
cybero designs
Quartz Composer Guide :
Quartz Composer JavaScript Guide :
Archive :
cybero productions
Services :
E: contact
Friday, June 26, 2009

Home

- Cybero Site
- Resources & Links
- JavaScript Patch
- WebKit Plugin
- QC JavaScript Guide
- Quartz Composer Guide
- Cybero Archive

Instructions

- Comments / Notes
- break
- continue
- delete
- do ~ while
- for
- for ... in
- function
- ifelse
- new
- return
- switch
- this
- throw
- try
- var
- while
- with

Boolean

- Boolean

Function

- arguments
- Function

Number

- Number
- MAX_VALUE
- MIN_VALUE
- NaN
- NEGATIVE_INFINITY
- POSITIVE_INFINITY
- toExponential
- toFixed

Quartz Composer JavaScript Reference

These pages provide a reference for the JavaScript facilities and API built into Quartz Composer , Apple's Xcode Tool, (included in the default installation of Applications within the OS X Developer Tools folder, as of OS X Tiger onwards). In QC2 users were somewhat constrained as to how inputs and outputs were created by configuration of the JavaScript patch. The JavaScript performance was by no means the snappiest in QC2.

With the release of Leopard , came Quartz Composer 3 and a few changes to how JavaScript is formatted in the JavaScript patch. QC3 users were presented with a scripting patch that gave them far greater liberty to fully code in standard JavaScript. The primary difference between QC3 JavaScript and browser JavaScript rests in the formatting of the function. Otherwise, it is pretty easy to port a lot of your existing JS code, sans DOM, to QC3.

QC3 JavaScript, is quite a remarkable assemblage of Core JavaScript objects, properties and methods. It supports both current, deprecated and cutting edge JavaScript properties [properties, indeed, still being rolled out in the two most frequently installed browsers, Firefox and Internet Explorer respectively].

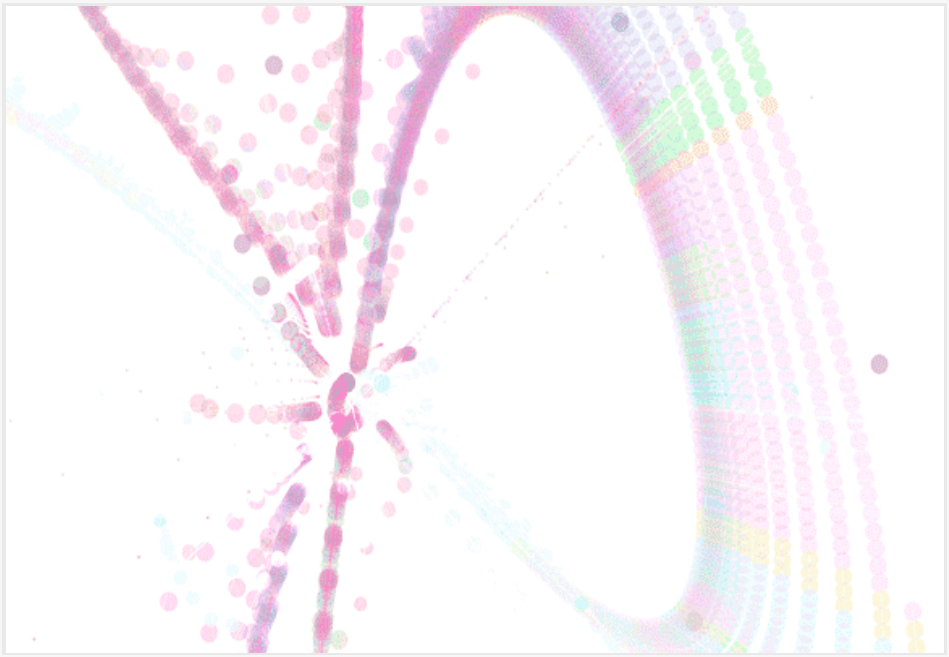
Using JavaScript in Quartz Composer is notably different from working with JavaScript in almost any other application, even in Flash, wherein much of the in built scripting support is understandably oriented towards the production and support of browser delivered, event driven , rich media interfaces and presentations.

Currently their is some browser support for QC files, however it limited by reasons of framework and plugin distribution base to Safari and any other WebKit plugin capable, OS X installed browser. See the WebKit Plugin page for further details on exploiting Quartz Composer compositions within Safari [WebKit plugin browsers].

This current lack of cross platform, cross browser suport for Quartz Composer compositions really matters little as such on the only platform currently supporting such compositions, namely OS X Tiger and Leopard.

For those of us wanting to to originate and program our own compositions, plugins and patches we only have to install the standard Developer's Tool package and open up Quartz Composer. For those wishing to only be consumers and users of Quartz Compositions, most will work in QuickTime Player unless specifically protocolised to only work as visualizers, screensavers and such. In that case, they will work as expected in their native environment.

QC JavaScript is radically direct in the way it operates upon input data and outputs results that support the generation of graphical structures and images.



Although Quartz Composer uses JavaScript Core Classes, it does not cover the DOM based use of JavaScript that is probably running in your browser today, Friday, June 26, 2009 . Of course, learning about JavaScript in Quartz Composer is hardly going to dent your learning curve with DOM based prototype JS. It will, in all probability, help you to gain greater facility in your use of this widely employed scripting language.

- toPrecision

Array

- Array
- concat
- join
- length
- pop
- push
- reverse
- shift
- slice
- sort
- splice
- toString
- unshift

Math

- abs
- acos
- asin
- atan
- atan2
- ceil
- cos
- E
- exp
- floor
- LN2
- LN10
- log
- LOG2E
- LOG10E
- Math
- max
- min
- PI
- pow
- random
- round
- sin
- sqrt
- SQRT1_2
- SQRT2
- tan

Anyone wanting to specifically research or learn about JavaScript in their browser should look instead to the following links:-

- Mozilla Dev ECMA specifications
- W3.Org DOM Specification
- W3Schools JavaScript Tutorial
- Internet.com

See the Apple Developer documents for a simple guide to using JavaScript in Quartz Composer.

See the resources page for a fuller set of Quartz Composer and QC JavaScript links.

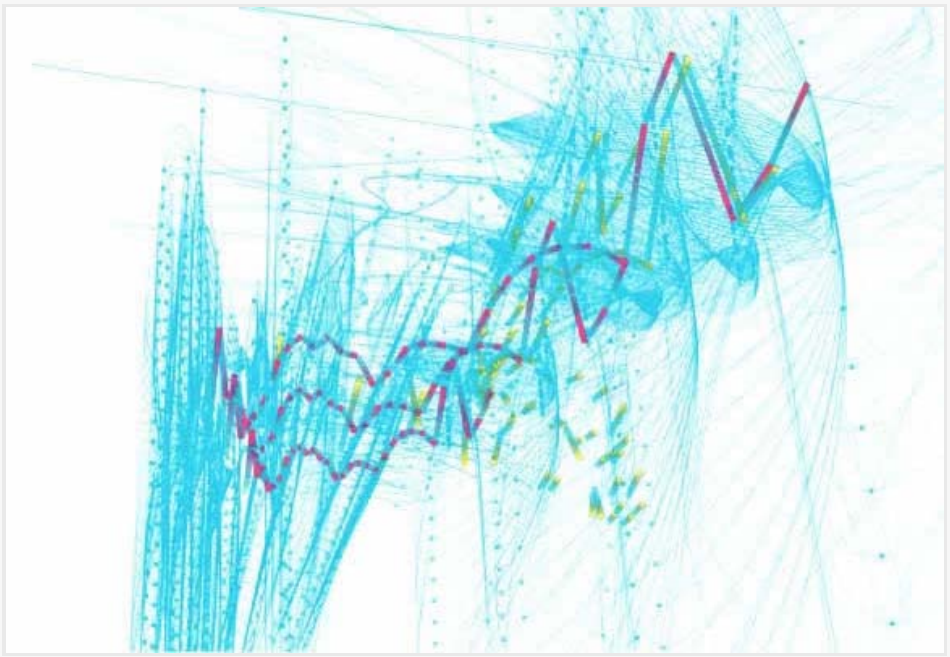
All QC 2 example scripts point to their original location at OpenSPC

All QC 3 example scripts point to their original location at Cybero Designs

Errors, updates, insights and any links to contact

If requesting redistribution from Kazuhiro Furuhata , enquire via OpenSPC .

Restyling, revisions & editing by Cybero Designs (with kind permission of Kazuhiro Furuhata)



Last modified

Date

- Date
- getDate
- getDay
- getFullYear
- getHours
- getMilliseconds
- getMinutes
- getMonth
- getSeconds
- getTime
- getTimezoneOffset
- GetUtcDate
- getUTCDay
- getUTCFullYear
- getUTCHours
- getUTCMilliseconds
- getUTCMinutes
- getUTCMonth
- getUTCSeconds
- getYear
- parse
- setDate
- setFullYear
- setHours
- setMilliseconds
- setMinutes
- setMonth
- setSeconds
- setTime
- setUTCDate
- setUTCFullYear
- setUTCHours
- setUTCMilliseconds
- setUTCMinutes
- setUTCMonth
- setUTCSeconds
- setYear
- toDateString
- toGMTString
- toLocaleDateString
- toLocaleString
- toLocaleTimeString
- toTimeString
- toUTCString
- UTC

String

- charAt
- charCodeAt
- fromCharCode
- indexOf
- lastIndexOf
- length
- match
- replace
- search
- slice
- split
- substr
- substring
- toLowerCase
- toUpperCase

RegExp / regex

- $1 $9 $ 1 - $ 9
- compile
- exec
- sticky
- global
- ignoreCase
- input
- lastIndex
- lastMatch
- lastParen
- leftContext
- multiline
- RegExp
- rightContext
- source
- test
- Special Characters

resources

Applications & Plugins

The following applications all use or exploit Quartz Composer files. This is by no means an exhaustive list.

OS X & iPhone OS
iWork
Final Cut Studio
coge
Photo Presenter
VDMX
AudioCodex
FXFactory
Q@mera
Eskatonia - VFX, CoreMelt, Inside Us All, Quartonian

Tutorials

Apple Developer Tutorials - 2005 Apple Developer Tutorials - 2008
Kineme Vimeo Tutorials Kineme Wiki Kineme Tutorials
QuartzCompositions - some broken links
whitsitt - 2009
vjkungfu - 2009
goto10 Vimeo - 2009
qtzcodex - 2008
hybrid visuals - 2008
dvcreators.net - 2008
Digital Motion - 2007 Digital Motion - 2008
Mac apper - Part1 & Part 2 - 2007
fdiv - Tiger only - 2007
O'Reilly Mac Dev Center - 2006
Steam Shift - 2006
Eskatonia Archive - 2006
createdigitalmusic - 2005
wonder how to - 2005 - 2009
podcast producers - Filters List

Examples & Samples

Futurismo Zugakousaku
Quartzcompositions.com

Kineme Interactive Media - especially the Applications section, although the Forums also contain example files.

qtzlcodex

Quartz Candy

toneburst - machines don't care

memo.tv

VIDVOX WIKI

cybero

Mailing Lists

Apple's QC Dev List

Kineme Info List

Custom Patches

Google Patches

Kineme Patches

Quartzcandy

Qtzlcodex

Books

Zugakousaku Quartz Composer Book

Images / Videos

Flikr

Vimeo

YouTube

Wiki

QC Wiki QC2

QC Wiki QC3

Kineme Wiki

Wikipedia

Origins

pixelshox - where Quartz Composer came from :-)

External Links - JavaScript

Mozilla Dev ECMA specifications

W3.Org DOM Specification

W3Schools JavaScript Tutorial

Internet.com

Return to the Start Page

Last modified

JavaScript Patch - some notes

Examples Notes

Caveats & Discoveries

The following has been found to hold true for the JavaScript Patch in Quartz Composer.

If all you really need is outputs[0] = inputs [0] + 1 then you would be better served by a math expression patch - honestly.

Do not try to edit a script whilst rendering is in progress within the edited composition. Will result in a hang.

Do apply a random or patch time to the JavaScript's timebase, otherwise it will likely as not stop refreshing inputs.

There is a maximum amount of code you can run in any one JavaScript patch. The workaround to this is to compartmentalise one's code if possible.

Under certain conditions, you can find yourself working unintentionally in Native Code when using the JavaScript Patch.

Wish List

Screen Grabs

Notes

External Links

Mozilla Dev ECMA specifications
W3.Org DOM Specification
W3Schools JavaScript Tutorial
Internet.com

Return to the Start Page

Last modified

Comments, annotations

T y p e Comment

F o r m a t
/ / Comment line
/ *
Multi-line comments
* /

C o m m e n t a r y
Comments and annotations. Comments will not run in the code.
// creates a single line comment and will comment until the beginning of the next line of uncommented code.
It is often used to define header comments in code or to provide a line by line commentary upon the code preceeding.
Multi Line comments can be made within the slash & asterisk area. /~*/ .*
Both single line & multiline comments can be freestanding of each other.
Useful for when you want to redraft code or pass code around.

Sample code - QC 2 & QC 3 JavaScript

```
/ / This is a comment. It doesn't affect the uncommented code following below.
a = 12;
/ *
Is a multi-line comments. Consequently the variable a won't hold a value of 34.
a = 34;
* /
outputs [0] = a;
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

break

Type Instruction

Format break

Commentary *Creates a conditionally based break from a looped routine. If a given condition is met , break the loop and continue executing any remaining code following after the break statement. The other command that breaks from a current loop is continue.*

Sample code - QC 2 JavaScript

```
for (i = 0; i <10; i ++)
{ (
outputs [0] = i;
f (i == 2) break;
} )
```

Sample code - QC 3 JavaScript

```
function (__number boolOutput) main (__number i, __number j)
{
if(i > j) if (j == 3) break;

var result = new Object();
result.boolOutput = j;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

continue

Type Instruction

Format continue expression

Commentary *The continue command will break a current loop upon a given condition being arrived at and continue on to the next value achievable.*

Sample code - QC 2 JavaScript

```
for (i = 0; i <5; i ++)
{(
if (i == 2) continue;
outputs [0] = i;
} )
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
for (i=0; i<5; i++)
{
if (i == 2) continue; }

result.outputNumber = i;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

delete

T y p e Instruction

F o r m a t delete expression //objects or variables

C o m m e n t a r y *The delete operator deletes an object, an object's property, or an element at a specified index in an array.*

Sample code - QC 2 JavaScript

```
var result = new Object();
arbor = new Array ("oak", "maple", "beech", "pine", "fir");
delete arbor[3];
if(3 in arbor)
{ result.outputNumber = false; }
else { result.outputNumber = true; }
return result;
```

Sample code - QC 3 JavaScript

```
function (__string outputArray) main ()
{var result = new Object();
var myArrayB = new Array("redwood","bay", "apple","cedar","oak","maple");
delete myArrayB[3];
if ( 1 in myArrayB)
{
myArrayB.sort ();
result.outputArray = myArrayB.toString();
}
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

do ... while

T y p e Instruction

F o r m a t

{do (
Processing
}) while (condition)

C o m m e n t a r y *The do...while loop is a variant of the while loop. This loop will always execute a block of code ONCE, and then it will repeat the loop as long as the specified condition is true. This loop will always be executed at least once, even if the condition is false, because the code is executed before the condition is tested.*

Sample code - QC 2 JavaScript

```
n = 0;
do {
n ++;
outputs [0] = n;
} while (n <3);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{ var result = new Object();
n = 0;
do { n++; } while (n<3);
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

for

T y p e Instruction

F o r m a t
for ([initial-expression]; [condition]; [final-expression]) statement

C o m m e n t a r y *Creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons, followed by a statement executed in the loop.*

Sample code - QC 2 JavaScript

```
for (i = 0; i <3; i ++)
(
outputs [0] = i;
)
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{var result = new Object();
var inputNumber = new Array();
var i = new Array();
var outputNumber = i;
for (i=0; i<36; i++)
result.outputNumber =i;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

for ... in

T y p e Instruction

F o r m a t
for (variable in object name)
(
Processing
)

C o m m e n t a r y *Iterates a specified variable over all the properties of an object, in arbitrary order. For each distinct property, the specified statement is executed.*

Sample code - QC 2 JavaScript

```
for (i in Math)
(
outputs [0] = i;
)
```

Sample code - QC 3 JavaScript

```
function (__boolean outputNumber) main () //
{var result = new Object();//
arbor = new Array ("oak", "maple", "beech", "pine", "fir");
delete arbor[3];
if(3 in arbor)//
{
result.outputNumber = false;
}
else
{
result.outputNumber = true;
}

return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

function

T y p e Instruction

F o r m a t
function name (argument1, argument 2, ...)
(
Processing Statements
)

C o m m e n t a r y *Declares a function with the specified parameters. Arguments, () can be a comma separated list. The statements comprise the body of the function.To return a value a return statement is needed to specify the returnable value.*

Sample code - QC 2 JavaScript

```
function calc(a,b)
{ return a * b;
}
outputs[0] = calc(2,4);
```

Sample code - QC 3 JavaScript

```
var inputNumber = new Array();
var outputNumber = new Array();
var c = outputNumber;
var a = inputNumber[1];
var b = inputNumber[0];
//var a = 32;
//var b = 6.752356;
function (__number outputNumber) main (__number a, __number b )
{var result = new Object();
function calc(a,b)
{
return a * b;
}
result.outputNumber = calc(a,b);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

if ... else

T y p e Instruction

F o r m a t

if (condition1)
statement1
else if (condition2)
statement2
else if (condition3)
statement3
...
else
statementN

C o m m e n t a r y *If you meet the criteria if () to run the following. else else after the operation, which does not meet the criteria. if the multiple nested (nested) can be.*

Sample code - QC 2 JavaScript

```
n = (new Date ()). getSeconds ();
if (n <30)
{ (
outputs [0] = "30 is less than";
else (
outputs [0] = "30 is over";
} )
```

Sample code - QC 3 JavaScript

```
function ( __string seconds) main ( __number TimeIn[1])
{var result = new Object();
var TimeIn = (new Date()).getSeconds();
if (TimeIn < 30)
{
seconds = TimeIn +" seconds is less than 30 seconds ";
}else{
seconds = TimeIn +" seconds is more than 30 seconds ";
}
result.seconds = seconds;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

new

Type Instruction

Format new object name

Commentary *Creates a new object.*

Sample code - QC 2 JavaScript

```
dObj = new Date ();
outputs [0] = "now" + dObj.getHours () + "is the time";
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[1])
{
var result = new Object();
var dObj = new Date();
result.outputNumber = "The Time Now Is :- "+ dObj.getHours() + " Hours " +dObj.getMinutes(2) + "
Minutes ";
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

return

Type Instruction

Returns return **Format**

Commentary *Specify the return value from function. If you want to specify multiple return values return ["Ab", 12, "OK"] is used as an array.*

Sample code - QC 2 JavaScript

```
function calc (a, b)
(
return a + b;
)
outputs [0] = calc (2,3);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
function calc(a,b)
{
return a + b;
}
result.outputNumber = calc(2,3);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

switch

T y p e Instruction

F o r m a t

switch (check value)
(
ase value 1: If the value of a process; break;
case value 2: The values of the two cases were handled; break;
value case 3: if the value of treatment was 3; break;
:
default: case processing in the case of non-specified value;
)

C o m m e n t a r y *Evaluates an expression, matching the expression's value to a case label, and executes statements associated with that case.*

Sample code - QC 2 JavaScript

```
n = (new Date ()). getSeconds ();
n = n% 3;
switch (n)
(
case 0: outputs [0] = "is zero"; break;
case 1: outputs [0] = "is one"; break;
default: outputs [0] = "0 and 1 are non-";
)
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{var result = new Object();
var outputNumber = new Array();
var n = new Array();
n = (new Date()).getSeconds();
n = n % 3;
switch(n)
{
case 0:outputNumber[0] = "0"; break;
case 1:outputNumber[0] = "1"; break;
default:outputNumber[0] = "2";
}
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

this

Type Instruction

Format this This

Commentary *Shows the object itself.*

Sample code - QC 2 JavaScript

```
function calc(a,b) function calc (a, b)
(
this.mul = a * b;
this.div = a / b;
this.add = a + b;
this.sub = a - b;
)
cObj = new calc (2,3);
outputs [0] = "Addition result is" + cObj.add + "is";
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[2])
{
var result = new Object();
function calc(a,b)
{
this.mul = a * b;
this.div = a / b;
this.add = a + b;
this.sub = a - b;
}
cObj = new calc(2,3);

result.outputNumber = "Number "+cObj.add+" Results";
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

throw

Type Instruction

Format throw error

Commentary *Cause any errors catch () you can pass as an error code.*

Sample code - QC 2 JavaScript

```
try (
throw "zero";
)
catch (e)
(
outputs [0] = "content error (" + e +")";
)
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[1])
{
var result = new Object();
try {
throw "zero";
}
catch(e)
{
result.outputNumber = "content error ("+e+")";
}
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

try

Type Instruction

Format

try
(
Processing potential of the error
)
catch (argument)
(
For treatment of specific causes of error
)
finally
(
Error of the entire process
)

Commentary *instruction that may try errors, and surround processing. If an error occurs catch, finally in the process when an error occurs. try is nested (nested) can be.*

Sample code - QC 2 JavaScript

```
try (
a = 1 / 0;
)
catch (e)
(
alert ( "content error (" + e +")");
)
y finally
(
outputs [0] = "Error";
)
```

Sample code - QC 3 JavaScript

```
function (__string strOut) main ()
{var result = new Object();
try {
a = 10 / 3;
}
catch(e)
{
alert("Error contents("+e+")");
}
finally
{
} result.strOut ="Error";
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

var

Type Instruction

Format var variable name = value or sum;

Commentary *Declare the variables. If the function is declared in a local variable. You can also declare and assign a value. var a = b = c = 9; var a = b = c = 9; variables and a, b, c will be assigned to 9.*

Sample code - QC 2 JavaScript

```
var a = 12;
outputs [0] = a;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
var a = 1.6359867;
result.outputNumber = a;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

while

Type Instruction

Format
while (
Processing
)

Commentary *Repeat the block between the conditions are met. Decision is to check the first condition, one may also repeat the process.*

Sample code - QC 2 JavaScript

```
n = 0;
while (n <3) (
n ++;
outputs [0] = n;
)
```

Sample code - QC 3 JavaScript

```
var outputNumber = new Number();
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
var n = inputNumber[0];
var x = inputNumber[1];
while (n < 3) {
n ++;
x += n;
}
result.outputNumber = x;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

with

T y p e Instruction

F o r m a t

with (name omitted object) (
Processing
)

C o m m e n t a r y *You can omit the object name and description specified. You can also nest.*

Sample code - QC 2 JavaScript

```
with (Math) (
outputs [0] = PI;
)
```

Sample code - QC 3 JavaScript

```
function (__number out) main ()
{
var result = new Object();
var out = new Number();
var x = out;
with(Math)
var x = PI;
{
result.out = x;
}
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

Boolean

T y p e Object

F o r m a t new Boolean (value)

C o m m e n t a r y *he Boolean object is an object wrapper for a boolean value.The value passed as the first parameter is converted to a boolean value, if necessary. If value is omitted or is 0, -0, null, false, NaN, undefined, or the empty string (""), the object has an initial value of false. All other values, including any object or the string "false", create an object with an initial value of true.*

Sample code - QC 2 JavaScript

```
flag = new Boolean (true);
outputs [0] = flag.toString ();
```

Sample code - QC 3 JavaScript

```
var flag = new Boolean(true);
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = flag.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

arguments

T y p e Global Object

F o r m a t
arguments.length
arguments [reference number]

C o m m e n t a r y
An array-like object corresponding to the arguments passed to a function.The arguments object is a local variable available within all functions. Has arguments.callee and arguments.length properties.

Sample code - QC 2 JavaScript

```
outputs [0] = test (123,456);
{ function test () (
var n = arguments.length;
return n.toString ();
} )
```

Sample code - QC 3 JavaScript

```
var test = new Array(); // initial value of false
function (__string outputArray) main ()
{ function test(){
var n = arguments.length;
return n.toString();
} var result = new Object();
result.outputArray = test(123,456);
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

Function

T y p e Function

F o r m a t

function (__number outputNumber) main (__number inputNumber[2])

function ([arg1[, arg2[, ... argN]],]
(
Processing Statements
)
functionBody
)

C o m m e n t a r y *Every function is actually a Function object. To return a value a return statement is needed to specify the returnable value.*

Sample code - QC 2 JavaScript

```
outputs[0] = getSec();
function getSec(){
var dObj = new Date();
var n = dObj.getSeconds();
return n.toString();
}
```

Sample code - QC 3 JavaScript

```
var test = new Array();
var a = 2;
var b = 4; //
function (__number outputNumber) main (__number a, __number b)
{ function calc(a,b)
{
return a * b;
}
var result = new Object();
result.outputNumber = calc(a,b);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

Number

Type Global Object

Format new Number (value)

Commentary *Number (Number) object. You can specify the numeric argument (or omitted). If the argument cannot be converted into a number, it returns Nan.*

Sample code - QC 2 JavaScript

```
n = new Number (12.345);
outputs [0] = n.toString ();
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[2])
{
var result = new Object();
n = new Number(12.345);
result.outputNumber = n.toString();
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

MAX_VALUE

Type Property (R)

Format Number.MAX_VALUE

Commentary *The MAX_VALUE property has a value of approximately 1.79E+308. Values larger than MAX_VALUE are represented as "Infinity".*

Sample code - QC 2 JavaScript

```
outputs[0] = Number.MAX_VALUE; outputs [0] = Number.MAX_VALUE;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
result.outputNumber = Number.MAX_VALUE;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

MIN_VALUE

Type Property (R)

Format Number.MIN_VALUE

Commentary *The smallest representable number. The MIN_VALUE property is the number closest to 0, not the most negative number, that JavaScript can represent. MIN_VALUE has a value of approximately 5e-324. Values smaller than MIN_VALUE ("underflow values") are converted to 0.*

Sample code - QC 2 JavaScript

```
outputs [0] = Number.MIN_VALUE;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
result.outputNumber = Number.MIN_VALUE;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

NaN

Type Property (R)

Format Number.NaN

Commentary *Indicates Non-numeric (Not a Number) . Special "not a number" value.*

Sample code - QC 2 JavaScript

```
outputs [0] = Number.NaN;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
result.outputNumber = Number.NaN;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

NEGATIVE_INFINITY

Type Property (R)

Format Number.NEGATIVE_INFINITY

Commentary *Special value representing negative infinity; returned on overflow.This value behaves slightly differently than mathematical infinity:*
Any positive value, including POSITIVE_INFINITY, multiplied by NEGATIVE_INFINITY is NEGATIVE_INFINITY.
Any negative value, including NEGATIVE_INFINITY, multiplied by NEGATIVE_INFINITY is POSITIVE_INFINITY.
Zero multiplied by NEGATIVE_INFINITY is NaN.
NaN multiplied by NEGATIVE_INFINITY is NaN.
NEGATIVE_INFINITY, divided by any negative value except NEGATIVE_INFINITY, is POSITIVE_INFINITY.
NEGATIVE_INFINITY, divided by any positive value except POSITIVE_INFINITY, is NEGATIVE_INFINITY.
NEGATIVE_INFINITY, divided by either NEGATIVE_INFINITY or POSITIVE_INFINITY, is NaN.
Any number divided by NEGATIVE_INFINITY is Zero.

Sample code - QC 2 JavaScript

```
outputs[0] = Number.NEGATIVE_INFINITY; outputs [0] = Number.NEGATIVE_INFINITY;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
result.outputNumber = Number.NEGATIVE_INFINITY;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

POSITIVE_INFINITY

Property Type (R)

Format Number.NEGATIVE_INFINITY

Commentary *Special value representing infinity; returned on overflow. This value behaves slightly differently than mathematical infinity:*
Any positive value, including POSITIVE_INFINITY, multiplied by POSITIVE_INFINITY is POSITIVE_INFINITY.
Any negative value, including NEGATIVE_INFINITY, multiplied by POSITIVE_INFINITY is NEGATIVE_INFINITY.
Zero multiplied by POSITIVE_INFINITY is NaN.
NaN multiplied by POSITIVE_INFINITY is NaN.
POSITIVE_INFINITY, divided by any negative value except NEGATIVE_INFINITY, is NEGATIVE_INFINITY.
POSITIVE_INFINITY, divided by any positive value except POSITIVE_INFINITY, is POSITIVE_INFINITY.
POSITIVE_INFINITY, divided by either NEGATIVE_INFINITY or POSITIVE_INFINITY, is NaN.
Any number divided by POSITIVE_INFINITY is Zero.

Sample code - QC 2 JavaScript

```
outputs [0] = Number.POSITIVE_INFINITY;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
result.outputNumber = Number.POSITIVE_INFINITY;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toExponential

T y p e method

F o r m a t number.toExponential([fractionDigits])

C o m m e n t a r y *Returns a string representing the number in exponential notation. Sets the number of digits before and after the decimal point.*

Sample code - QC 2 JavaScript

```
n = inputs [0]. toExponential (3);
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[2])
{
var result = new Object();
n = inputNumber[0].toExponential(3);
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toFixed

T y p e method

F o r m a t Numeric. toFixed (digits)

C o m m e n t a r y *To specify the decimal position. Returns a string representing the number in fixed-point notation, range is 0 to 20.(Unconfirmed)*

Sample code - QC 2 JavaScript

```
n = inputs [0]. toFixed (1);
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[2])
{var result = new Object();
n = inputNumber[0].toFixed(1);
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getTime

Type method

Format Date object. GetTime ()

Commentary *1 January 1970 at returns 0 milliseconds from the time.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getTime ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var Time = str;
var dateObj = new Date();
var Clock = dateObj.getTime();
var str = dateObj.toTimeString(); function (__string Time) main (__number Clock[1])
{
var result = new Object();
if (dateObj.getTime = 16)
{
result.Time = " The Time is exactly" + str;
}
else
if (dateObj.getTime < 16)
{
result.Time = " The Time is before 16:00" + str;
}
else
if (dateObj.getTime > 30)
{
result.Time = " The Time is after 16:00" + str;
}
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toPrecision

Type method

Format Numeric . toPrecision (digits)

Commentary *The specified number of digits in value. Returns a string representing the number to a specified precision in fixed-point or exponential notation. Value range for notation is 1 to 100.(Unconfirmed)*

Sample code - QC 2 JavaScript

```
n = inputs [0]. toPrecision (4);
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[2])
{
var result = new Object();
n = inputNumber[0].toPrecision(4);
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

Array

T y p e Object

F o r m a t
Object array = new Array ()
Object array = new Array (value)
Object array = new Array (value value ,.....,)
Object array = []
Object array = [value ,..., value]
Array object. Sequence object method. Property

C o m m e n t a r y *Array object. If an argument, will generate as many elements as in the specified array.*

Sample code - QC 2 JavaScript

```
myAry = new Array (12,34,56);
n = myAry [1];
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var myAry = new Array(66,12,26,75);
var outputArray = new Array();
n = myAry[2];
function (__number outputArray) main ()
{ var result = new Object();
result.outputArray = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

concat

Type method

Format concat(string2, string3[, ..., stringN])

Commentary *Combines the text of two or more strings and returns a new string..*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34,56);
myAry2 = new Array ( "AB", "CD", "EF");
n = myAry1.concat (myAry2);
outputs [0] = n.toString ();
```

Sample code - QC 3 JavaScript

```
var myAryA = new Array(66,12,26,75);
var myAryB = new Array("AB","CD","EF");
var outputArray = new Array();
var n = myAryA.concat(myAryB);
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = n.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

join

Type method

Format String = array object. Join (letters) join(separator)

Commentary *The string conversions of all array elements are joined into one string. Does not modify array*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34, "AB", "CD");
n = myAry1.join ("/");
outputs [0] = n.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD");
var outputArray = new Array();
var n = myArrayB.join("/");
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = n.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

length

T y p e Property (R / W)

F o r m a t Object = sequence number of elements. Length

C o m m e n t a r y *Returns number of array elements. The value of the length property is an integer with a positive sign and a value less than 2 to the 32 power (232).*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34, "AB", "CD");
n = myAry1.length;
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD");
var outputArray = new Array();
var n = myArrayB.length;
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = n.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

pop

Type method

Format array elements = array object. Pop () array.pop()

Commentary *Remove the array element at the end. Removes the last element from an array and returns that element.*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34, "AB", "CD");
n = myAry1.pop ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD");
var outputArray = new Array();
var n = myArrayB.pop();
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = n.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

push

Type method

Format Object array . Push (additional elements) array.push(element1, ..., elementN)

Commentary *Add an array element to the end. Adds one or more elements to the end of an array and returns the new length of the array.*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34, "AB", "CD");
myAry1.push ( "XYZ");
outputs [0] = myAry1.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD");
var outputArray = new Array();
var n = myArrayB.push("XYZ");
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = myArrayB.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

reverse

Type method

Format Object array.reverse();

Commentary *Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34, "AB", "CD");
n = myAry1.reverse ();
outputs [0] = n.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD");
var outputArray = new Array();
var n = myArrayB.reverse();
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = myArrayB.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

shift

Type method

Format array.shift()

Commentary *Removes the first element from an array and returns that element.*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34, "AB", "CD");
myAry1.shift ();
outputs [0] = myAry1.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD");
var outputArray = new Array();
var n = myArrayB.shift();
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = myArrayB.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

slice

Type method

Format slice(begin[,end])

Commentary *Extracts array elements in the range specified. The end is optional. Does not modify array*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34, "AB", "CD", 56,78);
n = myAry1.slice (1,3);
outputs [0] = n.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD",56,78);
var outputArray = new Array();
var n = myArrayB.slice(1,3);
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = n.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

sort

T y p e method

F o r m a t Object array . Sort (compare function)

C o m m e n t a r y *Sort the elements of an array . Compare function is optional. Specifies a function that defines the sort order. If omitted, the array is sorted lexicographically (in dictionary order) according to the string conversion of each element.*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (5,7,2,8,1);
n = myAry1.sort ();
outputs [0] = n.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(5,7,2,8,1);
var outputArray = new Array();
var n = myArrayB.sort();
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = n.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

splice

Type method

Format Extract elements = array object. Splice (position, length, string replacement)

Commentary *Splice the split location for the specified range or an array element. The return value is the element of the range specified in the original sequence and the remaining element.If the replacement string is specified, the replacement string is inserted after the position or range specified in the original sequence. Replacement string can be listed separated by commas.*

Sample code - QC 2 JavaScript

```
a = new Array (12,34, "AB", "CD", "EF");
b = a.splice (1,2, "ZZ");
outputs [0] = a.toString () + String.fromCharCode (10) + b.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD","EF");
var b = myArrayB.splice(1,2,"ZZ");
var outputArray = new Array();
var n = myArrayB.toString()+String.fromCharCode(10)+b.toString();
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = n.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toString

Type method

Format array . toString ()

Commentary *Converts a string array element. Returns a string representing the array and its elements.The Array object overrides the toString method of Object.*

Sample code - QC 2 JavaScript

```
myAry = new Array (12,34,56);
outputs [0] = myAry.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,24,36,48,60);
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = myArrayB.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

unshift

T y p e method

F o r m a t Object array . Unshift (,..., elements add additional elements)

C o m m e n t a r y *Adds one or more elements to the beginning of an array and returns the new length of the array.*

Sample code - QC 2 JavaScript

```
myAry1 = new Array (12,34, "AB", "CD");
myAry1.unshift ( "XYZ");
outputs [0] = myAry1.toString ();
```

Sample code - QC 3 JavaScript

```
var myArrayB = new Array(12,34,"AB","CD");
var outputArray = new Array();
var n = myArrayB.unshift("XYZ");
function (__string outputArray) main ()
{ var result = new Object();
result.outputArray = myArrayB.toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

abs

Type method

Format Math.abs (value)

Commentary *Determine the absolute value of value. If one is -1. 0 is still 0.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.abs (inputs [0]);
```

Sample code - QC 3 JavaScript

```
var input = new Array();
var out = new Array();
function (__number out[1]) main (__number input[1])
{
var result = new Object();
result.out = Math.abs(input);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

acos

Type method

Format Math.acos (value)

Commentary *Takes the arc cosine of value.*

Sample code - QC 2 JavaScript

```
n = inputs[0] / 3; n = inputs [0] / 3;
outputs [0] = Math.acos (n);
```

Sample code - QC 3 JavaScript

```
var out = new Number();
var input = new Number();
var n = input / 3;
var out = new Number();
function (__number out[1]) main (__number input[1])
{
var result = new Object();
var n = input / 3;
result.out = Math.acos(n);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

asin

Type method

Math.asin() **Format** Math.asin (value)

Commentary *Takes the arc sine of value.*

Sample code - QC 2 JavaScript

```
n = inputs [0] / 3;
outputs [0] = Math.asin (n);
```

Sample code - QC 2 JavaScript

```
var out = new Number();
var input = new Number();
var n = input / 3;
var out = new Number();
function (__number out[1]) main (__number input[1])
{
var result = new Object();
var n = input / 3;
result.out = Math.asin(n);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

atan

Type method

Format Math.atan (value)

Commentary *Takes the arctangent of the value.*

Sample code - QC 2 JavaScript

```
n = inputs [0] / 3;
outputs [0] = Math.atan (n);
```

Sample code - QC 3 JavaScript

```
var out = new Number();
var input = new Number();
var n = input / 3;
var out = new Number();
function (__number out[1]) main (__number input[1])
{
var result = new Object();
var n = input / 3;
result.out = Math.atan(n);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

atan2

Type method

Format Math.atan (Y, X)

Commentary *Angle from the XY coordinates (in radians) calculated.*

Sample code - QC 2 JavaScript

```
x = inputs [0];
y = inputs [1];
rad = Math.atan2 (y, x);
outputs [0] = rad * 180/Math.PI;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
x = inputNumber[0];
y = inputNumber[1];
rad = Math.atan2(y,x);
result.outputNumber = rad * 180/Math.PI;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

ceil

Type method | Static

Format Math.ceil (value)

Commentary *Returns the smallest integer greater than or equal to a number.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.ceil (inputs [0]);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
result.outputNumber = Math.ceil(inputNumber);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

cos

Type method

Format Math.cos (value)

Commentary *Takes the cosine of the value.*

Sample code - QC 2 JavaScript

```
n = inputs [0] / 3;
outputs [0] = Math.cos (n);
```

Sample code - QC 3 JavaScript

```
function (__number out) main (__number input[1])
{
var result = new Object();
var n = input / 3;
result.out = Math.cos(n);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

E

E

Type Property (R)

Format Math.E

Commentary

E is the value base of natural logarithms. Euler's constant and the base of natural logarithms, approximately 2.718.

Sample code - QC 2 JavaScript

```
outputs [0] = Math.E;
```

Sample code - QC 3 JavaScript

```
var outputNumber = new Number();
function (__number outputNumber) main ()
{
var result = new Object();
result.outputNumber = Math.E;
return result;
}
```

Download QC2 example script

Return to the Start Page

Last modified

exp

Type method

Format Math.exp (value)

Commentary *Return the power of e.*

Sample code - QC 2 JavaScript

```
n = inputs [0] / 3;
outputs [0] = Math.exp (n);
```

Sample code - QC 3 JavaScript

```
function (__number out) main (__number input[1])
{
var result = new Object();
var n = input / 3;
result.out = Math.exp(n);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

floor

T y p e method

F o r m a t Math.floor (value)

C o m m e n t a r y *Floor value.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.floor (inputs [0]);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
result.outputNumber = Math.floor(inputNumber);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

LN2

T y p e Property (R) Static | Global Math Object | Static

F o r m a t Math.LN2

C o m m e n t a r y *2 is the value of the natural logarithm.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.LN2;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
result.outputNumber = Math.LN2;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

LN10

T y p e Property (R)

F o r m a t Math.LN10

C o m m e n t a r y *Natural logarithm to the value of 10.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.LN10;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
result.outputNumber = Math.LN10;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

log

Type method

Format Math.log (value)

Commentary *Logarithmic returns.*

Sample code - QC 2 JavaScript

```
n = inputs [0] / 10;
outputs [0] = Math.log (n);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
var n = inputNumber / 10;
result.outputNumber = Math.log(n);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

LOG2E

T y p e Property (R) Static | Global Math Object | Static

F o r m a t Math.LOG2E

C o m m e n t a r y *Base 2 logarithm of E, approximately 1.442.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.LOG2E;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
result.outputNumber = Math.LOG2E;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

LOG10E

T y p e Property (R) Static, Read-only

F o r m a t Math.LOG10E

C o m m e n t a r y *Base 10 logarithm of E, approximately 0.434.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.LOG10E;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
result.outputNumber = Math.LOG10E;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

Math

Type Object

Format
Math. Methods
Math. Property

Commentary *Perform mathematical operations and is an object with a property that holds a specific value.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.PI;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
str = "Sample_sample123";
n = str.match(/mp/g);
n = n.length;
result.outputNumber = n;
return result;
}
```

Download QC2 example script

Return to the Start Page

max

T y p e method

F o r m a t Math.max (value1, value 2)

C o m m e n t a r y *2 returns a larger value compared to the value of one.*

Sample code - QC 2 JavaScript

```
n = inputs [0];
outputs [0] = Math.max (n, 0.5);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
str = "Sample_sample123";
n = str.match(/mp/g);
n = n.length;
result.outputNumber = n;
return result;
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

min

Type method

Format Math.min (value1, value 2)

Commentary *2 returns a small value compared to the value of one.*

Sample code - QC 2 JavaScript

```
n = inputs [0];
outputs [0] = Math.min (n, 0.5);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{var result = new Object();
n = inputNumber[0];
result.outputNumber = Math.min(n,0.085);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

PI

Property Type (R)

Format Math.PI

Commentary *The value of π is pi.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.PI;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
result.outputNumber = Math.PI;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

pow

Type method

Format Math.pow (base, index)

Commentary *Power returns.*

Sample code - QC 2 JavaScript

```
n = inputs [0];
outputs [0] = Math.pow (n, 2);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
n =inputNumber[0];
result.outputNumber = Math.pow(n,2);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

random

Type method

Format Math.random ()

Commentary *Returns a random value.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.random ();
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
result.outputNumber = Math.random();
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

round

Type method

Format Math.round (value)

Commentary *The rounded value of the stated input.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.round (inputs [0]);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
result.outputNumber = Math.round(inputNumber[0]);
return result;
}
```

Download QC2 example script

Return to the Start Page

sin

Type method

Format Math.sin (value)

Commentary *Returns the sine of a number.*

Sample code - QC 2 JavaScript

```
n = inputs [0] / 3;
outputs [0] = Math.sin (n);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{
var result = new Object();
n = inputNumber[0] / 3;
result.outputNumber = Math.sin(n);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

sqrt

Type method

Format Math.sqrt (value)

Commentary *Returns the square root value.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.sqrt (inputs [0]);
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{var result = new Object();
result.outputNumber = Math.sqrt(inputNumber[0]);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

SQRT1_2

Type Property (R)

Format Math.SQRT1_2

Commentary *2 is the square root of half the value.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.SQRT1_2;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
result.outputNumber = Math.SQRT1_2;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

SQRT2

Property Type (R)

Format Math.SQRT2

Commentary *The value of the square root of two.*

Sample code - QC 2 JavaScript

```
outputs [0] = Math.SQRT2;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
result.outputNumber = Math.SQRT2;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

tan

Type method

Format Math.tan (value)

Commentary *Takes the tangent of the value.*

Sample code - QC 2 JavaScript

```
n = inputs [0] / 3;
outputs [0] = Math.tan (n);
```

Sample code - QC 3 JavaScript

```
function (__number out[1]) main (__number input[1])
{
var result = new Object();
var n = input[0] / 3;
result.out = Math.tan(n);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

Date

Type Object

Format

new Date()
new Date(milliseconds)
new Date(dateString)
new Date(year, month, date [, hour, minute, second, millisecond])

Commentary *Creates Date instances which let you work with dates and times. If you supply no arguments, the constructor creates a Date object for today's date and time according to local time. the system (OS) does not affect the clock.*

Sample code - QC 2 JavaScript

```
outputs [0] = (new Date ()). toString ();
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[1])
{ var result = new Object();
result.outputNumber = (new Date()).toString();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getDate

Type method

Format Date object. GetDate ()

Commentary
Return the Day's date.

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
d = dateObj.getDate ();
outputs [0] = d;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
n = dateObj.getDate();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getDay

Type method

Format Date object. GetDay ()

Commentary *Returns the day of the week.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getDay ();
d = "Saturday,Tuesday Mizuki Kimu Moon". charAt (n);
outputs [0] = d;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
var d = new Array();
function (__string outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
n = dateObj.getDate();
d = "Today is Day "+ n;
result.outputNumber = d;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getFullYear

T y p e method

F o r m a t Date object. GetFullYear ()

C o m m e n t a r y
Year (4 digit year) returns.

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getFullYear ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
n = dateObj.getFullYear();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getHours

Type method

Format Date object. GetHours ()

Commentary *Returns.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getHours ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{ var result = new Object();
dateObj = new Date();
n = dateObj.getHours();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getMilliseconds

T y p e method

F o r m a t Date object. GetMilliseconds ()

C o m m e n t a r y *Returns the milliseconds.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getMilliseconds ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
function (__number outputNumber) main (__number inputNumber[1])
{ var result = new Object();
dateObj = new Date();
n = dateObj.getMilliseconds();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getMinutes

Type method

Format Date object. GetMinutes ()

Commentary *Returns the seconds.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getMinutes ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
function (__number outputNumber) main (__number inputNumber[1])
{ var result = new Object();
dateObj = new Date();
n = dateObj.getMinutes();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getMonth

Type method

Format Date object. GetMonth ()

Commentary *Returns. Than the actual value a little.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getMonth () + 1;
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
function (__number outputNumber) main ()
{
var result = new Object();
dateObj = new Date();
n = dateObj.getMonth() + 1;
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getSeconds

Type method

Format Date object. GetSeconds ()

Commentary *Returns the seconds.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getSeconds ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
dateObj = new Date();
n = dateObj.getSeconds() + 1;
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getTimezoneOffset

Type method

Format Date object. GetTimezoneOffset ()

Commentary *Difference (minutes) is returned.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getTimezoneOffset ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
dateObj = new Date();
n = dateObj.getTimezoneOffset();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

GetUtcDate

T y p e method

F o r m a t Date object. GetUTCDate ()

C o m m e n t a r y *Return the date in Coordinated Universal Time.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
d = dateObj.getUTCDate ();
outputs [0] = d;
```

Sample code - QC 3 JavaScript

```
var d = new Array();
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
dateObj = new Date();
d = dateObj.getUTCDate();
result.outputNumber = d;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getUTCDay

Type method

Format Date object. GetUTCDay ()

Commentary *Returns the day of the Coordinated Universal Time.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getUTCDay ();
d = "Saturday,Sunday, Monday, Tuesday, Wednesday, Thursday, Friday". charAt (n);
outputs[0] = d; outputs [0] = d;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
n = dateObj.getUTCDay();
d = "SMTWTFS".charAt(n);
result.outputNumber = d;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getUTCFullYear

Type method

Format Date object. GetUTCFullYear ()

Commentary *Years of Coordinated Universal Time (4 digit year) returns.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getUTCFullYear ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
dateObj = new Date();
n = dateObj.getUTCFullYear();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getUTCHours

Type method

Format Date object. GetUTCHours ()

Commentary *Returns the Universal Coordinated Time.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getUTCHours ();
outputs [0] = n;
```

Sample code - QC 2 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
dateObj = new Date();
n = dateObj.getUTCHours();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getUTCMilliseconds

T y p e method

F o r m a t Date object. GetUTCMilliseconds ()

C o m m e n t a r y *Returns the milliseconds in Universal Coordinated Time.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getUTCMilliseconds ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
dateObj = new Date();
n = dateObj.getUTCMilliseconds();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getUTCMinutes

Type method

Format Date object. GetUTCMinutes ()

Commentary *Returns the minutes in Universal Coordinated Time.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getUTCMinutes ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
dateObj = new Date();
n = dateObj.getUTCMinutes();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getUTCMonth

Type method

F o r m a t Date object. GetUTCMonth ()

C o m m e n t a r y *Returns the month in Universal Coordinated Time.Than the actual value a little.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getUTCMonth () + 1;
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
dateObj = new Date();
n = dateObj.getUTCMonth() + 1;
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getUTCSeconds

Type method

F o r m a t Date object. GetUTCSeconds ()

C o m m e n t a r y *Return the second of Coordinated Universal Time.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getUTCSeconds ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[1])
{
var result = new Object();
dateObj = new Date();
n = dateObj.getUTCSeconds();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

getYear

Type method

Format Date object. GetYear ()

Commentary *Return the number of years from 1900.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
n = dateObj.getYear ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
dateObj = new Date();
n = dateObj.getYear();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

parse

Type method

Format Date.parse (string date)

Commentary *1 January 1970 at returns 0 milliseconds from the time. Mon Oct 03 2005 17:32:55」The date string "Mon Oct 03 2005 17:32:55" to specify the string. 2005/10/3 not handled correctly and to specify a string formatted as.*

Sample code - QC 2 JavaScript

```
n = Date.parse ( "Mon Oct 03 2005 17:32:55");
outputs [0] = n.toString ();
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number time)
{
var result = new Object();
n = Date.parse("Mon Oct 03 2005 17:32:55");
result.outputNumber = n.toString();
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setDate

Type method

Format Date object. SetDate (date)

Commentary *Set the date. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setDate (2);
n = dateObj.getDate ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
var n = new Array();
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
dateObj.setDate(30);
result.outputNumber = dateObj.getDate();
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setFullYear

T y p e method

F o r m a t Date object. SetFullYear (year, month, day)

C o m m e n t a r y *Years AD (four digits). Current year and will be omitted. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setFullYear (2007);
n = dateObj.getFullYear ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
dateObj.setFullYear(2007);
n = dateObj.getFullYear();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setHours

Type method

Format Date object. SetHours (time)

Commentary *Set the time. 。 The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setHours (5);
n = dateObj.getHours ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
dateObj.setHours(5);
n = dateObj.getHours();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setMilliseconds

T y p e method

F o r m a t Date object. SetMilliseconds (ms)

C o m m e n t a r y *Sets the milliseconds. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setMilliseconds (987);
n = dateObj.getMilliseconds ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
dateObj.setMilliseconds(987);
n = dateObj.getMilliseconds();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setMinutes

T y p e method

F o r m a t Date object. SetMinutes (minutes)

C o m m e n t a r y *Set the minute. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setMinutes (35);
n = dateObj.getMinutes ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
dateObj.setMinutes(35);
n = dateObj.getMinutes();
result.outputNumber = n;
return result;}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setMonth

Type method

Format Date object. SetMonth (Monday)

Commentary *Set. Than the actual value a little. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setMonth (9);
n = dateObj.getMonth () + 1;
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
dateObj.setMonth(9);
n = dateObj.getMonth() + 1;
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setSeconds

T y p e method

F o r m a t Date object. SetSeconds (s)

C o m m e n t a r y *Set seconds. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
ateObj = new Date ();
dateObj.setSeconds (49);
n = dateObj.getSeconds ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
dateObj.setSeconds(48);
n = dateObj.getSeconds();
result.outputNumber = n;
return result; }
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setTime

Type method

Format Date object. SetTime (milliseconds)

Commentary *1 January 1970 is set at 0 ms from the time. 。 The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setTime (123456);
n = dateObj.getTime ();
outputs [0] = n;
```

Sample code - QC 2 JavaScript

```
function (__number outputNumber) main ()
{var result = new Object();
dateObj = new Date();
dateObj.setSeconds(48);
n = dateObj.getSeconds();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setUTCDate

Type method

Format Date object. SetUTCDate (date)

Commentary *Set the date in Coordinated Universal Time. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setUTCDate (1);
n = dateObj.getUTCDate ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{var result = new Object();
dateObj = new Date();
dateObj.setUTCDate(13);
n = dateObj.getUTCDate();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setUTCFullYear

T y p e method

F o r m a t Date object. SetUTCFullYear (year, month, day)

C o m m e n t a r y *AD Years of Coordinated Universal Time (4 digits). 。 Current year end will be omitted. 。 You can also omit month. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setUTCFullYear (2007);
n = dateObj.getUTCFullYear ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{var result = new Object();
dateObj = new Date();
dateObj.setUTCFullYear(2007);
n = dateObj.getUTCFullYear();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setUTCHours

T y p e method

F o r m a t Date object. SetUTCHours (time)

C o m m e n t a r y *Set the time of the Coordinated Universal Time. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setUTCHours (6);
n = dateObj.getUTCHours ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{ var result = new Object();
dateObj = new Date();
dateObj.setUTCHours(6);
n = dateObj.getUTCHours();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setUTCMilliseconds

Type method

Format Date object. SetUTCHours (time)

Commentary *Sets the milliseconds in Universal Coordinated Time. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setUTCMilliseconds (567);
n = dateObj.getUTCMilliseconds ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{var result = new Object();
dateObj = new Date();
dateObj.setUTCMilliseconds(567);
n = dateObj.getUTCMilliseconds();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setUTCMinutes

Type method

Format Date object. SetUTCMinutes (minutes)

Commentary *Set the minute. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setUTCMinutes (39);
n = dateObj.getUTCMinutes ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{var result = new Object();
dateObj = new Date();
dateObj.setUTCMinutes(39);
n = dateObj.getUTCMinutes();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setUTCMonth

T y p e method

F o r m a t Date object. SetUTCMonth (Monday)

C o m m e n t a r y *Set the World on Monday Agreement. Than the actual value a little. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setUTCMonth (9);
n = dateObj.getUTCMonth () + 1;
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{var result = new Object();
dateObj = new Date();
dateObj.setUTCMonth(9);
n = dateObj.getUTCMonth() + 1;
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setUTCSeconds

T y p e method

F o r m a t Date object. SetUTCSeconds (s)

C o m m e n t a r y *Set the second of Coordinated Universal Time.。 The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setUTCSeconds (23);
n = dateObj.getUTCSeconds ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{var result = new Object();
dateObj = new Date();
dateObj.setUTCSeconds(23);
n = dateObj.getUTCSeconds();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

setYear

Type method

Format Date object. SetYear (years)

Commentary *Set the number of years. The clock does not affect systems.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
dateObj.setYear (107);
n = dateObj.getYear ();
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main (__number inputNumber[2])
{var result = new Object();
dateObj = new Date();
dateObj.setYear(107);
n = dateObj.getYear();
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toDateString

Type method

Format Date object. SetDateString ()

Commentary *The date and time format.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
str = dateObj.toDateString ();
outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
var Today = str;
dateObj = new Date();
var str = dateObj.toDateString();
function (__string Today) main (__number Time[1])
{
var result = new Object();
result.Today = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toGMTString

Type method

Format Date object. ToGMTString ()

Commentary *Converts a date to a string, using the Internet GMT conventions. [Use toUTCString instead.]*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
str = dateObj.toGMTString ();
outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main ()
{var result = new Object();
dateObj = new Date();
str = dateObj.toGMTString();
result.outputNumber = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toLocaleDateString

Type method

Format Date object. ToLocaleDateString ()

Commentary *Returns the "date" portion of the Date as a string, using the current locale's conventions.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
str = dateObj.toLocaleDateString ();
outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main ()
{var result = new Object();
dateObj = new Date();
str = dateObj.toLocaleDateString();
result.outputNumber = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toLocaleString

Type method

Format Date object. ToLocaleString ()

Commentary *Converts a date to a string, using the current locale's conventions. Overrides the Object.toLocaleString method.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
str = dateObj.toLocaleString ();
outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main ()
{
var result = new Object();
dateObj = new Date();
str = dateObj.toLocaleString();
result.outputNumber = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toLocaleTimeString

Type method

Format Date object. ToLocaleTimeString ()

Commentary *The dateObj converted to local time. Returns the "time" portion of the Date as a string, using the current locale's conventions.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
str = dateObj.toLocaleTimeString ();
outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main ()
{var result = new Object();
dateObj = new Date();
str = dateObj.toLocaleTimeString();
result.outputNumber = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toTimeString

T y p e method

F o r m a t Date object. ToTimeString ()

C o m m e n t a r y *Converts the time format.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
str = dateObj.toTimeString ();
outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main ()
{var result = new Object();
dateObj = new Date();
str = dateObj.toTimeString();
result.outputNumber = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toUTCString

Type method

Format Date object. ToUTCString ()

Commentary *When you convert to a string in UTC.*

Sample code - QC 2 JavaScript

```
dateObj = new Date ();
str = dateObj.toUTCString ();
outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main ()
{var result = new Object();
dateObj = new Date();
str = dateObj.toUTCString();
result.outputNumber = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

UTC

Type method

Format Date.UTC (year, month, day, hour, minute, second)

Commentary *Returns the milliseconds until the specified date.*

Sample code - QC 2 JavaScript

```
str = Date.UTC (2006,2,15,9,35,20);
outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
dateObj = new Date();
str = Date.UTC(2006,2,15,9,35,20);
result.outputNumber = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

charAt

Type method

.() Format **string**. CharAt (position)

Commentary *characters from a string.The first character position 0, the next one will be a subsequent increase.*

Sample code - QC 2 JavaScript

```
str = "Sample"; str = "Sample";
n = str.charAt(1); n = str.charAt (1);
outputs[0] = n; outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__string output) main (__number inputNumber[2])
{
var result = new Object();
str = "Sample";
n = str.charAt(1);
result.output = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

charCodeAt

Type method

Format **string.** CharCodeAt (position)

Commentary *Returns the character code of the specified position.*

Sample code - QC 2 JavaScript

```
str = "A";
n = str.charCodeAt (0);
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
str = "A";
n = str.charCodeAt(0);
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

fromCharCode

T y p e method

F o r m a t String.fromCharCode (code letter)

C o m m e n t a r y

Characters and character codes. Character code, (comma) can be separated by a row.

Sample code - QC 2 JavaScript

```
str = String.fromCharCode(66); str = String.fromCharCode (66);
outputs[0] = str; outputs [0] = str;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main ()
{
var result = new Object();
str = String.fromCharCode(66);
result.outputNumber = str;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

indexOf

Type method

Format **s t r i n g .** IndexOf (search string, position search)

C o m m e n t a r y *Search for string. 。 Find the starting position can be omitted. If you omit the search string from the beginning. Results indicate the position MITSUKATTA string. If not found returns -1.*

Sample code - QC 2 JavaScript

```
str = "Sample";
n = str.indexOf ( "m");
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
str = "Sample";
n = str.indexOf("m");
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

lastIndexOf

Type method

Format **s t r i n g .** LastIndexOf (search string, position search)

C o m m e n t a r y *You can search a string from the end.Find the starting position can be omitted. If you omit from the end of the search string. Results indicate the position MITSUKATTA string. If not found returns -1.*

Sample code - QC 2 JavaScript

```
tr = "Quartz Extreme";
n = str.lastIndexOf ( "a");
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
str = "Quartz Extreme";
n = str.lastIndexOf("a");
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

length

Type property

s t r i n g . Length

C o m m e n t a r y *Returns the length of the character. Japanese one is counted as a character.*

Sample code - QC 2 JavaScript

```
str = "Sample";
n = str.length;
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number output) main ()
{
var result = new Object();
str = "Sample";
n = str.length;
result.output = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

match

T y p e method

Format **s t r i n g .** Match (/ string search / optional)

C o m m e n t a r y *You can search using a regular expression string. The optional g (global matching), i (in English capital letters, lower case ignored), m (in units of matched lines) can be omitted. Search results will return an array of strings found. You can check to see if the number of matches by examining the number of the array. Otherwise returns null.*

Sample code - QC 2 JavaScript

```
str = "Sample_sample123";
n = str.match (/ mp / g);
n = n.length;
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
str = "Sample_sample123";
n = str.match(/mp/g);
n = n.length;
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

replace

T y p e method

Format **s t r i n g .** Replace (/ search text / options, string replacement)

C o m m e n t a r y *Replace the same characters if using a regular expression search strings. The optional g (global matching), i (in English capital letters, lower case ignored), m (in units of matched lines) can be omitted. Returns the result after execution.*

Sample code - QC 2 JavaScript

```
str = "Sample_sample123";
n = str.replace (/ mp / g, "MP");
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
str = "Sample_sample123";
n = str.replace(/mp/g,"MP");
result.output = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

search

T y p e method

Format **s t r i n g .** Search (/ search text / options)

C o m m e n t a r y *You can search using a regular expression string. The optional g (global matching), i (in English capital letters, lower case ignored), m (in units of matched lines) can be omitted.。 Search results will return an array of strings found.You can check to see if the number of matches by examining the number of the array. 。 Otherwise returns null.*

Sample code - QC 2 JavaScript

```
str = "Sample_sample123";
n = str.search (/ 12 / g);
outputs [0] = n;
```

Sample code - QC 3 JavaScript

```
function (__number outputNumber) main ()
{
var result = new Object();
str = "Sample_sample123";
n = str.search(/12/g);
result.outputNumber = n;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

slice

T y p e method

Format **s t r i n g .** Slice (start position, end position)

C o m m e n t a r y *only the range specified string. The first character position to the left of 0, since the next one will be an increase.*

Sample code - QC 2 JavaScript

```
str = "ABCDEFG";
outputs [0] = str.slice (2,5);
```

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
str = "ABCDEFG";
result.output = str.slice(2,5);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

split

Type method

Format string.Split (split string)

Commentary *Split by a specified character string. Results are returned in a string array that is split.*

Sample code - QC 2 JavaScript

```
tr = "AB: CDE: FG";
result = str.split (":");
outputs [0] = result.toString ();
```

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
str = "AB:CDE:FG";
result = str.split(":");
result.output = result.toString();
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

substr

Type method

Format **string.** Substr (start position, extract)

Commentary *Minutes抜KI出SHIMASU specified character string from the specified location.*

Sample code - QC 2 JavaScript

```
str = "ABCDEFG";
outputs [0] = str.substr (2,5);
```

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
str = "ABCDEFG";
result.output = str.substr(2,5);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

substring

T y p e method

Format **s t r i n g .** Substring (start position, end position)

C o m m e n t a r y *string from the specified range. Start position is left of the first string is 0 second, followed by a second, and after an increase.*

Sample code - QC 2 JavaScript

```
str = "ABCDEFG";
outputs [0] = str.substring (2,5);
```

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
str = "ABCDEFG";
result.output = str.substring(2,5);
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toLowerCase

T y p e method

F o r m a t string.ToLowerCase ()

C o m m e n t a r y *Returns the calling string value converted to lowercase.*

Sample code - QC 2 JavaScript

```
str = "AmIgA";
outputs [0] = str.toLowerCase ();
```

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
str = "AmiGa";
result.output = str.toLowerCase();
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

toUpperCase

T y p e method

F o r m a t string.ToUpperCase ()

C o m m e n t a r y *Converted to uppercase and lowercase letters in English to English.*

Sample code - QC 2 JavaScript

```
str = "Amiga";
outputs [0] = str.toUpperCase ();
```

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
str = "commodore";
result.output = str.toUpperCase();
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

$1 $9 $ 1 - $ 9

Type Properties (Deprecated)

Format

Commentary

Deprecated Pattern Matching - see links for examples of working deprecated RegExp Properties. No QC2 or QC3 script.

Property Description

```
$1, ..., $9 Parenthesized substring matches, if any.
$_ See input.
$* See multiline.
$& See lastMatch.
$+ See lastParen.
$` See leftContext.
$' See rightContext.
```

Return to the Start Page

Last modified

compile

Type Method (Deprecated)

Format new RegExp (pattern matching string, optional)

Commentary

Regex (regular expressions) object. String pattern matching should be of most use.
(Unconfirmed)
Option "g" in the global match, "i" if it ignored the match in the case, m determine whether to match the line unit. No QC2 code sample.

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{var result = new Object();
var str="Drink Coffee";
var pattern=new RegExp("Drink Tea");
if (pattern.test(str)==true)
{
result.output = "Match found! "
}
else
{
result.output = "Match not found"
}
pattern.compile("Drink Coffee");
if (pattern.test(str)==true)
{
result.output = "Match found!"
}
else
{
result.output = "Match not found"
}
return result;
}
```

Download QC3 example script

Return to the Start Page

Last modified

exec

Type Method

Format

```
var result1 = regexp.exec(str);
var result2 = regexp(str); //
```

Commentary

Regex (regular expressions) object. String pattern matching should be of most use.
(Unconfirmed)
Option "g" in the global match, "i" if it ignored the match in the case, m determine whether to match the line unit.

Sample code - QC 2 JavaScript

```
str = "RegExp Sample Text. String Match Test.";
reObj = new RegExp ( "S +", "g");
result = str.match (reObj);
outputs [0] = "number of matches" + result.length + "is";
```

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
var myRe = /ab*/g;
var str = "abbcdefabh";
var myArray;
while ((myArray = myRe.exec(str)) != null)
{
var msg = "Found " + myArray[0] + ". ";
}
result.output = msg += "Next match starts at " + myRe.lastIndex;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

sticky

Type Method

Format

Commentary
Supposedly only supported in Firefox, perhaps due to the JS engine QC uses being Mozilla compliant, does work in QC too.

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
var supports_sticky;
try { RegExp('','y'); supports_sticky = true; }
catch(e) { supports_sticky = false; }
result.output = "supports_sticky";
return result;
}
```

Download QC3 example script

Return to the Start Page

Last modified

global

T y p e Property (R)

F o r m a t object . Global

C o m m e n t a r y

Indicates the state of global match. Whether to test the regular expression against all possible matches in a string, or only against the first. You cannot change this property directly.

Sample code - QC 2 JavaScript

```
str = "RegExp Sample Text. String Match Test.";
reObj = new RegExp ( "S +", "g");
result = str.match (reObj);
outputs [0] = "number of matches" + result.length + ", GlobalFlag:" + reObj.global;
```

Sample code - QC 3 JavaScript

```
var str = "RegExp Sample Text. String Match Test.";
function (__string output) main ()
{
var result = new Object();
reObj = new RegExp("S+","g");
result = str.match(reObj);
result.output = "Number of matches"+result.length+"、GlobalFlag "+reObj.global;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

ignoreCase

Type Property (R)

Format object . IgnoreCase

Commentary

Indicates whether to ignore the case. To ignore is true, otherwise false.
(Unconfirmed)
Option "g" in the global match, "i" if it ignored the match in the case, m determine whether to match the line unit.
You cannot change this property directly.

Sample code - QC 2 JavaScript

```
str = "RegExp Sample Text. String Match Test.";
reObj = new RegExp ( "S +", "gi");
result = str.match (reObj);
outputs [0] = "number of matches" + result.length + ". ignore flag:" + reObj.ignoreCase;
```

Sample code - QC 3 JavaScript

```
function (__string output) main (__number inputNumber[2])
{
var result = new Object();
str = "RegExp Sample Text. String Match Test.";
reObj = new RegExp("S+","gi");
result = str.match(reObj);
result.output = "Number of matches"+result.length+"。ignore Flag "+reObj.ignoreCase; return
result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

input

T y p e Property (Deprecated)

F o r m a t

C o m m e n t a r y
Deprectated, but still functional in QC. No QC 2 code for this Reg Expression property.

Sample code - QC 3 JavaScript

```
function (__string output) main (__string input)
{var result = new Object();
var pattern = new RegExp("Cybero");
var str = input;
pattern.test(str);
if(RegExp.input)
{
result.output = "The RegExp.input is: " + RegExp.input;
}
else
{
result.output = "The input does not contain Cyber0";
}
return result;}
```

Download QC3 example script

Return to the Start Page

Last modified

lastIndex

Type Property (Deprecated)

Format new RegExp (pattern matching string, optional)

Commentary
The index at which to start the next match. No QC2 code sample.

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
var str = "Visit cybero.co.uk (now [and then])";
var pattern = new RegExp("(then)", "g");
pattern.test(str);
result.output = "Last parenthesized substring is: " + RegExp.lastParen;
return result;
}
```

Download QC3 example script

Return to the Start Page

Last modified

lastMatch

Type Property (Deprecated)

Format new RegExp (pattern matching string, optional)

Commentary
The last matched characters. No QC2 code sample.

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
var str = "The rain in Spain stays mainly in the plain";
var pattern = new RegExp("ain", "g");
pattern.test(str);
result.output = "Match found. index now at: " + RegExp.leftContext;
return result;
}
```

Download QC3 example script

Return to the Start Page

Last modified

lastParen

T y p e Property (Deprecated)

F o r m a t new RegExp (pattern matching string, optional)

C o m m e n t a r y

The last parenthesized substring match, if any. No QC2 code sample.

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
var str = "Visit cybero.co.uk (now [and then])";
var pattern = new RegExp("(then)", "g");
pattern.test(str);
result.output = "Last parenthesized substring is: " + RegExp.lastParen;
return result;
}
```

Download QC3 example script

Return to the Start Page

Last modified

leftContext

Type Property (Deprecated)

Format new RegExp (pattern matching string, optional)

Commentary
The substring preceding the most recent match. No QC2 code sample.

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{
var result = new Object();
var str = "The rain in Spain stays mainly in the plain";
var pattern = new RegExp("ain", "g");
pattern.test(str);
result.output = "Match found. index now at: " + RegExp.leftContext;
return result;
}
```

Download QC3 example script

Return to the Start Page

Last modified

multiline

Property Type (R)

Format object. Multiline Regular expression

Commentary

Indicates whether to ignore the line breaks. Whether or not to search in strings across multiple lines.You cannot change this property directly.The value of multiline is true if the "m" flag was used; otherwise, false. The "m" flag indicates that a multiline input string should be treated as multiple lines. For example, if "m" is used, "^" and "$" change from matching at only the start or end of the entire string to the start or end of any line within the string.

Sample code - QC 2 JavaScript

```
CR = String.fromCharCode (13);
LF = String.fromCharCode (10);
str = "RegExp Sample Text." + CR + LF + "String Match Test.";
reObj = new RegExp ( "S +", "g");
reObj.multiline = true;
result = str.match (reObj);
outputs [0] = "number of matches" + result.length + ". multiline flag:" + reObj.multiline;
```

Sample code - QC 3 JavaScript

```
function (__string outputNumber) main (__number inputNumber[2])
{
var result = new Object();
CR = String.fromCharCode(13);
LF = String.fromCharCode(10);
str = "RegExp Sample Text."+CR+LF+"String Match Test.";
reObj = new RegExp("S+","g");
reObj.multiline = true;
result = str.match(reObj);
result.outputNumber = "Number of matches"+result.length+"。multilineFlag "+reObj.multiline;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

RegExp

T y p e Object

F o r m a t new RegExp (pattern matching string, optional)

C o m m e n t a r y

Regex (regular expressions) object. String pattern matching should be of most use.
(Unconfirmed)
Option "g" in the global match, "i" if it ignored the match in the case, m determine whether to match the line unit.

Sample code - QC 2 JavaScript

```
str = "RegExp Sample Text. String Match Test.";
reObj = new RegExp ( "S +", "g");
result = str.match (reObj);
outputs [0] = "number of matches" + result.length + "is";
```

Sample code - QC 3 JavaScript

```
function (__string output) main (__number inputNumber[2])
{
var result = new Object();
str = "RegExp Sample Text. String Match Test.";
reObj = new RegExp("S+","g");
result = str.match(reObj);
result.output = " Number of matches "+result.length+" Is "; return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

rightContext

Type Property (Deprecated)

Format new RegExp (pattern matching string, optional)

Commentary

The substring following the most recent match. No QC2 code sample.

Sample code - QC 3 JavaScript

```
function (__string output) main ()
{var result = new Object();
var str="Drink Coffee";
var pattern=new RegExp("Drink Tea");
if (pattern.test(str)==true)
{
result.output = "Match found! "
}
else
{
result.output = "Match not found"
}
pattern.compile("Drink Coffee");
if (pattern.test(str)==true)
{
result.output = "Match found!"
}
else
{
result.output = "Match not found"
}
return result;
}
```

Download QC3 example script

Return to the Start Page

Last modified

source

Type Property (R)

Format Regular expression.object Source

Commentary

Creates a regular expression object for matching text according to a pattern.
You cannot change this property directly.

Sample code - QC 2 JavaScript

```
str = "RegExp Sample Text. String Match Test.";
reObj = new RegExp ( "S +", "g");
outputs [0] = reObj.source;
```

Sample code - QC 3 JavaScript

```
var str = "RegExp Sample Text. String Match Test.";
var reObj = new RegExp("S+","g");
var myRe = new RegExp(/d(b+)(d)/ig);
var strOut = new Array();
var myArray = new Array();
function (__string strOut) main ()
{
var result = new Object();
myRe = /d(b+)(d)/ig;
result.strOut = myRe.source;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

test

Type Property (R)

Format Regularexpression object . Test (string contents) regexp.test([str])

Commentary

Tests for a match in its string parameter..Determines whether the string matches the pattern string check.If the string matches is true, if not it returns false.

Sample code - QC 2 JavaScript

```
str = "RegExp Sample Text. String Match Test.";
reObj = new RegExp ();
result = reObj.test ( "Te");
outputs [0] = "test result" + result + "";
```

Sample code - QC 3 JavaScript

```
var str = "ABCDEFGH";
var reObj = new RegExp();
var strOut = new Array();
var re = new RegExp();
function (__string strOut) main ()
{
var result = new Object();
if (re.test(str))
midstring = " contains ";
else
midstring = " does not contain ";
result.strOut =str + midstring + re.source;
return result;
}
```

Download QC2 example script Download QC3 example script

Return to the Start Page

Last modified

Special characters in regular expressions

Character	Meaning
\	For characters that are usually treated literally, indicates that the next character is special and not to be interpreted literally. For example, `/b/` matches the character 'b'. By placing a backslash in front of b, that is by using `/\b/`, the character becomes special to mean match a word boundary. *or* For characters that are usually treated specially, indicates that the next character is not special and should be interpreted literally. For example, * is a special character that means 0 or more occurrences of the preceding character should be matched; for example, `/a*/` means match 0 or more "a"s. To match `*` literally, precede it with a backslash; for example, `/a\*/` matches 'a*'.
^	Matches beginning of input. If the multiline flag is set to true, also matches immediately after a line break character. For example, `/^A/` does not match the 'A' in "an A", but does match the first 'A' in "An A."
$	Matches end of input. If the multiline flag is set to true, also matches immediately before a line break character. For example, `/t$/` does not match the 't' in "eater", but does match it in "eat".
*	Matches the preceding item 0 or more times. For example, `/bo*/` matches 'boooo' in "A ghost booooed" and 'b' in "A bird warbled", but nothing in "A goat grunted".
+	Matches the preceding item 1 or more times. Equivalent to `{1,}`. For example, `/a+/` matches the 'a' in "candy" and all the a's in "caaaaaaandy".
?	Matches the preceding item 0 or 1 time. For example, `/e?le?/` matches the 'el' in "angel" and the 'le' in "angle." If used immediately after any of the quantifiers `*`, `+`, `?`, or `{}`, makes the quantifier non-greedy (matching the minimum number of times), as opposed to the default, which is greedy (matching the maximum number of times). Also used in lookahead assertions, described under `(?=)`, `(?!)`, and `(?:)` in this table.
.	(The decimal point) matches any single character except the newline characters: \n \r \u2028 or \u2029. (`[\s\S]` can be used to match any character including newlines.) For example, `/.n/` matches 'an' and 'on' in "nay, an apple is on the tree", but not 'nay'.
(*x*)	Matches *x* and remembers the match. These are called capturing parentheses. For example, `/(foo)/` matches and remembers 'foo' in "foo bar." The matched substring can be recalled from the resulting array's elements `[1], ..., [n]` or from the predefined `RegExp` object's properties `$1, ..., $9`.
(?:*x*)	Matches *x* but does not remember the match. These are called non-capturing parentheses. The matched substring can not be recalled from the resulting array's elements `[1], ..., [n]` or from the predefined `RegExp` object's properties `$1, ..., $9`.
x(?=*y*)	Matches *x* only if *x* is followed by *y*. For example, `/Jack(?=Sprat)/` matches 'Jack' only if it is followed by 'Sprat'. `/Jack(?=Sprat|Frost)/` matches 'Jack' only if it is followed by 'Sprat' or 'Frost'. However, neither 'Sprat' nor 'Frost' is part of the match results.
x(?!*y*)	Matches *x* only if *x* is not followed by *y*. For example, `/\d+(?!\.)/` matches a number only if it is not followed by a decimal point. `/\d+(?!\.)/.exec("3.141")` matches 141 but not 3.141.
x|*y*	Matches either *x* or *y*. For example, `/green|red/` matches 'green' in "green apple" and 'red' in "red apple."
{*n*}	Where *n* is a positive integer. Matches exactly *n* occurrences of the preceding item. For example, `/a{2}/` doesn't match the 'a' in "candy," but it matches all of the a's in "caandy," and the first two a's in "caaandy."
{*n*,}	Where *n* is a positive integer. Matches at least *n* occurrences of the preceding item. For example, `/a{2,}/` doesn't match the 'a' in "candy", but matches all of the a's in "caandy" and in "caaaaaaandy."
{*n*,*m*}	Where *n* and *m* are positive integers. Matches at least *n* and at most *m* occurrences of the preceding item. For example, `/a{1,3}/` matches nothing in "cndy", the 'a' in "candy," the first two a's in "caandy," and the first three a's in "caaaaaaandy". Notice that when matching "caaaaaaandy", the match is "aaa", even though the original string had more a's in it.

[*xyz*]	A character set. Matches any one of the enclosed characters. You can specify a range of characters by using a hyphen. For example, `[abcd]` is the same as `[a-d]`. They match the 'b' in "brisket" and the 'c' in "ache".
[^*xyz*]	A negated or complemented character set. That is, it matches anything that is not enclosed in the brackets. You can specify a range of characters by using a hyphen. For example, `[^abc]` is the same as `[^a-c]`. They initially match 'r' in "brisket" and 'h' in "chop."
[\b]	Matches a backspace. (Not to be confused with `\b`.)
\b	Matches a word boundary, such as a space. (Not to be confused with `[\b]`.) For example, `/\bn\w/` matches the 'no' in "noonday"; `/\wy\b/` matches the 'ly' in "possibly yesterday."
\B	Matches a non-word boundary. For example, `/\w\Bn/` matches 'on' in "noonday", and `/y\B\w/` matches 'ye' in "possibly yesterday."
\c*X*	Where *X* is a letter from A - Z. Matches a control character in a string. For example, `/\cM/` matches control-M in a string.
\d	Matches a digit character in the basic Latin alphabet. Equivalent to `[0-9]`. **Note**: In Firefox 2 and earlier, matches a digit character from any alphabet. (bug 378738) For example, `/\d/` or `/[0-9]/` matches '2' in "B2 is the suite number."
\D	Matches any non-digit character in the basic Latin alphabet. Equivalent to `[^0-9]`. **Note**: In Firefox 2 and earlier, all alphabet. (bug 378738) For example, `/\D/` or `/[^0-9]/` matches 'B' in "B2 is the suite number."
\f	Matches a form-feed.
\n	Matches a linefeed.
\r	Matches a carriage return.
\s	Matches a single white space character, including space, tab, form feed, line feed and other unicode spaces. equivalent_s For example, `/\s\w*/` matches ' bar' in "foo bar."
\S	Matches a single character other than white space. equivalent_S For example, `/\S\w*/` matches 'foo' in "foo bar."
\t	Matches a tab.
\v	Matches a vertical tab.
\w	Matches any alphanumeric character from the basic Latin alphabet, including the underscore. Equivalent to `[A-Za-z0-9_]`. For example, `/\w/` matches 'a' in "apple," '5' in "$5.28," and '3' in "3D."
\W	Matches any character that is not a word character from the basic Latin alphabet. Equivalent to `[^A-Za-z0-9_]`. For example, `/\W/` or `/[^A-Za-z0-9_]/` matches '%' in "50%."
n	Where *n* is a positive integer. A back reference to the last substring matching the n parenthetical in the regular expression (counting left parentheses). For example, `/apple(,)\sorange\1/` matches 'apple, orange,' in "apple, orange, cherry, peach." A more complete example follows this table.
\0	Matches a NUL character. Do not follow this with another digit.
\x*hh*	Matches the character with the code *hh* (two hexadecimal digits)
\u*hhhh*	Matches the character with the Unicode value *hhhh* (four hexadecimal digits).

The literal notation provides compilation of the regular expression when the expression is evaluated. Use literal notation when the regular expression will remain constant. For example, if you use literal notation to construct a regular expression used in a loop, the regular expression won't be recompiled on each iteration. The constructor of the regular expression object, for example, `new RegExp("ab+c")`, provides runtime compilation of the regular expression. Use the constructor function when you know the regular expression pattern will be changing, or you don't know the pattern and are getting it from another source, such as user input.

1. **Note:** equivalent_s
   Equivalent to:

```
[\t\n\v\f\r \u00a0\u2000\u2001\u2002\u2003\u2004\u2005\u2006\u2007\u2008\u2009\u200a\u200b\u2028\u2029\u3000]
```

1. **Note:** equivalent_S
   Equivalent to:

```
[^\t\n\v\f\r \u00a0\u2000\u2001\u2002\u2003\u2004\u2005\u2006\u2007\u2008\u2009\u200a\u200b\u2028\u2029\u3000]
```