JavaScript Regular Expressions

A regular expression is a sequence of characters that forms a search pattern.

The search pattern can be used for text search and text replace operations.

What Is a Regular Expression?

A regular expression is a sequence of characters that forms a **search pattern**.

When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of **text search** and **text replace** operations.

Syntax

/pattern/modifiers;

Example

/w3schools/i;

Example explained:

/w3schools/i is a regular expression.

w3schools is a pattern (to be used in a search).

i is a modifier (modifies the search to be case-insensitive).

Using String Methods

In JavaScript, regular expressions are often used with the two **string methods**: search() and replace().

The search() method uses an expression to search for a match, and returns the position of the match.

The replace() method returns a modified string where the pattern is replaced.

Using String search() With a String

The search() method searches a string for a specified value and returns the position of the match:

Example

Use a string to do a search for "W3schools" in a string:

```
let text = "Visit W3Schools!";
let n = text.search("W3Schools");
```

The result in *n* will be:

6

Try it Yourself »

Using String search() With a Regular Expression

Example

Use a regular expression to do a case-insensitive search for "w3schools" in a string:

```
let text = "Visit W3Schools";
let n = text.search(/w3schools/i);
```

The result in *n* will be:

6

Try it Yourself »

Using String replace() With a String

The replace() method replaces a specified value with another value in a string:

```
let text = "Visit Microsoft!";
let result = text.replace("Microsoft", "W3Schools");
```

Try it Yourself »

Use String replace() With a Regular Expression

Example

Use a case insensitive regular expression to replace Microsoft with W3Schools in a string:

```
let text = "Visit Microsoft!";
let result = text.replace(/microsoft/i, "W3Schools");
```

The result in *res* will be:

Visit W3Schools!

Try it Yourself »

Did You Notice?

Regular expression arguments (instead of string arguments) can be used in the methods above.
Regular expressions can make your search much more powerful (case insensitive for example).

Regular Expression Modifiers

Modifiers can be used to perform case-insensitive more global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all)
m	Perform multiline matching
d	Perform start and end matching (New in ES2022)

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

Metacharacters are characters with a special meaning:

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word li this: WORD\b

\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*

Using the RegExp Object

In JavaScript, the RegExp object is a regular expression object with predefined properties and methods.

Using test()

The test() method is a RegExp expression method.

It searches a string for a pattern, and returns true or false, depending on the result.

The following example searches a string for the character "e":

Example

```
const pattern = /e/;
pattern.test("The best things in life are free!");
```

Since there is an "e" in the string, the output of the code above will be:

true

Try it Yourself »

You don't have to put the regular expression in a variable first. The two lines above can be shortened to one:

```
/e/.test("The best things in life are free!");
```

Using exec()

The exec() method is a RegExp expression method.

It searches a string for a specified pattern, and returns the found text as an object.

If no match is found, it returns an empty *(null)* object.

The following example searches a string for the character "e":

Example

```
/e/.exec("The best things in life are free!");
```

Try it Yourself »

Complete RegExp Reference

For a complete reference, go to our Complete JavaScript RegExp Reference.

The reference contains descriptions and examples of all RegExp properties and methods.