

Package: corvusregexecma262 (via r-universe)

October 22, 2026

Title 'ECMA-262' ('JavaScript') Regular Expressions

Version 0.2.0

Description Regular expressions with the grammar and the answers of 'ECMA-262', the 'RegExp' of 'JavaScript': every flag, lookbehind, named groups, modifier groups and the Unicode properties, so that a pattern answers in R as it does in a browser or 'Node.js'. Test, match, locate, count, replace and split over character vectors. A pattern without a backreference is matched in time proportional to its size times the text. The engine is the 'corvus-regex-ecma262' 'Rust' crate.

License Apache License (>= 2)

URL <https://github.com/corvus-dotnet/corvus-regex-ecma262-r>

BugReports <https://github.com/corvus-dotnet/corvus-regex-ecma262-r/issues>

Depends R (>= 4.2)

SystemRequirements Cargo (Rust's package manager), rustc (>= 1.89)

Encoding UTF-8

NeedsCompilation yes

Config/pak/sysreqs libclang-dev

Repository <https://corvus-dotnet.r-universe.dev>

Date/Publication 2026-10-22 15:53:05 UTC

RemoteUrl <https://github.com/corvus-dotnet/corvus-regex-ecma262-r>

RemoteRef v0.2.0

RemoteSha 3f638ea57c6bce81ec51176526705c164bbe9671

Contents

corvusregexecma262-package	2
ecma_match_all	3
ecma_regex	4
ecma_replace	5

ecma_split	6
ecma_test	6
ecma_version	7
Index	9

corvusregexecma262-package	
	<i>ECMA-262 (JavaScript) Regular Expressions</i>

Description

Regular expressions with the grammar and the answers of ECMA-262, the RegExp of JavaScript, so that a pattern answers in R as it does in a browser or Node.js. Every flag: d, g, i, m, s, u, v and y, with the grammar of each; lookbehind, named groups, modifier groups such as (?i:...), and every Unicode property ECMA-262 lists.

Compile a pattern with [ecma_regex](#), then use it with [ecma_test](#), [ecma_match](#), [ecma_locate](#), [ecma_match_all](#), [ecma_locate_all](#), [ecma_count](#), [ecma_replace](#) and [ecma_split](#), each over a character vector. Each also takes a pattern as a string, compiled without flags.

Strings and positions

R's strings are matched as UTF-8: one marked as UTF-8, or in the native encoding of a UTF-8 session, is read in place (after a check that it is UTF-8); one in another encoding is translated. A string marked as bytes is refused. Positions are in characters from 1, as nchar counts them, an end inclusive, as regexpr gives them.

Without u or v a pattern is matched by UTF-16 code unit, as ECMA-262 matches it, so a match can take half of a character beyond the Basic Multilingual Plane. Its text then holds that half as U+FFFD, and its positions are the whole character's.

Conditions

A pattern that is not valid raises `ecma_invalid_pattern`; flags that are not, `ecma_invalid_flags`; a pattern over a limit of size, `ecma_pattern_too_large`; a match that reaches its timeout or its cap on backtracking memory, `ecma_match_limit`; a string that is not UTF-8 or is marked as bytes, `ecma_encoding_error`; and an argument of the wrong kind, `ecma_argument_error`. All inherit from `ecma_regex_error`.

See Also

<https://github.com/corvus-dotnet/corvus-regex-ecma262-r>

ecma_match_all	<i>Every Match of a Global Search</i>
----------------	---------------------------------------

Description

For each string, `ecma_match_all` gives every match of a global search and what each group captured (`matchAll`), `ecma_locate_all` where each match, or one of its groups, starts and ends, and `ecma_count` how many there are. After an empty match the search goes on one character later (one code unit without `u` or `v`); with the `y` flag each match must start where the one before it ended.

Usage

```
ecma_match_all(x, regex)
```

```
ecma_locate_all(x, regex, group = 0L)
```

```
ecma_count(x, regex)
```

Arguments

<code>x</code>	A character vector.
<code>regex</code>	A regular expression from ecma_regex , or a pattern compiled without flags. Whether or not it has the <code>g</code> flag, every match is found.
<code>group</code>	The group to locate in each match, by number (0 is the whole match) or by name.

Value

`ecma_match_all` returns a list with a character matrix for each string, laid out as [ecma_match](#)'s with a row for each match. `ecma_locate_all` returns a list with an integer matrix for each string, with the columns `start` and `end` (NA for a group that did not take part). Each is NULL for an NA string. `ecma_count` returns an integer vector, NA for an NA string; it makes no match, and is the fastest way to know how many there are.

Examples

```
ecma_match_all("1 22 333", "\\d+")
ecma_locate_all("x1y22z333", "\\d+")
ecma_count(c("a b a", "", NA), "a")
```

ecma_regex

*Compile an ECMA-262 Regular Expression***Description**

ecma_regex compiles a pattern as new RegExp(pattern, flags) does, with a match timeout. ecma_valid says whether each pattern is valid with the flags, without compiling it. ecma_set_default_timeout sets the timeout of every pattern compiled without one, and ecma_default_timeout gives it.

Usage

```
ecma_regex(pattern, flags = "", timeout = NULL)
```

```
ecma_valid(pattern, flags = "")
```

```
ecma_set_default_timeout(seconds)
```

```
ecma_default_timeout()
```

```
## S3 method for class 'ecma_regex'
print(x, ...)
```

```
## S3 method for class 'ecma_regex'
format(x, ...)
```

Arguments

pattern	For ecma_regex, one string; for ecma_valid, a character vector.
flags	The flags, such as "", "u" or "gimsuy": for ecma_valid, recycled.
timeout	The time in seconds each match may take before it is abandoned with ecma_match_limit, 0 for no limit, or NULL for ecma_default_timeout().
seconds	The default timeout in seconds, or NULL (or 0) for no limit.
x	A compiled regular expression.
...	Not used.

Details

The g flag makes [ecma_replace](#) replace every match, and the y flag makes each search match where it starts. The d flag is accepted and changes nothing, since every match gives its groups' positions.

A pattern has a match timeout, as a Regex of .NET has: every function that matches gives each of its matches the timeout, and a match that reaches it, or the cap on backtracking memory that comes with it, raises ecma_match_limit. A pattern compiled without one takes ecma_default_timeout(), which is NULL (no limit) unless ecma_set_default_timeout set it. Without a timeout a match has no limit of time or of memory, so give one to a pattern or a text from outside. A pattern given as a string to the other functions takes the default.

A compiled regular expression is not saved with the session: compile it again in a new one.

Value

ecma_regex returns an object of class ecma_regex, with the attributes source, flags (in the order RegExp gives them), group_count and group_names (the distinct names of the groups, in the order of the first group of each) and timeout (NULL for none). It is used by every other function, and from any number of calls. ecma_default_timeout returns the default timeout in seconds, or NULL. ecma_valid returns a logical vector, NA for an NA pattern.

Examples

```
date <- ecma_regex("(?<year>\\d{4})-(?<month>\\d{2})", "g")
date
attr(date, "group_names")
ecma_valid(c("a+", "a{2,1}", "("), "u")
timed <- ecma_regex("^(?:(a+)\\1*)+$", "u", timeout = 0.05)
attr(timed, "timeout")
```

ecma_replace	<i>Replace Matches</i>
--------------	------------------------

Description

ecma_replace replaces the first match in each string, or every match when the pattern has the g flag (String.prototype.replace); ecma_replace_all replaces every match (replaceAll).

Usage

```
ecma_replace(x, regex, replacement)

ecma_replace_all(x, regex, replacement)
```

Arguments

x	A character vector.
regex	A regular expression from ecma_regex , or a pattern compiled without flags.
replacement	A character vector of templates (recycled), or a function. In a template, && is the match, \$` and \$' the text before and after it, \$1 to \$99 a group, \$<name> a named group, and \$\$ a dollar sign; anything else is itself. A function is called with each match's row, as ecma_match lays it out (a named character vector), and returns one string.

Value

A character vector, NA where the string or its template is NA. A match that takes half of a character leaves the other half as U+FFFD.

Examples

```
date <- ecma_regex("(?<year>\\d{4})-(?<month>\\d{2})", "g")
ecma_replace("2026-10, 2027-01", date, "$<month>/$1")
ecma_replace_all("1 2 3", "\\d+", function(m) as.character(as.integer(m[[1]]) * 2))
```

ecma_split	<i>Split Strings Around Matches</i>
------------	-------------------------------------

Description

Splits each string around the matches, as `String.prototype.split` does: what each group captured is spliced in after the piece before it, and an empty match at the start or the end of the string splits nothing.

Usage

```
ecma_split(x, regex, limit = NULL)
```

Arguments

- `x` A character vector.
- `regex` A regular expression from [ecma_regex](#), or a pattern compiled without flags.
- `limit` The most pieces to give, or `NULL` for every one.

Value

A list with a character vector for each string, `NA` for a group that did not take part, and `NULL` for an `NA` string.

Examples

```
ecma_split("a ; b;c", "\\s*(;)\\"s*")
ecma_split("a,b", "(x)?,")
```

ecma_test	<i>Test, Match and Locate the First Match</i>
-----------	---

Description

For each string, `ecma_test` says whether the pattern matches (`RegExp.prototype.test`), `ecma_match` gives the first match and what each group captured (`exec`), and `ecma_locate` gives where the first match, or one of its groups, starts and ends.

Usage

```
ecma_test(x, regex, start = 1L, timeout = NULL)

ecma_match(x, regex, start = 1L, timeout = NULL)

ecma_locate(x, regex, group = 0L, start = 1L, timeout = NULL)
```

Arguments

x	A character vector.
regex	A regular expression from ecma_regex , or a pattern compiled without flags.
start	Where each search starts, in characters from 1 (recycled), as lastIndex sets it: with the y flag a match must start there.
timeout	A limit in seconds on the time this match may take in place of the pattern's timeout (0 for none), or NULL for the pattern's.
group	The group to locate, by number (0 is the whole match) or by name.

Value

ecma_test returns a logical vector. ecma_match returns a character matrix with a row for each string: the match, then each group, then one column for each group name (named by it: of the groups that share a name, the one that took part). ecma_locate returns an integer matrix with the columns start and end. An empty match ends one character before it starts. There is NA for an NA string, for a string the pattern does not match, and for a group that did not take part.

Limits

A match that reaches the pattern's timeout (see [ecma_regex](#)) or timeout, or the cap on backtracking memory, raises ecma_match_limit. A pattern without a backreference is matched in time proportional to its size times the text.

Examples

```
date <- ecma_regex("(?<year>\\d{4})-(?<month>\\d{2})")
ecma_test(c("due 2026-10", "none", NA), date)
ecma_match("due 2026-10", date)
ecma_locate("due 2026-10", date, group = "month")
ecma_test(c("foo", "foo"), ecma_regex("o", "y"), start = c(1L, 2L))
```

ecma_version	<i>Versions</i>
--------------	-----------------

Description

The version of this package's native code and of the corvus-regex-ecma262 crate it was built from.

Usage

```
ecma_version()
```

Value

A character vector with the elements package and crate.

Examples

```
ecma_version()
```

Index

* package

corvusregexecma262-package, [2](#)

corvusregexecma262

(corvusregexecma262-package), [2](#)

corvusregexecma262-package, [2](#)

ecma_count, [2](#)

ecma_count (ecma_match_all), [3](#)

ecma_default_timeout (ecma_regex), [4](#)

ecma_locate, [2](#)

ecma_locate (ecma_test), [6](#)

ecma_locate_all, [2](#)

ecma_locate_all (ecma_match_all), [3](#)

ecma_match, [2](#), [3](#), [5](#)

ecma_match (ecma_test), [6](#)

ecma_match_all, [2](#), [3](#)

ecma_regex, [2](#), [3](#), [4](#), [5–7](#)

ecma_replace, [2](#), [4](#), [5](#)

ecma_replace_all (ecma_replace), [5](#)

ecma_set_default_timeout (ecma_regex), [4](#)

ecma_split, [2](#), [6](#)

ecma_test, [2](#), [6](#)

ecma_valid (ecma_regex), [4](#)

ecma_version, [7](#)

format.ecma_regex (ecma_regex), [4](#)

print.ecma_regex (ecma_regex), [4](#)