

JavaScript Regular Expressions

SINGLE CHARACTERS

Use	To match any character
[<i>set</i>]	In that set
[^ <i>set</i>]	Not in that set
[<i>a-z</i>]	In the <i>a-z</i> range
[^ <i>a-z</i>]	Not in the <i>a-z</i> range
.	Any except \n (new line)
\char	Escaped special character

CONTROL CHARACTERS

Use	To match	Unicode
\t	Horizontal tab	\u0009
\v	Vertical tab	\u000B
\b	Backspace	\u0008
\e	Escape	\u001B
\r	Carriage return	\u000D
\f	Form feed	\u000C
\n	New line	\u000A
\a	Bell (alarm)	\u0007

NON-ASCII CODES

Use	To match character with
\x hex	2-digit hex character code
\u hex	4-digit hex character code

CHARACTER CLASSES

Use	To match character
\w	Word character. [0-9_a-zA-Z] and Unicode word characters
\W	Non-word character
\d	Decimal digit and Unicode digits
\D	Not a decimal digit
\s	White-space character [\t\n\r\f\v] and Unicode spaces
\S	Non-white-space char
\p{ctgry}	Unicode category or block
\P{ctgry}	Not in that Unicode category or block

QUANTIFIERS

Greedy	Lazy	Matches
*	*?	0 or more times
+	+	1 or more times
?	??	0 or 1 time
{ <i>n</i> }	{ <i>n</i> }?	Exactly <i>n</i> times
{ <i>n</i> ,}	{ <i>n</i> ,}?	At least <i>n</i> times
{ <i>n</i> , <i>m</i> }	{ <i>n</i> , <i>m</i> }?	From <i>n</i> to <i>m</i> times

ANCHORS

Use	To specify position
^	At start of string or line
\$	At end of string or line
\b	On word boundary
\B	Not on word boundary

GROUPS

Use	To define
(<i>exp</i>)	Indexed group
(?< <i>name</i> > <i>exp</i>)	Named group
(?: <i>exp</i>)	Non-capturing group
(?= <i>exp</i>)	Zero-width positive lookahead
(?! <i>exp</i>)	Zero-width negative lookahead
(?<= <i>exp</i>)	Zero-width positive lookbehind. <i>exp</i> is fixed width
(?!< <i>exp</i>)	Zero-width negative lookbehind. <i>exp</i> is fixed width

INLINE OPTIONS

Option	Effect on match
i	Case-insensitive
m	Multiline mode
g	Global
u	Unicode dependent
s	Dot . wildcard character matches new line
x	Ignore white space

Updated: October 2020

Chandra Lingam, Cloud Wave LLC
<https://github.com/ChandraLingam/PyRegex>
[Microsoft/MSDN .NET Regular Expressions \(Template\)](#)
[Mozilla JavaScript Regex Syntax Cheat sheet](#)

BACKREFERENCES

Use	To match
<code>\n</code>	Indexed group
<code>\k<name></code>	Named group

ALTERNATION

Use	To match
<code>a b</code>	Either <i>a</i> or <i>b</i>

REPLACEMENT

Use	To substitute
<code>\$n</code>	Substring matched by group number <i>n</i>
<code>\$<name></code>	Substring matched by group <i>name</i>

REGULAR EXPRESSION OPERATIONS

Class: `RegExp`, `String`

Pattern matching with Compiled objects

To initialize with	Use constructor
Pattern	<code>RegExp(pattern)</code>
+ flags	<code>RegExp(pattern, flags)</code>

Finding and replacing matched patterns

Use method	To
<code>re.exec</code>	Iterate all matches (/g)
<code>re.test</code>	Test for a match (boolean)
<code>string.search</code>	Index of first match
<code>string.match</code>	Retrieve all matching strings
<code>string.matchall</code>	Iterate all matches
<code>string.replace</code>	Replace a matching string
<code>string.split</code>	Split text based on match

Getting info about regular expression patterns

Use compiled object API	To get
<code>lastIndex</code>	Index location where last match ended. Valid when global flag is set
<code>source</code>	Pattern for compiled object

Processing a match

Use method	To
<code>[n]</code>	Retrieve value of a group by number
<code>groups</code>	Retrieve all subgroups as name-value pairs
<code>index</code>	Find starting index position of a match
<code>length</code>	Find the number of indexed groups

Updated: October 2020

Chandra Lingam, Cloud Wave LLC
<https://github.com/ChandraLingam/PyRegex>
[Microsoft/MSDN .NET Regular Expressions \(Template\)](#)
[Mozilla JavaScript Regex Syntax Cheat sheet](#)