Example

Do a global search for the character-span from uppercase "A" to lowercase "e" (will search for all uppercase letters, but only lowercase letters from a to e.)

```
var str = "I Scream For Ice Cream, is that OK?!";
var patt1 = /[A-e]/g;
```

Do a global, case-insensitive search for the character-span [a-s]:

```
var str = "I Scream For Ice Cream, is that OK?!";
var patt1 = /[a-s]/gi;
```

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global, case-insensitive search for the character-
span [a-s].</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "I Scream For Ice Cream, is that OK?!";
  var patt1 = /[a-s]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global, case-insensitive search for the character-span [a-s].

Try it

I,S,c,r,e,a,m,F,o,r,I,c,e,C,r,e,a,m,i,s,h,a,O,K

Example

A demonstration of "g" and "gi"-search for characters:

```
var str = "THIS This this";
var patt1 = /[THIS]/g;

var str = "THIS This this";
var patt1 = /[THIS]/gi;
```

```
a string.</p>

<button onclick="myFunction()">Global case-sensitive search</button>
<button onclick="myFunction2()">Global case-insensitive search</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "THIS This this";
  var patt1 = /[THIS]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

<script>
function myFunction2() {
  var str = "THIS This this";
  var patt1 = /[THIS]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click one of the buttons to perform a search for the characters "THIS" in a string.

Global case-sensitive search | Global case-insensitive search

T,H,I,S,T,h,i,s,t,h,i,s

JavaScript RegExp [^abc] Expression

‹ JavaScript RegExp Object

Example

Do a global search for characters that are NOT inside the brackets [h]:

```
var str = "Is this all there is?";
var patt1 = /[^h]/g;
```

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for characters NOT inside the
brackets [h] in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Is this all there is?";
  var patt1 = /[^h]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for characters NOT inside the brackets [h] in a string.

Try it

I,s, ,t,i,s, ,a,l,l, ,t,e,r,e, ,i,s,?

Syntax

```
new RegExp("[^xyz]")

or simply:

/[^xyz]/
```

Syntax with modifiers

```
new RegExp("[^xyz]", "g")

or simply:

/\[^xyz]/g
```

Example

Do a global search for characters that are NOT "i" and "s" in a string:

```
var str = "Do you know if this is all there is?";
var patt1 = /[^is]/gi;
```

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for characters that are NOT "i" and "s" in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Do you know if this is all there is?";
  var patt1 = /[^is]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for characters that are NOT "i" and "s" in a string.

Try it

D,o, ,y,o,u, ,k,n,o,w, ,f, ,t,h, , ,a,l,l, ,t,h,e,r,e, ,?

Do a global search for the character-span NOT from lowercase "a" to lowercase "h" in a string:

```
var str = "Is this all there is?";
var patt1 = /[^a-h]/g;
```

Do a global search for the character-span NOT from uppercase "A" to uppercase "E":

```
var str = "I SCREAM FOR ICE CREAM!";
var patt1 = /[^A-E]/g;
```

Do a global search for the character-span NOT from uppercase "A" to lowercase "e":

```
var str = "I Scream For Ice Cream, is that OK?!";
var patt1 = /[^A-e]/g;
```

Do a global, case-insensitive search for the character-span that's NOT [a-s]:

```
var str = "I Scream For Ice Cream, is that OK?!";
var patt1 = /[^a-s]/gi;
```

JavaScript RegExp [0-9] Expression

‹ JavaScript RegExp Object

Example

Do a global search for the numbers 1, 2, 3 and 4 in a string:

```
var str = "123456789";
var patt1 = /[1-4]/g;
```

JavaScript RegExp (x|y) Expression

‹ JavaScript RegExp Object

Example

Do a global search to find any of the specified alternatives (red|green):

```
var str = "re, green, red, green, gren, gr, blue, yellow";
var patt1 = /(red|green)/g;
```

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for any of the specified alternatives
(red|green).</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "re, green, red, green, gren, gr, blue, yellow";
  var patt1 = /(red|green)/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for any of the specified alternatives (red|green).

Try it

green,red,green

Do a global search to find any of the specified alternatives (0|5|7):

```
var str = "01234567890123456789";
var patt1 = /(0|5|7)/g;
```

Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character

\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

JavaScript RegExp . Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for "h.t" in a string:

```
var str = "That's hot!";
var patt1 = /h.t/g;
```

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for "h.t" in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "That's hot!";
  var patt1 = /h.t/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for "h.t" in a string.

Try it

hat,hot

Definition and Usage

The . metacharacter is used to find a single character, except newline or other line terminators.

JavaScript RegExp \w Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for word characters in a string:

```
var str = "Give 100%!";
var patt1 = /\w/g;
```

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for word characters in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Give 100%!";
  var patt1 = /\w/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for word characters in a string.

Try it

G,i,v,e,1,0,0

JavaScript RegExp \d Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for digits:

```
var str = "Give 100%!";
var patt1 = /\d/g;
```

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for digits in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Give 100%!";
  var patt1 = /\d/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for digits in a string.

Try it

1,0,0

JavaScript RegExp \D Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for non-digit characters:

```
var str = "Give 100%!";
var patt1 = /\D/g;
```

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to do a global search for non-digit characters in
a string.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var str = "Give 100%!";
var patt1 = /\D/g;
var result = str.match(patt1);
document.getElementById("demo").innerHTML=result;
}
</script>

</body>
</html>
```

G,i,v,e, ,%,!

Try it

JavaScript RegExp \s Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for whitespace characters in a string:

```
var str = "Is this all there is?";
var patt1 = /\s/g;
```

Definition and Usage

The \s metacharacter is used to find a whitespace character.

A whitespace character can be:

- A space character
- A tab character
- A carriage return character
- A new line character
- A vertical tab character
- A form feed character

Syntax

```
new RegExp("\\s")

or simply:

/\s/
```

Syntax with modifiers

```
new RegExp("\\s", "g")

or simply:

/\s/g
```

JavaScript RegExp \S Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for non-whitespace characters in a string:

```
var str = "Is this all there is?";
var patt1 = /\S/g;
```

Syntax

```
new RegExp("\\S")

or simply:

/\S/
```

Syntax with modifiers

```
new RegExp("\\S", "g")

or simply:

/\S/g
```

JavaScript RegExp \b Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for "W3" at the beginning or end of a word in a string:

```
var str = "Visit W3Schools";
var patt1 = /\bW3/g;
```

JavaScript RegExp \B Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for "Schools" NOT at the beginning or end of a word in a string:

```
var str = "Visit W3Schools";
var patt1 = /\BSchool/g;
```

JavaScript RegExp \0 Metacharacter

‹ JavaScript RegExp Object

Example

Search for a NUL character in a string:

```
var str = "Visit W3Schools.\0Learn Javascript.";
var patt1 = /\0/;
```

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to return the position where the NUL character was found in
a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Visit W3Schools.\0Learn JavaScript.";
  var patt1 = /\0/;
  var result = str.search(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to return the position where the NUL character was found in a string.

Try it

16

JavaScript RegExp \n Metacharacter

‹ JavaScript RegExp Object

Example

Search for a newline character in a string:

```
var str = "Visit W3Schools.\nLearn Javascript.";
var patt1 = /\n/;
```

Definition and Usage

The \n character is used to find a newline character.

\n returns the position where the newline character was found. If no match is found, it returns -1.

JavaScript RegExp \f Metacharacter

‹ JavaScript RegExp Object

Example

Search for a form feed character in a string:

```
var str = "Visit W3Schools.\fLearn Javascript.";
var patt1 = /\f/;
```

Definition and Usage

The \f metacharacter is used to find a form feed character.

\f returns the position where the form feed character was found. If no match is found, it returns -1.

JavaScript RegExp \r Metacharacter

‹ JavaScript RegExp Object

Example

Search for a carriage return character in a string:

```
var str = "Visit W3Schools.\rLearn Javascript.";
var patt1 = /\r/;
```

JavaScript RegExp \t Metacharacter

‹ JavaScript RegExp Object

Example

Search for a tab character in a string:

```
var str = "Visit W3Schools.\tLearn Javascript.";
var patt1 = /\t/;
```

JavaScript RegExp \v Metacharacter

‹ JavaScript RegExp Object

Example

Search for a vertical tab character in a string:

```
var str = "Visit W3Schools.\vLearn Javascript.";
var patt1 = /\v/;
```

JavaScript RegExp \xdd Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for the hexadecimal number 57 (W) in a string:

```
var str = "Visit W3Schools. Hello World!";
var patt1 = /\x57/g;
```

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for the hexadecimal number 57 (W) in
a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Visit W3Schools. Hello World!";
  var patt1 = /\x57/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for the hexadecimal number 57 (W) in a string.

Try it

W,W

JavaScript RegExp \uxxxx Metacharacter

‹ JavaScript RegExp Object

Example

Do a global search for the hexadecimal number 0057 (W) in a string:

```
var str = "Visit W3Schools. Hello World!";
var patt1 = /\u0057/g;
```

Definition and Usage

The \uxxxx character is used to find the Unicode character specified by a hexadecimal number xxxx.

If no match is found, it returns null.

Quantifiers

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*
n{X}	Matches any string that contains a sequence of *X* *n*'s
n{X,Y}	Matches any string that contains a sequence of X to Y *n*'s
n{X,}	Matches any string that contains a sequence of at least X *n*'s
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it
?=n	Matches any string that is followed by a specific string *n*
?!n	Matches any string that is not followed by a specific string *n*

JavaScript RegExp + Quantifier

‹ JavaScript RegExp Object

Example 1

Do a global search for at least one "o":

```
var str = "Hellooo World! Hello W3Schools!";
var patt1 = /o+/g;
```

JavaScript RegExp * Quantifier

‹ JavaScript RegExp Object

Example 1

Do a global search for an "l", followed by zero or more "o" characters:

```
var str = "Hellooo World! Hello W3Schools!";
var patt1 = /lo*/g;
```

JavaScript RegExp ? Quantifier

‹ JavaScript RegExp Object

Example

Do a global search for a "1", followed by zero or one "0" characters:

```
var str = "1, 100 or 1000?";
var patt1 = /10?/g;
```

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for a "1", followed by zero or one "0" characters.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "1, 100 or 1000?";
  var patt1 = /10?/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for a "1", followed by zero or one "0" characters.

Try it

1,10,10

JavaScript RegExp {X} Quantifier

‹ JavaScript RegExp Object

Example

Do a global search for a substring that contains a sequence of four digits:

```
var str = "100, 1000 or 10000?";
var patt1 = /\d{4}/g;
```

JavaScript RegExp {X,Y} Quantifier

‹ JavaScript RegExp Object

Example

Do a global search for a substring that contains a sequence of three to four digits:

```
var str = "100, 1000 or 10000?";
var patt1 = /\d{3,4}/g;
```

JavaScript RegExp {X,} Quantifier

‹ JavaScript RegExp Object

Example

Do a global search for a substring that contains a sequence of at least three digits:

```
var str = "100, 1000 or 10000?";
var patt1 = /\d{3,}/g;
```

JavaScript RegExp $ Quantifier

‹ JavaScript RegExp Object

RegExp Object Reference

Example

Do a global search for "is" at the end of a string:

```
var str = "Is this his";
var patt1 = /is$/g;
```

JavaScript RegExp ^ Quantifier

‹ JavaScript RegExp Object

Example

Do a global search for "Is" at the beginning of a string:

```
var str = "Is this his";
var patt1 = /^Is/g;
```

JavaScript RegExp ?= Quantifier

‹ JavaScript RegExp Object

Example

Do a global search for "is" followed by " all":

```
var str = "Is this all there is";
var patt1 = /is(?= all)/g;
```

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to global search for "is" followed by " all".</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Is this all there is";
  var patt1 = /is(?= all)/;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to global search for "is" followed by " all".

Try it

is

JavaScript RegExp ?! Quantifier

‹ JavaScript RegExp Object

Example

Do a global, case insensitive search for "is" not followed by " all":

```
var str = "Is this all there is";
var patt1 = /is(?! all)/gi;
```

RegExp Object Properties

Property	Description
constructor	Returns the function that created the RegExp object's prototype
global	Checks whether the "g" modifier is set
ignoreCase	Checks whether the "i" modifier is set
lastIndex	Specifies the index at which to start the next match
multiline	Checks whether the "m" modifier is set
source	Returns the text of the RegExp pattern

JavaScript lastIndex Property

‹ JavaScript RegExp Object

Example

Do a global search for "ain" in a string, and output the index after a match is found:

```
var str = "The rain in Spain stays mainly in the plain";
var patt1 = /ain/g;

while (patt1.test(str) == true) {
  document.write("'ain' found. Index now at: "+patt1.lastIndex);
  document.write("<br>");
}
```

JavaScript multiline Property

‹ JavaScript RegExp Object

Example

Check whether or not the "m" modifier is set:

```
var str = "Visit W3Schools!";
var patt1 = /W3S/gi; // "g" and "i" is set, "m" is not.
var res = patt1.multiline;
```

RegExp Object Methods

Method	Description
compile()	Deprecated in version 1.5. Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false
toString()	Returns the string value of the regular expression

Example

Do a global search for "man" in a string, and replace it with "person". Then change the regular expression and replace either "man" or "woman" with "person", with the compile() method:

```
var str = "Every man in the world! Every woman on earth!";
var patt = /man/g;
var str2 = str.replace(patt,"person");
document.write(str2 + "<br>");

patt = /(wo)?man/g;
patt.compile(patt);
str2 = str.replace(patt, "person");
document.write(str2);
```

```
<!DOCTYPE html>
<html>
<body>

<script>

var str="Every man in the world! Every woman on earth!";
var patt=/man/g;
var str2=str.replace(patt,"person");
document.write(str2+"<br>");
patt=/(wo)?man/g;
patt.compile(patt);
str2=str.replace(patt,"person");
document.write(str2);

</script>

</body>
</html>
```

Every person in the world! Every woperson on earth!
Every person in the world! Every person on earth!

JavaScript exec() Method

‹ JavaScript RegExp Object

Example

Search a string for the character "e":

```
var str = "The best things in life are free";
var patt = new RegExp("e");
var res = patt.exec(str);
```

JavaScript exec() Method

‹ JavaScript RegExp Object

Example

Search a string for the character "e":

```
var str = "The best things in life are free";
var patt = new RegExp("e");
var res = patt.exec(str);
```

Try it Yourself »

Definition and Usage

The exec() method tests for a match in a string.

This method returns the matched text if it finds a match, otherwise it returns null.

```
<!DOCTYPE html>
<html>
<body>

<p>The exec() method returns the matched text if it finds a match, otherwise it
returns null.</p>
<p>Click the button to search a string for the character "e".</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "The best things in life are free";
  var patt = new RegExp("e");
  var res = patt.exec(str);
  document.getElementById("demo").innerHTML = res;
}
</script>

</body>
</html>
```

The exec() method returns the matched text if it finds a match, otherwise it returns null.

Click the button to search a string for the character "e".

Try it

e

JavaScript test() Method

‹ JavaScript RegExp Object

Example

Search a string for the character "e":

```
var str = "The best things in life are free";
var patt = new RegExp("e");
var res = patt.test(str);
```

Try it Yourself »

Definition and Usage

The test() method tests for a match in a string.

This method returns true if it finds a match, otherwise it returns false.

```
<!DOCTYPE html>
<html>
<body>

<p>The test() method returns true if it finds a match, otherwise it returns
false.</p>
<p>Click the button to search a string for the character "e".</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "The best things in life are free";
  var patt = new RegExp("e");
  var res = patt.test(str);
  document.getElementById("demo").innerHTML = res;
}
</script>

</body>
</html>
```

The test() method returns true if it finds a match, otherwise it returns false.

Click the button to search a string for the character "e".

Try it

true

JavaScript RegExp toString Method

‹ JavaScript RegExp Object

Example

Return the string value of the regular expression:

```
var patt = new RegExp("Hello World", "g");
var res = patt.toString();
```

Try it Yourself »

Definition and Usage

The toString() method returns the string value of the regular expression.

```html
<!DOCTYPE html>
<html>
<body>

<p>Click the button to return the string value of the regular expression.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var patt = new RegExp("Hello World", "g");
  var res = patt.toString();
  document.getElementById("demo").innerHTML = res;
}
</script>

</body>
</html>
```

Click the button to return the string value of the regular expression.

Try it

/Hello World/g

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

```
var patt = /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:

```
var str = "Visit W3Schools!";
var n = str.search("W3Schools");
```

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:

```
var str = "Visit W3Schools";
var n = str.search(/w3schools/i);
```

The result in *n* will be:

```
6
```

Using String replace() With a String

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

```
var str = "Visit Microsoft!";
var res = str.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:

```
var str = "Visit Microsoft!";
var res = str.replace(/microsoft/i, "W3Schools");
```

Regular Expression Modifiers

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

Modifier	Description	Try it
i	Perform case-insensitive matching	Try it »
g	Perform a global match (find all matches rather than stopping after the first match)	Try it »
m	Perform multiline matching	Try it »

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a multiline search for "is" at the beginning of each
line in a string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "\nIs th\nis it?";
  var patt1 = /^is/m;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a multiline search for "is" at the beginning of each line in a string.

Try it

is

Regular Expression Patterns

Brackets are used to find a range of characters:

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

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x|y)	Find any of the alternatives specified

JavaScript RegExp [abc] Expression

‹ JavaScript RegExp Object

Example

Do a global search for the character "h" in a string:

```
var str = "Is this all there is?";
var patt1 = /[h]/g;
```

Definition and Usage

The [abc] expression is used to find any character between the brackets.

The characters inside the brackets can be any characters or span of characters:

- [abcde..] - Any character between the brackets
- [A-Z] - Any character from uppercase A to uppercase Z
- [a-z] - Any character from lowercase a to lowercase z
- [A-z]- Any character from uppercase A to lowercase z

Syntax

```
new RegExp("[abc]")

or simply:

/[abc]/
```

Syntax with modifiers

```
new RegExp("[abc]", "g")

or simply:

/\[abc]/g
```

course there's a lot in common, but they are a somewhat different in Perl, Ruby, PHP etc.

Regular expressions

A regular expression (also "regexp", or just "reg") consists of a *pattern* and optional *flags*.

There are two syntaxes to create a regular expression object.

The long syntax:

```
regexp = new RegExp("pattern", "flags");
```

...And the short one, using slashes `"/"`:

```
regexp = /pattern/; // no flags
regexp = /pattern/gmi; // with flags g,m and i (to be covered soon)
```

Usage

To search inside a string, we can use method search.

Here's an example:

```
let str = "I love JavaScript!"; // will search here

let regexp = /love/;
alert( str.search(regexp) ); // 2
```

The code above is the same as:

```
let str = "I love JavaScript!"; // will search here

let substr = 'love';
alert( str.search(substr) ); // 2
```

When to use `new RegExp`?

Normally we use the short syntax `/.../`. But it does not allow any variable insertions, so we must know the exact regexp at the time of writing the code.

On the other hand, `new RegExp` allows to construct a pattern dynamically from a string.

So we can figure out what we need to search and create `new RegExp` from it:

```
let search = prompt("What you want to search?", "love");
let regexp = new RegExp(search);

// find whatever the user wants
alert( "I love JavaScript".search(regexp));
```

The “i” flag

The simplest flag is `i`.

An example with it:

```
let str = "I love JavaScript!";

alert( str.search(/LOVE/) ); // -1 (not found)
alert( str.search(/LOVE/i) ); // 2
```

1. The first search returns `-1` (not found), because the search is case-sensitive by default.
2. With the flag `/LOVE/i` the search found `love` at position 2.

So the `i` flag already makes regular expressions more powerful than a simple substring search. But there’s so much more. We’ll cover other flags and features in the next chapters.

Summary

- A regular expression consists of a pattern and optional flags: `g`, `i`, `m`, `u`, `y`.
- Without flags and special symbols that we’ll study later, the search by a regexp is the same as a substring search.
- The method `str.search(regexp)` returns the index where the match is found or `-1` if there’s no match.

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for the characters "i" and "s" in a
string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Do you know if this is all there is?";
  var patt1 = /[is]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Example

Do a global search for the characters "i" and "s" in a string:

```
var str = "Do you know if this is all there is?";
var patt1 = /[is]/gi;
```

Click the button to do a global search for the characters "i" and "s" in a string.

Try it

i,i,s,i,s,i,s

Example

Do a global search for the character-span from lowercase "a" to lowercase "h" in a string:

```
var str = "Is this all there is?";
var patt1 = /[a-h]/g;
```

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to do a global search for the character-span [a-h] in a
string.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var str = "Is this all there is?";
  var patt1 = /[a-h]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>

</body>
</html>
```

Click the button to do a global search for the character-span [a-h] in a string.

Try it

h,a,h,e,e

Example

Do a global search for the character-span from uppercase "A" to uppercase "E":

```
var str = "I SCREAM FOR ICE CREAM!";
var patt1 = /[A-E]/g;
```