code> for details or <code>chrome_versions_path()</code> for paths from the chromote-managed cache.

## Details

This function downloads the specified binary, if not already available and configures `find_chrome()` to use the specified binary while evaluating code or within the local scope. It uses the "known-good-versions" list from the Google Chrome for Testing versions at <https://googlechromelabs.github.io/chrome-for-testing/>.

## Value

Temporarily sets the `CHROMOTE_CHROME` environment variable and returns the result of the code argument.

**Functions**

- `with_chrome_version()`: Temporarily use a specific version of Chrome during the evaluation of code.
- `local_chrome_version()`: Use a specific version of Chrome within the current scope.
- `local_chromote_chrome()`: Use a specific Chrome, by path, within the current scope.
- `with_chromote_chrome()`: Temporarily use a specific Chrome version, by path, for the evaluation of code.

**Examples**

```
# Use the latest version of Chrome
local_chrome_version()
```

```
# Use a specific version of chrome-headless-shell
local_chrome_version("114.0.5735.90", binary = "chrome-headless-shell")
```

# Index

Browser, [2](#), [3](#), [9](#), [12](#)  
browseURL(), [9](#), [15](#)

Chrome, [3](#), [9](#), [15](#), [26](#), [27](#)  
chrome\_versions, [5](#)  
chrome\_versions\_add(chrome\_versions), [5](#)  
chrome\_versions\_list, [7](#)  
chrome\_versions\_list(), [6](#)  
chrome\_versions\_path(chrome\_versions),  
[5](#)  
chrome\_versions\_path(), [30](#)  
chrome\_versions\_path\_cache  
(chrome\_versions), [5](#)  
chrome\_versions\_path\_cache(), [7](#)  
chrome\_versions\_remove  
(chrome\_versions), [5](#)  
ChromeRemote, [4](#)  
Chromote, [8](#), [13](#), [15](#), [22](#), [27](#), [28](#)  
chromote-options, [13](#)  
chromote::Browser, [3](#), [4](#)  
chromote\_info, [25](#)  
ChromoteSession, [10](#), [11](#), [13](#), [26](#), [27](#)

data.frame(), [7](#)  
default\_chrome\_args, [26](#)  
default\_chromote\_object, [27](#)  
default\_chromote\_object(), [15](#)

find\_chrome, [28](#)  
find\_chrome(), [30](#)

get\_chrome\_args(default\_chrome\_args),  
[26](#)  
get\_chrome\_args(), [4](#)

has\_default\_chromote\_object  
(default\_chromote\_object), [27](#)

local\_chrome\_version  
(with\_chrome\_version), [29](#)  
local\_chrome\_version(), [30](#)

local\_chromote\_chrome  
(with\_chrome\_version), [29](#)

processx::process, [3](#)  
promises::promise(), [10](#), [15](#), [16](#), [20](#), [21](#)

set\_chrome\_args(default\_chrome\_args),  
[26](#)  
set\_default\_chromote\_object  
(default\_chromote\_object), [27](#)

with\_chrome\_version, [29](#)  
with\_chrome\_version(), [30](#)  
with\_chromote\_chrome  
(with\_chrome\_version), [29](#)