Package 'QuickJSR'

November 24, 2023

Title Interface for the 'QuickJS' Lightweight 'JavaScript' Engine

Version 1.0.8

Description An 'R' interface to the 'QuickJS' portable 'JavaScript' engine.
The engine is bundled entirely within the package, requiring no external system dependencies beyond a 'C' compiler.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation yes

SystemRequirements GNU make

Imports jsonlite, R6, Rcpp

LinkingTo Rcpp

URL https://github.com/andrjohns/QuickJSR https://bellard.org/quickjs/ (upstream)

BugReports https://github.com/andrjohns/QuickJSR/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/testthat/parallel true

Language en-GB

Author Andrew R. Johnson [aut, cre] (<https://orcid.org/0000-0001-7000-8065>),
Fabrice Bellard [ctb, cph] (Author of QuickJS sources and headers),
Charlie Gordon [ctb, cph] (Author of QuickJS sources and headers)

Maintainer Andrew R. Johnson <andrew.johnson@arjohnsonau.com>

Repository CRAN

Date/Publication 2023-11-24 15:50:02 UTC

R topics documented:

`QuickJSR-package` *The QuickJSR package.*

Description

An interface to the QuickJS lightweight Javascript engine

`cxxflags` *cxxflags*

Description

Function for returning the C/C++ flags needed for compilation using the package's headers

Usage

```
cxxflags(to_console = FALSE)
```

Arguments

`to_console`	Whether the result should be returned as a string

Value

Character string of CXX flags, or print flags to console and invisibly return NULL (for use in package Makevars or similar)

`JSContext` *JSContext object*

Description

An initialised context within which to evaluate Javascript scripts or commands.

Value

A JSContext object containing an initialised JavaScript context for evaluating scripts/commands

Methods

Public methods:

- `JSContext$new()`
- `JSContext$validate()`
- `JSContext$source()`
- `JSContext$call()`
- `JSContext$clone()`

Method `new()`**:** Creates a new JSContext instance and initialises the QuickJS runtime and evaluation context

Usage:

```
JSContext$new(stack_size = NULL, disable_stack_size_check = TRUE)
```

Arguments:

`stack_size` An optional fixed value for the stack size (in bytes)

`disable_stack_size_check` Disable fixed/automatic stack size allocation.

Returns: No return value, used internally to initialise the JSContext object

Method `validate()`**:** Checks whether JS code string is valid code in the current context

Usage:

```
JSContext$validate(code_string)
```

Arguments:

`code_string` The JS code to check

Returns: A boolean indicating whether code is valid

Method `source()`**:** Evaluate a provided JavaScript file or string within the initialised context. Note that this method should only be used for initialising functions or values within the context, no values are returned from this function. See the `$call()` method for returning values.

Usage:

```
JSContext$source(file = NULL, code = NULL)
```

Arguments:

`file` A path to the JavaScript file to load

`code` A single string of JavaScript to evaluate

Returns: No return value, called for side effects

Method `call()`**:** Call a specified function in the JavaScript context with the provided arguments.

Usage:

```
JSContext$call(function_name, ...)
```

Arguments:

`function_name` The function to be called

`...` The arguments to be passed to the function

Returns: The result of calling the specified function, the return type is mapped from JS to R using `jsonlite::fromJSON()`

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ldflags` *ldflags*

Description

Function for returning the flags needed for linking to the package

Usage

```
ldflags(to_console = FALSE)
```

Arguments

`to_console`	Whether the result should be returned as a string

Value

Character string of linker flags, or print flags to console and invisibly return NULL (for use in package Makevars or similar)

qjs_eval *qjs_eval*

Description

Evaluate a single Javascript expression.

Usage

```
qjs_eval(eval_string)
```

Arguments

eval_string	A single string of the expression to evaluate

Value

The result of the provided expression, the return type is mapped from JS to R using `jsonlite::fromJSON()`

Examples

```
# Return the sum of two numbers:
qjs_eval("1 + 2")

# Concatenate strings:
qjs_eval("'1' + '2'")

# Create lists from objects:
qjs_eval("var t = {'a' : 1, 'b' : 2}; t")
```

Index