RollOvers in Javascript

RollOvers

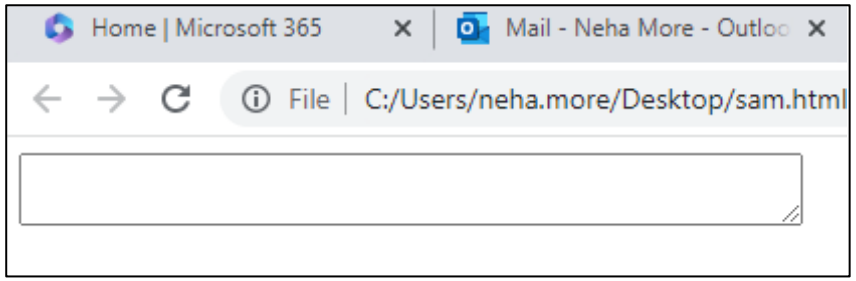
Rollover means a webpage changes when the user moves his or her mouse over an object on the page. It is often used in advertising.There are two ways to create rollover, using plain HTML or using a mixture of JavaScript and HTML.

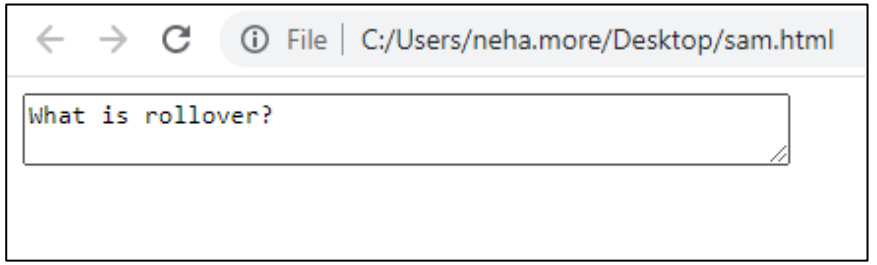
Creating Rollovers using HTML

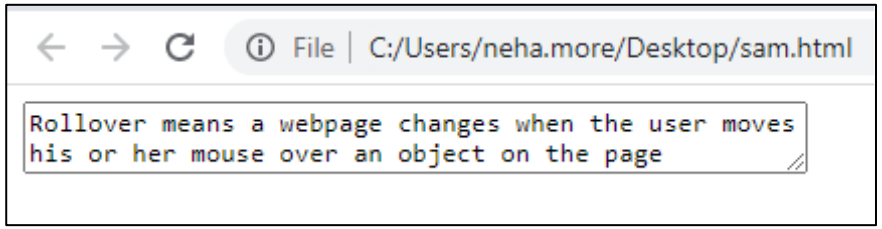
The keyword that is used to create rollover is the <onmousover> event. For example, we want to create a rollover text that appears in a text area. The text *"What is rollover?"* appears when the user place his or her mouse over the text area and the rollover text changes to *"Rollover means a webpage changes when the user moves his or her mouse over an object on the page"* when the user moves his or her mouse away from the text area.

EXAMPLE

```
<HTML>
<head></head>
<Body>
<textarea rows="2" cols="50" name="rollovertext"
onmouseover="this.value='What is rollover?'"
onmouseout="this.value='Rollover means a webpage changes
when the user moves his or her mouse over an object on the
page'"></textarea>
</body>
</html>
```

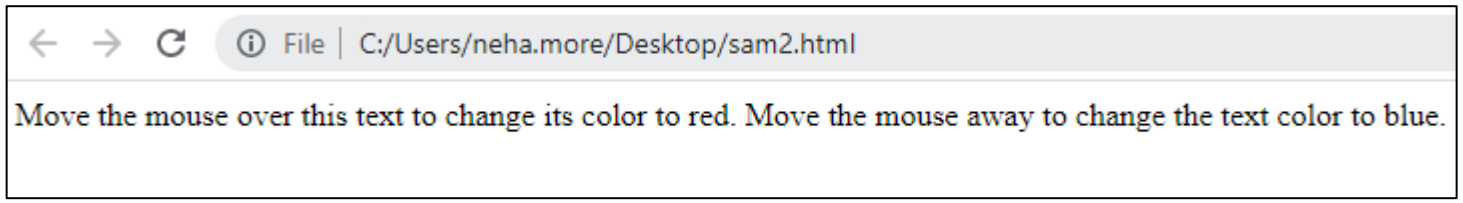


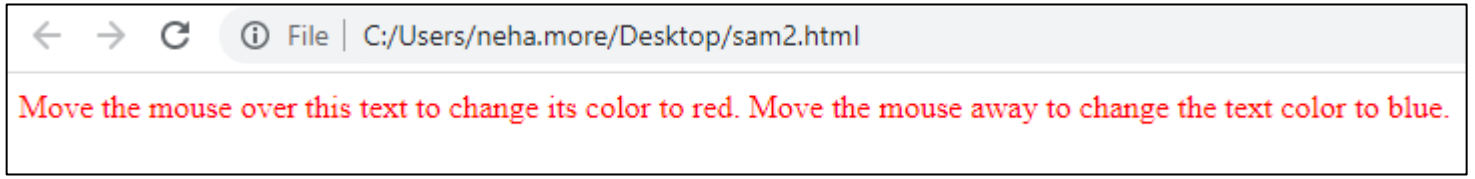


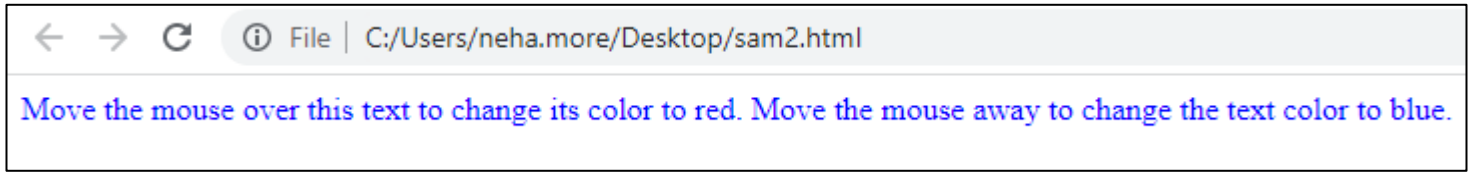


create a rollover effect that can change the color of its text using the style attribute.

```
<html>
<body>
<p
onmouseover="this.style.color='red'"
onmouseout="this.style.color='blue'">
Move the mouse over this text to change its color to red. Move
the mouse away to
change the text color to blue.
</p>
</body>
</html>
```

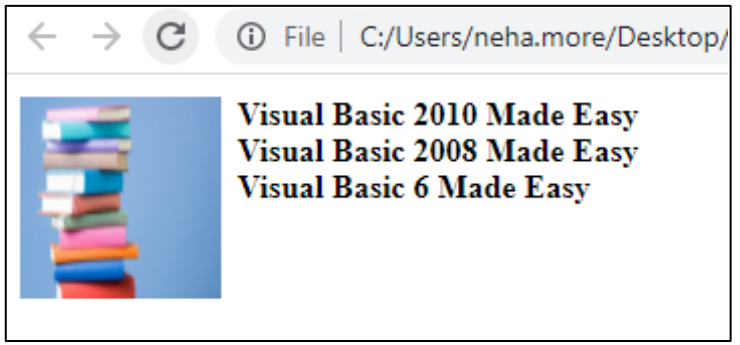


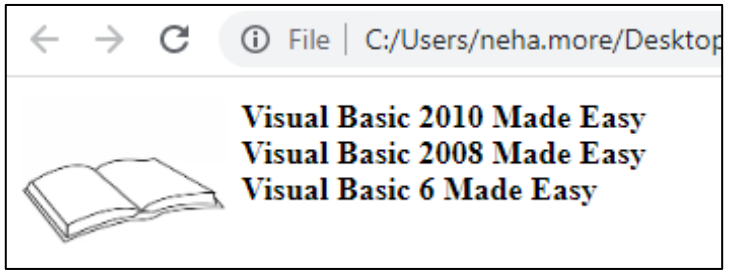


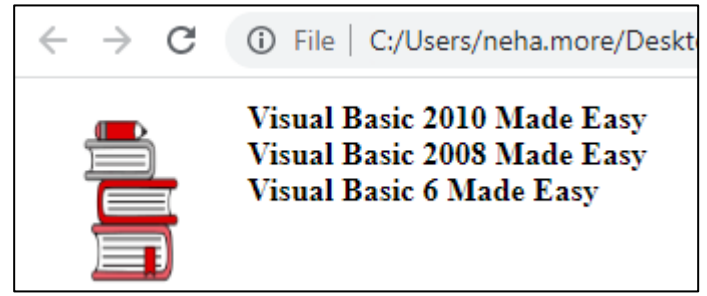


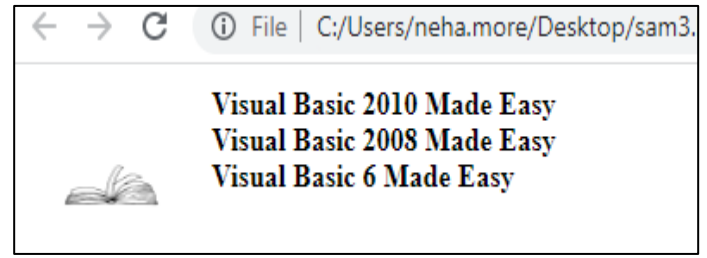
Example 3: To create rollover effect that involves text and images. When the user places his or her mouse pointer over a book title, the corresponding book image appears.

```
<html>
<head>
<title>Rollover Effect</title>
</head>
<body>
<table>
<tbody>
<tr valign="top">
<td width="50">
<a><img src="abc.jpg" name="book" height="100" width="100"></a>
</td><td></td>
<td><a onmouseover="document.book.src='a.png'"><b>Visual Basic 2010 Made
Easy</b></a>
<br>
<a onmouseover="document.book.src='b.png'"><b>Visual Basic 2008 Made Easy</b></a>
<br>
<a onmouseover="document.book.src='c.png'"><b>Visual Basic 6 Made Easy</b></a>
<br>
</td>
</tr>
</tbody>
</table>
</body>
</html>
```









Creating Rollovers Using JavaScript

Though HTML can be used to create rollovers , it can only performs simple actions. If you wish to create more powerful rollovers, you need to use JavaScript. To create rollovers in JavaScript, we need to create a JavaScript function.

Example 4 is similar to Example 3 but we have added some JavaScript code. In this example, we have create an array MyBooks to store the images of three book covers. Next, we create a ShowCover(book) to display the book cover images on the page. Finally, we call the ShowCover function using the onmouseover event.

Example:4

```
<html>
<head>
<script language="Javascript">
MyBooks=new Array('c.png','a.png','b.png')
book=0
function ShowCover(book){document.DisplayBook.src=MyBooks[book]
}</script></head>
<body>
<body>
<P align="center"><img src="abc.jpg" name="DisplayBook" height="50" width="50"><p>
<center>
<table border=0>
<tr>
<td align=center><a onmouseover="ShowCover(0)"><b>Visual Basic 2010 Made
Easy</b></a><br>
<a onmouseover="ShowCover(1)"><b>Visual Basic 2008 Made Easy</b></a><br>
<a onmouseover="ShowCover(2)"><b>Visual Basic 6 Made Easy</b></a><br>
</td>
</tr>
</table>
</body>
</html>
```

Visual Basic 2010 Made Easy
Visual Basic 2008 Made Easy
Visual Basic 6 Made Easy

Visual Basic 2010 Made Easy
Visual Basic 2008 Made Easy
Visual Basic 6 Made Easy

Visual Basic 2010 Made Easy
Visual Basic 2008 Made Easy
Visual Basic 6 Made Easy

Visual Basic 2010 Made Easy
Visual Basic 2008 Made Easy
Visual Basic 6 Made Easy

Regular Expression

RegExp Object

- A regular expression is a **pattern** of characters.
- The pattern is used to do pattern-matching **"search-and-replace"** functions on text.
- In JavaScript, a **RegExp Object** is a pattern with **Properties** and **Methods**.

Syntax

```
/pattern/modifier(s);
```

Example 1

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>Do a case-insensitive search for "w3schools" in a string:</p>
<p id="demo"></p>
<script>
let text = "Visit W3Schools";
let pattern = /w3schools/i;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

W3schools : The pattern to search for
/w3schools/ : A regular expression
/w3schools/i : A case-insensitive regular expression

Modifiers:

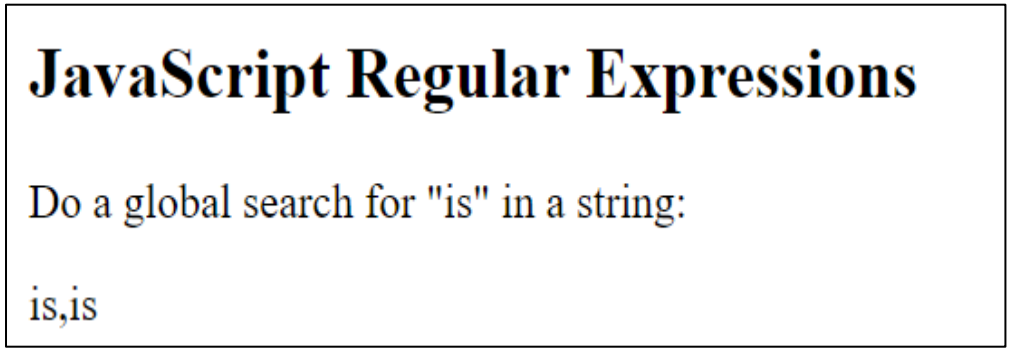
Modifiers are used to perform case-insensitive and global searches:

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

JavaScript RegExp g Modifier:

The "g" modifier specifies a global match.
A global match finds all matches.

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>Do a global search for "is" in a string:</p>

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

<script>
let text = "Is this all there is?";
let pattern = /is/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```



JavaScript RegExp i Modifier:

The "i" modifier specifies a case-insenitive match.

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>The i modifier performs a case-insenitive match:</p>

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

<script>
let text = "Visit W3Schools";
let pattern = /w3schools/i;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions

The i modifier performs a case-insenitive match:
W3Schools

JavaScript RegExp m Modifier

- The "m" modifier specifies a **multiline match**.
- It only affects the behavior of start **^** and end **$**.
- **^** specifies a match at the start of a string.
- **$** specifies a match at the end of a string.
- With the "m" set, ^ and $ also match at the beginning and end of each line.

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>Do a multiline search for "is" at the beginning of each line
in a string:</p>
<p id="demo"></p>
<script>
let text = `Is this
all there
is`
let pattern = /^is/m;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions

Do a multiline search for "is" at the beginning of each line in a string:

is

The "m" modifier is case-sensitive and not global.
To perform a global, case-insensitive search, use "m" with "g" and "i".

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>

<p>Do a global, multiline search for "is" at the beginning of
each line in a string.</p>

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

<script>
let text = `Is this
all there
is`

let pattern = /^is/gm;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions
Do a global, multiline search for "is" at the beginning of each line in a string.
is

A global, case-insensitive, multiline search for "is" at the beginning of each string line:

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>Do a global, case-insensitive, multiline search for "is" at the beginning of each line in a string.</p>

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

<script>
let text = `Is this
all there
is`

let pattern = /^is/gmi;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions

Do a global, case-insensitive, multiline search for "is" at the beginning of each line in a string.

Is,is

A global, multiline search for "is" at the end of each string line

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>

<p>Do a global, multiline search for "is" at the end of each line
in a string.</p>

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

<script>
let text = `Is this
all there
is`

let pattern = /is$/gm;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions
Do a global, multiline search for "is" at the end of each line in a string.

is,is

Regular Expression Search Methods:

In JavaScript, a regular expression text search, can be done with different methods.
With a **pattern** as a regular expression, these are the most common methods:

Example	Description
text.match(pattern)	The String method match()
text.search(pattern)	The String method search()
pattern.exec(text)	The RexExp method exec()
pattern.test(text)	The RegExp method test()

JavaScript String match()

- The match() method matches a string against a regular expression **
- The match() method returns an array with the matches.
- The match() method returns null if no match is found.

Syntax:

```
string.match(match)
```

Parameters:

Parameter	Description
match	Required. The search value. A regular expression (or a string that will be converted to a regular expression).

Return Values

Type	Description
An array or null	An array containing the matches. null if no match is found.

A search for "ain" using a string:

```
<html>
<body>

<h1>JavaScript Strings</h1>
<h2>The match() Method</h2>

<p>match() searches for a match in a string.</p>

<p>Do a search for "ain":</p>

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

<script>
let text = "The rain in SPAIN stays mainly in the plain";
let result = text.match("ain");

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Strings
The match() Method
match() searches for a match in a string.

Do a search for "ain":

ain

A search for "ain" using a regular expression:

```
<html>
<body>

<h1>JavaScript Strings</h1>
<h2>The match() Method</h2>

<p>match() searches for a match against a regular
expression.</p>

<p>Do a search for "ain":</p>

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

<script>
let text = "The rain in SPAIN stays mainly in the plain";
let result = text.match(/ain/);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Strings
The match() Method
match() searches for a match against a regular expression.

Do a search for "ain":

ain

JavaScript String search():

- The search() method matches a string against a regular expression **
- The search() method returns the index (position) of the first match.
- The search() method returns -1 if no match is found.
- The search() method is case sensitive.

Syntax

```
string.search(searchValue)
```

Parameters

Parameter	Description
searchValue	Required. The search value. A regular expression (or a string that will be converted to a regular expression).

Return Value

Type	Description
A number	The position of the first match. -1 if no match.

Example:

Search for "Blue":

```
<html>
<body>

<h1>JavaScript Strings</h1>
<h2>The search() Method</h2>

<p>search() searches a string for a value and returns the
position of the first match:</p>

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

<script>
let text = "Mr. Blue has a blue house"
let position = text.search("Blue");

document.getElementById("demo").innerHTML = position;
</script>

</body>
</html>
```

JavaScript Strings
The search() Method
search() searches a string for a value and returns the
position of the first match:

4

Search for "blue":

```
<html>
<body>

<h1>JavaScript Strings</h1>
<h2>The search() Method</h2>

<p>search() searches a string for a value and returns the
position of the first match:</p>

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

<script>
let text = "Mr. Blue has a blue house"
let position = text.search("blue");

document.getElementById("demo").innerHTML = position;
</script>

</body>
</html>
```

JavaScript Strings
The search() Method
search() searches a string for a value and returns the position of the first match:

15

Search for /Blue/:

```
<html>
<body>

<h1>JavaScript Strings</h1>
<h2>The search() Method</h2>

<p>search() searches a string for a value and returns the
position of the first match:</p>

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

<script>
let text = "Mr. Blue has a blue house"
let position = text.search(/Blue/);

document.getElementById("demo").innerHTML = position;
</script>

</body>
</html>
```

JavaScript Strings
The search() Method
search() searches a string for a value and returns the position of the first match:

4

Search for /blue/:

```
<html>
<body>

<h1>JavaScript Strings</h1>
<h2>The search() Method</h2>

<p>search() searches a string for a value and returns the
position of the first match:</p>

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

<script>
let text = "Mr. Blue has a blue house"
let position = text.search(/blue/);

document.getElementById("demo").innerHTML = position;
</script>

</body>
</html>
```

JavaScript Strings
The search() Method
search() searches a string for a value and returns the position of the first match:

15

Search case insensitive:

```
<html>
<body>

<h1>JavaScript Strings</h1>
<h2>The search() Method</h2>

<p>search() searches a string for a value and returns the
position of first the match:</p>

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

<script>
let text = "Mr. Blue has a blue house"
let position = text.search(/blue/i);

document.getElementById("demo").innerHTML = position;
</script>

</body>
</html>
```

JavaScript Strings
The search() Method
search() searches a string for a value and returns the position of first the match:

4

The Difference Between

String search() and String match()

The search() method returns the position of the first match.

The match() method returns an array of matches.

JavaScript RegExp exec()

- The exec() method tests for a match in a string.
- If it finds a match, it returns a result array, otherwise it returns null.

Syntax

```
RegExpObject.exec(string)
```

Parameter Values

Parameter	Description
string	Required. The string to be searched

Return Value

Type	Description
Array	An array containing the matched text if it finds a match, otherwise it returns null

Example: Search a string for the character "e":

```
<html>
<body>

<h2>JavaScript RegExp</h2>

<p>The exec() method tests for a match in a string:</p>

<p>Search a string for the character "e":</p>

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

<script>
let text = "The best things in life are free";
let result = /e/.exec(text);
document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript RegExp

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

Search a string for the character "e":

e

Example:
Do a global search for "Hello" and "W3Schools" in a string:

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>The exec() method tests for a match in a string:</p>

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

<script>
let text = "Hello world!";

// look for "Hello"
let result1 = /Hello/.exec(text);

// look for "W3Schools"
let result2 = /W3Schools/.exec(text);

document.getElementById("demo").innerHTML =
result1 + "<br>" + result2;
</script>
</body>
</html>
```

JavaScript Regular Expressions
The exec() method tests for a match in a string:

Hello
null

JavaScript RegExp test():

- The test() method tests for a match in a string.
- If it finds a match, it returns true, otherwise it returns false.

Syntax

```
RegExpObject.test(string)
```

Parameter Values

Parameter	Description
string	Required. The string to be searched

Return Value

Type	Description
Boolean	Returns true if it finds a match, otherwise false

Example 1

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>The test() method returns true if it finds a match, otherwise
false.</p>

<p>Search a string for the character "e":</p>
<p id="demo"></p>

<script>
let text = "The best things in life are free";
let pattern = /e/;
let result = pattern.test(text);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

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

Search a string for the character "e":

true

Example 2

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>Do a global search for "Hello" and "W3Schools" in a string:</p>
<p id="demo"></p>

<script>
let text = "Hello world!";

// look for "Hello"
let pattern1 = /Hello/g;
let result1 = pattern1.test(text)

// look for "W3Schools"
let pattern2 = /W3Schools/g;
let result2 = pattern2.test(text);

document.getElementById("demo").innerHTML = result1 + "<br>" + result2;
</script>
</body>
</html>
```

JavaScript Regular Expressions

Do a global search for "Hello" and "W3Schools" in a string:

true
false

JavaScript RegExp Group [abc]

- Brackets [abc] specifies matches for the characters inside the brackets.
- Brackets can define single characters, groups, or character spans:

[abc]	Any of the characters a, b, or c
[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

Example:

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>A global search for the character "h" in a string:</p>
<p id="demo"></p>
<script>
let text = "Is this all there is?";
let pattern = /[h]/g;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions
A global search for the character "h" in a string:

h,h

Example

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

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>A global search for the characters "i" and "s" in a string:</p>

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

<script>
let text = "Do you know if this is all there is?";
let pattern = /[is]/gi;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions
A global search for the characters "i" and "s" in a string:

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

Example

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

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>A global search for the character span [a-h] in a string:</p>

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

<script>
let text = "Is this all there is?";
let pattern = /[a-h]/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions
A global search for the character span [a-h] in a string:

h,a,h,e,e

Example

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

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>
<p>A global search for any character from uppercase A to
uppercase E.</p>
<p id="demo"></p>
<script>
let text = "I SCREAM FOR ICE CREAM!";
let pattern = /[A-E]/g;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions
A global search for any character from uppercase A to uppercase E.

C,E,A,C,E,C,E,A

Example

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**.)

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>A global search for any character from uppercase "A" to lowercase "e":</p>

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

<script>
let text = "I Scream For Ice Cream, is that OK?!";
let pattern = /[A-e]/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML=result;
</script>

</body>
</html>
```

JavaScript Regular Expressions
A global search for any character from uppercase "A" to lowercase "e":

I,S,c,e,a,F,I,c,e,C,e,a,a,O,K

Example

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

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>A global, case-insensitive search for the character span [a-s]:</p>

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

<script>
let text = "I Scream For Ice Cream, is that OK?!";
let pattern = /[a-s]/gi;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions
A global, case-insensitive search for the character span [a-s]:

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

A "g" and "gi" search for characters:

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>
<p>Perform a search for the characters "THIS" in a string.</p>
<p id="demo"></p>

<script>
let text = "THIS This this";

let result1 = text.match(/[THIS]/g);
let result2 = text.match(/[THIS]/gi);

document.getElementById("demo").innerHTML =
result1 + "<br>" +result2;
</script>
</body>
</html>
```

JavaScript Regular Expressions
Perform a search for the characters "THIS" in a string.

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

JavaScript RegExp Group [^abc]

Brackets [^abc] specifies matches for any character NOT between the brackets.

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>Do a global search for that characters that are not "h":</p>

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

<script>
let text = "Is this all there is?";
let pattern = /[^h]/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions
Do a global search for that characters that are not "h":

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

Example

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

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>A global search for characters that are NOT "i" or "s":</p>

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

<script>
let text = "Do you know if this is all there is?";
let pattern = /[^is]/gi;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions
A global search for characters that are NOT "i" or "s":

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

Example:

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

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>

<p>A global search for characters NOT in the span from
lowercase "a" to lowercase "h":</p>

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

<script>
let text = "Is this all there is?";
let pattern = /[^a-h]/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions
A global search for characters NOT in the span from
lowercase "a" to lowercase "h":

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

Example

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

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>A global search for characters NOT in the span, from
uppercase A to uppercase E:</p>

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

<script>
let text = "I SCREAM FOR ICE CREAM!";
let pattern = /[^A-E]/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions
A global search for characters NOT in the span, from uppercase A to uppercase E:

I, ,S,R,M, ,F,O,R, ,I, ,R,M,!

Example

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

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>A global search for characters NOT in the span, from
uppercase "A" to lowercase "e":</p>

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

<script>
let text = "I Scream For Ice Cream, is that OK?!";
let pattern = /[^A-e]/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions
A global search for characters NOT in the span, from uppercase "A" to lowercase "e":

,r,m, ,o,r, , ,r,m,,, ,i,s, ,t,h,t, ,?,!

Example:

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

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>A global, case-insensitive search for the characters NOT in
the span [a-s]:</p>

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

<script>
let text = "I Scream For Ice Cream, is that OK?!";
let pattern = /[^a-s]/gi;
let result = text.match(pattern);

document.getElementById("demo").innerHTML= result;
</script>

</body>
</html>
```

JavaScript Regular Expressions

A global, case-insensitive search for the characters NOT in the span [a-s]:

, , , ,,, , ,t,t, ,?,!

JavaScript RegExp Group [0-9]

- The [0-9] expression is used to find any character between the brackets.
- The digits inside the brackets can be any numbers or span of numbers from 0 to 9.

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>

<p>A global search for the numbers 1 to 4:</p>

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

<script>
let text = "123456789";
let pattern = /[1-4]/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions
A global search for the numbers 1 to 4:

1,2,3,4

JavaScript RegExp Group [^0-9]

A global search for numbers that are NOT from 1 to 4:

```
<html>
<body>

<h2>JavaScript Regular Expressions</h2>

<p>A global search for numbers that are NOT from 1 to 4:</p>

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

<script>
let text = "123456789";
let pattern = /[^1-4]/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions

A global search for numbers that are NOT from 1 to 4:

5,6,7,8,9

JavaScript RegExp Group (x|y)

- The (x|y) expression is used to find any of the alternatives specified.
- The alternatives can be of any characters.

A global search for any of the alternatives (red|green):

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>A global search for the specified alternatives
(red|green):</p>
<p id="demo"></p>
<script>

let text = "re, green, red, green, gren, gr, blue, yellow";
let pattern = /(red|green)/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript Regular Expressions

A global search for the specified alternatives (red|green):

green,red,green

Example

Global search to find any of the specified alternatives (0|5|7):

```
<html>
<body>
<h2>JavaScript Regular Expressions</h2>
<p>A global search for any of the specified alternatives
(0|5|7):</p>
<p id="demo"></p>

<script>
let text = "01234567890123456789";
let pattern = /(0|5|7)/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
</body>
</html>
```

JavaScript Regular Expressions
A global search for any of the specified alternatives
(0|5|7):

0,5,7,0,5,7

Metacharacter

JavaScript RegExp . Metacharacter

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for "h.t":</p>

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

<script>
let text = "That's hot!";
let pattern = /h.t/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A global search for "h.t":
hat,hot

```
let text = "Thaat's hot!";
let pattern = /h..t/g;
let result = text.match(pattern);
```

A global search for "h.t":
haat

JavaScript RegExp \w Metacharacter

- The \w metacharacter matches word characters.
- A word character is a character a-z, A-Z, 0-9, including _ (underscore).

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for word characters:</p>

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

<script>
let text = "Give 100%!";
let pattern = /\w/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A global search for word characters:

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

JavaScript RegExp \W Metacharacter

- The \W metacharacter matches non-word characters:
- A word character is a character a-z, A-Z, 0-9, including _ (underscore).

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for non-word characters:</p>

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

<script>
let text = "Give 100%!";
let pattern = /\W/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A global search for non-word characters:

,%,!

JavaScript RegExp \d Metacharacter

The \d metacharacter matches digits from 0 to 9.

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for digits:</p>

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

<script>
let text = "Give 100%!";
let pattern = /\d/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A global search for digits:

1,0,0

JavaScript RegExp \D Metacharacter

```
let text = "Give 100%!";
let pattern = /\D/g;
```

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

JavaScript RegExp \s Metacharacter

```
<h2>JavaScript Regular Expressions</h2>
<p>A global search for whitespace characters:</p>
<p id="demo"></p>

<script>
let text = "Is this all there is?";
let pattern = /\s/g;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

A global search for whitespace characters:

, , ,

JavaScript RegExp \S Metacharacter

```
let text = "Is this all there is?";
let pattern = /\S/g;
```

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

JavaScript RegExp \b Metacharacter

The \b metacharacter matches at the beginning or end of a word.

Search for "LO" at the beginning of a word:

```
<h2>JavaScript Regular Expressions</h2>
<p>Search for the characters "LO" in the <b>beginning</b> of a word:</p>
<p>"HELLO, LOOK AT YOU!"</p>
<p id="demo"></p>
<script>
let text = "HELLO, LOOK AT YOU!";
let pattern = /\bLO/;
let result = text.search(pattern);
document.getElementById("demo").innerHTML = "Found in position: " + result;
</script>
```

JavaScript Regular Expressions
Search for the characters "LO" in the beginning of a word:

"HELLO, LOOK AT YOU!"

Found in position: 7

- Search for the pattern LO at the beginning of a word like this:

\bLO

- Search for the pattern LO at the end of a word like this:

LO\b

JavaScript RegExp \B Metacharacter

```
<h2>JavaScript Regular Expressions</h2>

<p>Search for the characters "LO" the phrase: "HELLO, LOOK AT YOU!" and return the first position where it is present, NOT in the <b>beginning</b> of a word:</p>

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

<script>
let text = "HELLO, LOOK AT YOU!";
let pattern = /\BLO/;
let result = text.search(pattern);

document.getElementById("demo").innerHTML = "Found in position: " + result;
</script>
```

JavaScript Regular Expressions

Search for the characters "LO" the phrase: "HELLO, LOOK AT YOU!" and return the first position where it is present, NOT in the **beginning** of a word:

Found in position: 3

The **\B** metacharacter matches NOT at the beginning/end of a word.

- Search for the pattern LO, not at the beginning of a word like this:

\BLO

- Search for the pattern LO, not at the end of a word like this:

LO\B

JavaScript RegExp \0 Metacharacter

The \0 metacharacter maches NUL characters.

```
<h2>JavaScript Regular Expressions</h2>

<p>Find the position where a NUL character is found:</p>

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

<script>
let text = "Visit W3Schools.\0Learn JavaScript.";
let pattern = /\0/;
let result = text.search(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

Find the position where a NUL character is found:

16

JavaScript RegExp \n Metacharacter

The \n character matches newline characters.

```
<h2>JavaScript Regular Expressions</h2>

<p>find the position where a newline character is found:.</p>

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

<script>
let text = "Visit W3Schools.\nLearn JavaScript.";
let pattern = /\n/;
let result = text.search(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
find the position where a newline character is found:.
16

Quantifier

JavaScript RegExp + Quantifier

The n+ quantifier matches any string that contains at least one n.

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for at least one "o" in a string:</p>

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

<script>
let text = "Hellooo World! Hello W3Schools!";
let pattern = /o+/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

A global search for at least one "o" in a string:

ooo,o,o,oo

JavaScript RegExp * Quantifier

The **n*** quantifier matches any string that contains zero or more occurrences of n.

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for an "l", followed by zero or more "o"
characters:</p>

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

<script>
let text = "Hellooo World! Hello W3Schools!";
let pattern = /lo*/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A global search for an "l", followed by zero or more "o" characters:

l,looo,l,l,lo,l

```
<script>
let text = "1, 100 or 1000?";
let pattern = /10*/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

1,100,1000

JavaScript RegExp ? Quantifier

The n? quantifier matches any string that contains zero or one occurrences of n.

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for a "1", followed by zero or one "0"
characters:</p>

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

<script>
let text = "1, 100 or 1000?";
let pattern = /10?/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

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

1,10,10

JavaScript RegExp {X} Quantifier

- The n{X} quantifier matches any string that contains a sequence of X n's.
- X must be a number.

```
<h2>JavaScript Regular Expressions</h2>
<p>A global search for sequence of four digits:</p>
<p id="demo"></p>

<script>
let text = "100, 1000 or 10000?";
let pattern = /\d{4}/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A global search for sequence of four digits:

1000,1000

JavaScript RegExp {X,Y} Quantifier

- The n{X,Y} quantifier matches any string that contains a sequence of X to Y n's.
- X and Y must be a number.

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for a sequence of three to four digits:</p>

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

<script>
let text = "100, 1000 or 10000? 10";
let pattern = /\d{3,4}/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

A global search for a sequence of three to four digits:

100,1000,1000

JavaScript RegExp {X,} Quantifier

- The n{X,} quantifier matches any string that contains a sequence of at least X n's.
- X must be a number.

```
<h2>JavaScript Regular Expressions</h2>

<p>A global search for a sequence of at least three digits:</p>

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

<script>
let text = "100, 1000 or 10000? 10000000000 10";
let pattern = /\d{3,}/g;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

A global search for a sequence of at least three digits:

100,1000,10000,10000000000

JavaScript RegExp $ Quantifier

The *n*$ quantifier matches any string with *n* at the end of it.

```
<h2>JavaScript Regular Expressions</h2>
<p>A search for "is" at the end of a string:</p>
<p id="demo"></p>

<script>
let text = "Is this his";
let pattern = /is$/;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A search for "is" at the end of a string:

is

The **^n** quantifier matches any string with n at the beginning of it.

```
<h2>JavaScript Regular Expressions</h2>
<p>A search for "Is" at the beginning of a string:</p>
<p id="demo"></p>
<script>
let text = "Is this his";
let pattern = /^Is/g;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A search for "Is" at the beginning of a string:
Is

JavaScript RegExp ?= Quantifier

The ?=n quantifier matches any string that is followed by a specific string n.

A search for "**is**" followed by " **all**":

```
<h2>JavaScript Regular Expressions</h2>
<p>A search for "is" followed by " all".</p>
<p id="demo"></p>
<script>
let text = "Is this all there isall is all ";
let pattern = /is(?= all)/g;
let result = text.match(pattern);
document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A search for "is" followed by " all".
is,is

JavaScript RegExp ?! Quantifier

```
<p>A global, case insensitive search for "is" not followed by " all":</p>

let text = "Is this all there is";
let pattern = /is(?! all)/gi;
```

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

Is,is

JavaScript RegExp ?! Quantifier

The ?!*n* quantifier matches any string that is not followed by a specific string *n*.

```
<h2>JavaScript Regular Expressions</h2>

<p>A global, case insensitive search for "is" not followed by "
all":</p>

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

<script>
let text = "Is this all there is";
let pattern = /is(?! all)/gi;
let result = text.match(pattern);

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
A global, case insensitive search for "is" not followed by " all":
Is,is

JavaScript RegExp constructor Property

- The constructor property returns the function that created the RegExp prototype.
- For a regular expression the constructor property returns:
- function RegExp() { [native code] }

```
<h2>JavaScript Regular Expressions</h2>

<p>The constructor property returns the function that created the RegExp prototype:</p>

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

<script>
let pattern = /Hello World/g;
let text = pattern.constructor;

document.getElementById("demo").innerHTML = text;
</script>
```

JavaScript Regular Expressions

The constructor property returns the function that created the RegExp prototype:

function RegExp() { [native code] }

Note: Native code is binary data compiled to run on a processor, such as an Intel x86-class processor. The code is written in all 1s and 0s that must conform to the processor's instruction set architecture (ISA). Native code provides instructions to the processor that describe what tasks to carry out.

JavaScript RegExp global

```
<h2>JavaScript Regular Expressions</h2>

<p>The global property returns true if the "g" modifier is set,
otherwise false:</p>

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

<script>
let pattern = /W3S/g;
let result = pattern.global;

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

The global property returns true if the "g" modifier is set, otherwise false:

true

JavaScript RegExp ignoreCase

The ignoreCase property specifies whether or not the "i" modifier is set.

```
<h2>JavaScript Regular Expressions</h2>

<p>The ignoreCase property returns true if the "i" modifier is set, otherwise false:</p>

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

<script>
let text = "Visit W3Schools!";
let pattern = /W3S/i;
let result = pattern.ignoreCase;
document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions
The ignoreCase property returns true if the "i" modifier is set, otherwise false:

true

JavaScript lastIndex Property:

- The lastIndex property specifies the index at which to start the next match.

Note: This property only works if the "g" modifier is set.

- This property returns an integer that specifies the character position immediately after the last match found by exec( ) or test( ) methods.

Note: exec( ) and test( ) reset lastIndex to 0 if they do not get a match.

```
<h2>JavaScript Regular Expressions</h2>
<p>The lastIndex property specifies the index at which to start the next match:</p>

<p id="demo"></p>
<script>
let text = "The rain in Spain stays mainly in the plain";
let pattern = /ain/g;
let result = "";
while (pattern.test(text)==true)  {
  result += "Found at position " + pattern.lastIndex + "<br>";
}
document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

The lastIndex property specifies the index at which to start the next match:

Found at position 8
Found at position 17
Found at position 28
Found at position 43

JavaScript multiline Property

- The multiline property specifies whether or not the m modifier is set.
- This property returns true if the "m" modifier is set, otherwise it returns false.

```
<h2>JavaScript Regular Expressions</h2>

<p>The multiline property returns true if the "m" modifier is
set, otherwise false:</p>

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

<script>
let text = "Visit W3Schools!";
let pattern = /W3S/gi;
let result = pattern.multiline;

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

The multiline property returns true if the "m" modifier is set, otherwise false:

false

JavaScript source Property

The source property returns the text of the RegExp pattern.

```
<h2>JavaScript Regular Expressions</h2>

<p>The source property returns the text of a RegExp
pattern:</p>

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

<script>
let text = "Visit W3Schools";
let pattern = /W3S/g;
let result = pattern.source;

document.getElementById("demo").innerHTML = result;
</script>
```

JavaScript Regular Expressions

The source property returns the text of a RegExp pattern:

W3S

Regular expression to check whether first letter is capital or not

```
<html>
<head>
   <title>Check First Letter Capital</title>
   <script>
      function checkFirstLetterCapital() {
         // Get the input value from the textbox
         var inputText = document.getElementById("textInput").value;

         // Regular expression to match the first letter being capitalized
         var regex = /^[A-Z]/;

         // Check if the first letter is capitalized or not
         if (inputText.match(regex)) {
            alert("First letter is capitalized!");
         } else {
            alert("First letter is not capitalized!");
         }
      }
   </script>
</head>
<body>
   <input type="text" id="textInput" placeholder="Enter a string">
   <button onclick="checkFirstLetterCapital()">Check</button>
</body>
</html>
```

Regular expression to enter only digits between 0-5

```
<html>
<head>
   <title>Accept Numbers 0 to 5</title>
</head>
<body>
   <input type="text" id="numberInput" placeholder="Enter a number between 0 and 5">
   <button onclick="validateInput()">Check</button>
   <script>
      function validateInput() {
         var inputValue = document.getElementById("numberInput").value;

         if (inputValue.match(/^[0-5]$/)) {
            // Input is valid
            alert("Input is valid: " + inputValue);
         } else {
            // Input is invalid
            document.getElementById("numberInput").value = ""; // Clear the input value
            alert("Please enter a number between 0 and 5.");
         }
      }
   </script>
</body>
</html>
```

Form Validation using Regular Expression---email validation

```
<html>

<head>
   <title>creating mailing system</title>
   <style>
      legend {
         display: block;
         padding-left: 2px;
         padding-right: 2px;
         border: none;
      }
   </style>

   <script type="text/javascript">
```

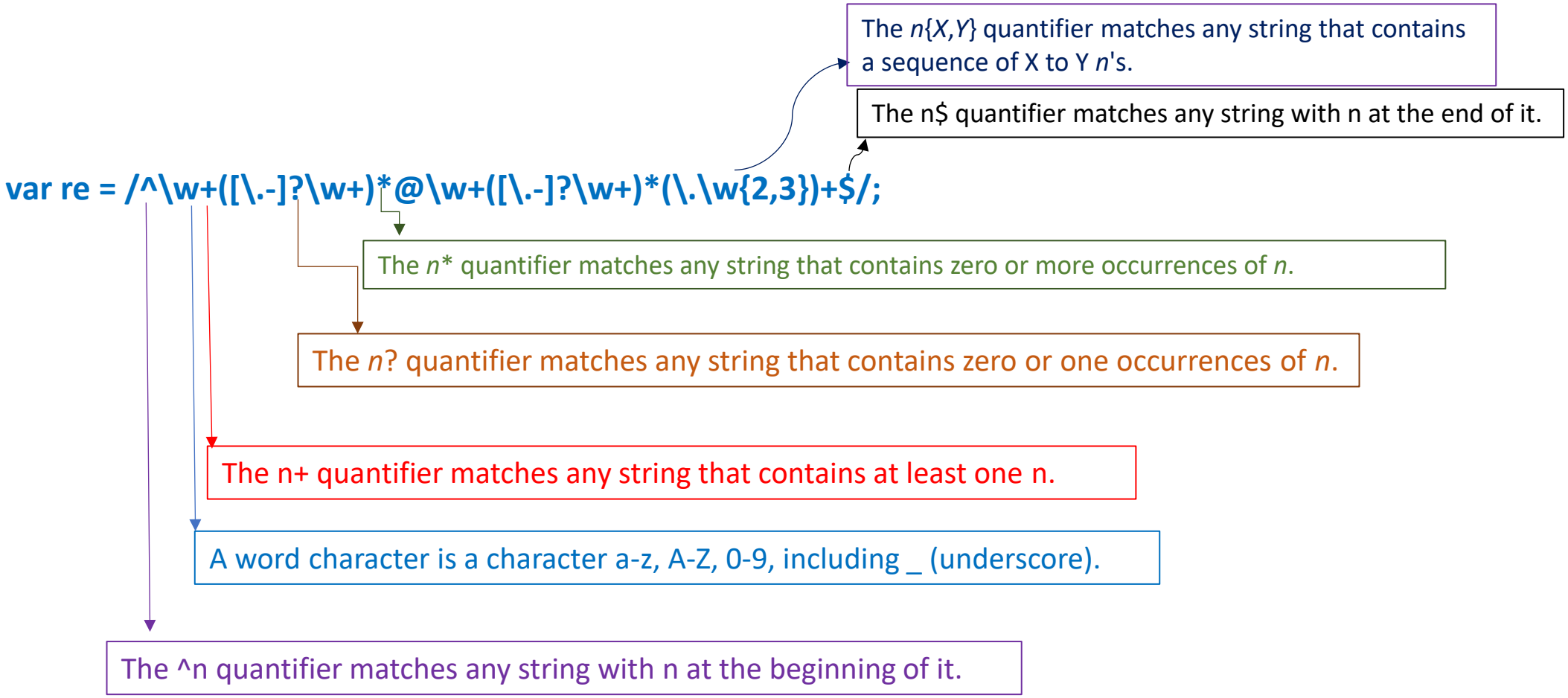
```
function validate() {

        var user = document.getElementById("e").value;
        var user2 = document.getElementById("e");

        var re = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
                        ^[a-zA-Z0-9+_.-]+@[a-zA-Z0-9.-]+$

        if (re.test(user))
            {
            alert("done");
            return true;
            }

        else
            {
           user2.style.border = "red solid 3px";
           return false;
        }
      }
   </script>
</head>
```

- The **^n** quantifier matches any string with n at the beginning of it.
- The **\w** metacharacter matches word characters.
- A word character is a character a-z, A-Z, 0-9, including _ (underscore).
- The ***n*+** quantifier matches any string that contains at least one *n*.
- The ***n*?** quantifier matches any string that contains zero or one occurrences of *n*.
- The **n*** quantifier matches any string that contains zero or more occurrences of n.
- The **n$** quantifier matches any string with n at the end of it.

```
<body bgcolor="cyan">
   <center>
      <h1>Email Registration</h1>
      <form>
         <fieldset style="width:300px">
            <legend>Registation Form</legend>
            <table>
               <tr>
                  <input type="text" placeholder="firstname" maxlength="10">
               </tr>
               <br><br>
               <tr>
                  <input type="text" placeholder="lastname" maxlength="10">
               </tr>
               <br><br>
               <tr>
                  <input type="email"
                     placeholder="username@gmail.com" id="e">
               </tr>
               <br><br>
               <tr>
                  <input type="password" placeholder="password">
               </tr>
```

```
<br><br>
            <tr>
              <input type="password" placeholder="confirm">
            </tr>
            <br><br>
            <tr>
              <input type="text" placeholder="contact">
            </tr>
            <br><br>
            <tr>
              <label>Gender:</label>
              <select id="gender">
                <option value="male">male</option>
                <option value="female">female</option>
                <option value="others">others</option>
              </select>
            </tr>
            <br><br>
            <tr><input type="submit" onclick="validate()" value="create">
            </tr>
          </table>
        </fieldset>
      </form>
   </center>
</body>
 </html>
```

Regular for Email verification:



var re = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;

^: Asserts the start of the string.

\w+: Matches one or more word characters (letters, digits, or underscores). This represents the username part of the email address.

([\.-]?\w+)*: This part represents the domain name.

[\.-]?: Matches an optional **dot (.) or hyphen (-).**
\w+: Matches one or more word characters.
*: Allows for zero or more occurrences of the preceding group, allowing for multiple domain segments.

@: Matches the @ symbol, which separates the username from the domain name.

\w+: Matches one or more word characters. This represents the domain name.

([\.-]?\w+)*: Similar to the username part, this allows for multiple domain segments separated by dots or hyphens.

(\.\w{2,3})+: This part matches the top-level domain (TLD), which consists of a dot followed by 2 or 3 word characters (e.g., .com, .org, .net).

\.\w{2,3}: Matches a dot followed by 2 or 3 word characters.

+: Allows for one or more occurrences of the preceding group, enabling support for multiple TLDs like .co.uk or .com.au.
$: Asserts the end of the string.

Example1:

txt = "hello world"

#Search for a sequence that starts with "he", followed by two (any) characters, and an "o":

```
let pattern = /he..o/g;
let result = text.match(pattern);
```

Examples:

JavaScript Regex that you will use for phone number validation:

```
/^\(?(\d{3})\)?[- ]?(\d{3})[- ]?(\d{4})$/
```

I. /^\(?: The number may start with an open parenthesis.

II. (\d{3}): Then three numeric digits must be entered for valid formats. If there is no parenthesis, the number must start with these digits.

III. \)?: It allows you to include a close parenthesis.

IV. [-]?: The string may optionally contain a hyphen. It can be placed either after the parenthesis or following the first three digits. For example, (123)- or 123-.

V. (\d{3}): Then the number must contain another three digits. For example, it can look like this: (123)-456, 123-456, or 123456.

VI. [-]?: It allows you to include an optional hyphen in the end, like this: (123)-456-, -123- or 123456-.

VII. (\d{4})$/: Finally, the sequence must end with four digits. For example, (123)-456-7890, 123-456-7890, or 123456-7890.

(123) 456-7890

(123)456-7890

123-456-7890

1234567890