Package ‘js’

February 24, 2015

Type Package

Title Tools for Working with JavaScript in R

Version 0.2

Date 2015-02-23

Author Jeroen Ooms

Maintainer Jeroen Ooms <jeroen.ooms@stat.ucla.edu>

Description A set of utility functions for working with JavaScript in R.
Currently includes functions to compile, validate, reformat, optimize
and analyze JavaScript code.

License MIT + file LICENSE

URL https://github.com/jeroenooms/js

BugReports https://github.com/jeroenooms/js/issues

VignetteBuilder knitr

Imports V8 (>= 0.5)

Suggests knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-24 08:03:39

R topics documented:

`coffee_compile` *Coffee Script*

Description

Compiles coffee script into JavaScript.

Usage

```
coffee_compile(code, ...)
```

Arguments

`code`	a string with JavaScript code
`...`	additional options passed to the compiler

Examples

```
# Hello world
coffee_compile("square = (x) -> x * x")
coffee_compile("square = (x) -> x * x", bare = TRUE)

# Simple script
demo <- readLines(system.file("example/demo.coffee", package = "js"))
js <- coffee_compile(demo)
cat(js)
cat(uglify_optimize(js))
```

`jshint` *Static analysis tool for JavaScript*

Description

JSHint is a community-driven tool to detect errors and potential problems in JavaScript code. It is very flexible so you can easily adjust it to your particular coding guidelines and the environment you expect your code to execute in.

Usage

```
jshint(text, ..., globals = NULL)
```

Arguments

`text`	a string of JavaScript code
`...`	additional jshint configuration options
`globals`	a white list of global variables that are not formally defined in the source code

Value

a data frame where each row represents a jshint error or `NULL` if there were no errors

Examples

```
code = "var foo = 123"
jshint(code)
jshint(code, asi = TRUE)
```

`js_eval` *Evaluate JavaScript*

Description

Evaluate a piece of JavaScript code in a disposable context.

Usage

```
js_eval(text)
```

Arguments

`text`	JavaScript code

Examples

```
# Stateless evaluation
js_eval("(function() {return 'foo'})()")

# Use V8 for stateful evaluation
ct <- V8::new_context()
ct$eval("var foo = 123")
ct$get("foo")
```

`js_typeof` *Get the type of a JavaScript object*

Description

JavaScript wrapper to `typeof` to test if a piece of JavaScript code is syntactically valid, and the type of object it evaluates to. Useful to verify that a piece of JavaScript code contains a proper function/object.

Usage

```
js_typeof(text)
```

Arguments

`text`	JavaScript code

Examples

```
js_typeof("function(x){return x+1}")
js_typeof("(function() {return 'foo'})()")
js_typeof("{foo : 123, bar : true}")
```

`js_validate_script` *Validate JavaScript*

Description

Simple wrapper for `ct$validate` in `V8`. Tests if code constitutes a syntactically valid JS script.

Usage

```
js_validate_script(text, error = TRUE)
```

Arguments

`text`	character vector with JavaScript code
`error`	raise error on invalid code

Examples

```
js_validate_script("function foo(x){2*x}") #TRUE
js_validate_script("foo = function(x){2*x}") #TRUE

# Anonymous functions in global scope are invalid
js_validate_script("function(x){2*x}", error = FALSE) #FALSE

# Use ! or () to check anonymous function syntax
js_validate_script("!function(x){2*x}") #TRUE
js_validate_script("(function(x){2*x})") #TRUE
```

`uglify` *Compress and Reformat JavaScript Code*

Description

UglifyJS is a JavaScript compressor/minifier written in JavaScript. It also contains tools that allow one to automate working with JavaScript code.

Usage

```
uglify_reformat(text, beautify = FALSE, ...)

uglify_optimize(text, ...)
```

Arguments

`text`	a character vector with JavaScript code
`beautify`	prettify (instead of minify) code
`...`	additional arguments for the optimizer or generator.

References

UglifyJS2 Documentation: `http://lisperator.net/uglifyjs/`.

Examples

```
code <- "function test(x, y){ x = x || 1; y = y || 1; return x*y;}"
cat(uglify_optimize(code))
cat(uglify_reformat(code, beautify = TRUE, indent_level = 2))
```

Index