Package: websocket (via r-universe)

September 20, 2024

Version 1.4.2

Title 'WebSocket' Client Library

Description Provides a 'WebSocket' client interface for R. 'WebSocket' is a protocol for low-overhead real-time communication: <https://en.wikipedia.org/wiki/WebSocket>.

License GPL-2

Encoding UTF-8

ByteCompile true

Imports R6, later (>= 1.2.0)

LinkingTo cpp11, AsioHeaders, later

BugReports https://github.com/rstudio/websocket/issues

SystemRequirements GNU make, OpenSSL >= 1.0.2

RoxygenNote 7.3.2

Suggests httpuv, testthat, knitr, rmarkdown

VignetteBuilder knitr

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

RemoteUrl https://github.com/rstudio/websocket

RemoteRef HEAD

RemoteSha 380d4db886c2a377a6a9878ef90338b1ccc692ec

Contents

WebSocket *Create a WebSocket client*

Description

```
WebSocket$new(url,
  protocols = character(0),
  headers = NULL,
  autoConnect = TRUE,
  accessLogChannels = c("none"),
  errorLogChannels = NULL,
  maxMessageSize = 32 * 1024 * 1024)
```

Arguments

`url`	The WebSocket URL. Should begin with `ws://` or `wss://`.
`protocols`	Zero or more WebSocket sub-protocol names to offer to the server during the opening handshake.
`headers`	A named list or character vector representing keys and values of headers in the initial HTTP request.
`autoConnect`	If set to 'FALSE', then constructing the WebSocket object will not automatically cause the connection to be established. This can be used if control will return to R before event handlers can be set on the WebSocket object (i.e. you are constructing a WebSocket object manually at an interactive R console); after you are done attaching event handlers, you must call 'ws$connect()' to establish the WebSocket connection.
`accessLogChannels`	A character vector of access log channels that are enabled. Defaults to `"none"`, which displays no normal, websocketpp logging activity. Setting `accessLogChannels = NULL` will use default websocketpp behavior. Multiple access logging levels may be passed in for them to be enabled. A few commonly used access logging values are: `"all"` Special aggregate value representing "all levels" `"none"` Special aggregate value representing "no levels" `"rerror"` Recoverable error. Recovery may mean cleanly closing the connection with an appropriate error code to the remote endpoint. `"fatal"` Unrecoverable error. This error will trigger immediate unclean termination of the connection or endpoint. All logging levels are explained in more detail at https://docs.websocketpp.org/reference_8logging.html.
`errorLogChannels`	A character vector of error log channels that are displayed. The default value is `NULL`, which will use default websocketpp behavior. Multiple error logging levels may be passed in for them to be enabled. A few commonly used error logging values are:

"all" Special aggregate value representing "all levels"

"none" Special aggregate value representing "no levels"

"connect" One line for each new connection that is opened

"disconnect" One line for each new connection that is closed

All logging levels are explained in more detail at https://docs.websocketpp.org/reference_8logging.html.

maxMessageSize The maximum size of a message in bytes. If a message larger than this is sent, the connection will fail with the message_too_big protocol error.

Details

A WebSocket object has four events you can listen for, by calling the corresponding 'onXXX' method and passing it a callback function. All callback functions must take a single 'event' argument. The 'event' argument is a named list that always contains a 'target' element that is the WebSocket object that originated the event, plus any other relevant data as detailed below.

onMessage Called each time a message is received from the server. The event will have a 'data' element, which is the message content. If the message is text, the 'data' will be a one-element character vector; if the message is binary, it will be a raw vector.

onOpen Called when the connection is established.

onClose Called when a previously-opened connection is closed. The event will have 'code' (integer) and 'reason' (one-element character) elements that describe the remote's reason for closing.

onError Called when the connection fails to be established. The event will have an 'message' element, a character vector of length 1 describing the reason for the error.

Each 'onXXX' method can be called multiple times to register multiple callbacks. Each time an 'onXXX' is called, its (invisible) return value is a function that can be invoked to cancel that particular registration.

A WebSocket object also has the following methods:

connect() Initiates the connection to the server. (This does not need to be called unless you have passed 'autoConnect=FALSE' to the constructor.)

send(msg) Sends a message to the server.

close() Closes the connection.

readyState() Returns an integer representing the state of the connection.

- 0L: **Connecting** The WebSocket has not yet established a connection with the server.
- 1L: **Open** The WebSocket has connected and can send and receive messages.
- 2L: **Closing** The WebSocket is in the process of closing.
- 3L: **Closed** The WebSocket has closed, or failed to open.

setAccessLogChannels(channels) Enable the websocket Access channels after the websocket's creation. A value of NULL will not enable any new Access channels.

setErrorLogChannels(channels) Enable the websocket Error channels after the websocket's creation. A value of NULL will not enable any new Error channels.

clearAccessLogChannels(channels) Disable the websocket Access channels after the websocket's creation. A value of NULL will not clear any existing Access channels.

clearErrorLogChannels(channels) Disable the websocket Error channels after the websocket's creation. A value of NULL will not clear any existing Error channels.

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {

# Create a websocket using the websocket.org test server
ws <- WebSocket$new("ws://echo.websocket.org/")
ws$onMessage(function(event) {
  cat("Client got msg:", event$data, "\n")
})
ws$onClose(function(event) {
  cat("Client disconnected\n")
})
ws$onOpen(function(event) {
  cat("Client connected\n")
})

# Try sending a message with ws$send("hello").
# Close the websocket with ws$close() after you're done with it.
}
```

Index