Package 'splashr'

January 16, 2018

Type Package

Title Tools to Work with the 'Splash' 'JavaScript' Rendering and Scraping Service

Version 0.4.1

Date 2018-01-16

Encoding UTF-8

Maintainer Bob Rudis <bob@rud.is>

Description 'Splash' <https://github.com/scrapinghub/splash> is a 'JavaScript' rendering service. It is a lightweight web browser with an 'HTTP' API, implemented in 'Python' using 'Twisted' and 'QT' and provides some of the core functionality of the 'RSelenium' or 'seleniumPipes' R packages in a lightweight footprint. Some of 'Splash' features include the ability to process multiple web pages in parallel; retrieving 'HTML' results and/or take screen shots; disabling images or use 'Adblock Plus' rules to make rendering faster; executing custom 'JavaScript' in page context; getting detailed rendering info in 'HAR' format.

URL http://github.com/hrbrmstr/splashr

BugReports https://github.com/hrbrmstr/splashr/issues

License AGPL

Suggests testthat, tibble, jpeg, png, covr, knitr, rmarkdown

Depends R (>= 3.2.0)

Imports xml2, curl, httr, purrr, stats, utils, docker, magick, scales, formatR, openssl, stringi, jsonlite, HARtools, lubridate

RoxygenNote 6.0.1.9000

VignetteBuilder knitr

NeedsCompilation no

Author Bob Rudis [aut, cre] (0000-0001-5670-2640)

Repository CRAN

Date/Publication 2018-01-16 04:38:22 UTC

R topics documented:

`as_har` *Turn a generic Splash HAR response into a HAR object*

Description

Turn a generic Splash HAR response into a HAR object

Usage

```
as_har(splash_resp)
```

Arguments

`splash_resp`	splash response object

`as_httr_req` *Create an httr verb request function from an HAR request*

Description

This function is very useful if you used `splashr` to find XHR requests in a dynamic page and want to be able to make a call directly to that XHR resource. Once you identify the proper HAR entry, pass it to this function and fully working function that makes an `httr::VERB()` request will be created and returned.

Usage

```
as_httr_req(entry, quiet = TRUE)
```

Arguments

`entry`	HAR entry
`quiet`	quiet (no messages)

as_response *Return a HAR entry response as an httr::response object*

Description

Return a HAR entry response as an httr::response object

Usage

```
as_response(har_entry)
```

Arguments

har_entry	a HAR object (should contain a response body to be most useful)

Examples

```
## Not run:
library(purrr)

URL <- "http://www.svs.cl/portal/principal/605/w3-propertyvalue-18554.html"

splash_local %>%
  splash_response_body(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go(URL) %>%
  splash_wait(2) %>%
  splash_har() -> har

keep(har$log$entries, is_xhr) %>%
  map(as_request) %>%
  map(httr::content, as="parsed")

## End(Not run)
```

execute_lua *Execute a custom rendering script and return a result.*

Description

Execute a custom rendering script and return a result.

Usage

```
execute_lua(splash_obj, lua_source, timeout = 30, allowed_domains, proxy,
  filters, save_args, load_args)
```

Arguments

splash_obj	Object created by a call to splash()
lua_source	Browser automation script. See Splash Script Tutorial for more info.
timeout	A timeout (in seconds) for the render (defaults to 30). Without reconfiguring the startup parameters of the Splash server (not this package) the maximum allowed value for the timeout is 60 seconds.
allowed_domains	Comma-separated list of allowed domain names. If present, Splash won't load anything neither from domains not in this list nor from subdomains of domains not in this list.
proxy	Proxy profile name or proxy URL.
filters	Comma-separated list of request filter names.
save_args	A list of argument names to put in cache.
load_args	Parameter values to load from cache

Value

raw content from the httr call. Given the vast diversity of possible return values, it's up to the caller to handle the return value.

See Also

Other splash_renderers: render_har, render_html, render_jpeg, render_json, render_png

Examples

```
## Not run:
splash_local %>%
  execute_lua('
function main(splash)
  splash:go("https://projects.fivethirtyeight.com/congress-trump-score/")
  splash:wait(0.5)
  return splash:evaljs("memberScores")
end
') -> res

rawToChar(res) %>%
  jsonlite::fromJSON(flatten=TRUE) %>%
  purrr::map(tibble::as_tibble) -> member_scores

member_scores

## End(Not run)
```

get_content_size *Retrieve size of content | body | headers*

Description

Retrieve size of content | body | headers

Usage

```
get_content_size(har_resp_obj)

get_body_size(har_resp_obj)

get_headers_size(har_resp_obj)
```

Arguments

har_resp_obj	HAR response object

See Also

Other splash_har_helpers: get_content_type, get_har_entry, get_request_type, get_request_url, get_response_body, har_entry_count

get_content_type *Retrieve or test content type of a HAR request object*

Description

Retrieve or test content type of a HAR request object

Usage

```
get_content_type(har_resp_obj)

is_content_type(har_resp_obj, type = "application/json")

is_json(har_resp_obj)

is_xml(har_resp_obj)

is_css(har_resp_obj)

is_plain(har_resp_obj)

is_binary(har_resp_obj)
```

```
is_javascript(har_resp_obj)

is_html(har_resp_obj)

is_jpeg(har_resp_obj)

is_png(har_resp_obj)

is_svg(har_resp_obj)

is_gif(har_resp_obj)

is_xhr(har_resp_obj)
```

Arguments

har_resp_obj	a reponse object from [render_har()] or [execute_lua()]
type	content type to compare to (default: "application/json")

See Also

Other splash_har_helpers: get_content_size, get_har_entry, get_request_type, get_request_url, get_response_body, har_entry_count

---

get_har_entry *Retrieve an entry by index from a HAR object*

Description

Retrieve an entry by index from a HAR object

Usage

```
get_har_entry(x, i = 1)
```

Arguments

x	can be a ‘har‘ object, ‘harlog‘ object or ‘harentries‘ object
i	index of the HAR entry to retrieve

See Also

Other splash_har_helpers: get_content_size, get_content_type, get_request_type, get_request_url, get_response_body, har_entry_count

get_request_type *Retrieve or test request type*

Description

Retrieve or test request type

Usage

```
get_request_type(har_resp_obj)

is_get(har_resp_obj)

is_post(har_resp_obj)
```

Arguments

har_resp_obj	HAR response object

See Also

Other splash_har_helpers: get_content_size, get_content_type, get_har_entry, get_request_url, get_response_body, har_entry_count

get_request_url *Retrieve request URL*

Description

Retrieve request URL

Usage

```
get_request_url(har_resp_obj)
```

Arguments

har_resp_obj	HAR response object

See Also

Other splash_har_helpers: get_content_size, get_content_type, get_har_entry, get_request_type, get_response_body, har_entry_count

get_response_body *Retrieve the body content of a HAR entry*

Description

Retrieve the body content of a HAR entry

Usage

```
get_response_body(har_resp_obj, type = c("raw", "text"))
```

Arguments

har_resp_obj	HAR response object
type	return type. If raw (default) then a raw vector of the content is returned. If text then a character vector.

Value

A raw vector of the content or NULL or a character if type == text

See Also

Other splash_har_helpers: get_content_size, get_content_type, get_har_entry, get_request_type, get_request_url, har_entry_count

har_entries *Retrieve just the HAR entries from a splashr request*

Description

Retrieve just the HAR entries from a splashr request

Usage

```
har_entries(x)
```

Arguments

x	can be a ‘har‘ object, ‘harlog‘ object or ‘harentries‘ object

har_entry_count *Retrieves number of HAR entries in a response*

Description

Retrieves number of HAR entries in a response

Usage

```
har_entry_count(x)
```

Arguments

x	can be a ‘har‘ object, ‘harlog‘ object or ‘harentries‘ object

See Also

Other splash_har_helpers: get_content_size, get_content_type, get_har_entry, get_request_type, get_request_url, get_response_body

install_splash *Retrieve the Docker image for Splash*

Description

Retrieve the Docker image for Splash

Usage

```
install_splash(tag = "3.0")
```

Arguments

tag	Splash Docker image tag to install

See Also

Other splash_docker_helpers: start_splash, stop_splash

Examples

```
## Not run:
install_splash()
splash_container <- start_splash()
stop_splash(splash_container)

## End(Not run)
```

json_fromb64 *Convert a Base64 encoded string into an R object*

Description

A simple wrapper around calls to `openssl::base64_decode()` and `jsonlite::fromJSON()`.

Usage

```
json_fromb64(x, flatten = TRUE, ...)
```

Arguments

`x`	a string
`flatten`	flatten JSON structures upon conversion?
`...`	passed on to `jsonlite::fromJSON()`

killall_splash *Prune all dead and running Splash Docker containers*

Description

This is a destructive function. It will stop **any** Docker container that is based on an image matching "‘scrapinghub/splashr‘". It's best used when you had a session forcefully interuppted and had been using the R helper functions to start/stop the Splash Docker container. You may want to consider using the Docker command-line interface to perform this work manually.

Usage

```
killall_splash()
```

render_har *Return information about Splash interaction with a website in HAR format.*

Description

It includes information about requests made, responses received, timings, headers, etc and is incredibly detailed, full of information on every component loaded.

Usage

```
render_har(splash_obj = splash_local, url, base_url, response_body = FALSE,
  timeout = 30, resource_timeout, wait = 0, proxy, js, js_src, filters,
  allowed_domains, allowed_content_types, forbidden_content_types,
  viewport = "1024x768", images, headers, body, http_method, save_args,
  load_args)
```

Arguments

splash_obj	Object created by a call to splash()
url	The URL to render (required)
base_url	The base url to render the page with.
response_body	When TRUE, response content is included in the HAR records
timeout	A timeout (in seconds) for the render (defaults to 30). Without reconfiguring the startup parameters of the Splash server (not this package) the maximum allowed value for the timeout is 60 seconds.
resource_timeout	A timeout (in seconds) for individual network requests.
wait	Time (in seconds) to wait for updates after page is loaded (defaults to 0).
proxy	Proxy profile name or proxy URL.
js	Javascript profile name.
js_src	JavaScript code to be executed in page context.
filters	Comma-separated list of request filter names.
allowed_domains	Comma-separated list of allowed domain names. If present, Splash won't load anything neither from domains not in this list nor from subdomains of domains not in this list.
allowed_content_types	Comma-separated list of allowed content types. If present, Splash will abort any request if the response's content type doesn't match any of the content types in this list. Wildcards are supported.
forbidden_content_types	Comma-separated list of forbidden content types. If present, Splash will abort any request if the response's content type matches any of the content types in this list. Wildcards are supported.
viewport	View width and height (in pixels) of the browser viewport to render the web page. Format is "<width>x<height>", e.g. 800x600. Default value is "full".
images	Whether to download images.
headers	HTTP headers to set for the first outgoing request.
body	Body of HTTP POST request to be sent if method is POST.
http_method	HTTP method of outgoing Splash request.
save_args	A list of argument names to put in cache.
load_args	Parameter values to load from cache

Value

a HARtools `har` object

References

Splash docs

See Also

Other splash_renderers: `execute_lua`, `render_html`, `render_jpeg`, `render_json`, `render_png`

`render_html` *Return the HTML of the javascript-rendered page.*

Description

Similar (i.e. a dynamic equivalent) to `rvest::read_html`.

Usage

```
render_html(splash_obj = splash_local, url, base_url, timeout = 30,
  resource_timeout, wait = 0, proxy, js, js_src, filters, allowed_domains,
  allowed_content_types, forbidden_content_types, viewport = "1024x768",
  images, headers, body, http_method, save_args, load_args, raw_html = FALSE)
```

Arguments

`splash_obj`	Object created by a call to `splash()`
`url`	The URL to render (required)
`base_url`	The base url to render the page with.
`timeout`	A timeout (in seconds) for the render (defaults to 30). Without reconfiguring the startup parameters of the Splash server (not this package) the maximum allowed value for the timeout is 60 seconds.
`resource_timeout`	A timeout (in seconds) for individual network requests.
`wait`	Time (in seconds) to wait for updates after page is loaded (defaults to 0).
`proxy`	Proxy profile name or proxy URL.
`js`	Javascript profile name.
`js_src`	JavaScript code to be executed in page context.
`filters`	Comma-separated list of request filter names.
`allowed_domains`	Comma-separated list of allowed domain names. If present, Splash won't load anything neither from domains not in this list nor from subdomains of domains not in this list.

allowed_content_types
: Comma-separated list of allowed content types. If present, Splash will abort any request if the response's content type doesn't match any of the content types in this list. Wildcards are supported.

forbidden_content_types
: Comma-separated list of forbidden content types. If present, Splash will abort any request if the response's content type matches any of the content types in this list. Wildcards are supported.

viewport
: View width and height (in pixels) of the browser viewport to render the web page. Format is "<width>x<height>", e.g. 800x600. Default value is "full".

images
: Whether to download images.

headers
: HTTP headers to set for the first outgoing request.

body
: Body of HTTP POST request to be sent if method is POST.

http_method
: HTTP method of outgoing Splash request.

save_args
: A list of argument names to put in cache.

load_args
: Parameter values to load from cache

raw_html
: if TRUE then return a character vector vs an XML document. Only valid for render_html

Value

An XML document. Note that this is processed by xml2::read_html() so it will not be the pristine, raw, rendered HTML from the site. Use raw_html=TRUE if you do not want it to be processed first by xml2. If you choose raw_html=TRUE you'll get back a character vector.

References

Splash docs

See Also

Other splash_renderers: execute_lua, render_har, render_jpeg, render_json, render_png

---

render_jpeg *Return a image (in JPEG format) of the javascript-rendered page.*

Description

Return a image (in JPEG format) of the javascript-rendered page.

Usage

```
render_jpeg(splash_obj = splash_local, url, base_url = NULL, quality = 75,
  width, height, timeout = 30, resource_timeout, wait = 0,
  render_all = TRUE, proxy, js, js_src, filters, allowed_domains,
  allowed_content_types, forbidden_content_types, viewport = "full", images,
  headers, body, http_method, save_args, load_args)
```

Arguments

splash_obj	Object created by a call to splash()
url	The URL to render (required)
base_url	The base url to render the page with.
quality	JPEG quality parameter in range from 0 to 100. Default is quality=75.
width	Resize the rendered image to the given width/height (in pixels) keeping the aspect ratio. These are optional
height	Resize the rendered image to the given width/height (in pixels) keeping the aspect ratio. These are optional
timeout	A timeout (in seconds) for the render (defaults to 30). Without reconfiguring the startup parameters of the Splash server (not this package) the maximum allowed value for the timeout is 60 seconds.
resource_timeout	A timeout (in seconds) for individual network requests.
wait	Time (in seconds) to wait for updates after page is loaded (defaults to 0).
render_all	If TRUE extend the viewport to include the whole webpage (possibly very tall) before rendering.
proxy	Proxy profile name or proxy URL.
js	Javascript profile name.
js_src	JavaScript code to be executed in page context.
filters	Comma-separated list of request filter names.
allowed_domains	Comma-separated list of allowed domain names. If present, Splash won't load anything neither from domains not in this list nor from subdomains of domains not in this list.
allowed_content_types	Comma-separated list of allowed content types. If present, Splash will abort any request if the response's content type doesn't match any of the content types in this list. Wildcards are supported.
forbidden_content_types	Comma-separated list of forbidden content types. If present, Splash will abort any request if the response's content type matches any of the content types in this list. Wildcards are supported.
viewport	View width and height (in pixels) of the browser viewport to render the web page. Format is "<width>x<height>", e.g. 800x600. Default value is "full".
images	Whether to download images.
headers	HTTP headers to set for the first outgoing request.
body	Body of HTTP POST request to be sent if method is POST.
http_method	HTTP method of outgoing Splash request.
save_args	A list of argument names to put in cache.
load_args	Parameter values to load from cache

Value

a magick image object

References

Splash docs

See Also

Other splash_renderers: execute_lua, render_har, render_html, render_json, render_png

---

render_json *Return a json-encoded dictionary with information about javascript-rendered webpage.*

Description

It can include HTML, PNG and other information, based on arguments passed.

Usage

```
render_json(splash_obj = splash_local, url, base_url = NULL, quality = 75,
  width, height, timeout = 30, resource_timeout, wait = 0,
  render_all = FALSE, proxy, js, js_src, filters, allowed_domains,
  allowed_content_types, forbidden_content_types, viewport = "1024x768",
  images, headers, body, http_method, save_args, load_args, html = TRUE,
  png = FALSE, jpeg = FALSE, iframes = TRUE, script = TRUE,
  console = TRUE, history = TRUE, har = TRUE, response_body = FALSE)
```

Arguments

splash_obj	Object created by a call to splash()
url	The URL to render (required)
base_url	The base url to render the page with.
quality	JPEG quality parameter in range from 0 to 100. Default is quality=75.
width	Resize the rendered image to the given width/height (in pixels) keeping the aspect ratio. These are optional
height	Resize the rendered image to the given width/height (in pixels) keeping the aspect ratio. These are optional
timeout	A timeout (in seconds) for the render (defaults to 30). Without reconfiguring the startup parameters of the Splash server (not this package) the maximum allowed value for the timeout is 60 seconds.
resource_timeout	A timeout (in seconds) for individual network requests.
wait	Time (in seconds) to wait for updates after page is loaded (defaults to 0).

render_all — If TRUE extend the viewport to include the whole webpage (possibly very tall) before rendering.

proxy — Proxy profile name or proxy URL.

js — Javascript profile name.

js_src — JavaScript code to be executed in page context.

filters — Comma-separated list of request filter names.

allowed_domains — Comma-separated list of allowed domain names. If present, Splash won't load anything neither from domains not in this list nor from subdomains of domains not in this list.

allowed_content_types — Comma-separated list of allowed content types. If present, Splash will abort any request if the response's content type doesn't match any of the content types in this list. Wildcards are supported.

forbidden_content_types — Comma-separated list of forbidden content types. If present, Splash will abort any request if the response's content type matches any of the content types in this list. Wildcards are supported.

viewport — View width and height (in pixels) of the browser viewport to render the web page. Format is "<width>x<height>", e.g. 800x600. Default value is "full".

images — Whether to download images.

headers — HTTP headers to set for the first outgoing request.

body — Body of HTTP POST request to be sent if method is POST.

http_method — HTTP method of outgoing Splash request.

save_args — A list of argument names to put in cache.

load_args — Parameter values to load from cache

html — Whether to include HTML in output.

png — Whether to include PNG in output.

jpeg — Whether to include JPEG in output.

iframes — Whether to include information about child frames in output.

script — Whether to include the result of the custom executed javascript final statement in output

console — Whether to include the executed javascript console messages in output.

history — Whether to include the history of requests/responses for webpage main frame. Use it to get HTTP status codes and headers. Only information about "main" requests/responses is returned (i.e. information about related resources like images and AJAX queries is not returned). To get information about all requests and responses use har parameter.

har — Whether to include HAR in output. If TRUE the result will contain the same data as render_har() provides under har list entry. By default, response content is not included. To enable it use response_body parameter.

response_body — Used with har parameter.

Value

a huge `list`

Note

All "whether to include..." parameters are default `TRUE` except for `png` and `jpeg` and a custom `print` method is defined to stop your console from being overwhelmed with data. Use `str()` to inspect various portions of the result.

References

Splash docs

See Also

Other splash_renderers: `execute_lua`, `render_har`, `render_html`, `render_jpeg`, `render_png`

`render_png` *Return a image (in PNG format) of the javascript-rendered page.*

Description

Return a image (in PNG format) of the javascript-rendered page.

Usage

```
render_png(splash_obj = splash_local, url, base_url = NULL, width, height,
  timeout = 30, resource_timeout, wait = 0, render_all = TRUE, proxy, js,
  js_src, filters, allowed_domains, allowed_content_types,
  forbidden_content_types, viewport = "full", images, headers, body,
  http_method, save_args, load_args)
```

Arguments

`splash_obj`	Object created by a call to `splash()`
`url`	The URL to render (required)
`base_url`	The base url to render the page with.
`width, height`	Resize the rendered image to the given width/height (in pixels) keeping the aspect ratio. These are optional
`timeout`	A timeout (in seconds) for the render (defaults to 30). Without reconfiguring the startup parameters of the Splash server (not this package) the maximum allowed value for the timeout is 60 seconds.
`resource_timeout`	A timeout (in seconds) for individual network requests.
`wait`	Time (in seconds) to wait for updates after page is loaded (defaults to 0).

render_all	If TRUE extend the viewport to include the whole webpage (possibly very tall) before rendering.
proxy	Proxy profile name or proxy URL.
js	Javascript profile name.
js_src	JavaScript code to be executed in page context.
filters	Comma-separated list of request filter names.
allowed_domains	Comma-separated list of allowed domain names. If present, Splash won't load anything neither from domains not in this list nor from subdomains of domains not in this list.
allowed_content_types	Comma-separated list of allowed content types. If present, Splash will abort any request if the response's content type doesn't match any of the content types in this list. Wildcards are supported.
forbidden_content_types	Comma-separated list of forbidden content types. If present, Splash will abort any request if the response's content type matches any of the content types in this list. Wildcards are supported.
viewport	View width and height (in pixels) of the browser viewport to render the web page. Format is "<width>x<height>", e.g. 800x600. Default value is "full".
images	Whether to download images.
headers	HTTP headers to set for the first outgoing request.
body	Body of HTTP POST request to be sent if method is POST.
http_method	HTTP method of outgoing Splash request.
save_args	A list of argument names to put in cache.
load_args	Parameter values to load from cache

Value

a magick image object

References

Splash docs

See Also

Other splash_renderers: execute_lua, render_har, render_html, render_jpeg, render_json

Examples

```
## Not run:
render_png(url = "https://httpbin.org/")

## End(Not run)
```

splash *Configure parameters for connecting to a Splash server*

Description

Configure parameters for connecting to a Splash server

Usage

```
splash(host, port = 8050L)

splash_local
```

Arguments

host	host or IP address
port	port the server is running on (default is 8050)

Format

An object of class `list` of length 2.

Examples

```
## Not run:
sp <- splash()

## End(Not run)
```

splashr *Tools to Work with the 'Splash' JavaScript Rendering Service*

Description

'Splash' https://github.com/scrapinghub/splash is a 'JavaScript' rendering service. It's a lightweight web browser with an 'HTTP' API, implemented in 'Python' using 'Twisted' and 'QT' and provides some of the core functionality of the 'RSelenium' or 'seleniumPipes' R pacakges in a lightweight footprint.

Details

Some of 'Splash' features include the ability to process multiple webpages in parallel; retrieving 'HTML' results and/or take screenshots; disabling images or use 'Adblock Plus' rules to make rendering faster; executing custom 'JavaScript' in page context; getting detailed rendering info in 'HAR' format.

Author(s)

Bob Rudis (bob@rud.is)

`splashr-exports` *splashr exported operators*

Description

The following functions are imported and then re-exported from the splashr package to enable use of the magrittr pipe operator with no additional library calls

---

`splash_active` *Test if a Splash server is up*

Description

Test if a Splash server is up

Usage

```
splash_active(splash_obj = splash_local)
```

Arguments

`splash_obj`	A splash connection object

Value

`TRUE` if Slash server is running, otherwise `FALSE`

See Also

Other splash_info_functions: `splash_debug`, `splash_history`, `splash_perf_stats`, `splash_version`

Examples

```
## Not run:
sp <- splash()
splash_active(sp)

## End(Not run)
```

splash_add_lua *Add raw lua code into DSL call chain*

Description

The splashr lua DSL (domain specific language) wrapper wraps what the package author believes to be the most common/useful lua functions. Users of the package may have need to insert some custom lua code within a DSL call chain they are building. You can insert any Splash lua code you like with this function call.

Usage

```
splash_add_lua(splash_obj, lua_code)
```

Arguments

splash_obj	splashr object
lua_code	length 1 character vector of raw lua code

Details

The code is inserted at the position the splash_add_lua() is called in the chain which will be within the main "splash' function which is defined as:

```
function main(splash)
  ...
end
```

If you need more flexibility, use the execute_lua() function.

See Also

Other splash_dsl_functions: splash_click, splash_focus, splash_go, splash_har_reset, splash_har, splash_html, splash_png, splash_press, splash_release, splash_send_keys, splash_send_text, splash_wait

splash_click *Trigger mouse click event in web page.*

Description

Trigger mouse click event in web page.

Usage

```
splash_click(splash_obj, x, y)
```

Arguments

splash_obj	splashr object
x, y	coordinates (distances from the left or top, relative to the current viewport)

See Also

Other splash_dsl_functions: splash_add_lua, splash_focus, splash_go, splash_har_reset, splash_har, splash_html, splash_png, splash_press, splash_release, splash_send_keys, splash_send_text, splash_wait

---

splash_enable_javascript

Enable or disable execution of JavaSript code embedded in the page.

Description

JavaScript execution is enabled by default.

Usage

```
splash_enable_javascript(splash_obj, enable = TRUE)
```

Arguments

splash_obj	splashr object
enable	logical

See Also

Other splash_dsl_attributes: splash_images, splash_plugins, splash_private_mode, splash_response_body

Examples

```
## Not run:
splash_local %>%
  splash_response_body(TRUE) %>%
  splash_private_mode(TRUE) %>%
  splash_enable_javascript(FALSE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_har() -> rud_har

## End(Not run)
```

splash_focus *Focus on a document element provided by a CSS selector*

Description

Focus on a document element provided by a CSS selector

Usage

```
splash_focus(splash_obj, selector)
```

Arguments

splash_obj	splashr object
selector	valid CSS selector

References

See the docs for more info

See Also

Other splash_dsl_functions: splash_add_lua, splash_click, splash_go, splash_har_reset, splash_har, splash_html, splash_png, splash_press, splash_release, splash_send_keys, splash_send_text, splash_wait

splash_go *Go to an URL.*

Description

This is similar to entering an URL in a browser address bar, pressing Enter and waiting until page loads.

Usage

```
splash_go(splash_obj, url)
```

Arguments

splash_obj	splashr object
url	- URL to load;

See Also

Other splash_dsl_functions: splash_add_lua, splash_click, splash_focus, splash_har_reset, splash_har, splash_html, splash_png, splash_press, splash_release, splash_send_keys, splash_send_text, splash_wait

Examples

```
## Not run:
splash_local %>%
  splash_response_body(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_har() -> rud_har

## End(Not run)
```

---

splash_har *Return information about Splash interaction with a website in HAR format.*

Description

Similar to render_har() but used in a script context. Should be the LAST element in a DSL script chain as this will execute the script and return the HAR content

Usage

```
splash_har(splash_obj)
```

Arguments

splash_obj	splashr object

See Also

Other splash_dsl_functions: splash_add_lua, splash_click, splash_focus, splash_go, splash_har_reset, splash_html, splash_png, splash_press, splash_release, splash_send_keys, splash_send_text, splash_wait

Examples

```
## Not run:
splash_local %>%
  splash_response_body(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_har() -> rud_har
```

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

`splash_har_reset` *Drops all internally stored HAR records.*

Description

Drops all internally stored HAR records.

Usage

```
splash_har_reset(splash_obj)
```

Arguments

`splash_obj`	splashr object

See Also

Other splash_dsl_functions: `splash_add_lua`, `splash_click`, `splash_focus`, `splash_go`, `splash_har`, `splash_html`, `splash_png`, `splash_press`, `splash_release`, `splash_send_keys`, `splash_send_text`, `splash_wait`

`splash_history` *Get information about requests/responses for the pages loaded*

Description

Get information about requests/responses for the pages loaded

Usage

```
splash_history(splash_obj = splash_local)
```

Arguments

`splash_obj`	A splash connection object

See Also

Other splash_info_functions: `splash_active`, `splash_debug`, `splash_perf_stats`, `splash_version`

Examples

```
## Not run:
sp <- splash()
splash_history(sp)

## End(Not run)
```

`splash_html` *Return a HTML snapshot of a current page.*

Description

Similar to `render_html()` but used in a script context. Should be the LAST element in a DSL script chain as this will execute the script and return the HTML content

Usage

```
splash_html(splash_obj, raw_html = FALSE)
```

Arguments

`splash_obj`	splashr object
`raw_html`	if `TRUE` then return a character vector vs an XML document.

See Also

Other splash_dsl_functions: `splash_add_lua`, `splash_click`, `splash_focus`, `splash_go`, `splash_har_reset`, `splash_har`, `splash_png`, `splash_press`, `splash_release`, `splash_send_keys`, `splash_send_text`, `splash_wait`

Examples

```
## Not run:
splash_local %>%
  splash_response_body(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_html() -> rud_pg

## End(Not run)
```

splash_images *Enable/disable images*

Description

By default, images are enabled. Disabling of the images can save a lot of network traffic (usually around ~50 affect the JavaScript code inside page: disabling of the images may change sizes and positions of DOM elements, and scripts may read and use them.

Usage

```
splash_images(splash_obj, enable = TRUE)
```

Arguments

splash_obj	splashr object
enable	logical

See Also

Other splash_dsl_attributes: splash_enable_javascript, splash_plugins, splash_private_mode, splash_response_body

Examples

```
## Not run:
splash_local %>%
  splash_images(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_har() -> rud_har

## End(Not run)
```

splash_perf_stats *Get Splash performance-related statistics*

Description

Get Splash performance-related statistics

Usage

```
splash_perf_stats(splash_obj = splash_local)
```

Arguments

splash_obj	A splash connection object

See Also

Other splash_info_functions: splash_active, splash_debug, splash_history, splash_version

Examples

```
## Not run:
sp <- splash()
splash_perf_stats(sp)

## End(Not run)
```

---

splash_plugins *Enable or disable browser plugins (e.g. Flash).*

Description

Plugins are disabled by default.

Usage

```
splash_plugins(splash_obj, enable = FALSE)
```

Arguments

splash_obj	splashr object
enable	logical

See Also

Other splash_dsl_attributes: splash_enable_javascript, splash_images, splash_private_mode, splash_response_body

Examples

```
## Not run:
splash_local %>%
  splash_plugins(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_har() -> rud_har

## End(Not run)
```

splash_png *Return a screenshot of a current page in PNG format.*

Description

Similar to `render_png()` but used in a script context. Should be the LAST element in a DSL script chain as this will execute the script and return the PNG content

Usage

```
splash_png(splash_obj)
```

Arguments

splash_obj	splashr object

Value

a magick image object

See Also

Other splash_dsl_functions: `splash_add_lua`, `splash_click`, `splash_focus`, `splash_go`, `splash_har_reset`, `splash_har`, `splash_html`, `splash_press`, `splash_release`, `splash_send_keys`, `splash_send_text`, `splash_wait`

Examples

```
## Not run:
splash_local %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_png()

## End(Not run)
```

splash_press *Trigger mouse press event in web page.*

Description

Trigger mouse press event in web page.

Usage

```
splash_press(splash_obj, x, y)
```

Arguments

splash_obj	splashr object
x, y	coordinates (distances from the left or top, relative to the current viewport)

See Also

Other splash_dsl_functions: splash_add_lua, splash_click, splash_focus, splash_go, splash_har_reset, splash_har, splash_html, splash_png, splash_release, splash_send_keys, splash_send_text, splash_wait

---

splash_private_mode *Enable or disable execution of JavaSript code embedded in the page.*

Description

Private mode is enabled by default unless you pass flag --disable-private-mode at Splash (server) startup. Note that if you disable private mode browsing data such as cookies or items kept in local storage may persist between requests.

Usage

```
splash_private_mode(splash_obj, enable = FALSE)
```

Arguments

splash_obj	splashr object
enable	logical

See Also

Other splash_dsl_attributes: splash_enable_javascript, splash_images, splash_plugins, splash_response_body

Examples

```
## Not run:
splash_local %>%
  splash_response_body(TRUE) %>%
  splash_private_mode(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_har() -> rud_har

## End(Not run)
```

splash_release *Trigger mouse release event in web page.*

Description

Trigger mouse release event in web page.

Usage

```
splash_release(splash_obj, x, y)
```

Arguments

splash_obj	splashr object
x, y	coordinates (distances from the left or top, relative to the current viewport)

See Also

Other splash_dsl_functions: splash_add_lua, splash_click, splash_focus, splash_go, splash_har_reset, splash_har, splash_html, splash_png, splash_press, splash_send_keys, splash_send_text, splash_wait

splash_response_body *Enable or disable response content tracking.*

Description

By default Splash doesn't keep bodies of each response in memory, for efficiency reasons.

Usage

```
splash_response_body(splash_obj, enable = FALSE)
```

Arguments

splash_obj	splashr object
enable	logical

See Also

Other splash_dsl_attributes: splash_enable_javascript, splash_images, splash_plugins, splash_private_mode

Examples

```
## Not run:
splash_local %>%
  splash_response_body(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_har() -> rud_har

## End(Not run)
```

---

splash_send_keys *Send keyboard events to page context.*

Description

- whitespace is ignored and only used to separate the different keys
- characters are literally represented

Usage

```
splash_send_keys(splash_obj, keys)
```

Arguments

splash_obj	splashr object
keys	string to send

Details

This is different from splash_send_text()

References

See the docs for more info

See Also

Other splash_dsl_functions: splash_add_lua, splash_click, splash_focus, splash_go, splash_har_reset, splash_har, splash_html, splash_png, splash_press, splash_release, splash_send_text, splash_wait

splash_send_text *Send text as input to page context, literally, character by character.*

Description

This is different from `splash_send_keys()`

Usage

```
splash_send_text(splash_obj, text)
```

Arguments

splash_obj	splashr object
text	string to send

Note

This adds a call to `splash:wait` so you do not have to

References

See the docs for more info

See Also

Other splash_dsl_functions: `splash_add_lua`, `splash_click`, `splash_focus`, `splash_go`, `splash_har_reset`, `splash_har`, `splash_html`, `splash_png`, `splash_press`, `splash_release`, `splash_send_keys`, `splash_wait`

splash_user_agent *Overwrite the User-Agent header for all further requests.*

Description

There are a few built-in user agents, all beginning with `ua_`.

Usage

```
splash_user_agent(splash_obj, user_agent = ua_splashr)

ua_splashr

ua_win10_chrome

ua_win10_firefox

ua_win10_ie11

ua_win7_chrome

ua_win7_firefox

ua_win7_ie11

ua_macos_chrome

ua_macos_safari

ua_linux_chrome

ua_linux_firefox

ua_ios_safari
```

Arguments

splash_obj	splashr object
user_agent	1 element character vector, defaults to splashr/#.#.#.

Format

An object of class character of length 1.

Examples

```
## Not run:
library(rvest)

URL <- "https://httpbin.org/user-agent"

splash_local %>%
  splash_response_body(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go(URL) %>%
  splash_html() %>%
  html_text("body") %>%
```

```
  jsonlite::fromJSON()

## End(Not run)
```

`splash_version` *Get Splash version information*

Description

Get Splash version information

Usage

```
splash_version(splash_obj = splash_local)
```

Arguments

`splash_obj`	A splash connection object

See Also

Other splash_info_functions: `splash_active`, `splash_debug`, `splash_history`, `splash_perf_stats`

Examples

```
## Not run:
sp <- splash()
splash_version(sp)

## End(Not run)
```

`splash_wait` *Wait for a period time*

Description

When script is waiting WebKit continues processing the webpage

Usage

```
splash_wait(splash_obj, time = 2)
```

Arguments

`splash_obj`	splashr object
`time`	number of seconds to wait

See Also

Other splash_dsl_functions: splash_add_lua, splash_click, splash_focus, splash_go, splash_har_reset, splash_har, splash_html, splash_png, splash_press, splash_release, splash_send_keys, splash_send_text

Examples

```
## Not run:
splash_local %>%
  splash_response_body(TRUE) %>%
  splash_user_agent(ua_macos_chrome) %>%
  splash_go("https://rud.is/b") %>%
  splash_wait(2) %>%
  splash_har() -> rud_har

## End(Not run)
```

---

start_splash *Start a Splash server Docker container*

Description

If using this in an automation context, you should consider adding a ‘Sys.sleep(3)‘ (or higher) after starting the docker container.

Usage

```
start_splash(tag = "3.0")
```

Arguments

tag	Splash Docker image tag to start

Value

‘docker‘ ‘container‘ object

Note

you need Docker running on your system and have pulled the container with [install_splash] for this to work. You should save the resultant object for use in [stop_splash] otherwise you’ll have to kill it from the command line interface.

See Also

Other splash_docker_helpers: install_splash, stop_splash

Examples

```
## Not run:
install_splash()
splash_container <- start_splash()
stop_splash(splash_container)

## End(Not run)
```

`stop_splash` *Stop a running a Splash server Docker container*

Description

Stop a running a Splash server Docker container

Usage

```
stop_splash(splash_container)
```

Arguments

`splash_container`
: Docker ‘container‘ object created by [start_splash()]

Note

you need Docker running on your system and have pulled the container with [install_splash()] and started the Splash container with [start_splash()] for this to work. You will need the ‘container‘ object from [start_splash()] for this to work.

See Also

Other splash_docker_helpers: `install_splash`, `start_splash`

Examples

```
## Not run:
install_splash()
splash_container <- start_splash()
stop_splash(splash_container)

## End(Not run)
```

Index