

Regular Expression in Javascript

JavaScript Regex

In JavaScript, a Regular Expression (Regex) is an object that describes a sequence of characters used for defining a search pattern. For example,

```
/^a...s$/
```

The above code defines a Regex pattern. The pattern is: any five letter string starting with a and ending with s.

A pattern defined using Regex can be used to match against a string.

Expression	String	Matched?
<code>/^a...s\$/</code>	<code>abs</code>	No match
	<code>alias</code>	Match
	<code>abyss</code>	Match
	<code>Alias</code>	No match
	<code>An abacus</code>	No match

There are two ways you can create a regular expression in JavaScript.

1. Using a regular expression literal:

The regular expression consists of a pattern enclosed between slashes /. For example,

```
const regularExp = /abc/;
```

Here, /abc/ is a regular expression.

2. Using the RegExp() constructor function:

You can also create a regular expression by calling the RegExp() constructor function. For example,

```
const regex = new RegExp(/^a...s$/);
```

```
console.log(regex.test('alias')); // true
```

In the above example, the string alias matches with the RegEx pattern `/^a...s$/`.

Here, the `test()` method is used to check if the string matches the pattern.

Specify Pattern Using RegEx

MetaCharacters

Metacharacters are characters that are interpreted in a special way by a RegEx engine.

[] . ^ \$ * + ? { } () \ |

[] - Square brackets- specify a set of characters we wish to match

Expression	String	Matched?
[abc]	a	1 match
	ac	2 matches
	Hey Jude	No match
	abc de ca	5 matches

Also specify a range of characters using - inside square brackets.

[a-e] is the same as [abcde].

[1-4] is the same as [1234].

[0-39] is the same as [01239].

Complement (invert) the character set by using caret ^ symbol at the start of a square-bracket.

[^abc] means any character except a or b or c.

[^0-9] means any non-digit character.

^ - Caret

The caret symbol ^ is used to check if a string starts with a certain character.

Expression	String	Matched?
^a	a	1 match
	abc	1 match
	bac	No match
^ab	abc	1 match
	acb	No match (starts with a but not followed by b)

A. var pattern = /^hello/;

console.log(pattern.test("hello world")); // Output: true

console.log(pattern.test("hi hello")); // Output: false

B. var pattern = /^[^hello]/;

console.log(pattern.test("hello world")); // Output: false

\$ - Dollar

The dollar symbol \$ is used to check if a string ends with a certain character.

Expression	String	Matched?
a\$	a	1 match
	formula	1 match
	cab	No match

* - Star

The star symbol * matches zero or more occurrences of the pattern left to it.

Matches the preceding element zero or more times.

Expression	String	Matched?
ma*n	mn	1 match
	man	1 match
	mann	1 match
	main	No match (a is not followed by n)
	woman	1 match

+ - Plus

The plus symbol + matches one or more occurrences of the pattern left to it.

Expression	String	Matched?
ma+n	mn	No match (no a character)
	man	1 match
	mann	1 match
	main	No match (a is not followed by n)
	woman	1 match

? - Question Mark

The question mark symbol ? matches zero or one occurrence of the pattern left to it.

Matches the preceding element either zero or one time.

Expression	String	Matched?
ma?n	mn	1 match
	man	1 match
	maan	No match (more than one a character)
	main	No match (a is not followed by n)
	woman	1 match

{ } - Braces

Consider this code: {n,m}. This means at least n, and at most m repetitions of the pattern left to it.

Expression	String	Matched?
a{2,3}	abc dat	No match
	abc daat	1 match (at <u>daa</u> t)
	aabc daaat	2 matches (at <u>aabc</u> and <u>daaat</u>)
	aabc daaaat	2 matches (at <u>aabc</u> and <u>daaaat</u>)

This RegEx `[0-9]{2,4}` matches at least 2 digits but not more than 4 digits.

Expression	String	Matched?
	ab123csde	1 match (match at <u>ab123</u> csde)
<code>[0-9]{2,4}</code>	12 and 345673	3 matches (<u>12</u> , <u>3456</u> , <u>73</u>)
	1 and 2	No match

| - Alternation

Vertical bar | is used for alternation (or operator).

Expression	String	Matched?
a b	cde	No match
	ade	1 match (match at <u>a</u> de)
	acdbea	3 matches (at <u>a</u> <u>c</u> <u>d</u> bea)

Here, a|b match any string that contains either a or b

() - Group

Parentheses () is used to group sub-patterns. For example, (a|b|c)xz match any string that matches either a or b or c followed by xz

Expression	String	Matched?
	ab xz	No match
(a b c)xz	abxz	1 match (match at <u>abxz</u>)
	axz cabxz	2 matches (at <u>axzbc</u> <u>cabxz</u>)

\ - Backslash

Backslash \ is used to escape various characters including all metacharacters. For example,

\\$a match if a string contains \$ followed by a. Here, \$ is not interpreted by a RegEx engine in a special way.

Special Sequences

Special sequences make commonly used patterns easier to write. Here's a list of special sequences:

`\A` - Matches if the specified characters are at the start of a string.

Expression	String	Matched?
<code>\Athe</code>	the sun	Match
	In the sun	No match